Simulating the Universe using Machine Learning

Toby Kidd & Thomas Watson

Fourth Year Project Report

Supervisor:
Dr Adam Moss

School of Physics & Astronomy
University of Nottingham
May 2024

Abstract

The goal of this project is to show that machine learning (ML), a form of artificial intelligence (AI), is a viable technique to use for cosmological and astrophysical research. The project presents the uses of machine learning in two separate areas: analysis of existing cosmological simulations and generation of new physical data. Convolutional neural networks are used to analyse the underlying physical parameters behind cosmological data from the CAMELS Multifield Dataset. The presented models are able to predict the Ω_M and σ_8 parameters, with prediction errors of 2.37% and 1.62% respectively. These neural networks also allow for analysis of both the aleatoric and epistemic errors, meaning that this approach may be extended to many other areas of scientific research. Moreover, generative AI models are used to produce new cosmological data, via both conditional and unconditional denoising diffusion implicit models, in a less computationally and financially taxing way as compared to traditional simulations. Also, generative methods may aid traditional simulation techniques, such as by using N-Body simulations to produce new hydrodynamic data. The new data generated in this project is tested via standard generative AI metrics, namely the Fréchet and kernel inception distances, and via physical tests, such as power spectrum analysis. These tests show that generative AI can produce new astrophysical data, from specified cosmological parameters, accurately. However, the method struggles to reproduce the small-scale features found in the CAMELS dataset. Additionally, this project showcases some recent advancements in ML techniques, with the aim of highlighting how new scientific research can benefit from ML methods.

Contents

1 Introduction and Background **2**
- 1.1 Project Introduction . . . 2
- 1.2 Physics Background . . . 2
- 1.3 Machine Learning Background . . . 5
  - 1.3.1 Machine Learning . . . 5
  - 1.3.2 Convolutional Neural Networks . . . 7
  - 1.3.3 Denoising Diffusion Implicit Models . . . 8
  - 1.3.4 Classifier-Free Conditional Diffusion Model . . . 9

2 Parameter Inference **11**
- 2.1 Introduction . . . 11
- 2.2 Method . . . 11
  - 2.2.1 Predicting Parameter Distributions . . . 11
  - 2.2.2 Epistemic Errors . . . 13
- 2.3 MSE Results and Discussion . . . 14
- 2.4 KL Results . . . 15
- 2.5 Discussion . . . 19
  - 2.5.1 Overall Performance . . . 19
  - 2.5.2 Data Errors . . . 19
  - 2.5.3 Model Errors . . . 21

3 Generating Data with Diffusion Models **22**
- 3.1 Introduction . . . 22
- 3.2 Method . . . 22
  - 3.2.1 Testing Methods . . . 23
- 3.3 Results . . . 25
  - 3.3.1 Diffusion Steps . . . 25
  - 3.3.2 Conditional Toggle . . . 25
  - 3.3.3 Conditional Parameters . . . 25
  - 3.3.4 Map Types . . . 25
- 3.4 Discussion . . . 28

4 Final Discussions and Conclusion **29**
- 4.1 Dataset Limitations . . . 29
- 4.2 Improvements in Machine Learning . . . 33
- 4.3 Machine Learning in Physical Research . . . 34
- 4.4 Project Conclusion . . . 35

Chapter 1

Introduction and Background

1.1 Project Introduction

Computational simulations of the Universe, in conjunction with observational data, are used to study the formation of galaxies and the structure of the cosmos [1]. Simulations provide information regarding the impact of dark energy and matter on the construction of the Universe, by comparing observed structures to those within simulation catalogues [2]. They also allow for constraints to be placed on cosmological parameters, which provides information about the composition of the Universe, such as the neutrino mass [3]. This information may be recovered from analysis of the matter power spectra of both simulation and observational cosmological data [4].

However, cosmological simulations are computationally taxing, with supercomputers often used to reduce this expense [5]. Although these simulations produce extremely useful data, using supercomputers leads to a substantial financial cost. Therefore, investigating new methods to produce cosmological data, in order to reduce the financial and computational costs of existing methods, is a worthwhile goal. Generative artificial intelligence (AI) models are a way of achieving this objective [6][7]. AI may also be used to analyse existing cosmological data [8], by deducing the values of certain cosmological and astrophysical parameters underpinning the data.

Machine learning (ML), a subset of AI, is a nascent and rapidly advancing technology, with new approaches being developed to achieve research goals in physics [9]. This project serves as a demonstration of the usefulness of ML in cosmology and astrophysics, by exploring its inference and generative capabilities. To achieve this goal, the project is separated into two areas of research: Chapter 2 focuses on the inference abilities of ML via convolutional neural networks; Chapter 3 investigates the generative functionality of AI via diffusion models .

1.2 Physics Background

In order to analyse and generate new cosmological simulation data, a suitable dataset is needed. In this project, the CAMELS Multifield Dataset (CMD) [10] serves this purpose. CMD is a collection of over 12,000 hydrodynamic and gravity-only cosmological simulations composed of different simulation types, including the IllustrisTNG [11] and SIMBA [12] suites.

Gravity-only, or N-Body, simulations consist of the gravitational interactions of dark matter within a simulated universe. The simulations are calculated using a tree-structure approximation [13], which uses the fact that neighbouring bodies have a greater influence on each other when compared to distant bodies. Such simulations have a complexity $O(N \log N)$, where $N \gtrsim 10^9$ is the number of simulated particles. This highlights the computational expense of large simulations. Hydrodynamic simulations include more complex interactions, such as those between dark matter and baryons [1], and thus are more computationally taxing.

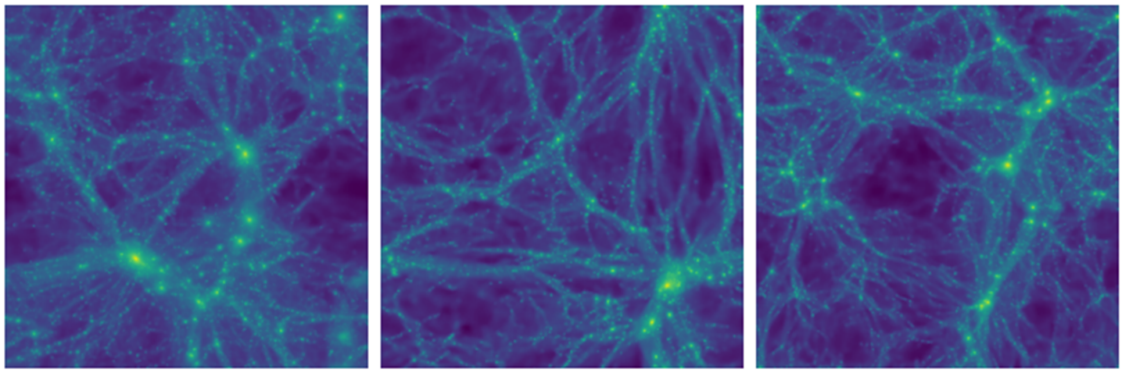


Figure 1.1: IllustrisTNG hydrodynamic simulation maps, 25 Mpc h^{-1} per side, showing the total matter density of different universes with varying cosmological and astrophysical parameters. A brighter colour indicates a higher matter density.

CMD is composed of a vast collection of two-dimensional (2D) maps showing the different properties of a simulated universe at redshift $z = 0$, due to the initial parameters of the simulation. Each map has a side comoving length of 25 Mpc h^{-1}, where h is the dimensionless Hubble constant. The maps are generated by taking a 2D slice of a larger $(25\ \text{Mpc}\, h^{-1})^3$ volume and integrating along a perpendicular line with respect to a measured quantity, such as the total matter density, within a 256x256 pixel region [10]. Examples of such maps are shown in Fig 1.1. Each hydrodynamic simulation has been calculated by following the evolution of 256^3 dark matter particles and 256^3 fluid elements within in a periodic box from $z = 127$ to $z = 0$. N-Body simulations follow the evolution of dark matter particles only.

Notably, due to the size of the maps used in CMD, the maps cannot contain extreme astronomical objects, such as voids or large galaxy clusters [10]. The limitations imposed by the map size are fully explored in Chapter 4.1. However, these maps still contain a sizeable amount of physical phenomena, allowing for the demonstration of the usefulness of inference and generative AI in cosmology and astrophysics. The objects found in these maps include dark matter halos, galaxies, and smaller galaxy clusters [14].

Each universe is characterised by six cosmological and astrophysical parameters. In the N-Body case, only the two cosmological parameters are considered. Each parameter is varied to produce a different universe. The schedule by which each parameter is changed is given by the chosen set, within CMD. For this project, the Latin hypercube set is used, where every parameter is varied within a certain range per simulation. This produces universes with different initial conditions and generates a large universe catalogue, which is ideal for the training of AI models.

The two cosmological parameters that appear in both the N-Body and hydrodynamic simulations are σ_8 and Ω_M. Each parameter is varied over a large range in the CMD maps. While many maps do not have parameter values similar to those expected for the Universe, the large ranges allow for ML models to fully understand the effects of these parameters. Both σ_8 and Ω_M dictate how matter is distributed and structured within a given universe.

σ_8 represents the amplitude of matter fluctuations over an 8 Mpc h^{-1} scale [15], i.e. how the matter is distributed within a given universe. In the Universe, σ_8 is expected to be approximately 0.81 [4], while in CMD the parameter is varied from 0.6 to 1.0. A higher σ_8 value leads to larger fluctuations in the matter power spectrum; for a value close to 1.0, the majority of matter will have collapsed into massive bodies such as galaxies [16][17], while the matter will be more distributed across a universe for a value close to 0.6. Therefore, σ_8 helps to control the matter distribution within a map and the structures present.

Ω_M is the total matter density within a universe [4], this includes the densities of cold dark matter and baryons. In CMD, Ω_M varies from 0.1 to 0.5, while it is believed to have a value of approximately 0.32 in the Universe. Combined with the effective mass densities of photons,

relativistic particles, and dark energy, which is related to the cosmological constant Λ [16], the total density parameter Ω is a ratio of the Universe's density with respect to a critical density, for which the Universe will expand perpetually [18]. Therefore, the value of Ω_M gives information about the expansion rate and composition of a universe.

There are four astrophysical feedback parameters in hydrodynamic CMD simulations, which have slightly different meanings and values depending on which simulation suite is used. In CMD, all astrophysical parameters are given as a multiplier to the fiducial value given in the original SIMBA or IllustrisTNG simulations, ranging from 0.25 to 4.

Active Galactic Nuclei (AGN) [18], which are supermassive black holes at the centre of galaxies, create galactic winds and jets of dust from their accretion disks [19]. Processes within these disks cause ejection of high energy jets, which shape the look of galaxies. They are measured by AGN feedback. AGN are characterised in CMD by the parameters A_{AGN1} and A_{AGN2}, which describe the momentum flux and speed of AGN jets respectively in SIMBA, or the energy accretion rate and ejection speed respectively in IllustrisTNG.

Supernovae, extremely powerful explosion of stars, also influence the evolution of the Universe by producing galactic winds [20]. Galactic winds contain charged particles as well as hot and cold gases, which interact with each other to convert kinetic to thermal energy. This causes the gases to expand, moving heavy metals across a galaxy [21]. The feedback from these winds is represented in CMD by A_{SN1} and A_{SN2}. In SIMBA these represent the mass loading and speed of the galactic winds, while they describe the energy and speed of the winds in IllustrisTNG. Both of these phenomena impact the evolution of galaxies within a universe [22] and their related values in the simulations impact the physics and appearance of the resulting maps.

1.3 Machine Learning Background

1.3.1 Machine Learning

The overarching aim of machine learning is to enable computers to perform specific tasks by generalising from patterns and examples. This is achieved by optimising a neural network. Neural networks are made up of neurons, as well as weights and biases between neurons, as shown by Fig 1.2.

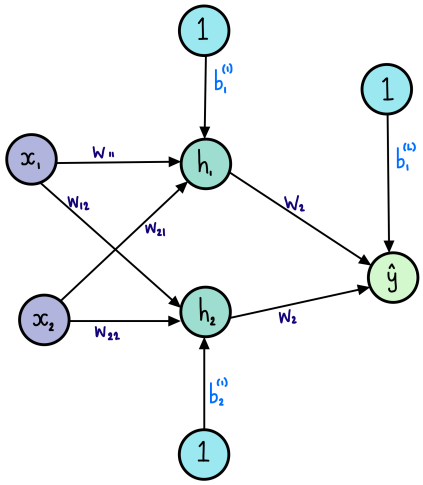


Figure 1.2: A neural network with two inputs x_1 and x_2, one hidden layer containing two neurons h_1 and h_2, and one output neuron $\hat{y}$, where all layers are fully connected, dense layers.

A neural network can be constructed and connected in any way the user sees fit, preferably in the best way to solve the given problem. The input data, e.g. pixel intensities, are converted into neurons, set to those values, which are then connected to further layers of trainable neurons. The connections are the weights ($\mathbf{W}$) and biases ($\mathbf{b}$). The weights are adjusted during training to best solve the task, along with an activation function (Φ). This combination of weights and biases means that each neuron layer can be represented mathematically as a linear equation [23]:

1. The first layer (denoted with the superscript $^{(1)}$):

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)T}\mathbf{x} + \mathbf{b}^{(1)}, \quad \mathbf{a}^{(1)} = \Phi^{(1)}(\mathbf{z}^{(1)}) \tag{1.1}$$

2. The second layer:

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)T}\mathbf{a}^{(1)} + \mathbf{b}^{(2)}, \quad \mathbf{a}^{(2)} = \Phi^{(2)}(\mathbf{z}^{(2)}) \tag{1.2}$$

The output neurons then provide an estimation of the solution to the task. For example, this could be the probability that an image is of a dog, or the matter density within a cosmological map, seen in Chapter 2. To this end, a loss function is used to define how close the output is to the expected results [24]. An example is the the mean square error (1.3), where $\hat{\mathbf{y}}$ is the expected value and $\mathbf{y}$ is the model predicted value:

$$l(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2 \tag{1.3}$$

The weights and biases of the neural network are therefore adjusted to minimise the loss through the processes defined below, known as back-propagation: [25]:

1. Forward pass: $z^{(l)}$ and $a^{(l)}$ are calculated for each layer using the above equations (1.1, 1.2).
2. Calculate the error at the output layer using the chosen loss function, J: $\mathbf{\Delta}^{(L)} := \frac{\partial J}{\partial \mathbf{a}^{(L)}} \Phi'^{(L)}(\mathbf{z}^{(L)})$
3. Backward pass: propagate the error backwards to calculate the error at each layer: $\mathbf{\Delta}^{(l)} := (\mathbf{W}^{(l+1)}\mathbf{\Delta}^{(l+1)}) \odot \Phi'^{(l)}(\mathbf{z}^{(l)})$

4. Calculate gradients: how much to adjust the weights and biases by: $\frac{\partial J}{\partial W_{kj}^{(l)}} = (\frac{\partial J}{\partial \mathbf{b}^{(l)}})_j a_k^{(l-1)}$

5. Update weights and biases: $W_{11}^{(2)} \to W_{11}^{(2)} - \alpha \frac{\partial J}{\partial W_{11}^{(2)}}$, where α is the learning rate, a hyper-parameter that controls the size of the update.

