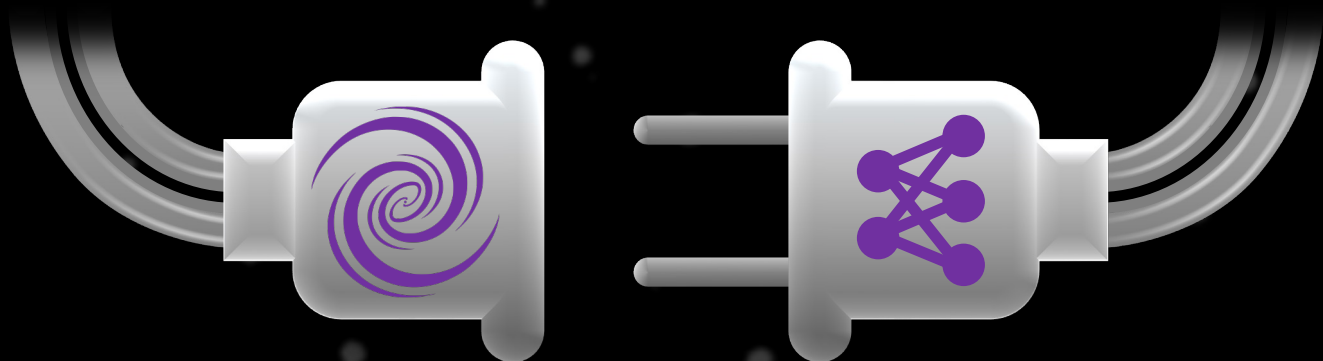


MAKING EVERYTHING CONNECT

Optimising Cosmological
Inference Through Emulation



Andreas Nygaard

MAKING EVERYTHING CONNECT

OPTIMISING COSMOLOGICAL INFERENCE
THROUGH EMULATION

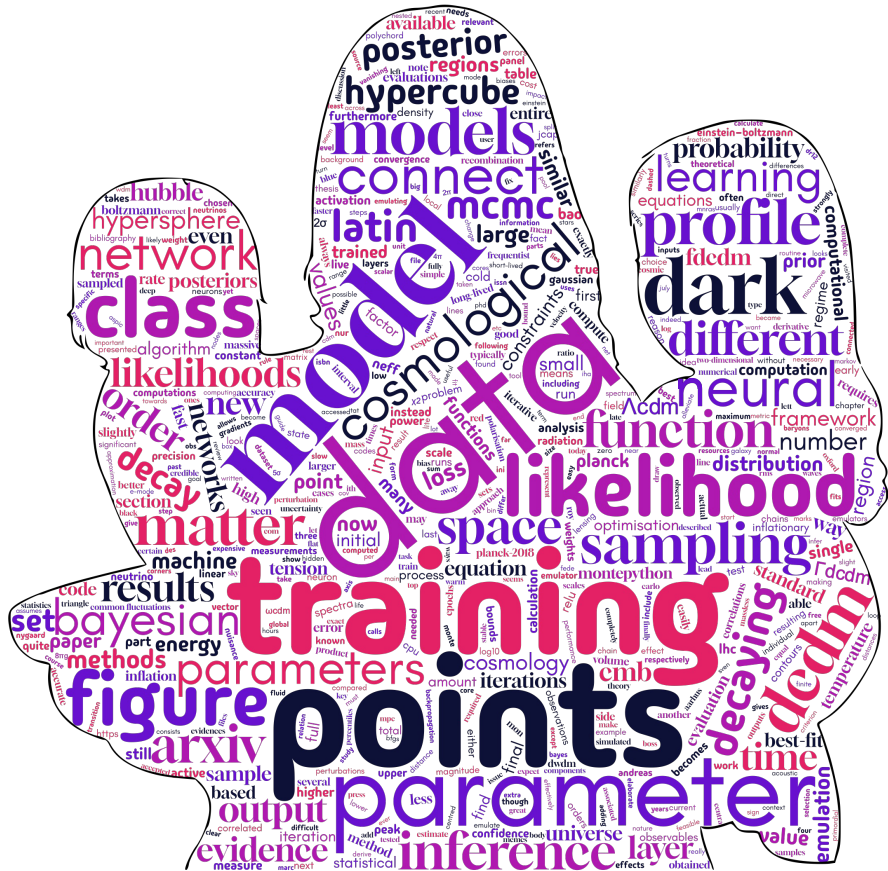
A DISSERTATION PRESENTED TO THE FACULTY OF NATURAL
SCIENCES OF AARHUS UNIVERSITY IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE PhD DEGREE

PHD THESIS BY
ANDREAS NYGAARD
ANDREAS@PHYS.AU.DK

SUPERVISORS
THOMAS TRAM
STEEN HANNESTAD

JANUARY 2025
DEPARTMENT OF PHYSICS & ASTRONOMY
AARHUS UNIVERSITY

*To my lovely wife
and beautiful daughters*



ACKNOWLEDGEMENTS

This thesis marks the end of an era. An era filled with joy, challenges, life-changing events, and unforgettable moments. Nearly half a decade has passed since I began my PhD, and the time before that now feels like a different life. Over these years, I have had the privilege of working with incredible people, travelling to exotic places for conferences, and forming friendships with fellow young researchers from around the world. I have engaged in fascinating scientific discussions, contributed meaningful research to the community, taught classes as a teaching assistant, and even become a father – twice! I have learnt a great deal, experienced the joy of my car being broken into (also twice!), and through it all, I am immensely grateful for every experience¹.

I faced it all, and I stood tall.
And did it my way.

Frank Sinatra

I would like to express my deepest gratitude to my two exceptional supervisors, Thomas Tram and Steen Hannestad. Working with you has been a privilege. Your invaluable guidance and delightful conversations have created a working environment that strikes the perfect balance between rigorous research and lighthearted discussions on just about anything. I could not have wished for more capable and supportive supervisors.

Being a part of the cosmology group at Aarhus University has been immensely rewarding, and I am pleased to have worked with so many bright students, one of whom is my close childhood friend Ajanthan Ketheeswaran (colloquially known as AK), who never fails to amaze me with his antics. Whether it is a sudden phone call about absolutely nothing or helping me with practicalities, I can always count on you to be in my life. Stay true – stay you!

My sincere appreciation goes to my great friend and former PhD colleague, Emil Brinch Holm, as well. Sharing an office with you for

¹ Except for my car being broken into. That was, in fact, mildly unpleasant.

four years has fostered a close bond, and our shared interests in both science and music, coupled with our daily conversations, have made this journey truly enjoyable. Whether travelling across the planet for conferences or venturing out for a power metal concert, I could not have asked for a better companion. Your friendship has been a cornerstone of my PhD experience and you have truly helped me set the universe on fire!

I would also like to extend my thanks to the many friends I've made internationally during my travels, particularly Sven Günther and Jonas El Gammal. Together with Emil, we achieved the remarkable feat of publishing an April Fools' paper on varying the mathematical constant π . The laughs I had while making that paper are so numerous that they form an uncountable set.

Finally, my heartfelt thanks go to my wonderful family. My dear wife Anne-Sofie has been my steadfast support throughout this journey. Her incredible flexibility allowed me to travel the world, and she even accompanied me during my research stay in England. You are my rock, and I could not have completed this journey without your unwavering support. To my beautiful daughters, Olivia and Mathilda, you have brought immeasurable joy into my life, even when balancing fatherhood with a PhD was challenging. Your smiles and laughter remind me of the most important things in life and have been a constant source of inspiration.

ABSTRACT

As cosmological models grow increasingly complex, the computational demands for their analysis have significantly increased, making parameter inference especially challenging. This thesis presents a framework, `CONNECT`, designed to alleviate these computational bottlenecks through the emulation of cosmological observables. Initially developed with the analysis of decaying dark matter models as motivation, `CONNECT` has evolved into a versatile tool capable of emulating observables across a wide range of cosmological models. By reducing the computational cost of these models, `CONNECT` facilitates more efficient parameter inference, enabling faster exploration of cosmological parameter spaces.

The thesis offers a comprehensive overview of the statistical methods used for parameter inference, including Bayesian and frequentist approaches, and provides the theoretical foundations of cosmological models, focusing on the standard Λ CDM model and its extensions involving decaying dark matter. The development of the `CONNECT` framework is discussed in detail, and so are its various applications. Results demonstrate the effectiveness of emulation in accelerating the estimation of model parameters, particularly for complex cosmological models.

While the work behind this thesis provides a contribution towards addressing the computational challenges in cosmology, it also highlights that there is still much to explore. The emulation framework presented here offers a promising tool, and future improvements and applications may help further streamline and optimise cosmological analyses.

DANISH ABSTRACT

Efterhånden som kosmologiske modeller bliver mere komplekse, er de beregningsmæssige krav til analyse af dem steget markant, hvilket gør parameterestimering særligt udfordrende. Denne afhandling præsenterer et framework, CONNECT, der er udviklet for at afhjælpe disse beregningsmæssige flaskehalse gennem emulering af kosmologiske observabler. Selvom CONNECT oprindeligt blev motiveret af modeller med henfaldende mørkt stof, er CONNECT blevet et alsidigt værktøj, der kan emulere observabler for en bred vifte af kosmologiske modeller. Ved at reducere de beregningsmæssige omkostninger ved disse modeller, gør CONNECT parameterestimeringen lettere og gør det derved muligt at udforske kosmologiske parameterrum hurtigere.

Afhandlingen giver et grundigt overblik over de statistiske metoder, der anvendes til parameterestimering, herunder både Bayesianske og frekventistiske tilgange, og præsenterer de teoretiske fundament bag kosmologiske modeller med særlig vægt på Λ CDM-modellen og dens udvidelser med henfaldende mørkt stof. Udviklingen af CONNECT koden diskuteres detaljeret, og det samme gør sig gældende for dens mange anvendelser. Resultaterne viser, hvordan emulering kan accelerere estimeringen af modelparametre, især for komplekse kosmologiske modeller.

Selvom arbejdet bag denne afhandling bidrager til at løse nogle af de beregningsmæssige udfordringer i kosmologi, viser det også, at der stadig er meget endnu at udforske. CONNECT rummer et lovende potentiale, og fremtidige forbedringer og anvendelsesmuligheder vil sandsynligvis bidrage til at strømline og optimere kosmologiske analyser.

LIST OF PUBLICATIONS

During my PhD, I have authored and co-authored a total number of 8 research papers. The following papers are the ones to which I have made major contributions:

- *Andreas Nygaard, Thomas Tram, and Steen Hannestad. “Updated constraints on decaying cold dark matter.” In: JCAP 05 (2021), p. 017. doi: 10.1088/1475-7516/2021/05/017. arXiv: 2011.01632 [astro-ph.CO]. [1]*
- *Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. “CONNECT: a neural network based framework for emulating cosmological observables and cosmological parameter inference.” In: JCAP 05 (2023), p. 025. doi: 10.1088/1475-7516/2023/05/025. arXiv: 2205.15726 [astro-ph.IM]. [2]*
- *Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. “Fast and effortless computation of profile likelihoods using CONNECT.” In: JCAP 11 (2023), p. 064. doi: 10.1088/1475-7516/2023/11/064. arXiv: 2308.06379 [astro-ph.CO]. [3]*
- *Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. “Cutting corners: hypersphere sampling as a new standard for cosmological emulators.” In: JCAP 10 (2024), p. 073. doi: 10.1088/1475-7516/2024/10/073. arXiv: 2405.01396 [astro-ph.CO]. [4]*

Except for the first one on decaying cold dark matter, they all describe the development and applications of my emulation framework for cosmology, CONNECT. These papers will all be presented in their entirety in this thesis.

Additionally, I have made significant contributions to the following papers:

- *Andreas Nygaard, Emil Brinch Holm, Thomas Tram, and Steen Hannestad. “Decaying Dark Matter and the Hubble Tension.” In: The Hubble Constant Tension. Ed. by Eleonora Di Valentino and Dillon Brout. Springer Series in Astrophysics and Cosmology. Springer, July 2024. Chap. 25, pp. 481–492. isbn: 978-981-99-0176-0, 978-981-99-0179-1, 978-981-99-0177-7. doi: 10.1007/978-981-99-0177-7. [5]*
- *Camilla T. G. Sørensen, Steen Hannestad, Andreas Nygaard, and Thomas Tram. “Calculating Bayesian evidence for inflationary models using CONNECT.” In: (June 2024). arXiv: 2406.03968 [astro-ph.CO]. [6]*

The first one is a review chapter for a book on the Hubble tension, and it has a small overlap with the Ref. [1]. This overlap will be excluded, but otherwise, the two papers will also be featured in their entirety.

Lastly, I have co-authored the following papers where my contribution has been minor relative to other authors in the author lists:

- *Emil Brinch Holm, Laura Herold, Steen Hannestad, Andreas Nygaard, and Thomas Tram. “Decaying dark matter with profile likelihoods.” In: Phys. Rev. D 107.2 (2023), p. L021303. doi: 10.1103/PhysRevD.107.L021303. arXiv: 2211.01935 [astro-ph.CO]. [7]*
- *Emil Brinch Holm, Andreas Nygaard, Jeppe Dakin, Steen Hannestad, and Thomas Tram. “PROSPECT: A profile likelihood code for frequentist cosmological parameter inference.” In: (Dec. 2023). arXiv: 2312.02972 [astro-ph.CO]. [8]*

No parts of these two papers will be presented in this thesis.

Papers and excerpts presented in the thesis have been modified slightly to align with the overall layout and ensure coherence within the context of the thesis.

CONTENTS

ACKNOWLEDGEMENTS	III
ABSTRACT	V
LIST OF PUBLICATIONS	VII
CONTENTS	IX
INTRODUCTION	XIII

I THEORY

1 STATISTICAL METHODS FOR PARAMETER INFERENCE	3
1.1 Bayesian parameter inference	3
1.1.1 Markov chain Monte Carlo	7
1.1.2 Metropolis-Hastings algorithm	8
1.2 Frequentist parameter inference	14
1.2.1 Maximum-likelihood estimation	15
1.2.2 Profile likelihoods	16
2 Λ CDM COSMOLOGY	19
2.1 Observations and the Hubble tension	20
2.2 The cosmic microwave background	23
2.2.1 Spherical harmonics expansion	24
2.2.2 The CMB power spectra	28
2.3 Homogenous and isotropic cosmology	32
2.3.1 Background evolution	35
2.4 Inhomogeneities and anisotropies	37
2.4.1 The Boltzmann equation	38
2.4.2 Linear perturbation theory in cosmology	39
3 DECAYING DARK MATTER	45
3.1 Introduction	49

3.1.1	Previous work and constraints	50
3.1.2	Outline of this paper	52
3.1.3	Cosmological data	53
3.2	Boltzmann equations for Λ CDM and DR	54
3.2.1	Background equations	54
3.2.2	Perturbation equations	54
3.3	Mapping short-lived Λ CDM to N_{eff}	56
3.3.1	Analytic solution of Boltzmann equations	57
3.3.2	Numerical analysis using CLASS	58
3.4	Current constraints on decay parameters	61
3.4.1	Long-lived Λ CDM regime	62
3.4.2	Short-lived Λ CDM regime	65
3.5	Impact on the Hubble and σ_8 tensions	69
3.6	Conclusion	71
3.7	Other decaying dark matter models	73
3.7.1	Λ CDM \rightarrow DR with profile likelihoods	73
3.7.2	Λ CDM \rightarrow DR+WDM	75
3.7.3	DWDM \rightarrow DR	78
3.7.4	Discussion and conclusion	80
4	MACHINE LEARNING	83
4.1	Learning scenarios	83
4.1.1	Supervised learning	84
4.1.2	Active learning	88
4.2	Neural networks	92
4.2.1	Multilayer neural networks	94
4.2.2	Activation functions	97
4.2.3	Backpropagation	100
4.2.4	Common challenges during training	108
II	EMULATION OF COSMOLOGY	
5	THE CONNECT FRAMEWORK	115
5.1	Introduction	117
5.2	Neural network design	120
5.2.1	Network architecture	121
5.2.2	Choosing a loss function	122
5.2.3	Choosing an activation function	124
5.2.4	Normalisation of inputs and outputs	125

5.3	Sampling of training data	127
5.3.1	Iterative sampling	129
5.3.2	Reduction of over-densities in sample points . . .	131
5.4	Integration with MontePython	133
5.4.1	Considerations and usage	133
5.4.2	Inference with Planck lite	134
5.5	Discussion and Conclusions	140
	Appendix 5.A: Structure of the code	147
6	USING CONNECT FOR PROFILE LIKELIHOODS	149
6.1	Introduction	151
6.2	Profile likelihoods and Bayesian inference	153
6.3	Optimisation of the likelihood function	155
6.3.1	Global optimisation	156
6.3.2	Local optimisation	157
6.4	Implementation	158
6.4.1	Auto-differentiability	158
6.4.2	Ensemble basin-hopping	159
6.4.3	Constraints on parameter space	160
6.4.4	Workflow	161
6.5	Results and discussion	162
6.5.1	Λ CDM	162
6.5.2	Massive neutrinos	164
6.5.3	Decaying cold dark matter	166
6.5.4	Computational performance	170
6.6	Conclusion and outlook	171
	Appendix 6.A: Choosing points for two-dimensional profile likelihoods	174
	Appendix 6.B: Recomputing and adding points	175
7	USING CONNECT FOR BAYESIAN EVIDENCES	179
7.1	Introduction	181
7.2	The CONNECT framework and Bayesian evidence	183
7.3	Validation of CONNECT for evidence computation	184
7.4	Inflationary model parameterisation	188
7.5	Numerical results	189
7.6	Runtime considerations	191
7.7	Discussion and conclusions	193
	Appendix 7.A: Convergence issues	195

8 IMPROVEMENT OF TRAINING DATA USING	
 HYPERSPHERE SAMPLING	197
8.1 Introduction	199
8.2 Sampling methods for training data	200
8.2.1 Latin hypercube sampling	201
8.2.2 Latin hypersphere sampling	202
8.2.3 Random uniform sampling from a hypersphere	203
8.2.4 Sampling near boundaries	205
8.2.5 Hyperellipsoid with correlations	205
8.3 Comparisons using CONNECT	206
8.3.1 Λ CDM	209
8.3.2 EDE+ M_ν + N_{ur}	214
8.4 Conclusion and outlook	221
FINAL THOUGHTS AND PERSPECTIVES	225
BIBLIOGRAPHY	227

INTRODUCTION

The study of cosmology seeks to understand the fundamental properties and evolution of the universe. Central to this pursuit are cosmological models that describe the behaviour of dark matter, dark energy, and other cosmic phenomena. As these models become increasingly sophisticated, the computational demands for their analysis rise significantly. This challenge is particularly evident in the context of decaying dark matter models, where traditional computational techniques, such as Markov chain Monte Carlo (MCMC) sampling, struggle with the increasing complexity of the models. The inefficiency of repeated computations has made the inference of cosmological parameters from these models a challenging task.

My research journey began with a focus on decaying dark matter models. These models, which hypothesise that dark matter can decay into other particles in the dark sector over time, offer intriguing insights into some of the unresolved questions in cosmology, including the Hubble tension. However, the computational intensity of these models soon became a major challenge, limiting the ability to perform robust parameter inference. To address this, I began working on a side project to develop an emulation framework that would expedite the calculation of cosmological observables, enabling more efficient use of MCMC sampling for parameter inference. This “side” project would, however, turn out to completely take over the focus of my PhD. The result of this endeavour is CONNECT, a neural network based framework designed to emulate cosmological observables. By approximating the outputs of cosmological computations of observables, CONNECT reduces the computational burden associated with these models, allowing for a quicker and more efficient analysis. What began as a tool tailored for decaying dark matter models evolved into a more comprehensive framework capable of handling a variety of cosmological models. This versatility has expanded the potential applications of CONNECT beyond its initial scope, making it a valuable asset for the broader cosmological community.

This thesis is structured to reflect the development and application of CONNECT in the context of cosmological research. It is split into two parts; the first one describes all the necessary theory for understanding

the emulation framework along with its cosmological motivation and applications, while the second part describes the results obtained using the framework and all the considerations that have gone into it.

The first part begins with a discussion of statistical methods for parameter inference, including both Bayesian and frequentist approaches. Chapter 1 provides a detailed overview of these methods, focusing on MCMC techniques and, in particular, the Metropolis-Hastings algorithm, as well as maximum-likelihood estimation and profile likelihoods. These methods are foundational to the subsequent discussions of cosmological modelling and inference. Chapter 2 introduces the standard cosmological model, Λ CDM, which serves as a baseline for exploring more complex models such as those involving decaying dark matter. This chapter covers key observations, such as the cosmic microwave background (CMB), and the Hubble tension, and delves into the theoretical framework describing both homogeneous and isotropic cosmology as well as inhomogeneities and structure formation. The focus shifts to the class of extensions to the Λ CDM model involving decaying dark matter in chapter 3, where various models are explored, with particular emphasis on the simplest decaying cold dark matter model (DCDM). The impact of DCDM on the Hubble and S_8 tensions is also examined, along with current constraints on decay parameters. The ending of this chapter lays the groundwork for understanding the computational challenges associated with more complex decaying dark matter models, which motivated the development of `CONNECT`. Chapter 4 is the last chapter of the theoretical part of the thesis, and it introduces the field of machine learning, with a focus on supervised learning and active learning scenarios. Neural networks, particularly multilayer neural networks, are discussed in detail, including their architecture, activation functions, and the backpropagation algorithm. This chapter furthermore addresses common training challenges and sets the stage for the development of the `CONNECT` framework.

The second part of the thesis (chapters 5 through 8) details the development, implementation, and application of `CONNECT`. Chapter 5 describes the neural network design, including the choice of loss functions, activation functions, and the normalisation of inputs and outputs. The integration of `CONNECT` with the popular MCMC sampler codes, `MONTYPYTHON` and `Cobaya`, for cosmological parameter inference, is also covered, demonstrating its practical applications in cosmological research. Chapter 6 focuses on the use of `CONNECT` for profile like-

lihoods, discussing the optimisation of the likelihood function and the implementation of numerical methods to accomplish this. The results of applying CONNECT to Λ CDM, massive neutrinos, and DCDM models are presented, along with an evaluation of its computational performance. Chapter 7 explores the use of CONNECT for Bayesian evidence computation, validating its accuracy and discussing its application to inflationary model parameterisation. Finally, chapter 8 examines the improvement of training data using hypersphere sampling methods, and these methods are compared with Latin hypercube sampling in the context of CONNECT's performance.

This thesis aims to address some of the computational challenges in cosmology by presenting a tool designed to emulate cosmological observables efficiently. By reducing the computational strain of complex models, CONNECT aims to enable faster and more accessible exploration of cosmological parameters. It is intended to serve as a useful resource for researchers, offering a means to streamline analyses and support ongoing investigations into the fundamental nature of the universe.

Part I

THEORY

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened.

Douglas Adams

STATISTICAL METHODS FOR PARAMETER INFERENCE

In order to perform detailed analyses of cosmological observations so we can unlock the secrets of our very universe, a robust mathematical foundation on which our methods can rely is essential. A crucial part of this foundation is the field of *statistics*. While statistical methods are often perceived as abstract or complex, they are indispensable for interpreting and analysing cosmological data. Without the application of rigorous statistical methods, cosmologists would not have the tools to derive meaningful insights from observational data.

It is therefore important to describe the statistical frameworks and their methodologies that are relevant to the analyses in this thesis. Throughout my PhD, I have had the opportunity to explore a range of statistical methods and this experience has taught me an important lesson: the importance of remaining open to alternative approaches, as embracing flexibility and adaptability often leads to deeper insights.

In this chapter, we will have a look at both Bayesian statistics and frequentist statistics – two different paradigms in which something as fundamental as the concept of probability is treated differently. We will go over different methods used for cosmological parameter inference of both paradigms and explain how they differ and complement each other.

1.1 BAYESIAN PARAMETER INFERENCE

Bayesian statistics, named after the English statistician Thomas Bayes, is currently the dominant statistical framework within the field of cosmological parameter inference. It revolves around a certain interpretation of probability where probability is regarded as a degree of belief in an event. This degree of belief may differ from person to person depending on the knowledge available to the respective individual, and it is thus also referred to as a *subjective probability*. It is furthermore known as

Bayesian probability due to its reliance on *Bayes' theorem* which is a fundamental, yet simple, theorem of probability theory regardless of the definition of probability in use [9].

The theorem describes the conditional probability $P(A|B)$ of event A given that B is true. The theorem can be derived easily by noting that the probability $P(A \cup B)$ of both events A and B occurring can be expressed as the product of the conditional probability of A given B and the probability of B occurring, i.e., $P(A \cup B) = P(A|B)P(B)$. There is, however, nothing special about either event over the other, so we can of course switch A and B in this expression. Doing that and equating the two expressions for the combined probability of both events occurring, we can derive an expression for the conditional probability of A given B ,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (1.1)$$

This is exactly Bayes' theorem and it has many uses within probability theory. In Bayesian statistics, the theorem is typically interpreted such that B represents experimental data while A represents a theoretical model. The theorem then suggests that the probability of the theoretical model being correct given the measured data ($P(A|B)$) is proportional to the probability of measuring that exact data if the theoretical model is correct ($P(B|A)$) multiplied by the probability of the theoretical model being correct ($P(A)$). $P(A)$ is in this case referred to as the *prior probability* that you would assign to the theoretical model before considering the data B , while $P(A|B)$ is referred to as the *posterior probability* of the theoretical model in light of the new data. Bayes' theorem in this case expresses a way to update one's subjective belief in a theoretical model when new data is available. The factor of $P(B)$ in the denominator describes the probability of measuring the data at all and is also referred to as the *evidence*.

It is useful to define the quantity $L(\mathbf{x}, \boldsymbol{\theta})$, known as the *likelihood function*, describing the probability of obtaining the data \mathbf{x} given the model parameter vector $\boldsymbol{\theta}$. This sounds a lot like the factor $P(B|A)$ in Bayes' theorem shown above, although that described probabilities of single events. We can, however, generalise Bayes' theorem to continuous variables and probability density functions [9],

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{L(\mathbf{x}, \boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int L(\mathbf{x}, \boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}, \quad (1.2)$$

where $p(\boldsymbol{\theta}|\mathbf{x})$ denotes the posterior probability density function over the model parameters $\boldsymbol{\theta}$ given the data \mathbf{x} , $\pi(\boldsymbol{\theta})$ is the prior probability distribution of the model parameters, $L(\mathbf{x}, \boldsymbol{\theta})$ is the likelihood function as described before, and the denominator is once again the evidence given by an integral of the product of the likelihood function and the prior distribution over the entire model parameter space. This integral in the denominator serves as a normalisation factor of the posterior probability distribution. The factor is usually disregarded when performing Bayesian parameter inference due to the integral over the entire parameter space being a computationally expensive endeavour. Within the field of cosmology, the evidence is used to compare cosmological models in order to determine which one generally fits the data better across all the different realisations of the models (points in their respective parameter spaces). It is important to stress that the likelihood function is not itself a probability density, since $\boldsymbol{\theta}$ is regarded as a fixed (but unknown) variable instead of a random variable.

The posterior is key to Bayesian parameter inference, but it can be difficult to visualise and investigate when the dimensionality of the parameter space is high. In this case, it is useful to look at only a single model parameter θ_i at a time. An expression of the one-dimensional probability function describing this parameter can be found through *marginalisation*, i.e., integrating out all other model parameters $\boldsymbol{\theta}_{j \neq i}$, and it is thus referred to as the *marginalised posterior*,

$$p(\theta_i|\mathbf{x}) = \int p(\boldsymbol{\theta}|\mathbf{x}) d\boldsymbol{\theta}_{j \neq i} = \frac{\int L(\mathbf{x}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}_{j \neq i}}{\int L(\mathbf{x}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}}. \quad (1.3)$$

Similarly, a two-dimensional marginalised posterior can be obtained by integrating out all model parameters except the two of interest.

Even though all information about the inference is contained in the posterior, it is in practice more useful to summarise the posterior with a measure of *location* and a measure of *spread*. One has a few choices for the location, i.e., the best estimate of the parameters; the maximum of the posterior called the *mode* of the posterior, the median of the posterior, or the mean of the posterior. The mode has the advantage that it coincides with the *maximum likelihood estimate* (MLE) if the prior distribution is uniform, but it can be misleading in skewed or multimodal distributions or be located near the ends of the distribution. The median has the advantage of usually being the most robust estimate by being invariant under *affine transformations* (such as taking the

logarithm), and it is also less affected by the tails of the distribution. The mean is widely used as the best estimate in Bayesian analysis, due to it using information from the entire distribution and being less affected by the tails of the distribution compared to the mode, but a very heavy tail can potentially place it far from most of the probability [10]. It is important to stress that none of these ways of summarising the posterior are necessarily better than the other; they simply express different information.

The uncertainty in Bayesian inference is the *credible interval* in the one-dimensional case or the *credible region* for a multi-dimensional parameter space. For a single parameter of interest, θ , the credible interval at a level of significance of $1 - \alpha$ is defined as the interval $[\theta_{\text{low}}, \theta_{\text{up}}]$ satisfying

$$P(\theta_{\text{low}} < \theta < \theta_{\text{up}}) = \int_{\theta_{\text{low}}}^{\theta_{\text{up}}} p(\theta|\mathbf{x})d\theta = 1 - \alpha. \quad (1.4)$$

That is, the probability of the true value of the parameter being within the interval is $1 - \alpha$. This is, however, ambiguous since multiple intervals will satisfy this equation, and it is therefore necessary to impose an additional condition [9]. The most commonly chosen condition is to use the shortest interval, meaning that the difference between the lower and upper limits should be minimised. In the case of a single maximum in the distribution, the shortest interval will always contain the mode of the posterior, so this interval is typically used when using the mode as the best estimate of the parameter. When using the mean or the median as the best estimate, the additional condition is often chosen to be that of the central interval, also known as the equal-tailed interval. This ensures that there is the same probability of being above and below the interval, i.e.,

$$P(-\infty < \theta < \theta_{\text{low}}) = P(\theta_{\text{up}} < \theta < \infty) = \frac{\alpha}{2}. \quad (1.5)$$

For multi-dimensional cases, the credible regions are typically defined as the contour regions with the smallest hypervolume. Furthermore, if a parameter is bounded, e.g., by physical constraints such as requiring a mass to be non-negative, the best estimate can be exactly on this boundary. In this case, only an upper or lower limit on the parameter is quoted.

1.1.1 MARKOV CHAIN MONTE CARLO

The posterior probability density function is not easily obtainable in practice, since the likelihood function usually is not an analytic function. A single evaluation of the likelihood function can also be computationally expensive, thus making grid-based sampling impossible in a high-dimensional parameter space. In order to be able to sample the posterior well, methods like *Markov chain Monte Carlo* (MCMC) are invaluable. This refers to a class of numerical methods allowing the user to draw samples from the proportional posterior (product of likelihood function and prior distribution) only in regions of the parameter space with significant contributions to the posterior [10]. In statistics, a *Markov chain* is a process that moves around different states stochastically, with the next state only depending on the current state (*memoryless* property), while *Monte Carlo* refers to a class of methods using random sampling to obtain numerical results. MCMC thus combine the random walk process by Markov chains with the randomised sampling aspect of Monte Carlo methods. The basic algorithm is as follows:

1. Select a starting point for the process (usually a best guess of a location measure of the posterior).
2. Select a new point randomly from a proposal distribution (depends on the specific kind of MCMC).
3. Assess whether or not to switch to the new point and append it to the Markov chain.
4. Repeat steps 2-3 until the distribution described by the chain converges.

There are various different options for the proposal distribution and how to accept/reject points, and each of these describes a specific MCMC algorithm. One such algorithm that is widely used for parameter inference due to its simplicity is the *Metropolis-Hastings algorithm*. This only needs the function value of the likelihood function, unlike more sophisticated MCMC methods like *Hamiltonian Monte Carlo* relying also on the gradient of the likelihood function. The Metropolis-Hastings algorithm is, however, the only one relevant to this thesis, so this will be addressed in detail.

1.1.2 METROPOLIS-HASTINGS ALGORITHM

The Metropolis-Hastings algorithm is particularly useful for sampling the Bayesian posterior distribution, since it is sufficient to only be able to compute a quantity proportional to the posterior. This means that we can ignore the integral in the denominator of equation (1.2) (the Bayesian evidence) since this is difficult to compute in practice.

For the algorithm to be useful in practice, the Markov process must converge to a unique stationary distribution. The Markov process being *ergodic* ensures the existence of such a unique stationary distribution, and it being *reversible* ensures that the distribution is the target distribution (the true posterior). The ergodicity means that the Markov process can reach every state from every other state and that the expected number of steps for returning to a state is finite, while the reversibility means that the transition between two states must be symmetric, i.e., being in state θ and transitioning to state θ' must have the same probability as being in state θ' and transitioning to state θ [11]. Using this, we may derive the acceptance criterion used in the Metropolis-Hastings algorithm [12].

The probability of transitioning from state θ to state θ' must be the product of the probability of selecting the state θ' as a new candidate given the current state θ and the probability of accepting the new state given the current state. The conditional probability of selecting a new candidate state θ' from the current state θ is described by the proposal function, $\mathcal{P}(\theta'|\theta)$, and we similarly denote the acceptance probability as a conditional probability, $\mathcal{A}(\theta'|\theta)$. The transition probability, τ , from θ to θ' is thus

$$\tau(\theta'|\theta) = \mathcal{P}(\theta'|\theta)\mathcal{A}(\theta'|\theta). \quad (1.6)$$

The reversibility of the Markov process results in the product of the transition probability $\tau(\theta'|\theta)$ and the probability of being in the state θ is equal to the product of the reverse transition probability and the probability of being in the state θ' . The probability of being in a certain state is exactly the target distribution evaluated at that state, i.e., the posterior, $p(\theta)$ (we keep the conditional dependence on the data implicit for now). We may then write

$$p(\theta)\tau(\theta'|\theta) = p(\theta')\tau(\theta|\theta'), \quad (1.7)$$

into which we can substitute the expression for the transition probability and arrive at an expression for the ratio of acceptance probabilities,

$$\frac{\mathcal{A}(\theta'|\theta)}{\mathcal{A}(\theta|\theta')} = \frac{p(\theta')\mathcal{P}(\theta|\theta')}{p(\theta)\mathcal{P}(\theta'|\theta)}. \quad (1.8)$$

There now is some freedom in choosing the acceptance probability, $\mathcal{A}(\theta'|\theta)$, such that it satisfies the above equation, but the choice used in the Metropolis-Hastings algorithm is using a function equal to the right-hand side of equation (1.8) when its function value is less than 1 and a constant function of 1 when the function value is greater than 1. This can be written as

$$\mathcal{A}(\theta'|\theta) = \min\left(1, \frac{p(\theta')\mathcal{P}(\theta|\theta')}{p(\theta)\mathcal{P}(\theta'|\theta)}\right). \quad (1.9)$$

In the case of a symmetric proposal function, these will cancel in the expression and the acceptance will alone depend on the ratio between the posterior values in the two states. This is known as just the *Metropolis algorithm* as this was the one used in the original paper from 1953 [13]. The original paper also described a uniform distribution with a given range centred at the current state as the proposal distribution, while a more common choice today is a Gaussian distribution centred at the current state described by an estimated covariance matrix (also a symmetric choice). Using a Gaussian proposal distribution, the complete Metropolis-Hastings algorithm is then described by the following pseudo-code:

```

chain      = []
θcurrent   = [θ1initial, θ2initial, ..., θNinitial]
Lcurrent   = posterior(θcurrent)
Cov        = ⟨Estimated N × N covariance matrix⟩
for 0 ≤ i < Nsteps do
    Pproposal(θ) = MultivariateGaussian(θcurrent, Cov)
    θnew         = ⟨Draw point from Pproposal⟩
    Lnew         = posterior(θnew)
    u             = RandomUniform([0, 1])
    a             = min(1, Lnew / Lcurrent)
    if u ≤ a then
        θcurrent = θnew
        Lcurrent = Lnew
    ⟨append θcurrent to chain⟩
    
```

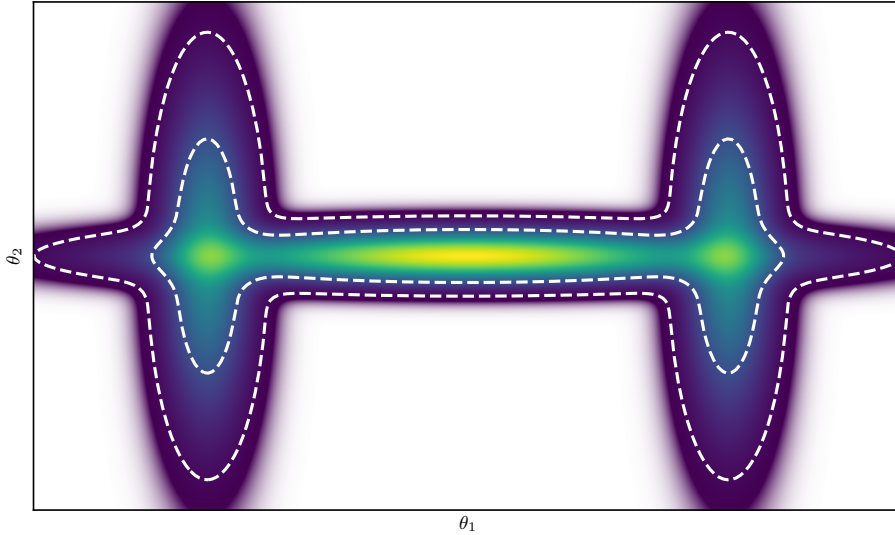


Figure 1.1: Example distribution used to test the Metropolis-Hastings algorithm. This two-dimensional distribution consists of three superimposed Gaussian distributions with a global maximum in the centre of the parameter space. The two contours mark the 68.27% and 95.45% credible regions.

Let us have a look at this algorithm in action. Consider the (somewhat contrived) two-parameter example distribution in figure 1.1 consisting of three superimposed Gaussian distributions. It has the two parameters θ_1 and θ_2 , and I will (for obvious reasons) refer to the distribution as the *H-distribution* denoted by the function $L_H(\theta_1, \theta_2)$. We now imagine a case where the H-distribution is the true posterior that we want to sample. Before using the algorithm, it is worth considering how many steps we might need to take before the distribution is sampled well, but this is not easy to know from the beginning. Luckily, because the transition to the next point only depends on the current point, we may choose any number, and if it is insufficient, we can just continue. The process of checking for convergence manually several times can, however, be tedious, so in practice, this is often automated. For this example, we will repeat the sampling for four different numbers of steps, 1,000, 5,000, 10,000, and 100,000, and set the centre of the parameter space (the global maximum) as the starting point.

The points sampled in these four MCMC runs are shown in figure 1.2 along with the 68.27% and 95.45% credible regions computed from the points and the true posterior contours as well. It is clearly visible that 1,000 points are too few to accurately represent the H-distribution. The

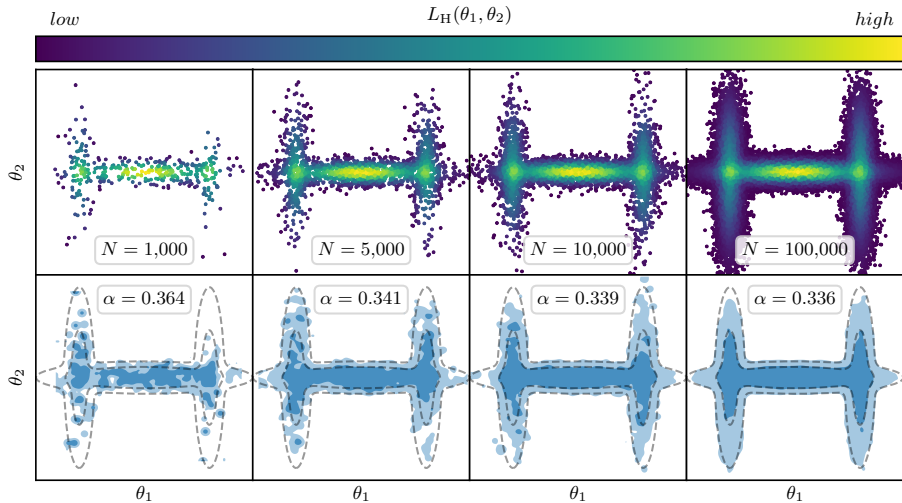


Figure 1.2: The top row of panels shows the points sampled by the Metropolis-Hastings algorithm for different numbers of steps in the MCMC runs, coloured to reflect the value of the true posterior denoted $L_H(\theta_1, \theta_2)$. The acceptance rates α for the individual runs are also shown in the panels. The bottom row shows the resulting 68.27% and 95.45% credible regions (blue filled contours) obtained using the sampled points along with the contours of the true posterior (dashed grey line).

algorithm had started to sample the sides of the H-shape but it had not had the time necessary to explore the entire distribution. Only 364 points were actually accepted (and thus included in the figure) due to the acceptance rate being 0.364 (also shown in the figure), so one needs to also take this into account when choosing the number of steps. The optimal acceptance rate is between 0.2 and 0.4 depending on the dimensionality of the problem [14]. The acceptance rate is similar for the other amounts of sampling points, and already at 5,000 we see that the entire shape has been visited. It is not, however, until we use on the order of 100,000 points that we get smooth and converged posterior contours. Depending on the shape of the posterior and the dimensionality, it can potentially require a much larger number of steps before convergence is reached.

It is usually necessary to marginalise the posterior in order to present it, although we, in fact, can present the entire posterior in this case since the parameter space is only two-dimensional. We compute the one-dimensional marginalised posteriors for the two parameters by in-

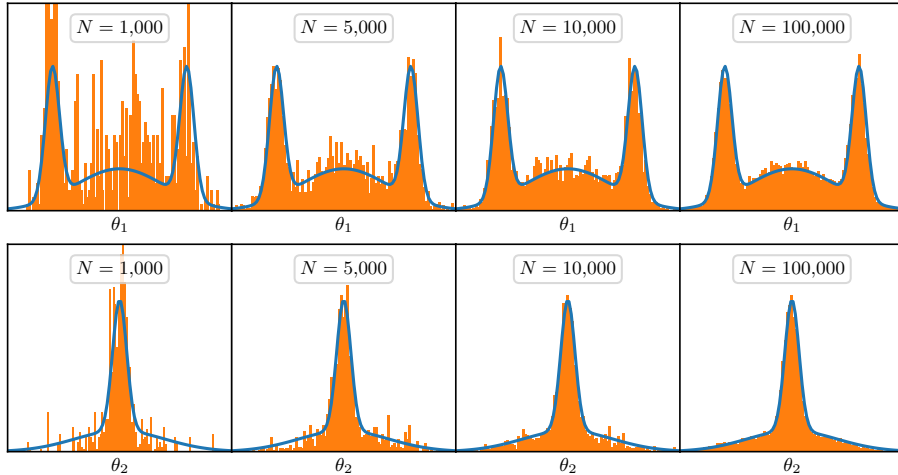


Figure 1.3: One-dimensional histograms (orange bars) for each of the two parameters and for different numbers of steps in the MCMC runs. The true marginalised posteriors are also shown (blue lines).

tegrating the other parameter out, e.g., the marginalised posterior for θ_1 is given by

$$p(\theta_1) = \int L_H(\theta_1, \theta_2) d\theta_2. \quad (1.10)$$

The true marginalised posteriors for θ_1 and θ_2 are shown in figure 1.3 along with the one-dimensional histograms of the sampled data. Here, we can see how the histograms converge towards the marginalised posteriors. It is interesting to note that the two marginalised posteriors have maxima located differently. The θ_1 posterior has two equal maxima near the edges and a smaller bump in the middle, while the θ_2 posterior has a narrow central peak with broad tails. This is despite the true two-dimensional posterior having its global maximum in the centre of the parameter space. The reason for the θ_1 posterior not having its central bump as the highest peak is due to it only contributing significantly in a narrow interval when integrating over θ_2 . The marginalised posterior is thus sensitive to the distribution of probability volume and will not necessarily peak at the maximum of the true multi-dimensional posterior. This is known as a *volume effect* and it is necessary to be aware of how this potentially can hide certain features of the multi-dimensional posterior.

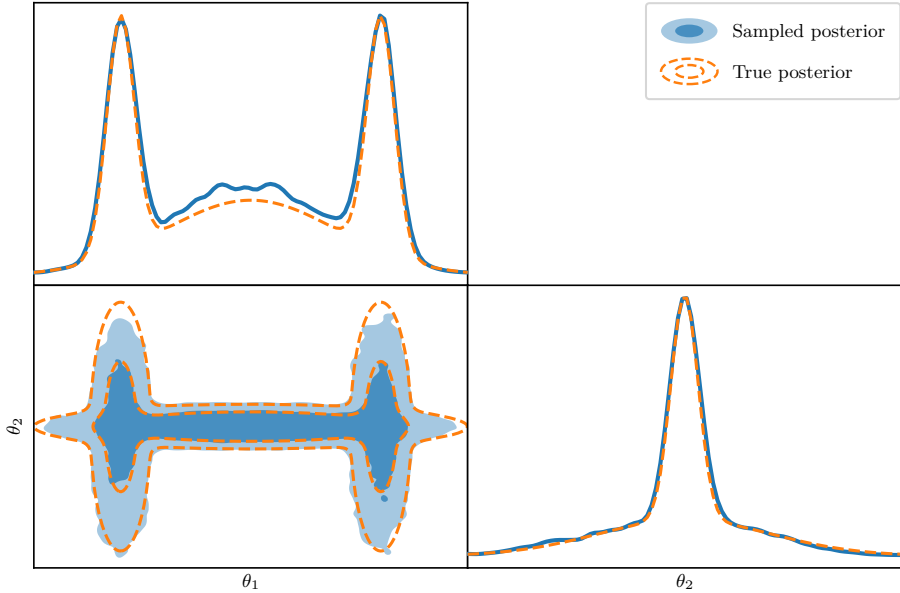


Figure 1.4: Triangle plot showing the one-dimensional marginalised posterior distributions and the contour lines of the two-dimensional posterior obtained through 100,000 MCMC steps (blue contours/lines) along with the contours of the true posterior and true marginalised posteriors (dashed orange contours/-lines).

Typically, the results of such an MCMC analysis are summarised by a *triangle plot* or *corner plot* where the one-dimensional marginalised posteriors are shown on the diagonal of an $N \times N$ grid of subplots (for an N -dimensional parameter space), the two-dimensional marginalised (in our case, the two-dimensional posterior is not marginalised since $N = 2$) posteriors are shown as contour lines enclosing the 68.27% and 95.45% credible regions in the lower triangle of the grid, and the upper triangle is kept empty. The triangle plot for this example (the results after 100,000 steps) is shown in figure 1.4 along with the true contours and true marginalised posteriors. One usually uses the data from histograms or *Gaussian kernel density estimation* to obtain an estimate of the marginalised posteriors, and in this case, we use data from histograms and subsequently smooth it using a Gaussian filter.

Aside from the volume effects, there are other common points of criticism when it comes to Bayesian inference. One of these challenges is the dependence on prior distributions. Although this can be a very useful

feature of Bayesian inference, the subjectivity related to the choice of prior can lead to different results in the end. Usually, when nothing is known about the model parameter in question, the prior is chosen to be uniform. One would naïvely think that a uniform prior does not impose any prior beliefs, but that would be wrong. A uniform prior is only uniform in that particular parametrisation. If a transformation is made to another parameter, the prior would not remain uniform. An example of this is the transformation of a uniform prior in a parameter θ to the prior in the parameter $\phi = \exp(\theta)$, where the resulting prior in ϕ is hyperbolic (Jacobian of the inverse transformation) and smaller values will then be preferred over larger values. It is possible to determine the least informative prior known as the Jeffreys prior [15], but this is just *least informative* and not completely *uninformative*. This is not so much a problem as an inherent part of Bayesian analysis; one just needs to be aware that priors carry information, and results, therefore, might be artificially induced when the data is not sufficiently constraining.

1.2 FREQUENTIST PARAMETER INFERENCE

Frequentist methods for parameter inference have only recently begun to regain popularity within cosmological parameter inference. Bayesian parameter inference has dominated the field in the past decades, with only a few instances of frequentist inference along the way. A reason for this is the computational cost of performing a frequentist analysis as global optimisation is difficult when the evaluation of the function is rather slow.

Frequentist statistics gets its name from its interpretation of probability. Imagine flipping a coin, hiding it, and asking a friend what the probability of the outcome being heads is. If he is Bayesian at heart, he will likely answer “50%, if the coin is fair”, but if he is a frequentist, he might just say “either 100% or 0%”. In a frequentist’s mind, the question makes little sense, since the outcome is already determined, regardless of him knowing the result, while a Bayesian interprets the probability as a subjective belief given the available information. The way a frequentist deals with probability is through the frequency of outcomes when repeating an experiment. It is defined as the fraction of times an outcome occurs in the limit of infinitely many repetitions of the experiment [9].

Returning to the coin-flip example, a frequentist would instead express his confidence in a statement such as “the outcome is heads” as being 50%. When estimating a parameter through a set of measurements leading to an estimate x slightly different from the true value μ , and one determines an uncertainty σ based on the spread of the data (normally distributed), it is tempting to interpret the result, i.e., $x \pm \sigma$, as a 68.27% probability of μ being in the interval $[x - \sigma, x + \sigma]$, but this does not make sense in a frequentist perspective. Either μ lies in the range, or it does not. Similarly to the coin-flip example, we may, however, state that the statement “ μ is within the range” has a 68.27% chance of being true, or you have 68.27% confidence in the statement.

1.2.1 MAXIMUM-LIKELIHOOD ESTIMATION

The likelihood function as described in the previous section on Bayesian inference is also widely used in frequentist methods. One such method of parameter estimation is the *maximum-likelihood estimate* (MLE). Given N statistically independent measurements of a quantity, $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ each following the same probability density, $f(x; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a set of model parameters with unknown values to be estimated. We can then express the likelihood function as [9]

$$L(\mathbf{x}, \boldsymbol{\theta}) = \prod_{i=1}^N f(x_i; \boldsymbol{\theta}), \quad (1.11)$$

where $f(x; \boldsymbol{\theta})$ is properly normalised. The maximum likelihood estimate of the model parameters $\boldsymbol{\theta}$ are then the values $\hat{\boldsymbol{\theta}}$ that maximises the likelihood function. In the limit of infinitely many measurements, the estimate will converge towards the true values of the model parameters, but for a finite number of measurements, the estimator generally has a bias proportional to $1/N$. A very useful property of the estimator is its invariance under parameter transformations, where the maximum likelihood estimate $\hat{\phi}$ of a parameter ϕ resulting from the transformation $\phi = g(\theta)$ will be exactly given as $\hat{\phi} = g(\hat{\theta})$.

In practice, it is generally more convenient to compute the negative of the logarithm to the likelihood and then minimise this quantity referred to as the *log-likelihood*. This turns the product into a sum instead,

$$-\ln L(\mathbf{x}, \boldsymbol{\theta}) = - \sum_{i=1}^N \ln f(x_i; \boldsymbol{\theta}) . \quad (1.12)$$

The uncertainty of the estimate $\hat{\theta}$ of one parameter is the interval described by the values where the log-likelihood drops 1/2 from its maximum value. The estimate does not generally lie in the centre of this interval for a small number of data points, but as that number increases, the likelihood will converge to a Gaussian distribution due to the *central limit theorem* [9], and the uncertainty can thus be computed from the curvature (second derivative) of the log-likelihood at its peak.

1.2.2 PROFILE LIKELIHOODS

If the likelihood function depends on many parameters, but one is only interested in a single parameter or a subset of parameters, it is convenient to use a *profile likelihood*. Similarly to how Bayesian marginalisation decreases the dimensionality by integrating out additional parameters, profile likelihoods achieve this by optimising in the additional parameters. If we split the parameter vector $\boldsymbol{\Theta}$ into two parts consisting of the parameters of interest, $\boldsymbol{\theta}$, and the parameters to optimise over, $\tilde{\boldsymbol{\theta}}$, the profile likelihood is defined as

$$L(\boldsymbol{\theta}) = \max_{\tilde{\boldsymbol{\theta}}} (L(\boldsymbol{\Theta})) . \quad (1.13)$$

The resulting profile likelihoods have the nice properties of not being dependent on any subjective choice (unlike Bayesian posteriors which depend on the choice of prior), and they do not suffer from volume effects like marginalised posteriors do. The location of the maximum of a profile likelihood is thus the same (in that parameter subspace) as the maximum of the full likelihood function.

Profile likelihoods are the focus of the paper presented in chapter 6 and the topic of profile likelihoods and the construction of confidence intervals will thus be further explored in section 6.2. It should, however, be mentioned that the construction of intervals based on profile likeli-

hoods only makes sense in the large data limit where *Wilks' theorem* holds true, meaning that the *Wilks likelihood ratio statistic*,

$$W(\boldsymbol{\theta}) = -2 \ln \frac{L(\boldsymbol{\theta}, \hat{\tilde{\boldsymbol{\theta}}})}{L(\hat{\boldsymbol{\Theta}})} , \quad (1.14)$$

follows a χ^2 distribution with $\dim(\boldsymbol{\theta})$ degrees of freedom [16], e.g., 1 degree of freedom for one-dimensional profile likelihoods, where $L(\boldsymbol{\theta}, \hat{\tilde{\boldsymbol{\theta}}})$ refers to the likelihood where all parameters in $\tilde{\boldsymbol{\theta}}$ assume the values of their MLE for a given $\boldsymbol{\theta}$ and $L(\hat{\boldsymbol{\Theta}})$ is the global maximum for the entire parameter space. In reality, Wilks' theorem only applies to the full likelihood function, so applying it to a profile likelihood assumes that the MLEs of the parameters in $\tilde{\boldsymbol{\theta}}$ are equal to their true values. This is only true in the large data limit, but it can be difficult to assess whether or not the data is in this limit. If not in the large data limit, one introduces an error in the results for each parameter optimised over in the profile likelihood. In high dimensionality, this error can potentially grow large for the one-dimensional profile likelihood, so interval construction might not always be exact.

Λ CDM COSMOLOGY

In the era of modern-day cosmology, one model has emerged that is particularly good at describing cosmological observations while simultaneously being rather simple. This model assumes a universe consisting of radiation, baryonic matter, dark matter, and a *cosmological constant*, Λ . The standard model of particle physics well describes the baryonic matter and radiation, and the cosmological constant can be interpreted as an extra term in the Einstein field equations such that it is just an inherent property of gravitation on large scales. Dark matter, though, has puzzled cosmologists for nearly a century, and its properties and origin remain clouded in mystery. We have, however, been able to deduce some properties based on assumptions about dark matter. If we assume it to be a particle, then observations tell us that it should be *cold*, i.e., be massive and only have small velocity, and we refer to this theoretical particle as a Weakly Interacting Massive Particle (WIMP). Cold dark matter (CDM) together with the cosmological constant, Λ thus leads to the name for this well-fitting model, the Λ CDM *model*.

This is regarded as the standard model of cosmology, but quite unsatisfactorily, it leaves us with some unanswered questions. Even though the Λ CDM model fits our observations well, there are some discrepancies and anomalies that it cannot accommodate. One such discrepancy is the *Hubble tension* which we will go over in the next section. During this chapter we will also have a look at one of the types of measurements particularly relevant to this thesis, the cosmic microwave background radiation, and we will lastly discuss the equations governing the evolution of matter and energy as well as those describing the formation of structure in the universe.

2.1 OBSERVATIONS AND THE HUBBLE TENSION

The driving power behind the field of cosmology is the large variety of observational data collected by different national and international experiments over the course of several decades. It is these observations and experiments that inspire theorists to construct models of the universe that neatly predict the observations. Since we only have one single universe to observe, we cannot easily test cosmological theories as if everything were reproducible in a laboratory, and we thus rely heavily on these observations to indirectly infer our results.

Generally speaking, the observations can be divided into two categories: early-time observations and late-time observations, depending on when the observed light was emitted. An example of early-time observations is the *cosmic microwave background* (CMB) which is light emitted from the last scattering of photons on electrons in the early universe, whereas an example of a late-time observation is that of the large-scale structure of matter.

Almost a century ago in 1929, the American astronomer Edwin Hubble cemented the idea of an expanding universe by demonstrating an increasing receding velocity of more distant galaxies, known as *Hubble's law*:

$$v = H_0 D. \quad (2.1)$$

The constant of proportionality between the distance, D , and receding velocity, v , is known as the *Hubble constant*, H_0 , and corresponds to the rate of expansion of the universe today. The best estimates of this constant from direct measurements come from Supernova type Ia data since these function as so-called *standard candles*, which makes very large distance measurements quite accurate. At the time of writing, the current best estimate from the *SH0ES* collaboration is $(73.04 \pm 1.04) \text{ km s}^{-1} \text{ Mpc}^{-1}$ [17].

This parameter can, however, also be inferred from early-time measurements like that of the CMB. In this case, one has to assume an underlying model of the universe¹. In the standard ΛCDM model, the value inferred from CMB by the Planck collaboration is $(67.4 \pm 0.5) \text{ km s}^{-1} \text{ Mpc}^{-1}$ [21]. This expresses a tension in cosmology

¹ In reality, the same is true for the direct measurement, since the relationship between distance and redshift depends on the cosmological model.

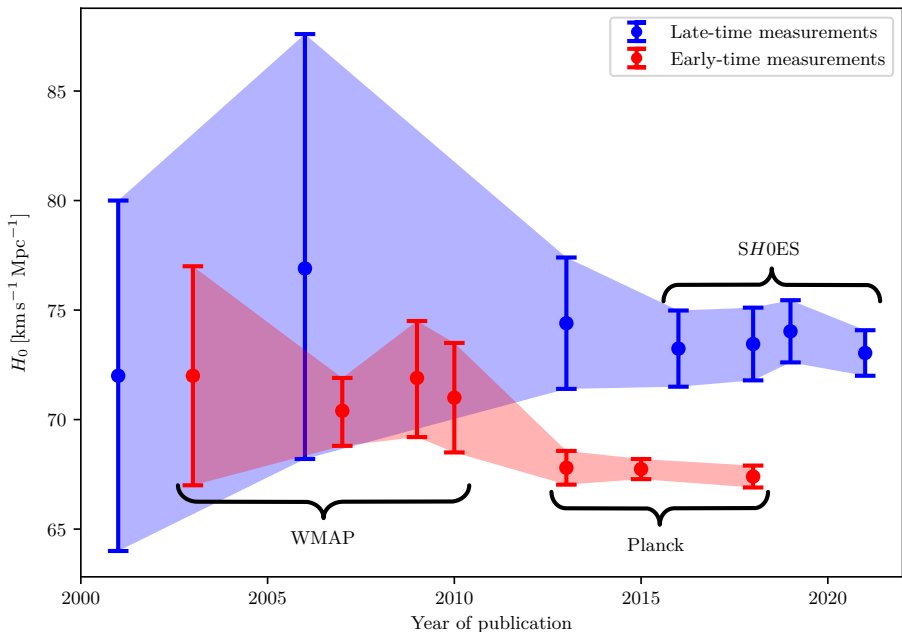


Figure 2.1: The evolution of the early-time (red) and late-time (blue) measurements of the Hubble constant in the past couple of decades. This is only a selection of the measurements in this period of time, but these have been chosen to highlight the discrepancy in a simple manner. The three unlabelled measurements are (in chronological order) from the Hubble Space Telescope Key Project [18], the Chandra X-ray Observatory [19], and Cosmicflows-2 [20].

of more than 5 standard deviations known as the *Hubble Tension*, and it is one of the great unsolved problems of modern-day cosmology. As data has progressively improved, uncertainties have lowered and the tension has worsened. This is depicted in figure 2.1 which shows the evolution of the late-time (blue) and early-time (red) measurements in the past couple of decades. It is possible that this tension is solely due to unknown systematics, but the far more interesting reason would be new physics beyond the Λ CDM model. This has been investigated for several years with many different types of models, some of which will be presented in chapter 3.

Both early-time and late-time measurements are important to accurately test the prediction power of a new proposed cosmological model. Some of the popular observational experiments widely used in cosmological analyses include the following:

Early-time measurements

- Cosmic Microwave Background (CMB): *Measurements of the photons from their last scattering on electrons, 380.000 years after the Big Bang.*
- Big Bang Nucleosynthesis (BBN): *Measurements of the primordial abundance of light elements produced in the first few minutes after the Big Bang.*
- Primordial gravitational waves: *Measurements of the gravitational wave background in the nanohertz frequency using pulsar timing arrays (PTAs).*

Late-time measurements

- Baryonic Acoustic Oscillations (BAO): *Measurements of the characteristic scale imprinted by oscillations in the photon-baryon plasma prior to recombination.*
- Weak gravitational lensing: *Measurements of matter distributions by observing the distorting effects of weak gravitational lensing.*
- Supernovae type Ia: *Measurements of the distances to faraway galaxies by using SNe type Ia as standard candles.*
- Galaxy surveys: *Measurements of matter distributions and large scale structure by measuring distances to galaxies.*
- Gravitational waves from mergers: *Measurements of gravitational waves due to mergers of stellar black holes and neutron stars using laser interferometry.*

The early-time measurement of the CMB is most relevant to the results presented in this thesis, so we will have a detailed look at this in the next section.

2.2 THE COSMIC MICROWAVE BACKGROUND

One of the most important discoveries in modern cosmology, allowing for measurements of great precision and thus a high constraining power of theoretical models, is the discovery of the cosmic microwave background (CMB). When measuring the radiation received from all directions in space and subtracting that of astrophysical objects and phenomena, what we are left with seems to be isotropic microwave radiation. Not only that, the radiation seems to have a near-perfect thermal blackbody spectrum with a temperature of 2.7255 K. This suggests that the radiation is a relic from a hot and dense state of the universe, and thus further supports the idea of the Big Bang Theory [22].

When examining the radiation more closely, it becomes apparent that there are in fact anisotropies in the distribution. First of all, there is a dipole distortion resulting in one half of the sky being slightly blueshifted and similarly the other half slightly redshifted. This is due to the peculiar velocity of our Solar System relative to the comoving rest frame in which the CMB is isotropic [22]. Furthermore, when removing the dipole distortion, the remaining fluctuations have a very small amplitude and are in the order of 10^{-5} K. Despite this, our observations of these anisotropies are very detailed, with the sensitivity of the Planck satellite being high enough to distinguish temperature variations of only $\sim 10^{-6}$ K [23].

These small anisotropies are a relic from the anisotropies in the early universe. At the age of approximately 380,000 years, the universe cooled enough for protons and electrons to combine into neutral hydrogen atoms. This is known as *recombination*. Before this sudden change, the photons were strongly coupled to the electrons and protons through Thomson scattering,

$$\gamma + e^- \rightarrow \gamma + e^- ,$$

resulting in an opaque state of the universe. When neutral atoms (mostly hydrogen and helium) were formed instead, the photons could no longer scatter off any free charged particles and the universe thus became transparent. The photons emitted from this last scattering form the cosmic microwave background². The anisotropies in the plasma of

² It is microwave radiation today due to it being subjected to cosmological redshift as the universe expands.

coupled photons and baryons are thought to have arisen from quantum fluctuations before the period of *inflation*, and they were the basis of all structure formation in the later evolution of the universe.

2.2.1 SPHERICAL HARMONICS EXPANSION

In order to statistically describe the anisotropies in the CMB, it is useful to describe the fluctuations as a difference between the temperature $T(\hat{n})$ observed in a direction characterised by the unit vector \hat{n} and the mean temperature $\langle T \rangle$ when averaging across the entire sky [24],

$$\Delta T(\hat{n}) \equiv T(\hat{n}) - \langle T \rangle, \quad \langle T \rangle \equiv \frac{1}{4\pi} \int d^2\hat{n} T(\hat{n}). \quad (2.2)$$

We can then expand this quantity in spherical harmonics,

$$\Delta T(\hat{n}) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} a_{\ell m} Y_{\ell}^m(\hat{n}), \quad (2.3)$$

where $Y_{\ell}^m(\hat{n})$ is the spherical harmonic function of degree ℓ and order m . Since we are expanding a real function, the expansion coefficients must satisfy the relation

$$a_{\ell m}^{\star} = a_{\ell -m}, \quad (2.4)$$

where the \star denotes complex conjugate, and we are using the convention of the spherical harmonics without the phase factor of $(-1)^m$ on the complex conjugate, i.e., $Y_{\ell}^m(\hat{n})^{\star} = Y_{\ell}^{-m}(\hat{n})$ [24]. If we assume the universe to be rotationally invariant on averages, such that any average over the possible positions from which to view the CMB (denoted by $\langle \dots \rangle$) will be independent of \hat{n} , it follows that the average of the product of two expansion coefficients must take the form

$$\langle a_{\ell m} a_{\ell' m'}^{\star} \rangle = \delta_{\ell \ell'} \delta_{m m'} C_{\ell}, \quad (2.5)$$

where δ_{ij} is the Kronecker delta and C_{ℓ} is the variance of the $a_{\ell m}$ coefficients for a given ℓ [25]. We can now write the rotationally invariant correlation between the temperature fluctuations of two separate points on the sky described by the unit vectors \hat{n} and \hat{n}' ,

$$\begin{aligned}
 \langle \Delta T(\hat{n}) \Delta T(\hat{n}') \rangle &= \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} C_{\ell} Y_{\ell}^m(\hat{n}) Y_{\ell}^{-m}(\hat{n}') \\
 &= \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} C_{\ell} P_{\ell}(\hat{n} \cdot \hat{n}') ,
 \end{aligned} \tag{2.6}$$

where P_{ℓ} are the *Legendre polynomials* and the dot product $\hat{n} \cdot \hat{n}' = \cos\theta$ describes an angular separation of θ on the sky. We can now exploit the orthogonality of the Legendre polynomials, i.e.,

$$\int_{-1}^1 dx P_{\ell}(x) P_{\ell'}(x) = \delta_{\ell\ell'} \frac{2}{2\ell+1} , \tag{2.7}$$

and find an expression for C_{ℓ} by integrating both sides of equation (2.6) multiplied by P_{ℓ} over the domain,

$$\begin{aligned}
 &\int d^2\hat{n} d^2\hat{n}' P_{\ell}(\hat{n} \cdot \hat{n}') \langle \Delta T(\hat{n}) \Delta T(\hat{n}') \rangle \\
 &= \int d^2\hat{n} d^2\hat{n}' P_{\ell}(\hat{n} \cdot \hat{n}') \sum_{\ell'=0}^{\infty} \frac{2\ell'+1}{4\pi} C_{\ell'} P_{\ell'}(\hat{n} \cdot \hat{n}') ,
 \end{aligned} \tag{2.8}$$

where $d^2\hat{n}$ and $d^2\hat{n}'$ are the solid angles (the surface element on the unit sphere) corresponding to the two unit vectors \hat{n} and \hat{n}' . In order to use the orthogonality, we need to change integration variables on the right-hand side of equation (2.8) such that the integration limits are $[-1, 1]$. The dot product, $\hat{n} \cdot \hat{n}'$, of the unit vectors, is equal to the cosine of their angle of separation, θ , and this quantity has exactly $[-1, 1]$ as its range. We can then change integration variables to this dot product along with the azimuthal angles ϕ and ϕ' and a factor of 2 to ensure the preservation of the total volume of the integral,

$$\begin{aligned}
 &\int d^2\hat{n} d^2\hat{n}' P_{\ell}(\hat{n} \cdot \hat{n}') \langle \Delta T(\hat{n}) \Delta T(\hat{n}') \rangle \\
 &= 2 \int_{-1}^1 d(\hat{n} \cdot \hat{n}') P_{\ell}(\hat{n} \cdot \hat{n}') \sum_{\ell'=0}^{\infty} \frac{2\ell'+1}{4\pi} C_{\ell'} P_{\ell'}(\hat{n} \cdot \hat{n}') \int_0^{2\pi} d\phi \int_0^{2\pi} d\phi' \\
 &= 2(2\pi)^2 \int_{-1}^1 d(\hat{n} \cdot \hat{n}') P_{\ell}(\hat{n} \cdot \hat{n}') \sum_{\ell'=0}^{\infty} \frac{2\ell'+1}{4\pi} C_{\ell'} P_{\ell'}(\hat{n} \cdot \hat{n}') ,
 \end{aligned} \tag{2.9}$$

where each ϕ integral results in a factor of 2π since the expression has no dependence on the azimuthal angles. We can now use the orthogonality of the Legendre polynomials and extract C_ℓ from the sum,

$$\begin{aligned}
 & \int d^2\hat{n} d^2\hat{n}' P_\ell(\hat{n} \cdot \hat{n}') \langle \Delta T(\hat{n}) \Delta T(\hat{n}') \rangle \\
 &= 2(2\pi)^2 \int_{-1}^1 d(\hat{n} \cdot \hat{n}') P_\ell(\hat{n} \cdot \hat{n}')^2 \frac{2\ell+1}{4\pi} C_\ell \\
 &= 2(2\pi)^2 \frac{2}{2\ell+1} \frac{2\ell+1}{4\pi} C_\ell \\
 &= 4\pi C_\ell.
 \end{aligned} \tag{2.10}$$

This leads us to the equation for the C_ℓ coefficients,

$$C_\ell = \frac{1}{4\pi} \int d^2\hat{n} d^2\hat{n}' P_\ell(\hat{n} \cdot \hat{n}') \langle \Delta T(\hat{n}) \Delta T(\hat{n}') \rangle. \tag{2.11}$$

When averaging over the entire sky, we are not accounting for the fact that we only observe the CMB from one point in space while our statistical treatment of it suggests that the averages denoted by $\langle \dots \rangle$ are really averages over possible positions in the universe from which the CMB can be observed [24]. This results in the observed C_ℓ^{obs} coefficients being slightly different since we cannot average over positions from which to view the CMB. The observed coefficients are instead only averaged over the order m of which there are $2\ell+1$ for a given multipole ℓ ,

$$\begin{aligned}
 C_\ell^{\text{obs}} &\equiv \frac{1}{2\ell+1} \sum_{m=-\ell}^{\ell} a_{\ell m} a_{\ell -m} \\
 &= \frac{1}{4\pi} \int d^2\hat{n} d^2\hat{n}' P_\ell(\hat{n} \cdot \hat{n}') \Delta T(\hat{n}) \Delta T(\hat{n}').
 \end{aligned} \tag{2.12}$$

The mean squared relative difference between the observed and theoretical coefficients is known as *cosmic variance*, and it is a fundamental uncertainty in the knowledge that we can get from the C_ℓ coefficients [25]. We can think of it in terms of the number of orders m for a given multipole ℓ . Only a few values of m are available for a small multipole ℓ ($2\ell+1$ to be exact), which means that we only average over a few numbers in this case and thus obtain a result with a large uncertainty. For larger values of ℓ , we will, on the contrary, have many available values of m to average over resulting in a low uncertainty. The multipole ℓ is a

measure of the angular separation on the sky, i.e., $\theta \approx 180^\circ/\ell$, with low values of ℓ corresponding to large angular scales and high values corresponding to small scales. This then translates to the C_ℓ coefficients being much more precisely measured for small angular scales compared to large scales. We can quantify the cosmic variance using the first equality in equation (2.12),

$$\left\langle \left(\frac{C_\ell - C_\ell^{\text{obs}}}{C_\ell} \right)^2 \right\rangle = 1 - 2 \frac{C_\ell^{\text{obs}}}{C_\ell} + \frac{\sum_{m=-\ell}^{\ell} \sum_{m'=-\ell}^{\ell} \langle a_{\ell m} a_{\ell-m} a_{\ell m'} a_{\ell-m'} \rangle}{(2\ell+1)^2 C_\ell^2}. \quad (2.13)$$

For Gaussian temperature fluctuations, the multipole coefficients $a_{\ell m}$ will also have a Gaussian distribution. Furthermore, the mean of the $a_{\ell m}$ coefficients is vanishing, i.e., $\langle a_{\ell m} \rangle = 0$, since the temperature fluctuations have zero mean as well, resulting in the average in the double sum being separable due to Isserlis' theorem [24, 26],

$$\begin{aligned} \langle a_{\ell m} a_{\ell-m} a_{\ell m'} a_{\ell-m'} \rangle &= \langle a_{\ell m} a_{\ell-m} \rangle \langle a_{\ell m'} a_{\ell-m'} \rangle \\ &= \langle a_{\ell m} a_{\ell m'} \rangle \langle a_{\ell-m} a_{\ell-m'} \rangle \\ &= \langle a_{\ell m} a_{\ell-m'} \rangle \langle a_{\ell-m} a_{\ell m'} \rangle. \end{aligned} \quad (2.14)$$

Looking at these terms in context, we can get their individual contributions to the double sum (adopting the $\sum_{mm'}$ notation for the double sum),

$$\begin{aligned} \sum_{mm'} \langle a_{\ell m} a_{\ell-m} \rangle \langle a_{\ell m'} a_{\ell-m'} \rangle &= \sum_{mm'} \delta_{\ell\ell} \delta_{mm'} C_\ell \delta_{\ell\ell} \delta_{m'm'} C_\ell \\ &= (2\ell+1)^2 C_\ell^2, \\ \sum_{mm'} \langle a_{\ell m} a_{\ell m'} \rangle \langle a_{\ell-m} a_{\ell-m'} \rangle &= \sum_{mm'} \delta_{\ell\ell} \delta_{m-m'} C_\ell \delta_{\ell\ell} \delta_{m-m'} C_\ell \\ &= (2\ell+1) C_\ell^2, \\ \sum_{mm'} \langle a_{\ell m} a_{\ell-m'} \rangle \langle a_{\ell-m} a_{\ell m'} \rangle &= \sum_{mm'} \delta_{\ell\ell} \delta_{mm'} C_\ell \delta_{\ell\ell} \delta_{mm'} C_\ell \\ &= (2\ell+1) C_\ell^2. \end{aligned} \quad (2.15)$$

We can now substitute this into equation (2.13) and obtain an expression for the cosmic variance,

$$\begin{aligned} \left\langle \left(\frac{C_\ell - C_\ell^{\text{obs}}}{C_\ell} \right)^2 \right\rangle &= 1 - 2 \frac{C_\ell^{\text{obs}}}{C_\ell} + \frac{(2\ell + 1)^2 C_\ell^2 + 2(2\ell + 1) C_\ell^2}{(2\ell + 1)^2 C_\ell^2} \\ &= 2 - 2 \frac{C_\ell^{\text{obs}}}{C_\ell} + \frac{2}{2\ell + 1} \\ &\approx \frac{2}{2\ell + 1}, \end{aligned} \quad (2.16)$$

where we have used the fact that $C_\ell^{\text{obs}}/C_\ell$ is very close to unity.

