Application of Machine Learning Coursework Report

James Hughes

March 25, 2024

Contents

1 Gaussian Noise Diffusion

1.1 Diffusion Model & Training Algorithm

The code found in the repository for this project was based on the Jupyter notebook found in the `assignment` directory. This notebook contains a standard pipeline which trains a Denoising Diffusion Probabilistic Model, a kind of generative model first appearing in the literature just four years ago [1]. The model is trained using the training partition of the MNIST dataset [2], which contains monochrome images of handwritten digits.

Fundamental to this model is an underlying image-to-image convolutional neural network, which tries to predict a ground truth MNIST image from the same image, with added Gaussian noise, as an input. This netowrk has a deep, symmetrical architecture based on convolutional layers which initially increase the number of channels, which extracts features from the input, and then later combines these channels to eventually produce a monochrome (single-channel) generated image. These convolutional layers are combined with a layer normalisation, which normalises the pre-activations of the hidden units separately for each input of the batch [3]. This step builds on the batch normalisation approach, improving the stability and speed of training the neural network [3], but has the advantage of being more suitable when batches are small or have a large within-batch variation [4]. The results of the layer normalisation are then passed through a GeLU activation function [5], which is a close variant of the standard ReLU activation function, but which mitigates the issue of vanshing gradients by having non-zero gradients for small negative inputs [6].

The underlying CNN model also takes a scalar time step `t` in $[0, 1]$ as part of its raw input; this becomes a vector when inputs are batched, with one scalar for each batch member. Each of these scalars is in turn encoded as a vector, whose entries are a periodic function (of progressively increasing frequency) of the scalar value. These vectors are then passed through a series of learned linear layers and non-linear activations, changing their length to equal the number of hidden channels in the first layer of the CNN. These time embeddings are then combiend with the first hidden activations via addition.

This underlying model is then wrapped in a `DDPM` class, implementing a `forward` and `sample` method. The `forward` method takes a batch of MNIST images input. It generates a random time step for each image, which controls the strength of degradation applied to each, taking the form of addition with

Gaussian pixel noise. It feeds the degraded image and the time step to the CNN, and returns the mean squared error between the CNN output and the original noise map. During training this incentivises the CNN to predict the Gaussian noise applied to a degraded image. The `sample` method implements Algorithm 18.2 in [4]. It starts with a random Gaussian tensor the same size as the MNIST images, which it then passes through the trained CNN, and subsequently degrades with Gaussian noise. These two transformations are repeated for smaller and smaller timesteps until the degradation is minimal.

The initial codebase was created by refactoring this notebook into a library, `diffusiontools`. The training procedure was then altered to make training runs more repeatable. Alongside the samples which were produced at each epoch, the trained model was itself saved as a `.pt` file. This enabled further investigation into intermediate models after the training, and indeed meant that later on, the periodic sampling could be removed altogether to speed up training (as sampling could be done in post). The hyperparameter settings were also controlled via a `config.ini` to enable different configurations to be easily tested, and tracked.

1.2 Analysis of Training

The first model trained had the same hyperparameters as found in the notebook. It was trained for 500 epochs by uploading the repository to the CSD3 systems. The MNIST training dataset was split further into a training set (80%) and a validation set (20%), to monitor the model's ability to generalise during the training process.

In Figure 1, it can be seen that the validation loss appears to converge roughly around the 100 epoch stage, with no significant improvements from further training. This matches the findings of generated samples in Figure 2. We see that around the 40 epoch mark, the model begins to produce symbols but these are not recognisable digits. The images generated at epoch 60 are the first that resemble numerical digits – images in Figure 2b appear to show a 7, 5, 2, 1, and 9. Figure 2c appears to show higher fidelity images in the first three samples, although the other two cannot be discerned as digits. This model was selected for further analysis as the final trained model.