Therefore, this process can be thought of as a gradient step method towards the minimum, as represented in Fig 1.3.

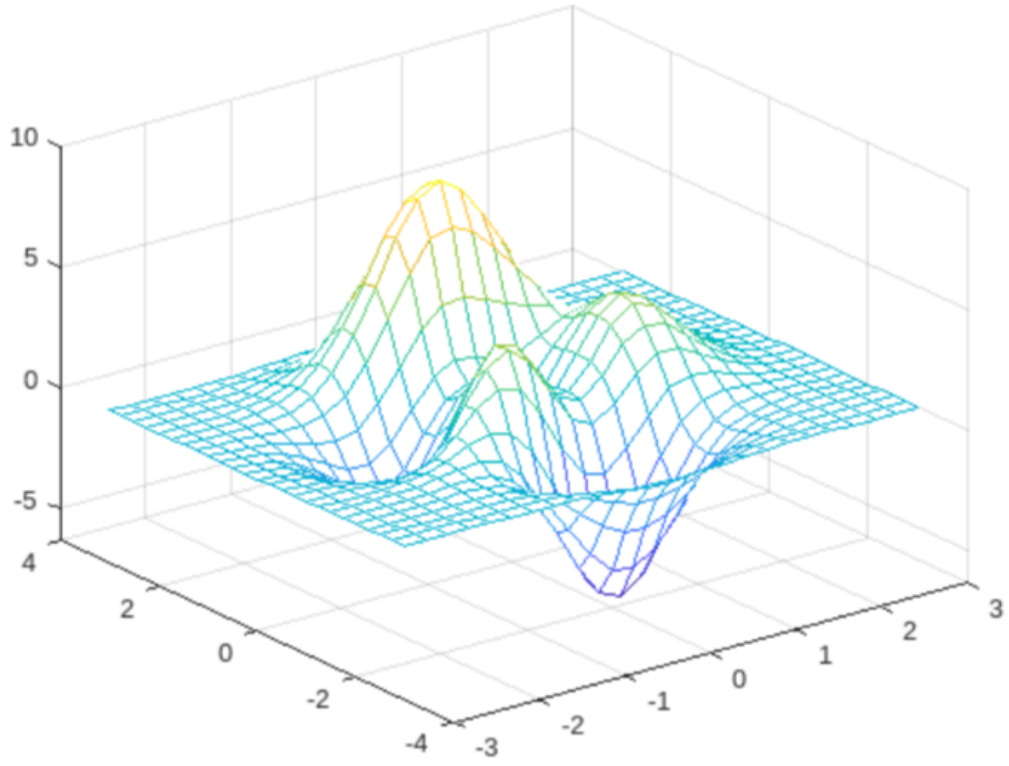


Figure 1.3: MATLAB peaks function graph, showing multiple peaks and troughs

However, in order to ensure that the network does not get stuck into local minima far away from the true global minimum, optimisation methods [26] are used. An example is the AdamW optimiser, which alters the size of the gradient descent step, in order to not overshoot the minimum and to accelerate training and convergence. Data pre-processing is also important to ensure standardisation between different input data, to ensure best suitability with the activation functions, and to improve efficiency of the model.

Throughout this project, ease-of-use methods are used. This includes early stopping: the validation loss is allowed to increase for at most 10 epochs before training is stopped and the weights are restored to those that produced the lowest validation loss. This is the loss for a reserved set of input data, the validation set, that is not used during training. Monitoring the validation loss helps to track how the model generalises to any input data. Checkpoints are also used: the weights that produce the lowest validation loss overall are saved and used for future predictions.

ML methods can be used in many areas of physical research. Modelling through machine learning can: capture non-linear relationships in cosmic phenomena [27]; refine theoretical models and improve the accuracy for the prediction of universal behaviours [28]. Machine learning can also accelerate computational simulations. It can generate realistic, time-efficient cosmological images and models, which facilitates rapid hypothesis testing and theoretical exploration. Unsupervised ML techniques hold the potential for discovering new physical laws by identifying novel patterns in existing data, potentially unveiling physical phenomena and structure.

1.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are used to make learning more efficient, for more complicated inputs and tasks [29]. This is accomplished through the use of a stack of convolutional and pooling layers in the neural network. The convolution layer works by applying a kernel to each pixel, which allows the model to investigate the important features within an image. 0-padding ensures that all edge values can be calculated as well, allowing the model to learn edge features at the boundaries of the input data. This is shown in Fig 1.4.

Pre-processing layers (e.g. flips and rotations) are of particular importance in CNN models. The model is able to detect the same features regardless of their positions in the input, known as translational invariance [30]. Pre-processing layers can also augment the data, which increases the size of the training dataset. This allows for more successful models, due to a reduction in overfitting, as the model has more data on which to train. Overfitting is when the model fits too closely to input data, and cannot generalise to unseen data [31]. Dropout layers also reduce overfitting [32]; on each training epoch there is a chance, a hyperparameter determined when setting up the network, that a neuron may be temporarily removed from the network, creating a sub-network. The neuron is removed by setting the preceding weights from other neurons to 0. This reduces overfitting, preventing the model from becoming overly dependent on any single neuron and the networks learn to find alternative routes to solve a problem.

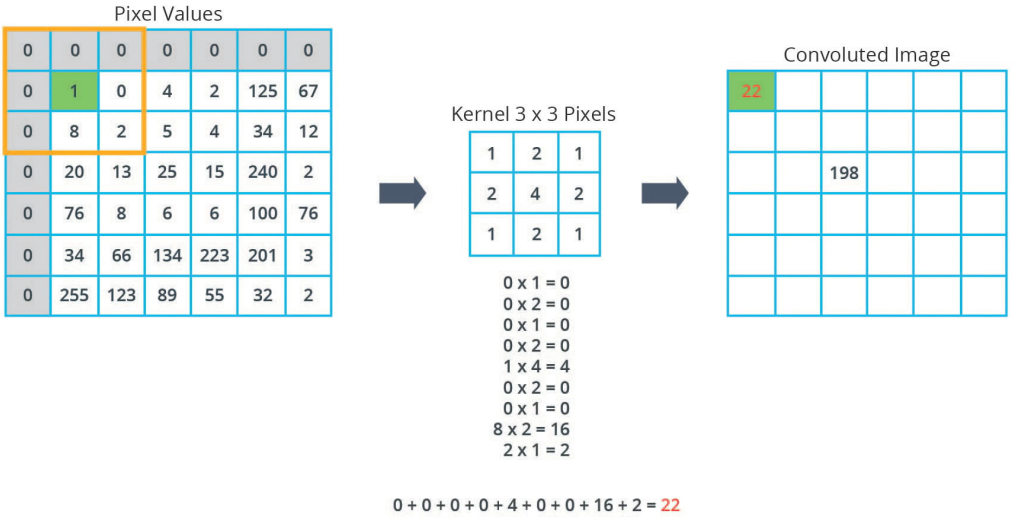


Figure 1.4: A convolutional kernel is applied to each pixel, and performing element wise multiplication. Image taken from a TensorFlow (python library for ML applications) course [33].

The max pooling layer creates a new matrix with elements consisting of the maximum element within a kernel applied to an input, shown in Fig 1.5. This down-sampling process helps to detect features which are more abstract in higher layers of the network, while simultaneously reducing the computational load and preventing overfitting.

Therefore, CNNs are effective at parameter inference tasks, as they are able to learn the overall patterns within input data, instead of learning the representation of each pixel or data point individually. Physical data can benefit from analysis by CNNs. The data often encompasses complex spatial structures and patterns. CNNs excel in extracting relevant features from such data, due to learning hierarchical representations of the data. This means that the CNN can automatically detect and learn features at multiple level of abstraction, such as: dots, lines, edges and combinations of these features. This capability makes CNNs adept at identifying subtle correlations and features in experimental data, which might be overlooked by traditional data analysis techniques.

Due to being able to efficiently process large-scale data, through down-sampling, and being able to identify complex patterns, CNNs can be used to analyse data, test models and theoretical

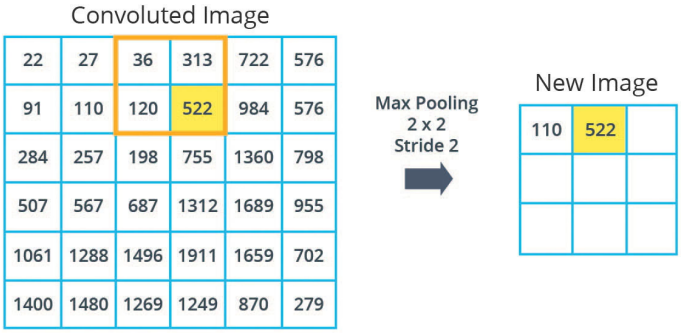


Figure 1.5: A max pooling kernel is applied across the whole matrix and the maximum element from each step is entered into the output matrix. Image taken from a TensorFlow course [33].

predictions rapidly, which may be quicker and simpler than traditional methods. This is explored in Chapter 2.

1.3.3 Denoising Diffusion Implicit Models

Denoising Diffusion Implicit Models (DDIMs) [34] are an approach to deep generative machine learning models. DDIMs have demonstrated remarkable proficiency in generating high-quality generated data, including images, audio, and text. DDIMs are a subset of the more general case of denoising diffusion probabilistic models (DDPMs) [35]. DDIMs provide faster sampling, or image generation, than DDPMs.

The core principle of denoising models involves modelling the data generation process as a reverse diffusion process [35]. This is done via a Markov chain of gradual denoising steps that transforms noise into a coherent sample. DDIMs can be described in two complementary processes [35]:

1. Forward diffusion process: Gaussian noise is gradually added to an input image according to a noise schedule, β_t, until data is isotropic Gaussian noise [35]. Here $\mathbf{x}_t$ is the image at each time step.

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{1.4}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{1.5}$$

   The model then learns to predict the noise additions at each time step, t. The model is fed noisy images at each time-step, then outputs a prediction of the noise addition.

2. Reverse diffusion process: iteratively denoises noisy data to recover the original data distribution, where $\mu(\mathbf{x}_t, t)$ is the model prediction for the mean of the Gaussian distribution used to sample $\mathbf{x}_{t-1}$, and $\Sigma_\theta(\mathbf{x}_t, t)$ represents the prediction on the variance, with θ representing model parameters.

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{1.6}$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{1.7}$$

A re-parametrisation trick can be used such that, $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. This simplifies the forward process to:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{1.8}$$

Therefore, $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon$. Where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, is the noise added to the image. A prediction of $\mathbf{x}_0$ can therefore be represented by:

$$f_\theta^{(t)}(\mathbf{x}_t) := \frac{(\mathbf{x}_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta^{(t)}(\mathbf{x}_t))}{\sqrt{\alpha_t}} \tag{1.9}$$

From this the reverse process can be defined as:

$$p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(\mathbf{x}_1), \sigma_1^2\mathbf{I}) & \text{if } t = 1 \\ q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, f_\theta^{(t)}(\mathbf{x}_t)) & \text{otherwise} \end{cases} \tag{1.10}$$

Leading to the following equation for each time-step, where $\sigma_t\epsilon_t$ is the random noise, such that different σ lead to different generation processes for the same model:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \epsilon_\theta^{(t)} + \sigma_t\epsilon_t \tag{1.11}$$

When the stochatisty of the model is set to 0, $\sigma_t = 0$, the model becomes deterministic and is known as a DDIM. This can lead to accelerated sampling, as the sampling process becomes deterministic, meaning that the trajectory through latent space, an abstract representation of image data, can be shortened by using an arbitrary subset of sampling steps.

Overall, DDIMs offer faster sampling compared to traditional DDPMs by enabling fewer time-steps without significantly compromising the quality of the generated samples [34]. DDIMs can generate many high-quality images, with controllable sample diversity. The stochasticity, σ, of the machine can be changed to give a mixture of deterministic and probabilistic based diffusion pathways, leading to a wide array of images. DDIMs are also highly customisable, given that the architecture as well as the diffusion and sampling steps can be changed to suit the project. DDIMs can be more rapidly tested that DDPMs, due to their faster sampling speed.

U-net Architecture

U-nets are an important part of the DDIM architecture, which allow for the prediction of the noise applied to an image to be made. The U-net architecture is a highly effective CNN model, initially designed for biomedical image segmentation [36], whose structure learns precise localisation information. The U-net architecture is characterised by its U-shaped design, consisting of two symmetrical main pathways: the contracting (downsampling) path and the expanding (upsampling) path [36].

The contracting path consists of subsequent convolutional and maxpooling layers, which captures the context, the information surrounding a pixel or group of pixels, in an image. This path captures high-level features by decreasing the spatial dimensions of the feature maps, while also increasing the depth at a reduced computational cost. Each convolutional layer in this path involves the application of a (n,n) kernel. Subsequent pooling layers further reduce the spatial dimension. After the feature maps' spatial dimensions have been reduced to the desired amount, there is a bottleneck layer, which consists of only convolutional layers. This layer acts as a bridge between the contracting and expanding paths. [36]

The expanding path represents the opposite process of the contracting path by replacing the pooling operations with upsampling operations, which recover the higher spatial dimensions needed for precise localisation. Following this, a convolutional layer is applied and concatenated with the corresponding cropped feature map from the contracting path. This combines the high-level features learned in the contracting path with the localisation from the expanding path. This process is repeated until the original image dimensions are recovered. [36]

1.3.4 Classifier-Free Conditional Diffusion Model

A diffusion model can be trained on conditional information, meaning that image samples can be generated according to conditional information. E.g. a model trained to produce images of dogs may be trained to generate images of Labradors. This is done by concatenating the conditional information to the input images. This means that the network can learn from the images as well as the parameters, increasing sample quality. However, this increased image quality comes at a trade-off [37], as conditioning information reduces sample diversity. This is already a problem for DDIMs, as they lose sample diversity through their inherent lack of stochasticity.

This is resolved through the use of a conditioning toggle, such that the model randomly drops the conditioning information while training, so that the model can learn more about the images and avoid overfitting to the conditioning information. This increases sampling diversity over a fully conditional model, but with less adherence to the conditional information. Therefore, training a classifier-free guidance model can be represented by the following process [37]:

1. $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$, describes a joint probability distribution, indicating that pairs of input images $\mathbf{x}$ and their associated conditioning information $\mathbf{c}$.
2. $\mathbf{c} \leftarrow \phi$ with probability p_{uncond}, which randomly drops the conditioning information to train unconditionally.
3. Noise is added to the image $\mathbf{x}$ until it becomes isotropic Gaussian noise, $\mathbf{x}_T$, as in typical DDPM.

