SCORE-BASED METHODS: STOCHASTIC AND ORDINARY DIFFERENTIAL EQUATIONS FOR GENERATIVE DIFFUSION

Maria DaRocha

VUW, AIML425: Neural Networks and Deep Learning

1. INTRODUCTION

The following is my own work and supported by code available on GitHub and Google Collab.

In the context of generative modeling, it is very useful to randomly sample from a simple distribution mapped (by a neural network) to a desired distribution. The following paper investigates the use of diffusion models for mapping simplified (2D) representations of cats, dogs, and Gaussians. For visualized target examples, see ***Figure 1***, *Appendix.*

2. THEORY

The first diffusion model used in the context of generative modeling, the Denoising Diffusion Probabilistic Model (DDPM), was derived from the Markov chain and nonequilibrium thermodynamics theory (particle diffusion). [1] The model is expressed as a forward Markov process that gradually replaces a clean signal with Gaussian noise,[1] followed by an approximation of the process's probabilistic inverse to reconstruct a random variable input using parameters θ *(**Equation 1**, Appendix).*

This motivated the Stochastic Differential Equation (SDE) formulation of the Markov chain (general notation in ***Equation 2***, *below with further details in Appendix*).

$$\textbf{(1)}\quad dx = f(x,t)dt + g(t)d\omega$$

$$\textbf{(2)}\quad dx = \left(f(x,t) - g^2(t)\nabla_x log\big(p(x,t)\big)\right)dt + g(t)d\omega$$

where… $f(x,t)$ *is the drift coefficient,*
$g(t)$ *is the diffusion coefficient,*
$\nabla_x log\big(p(x,t)\big)$ *is the* ***score function****,*
$d\omega$ *is the Wiener process*

Equation 2: *Forward and Reverse SDE, Appendix.*

The score function estimates the clean signal based on the difference between the noisy signal x at time t and the clean signal Y. The method used to learn the score function is referred to as **score matching**.

In score matching, the forward process of summing random Gaussian variables is equated to convolving their densities; this is helpful for formulating an analytic description of the score (***Equation 3***, *below).*

$$\textbf{(3)}\quad \nabla_x log\big(p(x,t)\big) = \frac{1}{\sigma^2(t)}\left(E[Y|x] - x\right)$$

Equation 3: *Analytic Description of the Score Function.*

[1] A Markov process is a system in which the next state depends on all previous states.

The next step is to define an MMSE estimator D for Y, in which Y is conditional on current data x *(**Equation 4**, below).*

$$\textbf{(4)}\quad D^*(x) = argmin_D \int dy\left|\left|y - D(x)\right|\right|^2 p(y|x) = \int dy\; y\; p_x(y|x) = E[Y|x]$$

Equation 4: *Optimal MMSE Estimate for* $E[Y|x]$.

Finally, it is possible to approximate $E[Y|x]$ with a D_θ, an all-of-t neural network denoiser $D_\theta(x;t)$, by sampling a clean image, selecting a random time t to obtain a controlled noisy signal (with mean zero, covariance as per ***Equation 1****, Appendix*)**,** and gradient step training to reduce the mean squared error between $D_\theta(x;t)$ and clean signal y (***Equation 5****, Appendix).*

$$\textbf{(5)}\quad \nabla_x log\big(p(x,t)\big) = \frac{1}{\sigma^2(t)}\left(D_\theta(x;t) - x\right)$$

Equation 5: *Score Function Final Training Objective, Appendix.*

Once the score function $\nabla_x log\big(p(x,t)\big)$ is learned via a neural network as a function of x and t, it can be used within the Euler-Maruyama method **(2)** to solve the backwards SDE (or in the ODE, discussed further on). [2] The cumulative process describes the evolution of an X (e.g., an image) and effectively maps one distribution to another.

Notably, the reversal is not exact due to the Wiener term (stochasticity) in the forward process, but after many iterations the result still follows the original (intended) distribution. Further, the act of fixing (or 'freezing') the score function and allowing the SDE to run results in a Langevin equation with self-healing behaviour.