2.2.2 THE CMB POWER SPECTRA

The C_ℓ coefficients represent the contribution to the total variance of the field from anisotropy on angular scales corresponding to the multipole ℓ . It is, however, customary to multiply the coefficients with a factor of $\ell(\ell + 1)/(2\pi)\langle T \rangle^2$ to account for the increased number of modes for higher values of ℓ . The factor of $\lambda = \ell(\ell + 1)$ arises from the fact that the spherical harmonics are the eigenfunctions of the Laplacian on the surface of a sphere with eigenvalues $-\lambda$, i.e.,

$$r^2 \nabla^2 Y_\ell^m(\theta, \phi) = -\ell(\ell + 1) Y_\ell^m(\theta, \phi), \quad (2.17)$$

and the factor of 2π is a conventional normalisation to ensure consistency with the units. The quantity that we usually depict is then given by

$$\mathcal{D}_\ell^{TT} = \frac{\ell(\ell + 1) C_\ell}{2\pi} \langle T \rangle^2, \quad (2.18)$$

where the superscript TT refers to this representing the variance of the temperature fluctuations.

This quantity can now be used to make the *CMB temperature power spectrum* describing the temperature anisotropies on different scales. This is shown in figure 2.2 where the observed coefficients are plotted along with the theoretical prediction best describing the data within the context of the Λ CDM model. We can clearly see the effects of cosmic variance here since the errors on larger scales are much more profound than on small scales. There is a very noticeable wave struc-

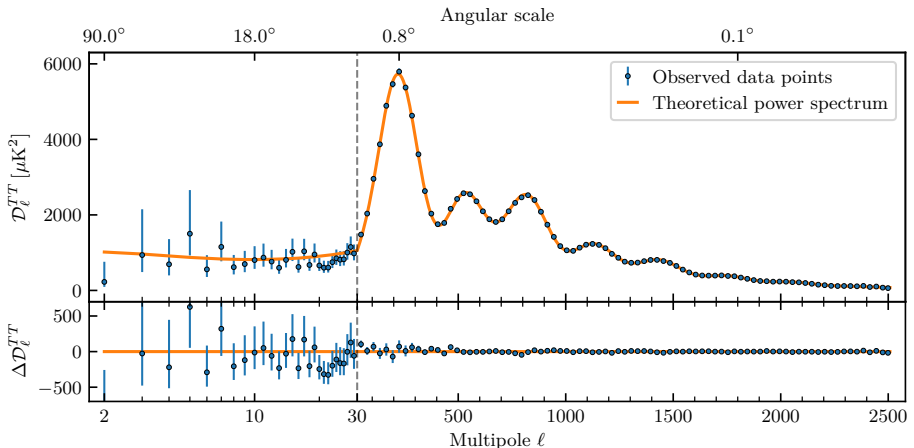


Figure 2.2: The angular power spectrum for CMB temperature fluctuations. The observed data from Planck [21] is shown as blue points with error bars and the best-fitting theoretical prediction within the Λ CDM model is shown as an orange line. The vertical dashed line at $\ell = 30$ marks the transition between a logarithmic axis and a linear axis.

ture in the power spectrum which provides a very detailed description of the universe at the time of recombination.

Prior to last scattering, the plasma of photons, electrons, and nucleons was not at all static or homogeneous. Overdense regions in the plasma due to slight perturbations collapse gravitationally and grow until the plasma is heated enough for the pressure to become too great and start counteracting the gravitational collapse, leading to re-expansion. The plasma will then cool again until the pressure drops and gravity once again can collapse the plasma. This leads to oscillations or *sound waves* in the plasma on different scales and these oscillations are exactly what is captured in the CMB power spectrum. At the time of last scattering, the oscillations stopped and the modes at their extrema at this time are the ones exhibiting the most anisotropy in the power spectrum, i.e., the peaks. The first peak corresponds to the largest mode that never experienced a re-expansion by the pressure, so this marks the maximum proper distance a sound wave could have travelled between the Big Bang and last scattering, also known as the *sound horizon*, r_s [22]. The second peak is likewise a mode that has collapsed and expanded once, the third peak is a mode that has collapsed, expanded and then collapsed again, and so on.

We notice as well that higher peaks in the power spectrum are dampened and the spectrum seems to flatten for larger multipoles. The reason for this damping effect, known as *diffusion damping*, can be found in the coupling between the photons, electrons, and nucleons in the plasma before recombination. The universe was not perfectly opaque prior to last scattering since the photons did have a mean free path and the damping occurs at physical scales comparable to this mean free path. The photons are able to mix on these scales and it averages out the anisotropies in the temperature field [25].

Apart from the intrinsic temperature fluctuations, there are a few more effects affecting the CMB power spectrum that we may observe today [24]:

- *The Doppler effect* affects the emitted photons from last scattering due to fluctuations in the velocity field.
- *The Sachs-Wolfe effect* is the gravitational redshifting and blueshifting of the photons from last scattering due to fluctuations in the gravitational potential.
- *The integrated Sachs-Wolfe effect* affects the photons after last scattering and up until today. When the universe expands, large gravitational wells will change over time fast enough for a photon entering and exiting to experience a net blueshift or redshift. This only occurs when matter is not the dominating contribution to the energy density. This means that there are two contributions, the *late integrated Sachs-Wolfe effect* and the *early integrated Sachs-Wolfe effect*, for the dark energy domination in the late universe and the radiation domination still in effect immediately after last scattering, respectively.

These all contribute to the observed CMB power spectrum as depicted in figure 2.3.

Apart from the temperature fluctuations, there are also polarisation fluctuations in the CMB. These arise from the Thomson scattering at the time of recombination. The scattering allows for transverse radiation to be completely transmitted while radiation parallel to the outgoing direction is stopped [25]. In the isotropic case, photons will scatter off the charged particle from all directions and the net result will be unpolarised radiation. In order to produce polarisations we need a

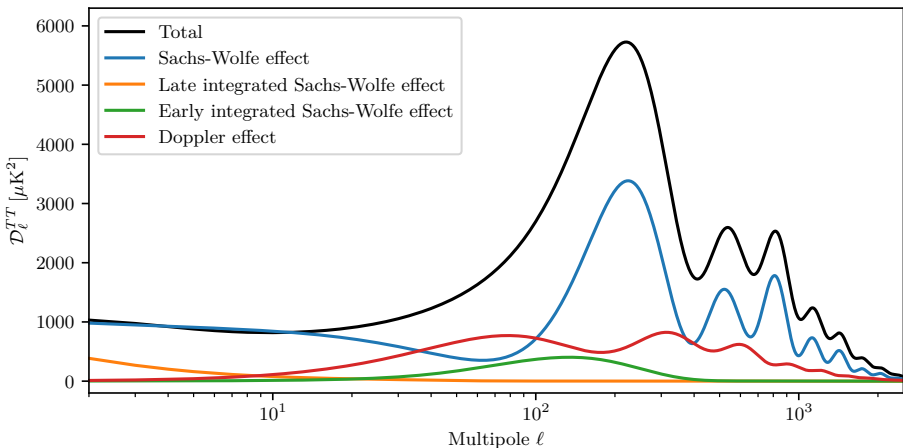


Figure 2.3: Different contributions to the angular power spectrum for CMB temperature fluctuations. The contributions have been computed using CLASS [27].

quadrupole anisotropic pattern (since a dipole only leads to cancellations between radiation from hot and cold regions of similar departure from the mean temperature), and this quadrupole moment is small prior to recombination. Only near the end of recombination can a significant quadrupole moment form due to diffusion of photons between hot and cold regions.

Only the E -mode polarisation is sourced by scalar perturbations and we need tensor perturbations in order to have non-zero B -mode polarisation. Because the signal of tensor perturbations is much smaller than that of scalar perturbations, they are difficult to measure and that holds true for the B -mode polarisation as well. Tensor perturbations are fundamentally different from scalar perturbations in the sense that they correspond to primordial gravitational waves generated during inflation while scalar perturbations are perturbations in density and potential of matter and energy. Although tensor perturbations also contribute to the E -mode polarisation, it is not easy to split the observed signal up into the contributions from scalar and tensor perturbations, so we cannot currently conclude a detection of tensor perturbations (and thus primordial gravitational waves) from the E -mode polarisation alone. B -mode polarisation in the CMB can, however, only have a contribution from tensor perturbations, so detection of this would be a clear indication of primordial gravitational waves and thereby also inflationary models producing gravitational waves [25]. We have yet to obtain a

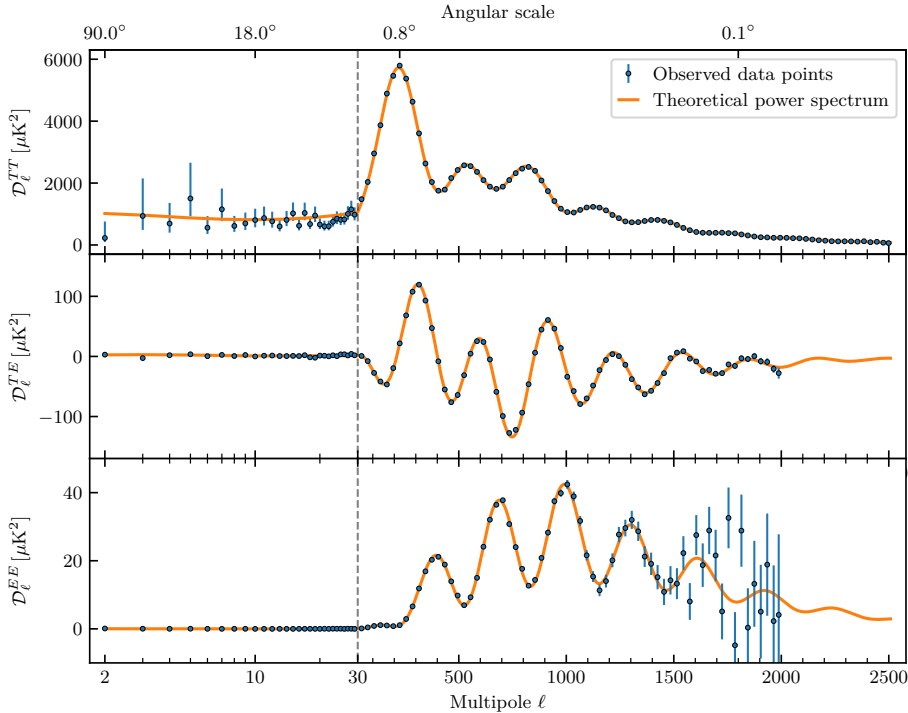


Figure 2.4: Angular power spectra for temperature fluctuations (top), E -mode polarisation fluctuations (bottom), and a cross-correlation between the two (middle). The figure includes the observed data from Planck [21] as well as the theoretical spectra for the set of parameters (within the Λ CDM model) that fit all three sets of data best. The vertical dashed line at $\ell = 30$ marks the transition between a logarithmic axis and a linear axis.

clear detection of the B -mode polarisation in support of tensor perturbations (foreground effects and gravitational lensing can induce B -modes as well) although the *tensor-to-scalar ratio* has been constrained by various experiments, including the BICEP/Keck experiments [28]. It is thus more common to only include the E -mode polarisation along with a cross-correlation between fluctuations in E -mode polarisation and temperature fluctuations. The three different CMB power spectra, commonly referred to as TT , TE , and EE , are shown in figure 2.4.

2.3 HOMOGENOUS AND ISOTROPIC COSMOLOGY

On large scales, the universe seems quite homogeneous and isotropic given the fact that the matter distributions look similar no matter where

we look³. It is thus useful to consider a completely smooth, homogeneous, and isotropic universe in the context of general relativity.

The *metric* used to describe an expanding universe is the Friedmann-Robertson-Walker (FRW) metric relating the coordinate distance, dx^μ , and the physical distance, ds , through

$$ds^2 = \sum_{\mu,\nu=0}^3 g_{\mu\nu} dx^\mu dx^\nu, \quad (2.19)$$

where the Greek letters, μ and ν , as indices represent the components of four-vectors with the 0th component being the temporal component and the three other components being the spatial components, while Latin letters as indices are used to denote just the spatial components. The FRW metric, $g_{\mu\nu}$, in an expanding, flat universe⁴ is given as [25]

$$g_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & a^2(t) & 0 & 0 \\ 0 & 0 & a^2(t) & 0 \\ 0 & 0 & 0 & a^2(t) \end{pmatrix}, \quad (2.20)$$

with $a(t)$ being the *scale factor* of the universe with a value of 1 at the time today, t_0 , i.e., $a_0 \equiv a(t_0) = 1$. The metric describes the geometry of the universe, but we still need something to relate its geometry and its energy density. This is exactly the role of the *Einstein equations*:

$$G_{\mu\nu} \equiv R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}\mathcal{R} = 8\pi GT_{\mu\nu}, \quad (2.21)$$

where $G_{\mu\nu}$ is the Einstein tensor, $R_{\mu\nu}$ is the *Ricci tensor*, \mathcal{R} is the *Ricci scalar*, G is Newton's gravitational constant, and $T_{\mu\nu}$ is the *stress-energy* tensor. The Ricci scalar is a contraction of the Ricci tensor which in turn is purely related to the geometry depending on the metric and its derivatives. The left-hand side thus describes the geometry of the universe, while the right-hand side describes the energy contents.

³ With distances confined to the late universe where we expect similar amounts of structure.

⁴ Including curvature is certainly possible but will not be treated in this thesis.

In a perfect isotropic fluid, we cannot have any net transport of momentum, and the stress-energy tensor thus becomes diagonal (in its mixed tensor form to avoid factors of the scale factor),

$$T^\mu{}_\nu = \begin{pmatrix} -\rho & 0 & 0 & 0 \\ 0 & P & 0 & 0 \\ 0 & 0 & P & 0 \\ 0 & 0 & 0 & P \end{pmatrix}, \quad (2.22)$$

where ρ is the energy density of the fluid and P is the pressure. From this, we can now derive two useful equations. The first one is obtained from the $(0,0)^{\text{th}}$ component of the Einstein equations [29],

$$H(t) \equiv \left(\frac{\dot{a}(t)}{a(t)} \right)^2 = \frac{8\pi G}{3} \rho(t), \quad (2.23)$$

and this is the *Friedmann equation* for a flat universe, where we have defined the Hubble parameter, $H(t)$, describing the rate of expansion at time t , and the dot ($\dot{}$) denotes the derivative with respect to proper time. The second equation is obtained from the conservation of the stress-energy tensor, which corresponds to the vanishing of its *covariant derivative* [25]. From the $\nu = 0$ components, we have that,

$$T^\mu{}_{0;\mu} = \frac{\partial T^\mu{}_0}{\partial x^\mu} + \Gamma^\mu{}_{\alpha\mu} T^\alpha{}_0 - \Gamma^\alpha{}_{0\mu} T^\mu{}_\alpha = 0, \quad (2.24)$$

where

$$\Gamma^\mu{}_{\alpha\beta} = \frac{g^{\mu\nu}}{2} \left(\frac{\partial g_{\alpha\nu}}{\partial x^\beta} + \frac{\partial g_{\beta\nu}}{\partial x^\alpha} - \frac{\partial g_{\alpha\beta}}{\partial x^\nu} \right) \quad (2.25)$$

denotes the *Christoffel symbol*. This leads to the *continuity equation*,

$$\frac{\partial \rho}{\partial t} + 3 \frac{\dot{a}}{a} (\rho + P) = 0. \quad (2.26)$$

We now have two equations featuring the quantities, a , ρ and P , so we need one more in order to solve for the evolution of the universe. This

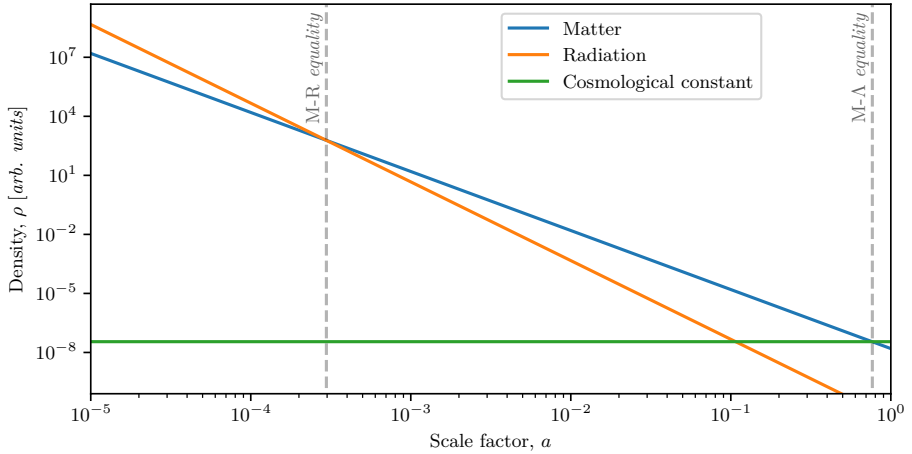


Figure 2.5: The evolution of the energy densities of matter, radiation, and a cosmological constant in the Λ CDM model. Different components dominate different epochs of the evolution. The dashed vertical lines signify the scale factors of the matter-radiation and matter- Λ equalities.

last equation relates the pressure and the energy density, and is known as the equation of state,

$$P = w\rho, \quad (2.27)$$

where the equation-of-state parameter w depends on the fluid. For nonrelativistic matter, we have that $w_M = 0$ since it exhibits vanishing pressure, for radiation, we have that $w_R = 1/3$, and for a cosmological constant, we have that $w_\Lambda = -1$, thus resulting in negative pressure. The cosmological constant (as an extra term in the Einstein tensor) may equivalently be interpreted as dark energy, even though more elaborate models of dark energy exist whose effects do not equal those of a cosmological constant. For the remaining part of this chapter, we will use these two terms interchangeably about the component with $w_\Lambda = -1$.

2.3.1 BACKGROUND EVOLUTION

With these three equations in our toolkit, we may now describe the expansion and the background evolution of universes of different composi-

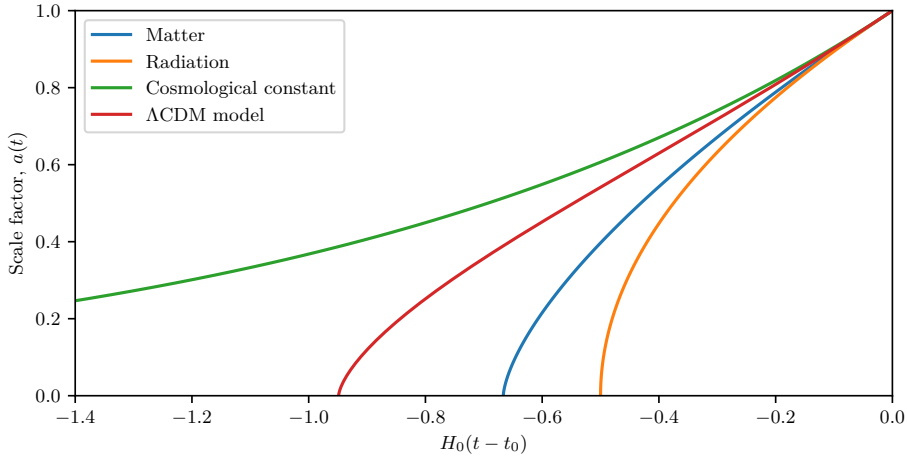


Figure 2.6: The evolution of the scale factor, $a(t)$, for universes of different compositions. It is shown for the single-component universes consisting of matter, radiation, and a cosmological constant, along with that of the Λ CDM model with parameters in best agreement with observations.

tions. From equations (2.26) and (2.27) we get the following evolution of the energy densities of different components:

$$\begin{aligned}
 \rho_M(t) &= \rho_M(t_0) a(t)^{-3}, \\
 \rho_R(t) &= \rho_R(t_0) a(t)^{-4}, \\
 \rho_\Lambda(t) &= \rho_\Lambda(t_0), \\
 \rho'(t) &= \rho'(t_0) a(t)^{-3(1+w')},
 \end{aligned} \tag{2.28}$$

where we have included the energy density ρ' of a general component with equation-of-state parameter w' . Cosmological observations tell us that matter contributes $\sim 30\%$ to the total energy density today while radiation contributes a negligible amount of less than a $10^{-2}\%$ and dark energy (or cosmological constant) contributes the remaining $\sim 70\%$. In order to achieve this according to the evolutions described in equation 2.28, we must have had a period of radiation domination in the early universe followed by a period of matter domination and lastly, a period dominated by dark energy. This is shown in figure 2.5 where the individual energy densities are graphed as a function of the scale factor.

Using the Friedmann equation we can furthermore compute the evolution of the scale factor for universes of different compositions. For a universe with only a single component, we may insert the expression

of the component's energy density in terms of the scale factor into the Friedmann equation,

$$\left(\frac{\dot{a}(t)}{a(t)}\right)^2 = \frac{8\pi G}{3}\rho_0 a(t)^{-3(1+w)}, \quad (2.29)$$

where $\rho_0 = \rho(t_0)$. Solving this differential equation yields an expression for the scale factor as a function of time,

$$a(t) = \left(\frac{t}{t_0}\right)^{2/(3+3w)}, \quad w \neq -1. \quad (2.30)$$

The scale factor for a universe with only a cosmological constant cannot be described by this equation. We can, however, derive the expression for the scale factor directly from the Friedmann equation by setting $w = -1$. This results in the following expression valid for a flat universe with only a cosmological constant,

$$a(t) = e^{H_0(t-t_0)}, \quad H_0 = \left(\frac{8\pi G}{3}\rho_{\Lambda,0}\right)^{1/2}. \quad (2.31)$$

Universes of two components can also (to some extent) be treated analytically, thus making it possible to describe the evolution around *matter-radiation equality* and *matter-Λ equality*. A universe with three or more components, however, requires a numerical solution of the Friedmann equation. The scale factor as a function of time can be seen in figure 2.6 for different single-component universes as well as for the three-component model matching our observations.

2.4 INHOMOGENEITIES AND ANISOTROPIES

It is clear from the fact that you are reading this thesis, that our universe is not homogeneous and isotropic on all scales. If that were the case, no galaxies, stars, planets, or theses would exist. We thus owe our very existence to the effects described in this section.

Small perturbations in the early universe have grown into all the galaxies, clusters, etc. that we observe today through the process of *structure formation*. In this section, we will have a look at the mechanisms behind this phenomenon and describe them mathematically using linear perturbation theory.

2.4.1 THE BOLTZMANN EQUATION

When statistically describing thermodynamical systems away from equilibrium, it is necessary to consider the *phase space* of the system. This is the six-dimensional space described by three dimensions of position and three dimensions of momentum. The *phase space distribution function*, $f(x^i, p_j, \tau)$, of the particles in the system describes the statistical state of the system at conformal time τ , and the integral of it over the entire six-dimensional phase space gives the number of particles in the system. The distribution function can give us all the statistical information of the system, so we can similarly find expressions of thermodynamical quantities, such as the number density, the energy density, and the pressure, as weighted integrals of the distribution function. Bosons and fermions behave differently in thermodynamical systems due to the *Pauli exclusion principle* for fermions, and in equilibrium, they follow a *Bose-Einstein distribution* (− sign) or a *Fermi-Dirac distribution* (+ sign), respectively [30],

$$f(\epsilon) = \frac{g_s}{h_P^3} \frac{1}{e^{\epsilon/(k_B T)} \pm 1}, \quad (2.32)$$

where $\epsilon = \sqrt{q^2 + a^2 m^2}$ with q as the magnitude of the *comoving* momentum given by the product of the scale factor and magnitude of the proper momentum, h_P is the Planck constant, k_B is the Boltzmann constant, g_s is the number of spin degrees of freedom, and T denotes the temperature of the particles.

When a system is out of equilibrium, the evolution of the phase space distribution is governed by the *Boltzmann equation* [25],

$$\frac{df}{dt} = C[f], \quad (2.33)$$

where the right-hand side is a functional expressing all the possible collision terms of the distribution function. In the case of no collisions between particles, e.g. for cold dark matter, the right-hand side is zero. It is useful to express the left-hand side in terms of its partial derivatives using the chain rule,

$$\frac{df}{d\tau} = \frac{\partial f}{\partial \tau} + \frac{\partial f}{\partial x^i} \cdot \frac{dx^i}{d\tau} + \frac{\partial f}{\partial q} \frac{dq}{d\tau} + \frac{\partial f}{\partial \hat{n}_i} \cdot \frac{d\hat{n}_i}{d\tau} = C[f], \quad (2.34)$$

where we have split the comoving momentum dependence into its magnitude, q , and its direction \hat{n}^i . Similarly to how we can integrate the distribution function with different weights to obtain thermodynamical quantities, we can obtain conservation equations by taking the moments of the Boltzmann equation.

2.4.2 LINEAR PERTURBATION THEORY IN COSMOLOGY

Equipped with the Boltzmann equation and Einstein equations, we can now introduce perturbations to the homogenous and isotropic solution. In the *conformal Newtonian gauge*, these perturbations are described by two scalar potentials, ψ and ϕ , modifying the metric [30], i.e.,

$$g_{\mu\nu} = \begin{pmatrix} -(1+2\psi) & 0 & 0 & 0 \\ 0 & a^2(1-2\phi) & 0 & 0 \\ 0 & 0 & a^2(1-2\phi) & 0 \\ 0 & 0 & 0 & a^2(1-2\phi) \end{pmatrix}. \quad (2.35)$$

We furthermore introduce perturbations to the stress-energy tensor as $T^\mu_\nu = \bar{T}^\mu_\nu + \delta T^\mu_\nu$, where the bar ($\bar{}$) denotes the background quantity, and δ denotes a perturbation to the subsequent quantity. This introduces off-diagonal components, and the perturbed stress-energy tensor becomes

$$\begin{aligned} T^0_0 &= -(\bar{\delta} + \delta\rho), \\ T^0_i &= (\bar{\delta} + \bar{P})v_i, \\ T^i_j &= (\bar{P} + \delta P)\delta^i_j + \Sigma^i_j, \end{aligned} \quad (2.36)$$

where v_i is a small coordinate velocity of the fluid and Σ^i_j is the traceless component of T^i_j .

Using the perturbed metric and the perturbed stress-energy tensor, one can compute the perturbed Einstein equations (most conveniently expressed in Fourier space with k as the wavenumber). Given small perturbations, it is sufficient to linearise the equations by only keeping terms up to first order. These equations relate the metric perturbations with perturbations of matter and energy, thus describing how inhomogeneities in matter and energy distort spacetime and similarly how mat-

ter and energy distributions evolve due to these distortions. Defining the quantities $\delta \equiv \delta\rho/\bar{\rho}$, $\theta \equiv ik^j v_j$, and $\sigma \equiv -(\hat{k}_i \hat{k}_j - \delta_{ij}/3)\Sigma^i_j/(\bar{\rho} + \bar{P})$, where \hat{k}_i is the direction of the wave vector, and using $w = P/\rho$, we can arrive at the following two equations:

$$\begin{aligned}\dot{\delta} &= -(1+w)(\theta - 3\dot{\phi}) - 3\frac{\dot{a}}{a}\left(\frac{\delta P}{\delta\rho} - w\right)\delta, \\ \dot{\theta} &= -\frac{\dot{a}}{a}(1-3w)\theta - \frac{\dot{w}}{1+w}\theta + \frac{\delta P/\delta\rho}{1+w}k^2\delta - k^2\sigma + k^2\psi,\end{aligned}\tag{2.37}$$

which are valid for any uncoupled fluid, i.e., no collisions in the Boltzmann equation. The dependence on the anisotropic stress, σ , in the equation for the divergence, θ , of the fluid velocity tells us that this is not a closed set of equations. Additional equations for higher moments are needed, and just as the equation for δ includes θ and the equation for θ includes σ , the equation for the l^{th} moment will generally depend on the $(l+1)^{\text{th}}$ moment. The equations for all moments can be found through the Boltzmann equation, which is also needed to introduce collisions into the fluid.

Cold dark matter, however, is sufficiently described by the two equations above since we can treat it as a pressureless fluid, i.e., $w = \dot{w} = 0$ and $\sigma = 0$,

Perturbation equations for cold dark matter

$$\begin{aligned}\dot{\delta}_{\text{cdm}} &= -\theta_{\text{cdm}} + 3\dot{\phi}, \\ \dot{\theta}_{\text{cdm}} &= -\frac{\dot{a}}{a}\theta_{\text{cdm}} + k^2\psi,\end{aligned}\tag{2.38}$$

where the subscript “cdm” refers to cold dark matter.

Using the Boltzmann equation requires a slightly different approach to the perturbations. We now introduce a perturbation Ψ to the distribution function instead,

$$f(x^i, q, \hat{n}_j, \tau) = f_0(q) (1 + \Psi(x^i, q, \hat{n}_j, \tau)).\tag{2.39}$$

Using this perturbation and the Boltzmann equation as written in equation (2.34), we can arrive at the following first-order perturbation equation in Fourier space [30],

$$\frac{\partial \Psi}{\partial \tau} + i \frac{q}{\epsilon} (\mathbf{k} \cdot \hat{n}) \Psi + \frac{d \ln f_0}{d \ln q} \left[\dot{\phi} - i \frac{\epsilon}{q} (\mathbf{k} \cdot \hat{n}) \psi \right] = \frac{1}{f_0} C[f]. \quad (2.40)$$

This equation may be used to derive perturbation equations for the other components, such as neutrinos, baryons, and photons.

Neutrinos with mass m can be treated by expanding the perturbation Ψ in Legendre polynomials, P_l ,

$$\Psi(\mathbf{k}, \hat{n}, q, \tau) = \sum_{l=0}^{\infty} (-i)^l (2l+1) \Psi_l(\mathbf{k}, q, \tau) P_l(\mathbf{k} \cdot \hat{n}). \quad (2.41)$$

This leads to equations for infinitely many Ψ_l coefficients, but $l \geq 2$ can be expressed through a recursion relation,

Perturbation equations for neutrinos

$$\begin{aligned} \dot{\Psi}_0^\nu &= -\frac{qk}{\epsilon} \Psi_1^\nu - \dot{\phi} \frac{d \ln f_0}{d \ln q}, \\ \dot{\Psi}_1^\nu &= \frac{qk}{3\epsilon} (\Psi_0^\nu - 2\Psi_2^\nu) - \frac{\epsilon k}{3q} \psi \frac{d \ln f_0}{d \ln q}, \\ \dot{\Psi}_l^\nu &= \frac{qk}{(2l+1)\epsilon} [l\Psi_{l-1}^\nu - (l+1)\Psi_{l+1}^\nu], \quad l \geq 2, \end{aligned} \quad (2.42)$$

where the “ ν ” superscript denotes that this is the *Boltzmann hierarchy* for neutrinos.

Photons are simpler in the sense that one can integrate out the momentum dependence due to them being massless, but they are also more complicated since they couple to baryons through Thomson scattering prior to recombination. This means that there are non-zero collision terms that depend on polarisation. We define the quantities $F_\gamma(\mathbf{k}, \hat{n}, \tau)$ and $G_\gamma(\mathbf{k}, \hat{n}, \tau)$ which are the momentum averaged perturbation of the phase space distribution summed over polarisations, and the difference

of two linear polarisation components, respectively [30]. The right-hand side of the Boltzmann equation then reads,

$$C[f] = \left(\frac{\partial F_\gamma}{\partial \tau} \right)_C + \left(\frac{\partial G_\gamma}{\partial \tau} \right)_C. \quad (2.43)$$

Both F_γ and G_γ can be expanded in Legendre polynomials and the collision terms can thus be expressed in terms of the expansion coefficients, F_l^γ and G_l^γ , the velocity divergences of photons and baryons, θ_γ and θ_b , the number density of electrons, n_e , and the cross-section for Thomson scattering, σ_T . Including these terms in the Boltzmann equations likewise results in infinitely many coupled differential equations,

Perturbation equations for photons

$$\begin{aligned} \dot{\delta}_\gamma &= -\frac{4}{3}\theta_\gamma + 4\dot{\phi}, \\ \dot{\theta}_\gamma &= k^2 \left(\frac{1}{4}\delta_\gamma - \sigma_\gamma \right) + k^2\psi + an_e\sigma_T(\theta_b - \theta_\gamma), \\ \dot{F}_2^\gamma &= 2\dot{\sigma}_\gamma = \frac{8}{15}\theta_\gamma - \frac{3}{5}kF_3^\gamma + an_e\sigma_T \left[\frac{1}{10}(G_0^\gamma - G_2^\gamma) - \frac{9}{5}\sigma_\gamma \right], \\ \dot{F}_l^\gamma &= \frac{k}{2l+1} [lF_{l-1}^\gamma - (l+1)F_{l+1}^\gamma] - an_e\sigma_T F_l^\gamma, \quad l \geq 3, \\ \dot{G}_l^\gamma &= \frac{k}{2l+1} [lG_{l-1}^\gamma - (l+1)G_{l+1}^\gamma] \\ &\quad - an_e\sigma_T \left[G_l^\gamma - \frac{1}{2}(F_2^\gamma + G_0^\gamma + G_2^\gamma) \left(\delta_{l,0} + \frac{\delta_{l,2}}{5} \right) \right], \end{aligned} \quad (2.44)$$

where we notice that the equations for δ_γ and θ_γ are exactly what was expected from equations (2.37) apart from the extra term describing the effects of Thomson scattering.

Baryons are also influenced by collisions due to their coupling to photons before recombination. Baryons are, however, also nonrelativistic matter, so we only get 2 equations as with cold dark matter. We can simply use equations (2.37) to find the equations for baryons and then

add the coupling term due to Thomson scattering. Looking at the equation for θ_γ , we can derive the necessary extra term in order to satisfy momentum conservation. The equations for baryons then become

Perturbation equations baryons

$$\begin{aligned}\delta_b &= -\theta_b + 3\dot{\phi}, \\ \dot{\theta}_b &= -\frac{\dot{a}}{a}\theta_b + c_s^2 k^2 \delta_b + k^2 \psi + \frac{4\bar{\rho}_\gamma}{3\bar{\rho}_b} a n_e \sigma_T (\theta_\gamma - \theta_b),\end{aligned}\tag{2.45}$$

where the sound speed $c_s^2 = \delta P_b / \delta \rho_b = w \ll 1$.

These equations are all we need in order to describe linear structure formation in the Λ CDM model. Of course, we cannot use an infinite set of coupled differential equations in practice, but luckily, the higher moments usually contribute less and less, so we are able to obtain nice results by truncating the Boltzmann hierarchies at some maximum multipole order, l_{\max} .

Using $l_{\max} = 17$ has been shown to work quite well for the neutrino hierarchy and is the default in the popular Einstein–Boltzmann solver code, CLASS [27], with which we obtain the resulting evolution of the density perturbations, δ , for the different components at a scale of $k = 1.0 \text{ Mpc}^{-1}$, shown in figure 2.7. Here, we can see the coupling between the photons and baryons prior to recombination, and afterwards the baryons evolve similarly to cold dark matter since the gravitational wells formed by the cold dark matter attract the baryons. The neutrinos free stream from the moment they decouple in the very early universe and the photons do as well after recombination. As with the CMB power spectrum, we also here notice the effects of the photon diffusion prior to recombination which decreases the perturbations of photons and baryons until they decouple.

We have only focussed on structure formation to linear order in this section and this is a very good tool to describe the universe on large scales and it is especially useful to describe the physics of the early universe. As structure grows, though, perturbations collapse due to *Jeans instability* and this leads to non-linear effects that we cannot describe with the equations presented in this section. The distribution of matter that we can observe is highly non-linear and we thus need a way to de-

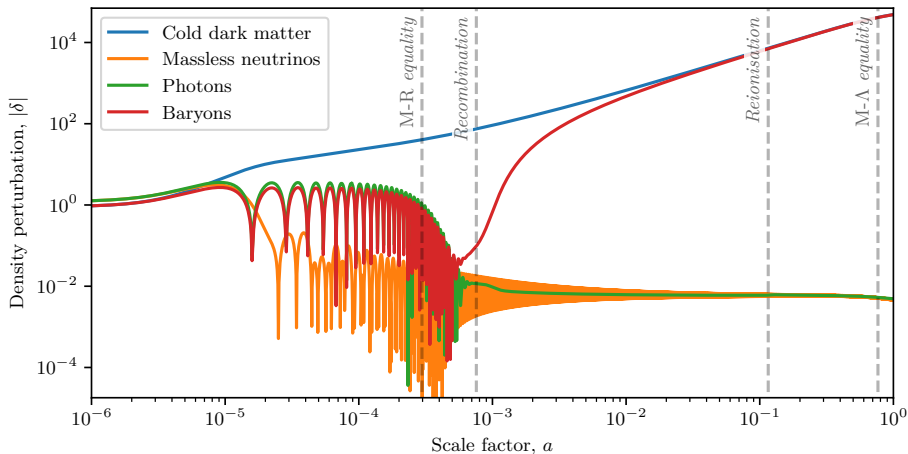


Figure 2.7: The evolution of the density perturbations of cold dark matter, massless neutrinos, photons, and baryons. These have been computed using CLASS [27]. The vertical dashed lines signify the scale factors where the matter-radiation and matter- Λ equalities, recombination, and reionisation happen.

scribe this. It is possible to do perturbation theory of higher orders, but there is a limit to the results one might feasibly get this way. It is much more common to rely on numerical simulations of interacting particles, so-called *N-body codes*. These simulations are, however, much more computationally demanding than the relatively simple computations of linear perturbation theory, so obtaining a theoretical distribution of matter from simulations can be challenging and slow. Although an interesting topic, a further discussion of non-linear structure formation is beyond the scope of this thesis.

DECAYING DARK MATTER

In an attempt to describe observations better than the Λ CDM model is able to, one can make an extension to the model and test whether or not data prefers the newly introduced extension. One such extension is giving the cold dark matter the ability to decay into invisible *dark radiation*. This model (and generalisations hereof) was initially the foundation for my PhD project. My bachelor’s project was to explore this model with the newest CMB data from the Planck collaboration’s 2018 data release, and once I started my PhD, this quickly turned into the paper

- *Andreas Nygaard, Thomas Tram, and Steen Hannestad. “Updated constraints on decaying cold dark matter.” In: JCAP 05 (2021), p. 017. doi: 10.1088/1475-7516/2021/05/017. arXiv: 2011.01632 [astro-ph.CO],*

which constitutes the majority of this chapter. The remaining part of the chapter discusses more elaborate models of decaying dark matter and this consists of an excerpt from a review chapter that I co-authored for a book on the Hubble tension,

- *Andreas Nygaard, Emil Brinch Holm, Thomas Tram, and Steen Hannestad. “Decaying Dark Matter and the Hubble Tension.” In: The Hubble Constant Tension. Ed. by Eleonora Di Valentino and Dillon Brout. Springer Series in Astrophysics and Cosmology. Springer, July 2024. Chap. 25, pp. 481–492. isbn: 978-981-99-0176-0, 978-981-99-0179-1, 978-981-99-0177-7. doi: 10.1007/978-981-99-0177-7.*

The introduction from this review chapter summarises the ideas behind the decaying dark matter models wonderfully, and I will thus include it here.

Beginning of excerpt of reference [5]

Although the nature of dark matter remains unknown, a brief look at the Standard Model contents of the universe reveals that a majority of the known particles are unstable and decay. By analogy, a natural question to ask is whether dark matter may decay on cosmological timescales. Decays of dark matter into electromagnetically interacting particles are strongly constrained by CMB observations [31]. Decays into a dark sector, so-called *invisible decays*, on the other hand, are much less constrained because no direct observation channel exists. Nonetheless, there are strong constraints on models that assume *all* of dark matter to decay on cosmological timescales (e.g., the simple observation that we observe it today) [32, 33]. However, these constraints may always be evaded by considering a scenario where only a fraction of the dark matter decays invisibly. It is this class of models we study in this chapter.

There exist several phenomenological models of invisibly decaying dark matter, largely varying in their assumptions on the decaying particle (cold or warm) and on the decay products (massive or massless, two- or many-body decays). In this chapter, we review constraints on the three most studied models:

DCDM→DR: Decaying cold dark matter (DCDM) decaying into dark radiation (DR). Presented from section 3.1 to section 3.7.1.

DCDM→DR+WDM: Decaying cold dark matter decaying into warm dark matter (WDM) *and* dark radiation. Presented in section 3.7.2.

DWDM→DR: Decaying *warm* dark matter (DWDM) decaying into dark radiation. Presented in section 3.7.3.

Why are these models relevant to the Hubble tension? Figure 3.1 shows the relative change in the Hubble parameter $H(a)$ as a function of the scale factor a for the DWDM→DR and DCDM→DR models, as well as a model with additional relativistic degrees of freedom, ΔN_{eff} . The abundances of the species have been fixed such that they all contribute $\Delta N_{\text{eff}} = 0.5$ to the number of relativistic degrees of freedom at $a = 1$, the acoustic scale is fixed (thus letting H_0 vary) to simulate observational constraints, and the DWDM and DCDM models have the same decay rate Γ .

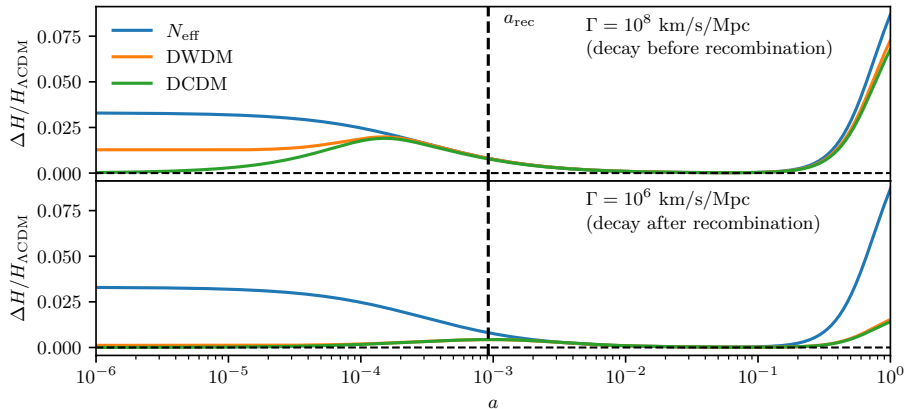


Figure 3.1: Hubble parameter H as a function of scale factor a , relative to its value in the Λ CDM model, for models of decaying cold dark matter (DCDM), decaying warm dark matter (DWDM), and additional relativistic degrees of freedom, ΔN_{eff} . Taken from Ref. [34].

In the top panel, Γ is such that both models decay before recombination. In this case, the energy density is greater than that of Λ CDM, increasing the value of the Hubble parameter. As the species decay, their values of H converge to that of the N_{eff} model. In all three models, the increase in $H(a)$ prior to recombination decreases the sound horizon at recombination, $r_s(a_*)$. Since observations essentially fix the acoustic scale to $\theta_s = r_s(a_*)/D_A(a_*)$, where $D_A(a_*)$ is the angular diameter distance to recombination, this results in a decrease of $D_A(a_*)$, which manifests in an increased value of H_0 , as can be seen in the figure¹. The model of extra relativistic degrees of freedom obtains the largest increase in H_0 but is known to have too strong an impact on the CMB spectrum to satisfactorily solve the Hubble tension [35]. The motivation for the decaying models, then, is that they may be able to circumvent these constraints by virtue of injecting the radiation energy density more locally around recombination.

————— End of excerpt of reference [5] —————

The following paper on decaying cold dark matter includes an analysis using the newest Planck CMB data in order to update the constraints on

¹ However, in the bottom panel, where Γ is such that both models decay after recombination, only a comparatively negligible increase in H_0 is seen.

the model parameters. It also includes an analytic treatment resulting in a relation between very rapidly decaying cold dark matter and a model with additional relativistic degrees of freedom. It furthermore presents how well the model is able to alleviate the Hubble tension and the more mild S_8 tension.

 Beginning of reference [1]

UPDATED CONSTRAINTS ON DECAYING COLD DARK MATTER

Andreas Nygaard^a, Thomas Tram^a, Steen Hannestad^a

^a*Department of Physics and Astronomy, Aarhus University, DK-8000 Aarhus C, Denmark*

Abstract. In this paper we update the constraints on the simple decaying cold dark matter (DCDM) model with dark radiation (DR) as decay product. We consider two different regimes of the lifetime, i.e. short-lived and long-lived, and use the most recent CMB data from Planck (2018) to infer new constraints on the decay parameters with which we compare the constraints inferred by the previous Planck data (2015). We hereby show that the newest CMB data constrains the fractional amount of DCDM twice as much as the previous data in the long-lived regime, leading to our current best 2σ upper bound of $f_{\text{dcdm}} < 2.44\%$. In the short-lived regime, we get a slightly looser 2σ upper bound of $f_{\text{dcdm}} < 13.1\%$ compared to the previous CMB data. If we include Baryonic Acoustic Oscillations data from BOSS DR-12, the constraints in both the long-lived and the short-lived regimes relax to $f_{\text{dcdm}} < 2.62\%$ and $f_{\text{dcdm}} < 1.49\%$, respectively. We also investigate how this model impacts the Hubble and σ_8 tensions, and we find that each of the decay regimes can slightly relieve a different one of the tensions. The model can thus not accommodate both tensions at once, and the improvements on each are not significant. We furthermore improve on previous work by thoroughly analysing the impacts of short-lived DCDM on the radiation density and deriving a mapping between short-lived DCDM and a correction, ΔN_{eff} , to the effective number of massless neutrino species.

3.1 INTRODUCTION

Ever since the first inference of the existence of dark matter almost a century ago, the nature of it has remained elusive. Significant progress has been made in establishing its properties. For example it is known that dark matter can only interact weakly with standard model particles (perhaps only through gravitation), and that it must be *cold* in the sense of having sufficiently small thermal velocity so that small scale structure can form. The standard Λ CDM model of cosmology is based on dark matter with exactly these properties and in general provides an excellent fit to observational data.

However, despite many advances in both theory and observations much still remains unknown about the nature of dark matter. For example dark matter could have significant interactions, either with itself or with other particles in a dark sector, separate from the standard model. Such models have been invoked as possible explanations of a variety of anomalies in cosmology, such as the missing satellites problem, the cusp-core problem, and the Hubble tension.

In this paper we investigate the possibility that dark matter is cold and consists of massive particles, but that these particles are unstable and decay. The decay product cannot simply be electromagnetic radiation, since that would be detectable, so instead it is assumed that the decaying cold dark matter (DCDM) decays into massless particles in a dark sector. In line with previous work on the topic we shall refer to the decay product as *dark radiation* (DR), so that we can sketch the process as

$$X_{\text{cdm}} \rightarrow \gamma_{\text{dr}}.$$

When comparing a model where all dark matter is DCDM with our observational data (Planck, etc.), we need a very large lifetime which renders our DCDM basically stable. Because of that, we add another degree of freedom to our model which is the initial fraction of DCDM to the total amount of cold dark matter (CDM),

$$f_{\text{dcdm}} = \frac{\Omega_{\text{dcdm}}^{\text{ini}}}{\Omega_{\text{dcdm}}^{\text{ini}} + \Omega_{\text{scdm}}}, \quad (3.1)$$

where SCDM is the stable CDM component. The physical interpretation of having this fractional decay could be [36]:

- 1) CDM is multi-component such that only a part of the dark matter decays,
- 2) all dark matter decays, but the decay product is both dark radiation and a stable CDM component with fraction $(1 - f_{\text{dcdm}})$.

Since the amount of dark matter decreases, we define $\Omega_{\text{dcdm}}^{\text{ini}}$ as the density parameter of DCDM today as if none of it had decayed. We otherwise adopt the same notation as in Ref. [30], which is now standard.

3.1.1 PREVIOUS WORK AND CONSTRAINTS

The field of decaying cold dark matter has been growing ever since the first analysis by Ref. [37] in 2004, where the simple model with dark matter decaying into dark radiation was also used. A lot has happened in the last two decades with the field being much wider now with numerous models of varying sophistication. The different regimes of the decay rate, Γ_{dcdm} , is investigated in Ref. [36] and the fractional amount of DCDM is constrained to $f_{\text{dcdm}} < 3.8 \times 10^{-2}$ using Cosmic Microwave Background (CMB) data from Planck-2015. This is in agreement with the results found in Ref. [38] where the same CMB data was used. In this paper we update the results and constraints from Ref. [36] using the newest Planck-2018 data as well as Baryonic Acoustic Oscillations (BAO) data.

It is well known that allowing dark matter to decay can potentially relieve the Hubble tension as proposed in Ref. [39], and this has been demonstrated numerous times in the literature. This includes Ref. [40] where both the Hubble tension and the milder S_8 tension of matter fluctuations between Λ CDM-model and the model-independent measurements in the local universe were relieved using CMB data from Planck-2018 and the same DCDM model as we are using in this paper. Including other data sets such as BAO data and intermediate-redshift data, however, leads to only a slight decrease in both tensions. Another attempt at relieving the Hubble tension is made in Ref. [41] where the tension is reduced by 1.5σ using Planck-2015 data, BAO data, Redshift Space Distortion (RSD) measurements and a DCDM model with decay time after recombination. By allowing the CMB lensing power amplitude to be a free fitting parameter, the tension is reduced by 3.3σ . They furthermore found an upper limit on the fractional amount of DCDM at the level $f_{\text{dcdm}} \lesssim 5\%$ for DCDM decaying after recombination. The

fractional amount of short-lived DCDM is also constrained in Ref. [42] where CMB data from Planck-2015, BAO data, and RSD measurements are used to infer the upper limit $f_{\text{dcdm}} \lesssim 2.73\%$, but this only leads to a slight reduction of the Hubble tension. They furthermore show the impacts of DCDM on the kinetic Sunyaev-Zel'dovich effect and note that this could lead to further constraints on DCDM in the future.

Better and more data has allowed the constraints to be more refined, with different studies using different combinations of data sets. An upper bound on the decay rate of late time DCDM at $\Gamma_{\text{dcdm}} \leq (175 \text{ Gyr})^{-1}$ is inferred in Ref. [43] by using CMB data and cluster counts from Planck-2015 along with weak lensing data from KiDS450, while a more tight constraint of $\Gamma_{\text{eff}} < 9.1 \times 10^{-9} \text{ Gyr}^{-1}$ is found in Ref. [44] to the effective decay rate by using Planck-2015 data and analysing cosmic reionisation and dark matter decay simultaneously. A different approach is taken in Ref. [45] where they assume that the decay product is detectable, i.e. a pair production of an elementary particle from the Standard Model (quarks, leptons or bosons), which leads to lower limits on the lifetime in the range $\tau \sim (1 - 5) \times 10^{28} \text{ s}$ using data from the isotropic gamma-ray background.

There are many studies beyond the simple DCDM model, where the decay product is only DR, and a more agnostic approach is found in Ref. [46] where the conversion of dark matter to dark radiation is treated in a more general aspect without assuming the transition being caused by a decay. They then find the impacts of the conversion on the CMB spectrum as well as constraints on their general model parameters using CAMB and COSMOMC, which lead to a reduction in the Hubble tension. They then treat a conversion model in detail where dark matter particles interact via a light mediator particle leading to Sommerfeld-enhanced self-annihilation.

Another, more general analysis is done in Ref. [47] where they investigate Dynamical Dark Matter in which the dark sector is comprised of a large ensemble of particles with different decay widths. Their analysis shows that the constraints allow energy scales ranging from GeV scale to the Planck scale, but the cosmological abundance of the dark sector today must be spread across an increasing number of states in the ensemble as the energy scale is decreased down to GeV scale from the Planck scale. A third continuation of the DCDM model is found in Ref. [48] which features Partially Acoustic Dark Matter, where a subdominant part of the dark matter is strongly coupled to the DR

fluid, and this combined fluid undergoes acoustic oscillations below the effective dark sound horizon. This is used to reduce the tensions in the Hubble constant and the lensing amplitude between CMB data and direct measurements. They find that even though the model can be used to reduce the tensions separately, it cannot accommodate both at once, since additional CDM is required by the CMB data to preserve the shape of the acoustic peaks.

It has recently been increasingly popular to assume a slightly warm component of the decaying dark matter sector, and an analysis of the background equations can be found in Ref. [49], where the cold dark matter decays to both DR and a warm daughter particle. They show that this can potentially relieve the Hubble tension, which makes the model very interesting to future work in the field of cosmology. More thorough analyses of the two-body decay model are found in Refs. [50–52], and the same model has also been shown to potentially relieve the S_8 tension in various studies, including Refs. [53–55]. This is due to the recoil velocity of the massive daughter particle inducing a free-streaming suppression of matter fluctuations. The model has also shown promise when it comes to the problems of small-scale structure, and treatments of this using N-body simulations can be found in Refs. [56–59]. More general studies, where both daughter particles can have arbitrary masses, are treated in Refs. [60, 61]. A slightly different model, where the decaying component is warm and the decay product is only DR, is treated thoroughly in Ref. [62]. The idea of this model is to not modify the evolution of the gravitational potentials, which otherwise leads to inconsistencies with data as in the case of decaying cold dark matter. They find that this can significantly reduce the Hubble tension as well.

3.1.2 OUTLINE OF THIS PAPER

In this paper, we are updating the constraints on the DCDM model parameters from Ref. [36] using the Markov Chain Monte Carlo (MCMC) sampler MONTEPYTHON [63] and the Einstein–Boltzmann code CLASS [27]. To describe the cosmological framework we will use the following set of parameters:

$$\Theta = \{\Omega_b h^2, \Omega_{\text{cdm}} h^2, h^2, A_s, n_s, \tau_{\text{reio}}\}, \quad (3.2)$$

in addition to the decay parameter vector $\{f_{\text{dcdm}}, \Gamma_{\text{dcdm}}\}$. As was also done in Ref. [36], we will split our MCMC runs into two categories, named *short-lived* and *long-lived*. The reason for this is that the likelihood contour in the full parameter space has a shape which makes convergence exceedingly slow. When the lifetime becomes very short, essentially all DCDM has decayed well before matter-radiation equality. This makes it impossible to constrain Γ_{dcdm} , and the only constrainable quantities are then f_{dcdm} and the product Γf_{dcdm} . This leads to a funnel-like likelihood surface stretching towards very large values of Γ_{dcdm} . This particular part of the parameter space is best probed using a logarithmic prior on Γ_{dcdm} , whereas for the long-lived regime, where only a fraction of the dark matter has decayed before the present, a prior which is flat in Γ_{dcdm} is more suitable. We will elaborate on the technicalities of this split in section 3.4.

We will furthermore look at an analogous scheme to the short-lived DCDM, i.e. a model with an increase in N_{eff} instead of a decaying dark matter component. This is treated both analytically and numerically using the CLASS code.

The structure of this paper is as follows. We introduce the formal theoretical framework of DCDM in the synchronous gauge in section 3.2, where both the background equations and the perturbation equations are presented. In section 3.3 we will treat the mapping between a correction to N_{eff} and short-lived DCDM by deriving an analytical expression for ΔN_{eff} corresponding to a short-lived DCDM component and comparing this to numerical results from CLASS. In section 3.4 we will present our results from the MCMC sampler MONTEPYTHON along with improved constraints in the long-lived and short-lived regimes, and in section 3.5 we will investigate the impact of our model on the Hubble and σ_8 tensions. Lastly, we will conclude and summarise in section 3.6.

3.1.3 COSMOLOGICAL DATA

As our data sets we use the newest CMB data from Planck-2018 [21] which has a higher quality in the polarisation data than its predecessor from 2015 [64]. In all of our computations using Planck-2018, we use both polarisation and temperature likelihoods for both high- ℓ and low- ℓ as well as the lensing likelihood. When comparing to Planck-2015 data, we of course use the corresponding likelihoods from this data release, which is the same combination used in Ref. [36].

We furthermore use the BAO data from BOSS data release 12 (DR-12) [65]. While the BAO data presumably has little impact in parameter constraints in the short-lived regime, due to the decay happening much earlier than the formation of the BAO, it is quite important in the limit of very long-lived DCDM.

3.2 BOLTZMANN EQUATIONS FOR DCDM AND DR

The behaviour of DCDM and DR can be calculated using the Boltzmann equation, which takes the following form for DCDM [36]:

$$\frac{df}{d\tau} = \frac{\partial f}{\partial \tau} + \frac{\partial f}{\partial x^i} \frac{dx^i}{d\tau} + \frac{\partial f}{\partial p} \frac{dp}{d\tau} + \frac{\partial f}{\partial \hat{p}^i} \frac{d\hat{p}^i}{d\tau} = \pm a\Gamma_{\text{dcdm}} f_{\text{dcdm}} , \quad (3.3)$$

with $-$ and $+$ for DCDM and DR respectively.

3.2.1 BACKGROUND EQUATIONS

The zeroth moment of the Boltzmann equation, eq. (3.3), i.e. integrating it over phase-space and keeping only terms of zeroth order, leads to the continuity equation, which is different for DCDM and DR than for the homogeneous universe in having source terms dependent of the decay rate Γ_{dcdm} with respect to proper time [36]:

$$\begin{aligned} \rho'_{\text{dcdm}} &= -3\frac{a'}{a}\rho_{\text{dcdm}} - a\Gamma_{\text{dcdm}}\rho_{\text{dcdm}} , \\ \rho'_{\text{dr}} &= -4\frac{a'}{a}\rho_{\text{dr}} + a\Gamma_{\text{dcdm}}\rho_{\text{dcdm}} , \end{aligned} \quad (3.4)$$

where the prime denotes derivatives with respect to conformal time, τ .

3.2.2 PERTURBATION EQUATIONS

One way of obtaining the relevant perturbation equations is again through the Boltzmann equation, eq. (3.3). We are doing our calculations in the comoving synchronous gauge, so we need to express the perturbation equations in this gauge. Perturbations in the synchronous

gauge can be expressed in the following way using the space-time interval [30]:

$$ds^2 = a^2(\tau) (-d\tau^2 + [\delta_{ij} + h_{ij}(\mathbf{x}, \tau)] dx^i dx^j) , \quad (3.5)$$

where the scalar part of the perturbation $h_{ij}(\mathbf{x}, \tau)$ can be expressed through its Fourier transform

$$h_{ij}(\mathbf{x}, \tau) = \int d^3k e^{i\mathbf{k}\cdot\mathbf{x}} \left(h(\mathbf{k}, \tau) \hat{k}_i \hat{k}_j + 6\eta(\mathbf{k}, \tau) \left[\hat{k}_i \hat{k}_j - \frac{1}{3} \delta_{ij} \right] \right) , \quad (3.6)$$

with $h(\mathbf{k}, \tau)$ and $\eta(\mathbf{k}, \tau)$ being the metric perturbations in the synchronous gauge. Integrating the Boltzmann equation, eq. (3.3), over phase-space and keeping the first order terms leads to an expression for the evolution of the density perturbation, $\delta_{\text{dcdm}} = \rho_{\text{dcdm}} / \bar{\rho}_{\text{dcdm}} - 1$, where the bar denotes the average value as in a homogeneous universe. We can furthermore find the evolution of the divergence, θ_{dcdm} , of the fluid velocity by taking the divergence of the first moment of eq. (3.3), which is found by multiplying the equation by \vec{p}/E (with \vec{p} and E being the 3-momentum and the energy of a particle, respectively) and again integrating over phase-space. The resulting equations are

$$\delta'_{\text{dcdm}} = -\frac{h'}{2} , \quad (3.7)$$

$$\theta'_{\text{dcdm}} = -\mathcal{H}\theta_{\text{dcdm}} = 0 . \quad (3.8)$$

These two equations are enough to describe the evolution of DCDM since it per definition is cold, which means that we have neglected all second (or higher) order terms of the momentum in our calculations [25]. All higher moments would therefore be zero.

The equations turn out a bit more complicated for DR, since this species is not cold and we therefore cannot neglect higher orders of momentum. Following the same procedure as for DCDM in the synchronous gauge, we get the following equations for the evolution of the density perturbations and the velocity divergence:

$$\delta'_{\text{dr}} = -\frac{2}{3}h' + a\Gamma_{\text{dcdm}} \frac{\rho_{\text{dcdm}}}{\rho_{\text{dr}}} (\delta_{\text{dcdm}} - \delta_{\text{dr}}) , \quad (3.9)$$

$$\theta'_{\text{dr}} = \frac{k^2}{4}\delta_{\text{dr}} - k^2\sigma_{\text{dr}} - a\Gamma_{\text{dcdm}} \frac{\rho_{\text{dcdm}}}{\rho_{\text{dr}}} \theta_{\text{dr}} . \quad (3.10)$$

These certainly do not look as simple as those for Λ CDM, and we notice the parameter σ_{dr} in the bottom equation which is the next moment of the Boltzmann equation - the shear stress. Generally the evolution of a moment, l , will depend on the next moment, $l + 1$, which leads to an infinite Boltzmann hierarchy for the moments containing the same information as the momentum-dependent Boltzmann equation itself. We still need to truncate the hierarchy at some large l -value to work with it numerically. In the `CLASS` code, the cut-off l -value can be user-specified and is $l_{\text{max}} = 17$ by default.

3.3 MAPPING SHORT-LIVED DCDM TO N_{eff}

In the very short-lived regime, all DCDM has decayed well before matter-radiation equality. In this regime the primary effect of DCDM should be to enhance N_{eff} through the decay product, DR. We define the correction, ΔN_{eff} , as the ratio between the energy densities of the additional radiation (which is DR) and a single massless neutrino. However, we need to evaluate this ratio after the DCDM has fully decayed, where no more DR is produced and the energy density scales as a^{-4} (like any type of radiation),

$$\Delta N_{\text{eff}} = \left. \frac{\rho_{\text{dr}}}{\rho_{\nu}^{N=1}} \right|_{t \gg t_d}, \quad (3.11)$$

where $\rho_{\nu}^{N=1} = 7/8 (4/11)^{4/3} \rho_{\gamma}$ represents only a single massless neutrino [66], and t_d is the time of the decay.

We can estimate how ΔN_{eff} should scale by assuming an instant decay, so the energy density of DR just after the decay equals that of DCDM just before the decay. We can thus evaluate the energy density of DCDM at the time of decay instead of that of DR. We can then scale that to the current time, introducing a_d as the scale factor at the time of decay

$$\Delta N_{\text{eff}} \approx \left. \frac{\rho_{\text{dcdm}}}{\rho_{\nu}^{N=1}} \right|_{t=t_d} \sim a_d \cdot \frac{f_{\text{dcdm}}}{1 - f_{\text{dcdm}}} \cdot \frac{\Omega_{\text{dm},0}}{\Omega_{\nu,0}^{N=1}}. \quad (3.12)$$

Of course this is a very simplistic calculation, but it shows that, as $a_d \rightarrow 0$ (and thus $\Gamma_{\text{dcdm}} \rightarrow \infty$) the effect of DCDM vanishes and the model becomes identical to standard Λ CDM.

3.3.1 ANALYTIC SOLUTION OF BOLTZMANN EQUATIONS

We can derive an analytic approximation to the Boltzmann equations fairly easily. First, we define

$$Y = a^3 \rho_{\text{dcdm}} / \rho_{\nu,0}, \quad X = a^4 \rho_{\text{dr}} / \rho_{\nu,0}, \quad (3.13)$$

i.e. X and Y are constant in the absence of decays. Here $\rho_{\nu,0}$ is the energy density of all neutrinos today as described by the effective number N_{eff} . The Boltzmann equations (3.4), can then be expressed in proper time as

$$\frac{dY}{dt} = -\Gamma Y(t), \quad \frac{dX}{dt} = a\Gamma Y(t), \quad (3.14)$$

where we have dropped the subscript on the decay rate Γ . The Boltzmann equation for the decaying component then has the solution

$$Y(t) = Y_i e^{-\Gamma t}, \quad (3.15)$$

$$Y_i = a_i^3 \frac{(\rho_{\text{dcdm}})_i}{\rho_{\nu,0}} = \frac{\Omega_{\text{dcdm}}^{\text{ini}}}{\Omega_{\nu,0}} = \frac{\Omega_{\text{dm},0}}{\Omega_{\nu,0}} \cdot \frac{f_{\text{dcdm}}}{1 - f_{\text{dcdm}}}, \quad (3.16)$$

where the subscript i represents some initial starting point before the decay, and the last equal sign assumes that all DCDM has decayed today, thus making $\Omega_{\text{dm},0}$ both the current density parameter of all dark matter and that of only stable dark matter, since no decaying component is left.

The Boltzmann equation for DR can now be solved using the solution to that of DCDM. We switch coordinates to the scale factor divided by the initial scale factor at the starting point, $\tilde{a} \equiv a/a_i$, and assume that the universe is radiation dominated throughout the decay so that $a \propto t^{1/2}$. We furthermore define

$$\alpha \equiv \frac{H_i}{\Gamma} = \left(\frac{8\pi G}{3} \rho_{\text{r},0} a_i^{-4} \right)^{1/2} \cdot \frac{1}{\Gamma}, \quad (3.17)$$

$\Omega_b h^2$	$\Omega_{\text{cdm}} h^2$	h^2	$A_s \times 10^9$	n_s	τ_{reio}	N_{eff}
0.022032	0.11933	0.67556	2.215	0.9619	0.0925	3.046

Table 3.1: Table of input parameters in CLASS which are held constant through all calculations in section 3.3.2. N_{eff} is, however, modified with ΔN_{eff} in figure 3.3.

where $\alpha \gg 1$. We then integrate the differential equation from the starting point $\tilde{a} = 1$ to find the solution for X (a similar result can be found in Ref. [67])

$$X(\tilde{a}) = X_i + \left[\sqrt{\frac{\pi}{2}} \text{erf} \left(\frac{\tilde{a}}{\sqrt{2\alpha}} \right) - \frac{\tilde{a} e^{-\tilde{a}^2/(2\alpha)}}{\sqrt{\alpha}} \right] \times \left(\frac{8\pi G}{3} \rho_{\text{r},0} \right)^{1/4} \frac{\Omega_{\text{dm},0}}{\Omega_{\nu,0}} \cdot \frac{f_{\text{dcdm}}}{1 - f_{\text{dcdm}}} \Gamma^{-1/2}, \quad (3.18)$$

where X_i is the initial value of $X(\tilde{a})$ proportional to the initial DR component (here we have $X_i = 0$ since there is no pre-existing DR component). In the asymptotic limit ($\tilde{a} \rightarrow \infty$) we find

$$X(\tilde{a} \rightarrow \infty) = X_i + \sqrt{\frac{\pi}{2}} \left(\frac{8\pi G \rho_{\text{r},0}}{3} \right)^{1/4} \frac{\Omega_{\text{dm},0}}{\Omega_{\nu,0}} \cdot \frac{f_{\text{dcdm}}}{1 - f_{\text{dcdm}}} \Gamma^{-1/2}, \quad (3.19)$$

and we finally note that $\Delta N_{\text{eff}} = N_{\text{eff}} X(\tilde{a} \rightarrow \infty)$ because of our definition of X . Here we notice the scaling relation $\Delta N_{\text{eff}} \propto \Gamma^{-1/2} f_{\text{dcdm}} / (1 - f_{\text{dcdm}})$. Inserting numbers in eq. (3.19), we arrive at an approximate relation of the form

$$\Delta N_{\text{eff}} \sim 5.74 \Omega_{\text{dm},0} h^2 \frac{f_{\text{dcdm}}}{1 - f_{\text{dcdm}}} \Gamma_6^{-1/2}, \quad (3.20)$$

where $\Gamma_6 = \Gamma / (10^6 \text{ Gyr}^{-1})$.

3.3.2 NUMERICAL ANALYSIS USING CLASS

Using the CLASS code, we can get an exact numerical correction to N_{eff} caused by short-lived DCDM. The values of input parameters which are held constant in all of the calculations can be seen in table 3.1 (N_{eff} is modified by ΔN_{eff} in figure 3.3).

Running CLASS with a high $\Gamma_{\text{dcdm}} (> 10^3 \text{ Gyr}^{-1})$ ensures that all DCDM has decayed well before today, which means that the energy density of the resulting DR scales like normal radiation. DR and neutrinos therefore scale similarly, which makes the ratio of their energy densities constant after the decay. This ratio is used to define the correction ΔN_{eff} cf. eq. (3.11). However, because we do not only have a single neutrino, we need to divide the neutrino density with N_{eff} . The numerical value of ΔN_{eff} can in fact be evaluated at any time after the decay as long as the energy density of DCDM is no longer of any significant size. We, however, evaluate at the current time in order to get a density of DCDM as low as possible,

$$\Delta N_{\text{eff}}^{(\text{numerical})} = N_{\text{eff}} \left. \frac{\rho_{\text{dr}}}{\rho_{\nu}} \right|_{t=t_0}, \quad (3.21)$$

where ρ_{ν} is the total energy density of all massless neutrino species.

At first, we are interested in how well our analytical approach fits the numerical results. Using eq. (3.21), we calculate ΔN_{eff} for different decay rates, Γ_{dcdm} , and fractional amounts of DCDM, f_{dcdm} , and plot the results as functions hereof. This can be seen in the figures 3.2a and 3.2b respectively along with the analytical results and the ratio between the two. The numerical and analytical results agree with increasing precision for higher values of Γ_{dcdm} , which is also supported by their ratios approaching unity. Here we also note that both results approach zero for increasing decay rate, which supports that we should recover the Λ CDM model for $\Gamma_{\text{dcdm}} \rightarrow \infty$ as we argued in the beginning of section 3.3. From figure 3.2b we, however, see that the analytical and numerical results agree with decreasing precision for higher values of f_{dcdm} , which makes sense according to eq. (3.19) due to the divergence at $f_{\text{dcdm}} = 1$. It should of course behave this way because we cannot have that short-lived DCDM comprises all of the dark matter, since that would correspond to no dark matter at all after recombination, which contradicts some assumptions made in the derivation in section 3.3.1. Note that the value of ΔN_{eff} also approaches zero for $f_{\text{dcdm}} \rightarrow 0$ which again leads to the Λ CDM model. We also note that the ratio between the analytical and numerical results in figure 3.2b does not approach unity exactly, but rather a slightly higher value, since the decay rate is fixed at a finite value ($\Gamma_{\text{dcdm}} = 10^7 \text{ Gyr}^{-1}$) and the ratio only approaches unity for $\Gamma_{\text{dcdm}} \rightarrow \infty$, according to figure 3.2a.