Now when the DDIM method is used in the sampling process, a parameter can be parsed into the classifier-free guidance model called the guidance strength, $\mathbf{w}$, which represents how closely the generated images should adhere to the conditioning information.

Models in physical research often have many underlying parameters. Therefore, it is very important for any generative machine learning to be able to learn the relationships between the models and their parameters, in order to generate samples which are of practical significance. Being able to specify conditions on generated data allows for certain hypotheses to be tested, which may rely on specific data.

The combination of using a DDIM and classifier-free guidance architecture offers a lot of flexibility. The adherence to specific conditioning parameters can be specified, as well as the compromise between sample quality and diversity, as well as the trade-off between generation time and sample quality. This allows for good data to be generated in an efficient way, depending on the needs of the research project. By producing conditioned generated data, physicists can compare hypotheses and observed data to simulations of specific parameters.

Chapter 2

Parameter Inference

2.1 Introduction

Physicists investigate the statistical deviations from a homogeneous universe in order to see how the Universe was formed and what structures it contains. This is done by analysing the matter power spectrum [4], which is influenced by the aforementioned cosmological parameters. For deviations which follow Gaussian, or linear, statistics, the power spectrum fully contains all the necessary information. The power spectrum can be compared to simulation data, produced from a set of cosmological parameters. Bayesian analysis of likelihood functions can be conducted to find the most likely set of initial parameters [38], to within a given confidence interval. This allows for constrains to be placed on the cosmological parameters, usually to within a 95% confidence level [39]. For data generated with non-Gaussian, or non-linear, perturbations, such as CMD, the power spectrum is still an extremely useful summary statistic [10], though does not contain all the necessary information. There is not an equivalent summary statistic that contains all the required information in the non-linear case. Though similar likelihood methods to the linear case, are used to analyse such data; cosmological parameters are imposed onto simulations, which include non-linear fluctuations, and are matched against observation data [39]. The information obtained solely from the power spectrum in the non-Gaussian case can lead to large errors, for example a 20% error in the prediction of Ω_M on the CAMELS dataset [8].

The goal of Chapter 2 is to investigate whether machine learning (ML) methods are able to infer the values of the two cosmological (Ω_M, σ_8) and the four astrophysical $(A_{AGN1}, A_{AGN2}, A_{SN1}, A_{SN2})$ parameters with a low error; an overall model prediction error of 5% or lower will be deemed acceptable. The approach used follows the work of Villaescusa-Navarro et al in "Robust Marginalization of Baryonic Effects for Cosmological Inference at the Field Level". This section also investigates uncertainties in neural networks (NN) arising from both data and model error.

2.2 Method

2.2.1 Predicting Parameter Distributions

The goal of the inference NN is to be able to predict correctly the six parameters used to generate a CMD map. The model is trained to return a distribution for each parameter value per map in the chosen CMD suite, which allows for the error within the dataset to be predicted as well as the mean predicted parameter value.

To do this, both a large set of maps, X, and their corresponding parameter labels, Y, are fed into a convolutional neural network (CNN). The dataset used to train and test the inference model is the total matter density M_{tot} hydrodynamic IllustrisTNG set, with units $(M_\odot\, h^{-1})(\mathrm{Mpc}\, h^{-1})^{-2}$, where $M_\odot$ is the solar mass.

This set contains 15,000 different $(25\ \mathrm{Mpc}\,h^{-1})^2$ maps produced from a set of 1,000 different parameter combinations. Each set of parameters was used with a different random seed, given to the IllustrisTNG simulation to determine the initial density field, 15 times to produce 15,000 maps. 13,500 of these maps and the corresponding parameter values are used as the training dataset, X_{test} and Y_{test} respectively, of which 10% are used as a validation set after each training epoch. The remaining 1,500 entries are used to test the predictions of the trained model, X_{train} and Y_{train}.

Before being given to the NN, both X and Y need to be scaled. The pixel values of each map in X are Gaussian scaled, to reduce the values within the map, ensuring that they are better interpreted by the CNN. The label values are scaled using a min-max scaling for each parameter per map, ensuring that all values are in the set $[0, 1]$, allowing for a Sigmoid activation function [40] to be used. All label predictions produced by the CNN are later compared to the true values. To avoid any bias when training, the datasets are shuffled so that the network is not fed 15 maps of the same parameters in a row. This produces better prediction results on the testing dataset, as the model is presented with a varied set of images per optimisation step.

Once the datasets are prepared, they are fed into the CNN in batches of 32 maps. On every training epoch, each $x \in X_{test}$ has a chance to be augmented by random rotations of $90°, 180°, 270°$ (25% chance each) and flipped along the horizontal or vertical axis (50% chance each). This serves two purposes: the first is to artificially inflate the size of the training set to reduce overfitting; the second is to encourage the model to learn that these maps have both rotational and parity symmetry [41].

X, in training and testing, is then sent through a series of 7 blocks. Each block contains a 2D convolutional layer of (3,3) kernel size, with a LeakyReLU [40] activation function, and a maxpooling layer. The first convolutional layer has 32 filters, with the last having a 256. The number of filters increases across the blocks (32, 64, 64, 128, 128, 256, 256), allowing for the model to focus on the more important details of the map. This increases the likelihood of predicting an accurate parameter value. After the 7 convolutional blocks, X_{train} only is sent through a dropout layer, which also reduces the likelihood of overfitting. After dropout, there is a fully-connected layer with a LeakyReLU activation function.

The model then outputs two values, via two different fully-connected layers with Sigmoid, which produces outputs in the set $[0, 1]$, and Softplus [40], which provides only positive outputs, activation functions respectively. The outputs describe a distribution of the predicted parameter value, characterised by the mean value of the prediction, μ_i, and the standard deviation, σ_i. These are calculated via the following loss function $\mathcal{L}$, referred to as the KL loss function, used by Villaescusa-Navarro et al:

$$\mathcal{L} = \sum_{i=1}^{6} \log\left(\sum_{j\in\text{batch}} (\theta_{i,j} - \mu_{i,j})^2\right) + \sum_{i=1}^{6} \log\left(\sum_{j\in\text{batch}} \left((\theta_{i,j} - \mu_{i,j})^2 - \sigma_{i,j}^2\right)^2\right) \tag{2.1}$$

θ_i is the true value of the parameter. The index $i \in [1, 6]$ refers to each of the 6 parameters.

The loss function is separated into two terms. The first:

$$\mathcal{L}_1 = \sum_{i=1}^{6} \log\left(\sum_{j\in\text{batch}} (\theta_{i,j} - \mu_{i,j})^2\right) \tag{2.2}$$

is similar to the standard mean squared error (MSE) [24] and predicts the mean of the distribution. The second:

$$\mathcal{L}_2 = \sum_{i=1}^{6} \log\left(\sum_{j\in\text{batch}} \left((\theta_{i,j} - \mu_{i,j})^2 - \sigma_{i,j}^2\right)^2\right) \tag{2.3}$$

predicts the distribution's variance, which represents the error due to the data itself. The total loss function, $\mathcal{L}$, uses an estimate of the Kullback-Leibler (KL) divergence [42], along with a type of ML technique called a moment network [43], to predict both μ_i and σ_i.

Originally, the MSE loss function $\mathcal{L}_{MSE}$ (2.4), which is a much simpler function to implement, was used to test the feasibility of this model. Under this procedure, only the parameter values were predicted, not a distribution.

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{j=1}^{N}\left(\theta_{i,j} - \widehat{\theta}_{i,j}\right)^2 \tag{2.4}$$

N is the number of maps per batch. $\widehat{\theta}_i$ is the predicted label value.

The model is optimised by gradient descent, using the ADAM [44] optimisation algorithm, a precursor to the ADAMW loss function. The model is trained for a total of 150 epochs. For the first 50 epochs, the learning rate is constant, after which an exponentially decaying learning rate is. The model's weights are saved for each epoch that reduces the validation loss, when compared to the previous saved set of weights. Both of these techniques are employed to reduce overfitting.

The above method produces a prediction of Y, as well as an associated error due to the data, known as the aleatoric error [45]. Comparing the predicted Y_{test} to the real set gives the overall accuracy of the model. However, the method does not allow for any analysis of the model's own uncertainty. This can be produced by the ensemble method.

2.2.2 Epistemic Errors

The epistemic error [45] is the uncertainty due to the model itself. Changes in the model's weights, for the same architecture, can lead to different predictions for Y. Due to the nature of back-propagation, rebuilding and training the same model can lead to deviations in predictions of the parameters.

The epistemic error can be approximated in multiple ways. One such method is Monte Carlo (MC) dropout [46], where dropout layers are used when predicting values (i.e. not just during training) to reset some of the model's weights. The model, which is trained only once, is used to predict a set of parameters multiple times, producing a distribution. This is due to the random number of weights being set to 0 for each prediction, by the dropout layers. This method is not computationally demanding and approximates Bayesian statistical methods.

However, MC droput is difficult to implement. The placement of dropout layers, as well as their individual dropout rates, can lead to temperamental results and difficult analysis. A simpler approach, although more computationally stressful, is to produce an ensemble of results, from which a distribution can be made. This method is implemented in this project, known as the ensemble method.

The above training, testing, and gathering of results is run for a number of times, after which the means and standard deviations of the predicted label values can be calculated per map. After each model is used to predict a set of parameters, it is destroyed and a new model, of the same architecture, is trained. This leads to a set of models with different optimisation pathways and distinctive sets of weights within each NN, which results in different predictions per model.

In this project, the ensemble consists of 5 identical models trained for 150 epochs with the KL loss function architecture. The process gives an estimate of the standard error of the model, which can then be compared to the aleatoric error predicted by the KL loss function.

2.3 MSE Results and Discussion

Parameter	Percentage Errors (MSE)	Literature Percentage Errors (KL)
Ω_M	4.45	3.4
σ_8	2.28	2.4
A_{SN1}	54.27	38
A_{SN2}	22.66	17
A_{AGN1}	112.49	N/A
A_{AGN2}	33.68	N/A

Table 2.1: The mean prediction error, $\left\langle \frac{\Delta Y}{Y_{true}} \right\rangle$, for the MSE model for each parameter compared to the results by Villaescusa-Navarro et al using the KL loss function.

As mentioned, first MSE was used to assess the functionality of the model, by using the simpler loss function. This was to see if the model was unable to predict the value of certain parameters to within a reasonable error, allowing for focus to be placed on those which the model could predict well.

Table 2.1 shows low prediction errors Ω_M and σ_8, which are comparable to the existing literature results using the KL loss function and lower than 5%. This indicated the errors could be reduced with a more sophisticated loss function and a NN focused on the prediction of these two parameters.

The four astrophysical parameters could not be predicted to within a reasonable error, though A_{SN1} and A_{SN2} could be predicted to lower errors than the two AGN feedback parameters. This follows the results of Villaescusa-Navarro et al, who were unable to produce reasonable errors for the four astrophysical parameters, especially the two AGN feedback terms for which no results were presented.

The inaccuracies for these terms may be due to the size of the maps. Maps with a higher resolution, which focus on a smaller area of a universe, such as an individual galaxy, may be a better dataset on which to conduct inference on the feedback terms. The effects of supermassive blackholes, due to how they affect the makeup of individual galaxies, would be seen more clearly on such maps, potentially leading to a higher accuracy for the inference of the AGN feedback terms. Another possible reason for the higher errors for the astrophysical terms is that M_{tot} maps were used. These are dominated by the distribution of dark matter, a cosmological effect, so the impact of the feedback terms may be smaller than for other maps.

It is important to recall that the two simulation suites mentioned describe feedback in different ways, meaning direct comparison of parameter values between the two different suites is not useful. An A_{ANG1} value of 1.0 for IllustrisTNG does not mean the same as a value of 1.0 for SIMBA, as they do not describe the same effect. Although this does not affect the prediction accuracy of the model, the usefulness of predicting these values for astrophysical data is not as apparent as for the two cosmological parameters, which are mathematically consistent for both simulation and observation data. Also, training a CNN on one suite will lead to very poor predictions on the other suite for the feedback terms, due to the different definitions of the parameters per suite, as shown by Villaescusa-Navarro et al.

Given that the four astrophysical parameters could not be predicted to within reasonable errors by the less computationally taxing MSE model, the KL loss function is used only for the prediction of Ω_M and σ_8.

2.4 KL Results

Parameter	Percentage Errors (MSE)	Percentage Errors (KL)	Literature Percentage Errors (KL)
Ω_M	4.45	2.37	3.4
σ_8	2.28	1.62	2.4

Table 2.2: The mean prediction error, $\left\langle \frac{\Delta Y}{Y_{true}} \right\rangle$, for the cosmological parameters, from the MSE and KL models, compared to the results by Villaescusa-Navarro et al using the KL loss function.

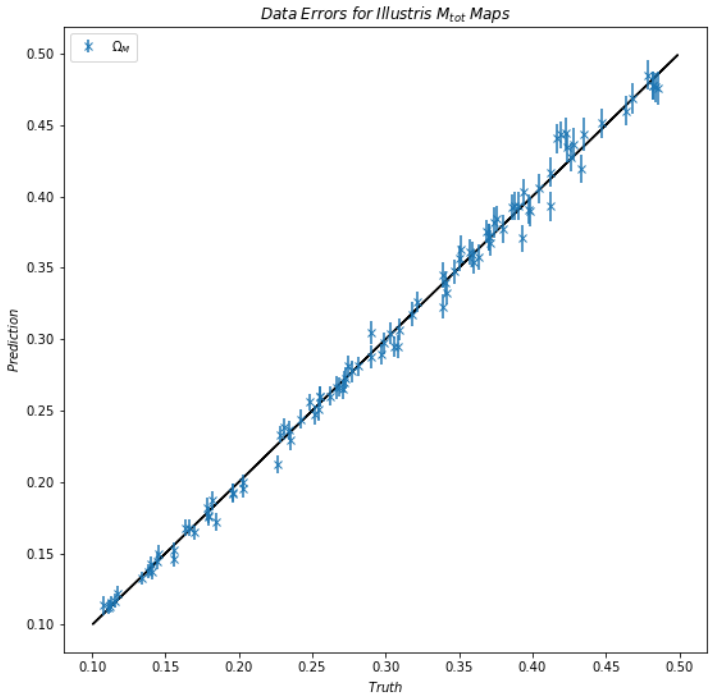
Table 2.2 shows that the CNN is able to infer the value of the cosmological parameters from the features within a map. The prediction errors produced by the inference model with the KL loss function are lower than the MSE case and lower than the results found by Villaescusa-Navarro et al. Interestingly, Villaescusa-Navarro et al trained their model for 200 epochs with a more complex architecture than the model used in this project. This shows that increasing the complexity of a NN does not always lead to better results, often due to overfitting [47]. It also demonstrates how focusing a model on the parameters it is able to understand leads to better performance.

