

DD2434 Project

Importance Weighted Autoencoders

GROUP 14

Manuel Bradicic, Michal Sitarz, Robert Skoglund, Ida Swartling

January 8th, 2024

Abstract

In this paper we investigate the Importance Weighted Autoencoder (IWAE) and reproduce the results of Burda et. al. [1]. Whilst the model architecture is identical to a variational autoencoder (VAE), the IWAE proposes a new loss function with a strictly tighter lower bound with looser assumptions, opening up for a broader parameter space and learning of richer latent space representations. In the *Introduction* we cover this fundamental theory behind VAEs and derive its loss function's gradient, followed by IWAE's loss function and its properties. In the *Method*, we cover our adaption of Burda's method, including model architectures. In the *Results* we present our experiment results, plotting negative log-likelihoods, training/testing loss as well as a comparison of ground truth samples, VAE predictions and IWAE predictions. Finally, in the *Discussion*, we discuss and justify differences in the *Results* relative to the original study, followed by an overview of related recent works. Overall, the obtained results successfully reflected the original results, showing a clear decrease of negative log-likelihood for larger values of k , i.e. the number of Monte Carlo samples, for both tested architectures. A minor deviation was shown for the MNIST dataset for the two-layer architecture for $k = 5$ which we attributed to potential differences in unspecified training hyperparameters in the original study.

Introduction

The importance weighted autoencoder (IWAE) is a generative model identical to the variational autoencoder (VAE) by Kingma & Welling [7] but with an improved loss function - a strictly tighter lower bound compared to the ELBO without the strong mean-field assumption. This enables for increased flexibility in modelling posteriors and learning of richer latent space representations. The ultimate aim of this paper is to reproduce the results from the original IWAE paper by Burda et. al. [1]. We divide the paper into four sections: (i) *Background* - covering fundamental theory behind the VAE and IWAE, (ii) *Methods* - presenting our adaption to the original study's method, including model architectures (iii) *Results* - the experimental results behind the implemented method, and finally (iv) *Discussion* - a comparison of the obtained results to the original results as well as an overview of related, recent works.

Background

Variational Autoencoder

The VAE models the data $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^n$ with the two following generative processes:

1. A latent variable $\mathbf{z}^{(i)}$ is generated from a prior distribution $p_\theta(\mathbf{z})$
2. A data point $\mathbf{x}^{(i)}$ is generated from the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z}^{(i)})$

The ultimate goal is to perform inference with our posterior $p(\mathbf{z}|\mathbf{x})$, enabling us to conduct tasks such as classification and denoising. By Bayes' rule, the posterior can be expanded as follows:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{\int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}},$$

where $p(\mathbf{x}|\mathbf{z})$, $p(\mathbf{z})$, $p(\mathbf{x})$ are the likelihood, prior and evidence respectively. Unfortunately, however, the integral in the denominator is in many cases intractable, forcing us instead to estimate the posterior with a variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$, where ϕ represents its parameters. In order to make our variational approximation as close to the true posterior as possible, an optimization problem, an objective function first needs to be established. In practice the evidence lower bound (ELBO), denoted $\mathcal{L}(\mathbf{x})$, is used and is derived from evidence or log-likelihood as follows:

$$\begin{aligned} \text{evidence} := \log p(\mathbf{x}) &= \log \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \log \int p(\mathbf{x}, \mathbf{z}) \frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} = \log E_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right] \\ &\geq E_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right] = E_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right] =: \text{ELBO} \end{aligned}$$

The inequality follows from Jensen's inequality. It can further be shown that in fact the ELBO is the difference between the evidence and the Kullback–Leibler (KL) divergence. This is shown by decomposing the KL-divergence below:

$$\begin{aligned} KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})) &:= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right] = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log q_\phi(\mathbf{z}|\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log \left[\frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} \right] \\ &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log q_\phi(\mathbf{z}|\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}, \mathbf{z})] + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}) \\ &= \log p(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \\ \iff \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] &= \log p(\mathbf{x}) - KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})) \end{aligned} \tag{1}$$

Since the evidence is not dependent on the variational parameters ϕ , it is trivial to see that minimizing the KL-divergence w.r.t. ϕ is equivalent to maximizing the ELBO w.r.t. ϕ . The last equation (1) can be re-written as:

$$\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})),$$

This formulation of the first term however poses a problem during training: due to the stochastic nature of the sampling operation, backpropagation cannot be appropriately performed. To overcome this, the so-called reparametrization trick is utilized. Letting $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$, the reparametrization trick substitutes the sampling $\mathbf{z} \sim q_{\phi}$ with a deterministic function g_{ϕ} such that

$$\mathbf{z} = g_{\phi}(\boldsymbol{\epsilon}, \mathbf{x}) = \mu_{\mathbf{x}} + \boldsymbol{\epsilon} \odot \sigma_{\mathbf{x}}$$

where $\boldsymbol{\epsilon}$ is an auxiliary noise variable sampled from a standard multivariate Normal and \odot signifies an element-wise product.

