An Analysis of Score Based Modelling

Joshua Brown, with supervisor Susan Wei

University of Melbourne

Introduction

Consider the challenge of fitting an unnormalised model to observed data. Specifically, we have data $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n \sim p_d(\mathbf{x})$ to which we wish to fit a model density $p_m(\mathbf{x}|\theta) = q(\mathbf{x}|\theta)/Z(\theta)$ known only up to an intractable normalising constant $Z(\theta)$. One technique to circumvent this difficulty is called score matching [2]. This poster takes a closer look at this estimation method, its contemporary descendant and explores possible applications.

Score Matching & Sliced Score Matching

We define the score function of our data and our model to be $s_d(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_d(\mathbf{x})$ and $s_m(\mathbf{x}|\theta) = \nabla_{\mathbf{x}} \log p_m(\mathbf{x}|\theta)$, respectively. It is straightforward to see that $\nabla_{\mathbf{x}} \log p_m(\mathbf{x}) = \nabla_{\mathbf{x}} \log q(\mathbf{x}|\theta)$. Our estimator for θ comes from minimising the expected squared distance between $s_d(\mathbf{x})$ and $s_m(\mathbf{x}|\theta)$. This expectation can be reformulated through a trick with integration by parts:

$$\begin{aligned} J_1(\theta) &:= \mathbb{E}_{p_d}\left[\frac{1}{2}\left\|s_m(\mathbf{x}|\theta) - s_d(\mathbf{x})\right\|^2\right] \\ &= \mathbb{E}_{p_d}\left[\text{tr}(\nabla_{\mathbf{x}} s_m(\mathbf{x}|\theta)) + \frac{1}{2}\left\|s_m(\mathbf{x}|\theta)\right\|^2\right] + C, \end{aligned}$$

where C does not depend on θ. An estimator of this expectation can be computed through the law of large numbers: $\hat{J}_1(\theta) = \frac{1}{n}\sum_{i=1}^n \left[\text{tr}(\nabla_{\mathbf{x}} s_m(\mathbf{x}_i|\theta)) + \frac{1}{2}\left\|s_m(\mathbf{x}_i|\theta)\right\|^2\right]$, assuming the data is a random sample. Then our estimator for θ is $\hat{\theta} = \arg\min_{\theta} \hat{J}_1(\theta)$.

One problem with this estimator is that when the dimension of the data $\mathbf{x}$ is high, computation of $\text{tr}(\nabla_{\mathbf{x}} s_m(\mathbf{x}_i|\theta))$ is very costly. An idea presented in [5] proposes projecting random vectors v_i sampled from p_v (usually Gaussian or Rademacher) to approximate this trace. The objective function then becomes