Thus, the forward SDE goes towards a flat distribution, the reverse SDE goes towards a desired distribution, and accumulated precision errors are mitigated by using the Langevin equation locally for the probability distribution $p(x,t)$ (i.e., we converge to distribution $p(x,t)$ independently of past events, including numerical errors).

In contrast to the Euler-Maruyama method, the Fokker-Planck equation for $p(x,t)$ corresponds to the Ordinary Differential Equation (ODE) for x, which describes the time evolution of a probability distribution. By extension, the ODE can be understood as describing a deterministic particle path, rather than a stochastic one (SDE). [3] This formulation is also known as the Euler method (***Equation 6***, *below with further details in Appendix).*

$$(6)\quad dx = \left(f(x,t) - \frac{1}{2}g^2(t)\nabla_x log\big(p(x,t)\big)\right)dt$$

Equation 6: *ODE for a Probability Evolution, Appendix.*

The ODE is simpler due to the absence of the Wiener term and ease of function reversal (added minus, ***Equation 6***, *Appendix*), but the self-healing property is lost, making the ODE more susceptible to error accrual and slightly worse (best) performance.

Crucially, in this setup, the RHS of the equation represents an interpolant-defined velocity field (***Figure 2***, *Appendix*). When using interpolants for ODEs, the training objective is $v_\theta(x,t)$ that matches $x_0 - x_1$ at (x,t) best on average *(**Equation 7**, below).*

$$(7)\quad min_\theta = E_{\tau\in U[0,1]} E_{x_1\sim p_1, x_0\sim p_0}\left[\left|\left|X_1 - X_0 - v_\theta(x_t,\tau)\right|\right|^2 \middle| X_t = x_t\right]$$

Equation 7: *Interpolants for Velocity Field Matching.*

That is, the neural network $v_\theta(x_t,\tau)$ target is to learn the average of the velocity field, then minimise the squared error between each instance of the velocity field over all instances of both distributions for a random location between zero and one (setup example only).

Lastly, implementing either the SDE or ODE requires discretization of their respective differential equations.

3. EXPERIMENT

The experiment was set up using Python in Jupyter Lab.

The first phase of the experiment focused on learning the score function for a Gaussian to Dog mapping via the score matching process previously outlined. The all-of-***t*** denoiser had an input dimensionality of three ($x \in R^2, t \in R$), a flexible number of hidden layers and dimensions, and an output dimensionality of two (x,t). The denoiser's loss (objective function) was a predicted MMSE residual $(E[Y|x] - x)$**.**

The next phase of the experiment took the learned Gaussian to Dog scores and used them within first-order Euler-Maruyama and Euler methods to compare the resulting maps. Within the SDE and ODE setup, for simplicity the chosen drift coefficient was ***zero*** and the diffusion coefficient was ***t*** (i.e., $f(x,t) = 0$, and $g(t) = t$, respectively). The SDE used a variance exploding configuration, in which the noise term (scaled by t) increased as time progressed. The models also presented velocity fields for further comparison. Model performance was assessed using Number of Function Evaluations (NFE), Mean Squared Error (MSE), and Wasserstein (Earth mover's) distance.

The final phase of the experiment consisted of a second mapping (third diffusion model) from cats to dogs, using a similar approach to score function derivation and ODE (interpolant) formulation as the previous phase.

3.2. Results

The denoiser for the Gaussian to Dog mapping attained low training and validation losses at 0.073 and 0.095, respectively. The loss function demonstrated some instability, but this did not seem to adversely affect performance. The score function's probability density also sat within an expected range (-2.912, 3.22) and the density plot of the score function displayed an anticipated "blurry dog" appearance. Full model results, including probability densities and partial (last index) predicted residual and training scores can be seen in ***Tabel 1***, *Appendix*.

The SDE solver of a first-order variance exploding Euler-Maruyama configuration performed better than its counterpart ODE. Both models' parameters and performance results are captured in ***Table 2*** and ***Table 3***, respectively. For complete visual outputs, such as the final mapping and velocity fields please refer to ***Table 2*** and ***Table 3***, *Appendix*.