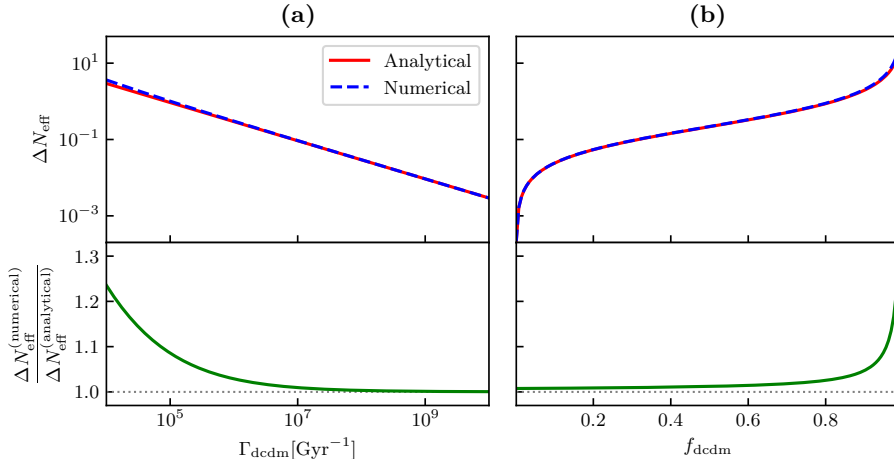


Figure 3.2: ΔN_{eff} as a function of different parameters for both the numerical result calculated using eq. (3.21) and the analytical result calculated using eq. (3.19). The lower panels show the ratio between the numerical and analytical ΔN_{eff} as a function of the same parameters. The additional parameters used for the simulations are found in table 3.1. (a) ΔN_{eff} as a function of the decay rate, Γ_{dcdm} , with the fractional amount of DCDM fixed to $f_{\text{dcdm}} = 0.3$. (b) ΔN_{eff} as a function of the fractional amount of DCDM, f_{dcdm} , with the decay rate fixed to $\Gamma_{\text{dcdm}} = 10^7 [\text{Gyr}^{-1}]$.

We can now run CLASS again without DCDM, but with the corresponding $\Delta N_{\text{eff}}^{(\text{numerical})}$ from the DCDM computation instead. According to the theory, we would expect the cosmology of such a run to be similar to that of very short-lived DCDM. Figure 3.3 shows the CMB TT power spectra of the simulations with DCDM and $\Delta N_{\text{eff}} = \Delta N_{\text{eff}}^{(\text{numerical})}$ for two values of the decay rate, $\Gamma_{\text{dcdm}} \in \{10^5 \text{ Gyr}^{-1}, 10^8 \text{ Gyr}^{-1}\}$, and a fraction of initial DCDM to all dark matter of $f_{\text{dcdm}} = 0.2$. From this figure it is very clear that the mapping to ΔN_{eff} becomes more accurate for higher values of Γ_{dcdm} , as predicted by the theory. For the lower value of $\Gamma_{\text{dcdm}} = 10^5 \text{ Gyr}^{-1}$, we still see the same behaviour in the power spectra, but they do not overlap as well due to the decay happening much closer to recombination, so a more significant amount of DR production is still ongoing at the time of the primary CMB signal formation. We also note that the magnitude of the relative power spectra is decreasing for larger Γ_{dcdm} as we would also expect from the theory.

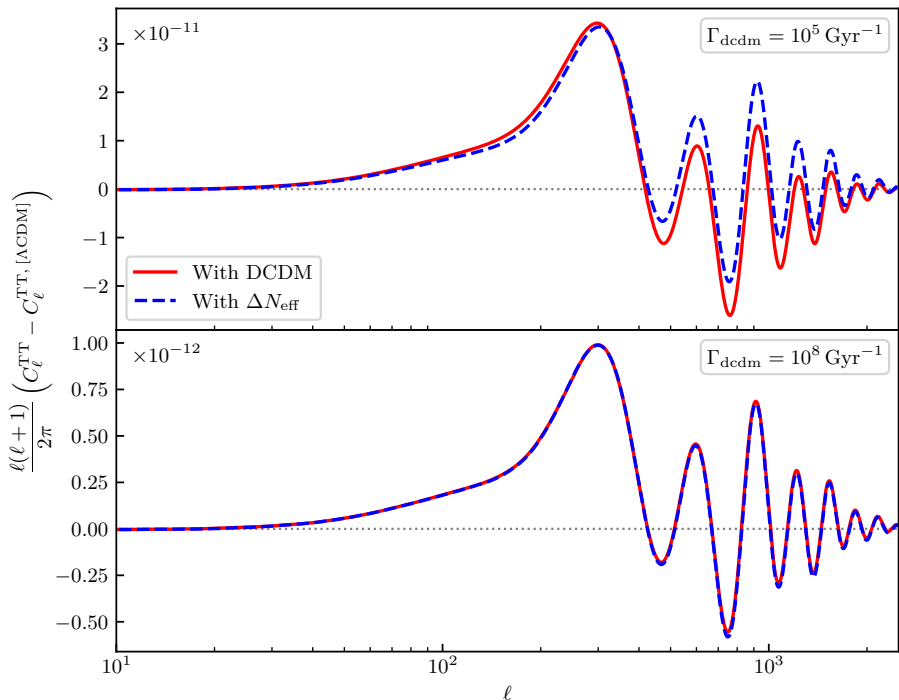


Figure 3.3: CMB TT power spectra for different values of Γ_{dcdm} and the corresponding $\Delta N_{\text{eff}}^{\text{numerical}}$ (calculated using eq. (3.21)) relative to the power spectrum of the ΛCDM model with no correction to N_{eff} . The additional parameters used for the simulations are found in table 3.1, and the fractional amount of DCDM is fixed to $f_{\text{dcdm}} = 0.2$.

3.4 CURRENT CONSTRAINTS ON DECAY PARAMETERS

In order to obtain constraints on decay parameters we have used the publicly available code MONTEPYTHON [63]. This Markov Chain Monte Carlo (MCMC) sampler uses the Metropolis-Hastings algorithm to sample the probability distribution assuming flat priors. The code is run with four or seven sample chains on a computer cluster until convergence of the chains. Our Gelman-Rubin criterion for convergence of the chains is $R - 1 \lesssim 0.01$, which means that the largest $R - 1$ value of any parameter should be around or smaller than 0.01. The prior of the Γ_{dcdm} parameter is set with different upper and lower bounds depending on which regime we want to analyse. These bounds and the number of sample chains will be stated when necessary.

3.4.1 LONG-LIVED DCDM REGIME

The long-lived regime with a decay rate in the interval $\Gamma_{\text{dcdm}} \in [0, 10^3] \text{ Gyr}^{-1}$ corresponds to anything in between the DCDM starting to decay around the time of recombination (and thus having fully decayed by now) and an infinite lifetime where none of the DCDM has decayed yet. The common thing here is that the decay has not finished before recombination which leads to an ongoing DR production after the emission of the CMB. The late decay will thus have an effect on e.g. the angular diameter distance and other background parameters.

Using MONTEPYTHON we search the parameter space with four sample chains using the likelihoods of Planck-2015 and Planck-2018 to see the differences of the two data sets. The result of this is found in figure 3.4. We have also tried to further constrain the parameters using a combination of Planck-2018 and Baryonic Acoustic Oscillation data (BAO) from BOSS DR12. These results are plotted along with that of only Planck-2018 in figure 3.5.

From figure 3.4 we see that the Planck-2018 data constrains the amount of DCDM through the parameter f_{dcdm} much more than the Planck-2015 data does, and our results using Planck-2015 is in great agreement with those found in Ref. [36]. The same is true regarding the constraints on the decay rate, Γ_{dcdm} , which means that the Planck-2018 data favours very long-lived DCDM even more than Planck-2015 data does. The plateau for higher values of Γ_{dcdm} is due to the fact that the data cannot distinguish between the models corresponding to this plateau given that they all imply a very late decay of DCDM (much later than recombination). The plateau continues all the way to the short-lived regime, but we only show the lowest values here. The disagreement in the parameter $\omega_{\text{cdm}}^{\text{ini}} \equiv (\Omega_{\text{dcdm}}^{\text{ini}} + \Omega_{\text{cdm}})h^2$ is mainly due to the disagreement in Ω_{cdm} of the two Planck data sets [21, 64], since the very long-lived DCDM behaves like a stable component and we thus would not expect a significant change to the total dark matter component from that of the Λ CDM model.

In figure 3.5 we see that the addition of BAO data relaxes the amount of DCDM through f_{dcdm} a little, and we see a slight change in the posterior of the decay rate as well, raising the plateau for higher values while faintly narrowing the peak towards lower values. The parameter $\omega_{\text{dcdm}}^{\text{ini}}$ is lowered by BAO as well, and this is in agreement with a narrower peak in the decay rate when including BAO, i.e. if the decay rate is smaller, the lifetime is longer, and we therefore need a smaller

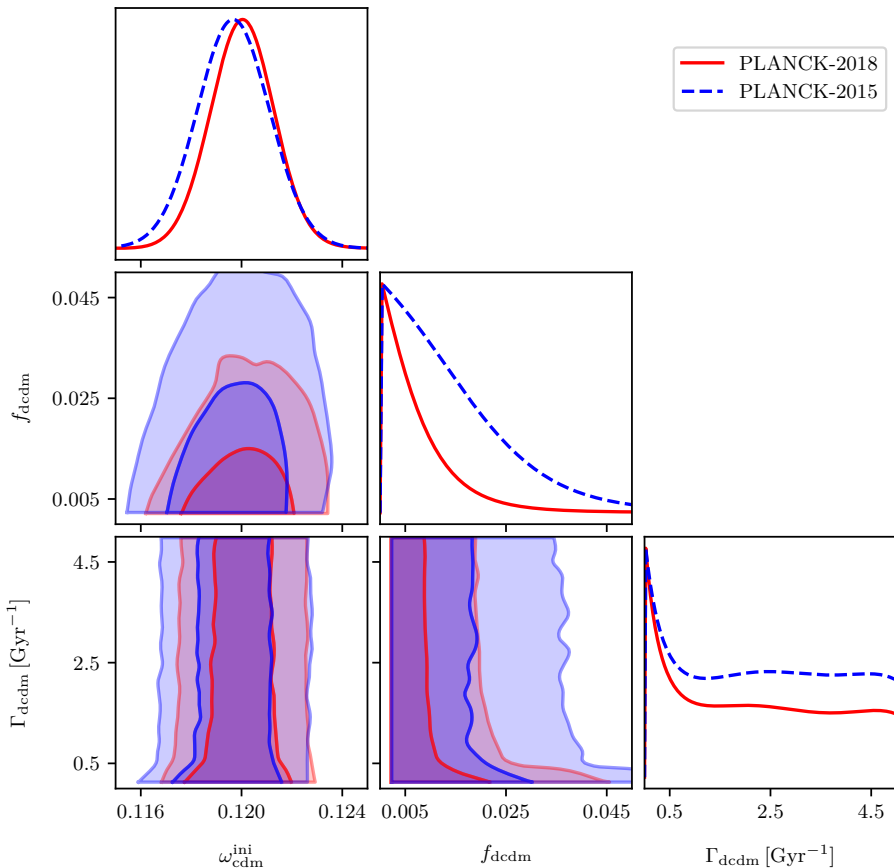


Figure 3.4: Triangle plot of posteriors in the long-lived regime using the two Planck data sets from 2015 and 2018.

amount of initial dark matter since less of it has time to decay before the present.

The relevant best-fit parameters and constraints in the long-lived regime can be seen in the middle panel of table 3.2 where we see a much tighter constraint on f_{ddm} inferred by data from Planck-2018 as opposed to data from Planck-2015, and the same is true for the parameter Γf_{ddm} . We also notice that the inclusion of BAO data loosens the constraints of the parameters while slightly modifying the best-fit values of H_0 and $\Omega_{\text{cdm}}^{\text{ini}}$ as well.

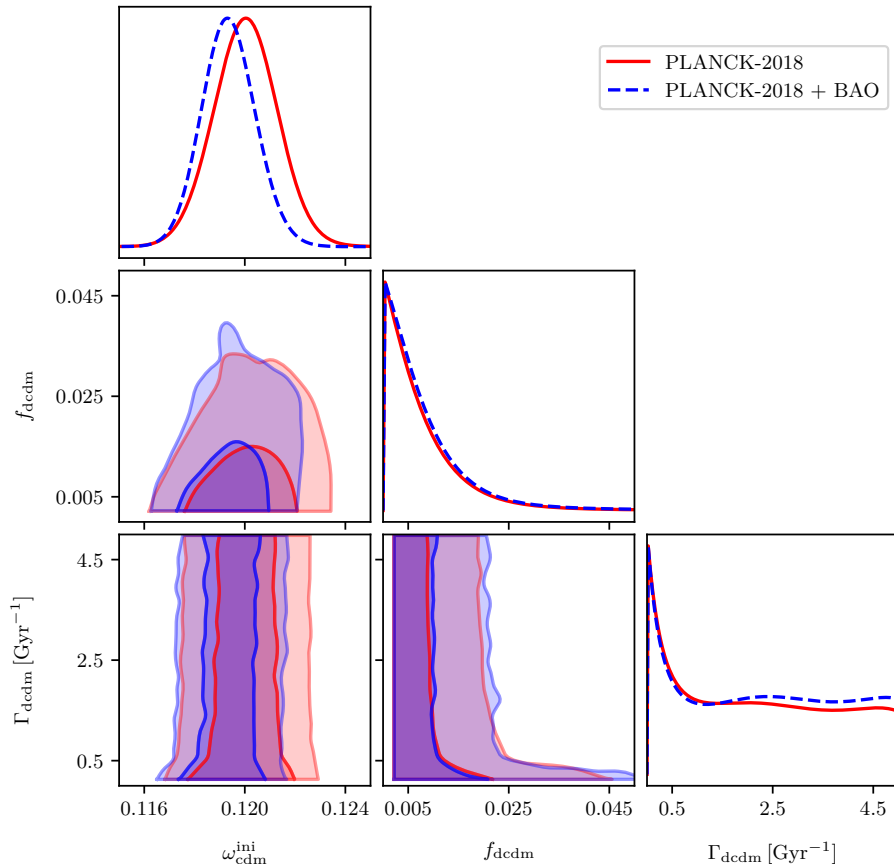


Figure 3.5: Triangle plot of posteriors in the long-lived regime using Planck-2018 data with and without BAO data from BOSS DR12.

VERY LONG-LIVED DCDM REGIME

In order to compare with the previous results of Ref. [36], we have also investigated a *very long-lived* sub-regime with $\Gamma_{\text{dcdm}} \lesssim H_0$. This yields slightly different results, since the long-lived runs are strongly undersampled in the very long-lived limit, but the tendencies are the same with tighter constraints from the newest Planck-2018 data. The inclusion of BAO data actually further tightens the constraint of the parameter Γf_{dcdm} even though this was opposite in the long-lived regime.

We find that this regime allows for a much larger fractional amount of DCDM ($f_{\text{dcdm}} \rightarrow 1$), which is clear from figure 3.6, since the lifetime is larger than the age of the universe and the fractional amount thus has only little impact on current observables and no impact on early-

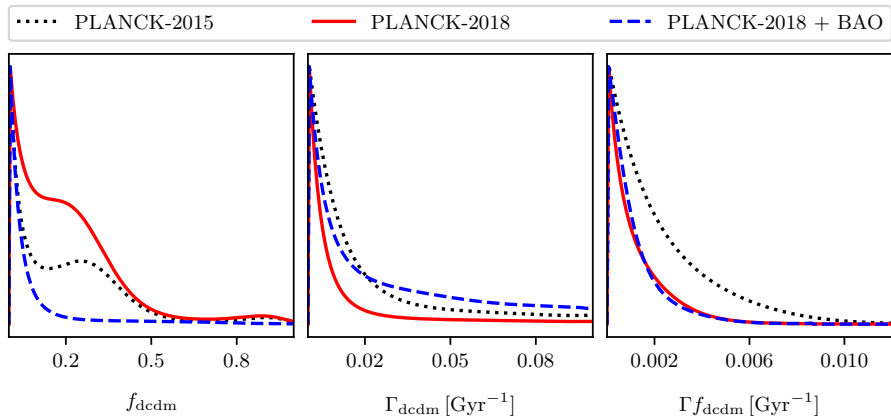


Figure 3.6: 1D posteriors of DCDM parameters in the very long-lived sub-regime using Planck-2015 data along with Planck-2018 data with and without BAO data.

time observables such as the CMB. Constraints from this regime can be found in the upper panel of table 3.2. We notice that we are only able to constrain the parameter f_{dcdm} when including the BAO data, and even then it is only possible to 1σ . This is due to the degenerate nature of the parameter in this regime, where all values are allowed by the data. The degeneracy is lifted when including BAO data since late-time measurements can rule out too high values of f_{dcdm} , given that these would feel the effects of the decay as opposed to the early-time measurements. This is also apparent from figure 3.6.

3.4.2 SHORT-LIVED DCDM REGIME

In the short-lived regime, the decay rate is rather high ($\Gamma_{\text{dcdm}} > 10^3 [\text{Gyr}^{-1}]$) resulting in most of the DCDM decaying well before recombination. We now search the parameter space using seven sample chains, and in order to search more efficiently, we implement the logarithm of the decay rate, $\log_{10}(\Gamma_{\text{dcdm}})$, as a parameter in CLASS, which allows us to use this parameter in MONTEPYTHON as well. The parameter space is effectively reduced in size which makes the sampling much faster. We cut the prior at the upper bound $\Gamma_{\text{dcdm}} < 10^6 [\text{Gyr}^{-1}]$ (similar to Ref. [36]) for convergence reasons, but we have checked that the posterior for the decay rate flattens out for increasing values, thus creating another plateau where the data cannot distinguish between the models. To further increase the efficiency we use the flag `-T=2.0` (de-

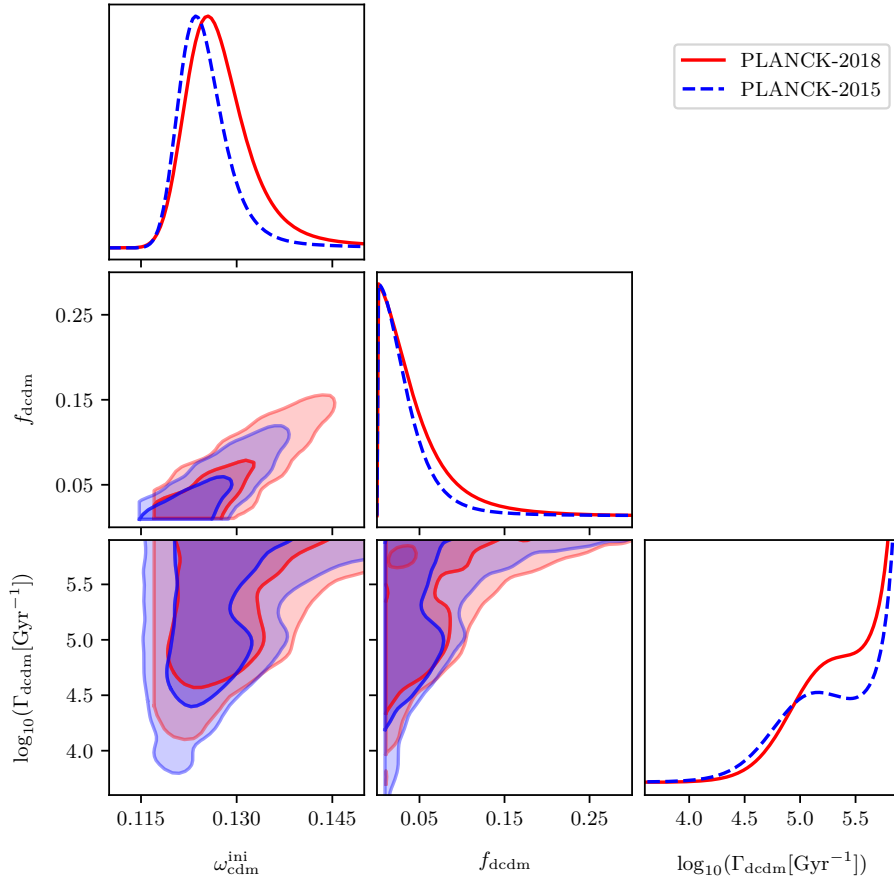


Figure 3.7: Triangle plot of posteriors in the short-lived regime using the two Planck data sets from 2015 and 2018.

fault 1.0) when running the code, which corresponds to increasing the statistical temperature of the chains so they are more likely to jump further in the parameter space.

The figures 3.7 and 3.8 show the posterior distributions in the short-lived regime of Planck-2018 data and either Planck-2015 data or including BAO data, respectively. The 1D-posteriors of the parameter $\log_{10}(\Gamma_{\text{dcdm}} [\text{Gyr}^{-1}])$ are not shown in their entirety, but rather a closeup of the interesting region is presented by only showing the vertical interval $[0, 0.2]$ (the posteriors are normalised so highest value equals unity).

Figure 3.7 shows that Planck-2018 data allows for a slightly higher initial dark matter density, $\omega_{\text{cdm}}^{\text{ini}}$, than Planck-2015 data does, which is apparent from the broader peak and longer exponential tail towards higher

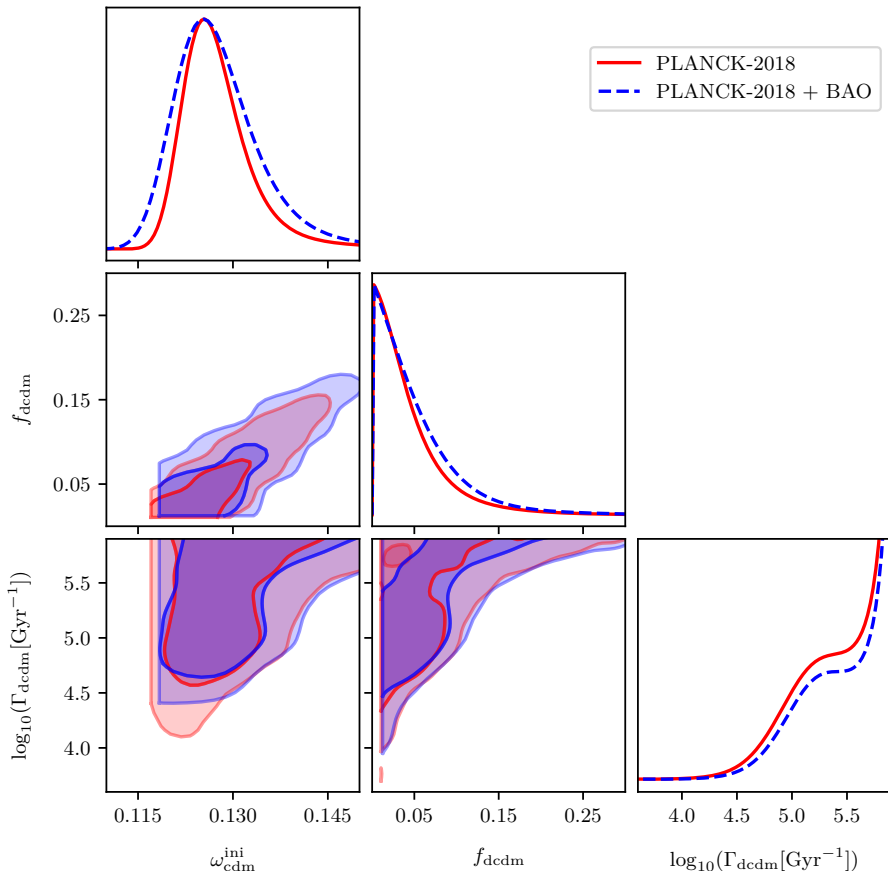


Figure 3.8: Triangle plot of posteriors in the short-lived regime using Planck-2018 data with and without BAO data from BOSS DR12.

values in the 1D-posterior. The amount of DCDM allowed by Planck-2018 is also slightly higher which is apparent from the 1D-posterior of f_{dcdm} being slightly broader than that of Planck-2015. The posterior of the decay rate consists of two sub-regimes as explained in Ref. [36], where the small peak (plateau) in the 1D-posterior of Planck-2015 (2018) represents a decay happening mostly in between matter-radiation equality and recombination while the increase in the posterior at larger values of the decay rate represents a decay taking place mostly before matter-radiation equality. Our 1D-posterior of the decay rate for the Planck-2015 data is reminiscent of the one presented in Ref. [36], but with a much less significant peak even though the same data has been used. The difference here could possibly be due to the Gelman-Rubin

VLL	$\Omega_{\text{cdm}}^{\text{ini}} h^2 \times 10^2$	H_0	$f_{\text{dcdm}} \times 10^2$	$\Gamma f_{\text{dcdm}} \times 10^3$
		$[\text{km s}^{-1} \text{Mpc}^{-1}]$		$[\text{Gyr}^{-1}]$
P15	$11.9^{+0.3}_{-0.3}$	$67.7^{+1.3}_{-1.2}$	—	< 7.12
P18	$11.9^{+0.2}_{-0.2}$	$67.5^{+1.2}_{-1.2}$	—	< 4.01
+ BAO	$11.9^{+0.2}_{-0.2}$	$67.6^{+0.9}_{-0.9}$	$< 8.05^*$	< 3.72
LL	$\Omega_{\text{cdm}}^{\text{ini}} h^2 \times 10^2$	H_0	$f_{\text{dcdm}} \times 10^2$	$\Gamma f_{\text{dcdm}} \times 10^2$
		$[\text{km s}^{-1} \text{Mpc}^{-1}]$		$[\text{Gyr}^{-1}]$
P15	$12.0^{+0.3}_{-0.3}$	$67.3^{+1.6}_{-1.5}$	< 4.14	< 10.99
P18	$12.1^{+0.3}_{-0.3}$	$67.1^{+1.2}_{-1.2}$	< 2.44	< 5.18
+ BAO	$11.9^{+0.2}_{-0.2}$	$67.7^{+1.0}_{-0.9}$	< 2.62	< 5.84
SL	$\Omega_{\text{cdm}}^{\text{ini}} h^2 \times 10^2$	H_0	$f_{\text{dcdm}} \times 10^1$	$\Gamma f_{\text{dcdm}} \times 10^{-4}$
		$[\text{km s}^{-1} \text{Mpc}^{-1}]$		$[\text{Gyr}^{-1}]$
P15	$12.6^{+1.5}_{-0.5}$	$67.7^{+1.6}_{-1.6}$	< 0.98	< 2.35
P18	$12.7^{+1.6}_{-0.8}$	$67.8^{+1.4}_{-1.5}$	< 1.31	< 3.01
+ BAO	$13.1^{+1.8}_{-1.0}$	$68.6^{+1.2}_{-1.4}$	< 1.49	< 3.78

Table 3.2: Table of parameter constraints in the long-lived (LL) and short-lived (SL) regimes as well as in the very long-lived (VLL) sub-regime. We present 2σ constraints corresponding to a 95% confidence level. The * refers to 1σ constraints only. Results are shown using data from Planck-2015 (P15), data from Planck-2018 (P18), and a combination of data from Planck-2018 and BAO.

convergence criterion of the MCMC sampling, where our criterion is an order of magnitude lower than the one used in Ref. [36]. The conclusion, however, does not change by letting the chains converge further, but we do get that the region between the small peak and the subsequent increase is populated more frequently with a stronger convergence criterion, thus diminishing the profoundness of the peak and making it more like a plateau. Using Planck-2018 data we completely transform the peak into a plateau instead.

From figure 3.8 it is clear that the inclusion of BAO data does not have a significant impact in the short-lived regime, as we previously

argued. We see almost the exact same features in the posteriors with the exception that the 1D-posteriors of the initial dark matter density and the fractional amount of DCDM seem broader when including BAO data as opposed to just using Planck-2018 data. This is, however, not significant and we should not draw any conclusions of it, since it could very well be due to our chains not being converged enough. Increasing the amount of sample points or chains could possibly make the difference between the posteriors vanish.

The lower panel of table 3.2 shows the relevant best-fit values and constraints in the short-lived regime. Here we see more loose constraints of the DCDM parameters from Planck-2018 data than from Planck-2015 data, and even more so when including BAO data. The DCDM parameter constraints are, however, not affected as much in the short-lived regime as in the very long-lived limit, where the inclusion of BAO data lifts the degeneracy of the f_{dcdm} parameter, and this agrees with the idea that BAO data only has little impact in short-lived regime.

We note that at first it might seem somewhat counter-intuitive that the constraint on f_{dcdm} is actually loosened by the inclusion of more data, rather than strengthened. The reason for this shift is, however, easy to understand from figure 3.8. Given the strong correlation between f_{dcdm} and Γ_{dcdm} , the shift towards higher values of Γ_{dcdm} enforced by the Planck-2018 (and BAO) data removes a large fraction of the low f_{dcdm} parameter space allowed by the Planck-2015 data. This automatically shifts the preferred range of f_{dcdm} upwards and leads to a less restrictive upper bound on this parameter.

3.5 IMPACT ON THE HUBBLE AND σ_8 TENSIONS

As mentioned in section 3.1.1, the idea of decaying dark matter has been able to relieve both the Hubble tension and the σ_8 tension in matter fluctuations to some extent.

From the 1D-posteriors of figure 3.9 we can clearly see that our model is indeed capable of relieving the Hubble tension to some extent using the various data sets. The best result in this regard is obtained using both Planck-2018 data and BAO data. This seems to be the case in both regimes while the short-lived regime is more successful in relieving the tension, which is also apparent from table 3.2. Compared to standard Λ CDM, our best case scenario is, however, only able to relieve the tension with $\sim 1\sigma$, so the simple DCDM model does not offer the solution to

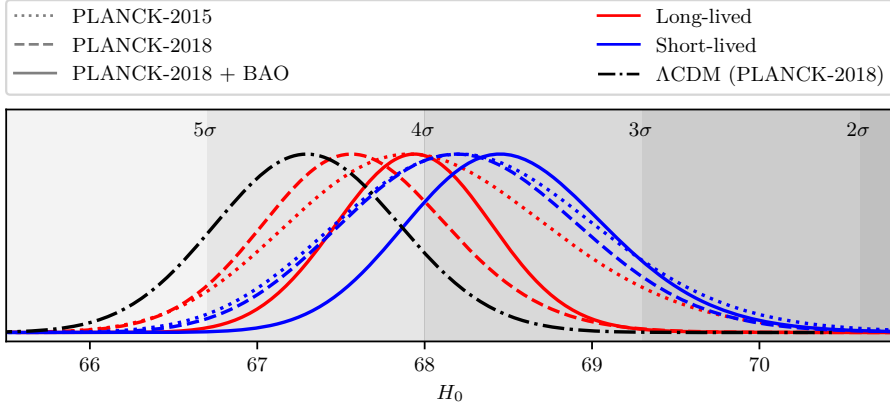


Figure 3.9: 1D-posteriors of H_0 using all the data set combinations in both regimes. A posterior for the Λ CDM model using the newest Planck-2018 data has been included for reference. The shaded areas represent the latest confidence level of measurements in the local universe of $73.2 \pm 1.3 \text{ km s}^{-1} \text{ Mpc}^{-1}$ from Ref. [68].

this discrepancy. Perhaps one might have expected that the short-lived case would offer a better fit with high values of H_0 . It is well known that using only CMB temperature data, there is a very strong positive correlation between N_{eff} and H_0 , and since the short-lived case can be almost exactly mapped to a model with increased N_{eff} (see section 3.3), it is perhaps natural to expect that a high H_0 can be accommodated. However, the high (H_0, N_{eff}) region is no longer allowed when polarisation data is added, thus disallowing this possibility, and in the end the preferred range of H_0 is only shifted marginally towards higher values in this case.

The σ_8 tension is also not solved with this model, which is apparent from the contours of figure 3.10. This figure includes the contours of the combined DES Year 1 data from local measurements of the matter fluctuations using galaxy clustering and lensing. We can see that the contours of the long-lived regime move towards that of DES, but the tension is only relieved slightly with the 2σ contours of the long-lived results being within the 1σ contour of DES and vice versa. In the short-lived regime, all contours remain outside the 1σ contour of DES, as in the case of standard Λ CDM, and they only move tangentially along the DES contour in comparison to the Λ CDM contour, thus yielding no better (nor worse) result.

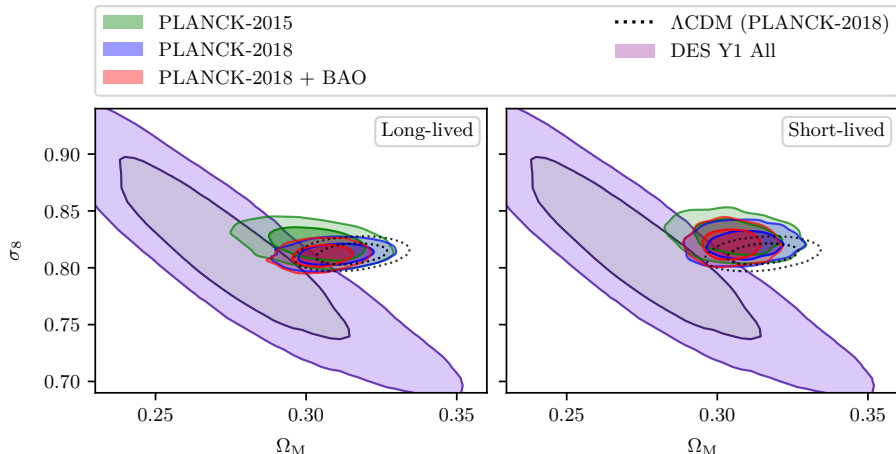


Figure 3.10: 2D-posteriors of Ω_M and σ_8 using all the data set combinations in both regimes. A posterior from the Λ CDM model using the newest Planck-2018 data has been included for reference (dotted contours) as well as a posterior from DES Year 1 (purple contour) combining galaxy-galaxy clustering, galaxy-galaxy lensing and cosmic shear (Ref. [69]).

Since the σ_8 tension can be relieved (slightly) only in the long-lived regime and since the Hubble tension can be relieved best in the short-lived, we recover a problem stated in Ref. [40] saying that attempts to relieve one of the tensions often worsens the other. We, however, do not get a worse result for the other when attempting to relieve one of the tension, but since the two tensions are relieved in different decay regimes, the simple DCDM model cannot be a solution to both tensions at once.

3.6 CONCLUSION

Using the MCMC sampler MONTEPYTHON and the most recent CMB data from Planck-2018, we have improved on the previous constraints on the fractional amount of DCDM, f_{dcdm} , as well as the parameter Γf_{dcdm} in the two regimes (long-lived and short-lived) along with the very long-lived sub-regime. In the very long-lived sub-regime, we find that the fractional amount of DCDM cannot be constrained by Planck-2018 data alone due to the degenerate nature of the parameter in this regime. We can however get a constraint for Γf_{dcdm} for which we find an upper bound at 2σ of $\Gamma f_{\text{dcdm}} < 4.01 \times 10^{-3} \text{ Gyr}^{-1}$. In the long-lived regime, we find that the fractional amount of DCDM is much better constrained

by Planck-2018 data (as opposed to Planck-2015 data) leading to an upper bound at 2σ of $f_{\text{dcdm}} < 2.44\%$, and the same is true for Γf_{dcdm} for which we find an upper bound at 2σ of $\Gamma f_{\text{dcdm}} < 5.18 \times 10^{-2} \text{ Gyr}^{-1}$. The constraints in the long-lived and very long-lived regimes are thus all tighter than the constraints inferred by Planck-2015 data by a factor of ~ 2 . For the short-lived regime, the Planck-2018 data leads to more loose constraints than Planck-2015 data does, and we thus find 2σ upper bounds of $f_{\text{dcdm}} < 13.1\%$ and $\Gamma f_{\text{dcdm}} < 3.01 \times 10^4 \text{ Gyr}^{-1}$, which are both higher than those of Planck-2015 by a factor of ~ 1.3 .

To further constrain the decay parameters, we have included BAO data from BOSS DR-12 as well. This leads to an even tighter constraint in the very long-lived sub-regime for Γf_{dcdm} with a 2σ upper bound of $\Gamma f_{\text{dcdm}} < 3.72 \times 10^{-3} \text{ Gyr}^{-1}$, while it is now also possible to constrain f_{dcdm} due to the BAO data lifting the degeneracy. We then get a 1σ upper bound of $f_{\text{dcdm}} < 8.05\%$. In both the long-lived and the short-lived regimes we, however, get looser constraints. In the long-lived regime we get 2σ upper bounds of $f_{\text{dcdm}} < 2.62\%$ and $\Gamma f_{\text{dcdm}} < 5.84 \times 10^{-2} \text{ Gyr}^{-1}$, while we in the short-lived regime get $f_{\text{dcdm}} < 14.9\%$ and $\Gamma f_{\text{dcdm}} < 3.78 \times 10^4 \text{ Gyr}^{-1}$.

The impact on the Hubble and σ_8 tensions has also been investigated by comparing our posterior distributions from the MCMC runs with the data from measurements in the local universe, i.e. the latest value of the Hubble parameter inferred by the distance ladder in Ref. [68] and the combined data of DES Y1 (Ref. [69]) measuring the matter fluctuations using galaxy clustering and lensing. We get a smaller discrepancy in the Hubble parameter in the short-lived regime, reducing the tension by $\sim 1\sigma$, but in this same regime we get no improvement of the σ_8 tension. The case is opposite for the long-lived regime, where we are able to relieve the σ_8 tension slightly, but not as much in the Hubble tension. We must therefore conclude that the simple Λ CDM model cannot accommodate both tensions at once.

We have in addition to this investigated how the short-lived Λ CDM (decaying much earlier than matter-radiation equality) is analogous to a universe without Λ CDM but with a larger initial amount of non-EM radiation (e.g. massless neutrinos). This has been analytically mapped to a correction to the effective number of massless neutrinos species, N_{eff} , in eq. (3.19). Using the Boltzmann code CLASS we also calculated the actual correction to N_{eff} and saw that this scales in the exact same way as our analytical expression. The analytical and numerical results

are also shown to agree with increasing precision as $\Gamma_{\text{dcdm}} \rightarrow \infty$, where we recover the standard Λ CDM model.

Reproducibility. The modified version of CLASS used to obtain the results in this paper is available at https://github.com/AarhusCosmology/CLASSpp_public/ as branch 2011.01632 with SHA 767fcdec52f8135dd8cebfcba7e1b2b3cdc7bc6a. The version of MONTEPYTHON used as well as parameter files and scripts are available at https://github.com/AarhusCosmology/montepython_public/ as branch 2011.01632 with SHA e2a8af41725ce31d63718eafe7ec614801b291f6.

ACKNOWLEDGEMENTS

The numerical results presented in this work were obtained at the Centre for Scientific Computing, Aarhus <http://phys.au.dk/forskning/cscaa/>. A.N. and T.T. was supported by a research grant (29337) from VILLUM FONDEN.

Ending of reference [1]

3.7 OTHER DECAYING DARK MATTER MODELS

As mentioned in the introduction to the chapter, I have co-authored a review chapter on decaying dark matter models for a book on the Hubble tension. This review was in part a review of the paper I have just presented, but I find it relevant to include the parts from the review that was based on other work.

Beginning of excerpt of reference [5]

3.7.1 DCDM \rightarrow DR WITH PROFILE LIKELIHOODS

The DCDM model has been investigated in numerous papers with Bayesian statistics [36–42, 44] and recently also with frequentist statis-

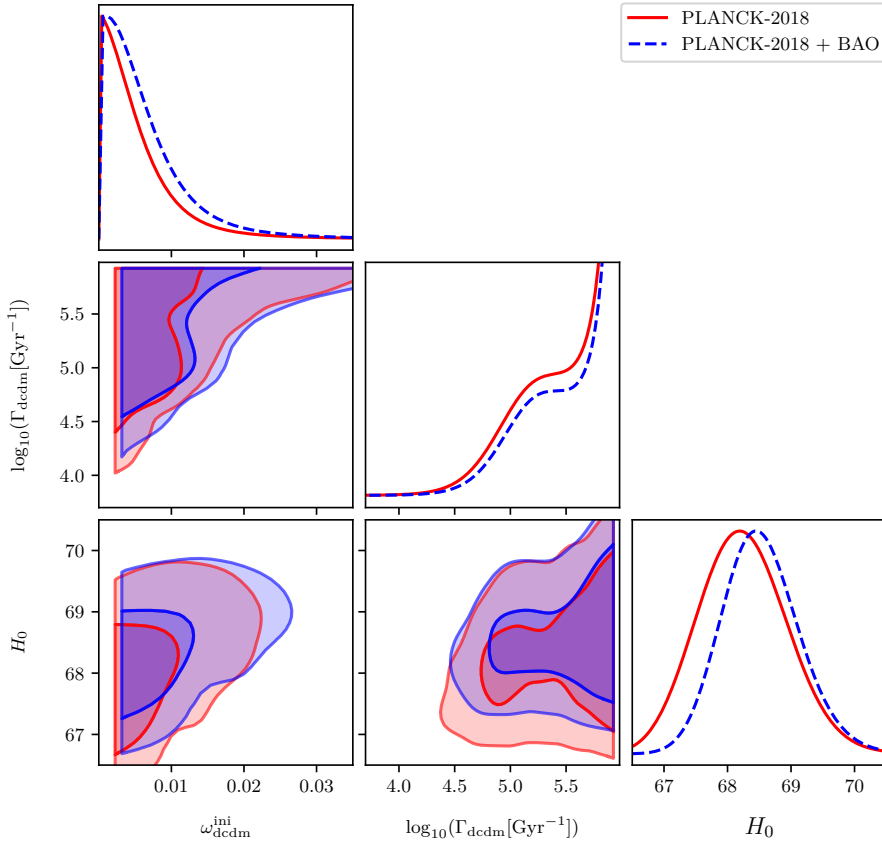


Figure 3.11: Triangle plot of posteriors in the short-lived regime using Planck 2018 data with and BAO peak data. The figure shows the model-specific parameters, $\omega_{\text{dcdm}}^{\text{ini}}$ and Γ_{dcdm} , along with H_0 , and has been produced using the same MCMC runs used in Ref. [1].

tics and profile likelihoods [7]. Figure 3.11 shows the posteriors for the short-lived regime in the model-specific parameters (along with H_0) using high- ℓ and low- ℓ temperature and polarization as well as lensing data from Planck 2018 [21] (from now on just referred to as Planck 2018 data) with and without the BOSS DR12 BAO peak data [70] (from now on referred to as BAO peak data). The posterior of the decay rate, Γ_{dcdm} , shows a volume effect towards higher values, where a large part of the parameter space fits the data well. Because of this, it is not possible to assign a reasonable credible interval to the decay rate since any such interval would be prior dependent. It is, however, also apparent that a region around $\log_{10}(\Gamma/\text{Gyr}) \in [4.5, 5.5]$ includes a separate effect in the posterior, visualized as a small “bump” or plateau,

which marks this as a region of interest. By investigating the same region in the decay rate with profile likelihoods, we find that a peak in the likelihood (a “well” in $\chi^2(\Gamma) \equiv -2 \log(\mathcal{L}(\Gamma) / \max(\mathcal{L}(\Gamma)))$) arises here, as shown in figure 3.12. This study has been performed with the same data (Planck 2018 data and BAO peak data) along with low redshift BAO data from the 6dF survey [71] and the BOSS main galaxy sample [72] (the inclusion of which will be referred to as the full BOSS DR12 data set). The approximate 68% confidence interval obtained is $\log_{10} \Gamma \text{ Gyr}^{-1} = 4.763^{+0.214}_{-0.290}$, while it is unconstrained at 95%. The global best-fit of the model lies in this region at $\Delta\chi^2 = 2.8$ relative to the Λ CDM model, hinting at a mild preference for the DCDM model over Λ CDM. Interestingly, the best-fitting parameters $\log_{10} \Gamma \text{ Gyr}^{-1} = 4.763$ and $\omega_{\text{dcdm}}^{\text{ini}} = 0.00429$ correspond to a scenario where about 3% of the cold dark matter decays just prior to recombination, in support of the tendency of early time solutions to the Hubble tension to preferentially modify physics temporally close to recombination [35, 73]. Despite the stronger signature of DCDM in the frequentist analysis, the resulting 68% constraint $H_0 = 68.14^{+0.54}_{-0.49} \text{ km s}^{-1} \text{ Mpc}^{-1}$ solidifies the conclusion that the DCDM→DR model is unable to solve the H_0 tension.

3.7.2 DCDM→DR+WDM

A reasonable increase in model complexity is to allow one of the daughter particles to be massive. We still have a cold mother particle decaying into dark radiation, but now also accompanied by a massive daughter particle acting as *warm dark matter* (WDM). We can write this process as

$$X_{\text{dcdm}} \rightarrow \gamma_{\text{dr}} + Y_{\text{wdm}}.$$

This model has the same model parameters as the DCDM→DR model (abundance and decay rate), but it also has a new parameter: The mass ratio of the WDM particle to the DCDM particle, \tilde{m} . Using conservation of energy and momentum, one can relate this mass ratio to the WDM velocity and the fraction of energy transferred to the massless DR particle in the decay, ϵ , through the following two equations [50],

$$\epsilon = \frac{1}{2}(1 - \tilde{m}^2), \quad \beta_{\text{wdm}}^2 = \frac{\epsilon^2}{(1 - \epsilon)^2}, \quad (3.22)$$

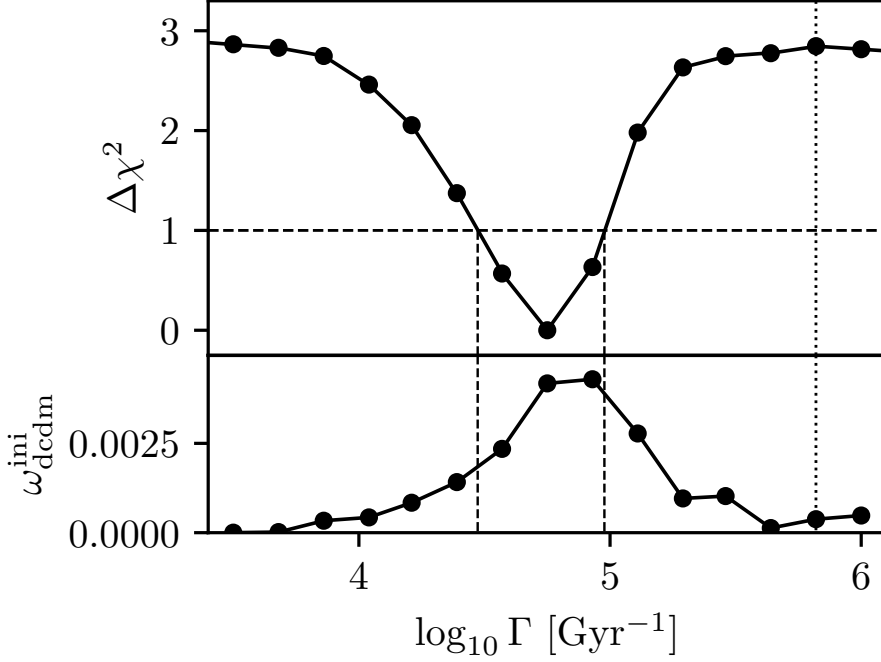


Figure 3.12: Top panel: One-dimensional profile likelihood of $\log_{10} \Gamma / \text{Gyr}$. Bottom panel: The abundance of decaying cold dark matter $\omega_{\text{dcdm}}^{\text{ini}}$ associated with every point in the profile above. Dashed lines indicate the $\Delta\chi^2 = 1$ intersections, giving the 1σ CIs. Taken from Ref. [7].

where β_{wdm} is the velocity of the massive daughter in natural units.

The background and perturbation equations are the same for DCDM and DR as in the DCDM \rightarrow DR model, except for an additional factor of ϵ on the source term in the background equation for DR. The equations for the massive daughter can be expressed in terms of the equation-of-state parameter, $w_{\text{wdm}}(a)$, which can be shown to have the following form [50],

$$w_{\text{wdm}}(a) = \frac{1}{3} \frac{\Gamma \beta^2}{1 - e^{-\Gamma t}} \int_{a_{\text{ini}}}^{a(t)} \frac{e^{-\Gamma t_D} d \ln(a_D)}{H_D [(a/a_D)^2 (1 - \beta^2) + \beta^2]}, \quad (3.23)$$

where a_{ini} refers to some initial value of a where our numerical integration begins, and the subscript “ D ” refers to the integration variable. The background equation for the massive daughter particle is then

$$\dot{\rho}_{\text{wdm}} = -3(1 + w_{\text{wdm}}(a)) \frac{a'}{a} \rho_{\text{wdm}} + (1 - \epsilon) a \Gamma \rho_0. \quad (3.24)$$

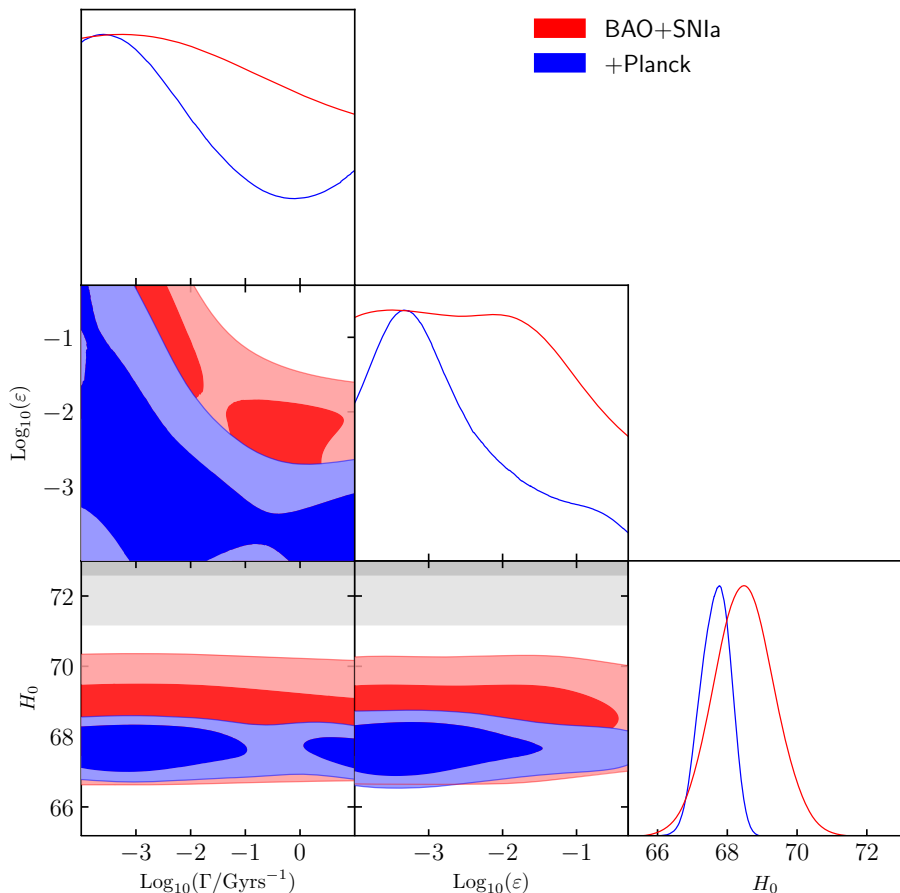


Figure 3.13: Triangle plot of the decay rate, the energy transfer ratio, and the Hubble constant. The red lines and contours are from a background-only MCMC run with the full BOSS DR12 data set and a SH0ES prior, while the blue lines and contours are from an MCMC run including perturbations (with the fluid approximation) and also the Planck 2018 data in addition to the other data sets. The gray contours are the H_0 measurement by the SH0ES collaboration. The figure is taken from Ref. [74].

The perturbation equations resemble those of massive neutrinos but can be rewritten in terms of w_{wdm} in a similar manner [74]. An accurate implementation of the full Boltzmann hierarchy fast enough for MCMC runs to be feasible is still missing, but results have been produced using a fluid approximation [33, 35, 74–76].

Figure 3.13 shows the posteriors for the model-specific parameters, Γ and ϵ , along with H_0 . These results assume that all dark matter is decaying, i.e., the abundance ratio, f_{dcdm} , is fixed to unity. Because of

this, the results are within the very long-lived regime, and the energy transfer ratio, ϵ , is rather low as well, resulting in a heavy WDM daughter particle acting much like CDM. In particular, if $\log_{10}(\epsilon) \lesssim -2.7$, the decay rate becomes unconstrained because the massive daughter particle becomes virtually indistinguishable from the cold mother particle. The 68.27% credible interval for the Hubble constant from this analysis is $H_0 = 67.71^{+0.42}_{-0.43} \text{ km s}^{-1} \text{ Mpc}^{-1}$, which is similar to the result from the long-lived regime in the simple $\text{DCDM} \rightarrow \text{DR}$ model. We would expect that an analysis with a fractional decay in the short-lived regime would yield a higher value of H_0 . In order to do this accurately, the full hierarchy should be solved for WDM instead of using a fluid approximation.

3.7.3 DWDM \rightarrow DR

Another extension of the simplest decaying dark matter model in section 3.7.1 is to allow the decaying particle a non-negligible momentum, making it a warm dark matter species. The model of a decaying warm dark matter (DWDM) species decaying to dark radiation was studied in Refs. [34, 62]. This model has several interesting particle physics realizations, such as decaying neutrinos [77] and majoron decays [78]. In particular, in [79] it was used to constrain the lifetime of the active neutrino species.

The fundamental characteristic that separates the DWDM model from the other decaying dark matter models is that the non-negligible momentum increases the lifetime by a factor of E/m through time dilation, where E and m denote the energy and mass of the decaying particle, respectively. In the general case, the model contains three parameters: The decay constant Γ (or, equivalently, the lifetime $\tau \equiv 1/\Gamma$), the DWDM mass m and the abundance of the DWDM species, specified either at final or initial time. The background equations are [34]

$$\begin{aligned} \rho'_{\text{dwdm}} &= -3 \frac{a'}{a} (\rho_{\text{dwdm}} + p_{\text{dwdm}}) - a \Gamma m n_{\text{dwdm}}, \\ \rho'_{\text{dr}} &= -4 \frac{a'}{a} \rho_{\text{dr}} + a \Gamma_{\text{dcdm}} m n_{\text{dwdm}}, \end{aligned} \tag{3.25}$$

where ρ_i and p_i denote the energy and pressure density of the i 'th species and n_{dwdm} the number density of the decaying species. Apart from the addition of the non-zero DWDM pressure p_{dwdm} , these equations are

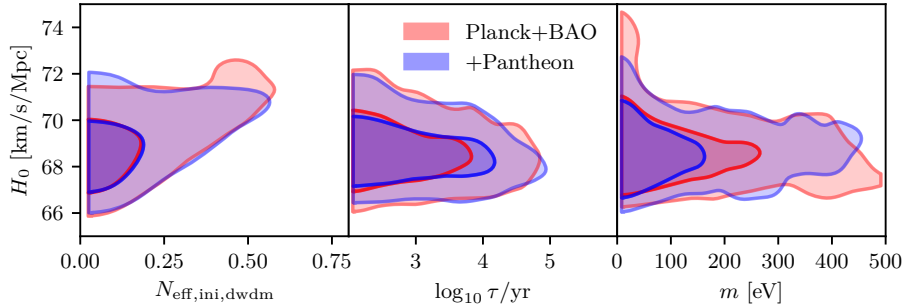


Figure 3.14: Two-dimensional marginalized posteriors for the Hubble constant H_0 and the three parameters of the DWDM \rightarrow DR model: The initial abundance, $N_{\text{eff,ini,dwdm}}$, parametrized as its contribution to the effective number of relativistic degrees of freedom, the DWDM lifetime, τ , and the DWDM mass, m . Taken from Ref. [34].

identical to the case of a cold decaying species (3.4) up to the substitution $\rho \rightarrow mn$, i.e., it is the rest mass and not the total energy that drives the decay.

As in the DCDM \rightarrow DR model, the perturbations of the DWDM species are identical to those of massive neutrinos [30]. Moreover, the decay product perturbations are influenced by a collision term, which captures the fact that the DWDM species decays preferentially at low momenta [34].

Figure 3.14, adopted from [34], shows contours of the two-dimensional marginalized posterior distributions for H_0 and three parameters of the DWDM \rightarrow DR model using Planck 2018 data combined with the full BOSS DR12 data set. Here, the initial abundance is parametrized as $N_{\text{eff,ini,dwdm}}$, its contribution to the effective number of relativistic degrees of freedom. The prior on the lifetime studied here is roughly equivalent to the short-lived regime explored earlier.

Evidently, although the maximum a posteriori estimate favors the Λ CDM limit, a considerable amount of the DWDM species is permitted by the data. Ref. [34] finds a 68% credible upper bound of $N_{\text{eff,ini,dwdm}} < 1.05$. Furthermore, as seen in figure 3.14, there is a mild correlation between H_0 and the DWDM abundance, indicating that the DWDM \rightarrow DR model may admit a larger value of H_0 than Λ CDM. A 68% credible interval of $H_0 = 68.73^{+0.81}_{-1.3} \text{ km s}^{-1} \text{ Mpc}^{-1}$, at a 2.7σ Gaussian tension with the representative local value $H_0 = 73.2 \pm 1.3 \text{ km s}^{-1} \text{ Mpc}^{-1}$ [68], is presented in this study. Thus, the broad conclusion is that there is

no evidence from CMB data that the DWDM→DR model solves the Hubble tension.

Although the constraints on the lifetime τ and mass m are strongly driven by the bounds of the uniform prior chosen, in broad terms, the data prefers small masses and small lifetimes. Unfortunately, in all treatments of the DWDM→DR model to date, the effects of *inverse decays* have been neglected. The inverse decay process is kinematically allowed when the DWDM species decays while still relativistic, which is exactly the scenario in the preferred region of parameter space [34]. Thus, a complete understanding of the DWDM→DR model inevitably requires a numerical implementation of the inverse decay processes and their quantum statistical corrections [77]. Finally, we also note that the Bayesian constraints on the DWDM→DR model are expected to be strongly influenced by prior volume effects [34] since it reduces to the Λ CDM model in the limit of vanishing abundance, making the lifetime and mass unconstrained and thereby storing a large probability mass in the posterior distribution. At this time, a frequentist analysis of the model has not been conducted.

3.7.4 DISCUSSION AND CONCLUSION

The three decay models presented in this chapter all show an ability to slightly alleviate the Hubble tension. The simplest of the models (DCDM→DR), has been exhaustively studied, and it appears that it has reached its limits regarding how much it can alleviate the tension. The other two models, however, still have potential for further investigation. The DCDM→DR+WDM model still needs a fast and accurate implementation of the full Boltzmann hierarchy, and with a fractional decay, this model can also be studied in the short-lived regime. The DWDM→DR model also prefers slightly larger values of H_0 , but a study including the inverse decay process is needed for a definite conclusion. Further investigation of the latter two models, along with improvements, could potentially alleviate the Hubble tension further. Ultimately, at the time of writing, we cannot definitively say which decaying dark matter model has the strongest alleviation of the Hubble tension, although there is a slight preference for the DWDM→DR model with the 68% C.I. $H_0 = 68.73^{+0.81}_{-1.3} \text{ km s}^{-1} \text{ Mpc}^{-1}$ at a 2.7σ Gaussian tension with the representative local value $H_0 = 73.2 \pm 1.3 \text{ km s}^{-1} \text{ Mpc}^{-1}$ [68]. Thus, although decaying dark matter models are somewhat more nat-

ural than several other proposed solutions to the H_0 discrepancy, they only accomplish a mild alleviation of the latter [35].

The next natural step to take in the hierarchy of numerical complexity would be a fully general two-body decay from a mother particle with a definite mass to two daughter particles with different masses, i.e., $\text{DWDM} \rightarrow \text{WDM}_{(1)} + \text{WDM}_{(2)}$. This is a challenging task since it combines all the most difficult aspects of the previous models while also introducing new ones. Although the full set of perturbation equations has been derived [77], an implementation in a numerical Einstein–Boltzmann solver code still remains. Furthermore, we expect the full model to be very computationally expensive to evaluate, possibly making it unfeasible for immediate inference purposes. Nevertheless, a fully general decay scheme like this would, apart from other use cases like neutrino decays, possibly be able to affect the physics around recombination in just the right way for the Hubble tension to be relieved.

ACKNOWLEDGEMENTS

A.N., E.B.H., and T.T. were supported by a research grant (29337) from VILLUM FONDEN. We would like to thank Guillermo Franco Abellán (first author of Ref. [74]) for letting us use figure 3.13 in this chapter.

End of excerpt of reference [5]

MACHINE LEARNING

Machine Learning (ML) has quickly grown from a specialised area of computer science into a key driver of modern technology, influencing both scientific advancements and everyday life. Its ability to identify patterns, make predictions, and improve decision-making has made it a vital tool in industries such as healthcare, finance, and personalised services. From diagnosing diseases to personal recommendations, ML’s impact is widespread and continues to grow.

The work I have done during my PhD has mainly been focussed on machine learning and how to utilise it to make cosmological analyses much faster and results more easily obtainable. In this regard, I have had the pleasure of exploring many aspects of machine learning as well as the challenges and frustrations that come with trying to make it work.

This chapter will introduce key concepts and techniques in ML, focusing on different learning approaches including supervised and active learning. The fundamentals of neural networks such as applicabilities and training will also be presented, and lastly, we will discuss the challenges faced during the training process and how these can be addressed.

4.1 LEARNING SCENARIOS

Depending on the task at hand, a machine learning (ML) model can be trained in a variety of ways, and in most cases, it boils down to a question of how difficult training data is to obtain. If training data is readily obtained as corresponding pairs of input and output, one can make use of *supervised learning*, where all input data has a “correct” output that the ML model should strive to reproduce through its training phase. This is opposed to the case of *unsupervised learning* where all data is unlabelled, meaning that only the input to the ML model is available [80]. It stands to reason that the performance of the ML

model, in this case, is more difficult to quantitatively evaluate, and this makes the use cases of this particular learning scenario quite different from those of supervised learning. Common applications of unsupervised learning include cluster analysis, where clusters in the data are identified by the ML model, and dimensionality reduction, where the dimensionality of the parameter space is decreased whilst retaining the same meaningful properties as the original data.

One does not simply¹ always possess all training data needed for a good ML model initially due to, e.g., a very large parameter space or a high computational cost of obtaining training data. After training an ML model with initial training data, the accuracy of the model might not be sufficient in certain regions of the parameter space due to undersampling in the region or a rapidly changing function value. In this case, one can add new points (either manually or automatically) to the original training data to accommodate this problem. This is known as *active learning*, and the goal here often is to obtain a high performance and accuracy while limiting the amount of training data [80]. This way of automatically collecting new data is more pronounced in the case of *reinforcement learning*, where the ML model is collecting information based on a reward system. The training and testing is in this case intermixed such that the ML model iteratively learns to maximise the reward through a series of actions.

In this section, we will have a deeper look into supervised learning and active learning since these are the only learning scenarios relevant to this thesis.

4.1.1 SUPERVISED LEARNING

Supervised learning uses training data consisting of inputs and outputs. The goal is to produce a machine learning model that can accurately predict the outputs based on only the inputs. In order to do this, it needs a way to accurately quantify its performance such that this performance can be optimised. The error of the model is quantified through the *loss function* which is a function of the machine learning model's output. The loss is then minimised using an optimiser resulting in a machine learning model with good predictive capabilities. The optimisation of the model is typically iterative, allowing the model to learn over time and get increasingly better predictive accuracy.

¹ ... walk into Mordor.

Generally, supervised learning can be separated into two different types of problems – classification and regression [81].

- *Classification* aims at assigning the input data to certain predefined categories. The model learns to recognise specific features or characteristics in the input data and then decides which category the data belongs to. An example of classification could be image recognition, where the training data consists of images (input) and associated labels of the categories (output). For certain machine learning algorithms, one would typically have a discretised probability distribution as the output, indicating which category has the highest probability of matching the input. A common loss function quantising the error, in this case, is the *categorical cross-entropy* (CCE) loss function which is defined (for a single training data example) as $\mathcal{L}_{CCE}(\mathbf{y}) = -\log(y_i)$, where $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$ are the probabilities (of n categories) outputted by the model and the i^{th} category is the true one [82]. The logarithm ensures that incorrect predictions are heavily penalised. Several algorithms for classification exist including neural networks, linear classifiers, support vector machines (SVMs), decision trees, k -nearest neighbour (k NN), and random forest [81].
- *Regression* is used to approximate functions, thus providing a relation between dependent and independent variables. The inputs and outputs of the training data are exactly the values of the independent and corresponding dependent variables, respectively. The output of the machine learning model will then be a prediction of the dependent variable corresponding to the independent variable used as input to the model. A common loss function in this case is the *mean squared error* (MSE) loss function where the difference between the true output and the predicted output is squared and then averaged across the number of dependent variables, i.e., $\mathcal{L}_{\text{MSE}}(\mathbf{y}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$, where $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$ are the predicted values (for a single training data example) of the n dependent variables and \hat{y}_i is the true output of the i^{th} dependent variable. Regression can be performed using linear regression and non-linear regression, but also more advanced algorithms such as neural networks [81].

Machine learning is not just a one-thing-fits-all kind of field only relying on a single algorithm to perform all tasks. Instead, numerous algo-

rithms are available depending on the task at hand, each with strengths and weaknesses. Some of the most common algorithms for supervised learning are the following [81]:

- *Neural networks* are a popular choice for various machine learning tasks due to their high number of free parameters. It draws inspiration from biological neural networks and mimics the interconnectivity herein using layers of artificial neurons connected to each other. The input is passed through the network to compute the predicted output which is then used to compute the loss. This type of algorithm is well suited for both supervised learning, unsupervised learning, and semi-supervised learning.
- *Naïve Bayes* are classifiers based on Bayes' theorem and "naïvely" assumes that features of the model are independent. Even though this assumption is often violated in realistic scenarios², the performance of the Naïve Bayes classifiers is often quite good. There are three different types of Naïve Bayes; Multinomial, Gaussian, and Bernoulli.
- *Support vector machines* (SVM) are most often used for classification, and they involve the construction of hyperplanes between classes of data points in the parameter space. This hyperplane is known as the *decision boundary* separating different classes of data points on either side.
- *k-nearest neighbour* (kNN) is an algorithm that assumes similar data being in the vicinity of each other in the parameter space. A new classification can then be made by looking at the k nearest neighbours of the new point in the parameter space and assigning it the same class as the majority within the k nearest neighbours. For small test data sets, this is a widely used algorithm for classification, but for larger data sets, the computation time grows too large, since the distances to all points need to be computed.
- *Random forest* is a collection of uncorrelated decision trees that are used for both classification and regression. The algorithm seeks to solve the inaccuracy of basic decision trees by having an ensemble of different trees all predicting the output at once. For classification, the most popular output is used, and for regression, the average of the outputs is used.

² In text classification, words are typically dependent upon preceding words.

- *Linear regression* is used to find a straight line that best describes the relationship between the dependent and independent variables. This is done by minimising the squared distances between the line and all the data points.
- *Logistic regression* is used for classification (despite the name), utilising the logistic function (a sigmoid function asymptotically approaching 0 and 1 for increasingly negative and positive inputs, respectively) to solve binary classification problems.

As an example of supervised learning, let us take a look at simple linear regression. Let $\hat{\mathbf{x}} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n\}$ be the data for our independent variable and $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n\}$ be the data for our dependent variable. We will relate the dependent variable to the independent variable through this linear relation,

$$y(x) = m x + b, \quad (4.1)$$

where m is the slope of the best-fitting line and b is the y -intercept of the line. Both of these parameters are unknown and should thus be determined through optimisation. We will use the mean squared error loss function for the optimisation which in this case looks like

$$\mathcal{L}_{\text{MSE}}(\mathbf{y}) = \frac{1}{n} \sum_{i=1}^n (y(\hat{x}_i) - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (m \hat{x}_i + b - \hat{y}_i)^2, \quad (4.2)$$

where $\mathbf{y} = \{y(\hat{x}_1), y(\hat{x}_2), \dots, y(\hat{x}_n)\}$ are the outputs of the linear model for the data of the independent variable. This quantity can be optimised analytically, which isn't the case for more advanced machine learning algorithms. In order to optimise this, both partial derivatives of the loss function with respect to the two parameters, m and b , need to equate zero, i.e.,

$$\frac{\partial \mathcal{L}}{\partial m} = \frac{2}{n} \sum_{i=1}^n (m \hat{x}_i + b - \hat{y}_i) \hat{x}_i = 0, \quad (4.3)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{2}{n} \sum_{i=1}^n m \hat{x}_i + b - \hat{y}_i = 0. \quad (4.4)$$

x	0	1	2	3	4	5	6	7	8	9	10
y	5	8	13	15	18	21	23	28	30	34	36

Table 4.1: Example data for linear regression. x and y are the independent and dependent variables, respectively.

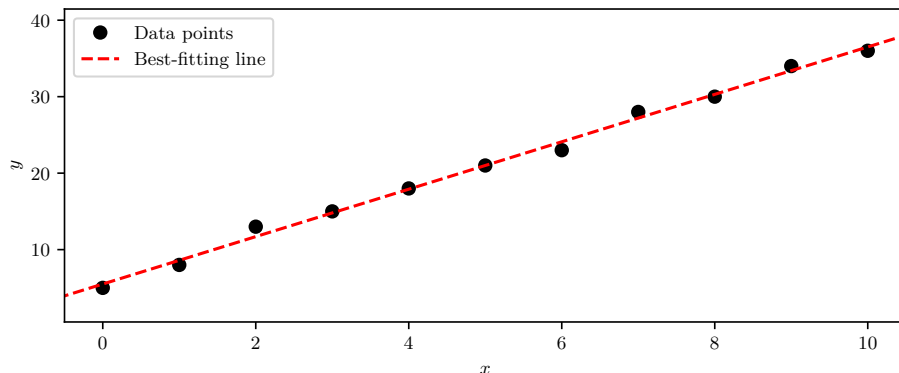


Figure 4.1: Data points from the example data in tabel 4.1 along with the best-fitting line found through simple linear regression.

These are now two equations with two unknowns, and the solutions are straightforwardly found to be

$$m = \frac{n \sum_{i=1}^n \hat{x}_i \hat{y}_i - \sum_{i=1}^n \hat{x}_i \sum_{i=1}^n \hat{y}_i}{n \sum_{i=1}^n \hat{x}_i^2 - (\sum_{i=1}^n \hat{x}_i)^2}, \quad (4.5)$$

$$b = \frac{1}{n} \left(\sum_{i=1}^n \hat{y}_i - m \sum_{i=1}^n \hat{x}_i \right). \quad (4.6)$$

These equations allow us to infer the best-fitting line of any simple linear regression problem without having to optimise the parameters numerically. For the data in table 4.1 the parameters equate to $m = 3.1$ and $b = 5.5$ and the best-fitting line along with the data points are shown in figure 4.1.

4.1.1.2 ACTIVE LEARNING

Having a representative and broad data set as training data is key when training a machine learning model to perform well. If a large part of the parameter space is only sparsely represented in the training data, the model will not have enough information to accurately predict the

output of new input located in this sparsely sampled region. A way to be sure to be able to accurately predict the output of any reasonable input is to include a large number of labelled points (pairs of input and true output) in the training data and train the model extensively on this broad set of data. This ensures that the training only needs to be done a single time in order to yield a well-rounded machine learning model with accurate predictive ability across the entire parameter space. This is known as *passive learning*. For a continuous parameter space, one might wonder how finely sampled it needs to be to ensure accurate predictions, and quite frankly this can be difficult to know beforehand. One might also be limited in the amount of obtainable labelled data by the dimensionality of the parameter space. To solve these issues, one can let the machine learning model choose which data points to label (or compute the true output for) and use for training. This is the basic idea behind *active learning* [83].