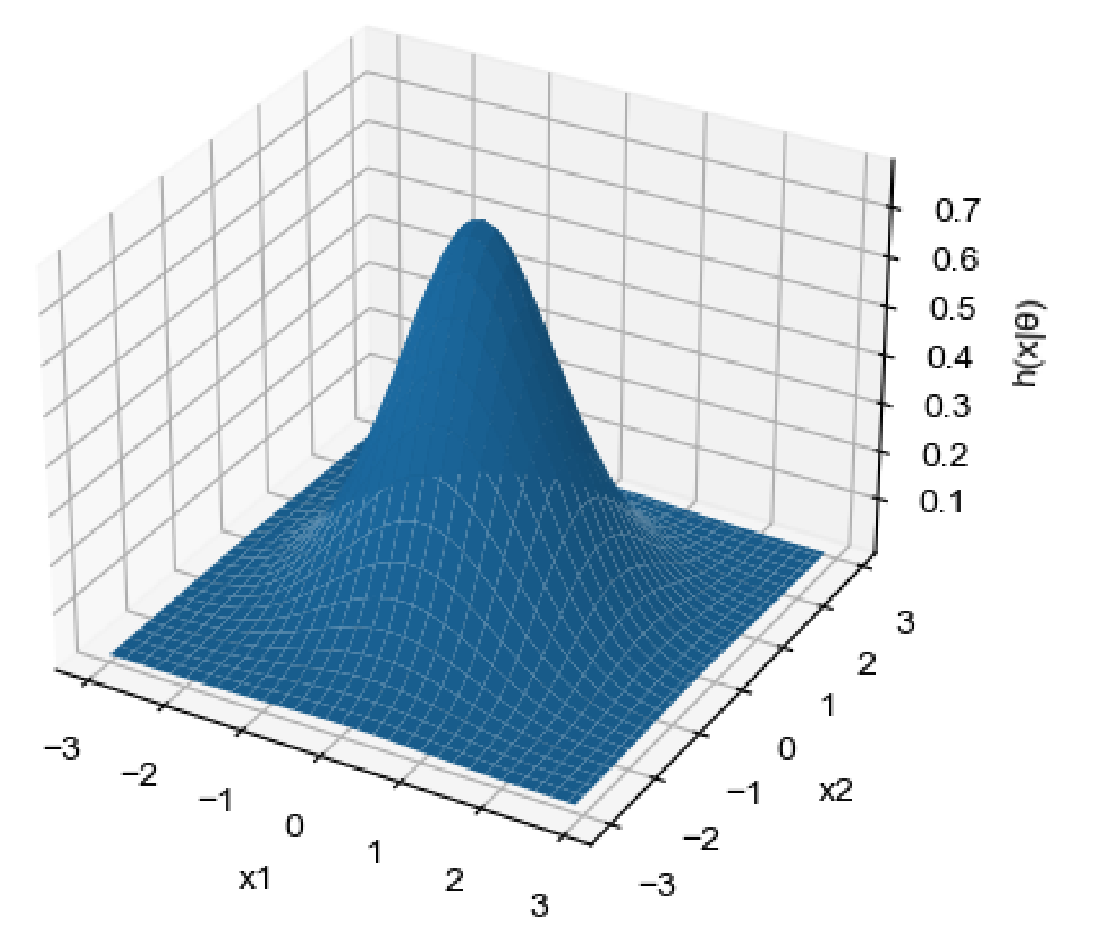
$$J_2(\theta) = \mathbb{E}_{p_v}\mathbb{E}_{p_d}\left[\mathbf{v}^T \nabla_{\mathbf{x}} s_m(\mathbf{x}|\theta)\mathbf{v} + \frac{1}{2}\left\|s_m(\mathbf{x}|\theta)\right\|^2\right]$$

Using auto-differentiation methods and optimising the above using θ gives us a computationally effective estimator for θ.

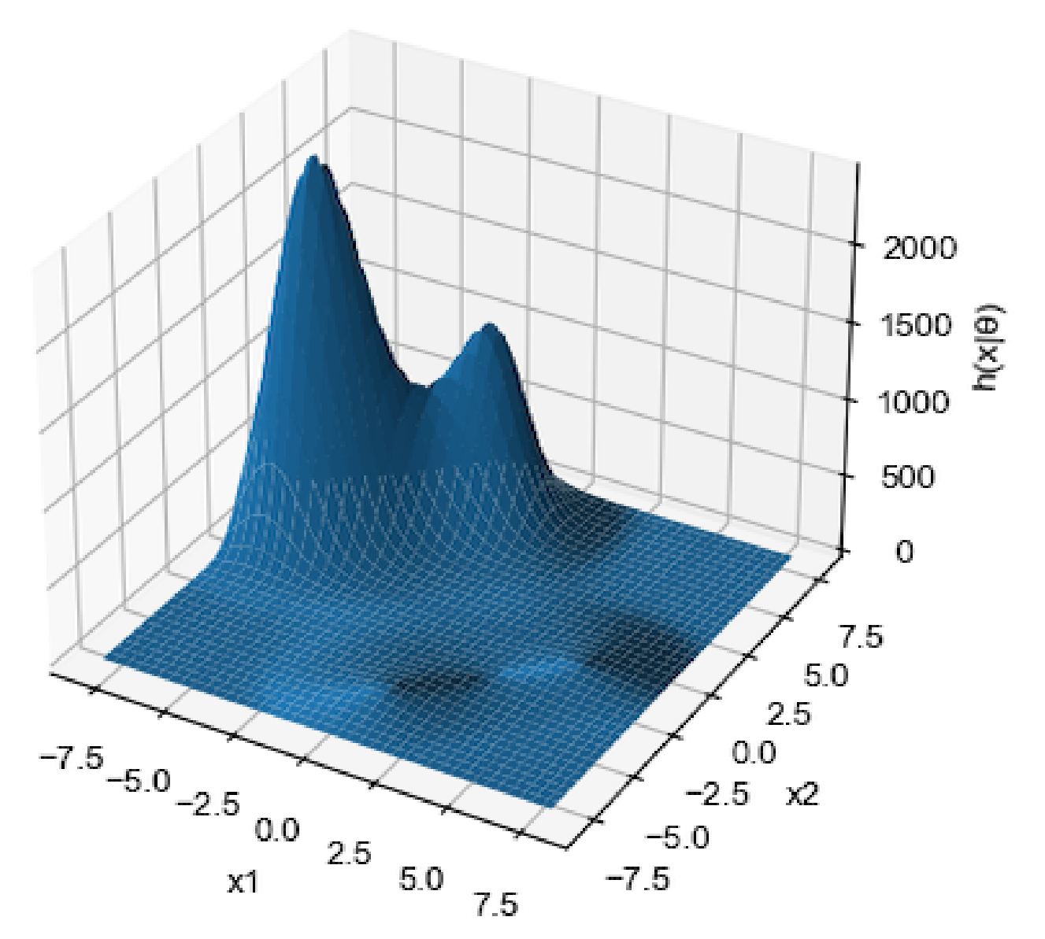
Score Matching for Energy Based Model Estimation