SDE Model	**Euler-Maruyama Method, Variance Exploding**
Mapping	Gaussian to Dog
SDE Parameters	T = 1, N = 1000, dt = 0.001
Scaling Factor (s)	0.050
NFE	1000
MSE	0.03706
Wasserstein Dist.	0.36373

Table 2: *SDE Formulation of Diffusion Model, Gaussian to Dog, Appendix.*

ODE Model	**ODE (Euler Method), Interpolation**
Mapping	Gaussian to Dog
SDE Parameters	T = 1, N = 40, dt = 0.025
Scaling Factor (s)	1.000
NFE	40
MSE	0.10783
Wasserstein Dist.	0.62838

Table 3: *ODE Formulation of Diffusion Model, Gaussian to Dog Appendix.*

The third model used a similar ODE formulation but required a new score derivation for the task of mapping cats to dogs. The final score function losses were 0.069 (training) and 0.076 (validation). Model results are as follows, with further details and diagrams located in ***Table 4*** of the *Appendix*.

ODE Model	**ODE (Euler Method), Interpolation**
Mapping	Gaussian to Dog
SDE Parameters	T = 1, N = 40, dt = 0.025
Scaling Factor (s)	1.000
NFE	40
MSE	0.26016
Wasserstein Dist.	1.01490

Table 4: *ODE Formulation of Diffusion Model, Cat to Dog, Appendix.*

4. CONCLUSION

After fine-tuning model parameters, such as the time horizon and number of time steps, the experiment concluded with relatively positive results. The SDE model slightly outperformed the ODE models, which is consistent with real-world expectations, since SDEs account for randomness within the system.

Further, the ODE velocity fields demonstrated structured, directional flow and the arrows representing velocity followed consistent, organised paths; this follows the expected behaviour of a deterministic setting.

By comparison, the SDE velocity field demonstrated chaotic, radial patterns. This is reflective of how SDEs introduces a noise tap (Wiener process) that perturbs the velocity field in all directions, where randomness diffuses particles away from their initial trajectories.

In conclusion, all the models performed moderate to well at the chosen mappings and provide firm evidence of the utility of score-based methods in guiding complex transformations between distributions. The SDE's ability to account for randomness and uncertainty gives it a slight edge in capturing real-world variability, while the ODE's structured, deterministic nature ensures more predictable, directional mappings. Together, these approaches highlight the versatility and robustness of diffusion models for handling distributional shifts, such as mapping from cats (or Gaussians) to dogs.

5. APPENDIX

5.1. Equations

Forward Markov Process:

$$(1)\quad q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right), \textit{ where…}$$

$\beta_1, \dots, \beta_t$ is a noise schedule, hence $\beta_t I$ is the diagonal covariance matrix of noise with a gain β_t.

Reverse Markov Process (probabilistic inverse):

$$(2)\quad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma_\theta(x_t, t))$$

Early implementations synchronized the forward and backwards process, but this is not necessary.

Equation 1: Denoising Diffusion Probabilistic Model (DDPM), Forward and Reverse Markov Formulation *[4], [5]**.***

$$(3)\quad dx = f(x,t)dt + g(t)d\omega$$

$$(4)\quad dx = \left(f(x,t) - g^2(t)\nabla_x log\big(p(x,t)\big)\right)dt + g(t)d\omega$$

where… $f(x,t)$ *is the drift coefficient,*
$g(t)$ *is the diffusion coefficient,*
$\nabla_x log\big(p(x,t)\big)$ *is the score function,*
$d\omega$ *is the Wiener process*

The variance (energy) preserving case allows movement in random directions while keeping particle distance on average equally far from the origin.