A second similar model was then trained with a slightly different Gaussian noising schedule. In this second model, the number of discrete timesteps in the diffusion process was reduced ten-fold to 100. The noise parameter limits were accordingly increased by a factor of ten, ensuring that the amount

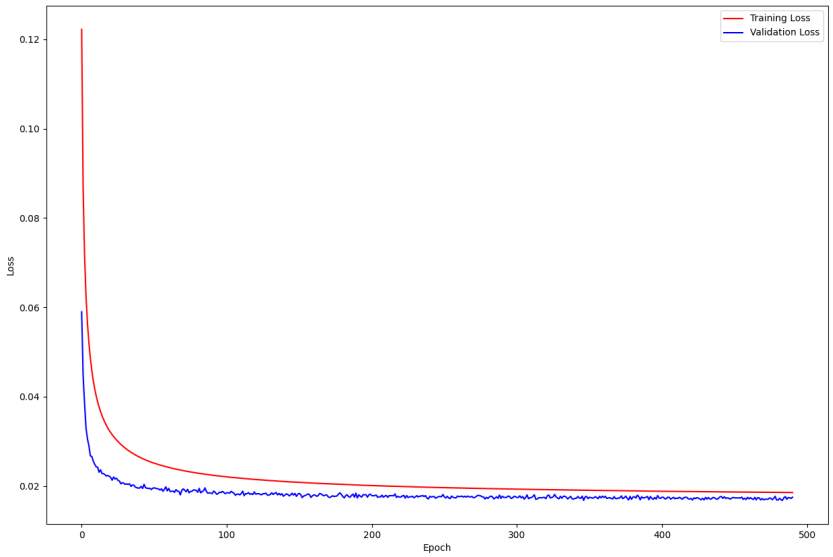
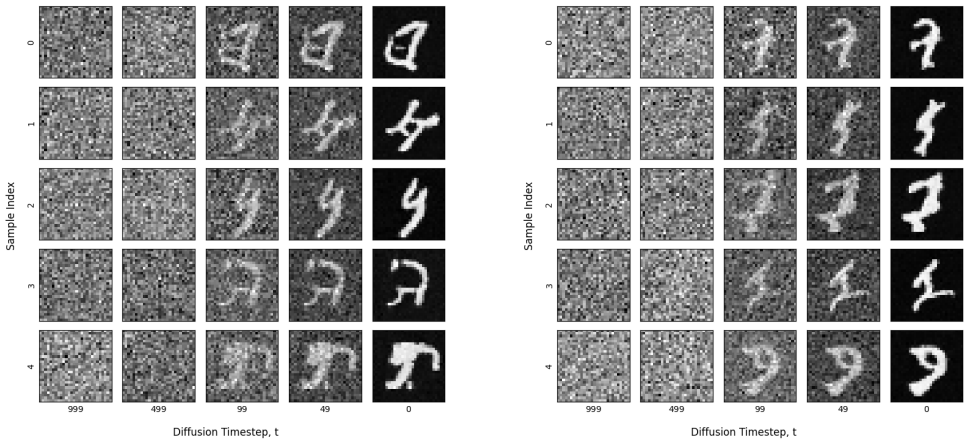
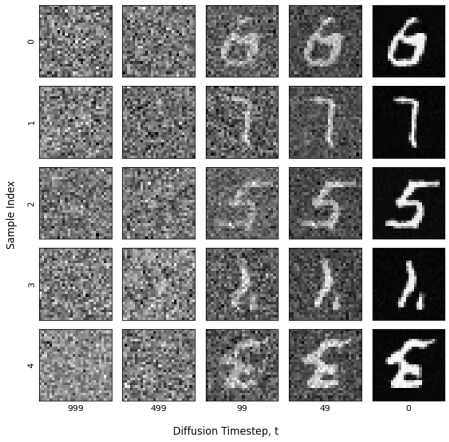


Figure 1: Learning Curve for Model 1 Trained over 500 Epochs.

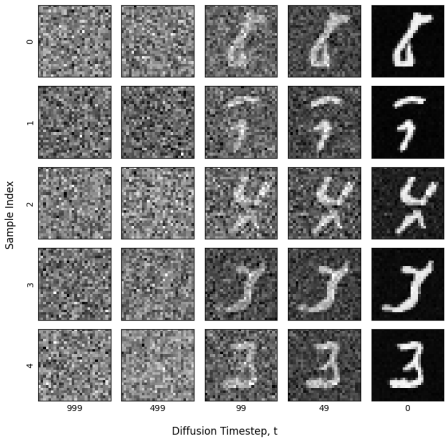


(a) Epoch 40

(b) Epoch 60



(c) Epoch 120



(d) Epoch 300

Figure 2: Model 1 diffusion process samples at various stages of training.

of signal-to-noise ratio of an image at the final time step remained approximately zero.