While the overall percentage errors give an indication of the accuracy of the model, analysis of individual predictions and their associated errors gives more insight into the feasibility of parameter inference for cosmological data.

	Ω_M	σ_8
Fraction of points within $\pm\sigma$ (Data Error)	0.65	0.49
Fraction of points within $\pm s$ (Model Error)	0.19	0.14
Average predicted data error (σ)	0.0078	0.011
Average model error (s)	0.0021	0.0030
Average model error of the standard deviation s_σ	0.00042	0.00057

Table 2.3: Results for the cosmological parameters using the KL loss function. σ represents the standard deviation predicted by the model, i.e. the data error. While s is the standard error approximation calculated from the ensemble method, i.e. the model error. s_σ is the model's error for its calculation of σ.

The KL loss function gives the mean and standard deviation of a distribution of predictions, for a particular parameter from a given map. Fig 2.1 shows the true, un-scaled Ω_M values against the NN's predicted mean value of the same Ω_M. This plot is 100 random Ω_M values taken from Y_{test}. In the data errors case, the errors shown are the predicted standard deviations produced by the model. The model errors are an estimate of the standard error produced by the distribution of predictions from the ensemble method. Likewise, Fig 2.2 shows the results for σ_8. Table 2.3 shows the average errors for each parameter, calculated using the two error techniques.



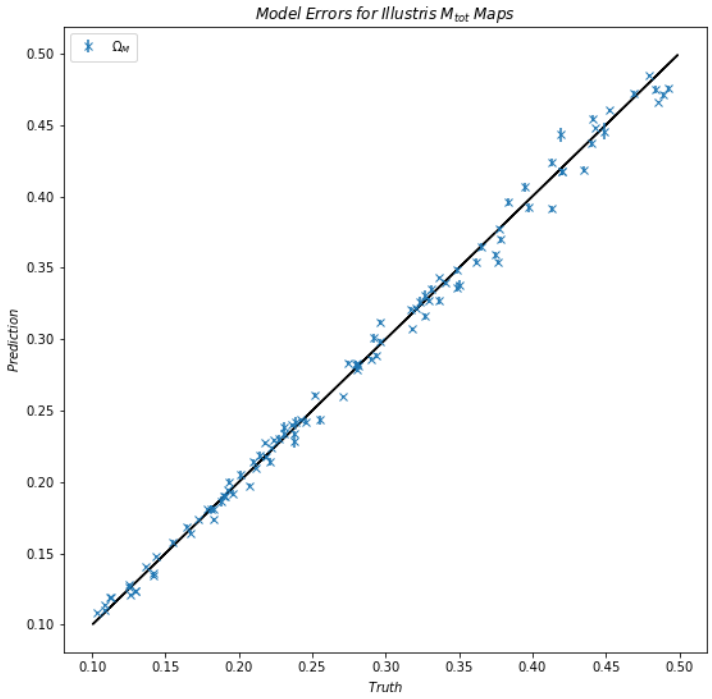
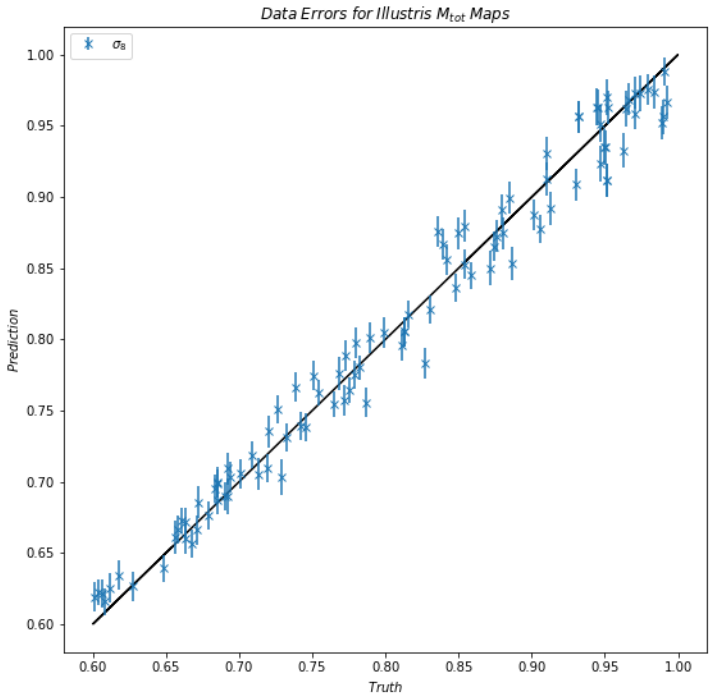


Figure 2.1: Data and model errors produced by the CNN for Ω_M. The data errors are produced by the KL loss function, while the model errors are given by the ensemble method. The black line shows the expected value.



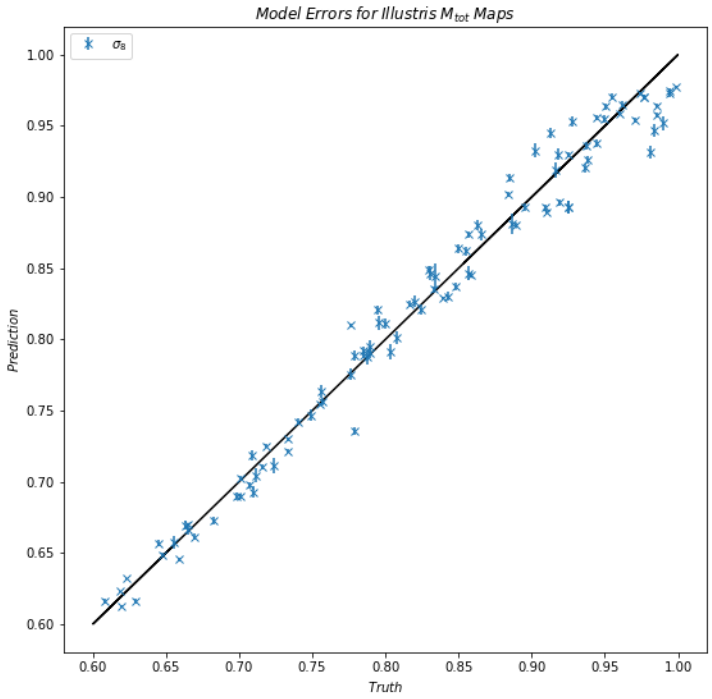


Figure 2.2: Data and model errors produced by the CNN for σ_8. The data errors are produced by the KL loss function, while the model errors are given by the ensemble method. The black line shows the expected value.

Parameter	**Average Relative Data (σ) Percentage Uncertainty**	**Average Relative Model (s) Percentage Uncertainty**	**Average Relative Model Percentage Uncertainty for Data Error Prediction**
Ω_M	2.69	0.72	5.4
σ_8	1.38	0.38	5.2

Table 2.4: The average relative data and model percentage uncertainties for Ω_M and σ_8.

Table 2.4 shows the average relative data and model percentage uncertainties. Calculated by comparing σ and s to the mean parameter value in Y_{test}. The relative model uncertainty for the data error prediction compares s_σ to σ.

Parameter	**M_{tot} Prediction Percentage Error**	**n_e Prediction Percentage Error**
Ω_M	2.37	3.52
σ_8	1.62	3.50

Table 2.5: Prediction error for both the M_{tot} and n_e maps using the KL loss function architecture.

The KL model was also trained and tested on the hydrodynamic IllustrisTNG electron number density maps n_e, with units $h^2 cm^{-3}(\mathrm{Mpc}\, h^{-1})^{-1}$. The training and testing process was the same as the M_{tot} case, but only the overall predictions errors were calculated. This was to investigate the performance of the model on a different map type. Table 2.5 shows the results.

2.5 Discussion

2.5.1 Overall Performance

The model is able to ascertain the cosmological parameters behind a set of maps to within a average error lower than 2.5%. Compared to power spectrum analysis on the M_{tot} dataset, the model performs much better than the 20% error for the calculation of Ω_M [8]. This highlights the ability of CNNs to infer the parameters behind cosmological data, allowing for universe catalogues to be analysed accurately. The model is able to learn quickly how these parameters shape the look of a universe and the physical quantity represented by the data. Therefore, using CNNs to infer these parameters within the actual Universe is possible. This is a much simpler process than matching observational data to a set of simulation data, to constrain the values for the parameters; the traditional statistical approach is further complicated by the choice of confidence levels around which to conduct the analysis [39]. However, the inference approach is dependent upon there being a large dataset, which accurately models the entire Universe, available in order to train an AI model.

As will be explored in Chapter 4.1, $(25\ \mathrm{Mpc}\,h^{-1})^2$ maps are not representative of the wider Universe, as they are too small for the cosmological principle [16] to hold. The observable Universe has a side length of $\sim 10\ \mathrm{Gpc}\,h^{-1}$ [48], which is much larger than the maps used in this project. Thus, any conclusions about the validity of inference models on a much larger universe need to account for this, though the is no reason to suggest ML methods cannot gather information about larger length scales with similar levels of accuracy as those shown in this project. Being able to look at smaller length scales may also lead to a better approximation of both cosmological parameters, as the more intricate distribution of matter within a universe may be seen. Therefore, a higher map resolution may also help an AI inference model constrain the cosmological parameters more tightly.

As shown by table 2.5, the dataset being used has an impact on the prediction accuracy. The predictions for the M_{tot} set are lower than the n_e set, which is expected as the two parameters directly impact the matter density within a map. While the two parameters do impact the appearance of the maps in the n_e case, it is not as direct as for the M_{tot} case. It is possible the NN has to conduct more inference on the n_e maps. Interestingly, the model produces very similar errors for both parameters on the n_e dataset. This may indicate that the model is unable to find any more information to aid in its prediction of the parameters, giving a lower limit on the error correction. Maps with a higher resolution may increase the accuracy of predictions from such maps, where the effects of astrophysical processes on n_e may be investigated more closely. However, the model is still able to predict the cosmological parameters to a low error using the more indirect data. Therefore, training an inference model using a form of observational data that is not the total matter density, will still lead to good predictions of the cosmological parameters.

Overall, the model presented in this project is able to infer the cosmological parameters with low errors on multiple CMD sets. Due to the ensemble method being used with the KL loss function, it is possible to compare the data and model uncertainties to each other and see where the limitations of this process are located.

2.5.2 Data Errors

The average predicted standard deviations are low and account for the majority of the errors of the inference approach. Table 2.4 shows that the model's percentage uncertainty due to the data is 2.7% or lower, highlighting that better data can reduce the overall error. While the overall average error for σ_8 is lower than for Ω_M, fewer predictions are within one standard deviation of the correct parameter value, as shown in Figs 2.1, 2.2 and table 2.3. This is due to the model being more certain in its prediction of σ_8, shown in table 2.4. Overall, table 2.2 shows a lower prediction error for σ_8 than for Ω_M. This coincides with the findings of Villaescusa-Navarro et al, who also found that inference models were better suited to find the value of σ_8 than Ω_M, at least on the M_{tot} dataset.

The main source of the data errors is most likely the map size. The model is able to predict σ_8

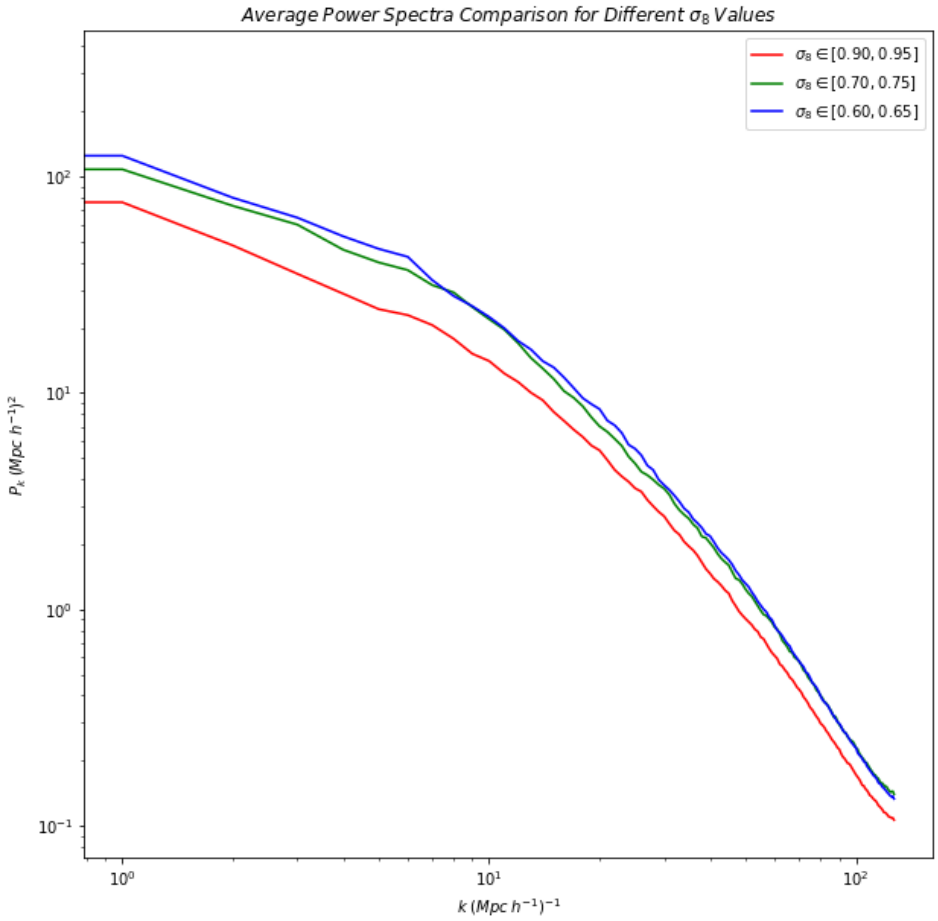


Figure 2.3: The average power spectra, P_k, of M_{tot} maps with different σ_8 value ranges.

reasonably well, yet there is still some uncertainty and error in the its prediction. Given that σ_8 represents the amplitude of matter density fluctuations over a sphere of radius 8 Mpc h^{-1}, the ability of the model to constrain the value of σ_8 will be limited due to the size of the maps. The maps do not represent enough space to fully capture the effects of σ_8, especially when comparing those maps for which σ_8 varies only slightly. Increasing the size of the maps may reduce this error.