Variance Preserving Case:

$$f(x,t) = -\tfrac{1}{2}\sqrt{\beta_t}x\,,\ \ g(t) = \sqrt{\beta_t}$$

Equation 2: Forward and Reverse Stochastic Differential Equation (SDE). *[5], [6]*

$$(5)\quad dx = \left(f(x,t) - \tfrac{1}{2}g^2(t)\nabla_x log\big(p(x,t)\big)\right)dt$$

As observed in ResNets and continuous flows, the reverse ODE is the same as the forward ODE but with a minus sign at the front. [7]

Equation 6: Ordinary Differential Equation (ODE) Expression for a Probability Evolution. *[3], [5]*

5.2. Figures & Tables

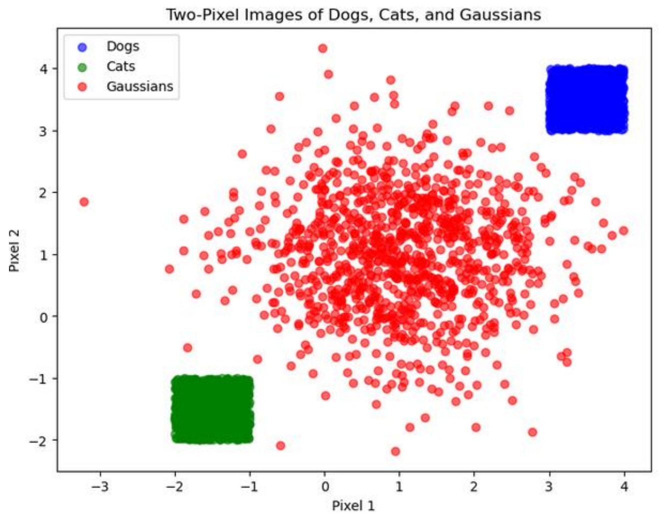


Figure 1: 2D Representation of Cats, Dogs, and Gaussians.

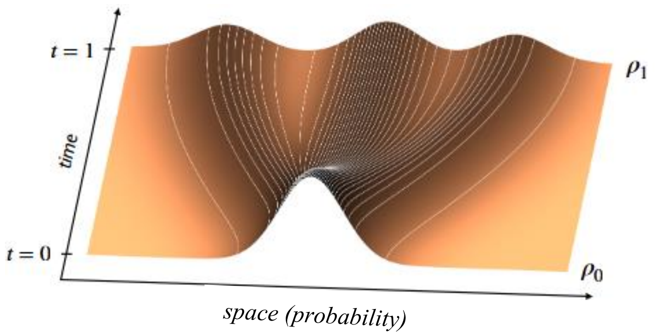
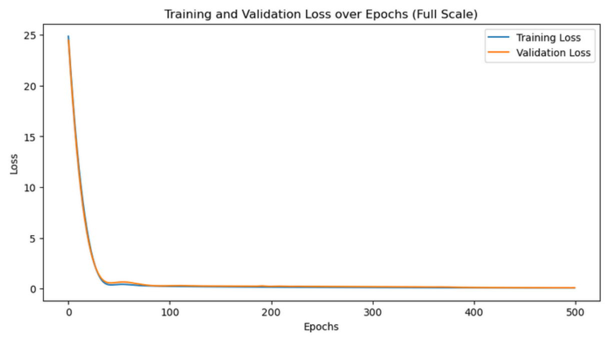


Figure 2: Velocity Field (Example) *[5]**.***

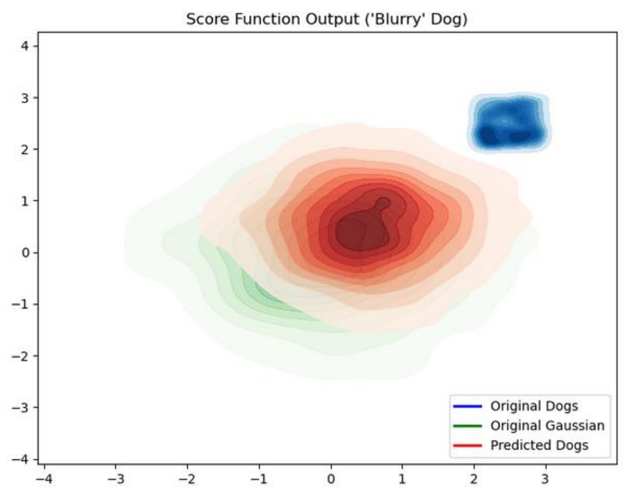
Model f_1: Gaussian to Dog (Score Matching)