The active learning scenario operates through an iterative process of selection and retraining, and the basic steps usually involved in this are as follows [84]:

1. **Initialisation:** An initial set of training data needs to be selected and properly labelled with categories for classification problems or true output values for regression problems. This will serve as the basis for the first model training.
2. **Model training:** Using the available labelled data, a machine learning model is trained through supervised learning.
3. **Query strategy:** New data points to be labelled will be selected by the trained model based on a query strategy either prioritising decreasing the uncertainty or selecting the most dissimilar points to the ones already in the training data. Notable query strategies include *uncertainty sampling*, *diversity sampling*, and *query-by-committee sampling*.
4. **Labelling:** The newly selected points will need to be labelled by an *oracle*, i.e., the source of ground truth. This can be done by either a human annotator or an automatic computation. In the case of classification such as image classification, a human annotator is usually required, but for regression problems where the ground truth is the output of a computer program or similar, the labelling can be done automatically without the need for human interference.

5. **Model update:** The newly labelled points are added to the training data and used to retrain the machine learning model.
6. **Active learner loop:** Steps 3 to 5 are repeated iteratively until the addition of new data no longer provides significant improvements to the model's performance.

The benefits of active learning are many, and it is especially advantageous when the process of labelling the input data is costly or time-consuming. Active learning typically leads to the same (or better) performance as passive learning using much less training data since this training data consists of the most informative samples meticulously chosen to best represent the features of the entire data set. This leads to only a small fraction of the data having to be labelled which is extremely cost effective when labelling is challenging. By only keeping the most relevant samples of the training data, both the accuracy and convergence improve as well since the set of training data is smaller so as to not have the machine learning model waste resources on the more irrelevant samples during the training process. This also leads to a higher robustness to noise and, depending on the query strategy, the active learning algorithm can choose the most diverse points in order to improve generalisation as well.

There are different measures of how informative a data point is to the machine learning model and these form the bases of different query strategies [84]. *Uncertainty sampling* selects the samples that are expected to reduce the uncertainty of the machine learning model the most. This uncertainty is typically entropy or margin-based uncertainty. *Diversity sampling* seeks to select new data points based on their diversity in the current set of training data. The measure of diversity could be dissimilarity between samples, e.g., for a continuous parameter space, it could be the Euclidean distance between data points. This helps to improve the model's generalisation ability by providing data points representative of the entire parameter space. *query-by-committee sampling* involves training several individual machine learning models on different subsets of the labelled data and selecting new points based on where the models disagree. The idea is that new points in regions where the models disagree are very informative and can enable the next ensemble of models to perform better in these regions. *Expected model-change-based sampling* aims at selecting the points that are expected to lead to the largest difference in the predicted output when added to the training data. The existence

of points outside the training data that would lead to a large variation in predictions when included in the training data is a sign that these points contain information that the current training data is lacking. Adding them to the training data thus ensures a better performance by the model.

As a simple example of active learning, we will use the Fashion-MNIST dataset [85] consisting of 70,000 28×28 pixel greyscale images of clothing items in 10 different categories. We will set aside 10,000 points as a set of test data which we can use to evaluate the performance of our trained models on samples not in the training data. This leaves us with 60,000 samples in the data pool from which the active learner can request new samples. We use a neural network with a single hidden layer with 128 neurons and *ReLU* as the activation function (see section 4.2), as well as a *softmax* activation function on the output layer in order to have the 10 outputs as a discrete probability distribution. The active learner is initialised with 500 randomly selected data points from the data pool, and in each iteration, the trained model is used on the remaining samples in the data pool. This gives us a (discrete) probability distribution for each sample in the data pool that we can use to compute an uncertainty measure for every sample. We use the entropy as the uncertainty measure [86],

$$E_i = - \sum_{k=1}^{10} p_k^i \log(p_k^i), \quad (4.7)$$

where E_i is the entropy of the i^{th} sample in the data pool and p_k^i is the probability of the i^{th} input image in the data pool belonging to the k^{th} category. Using this measure, the 500 samples from the data pool with the highest entropy are selected to be labelled and included in the training data of the active learner. We continue this for 50 iterations and compute the accuracy on the 10,000 test data images each time. We also compute the accuracy on the test data of passive learners trained with different amounts of randomly selected points from the data pool to see the difference when using the uncertainty measure. The results are shown in figure 4.2, and from this, we can see that the active learner is able to achieve the same accuracy as a passive learner using the entire data pool with less than a third of the samples. The images in this data set were already labelled, but in a realistic scenario, this would

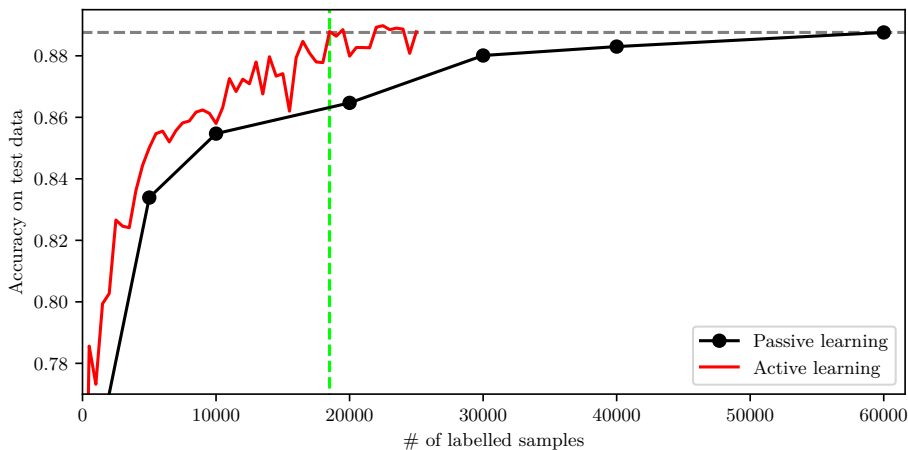


Figure 4.2: Accuracy on 10,000 test data samples as a function of amount of labelled training data. Curves for both active (red) and passive (black) learning are shown. The horizontal, grey, dashed line represents the accuracy when using the entire data set of 60,000 samples for passive training, and the vertical, green, dashed line represents the amount of labelled data acquired by the active learner in order to achieve the same accuracy.

save the oracle (human annotator or costly computation) nearly 70% of the work.

4.2 NEURAL NETWORKS

Artificial neural networks (ANNs) are foundational in modern machine learning and underpin many groundbreaking advancements in artificial intelligence. The concept of ANNs originated in 1943 when Warren McCulloch and Walter Pitts modelled a simple system of artificial neurons using electrical circuits to describe biological neural networks [87]. Heavily inspired by biological neural circuitry, ANNs adopt much of their terminology from neuroscience. In 1957, Frank Rosenblatt introduced the *perceptron* [88], the simplest form of a neural network, consisting of input neurons connected to an output neuron. The perceptron could “learn” patterns by adjusting connection weights to minimise the difference between predicted and actual outputs, an early example of supervised learning. However, scalability issues soon became apparent, as Marvin Minsky highlighted in his 1969 book *Perceptrons* [89], noting the challenges of extending perceptrons to multiple layers. This limitation stalled the progress in neural networks, with little to no advancements

in more than a decade – now known as the *AI winter* [90]. Interest revived in 1985 when David Rumelhart, Geoffrey Hinton, and Ronald Williams formalised the use of *backpropagation* for training multi-layer networks [91]. This breakthrough enabled the development of advanced architectures such as *convolutional neural networks* and *recurrent neural networks*, propelling the field to its current status.

If we take a closer look at Rosenblatt’s perceptron, it uses the McCulloch-Pitts neuron model, where inputs and outputs are binary numbers. This is inspired by biological neurons either firing or not based on some kind of threshold. We might label the binary inputs from a layer of N input neurons as x_1, x_2, \dots, x_N , and the (real-numbered) weights similarly as w_1, w_2, \dots, w_N . Rosenblatt proposed the idea of computing the binary output as

$$\text{output} = \begin{cases} 1, & \text{if } \sum_{i=1}^N w_i x_i \geq \text{threshold} \\ 0, & \text{if } \sum_{i=1}^N w_i x_i < \text{threshold} \end{cases}, \quad (4.8)$$

where the threshold is a real number like the weights and just an intrinsic parameter of the neuron [92]. A more general way of expressing this threshold is through a *bias* of the neuron. This can be interpreted as the invariant part of the prediction if the predicted outputs are centred around a non-zero mean [82]. Equation (4.8) then becomes

$$\text{output} = \begin{cases} 1, & \text{if } \sum_{i=1}^N w_i x_i + b \geq 0 \\ 0, & \text{if } \sum_{i=1}^N w_i x_i + b < 0 \end{cases}, \quad (4.9)$$

where b is the bias of the neuron.

Modern neural networks typically use other kinds of neuron models that work with real-numbered inputs and outputs instead of binary numbers. An example of this is the *sigmoid neuron* where the output is computed using the sigmoid function³,

$$\text{output} = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad \text{where } z = \sum_{i=1}^N w_i x_i + b. \quad (4.10)$$

This ensures a real-valued output between 0 and 1 instead of a binary output, and it also introduces non-linearity into the network, which

³ The sigmoid function refers to the logistic function in this case.

greatly improves the usability for non-trivial problems. Furthermore, it is a smooth function, which means that changes in the output depend linearly on changes in the weights and the bias, i.e., small changes in weights and bias lead to a small change in output. This linearity is very helpful during the training process, since knowing the response of the output makes adjusting the weights and bias much easier [92].

We can generalise this idea by replacing the sigmoid function with an arbitrary function, known as the *activation function*, $a(x)$. This now turns the output computation into

$$\text{output} = a \left(\sum_{i=1}^N w_i x_i + b \right). \quad (4.11)$$

Different activation functions have different strengths and use cases, and despite it making the training easier, they do not necessarily have to be smooth, e.g., using a step function would result exactly in Rosenblatt's perceptron.

4.2.1 MULTILAYER NEURAL NETWORKS

A perceptron only consists of a single computational layer (the output layer), which limits the usability for more complicated problems. In order to be able to fully approximate any function, multiple layers are needed. The additional layers between the input and the output layer are called *hidden layers* due to their computations being hidden from the user [82]. This multilayered architecture is known as *feed-forward* networks because computations from one layer feed into the next layer as input. This becomes apparent when describing the network mathematically. A schematic of a fully connected neural network is shown in figure 4.3.

We will consider a network with a total of M layers with different numbers of neurons in each layer. We label layers with the index $j = 0, 1, 2, \dots, M-1$ where $j = 0$ refers to the input layer and $j = M-1$ refers to the output layer. This means that we will have $M-1$ computational layers, as the input layer does no computations. All neurons within a layer will similarly be labelled with the index $i = 1, 2, \dots, N_j$ where N_j is the total number of neurons in the j^{th} layer. Each neuron in layer $j > 0$ receives input from all neurons in the previous layer, so these inputs can be labelled using three indices; the layer of the receiving neuron, the index of the receiving neuron within that layer, and the

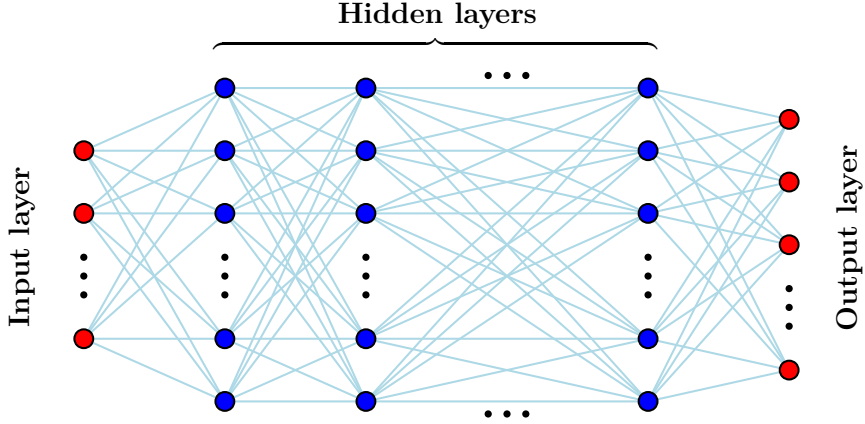


Figure 4.3: A depiction of a fully connected neural network showing the input layer, the hidden layers, and the output layer. The lines connect any two neurons in consecutive layers and have an associated weight, w , while the neurons (except the ones in the input layer) each have an associated bias, b .

index of the transmitting neuron in the previous layer. We can now compute the output, x_i^j , of the i^{th} neuron in the j^{th} layer as

$$x_i^j = a_j \left(\sum_{k=1}^{N_{j-1}} x_k^{j-1} w_{i,k}^j + b_i^j \right), \quad j > 0, \quad (4.12)$$

where the superscripts (j) of x_k^{j-1} , $w_{i,k}^j$, and b_i^j refer to the layer of the neuron in question and the subscripts (i and k) refer to the indices of the neuron in the current layer and of the neuron in the previous layer, respectively, and $a_j(x)$ is the activation function of the neurons in the j^{th} layer.

This notation is a bit cumbersome, due to the number of indices, so we will switch to a vector-based notation instead. We will define the vector $\mathbf{X}_j = \{x_1^j, x_2^j, \dots, x_{N_j}^j\}$ as the vector of the outputs from the neurons of the j^{th} layer (and thus input to the next layer) and similarly the vector $\mathbf{B}_j = \{b_1^j, b_2^j, \dots, b_{N_j}^j\}$ for the biases of the j^{th} layer. We will also define the weight matrices \overline{W}_j ($1 \leq j \leq M-1$), containing all the

weights between the j^{th} and $(j-1)^{\text{th}}$ layers. These matrices will have the dimensions $N_j \times N_{j-1}$ and will look like this:

$$\overline{W}_j = \begin{bmatrix} w_{1,1}^j & w_{1,2}^j & \cdots & w_{1,N_{j-1}}^j \\ w_{2,1}^j & w_{2,2}^j & \cdots & w_{2,N_{j-1}}^j \\ \vdots & \vdots & \ddots & \vdots \\ w_{N_j,1}^j & w_{N_j,2}^j & \cdots & w_{N_j,N_{j-1}}^j \end{bmatrix}. \quad (4.13)$$

We can now rewrite equation (4.12) more compactly as

$$\mathbf{X}_j = \mathbf{a}_j (\overline{W}_j \cdot \mathbf{X}_{j-1} + \mathbf{B}_j), \quad j > 0. \quad (4.14)$$

To get the full output vector of the entire neural network, we must feed the output as input to the next layer throughout all the layers resulting in a nested structure of equation (4.14). In the case of four layers ($j = 0, 1, 2, 3$), this will look like

$$\mathbf{X}_3 = \mathbf{a}_3 (\overline{W}_3 \cdot \mathbf{a}_2 (\overline{W}_2 \cdot \mathbf{a}_1 (\overline{W}_1 \cdot \mathbf{X}_0 + \mathbf{B}_1) + \mathbf{B}_2) + \mathbf{B}_3), \quad (4.15)$$

where \mathbf{X}_0 refers to the original inputs to the network.

From this, we can really see the strength of the non-linearity introduced by the activation functions if we consider a trivial linear activation function $a(x) = x$. This means that we can get rid of the activation functions altogether, and using the distributive and associative properties of the matrix product, we arrive at the following:

$$\begin{aligned} \mathbf{X}_3 &= \overline{W}_3 \cdot (\overline{W}_2 \cdot (\overline{W}_1 \cdot \mathbf{X}_0 + \mathbf{B}_1) + \mathbf{B}_2) + \mathbf{B}_3 \\ &= \overline{W}_3 \cdot \overline{W}_2 \cdot \overline{W}_1 \cdot \mathbf{X}_0 + \overline{W}_3 \cdot \overline{W}_2 \cdot \mathbf{B}_1 + \overline{W}_3 \cdot \mathbf{B}_2 + \mathbf{B}_3 \\ &= \underbrace{(\overline{W}_3 \cdot \overline{W}_2 \cdot \overline{W}_1)}_{\text{Effective weight matrix}} \cdot \mathbf{X}_0 + \underbrace{(\overline{W}_3 \cdot \overline{W}_2 \cdot \mathbf{B}_1 + \overline{W}_3 \cdot \mathbf{B}_2 + \mathbf{B}_3)}_{\text{Effective bias vector}}. \end{aligned} \quad (4.16)$$

This shows that without the non-linearity of the activation function, one will never achieve anything more than the performance of a single computational layer, thus rendering the network no better than a linear transformation. Activation functions are thus crucial for *deep learning* with many hidden layers. It can even be shown [93] that a network

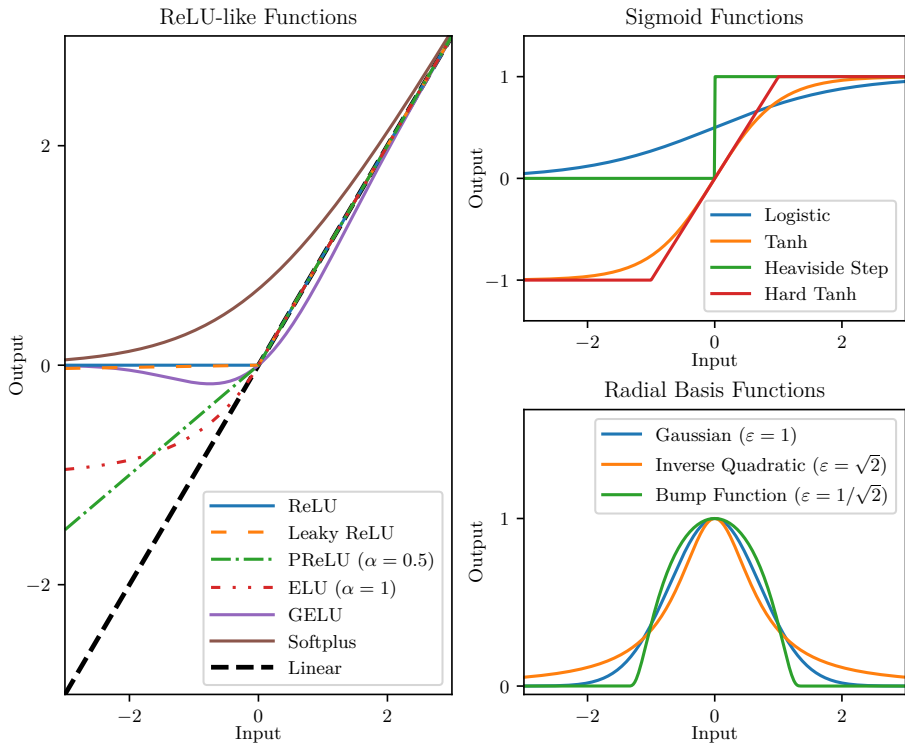


Figure 4.4: Graphs of the activation functions described in this section divided into three categories: ReLU-like functions, sigmoid functions, and radial basis functions. The values of parameters are shown in the legend where relevant.

with only a single hidden layer with a non-linear activation and a linear output layer is sufficient to approximate any function at all (as long as the hidden layer has enough neurons). This is known as *the universal approximation theorem*.

4.2.2 ACTIVATION FUNCTIONS

Now that we know activation functions are essential for deep learning, it would be beneficial to look at the most commonly used ones. Generally, one can identify a few families of activation functions, e.g., the ReLU-like functions, the sigmoid functions, and the radial basis functions. Almost all popular activation functions fall under one of these categories. This section will go through the most common choices in each category. They are all depicted in figure 4.4.

The *ReLU* (Rectified Linear Unit) function returns the non-negative part of its argument, meaning that it is linear for positive inputs and zero for negative inputs, i.e., $f(x) = \max(0, x)$. The negative part of the function renders the neuron inactive similar to how a biological neuron will not fire if it does not experience its minimum level of stimulus. There are numerous variations of this activation function in order to tackle its possible problems:

- *Leaky ReLU*: To mitigate the problem of *dying ReLU* (see section 4.2.4), one can introduce a small slope to the negative part of the activation function. This ensures that gradients remain nonzero, guiding the training process effectively.
- *Parametric ReLU* (PReLU): This generalises the Leaky ReLU by adding an arbitrary slope to the negative part.
- *Exponential Linear Unit* (ELU): This furthermore changes the negative part of the ReLU function to $f(x < 0) = \alpha(e^x - 1)$, and introduces a smooth transition between the positive and negative parts, ensuring differentiability everywhere.
- *Gaussian Error Linear Unit* (GELU): This is a different way of introducing a smooth transition in the ReLU function. The function looks like $f(x) = \frac{1}{2}x(1 + \operatorname{erf}(x/\sqrt{2}))$ and ensures the same asymptotic behaviour as the normal ReLU function.
- *Softplus*: Although this function might stand out in this category, it has the same asymptotic behaviour as the normal ReLU function with a large smoothing in between the two sides. The function is the antiderivative of the logistic function (which is also commonly used as an activation function), i.e., $f(x) = \ln(1 + e^x)$.
- *Linear*: We should also include the linear activation function $f(x) = x$ in this category since it is a special case of the parametric ReLU function. This is the trivial activation function that is typically only used in output layers.

Sigmoid functions are also widely popular as activation functions due to their squashing property, ensuring a bounded response of the neurons [94]. A sigmoid function is any function exhibiting an S-shaped behaviour, and there are thus various to choose from:

- *Logistic function*: This is what most people refer to when mentioning “the sigmoid function”. It looks like $f(x) = (1 + e^{-x})^{-1}$ and is thus bounded between 0 and 1, making it a natural extension of the binary McCulloch-Pitts neuron model.
- *Hyperbolic tangent*: The hyperbolic tangent takes the form $\tanh(x) = (e^x - e^{-x}) / (e^x + e^{-x})$, and it is equivalent to a scaled and offset logistic function. It is instead bounded between -1 and 1 and has thus a few advantages over the logistic function. Firstly, inputs keep their sign, i.e., near-zero maps to near-zero, negative to negative, and positive to positive, and secondly, the output of each layer is centred around zero from the beginning of the training process, which can lead to a faster convergence [95].
- *Heaviside step function*: This function is 0 for negative arguments and 1 for non-negative arguments, which makes it equivalent to the McCulloch-Pitts neuron model. It does not have much advantage over the other sigmoids apart from being easily computed due to it only needing a comparison operation.
- *Hard tanh*: This function is a piecewise linear function that is -1 for inputs below -1, 1 for inputs above 1 and linear between -1 and 1. It is sometimes preferred over the normal hyperbolic tangent due to it being computationally less expensive. A disadvantage is, however, that it saturates for inputs above 1 and below -1, similar to the dying ReLU problem [96].

Another class of activation functions are the *radial basis functions* (RBFs) specifically used in RBF networks. These are a special type of feed-forward network that only has a single hidden layer, but the RBFs ensure a very powerful approximating capability. They also exhibit faster convergence during training and are used for classification, regression, and time-series predictions [97]. RBFs are functions that only depend on the Euclidean distance between the input and a defined centre of the function. In RBF networks, this centre is defined using a “prototype” input from the training set that each neuron stores. There are various functions with this property that are used in RBF networks, and I will be using the parameter $r = ||\mathbf{x} - \mathbf{c}||$ as the Euclidean distance between the input \mathbf{x} and the centre \mathbf{c} :

- *Gaussian*: This is the most popular choice. The function is normalised such that its maximum is 1, i.e., $f(r) = \exp((\varepsilon r)^2)$,

where ε is a broadness parameter and inversely proportional to the Gaussian's standard deviation.

- *Inverse multiquadric*: This has the same asymptotic behaviour as the Gaussian but with a slightly different shape. Normalised in the same way, it looks like $f(r) = (1 + (\varepsilon r)^2)^{-1}$, where ε is parameter of the broadness.
- *Bump function*: This is a piecewise function with a value of $f(r < 1/\varepsilon) = \exp(-(1 + (\varepsilon r)^2)^{-1})$ for small distances away from the centre and 0 if the distance is larger than $1/\varepsilon$.

Lastly, it is worth mentioning the unique *softmax* activation function, which is almost only used in the output layer of classification networks. This activation function ensures that all output neurons sum to unity, such that each output can be considered a probability of the input matching the category corresponding to that output neuron. If a network is trained well and the input unambiguously belongs to one of the categories, the softmax activation function should ensure that every output neuron except for the one corresponding to the correct category returns a value close to zero and a value close to 1 for the correct category. If an answer is ambiguous, i.e., the input does not look entirely like the inputs in the training data of the same category, multiple output neurons will return probabilities of similar significance. With the k outputs denoted by $\mathbf{v} = \{v_1, v_2, \dots, v_k\}$, the activation function for the i^{th} output is defined as [82]

$$\Phi(\mathbf{v})_i = \frac{\exp(v_i)}{\sum_{j=1}^k \exp(v_j)} \quad \forall i \in \{1, 2, \dots, k\}. \quad (4.17)$$

4.2.3 BACKPROPAGATION

Ever since it was popularised by Rumelhart, Hinton and Williams [91], the backpropagation algorithm has been the backbone of modern-day neural networks, and many current applications would not be possible without it. The aim of backpropagation is to compute the partial derivatives $\partial \mathcal{L} / \partial w$ and $\partial \mathcal{L} / \partial b$ of the loss function \mathcal{L} with respect to any bias, b , or weight, w [92]. The loss function is a function of the network's output that describes the difference between the predicted output and the true output. This is the function that we need to minimise during the training process. An important assumption of the loss function is

that one should be able to write it as an average of loss functions for individual examples of the training data, i.e.,

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n \mathcal{L}_i, \quad (4.18)$$

where \mathcal{L}_i is the loss for the i^{th} training data example. This assumption is crucial since it is in fact the partial derivatives of single training examples, $\partial \mathcal{L}_i / \partial w$ and $\partial \mathcal{L}_i / \partial b$, that we can compute, and the partial derivatives of the entire loss function are then computed as the average [92]. For now, we will drop the subscript on the loss function and \mathcal{L} will thus refer to the loss of a single training example. The loss function will then only depend on the weights and biases, since the input and output are considered constants.

It is useful to define a measure of “error”, δ_i^j , for the i^{th} neuron in the j^{th} layer as well as a shorthand notation of the weighted input to the neuron, $z_i^j = \sum_{k=1}^{N_{j-1}} x_k^{j-1} w_{k,i}^j + b_i^j$. We will define the error as

$$\delta_i^j \equiv \frac{\partial \mathcal{L}}{\partial z_i^j}. \quad (4.19)$$

This is thus a measure of how much a change in the weighted input of the particular neuron can affect the loss function. If it is a small number, small changes to z_i^j will not have a significant impact on \mathcal{L} and the neuron is then close to optimal. In matrix form, the weighted input vector of the j^{th} layer is given by $\mathbf{Z}_j = \bar{\mathbf{W}}_j \cdot \mathbf{X}_{j-1} + \mathbf{B}_j$, and the error vector, $\mathbf{\Delta}_j$, of the j^{th} layer is similarly given by

$$\mathbf{\Delta}_j \equiv \nabla_{\mathbf{Z}_j} \mathcal{L}, \quad (4.20)$$

where $\nabla_{\mathbf{x}} f$ denotes the gradient of the scalar function f with respect to the vector \mathbf{x} and is a column vector with partial derivatives as elements.

Backpropagation is based on four equations computing this error measure of every neuron and relating it to the partial derivatives of the loss function with respect to weights and biases. Given a network with M layers indexed⁴ by $j = 1, 2, \dots, M$, the output of the final layer is denoted

⁴ The indices has been shifted by 1 compared to section 4.2.1 to ease the notation when dealing with the final layer.

by $\mathbf{X}_M = \mathbf{a}_M(\overline{\mathbf{W}}_M \cdot \mathbf{X}_{M-1} + \mathbf{B}_M)$. We can derive the error measure of the final layer in terms of its output using the chain rule,

$$\overline{\mathbf{J}}_{f \circ \mathbf{g}}(\mathbf{x}) = \overline{\mathbf{J}}_f(\mathbf{g}(\mathbf{x})) \cdot \overline{\mathbf{J}}_g(\mathbf{x}), \quad (4.21)$$

where the $\overline{\mathbf{J}}_f(\mathbf{x})$ denotes the Jacobian matrix for the vector function $\mathbf{f}(\mathbf{x})$, and the “ \circ ” denotes the composition of two functions, i.e., $\mathbf{f} \circ \mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{g}(\mathbf{x}))$. The elements of the Jacobian matrix are defined as

$$(\overline{\mathbf{J}}_f(\mathbf{x}))_{i,j} = \frac{\partial f_i}{\partial x_j}. \quad (4.22)$$

We can similarly write the loss function as a composite function of the weighted input vector, \mathbf{Z}_{M-1} ,

$$\mathcal{L} \circ \mathbf{a}_M(\mathbf{Z}_M) = \mathcal{L}(\mathbf{a}_M(\mathbf{Z}_M)), \quad (4.23)$$

where the loss function, \mathcal{L} is a scalar function with vector input, and the activation function, \mathbf{a}_M , is a vector function. This results in the composition $\mathcal{L} \circ \mathbf{a}_M$ being a scalar function as well. Using the chain rule from equation (4.21), we can now write

$$\overline{\mathbf{J}}_{\mathcal{L} \circ \mathbf{a}_M}(\mathbf{Z}_M) = \overline{\mathbf{J}}_{\mathcal{L}}(\mathbf{a}_M(\mathbf{Z}_M)) \cdot \overline{\mathbf{J}}_{\mathbf{a}_M}(\mathbf{Z}_M). \quad (4.24)$$

Recall that $\mathbf{X}_M = \mathbf{a}_M(\mathbf{Z}_M)$, and that the Jacobian of a scalar function equals the transpose of the gradient of that function. We can now use this to simplify the expression,

$$(\nabla_{\mathbf{Z}_M} \mathcal{L})^T = (\nabla_{\mathbf{X}_M} \mathcal{L})^T \cdot \overline{\mathbf{J}}_{\mathbf{a}_M}(\mathbf{Z}_M), \quad (4.25)$$

and transpose it,

$$\nabla_{\mathbf{Z}_M} \mathcal{L} = (\overline{\mathbf{J}}_{\mathbf{a}_M}(\mathbf{Z}_M))^T \cdot \nabla_{\mathbf{X}_M} \mathcal{L}. \quad (4.26)$$

By examining the last remaining Jacobian, we note that the $(i, j)^{\text{th}}$ element is the partial derivative $\partial a_{M,i} / \partial z_j^M$, where $a_{M,i} = x_i^M$ is the output of the i^{th} neuron in the M^{th} layer. The value of this partial derivative is thus only non-zero for diagonal elements since the weighted input to the j^{th} neuron does not affect the output of the i^{th} neuron unless $i = j$. The Jacobian is then a diagonal matrix, which means that the multiplication between the transpose of it and a column vector will be equivalent to

elementwise multiplication between the vector and a vector consisting of the diagonal elements of the Jacobian. This elementwise product is known as the Hadamard product [92], denoted by “ \odot ”, and we can now use it to simplify equation (4.26) and arrive at the expression for the error measure of the final (M^{th}) layer, by noting that the left-hand side of the equation exactly corresponds to our definition of the error measure from equation (4.20):

$$\Delta_M = \nabla_{\mathbf{Z}_M} \mathcal{L} = \nabla_{\mathbf{X}_M} \mathcal{L} \odot \mathbf{a}'_M(\mathbf{Z}_M), \quad (4.27)$$

where the prime denotes derivatives of elements with same indices, i.e., $\mathbf{f}'(\mathbf{x}) = \{\partial f_1/\partial x_1, \partial f_2/\partial x_2, \dots, \partial f_n/\partial x_n\}$. Equation (4.27) is the first equation necessary for backpropagation, and intuitively, it expresses that the error measure of the output neurons, i.e., how much we can do to improve the weights and biases associated with the neurons, is a product of the derivative of the loss function with respect to the output and the slope of the activation function for the weighted input. In other words, we need to have both a decent slope of the activation function and a significant derivative of the loss function for the particular neuron to be helpful in improving the network.

The second equation similarly computes the error measure of an arbitrary layer based on the error of the next layer, and this is the reason for the name “backpropagation”. First forward pass is executed in order to compute the loss function, and then one propagates backwards through the layers to compute the response of the loss function to small changes in each weight and bias. We can again use the chain rule to derive the error of the second-to-last layer by expressing the loss function as a function of the weighted input of that layer,

$$\mathcal{L}^\circ_{M-1}(\mathbf{Z}_{M-1}) = \mathcal{L}(\mathbf{a}_M(\overline{\mathbf{W}}_M \cdot \mathbf{a}_{M-1}(\mathbf{Z}_{M-1}) + \mathbf{B}_M)), \quad (4.28)$$

where \mathcal{L}°_j is introduced as a shorthand for the implicit composite function that captures the dependence of the loss function, \mathcal{L} , on the weighted input vector, \mathbf{Z}_j , of the j^{th} layer through successive compositions in the network. If we define the vector function $\mathbf{z}_j(\mathbf{x}) \equiv \overline{\mathbf{W}}_j \cdot \mathbf{x} + \mathbf{B}_j$, we can write the loss function as a composite function of four functions,

$$\mathcal{L}^\circ_{M-1}(\mathbf{Z}_{M-1}) = \mathcal{L} \circ \mathbf{a}_M \circ \mathbf{z}_M \circ \mathbf{a}_{M-1}(\mathbf{Z}_{M-1}). \quad (4.29)$$

Using the chain rule, the derivative with respect to \mathbf{Z}_{M-1} becomes

$$\begin{aligned} \bar{J}_{\mathcal{L} \circ \mathbf{a}_M \circ \mathbf{z}_M \circ \mathbf{a}_{M-1}}(\mathbf{Z}_{M-1}) \\ = \bar{J}_{\mathcal{L}}(\mathbf{X}_M) \cdot \bar{J}_{\mathbf{a}_M}(\mathbf{Z}_M) \cdot \bar{J}_{\mathbf{z}_M}(\mathbf{X}_{M-1}) \cdot \bar{J}_{\mathbf{a}_{M-1}}(\mathbf{Z}_{M-1}), \end{aligned} \quad (4.30)$$

which we can simplify using results from the previous derivation,

$$(\nabla_{\mathbf{Z}_{M-1}} \mathcal{L})^T = \underbrace{(\nabla_{\mathbf{X}_M} \mathcal{L} \odot \mathbf{a}'_M(\mathbf{Z}_M))^T}_{(\Delta_M)^T} \cdot \bar{J}_{\mathbf{z}_M}(\mathbf{X}_{M-1}) \cdot \bar{J}_{\mathbf{a}_{M-1}}(\mathbf{Z}_{M-1}). \quad (4.31)$$

The last Jacobian is also diagonal for the same reasons as its $j = M$ counterpart, and we can thus again use the Hadamard product (after transposing the equation),

$$\Delta_{M-1} = \nabla_{\mathbf{Z}_{M-1}} \mathcal{L} = \left[(\bar{J}_{\mathbf{z}_M}(\mathbf{X}_{M-1}))^T \cdot \Delta_M \right] \odot \mathbf{a}'_{M-1}(\mathbf{Z}_{M-1}). \quad (4.32)$$

We now only need to compute the Jacobian $\bar{J}_{\mathbf{z}_M}(\mathbf{X}_{M-1})$, which we can do by considering the form of the vector function $\mathbf{z}_M(\mathbf{X}_{M-1}) = \bar{W}_M \cdot \mathbf{X}_{M-1} + \mathbf{B}_M$. The first term on the right-hand side is linear in \mathbf{X}_{M-1} while the second term does not depend on \mathbf{X}_{M-1} . This means that the Jacobian is simply the weight matrix \bar{W}_M , which we can now substitute into equation (4.32),

$$\Delta_{M-1} = \left[(\bar{W}_M)^T \cdot \Delta_M \right] \odot \mathbf{a}'_{M-1}(\mathbf{Z}_{M-1}). \quad (4.33)$$

In order to arrive at the second equation of backpropagation, we note that the above equation relates the error of the $(M-1)^{\text{th}}$ layer in terms of that of the next layer. We can thus generalise this to a recurrence relation,

$$\Delta_j = \left[(\bar{W}_{j+1})^T \cdot \Delta_{j+1} \right] \odot \mathbf{a}'_j(\mathbf{Z}_j), \quad 1 < j < M. \quad (4.34)$$

We can think about this equation in much the same way as the previous one, i.e., the product of the derivative of the loss function with respect to the output of the layer and the slope of the activation function for the weighted input to the layer. The loss function, however, does not

appear explicitly, but only through the error measure of the next layer propagated backwards using the transpose of the weight matrix between the two layers. Just as the weight matrix is used to propagate inputs from one layer to the next during the forward pass, we can think of applying the transpose of the weight matrix as the opposite of that.

The final two equations of backpropagation relate the error measure to the partial derivatives of the loss function with respect to the biases and weights. If we take a look at the error measure of the j^{th} layer, recall that this is defined as $\Delta_j \equiv \nabla_{\mathbf{Z}_j} \mathcal{L}$. We can again write the loss function as a composite function to have it as a function of the bias, \mathbf{B}_j , of the j^{th} layer, but first, we need to express the weighted input, \mathbf{Z}_j , in terms of the bias using the following definition,

$$\mathbf{z}_j^B(\mathbf{B}_j) \equiv \overline{\mathbf{W}}_j \cdot \mathbf{X}_{j-1} + \mathbf{B}_j. \quad (4.35)$$

We can now write the composite function relating the loss function and the bias as

$$\mathcal{L}_j^\circ(\mathbf{z}_j^B(\mathbf{B}_j)) = \mathcal{L}_j^\circ \circ \mathbf{z}_j^B(\mathbf{B}_j), \quad (4.36)$$

and use the chain rule once more,

$$\overline{\mathcal{J}}_{\mathcal{L}_j^\circ \circ \mathbf{z}_j^B}(\mathbf{B}_j) = \overline{\mathcal{J}}_{\mathcal{L}_j^\circ}(\mathbf{Z}_j) \cdot \overline{\mathcal{J}}_{\mathbf{z}_j^B}(\mathbf{B}_j). \quad (4.37)$$

The Jacobian on the left-hand side is exactly the transpose of the gradient of the loss function with respect to the bias vector of the j^{th} layer, and the first Jacobian on the right-hand side is exactly the transpose of the gradient of the loss function with respect to the weighted input of the j^{th} layer, and thus the error measure of that layer. The last Jacobian can be found by considering the definition in equation (4.35). We note that only the second term depends on \mathbf{B}_j and is in fact exactly equal to it, so the last Jacobian is just the identity matrix. This means that the gradient of the loss function with respect to the bias vector of the j^{th} layer is equal to the error measure of that layer,

$$\nabla_{\mathbf{B}_j} \mathcal{L} = \Delta_j, \quad 1 < j < M. \quad (4.38)$$

The last equation can be derived in a similar fashion by defining yet another vector function with the weight matrix as the argument,

$$\mathbf{z}_j^W(\overline{\mathbf{W}}_j) \equiv \overline{\mathbf{W}}_j \cdot \mathbf{X}_{j-1} + \mathbf{B}_j. \quad (4.39)$$

The composite function relating the loss function and the weight matrix can be expressed as

$$\mathcal{L}_j^\circ(\mathbf{z}_j^W(\overline{\mathbf{W}}_j)) = \mathcal{L}_j^\circ \circ \mathbf{z}_j^W(\overline{\mathbf{W}}_j), \quad (4.40)$$

and we can now use the chain rule one last time,

$$\overline{J}_{\mathcal{L}_j^\circ \circ \mathbf{z}_j^W}(\overline{\mathbf{W}}_j) = \overline{J}_{\mathcal{L}_j^\circ}(\mathbf{Z}_j) \cdot \overline{J}_{\mathbf{z}_j^W}(\overline{\mathbf{W}}_j). \quad (4.41)$$

The elements of Jacobian on the left-hand side are the exact partial derivatives of the loss function with respect to each weight in the weight matrix that we want to compute, with the $(i, k)^{\text{th}}$ element being the partial derivative $\partial \mathcal{L} / \partial w_{i,k}^j$. The first Jacobian on the right-hand side is once again just the transpose of the error measure of the j^{th} layer, but the last Jacobian will be a rank 3 tensor since it is the derivative of a vector with respect to a matrix. To simplify this, it is easier to consider the individual components of the left-hand side of equation (4.41). We can write an expression for the $(i, k)^{\text{th}}$ component of the partial derivative of the loss function with respect to the weight matrix,

$$\frac{\partial \mathcal{L}}{\partial w_{i,k}^j} = (\Delta_j)^T \cdot \frac{\partial \mathbf{Z}_j}{\partial w_{i,k}^j}, \quad (4.42)$$

where the partial derivative on the right-hand side is a column vector with elements $\partial z_l^j / \partial w_{i,k}^j$, where we define the components of the weighted input vector as

$$z_l^j = \sum_{k=1}^{N_{j-1}} x_k^{j-1} w_{l,k}^j + b_l^j, \quad l = 1, 2, \dots, N_j. \quad (4.43)$$

We note that the partial derivative of this with respect to $w_{i,k}^j$ is x_k^{j-1} if $l = i$ and 0 for all other values of l . This turns the inner product with

the error vector in equation (4.42) into a product between just the i^{th} elements of the vectors,

$$\frac{\partial \mathcal{L}}{\partial w_{i,k}^j} = \delta_i^j x_k^{j-1}, \quad (4.44)$$

which is the final result for the individual components of the weight matrices. We, however, still want the last equation to be in matrix form as well as the other three, and to accomplish this, we note that the right-hand side of equation (4.44) is merely the $(i, k)^{\text{th}}$ element of the outer product between the vectors Δ_j and \mathbf{X}_j . This outer product can be substituted into the right-hand side of equation (4.41) where we also replace the notation of the Jacobian with a partial derivative of the loss function with respect to the weight matrix,

$$\frac{\partial \mathcal{L}}{\partial \overline{\mathbf{W}}_j} = \Delta_j \cdot (\mathbf{X}_{j-1})^T, \quad 1 < j < M. \quad (4.45)$$

This concludes the derivation of the equations for backpropagation in neural networks, and in summary, the four equations are

Equations for backpropagation

$$\Delta_M = \nabla_{\mathbf{Z}_M} \mathcal{L} = \nabla_{\mathbf{X}_M} \mathcal{L} \odot \mathbf{a}'_M(\mathbf{Z}_M), \quad (4.27)$$

$$\Delta_j = \left[(\overline{\mathbf{W}}_{j+1})^T \cdot \Delta_{j+1} \right] \odot \mathbf{a}'_j(\mathbf{Z}_j), \quad (4.34)$$

$$\nabla_{\mathbf{B}_j} \mathcal{L} = \Delta_j, \quad (4.38)$$

$$\frac{\partial \mathcal{L}}{\partial \overline{\mathbf{W}}_j} = \Delta_j \cdot (\mathbf{X}_{j-1})^T, \quad 1 < j < M. \quad (4.45)$$

Using these equations, one can compute all partial derivatives after a single forward pass through the network for a single training example. In practice, when one has a batch of several training examples, a forward pass is made for each of them and the partial derivatives are averaged and used to update the weights and biases. This is performed within an outer loop generating batches from the entire set of training data and yet another outer loop stepping through multiple *epochs* of training [92].

4.2.4 COMMON CHALLENGES DURING TRAINING

While we dream of neural networks being mystical black boxes that solve all our problems with a wave of a mathematical⁵ wand, the reality is far less enchanting. Instead of magic, we encounter a series of puzzles, missteps, and challenges that demand patience, precision, and an occasional urge to yell at our screens. From vanishing gradients to overfitting, the training process is as much art as it is science – albeit without the stage lights and applause. We will go through the most common pitfalls and challenges when training neural networks as well as how one might avoid them or overcome them.

Overfitting occurs when the neural network tries to learn the noise of the training data instead of the features. The network will try to learn the behaviour of the individual samples of the training data instead of the overall trend in the data. This of course minimises the loss function specifically for the training data, but any new data sample given to the network will result in a large error. Overfitting typically occurs when the training data set is too small and the network is too complex. In this case, the network will have enough weights and biases to represent the entire data set exactly, and given enough epochs during training, it will do just that simply because it has the capacity to do so to further minimise the loss function. There are various ways to accommodate this problem depending on the task at hand [98].

One solution is to decrease the complexity of the network to have fewer free parameters to train. One, however, needs to be careful that the complexity is not decreased so much as to cause the problem of *underfitting* where the network does not have the capacity to represent the trends of the training data. This, however, results in poor predictive ability for both the training data and the test data, so it is much easier to spot.

Another solution is to increase the training dataset. However, this may not always be feasible, either because generating more data is impractical or because the available pool of training data has already been exhausted. For certain problems such as image recognition, it is possible to artificially produce more training data through *data augmentation*. This means that images in the training data set are repeated with alterations such as rotations, translations, adjusted brightness, added noise,

⁵ This is not a typo.

etc. This will increase the number of training data images and the network will then be trained on multiple instances of the same object in different perspectives.

An overfitted neural network typically has weights with large values, since small changes in input can lead to big changes in the output. To avoid this, a third solution is to use *regularisation*. One type of regularisation introduces a penalty term in the loss function adding either the absolute values of the weights (L1 regularisation) or the square of the weights (L2 regularisation),

$$\mathcal{L}_{L1}(x) = \mathcal{L}(x) + \lambda \sum_i |w_i| \quad (4.46)$$

$$\mathcal{L}_{L2}(x) = \mathcal{L}(x) + \lambda \sum_i w_i^2, \quad (4.47)$$

where λ is a hyperparameter controlling how much the regularisation term should influence the total loss function. Choosing one or the other depends on the simplicity/complexity of the data as L2 is able to model the inherent patterns in the training data for more complex data. Regularisation can also be done without modifying the loss function by using *dropout* instead. This deactivates a certain number of neurons within a layer randomly with each forward pass during training. This ensures more stability and robustness in the entire network since individual neurons will have less effect on the final output.

A more simple solution to overfitting is *early stopping* where the training process is halted just before overfitting begins. Usually, both the total loss for the training data and the test data will decrease with the epoch number until they start to flatten. At this point, the test loss may start to increase again if the network is prone to overfitting, and it is at this point early stopping should be done. To further decrease the loss after this point, the training may be resumed with a smaller learning rate in the optimiser or a different optimiser altogether.

Vanishing gradients occur when the gradients of the loss function with respect to the weights in the early layers become extremely small. This leads to minimal updates in the weights of the early layers, resulting in very slow convergence during training. In order to understand the vanishing gradients problem, we can take a look at the derivations in section 4.2.3 and specifically the equations (4.27), (4.34), (4.38), and (4.45). These tell us that the gradient of the loss function with respect to the

weight matrix of the j^{th} layer is a product of all the weight matrices and the derivative of the activation functions from the subsequent layers. If many weights are initialised with small values (less than 1), the product of these weights becomes increasingly small for earlier layers. Furthermore, if the activation functions saturate (as is the case with sigmoid functions), their derivatives also yield small values, compounding the issue and further reducing the gradients in the earlier layers [99]. A similar problem can arise for weights with large values known as the *exploding gradients* problem [92]. There are a few ways to mitigate these kinds of problems.

The architecture of the network can greatly impact how profound the problem of vanishing gradients is. By reducing the complexity of the network (especially the number of layers), one will typically reduce the problem as well, since fewer small numbers are multiplied when computing gradients for the earliest layers. This, however, might impact the performance as well since the capacity of the network is also reduced. To circumvent this while still mitigating the problem of vanishing gradients, other types of network architecture might be used. In particular, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are designed to have much better gradient flow and propagation of information across the layers, which in turn reduces the vanishing gradients problem. Another type of architecture that better facilitates the flow of gradients is residual networks (ResNets) that introduce a *skip connection* adding the output of one layer to the input of the next. This type of architecture can make the training of deeper models easier.

A better flow of gradients across layers can also be accomplished with *batch normalisation*, where the input to each layer is normalised to zero mean and unit variance. This in turn leads to faster convergence and it also acts as a method of regularisation resulting in better performance. Another benefit of batch normalisation is that it is robust to changes in hyperparameters and improves stability during the training process.

Since the derivative of the activation function contributes to the gradients, choosing an activation function without saturation can prevent the vanishing gradients. Choosing ReLU as the activation function ensures that gradients flow freely between layers, and its computational simplicity makes it a popular choice for deep learning. The function, however, has a constant value of zero for neg-

active inputs, which can lead to a whole separate problem (dying ReLU).

Dying ReLU refers to the persistent inactiveness of neurons with very negative weighted inputs. Since the negative part of the ReLU function is constant, gradients cannot help the optimiser push the weights to other values resulting in positive weighted input. As a result, neurons with associated weights that always lead to negative values will be inactive (*dead state*) and usually remain inactive through the entire training process. It is not a problem to have neurons receiving a negative weighted input, since the entire non-linearity of the ReLU relies on this, i.e., without the negative part, it would just be a linear activation function effectively resulting in a single-layer network. The problem is if several neurons end up in the dead state where they remain inactive for all realistic input. Dying ReLU can occur if the learning rate is too high since the weights are updated by subtracting the factor of change, i.e., the partial derivative of the loss with respect to the weight, multiplied by the learning rate. If this update term is large, the new weight will be very negative and drive the weighted input towards negative values. Another reason for the problem could be having a large negative bias since this is also a part of the weighted input entering the activation function [100].

A solution to this could be to lower the learning rate and also initialise positive biases for the neurons such that the weights do not get pushed to negative values and the bias can help push the weighted input to positive values. Another solution is to slightly modify the ReLU function to introduce gradients. An example is the Leaky ReLU function (see section 4.2.2) which substitutes the constant function for negative inputs with a function with a tiny positive slope. This will not significantly impact the overall behaviour of the function since a negative input still results in a near-zero value, but the optimiser can now have access to information about the gradients in order to help adjust the weights and avoid the problem of dying ReLU completely.

Part II

EMULATION OF COSMOLOGY

The difference between us and a computer is that, the computer is blindingly stupid, but it is capable of being stupid many, many million times a second.

Douglas Adams

THE CONNECT FRAMEWORK

As cosmological models become more advanced, the computational requirements for their analysis increase exponentially. This challenge became especially evident when dealing with more complex models of decaying dark matter, which highlighted the limitations of conventional parameter inference in cosmology. The need for faster computations in these models became urgent, propelling my PhD in a completely new direction: finding a solution to the problem of slow model computation. Initially, our goal was simply to leverage the codes that were available at the time, but these were not easily accessible for public use and did not fully accommodate our needs. This led me to develop what would later become the neural network framework CONNECT.

This chapter presents the following release paper of the CONNECT code in its entirety:

- *Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. “CONNECT: a neural network based framework for emulating cosmological observables and cosmological parameter inference.” In: JCAP 05 (2023), p. 025. doi: 10.1088/1475-7516/2023/05/025. arXiv: 2205.15726 [astro-ph.IM].*

The paper delves into the key considerations that guided the development of the code, as well as the active learning algorithm that I designed to enable more efficient training. This algorithm serves as the core of the CONNECT framework, distinguishing it from other neural network-based tools used for cosmological parameter inference. Its ability to intelligently select data for training has made it a valuable resource for a variety of applications, as we will explore in the chapters that follow.

Developing CONNECT has been an exciting and rewarding experience. Over time, the framework has grown into a highly adaptable tool for cosmological emulation. However, as mentioned in the accompanying paper, it remains a work in progress. Given the rapidly advancing field of machine learning, the methods behind CONNECT will likely continue

to evolve, and it is my hope that it will remain a valuable asset in cosmological research for many years to come.

————— Beginning of reference [2] —————

CONNECT: A NEURAL NETWORK BASED FRAMEWORK FOR EMULATING COSMOLOGICAL OBSERVABLES AND COSMOLOGICAL PARAMETER INFERENCE


Andreas Nygaard^a, Emil Brinch Holm^a, Steen Hannestad^a, Thomas Tram^a

^aDepartment of Physics and Astronomy, Aarhus University, DK-8000 Aarhus C, Denmark

Abstract. Bayesian parameter inference is an essential tool in modern cosmology, and typically requires the calculation of 10^5 – 10^6 theoretical models for each inference of model parameters for a given dataset combination. Computing these models by solving the linearised Einstein–Boltzmann system usually takes tens of CPU core-seconds per model, making the entire process very computationally expensive.

In this paper we present CONNECT, a neural network framework emulating CLASS computations as an easy-to-use plug-in for the popular sampler MONTEPYTHON. CONNECT uses an iteratively trained neural network which emulates the observables usually computed by CLASS. The training data is generated using CLASS, but using a novel algorithm for generating favourable points in parameter space for training data, the required number of CLASS-evaluations can be reduced by two orders of magnitude compared to a traditional inference run. Once CONNECT has been trained for a given model, no additional training is required for different dataset combinations, making CONNECT many orders of magnitude faster than CLASS (and making the inference process entirely dominated by the speed of the likelihood calculation).

For the models investigated in this paper we find that cosmological parameter inference run with CONNECT produces posteriors which differ from the posteriors derived using CLASS by typically less than 0.01–0.1 standard deviations for all parameters. We also stress that the

training data can be produced in parallel, making efficient use of all available compute resources. The `CONNECT` code is publicly available for download on GitHub .

5.1 INTRODUCTION

For the past two decades the method of choice for cosmological parameter estimation has been based on stochastic optimisation techniques, typically Markov-chain Monte Carlo (MCMC) methods. These methods have the advantage that they are very robust and do not require derivatives of the cost (likelihood) function. They also easily scale to large numbers of parameters which allows for a simple treatment of nuisance parameters. However, a major disadvantage is that a single calculation of the cost function in cosmology can be very expensive because it requires a full solution of the Einstein–Boltzmann equations of linear perturbation theory (and perhaps even a calculation of non-linear corrections). Such a computation typically takes tens of seconds on a single CPU core, and does not parallelise well beyond 10 cores. A fully converged MCMC run, typically requires between 10^5 and 10^6 solutions of the Einstein–Boltzmann solver, so the total computation time can easily reach days or weeks, in particular for more complex cosmological models. Furthermore, a new MCMC run must be performed either when a new cosmological model is required or when new data is added. In the latter case, which is common for modern application, the complete analysis with several datasets can be prohibitively expensive numerically.

The purpose of the present paper is to remedy this through a new framework for emulating cosmological observables based on machine learning via neural networks (NN) which we call `CONNECT` (**C**osmological **N**eural **N**etwork **E**mulator of **C**lass using **T**ensorFlow). We demonstrate, via a new plug-in written for the publicly available `MONTEPYTHON` MCMC code [63, 101] that we can reduce the time required for a full MCMC run to hours rather than days or weeks. A similar plug-in for the code `Cobaya` [102] has also been implemented. `CONNECT` assumes a cosmological model but is independent of the targeted dataset, and separates itself from other Einstein–Boltzmann emulators by allowing for user-friendly plug-and-play generation of a neural network emulator for any cosmology that the user may want to investigate, the only requirement being a work-

ing CLASS implementation. With a simple Boolean input argument supplied, we have modified MONTEPYTHON to automatically generate training data with CLASS, train a neural network emulator to sufficient precision and conduct the MCMC analysis using this emulator, with a very significant decrease in total computation time relative to simply running an MCMC analysis directly with CLASS.

The idea of using machine learning and specifically neural networks to speed up computations in cosmology has existed for several years. Much focus has been on emulating N -body codes (e.g. [103]) due to them being massively time consuming. Since the training data is expensive to generate, the field of emulating N -body codes is markedly data starved. Accordingly, the machine learning tools usually employed in that context include Gaussian processes [104, 105] and polynomial chaos expansion coupled with principal component analysis [103]. However, when large data samples are available, and especially when the dimensionality is large, neural networks are often superior to other supervised learning strategies (as evident, for example, in the recent dominance of neural networks in the ImageNet Large Scale Visual Recognition Challenge [106]), and are therefore the obvious choice of strategy for emulating Einstein–Boltzmann codes which are many orders of magnitude faster at generating data than N -body codes.

Examples of use of neural networks in emulation of Einstein–Boltzmann codes date back to the early COSMONET [107, 108], and the approach has since been revisited on numerous occasions with various target variables. CLASSNET [109, 110] is embedded in CLASS and learns the source functions, reducing the time required to solve linear perturbation equations. Ref. [111] targets LSS angular power spectra, whereas ref. [112] learns the linear matter power spectrum, both using neural networks. More recently, COSMOPOWER [113] emulates CLASS computations of CMB spectra with temperature, polarisation and lensing anisotropies, as well as the matter power spectrum. CONNECT, contrary to these works, emulates a wide range of customisable outputs: The user simply defines the desired CLASS output variables in an input file, and the CONNECT framework automatically generates a network emulating these.

Although Einstein–Boltzmann solvers generate data much faster than their N -body siblings, the total time required for the combined process of gathering data, training a network from it and performing parameter inference with the network, is still dominated by the data generation.

To optimise the emulation scheme, it is therefore most vital to improve on the data gathering method, e.g. by optimising the amount of information extracted from each Einstein–Boltzmann computation or by generating training data in the most important points of the cosmological parameter space. These topics fall under the machine learning field of active learning [114–116]. Most early active learning algorithms focused on selecting new data at regions of large uncertainty of the emulator in so-called uncertainty sampling. Individual active learning algorithms in uncertainty sampling typically differ on how they approximate the uncertainty of the emulator. Query-by-committee [115] algorithms estimate the uncertainty as the spread in predictions from a set of learners trained on the currently available dataset, whereas expected model change approaches [115], such as the expected gradient length algorithm [117], select new data that optimise an approximate expected improvement of the emulator; e.g. where the new training gradient has the largest magnitude in the case of expected gradient length sampling. In the case of a fully connected neural network emulator, however, a measure of network uncertainty is not readily available. Neural network uncertainties can be naturally estimated using architectures such as Monte Carlo dropout [118] or Bayesian neural networks [119], but given the scope of the paper, we leave such endeavours for future investigation.

Furthermore, an important distinction between the classical active learning applications and the one at hand is that in addition to minimising the global emulator uncertainty, we are especially interested in minimising the error in the regions of parameter space that correspond to cosmologies of large likelihood. This duality, i.e. selecting data where (i) the likelihood value is large and (ii) the current emulator uncertainty is large, has been explored previously in the context of cosmological inference. Particular examples of such active learning strategies in cosmology include ref. [120], in which a Gaussian process is used to emulate the likelihood function, from which new data can be selected based on their weighting according to some balance of the Gaussian process uncertainty and its current estimate of the likelihood at the proposed points. A similar approach was adopted in ref. [121] in an iterative fashion, as well as in ref. [122], where it was found that the exploratory behaviour is increasingly important in many-dimensional problems. However, with Einstein–Boltzmann emulators, the overhead introduced by the Gaussian processes in the Bayesian optimisation may

cancel this gain in efficiency since Gaussian processes are known to scale disadvantageously with the size of the training data [123]. Additionally, batch acquisition can be non-trivial, and the optimization of the acquisition function itself contributes considerable overhead, rendering such more advanced methods of active learning useful when the generation of data is slow, e.g. for N-body simulations. Indeed, it was shown in [122] that the overhead involved in these acquisition methods often becomes the computational bottleneck when emulating the relatively fast Einstein–Boltzmann emulators.

In this work, we present an iterative data generating procedure that with little overhead combines the focus of data generation around regions of large likelihood while still being spread to reduce uncertainty far from the maximum likelihood. Our algorithm produces parameter space samples with an MCMC chain run by a neural network iteratively trained on the same points, including a method of protecting against spuriously oversampled regions. With this, we find vastly increased emulator accuracy relative to a standard Latin hypercube sampling of training data. We developed the framework with the notoriously difficult posterior of decaying cold dark matter [1] as a reference, and tested it *blindly* on a Λ CDM model with variable neutrino mass and degeneracy parameter, on both of which it performs excellently.

This paper is structured as follows. In section 5.2 we specify the design of the neural network architecture employed in CONNECT, and in section 5.3 we describe the novel iterative algorithm for placing training data at advantageous points in the parameter space. In section 5.4 we describe the use of CONNECT through MONTEPYTHON and present resulting MCMC analyses, using CONNECT, for the decaying cold dark matter and massive neutrino cosmological models. Finally, we discuss and conclude on our findings in section 5.5.

5.2 NEURAL NETWORK DESIGN

The method used for the emulation is a fully connected deep neural network consisting of an input layer, multiple hidden layers, and an output layer (see e.g. [124] for a recent overview). The input layer consists of the cosmological parameters from which we would like to extract an output, i.e. any numeric parameter that the Einstein–Boltzmann solver code CLASS takes as input [124]. The hidden layers have a much larger dimension of several hundreds of nodes, in order to create enough train-

able weights for the network to find the correct behaviour of the CLASS computations. The output layer consists of all the specified spectra and output parameters one wishes to emulate — this being any output that CLASS can compute (CMB spectra, matter power spectra, background functions, thermodynamical parameters, and derived parameters).

The first step is to gather training data for the network which requires a method for sampling in the space of cosmological input parameters. The construction of this method will be further discussed in section 5.3. When the sample of input values has been constructed, we can use CLASS to calculate the specified output values for each point in the sampled data. This is then combined to a single output array for each point while the cosmological input parameters are put in a single input array for each point. Together, the set of input arrays and output arrays constitute our training data.

Using the TensorFlow framework [125] we can now train the network on the training data for a specified number of *cycles*, where a cycle refers to an update of the network weights. Each network used for our results has been trained for 300 cycles¹ and with batch-sizes of 512. The *loss function* is then minimised using a specified optimisation algorithm (We have used the ADAM optimiser [126] for this work and as the default in CONNECT) which slightly tweaks the weights of the network while propagating backwards. The Network is then ready for the next cycle where the whole procedure is repeated in order for the network to perform better with each cycle.

5.2.1 NETWORK ARCHITECTURE

An Einstein–Boltzmann solver can be seen as a function mapping the cosmological parameters into a set of observables such as the CMB anisotropy coefficients or the linear matter power spectrum. Since this mapping can be very general, the most conservative neural network structure to employ is a fully connected, feed-forward, deep neural network [124]². This involves several hidden layers where each node in a layer is connected to each node in the next layer. For all results in this

1 Except for a single massive neutrino model trained on a Latin hypercube consisting of 10^6 points used solely for comparison in figure 5.13. This has only trained for 100 cycles due to the large amount of data causing the optimiser to diverge, but the accuracy of the network stagnates quickly so there would be little to no gain with more cycles anyway.

2 Other Einstein–Boltzmann emulators have used different architectures. For example, ref. [110] used convolutional layers [124]. However, such choices are always motivated

paper we use 6 hidden layers with 512 nodes in each, inspired by the architecture in ref. [113]. Too few nodes and layers restrict the ability of the network to emulate the desired computation, and too many both make it prone to over-fitting [124] and require larger datasets. We find that our chosen values evade both of these concerns, and by varying these network parameters slightly, we find only modest changes in the network performance. We therefore leave a more thorough investigation of the optimal network architecture for future work. However, one soft requirement is that the evaluation time of the CONNECT architecture must not exceed the evaluation time of typical likelihood codes such as Planck [127] or Planck lite [128]. We have conducted rough benchmarking of the likelihood codes and the CONNECT evaluation time, and find that at around 12 layers, the evaluation time of CONNECT becomes greater than the evaluation time of *Planck lite*, giving an approximate upper bound on the architecture complexity. Furthermore, since CONNECT allows the user to choose the outputs to emulate, one should keep in mind that the ideal architecture will vary with the size of the output, with larger outputs naturally requiring a larger network complexity. For example, Einstein–Boltzmann solvers such as CLASS do not evaluate C_ℓ coefficients for each ℓ , but rather at a reduced set of approximately 10^2 ℓ -values from which the full sets of C_ℓ coefficients are constructed by interpolation. This significantly reduces the output dimension and we have consequently chosen the set of ℓ -values directly computed by CLASS for the output layer.

5.2.2 CHOOSING A LOSS FUNCTION

When training a neural network, one always has to make choices regarding the optimisation of the network. First of all, we need a way of quantifying how well the output from the network fits the desired output from the training data – the *loss function* [124]. A simple choice for a loss function would be the widely-used *mean squared error* (MSE) function,

$$\mathcal{L}_{\text{MSE}}(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2, \quad (5.1)$$

by some properties of the underlying physics, and since CONNECT emulates customizable CLASS outputs, we cannot directly make such assumptions.

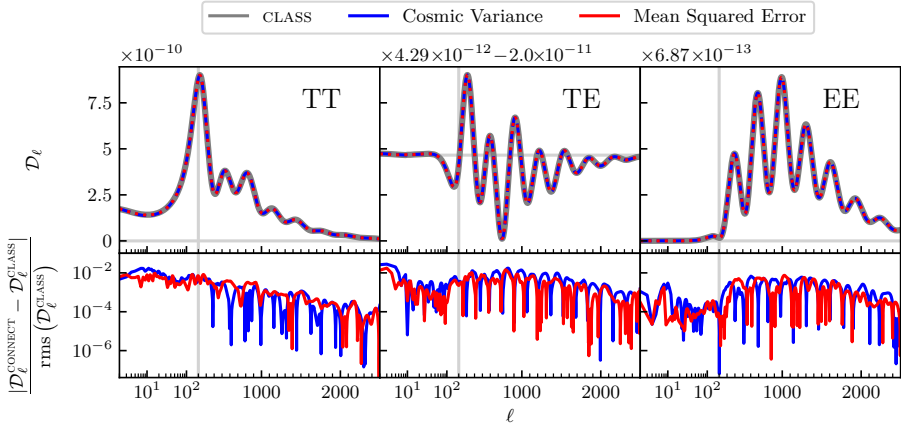


Figure 5.1: CMB spectra as calculated by CLASS along with two neural network models with different loss functions. The difference is not really visible in the spectra, so the errors are included as well. Due to some of the spectra values being close to zero, a relative error would be misleading. The absolute errors are therefore scaled by the rms-values of the spectra.

where \mathbf{x} is the output from the network and \mathbf{y} is the output from the training data. This loss function ensures that the network performs equally well on every output node and is thus the apparent choice if we are to remain agnostic about our network.

There are, however, various situations where this approach is not the most optimal, and the CMB spectra are examples hereof. Measurement errors on the CMB temperature and polarisation power spectra are a combination of cosmic (sample) variance, noise, and finite beam width (see e.g. [129]). Modern CMB probes, such as Planck, in general provide spectra which are cosmic variance limited (except for B -mode polarisation) effectively out to the maximum ℓ -value measurable with the given beam width, and can therefore be reasonably approximated by assuming cosmic variance out to some maximum ℓ beyond which the error goes to infinity. Using this observation as a guide we therefore modify equation (5.1) with ℓ -dependent coefficients,

$$\mathcal{L}_{\text{CV}}(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n \frac{2\ell_i + 1}{2} (x_i - y_i)^2, \quad (5.2)$$

Figure 5.1 shows the CMB spectra of a Λ CDM model as calculated by CLASS and two neural network models with different loss functions, \mathcal{L}_{MSE} and \mathcal{L}_{CV} . The figure also includes the errors between the spectra

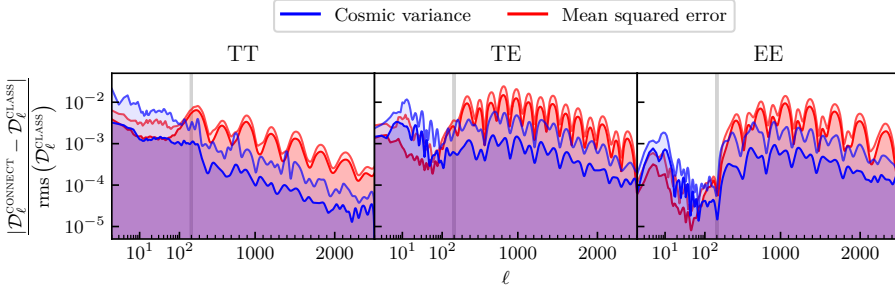


Figure 5.2: Percentiles of errors in the CMB spectra of neural network models with different loss functions when using a test dataset of 20,000 points from a high-temperature MCMC sampling of the Λ CDM posterior. Both the 1σ and 2σ percentiles are included for each model.

from the neural network models and CLASS. The errors are calculated as the absolute difference scaled by the rms-values of the spectra. This is due to the fact that a normal relative error is misleading when the values of the spectra are close to zero, since the error would be very large even though the discrepancy is rather small.

Evaluating the loss functions on a single cosmological model may misrepresent the performances on a larger region of cosmological parameter space. We therefore use the neural network models on a set of data from a high-temperature MCMC sampling containing 20,000 points and calculate the error in the same way. Figure 5.2 shows the 1σ and 2σ percentiles of this set of errors for both loss functions. It is clear from figure 5.2 that the cosmic variance loss function has the desired effect of improving the accuracy at high l s at the cost of accuracy at low l s. This in turn improves results of parameter inference compared to using the mean squared error loss function. When including additional output, such as derived parameters, we need to use a combination of both loss functions, but we also need to assign an importance to all non-CMB output similar to that of the highest l -mode, as to not get a low accuracy on these.

5.2.3 CHOOSING AN ACTIVATION FUNCTION

Another choice we need to make is the choice of an *activation function* [124]. The traditional choice is the *Rectified Linear Unit* (ReLU) function [130]. However, the main drawback of ReLU is that the training might become more difficult due to the derivative being exactly zero

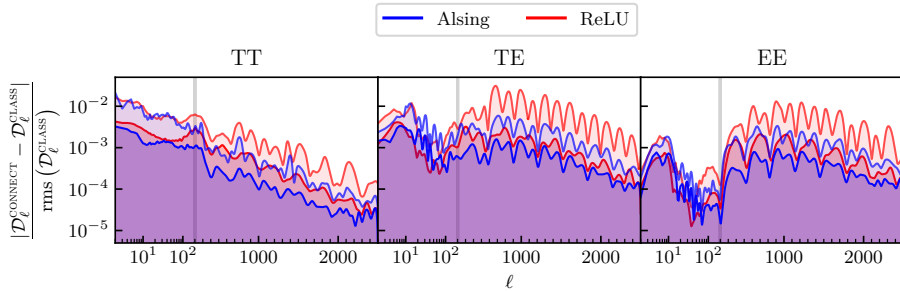


Figure 5.3: Percentiles of errors in the CMB spectra of neural network models with different activation functions when using a test dataset of 20,000 points from a high-temperature MCMC sampling of the Λ CDM posterior. Both the 1σ and 2σ percentiles are included for each model. “Alsing” refers to equation (5.3) as presented in ref. [131].

for negative input. For our application, we found that the following parameterised ReLU with a smoothing between the positive and negative parts, as suggested in ref. [131], works well. There are two free parameters of this activation function, one for the slope of the negative part and one for the smoothing, and we allow these to be trained alongside the weights of the network. We can furthermore assign different parameters for each node in a layer which will then be optimised during training. This leads to the form of the activation function as presented in ref. [131],

$$f(x) = \left(\gamma + \left(1 + e^{-\beta \odot x} \right)^{-1} \odot (1 - \gamma) \right) \odot x, \quad (5.3)$$

where the parameters β and γ control the smoothing and slope of the negative part, respectively, and the \odot represents elementwise multiplication. From figure 5.3, it is evident that this activation function performs better than the simple ReLU activation function.

5.2.4 NORMALISATION OF INPUTS AND OUTPUTS

When using an artificial neural network it is beneficial, and often necessary, to consider scaling of the training data [132]. This is especially true in our case, since the input nodes and output nodes vary with several orders of magnitude. If we were to not consider this at all, the loss function would only have significant contributions from the larger values, while small numbers, such as the C_ℓ s, would have a vanishing

impact on the total loss. We are therefore required to address this problem in some manner. There are several ways to deal with this and they include the following:

1. **Min-Max scaling**, where data belonging to each node in the input (output) layer is transformed to the same interval, e.g. $[0, 1]$, using the minimal and maximal values of the data within the node: $X_{\text{new}} = (X - X_{\min}) / (X_{\max} - X_{\min})$.
2. **Logarithmic scaling**, where the data is transformed to logarithmic space in order for values differing by several orders of magnitude to lie within the same order of magnitude (or few apart). Since $X_1 / X_2 = \exp[\log(X_1) - \log(X_2)]$, this also ensures that optimisation of absolute loss in logarithmic space is equivalent to an actual optimisation of relative loss, meaning that larger orders of magnitude will not be favoured above smaller orders of magnitude.
3. **Standardisation**, where data belonging to each node in the input layer (or the output layer) is transformed to a normal distribution with zero mean ($\mu = 0$) and a variance of unity ($\text{Var} = 1$): $X_{\text{new}} = (X - \mu) / \sqrt{\text{Var}}$.