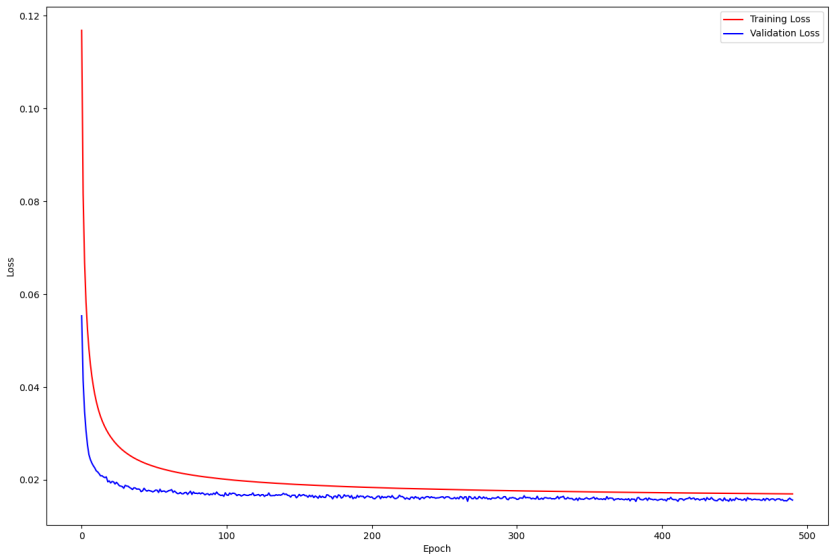
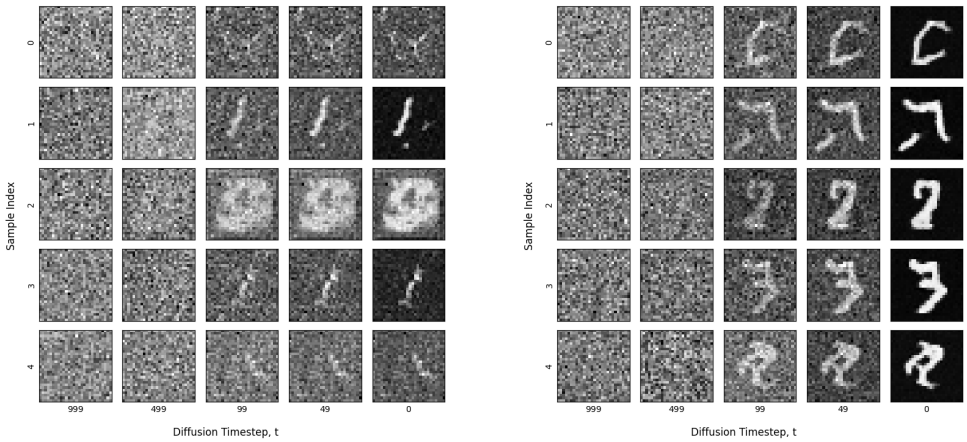


Figure 3: Learning Curve for Model 2 Trained over 500 Epochs.

A similar learning curve can be seen for the training of the second model in Figure 3. However, inspecting some of the images produced by intermediate models shows that the quality of the generated images deteriorates much earlier than in Model 1. Even images produced at epoch 100 in 4 appear to be low quality, with 'fragmented' symbols rather than one continuous digit. The images produced at epoch 80 appear to be the highest quality and this model was used in the final analysis.