Score Function Training & Validation Loss Over Epochs

Epoch	Train Loss	Val Loss	Predicted Train Residual (Last Entry)	Final Train Score (Last Entry)
0	24.8	24.5	[-2.738, -4.150]	[-4.280, -6.48]
50	0.425	0.646	[-1.181, 2.724]	[-1.846, 4.258]
100	0.232	0.293	[-0.735, 2.294]	[-1.149, 3.586]
150	0.187	0.255	[-0.317, 2.222]	[-0.495, 3.473]
200	0.150	0.226	[-0.070, 2.149]	[-0.110, 3.359]
250	0.132	0.214	[-0.042, 2.064]	[-0.066, 3.227]
300	0.120	0.196	[-0.032, 2.044]	[-0.051, 3.195]
350	0.108	0.170	[-0.046, 2.021]	[-0.071, 3.159]
400	0.089	0.120	[-0.147, 1.888]	[-0.230, 2.951]
450	0.077	0.100	[-0.154, 1.889]	[-0.240, 2.952]
500	**0.073**	**0.095**	**[-0.172, 1.872]**	**[-0.269, 2.926]**
Final Train Loss = 0.073 | Final Val Loss = 0.095				

Cont'd.



Score Function Output, All Probability Densities

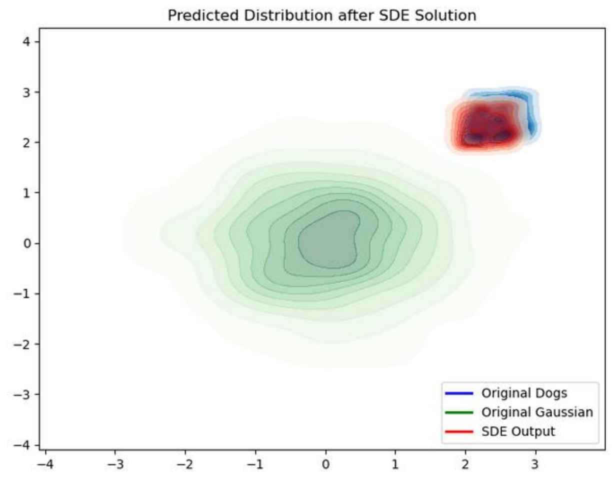
Dog Train Range: (2.00, 2.99)
Gaussian Train Range: (-1.13, 6.32)
Predicted Train Range: (-2.912, 3.22)

Table 1: Score Mapping Summary and Output.

Model f_2: SDE Formulation of Diffusion Model

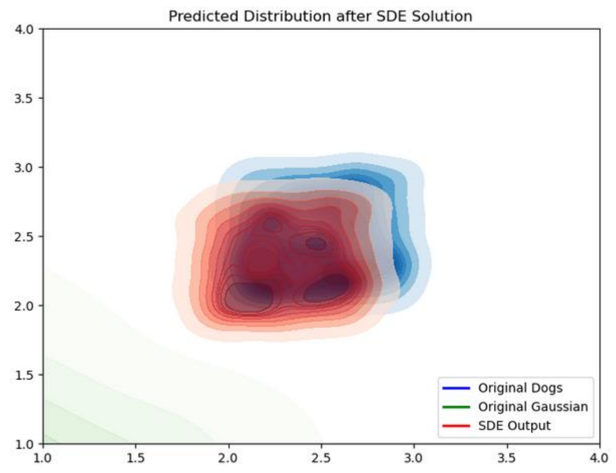
Model Summary

SDE Model	**Euler-Maruyama Method, Variance Exploding**
Mapping	Gaussian to Dog
SDE Parameters	T = 1, N = 1000, dt = 0.001
Scaling Factor (s)	0.050
NFE	1000
MSE	0.03706
Wasserstein Dist.	0.36373