However, even in a relatively small map, σ_8 values of 0.6 and 0.9 will produce universes that will appear quite different. Less clustering in the 0.6 case will give a more homogeneously populated universe, which will especially be apparent on the M_{tot} map, allowing for the good level of inference by the model. This is shown by Fig 2.3, where different average power spectra are shown from three sets of maps with different σ_8 values from the M_{tot} set. An average of maps with σ_8 within these ranges is taken due to the limitations of the Latin Hypercube set, where direct comparison of individual parameters is difficult. The power spectrum of $\sigma_8 \in [0.60, 0.65]$ (blue) can be differentiated from $\sigma_8 \in [0.90, 0.95]$ (red) more easily than from $\sigma_8 \in [0.70, 0.75]$ (green), for maps of an area of (25 Mpc h^{-1})^2. The different σ_8 values are easiest to distinguish at small k modes, i.e. for larger wavelengths. To access larger wavelengths, in order to distinguish the σ_8 values more easily, a larger map is needed.

Ω_M, which measures the matter density within a universe, has a higher relative data uncertainty and a higher overall average prediction error than σ_8. While a similar argument holds as for why the overall prediction is low, the CNN is less accurate and more uncertain in its prediction of Ω_M. The maps in CMD do not show larger galaxy clusters nor voids, which will affect the prediction of Ω_M more so than σ_8. Without large voids, a simulated universe will appear to be much more densely populated by matter. The cosmological principle does not hold and a (25 Mpc h^{-1})^2 area

does not have a matter density that reflects that of the whole universe, due to the absence of certain structures. Therefore, the inaccuracy in the prediction of Ω_M may also be reduced with larger maps, which represent the overall universe much more accurately.

The error due to the data is difficult for a NN to predict, given it cannot be provided with 'true' error values to which it can compare. This is an important consideration for if ML is used in physical research. This project demonstrates that NNs are able to produce uncertainties from datasets, highlighting their research potential. Yet to be able to utilise this feature of NNs, the uncertainty of the model itself needs to be analysed, especially the uncertainties of the data error estimates.

2.5.3 Model Errors

Overall, the model uncertainties are much lower than the data error estimates, as shown by table 2.4. Given that the data uncertainties are predictions from the model itself, it is difficult to fully separate the two sources of uncertainty. However, the model uncertainty of the predicted standard deviation shows that the model is approximately 95% certain in its calculation of the data error. Table 2.4 also shows that the relative percentage uncertainties, due to the model itself, are less than 0.8% for the predictions of both parameters. The model is quite certain in its prediction of the mean parameter values, with small error bars around each point, shown in Figs 2.1 and 2.2.

For both parameters, $\sigma \pm s_\sigma$ leads to a change in the relative percentage uncertainty by at most 0.1%. Also, when model errors alone are taken into account, few of the mean predictions lie within an error bar of the expected result. This highlights that the data errors are the main limitation of this approach, not the model's architecture. While being aware of the model's own error is important, knowledge of the data's error is far more vital.

The uncertainty of the model is due to how backpropagation works. There are many minima, for which the weights can be optimised. It is extremely unlikely for a fresh model to find the same local minima as a previous iteration, leading to different optimisation pathways and different values for the network's weights. Yet, as shown by table 2.4, this is not an issue as the spread of predictions is low and overall model accuracies are high. This highlights that the global minima need not be found for an accurate network, as many of the local minima lead to predictions that are accurate.

Combining the results for the overall accuracy of the model with the low percentage uncertainties due to the model's architecture, it can be concluded that the approach taken in this project is a useful inference tool for cosmological data. The main source of errors lie within the data and the uncertainties within the cosmological maps. However, for smaller data uncertainties, model uncertainty becomes more apparent. Therefore, it is important that an approach exists which can produce the uncertainties of the model's architecture.

Chapter 3

Generating Data with Diffusion Models

3.1 Introduction

Current simulations enable physicists to evaluate the accuracy of various cosmological models by comparing them against observed data. Generative AI, namely diffusion models, offer a novel approach for creating simulation maps, offering new possibilities for testing and refining these theoretical models.

A benefit of using classifier-free denoising diffusion implicit models is that a single model trained on a dataset can produce simulation data with different cosmological parameter values, without needing to be re-trained. The neural network learns to represent the relationship between the parameters and the output data. Additionally, generative AI can learn directly from real observational data, allowing it to reproduce data which is more representative of observed phenomena. In this project, a classifier-free guidance denoising diffusion implicit model (DDIM) is trained on data from the CAMELS Multifield Dataset (CMD), in order to produce 2D galaxy cluster maps that are representative of CMD data. Therefore, generated maps will have cosmological parameters which follow those in CMD and should have similar summary statistics to the real dataset.

The DDIM models created in this project are able to produce maps which follow certain cosmological parameter values. This is even the case for values not necessarily in, though closely related to, the training dataset. For example, the model can generate maps with $\Omega_m = 0.35$ even though there are no such maps in CMD. Another important application of generative AI in cosmological research, is the high level by which the model can be customised for a specific goal, allowing researchers from many different fields to use DDIMs for interdisciplinary applications.

3.2 Method

A classifier-free conditional DDIM with a U-net architecture is explored in this project. The U-net architecture employs 4 blocks with widths 32, 64, 96, and 128, as this was found to produce the highest quality data. The inputs to the model are noisy images produced in the forward diffusion process, created by gradually adding noise to the 2D grids from the CMD dataset. This includes the total matter density, M_{tot}, maps from the IllustrisTNG [11] hydrodynamic simulation set and from the SIMBA [12] N-Body set.

The majority of results shown in this subsection are for CMD maps downsampled to a pixel size of 64x64 from 256x256, unless specified otherwise. This downsampling is primarily utilised to accelerate the training process, allowing for a quicker and more efficient investigation into the impact of changes to variables and model architecture on performance. This is due to the fact that full-scale 256x256 models take a much longer time to train. Training for a single epoch found this

amount of time to be roughly 16.4 times that of 64x64, and also requires gradient accumulation [49]. Gradient accumulation is when the model applies gradient steps over the average of mini-batches. This method was employed due to lack of GPU memory on the machine hosting the DDIM.

The following variables are investigated: model architecture, which heavily influences how well the model can learn from training data; the number of sampling diffusion steps and reverse diffusion steps, which both represent the balance between sample quality and computational demand; conditioning toggle, the value for the sampling guidance strength and 1 minus the probability to randomly drop conditioning information; conditional parameters, the cosmological parameters used to generate new conditional data; simulation type, N-Body SIMBA and hydrodynamic IllustrisTNG M_{tot} maps are both used to generate new data to see how the model performs on different datasets. Each of the above variables is optimised to produce the most accurate data possible, while also taking into account sampling time and exploring the limitations of each method used.

The results of each subsection are put through a testing program, explained in Section 3.2.1, which includes standard generative AI metrics and tests based around the physical data. The generated images are compared to maps from CMD, which are of the same simulation type and have similar cosmological parameters.

3.2.1 Testing Methods

To verify the validity of the generated maps, for both the conditional and unconditional methods, four tests are carried out. Two of these are standard metrics used to assess images generated by generative AI methods, while the remaining two are physical tests.

The two metrics used to test the validity of generated AI images are the Fréchet inception distance [50] (FID) and the kernel inception distance [51] (KID). Both tests measure the similarity between two multivariate Gaussian distributions and are widely used in generative AI research. The distributions are created by feeding two sets of images into the pre-trained InceptionV3 [52] CNN, from which a set of activations from the penultimate layer of the network is received per image set. These activations are a representation of the images in latent, or feature, space; images that are a close distance to each other in this space visually appear similar in real space. This means a low FID or KID score indicates similarity between two image sets, with a score of zero meaning the two sets are identical.

FID is a 2^{nd} order measure, comparing the means and covariances of the distributions:

$$\text{FID}(A, B) = |\boldsymbol{\mu_A} - \boldsymbol{\mu_B}|^2 + Tr\left(\boldsymbol{C_A} + \boldsymbol{C_B} - 2(\boldsymbol{C_A}\boldsymbol{C_B})^{1/2}\right) \tag{3.1}$$

Where $\boldsymbol{\mu_A}$ is the mean and $\boldsymbol{C_A}$ is the covariance matrix of the Gaussian distribution, A.

A known fault with FID scores is that large image sets are required for accurate scores, otherwise it is said to be biased. Heusel et al [50] recommend that at least 10,000 samples are used to generate the activation distributions, on which to measure FID. As such, the FID score is not used alone to indicate whether the generative process is working correctly. KID is an additional test that is conducted.

KID is also a distance in latent space that uses a kernel to approximate the distribution to an infinite order. In most ML applications the kernel used is 3^{rd} order, with the following form:

$$K(\boldsymbol{a}, \boldsymbol{b}) = \left(\frac{1}{d}\boldsymbol{a}\boldsymbol{b}^T + 1\right)^3 \tag{3.2}$$

Where $\boldsymbol{a} \sim A$, $\boldsymbol{b} \sim B$, and d is the number of elements of both $\boldsymbol{a}$ and $\boldsymbol{b}$.

From this kernel (3.2), the KID may be calculated:

$$\text{KID}(A, B) = \overline{K(A, A)} + \overline{K(B, B)} - 2\overline{K(A, B)} \tag{3.3}$$

Where $\overline{K(A, B)}$ is the mean value of $K(A, B) = \{K(\boldsymbol{a}, \boldsymbol{b}) \ : \ \boldsymbol{a} \sim A, \boldsymbol{b} \sim B\}$

KID is not biased with regards to the size of the image set, although it has a higher variance of scores [53]. Therefore, both tests are employed to test the validity of generated images.

To use these metrics to test the validity of the results, the following reference values, table 3.1, were calculated by comparing two subsets of M_{tot} maps. Achieving similar, or lower, scores, by comparing generated maps to real data, indicates that the generated maps are realistic.

Inception Distance	**Reference Value Range**
FID	2.0 - 3.0
KID	10^{-5} - 10^{-4}

Table 3.1: FID and KID reference scores.

Industry standard AI metrics are a good measure of how images look visually, but do not indicate if the underlying physics is being learned by the generative model. Given that cosmological data is being created, the power spectrum and pixel count of the generated and real maps may be calculated and compared. Sets of maps generated by either the conditional or unconditional model can be compared to real datasets with similar cosmological and astrophysical parameters. Close alignment between the average power spectra and average pixel values of each set indicates that the generated maps are realistic. For testing of conditionally generated maps, the real dataset being tested against needs to represent the desired value of the conditioned parameter. For unconditional data, the real dataset needs to be a sample of the training data given to the generative model.

As mentioned, the power spectrum is an extremely useful tool for analysing cosmological data. It is calculated from the squared amplitudes of the 2D Fourier Transform of an image, which can be plotted against different values of the wavenumber k, with units $(\mathrm{Mpc}\, h^{-1})^{-1}$. Given that the images represent a physical quantity, in this case the total matter density M_{tot}, it is expected that maps with similar initial parameters will have similar distributions of pixel values. Therefore, histograms of the pixel values for both real and generated data will be similar.

The pixel count histograms and power spectra are analysed using the chi-square test [54]. For the pixel count, the p-value from a null hypothesis (H_0) that the histograms are the same is displayed. This is useful as it can be determined whether conditioned generated maps have the same summary statistics as the real data with the same parameter values, and also to compare that generated maps with different parameter values have different summary statistics. It is taken that a p-value less than 0.05 indicates that there is sufficient evidence to suggest that the results are different, and H_0 is rejected.

The generated maps in this project do not accurately reproduce the power spectra at large k values. To investigate and draw conclusions from this discrepancy, the highest k value for which the null hypothesis, that the power spectra match, is still valid. This is achieved by verifying that the p-value from the chi-square statistic exceeds 0.05. This is done by first truncating the power spectrum to only the first data point and then truncating one fewer data point, up to a maximum of 32 points, until the p-value is less than 0.05. This should give a clear indication of at what k value the model begins to fail to reproduce the power spectra accurately. Furthermore, when testing conditioned data, maps from CMD are taken which match the intended parameter value ± 0.01.

Importantly, no single test is used to prove that the generated images are realistic. By comparing real and generated data using these four tests, and if all tests show alignment between the two image sets, it can be concluded that the generated physical data is realistic.

3.3 Results

3.3.1 Diffusion Steps

The following results are obtained from an unconditional DDIM. A larger number of diffusion steps leads to higher quality generated samples, however this comes at the trade-off of a larger sampling time. Therefore, the number of diffusion steps can be adjusted to either rapidly test models, and the resulting generated images, or to generate higher quality samples. Sampling times here were averaged over the generation of 1000 maps. For these results, the following table (3.2) is presented. PC represents the p-value for the null hypothesis that pixel counts of real and generated maps match. P_k represents the percentage of the total power spectra that the model can reproduce to a statistically significant degree.

Diffusion Steps	PC	P_k	FID	KID	Sampling Times (s)
20	0.0	0%	1.48	0.00241	2.16
50	2.34e-07	0%	1.81	0.00300	5.10
100	1.64e-6	0%	1.99	0.00321	10.2
500	0.101	13%	1.99	0.00330	51.2

Table 3.2: The results of adjusting the diffusion steps during the sampling process.

3.3.2 Conditional Toggle

Increasing the conditional toggle leads to a greater adherence to the conditioning information, producing samples with the desired parameters to a higher accuracy. The following model is only conditioned on Ω_m and results are averaged over those for when the model is asked to produce maps for Ω_m equal to 0.15, 0.25, 0.3 and 0.35.

Conditional Toggle	PC	P_k	FID	KID
0.0	0.101	13%	1.99	0.00330
0.25	0.0708	25%	1.96	0.00332
0.5	0.355	31%	2.00	0.00337
1.0	0.685	31%	1.99	0.00334

Table 3.3: Analysis of the affect of different conditional toggles on learning and image generation.

3.3.3 Conditional Parameters

The model is trained on different cosmological and astrophysical parameters and combination of parameters, with a conditioning toggle of 1.

Conditional parameters	PC	P_k	FID	KID
Ω_m	0.685	40%	1.99	0.00334
Ω_m and σ_8	0.106	13%	2.01	0.00339
Astrophysical	0.0	0%	3.33	0.00674

Table 3.4: Analysis of the use of combinations of conditional parameters on the learning and image generation. Model trained on SIMBA N-Body M_{tot} maps.

3.3.4 Map Types

The following maps are generated using 128x128 M_{tot} Illusris and SIMBA N-Body data from CMD. The resolution is half of that of the maps within CMD. The U-net architecture used has widths 32, 64, 96, 128, 160 and 192, and a block depth of 6. The models are trained with the best results of

the above tests. Training the model on 128x128 decreases similarity of pixel counts, while keeping power spectrum similarity the same. However, for both SIMBA and IllustrisTNG trained models, the power spectra shows a 9% closer match at the highest k value. This could possibly imply that the model more easily picks out smaller-scale features present in the original datasets, due to the increased resolution, but the model is not optimised to generate the best results. However, this was not investigated, due to time constraints.

Map Type	PC	P_k	FID	KID
SIMBA	0.0781	31%	3.52	0.00337
IllustrisTNG	0.137	31%	2.62	0.00445

Table 3.5: Analysis of the effectiveness of using the optimised training parameters on different map types from CMD.