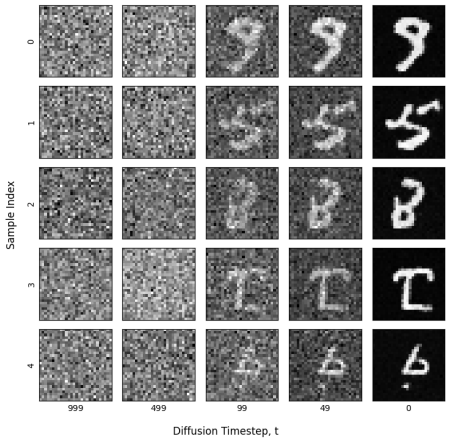
1.3 Analysis of Final Models

The evaluation of generative models is a challenging and ongoing subject of research [7],[8]. Unlike other tasks within machine learning such as classification, regression, and clustering, an ideal generative model generalises from its training data to produce content that is simultaneously novel, diverse and realistic. When diffusion models are used in a conditional generation setting,

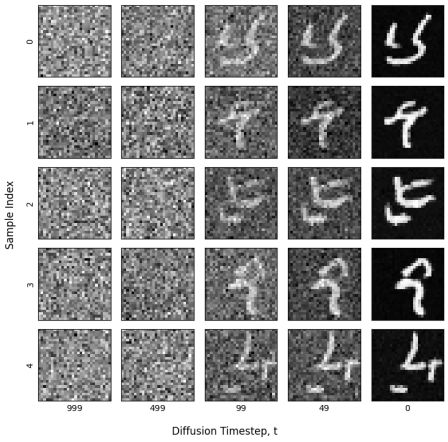


(a) Epoch 20

(b) Epoch 40



(c) Epoch 80



(d) Epoch 100

Figure 4: Model 2 diffusion process samples at various stages of training.

ground truth images can be degraded and then reconstructed by the diffusion model, allowing evaluation through image comparison metrics such as SSIM and MSE, as done in [9] – this is explored later. More broadly, a standard evaluation metric is the Fréchet Inception Distance [10]. In this project, the models are quantitatively assessed with the framework of comparing the statistical distributions of generated and ground truth images, just like with the FID score, but in a slightly simpler and more interpretable way. 200 samples were taken from each of the four final models and combined with the 10,000 MNIST test images into a single array. These were then transformed into two dimensions using t-SNE dimensionality reduction. The effect of this on the MNIST dataset alone can be seen in Figure 5, which shows that this technique is strikingly effective at encoding the true image class in a meaningful way. A Gaussian mixture model with 10 components (corresponding to the 10 digits) was then fitted to each sample, enabling an estimate of the KL divergence between the generated and ground truth image distributions.

Figure 6 shows that both models have good coverage across the whole distribution of ground truth distribution of data, suggesting they are capable of producing a variety of different digits. However, we can also observe that neither distribution of points appears to have a similar clustering pattern to the ground truth data, which suggests that both models also have a tendency to produce low quality samples that do correspond to a discernible digit. The KL divergences, computed as $D_K L(P||Q)$ where Q is the ground truth distribution, were similar with the model 1 sample at 0.81844, and the model 2 sample at 0.77298. This suggests that the distributions of generated images from both models imitate the ground truth with similar success. When their images are treated as random vectors, the total variance (trace of the covariance matrix) for the first sample was 56.7, and slightly higher for the second at 69.9. While this may suggest a greater diversity of samples from the second model, it may also reflect the greater noise present at each step, which in turn is clearly visible as the samples produced appear grainier.

2 Custom Diffusion Model

2.1 Custom Degradation Strategy

Earlier prototypes that were infeasible.

Demo plot for digits 1 to 4. Batch/timestep limitation.