We begin our exploration by considering score-matching within the context of deep learning. We consider having a neural network $h(\mathbf{x}|\theta)$ approximate our $p_d(\mathbf{x})$ through the objective detailed above, i.e setting $p_m(\mathbf{x}|\theta) = h(\mathbf{x}|\theta)$. Our architecture for the network is given by

- 3 hidden layers;
- 50 nodes in each hidden layer;
- Swish activation function



(a) Gaussian



(b) Gaussian Bernoulli RBM

Figure 1. Two distributions modelled by score matching.

Issues in Modelling Density

The main issue with this approach is that it is difficult to ensure the network $h(\mathbf{x}|\theta)$ satisfies all properties of a density function. Namely, while the network accurately models the "shape" of the data distribution, we face difficulty in enforcing the restriction

$$\int_{\mathbf{x}\in\mathbb{R}^d} h(\mathbf{x}|\theta)\, d\mathbf{x} = 1$$

This is why in figure 1 we see modes of unrealistic height for a density function. This network therefore approximates an unnormalised $p_d(\mathbf{x})$. Hence, a more sensible approach may be modelling the score function by the neural network, i.e setting $s_m(\mathbf{x}|\theta) = h(\mathbf{x}|\theta)$ in $\hat{J}_2(\theta)$. This circumvents having to take two derivatives in our loss function, reducing computation significantly, and achieves the same objective of minimising the distance between $s_d(\mathbf{x}|\theta)$ and $s_m(\mathbf{x}|\theta)$. The efficiency of this network compared to a baseline KDE for the score is expressed in table 1.

Dimension of x	Sliced Score Matching	Kernel Density Estimation
2	0.12	0.26
10	0.69	2.72
20	1.23	3.89
40	2.02	5.42

Table 1. Comparison of the average euclidean distance between estimated score and theoretical score of a Gaussian random variable using two different techniques. Each model was trained with 10,000 data points and evaluated on 1,000 points unseen in training. The neural network was evaluated after 50 epochs of training, and 1 projection vector was used to approximate the expectation in $\hat{J}_2(\theta)$.

Implicit VAEs with Score Matching

Variational Autoencoders are a form of neural network constructed to reduce the number of dimensions of data $\mathbf{x}$ in such a way to encode as much information from the original data as possible in the latent variable, $\mathbf{z}$ [3]. The network is built to model (and induce with each $\mathbf{x}$) an encoder distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and a decoder distribution $p_\theta(\mathbf{x}|\mathbf{z})$. It is traditionally optimised using the so-called ELBO objective:

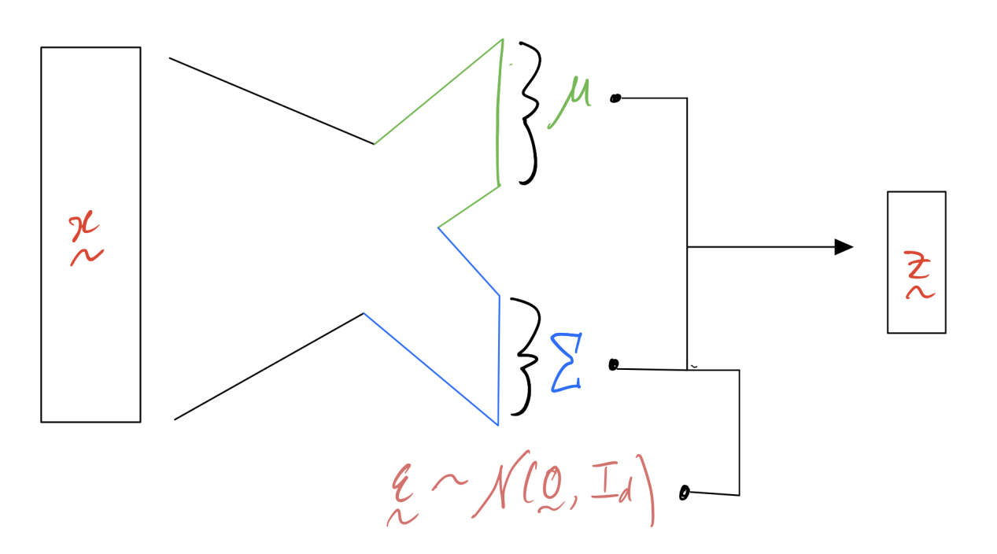
$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})]$$

which forms a lower bound for the log-likelihood of $\mathbf{x}$, $\log p(\mathbf{x})$. To perform gradient descent, we need to compute $\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})]$. Evaluating the gradient with respect to θ can be done quite easily, since q_ϕ is independent of θ, leading to $\mathbb{E}_{q_\phi(\mathbf{z},\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x}|\mathbf{z})]$. When evaluating the gradient with respect to ϕ, we typically find some g such that $\mathbf{z} = g_\phi(\boldsymbol{\epsilon},\mathbf{x})$, where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. This is called the "**reparameterisation trick**". Notice now that

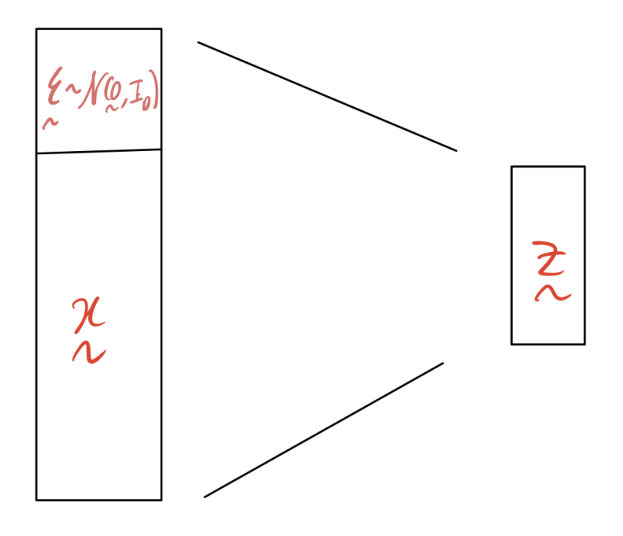
$$\begin{aligned} \mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})], \quad \mathbf{z} = g_\phi(\boldsymbol{\epsilon},\mathbf{x}) \\ \therefore \nabla_\phi \mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_\phi(\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))], \\ \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}) &= \nabla_{\mathbf{z}} \log q_\phi(\mathbf{z}|\mathbf{x})|_{\mathbf{z}=g_\phi(\boldsymbol{\epsilon},\mathbf{x})} \nabla_\phi g_\phi(\boldsymbol{\epsilon},\mathbf{x}) \end{aligned}$$

Hence, the score function of the distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is needed for evaluation of this gradient. Within this framework, there exists a bifurcation between:

1. **Choosing** $q_\phi(\mathbf{z}|\mathbf{x})$ so that $\nabla_{\mathbf{z}} \log q_\phi(\mathbf{z}|\mathbf{x})$ is tractable. This usually involves restricting $\mathbf{z}|\mathbf{x} \sim \mathcal{N}(\mu,\Sigma)$, so that $\mathbf{z}|\mathbf{x} = \mu + \Sigma\cdot\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I_d)$. If we use the neural network modelling the encoder to produce μ and Σ, our reparameterisation is then precisely $g_{\mu,\Sigma}(\boldsymbol{\epsilon},\mathbf{x}) = \mu(\mathbf{x}) + \Sigma(\mathbf{x})\cdot\boldsymbol{\epsilon}$.
2. **Approximating** the score function. We consider using sliced score matching to accomplish this objective. We let our encoder network be $g_\phi(\boldsymbol{\epsilon},\mathbf{x})$ itself, appending $\boldsymbol{\epsilon}$ to each $\mathbf{x}$. After each pass, we estimate $\nabla_{\mathbf{z}} \log q_\phi(\mathbf{z}|\mathbf{x})$ through a score network. In this way, we construct an "implicit" encoder, in the sense that we need make no assumptions about the distribution of $\mathbf{z}|\mathbf{x}$, besides specifying the functional form $\mathbf{z} = g_\phi(\boldsymbol{\epsilon},\mathbf{x})$.



(a) Traditional approach [4]