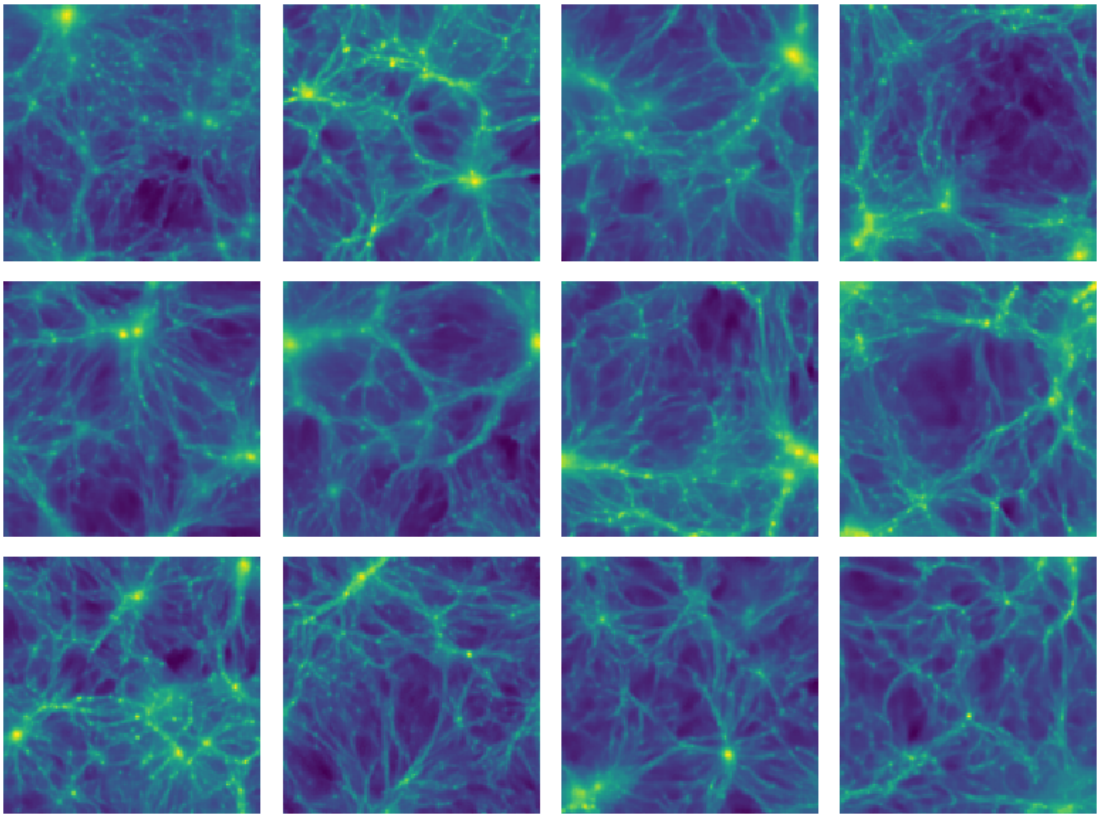


Figure 3.1: 128x128 generated images from the IllustrisTNG M_{tot} dataset

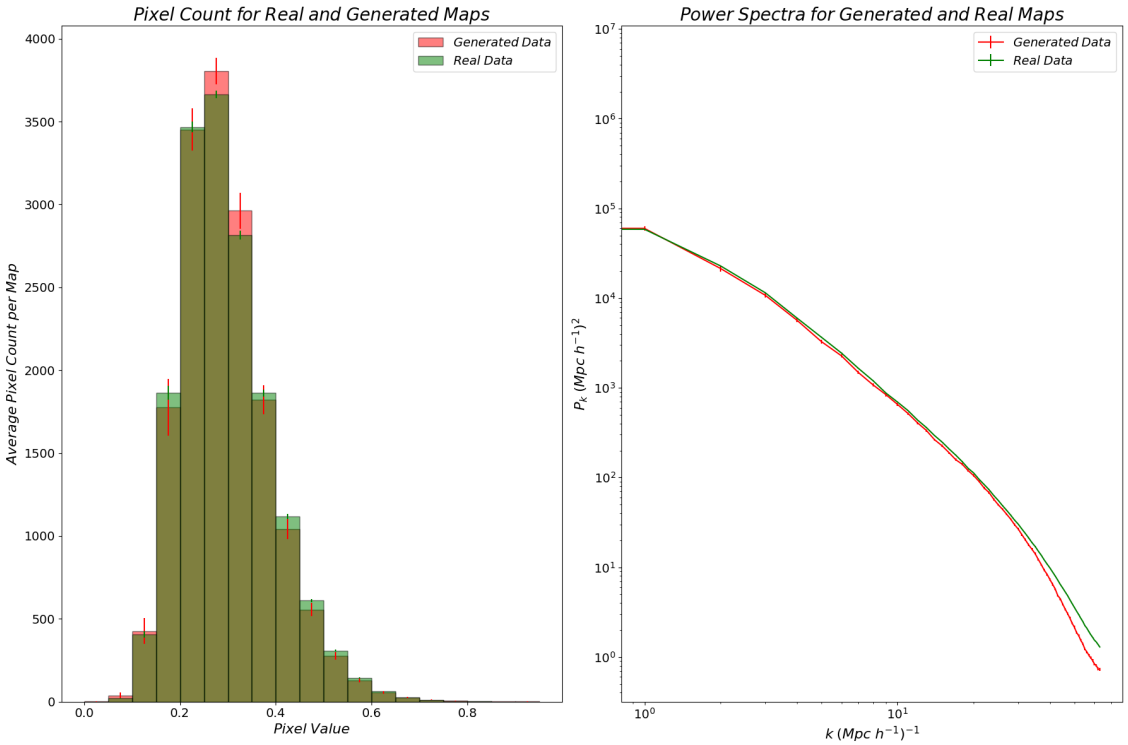


Figure 3.2: Pixel count and power spectra graphs for 128x128 generated images from the IllustrisTNG M_{tot} dataset 3.1

3.4 Discussion

This section outlines how changing different variables within the conditional DDIM model can lead to varying performances of the model to generate galaxy clusters from CMD data. It is important to mention, that none of the generated maps are present within the original dataset, and therefore the model is correctly learning to generate new maps.

The simplest way to improve the generated map quality is to increase the number of diffusion steps, as discussed in Section 3.3.1. An increase of which appears to raise the sampling time by 0.1s per generated map. To sample images with 500 diffusion steps, it requires 22.2 GPU hours to produce 100,000 64x64 $(25\ \mathrm{Mpc}\,h^{-1})^2$ maps. Therefore, such maps can be generated without having to do large hydrodynamic or N-body simulations of 256^3 particles from $z = 0$ to $z = 127$. Sampling with 500 diffusion steps leads to a large increase in sample quality, so this is likely much more desirable in most situations. The FID and KID scores for these results are relatively low. The FID scores are lower within expected ranges, while the KID scores are higher. This is most likely due to having to reduce the resolution of the images and the fact that not many generated images were compared to the real dataset, which is a known issue with the inception scores, as discussed in Section 3.2.2. It is most likely that the KID scores are more accurate than the FID scores, though neither test should be relied upon in isolation.

For the fully conditioned model with conditioning toggle = 1, the fact that the pixel counts match, and power spectrum up to 31% of the total spectrum, averaged over different Ω_m values, indicates that the conditional model is correctly implementing the conditioning information in the generation process. Furthermore, when comparing the generated maps for Ω_m equal to 0.25 and 0.35, the p-value for the pixel count histograms being statistically the same is 0.0001, implying no statistical correlation between these generated maps. However, individually they have p-values of 0.349 and 0.646 respectively when compared to real CMD data, implying statistical correlation.

It is shown that lower conditioning toggles can still create high-quality maps in Section 3.3.2, statistically matching the pixel count. Therefore, using a lower conditioning toggle is a completely viable option, and it is also assumed that decreasing the conditioning toggle should increase sample diversity [37]. A higher sample diversity is useful for generating universe catalogues, creating universes with a varied range of cosmological parameters.

The results from the classifier-free guidance conditional DDIM outlined in Section 3.3.3 clearly indicate that the model is able to learn from the cosmological parameters Ω_m and σ_8, however struggles to produce reliable results for the astrophysical parameters. This follows the results of Chapter 2. Applying a chi-square test to pixel count and power spectra for generated maps, it can be seen that the model can accurately learn the pixel count of the maps but struggles to reproduce the power spectrum. This implies that the model fails to learn small-scale features of the input maps. Therefore, a likely reason for this failure in the model's training is due to the resolution of the maps that the model is trained on. Training the model on 128x128 decreases similarity of pixel counts when comparing generated to real data with the same parameter values, while keeping power spectrum similarity the same. However, for both SIMBA and IllustrisTNG trained models, the power spectra show a 9% closer match for the highest k value. This could imply that the model can pick out smaller-scale features more accurately, but that it is not optimised to generate the best results. However, this was not investigated, due to time constraints. Therefore, it is a possibility that training the model on full-scale (256x256) images, may lead to greater similarity scores between power spectra. Further improvements to the model, outlined in Section 4.2, should enable the generation of maps which more faithfully reproduce the power spectra of real CMD maps.

These results show that the use of classifier-free guidance DDIM models have the potential to produce high-quality data, while also displaying some of the customisable features of the model and their trade-offs. Whether the goal is to achieve fast sampling times, enhanced image quality, or highly conditioned maps, this model offers adaptable solutions to meet the diverse requirements within the world of physics research.

Chapter 4

Final Discussions and Conclusion

4.1 Dataset Limitations

As has been mentioned, the dataset used in this project does not accurately represent the Universe. Although this project serves as a demonstration of the ability of ML methods to aid cosmological research, the limitations of the dataset needs to be addressed in order to make appropriate conclusions. There are three main limitations of the CMD maps that cause these inaccuracies: the relatively small box size, the number of particles per box, and that each map has a mean mass density that is the same as the mean mass density of the entire Universe. These limitations lead to a constraint on the total power contained the maps; they also mean that the cosmological principle cannot hold.

The cosmological principle [16] states the matter distribution in the universe is isotropic and homogeneous when considered over a large enough area. The principle means that the whole Universe may be composed of smaller map tiles joined to each other, which cannot be done with maps in CMD. This is because of both the missing power, due to map size, and the mean mass density considerations.

The mean mass density of all maps in CMD is equal to that of a hypothetical wider universe, by construction. In a simulation with map side lengths much larger than 25 $\text{Mpc}\,h^{-1}$, taking $(25\ \text{Mpc}\,h^{-1})^2$ boxes from within the larger map will lead to submaps with mean densities that are not always equal to the overall matter density of the wider universe. Submaps that contain mostly void will have a much lower mass density; while submaps filled with galaxy clusters will have a higher density. Thus, the CMD maps cannot accurately represent the whole Universe when tiled together, simply because, for a larger universe, each $(25\ \text{Mpc}\,h^{-1})^2$ area does not have a mass density equal to the overall mean mass density of said universe.

This is highlighted by Fig 4.1, which shows an N-Body, dark matter only, simulation of 256^3 particles with a 1 $\text{Gpc}\,h^{-1}$ side length. The data is taken from the Quijote simulation set [55], which is similar to CAMELS. Within this much larger map, which shows the dark matter mass density, $(25\ \text{Mpc}\,h^{-1})^2$ submaps are drawn at random. Fig 4.1 shows how small the CMD maps are compared to a much wider universe. Due to its size, a $(1\ \text{Gpc}\,h^{-1})^2$ map is much more representative of the whole Universe; the cosmological principle holds for such a map size.

By assuming a $(1\ \text{Gpc}\,h^{-1})^2$ map contains all the power of the Universe, comparisons may be made between the power contained within a $(25\ \text{Mpc}\,h^{-1})^2$ map. This is shown in Fig 4.2a. A $(25\ \text{Mpc}\,h^{-1})^2$ map contains approximately 70% of the total power of the universe, showing how the CMD maps miss much of the power within a simulated universe. Submaps of $(150\ \text{Mpc}\,h^{-1})^2$ contain nearly all the power in the $(1\ \text{Gpc}\,h^{-1})^2$ map; they would be much more representative of a wider Universe. A dataset composed of such maps would be more suited for the training of AI models. The missing power in the $(25\ \text{Mpc}\,h^{-1})^2$ maps is due to both the small and large scale mode considerations.