The input arrays in the training data are automatically normalised with standardisation using TensorFlow’s own built-in `preprocessing.Normalization` routine based on a usual batch normalisation scheme [132]. The means and variances are stored as weights in the input layer of the model and we thus do not need to do anything explicit to the inputs. It is not quite as easy with the output arrays, since no similar routine is available for the output layer. We instead have to normalise the output arrays manually, and we have therefore implemented all of the above three methods in CONNECT.

All three methods of normalisation yield good results when compared to simply multiplying all spectra with a constant factor of 10^{10} , but the accuracy is much better when using min-max scaling or standardisation. This is because all nodes in the output layer have the same span and the network treats them similarly. When using logarithmic scaling, the order of magnitude is still very similar, but there is a clear difference between the C_ℓ s and other kinds of output, such as derived parameters, since the C_ℓ s will have values around or lower than -20 while other parameters will have values closer to or above zero. This leads to a difference in how the output nodes are viewed and treated by the network,

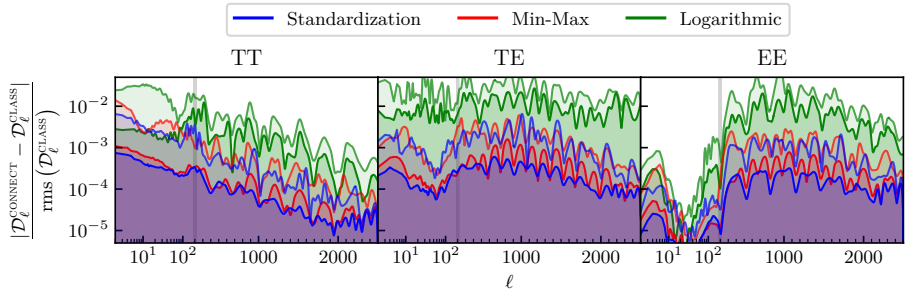


Figure 5.4: Percentiles of errors in the CMB spectra of neural network models with different normalisation methods when using a test dataset of 20,000 points from a high-temperature MCMC sampling of the Λ CDM posterior. Both the 1σ and 2σ percentiles are included for each model.

and it is thus harder to achieve convergence. In our implementation of logarithmic scaling, we found that the performance can be further increased by taking the logarithm twice (after a shift of all the data to positive values) since the difference in orders of magnitude for C_ℓ s and derived parameters is quite large. Standardisation yields a slightly better result than min-max scaling, as apparent from figure 5.4, and so it has been chosen as the default normalisation method. All results in this paper have been produced with standardisation as the normalisation method except for the comparisons between different methods of normalisation in this section.

5.3 SAMPLING OF TRAINING DATA

The training data can be sampled in various ways with different methods having different strengths and weaknesses. The most agnostic way of sampling the parameter space would be using a grid-based method. To get a good resolution these can, however, be very costly and we end up with many points that yield almost identical output since many of them only differ in a single parameter. To circumvent this, we can use *Latin hypercube sampling*, where no two models share any parameter values. This is much more efficient and proves sufficient for the training of the neural network. This way of sampling still yields a uniform distribution of points in the parameter space, so the trained network will be able to emulate the output for all models in the parameter space (within the boundaries of the Latin hypercube) with similar accuracy.

For a large Latin hypercube containing all reasonable models, it is, however, rare that we would ever need to use the network on the outer parts of the hypercube. This is due to the fact that most models near the edges (and especially the corners) have very low likelihoods since they are very far from the best-fit points of most datasets. If we disregard all such unlikely models, a better way of sampling would be by mimicking the shape of the actual posterior distribution. With high dimensionality in the parameter space, this proves much more efficient than using Latin hypercube sampling, since only a small fraction of the models are of actual use in the latter. A way of illustrating this effect is by imagining a simple hyperspherical posterior with radius R centred around the best-fit point. The ratio of the volume of the hypersphere to that of a hypercube with side length $2R$ is given by

$$r_n = \frac{V_n^{\text{sphere}}}{V_n^{\text{cube}}} = \frac{\pi^{n/2}}{2^n \Gamma(\frac{n}{2} + 1)}, \quad (5.4)$$

and in high-dimensional space this decreases rapidly. With just 3 parameters, the corners of the hypercube makes up almost half of the volume, and with 9 parameters, less than a percent of the volume is within the hypersphere. By only focussing on models within such a hypersphere, we could utilise our resources much better and increase the performance of the network on all the relevant models of interest. Actual posteriors typically have a much more complicated shape than a hypersphere, however, the argument still holds due to many of the cosmological parameters having a vanishing likelihood only a few standard deviations away from the best-fit point. We cannot simply expect that a hyperspherical sampling will be representative of the posterior distribution. We thus need a way of sampling training data from the actual posterior distribution instead. We therefore propose to sample the training data using an MCMC method with a high sampling temperature. It seems a little strange to use an MCMC method to create the training data for a neural network that is to be used in an MCMC analysis, but the idea is that we do not need anywhere near as many data points for the training data as we do for the actual MCMC analysis. A high-temperature MCMC sampling running for a few hours is sufficient to obtain the same (or better) accuracy on the relevant models as one would get from Latin hypercube sampling with 10^5 – 10^6 points. As noted in ref. [133], we thus obtain a set of training data from the exact region of the parameter space where emulation is relevant instead

of having the majority of the training data unrealistically far away from the best-fit point.

In this paper, we present results obtained with the default CONNECT temperature of $T = 5$. Since the temperature alters the likelihood \mathcal{L} as $\mathcal{L} \rightarrow \mathcal{L}^{1/T}$, a temperature of $T = 5$ corresponds to increasing the standard deviation of a Gaussian likelihood by a factor 5, to the effect that the generated training data mainly lies inside the 5σ contour of the posterior. However, this is a free input parameter and may be adjusted if the user desires higher accuracy further away from the posterior mode.

5.3.1 ITERATIVE SAMPLING

We can even improve on this and make the sampling even more efficient. We can exploit the fact that we only need to sample from something roughly similar to the posterior distribution and not the actual one, due to the high sampling temperature and the neural networks ability to interpolate in a well-sampled area. We therefore do not need to search the parameter space with the precision of CLASS resulting in many slow calculations of the likelihood. With a typical acceptance rate of 0.3 we are wasting a lot of computation time when calculating the likelihood of rejected steps. A way around this is to use a neural network trained with only a small number of Latin hypercube points ($\sim 10^4$) — enough to give a decent, but not great, accuracy. We then use this neural network model to calculate likelihoods during the MCMC sampling much faster and sample new points for the training data. We then only need to use CLASS to calculate the output-part of this new training data, which means that we effectively skip the CLASS computations of all the rejected steps. Since the neural network model had a relatively low accuracy, we have only gained a rough sample around the posterior distribution, but new models trained with this new training data shows a major improvement. We can even repeat the process starting from this new model, and the resulting models improve for each iteration. Using this method we can thus get rid of a huge part of the expensive CLASS computations only resulting in rejected steps. We could include the initial Latin hypercube data in the total dataset, but by doing so we lower the accuracy of the neural network model in the relevant parts of the parameter space. This is due to the fact that the network contains a limited set of weights, and by forcing it to learn the behaviour in the outermost regions of the parameter space, we inhibit the training in the

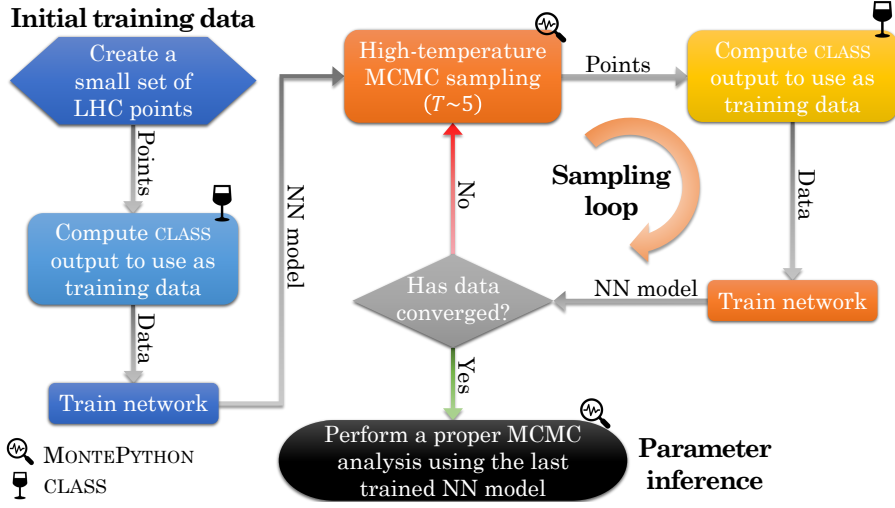


Figure 5.5: Flowchart of the iterative sampling algorithm.

more relevant regions. It is thus advantageous to exclude the initial training-data even though the CLASS computations have already been done. An illustrative flowchart of this sampling algorithm is shown in figure 5.5. During the completion of this paper a few similar approaches have been published [134–136], which further increases the confidence in this type of sampling.

When using this way of sampling, we are interested in the least amount of points possible with the best representation of the posterior for a good accuracy. We are therefore not interested in using all of the points from the high-temperature MCMC samplings, since the burn-in period yields unfavourable points to use as training data. In order to get a representative set of training data, we therefore sample longer MCMC chains and keep only the last N points for the training data. The question now remains how to determine when each high-temperature MCMC sampling should end as well as when the accuracy of an iteration is acceptable. We propose similar answers to the two questions, namely to stop when the variance falls below a certain threshold. For the individual high-temperature MCMC samplings it is the variance between the chains, and for the iterations it is the variance between the kept points from two consecutive iterations.

5.3.2 REDUCTION OF OVER-DENSITIES IN SAMPLE POINTS

We could use all of the kept points from each MCMC run, but we would then get a lot of similar points in our total dataset, since each iteration roughly samples from the same distribution. It would be beneficial to have a way of determining which points we can safely discard, as to not waste computational power increasing our dataset where it is already well sampled. A simple, yet effective, way of doing this is the following:

```

 $P_i$       = new points from current iteration
 $P_{i-1}$     = points in the data set from previous iterations
for  $p$  in  $P_i$  do
     $x$       = nearest point of  $p$  in  $P_{i-1}$ 
     $d_{\min}$   = distance between  $p$  and  $x$ 
     $D_{i-1}$   = array of distances between  $x$  and the
                $n$  nearest neighbours of  $x$  in  $P_{i-1}$ 
     $D_i$      = array of distances between  $p$  and the
                $n$  nearest neighbours of  $p$  in  $P_i$ 
    if  $d_{\min} > \text{average}(D_{i-1}) + 2 \cdot \text{std}(D_{i-1})$  then
         $p$  is accepted
    else if  $\text{average}(D_i) < \text{average}(D_{i-1}) - 2 \cdot \text{std}(D_{i-1})$  then
         $p$  is accepted
    Add all accepted points to the data set

```

The conditions of the if-statements might seem arbitrary at first, but we found that a tolerance of 2 standard deviations gave the most consistent results. There are several reasons why oversampling should be avoided, and computational waste is only one of them. Another reason is that an oversampling of certain regions leads to a bias in the training, since these regions are given more weight in the calculation of the total loss. When repeating the sampling for several iterations, the over-sampled regions may differ between two iterations and thus result in trained neural networks with different biases. These will in turn lead to different samples, and the convergence in the data between two consecutive iterations will be immensely difficult and require a large number of iterations.

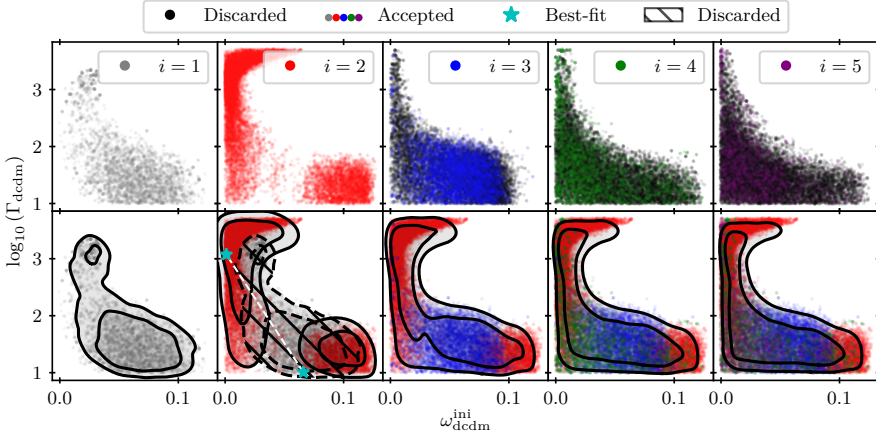


Figure 5.6: Upper panel: Data from each iteration of a sampling of a decaying cold dark matter cosmological model in the $(\omega_{\text{dcdm}}^{\text{ini}}, \log_{10}(\Gamma_{\text{dcdm}}))$ plane. Data points from each iteration are filtered to prevent oversampling of certain regions, and the accepted points are shown in color. Lower panel: Combined data after filtering including 1σ and 2σ contours. Note how the best-fit point of iteration $i = 1$ is several standard deviations away from the subsequent best-fit point. Including the $i = 1$ data in the final training set would degrade the performance of the network as argued in the text.

Next, in figure 5.6 we provide an actual demonstration of how the iterative procedure works for the case of a decaying cold dark matter (DCDM) model. This model is described by a number of cosmological parameters: $\omega_{\text{dcdm}}^{\text{ini}}$ and Γ_{dcdm} (see e.g. ref. [1] for details on the model parameters). The figure shows the training data acquired in each iteration in the 2-dimensional $(\omega_{\text{dcdm}}^{\text{ini}}, \log_{10}(\Gamma_{\text{dcdm}}))$ parameter space, and only the accepted points using the aforementioned algorithm are shown in color.

From the different iterations of figure 5.6 it is clear that we might need to further discard some of the points in our dataset. The points from the first iteration, as sampled by the neural network model trained on the initial Latin hypercube data, often have little to no overlap with those from the other iterations, and including them in the dataset thus leads to a worse accuracy in the relevant part of the parameter space, since the network has to focus some of the training on an irrelevant region. It would therefore be beneficial to remove the data from the first iteration altogether like we removed the initial Latin hypercube data. This way of discarding data is more wasteful than the filtration of points from the MCMC samplings, since we again are throwing away already

computed CLASS models. In order to combat this waste of resources, we decrease the number of points sampled by the initial neural network model. For some cosmological models the first sample is not far from the consecutive samples, and in those cases we could keep the data from the first iteration without lowering the accuracy. We have therefore included the option of keeping the data from the first iteration if one wishes to do so. When looking at figure 5.6, one could make an argument for keeping the first iteration (or, though wasteful, throwing away the second as well), but we discard it to be on the safe side. The first iteration contains 5,000 CLASS computations and the maximal amount of new points from each iteration is 20,000. The final dataset contains 19,999 points from $i = 2$ (one CLASS computation returned an error and was excluded), 7,475 points from $i = 3$, 4,781 points from $i = 4$, and 2,415 points from $i = 5$. We thus see that the amount of points taken from each iteration decreases due to convergence, so less and less CLASS computations need to be performed.

5.4 INTEGRATION WITH MONTEPYTHON

In order to gain any real benefits of a neural network emulating cosmological observables, we need to be able to use the network instead of an Einstein-Boltzmann solver code like CLASS in an MCMC analysis. We have therefore made a module for CONNECT as a plug-in for the popular MCMC code MONTEPYTHON. Using this plug-in along with the *Planck lite* likelihood [128], one can reach speedups of 2-3 orders of magnitude. This means that a reasonable inference can be done in mere minutes.

5.4.1 CONSIDERATIONS AND USAGE

Now that the computation speed of CMB spectra is increased significantly, it no longer dominates the computation time of each step in an MCMC chain. This means that the computation time of the likelihood dominates the computation time when using the CONNECT plug-in, and this means that we are limited to only certain likelihoods if we want the greatest speedups. When using the full *Planck clik* likelihood, we only see speedups of less than one order of magnitude using the CONNECT plug-in, and this is because of the vast number of nuisance parameters making the likelihood computation slow. To really see the benefits of

the plug-in, we need to use the much faster *Planck lite* likelihood which is marginalised over the nuisance parameters leaving only a single one for the likelihood computation. This leads to speedups of several orders of magnitude.

A few things in the source code of MONTEPYTHON are specific to CLASS and for the sake of usability we did not want to alter anything in the source code. The solution was therefore to make our new plug-in inherit from the cython wrapper of CLASS, `classy`, and trick MONTEPYTHON into believing that our CONNECT module is CLASS. This way, a user will not have to alter any code, and any version of MONTEPYTHON supporting CLASS can be used. One simply has to set the path of the cosmological module to that of the CONNECT plug-in instead of a CLASS installation and add the name of a trained CONNECT model to the `data.cosmo_arguments` dictionary in the parameter file.

Since the plug-in inherits from the `classy` wrapper, it will automatically use CLASS to compute any derived parameter that was not emulated by CONNECT. Since the background calculation of CLASS is at the same order of magnitude of computation time as a CONNECT emulation, we can just let CLASS take care of any derived parameter that is only dependent of the background module of CLASS. It is, however, a good idea to include any other derived parameter, needed for the MCMC analysis, in the emulation output, since all other modules of CLASS are slower and this would impact the computation time significantly.

5.4.2 INFERENCE WITH PLANCK LITE

Using the plug-in for MONTEPYTHON, we have first sampled the parameter space of the Λ CDM model which we used to test and validate the training algorithm. In figure 5.7 we show the posteriors resulting from both a standard CLASS-based MCMC run with approximately 5×10^5 accepted chain elements ($\sim 5,000$ CPU core-hours) and a CONNECT-based run with approximately the same number of chain elements (~ 10 CPU core-hours). As can be clearly seen the agreement is excellent! The difference on both means and standard deviations is around 0.01–0.1 standard deviations for all parameters. Figure 5.8 shows the CONNECT-based runs using data from iterations 1, 3 and 5 for a subset of the cosmological parameters ($\omega_{\text{dcdm}}^{\text{ini}}$, Γ_{dcdm} and H_0), and we can clearly see the progress through the iterations. The first iteration does not lead to a particularly good model, but it manages to find a rough area in

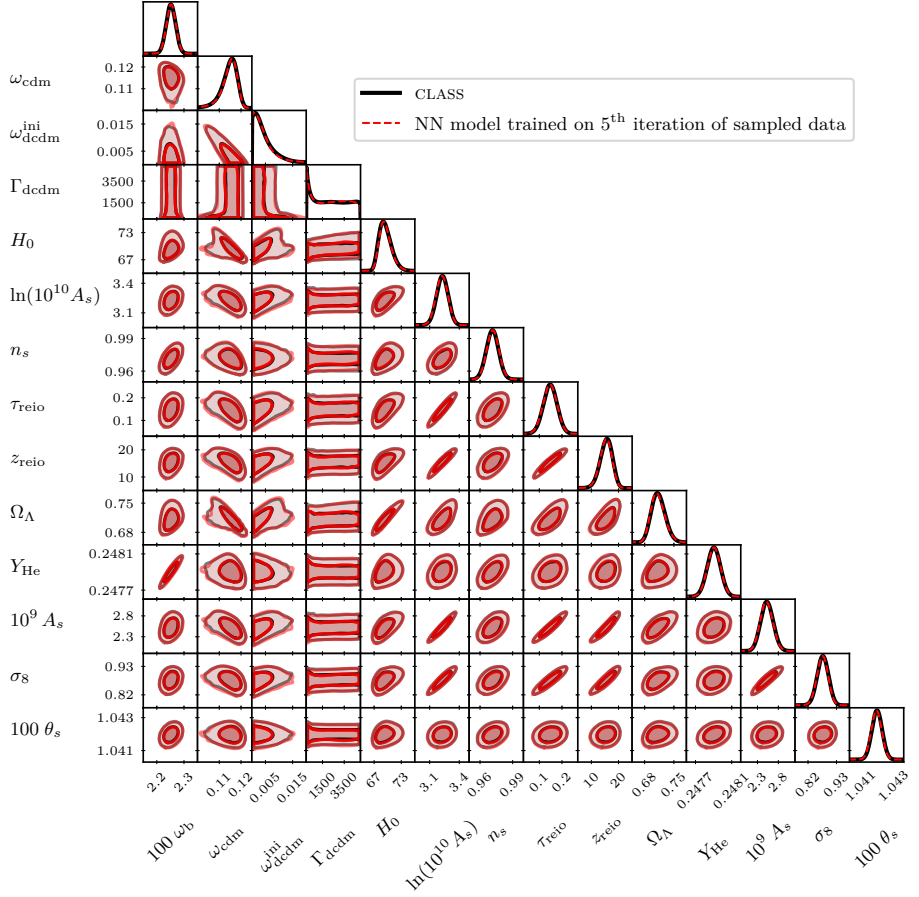


Figure 5.7: 1D and 2D posteriors for the DCDM model resulting from a standard CLASS-based run (black) and CONNECT (red).

which to sample during the following iterations. This improves until the iterations are halted by convergence of the data. When comparing the results of the iterative sampling method to the performance of Latin hypercube sampled NN models in figure 5.9, we see that not even 10^6 points (individual CLASS computations) are enough for the Latin hypercube sampling to match the results of the iterative sampling, and certainly not a Latin hypercube with as few points as in the dataset from the iterative sampling (34,670 CLASS computations). The Latin hypercubes are of course sampled logarithmically in the Γ_{dcdm} parameter as in the iterative case, but this does not help much, the reason being the finite size of the network having to accommodate a huge amount of points in the parameter space that are very far from the region of inter-

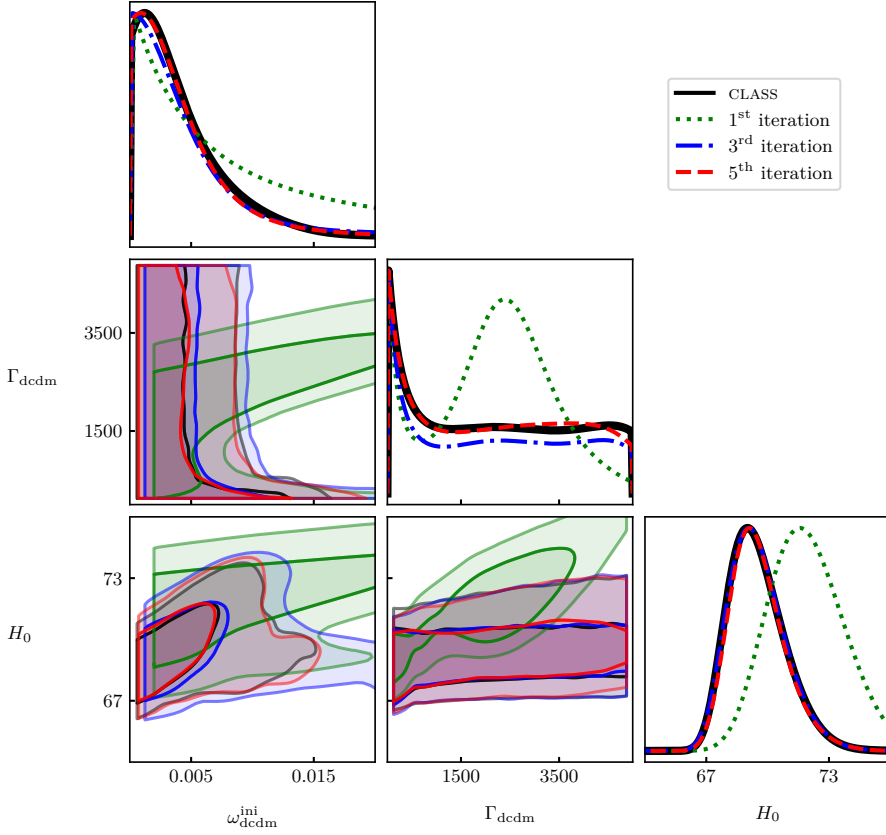


Figure 5.8: 1D and 2D posteriors of the model-specific parameters and H_0 for the DCDM model resulting from a standard CLASS-based run (black) and NN models trained on sampled data from iterations 5, 3, and 1 (red, blue, and green).

est. A network with many more nodes and hidden layers could perhaps be trained to behave nicely in the entire span of the Latin hypercube.

Next, we have used the exact same setup for a completely different massive neutrino model, described by parameters m_{ncdm} and deg_{ncdm} . In figure 5.10 we show the iterative acquisition of training data in the mass-degeneracy plane. As can clearly be seen, this model is significantly easier for the algorithm and converges in just 3 iterations because the likelihood function is significantly more Gaussian than that of the DCDM model. The first iteration again contains 5,000 CLASS computations and the maximal amount of new points from each iteration is now 30,000 (see section 5.5 for a discussion hereof). The final dataset contains 30,000 points from $i = 2$ and 8,959 points from $i = 3$. We

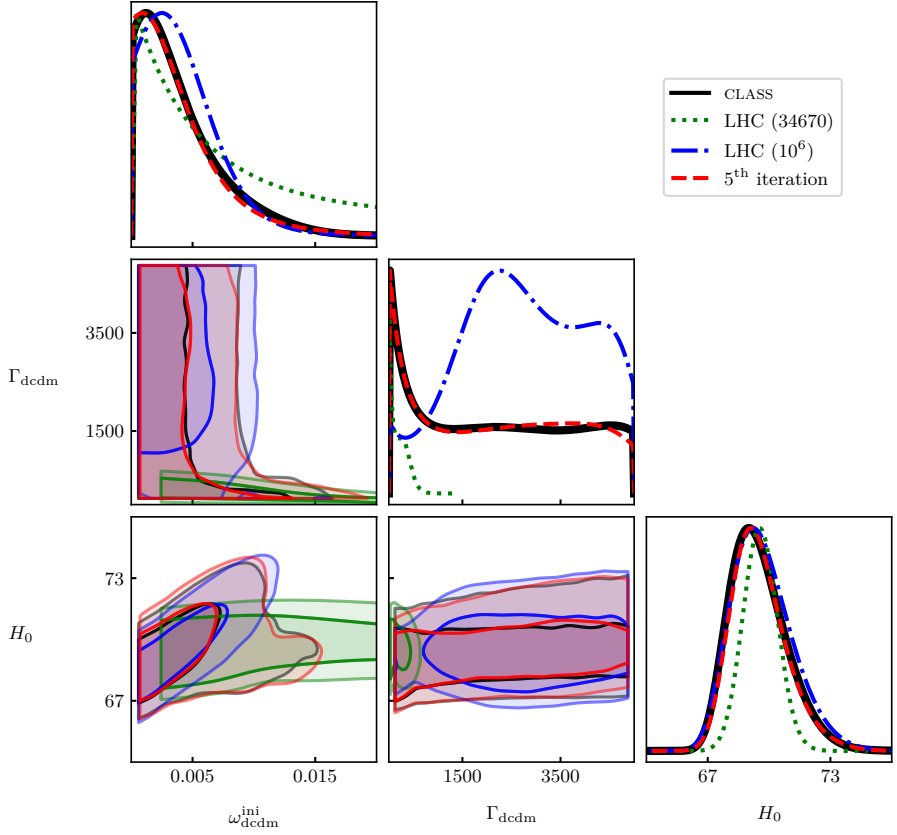


Figure 5.9: 1D and 2D posteriors of the model-specific parameters and H_0 for the DCDM model resulting from a standard CLASS-based run (black), an NN model trained on sampled data from iteration 5 (red), and NN models trained on Latin hypercube data with 10^6 and 34,670 points (blue, green).

then perform the same comparison between MCMC runs with CLASS and CONNECT as in the DCDM case, and the result is shown in figure 5.11. Clearly, in this case the agreement is even better than in the DCDM case, with means and confidence regions differing by around or less than 10^{-2} standard deviations for all parameters except H_0 , which differs by around 10^{-1} .

Figure 5.12 shows the results of MCMC runs based on the data from the three iterations for a subset of the cosmological parameters (m_{nCDM} , deg_{nCDM} and H_0), and we clearly see that convergence is very quickly achieved since the second iteration is very close to the third. This is quite remarkable due to the fact that the first iteration samples no training data in the immediate vicinity of the best-fit point, so the second

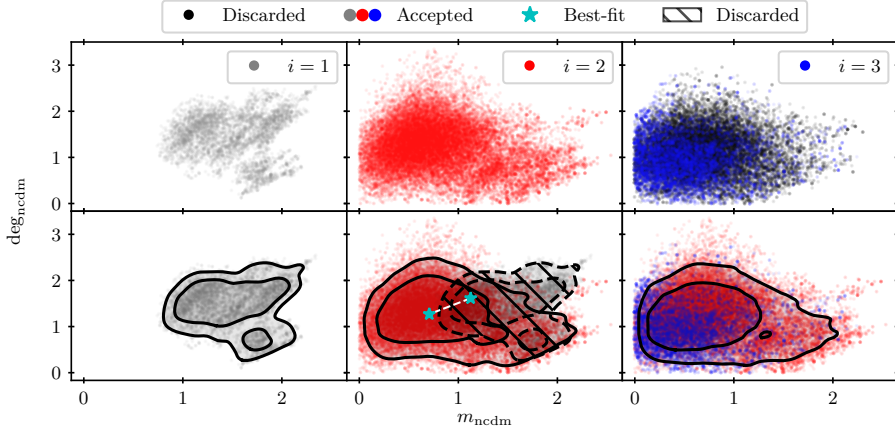


Figure 5.10: Upper panel: Data from each iteration of a sampling of a cosmological model including massive neutrinos marginalised to the mass-degeneracy plane. Data points from each iteration are filtered to prevent oversampling of certain regions, and the accepted points are shown in colour. Lower panel: Combined data after filtering including 1σ and 2σ contours. The first iteration is again removed from the dataset, since a better accuracy is achieved without it.

iteration being that good really demonstrates how well the sampling works when the likelihood function is simpler and more Gaussian. Figure 5.13 shows how the results of MCMC runs using models trained on Latin hypercubes of different sizes stack up against the results from the last iteration. It is apparent that a Latin hypercube of the same size as the dataset from the iterative process (38,959 points) in no way comes even remotely close to the performance of the last iteration. We also see that a number of points larger than 10^6 is needed to even come close to the same result as the iterative process, but a much larger dataset would require a larger network architecture as well to perform reasonable, and this would raise the evaluation time for each model and is thus not a viable solution — not to mention the huge number of CLASS computations that would make the whole idea of emulation obsolete. The larger Latin hypercube seems reasonable for the parameters with a Gaussian posterior, but in the case of a parameter whose likelihood function increases towards a boundary of the prior, we again conclude that the Latin hypercube does not have the ability to represent the cosmological models close to the boundary.

Lastly, we note that models trained by CONNECT are also fairly accurate beyond 2σ . Figure 5.14 shows the contours of posteriors in the

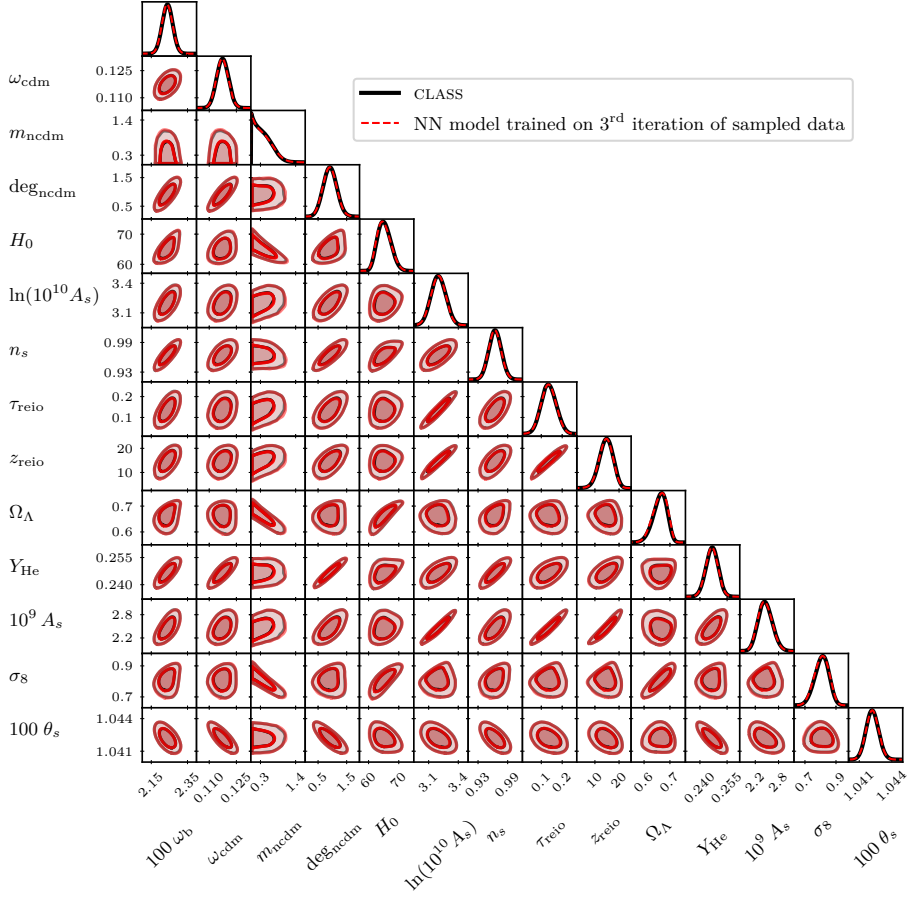


Figure 5.11: 1D and 2D posteriors for the massive neutrino model resulting from a standard CLASS-based run (black) and CONNECT (red).

massive neutrino model as estimated by CONNECT and CLASS, respectively, out to 5σ . Evidently, the agreement is reasonable, even out to 5 standard deviations for the massive neutrino model. This accuracy is a consequence of the increased temperature with which the training data sampling chains are run, and the training temperature can be further tuned manually to accommodate even stricter accuracy requirements on a broader region.

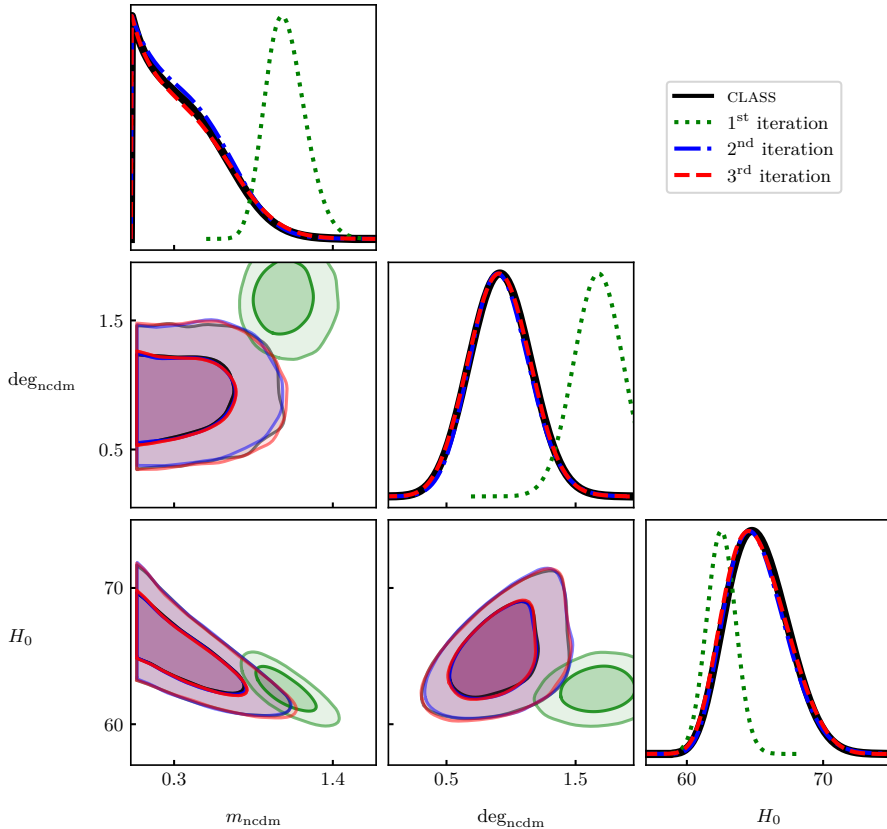


Figure 5.12: 1D and 2D posteriors of the model-specific parameters and H_0 for the massive neutrino model resulting from a standard CLASS-based run (black) and NN models trained on sampled data from iterations 3, 2, and 1 (red, blue, and green).

5.5 DISCUSSION AND CONCLUSIONS

We have presented CONNECT, a novel, neural network based framework for cosmological parameter inference. The method relies on an iterative MCMC-based generation of training data which proves highly efficient in training the network to perform extremely well for MCMC based cosmological parameter inference. We have tested the robustness and versatility of the method by first building the network structure and training algorithm on a decaying cold dark matter (DCDM) cosmological model and demonstrating that we can achieve results almost exactly identical to well-converged CLASS based MCMC runs, and subsequently used CONNECT in an off-the-shelf manner to run

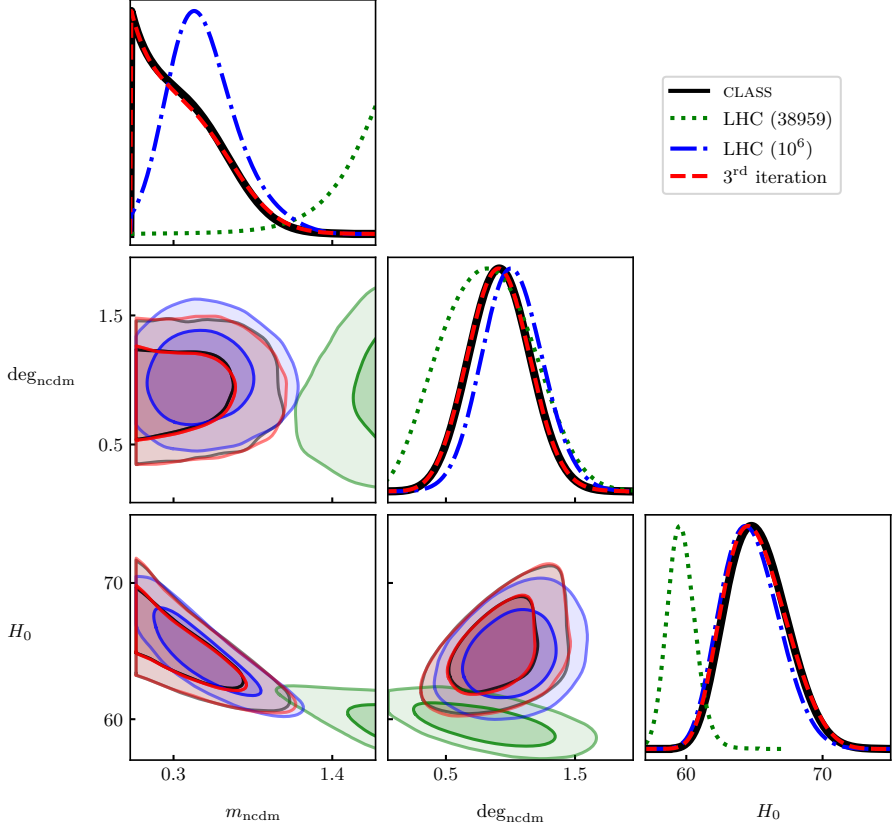


Figure 5.13: 1D and 2D posteriors of the model-specific parameters and H_0 for the massive neutrino model resulting from a standard CLASS-based run (black), an NN model trained on sampled data from iteration 3 (red), and NN models trained on Latin hypercube data with 10^6 and 34,670 points (blue, green).

parameter inference on a *different* massive neutrino cosmology. Even though the method was not previously tested or optimised on this model we find results which are just as impressive as for the Λ CDM model.

Hyperparameters of the iterative sampling. A few things can be customised when using the iterative sampling method, including convergence criteria, the size of the initial Latin hypercube, prior bounds for the individual high-temperature MCMC samplings, the maximum number of points to include from each iteration, and the filtration of new data points. The convergence criterion for the

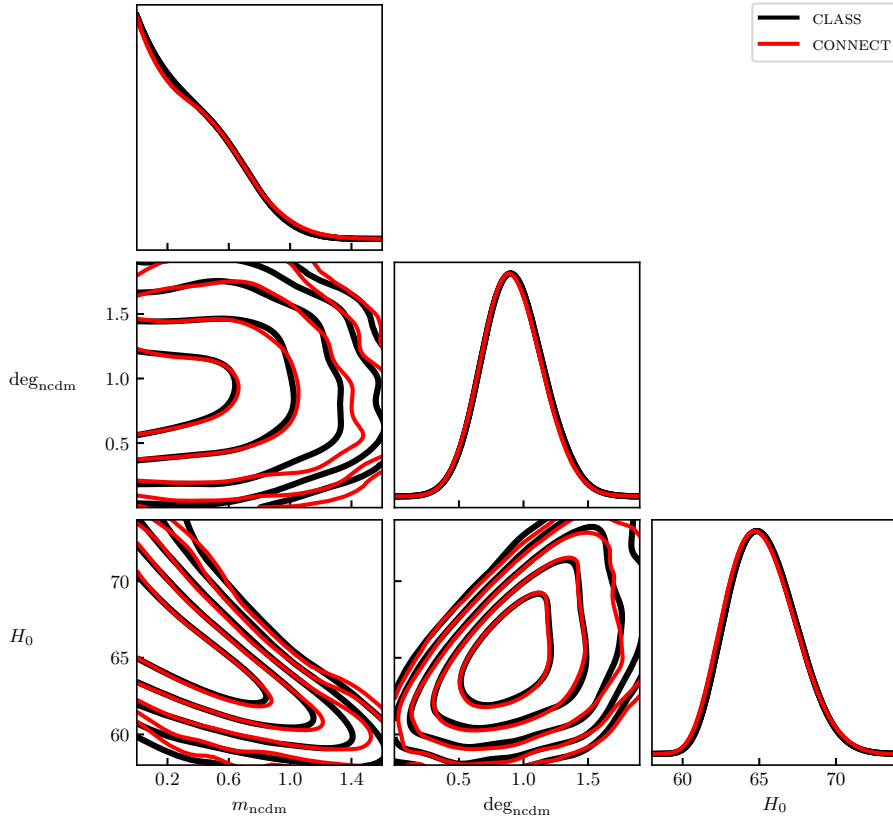


Figure 5.14: Triangle plot of the marginalised posteriors of selected parameters in the massive neutrino model, as computed directly by CLASS and by the CONNECT model. The contours represent the contours from 1σ to 5σ , respectively. There is a fair agreement between CONNECT and CLASS even out to 5σ .

individual MCMC runs is chosen to be when $R - 1 < 0.01$ between the chains is valid for all sampling parameters, and the criterion for halting the iterations is chosen to be when the variance between the data acquired from two consecutive iterations is also at $R - 1 < 0.01$. Depending on the complexity of the likelihood, these criteria could be loosened. When filtrating points from each iteration to avoid oversampling in certain regions, the amount of points actually included, of course, decreases over time, and so another halting criterion for the iterations could be when (almost) no new points are added to the total dataset. These two criteria could even be combined, thus giving larger robustness and a better guarantee of the whole best-fit region being sampled. The maximum amount of points included by each

iteration is the number of points extracted from the high-temperature MCMC samplings before doing any filtration. This number can be varied depending on the difficulty in sampling the likelihood function. A simpler likelihood function allows for very quick convergence when having a larger maximum amount, which is apparent from our results for massive neutrino model, where this parameter was set to 3×10^4 while it was set to 2×10^4 for the Λ CDM results. There can, however, also be a reason to raise this number for more complicated models, since we would then have a better representation from the MCMC chains. This is not ideal for the first few iterations though, since we would waste many more CLASS computations given the low accuracy of the first iterations. The size of the initial Latin hypercube was chosen to 10^4 points for both of our cosmological models, but this could perhaps be brought down due to the fact that the bounds are chosen to be quite large in order for it to encapsulate all of the regions of significant likelihood. If one knows roughly where the best-fit region is, a much smaller Latin hypercube could be sufficient. The priors on the high-temperature MCMC samplings were chosen to be the same as the bounds of the initial Latin hypercube, but if the hypercube is shrunk, the priors should be set differently than the bounds so as to not exclude significant parts of the parameter space. Due to the nature of the iterative method, convergence should be reached even if the initial Latin hypercube has little to no overlap with the best-fit region (it might take more iterations though), but more tests are needed regarding this.

CPU time comparisons. Comparing the CPU intensity of inference run with CONNECT and with standard CLASS based methods is somewhat involved. There is an initial overhead in training the CONNECT emulator for a given model of order 5×10^4 CLASS evaluations depending on the number of iterations and how many points each iteration should contribute. However, once the network has been trained the CPU consumption from the CONNECT based MONTEPYTHON inference comes almost exclusively from the likelihood evaluation when using the full Planck likelihood, not from the emulation. A CPU consumption of similar size as evaluations of the neural network arise when using the Planck lite likelihood instead, thus making the speedup much more profound during the MCMC runs. A single evaluation of the C_ℓ spectra for a cosmological model takes of order 5 s (depending on the cosmo-

logical model — about twice as much for Λ CDM) to evaluate using CLASS on a modern intel CPU core while CONNECT only uses around 3 ms (including interpolation of the C_ℓ s)! This is an immense speedup of three orders of magnitude, and this is also apparent from the time consumption of an MCMC. The MCMC runs using CLASS as cosmological module each took 200 hours with 6 chains and 6 CPU cores for each chain. This enables the CLASS computations to be parallelised which brings down the total time at the cost of using 6 times the CPU cores. When using CONNECT as cosmological module, we again use 6 chains, but only one CPU core per chain is necessary since the evaluation of the network is not parallelisable. These runs, however, take less than two hours to reach the same level of convergence and a similar amount of accepted steps, so the MCMC runs using CONNECT are sped up by a factor of more than 600. This is in great agreement with the difference in the evaluation times when factoring in the time consumption of the Planck lite likelihood evaluations.

We also need to consider the overhead from sampling of training data, which consists of a number of CLASS computations and, in the case of iterative sampling, high-temperature MCMC sampling and training of neural network models. There is not much to do about the CLASS computations except for limiting the number of them and using many CPU cores on a cluster. With the iterative sampling method the CLASS computations are embarrassingly parallelisable, and the only way to limit the number of computations is to optimise the choice of points in the parameter space to compute and not throw any away in the end. Unfortunately we have to throw away the initial Latin hypercube since the inclusion of this worsens the performance of the network, and for complicated likelihood shapes, we often need to discard the first iteration as well, as argued previously. This means that there are around 10^4 CLASS computation that we have performed, but cannot use in the final dataset. The high-temperature MCMC samplings are much less expensive in CPU time, but using a normal Metropolis-Hastings algorithm, the sampling is not very parallelisable, and so the time consumption is significant compared to the CLASS computations utilising hundreds of CPU cores at once. The training of the network is quite fast on a GPU, and it normally takes around 5-8 minutes to train our networks using datasets of $\sim 5 \times 10^4$ points on two GPUs with distributed training, so we can probably not bring this down any further. The computation time of a single iteration in the

sampling takes about an hour at this point with 500 CPU cores and two GPU cores allocated. The CLASS computation utilise all CPUs for around 10-15 minutes, whereafter the training uses both GPUs for 5-8 minutes, and lastly the MCMC sampling uses only a handful of CPU cores for around 20-40 minutes depending on the difficulty of achieving convergence. This means that we only use a fraction of the resources for most of the time, but limiting the amount of CPUs to match the MCMC sampling would increase the time of CLASS computations many times. The biggest speedup in the iterations would thus come from the high-temperature MCMC sampling being modified with a more parallelisable algorithm on either a GPU or the many CPU cores available anyway for the CLASS computations. This could bring the sampling part down to a few minutes or even seconds, leaving only CLASS computations and training as the time consuming parts and thus decreasing the time for each iteration to 20 minutes when using the same resources.

MCMC methods and other sampling strategies. For the results presented in this paper we have used the Metropolis-Hastings algorithm for the samplings of parameter space through integration with the popular code MONTEPYTHON. A benefit of doing this is that we can utilise the entire library of likelihoods contained within MONTEPYTHON. This means that nothing has to be rewritten and our CONNECT plug-in really provides a plug-and-play solution. The standard Metropolis-Hastings algorithm used for the sampling is, however, not parallelisable to more than a handful of chains running simultaneously, and therefore the MCMC is quite time consuming when compared to everything else during an iteration (assuming enough CPU cores for the CLASS computations to be parallelised). In order to speed up the process, we need to consider alternative sampling methods as well as how to utilise different likelihoods in the analysis.

There are many ways of sampling the parameter space of a model and MCMC with Metropolis-Hastings is widely chosen mainly due to the cost of evaluating Einstein-Boltzmann solver codes. With the use of neural networks for emulation, a whole new world of parameter inference beyond MCMC sampling opens. Due to the neural network being a smooth function, gradients are not only easy to access but also very numerically stable. This means that gradient-based methods like Hamiltonian Monte Carlo [137] are now a possibility. It is even possible now to move away from MCMC methods and compute profile likelihoods,

given that optimisation is so much faster and more robust than when using CLASS.

In order for us to really utilise the speedup of the emulation, we need some way around the time consumption of likelihood evaluations, which is especially cumbersome for the full Planck likelihood. An idea used by ref. [113] is to translate the likelihoods into TensorFlow syntax in order for it to run rapidly on a GPU. This way one can perform the sampling of the parameter space on the GPU as well, thus having parameter inference in mere seconds (using a parallelisable sampling method such as the affine invariant sampling algorithm [138]). It requires a lot of work to translate the more heavy likelihoods, so a full library of GPU-compatible likelihoods is probably not realisable in the next few years. A simple solution could also be optimisation of the existing likelihood codes (if possible), since many probably have been written knowing that the CLASS computations will always be much slower and therefore had no reason to be written in the most optimal way. This too requires a significant amount of work, and so this is most likely also not happening in the immediate future. A more easy-to-code solution could in fact be emulation of the likelihood functions themselves. This would allow for both a faster sampling with normal MCMC methods, since the evaluation of the likelihood functions would all be on the level of the CONNECT evaluations, and compatibility with GPUs allowing for more sophisticated gradient-based or parallelisable sampling methods leading to reliable parameter inference in seconds.

Reproducibility. We provide the complete CONNECT framework on GitHub for public use available at https://github.com/AarhusCosmology/connect_public. All the parameter files used for CONNECT are included as well as a brief description of how to use CONNECT on its own and with MONTEPYTHON. Any version of MONTEPYTHON supporting CLASS as cosmological module can be used with CONNECT without any alterations to the source code. We used our own version of CLASS written in C++, and thus named CLASS++, but any version supporting the wrapper functions `lensed_cl()` and `get_current_derived_parameters()` can be used. CLASS++ along with a forked version of MONTEPYTHON are both available from our GitHub organisation page <https://github.com/AarhusCosmology>.

ACKNOWLEDGEMENTS

We acknowledge computing resources from the Centre for Scientific Computing Aarhus (CSCAA). A.N., E.B.H. and T.T. were supported by a research grant (29337) from VILLUM FONDEN. We would like to thank Christian Fidler and Sven Günther for useful discussions regarding sampling methods and oversampling of regions in the parameter space. We also thank the newborn daughter of A.N., Olivia, for her willingness in letting her father finish this paper.

APPENDIX 5.A: STRUCTURE OF THE CODE

In our implementation we have collected all the classes and functions to be used in a **source** folder, all the parameter files in an **input** folder, and all the trained models in a folder called **trained_models**. We have also included a space for all the training data, which is automatically sorted in folders with a specified job-name and subfolders with the amount of data points (iteration number) as the name upon creation of the data when using Latin hypercube sampling (iterative sampling). These data folders are collected in the folder **data**. Within the **source** folder, the **custom_functions.py** file contains classes for adding new activation functions and loss functions, which gives the user an easy way of implementing new custom functions for training networks. Within the **source/architecture** folder, users can in addition easily define entirely new architectures for the network and tailor everything to their needs. In the folder **mcmc_plugin** we have made a wrapper for trained models to mimic **CLASS**. This structure is depicted in figure 5.15.

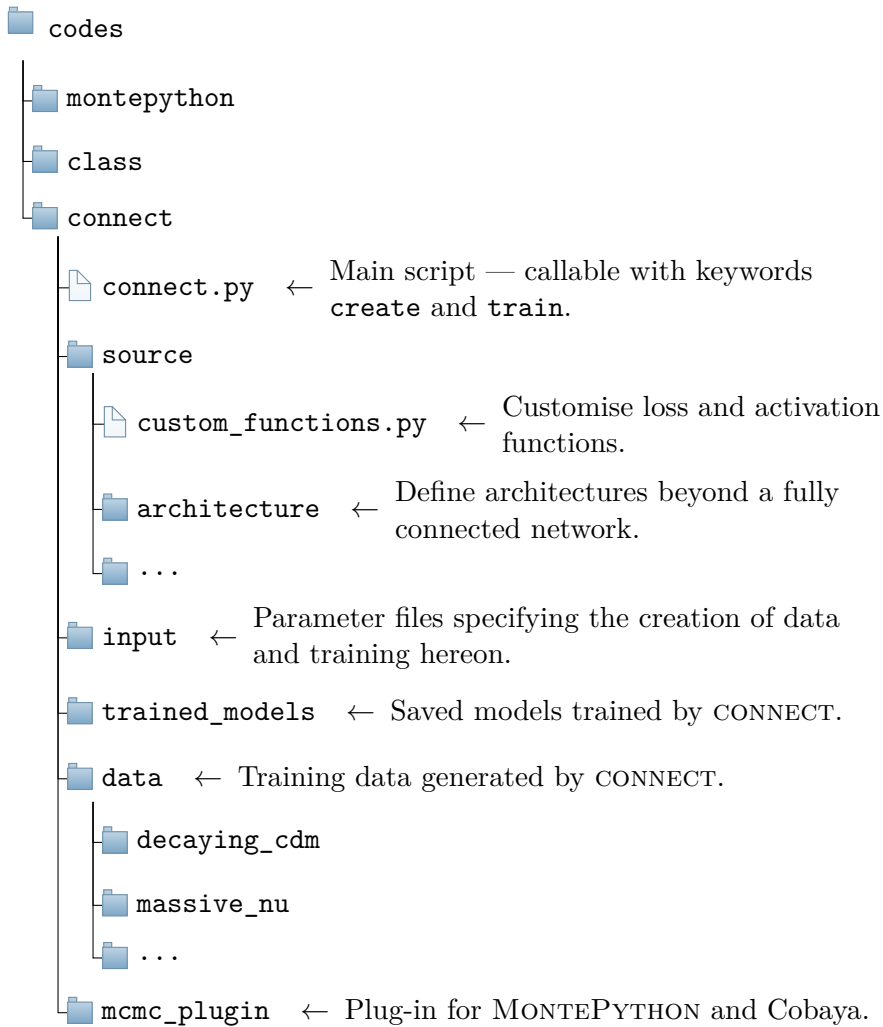


Figure 5.15: The directory structure of a typical installation. `MONTEPYTHON`, `CLASS` and `CONNECT` are located side-by-side in some root folder `codes`.

USING CONNECT FOR PROFILE LIKELIHOODS

An interesting aspect of using machine learning to emulate cosmological observables is the plethora of numerical and statistical methods instantly available that would otherwise be impossible or unfeasible with conventional means. This includes the frequentist method of profile likelihoods for parameter inference, which involves maximising the likelihood function, often requiring many function evaluations. Thus, a fast evaluation time is completely necessary to optimise the likelihood efficiently.

Another benefit of emulation apart from the much faster evaluation time is the access to gradients of the observables with respect to the cosmological parameters. Not only does this enable gradient-based sampling for Bayesian inference, but it also aids in the optimisation utilised when computing profile likelihoods.

This chapter presents the following paper in its entirety:

- *Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. “Fast and effortless computation of profile likelihoods using CONNECT.” In: JCAP 11 (2023), p. 064. doi: 10.1088/1475-7516/2023/11/064. arXiv: 2308.06379 [astro-ph.CO].*

The paper was meant as a proof of concept since only the marginalised Planck lite likelihood was used. This is due to it being the only CMB likelihood easily translatable to the syntax of TensorFlow, which is necessary to compute the gradient of the likelihood with respect to the observables. The paper discusses the problem of global optimisation and introduces a modified version of the *basin-hopping algorithm* which was used in order to utilise gradients.

Although efficient, the optimisation algorithms presented in this chapter have not been used with CONNECT since this paper came out due to the scarcity of differentiable likelihood codes. Subsequent projects

using CONNECT to compute profile likelihoods have used the *simulated annealing algorithm* which is standard for computing profile likelihoods conventionally in cosmology. When used this way, CONNECT also provides a faster and more robust optimisation, and with the development of efficient profile likelihood codes using simulated annealing, e.g., PROSPECT [8], the need for further study of differentiable likelihood codes has not been immediate. I believe, however, that it is important work in the long run, and it will eventually be useful if either differentiable likelihood codes are written, likelihood codes are emulated, or the entire computation from cosmological parameters to likelihood values is emulated instead of observables.

Beginning of reference [3]

FAST AND EFFORTLESS COMPUTATION OF PROFILE LIKELIHOODS USING CONNECT

Andreas Nygaard^a, Emil Brinch Holm^a, Steen Hannestad^a, Thomas Tram^a

^a*Department of Physics and Astronomy, Aarhus University, DK-8000 Aarhus C, Denmark*

Abstract. The frequentist method of profile likelihoods has recently received renewed attention in the field of cosmology. This is because the results of inferences based on the latter may differ from those of Bayesian inferences, either because of prior choices or because of non-Gaussianity in the likelihood function. Consequently, both methods are required for a fully nuanced analysis. However, in the last decades, cosmological parameter estimation has largely been dominated by Bayesian statistics due to the numerical complexity of constructing profile likelihoods, arising mainly from the need for a large number of gradient-free optimisations of the likelihood function.

In this paper, we show how to accommodate the computational requirements of profile likelihoods using the publicly available neural network framework CONNECT together with a novel modification of the gradient-based *basin-hopping* optimisation algorithm. Apart from the reduced evaluation time of the likelihood due to the neural network, we

also achieve an additional speed-up of 1–2 orders of magnitude compared to profile likelihoods computed with the gradient-free method of *simulated annealing*, with excellent agreement between the two. This allows for the production of typical triangle plots normally associated with Bayesian marginalisation within cosmology (and previously unachievable using likelihood maximisation because of the prohibitive computational cost). We have tested the setup on three cosmological models: the Λ CDM model, an extension with varying neutrino mass, and finally a decaying cold dark matter model. Given the default precision settings in CONNECT, we achieve a high precision in χ^2 with a difference to the results obtained by CLASS of $\Delta\chi^2 \approx 0.2$ (and, importantly, without any bias in inferred parameter values) – easily good enough for profile likelihood analyses.

6.1 INTRODUCTION

In the last few decades, parameter inference in cosmology has traditionally been done using Bayesian statistics. In Bayesian parameter inference, the goal is to characterise the multidimensional posterior probability distribution. This is often done using Markov-chain Monte Carlo sampling. Subsequently, estimates and uncertainties on single parameters are obtained by integrating the multidimensional posterior distribution over all other parameters, a process called marginalisation [139]. This is the main reason for the popularity of Bayesian parameter inference in cosmology: all credible intervals and two-dimensional posterior distributions are readily computable once a representative sample of the multidimensional posterior has been obtained.

Marginalisation requires a way to associate (prior) probability to volumes of parameter space, so the marginalised posterior distributions and the credible intervals [140] may sometimes be completely dominated by the choice of prior. This effect is sometimes referred to as volume effects, because the effect is associated with the prior volume being integrated. This can be seen as an advantage because it makes it easy to build e.g. theoretical prejudice into the statistical analysis. For instance, one may wish to penalise a model containing a parameter that needs to be extremely fine-tuned to provide a good fit to the data. However, given that we often do not know the true underlying model, it could very well be that the true underlying model is observationally equivalent to the one proposed, but in the true model, the equivalent

parameter is not fine-tuned. Thus, if we penalise the proposed model a priori, we might fail to discover the true model.

If we want to avoid this trap, we may instead employ frequentist statistics. Broadly speaking, this simply entails that we substitute marginalisation for maximisation; instead of integrating out the extra parameters, we maximise the likelihood over these parameters. The resulting object is the *profile likelihood*, which has recently gained increased popularity [7, 141–148]. From this profile likelihood, we may then derive *confidence intervals* [149], akin to the credible intervals in Bayesian statistics. The advantage of the profile likelihood is that it may reveal interesting features of the model that would be missed in the Bayesian approach, but the disadvantage is the difficulty and computational cost associated with the large number of multidimensional optimisations. Early papers on cosmological parameter inference have examples of both marginalisation (see e.g. [150]) and profiling (see e.g. [151–153]; see also [154] for an early discussion of profiling versus marginalisation). However, the introduction of MCMC techniques in marginalisation [155, 156] led to their increasing dominance in the field because of their speed and simplicity.

Many of the computational problems of profile likelihoods are solved by the recent advancements of machine learning within the field of cosmology. Many different cosmological emulators of observables, using e.g. neural networks [2, 109, 113, 134, 157] or Gaussian processes [121, 122, 158], have emerged in recent years, and these all benefit from much faster evaluation times than ordinary Einstein–Boltzmann solver codes.

In this paper, we show how CONNECT [2] can facilitate fast and accurate computation of one- and two-dimensional profile likelihoods at a tiny fraction of the cost of a more traditional approach. This performance enhancement derives both from the speed-up of evaluating the neural network instead of CLASS [27] or CAMB [159] but also because the gradient of the likelihood can be computed fast and accurately through automatic differentiation techniques. The paper is organised as follows: In section 6.1, we give an introduction; in section 6.2, we introduce profile likelihoods and discuss the difference between profile likelihoods and marginalised posteriors; in section 6.3, we give a brief overview of the optimisation algorithms we use; and in section 6.4, we provide some more practical details regarding the implementation. In section 6.5, we show, for the first time, triangle plots for the profile likelihood compared to the posterior distribution for the Λ CDM-model as well as for

an extension with varying neutrino mass and degeneracy. We also show a profile likelihood of the decay constant in a decaying cold dark matter model as an example where the training data of the neural network itself suffers from large volume effects. Finally, we give our conclusions and future outlook in section 6.6.

6.2 PROFILE LIKELIHOODS AND BAYESIAN INFERENCE

Given an N -dimensional parameter space Θ where we are mainly interested in constraining the parameters of an M -dimensional subset Ω , the profile likelihood $L(\boldsymbol{\theta})$ of a likelihood function $L(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})$ on the full parameter space is obtained by maximising the dimensions not contained in the reduced space Ω [160],

$$L(\boldsymbol{\theta}) = \max_{\tilde{\boldsymbol{\theta}}} \left(L(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \right), \quad \boldsymbol{\theta} \in \Omega, \tilde{\boldsymbol{\theta}} \in \Theta \setminus \Omega, \quad (6.1)$$

where $\boldsymbol{\theta}$ represents a vector in the parameter subspace of interest and $\tilde{\boldsymbol{\theta}}$ a parameter vector in the subspace of Θ we maximise over. Parameters in the latter subspace are commonly referred to as nuisance parameters, and usually we are interested in either one-dimensional profile likelihoods, $M = 1$, for one-parameter constraints, or two-dimensional profile likelihoods, $M = 2$, to constrain the correlation between pairs of parameters. Since the profile likelihood (6.1) is obtained by maximising a subset of the parameter space, frequentist inference based on it is an optimisation problem (as opposed to Bayesian inference, which is a sampling problem). The main contributions of this paper are novel strategies for carrying out this optimisation.

From the profile likelihood, confidence intervals (or *regions*, if $M > 1$) can be obtained using the Neyman construction [161], where 68.27% (95.45%) confidence regions for $\boldsymbol{\theta}$ are defined implicitly by the regions $\Delta\chi^2(\boldsymbol{\theta}) < 1.0$ (4.0) with the definition

$$\Delta\chi^2(\boldsymbol{\theta}) \equiv -2 \log \left(\frac{L(\boldsymbol{\theta})}{\max_{\tilde{\boldsymbol{\theta}}} (L(\boldsymbol{\theta}))} \right). \quad (6.2)$$

The coverage of the intervals constructed using the Neyman method is exact whenever the profile likelihood is Gaussian or whenever there

exists a reparametrisation in which it is. Alternative interval constructions, such as the Feldman-Cousins prescription [162], are known to produce more accurate interval coverages, but since the focus of this paper is on the optimisation and not the interval construction, we employ the Neyman construction throughout.

As seen in the definition (6.1), the profile likelihood is a maximum likelihood estimate in the reduced subspace and therefore inherits the invariance under reparametrisations of this subspace from the latter [160]. This is in contrast to posterior distributions from the Bayesian inferences, which may change with the specific parametrisation. To illustrate the difference between the marginalised posterior distribution and the profile likelihood, we investigate a toy likelihood comprised of two Gaussian distributions with different normalisations and covariance matrices. The two Gaussians have only a slight overlap, as shown in figure 6.1, and the one with the largest maximum is much more narrow in the θ_2 parameter. This makes the contribution of the taller peak to the posterior in the θ_1 parameter much less when marginalising over the θ_2 parameter in the case of uniform priors, even though the likelihood is actually larger in this peak. This shows how the greater volume of a less significant likelihood peak can dominate the marginalised posterior. When computing a profile likelihood in the θ_1 parameter, the likelihood is optimised for each fixed value of θ_1 , and so the profile looks like a projection of the two-dimensional surface onto the one-dimensional θ_1 -axis. This, of course, shows the two peaks with their actual height differences.

The two different ways of representing the likelihood function come from the two different ways of interpreting probability in frequentist and Bayesian statistics. Neither method is more correct; they simply answer different questions. Frequentist inference answers the question of how we can choose a range in the parameter space based on the data such that the true value of the parameter will fall within the range in $(1 - \alpha) \times 100\%$ of the asymptotically often repeated realisations of the experiment. This range is called the confidence interval with the level of significance α . Bayesian inference treats the true value of the parameter as a random parameter chosen from a distribution, and the question is which range of the parameter space we can choose so that we are $(1 - \alpha) \times 100\%$ certain that the true value has taken a value that lies within the range. This is called the credible interval with the level of significance α . The posterior can therefore be thought of as a probability distribution, while the profile likelihood cannot. Ultimately, it is this

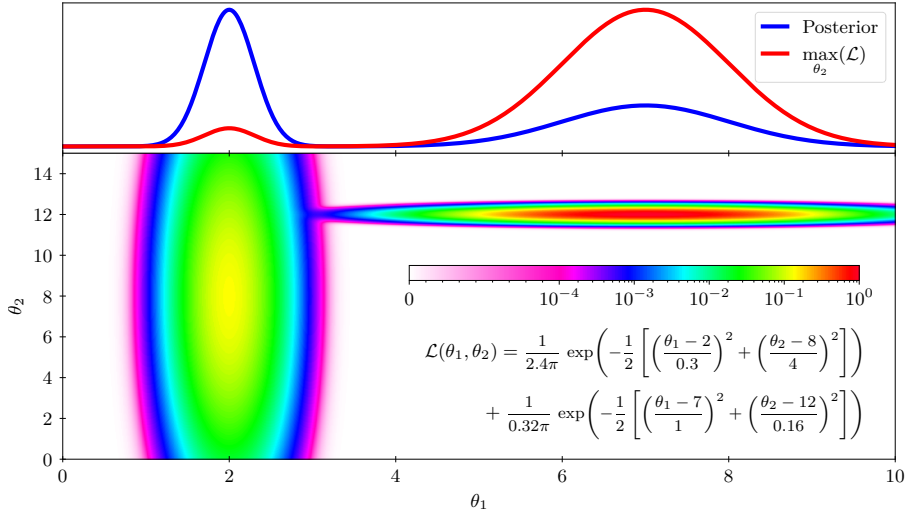


Figure 6.1: The bottom panel shows the function value of the likelihood function written in the panel, while the top panel shows the resulting marginalised posterior and the profile likelihood, both scaled to a maximum of unity. The posterior is dominated by the shorter large Gaussian, while the profile is dominated by the taller small Gaussian. The two one-dimensional statistics thus reveal complementary information about the actual likelihood.

difference in interpretation that manifests in the different constraints obtained from the two statistical paradigms.

6.3 OPTIMISATION OF THE LIKELIHOOD FUNCTION

An accurate and robust optimisation routine is crucial for profiling likelihoods, and this task can be difficult in certain situations. Optimisation routines typically require many function evaluations in order to perform well, and this is especially true if there is no prior knowledge of the cost function. In the case of using an Einstein–Boltzmann solver code like CLASS, the function takes on the order of 10 core seconds to evaluate (including both the Einstein–Boltzmann solver code and the likelihood codes), and this quickly adds up to large computation times. Certain optimisation methods make use of gradients, but Einstein–Boltzmann solver codes tend to be numerically unstable with respect to differentiation due to different approximation schemes toggling in different regions of parameter space. Furthermore, in order to construct profile

likelihoods, it is essential that the global optimum be found. Therefore, global optimisation routines are essential.

6.3.1 GLOBAL OPTIMISATION

The problem of global optimisation is numerically difficult. Methods like *gradient descent* [163] and the *simplex algorithm* [164] can get stuck in local optima, so this can be a problem whether or not one has access to gradients of the likelihood function. There exist, however, specific methods that can effectively search the parameter space in clever ways inspired by MCMC methods. One such method that has proven to be fairly efficient is *simulated annealing* [165], which does not require gradients and is therefore a suitable choice when dealing with Einstein–Boltzmann solver codes. In short, this method searches the parameter space as any other MCMC but gradually lowers the sampling temperature while doing so. This makes the features of the likelihood landscape more profound over time, allowing the MCMC chain to eventually settle on the global optimum. The optimisation is highly dependent on the chosen temperature schedule and somewhat inefficient when the acceptance rate of the MCMC is small. Despite the individual simulated annealing optimisation being sequential, each point in a profile likelihood can be optimised in parallel, so one can therefore do profile likelihoods based on simulated annealing in reasonable time, although the number of evaluations needed for an inference of all parameters in a typical cosmological model, as well as their correlations, makes it unfeasible to do with an Einstein–Boltzmann solver code.

Instead, we use a neural network from the CONNECT framework to speed up the evaluation of the observables. This means that the likelihood evaluation time is dominated by the comparison of theoretical observables to data. This is, however, a significant decrease in computational cost, and using the same method as before (simulated annealing), we can obtain profile likelihoods much quicker.

We can even improve on the method now that we are using a neural network that does not suffer the same numerical instabilities with respect to differentiation as Einstein–Boltzmann solver codes do. The auto-differentiation of the TensorFlow [125] framework lets us easily differentiate the neural network with respect to the input parameters, so we can now make use of gradient-based optimisation techniques. However, before that, we need to be able to auto-differentiate the likelihood

calculation as well. We are seeking the derivative of the likelihood value L with respect to the cosmological input parameters θ , and the network only provides us with the derivative of the observables \mathcal{O} (C_ℓ spectra, power spectra, etc.) with respect to the cosmological parameters. By the chain rule, we therefore need the derivative of the likelihood calculation with respect to its input (the cosmological observables computed by the network),

$$\frac{dL}{d\theta} = \frac{dL}{d\mathcal{O}} \frac{d\mathcal{O}}{d\theta}. \quad (6.3)$$

Obtaining the derivative of the likelihood with respect to cosmological observables proves to be quite elaborate, but this is discussed further in section 6.4.