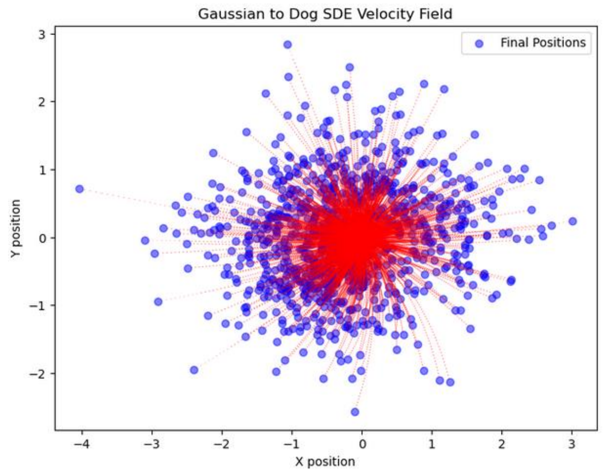


SDE Output, All Probability Densities

Cont'd.



SDE Output, All Probability Densities (Zoomed)



SDE Velocity Field

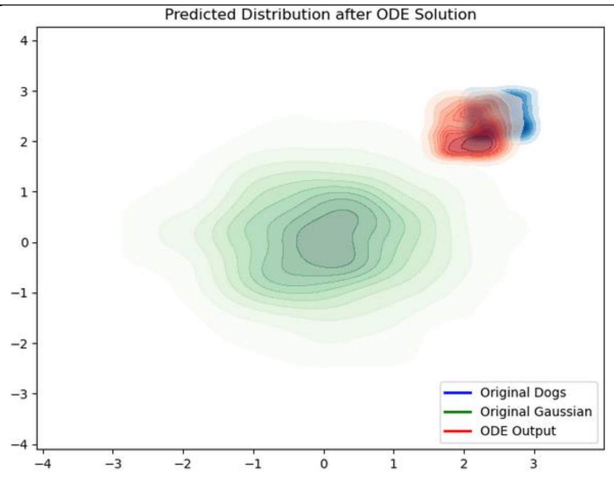
Comments: Good mapping; velocity field chaotic and radial, as expected of SDE (Wiener process; lack of meaningful latent space).

Table 2: SDE Gaussian to Dog Summary and Output.

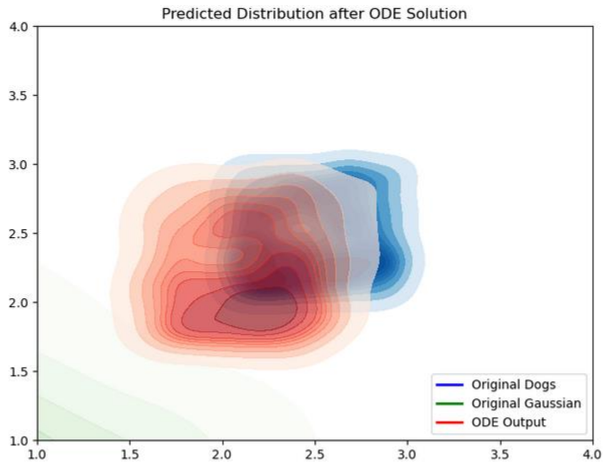
***Model f_3*: Variational Autoencoder (VAE)**

