AIML 425 Assignment 4

James Thompson
Victoria University of Wellington
300680096

I. Introduction

This document reports the implementation and results of Assignment 4. I implement Stochastic Differential Equations and Ordinary Differential Equation models. THe SDE model is a score based generative model to learn a mapping from Gaussian noise to dog images. The ODE model is a normalizing flow model that learns a mapping from Gaussian to dogs and cats to dogs.

II. Theory

I will be looking at two models, a Stochastic Differential Equation (SDE) model and an Ordinary Differential Equation (ODE) model. Both models are generative models that learn a mapping from one distribution to another. The key difference being that the SDE model is score based and stochastic in generation while the ODE model is a normalizing flow model that is deterministic in generation.

A. Flows

B. Stochastic Differential Equations

C. Ordinary Differential Equations

D. Performance comparison

The end objective of both models is to learn how to take data from one distribution to another iteratively. The loss shows us the difference between the flows. However to understand the performance we really want to know how well it can generate samples. A simple and effective measure is Mean Maximum Discrepancy (MMD) [1] which allows us to compare the generated samples to the real target distribution. The MMD is a measure of the distance between two distributions based on samples from each distribution. It is defined as

$$\text{MMD}^2(P, Q) = \mathbb{E}[k(x, x')] + \mathbb{E}[k(y, y')] - 2\mathbb{E}[k(x, y)] \quad (1)$$

$$\text{where } k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (2)$$

$$\text{and } x, x' \sim P, \quad y, y' \sim Q \quad (3)$$

The only parameter in this is the kernel bandwidth σ. The higher the sigma the smoother the kernel so focuses on global differences, where lower sigma focuses on local differences.

Other metrics to understand the performance can be learning efficiency and sample efficiency. This means that we can compare how many samples it takes to learn as well as how much compute time it takes to learn. This can be measured in terms of number of epochs or wall clock time.

III. Experiments

IV. Conclusion

STATEMENT

The code and report of the Assignment was solely completed by myself (Thompson James). The complete source code can be found here https://gitea.james-server.duckdns.org/james/AIML425_assignment_4, with a link to a colab notebook found here: https://colab.research.google.com/github/1jamesthompson1/AIML425_assignment_4/blob/main/main.ipynb. A complete run through of the notebook takes about 10 minutes on a GPU enabled machine.

I have kept the appendix as concise as possible, however the code is setup to produce many more figures and tables that can be used to understand the models. Most of these figures have been generated and stored in the 'figures' folder.

I completed my work using the following tools:

- **Jupyter Notebook [2] and JupyterText [3]:** For interactive development and hosting.
- **LaTeX**: For writing the report.
- **VSCode [4]:** As IDE, with Copilot to help with plotting and debugging.
- **JAX [5] and Flax [6]:** For implementing the neural network and training logic.
- **Matplotlib [7] and Pandas [8]:** For data visualization and management.

REFERENCES

[1] A. Gretton, K. Borgwardt, M. J. Rasch, B. Scholkopf, and A. J. Smola, "A Kernel Method for the Two-Sample Problem," May 2008.

[2] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter Notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87–90.

[3] M. Wouts, "Mwouts/jupytext," Aug. 2025.

[4] "Microsoft/vscode," Microsoft, Jul. 2025.

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: Composable transformations of Python+NumPy programs," 2018.

[6] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, "Flax: A neural network library and ecosystem for JAX," 2024.

[7] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[8] The pandas development team, "Pandas-dev/pandas: Pandas."

APPENDIX

A. Setup

B. Results

The models were trained using the hyperparameters in Table I. They were trained on a RTX A5000 GPU with a Xeon(R) Gold 6128 CPU.

TABLE I
HYPERPARAMETERS USED FOR EACH MODEL

Hyperparameter	SDE	ODE (Gauss to Dog)	ODE (Cat to Dog)
Learning Rate		0.0001	
Minibatch Size	512	256	
Hidden Dims	512	256	
Hidden Layers	4	3	
Num Epochs	500	200	
Optimizer		Adam	
Loss Function		MSE	

TABLE II
TRAINING RESULTS

Result	SDE	ODE (Gauss to Dog)	ODE (Cat to Dog)
Runtime (seconds)	?	?	?
MMD	?	?	?
Best validation loss	?	?	?