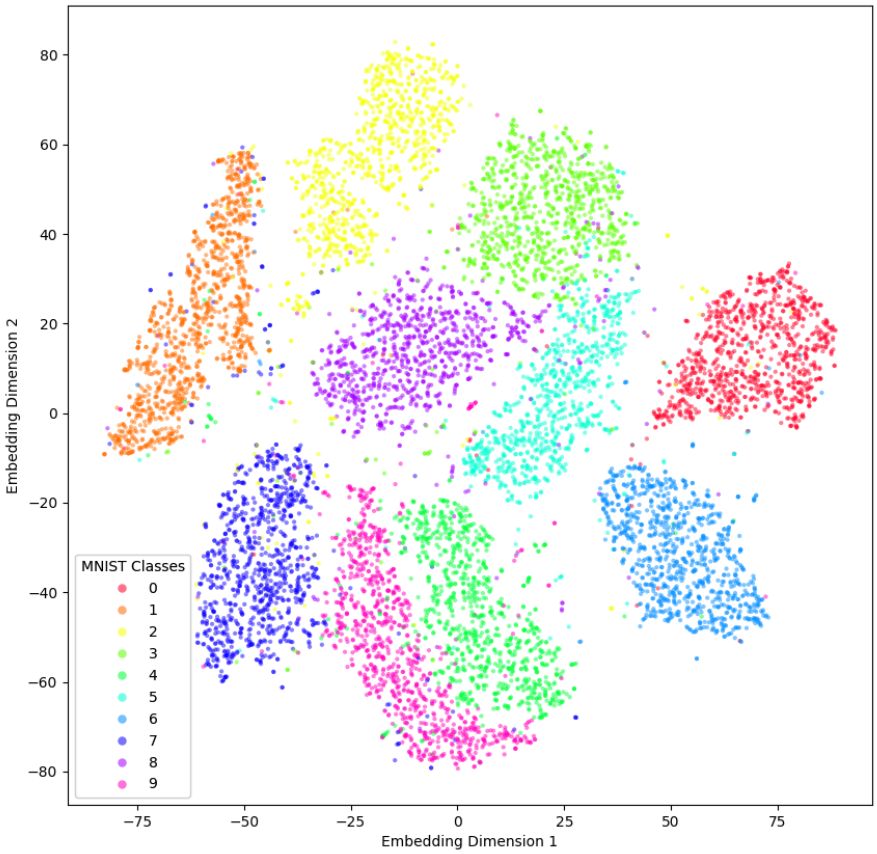


Figure 5: t-SNE dimensionality reduction applied to MNIST test dataset, with ground truth labels indicated.

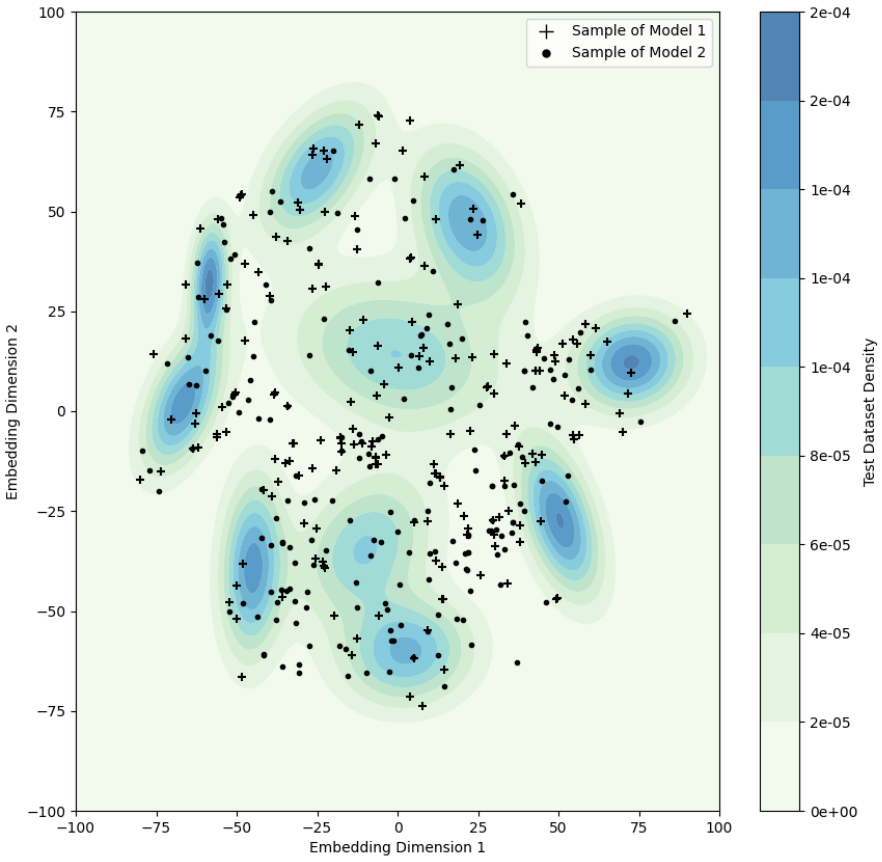


Figure 6: t-SNE dimensionality reduction applied to Gaussian model samples, with contour plot of GMM density estimate for MNIST test data.