Equipped with auto-differentiation all the way from cosmological input parameters to the likelihood value, one can now make use of gradient-based methods. Relying solely on these can be a problem since gradient-based methods are often unable to explore other optima than the closest. A global optimisation method that can circumvent this issue is the *basin-hopping* [166] algorithm. This method is reminiscent of simulated annealing, but with the addition of a local optimisation in each step. The method therefore “hops” between different local optima, or “basins”, instead of hopping between random points. This reaches convergence much quicker since fewer steps are needed due to the local optimiser always placing each point at an optimum. This method is very dependent on a good local optimisation method; otherwise, it will reduce to simulated annealing in the limit of a trivial optimiser. One drawback of this optimisation method is that one is not guaranteed to find the global optimum given the stochasticity of the optimiser [167], and indeed there is a probability of convergence at a suboptimal point (see appendix 6.B). In our case, however, this is a quite small probability due to the smoothness of the neural networks, and it can be heavily decreased through precision settings.

6.3.2 LOCAL OPTIMISATION

A local optimisation method should, in our case, take only a few steps in order to reach the optimum. Gradient-based methods are obviously suitable for this, but the simplest of such methods, gradient descent, does not perform well in this regard. The reason for this is that the

step size will decrease with the slope, and a lot of steps will be taken close to the optimum due to the vanishing gradient. Some parts of the likelihood function can even be close to flat, and this requires a lot of steps by the gradient descent optimiser. A better choice could be to also use second-order derivatives, which would mean that we could almost guess the correct optimum after one evaluation if the likelihood landscape is close to Gaussian. The second-order derivatives are, however, not easy to get in our case since the TensorFlow graph of this computation becomes too large to handle. An appropriate compromise could be a pseudo-second-order method like the *Levenberg–Marquardt* [168, 169] or *Broyden–Fletcher–Goldfarb–Shanno* [170–173] (BFGS) algorithms. These use first-order derivatives to approximate the Hessian in order to quickly locate the nearest optimum. TensorFlow has an implementation of BFGS, so this has been chosen as the local optimiser to use with the global basin-hopping method.

6.4 IMPLEMENTATION

6.4.1 AUTO-DIFFERENTIABILITY

In order for us to use gradient-based methods, we need auto-differentiation at each step in the evaluation of the likelihood function. We use the built-in *reverse mode differentiation* of TensorFlow, and this requires every operation during the evaluation to be written with TensorFlow syntax in order for them to be auto-differentiable. Many popular likelihood codes, such as the full Planck 2018 likelihood [127] (including low- ℓ TT , low- ℓ EE , high- ℓ $TT + TE + EE$, and lensing), are complex and tedious to translate to TensorFlow syntax, so as of now only the Planck lite likelihood has been translated, first to PYTHON by Ref. [128] and then to TensorFlow by Ref. [113]. We have altered the TensorFlow version from Ref. [113] to accommodate neural networks from CONNECT and this involved another rather tedious task of interpolation. CONNECT only computes the same ℓ -grid that CLASS does, and this is more than an order of magnitude fewer points than required by the likelihood code. We have therefore implemented a cubic spline interpolation routine in the likelihood code since no suitable interpolation method is implemented in TensorFlow. This has to be written not only in TensorFlow syntax but also using only differentiable

operations (some functionality in TensorFlow is not auto-differentiable). This adds an extra layer of computation to equation (6.3),

$$\frac{dL}{d\theta} = \underbrace{\frac{dL}{d\mathbf{C}_\ell^{[2508]}}}_{\text{Likelihood code}} \underbrace{\frac{d\mathbf{C}_\ell^{[2508]}}{d\mathbf{C}_\ell^{[100]}}}_{\text{Interpolation}} \underbrace{\frac{d\mathbf{C}_\ell^{[100]}}{d\theta}}_{\text{Neural network}}, \quad (6.4)$$

where \mathbf{C}_ℓ is a vector of C_ℓ values, which is the observable used by the likelihood code in our case, and the number in square brackets tells the number of C_ℓ values. One could simply emulate all 2508 values needed by the code, but this is a computational waste when training the network since all the information is contained in only the 100 values that CLASS actually calculates [2] (the number of points calculated by CLASS differs based on input and precision settings). Since each of the computational steps in equation (6.4) is now differentiable, the total derivative of the likelihood with respect to the cosmological parameters can be used for optimisation purposes. A single function evaluation of the gradients takes on the order of $\sim 10^{-2}$ seconds and is approximately twice as expensive as evaluating just the likelihood itself (using the neural network, the interpolation, and the data comparison).

6.4.2 ENSEMBLE BASIN-HOPPING

When training a neural network with CONNECT, the training data is gathered using multiple MCMC runs in an iterative manner [2], where each iteration uses an MCMC sampler to gather data using the neural network from the previous iteration. This means that we already have a covariance matrix and an estimate for the best-fit point as a starting point. This is a great help when doing the actual maximum likelihood optimisation, since the likelihood landscape is roughly known beforehand. We can use this information to slightly modify the standard basin-hopping algorithm to accommodate this additional information. Steps are normally taken sequentially, but since we have a covariance matrix and a best fit estimate, we can construct a proposal distribution and draw multiple points from it at once. These can then all be locally optimised in parallel, thus exploring different local optima simultaneously. We can then centre a new proposal distribution around the best point of the ensemble of optimised points and lower

the sampling temperature. Repeating this allows us to home in on the global optimum much faster due to the parallelisability. The altered algorithm is sketched below¹:

```

Cov( $\theta$ ) = covariance matrix from MCMC
 $\mathbf{b}$       = estimate of best-fit point
d        = dimension of parameter space
opt( $\theta$ ) = local optimiser (returns value and position)
p( $\theta$ )    = MultivariateGaussian( $\mathbf{b}, \textit{Cov}(\theta)$ ) (proposal dist.)
T        = 1.0 (sampling temperature)
Tmin    = minimal temperature (works as a tolerance)
N        = number of points in ensemble
while T > Tmin do
    Larray = zeros(N)
    Parray = zeros(N, d)
    for i in range(N) do (parallelisable)
         $\mathbf{X}$  = point drawn from p( $\theta$ )
        Larray[i], Parray[i] = opt( $\mathbf{X}$ )
    Lmin = min(Larray) (new best value)
    imin = argmin(Larray)
     $\mathbf{b}$  = Parray[imin] (new best-fit point)
    T = T/2 (reduce the temperature)
    p( $\theta$ ) = MultivariateGaussian( $\mathbf{b}, \textit{Cov}(\theta) \cdot T$ )
    if the majority of the ensemble finds the same optimum then
        ⟨break from while loop⟩
Lmin is now the optimised function value and  $\mathbf{b}$  is the best-fit point

```

6.4.3 CONSTRAINTS ON PARAMETER SPACE

Profile likelihoods are not subject to any priors as marginalised posteriors are, but there might still be benefits to confining the optimisation within certain ranges in the parameter space due to physical constraints. An example here could be the mass of a particle, which must be positive. When computing profile likelihoods using Einstein–Boltzmann solver codes and simulated annealing, this will never be an issue since the code will raise a computation error in such a case, and the like-

¹ A similar approach can be found in Ref. [174], where multiple individual workers perform independent basin-hopping optimisations but exchange information about optimal starting points through a master process.

likelihood code will then return a very low likelihood such that no point beyond this physical boundary will be accepted. Neural networks will, however, always produce output based on any input of the correct type and dimensionality, so they might learn some behaviour linked to a certain parameter as it decreases. It is then able to extrapolate, and the result might be a good fit to the data either by chance or because of this extrapolation. This is exactly the case for the model with a varying neutrino mass used as a test case in this paper. It is therefore beneficial to confine our optimisation within given parameter ranges. A variant of the BFGS algorithm dubbed BFGS-B [175] performs the optimisation within a confined box in the parameter space. This is very useful in our case, but unfortunately it is not implemented in TensorFlow. A solution is therefore to introduce a smooth and differentiable penalty function that penalises the likelihood when evaluating a point outside of the box. We have chosen a very steep exponential function that increases depending on the distance from the boundary, and this ensures that gradients are still meaningful in a way such that any evaluation outside the box will send the optimiser back inside the box. A proper implementation of BFGS-B might be beneficial in the future.

6.4.4 WORKFLOW

When doing profile likelihoods with `CONNECT`, there are a few steps. First of all, one needs to gather training data and train a neural network with a specific cosmological model implemented in `CLASS`. Then, it is a good idea to run a normal MCMC with the neural network in order to have a good covariance matrix along with Bayesian inference for comparison. The covariance matrix from the gathered training data can be used instead². Then one needs to choose at which points in the parameter space to optimise for both one- and two-dimensional profile likelihoods; the idea is to sample with more resolution where the likelihood function varies significantly as well as near its maximum. This can be tricky if one does not know any features of the likelihood function of the particular cosmological model, and this is another reason for doing an MCMC run with the neural network beforehand. When the posteriors are known, a good initial guess is that the profile likelihood will somewhat resemble them. This is exactly true when there

² The choice of covariance matrix does not significantly impact the result as long as the initial proposal distribution is wide enough to encapsulate the best-fit region.

are no volume effects and only flat priors are used, but in any case, it is reasonable to assume that at least some features of the profile likelihood will overlap with the features of the posterior. In the case of one-dimensional profile likelihoods, one can often get away with simply choosing a set of equally spaced points. In the case of two-dimensional profile likelihoods, it is not as straightforward, but a good guess is to use the points of the histograms from which the posteriors are computed. Any non-zero bin in these histograms corresponds to a region where the MCMC has accepted points, and so we can choose these bin centres as the points in our two-dimensional profiles (see appendix 6.A). If the computed profiles seem to not be best represented by these points, a routine to manually add points by clicking in the plots has been included (see appendix 6.B). A full automatisation of which points to choose is beyond the scope of this work but should be further investigated.

6.5 RESULTS AND DISCUSSION

In order to test the performance of the optimisation routine, we have chosen three cosmological models as examples: Λ CDM, massive neutrinos, and decaying cold dark matter. These three models each have features that are useful for highlighting different problems and their corresponding solutions.

6.5.1 Λ CDM

The posterior distribution of the standard Λ CDM model is almost perfectly Gaussian under standard CMB data [127] and therefore has no volume effects. In this case, we expect to see the profile likelihoods coinciding perfectly with the posteriors in both the one- and two-dimensional plots. We did not train a specific Λ CDM neural network with CONNECT since this will be contained in the neural networks of the other two models. We have therefore used the same network as for the massive neutrinos model, where the parameters m_{ncdm} and deg_{ncdm} were fixed to 0.06 and 1.0, respectively, in order to match a value of the effective number of degrees of freedom of $N_{\text{eff}} = 3.046$. When fixing the two model-specific parameters, the rest of the network behaves like a Λ CDM network trained with these two parameters fixed at the same values.

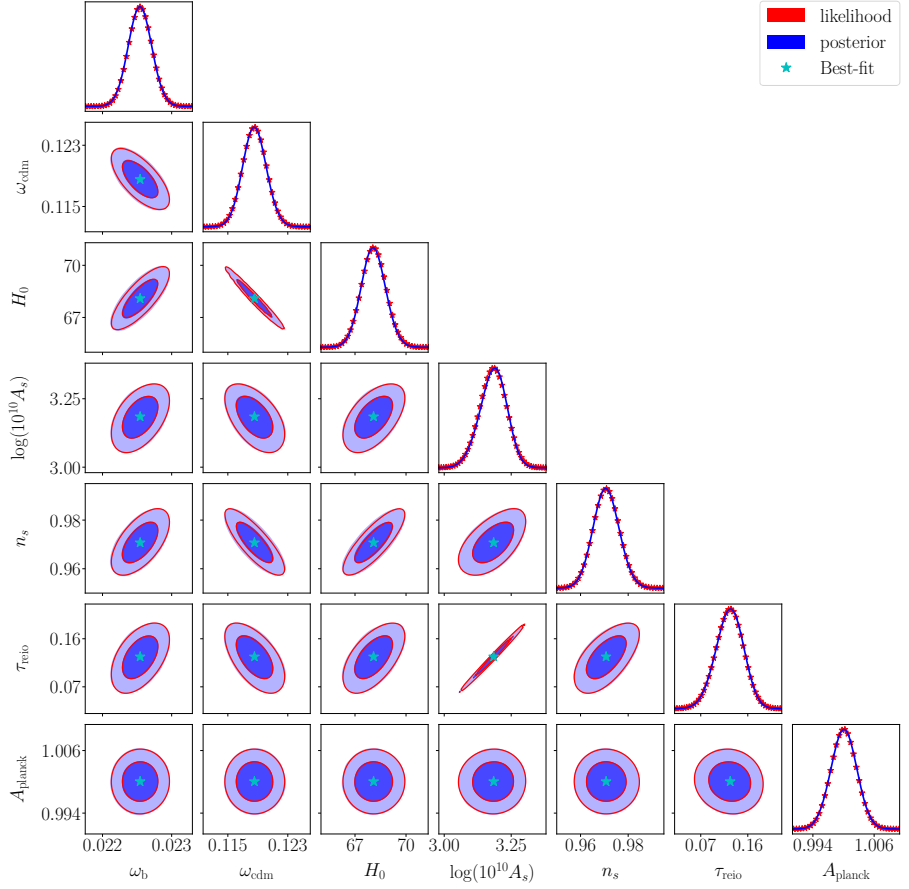


Figure 6.2: Posteriors and profile likelihoods of the Λ CDM model. The blue filled contours and the blue lines on the diagonal are the posteriors from an MCMC run with the neural network for one and two dimensions, respectively, and the red contour lines and the red stars on the diagonal are the profile likelihoods for one or two dimensions, respectively. The cyan star marks the best-fit point from a global optimisation of the entire neural network.

Figure 6.2 shows a full triangle plot of both posteriors and profile likelihoods for the Λ CDM model. The blue lines and filled contours are the one- and two-dimensional posteriors, respectively; the red stars and contour lines are the one- and two-dimensional profile likelihoods, respectively; and the cyan stars mark the best-fit point of the entire parameter space. The stars in the one-dimensional plots are the actual calculated points in the profile likelihoods, but for the two-dimensional plots, we only show the contour lines calculated from the computed points in the same way that the posterior contours are calculated from

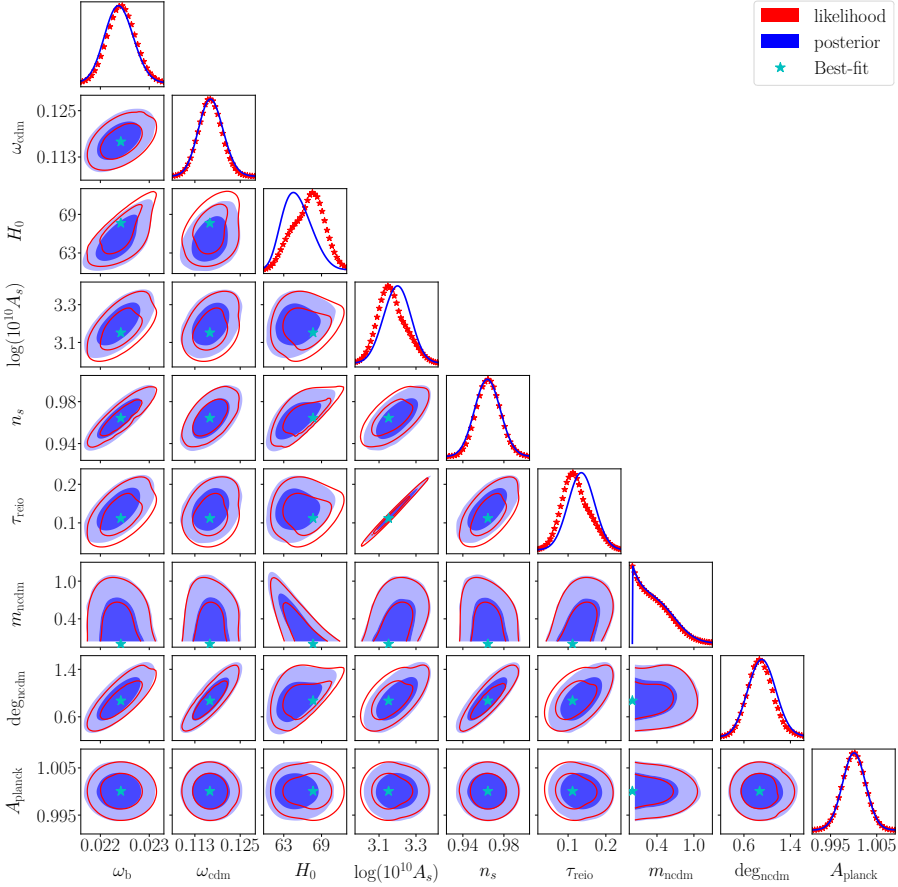


Figure 6.3: Posteriors and profile likelihoods of the massive neutrinos model. The blue filled contours and the blue lines on the diagonal are the posteriors from an MCMC run with the neural network for one and two dimensions, respectively, and the red contour lines and the red stars on the diagonal are the profile likelihoods for one or two dimensions, respectively. The cyan star marks the best-fit point from a global optimisation of the entire neural network.

the histograms. The agreement between the posteriors and the profile likelihood is excellent, and since we would expect this to be true for a near-Gaussian likelihood, this validates our optimisation routine.

6.5.2 MASSIVE NEUTRINOS

Now that we have tested our optimisation on a simple likelihood without any volume effects, like in the Λ CDM model, we can move on to a model that actually contains volume effects. The inclusion of a vari-

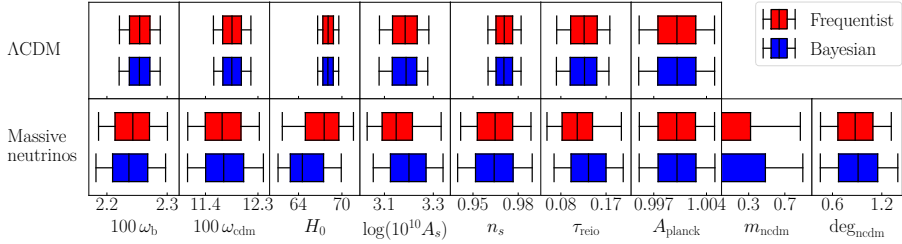


Figure 6.4: 68.27% and 95.45% confidence (credible) intervals found using frequentist (Bayesian) statistics for the two models, Λ CDM and massive neutrinos. We clearly see that the constraints from posteriors (blue) and profile likelihoods (red) are identical for Λ CDM (top panel) but differ somewhat for massive neutrinos (bottom panel). Along with the constraints, the best-fit point has been included as a centreline in the boxes.

able neutrino mass, m_{ncdm} , (two species are assumed to be massless) and the degeneracy, \deg_{ncdm} , introduces new volume in the parameter space, and the likelihood is not Gaussian for the neutrino mass. We should therefore expect to see some differences between the posteriors and the profile likelihoods. Figure 6.3 shows the posteriors and profile likelihoods of this model. The posteriors are again shown in blue, while the profile likelihoods are shown in red. We can clearly see differences between the posteriors and profile likelihoods now, with the profile likelihoods being shifted slightly compared to the posteriors. The most profound shifts are to higher values of H_0 and lower values of both A_s and τ_{reio} , and it is interesting that the actual best-fit point (cyan stars) lies on the edge of the 68.27% credible regions of the posteriors involving H_0 , and the 95.45% confidence regions of the profile likelihoods in H_0 seem to alleviate the Hubble tension significantly. Since we are only using the Planck lite likelihood, we cannot draw any reasonable conclusions based on this, but it really emphasises how the volume of the parameter space can impact parameter inference and that both a Bayesian analysis and a frequentist analysis should be performed in order to get the full picture.

The differences between a model with volume effects and one without are more apparent from figure 6.4, where the 68.27% and 95.45% constraints from both the profile likelihoods and the posteriors are shown for each of the two cosmological models, the Λ CDM model and the massive neutrinos model. The additional volume of the massive neutrinos model not only broadens the constraints but also shifts some of the parameters, as previously discussed.

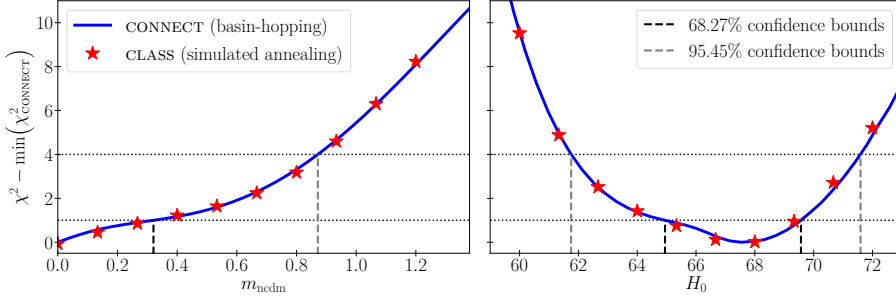


Figure 6.5: Comparison between one-dimensional profile likelihoods using CLASS and CONNECT with simulated annealing and basin-hopping as optimisation routines, respectively. The profile likelihoods are in the parameters m_{ncdm} and H_0 . The 68.27% and 95.45% confidence intervals are also shown.

In order to test the performance of the emulator, we compare the results with profile likelihoods obtained using CLASS and simulated annealing. These are shown in figure 6.5, along with the 68.27% and 95.45% confidence intervals calculated from the CONNECT profile likelihood. We see a great agreement between CLASS and CONNECT and this further strengthens our confidence in the results obtained using CONNECT in figures 6.2 and 6.3.

6.5.3 DECAYING COLD DARK MATTER

In order to really put the framework and optimisation to the test, we have included the model of decaying cold dark matter (DCDM) with dark radiation (DR) as the decay product. This likelihood is notoriously difficult to sample and is also quite challenging when doing profile likelihoods [7]. With Planck lite data, the likelihood features a very slight peak for high values of the decay rate, Γ_{dcdm} , with a height corresponding to only $\Delta\chi^2 \approx 0.5$. For this, we require much higher precision than for the other cosmological models, but achieving this turns out to be very difficult. The optimisation can only ever be as good as the neural network used by the optimisation routine, and a good network suitable for this particular optimisation requires a lot of training data around this subtle likelihood peak. As of now, we are limited by the fact that our training data for the neural network is generated by iteratively sampling from the posterior, and this makes it very hard to properly sample around the peak due to the large volume effects that the posterior is influenced by. It can, of course, be solved by sampling maybe an order

of magnitude more points to use as training data (as well as training for more epochs and with a larger network architecture), but this becomes unfeasible in terms of training the network.

Another approach would be to slice the parameter space at specific values for any parameters exhibiting such problems and train different networks for each value of this parameter. This also requires more computational resources, and if the goal is to have just the one-dimensional profile likelihood in the parameter in question (Γ_{dcdm} in our case), this is quite wasteful, and one might as well compute the profile likelihood with CLASS and simulated annealing. We can, however, use these individual networks for more, and so if we wanted a full triangle plot of profile likelihoods, we would just use the appropriate network for any point in a two-dimensional profile likelihood containing the sliced parameter and use all networks at once for any point in all other two-dimensional profile likelihoods and simply choose the lowest value of χ^2 . This adds a layer of complexity compared to using just a single network containing the whole parameter space, but if any two-dimensional profile likelihoods are needed, then this approach will be orders of magnitude less computationally expensive than using CLASS and simulated annealing.

Figure 6.6 shows the profile likelihood in the Γ_{dcdm} parameter using both CLASS and CONNECT. CONNECT has been tested both using single networks trained on the entire parameter space at different sampling temperatures and individual networks for each fixed value of Γ_{dcdm} . The individual networks for each fixed value of Γ_{dcdm} have been trained with the regular iterative approach of CONNECT [2], and the networks spanning the entire parameter space have been trained with a slight modification. The first part of the training is identical to the regular iterative approach with the default sampling temperature of $T = 5$, but when convergence is reached, the sampling and training do not halt. Instead, new data is gathered with a smaller temperature ($T = 4$) and used to train a new network, which is then used to gather data for a new network with an even lower temperature, etc. The training data from consecutive iterations are not merged in this second part since they have different statistical properties due to differences in sampling temperature. This annealing results in multiple networks with different sampling temperatures according to a predefined list that would not be possible to obtain with just the regular iterative approach. This is because the likelihood is difficult to sample with small temperatures, so

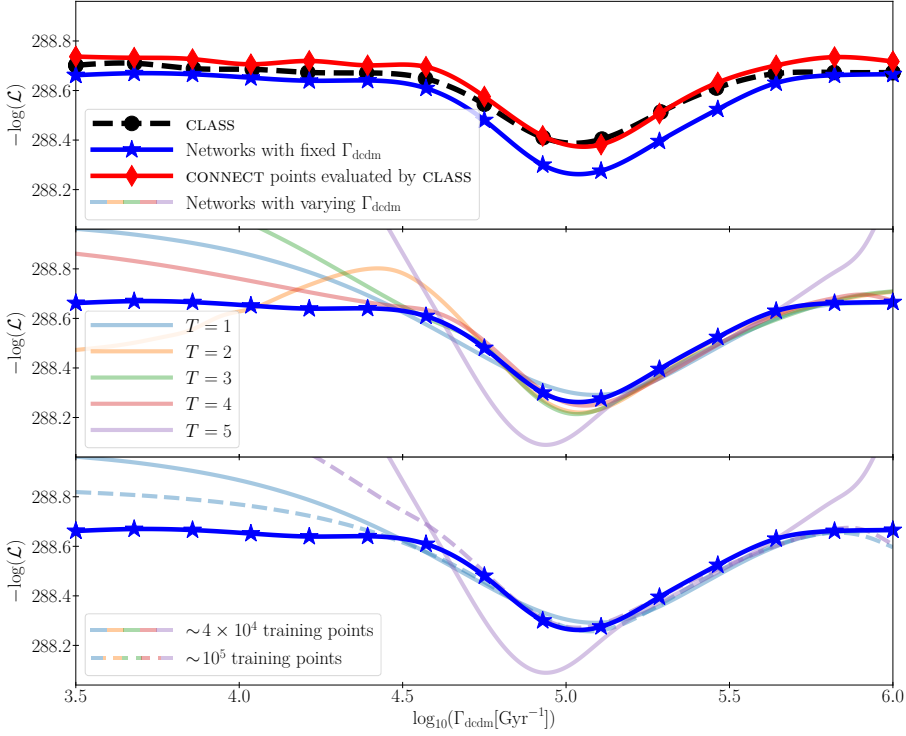


Figure 6.6: The top panel shows profile likelihoods in the Γ_{ddm} parameter with both CLASS (simulated annealing) shown with black circles and CONNECT (basin-hopping and BFGS) using several individual networks for each fixed value of Γ_{ddm} shown with blue stars. The parameter vectors resulting from the CONNECT optimisations have all been evaluated by CLASS and plotted as a profile likelihood, shown with red diamonds. The middle panel shows profile likelihoods resulting from neural networks with Γ_{ddm} as a varying parameter where training data has been gathered at different sampling temperatures (see text). The bottom panel shows the same kinds of profiles as the middle panel, but with different amounts of training data for the networks. Only the temperatures $T=1$ and $T=5$ are shown in order to keep the figure simpler.

having only a small temperature from the beginning will not result in a useable network for e.g. $T = 1$. By having the data gathered with a slightly larger temperature as the foundation for the network gathering the next data with a smaller temperature, it ensures that the network is always very accurate when gathering data. The reason for wanting to do this is that we cannot properly resolve this particular best-fit region with too large of a temperature given that it is a very narrow and slight band in the multidimensional likelihood surface. On the other hand, we

cannot perform the entire sampling at a low temperature because the accuracy would suffer. With inspiration from the simulated annealing algorithm, this new approach takes all of this into account.

Another problem apparent in figure 6.6 is that the networks with different temperatures only seem to agree just around the minimum and for higher values of Γ_{dcdm} ³. This is, however, not surprising since we are sampling training data from the posterior, which is heavily influenced by volume effects towards high values of Γ_{dcdm} in this case. We therefore have much more training data on the right side of the figure, which means that all of the networks have better performance here.

The figure also shows results from individual networks with fixed values of Γ_{dcdm} , and we can clearly see that these resemble the results from CLASS much more. There is a small discrepancy in the depth of the well, but this is due to small precision errors. The difference corresponds to $\Delta\chi^2 \approx 0.2$ and this is definitely not significant given that the same behaviour and shape are produced. The networks with different temperatures also seem to agree with the networks with fixed values of Γ_{dcdm} for high values, so it is highly probable that it is a systematic error founded in the precision of neural networks. These networks have, however, all been trained to per mille precision in all of the C_ℓ spectra, but when investigating the reason for the differences in χ^2 we found that the Planck lite likelihood is very sensitive to certain ranges in ℓ for the TE spectrum. A per mille error in the TE spectrum can lead to errors in χ^2 up to roughly 0.3 around the best-fit, and that seems to be in agreement with what we can see from the figure. This is only relevant for this very slight likelihood peak, and since the peak is more significant when using the full Planck 2018 likelihood [7], the precision is most likely not an issue. Increasing the precision in the networks, e.g. by training for more epochs, could potentially render the effect unnoticeable in figure 6.6, but this would be without merit due to the χ^2 precision already being much greater than what is needed in any actual case of use. Regardless of the minute χ^2 discrepancy, the CONNECT networks seem to find the same optimal parameter vectors that CLASS finds, given that evaluating the points with CLASS results in more or less the same profile as CLASS finds on its own (red diamonds in the top panel of the figure). The TT spectrum is furthermore known to have much more constraining power than both TE and EE , so we

³ The network with $T = 5$ is worse than the others due to it having training data from all previous iterations as well, which usually is not a problem, but in this case the precision needs to be very high.

should expect the same shape in the profile likelihoods due to the sufficiently high precision in the TT spectrum, and the small discrepancy from the precision in the TE spectrum only shifts the χ^2 values by a small amount and not the parameter vectors. Parameter inference is therefore not affected by this discrepancy. This is supported by the curve with the red diamonds in figure 6.6, which shows the same points in the parameter space obtained by the CONNECT networks with fixed values of Γ_{dcdm} but evaluated by CLASS. We see that this curve is very close to the one actually obtained by CLASS, and this indicates that the optimisations of the neural networks find the correct optima.

It stands to reason that we should be able to obtain the same level of accuracy with a single network spanning the entire parameter space as we can with individual networks at fixed values of Γ_{dcdm} , and indeed the results seem to converge if we generate larger amounts of training data, which is apparent from the bottom panel of figure 6.6. When increasing the amount of data much further than shown here, it is necessary to train the networks for more epochs and perhaps increase the size of the networks, i.e. hidden layers and nodes in each layer. Given that we have 15 individual fixed-value networks each with $\sim 5 \times 10^4$ training points, 6 hidden layers, and 1024 nodes in each hidden layer, we can definitely justify making a much larger network instead with close to $\sim 10^6$ training data points since this would be equally expensive in terms of computational power. Increasing the number of trainable parameters in the network also increases the amount of information it can hold, which eventually will bring the emulation error down to a point where there will be no discrepancy between CLASS and CONNECT.

6.5.4 COMPUTATIONAL PERFORMANCE

It is obvious that the computational cost of computing profile likelihoods with CONNECT is much lower than when using CLASS simply because the evaluation time of a CONNECT network is around 3 orders of magnitude faster than that of CLASS [2]. Just using CONNECT instead of CLASS with simulated annealing would therefore be a huge speed-up. The utilisation of gradients, however, boosts the speed-up even further since fewer function evaluations are needed. In order to use gradients in TensorFlow, a computational graph of the entire gradient computation needs to be constructed, which roughly takes around a minute, after which the evaluation of both the likelihood and the gradients takes

around $\sim 10^{-2}$ seconds. Given that CLASS is allowed to run on multiple threads, the evaluation time is around ~ 1 second. Using the basin-hopping algorithm described in section 6.4 with the BFGS optimiser as the local optimiser, each local optimisation requires $\sim 10^2$ function calls, each ensemble contains $\sim 10^1$ walkers, and the temperature is updated until convergence (usually around 2–4 times). This results in $\sim 10^3$ – 10^4 evaluations, each taking $\sim 10^{-2}$ seconds. This means that a single point will usually converge in less than a minute on a single CPU core. The simulated annealing algorithm requires on the order of $\sim 10^4$ – 10^5 evaluations for each point in the profiles presented in this paper, and with an evaluation time dominated by CLASS, a single point will converge in roughly a few days if run sequentially. There is, however, some degree of parallelisability, given that multiple chains can be utilised and CLASS can be parallelised to around 8–10 cores. The different points of a profile can, of course, be computed completely separately, regardless of which algorithm is used.

There is also the matter of gathering training data for a neural network if one is starting from scratch, and this process is quite time-consuming compared to the use cases of a trained network [2]. For most likelihoods, the network requires around $\sim 50,000$ points of training data, which means that CLASS needs to be evaluated $\sim 50,000$ times. This is, however, very parallelisable, and it is much faster than doing an actual MCMC run with CLASS – especially for beyond Λ CDM where the number of CLASS evaluations can be as high as 10^5 – 10^6 . If one only seeks to optimise a single point of a (Λ CDM) model, it might be better to use CLASS and simulated annealing, but for an entire one-dimensional profile, it is very beneficial to use CONNECT instead, and for two-dimensional profiles requiring several hundred points, it might be necessary to use CONNECT. If one seeks to perform a full frequentist analysis with triangle plots of one- and two-dimensional profiles, the task is virtually impossible at this point without using CONNECT.

6.6 CONCLUSION AND OUTLOOK

Using the ensemble basin-hopping algorithm for global optimisation combined with the BFGS optimiser for local optimisation has been demonstrated to yield robust, fast, and accurate results when calculating profile likelihoods from CMB data. By making use of gradients in the local optimiser, the number of function evaluations can be greatly

decreased when compared to gradient-free simulated annealing. This, together with the much faster evaluation time of `CONNECT` compared to `CLASS`, results in speed-ups of several orders of magnitude when constructing profile likelihoods. In addition to being fast, the method is also very robust and accurate, and given the smoothness of the neural networks, the global optimisations often converge with only a few local optimisations.

When a neural network has been trained, the profile likelihoods in any parameter or set of parameters are computationally very inexpensive, and entire triangle plots of profile likelihoods, typically consisting of more than 5000 individual points in parameter space for which optimisation must be performed, are easily computed. Each such point in the parameter space is independent of all other points, making the optimisations embarrassingly parallelisable. The optimisation of each point takes around a minute on a single modern CPU core. A full triangle plot could therefore be computed on a normal quad-core laptop in less than a day.

With fast and easy access to profile likelihoods in cosmology, it is easy to investigate cosmological models with both Bayesian and frequentist statistics. Neither of these statistical frameworks can claim superiority compared to the other, but the two different approaches answer different questions and complement each other. It is therefore very useful to have both a posterior and a profile likelihood in order to get the full picture and draw reasonable conclusions about the given model. The biggest reason for this not being done frequently in analyses of cosmological models is that the computational costs of profile likelihoods are much greater than those of posteriors. Having a fast emulation tool for quick computations of both posteriors and profile likelihoods makes it much easier and more appealing to (re)introduce the frequentist approach in cosmology.

While the optimisation is typically extremely precise, the precision is limited by the emulation precision of the underlying network. Therefore, it is quite important to use well-trained and precise networks to derive profiles consistent with those obtained using `CLASS` and simulated annealing. In this work, the same neural network has been used for all profile likelihoods and MCMC related to the Λ CDM model and the massive neutrino model due to its simple shape and only modest volume effects. The agreement with profile likelihoods from `CLASS` is excellent for these two models and very precise out to several standard

deviations in parameter space. For the Λ CDM model, the agreement with CLASS is also quite good when using either individual networks for each point in the decay rate, Γ_{dcdm} , or an annealed network with a large amount of training data, even though a small offset is visible between the profile likelihoods. This is, however, on a scale of $\Delta\chi^2 \approx 0.2$ which is much too small to be significant, and parameter inference would not be affected by this small discrepancy.

The training data for the CONNECT neural networks are sampled iteratively from the posterior, and with large volume effects, this can bias the accuracy of the network away from regions of maximal likelihood: Regions with a large volume are sampled much more than regions with a small volume, even though the small volumes might have better likelihood values. This creates a bias in the networks towards larger volumes due to these being much more represented in the training data. It is possible to overcome this problem using a larger network architecture, more training data, and more training epochs. The training data can also be weighted in a way that reduces the impact of data points with larger values of Γ_{dcdm} on the total loss during training. However, it might be beneficial to pursue new ways of sampling training data more suited for profile likelihoods in future works.

Another thing to consider is the complexity of the likelihood function at this point. We evaluate a network to get C_ℓ spectra, perform a cubic interpolation, and use a TensorFlow version of the Planck lite likelihood in order to go from cosmological parameters to a likelihood value, and this makes the computational graph of the gradients quite large in terms of memory. A way to speed up the computations even further and increase usability is to directly emulate the likelihood value of any given likelihood code. We then lose the dependency on having likelihood codes written in TensorFlow syntax or emulated separately. Most of the problems highlighted in this paper will most likely cease to exist with this approach of directly emulating likelihoods. An additional advantage is that any likelihood emulated in this way (e.g. BAO, SNI-a, etc.) will automatically lend itself to auto-differentiation, allowing the efficient combination of basin-hopping and BFGS local optimisation to be used.

Reproducibility. We have used the publicly available CONNECT framework available at https://github.com/AarhusCosmology/connect_public to create training data and train neural networks. The frame-

work has been extended with the basin-hopping optimiser and a module for computing profile likelihoods. Explanatory parameter files have been included in the repository in order to easily use the framework and reproduce results from this paper.

ACKNOWLEDGEMENTS

We acknowledge computing resources from the Centre for Scientific Computing Aarhus (CSCAA). A.N., E.B.H., and T.T. were supported by a research grant (29337) from VILLUM FONDEN.

APPENDIX 6.A: CHOOSING POINTS FOR TWO-DIMENSIONAL PROFILE LIKELIHOODS

When computing two-dimensional profile likelihoods, we often need many more points than what is suitable for one-dimensional profile likelihoods, especially if the points are not selected in a clever way. A square grid with the same ranges as the one-dimensional profile likelihoods is certainly possible, but not the most feasible in terms of computational resources. Ideally, we would like to choose a collection of points from the best-fit region stretching to at least the 3σ contours (99.73% confidence level) in order to have enough points to accurately compute the 68.27% and 95.45% confidence regions. A reasonable choice of points is choosing the bin centres of the histogram of an MCMC run (using e.g. MONTEPYTHON [63, 101]) with the same CONNECT model where the bin counts are different from zero. This means that each point will correspond to a small region where the MCMC sampler has accepted at least one point. This is what is used to compute the two-dimensional credible regions of the posterior, and the idea is that the same points should well represent at least part of the confidence regions of the profile likelihood. This will almost always be the case unless very significant volume effects are at play.

Figure 6.7 shows the bin centres of one such histogram, and the different panels are coloured according to either the bin counts or the likelihood value. We can see that the points encapsulate the entire 99.73% credible region but not the entire 99.73% confidence region. The 68.27% and 95.45% confidence regions are, however, well represented. More elaborate methods for obtaining points in a clever way could be

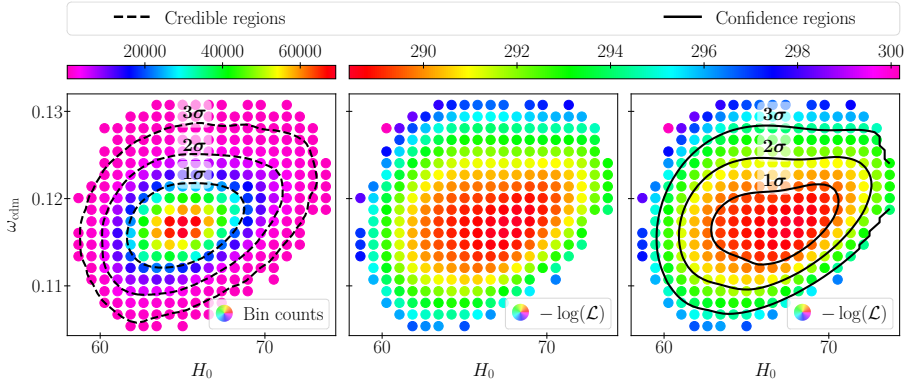


Figure 6.7: Bin centres from the histogram in the $(\omega_{\text{cdm}}-H_0)$ -plane from MONTEPYTHON. The left panel is coloured according to bin counts of the histogram, and the dashed 1σ , 2σ , and 3σ contours are the 68.27%, 95.45%, and 99.73% credible regions, respectively. The middle and right panels are coloured according to the likelihood values, and the solid 1σ , 2σ , and 3σ contours in the right panel are the 68.27%, 95.45%, and 99.73% confidence regions, respectively. The points from the histogram are suitable for the 68.27% and 95.45% confidence regions, but additional points are required for the 99.73% confidence region.

employed, but it is worth mentioning that this approach guarantees that the optimisation is embarrassingly parallelisable.

APPENDIX 6.B: RECOMPUTING AND ADDING POINTS

Since the optimisation routine, as described in section 6.4, has some level of stochasticity, the optimisation might fail by converging on a local optimum a small fraction of the time. By tweaking precision settings and hyper-parameters, the rate of failed optimisations can be greatly decreased, but there will always be some probability of not succeeding. With the default settings, a complete triangle plot of profile likelihoods containing $\sim 10^3$ points to compute will result in only $\sim 10^1$ failed points. This is difficult to detect automatically, but it is very easily seen when plotting the profile likelihoods. A routine to interactively choose failed points after plotting them has been implemented, and figure 6.8 shows the process of choosing points to recompute. These are then gathered in a file and recomputed. Given the low probability of getting stuck in a local optimum, the recomputation is almost always successful. In rare

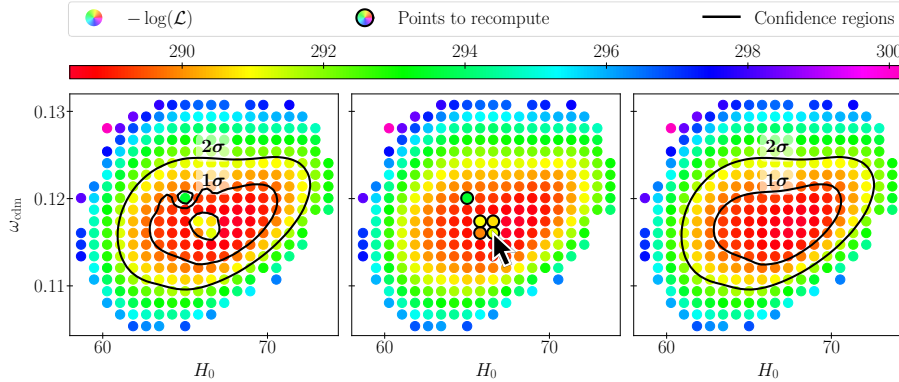


Figure 6.8: The panels from left to right show the process of recomputing specific points in the two-dimensional $(\omega_{\text{cdm}}-H_0)$ -profile likelihood if the optimiser converges on a local optimum. The points to recompute can be selected interactively and are then optimised again with a better result.

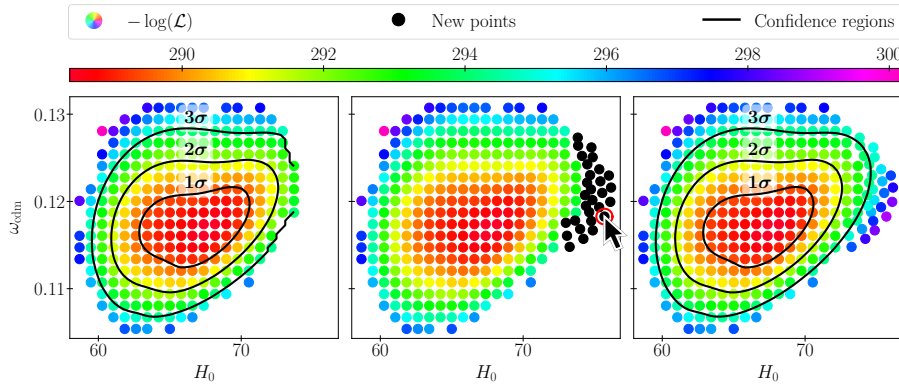


Figure 6.9: The panels from left to right show the process of adding extra points to the two-dimensional $(\omega_{\text{cdm}}-H_0)$ -profile likelihood if the set of points does not encapsulate the entire contour of a confidence level. New points can be chosen interactively and are then optimised to complete the 3σ contour.

cases, a few points need to be recomputed twice, and if a specific point turns out to be particularly difficult to optimise, then the precision settings might need adjustment for that single optimisation. This is still much more feasible than running with very precise settings for all points in the profile likelihoods.

If the computed points, chosen according to the process described in appendix 6.A, do not encapsulate the contour of a specific confidence level, additional points have to be selected and optimised. This is also

difficult to choose automatically, but it is very easy to pick new points by looking at the contours and previously optimised points. A routine for interactively choosing new points has also been implemented, and using this, one can easily choose points based on the location of all current points and the contours based on those. Figure 6.9 shows the process of choosing new points since the 99.73% confidence region is not entirely encapsulated by the previously chosen points. The new points are gathered in a file and optimised. After the inclusion of the new optimised points, the contour line looks as it should and is fully represented by the total set of points.

Ending of reference [3]

USING CONNECT FOR BAYESIAN EVIDENCES

Another task that has been difficult with conventional means, is the process of computing the Bayesian evidence, which is the normalisation of the posterior and a measure of the entire probability volume. This is a great quantity for comparing two cosmological models and determining the one that overall fits the data best. It is usually computed using *nested sampling* to find this probability volume, but it requires a huge number of function evaluations to accurately determine the integral over the entire parameter space.

The fast evaluation time of an emulator trained with CONNECT is very beneficial when a large number of evaluations is required, so it comes as no surprise that Bayesian evidence is much more easily obtained when utilising the emulator. This chapter presents a paper that I co-authored using CONNECT for exactly that:

- *Camilla T. G. Sørensen, Steen Hannestad, Andreas Nygaard, and Thomas Tram. “Calculating Bayesian evidence for inflationary models using CONNECT.” In: (June 2024). arXiv: 2406.03968 [astro-ph.CO].*

The paper was a continuation of a master’s project by Camilla Sørensen, for which I helped with a slight modification to the source code of CONNECT in order to implement the inflationary code ASPIC [176] and sample in inflationary parameters. We extensively investigated the convergence of the nested sampler POLYCHORD and discovered a default setting that was not appropriate when using an emulator. This is discussed in appendix 7.A.

In the paper, several inflationary models are treated and Bayesian evidences are computed and compared to results in the literature. As in the case of the former chapter, the paper here also represents a proof of concept more than an actual study of inflationary models. The difference in computational costs between conventional computations of the

Bayesian evidence and when using the emulator was immense, so this makes it a particularly good application of CONNECT. A common point of critique for active learning emulators like CONNECT is that they have to be retrained for different scenarios. In this case, however, we only needed to train a single emulator outputting the parameters that ASPIC uses as input. This further increases the difference in computational costs if all results were to be repeated without emulation.

While my contribution to the paper was primarily on the use of CONNECT as well as discussions regarding the convergence issues, I find the results of the paper relevant to this thesis, and I will therefore include the paper in its entirety.

Beginning of reference [6]

CALCULATING BAYESIAN EVIDENCE FOR INFLATIONARY MODELS USING CONNECT

Camilla T. G. Sørensen^a, Steen Hannestad^a, Andreas Nygaard^a, Thomas Tram^a

^a*Department of Physics and Astronomy, Aarhus University, DK-8000 Aarhus C, Denmark*

Abstract. Bayesian evidence is a standard tool used for comparing the ability of different models to fit available data and is used extensively in cosmology. However, since the evidence calculation involves performing an integral of the likelihood function over the entire space of model parameters this can be prohibitively expensive in terms of both CPU and time consumption. For example, in the simplest Λ CDM model and using CMB data from the Planck satellite, the dimensionality of the model space is over 30 (typically 6 cosmological parameters and 28 nuisance parameters). Even the simplest possible model requires $\mathcal{O}(10^6)$ calls to an Einstein–Boltzmann solver such as CLASS or CAMB and takes several days.

Here we present calculations of Bayesian evidence using the CONNECT framework to calculate cosmological observables. We demonstrate that we can achieve results comparable to those obtained using Einstein–Boltzmann solvers, but at a minute fraction of the computational cost.

As a test case, we then go on to compute Bayesian evidence ratios for a selection of slow-roll inflationary models.

In the setup presented here, the total computation time is completely dominated by the likelihood function calculation which now becomes the main bottleneck for increasing computation speed.

7.1 INTRODUCTION

Over the past three decades, a vast amount of cosmological data has yielded unprecedented knowledge of the physical model of our universe. The standard Λ CDM model is described in terms of relatively few free parameters and provides a very good fit to almost all observational data. Various statistical techniques have been used to infer the value of the fundamental physical parameters of the model, including Bayesian parameter inference through marginalisation of the likelihood function (see e.g. [101, 102, 156]), and maximum likelihood techniques in the form of profile likelihoods (see e.g. [7, 8, 146, 177, 178]). Another extremely useful tool is the calculation of Bayesian evidence when comparing different models (see e.g. [179] for a review). However, a major obstacle in evidence calculation is that it requires integration of the likelihood function over the entire prior volume, which, for high dimensional parameter spaces, can become prohibitively expensive.

Packages based on the nested sampling approach to likelihood integration [180, 181] are by now available for carrying out such analyses in a relatively efficient manner. POLYCHORD [182] and MULTINEST [183] are among the most commonly used within the field of cosmology (see e.g. [184] for a recent review of methods and packages). However, even with these packages, a reliable evidence calculation typically still requires millions of evaluations of the likelihood function. Each such evaluation requires running an Einstein–Boltzmann solver such as CLASS [27] or CAMB [159] to calculate the relevant cosmological observables and takes on the order of tens of seconds on a single CPU core (although a significant speed-up can be achieved in cases where the model parameter space can be split in “slow” (cosmological) and “fast” (nuisance) parameters). This makes evidence calculations extremely expensive, both in terms of time and computational resources.

A way to mitigate this could be to use a cosmological emulator instead of the Einstein–Boltzmann solver code. Recent years has seen a surge in popularity of such emulators and they have been applied in many dif-

ferent ways. The most common kinds of emulators are either based on Artificial Neural Networks [2, 109, 113, 157] or Gaussian Processes [122, 158], both with their respective advantages and drawbacks. The applications range from standard Bayesian marginalisation to frequentist profile likelihoods [3], and Refs. [185–187] furthermore employed emulators to approximate Bayesian evidence using posterior samples and a modification of the harmonic mean estimator [188]. While approximations of the Bayesian evidence are useful to roughly compare cosmological models with very different evidence, models that only differ slightly need better estimates (e.g. from nested sampling) in order to perform a meaningful comparison. Ref. [113] demonstrated that evidence computations could indeed be accurately computed using an emulator, albeit for the vanilla Λ CDM-model and using large-scale structure likelihoods and/or Planck Lite.

In this paper we test how the CONNECT [2] framework fares on evidence calculations by performing Bayesian model comparison of a variety of different slow roll inflationary models using the publicly available POLYCHORD package [182]. Accurate profile likelihoods require the emulator to be very accurate around the region of best fit, but in general they do not require very accurate emulation of other regions in the parameter space [3]. Marginalisation, on the other hand, requires integration over regions of parameter space. While this typically requires somewhat less precision around the absolute best fit, it requires the emulation to be reasonable over substantially larger regions. Evidence calculations are even more extreme in this regard since each evidence calculation requires integrating the likelihood function over the entire prior volume.

Given that evidence calculations are extremely time consuming due to the very large number of function evaluations required (typically millions of CLASS or CAMB evaluations, each requiring tens of CPU core seconds), it is of substantial interest to investigate whether the CONNECT emulator can also be used for this purpose. In order to compare our results to model comparisons using standard Einstein–Boltzmann solvers, we use the same prior ranges and model parameterisations as in Ref. [189].

Finally, since we are using inflationary model selection as our test case, we must of course credit the pioneering work in Refs. [190, 191]. (See also Ref. [192] for a very recent update.) In these papers, the authors computed an effective likelihood by integrating out all non-

inflationary parameters. A neural network was then trained to emulate this effective likelihood, allowing the authors to perform an exhaustive Bayesian model-comparison of most slow-roll inflationary models in the Λ CDM-model. Furthermore, we must also mention the early work done on inflationary model selection in Ref. [193].

The paper is structured as follows: In section 7.2 we provide an overview of both the CONNECT framework and of Bayesian evidence calculations. Section 7.3 contains a description of how the CONNECT neural network emulator is constructed and validated using standard inflationary observables. Section 7.4 is then devoted to a description of how we implement the ASPIC framework for describing slow-roll inflationary models and converting fundamental inflationary parameters to observables, and section 7.5 contains our numerical results. Section 7.6 contains our runtime considerations. Finally, we provide our conclusions in section 7.7.

7.2 THE CONNECT FRAMEWORK AND BAYESIAN EVIDENCE

The CONNECT framework for emulation of cosmological observables has been tested extensively for cosmological parameter inference, using both Bayesian marginalisation and frequentist profile likelihoods [2, 3]. A main advantage of CONNECT is that it contains both an emulator of cosmological observables as well as the framework needed to build and train the network. This means that emulators of new and non-standard cosmological models can easily be built and used to run cosmological parameter analyses within a single environment. CONNECT trains a neural network based on training data sampled iteratively to best represent the likelihood function. This ensures that the neural network is most precise where the likelihood is large, which makes it ideal for parameter inference. The training data is gathered using the fast Planck Lite likelihood [127]. The reason is that training requires a very large number of likelihood evaluations, which in the case of the full Planck likelihood would be prohibitively expensive. Because Planck Lite is somewhat less constraining than the full Planck likelihood, this gives us a set of training data that is more widely spread and this (along with a high sampling temperature, which guarantees adequate coverage of the training data set), yields a set of training data that can accurately represent several combinations of cosmological data sets (as long as either the full Planck

likelihood, Planck Lite or similar CMB data is included) without the need to retrain the network. Furthermore, `CONNECT` is both the emulator as well as the procedure that trains the emulator. This makes it easy to quickly train a new physics model.

However, parameter inference as a statistical technique is designed for determining parameter values within a given model, assuming the model to be correct, i.e. it is not designed to qualitatively compare how well different models fare in fitting the available data. For this purpose other techniques, such as the Akaike information criterion in frequentist analysis (see e.g. Ref. [194]) or evidence in Bayesian analysis, are used instead. The Akaike information criterion relies on maximising the likelihood function and is therefore closely related to the profile likelihood technique already tested extensively with `CONNECT` [3]. However, the Bayesian evidence calculation requires integrating the likelihood function over the entire prior volume, and testing the precision (and speed) with which `CONNECT` is able to perform this calculation is the main purpose of this work.

The Bayesian evidence has been calculated with the code `POLYCHORD` [182, 195], which uses a version of nested sampling [180] to calculate the evidence. The code is run from within the MCMC sampler `MONTYPYTHON` [63, 101] with either `CLASS` or `CONNECT` as the cosmological theory code. We finally note that it is only necessary to train one model with `CONNECT`, because all the inflationary parameters for the different inflationary models can be mapped to the same physical parameters.

7.3 VALIDATION OF `CONNECT` FOR EVIDENCE COMPUTATION

A natural first step is to validate results for Bayesian evidence calculated using `CONNECT` versus brute force calculations based on `CLASS` (or `CAMB`). The accuracy of `CONNECT` has been investigated thoroughly for both Bayesian parameter inference and profile likelihoods and found to be more than sufficiently accurate for such analyses, even in very extended parameter spaces (see [2]). However, the calculation of Bayesian evidence typically lends more weight to regions in parameter space where the likelihood is only moderately good. This means that one cannot directly infer from these previous tests that `CONNECT` performs Bayesian evidence calculations at the required level of precision.

Parameter	Minimum value of prior	Maximum value of prior
$100 \times \omega_b$	1.9	2.5
ω_{cdm}	0.095	0.145
$100 \times \theta_s$	1.03	1.05
$\ln 10^{10} A_s$	2.5	3.7
τ_{reio}	0.01	0.4
n_s	0.94	1.0
α_s	-0.3	0.3
r	0.0	0.3

Table 7.1: The parameter bounds used to validate the results for Bayesian evidence calculated using *CONNECT* versus calculations based on *CLASS*.

To test this, we calculate evidence in models based on Λ CDM, but with an extended inflationary component. The basis is the simplest inflationary slow-roll approximation in which the primordial fluctuations are adiabatic, Gaussian, and purely scalar and can be parameterised using only the amplitude, A_s , and the spectral index, n_s . Beyond this, we have added the tensor-to-scalar ratio, r , as well as the effective curvature of the primordial spectrum, α_s , so that the primordial fluctuation spectrum is fully described by four parameters¹ (A_s , n_s , r , and α_s) in addition to the parameters needed to describe the content of the flat Λ CDM model, i.e., $\omega_b, \omega_{\text{cdm}}, \theta_s, \tau_{\text{reio}}$.

The parameter bounds for $100 \times \omega_b$, ω_{cdm} , $100 \times \theta_s$, $\ln 10^{10} A_s$, and τ_{reio} is the same as in Ref. [189]. The bounds for these parameters as well as the bounds for n_s , α_s , and r can be seen in table 7.1.

Since our main goal is to demonstrate the feasibility of using the *CONNECT* framework for Bayesian evidence calculations, we will use the

¹ Validating *CONNECT* on this particular model has the advantage that since all the slow-roll models to be investigated can be mapped to the set of effective inflationary “observables”, A_s, n_s, α_s, r , we can infer that our set-up will also be valid for evidence computations using fundamental inflationary field parameters.

Case	$\log \mathcal{Z}$	Likelihood calls	CPU hours used
POLYCHORD with CLASS	-1860.4 ± 0.54	1,419,152/ 115,988,382	30,000
POLYCHORD with CONNECT	-1861.1 ± 0.55	1,569,491	125

Table 7.2: The Bayesian evidence calculated with POLYCHORD using both CONNECT and CLASS and 300 live points. The number of likelihood calls are shown for the “slow”/“fast” parameters for CLASS while this oversampling feature was turned off for CONNECT. Note that the log-evidence and the error estimates are comparable despite the CLASS-run using ~ 100 times more total likelihood calls. The last column shows the CPU core-hours used in each POLYCHORD run.

same data combination as in Ref. [189]. However, it should be stressed that evidence calculations are notoriously hard to compare because exact numbers are extremely sensitive to hyperparameter choices such as e.g. prior volume. Therefore, it is not to be expected that a direct, quantitative comparison can be made between our results and those of [189]. Our data sets therefore in all cases consist of the full Planck 2018 *TTTEEE*+low E data [127], the Planck 2018 lensing data [196], as well as the BICEP Keck 2015 data [197].

In the standard setting for POLYCHORD when run through MONTEPYTHON [101] there is a distinction between “slow” (cosmological) and “fast” (nuisance) parameters. The POLYCHORD wrapper for MONTEPYTHON is hard-coded to use 0.75 of the total wall time of the computation for integration of the cosmological parameter space and 0.25 on the nuisance parameter space. Given the difference in execution time between CLASS and the likelihood calls this typically leads to at least an order of magnitude more evaluation points in the nuisance parameter space than in the cosmological parameter space, but since the nuisance parameter space typically has much higher dimension the standard setting for POLYCHORD with MONTEPYTHON provides a reasonable division between the two sets of parameters.

However, when POLYCHORD is run using CONNECT this division between parameter spaces becomes catastrophically wrong. The reason is that *all* function calls in this case takes the same time because CPU time

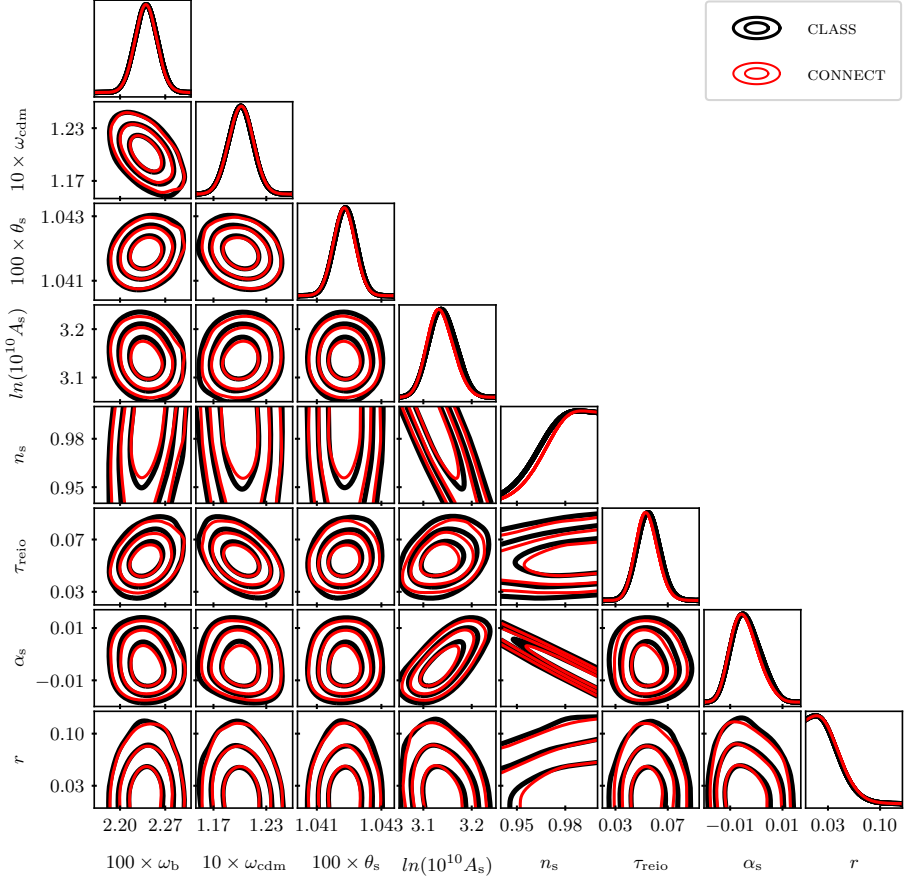


Figure 7.1: The posteriors for the physical and inflationary parameters with the bounds given in table 7.1. The contours correspond to 68.3%, 95.5%, and 99.7% credible intervals.

is entirely dominated by the time taken for likelihood calls. This means that the nuisance parameter space becomes severely under sampled and only if a much larger number of live points is used can convergence be achieved. The solution to this problem is to let POLYCHORD use its normal default setting in which all parameters are treated equally. In appendix 7.A we provide a more detailed discussion of the problem and its solution.

Using the new setting for POLYCHORD with CONNECT, the Bayesian evidence for the above model is then calculated with POLYCHORD using both CONNECT and CLASS using 300 live points in both cases. The values of the evidences can be seen in table 7.2 together with the total

number of likelihood calls in both cases. The resulting posteriors for the physical and inflationary parameters can be seen in figure 7.1.

In appendix 7.A we also discuss convergence in terms of the number of live points used. Although we do find that even using as little as 300 live points is enough to obtain robust results, the CPU requirements of the CONNECT-based runs are small enough that we opt to run all our inflationary model evidence calculations using 1200 live points.

7.4 INFLATIONARY MODEL PARAMETERISATION

In order to calculate Bayesian evidence for different inflationary models and their fundamental parameters, we have used the publicly available code ASPIC [176]. ASPIC takes the inflationary model and its inflation parameters as input and calculates n_s , α_s , and r , which can then be given as input to a neural network trained by CONNECT. The neural network then returns observables that can be used to compute a likelihood based on the given parameters of the inflationary model. Bayesian evidence is then computed using POLYCHORD.

ASPIC is written in FORTRAN, so in order to use the code with CONNECT and MONTEPYTHON, we have written a Python wrapper², PYASPIC for ASPIC that can be called by CONNECT.

The inflationary models used in this article have the following names in ASPIC: Higgs Inflation (HI), Large Field Inflation (LFI) with $p = 2$ and $p = 4$, Natural Inflation (NI), Loop Inflation (LI), and Coleman-Weinberg Inflation (CWI). The models, their potentials, as well as their names in Ref. [189] can be seen in table 7.3.

The bounds for the physical parameters ($100 \times \omega_b$, ω_{cdm} , $100 \times \theta_s$, $\ln 10^{10} A_s$, and τ_{reio}) are the same as for the validation of the CONNECT network, and they can be seen in table 7.1. The inflation parameters and their bounds are $\ln \rho_{\text{reh}}$ with bounds $\ln(1 \text{ TeV})^4$ and $\ln \rho_{\text{end}}$. The model NI has the parameter f with bounds in logspace been given as $0.3 \leq \ln f \leq 2.5$, the model LI has the parameter α with bounds in logspace given as $-2.5 \leq \ln \alpha \leq 1.0$, and the model CWI has a parameter α held constant at $4e$ as well as the parameter Q with bounds $0.00001 \leq Q \leq 0.001$. Furthermore, the model LFI also has a parameter p , and this model is run twice with p held constant at $p = 2$ and $p = 4$.

² Available at <https://github.com/AarhusCosmology/PyAspic>.

ASPIC model	Model name in Ref. [189]	Potential
Higgs Inflation (HI)	$R + R^2 / (6M^2)$	$M^4 \left(1 - e^{-\sqrt{2/3}\phi/M_{\text{pl}}}\right)^2$
Large Field Inflation (LFI ₂)	Power-Law Potential	$M^4 \left(\frac{\phi}{M_{\text{pl}}}\right)^2$
Large Field Inflation (LFI ₄)	Power-Law Potential	$M^4 \left(\frac{\phi}{M_{\text{pl}}}\right)^4$
Natural Inflation (NI)	Natural Inflation	$M^4 \left[1 + \cos\left(\frac{\phi}{f}\right)\right]$
Loop Inflation (LI)	Spontaneously broken SUSY	$M^4 \left[1 + \alpha \ln\left(\frac{\phi}{M_{\text{pl}}}\right)\right]$
Colemann- Weinberg Inflation (CWI)	Not in the reference	$M^4 \left[1 + \alpha \left(\frac{\phi}{Q}\right)^4 \ln\left(\frac{\phi}{Q}\right)\right]$

Table 7.3: The inflationary models used in this paper. See the text for details on the parameters and their bounds

respectively. The effective equation of state w is $1/3$ for LFI with $p = 4$ and 0 for all other models.

7.5 NUMERICAL RESULTS

After having validated the CONNECT framework for the purpose of calculating evidences, we now proceed to calculate evidence ratios for the selection of actual slow-roll models discussed in the previous section. All the inflationary models given in table 7.3 are run from MONTEPYTHON with POLYCHORD, CONNECT, and ASPIC. The number of live points for the nested sampling algorithm is 1200 for all models³. The calculated evidence for all models with respect to the calculated Bayesian evidence for Higgs Inflation can be seen in table 7.4.

³ As discussed in appendix 7.A, even 300 live points is enough to calculate reliable evidences, but the calculation is sufficiently fast that we can use 1200 live points and thereby also achieve a somewhat smaller statistical uncertainty on the obtained results.

ASPIC model	$\ln \mathcal{B}$
Large Field Inflation (LFI_2)	-8.8 ± 0.4
Large Field Inflation (LFI_4)	-51.2 ± 0.4
Natural Inflation (NI)	-4.6 ± 0.4
Loop Inflation (LI)	-4.7 ± 0.4
Colemann-Weinberg Inflation (CWI)	-19.7 ± 0.6

Table 7.4: The calculated Bayesian evidence of the inflationary models with respect to the calculated Bayesian evidence for Higgs inflation. In [189], corresponding values for LFI_2 , LFI_4 , NI, and LI are -11.5, -56.0, -6.6, and -6.8 respectively. Coleman-Weinberg inflation was not tested in that work. The uncertainties on the values from Ref. [189] is quoted as 0.3 in the article using 512 live points (note that estimated statistical uncertainties are typically significantly smaller for the same number of live points when using CLASS because of the very large number of likelihood evaluations in the nuisance parameter space).

When comparing Bayesian evidence from different models, the *Jeffreys scale* is often used [198]. Depending on the value of the Bayes factor between two models, the scale helps interpret if the strength of the evidence is either inconclusive, weak, moderate, or strong for one model compared to the other [179]. The threshold values for Jeffreys scale can be seen in table 7.5.

Using table 7.5 to interpret the results given in table 7.4, it can be seen that Large Field Inflation with both $p = 2$ and $p = 4$ as well as Coleman-Weinberg Inflation are strongly disfavoured compared to Higgs Inflation. Natural Inflation and Loop Inflation both have a value of the Bayes factor that puts them right on the threshold between being moderately or strongly disfavoured compared to Higgs Inflation. Taking into consideration the uncertainty of ± 0.9 for both models, it becomes impossible to put them into one category, and it is therefore concluded that the two models are moderately to strongly disfavoured compared to Higgs Inflation.

Comparing our results with Ref. [189], we find that they are in qualitative agreement regarding which models that are strongly disfavoured compared to Higgs Inflation. We note that the values for the Bayes factor found here are systematically more negative than the corresponding values in [189], and in most cases deviate more than the estimated statistical uncertainty. We stress that this is *not* due to problems related to the CONNECT emulation, but is most likely related to small differences in the prior volume and/or parameterisation.

Ref. [192] have also calculated the Bayesian evidence for different inflationary models using ASPIC and a neural network, but they have trained their neural network on the effective likelihood, where all non-inflationary parameters have already been integrated out. They have used some different data sets than us, and the priors are not the same. But it is still possible to compare our results with theirs for two models: Large Field Inflation with $p = 2$ and Natural Inflation (even though their prior on f is not identical to ours). They get $\ln \mathcal{B}_{\text{LFI}_2} = -7.35$ and $\ln \mathcal{B}_{\text{NI}} = -4.74$, which is in good agreement with our values seen in table 7.4.

7.6 RUNTIME CONSIDERATIONS

The main reason for calculating Bayesian evidence with CONNECT instead of CLASS is that it is much faster even though we first have to train

$ \ln \mathcal{B} $	Odds	Probability	Strength of evidence
<1.0	$<3:1$	<0.750	Inconclusive evidence
1.0	$\sim 3:1$	0.750	Weak evidence
2.5	$\sim 12:1$	0.923	Moderate evidence
5.0	$\sim 150:1$	0.993	Strong evidence

Table 7.5: The strength of the Bayesian evidence interpreted by using the Jeffreys scale. The threshold values for the odds are 3:1, 12:1, and 150:1, which represents weak, moderate and strong evidence respectively. The table is taken from Ref. [179].

a neural network for the model. This can clearly be seen by comparing the time it took to calculate the Bayesian evidence in section 7.3 with CLASS and the time it took to train the neural network and calculate the evidence with CONNECT respectively. The calculation of the Bayesian evidence using CLASS took $\sim 30,000$ CPU-hours on Intel Xeon E5-2680 v2 CPUs, whereas the calculation of the evidence with CONNECT (for $\Lambda\text{CDM}+\alpha_s+r$) took only ~ 125 CPU-hours on Intel Xeon Gold 6230 CPUs. The difference in hardware might have a small effect, but it is most likely not more than a factor of ~ 2 . The training of the neural network (including sampling and calculation of training data) took ~ 150 CPU-hours, so even with this included, the calculation of the Bayesian evidence is still much faster with CONNECT than with CLASS. Furthermore, the evidence for the different inflationary models all took less than ~ 3500 CPU hours combined to calculate with CONNECT and 1200 live points, which is considerably less than what was required for $\Lambda\text{CDM}+\alpha_s+r$ with CLASS despite the inflationary models being more complicated as well as having 4 times as many live points.

When calculating the Bayesian evidence using CLASS, the dominant part of the calculation is the evaluation of CLASS itself. By using CONNECT instead, the evidence can be calculated without evaluating any of the hundreds of coupled differential equations in CLASS, and the limiting factor therefore becomes the Planck likelihood. To train the neural network using CONNECT, CLASS still needs to calculate the Einstein-Boltzmann equations, but the number of times CLASS is called during the training is much less than the number of times it is called when calculating the evidence without a neural network. When using CLASS to calculate training data for the neural networks, the total number of evaluations is $\sim 50,000$, which is 30 times fewer evaluations than the POLYCHORD run using CLASS.