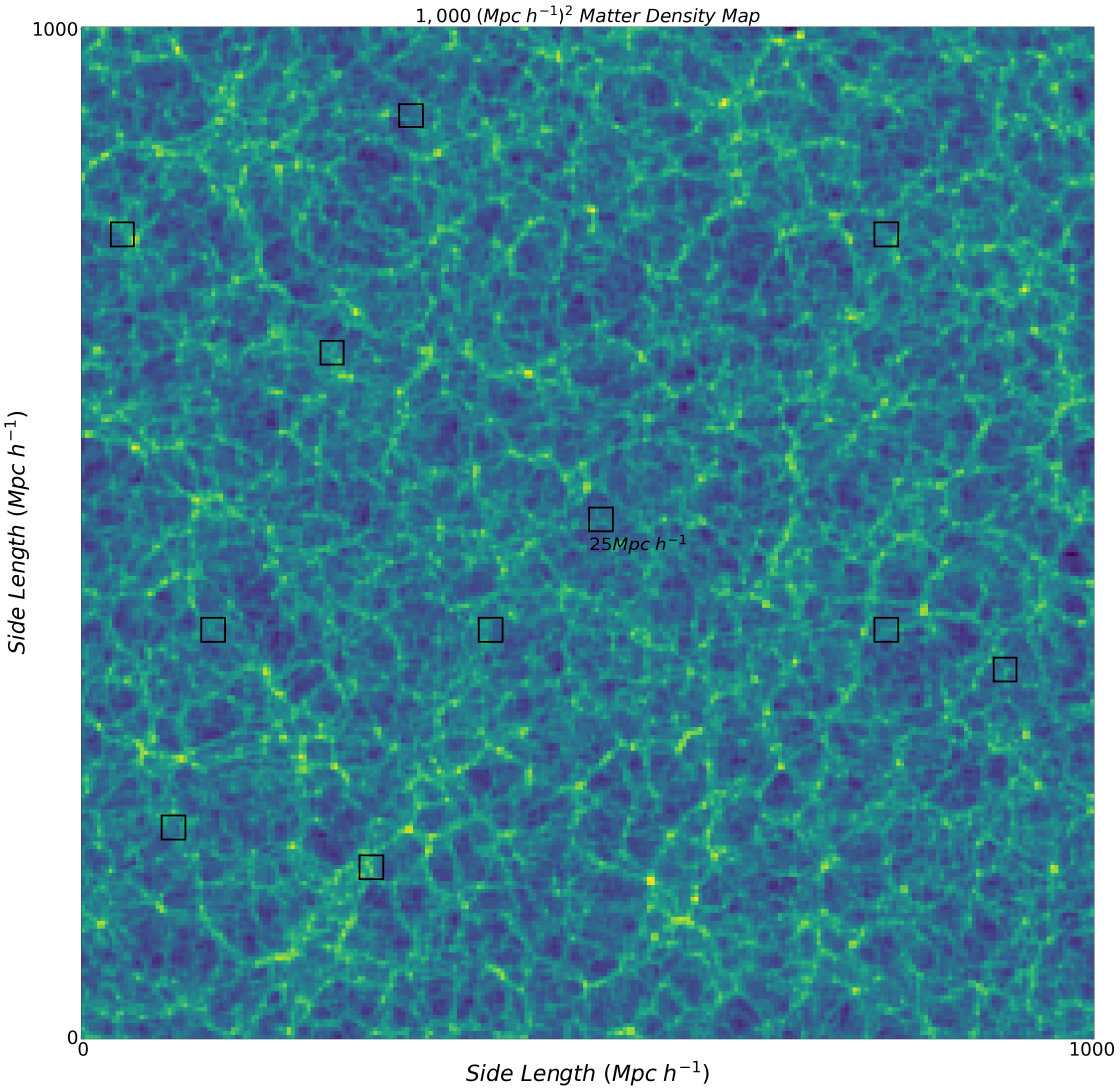
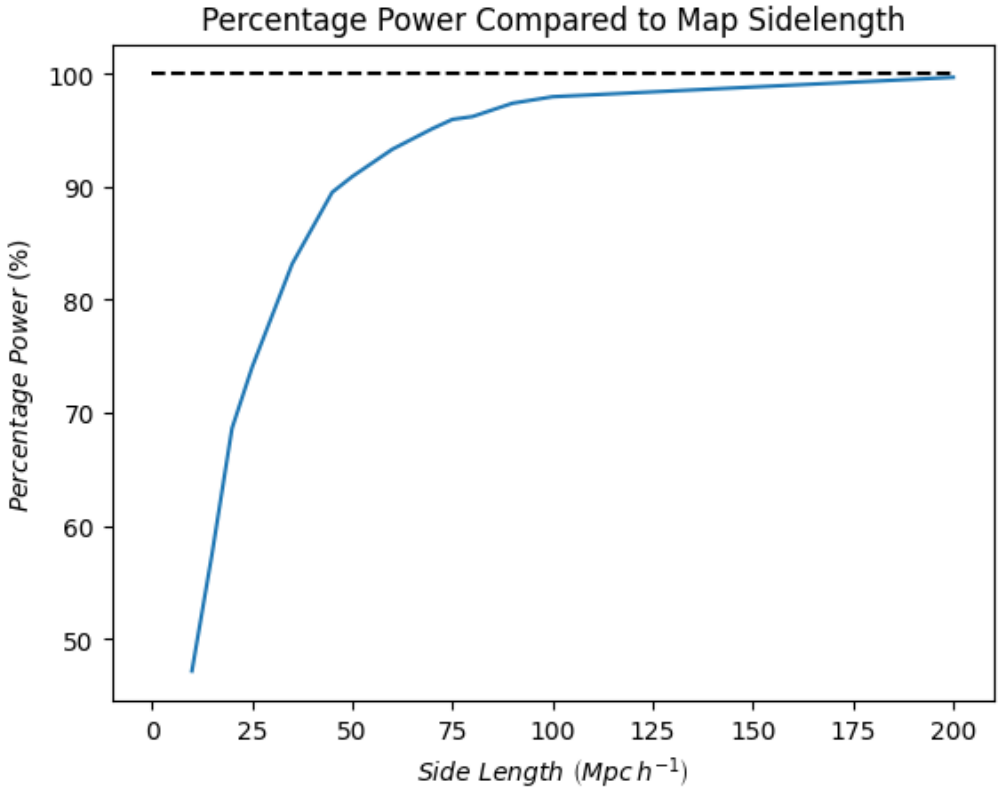


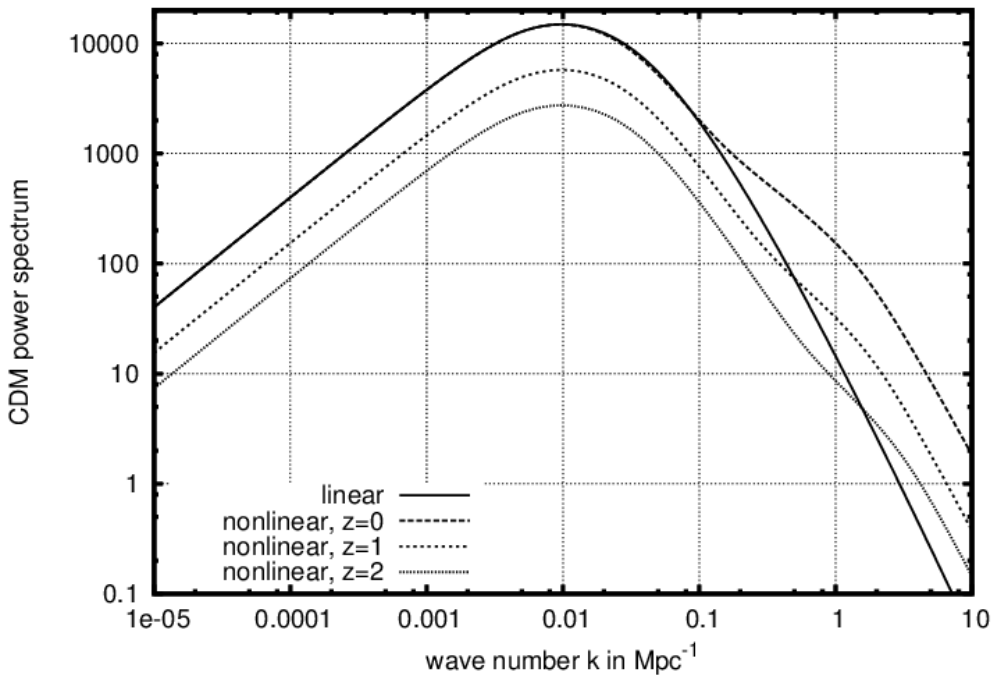
Figure 4.1: A $(1,000\ \mathrm{Mpc}\,h^{-1})^2$ N-Body simulation of 256^3 particles showing the dark matter density. A brighter colour indicates a region of higher matter density. $(25\ \mathrm{Mpc}\,h^{-1})^2$ submaps, representative of CMD maps, are highlighted at random.

The $(25\ \mathrm{Mpc}\,h^{-1})^2$ boxes cannot contain large scale modes in the matter power spectrum. There are k-values which cannot be accessed by a box of this size. Shown in Fig 4.2b, $(25\ \mathrm{Mpc}\,h^{-1})^2$ corresponds to $k \approx 0.25\ (\mathrm{Mpc}\,h^{-1})^{-1}$, meaning much of the power corresponding to small k-values is missing from the CMD maps. The CMD maps represent the more standard objects in the universe, those being galaxy clusters and very small voids. They cannot show large collapsed objects [17], e.g. super-clusters, nor large voids, which have a volume much larger than $(25\ \mathrm{Mpc}\,h^{-1})^3$. Therefore, some physics, related to these larger structures, is missing from CMD, which can only be seen with larger maps. However, these maps are more computationally demanding to produce; creating a large training dataset, with maps of this size, for AI models is not as easy as for the $(25\ \mathrm{Mpc}\,h^{-1})^2$ case.

Small scale modes are also limited by the resolution of the maps. By simulating more particles, e.g. 1024^3, more of the finer details of the universe can be seen, such as the distribution of matter on smaller scales. The CAMELS simulations cannot resolve length scales smaller than approximately $1\ \mathrm{kpc}\,h^{-1}$ [10], corresponding to $k \sim 10^4\ (\mathrm{Mpc}\,h^{-1})^{-1}$; this means that some of the large k power is also missing from these maps. This is a common problem in cosmological simulations, especially with the constraints imposed by the $O(N \log N)$ complexity for N-Body simulations. However, ML methods may be used to rectify this issue by increasing the resolution of existing maps [41], which is one of the goals of the wider CAMELS project.



(a) The amount of power contained within maps of a given side length relative to a wider $(1\ \mathrm{Gpc}\,h^{-1})^2$ map, which is representative of the whole Universe.



(b) The total linear and non-linear power spectra of the Λ-CDM (cold dark matter) model. Modes smaller than $k \sim (0.25\ \mathrm{Mpc}\,h^{-1})^{-1}$ are not contained within CMD maps. The plot is taken from Bartelmann's "The Dark Universe" [56].

Figure 4.2: Power spectrum comparisons for CMD maps.

4.2 Improvements in Machine Learning

The points within the CMD maps have dependencies on all other points within the map, including those at the opposite sides of the map. A transformer architecture might lead to a large increase in performance of the DDIM model. The Transformer model relies entirely on an attention mechanism to draw global dependencies between input and output data [57]. The self-attention mechanics used in the transformer architecture allows each part of the input data to interact directly with every other part. This also occurs due to how the input data is encoded; the specific sequence or location of data points does not matter. Improvements to the original visual transformer architecture have been made by also including the U-net architecture, as shown by He et al [58]. They showed that the their improved U-netmer++ (U-net transformer) design represents a 5.3% improvement for their image segmentation task [58] in the Dice similarity coefficient, a statistical measure quantifying the similarity between two sets.

Therefore, using self-attention for the parameter inference task in Chapter 2, should offer an improvement over using traditional CNN architecture [57]. It is also reasonable to assume that a similar improvement will be seen for the image generation task in Chapter 3, as the U-net in the DDIM may be replaced by a U-net transformer.

Furthermore, the CMD 2D grids are derived from the larger 3D grids from the CMD. Diffusion models can learn 3D features, as shown by Luo et al [59]. They used diffusion methods to model a point cloud, consisting of N points, as a set of particles in an evolving thermodynamic system, which could conceivably be thought of as particles. Furthermore, a more recent paper on 3D diffusion models: "LION: Latent Point Diffusion Models for 3D Shape Generation" [60], uses course voxel (3D pixel equivalent) inputs, converts them to latent points (an abstract, lower-dimensional space derived from the data), reconstructs clean point clouds, and then reconstructs meshes. As shown in this project, diffusion models struggle to pick out the finer details of the particle grids; this could be improved by LION. LION "sharpens" the point clouds into meshes, allowing for the model to identify the sharper details.

Therefore, the next steps of research into using classifier-free guidance DDIM models for the generation of images based on the CMD would be to use a U-net transformer architecture, in order to generate higher-quality samples, and use LION to generate 3D samples.

It is also worth mentioning that there are two severe limitations in the use of machine learning within physics. Firstly, as discussed in Section 4.1, ML models are only as good as the data that it is trained on, therefore they needs high-quality data to make accurate predictions, and cannot learn any information that is not present within the input data. The second point, is that the models also need a large quantity of high-quality data, so that the models can generalise to unseen data. There are techniques to overcome this problem, e.g. image flipping and rotations for isotropic data. Furthermore, a model can be trained to exploit known conditions of the dataset, for example using periodic padding to images. Additionally, if the output data must be in the set $[-1, 1]$, a tanh activation function could be used on the final layer, automatically scaling the data between -1 and 1. Therefore, for the case in section 2 and 3, periodic padding with any angle rotations could be used. This would allow for the models to be trained to learn the isotropic nature of the data more thoroughly.

In summary, there is a large scope for improvements to the model, which would not only enable it to better learn the complex relationships between data points across entire maps but also extend its learning capabilities to 3D maps from the CAMELS Multifield Dataset. The inherent flexibility and customisability of neural network architectures and overall model structures enable these advancements. By exploring various modifications and employing cutting-edge techniques, generative AI can be finely tuned, in order to meet certain demands of the problem. This adaptability underscores the transformative potential of machine learning in advancing the frontiers of scientific research.

4.3 Machine Learning in Physical Research

Both Chapters 2 and 3 highlight the usefulness of ML in physical research, especially in cosmology. Inference models are an obvious use of ML methods, provided large training datasets exist. They are useful tools for analysing complex data in a quick and inexpensive way, with a high level of accuracy, as shown in this project. The overall CAMELS dataset has also been used to train a NN to investigate sub-halo properties [61], with a higher accuracy than traditional regression techniques. NNs have also been used to predict redshifts from real galaxy data, without the need for expensive spectroscopic measurements [62]. Advancements in AI methods will lead to more accurate and wide-ranging uses of ML for cosmological and astrophysical inference, including being used with large three-dimensional (3D) datasets.

Diffusion models have been shown to generate realistic data in this project. While the maps produced are limited by their size and resolution, and so do not accurately represent the entire Universe, this project demonstrates that the concept of producing realistic cosmological data using AI methods is possible. Generative methods may also be used with existing traditional methods to accelerate the simulation process. An example is providing N-Body simulation data to a diffusion model, which can then output hydrodynamic data [63] by analysing the relationship between dark matter halos and regular matter. This is currently done via sub-halo abundance matching (SHAM) [64], though could soon be replaced by generative methods [7]. SHAM produces mixed results and is better for larger length scales, those of 2 Mpc h^{-1} per side or higher, which should not be an issue for ML methods.

The main advantage of using generative methods, especially diffusion models, is that data is produced in very short timescales. These models can be conditioned to produce maps with a range of desired characteristics, allowing for testing of cosmological hypotheses by producing large universe catalogues with desired characteristics. While the data generated in this project is two-dimensional, the recent advances in 3D diffusion [65][66] methods may lead to generation of 3D cosmological data.

It is worth noting that the ML methods presented in this project are black-box approaches. The programs are given input data and provide an output, which solves a particular problem. The mechanics behind this process are difficult to ascertain. Although this is an incredibly useful tool for data analysis, the underlying physics of the predictions made is difficult to see and may not provide an insight into why the Universe behaves as it does.

Training a NN with the aim of finding regression equations may be a way to open the black-box; analysing why the network chose certain equations or parameters may give insight into the underlying physics. Saliency analysis [67], where the most important terms within the input data can be highlighted, may be used to gather insight into how the network is coming to its conclusions by seeing which parameters, in the input data, the model thinks are the most important. This gives an insight into how the models think. Investigating why networks produce certain predictions may direct future research, provide understanding of the underlying physics, and improve traditional simulations.

The methods used in this project can be expanded to a wide range of areas of physics and mathematics, especially for data interpretation, modelling, and generation. For example, in solid state physics, ML applications have been used to investigate material properties [68]. Complex, high-dimensional mathematical research, such as in topology and graph theory, will also benefit from AI tools [69]. Similarly, AlphaFold [70], an AI program developed by DeepMind, has been used to investigate protein structures, significantly aiding disease research.