Figure 7: Samples from model 1.

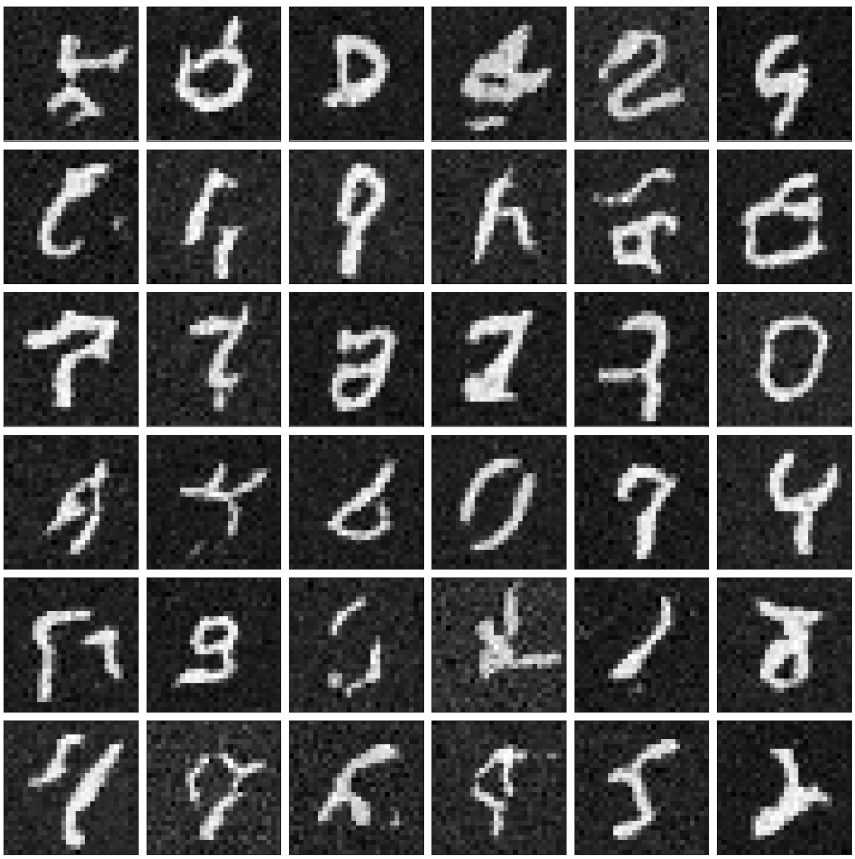


Figure 8: Samples from model 2.

2.2 Analysis of Custom Degradation Training

Early epochs diversity issues.

2.3 Comparison of Gaussian and Custom Degradation

TSNE plot and Bansal's

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[4] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023.

[5] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[6] Anh Nguyen, Khoa Pham, Dat Ngo, Thanh Ngo, and Lam Pham. An analysis of state-of-the-art activation functions for supervised deep neural network, 2021.

[7] Eyal Betzalel, Coby Penso, Aviv Navon, and Ethan Fetaya. A study on the evaluation of generative models, 2022.

[8] Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. The power of generative ai: A review of requirements, models, input-output formats, evaluation metrics, and challenges. *Future Internet*, 15(8), 2023.

[9] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S. Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise, 2022.

[10] 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. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

A Statement on the use of auto-generation tools

Auto-generation tools, such as GitHub or Microsoft Copilot, were not used at any stage during the development of the code within the repository of this project. Similarly, none of the content of this report was produced – or proofread – by modern Large Language Models such as ChatGPT at any point.

B High-Performance Computing Resources

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).