In summary, performing an evidence calculation using CLASS takes $X \sim 30,000$ CPU-hours, while an evidence calculation using CONNECT requires two steps: First, training data is generated iteratively and the neural network is trained and then the evidence calculation proceeds. The first step takes $X_1 \sim 150$ CPU-hours, while the second step takes $X_2 \sim 125$ CPU-hours. Because the model can be reused for different inflationary models and different dataset combinations, the combined run-time for m models and d dataset combinations will be $X_{\text{tot}} = m \times d \times X$ when using CLASS but only $X_{\text{tot}} = X_1 + m \times d \times X_2$ when using CONNECT. Thus, if we consider the case where we are computing Bayes

factors for a handful of models with a few dataset combinations, the training time becomes completely negligible, and CONNECT delivers a speedup factor of ~ 240 compared to CLASS.

7.7 DISCUSSION AND CONCLUSIONS

We have tested the use of the CONNECT framework for calculating Bayesian evidences in cosmology using inflationary models as a test case. CONNECT has previously been shown to emulate cosmological observables at a level of precision more than adequate for performing Bayesian parameter inference and for computing profile likelihoods. However, since the calculation of Bayesian evidence typically puts more weight on regions of parameter space in which the likelihood is only moderately good it cannot *a priori* be assumed that CONNECT delivers suitable precision for this task.

Using the standard set of “observational” parameters describing slow-roll inflation models, A_s, n_s, α_s, r , we found that running POLYCHORD with default settings through MONTEPYTHON leads to severe under-sampling of the nuisance parameter space when we use CONNECT rather than CLASS. We traced this problem to a default setting in the POLYCHORD wrapper which splits parameters into “slow” (cosmological) and “fast” (nuisance) parameters, and devotes 0.75 of the wall time to sampling the slow parameter space. When running POLYCHORD with CLASS this leads to a suitable division of labour between slow and fast parameters. However, when run with CONNECT it leads to the mentioned under sampling of nuisance parameters and poor convergence of the computation. In fact, the CONNECT-based runs typically required an order of magnitude more live points to achieve the same precision as the CLASS-based runs.

To fix the problem we ran POLYCHORD with all parameters treated equally (i.e. no splitting into “slow” and “fast” parameters) and found that results become compatible with CLASS-based results with the same number of live points, thus validating that CONNECT can replace the use of CLASS for evidence computations. This in turn reduced runtime tremendously with the evidence calculations now being completely dominated by the likelihood calls.

Having validated the CONNECT framework for this purpose we then proceeded to calculate Bayesian evidence for a number of slow-roll inflationary models by using the ASPIC library to convert inflationary

parameters to observable inflationary parameters. We found evidence ratios between models very similar to those reported in Ref. [189] and in all cases within the same evidence strength brackets. Furthermore, all the calculations of the Bayesian evidence was done with 1200 live points and on 24 tasks with 1 CPU for each task, and the calculations were done within 24 hours. Using a neural network therefore drastically reduces the runtime for these calculations, making it possible to easily use Bayesian evidence as a tool to compare different theoretical models.

Based on the tests carried out and presented here we are therefore confident that CONNECT can be used for calculations of Bayesian evidence in cosmology, vastly reducing the often prohibitive runtimes of such calculations. This will make it possible to start using Bayesian evidence as a tool in theoretical cosmology. Right now, theoretical cosmology is mostly done using Bayesian and frequentist parameter estimation, but it will now be possible to also use Bayesian evidence and thereby compare how good one cosmological model is compared to another cosmological model.

Reproducibility. We have used the publicly available CONNECT framework available at https://github.com/AarhusCosmology/connect_public to create training data and train neural networks. To calculate the Bayesian evidence, we have used the publicly available program POLYCHORD available at <https://github.com/PolyChord/PolyChordLite> as well as the program MONTEPYTHON publicly available at https://github.com/brinckmann/montepython_public. Lastly, we have used the program ASPIC to calculate the inflationary models and their fundamental parameters. This has been done with the publicly available PYTHON wrapper PYASPIC available at <https://github.com/AarhusCosmology/PyAspic>. ASPIC is publicly available at <http://cp3.irmp.ucl.ac.be/~ringeval/upload/patches/aspic/>.

ACKNOWLEDGEMENTS

We thank Jérôme Martin, Christophe Ringeval, and Vincent Vennin for valuable comments on the manuscript, and Will Handley for fruitful discussions on POLYCHORD. Furthermore, we acknowledge the use of computing resources from the Centre for Scientific Computing Aarhus (CSCAA). A.N. and T.T. was supported by a research grant (29337)

Case	Bayesian evidence ($\log \mathcal{Z}$)	“slow”/“fast” likelihood calls
POLYCHORD with CLASS	-1907.4 ± 0.92	398,478 / 25,689,051
POLYCHORD with CONNECT	-1898.2 ± 1.43	469,336 / 329,299

Table 7.6: The Bayesian evidence, as well as the number of likelihood evaluations, for the LFI_4 model calculated with POLYCHORD using both CONNECT and CLASS, both using the standard MONTEPYTHON settings for POLYCHORD in which parameters are split in “slow” and “fast” categories and 0.75 of the total wall time is spent integrating the slow parameter space. This test run was performed using 100 live points.

from VILLUM FONDEN. C.S. and S.H. were supported by a grant from the Danish Research Council (FNU).

APPENDIX 7.A: CONVERGENCE ISSUES

In this appendix we validate the use of POLYCHORD with CONNECT and discuss convergence in terms of the number of live points. As we discussed in section 7.3, the default setting for POLYCHORD in MONTEPYTHON leads to severe undersampling of the nuisance parameter space when using CONNECT⁴. This undersampling leads to a bias in the ensemble mean log-evidence, unless a very large number of live points is being used, as shown in figure 7.2.

That the nuisance parameter space becomes under sampled with standard settings is very evident from table 7.6 in which it can be seen that even though the number of evaluations in the slow parameters are comparable in the two cases, the number of evaluations in the fast parameters are a factor of 80 smaller when using CONNECT.

Once diagnosed this problem can be easily fixed by disabling the oversampling feature in `PolyChord.py` and treating all variables democratically. Running POLYCHORD with CONNECT using these settings

⁴ There are other situations where the default behaviour can be sub-optimal, see e.g. the issue at https://github.com/brinckmann/montepython_public/issues/374.

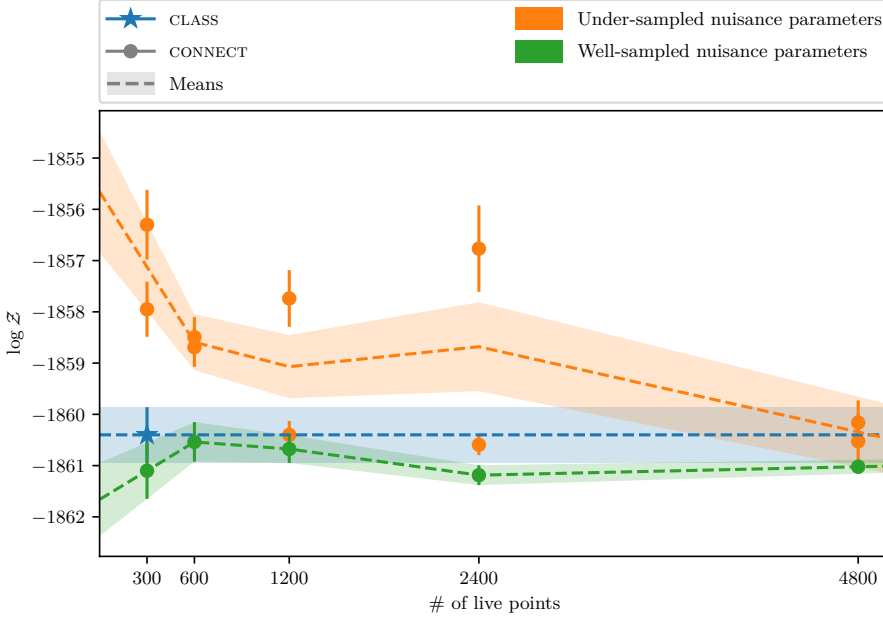


Figure 7.2: Evidence calculation of the the phenomenological (A_s, n_s, α_s, r) -model using POLYCHORD. We compare CLASS, CONNECT with default MONTEPYTHON, and CONNECT with our corrected MONTEPYTHON. Without the fix, 4800 live points are needed to obtain a converged result, whereas CLASS already seems converged when using 300 live points. With the fix to MONTEPYTHON, CONNECT is in agreement with CLASS and is no longer biased.

produces a Bayesian evidence of $\log \mathcal{Z} = -1906.2 \pm 0.9064$ using a total of 2,035,411 likelihood evaluations.

In order to further test convergence of POLYCHORD with both standard and “new” settings we have performed a series of test runs for the the phenomenological (A_s, n_s, α_s, r) -model, varying the number of live points. The results are shown in figure 7.2 from which we can conclude that CONNECT with standard POLYCHORD settings requires (at least) 4800 live points to achieve the same precision as CLASS-based runs with 300 live points. With the fix in place the CONNECT-based runs converge as quickly as the CLASS-based runs in terms of number of live points, but using a smaller total number of likelihood evaluations.

IMPROVEMENT OF TRAINING DATA USING HYPERSPHERE SAMPLING

As stressed several times during this thesis, proper selection of training data is vital for a good emulator. If the training data does not accurately represent the relevant parts of the parameter space, the emulator cannot be expected to perform well in the regions where it has little to no training data. While Latin hypercubes have been a standard for choosing training data in cosmological emulation, this is not very efficient when large parts of the parameter space are irrelevant to the cosmological analysis in question. The iterative sampling process of CONNECT’s active learning scheme solves this problem, but the process of collecting training data can be a bit slow, especially in cases where the initial points (from a sparse Latin hypercube) are poorly chosen.

A way to accommodate this problem such that the training data is more localised around the region of interest while simultaneously being easily computed is described in the following paper, which is presented in this chapter in its entirety:

- *Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. “Cutting corners: hypersphere sampling as a new standard for cosmological emulators.” In: JCAP 10 (2024), p. 073. doi: 10.1088/1475-7516/2024/10/073. arXiv: 2405.01396 [astro-ph.CO].*

This paper presents the idea of hypersphere sampling which had already been discussed to some extent in the literature. We implemented a new algorithm for efficiently obtaining a uniform hypersphere of training data points with the possibility of transforming the points using a covariance matrix if one is available. This way, one is rewarded with additional efficiency if extra information is available, but still without any known correlations, the hypersphere proves superior to a Latin hypercube for inference purposes.

This has been implemented as the new standard for the initial training data in CONNECT’s iterative sampling of training data, and it has greatly improved the convergence of the iterations.

————— Beginning of reference [4] —————

CUTTING CORNERS: HYPERSPHERE SAMPLING AS A NEW STANDARD FOR COSMOLOGICAL EMULATORS

Andreas Nygaard^a, Emil Brinch Holm^a, Steen Hannestad^a, Thomas Tram^a

^aDepartment of Physics and Astronomy, Aarhus University, DK-8000 Aarhus C, Denmark

Abstract. Cosmological emulators of observables such as the Cosmic Microwave Background (CMB) spectra and matter power spectra commonly use training data sampled from a Latin hypercube. This method often incurs high computational costs by covering less relevant parts of the parameter space, especially in high dimensions where only a small fraction of the parameter space yields a significant likelihood.

In this paper, we make use of hypersphere sampling, which instead concentrates sample points in regions with higher likelihoods, significantly enhancing the efficiency and accuracy of emulators. A novel algorithm for sampling within a high-dimensional hyperellipsoid aligned with axes of correlation in the cosmological parameters is presented. This method focuses the distribution of training data points on areas of the parameter space that are most relevant to the models being tested, thereby avoiding the computational redundancies common in Latin hypercube approaches.

Comparative analysis using the CONNECT emulation tool demonstrates that hypersphere sampling can achieve similar or improved emulation precision with more than an order of magnitude fewer data points and thus less computational effort than traditional methods. This was tested for both the Λ CDM model and a 5-parameter extension including Early Dark Energy, massive neutrinos, and additional ultra-relativistic degrees of freedom. Our results suggest that hypersphere sampling

holds potential as a more efficient approach for cosmological emulation, particularly suitable for complex, high-dimensional models.

8.1 INTRODUCTION

Using machine learning techniques to emulate observables such as CMB spectra or matter power spectra predicted by a cosmological model has become increasingly popular in recent years, mainly due to the high computational cost of directly computing the observables using standard tools (e.g. Einstein–Boltzmann solvers or N -body codes). For example, computing Bayesian evidence ratios between different cosmological models typically requires calculating observables in millions of points in the space of cosmological parameters, and using e.g. standard codes such as CLASS [27] or CAMB [159] therefore leads to millions of CPU core-seconds being consumed.

This CPU demand can be reduced by orders of magnitude using emulators, often without sacrificing precision. However, there is still a very substantial computational cost related to generating training data, i.e. the predicted observables at each point in the cosmological parameter space, for the emulator. This means that it is important that the training data represent the parameter space well, and necessitates a good balance between the amount of training data points and the relevance of each point.

A simple choice is to use Latin hypercube sampling of training points on some predefined (prior) volume of parameter space. This has the advantage of being simple to implement and assigning equal weight to all regions within the hypercube. Examples of this method used in cosmological emulator training include e.g. the emulators of Refs. [103, 113, 157, 199]. One could, however, ask why the feature of equal weight to all regions within the prior volume is desired, since most of the volume in higher dimensions is in the corners of the hypercube. Points in such regions are almost always associated with a very poor likelihood and when using the emulator to calculate for example profile likelihoods, Bayesian parameter inference, or Bayesian evidence, the emulator’s ability to accurately calculate observables in these corner regions is wasted. This leads to a very inefficient use of training data and puts substantially higher demands on the number of points in the training data. Some emulators, such as Refs. [122, 158, 200, 201], have circumvented this by using Gaussian processes (GP) instead of artificial neural networks. The

idea behind this is to let the acquisition function of the GP decide which new point to include in the training data based on where the emulation is uncertain or unexplored. The downside of this is the inferior scalability of GPs compared to neural networks, i.e. only a limited number of dimensions and amount of training data is feasible. Typically, Gaussian processes are therefore used in situations where calculating each individual training point becomes extremely expensive (a good example is emulation of observables based on N -body simulations).

In this paper, we illustrate that a better approach when dealing with neural networks is to sample uniformly in a hypersphere, which avoids the large amount of irrelevant points in the corners of the hypercube. This, however, requires prior knowledge about where the region of interest is located in the parameter space, but similar prior knowledge is also needed to construct a Latin hypercube. This kind of hypersphere (or hyperellipsoid) sampling has been used by Refs. [109, 133] but with a rather suboptimal way of sampling by rejecting points from a (Latin) hypercube that are outside the hypersphere. Ref. [103] used a normal Latin hypercube for the training data but sampled their validation data from a random distribution with an ellipsoidal mask, since rejection from a Latin hypercube was deemed too inefficient. In high dimensions, this becomes impossible since it requires a Latin hypercube too large to fit in the memory of a computer. There are, however, ways to effectively sample points within a hypersphere, if one omits the “Latin” requirement.

In section 8.2, we explore different sampling strategies and describe a novel, efficient method for sampling uniformly from a high-dimensional hypersphere. Next, we compare performances using the publicly available emulation tool CONNECT [2] in section 8.3, and give our conclusions and outlook in section 8.4.

8.2 SAMPLING METHODS FOR TRAINING DATA

When building an emulator of Einstein–Boltzmann solvers such as CLASS [27] or CAMB [159], one generates the training data by running the solver on a selected set of points in parameter space, giving corresponding pairs of cosmological parameters and their corresponding observables at each point. This leaves open the choice of the set of parameter space points at which to generate the data. There are different ways to sample training data depending on what the objective of the

network is. Ultimately, a neural network is only as good as its training data, and so a region of sparse data leads to the network having to interpolate over larger distances in this part of the parameter space, while regions of dense data leads to very accurate emulation in these regions. One's choice of training data might depend on various factors such as the number of points one is willing to compute (if each point requires expensive computations such as N -body codes, it might not be many) or whether or not the network should be more precise in some regions than others.

In this section, we will go through different ways one might sample data from a hypercube and a hypersphere, and present some of the strengths and drawbacks of each.

8.2.1 LATIN HYPERCUBE SAMPLING

The idea behind Latin hypercube sampling [202] is to create a set of data that is close to uniformly distributed throughout a hypercube without requiring a dense grid of points that scales exponentially with the dimensionality. When sampling a Latin hypercube of N points, each dimension is split into N even segments, and the points are then placed within the resulting N^d cells, where d is the dimensionality, in a way that ensures that only a single cell in a set of d rows is occupied by a point.

The Latin hypercube sampling does not, however, guarantee uniform sets of training data, given that having all points on a diagonal line also constitutes a Latin hypercube, but in practice, one will always get something very close to uniformity for a large amount of data points ($N > 10^3$). A variant of Latin hypercube sampling called *orthogonal sampling* furthermore ensures uniformity in the hypercube by splitting the cube into smaller segments [203]. This will result in uniformity on large scales (similar to the size of the Latin hypercube), but points on smaller scales tend to look more randomly distributed. This is due to most implementations of Latin hypercube sampling placing points randomly within the N^d cells [204]. Orthogonal sampling is more complex than Latin hypercube sampling and the added guarantee is not worth the extra layer of complexity, since Latin hypercubes virtually always produces a set of points that are close to uniformly distributed on large scales.

Having a Latin hypercube as training data thus ensures that the resulting neural network will be equally precise in all parts of the parameter space, which is beneficial if one wants to remain completely agnostic with respect to cosmological models and data. A drawback when sampling from a hypercube, however, is that the density of points around the region of interest (when computing the likelihood function) becomes very small in higher dimensions. Most of the volume is in the corners of the hypercube in higher dimensions, and this leads to a very sparse sampling for all feasible numbers of points, N . This means that possible features in the best-fit region are not resolved very well. A neural network trained on a hypercube of points still gains information from the outermost points, but it usually requires many more epochs of training (Ref. [157] reports 50,000 epochs for 10^4 points) in order to extract all the required information to perform well in the region of interest. Hence, only very smooth likelihoods can be accurately represented by this sparse sampling. If the likelihood is highly non-Gaussian, the information from the sparsely sampled points might be insufficient for accurate emulation.

8.2.2 LATIN HYPERSPHERE SAMPLING

This issue of the Latin hypercube can be circumvented by concentrating the sampling inside a hypersphere centred around the (approximate) best-fitting set of parameters. However, whereas the construction of the Latin hypercube is trivial, the procedure of sampling from a Latin hypersphere can be challenging in high dimensions. To illustrate, for generating a Latin hypersphere of N points, one might naively think that a good solution is to generate a Latin hypercube of M points, where $M = N \times r_d$ and r_d is the ratio of the volume of a d -dimensional hypercube to that of an inscribed hypersphere [2],

$$r_d = \frac{V_d^{\text{cube}}}{V_d^{\text{sphere}}} = \frac{2^d \Gamma(\frac{d}{2} + 1)}{\pi^{d/2}}, \quad (8.1)$$

and then reject all points with a Euclidean distance larger than 1 (radius of the hypersphere) to the centre. This is not ideal, however, since r_d grows to very large values for higher dimensions as seen in figure 8.1. For example, if one needs to sample 10^4 points in a 15-dimensional hypersphere, the required Latin hypercube would take up ~ 50 GB of

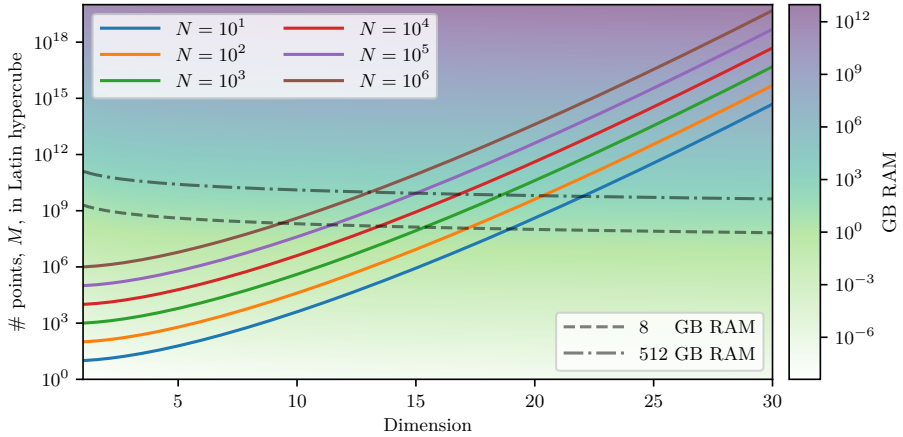


Figure 8.1: The figure shows how the ratio between the volumes of a hypercube and its inscribed hypersphere grows with higher dimensionality. The number of points, M , needed in a Latin hypercube in order to have N points within the hypersphere is depicted as a function of dimensionality for various values of N . The background colours indicate the minimum required RAM in order to store a Latin hypercube of M points of a certain dimension in memory (single precision), and two specific values of 8 GB and 512 GB have been highlighted by the dashed and dash-dotted lines, respectively.

memory. In practice, this makes it very unfeasible to go beyond 10 dimensions, and outright impossible to go beyond 15 dimensions.

However, since Latin hypercubes appear random on small scales, we might not need to enforce the Latin criterion on the sphere. If we can sample enough uniformly random points in the hypersphere, the density of points close to the best-fit region is still much greater than for any feasible Latin hypercube, and this set of points contains much more relevant information that can be extracted by a neural network in significantly fewer epochs during training. The question is then how to sample from a uniform hyperspherical distribution.

8.2.3 RANDOM UNIFORM SAMPLING FROM A HYPERSPHERE

A way to uniformly sample from a hypersphere is to sample from another isotropic distribution and transform the points to a sphere afterwards [205]. A standard multivariate normal distribution (i.e. a multivariate normal distribution with the identity covariance matrix $C = \mathbb{I}$ and zero mean $\mu = \mathbf{0}$) is one such isotropic distribution. Hence, if we

N	=	number of points to sample	
d	=	dimensionality	
S	=	<code>Normal(num=N, dim=d)</code>	normally distributed points
R	=	$\sqrt{\text{sum}_d(S^2)}$	distances to the centre
Π	=	<code>RandomUniform([0, 1], N)</code>	sample $\Pi(r)$ uniformly
R'	=	$\Pi^{1/d}$	compute new radii
S	=	$S \times R' / R$	rescale points to sphere

Algorithm 8.1: Pseudo-code for random uniform sampling directly from a hypersphere. A similar algorithm is presented in Ref. [205].

sample N points from a d -dimensional standard normal distribution and divide the coordinates of all points with their Euclidean distance from the centre, we obtain a sample of points uniformly distributed on the surface of a d -dimensional hypersphere with radius 1. We then need to distribute the points evenly throughout the hypersphere by multiplying the points by new radii. These new radii should be sampled from a non-uniform distribution in the interval $[0, 1]$ in order to account for more points required in the outer parts compared to around the centre. Each hyperspherical shell needs to be weighted by the volume in that shell (scaling as r^{d-1}), which means that we need to sample from the distribution $\pi(r) = d r^{d-1}$, where the dimension constitutes a normalisation factor. The Probability Integral Transformation [206] implies that the cumulative distribution function of $\pi(r)$ is uniformly distributed between 0 and 1, and we thus get

$$\Pi(r) = \int_0^r \pi(x) dx = \int_0^r d x^{d-1} dx = r^d \sim \mathcal{U}(0, 1). \quad (8.2)$$

This means that we can just sample $\Pi(r) = r^d$ uniformly and then take the d^{th} root of the samples in order to get the distribution of radii. This is summarised in the pseudo-code depicted in algorithm 8.1.

8.2.4 SAMPLING NEAR BOUNDARIES

In some cases we cannot sample from the entire hypersphere: If a parameter has a (physically motivated) hard boundary slicing the hypersphere, e.g. a non-negative particle mass, we cannot allow points outside of this boundary. In this case, the parameter space boundaries can be enforced through rejection sampling. One would then first sample from the hypersphere as if all of the parameter space is allowed, and then reject all points outside the boundaries. In practice, we implement this rejection sampling as a Python generator that maintains a cache of points generated using algorithm 8.1.

Contrary to the rejection sampling described in section 8.2.2, this is not expensive memory-wise since we can reject points on-the-fly. However, it might become computationally expensive if the boundaries of any parameter are such that only a thin slice of the hypersphere is allowed. In this case, the rejection rate would be close to 100%. However, this situation could be easily remedied by sampling uniformly from this thin slice along the parameter in question (good approximation for thin slices) while still sampling from a hypersphere in the other parameters.

8.2.5 HYPERELLIPSOID WITH CORRELATIONS

Finally, when sampling from a hypersphere, one will obviously need to scale the dimensions to fit the parameters (like one would scale a Latin hypercube), thus turning the hypersphere into a hyperellipsoid. This ellipsoid is uncorrelated in all parameters by construction, and in most cases this will be a very good sample, as we will see in section 8.3. We can, however, use additional information (if available) about correlations between parameters to significantly improve the performance by sampling along the known directions of correlation. In order to include correlations, we transform the sampled points from algorithm 8.1 using the *Cholesky transformation* [207, 208]. With the prior knowledge of the parameter correlations stored in the covariance matrix C , the lower triangular matrix L that satisfies $C = LL^T$ is determined from the Cholesky decomposition. From this, a transformed (correlated) point, \tilde{p} , is computed as the matrix multiplication $\tilde{p} = Lp$, where p is the uncorrelated point. This procedure, along with the rejection of points outside the parameter bounds described in section 8.2.4, is summarised in algorithm 8.2. It is usually better to start with a known Λ CDM covariance matrix and then treat other parameters as uncorrelated if

N = number of points to sample d = dimensionality B = bounding box C = covariance matrix M = buffer size larger than N L = CholeskyDecomposition(C) $points = \{ \}$ while Size($points$) < N do P = Hypersphere(M, d) P = MatrixMult(L, P) $P_{ac} = \{p \in B, \forall p \in P\}$ ⟨append P_{ac} to $points$ ⟩ $points$ = select N from $points$	<i>parameter boundaries</i> <i>buffer for rejection</i> <i>lower triangular matrix</i> <i>initialise set of points</i> <i>M points from alg. 8.1</i> <i>transform points</i> <i>reject outside boundaries</i> <i>keep only N points</i>
--	--

Algorithm 8.2: Pseudo-code for random sampling in a correlated hyperellipsoid. This procedure uses a covariance matrix to transform points computed by algorithm 8.1 to reflect correlations of the parameters.

no information is available about their correlations than to not use any correlations at all. This will be explored in section 8.3.

8.3 COMPARISONS USING CONNECT

In order to show the benefits of hypersphere sampling, we have tested this against the more conventional approach of using Latin hypercubes. We use the CONNECT framework¹ [2] to sample training data and train our neural networks. The sampling of training data is done using the iterative approach of CONNECT, where an initial neural network is trained on sparse uniformly distributed points (Latin hypercubes up until now) and then used to sample new points using MCMC. This continues until a set of training data points, representative of the likelihood, is built. The new implementation allows for a different sampling of the points for

¹ Publicly available at https://github.com/AarhusCosmology/connect_public.

	Lower boundary	Upper boundary
ω_b	0.014	0.039
ω_{cdm}	10^{-11}	0.25
H_0	30	120
$\log(10^{10} A_s)$	1	5
n_s	0.7	1.3
τ_{reio}	0.01	0.4
f_{EDE}	10^{-11}	0.3
$\log_{10}(z_c)$	3	4.3
θ_i^{scf}	0.1	3.1
m_{nCDM}	0.02	10
N_{ur}	0	6

Table 8.1: Lower and upper boundaries of the cosmological parameters. These are used for the initial Latin hypercubes and hyperspheres, and they are also enforced during the MCMC samplings. The first 6 parameters constitute the Λ CDM model, while all 11 parameters constitute the EDE+ M_ν + N_{ur} model.

the initial model than the regular Latin hypercube sampling. Instead, one can now use a uniformly sampled hypersphere, as presented in section 8.2, as a starting point for the iterative process. In this section we will explore how well the initial neural networks (using hyperspheres or Latin hypercubes) as well as the networks from the final iteration of the iterative sampling emulate the output from CLASS. We present results from two different cosmological models, i.e. the standard 6-parameter Λ CDM model and a 5-parameter extension with Early Dark Energy (EDE) [209–211], additional ultra-relativistic degrees of freedom N_{ur} and massive neutrinos M_ν to really challenge both the CONNECT framework as well as the two methods of initial sampling. For each cosmological model, we compare the following initial configurations:

- **HS (correlated):** Hypersphere with 1,000 points with correlations from a converged MCMC run.

- **HS (uncorrelated):** Hypersphere with 1,000 points with no correlations.
- **HS (Λ CDM correlated):** Hypersphere with 1,000 points with Λ CDM correlations and no correlations for the extended parameters (only applies to the $EDE+M_\nu+N_{\text{ur}}$ runs).
- **LHC (Small):** Latin hypercube with 1,000 points.
- **LHC (Medium):** Latin hypercube with 10,000 points.
- **LHC (Large):** Latin hypercube with 100,000 points.

The boundaries in the parameter space can be seen in table 8.1. All networks have been trained for 500 epochs with a batchsize of 256, and otherwise the same hyperparameters as in Ref [2]. The training data from the first iterations (using the initial neural networks) has been discarded for all Latin hypercubes (standard setting in CONNECT), since it is usually far from the best-fit region and therefore contaminates our total set of training data, while data from all iterations has been kept for the hyperspheres. For the subsequent MCMC runs using the neural networks, we have employed a data set consisting of:

- Planck 2018 high- ℓ $TTTEEE$, low- ℓ $TT+EE$, and lensing [21, 127].
- Baryon Acoustic Oscillations (BAO) measurements from BOSS DR12 [70], the main galaxy sample of SDSS DR7 [72] and 6dFGS [71].
- Pantheon supernova data [212].

The training data is gathered using the marginalised Planck 2018 high- ℓ $TTTEEE$ Lite likelihood due to its rapid evaluation time along with low- ℓ $TT+EE$. This data set is less constraining and always produces adequate training data [2] when the final data set includes the full Planck likelihood. For each MCMC, we have run 6 chains using MONTEPYTHON [63, 101], considering the runs to be converged when the Gelman-Rubin statistic fulfils $R - 1 < 0.01$. Some of the MCMC runs for the $EDE+M_\nu+N_{\text{ur}}$ model using the initial neural networks trained on either hyperspheres or Latin hypercubes have difficulties converging, and was stopped when the number of accepted points were similar to the converged runs.

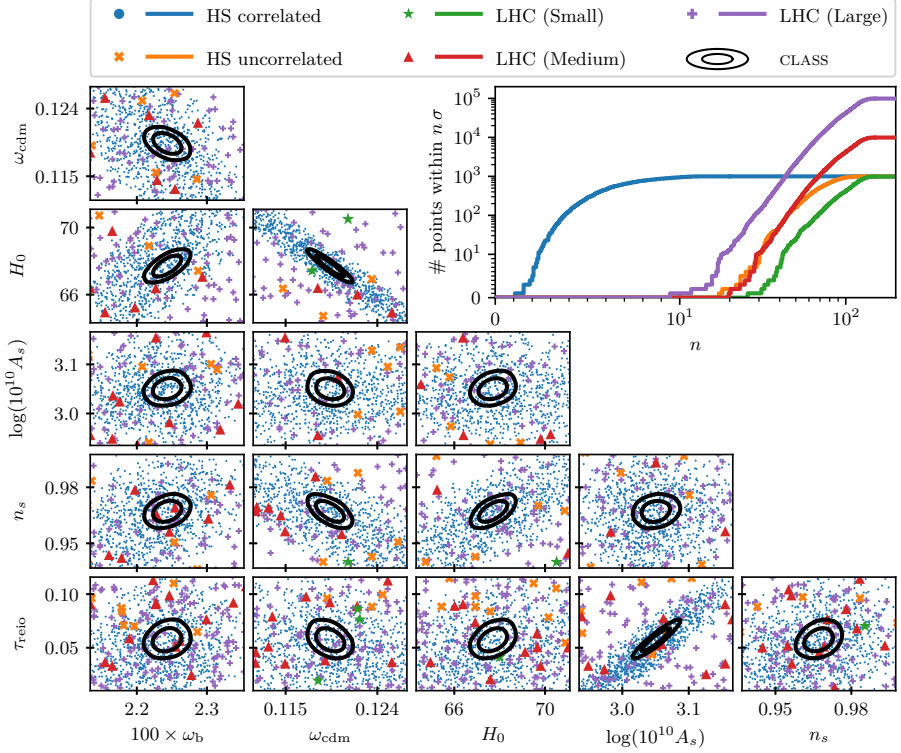


Figure 8.2: The triangle plot shows the point distributions of the various initial configurations in the Λ CDM model within 50 standard deviations of the best-fit point in all parameters along with the 2D posteriors when using CLASS. The top-right panel shows the number of points within n standard deviations of the best-fit point in all parameters as a function of n for the different configurations.

8.3.1 Λ CDM

The Λ CDM model has the parameter vector $\Theta = \{\omega_{\text{b}}, \omega_{\text{cdm}}, H_0, \log(10^{10} A_s), n_s, \tau_{\text{reio}}\}$ along with extra relativistic relics that fix $N_{\text{eff}} = 3.046$. This is often quite easy to sample because of its likelihood surface being almost perfectly Gaussian. Figure 8.2 highlights the differences in sampling density around the best-fit region between the various initial configurations. The triangle plot only includes points that are within 50 standard deviations (determined by an MCMC with CLASS) of the best-fit point in all parameters. It is evident that the correlated hypersphere has a much higher point density around the contours of the CLASS posterior, with the largest Latin hypercube with 100,000 points being

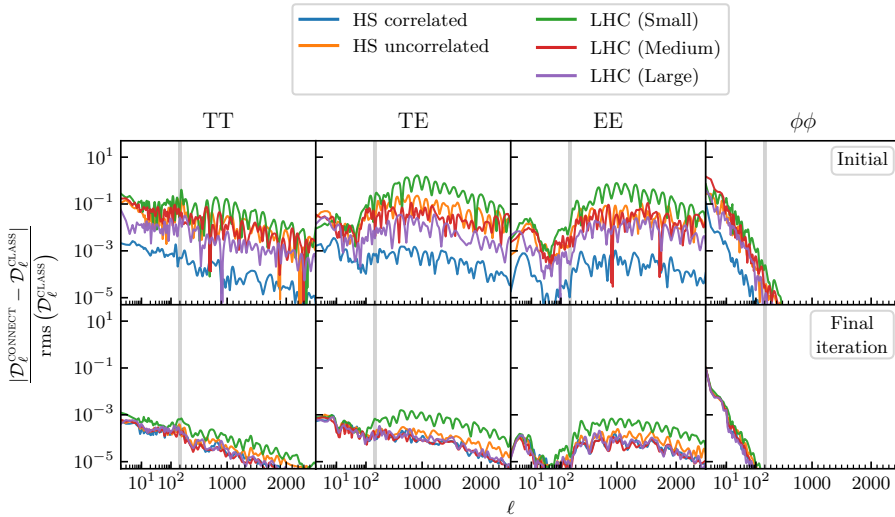


Figure 8.3: Errors of the neural networks emulating the Λ CDM model when emulating the CMB spectra of representative test data. The top panels show the errors of the initial models before the iterative process and the bottom panels show the errors of the networks from the final iterations of their respective runs. The test data is taken from a converged Λ CDM MCMC and is therefore the error around the best-fit and not an indicator of the training error. The lines correspond to a 95.45% confidence level where 95.45% of the computed points have errors beneath the lines.

second. It is also worth noticing that the uncorrelated hypersphere and the medium Latin hypersphere with 10,000 points have roughly similar point densities. This is also supported by the top-right panel, which shows the number of points within n standard deviations of the best-fit point in all parameters as a function of n for the different configurations, where the uncorrelated hypersphere and medium Latin hypersphere follow each other up until around $n = 50$. The limit of 50 standard deviations in the triangle plot was chosen such that the Latin hyperspheres would have around half of their points included. If we had chosen to include all points, the view of the best-fit region would have been obscured by the large amount of points from the Latin hyperspheres that are close to the best-fit in the 2 parameters of each 2D plane, but very far away in other parameters. The top-right panel also clearly shows how much closer to the best-fit point the points of the correlated hypersphere are. One would need a Latin hypersphere of many orders of magnitude more points to achieve the same density.

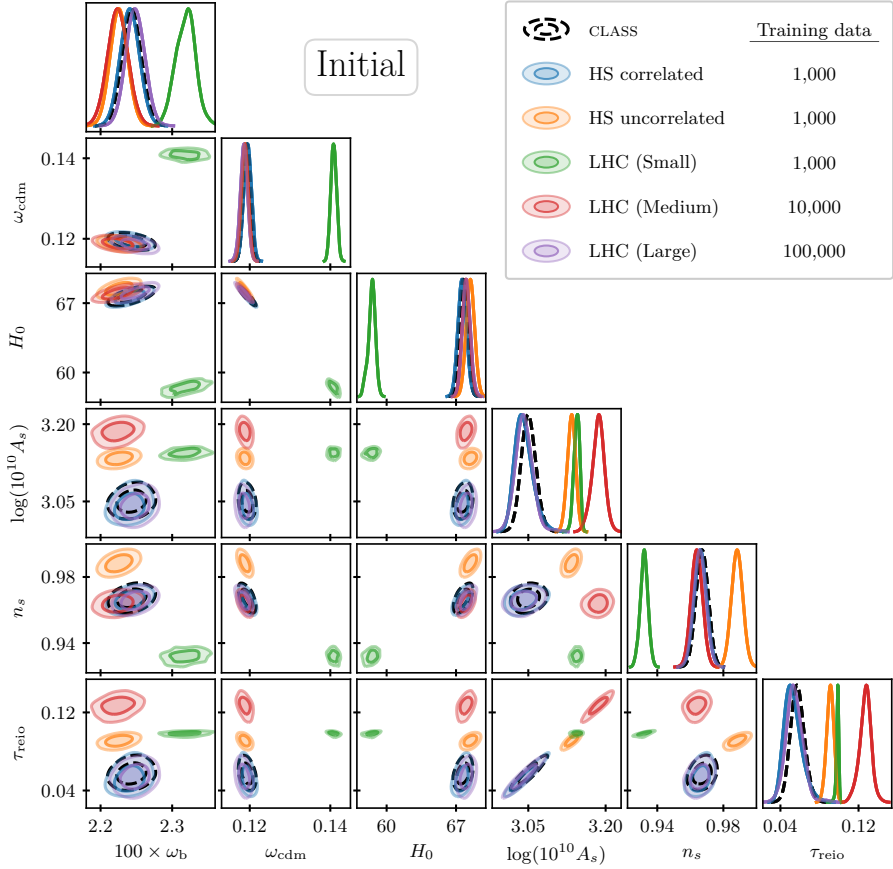


Figure 8.4: Posteriors from the MCMC runs using the initial neural networks emulating the Λ CDM model. A similar MCMC run using CLASS is also shown in black dashed lines as a reference. The table in the legend gives the number of training data points the respective networks used for the MCMCs were trained on.

Figure 8.3 shows, for each of the TT , TE , EE and $\phi\phi$ spectra, the difference in the $\mathcal{D}_\ell \equiv \ell(\ell+1)C_\ell/2\pi$ coefficients between CONNECT and CLASS relative to their root-mean-square values in CLASS. The top panel shows the 95.45% percentile of the error based on only the initial configurations while the bottom panel shows the same for the final iterations. It is apparent that a neural network trained on a correlated hypersphere outperforms all of the other initial networks in terms of preciseness around the best-fit region, since it has the most representable training data. In fact, in order to be as precise a hypersphere with

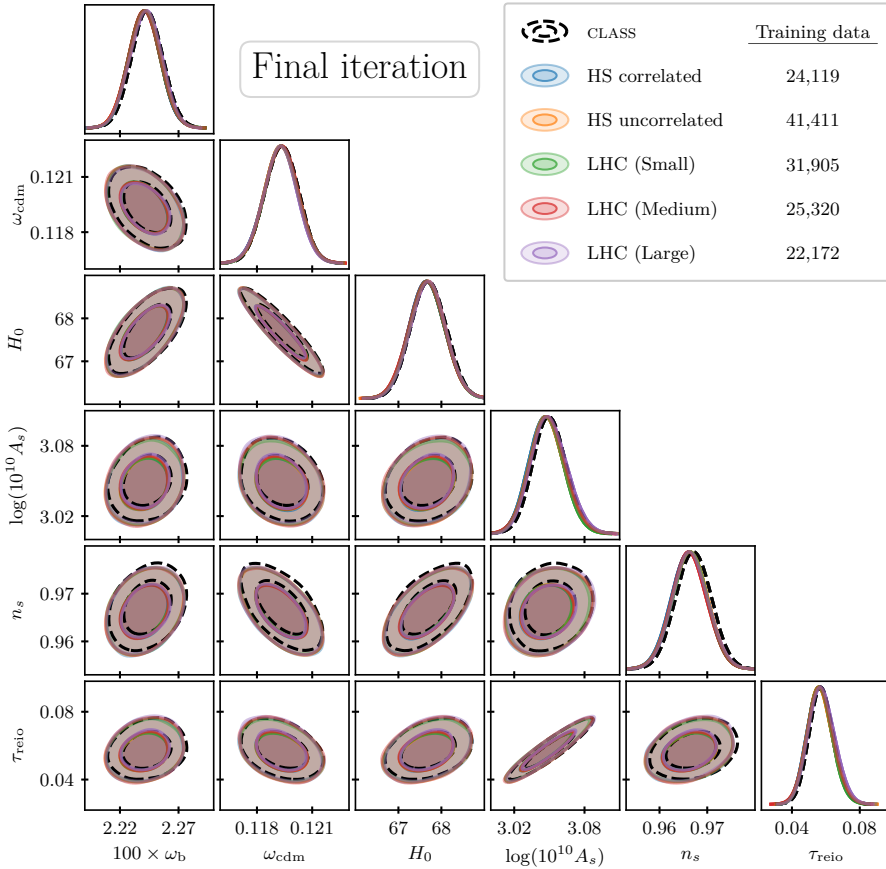


Figure 8.5: Posteriors from the MCMC runs using the neural networks emulating the Λ CDM model from the final iterations. A similar MCMC run using CLASS is also shown in black dashed lines as a reference. The table in the legend gives the number of training data points the respective networks used for the MCMCs were trained on.

only 1,000 points, a Latin hypercube would need more than 2 orders of magnitude more points (when restricting the training process to 500 epochs). Even if we lose the information of any correlations, the uncorrelated hypersphere seems to be just as good as a Latin hypercube with 10 times the amount of points, as also suggested by figure 8.2.

All of the runs, however, converge on good training data as can be seen in the lower panels of figure 8.3, where the precisions of the networks from all of the final iterations are shown. The only one that stands out as slightly less precise after the final iteration is the one with

an initial Latin hypercube of 1,000 points. This is most likely due to it having more training data away from the best-fit region. The initial network of this run is also worse than the rest, and it probably takes more iterations to locate the best-fit region than just the first one that is discarded afterwards. This leaves us with a lot of training data far from the region of interest, thus making the network less precise in order to accommodate the additional training data.

Figures 8.4 and 8.5 show the posteriors obtained from emulators trained on the initial and final configurations, respectively, along with posteriors obtained using CLASS. We can again see that the initial networks trained on a correlated hypersphere of 1,000 points and a Latin hypercube of 100,000 points perform similarly, while the initial network trained on an uncorrelated hypersphere of 1,000 points performs similarly to one trained on a Latin hypercube of 10,000 points. The small Latin hypercube of only 1,000 points is too sparse to give a good starting point, since its contours are far away from the rest. This is, however, not a problem when using the iterative approach, as is apparent from figure 8.5 where the posteriors from the final configuration emulators are seen to be nearly identical; instead, the downside of a poor initial configuration is that it requires more iterations and training data, reducing the total gain in computational efficiency from the emulation. One might think that the correlated hypersphere is very sensitive to the choice of centre point since it is much more narrow than the uncorrelated hypersphere or the Latin hypercube, but we have tested that the results are consistent with default settings even when offsetting the centre point as much as 10 standard deviations simultaneously in each parameter.

Table 8.2 shows the final number of iterations in each CONNECT training procedure along with how many CLASS evaluations have been used. The initial hypersphere and Latin hypercube data is always discarded and the first iteration consisting of 5,000 points is discarded for all Latin hypercubes. We can see that the run with the small Latin hypercube indeed takes more iterations, but the amount of CLASS evaluations is similar to the runs with the uncorrelated hypersphere and the medium Latin hypercube. Although the number CLASS evaluations is a good way to measure roughly how much CPU time is spent (since it is the slowest part of the sampling), there is an overhead from the MCMCs and training of each iteration that become more significant with several iterations. With a close-to-Gaussian likelihood like Λ CDM (or simple

	Iterations	CLASS evaluations
Correlated hypersphere	3	25,119
Uncorrelated hypersphere	3	42,411
Latin hypercube (Small)	5	37,905
Latin hypercube (Medium)	4	40,320
Latin hypercube (Large)	3	127,172

Table 8.2: Number of iterations and amount of CLASS evaluations (from both the initial sampling and the iterative process) in each Λ CDM CONNECT run. The initial data from hyperspheres and Latin hypercubes is always discarded and furthermore the first iteration of 5,000 points is discarded for all Latin hypercubes.

extensions), the configuration does not matter much for the final result when using relatively low precision settings (e.g. a low number of epochs), but one can significantly speed up the process and reduce the computational cost by using a hypersphere instead of a Latin hypercube.

8.3.2 EDE + M_ν + N_{ur}

This large extension model with a parameter vector of $\Theta = \{\omega_b, \omega_{\text{cdm}}, H_0, \log(10^{10} A_s), n_s, \tau_{\text{reio}}, f_{\text{EDE}}, \log_{10}(z_c), \theta_i^{\text{scf}}, m_{\text{ncdm}}, N_{\text{ur}}\}$ consists of the usual Λ CDM parameters, two massless neutrinos, a single neutrino with mass m_{ncdm} , additional ultra-relativistic degrees of freedom contributing a value N_{ur} to the amount of relativistic degrees of freedom in the early universe, and finally, an early dark energy (EDE) model [211]. The particular EDE model used here is the original axion-like model based on [209, 210], which involves an axion-like scalar field that is frozen at its initial field value θ_i^{scf} due to Hubble friction, until a redshift z_c , at which it rolls to the bottom of its potential, acting effectively as a fastly decaying fluid. Since it acts as a vacuum energy initially, its maximum fractional contribution to the energy budget, f_{EDE} , is realised at the decay time z_c . In the following, we use the implementation of the EDE model presented in [213]². This large, combined model

² Publicly available at https://github.com/mwt5345/class_ede.

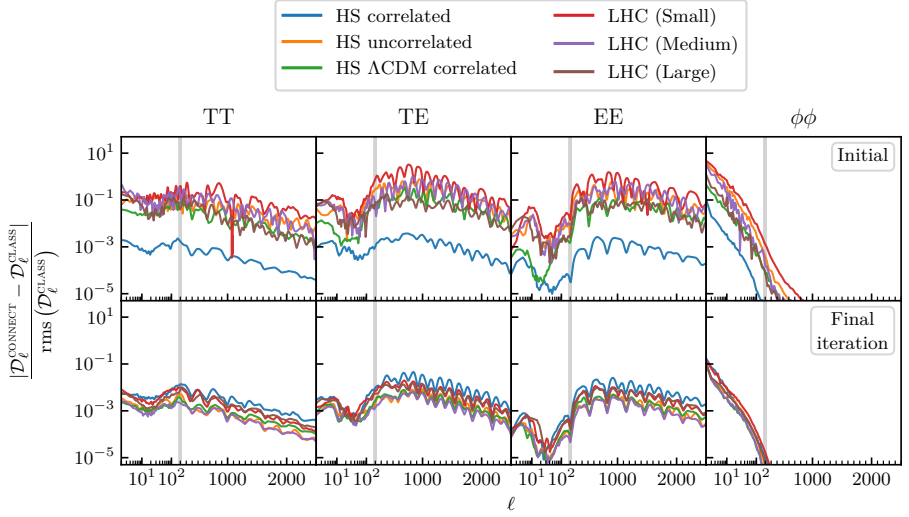


Figure 8.6: Errors of the neural networks emulating the $EDE+M_\nu+N_{\text{ur}}$ model when emulating the CMB spectra of representative test data. The top panels show the errors of the initial models before the iterative process and the bottom panels show the errors of the networks from the final iterations of their respective runs. The test data is taken from a converged $EDE+M_\nu+N_{\text{ur}}$ MCMC and is therefore the error around the best-fit and not an indicator of the training error. The lines correspond to a 95.45% confidence level where 95.45% of the computed points have errors beneath the lines.

was primarily chosen to showcase the potential of using hyperspheres instead of Latin hypercubes, and it also serves as a good test of the CONNECT framework.

The precision settings are the same as before with 500 epochs during training of the networks even though the MCMCs with this model take much more time to converge due to it being far from Gaussian in the extended parameters. Figure 8.6 shows the 95.45% percentiles of the errors in the CMB coefficients emulated by the initial configuration networks (top panel) and the networks from the final iterations (bottom panel) from CONNECT, compared to the values obtained directly from CLASS. It is clear from the figure that the initial neural network trained on a correlated hypersphere using the actual correlations of the model is much more precise than the rest of the initial networks within the best-fit region. Again we see a similar performance of the uncorrelated hypersphere and the medium Latin hypercube with 10,000 points, but in addition we also see a similar performance between the hypersphere

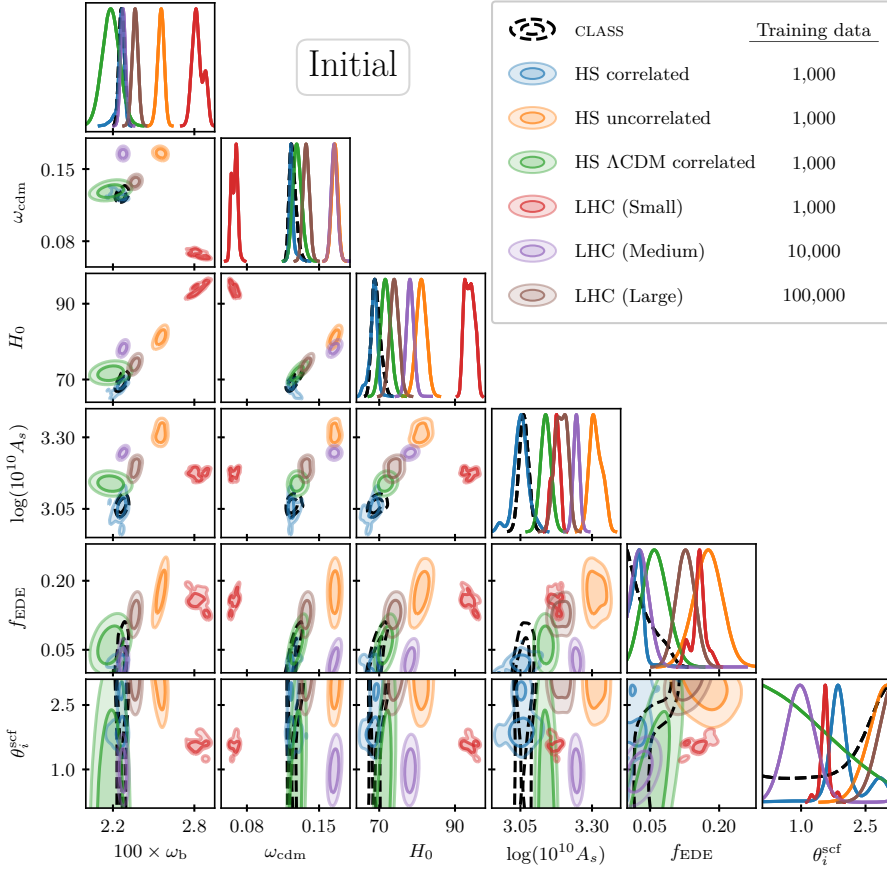


Figure 8.7: Posteriors from the MCMC runs using the initial neural networks emulating the $EDE+M_\nu+N_{\text{ur}}$ model. A similar MCMC run using CLASS is also shown in black dashed lines as a reference. The table in the legend gives the number of training data points the respective networks used for the MCMCs were trained on.

with only Λ CDM correlations (and no correlations in the extended parameters) and the large Latin hypercube with 100,000 points.

Figure 8.7 shows the posteriors from the initial neural networks of all runs. A mix between Λ CDM parameters and extended parameters have been chosen that best depicts how far some of the posteriors are from the best-fit region, since many of the extended parameters have significant posteriors all throughout their prior bounds. We clearly see that the correlated hypersphere has the most overlap with CLASS, although 1,000 points and 500 epochs is too small to correctly emulate

the model. The hypersphere with only Λ CDM correlations seems to be the one with the second most overlap, even though some correlations differ significantly from the Λ CDM model. This suggests that it is reasonable to use Λ CDM correlations if true correlations are not known beforehand, even though the model is significantly different. The reason for this is that the correlated hyperellipsoid is quite wide compared to the true posterior, so there is still a much higher density of points near the best-fit point compared to the uncorrelated hypersphere – even when using slightly wrong correlations. Extensions to the Λ CDM model might alter the correlations between the Λ CDM parameters, but usually not so much that this is not the case. The small Latin hypercube once again performs worse than the others, and it is much too sparse to discover any correlations in the model which is apparent from the $(H_0, \log(10^{10} A_s))$ –contour where all posteriors fall on the same line except that of the small Latin hypercube.

The figures 8.8 and 8.9 show the posteriors when sampling with the neural networks from the final iterations for hyperspheres and the Latin hypercubes, respectively. Only the extended parameters along with H_0 are shown, since all other posteriors are close to Gaussian and agree very well with CLASS for all runs. It is immediately clear that the posteriors calculated using the final iterations of the hyperspheres are much closer to the CLASS results than those calculated using the Latin hypercubes. It seems that using a Latin hypercube as initial training data (even with 100,000 points) for a complicated cosmological model does not produce the correct result with the default settings of CONNECT. Training for more epochs and collecting more data from each iteration will of course solve this, but this would also increase the computational costs significantly. The correlated hyperspheres (true correlations and Λ CDM correlations) are slightly more accurate than the one with no correlations, and this again suggests that slightly wrong correlations might be better than no correlations at all.

Indeed, the two correlated hyperspheres in figure 8.8 actually seem to agree quite well with CLASS and the subtle differences could be explained by the CLASS MCMC being slightly less converged. The curious “hole” in the $(\log_{10} z_c, \theta_i^{\text{scf}})$ –contour is also reproduced by the final iterations of the correlated hyperspheres and to a lesser extend also the uncorrelated hypersphere. This is not lack of convergence, since we expect this feature to be present (see e.g. [211, 213–215]).

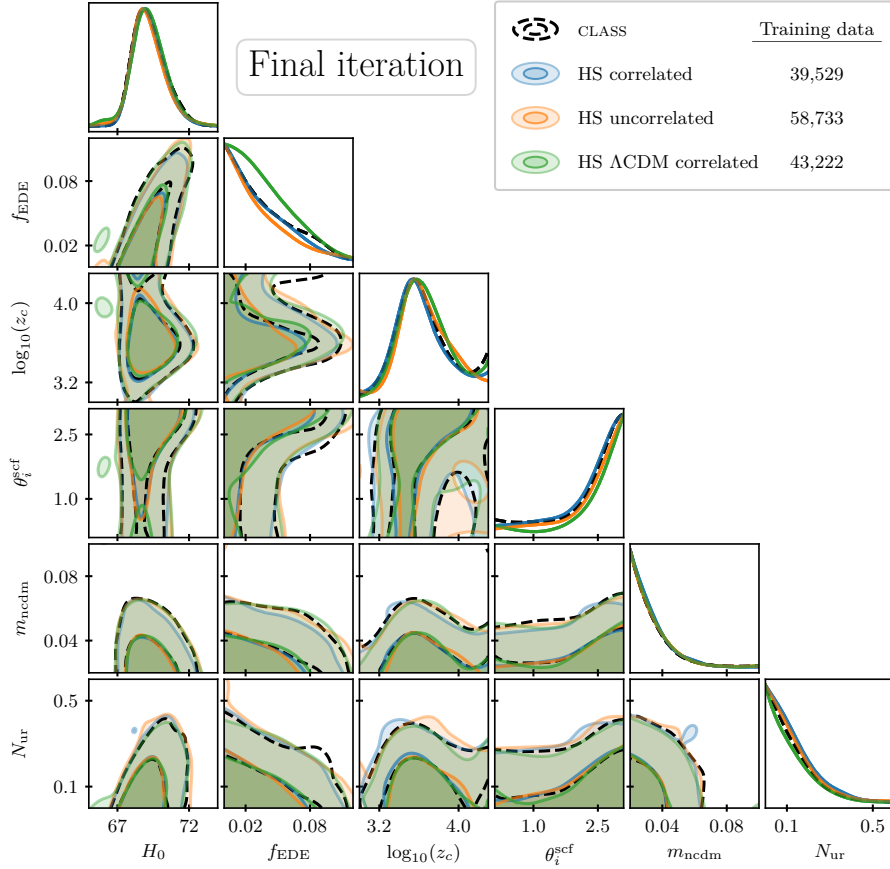


Figure 8.8: Posteriors from the MCMC runs using the neural networks emulating the $EDE+M_\nu+N_{\text{ur}}$ model from the final iterations of the hypersphere runs. A similar MCMC run using CLASS is also shown in black dashed lines as a reference. The table in the legend gives the number of training data points the respective networks used for the MCMCs were trained on.

Table 8.3 shows the number of iterations and total amount of CLASS evaluations from all these CONNECT runs, and it is apparent that the runs with the small and large Latin hypercubes took significantly more iterations. This typically indicates that the iterations “jump” around the parameter space and has difficulties homing in on the best-fit region.

Figure 8.10 illustrates the iterative training data sampling of CONNECT that has the small Latin hypercube (10^3 points) as initial configuration, in the (H_0, f_{EDE}) -plane. It is seen that the training data gathered by the small Latin hypercube run has difficulties converging,

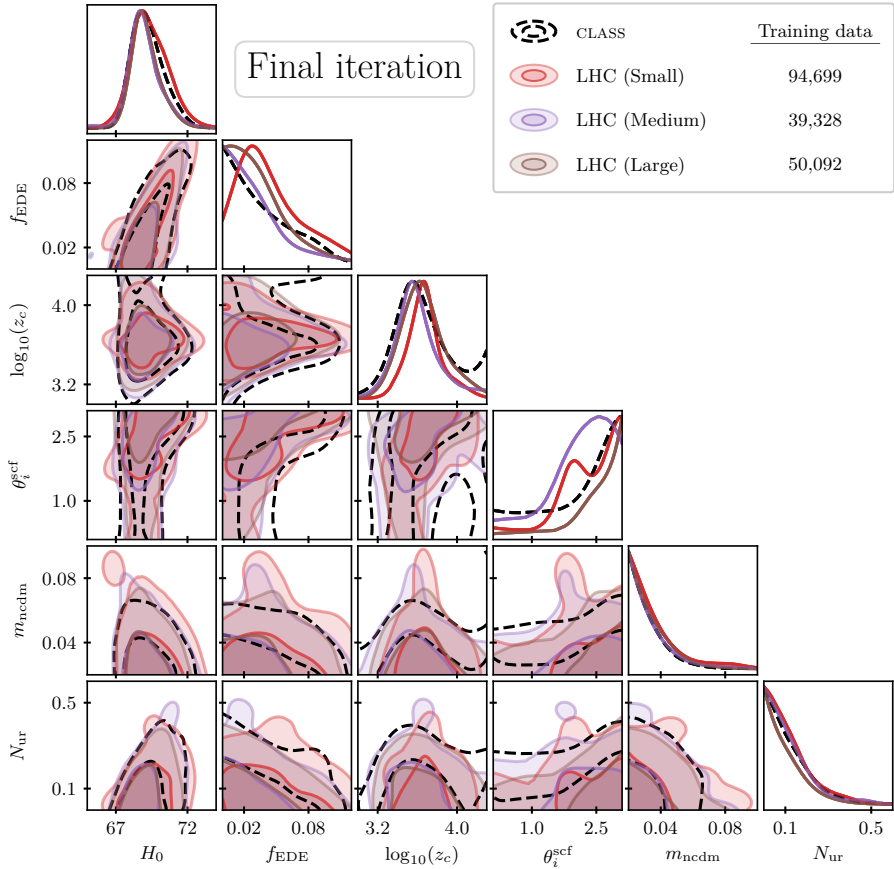


Figure 8.9: Posteriors from the MCMC runs using the neural networks emulating the $EDE+M_\nu+N_{\text{ur}}$ model from the final iterations of the Latin hypercube runs. A similar MCMC run using *CLASS* is also shown in black dashed lines as a reference. The table in the legend gives the number of training data points the respective networks used for the MCMCs were trained on.

resulting in training data from very different regions of the parameter space with little to no overlap in the first few iterations. This means that we end up with a contaminated set of training data where the neural network attempts to fit the large contamination during training at the cost of precision around the best-fit region (represented by contours from an MCMC using *CLASS* in the figure). Only in the final iterations does it seem to represent the best-fit region well. Usually, the iterations would overlap more, which filters away more points, so fewer *CLASS* evaluations are needed, but in this case, nearly all points

	Iterations	CLASS evaluations
Correlated hypersphere	5	40,529
Uncorrelated hypersphere	4	59,733
Λ CDM correlated hypersphere	4	44,222
Latin hypercube (Small)	8	100,699
Latin hypercube (Medium)	5	54,328
Latin hypercube (Large)	7	155,092

Table 8.3: Number of iterations and amount of CLASS evaluations (from both the initial sampling and the iterative process) in each $EDE+M_\nu+N_{\text{ur}}$ CONNECT run. The initial data from hyperspheres and Latin hypercubes is always discarded and furthermore the first iteration of 5,000 points is discarded for all Latin hypercubes.

are kept and points are only filtered out near the final iterations (aside from the first iteration which is completely discarded by default). There are fortunately simple solutions to accommodate this problem:

- using higher precision settings, e.g., training for many more epochs,
- throwing away training data from more iterations than the first, even though it can be difficult to know in advance,
- restarting the CONNECT run with the last model from the previous run as the initial neural network.

These solutions will all be able to solve the problem, but at much greater computational cost. Training for many more epochs can significantly slow down the iterative process, throwing away too many CLASS evaluations is wasteful and should be avoided, and restarting the run means that all previous training data is discarded, which is also wasteful. The least wasteful approach with the lowest computational cost will therefore be to use a hypersphere as initial guess instead (with correlations if available) and perhaps use more than 1,000 points for complicated cosmological models with a high dimensionality like this one.

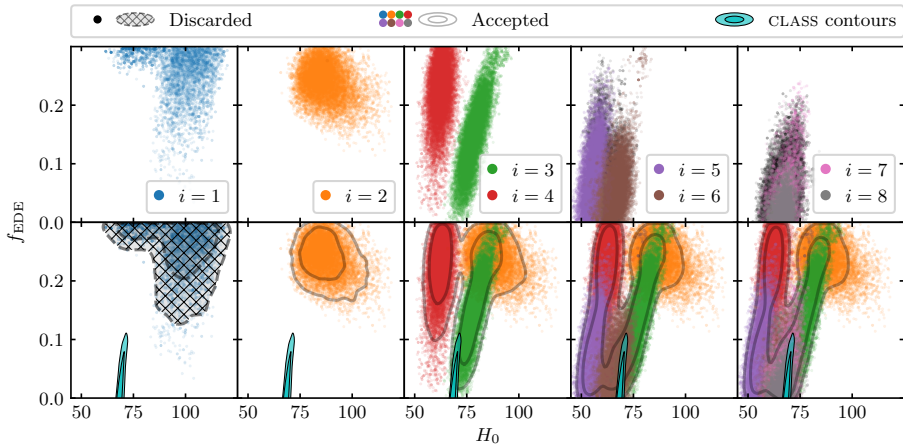


Figure 8.10: A depiction of how the training data was sampled iteratively in the CONNECT run with a small initial Latin hypercube of 1,000 points in the (H_0, f_{EDE}) -plane. The black points and the hatched contours represent discarded data. The CLASS contours have been included for reference.

8.4 CONCLUSION AND OUTLOOK

We have tested the performance of networks trained on hyperspheres and Latin hypercubes as well as how these impact the performance of the iterative approach of the CONNECT emulation framework. It is apparent that the neural networks trained on hyperspheres greatly outperform the networks trained on Latin hypercubes of similar size and with the same hyperparameters. Although weak knowledge about the shape and location of the posterior is required to initialise the hypersphere, this is not much different from the knowledge needed to initialise a Latin hypercube. We find that even using an uncorrelated hypersphere instead of a Latin hypercube cuts the amount of training data required for the same performances down by an order of magnitude, increasing to several orders of magnitude if correlations are included. Even in the case of a high-dimensional and highly non-Gaussian cosmological model like the $\text{EDE}+M_\nu+N_{\text{ur}}$ model, the correlated hypersphere with only 1,000 points trained for just 500 epochs was able to capture a lot of the behaviour of the observables under variations of the model parameters. These are quite low precision settings for the initial model, since its job in CONNECT’s iterative sampling is only to approximately locate the best-fit region, so if one wished to train a network just on

a hypersphere without the iterative approach, an order of magnitude more points and epochs, should be sufficient in nearly all cases. No matter the initial configuration presented in this paper, using a final CONNECT network for MCMC runs is computationally much cheaper than using CLASS directly in the MCMC, including sampling of training data, training the network, and the MCMC itself. This is especially true for elaborate cosmological models with difficulties converging.

As shown by other emulators [113, 157], Latin hypercubes are most certainly capable of good precision, but they require orders of magnitude more epochs, and for beyond- Λ CDM models also orders of magnitude more training data. If the aim of an emulator is to emulate a range in all parameters equally well, the Latin hypercube is still a good option, but if the aim is to use an emulator to compute likelihoods, the most effective option is to have the distribution of training data resemble the likelihood function. This is accomplished by the iterative approach of CONNECT, but one can get very close to the same effectiveness with a targeted uniform sampling representing the best-fit region, i.e. a correlated hypersphere. Especially in cases where the underlying cosmological code is prohibitively expensive, e.g. N -body codes, we conjecture that the performance would be greatly improved by switching to hypersphere sampling instead of Latin hypercube sampling. Indeed, emulators of N -body codes often only use very few training data points due to them being very slow to evaluate, but with a hypersphere, those few points would be much better distributed in order to capture the behaviour where the likelihood is significant.

Reproducibility. We have used the publicly available CONNECT framework available at https://github.com/AarhusCosmology/connect_public to create training data and train neural networks. The framework has been extended with the new way of sampling initial training data using hyperspheres. Explanatory parameter files have been included in the repository in order to easily use the framework and reproduce results from this paper.

ACKNOWLEDGEMENTS

We acknowledge computing resources from the Centre for Scientific Computing Aarhus (CSCAA). A.N., E.B.H., and T.T. were supported by a research grant (29337) from VILLUM FONDEN. We would like to thank Alessio Spurio Mancini and Sven Günther for their feedback on the draft of this paper.

 Ending of reference [\[4\]](#)

FINAL THOUGHTS AND PERSPECTIVES

Congratulations! You have reached the end of the thesis. During this thesis, you have been dragged through the formalities of the statistical frameworks employed, endured the vast number of equations involved in not only the treatment of cosmic microwave background radiation but also the evolution of matter, energy, and structure in the universe, persevered through the intricate derivation of the mathematics behind the training of neural networks, and you have survived the detailed presentation and description of the CONNECT framework and its applications.

The work presented here aimed to address a significant challenge in cosmology – the computational intensity associated with the analysis of complex cosmological models. By developing the CONNECT framework, I have demonstrated how neural networks can serve as a powerful tool for emulating cosmological observables, effectively alleviating the computational bottleneck that often limits the scope of parameter inference. This thesis explores not only the practicalities of implementing CONNECT but also the underlying theory, from statistical methods to machine learning techniques. Through the use of CONNECT, the cosmological community now has access to a more efficient way of conducting model parameter inference, which opens up new possibilities for exploring the universe. While this work has laid a strong foundation, it is only the beginning of what is possible with the integration of machine learning in cosmological research.

There are a few unfinished projects that I am eager to complete, including the implementation of direct likelihood emulation in CONNECT. Preliminary results have been promising, and the optimised approach to sampling training data makes this task feasible. This development could significantly enhance the efficiency of parameter inference, especially through the use of gradient-based methods. Some likelihood codes, which were originally designed to work with much slower Einstein–Boltzmann solvers, have now become bottlenecks due to their slower evaluation time compared to emulators. The addition of likelihood emulation would thus further improve the framework’s overall ef-

iciency. Other projects involve ongoing collaborations where CONNECT is being used. I take great pride in the growing interest surrounding the framework, and I am deeply grateful to the researchers who have embraced the challenge of trying out a new code.

While this thesis represents significant improvements in the application of emulation for cosmological parameter inference, much remains to be done. One of the next areas of focus is expanding the capabilities of CONNECT to emulate output from more than just Einstein–Boltzmann solver codes. N -body simulations, which face even greater challenges due to the slow computation of observables, present an exciting opportunity for CONNECT. Emulating power spectra obtained from N -body simulations is already underway, but current emulators often rely on Latin hypercube sampling [103, 201]. This method frequently includes points with very low likelihood, especially in the corners of the cuboid. By employing the active learning strategy of CONNECT or hypersphere sampling, this process could become much more efficient, making it possible to focus on more relevant points in the parameter space.

I mentioned in the very first sentence of this thesis that it marks the end of an era, and while that is indeed true, it also marks the beginning of a new one. As our understanding of the universe continues to expand and new theoretical models emerge, alongside the increasing availability of observational data through cosmological surveys, tools like CONNECT will be crucial in pushing the boundaries of what we can infer from this vast amount of data. I look forward to the continued development of this framework and the exciting discoveries that will undoubtedly follow as machine learning transforms the way we explore and understand the cosmos.

BIBLIOGRAPHY

- [1] Andreas Nygaard, Thomas Tram, and Steen Hannestad. „Updated constraints on decaying cold dark matter.“ In: *JCAP* 05 (2021), p. 017. DOI: [10.1088/1475-7516/2021/05/017](https://doi.org/10.1088/1475-7516/2021/05/017). arXiv: [2011.01632](https://arxiv.org/abs/2011.01632) [[astro-ph.CO](#)].
- [2] Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. „CONNECT: a neural network based framework for emulating cosmological observables and cosmological parameter inference.“ In: *JCAP* 05 (2023), p. 025. DOI: [10.1088/1475-7516/2023/05/025](https://doi.org/10.1088/1475-7516/2023/05/025). arXiv: [2205.15726](https://arxiv.org/abs/2205.15726) [[astro-ph.IM](#)].
- [3] Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. „Fast and effortless computation of profile likelihoods using CONNECT.“ In: *JCAP* 11 (2023), p. 064. DOI: [10.1088/1475-7516/2023/11/064](https://doi.org/10.1088/1475-7516/2023/11/064). arXiv: [2308.06379](https://arxiv.org/abs/2308.06379) [[astro-ph.CO](#)].
- [4] Andreas Nygaard, Emil Brinch Holm, Steen Hannestad, and Thomas Tram. „Cutting corners: hypersphere sampling as a new standard for cosmological emulators.“ In: *JCAP* 10 (2024), p. 073. DOI: [10.1088/1475-7516/2024/10/073](https://doi.org/10.1088/1475-7516/2024/10/073). arXiv: [2405.01396](https://arxiv.org/abs/2405.01396) [[astro-ph.CO](#)].
- [5] Andreas Nygaard, Emil Brinch Holm, Thomas Tram, and Steen Hannestad. „Decaying Dark Matter and the Hubble Tension.“ In: *The Hubble Constant Tension*. Ed. by Eleonora Di Valentino and Dillon Brout. Springer Series in Astrophysics and Cosmology. Springer, July 2024. Chap. 25, pp. 481–492. ISBN: 978-981-99-0176-0, 978-981-99-0179-1, 978-981-99-0177-7. DOI: [10.1007/978-981-99-0177-7](https://doi.org/10.1007/978-981-99-0177-7).
- [6] Camilla T. G. Sørensen, Steen Hannestad, Andreas Nygaard, and Thomas Tram. „Calculating Bayesian evidence for inflationary models using CONNECT.“ In: (June 2024). arXiv: [2406.03968](https://arxiv.org/abs/2406.03968) [[astro-ph.CO](#)].