Model Summary

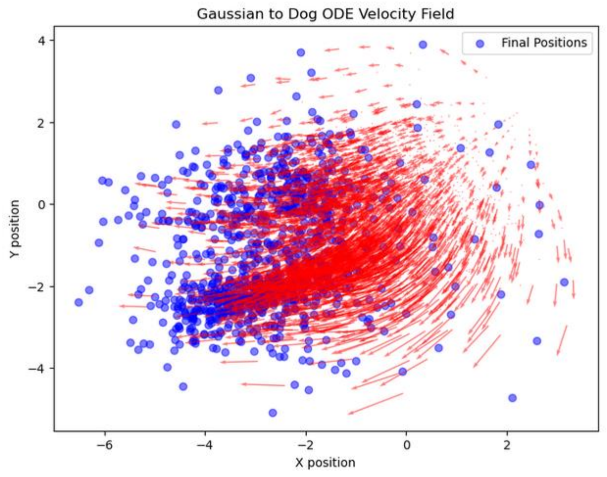
ODE Model	**ODE (Euler Method), Interpolation**
Mapping	Gaussian to Dog
SDE Parameters	T = 1, N = 40, dt = 0.025
Scaling Factor (s)	1.000
NFE	40
MSE	0.10783
Wasserstein Dist.	0.62838



ODE Output, All Probability Densities



ODE Output, All Probability Densities (Zoomed)

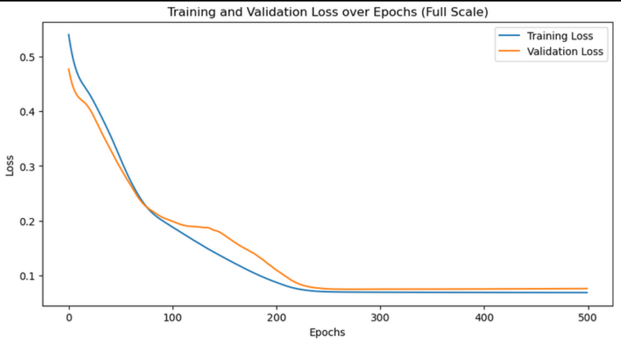


ODE Velocity Field

Comments: Also a good mapping; ODE more organized, as expected of linear interpolation.

Table 3: ODE Gaussian to Dog Summary and Output.

Model f_4: Cats to Dogs Complete Diffusion Model



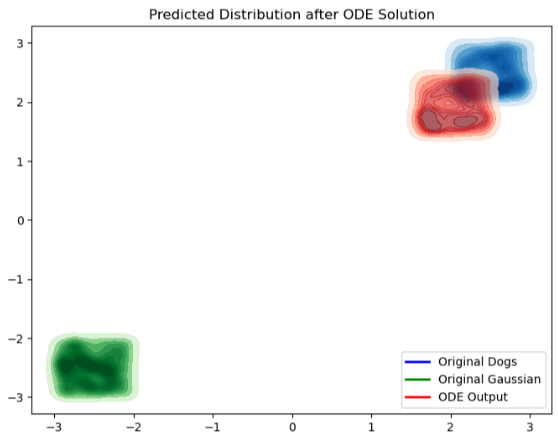
Score Function Training & Validation Loss Over Epochs

Epoch	Train Loss	Val Loss	Predicted Train Residual (Last Entry)	Final Train Score (Last Entry)
0	0.539	0.476	[-0.154, -0.199]	[-0.241, -0.31]
50	0.312	0.295	[-0.405, 0.480]	[-0.634, 0.750]
100	0.188	0.199	[-0.619, 1.203]	[-0.967, 1.881]
150	0.132	0.173	[-0.324, 1.400]	[-0.507, 2.188]
200	0.087	0.110	[-0.243, 1.503]	[-0.379, 2.349]
250	0.070	0.075	[-0.252, 1.497]	[-0.394, 2.340]
300	0.069	0.075	[-0.250, 1.510]	[-0.391, 2.361]
350	0.069	0.075	[-0.243, 1.513]	[-0.380, 2.366]
400	0.069	0.075	[-0.244, 1.514]	[-0.381, 2.368]
450	0.069	0.076	[-0.245, 1.516]	[-0.384, 2.370]
500	**0.069**	**0.076**	**[-0.247, 1.519]**	**[-0.385, 2.375]**