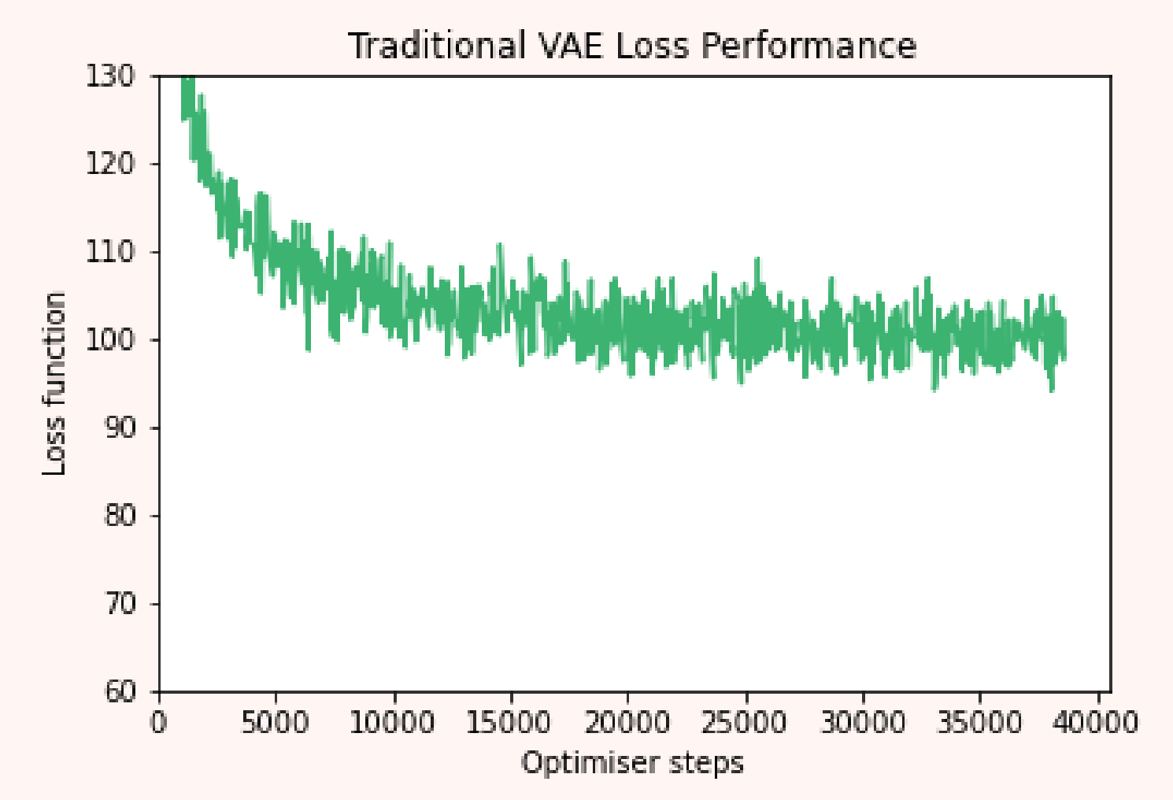


(b) Implicit approach

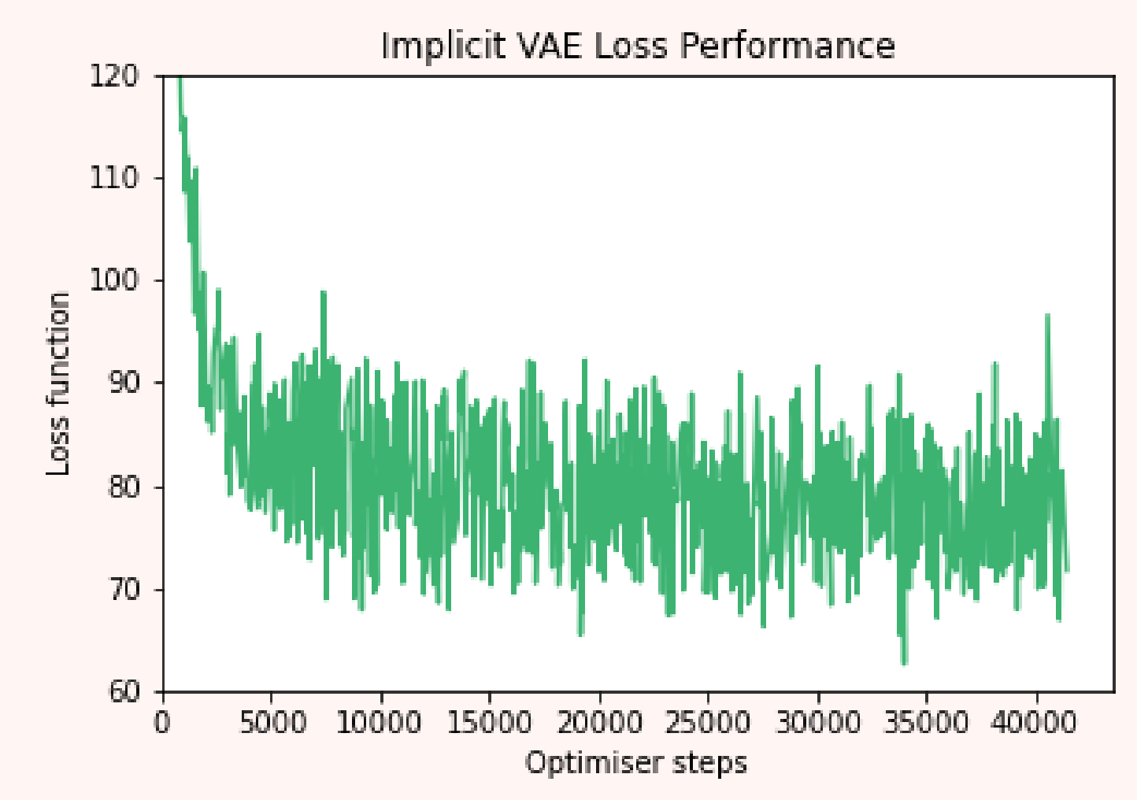
Figure 2. Visual comparison of the encoder section of VAE for each of the choices outlined.

Results & Conclusions

In practice, the way we implement an implicit encoder is by having three neural networks learn in parallel – one for each of the encoder, the decoder, and the score of the encoder. We train this model on MNIST and compare the loss of the autoencoder to that of the traditional approach.



(a) Traditional approach (assuming q_ϕ)



(b) Implicit approach

Figure 3. Loss functions over iterations for different VAE methods.

We expect the greater flexibility of the implicit approach to accommodate a more "optimal" solution. What we observe is a loss curve with a much higher variance than that of the traditional VAE, which suggests a trade-off between bias and variance. Song, Garg, Shi, and Ermon[5] used the negative log-likelihood as a separate evaluation metric and found that the results between the two methods were comparable. Given the computational cost of training the additional score network as opposed to the computationally efficient operations of scaling and translation in the traditional approach, this casts some doubt that the trade-off is ultimately worth it. Though score matching presents us an interesting technique, we wonder if its most promising applications lie elsewhere.

References

[1] Will Grathwohl, Kuan-Chieh Wang, Jorn-Henrik Jacobsen, David Duvenaud, and Richard Zemel. Learning the Stein Discrepancy for Training and Evaluating Energy-Based Models without Sampling. August 2020.

[2] Aapo Hyvarinen. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, May 2005.

[3] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. May 2014.

[4] Joseph Rocca. Understanding Variational Autoencoders (VAEs), March 2021. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

[5] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, page 204, 2019.