- [7] Emil Brinch Holm, Laura Herold, Steen Hannestad, Andreas Nygaard, and Thomas Tram. „Decaying dark matter with profile likelihoods.“ In: *Phys. Rev. D* 107.2 (2023), p. L021303. DOI: [10.1103/PhysRevD.107.L021303](https://doi.org/10.1103/PhysRevD.107.L021303). arXiv: [2211.01935](https://arxiv.org/abs/2211.01935) [[astro-ph.CO](#)].
- [8] Emil Brinch Holm, Andreas Nygaard, Jeppe Dakin, Steen Hannestad, and Thomas Tram. „PROSPECT: A profile likelihood code for frequentist cosmological parameter inference.“ In: (Dec. 2023). arXiv: [2312.02972](https://arxiv.org/abs/2312.02972) [[astro-ph.CO](#)].
- [9] Olaf Behnke, Kevin Kröninger, Thomas Schörner-Sadenius, and Gregory Schott, eds. *Data analysis in high energy physics: A practical guide to statistical methods*. Weinheim, Germany: Wiley-VCH, 2013. ISBN: 978-3-527-41058-3, 978-3-527-65344-7, 978-3-527-65343-0.
- [10] W.M. Bolstad and J.M. Curran. *Introduction to Bayesian Statistics*. Wiley, 2016. ISBN: 9781118593226. URL: <https://books.google.dk/books?id=UeT3DAAAQBAJ>.
- [11] Joao C. Pinto Barros and Marina Krstic Marinkovic. „Towards the Application of Skewed Detailed Balance in Lattice Gauge Theories.“ In: *PoS LATTICE2022* (2023), p. 028. DOI: [10.22323/1.430.0028](https://doi.org/10.22323/1.430.0028). arXiv: [2402.01046](https://arxiv.org/abs/2402.01046) [[hep-lat](#)].
- [12] W. K. Hastings. „Monte Carlo sampling methods using Markov chains and their applications.“ In: *Biometrika* 57.1 (Apr. 1970), pp. 97–109. ISSN: 0006-3444. DOI: [10.1093/biomet/57.1.97](https://doi.org/10.1093/biomet/57.1.97). eprint: <https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf>. URL: <https://doi.org/10.1093/biomet/57.1.97>.
- [13] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. „Equation of State Calculations by Fast Computing Machines.“ In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092. DOI: [10.1063/1.1699114](https://doi.org/10.1063/1.1699114). URL: <http://link.aip.org/link/?JCP/21/1087/1>.
- [14] A. Gelman, G. O. Roberts, and W. R. Gilks. „Efficient Metropolis Jumping Rules.“ In: *Bayesian Statistics 5*. Ed. by J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith. Oxford: Oxford University Press, 1996, pp. 599–608.

- [15] Harold Jeffreys. *The Theory of Probability*. Third. Oxford University Press, 1961.
- [16] Samuel S Wilks. „The large-sample distribution of the likelihood ratio for testing composite hypotheses.“ In: *The Annals of Mathematical Statistics* 9.1 (1938), pp. 60–62.
- [17] Adam G. Riess et al. „A Comprehensive Measurement of the Local Value of the Hubble Constant with $1 \text{ km s}^{-1} \text{ Mpc}^{-1}$ Uncertainty from the Hubble Space Telescope and the SH0ES Team.“ In: *Astrophys. J. Lett.* 934.1 (2022), p. L7. DOI: [10.3847/2041-8213/ac5c5b](https://doi.org/10.3847/2041-8213/ac5c5b). arXiv: [2112.04510](https://arxiv.org/abs/2112.04510) [[astro-ph.CO](#)].
- [18] W. L. Freedman et al. „Final results from the Hubble Space Telescope key project to measure the Hubble constant.“ In: *Astrophys. J.* 553 (2001), pp. 47–72. DOI: [10.1086/320638](https://doi.org/10.1086/320638). arXiv: [astro-ph/0012376](https://arxiv.org/abs/astro-ph/0012376).
- [19] Massimiliano Bonamente, Marshall K. Joy, Samuel J. La Roque, John E. Carlstrom, Erik D. Reese, and Kyle S. Dawson. „Determination of the Cosmic Distance Scale from Sunyaev-Zel’dovich Effect and Chandra X-ray Measurements of High Redshift Galaxy Clusters.“ In: *Astrophys. J.* 647 (2006), pp. 25–54. DOI: [10.1086/505291](https://doi.org/10.1086/505291). arXiv: [astro-ph/0512349](https://arxiv.org/abs/astro-ph/0512349).
- [20] R. Brent Tully et al. „Cosmicflows-2: The Data.“ In: *Astron. J.* 146 (2013), p. 86. DOI: [10.1088/0004-6256/146/4/86](https://doi.org/10.1088/0004-6256/146/4/86). arXiv: [1307.7213](https://arxiv.org/abs/1307.7213) [[astro-ph.CO](#)].
- [21] N. Aghanim et al. „Planck 2018 results. VI. Cosmological parameters.“ In: *Astron. Astrophys.* 641 (2020). [Erratum: *Astron. Astrophys.* 652, C4 (2021)], A6. DOI: [10.1051/0004-6361/201833910](https://doi.org/10.1051/0004-6361/201833910). arXiv: [1807.06209](https://arxiv.org/abs/1807.06209) [[astro-ph.CO](#)].
- [22] B. Ryden. *Introduction to cosmology*. Cambridge University Press, 1970. ISBN: 978-1-107-15483-4, 978-1-316-88984-8, 978-1-316-65108-7. DOI: [10.1017/9781316651087](https://doi.org/10.1017/9781316651087).
- [23] European Space Agency. *Planck and the cosmic microwave background*. https://www.esa.int/Science_Exploration/Space_Science/Planck/Planck_and_the_cosmic_microwave_background. Accessed: 2024-12-31.
- [24] Steven Weinberg. *Cosmology*. 2008. ISBN: 978-0-19-852682-7.
- [25] Scott Dodelson. *Modern Cosmology*. Amsterdam: Academic Press, 2003. ISBN: 978-0-12-219141-1.

- [26] J.V. Michalowicz, J.M. Nichols, F. Bucholtz, and C.C. Olson. „A general Isserlis theorem for mixed-Gaussian random variables.“ In: *Statistics & Probability Letters* 81.8 (2011), pp. 1233–1240. ISSN: 0167-7152. DOI: <https://doi.org/10.1016/j.spl.2011.03.022>. URL: <https://www.sciencedirect.com/science/article/pii/S0167715211001052>.
- [27] Diego Blas, Julien Lesgourgues, and Thomas Tram. „The Cosmic Linear Anisotropy Solving System (CLASS) II: Approximation schemes.“ In: *JCAP* 07 (2011), p. 034. DOI: [10.1088/1475-7516/2011/07/034](https://doi.org/10.1088/1475-7516/2011/07/034). arXiv: [1104.2933](https://arxiv.org/abs/1104.2933) [[astro-ph.CO](#)].
- [28] P. A. R. Ade et al. „Improved Constraints on Primordial Gravitational Waves using Planck, WMAP, and BICEP/Keck Observations through the 2018 Observing Season.“ In: *Phys. Rev. Lett.* 127.15 (2021), p. 151301. DOI: [10.1103/PhysRevLett.127.151301](https://doi.org/10.1103/PhysRevLett.127.151301). arXiv: [2110.00483](https://arxiv.org/abs/2110.00483) [[astro-ph.CO](#)].
- [29] Joan Arnau Romeu. „Derivation of Friedman Equations.“ Tutor: Eduard Salvador Solé. Bachelor’s thesis. Barcelona, Spain: University of Barcelona, 2014. URL: <https://hdl.handle.net/2445/59759>.
- [30] Chung-Pei Ma and Edmund Bertschinger. „Cosmological perturbation theory in the synchronous and conformal Newtonian gauges.“ In: *Astrophys. J.* 455 (1995), pp. 7–25. DOI: [10.1086/176550](https://doi.org/10.1086/176550). arXiv: [astro-ph/9506072](https://arxiv.org/abs/astro-ph/9506072).
- [31] Le Zhang, Xuelei Chen, Marc Kamionkowski, Zong-guo Si, and Zheng Zheng. „Constraints on radiative dark-matter decay from the cosmic microwave background.“ In: *Phys. Rev. D* 76 (2007), p. 061301. DOI: [10.1103/PhysRevD.76.061301](https://doi.org/10.1103/PhysRevD.76.061301). arXiv: [0704.2444](https://arxiv.org/abs/0704.2444) [[astro-ph](#)].
- [32] Benjamin Audren, Julien Lesgourgues, Gianpiero Mangano, Pasquale Dario Serpico, and Thomas Tram. „Strongest model-independent bound on the lifetime of Dark Matter.“ In: *JCAP* 12 (2014), p. 028. DOI: [10.1088/1475-7516/2014/12/028](https://doi.org/10.1088/1475-7516/2014/12/028). arXiv: [1407.2418](https://arxiv.org/abs/1407.2418) [[astro-ph.CO](#)].
- [33] Théo Simon, Guillermo Franco Abellán, Peizhi Du, Vivian Poulin, and Yuhsin Tsai. „Constraining decaying dark matter with BOSS data and the effective field theory of large-scale structures.“ In: *Phys. Rev. D* 106.2 (2022), p. 023516. DOI: [10.1103/PhysRevD.106.023516](https://doi.org/10.1103/PhysRevD.106.023516). arXiv: [2203.07440](https://arxiv.org/abs/2203.07440) [[astro-ph.CO](#)].

- [34] Emil Brinch Holm, Thomas Tram, and Steen Hannestad. „Decaying warm dark matter revisited.“ In: *JCAP* 08.08 (2022), p. 044. DOI: [10.1088/1475-7516/2022/08/044](https://doi.org/10.1088/1475-7516/2022/08/044). arXiv: [2205.13628](https://arxiv.org/abs/2205.13628) [[astro-ph.CO](#)].
- [35] Nils Schöneberg, Guillermo Franco Abellán, Andrea Pérez Sánchez, Samuel J. Witte, Vivian Poulin, and Julien Lesgourgues. „The H0 Olympics: A fair ranking of proposed models.“ In: *Phys. Rept.* 984 (2022), pp. 1–55. DOI: [10.1016/j.physrep.2022.07.001](https://doi.org/10.1016/j.physrep.2022.07.001). arXiv: [2107.10291](https://arxiv.org/abs/2107.10291) [[astro-ph.CO](#)].
- [36] Vivian Poulin, Pasquale D. Serpico, and Julien Lesgourgues. „A fresh look at linear cosmological constraints on a decaying dark matter component.“ In: *JCAP* 08 (2016), p. 036. DOI: [10.1088/1475-7516/2016/08/036](https://doi.org/10.1088/1475-7516/2016/08/036). arXiv: [1606.02073](https://arxiv.org/abs/1606.02073) [[astro-ph.CO](#)].
- [37] Kiyotomo Ichiki, Masamune Oguri, and Keitaro Takahashi. „WMAP constraints on decaying cold dark matter.“ In: *Phys. Rev. Lett.* 93 (2004), p. 071302. DOI: [10.1103/PhysRevLett.93.071302](https://doi.org/10.1103/PhysRevLett.93.071302). arXiv: [astro-ph/0403164](https://arxiv.org/abs/astro-ph/0403164).
- [38] A. Chudaykin, D. Gorbunov, and I. Tkachev. „Dark matter component decaying after recombination: Lensing constraints with Planck data.“ In: *Phys. Rev. D* 94 (2016), p. 023528. DOI: [10.1103/PhysRevD.94.023528](https://doi.org/10.1103/PhysRevD.94.023528). arXiv: [1602.08121](https://arxiv.org/abs/1602.08121) [[astro-ph.CO](#)].
- [39] Z. Berezhiani, A. D. Dolgov, and I. I. Tkachev. „Reconciling Planck results with low redshift astronomical measurements.“ In: *Phys. Rev. D* 92.6 (2015), p. 061303. DOI: [10.1103/PhysRevD.92.061303](https://doi.org/10.1103/PhysRevD.92.061303). arXiv: [1505.03644](https://arxiv.org/abs/1505.03644) [[astro-ph.CO](#)].
- [40] Kanhaiya L. Pandey, Tanvi Karwal, and Subinoy Das. „Alleviating the H_0 and σ_8 anomalies with a decaying dark matter model.“ In: *JCAP* 07 (2020), p. 026. DOI: [10.1088/1475-7516/2020/07/026](https://doi.org/10.1088/1475-7516/2020/07/026). arXiv: [1902.10636](https://arxiv.org/abs/1902.10636) [[astro-ph.CO](#)].
- [41] A. Chudaykin, D. Gorbunov, and I. Tkachev. „Dark matter component decaying after recombination: Sensitivity to baryon acoustic oscillation and redshift space distortion probes.“ In: *Phys. Rev. D* 97.8 (2018), p. 083508. DOI: [10.1103/PhysRevD.97.083508](https://doi.org/10.1103/PhysRevD.97.083508). arXiv: [1711.06738](https://arxiv.org/abs/1711.06738) [[astro-ph.CO](#)].

- [42] Linfeng Xiao, Le Zhang, Rui An, Chang Feng, and Bin Wang. „Fractional Dark Matter decay: cosmological imprints and observational constraints.“ In: *JCAP* 01 (2020), p. 045. DOI: [10.1088/1475-7516/2020/01/045](#). arXiv: [1908.02668 \[astro-ph.CO\]](#).
- [43] Kari Enqvist, Seshadri Nadathur, Toyokazu Sekiguchi, and Tomo Takahashi. „Constraints on decaying dark matter from weak lensing and cluster counts.“ In: *JCAP* 04 (2020), p. 015. DOI: [10.1088/1475-7516/2020/04/015](#). arXiv: [1906.09112 \[astro-ph.CO\]](#).
- [44] Isabel M. Oldengott, Daniel Boriero, and Dominik J. Schwarz. „Reionization and dark matter decay.“ In: *JCAP* 08 (2016), p. 054. DOI: [10.1088/1475-7516/2016/08/054](#). arXiv: [1605.03928 \[astro-ph.CO\]](#).
- [45] Carlos Blanco and Dan Hooper. „Constraints on Decaying Dark Matter from the Isotropic Gamma-Ray Background.“ In: *JCAP* 03 (2019), p. 019. DOI: [10.1088/1475-7516/2019/03/019](#). arXiv: [1811.05988 \[astro-ph.HE\]](#).
- [46] Torsten Bringmann, Felix Kahlhoefer, Kai Schmidt-Hoberg, and Parampreet Walia. „Converting nonrelativistic dark matter to radiation.“ In: *Phys. Rev. D* 98.2 (2018), p. 023543. DOI: [10.1103/PhysRevD.98.023543](#). arXiv: [1803.03644 \[astro-ph.CO\]](#).
- [47] Keith R. Dienes, Fei Huang, Shufang Su, and Brooks Thomas. „Dynamical Dark Matter from Strongly-Coupled Dark Sectors.“ In: *Phys. Rev. D* 95.4 (2017), p. 043526. DOI: [10.1103/PhysRevD.95.043526](#). arXiv: [1610.04112 \[hep-ph\]](#).
- [48] Marco Raveri, Wayne Hu, Timothy Hoffman, and Lian-Tao Wang. „Partially Acoustic Dark Matter Cosmology and Cosmological Constraints.“ In: *Phys. Rev. D* 96.10 (2017), p. 103501. DOI: [10.1103/PhysRevD.96.103501](#). arXiv: [1709.04877 \[astro-ph.CO\]](#).
- [49] Kyriakos Vattis, Savvas M. Koushiappas, and Abraham Loeb. „Dark matter decaying in the late Universe can relieve the H_0 tension.“ In: *Phys. Rev. D* 99.12 (2019), p. 121302. DOI: [10.1103/PhysRevD.99.121302](#). arXiv: [1903.06220 \[astro-ph.CO\]](#).
- [50] Gordon Blackadder and Savvas M. Koushiappas. „Dark matter with two- and many-body decays and supernovae type Ia.“ In:

- Phys. Rev. D* 90.10 (2014), p. 103527. DOI: [10.1103/PhysRevD.90.103527](#). arXiv: [1410.0683 \[astro-ph.CO\]](#).
- [51] Steven J. Clark, Kyriakos Vattis, and Savvas M. Koushiappas. „Cosmological constraints on late-Universe decaying dark matter as a solution to the H_0 tension.“ In: *Phys. Rev. D* 103.4 (2021), p. 043014. DOI: [10.1103/PhysRevD.103.043014](#). arXiv: [2006.03678 \[astro-ph.CO\]](#).
- [52] Balakrishna S. Haridasu and Matteo Viel. „Late-time decaying dark matter: constraints and implications for the H_0 -tension.“ In: *Mon. Not. Roy. Astron. Soc.* 497.2 (2020), pp. 1757–1764. DOI: [10.1093/mnras/staa1991](#). arXiv: [2004.07709 \[astro-ph.CO\]](#).
- [53] Mei-Yu Wang and Andrew R. Zentner. „Effects of Unstable Dark Matter on Large-Scale Structure and Constraints from Future Surveys.“ In: *Phys. Rev. D* 85 (2012), p. 043514. DOI: [10.1103/PhysRevD.85.043514](#). arXiv: [1201.2426 \[astro-ph.CO\]](#).
- [54] Guillermo F. Abellan, Riccardo Murgia, Vivian Poulin, and Julien Lavalle. „Hints for decaying dark matter from S_8 measurements.“ In: (Aug. 2020). arXiv: [2008.09615 \[astro-ph.CO\]](#).
- [55] Guillermo F. Abellán, Riccardo Murgia, and Vivian Poulin. „Linear cosmological constraints on 2-body decaying dark matter scenarios and robustness of the resolution to the S_8 tension.“ In: (Feb. 2021). arXiv: [2102.12498 \[astro-ph.CO\]](#).
- [56] Francesca Borzumati, Torsten Bringmann, and Piero Ullio. „Dark matter from late decays and the small-scale structure problems.“ In: *Phys. Rev. D* 77 (2008), p. 063514. DOI: [10.1103/PhysRevD.77.063514](#). arXiv: [hep-ph/0701007](#).
- [57] Annika H. G. Peter, Christopher E. Moody, and Marc Kamionkowski. „Dark-Matter Decays and Self-Gravitating Halos.“ In: *Phys. Rev. D* 81 (2010), p. 103501. DOI: [10.1103/PhysRevD.81.103501](#). arXiv: [1003.0419 \[astro-ph.CO\]](#).
- [58] Annika H. G. Peter and Andrew J. Benson. „Dark-matter decays and Milky Way satellite galaxies.“ In: *Phys. Rev. D* 82 (2010), p. 123521. DOI: [10.1103/PhysRevD.82.123521](#). arXiv: [1009.1912 \[astro-ph.GA\]](#).

- [59] Mei-Yu Wang, Annika H. G. Peter, Louis E. Strigari, Andrew R. Zentner, Bryan Arant, Shea Garrison-Kimmel, and Miguel Rocha. „Cosmological simulations of decaying dark matter: implications for small-scale structure of dark matter haloes.“ In: *Mon. Not. Roy. Astron. Soc.* 445.1 (2014), pp. 614–629. DOI: [10.1093/mnras/stu1747](https://doi.org/10.1093/mnras/stu1747). arXiv: [1406.0527](https://arxiv.org/abs/1406.0527) [[astro-ph.CO](#)].
- [60] Shohei Aoyama, Kiyotomo Ichiki, Daisuke Nitta, and Naoshi Sugiyama. „Formulation and constraints on decaying dark matter with finite mass daughter particles.“ In: *JCAP* 09 (2011), p. 025. DOI: [10.1088/1475-7516/2011/09/025](https://doi.org/10.1088/1475-7516/2011/09/025). arXiv: [1106.1984](https://arxiv.org/abs/1106.1984) [[astro-ph.CO](#)].
- [61] Shohei Aoyama, Toyokazu Sekiguchi, Kiyotomo Ichiki, and Naoshi Sugiyama. „Evolution of perturbations and cosmological constraints in decaying dark matter models with arbitrary decay mass products.“ In: *JCAP* 07 (2014), p. 021. DOI: [10.1088/1475-7516/2014/07/021](https://doi.org/10.1088/1475-7516/2014/07/021). arXiv: [1402.2972](https://arxiv.org/abs/1402.2972) [[astro-ph.CO](#)].
- [62] Nikita Blinov, Celeste Keith, and Dan Hooper. „Warm Decaying Dark Matter and the Hubble Tension.“ In: *JCAP* 06 (2020), p. 005. DOI: [10.1088/1475-7516/2020/06/005](https://doi.org/10.1088/1475-7516/2020/06/005). arXiv: [2004.06114](https://arxiv.org/abs/2004.06114) [[astro-ph.CO](#)].
- [63] Benjamin Audren, Julien Lesgourgues, Karim Benabed, and Simon Prunet. „Conservative Constraints on Early Cosmology: an illustration of the Monte Python cosmological parameter inference code.“ In: *JCAP* 02 (2013), p. 001. DOI: [10.1088/1475-7516/2013/02/001](https://doi.org/10.1088/1475-7516/2013/02/001). arXiv: [1210.7183](https://arxiv.org/abs/1210.7183) [[astro-ph.CO](#)].
- [64] P.A.R. Ade et al. „Planck 2015 results. XIII. Cosmological parameters.“ In: *Astron. Astrophys.* 594 (2016), A13. DOI: [10.1051/0004-6361/201525830](https://doi.org/10.1051/0004-6361/201525830). arXiv: [1502.01589](https://arxiv.org/abs/1502.01589) [[astro-ph.CO](#)].
- [65] Kyle S. Dawson et al. „The Baryon Oscillation Spectroscopic Survey of SDSS-III.“ In: *Astron. J.* 145 (2013), p. 10. DOI: [10.1088/0004-6256/145/1/10](https://doi.org/10.1088/0004-6256/145/1/10). arXiv: [1208.0022](https://arxiv.org/abs/1208.0022) [[astro-ph.CO](#)].
- [66] M. Tanabashi et al. „Review of Particle Physics.“ In: *Phys. Rev. D* 98.3 (2018), p. 030001. DOI: [10.1103/PhysRevD.98.030001](https://doi.org/10.1103/PhysRevD.98.030001).
- [67] Robert J. Scherrer and Michael S. Turner. „Decaying Particles Do Not Heat Up the Universe.“ In: *Phys. Rev. D* 31 (1985), p. 681. DOI: [10.1103/PhysRevD.31.681](https://doi.org/10.1103/PhysRevD.31.681).

- [68] Adam G. Riess, Stefano Casertano, Wenlong Yuan, J. Bradley Bowers, Lucas Macri, Joel C. Zinn, and Dan Scolnic. „Cosmic Distances Calibrated to 1% Precision with Gaia EDR3 Parallaxes and Hubble Space Telescope Photometry of 75 Milky Way Cepheids Confirm Tension with Λ CDM.“ In: *Astrophys. J. Lett.* 908.1 (2021), p. L6. DOI: [10.3847/2041-8213/abdbaf](https://doi.org/10.3847/2041-8213/abdbaf). arXiv: [2012.08534](https://arxiv.org/abs/2012.08534) [[astro-ph.CO](#)].
- [69] T. M. C. Abbott et al. „Dark Energy Survey year 1 results: Cosmological constraints from galaxy clustering and weak lensing.“ In: *Phys. Rev. D* 98.4 (2018), p. 043526. DOI: [10.1103/PhysRevD.98.043526](https://doi.org/10.1103/PhysRevD.98.043526). arXiv: [1708.01530](https://arxiv.org/abs/1708.01530) [[astro-ph.CO](#)].
- [70] Shadab Alam et al. „The clustering of galaxies in the completed SDSS-III Baryon Oscillation Spectroscopic Survey: cosmological analysis of the DR12 galaxy sample.“ In: *Mon. Not. Roy. Astron. Soc.* 470.3 (2017), pp. 2617–2652. DOI: [10.1093/mnras/stx721](https://doi.org/10.1093/mnras/stx721). arXiv: [1607.03155](https://arxiv.org/abs/1607.03155) [[astro-ph.CO](#)].
- [71] Florian Beutler, Chris Blake, Matthew Colless, D. Heath Jones, Lister Staveley-Smith, Lachlan Campbell, Quentin Parker, Will Saunders, and Fred Watson. „The 6dF Galaxy Survey: Baryon Acoustic Oscillations and the Local Hubble Constant.“ In: *Mon. Not. Roy. Astron. Soc.* 416 (2011), pp. 3017–3032. DOI: [10.1111/j.1365-2966.2011.19250.x](https://doi.org/10.1111/j.1365-2966.2011.19250.x). arXiv: [1106.3366](https://arxiv.org/abs/1106.3366) [[astro-ph.CO](#)].
- [72] Ashley J. Ross, Lado Samushia, Cullan Howlett, Will J. Percival, Angela Burden, and Marc Manera. „The clustering of the SDSS DR7 main Galaxy sample – I. A 4 per cent distance measure at $z = 0.15$.“ In: *Mon. Not. Roy. Astron. Soc.* 449.1 (2015), pp. 835–847. DOI: [10.1093/mnras/stv154](https://doi.org/10.1093/mnras/stv154). arXiv: [1409.3242](https://arxiv.org/abs/1409.3242) [[astro-ph.CO](#)].
- [73] Lloyd Knox and Marius Millea. „Hubble constant hunter’s guide.“ In: *Phys. Rev. D* 101.4 (2020), p. 043533. DOI: [10.1103/PhysRevD.101.043533](https://doi.org/10.1103/PhysRevD.101.043533). arXiv: [1908.03663](https://arxiv.org/abs/1908.03663) [[astro-ph.CO](#)].
- [74] Guillermo Franco Abellán, Riccardo Murgia, and Vivian Poulin. „Linear cosmological constraints on two-body decaying dark matter scenarios and the S8 tension.“ In: *Phys. Rev. D* 104.12 (2021), p. 123533. DOI: [10.1103/PhysRevD.104.123533](https://doi.org/10.1103/PhysRevD.104.123533). arXiv: [2102.12498](https://arxiv.org/abs/2102.12498) [[astro-ph.CO](#)].

- [75] Guillermo Franco Abellán, Riccardo Murgia, Vivian Poulin, and Julien Lavalle. „Implications of the S_8 tension for decaying dark matter with warm decay products.“ In: *Phys. Rev. D* 105.6 (2022), p. 063525. DOI: [10.1103/PhysRevD.105.063525](https://doi.org/10.1103/PhysRevD.105.063525). arXiv: [2008.09615](https://arxiv.org/abs/2008.09615) [[astro-ph.CO](#)].
- [76] Lea Fuß and Mathias Garny. „Decaying Dark Matter and Lyman- α forest constraints.“ In: (Oct. 2022). arXiv: [2210.06117](https://arxiv.org/abs/2210.06117) [[astro-ph.CO](#)].
- [77] Gabriela Barenboim, Joe Zhiyu Chen, Steen Hannestad, Isabel M. Oldengott, Thomas Tram, and Yvonne Y. Y. Wong. „Invisible neutrino decay in precision cosmology.“ In: *JCAP* 03 (2021), p. 087. DOI: [10.1088/1475-7516/2021/03/087](https://doi.org/10.1088/1475-7516/2021/03/087). arXiv: [2011.01502](https://arxiv.org/abs/2011.01502) [[astro-ph.CO](#)].
- [78] Miguel Escudero and Samuel J. Witte. „The hubble tension as a hint of leptogenesis and neutrino mass generation.“ In: *Eur. Phys. J. C* 81.6 (2021), p. 515. DOI: [10.1140/epjc/s10052-021-09276-5](https://doi.org/10.1140/epjc/s10052-021-09276-5). arXiv: [2103.03249](https://arxiv.org/abs/2103.03249) [[hep-ph](#)].
- [79] Guillermo Franco Abellán, Zackaria Chacko, Abhish Dev, Peizhi Du, Vivian Poulin, and Yuhsin Tsai. „Improved cosmological constraints on the neutrino mass and lifetime.“ In: *JHEP* 08 (2022), p. 076. DOI: [10.1007/JHEP08\(2022\)076](https://doi.org/10.1007/JHEP08(2022)076). arXiv: [2112.13862](https://arxiv.org/abs/2112.13862) [[hep-ph](#)].
- [80] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2012. ISBN: 9780262018258. URL: <https://books.google.dk/books?id=maz6AQAAQBAJ>.
- [81] IBM Cloud Education. *What Is Supervised Learning?* Accessed: 2024-12-17. 2023. URL: <https://www.ibm.com/topics/supervised-learning>.
- [82] Charu C. Aggarwal. *Neural Networks and Deep Learning. A Textbook*. Cham: Springer, 2018, p. 497. ISBN: 978-3-319-94462-3. DOI: [10.1007/978-3-319-94463-0](https://doi.org/10.1007/978-3-319-94463-0).
- [83] Farnaz Ghassemi Toudeshki. *All You Need to Know about Active Learning — Part 1*. Accessed: 2024-12-25. 2023. URL: <https://medium.com/@farnazgh73/ultimate-guide-for-active-learning-main-approaches-3cf53ce207f0>.

- [84] Frederik Hvilshøj. *Active Learning in Machine Learning: Guide & Strategies*. Accessed: 2024-12-25. 2023. URL: <https://encord.com/blog/active-learning-machine-learning-guide/>.
- [85] Han Xiao, Kashif Rasul, and Roland Vollgraf. „Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.“ In: *arXiv e-prints*, arXiv:1708.07747 (Aug. 2017), arXiv:1708.07747. DOI: [10.48550/arXiv.1708.07747](https://doi.org/10.48550/arXiv.1708.07747). arXiv: [1708.07747](https://arxiv.org/abs/1708.07747) [cs.LG].
- [86] Arindam Dey. *A Gentle Introduction to Active Learning*. Accessed: 2023-09-02. 2023. URL: <https://arindam-dey.medium.com/a-gentle-introduction-to-active-learning-e983b9d175cb>.
- [87] Warren McCulloch and Walter Pitts. „A Logical Calculus of Ideas Immanent in Nervous Activity.“ In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 127–147.
- [88] F. Rosenblatt. *The perceptron - A perceiving and recognizing automaton*. Tech. rep. 85-460-1. Ithaca, New York: Cornell Aeronautical Laboratory, 1957.
- [89] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.
- [90] Jaspreet. *A Concise History of Neural Networks*. Accessed: 2024-12-20. 2016. URL: <https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec>.
- [91] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. „Learning representations by back-propagating errors.“ In: *Nature* 323 (1986), pp. 533–536. URL: <https://api.semanticscholar.org/CorpusID:205001834>.
- [92] Michael A. Nielsen. *Neural Networks and Deep Learning*. Accessed: 2024-12-20. Determination Press, 2015. URL: <http://neuralnetworksanddeeplearning.com/>.
- [93] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. „Multilayer feedforward networks are universal approximators.“ In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8). URL: <https://www.sciencedirect.com/science/article/pii/0893608089900208>.

- [94] N. Gershenfeld. *When Things Start to Think*. Hodder & Stoughton, 1999. ISBN: 9780340750391. URL: <https://books.google.dk/books?id=y2SPPwAACAAJ>.
- [95] Brendan Patrick Purdy. *Hyperbolic Tangent Activation Function for Neural Networks*. Accessed: 2024-12-22. 2024. URL: <https://flatironschool.com/blog/hyperbolic-tangent-activation-for-neural-networks/>.
- [96] Richard Socher and Rohit Mundra. *Deep Learning for NLP - Lecture Notes: Part III*. https://cs224d.stanford.edu/lecture_notes/LectureNotes3.pdf. Accessed: 2024-12-22. 2015.
- [97] Mohamed Bakrey. *Radial Basis Function Networks (RBFNs)*. Accessed: 2024-12-23. 2022. URL: <https://mohamedbakrey094.medium.com/radial-basis-function-networks-rbfns-be2ec324d8fb>.
- [98] Shiv Vignesh. *The Perfect Fit for a DNN*. Accessed: 2024-12-25. 2020. URL: <https://medium.com/analytics-vidhya/the-perfect-fit-for-a-dnn-596954c9ea39>.
- [99] Amanatulla. *Vanishing Gradient Problem in Deep Learning: Understanding, Intuition, and Solutions*. Accessed: 2024-12-25. 2023. URL: <https://medium.com/@amanatulla1606/vanishing-gradient-problem-in-deep-learning-understanding-intuition-and-solutions-da90ef4ecb54>.
- [100] Shubham Koli. *The Dying ReLU Problem: Causes and Solutions*. Accessed: 2024-12-25. 2023. URL: <https://medium.com/@MrBam44/the-dying-relu-problem-causes-and-solutions-1a970f177c>.
- [101] Thejs Brinckmann and Julien Lesgourgues. „MontePython 3: boosted MCMC sampler and other features.“ In: *Phys. Dark Univ.* 24 (2019), p. 100260. DOI: 10.1016/j.dark.2018.100260. arXiv: 1804.07261 [astro-ph.CO].
- [102] Jesus Torrado and Antony Lewis. „Cobaya: Code for Bayesian Analysis of hierarchical physical models.“ In: *JCAP* 05 (2021), p. 057. DOI: 10.1088/1475-7516/2021/05/057. arXiv: 2005.05290 [astro-ph.IM].

- [103] M. Knabenhans et al. „Euclid preparation: IX. EuclidEmulator2 – power spectrum emulation with massive neutrinos and self-consistent dark energy perturbations.“ In: *Mon. Not. Roy. Astron. Soc.* 505.2 (2021), pp. 2840–2869. DOI: [10.1093/mnras/stab1366](#). arXiv: [2010.11288 \[astro-ph.CO\]](#).
- [104] Arrykrishna Mootoovaloo, Andrew H. Jaffe, Alan F. Heavens, and Florent Leclercq. „Kernel-based emulator for the 3D matter power spectrum from CLASS.“ In: *Astron. Comput.* 38 (2022), p. 100508. DOI: [10.1016/j.ascom.2021.100508](#). arXiv: [2105.02256 \[astro-ph.CO\]](#).
- [105] Ming-Feng Ho, Simeon Bird, and Christian R. Shelton. „Multifidelity emulation for the matter power spectrum using Gaussian processes.“ In: *Mon. Not. Roy. Astron. Soc.* 509.2 (2021), pp. 2551–2565. DOI: [10.1093/mnras/stab3114](#). arXiv: [2105.01081 \[astro-ph.CO\]](#).
- [106] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. „Imagenet: A large-scale hierarchical image database.“ In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [107] T. Auld, Michael Bridges, M. P. Hobson, and S. F. Gull. „Fast cosmological parameter estimation using neural networks.“ In: *Mon. Not. Roy. Astron. Soc.* 376 (2007), pp. L11–L15. DOI: [10.1111/j.1745-3933.2006.00276.x](#). arXiv: [astro-ph/0608174](#).
- [108] T. Auld, M. Bridges, and M. P. Hobson. „CosmoNet: Fast cosmological parameter estimation in non-flat models using neural networks.“ In: *Mon. Not. Roy. Astron. Soc.* 387 (2008), p. 1575. DOI: [10.1111/j.1365-2966.2008.13279.x](#). arXiv: [astro-ph/0703445](#).
- [109] Sven Günther, Julien Lesgourgues, Georgios Samaras, Nils Schöneberg, Florian Stadtmann, Christian Fidler, and Jesús Torrado. „CosmicNet II: emulating extended cosmologies with efficient and accurate neural networks.“ In: *JCAP* 11 (2022), p. 035. DOI: [10.1088/1475-7516/2022/11/035](#). arXiv: [2207.05707 \[astro-ph.CO\]](#).
- [110] Jasper Albers, Christian Fidler, Julien Lesgourgues, Nils Schöneberg, and Jesus Torrado. „CosmicNet. Part I. Physics-driven implementation of neural networks within Einstein-Boltzmann Solvers.“ In: *JCAP* 09 (2019), p. 028. DOI:

- 10 . 1088 / 1475 - 7516 / 2019 / 09 / 028. arXiv: 1907 . 05764 [astro-ph.CO].
- [111] Andrea Manrique-Yus and Elena Sellentin. „Euclid-era cosmology for everyone: neural net assisted MCMC sampling for the joint 3×2 likelihood.“ In: *Mon. Not. Roy. Astron. Soc.* 491.2 (2020), pp. 2655–2663. DOI: 10 . 1093 / mnras / stj3059. arXiv: 1907.05881 [astro-ph.CO].
 - [112] Giovanni Aricò, Raul E. Angulo, and Matteo Zennaro. „Accelerating Large-Scale-Structure data analyses by emulating Boltzmann solvers and Lagrangian Perturbation Theory.“ In: (Apr. 2021). arXiv: 2104.14568 [astro-ph.CO].
 - [113] Alessio Spurio Mancini, Davide Piras, Justin Alsing, Benjamin Joachimi, and Michael P. Hobson. „CosmoPower: emulating cosmological power spectra for accelerated Bayesian inference from next-generation surveys.“ In: *Mon. Not. Roy. Astron. Soc.* 511.2 (2022), pp. 1771–1788. DOI: 10 . 1093 / mnras / stac064. arXiv: 2106.03846 [astro-ph.CO].
 - [114] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. „A Survey of Deep Active Learning.“ In: (2020). DOI: 10.48550/ARXIV.2009.00236. URL: <https://arxiv.org/abs/2009.00236>.
 - [115] Burr Settles. „Active learning literature survey.“ In: (2009). URL: <https://burrsettles.com/pub/settles.activelearning.pdf>.
 - [116] Burcu Sayin, Evgeny Krivosheev, Jie Yang, Andrea Passerini, and Fabio Casati. „A review and experimental analysis of active learning over crowdsourced data.“ In: *Artificial Intelligence Review* 54.7 (2021), pp. 5283–5305. DOI: 10.1007/s10462-021-10021-3. URL: <https://doi.org/10.1007/s10462-021-10021-3>.
 - [117] Burr Settles, Mark Craven, and Soumya Ray. „Multiple-Instance Active Learning.“ In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt, D. Koller, Y. Singer, and S. Roweis. Vol. 20. Curran Associates, Inc., 2007. URL: <https://proceedings.neurips.cc/paper/2007/file/a1519de5b5d44b31a01de013b9b51a80-Paper.pdf>.

- [118] Yarin Gal and Zoubin Ghahramani. „Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.“ In: (2015). DOI: [10.48550/ARXIV.1506.02142](https://arxiv.org/abs/1506.02142). URL: <https://arxiv.org/abs/1506.02142>.
- [119] Ethan Goan and Clinton Fookes. „Bayesian Neural Networks: An Introduction and Survey.“ In: *Case Studies in Applied Bayesian Data Science*. Springer International Publishing, 2020, pp. 45–87. DOI: [10.1007/978-3-030-42553-1_3](https://doi.org/10.1007/978-3-030-42553-1_3). URL: https://doi.org/10.1007/978-3-030-42553-1_3.
- [120] Keir K. Rogers, Hiranya V. Peiris, Andrew Pontzen, Simeon Bird, Licia Verde, and Andreu Font-Ribera. „Bayesian emulator optimisation for cosmology: application to the Lyman-alpha forest.“ In: *JCAP* 02 (2019), p. 031. DOI: [10.1088/1475-7516/2019/02/031](https://arxiv.org/abs/1812.04631). arXiv: [1812.04631](https://arxiv.org/abs/1812.04631) [[astro-ph.CO](#)].
- [121] Marcos Pellejero-Ibañez, Raul E. Angulo, Giovanni Aricó, Matteo Zennaro, Sergio Contreras, and Jens Stücker. „Cosmological parameter estimation via iterative emulation of likelihoods.“ In: *Mon. Not. Roy. Astron. Soc.* 499.4 (2020), pp. 5257–5268. DOI: [10.1093/mnras/staa3075](https://arxiv.org/abs/1912.08806). arXiv: [1912.08806](https://arxiv.org/abs/1912.08806) [[astro-ph.CO](#)].
- [122] Jonas El Gammal, Nils Schöneberg, Jesús Torrado, and Christian Fidler. „Fast and robust Bayesian Inference using Gaussian Processes with GPry.“ In: (Nov. 2022). arXiv: [2211.02045](https://arxiv.org/abs/2211.02045) [[astro-ph.CO](#)].
- [123] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. „When Gaussian Process Meets Big Data: A Review of Scalable GPs.“ In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11 (2020), pp. 4405–4423. DOI: [10.1109/TNNLS.2019.2957109](https://arxiv.org/abs/1807.01065). arXiv: [1807.01065](https://arxiv.org/abs/1807.01065) [[cs.LG](#)].
- [124] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. „Dive into Deep Learning.“ In: (2021). DOI: [10.48550/ARXIV.2106.11342](https://arxiv.org/abs/2106.11342). URL: <https://arxiv.org/abs/2106.11342>.
- [125] Martín Abadi et al. „TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.“ In: (2015). Software available from [tensorflow.org](https://www.tensorflow.org/). URL: <https://www.tensorflow.org/>.
- [126] Diederik P. Kingma and Jimmy Ba. „Adam: A Method for Stochastic Optimization.“ In: (2014). DOI: [10.48550/ARXIV.1412.6980](https://arxiv.org/abs/1412.6980). URL: <https://arxiv.org/abs/1412.6980>.

- [127] N. Aghanim et al. „Planck 2018 results. V. CMB power spectra and likelihoods.“ In: *Astron. Astrophys.* 641 (2020), A5. DOI: [10 . 1051 / 0004 - 6361 / 201936386](https://doi.org/10.1051/0004-6361/201936386). arXiv: [1907 . 12875](https://arxiv.org/abs/1907.12875) [[astro-ph.CO](#)].
- [128] Heather Prince and Jo Dunkley. „Data compression in cosmology: A compressed likelihood for Planck data.“ In: *Phys. Rev. D* 100.8 (2019), p. 083502. DOI: [10 . 1103 / PhysRevD . 100 . 083502](https://doi.org/10.1103/PhysRevD.100.083502). arXiv: [1909.05869](https://arxiv.org/abs/1909.05869) [[astro-ph.CO](#)].
- [129] Yun Wang, David N. Spergel, and Michael A. Strauss. „Cosmology in the next millennium: Combining MAP and SDSS data to constrain inflationary models.“ In: *Astrophys. J.* 510 (1999), p. 20. DOI: [10 . 1086 / 306558](https://doi.org/10.1086/306558). arXiv: [astro-ph/9802231](https://arxiv.org/abs/astro-ph/9802231).
- [130] Abien Fred Agarap. „Deep Learning using Rectified Linear Units (ReLU).“ In: (2018). DOI: [10 . 48550 / ARXIV . 1803 . 08375](https://doi.org/10.48550/ARXIV.1803.08375). URL: <https://arxiv.org/abs/1803.08375>.
- [131] Justin Alsing, Hiranya Peiris, Joel Leja, ChangHoon Hahn, Rita Tojeiro, Daniel Mortlock, Boris Leistedt, Benjamin D. Johnson, and Charlie Conroy. „SPECULATOR: Emulating Stellar Population Synthesis for Fast and Accurate Galaxy Spectra and Photometry.“ In: *The Astrophysical Journal Supplement Series* 249 (2020), p. 5. DOI: [10 . 3847 / 1538 - 4365 / ab917f](https://doi.org/10.3847/1538-4365/ab917f). arXiv: [1911.11778](https://arxiv.org/abs/1911.11778) [[astro-ph.IM](#)].
- [132] Sergey Ioffe and Christian Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.“ In: (2015). DOI: [10 . 48550 / ARXIV . 1502 . 03167](https://doi.org/10.48550/ARXIV.1502.03167). URL: <https://arxiv.org/abs/1502.03167>.
- [133] Michael D. Schneider, Ó skar Holm, and Lloyd Knox. „INTELLIGENT DESIGN: ON THE EMULATION OF COSMOLOGICAL SIMULATIONS.“ In: *The Astrophysical Journal* 728.2 (2011), p. 137. DOI: [10 . 1088 / 0004 - 637x / 728 / 2 / 137](https://doi.org/10.1088/0004-637x/728/2/137). URL: [https://doi.org/10 . 1088 % 2F0004 - 637x % 2F728 % 2F2 % 2F137](https://doi.org/10.1088/0004-637x/728/2/137).
- [134] Chun-Hao To, Eduardo Roza, Elisabeth Krause, Hao-Yi Wu, Risa H. Wechsler, and Andrés N. Salcedo. „LINNA: Likelihood Inference Neural Network Accelerator.“ In: (2022). DOI: [10 . 48550 / ARXIV . 2203 . 05583](https://doi.org/10.48550/ARXIV.2203.05583). URL: <https://arxiv.org/abs/2203.05583>.

- [135] Supranta S. Boruah, Tim Eifler, Vivian Miranda, and Sai Krishanth P. M. „Accelerating cosmological inference with Gaussian processes and neural networks – an application to LSST Y1 weak lensing and galaxy clustering.“ In: (2022). DOI: [10.48550/ARXIV.2203.06124](https://arxiv.org/abs/2203.06124). URL: <https://arxiv.org/abs/2203.06124>.
- [136] Guo-Jian Wang, Si-Yao Li, and Jun-Qing Xia. „ECOPANN: A Framework for Estimating Cosmological Parameters Using Artificial Neural Networks.“ In: *The Astrophysical Journal Supplement Series* 249.2 (2020), p. 25. DOI: [10.3847/1538-4365/aba190](https://doi.org/10.3847/1538-4365/aba190). URL: <https://doi.org/10.3847/1538-4365/aba190>.
- [137] Amir Hajian. „Efficient Cosmological Parameter Estimation with Hamiltonian Monte Carlo.“ In: *Phys. Rev. D* 75 (2007), p. 083525. DOI: [10.1103/PhysRevD.75.083525](https://arxiv.org/abs/10.1103/PhysRevD.75.083525). arXiv: [astro-ph/0608679](https://arxiv.org/abs/astro-ph/0608679).
- [138] Daniel Foreman-Mackey, David W. Hogg, Dustin Lang, and Jonathan Goodman. „emcee: The MCMC Hammer.“ In: *Publ. Astron. Soc. Pac.* 125 (2013), pp. 306–312. DOI: [10.1086/670067](https://arxiv.org/abs/10.1086/670067). arXiv: [1202.3665](https://arxiv.org/abs/1202.3665) [[astro-ph](https://arxiv.org/abs/astro-ph).IM].
- [139] Miguel de Carvalho, Garritt L. Page, and Bradley J. Barney. „On the Geometry of Bayesian Inference.“ In: *Bayesian Analysis* 14.4 (2019), pp. 1013 –1036. DOI: [10.1214/18-BA1112](https://doi.org/10.1214/18-BA1112). URL: <https://doi.org/10.1214/18-BA1112>.
- [140] P.M. Lee. *Bayesian Statistics: An Introduction*. A Charles Griffin book. Oxford University Press, 1989. ISBN: 9780195208030. URL: https://books.google.dk/books?id=_hXvAAAAMAAJ.
- [141] Juan S. Cruz, Steen Hannestad, Emil Brinch Holm, Florian Niedermann, Martin S. Sloth, and Thomas Tram. „Profiling cold new early dark energy.“ In: *Phys. Rev. D* 108.2 (2023), p. 023518. DOI: [10.1103/PhysRevD.108.023518](https://arxiv.org/abs/10.1103/PhysRevD.108.023518). arXiv: [2302.07934](https://arxiv.org/abs/2302.07934) [[astro-ph](https://arxiv.org/abs/astro-ph).CO].
- [142] Laura Herold and Elisa G. M. Ferreira. „Resolving the Hubble tension with Early Dark Energy.“ In: (Oct. 2022). arXiv: [2210.16296](https://arxiv.org/abs/2210.16296) [[astro-ph](https://arxiv.org/abs/astro-ph).CO].
- [143] Alexander Reeves, Laura Herold, Sunny Vagnozzi, Blake D. Sherwin, and Elisa G. M. Ferreira. „Restoring cosmological concordance with early dark energy and massive neutrinos?“ In: *Mon.*

- Not. Roy. Astron. Soc.* 520.3 (2023), pp. 3688–3695. DOI: [10.1093/mnras/stad317](https://doi.org/10.1093/mnras/stad317). arXiv: [2207.01501](https://arxiv.org/abs/2207.01501) [[astro-ph.CO](#)].
- [144] Paolo Campeti and Eiichiro Komatsu. „New Constraint on the Tensor-to-scalar Ratio from the Planck and BICEP/Keck Array Data Using the Profile Likelihood.“ In: *Astrophys. J.* 941.2 (2022), p. 110. DOI: [10.3847/1538-4357/ac9ea3](https://doi.org/10.3847/1538-4357/ac9ea3). arXiv: [2205.05617](https://arxiv.org/abs/2205.05617) [[astro-ph.CO](#)].
- [145] Adrià Gómez-Valent. „Fast test to assess the impact of marginalization in Monte Carlo analyses and its application to cosmology.“ In: *Phys. Rev. D* 106.6 (2022), p. 063506. DOI: [10.1103/PhysRevD.106.063506](https://doi.org/10.1103/PhysRevD.106.063506). arXiv: [2203.16285](https://arxiv.org/abs/2203.16285) [[astro-ph.CO](#)].
- [146] Laura Herold, Elisa G. M. Ferreira, and Eiichiro Komatsu. „New Constraint on Early Dark Energy from Planck and BOSS Data Using the Profile Likelihood.“ In: *Astrophys. J. Lett.* 929.1 (2022), p. L16. DOI: [10.3847/2041-8213/ac63a3](https://doi.org/10.3847/2041-8213/ac63a3). arXiv: [2112.12140](https://arxiv.org/abs/2112.12140) [[astro-ph.CO](#)].
- [147] P. A. R. Ade et al. „A Constraint on Primordial B-modes from the First Flight of the Spider Balloon-borne Telescope.“ In: *Astrophys. J.* 927.2 (2022), p. 174. DOI: [10.3847/1538-4357/ac20df](https://doi.org/10.3847/1538-4357/ac20df). arXiv: [2103.13334](https://arxiv.org/abs/2103.13334) [[astro-ph.CO](#)].
- [148] P. A. R. Ade et al. „Planck intermediate results. XVI. Profile likelihoods for cosmological parameters.“ In: *Astron. Astrophys.* 566 (2014), A54. DOI: [10.1051/0004-6361/201323003](https://doi.org/10.1051/0004-6361/201323003). arXiv: [1311.1657](https://arxiv.org/abs/1311.1657) [[astro-ph.CO](#)].
- [149] B. Illowsky and S. Dean. *Introductory Statistics*. Samurai Media Limited, 2017. ISBN: 9789888407309. URL: <https://books.google.dk/books?id=25-RtAEACAAJ>.
- [150] Max Tegmark and Matias Zaldarriaga. „Current cosmological constraints from a 10 parameter CMB analysis.“ In: *Astrophys. J.* 544 (2000), pp. 30–42. DOI: [10.1086/317188](https://doi.org/10.1086/317188). arXiv: [astro-ph/0002091](https://arxiv.org/abs/astro-ph/0002091).
- [151] Charles H. Lineweaver and Domingos Barbosa. „Cosmic microwave background observations: implications for hubble’s constant and the spectral parameters n and q in cold dark matter critical density universes.“ In: (Dec. 1996). arXiv: [astro-ph/9612146](https://arxiv.org/abs/astro-ph/9612146).

- [152] Steen Hannestad. „New constraints on neutrino physics from BOOMERANG data.“ In: *Phys. Rev. Lett.* 85 (2000), pp. 4203–4206. DOI: [10.1103/PhysRevLett.85.4203](#). arXiv: [astro-ph/0005018](#).
- [153] Steen Hannestad. „Neutrino masses and the number of neutrino species from WMAP and 2dFGRS.“ In: *JCAP* 05 (2003), p. 004. DOI: [10.1088/1475-7516/2003/05/004](#). arXiv: [astro-ph/0303076](#).
- [154] Jan Hamann. „Evidence for extra radiation? Profile likelihood versus Bayesian posterior.“ In: *JCAP* 03 (2012), p. 021. DOI: [10.1088/1475-7516/2012/03/021](#). arXiv: [1110.4271](#).
- [155] Nelson Christensen, Renate Meyer, Lloyd Knox, and Ben Luey. „II. Bayesian methods for cosmological parameter estimation from cosmic microwave background measurements.“ In: *Class. Quant. Grav.* 18 (2001), p. 2677. DOI: [10.1088/0264-9381/18/14/306](#). arXiv: [astro-ph/0103134](#).
- [156] Antony Lewis and Sarah Bridle. „Cosmological parameters from CMB and other data: A Monte Carlo approach.“ In: *Phys. Rev. D* 66 (2002), p. 103511. DOI: [10.1103/PhysRevD.66.103511](#). arXiv: [astro-ph/0205436](#).
- [157] Marco Bonici, Federico Bianchini, and Jaime Ruiz-Zapatero. „Capse.jl: efficient and auto-differentiable CMB power spectra emulation.“ In: (July 2023). arXiv: [2307.14339 \[astro-ph.CO\]](#).
- [158] Sven Günther. „Uncertainty-aware and Data-efficient Cosmological Emulation using Gaussian Processes and PCA.“ In: (July 2023). arXiv: [2307.01138 \[astro-ph.CO\]](#).
- [159] Antony Lewis, Anthony Challinor, and Anthony Lasenby. „Efficient computation of CMB anisotropies in closed FRW models.“ In: *Astrophys. J.* 538 (2000), pp. 473–476. DOI: [10.1086/309179](#). arXiv: [astro-ph/9911177](#).
- [160] Yudi Pawitan. *In All Likelihood. Statistical Modelling and Inference Using Likelihood*. Oxford University Press, 2013.
- [161] J. Neyman. „Outline of a Theory of Statistical Estimation Based on the Classical Theory of Probability.“ In: *Philosophical Transactions of the Royal Society of London A* 236.767 (1937), pp. 333–380. DOI: [10.1098/rsta.1937.0005](#).

- [162] Gary J. Feldman and Robert D. Cousins. „A Unified approach to the classical statistical analysis of small signals.“ In: *Phys. Rev. D* 57 (1998), pp. 3873–3889. DOI: [10.1103/PhysRevD.57.3873](https://doi.org/10.1103/PhysRevD.57.3873). arXiv: [physics/9711021](https://arxiv.org/abs/physics/9711021).
- [163] J. Kiefer and J. Wolfowitz. „Stochastic Estimation of the Maximum of a Regression Function.“ In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. ISSN: 00034851. URL: <http://www.jstor.org/stable/2236690> (visited on 08/03/2023).
- [164] George B. Dantzig, Alex Orden, and Philip Wolfe. „The generalized simplex method for minimizing a linear form under linear inequality restraints.“ In: *Pacific Journal of Mathematics* 5.2 (1955), pp. 183–195.
- [165] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. „Optimization by Simulated Annealing.“ In: *Science* 220.4598 (1983), pp. 671–680. DOI: [10.1126/science.220.4598.671](https://doi.org/10.1126/science.220.4598.671). eprint: <https://www.science.org/doi/pdf/10.1126/science.220.4598.671>. URL: <https://www.science.org/doi/abs/10.1126/science.220.4598.671>.
- [166] David J. Wales and Jonathan P. K. Doye. „Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms.“ In: *The Journal of Physical Chemistry A* 101.28 (1997), pp. 5111–5116. DOI: [10.1021/jp970984n](https://doi.org/10.1021/jp970984n). eprint: <https://doi.org/10.1021/jp970984n>. URL: <https://doi.org/10.1021/jp970984n>.
- [167] Ce Zhou, Christian Ieritano, and William Scott Hopkins. „Augmenting Basin-Hopping With Techniques From Unsupervised Machine Learning: Applications in Spectroscopy and Ion Mobility.“ In: *Frontiers in Chemistry* 7 (2019). ISSN: 2296-2646. DOI: [10.3389/fchem.2019.00519](https://doi.org/10.3389/fchem.2019.00519). URL: <https://www.frontiersin.org/articles/10.3389/fchem.2019.00519>.
- [168] Kenneth Levenberg. „A Method for the Solution of Certain Non-Linear Problems in Least Squares.“ In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168. ISSN: 0033569X, 15524485. URL: <http://www.jstor.org/stable/43633451> (visited on 08/03/2023).

- [169] Donald W. Marquardt. „An Algorithm for Least-Squares Estimation of Nonlinear Parameters.“ In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441. DOI: [10.1137/0111030](https://doi.org/10.1137/0111030). eprint: <https://doi.org/10.1137/0111030>. URL: <https://doi.org/10.1137/0111030>.
- [170] C. G. BROYDEN. „The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations.“ In: *IMA Journal of Applied Mathematics* 6.1 (1970), pp. 76–90. DOI: [10.1093/imamat/6.1.76](https://doi.org/10.1093/imamat/6.1.76). eprint: <https://academic.oup.com/imamat/article-pdf/6/1/76/2233756/6-1-76.pdf>. URL: <https://doi.org/10.1093/imamat/6.1.76>.
- [171] R. Fletcher. „A new approach to variable metric algorithms.“ In: *The Computer Journal* 13.3 (1970). DOI: [10.1093/comjnl/13.3.317](https://doi.org/10.1093/comjnl/13.3.317). eprint: <https://academic.oup.com/comjnl/article-pdf/13/3/317/988678/130317.pdf>. URL: <https://doi.org/10.1093/comjnl/13.3.317>.
- [172] Donald Goldfarb. „A Family of Variable-Metric Methods Derived by Variational Means.“ In: *Mathematics of Computation* 24.109 (1970), pp. 23–26. ISSN: 00255718, 10886842. URL: <http://www.jstor.org/stable/2004873> (visited on 08/03/2023).
- [173] D. F. Shanno. „Conditioning of Quasi-Newton Methods for Function Minimization.“ In: *Mathematics of Computation* 24.111 (1970), pp. 647–656. ISSN: 00255718, 10886842. URL: <http://www.jstor.org/stable/2004840> (visited on 08/03/2023).
- [174] Steven McCarty, Laura Burke, and Melissa McGuire. „Parallel Monotonic Basin Hopping for Low Thrust Trajectory Optimization.“ In: Jan. 2018. DOI: [10.2514/6.2018-1452](https://doi.org/10.2514/6.2018-1452).
- [175] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. „A Limited Memory Algorithm for Bound Constrained Optimization.“ In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208. DOI: [10.1137/0916069](https://doi.org/10.1137/0916069). eprint: <https://doi.org/10.1137/0916069>. URL: <https://doi.org/10.1137/0916069>.
- [176] Jerome Martin, Christophe Ringeval, and Vincent Vennin. „Encyclopædia Inflationaris.“ In: *Phys. Dark Univ.* 5-6 (2014), pp. 75–235. DOI: [10.1016/j.dark.2014.01.003](https://doi.org/10.1016/j.dark.2014.01.003). arXiv: [1303.3787](https://arxiv.org/abs/1303.3787) [astro-ph.CO].

- [177] Tanvi Karwal, Yashvi Patel, Alexa Bartlett, Vivian Poulin, Tristan L. Smith, and Daniel N. Pfeffer. „Procoli: Profiles of cosmological likelihoods.“ In: (Jan. 2024). arXiv: [2401.14225 \[astro-ph.CO\]](#).
- [178] Steen Hannestad. „Stochastic optimization methods for extracting cosmological parameters from cosmic microwave background radiation power spectra.“ In: *Phys. Rev. D* 61 (2000), p. 023002. DOI: [10.1103/PhysRevD.61.023002](#). arXiv: [astro-ph/9911330](#).
- [179] Roberto Trotta. „Bayes in the sky: Bayesian inference and model selection in cosmology.“ In: *Contemp. Phys.* 49 (2008), pp. 71–104. DOI: [10.1080/00107510802066753](#). arXiv: [0803.4089 \[astro-ph\]](#).
- [180] John Skilling. „Nested Sampling.“ In: *AIP Conf. Proc.* 735.1 (2004), p. 395. DOI: [10.1063/1.1835238](#).
- [181] John Skilling. „Nested sampling for general Bayesian computation.“ In: *Bayesian Analysis* 1.4 (2006), pp. 833–859. DOI: [10.1214/06-BA127](#).
- [182] W. J. Handley, M. P. Hobson, and A. N. Lasenby. „PolyChord: nested sampling for cosmology.“ In: *Mon. Not. Roy. Astron. Soc.* 450.1 (2015), pp. L61–L65. DOI: [10.1093/mnrasl/slv047](#). arXiv: [1502.01856 \[astro-ph.CO\]](#).
- [183] F. Feroz, M. P. Hobson, and M. Bridges. „MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics.“ In: *Mon. Not. Roy. Astron. Soc.* 398 (2009), pp. 1601–1614. DOI: [10.1111/j.1365-2966.2009.14548.x](#). arXiv: [0809.3437 \[astro-ph\]](#).
- [184] Greg Ashton et al. „Nested sampling for physical scientists.“ In: *Nature* 2 (2022). DOI: [10.1038/s43586-022-00121-x](#). arXiv: [2205.15570 \[stat.CO\]](#).
- [185] Davide Piras, Alicja Polanska, Alessio Spurio Mancini, Matthew A. Price, and Jason D. McEwen. „The future of cosmological likelihood-based inference: accelerated high-dimensional parameter estimation and model comparison.“ In: (May 2024). arXiv: [2405.12965 \[astro-ph.CO\]](#).

- [186] Alicja Polanska, Matthew A. Price, Davide Piras, Alessio Spurio Mancini, and Jason D. McEwen. „Learned harmonic mean estimation of the Bayesian evidence with normalizing flows.“ In: (May 2024). arXiv: [2405.05969 \[astro-ph.IM\]](#).
- [187] Jason D. McEwen, Christopher G. R. Wallis, Matthew A. Price, and Alessio Spurio Mancini. „Machine learning assisted Bayesian model comparison: learnt harmonic mean estimator.“ In: (2023). arXiv: [2111.12720 \[stat.ME\]](#).
- [188] Michael A. Newton and Adrian E. Raftery. „Approximate Bayesian Inference with the Weighted Likelihood Bootstrap.“ In: *Journal of the Royal Statistical Society: Series B (Methodological)* 56.1 (1994), pp. 3–26. DOI: [https://doi.org/10.1111/j.2517-6161.1994.tb01956.x](#). eprint: [https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1994.tb01956.x](#). URL: [https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1994.tb01956.x](#).
- [189] Y. Akrami et al. „Planck 2018 results. X. Constraints on inflation.“ In: *Astron. Astrophys.* 641 (2020), A10. DOI: [10.1051/0004-6361/201833887](#). arXiv: [1807.06211 \[astro-ph.CO\]](#).
- [190] Christophe Ringeval. „Fast Bayesian inference for slow-roll inflation.“ In: *Mon. Not. Roy. Astron. Soc.* 439.4 (2014), pp. 3253–3261. DOI: [10.1093/mnras/stu109](#). arXiv: [1312.2347 \[astro-ph.CO\]](#).
- [191] Jérôme Martin, Christophe Ringeval, Roberto Trotta, and Vincent Vennin. „The Best Inflationary Models After Planck.“ In: *JCAP* 03 (2014), p. 039. DOI: [10.1088/1475-7516/2014/03/039](#). arXiv: [1312.3529 \[astro-ph.CO\]](#).
- [192] Jerome Martin, Christophe Ringeval, and Vincent Vennin. „Cosmic Inflation at the Crossroads.“ In: (Apr. 2024). arXiv: [2404.10647 \[astro-ph.CO\]](#).
- [193] Richard Easter and Hiranya V. Peiris. „Bayesian Analysis of Inflation II: Model Selection and Constraints on Reheating.“ In: *Phys. Rev. D* 85 (2012), p. 103533. DOI: [10.1103/PhysRevD.85.103533](#). arXiv: [1112.0326 \[astro-ph.CO\]](#).
- [194] Andrew R. Liddle. „How many cosmological parameters?“ In: *Mon. Not. Roy. Astron. Soc.* 351 (2004), pp. L49–L53. DOI: [10.1111/j.1365-2966.2004.08033.x](#). arXiv: [astro-ph/0401198](#).

- [195] W. J. Handley, M. P. Hobson, and A. N. Lasenby. „polychord: next-generation nested sampling.“ In: *Mon. Not. Roy. Astron. Soc.* 453.4 (2015), pp. 4385–4399. DOI: [10.1093/mnras/stv1911](https://doi.org/10.1093/mnras/stv1911). arXiv: [1506.00171](https://arxiv.org/abs/1506.00171) [[astro-ph.IM](#)].
- [196] N. Aghanim et al. „Planck 2018 results. VIII. Gravitational lensing.“ In: *Astron. Astrophys.* 641 (2020), A8. DOI: [10.1051/0004-6361/201833886](https://doi.org/10.1051/0004-6361/201833886). arXiv: [1807.06210](https://arxiv.org/abs/1807.06210) [[astro-ph.CO](#)].
- [197] P. A. R. Ade et al. „BICEP2 / Keck Array x: Constraints on Primordial Gravitational Waves using Planck, WMAP, and New BICEP2/Keck Observations through the 2015 Season.“ In: *Phys. Rev. Lett.* 121 (2018), p. 221301. DOI: [10.1103/PhysRevLett.121.221301](https://doi.org/10.1103/PhysRevLett.121.221301). arXiv: [1810.05216](https://arxiv.org/abs/1810.05216) [[astro-ph.CO](#)].
- [198] Harold Jeffreys. *The Theory of Probability*. Oxford Classic Texts in the Physical Sciences. Oxford University Press, 1939. ISBN: 978-0-19-850368-2, 978-0-19-853193-7.
- [199] Mischa Knabenhans et al. „Euclid preparation: II. The EuclidEmulator – A tool to compute the cosmology dependence of the nonlinear matter power spectrum.“ In: *Mon. Not. Roy. Astron. Soc.* 484 (2019), pp. 5509–5529. DOI: [10.1093/mnras/stz197](https://doi.org/10.1093/mnras/stz197). arXiv: [1809.04695](https://arxiv.org/abs/1809.04695) [[astro-ph.CO](#)].
- [200] J. Ruiz-Zapatero, D. Alonso, C. García-García, A. Nicola, A. Mootoovaloo, J. M. Sullivan, M. Bonici, and P. G. Ferreira. „LimberJack.jl: auto-differentiable methods for angular power spectra analyses.“ In: (Oct. 2023). DOI: [10.21105/astro.2310.08306](https://doi.org/10.21105/astro.2310.08306). arXiv: [2310.08306](https://arxiv.org/abs/2310.08306) [[astro-ph.CO](#)].
- [201] Raul E. Angulo, Matteo Zennaro, Sergio Contreras, Giovanni Aricò, Marcos Pellejero-Ibañez, and Jens Stücker. „The BACCO simulation project: exploiting the full power of large-scale structure for cosmology.“ In: *Mon. Not. Roy. Astron. Soc.* 507.4 (2021), pp. 5869–5881. DOI: [10.1093/mnras/stab2018](https://doi.org/10.1093/mnras/stab2018). arXiv: [2004.06245](https://arxiv.org/abs/2004.06245) [[astro-ph.CO](#)].
- [202] M. D. McKay, R. J. Beckman, and W. J. Conover. „A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code.“ In: *Technometrics* 21.2 (1979), pp. 239–245. ISSN: 00401706. URL: <http://www.jstor.org/stable/1268522> (visited on 04/23/2024).

- [203] Boxin Tang. „Orthogonal Array-Based Latin Hypercubes.“ In: *Journal of the American Statistical Association* 88.424 (1993), pp. 1392–1397. ISSN: 01621459. URL: <http://www.jstor.org/stable/2291282> (visited on 04/23/2024).
- [204] R. L. Iman, J. M. Davenport, and D. K. Zeigler. *Latin hypercube sampling (program user's guide). [LHC, in FORTRAN]*. Program User's Guide. United States, 1980. URL: <https://www.osti.gov/biblio/5571631>.
- [205] Werner Krauth. *Statistical Mechanics Algorithms and Computations*. Oxford: Oxford University Press, 2006. ISBN: 9781429459501 1429459506.
- [206] George Casella and Roger Berger. *Statistical Inference*. Duxbury Resource Center, 2001. ISBN: 0534243126.
- [207] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. Cambridge University Press, 2007. ISBN: 0521880688.
- [208] Antony Lewis. „Efficient sampling of fast and slow cosmological parameters.“ In: *Phys. Rev. D* 87.10 (2013), p. 103529. DOI: [10.1103/PhysRevD.87.103529](https://doi.org/10.1103/PhysRevD.87.103529). arXiv: [1304.4473](https://arxiv.org/abs/1304.4473) [astro-ph.CO].
- [209] Tanvi Karwal and Marc Kamionkowski. „Dark energy at early times, the Hubble parameter, and the string axiverse.“ In: *Phys. Rev. D* 94.10 (2016), p. 103523. DOI: [10.1103/PhysRevD.94.103523](https://doi.org/10.1103/PhysRevD.94.103523). arXiv: [1608.01309](https://arxiv.org/abs/1608.01309) [astro-ph.CO].
- [210] Vivian Poulin, Tristan L. Smith, Tanvi Karwal, and Marc Kamionkowski. „Early Dark Energy Can Resolve The Hubble Tension.“ In: *Phys. Rev. Lett.* 122.22 (2019), p. 221301. DOI: [10.1103/PhysRevLett.122.221301](https://doi.org/10.1103/PhysRevLett.122.221301). arXiv: [1811.04083](https://arxiv.org/abs/1811.04083) [astro-ph.CO].
- [211] Vivian Poulin, Tristan L. Smith, and Tanvi Karwal. „The Ups and Downs of Early Dark Energy solutions to the Hubble tension: A review of models, hints and constraints circa 2023.“ In: *Phys. Dark Univ.* 42 (2023), p. 101348. DOI: [10.1016/j.dark.2023.101348](https://doi.org/10.1016/j.dark.2023.101348). arXiv: [2302.09032](https://arxiv.org/abs/2302.09032) [astro-ph.CO].

- [212] D. M. Scolnic et al. „The Complete Light-curve Sample of Spectroscopically Confirmed SNe Ia from Pan-STARRS1 and Cosmological Constraints from the Combined Pantheon Sample.“ In: *Astrophys. J.* 859.2 (2018), p. 101. DOI: [10.3847/1538-4357/aab9bb](#). arXiv: [1710.00845 \[astro-ph.CO\]](#).
- [213] J. Colin Hill, Evan McDonough, Michael W. Toomey, and Stephon Alexander. „Early dark energy does not restore cosmological concordance.“ In: *Phys. Rev. D* 102.4 (2020), p. 043507. DOI: [10.1103/PhysRevD.102.043507](#). arXiv: [2003.07355 \[astro-ph.CO\]](#).
- [214] J. Colin Hill et al. „Atacama Cosmology Telescope: Constraints on prerecombination early dark energy.“ In: *Phys. Rev. D* 105.12 (2022), p. 123536. DOI: [10.1103/PhysRevD.105.123536](#). arXiv: [2109.04451 \[astro-ph.CO\]](#).
- [215] Evan McDonough, J. Colin Hill, Mikhail M. Ivanov, Adrien La Posta, and Michael W. Toomey. „Observational constraints on early dark energy.“ In: (Oct. 2023). arXiv: [2310.19899 \[astro-ph.CO\]](#).

Never before has cosmology seen a transformation such as the one going on right now. Artificial intelligence is very rapidly advancing to aid in scientific endeavors and the continued advancements will give us the opportunity to efficiently analyse vast amounts of data. You can rest assured that cosmologists will work relentlessly to keep up with the advancements and utilise them the best we can. We might never unlock all of the secrets of the Universe, but knowing what is going to happen, would take the fun out of the pursuit anyway.

To exploit the many aspects of the field of machine learning that will let us efficiently analyse cosmological models, this thesis presents you with the emulation framework CONNECT. The thesis breaks down the development of the framework and its many applications never before possible without emulation. During the thesis, we are going to cover the basics of cosmology and machine learning, how to analyse cosmological models using emulation as well as how to run the CONNECT code. The work presented here is the product of around four years as a PhD student at the Department of Physics and Astronomy at Aarhus University. In the enormous cosmological desert, this amazing emulation tool might just be the beacon of light you have all been waiting for.