The potential usage of AI in scientific research are numerous and could be revolutionary. ML methods are incredibly potent tools for data analysis in all areas of research and industry. With a greater understanding of how these models think, the underlying principles of scientific processes may be better understood with the help of ML models.

4.4 Project Conclusion

The results presented in Chapter 2 show the data inference capabilities of ML methods. The relative ease by which the model is able to learn the effects of the cosmological parameters Ω_M and σ_8, with prediction errors of 2.69% and 1.38% respectively, highlights the usefulness of ML in cosmological research. Additionally, Chapter 2 presents methods which can calculate associated uncertainties for its predictions, both due to the data and the model architecture, showcasing the versatility of using such methods in broader physical research. Chapter 2 also highlights how the data used to train AI models must be suitable when any predictions are made; the data must be representative of the entire Universe, if it were to be used to make predictions about the Universe. Overall, inference ML models can be employed in many areas of scientific data analysis. However, uncovering the black-box nature of NNs is an important step in order to use such networks for wider scientific goals, which may aid in the discovery of new underlying physics and scientific processes.

The results shown in Chapter 3 demonstrate that generative ML models are able to produce new cosmological data, which is consistent with existing datasets. This may be done via conditional and unconditional methods, and this allows for the production of vast universe catalogues, once a model is trained on a suitable dataset. This is shown via the physical tests conducted on the maps generated: shown by the the power spectrum and pixel count comparisons in Fig 3.2, and by the analysis in Chapter 3.3. The largest limitation was, again, CMD: large scale objects, such as large voids, could not be generated; the map resolution also meant that smaller scale structures could not be reproduce by the generated maps, shown by the deviations in the power spectra at large k values. Hardware was also a limiting factor, given that CMD maps had to be reduced to a lower pixel resolution for the model to work. However, with a more representative dataset and more sophisticated hardware, the generation of new, useful cosmological and astrophysical data by diffusion models is possible.

To conclude, this project demonstrates that machine learning methods, particularly inference and generative models, are a useful tool in cosmological research. The models are able to learn how the composition of a universe is underpinned by certain cosmological parameters. Using this knowledge, they can accurately generate new physical data, when provided with a large training dataset. The biggest consideration when using these methods for research is that realistic and representative data must be used to train the models in order to produce accurate results. Provided such data is available, the potential uses for ML in scientific research are extensive and potentially revolutionary. Once the inner-workings of such processes are understood more clearly, powerful machine learning techniques may lead to great advances in the way scientific research is undertaken, allowing for a greater understanding of the Universe.

Bibliography

[1] Mark Vogelsberger, Federico Marinacci, Paul Torrey, and Ewald Puchwein. Cosmological simulations of galaxy formation, 2019.

[2] Volker Springel, Simon D. M. White, Adrian Jenkins, Carlos S. Frenk, Naoki Yoshida, Liang Gao, Julio Navarro, Robert Thacker, Darren Croton, John Helly, John A. Peacock, Shaun Cole, Peter Thomas, Hugh Couchman, August Evrard, Jörg Colberg, and Frazer Pearce. Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature*, 435(7042):629–636, June 2005.

[3] Cora Dvorkin, Martina Gerbino, David Alonso, Nicholas Battaglia, Simeon Bird, Ana Diaz Rivero, Andreu Font-Ribera, George Fuller, Massimiliano Lattanzi, Marilena Loverde, Julian B. Muñoz, Blake Sherwin, Anže Slosar, and Francisco Villaescusa-Navarro. Neutrino mass from cosmology: Probing physics beyond the standard model, 2019.

[4] Ofer Lahav and Andrew R Liddle. The cosmological parameters, 2022.

[5] Wen-Long Zhao, Wu Wang, and Qiao Wang. Optimization of cosmological n-body simulation with fmm-pm on simt accelerators. *The Journal of Supercomputing*, 78, 04 2022.

[6] Nathanaël Perraudin, Ankit Srivastava, Aurelien Lucchi, Tomasz Kacprzak, Thomas Hofmann, and Alexandre Réfrégier. Cosmological n-body simulations: a challenge for scalable generative models, 2019.

[7] Yin Li, Yueying Ni, Rupert A. C. Croft, Tiziana Di Matteo, Simeon Bird, and Yu Feng. Ai-assisted superresolution cosmological simulations. *Proceedings of the National Academy of Sciences*, 118(19):e2022038118, 2021.

[8] Francisco Villaescusa-Navarro, Shy Genel, Daniel Angles-Alcazar, David N. Spergel, Yin Li, Benjamin Wandelt, Leander Thiele, Andrina Nicola, Jose Manuel Zorrilla Matilla, Helen Shao, Sultan Hassan, Desika Narayanan, Romeel Dave, and Mark Vogelsberger. Robust marginalization of baryonic effects for cosmological inference at the field level, 2021.

[9] Kazuo Yonekura. Physics-guided generative adversarial network to learn physical models, 2023.

[10] Francisco Villaescusa-Navarro, Daniel Anglés-Alcázar, Shy Genel, David N. Spergel, Rachel S. Somerville, Romeel Dave, Annalisa Pillepich, Lars Hernquist, Dylan Nelson, Paul Torrey, Desika Narayanan, Yin Li, Oliver Philcox, Valentina La Torre, Ana Maria Delgado, Shirley Ho, Sultan Hassan, Blakesley Burkhart, Digvijay Wadekar, Nicholas Battaglia, Gabriella Contardo, and Greg L. Bryan. The camels project: Cosmology and astrophysics with machine-learning simulations. *The Astrophysical Journal*, 915(1):71, July 2021.

[11] Dylan Nelson, Volker Springel, Annalisa Pillepich, Vicente Rodriguez-Gomez, Paul Torrey, Shy Genel, Mark Vogelsberger, Ruediger Pakmor, Federico Marinacci, Rainer Weinberger, Luke Kelley, Mark Lovell, Benedikt Diemer, and Lars Hernquist. The illustristng simulations: Public data release, 2021.

[12] Romeel Davé, Daniel Anglés-Alcázar, Desika Narayanan, Qi Li, Mika H Rafieferantsoa, and Sarah Appleby. simba: Cosmological simulations with black hole growth and feedback. *Monthly Notices of the Royal Astronomical Society*, 486(2):2827–2849, April 2019.

[13] W. Dehnen and J. I. Read. N-body simulations of gravitational dynamics. *The European Physical Journal Plus*, 126(5), May 2011.

[14] Ana Maria Delgado, Daniel Angles-Alcazar, Leander Thiele, Shivam Pandey, Kai Lehman, Rachel S. Somerville, Michelle Ntampaka, Shy Genel, Francisco Villaescusa-Navarro, and Lars Hernquist. Predicting the impact of feedback on matter clustering with machine learning in camels, 2023.

[15] Alain Blanchard, Jean-Yves Héloret, Stéphane Ilić, Brahim Lamine, and Isaac Tutusaus. λcdm is alive and well, 2023.

[16] Viatcheslav F Mukhanov. *Physical foundations of cosmology*. Cambridge university press, 2005.

[17] Scott Dodelson and Fabian Schmidt. *Modern cosmology*. Academic press, 2020.

[18] Dale A Ostlie and Bradley W Carroll. *An introduction to modern astrophysics*. Addison-Wesley Reading, MA, USA, 2007.

[19] D Alloin. Physics of active galactic nuclei at all scales. *The Messenger*, 115:46–46, 2004.

[20] Drummond Fielding, Eliot Quataert, Davide Martizzi, and Claude-André Faucher-Giguère. How supernovae launch galactic winds? *Monthly Notices of the Royal Astronomical Society: Letters*, 470(1):L39–L43, 2017.

[21] Sylvain Veilleux, Gerald Cecil, and Joss Bland-Hawthorn. Galactic winds. *Annu. Rev. Astron. Astrophys.*, 43:769–826, 2005.

[22] Raffaella Morganti. The many routes to agn feedback. *Frontiers in Astronomy and Space Sciences*, 4:42, 2017.

[23] Saksham Garg. A study on the structure of neural networks and the mathematics behind backpropagation. 06 2020.

[24] Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. A survey and taxonomy of loss functions in machine learning. *arXiv preprint arXiv:2301.05579*, 2023.

[25] Kevin Clark. Computing neural network gradients. Accessed: 2023-12-20.

[26] Rohan Kashyap. A survey of deep learning optimizers – first and second order methods, 2023.

[27] Renan Alves de Oliveira, Yin Li, Francisco Villaescusa-Navarro, Shirley Ho, and David N. Spergel. Fast and accurate non-linear predictions of universes with deep learning, 2020.

[28] Cora Dvorkin, Siddharth Mishra-Sharma, Brian Nord, V. Ashley Villar, Camille Avestruz, Keith Bechtol, Aleksandra Ćiprijanović, Andrew J. Connolly, Lehman H. Garrison, Gautham Narayan, and Francisco Villaescusa-Navarro. Machine learning and cosmology, 2022.

[29] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[30] Valerio Biscione and Jeffrey Bowers. Learning translation invariance in cnns, 2020.

[31] What is overfitting? Accessed: 2024-04-20.

[32] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. Survey of dropout methods for deep neural networks, 2019.

[33] Udacity. L3m2 convolutions v2, Feb 2019.

[34] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[35] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[37] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[38] Martina Gerbino, Massimiliano Lattanzi, Marina Migliaccio, Luca Pagano, Laura Salvati, Loris Colombo, Alessandro Gruppuso, Paolo Natoli, and Gianluca Polenta. Likelihood methods for cmb experiments. *Frontiers in Physics*, 8:15, 2020.

[39] Nabila Aghanim, Yashar Akrami, Mark Ashdown, Jonathan Aumont, Carlo Baccigalupi, Mario Ballardini, Anthony J Banday, RB Barreiro, N Bartolo, S Basak, et al. Planck 2018 results-vi. cosmological parameters. *Astronomy & Astrophysics*, 641:A6, 2020.

[40] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022.

[41] Francisco Villaescusa-Navarro, Shy Genel, Daniel Angles-Alcazar, Leander Thiele, Romeel Dave, Desika Narayanan, Andrina Nicola, Yin Li, Pablo Villanueva-Domingo, Benjamin Wandelt, et al. The camels multifield data set: Learning the universe's fundamental parameters with artificial intelligence. *The Astrophysical Journal Supplement Series*, 259(2):61, 2022.

[42] Jonathon Shlens. Notes on kullback-leibler divergence and likelihood theory. *Systems Neurobiology Laboratory*, 92037:1–4, 2007.

[43] Niall Jeffrey and Benjamin D Wandelt. Solving high-dimensional parameter inference: marginal posterior densities & moment networks. *arXiv preprint arXiv:2011.05991*, 2020.

[44] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[45] Matias Valdenegro-Toro and Daniel Saromo. A deeper look into aleatoric and epistemic uncertainty disentanglement, 2022.

[46] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[47] Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. Model complexity of deep learning: A survey. *Knowledge and Information Systems*, 63:2585–2619, 2021.

[48] Paul Halpern and Nick Tomasello. Size of the observable universe. *Advances in Astrophysics*, 1, 11 2016.

[49] Axel Andersson, Nadezhda Koriakina, Nataša Sladoje, and Joakim Lindblad. End-to-end multiple instance learning with gradient accumulation, 2022.

[50] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[51] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans, 2021.

[52] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[53] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks, 2021.

[54] Cencheng Shen, Sambit Panda, and Joshua T. Vogelstein. The chi-square test of distance correlation. *Journal of Computational and Graphical Statistics*, 31(1):254–262, July 2021.

[55] Francisco Villaescusa-Navarro, ChangHoon Hahn, Elena Massara, Arka Banerjee, Ana Maria Delgado, Doogesh Kodi Ramanah, Tom Charnock, Elena Giusarma, Yin Li, Erwan Allys,

Antoine Brochard, Cora Uhlemann, Chi-Ting Chiang, Siyu He, Alice Pisani, Andrej Obuljen, Yu Feng, Emanuele Castorina, Gabriella Contardo, Christina D. Kreisch, Andrina Nicola, Justin Alsing, Roman Scoccimarro, Licia Verde, Matteo Viel, Shirley Ho, Stephane Mallat, Benjamin Wandelt, and David N. Spergel. The quijote simulations. *The Astrophysical Journal Supplement Series*, 250(1):2, August 2020.

[56] Matthias Bartelmann. The dark universe. *Review of Modern Physics*, 82:331–382, 02 2010.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[58] Sheng He, Rina Bao, P. Ellen Grant, and Yangming Ou. U-netmer: U-net meets transformer for medical image segmentation, 2023.

[59] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation, 2021.

[60] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation, 2022.

[61] Helen Shao, Francisco Villaescusa-Navarro, Shy Genel, David N Spergel, Daniel Anglés-Alcázar, Lars Hernquist, Romeel Davé, Desika Narayanan, Gabriella Contardo, and Mark Vogelsberger. Finding universal relations in subhalo properties with artificial intelligence. *The Astrophysical Journal*, 927(1):85, 2022.

[62] S Schuldt, SH Suyu, R Cañameras, S Taubenberger, T Meinhardt, L Leal-Taixé, and BC Hsieh. Photometric redshift estimation with a convolutional neural network: Netz. *Astronomy & Astrophysics*, 651:A55, 2021.

[63] Benjamin Horowitz, Max Dornfest, Zarija Lukić, and Peter Harrington. Hyphy: Deep generative conditional posterior mapping of hydrodynamical physics. *The Astrophysical Journal*, 941(1):42, 2022.

[64] Jonás Chaves-Montero, Raul E Angulo, Joop Schaye, Matthieu Schaller, Robert A Crain, Michelle Furlong, and Tom Theuns. Subhalo abundance matching and assembly bias in the eagle simulation. *Monthly Notices of the Royal Astronomical Society*, 460(3):3100–3118, 2016.

[65] Jiatao Gu, Qingzhe Gao, Shuangfei Zhai, Baoquan Chen, Lingjie Liu, and Josh Susskind. Learning controllable 3d diffusion models from single-view images. *arXiv preprint arXiv:2304.06700*, 2023.

[66] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.

[67] Vitali Petsiuk, Rajiv Jain, Varun Manjunatha, Vlad I Morariu, Ashutosh Mehra, Vicente Ordonez, and Kate Saenko. Black-box explanation of object detectors via saliency maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11443–11452, 2021.

[68] Dmitrii Kapitan, Alena Korol, Egor Vasiliev, Pavel Ovchinnikov, Alexey Rybin, Eliza Lobanova, Konstantin Soldatov, Yuriy Shevchenko, and Vitalii Kapitan. Chapter one - application of machine learning in solid state physics. In Rair Macedo and Robert L. Stamps, editors, *Solid State Physics*, volume 74 of *Solid State Physics*, pages 1–65. Academic Press, 2023.

[69] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

[70] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.