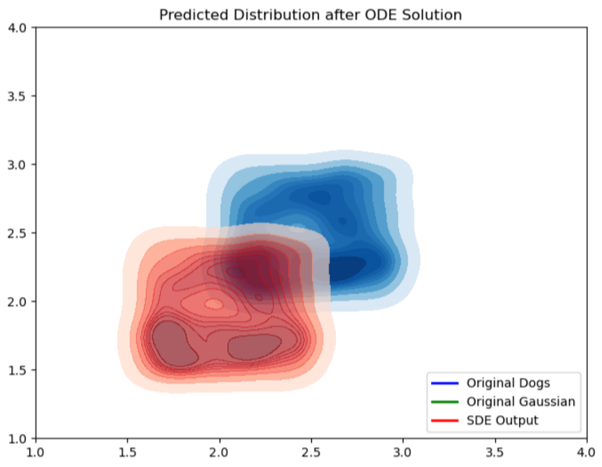
Final Train Loss = 0.069 | Final Val Loss = 0.076

Model Summary

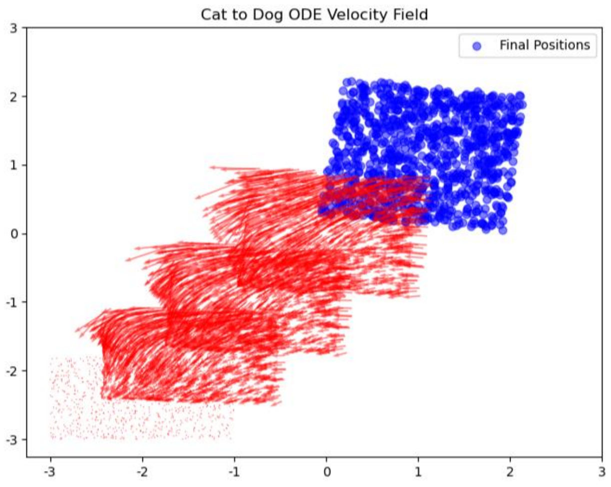
ODE Model	**ODE (Euler Method), Interpolation**
Mapping	Cat to Dog
SDE Parameters	T = 1, N = 40, dt = 0.025
Scaling Factor (s)	1.000
NFE	40
MSE	0.26016
Wasserstein Dist.	1.01490



ODE Output, Cats to Dogs



ODE Output, Cats to Dogs (Zoomed)



ODE Velocity Field, Cats to Dogs

Comments: Decent mapping; ODE shows a logical transformation across the velocity field.

Table 4: Cats to Dogs Score Matching and ODE Mapping.

6. REFERENCES

[1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep Unsupervised Learning using Nonequilibrium Thermodynamics,” in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., in Proceedings of Machine Learning Research, vol. 37. Lille, France: PMLR, Oct. 2015, pp. 2256–2265. [Online]. Available: https://proceedings.mlr.press/v37/sohl-dickstein15.html

[2] A. Hyvärinen, “Estimation of Non-Normalized Statistical Models by Score Matching.,” *Journal of Machine Learning Research*, vol. 6, pp. 695–709, Jan. 2005.

[3] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-Based Generative Modeling through Stochastic Differential Equations,” 2021. [Online]. Available: https://arxiv.org/abs/2011.13456

[4] J. Ho, A. Jain, and P. Abbeel, “Denoising Diffusion Probabilistic Models,” 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[5] B. Kleijn, “Diffusion [Lecture 9 Notes],” *Master of Artificial Intelligence, Victoria University of Wellington*, Sep. 2024, Accessed: Oct. 04, 2024. [Online]. Available: https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML425_2024T2/LectureSchedule/diffusion425.pdf

[6] B. D. O. Anderson, “Reverse-time diffusion equation models,” *Stoch Process Their Appl*, vol. 12, no. 3, pp. 313–326, May 1982, [Online]. Available: https://ideas.repec.org/a/eee/spapps/v12y1982i3p313-326.html

[7] B. Kleijn, “Normalising Flows [Lecture 8 Notes],” *Master of Artificial Intelligence, Victoria University of Wellington*, Sep. 2024, Accessed: Oct. 06, 2024. [Online]. Available: https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML425_2024T2/LectureSchedule/flows.pdf