The formulas above can be explicitly interpreted as components of a probabilistic autoencoder, where $q_{\phi}(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$ are the encoder and decoder networks respectively. In practice, there may be many layers of latent variables, each latent variable sampled from a distribution whose parameters are estimated by the output of a previous layer. Note that these layers may resemble complicated non-linear relationships, such as those in multilayer neural networks. Denoting the stochastic layers as $h = \{h_1, \dots, h_L\}$, the encoder and decoder can be fully expressed as:

$$p(\mathbf{x}|\theta) = \sum_{h^1, \dots, h^L} p(h^L|\theta)p(h^{L-1}|h^L, \theta) \cdots p(\mathbf{x}|h^1, \theta),$$

$$q(h|\mathbf{x}) = q(h^1|\mathbf{x})q(h^2|h^1) \cdots q(h^L|h^{L-1}),$$

The prior for the topmost hidden layer $p(h^L)$ is established to be a Gaussian distribution with zero mean and unit variance, i.e. $p(h^L) \sim \mathcal{N}(0, I)$. For each layer of hidden variables, the conditional distributions $p(h^l|h^{l+1})$ and $q(h^l|h^{l-1})$ are also Gaussian with diagonal covariance matrices. The mean and variance of these distributions are not fixed but are computed through a deterministic feed-forward neural network:

$$p(h^l|h^{l+1}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(h^{l+1}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(h^{l+1}))$$

$$q(h^l|h^{l-1}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(h^{l-1}), \boldsymbol{\Sigma}_{\boldsymbol{\phi}}(h^{l-1}))$$

Here $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ represent the neural network-determined mean and covariance for the generative model, while $\boldsymbol{\mu}_{\boldsymbol{\phi}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}$ represent these parameters for the recognition model. For continuous observations \mathbf{x} , the distribution $p(\mathbf{x}|h^1)$ is also a Gaussian whereas for binary observations it is Bernoulli-distributed.

The combined encoding distribution $q(h|\mathbf{x}, \theta)$, encompassing all latent variables, can be reformulated as a deterministic function $\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \theta)$ of both the input \mathbf{x} and a noise vector $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}^1, \dots, \boldsymbol{\epsilon}^L)$, with each $\boldsymbol{\epsilon}_i$ drawn from a standard multivariate Normal distribution. When computing the gradient of our objective function (the ELBO) during backpropagation to pursue the following re-formulation:

$$\nabla_{\theta} \log E_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x}, \theta)} \left[\frac{p(\mathbf{x}, h|\theta)}{q(\mathbf{h}|\mathbf{x}, \theta)} \right] = \nabla_{\theta} E_{\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_L \sim \mathcal{N}(0, I)} \left[\log \frac{p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \theta)|\theta)}{q(\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \theta)|\mathbf{x}, \theta)} \right]$$

$$= E_{\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_L \sim \mathcal{N}(0, I)} \left[\nabla_{\theta} \log \frac{p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \theta)|\theta)}{q(\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \theta)|\mathbf{x}, \theta)} \right].$$

where the first equality is granted by the law of the unconscious statistician (LOTUS) and the second since $\boldsymbol{\epsilon}$ is independent of θ [10]. In practice, we compute the last expression with the following k -sample Monte Carlo approximation:

$$\frac{1}{k} \sum_{i=1}^k \nabla_{\theta} \log w(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \theta), \theta),$$

where $w(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \theta), \theta) = p(\mathbf{x}, \mathbf{h}|\theta)/q(\mathbf{h}|\mathbf{x}, \theta)$. The approximation is an unbiased estimate of the ELBO gradient.

Importance Weighted Autoencoders [1]

The lower bound of the log-likelihood used in the IWAE is a k-sample importance weighting estimate given by

$$\mathcal{L}_k^{IWAE}(\mathbf{x}) = \mathbb{E}_{\mathbf{h}_1, \dots, \mathbf{h}_k \sim q(\mathbf{h}|\mathbf{x})} \left[\log \frac{1}{k} \sum_{i=1}^k \frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})} \right].$$

where $\mathbf{h}_1, \dots, \mathbf{h}_k$ are independent samples from the recognition model. The unnormalized importance weights are denoted by $w_i = p(\mathbf{x}, \mathbf{h}_i)/q(\mathbf{h}_i|\mathbf{x})$. We use Jensen's Inequality, note here that the logarithm is a concave function,

and the fact that we are using an unbiased estimator of the marginal likelihood to show that \mathcal{L}_k is in fact a lower bound on $\log p(\mathbf{x})$:

$$\mathcal{L}_k^{IWAE} = \mathbb{E} \left[\log \frac{1}{k} \sum_{i=1}^k w_i \right] \leq \log \mathbb{E} \left[\frac{1}{k} \sum_{i=1}^k w_i \right] = \log p(\mathbf{x}).$$

For every k we have that

$$\log p(\mathbf{x}) \geq \mathcal{L}_{k+1}^{IWAE} \geq \mathcal{L}_k^{IWAE},$$

and if $p(\mathbf{x}, \mathbf{h})/q(\mathbf{h}|\mathbf{x})$ is bounded it holds when k goes to infinity, \mathcal{L}_k approaches $\log p(\mathbf{x})$. When $k = 1$ the log-likelihood lower bound is the same for both VAE and IWAE:

$$\mathcal{L}_1^{IWAE} = \mathbb{E} \left[\log \frac{1}{1} \sum_{i=1}^1 w_i \right] = \mathbb{E} [\log w] = \mathbb{E} \left[\log \frac{p(\mathbf{x}, \mathbf{h})}{q(\mathbf{h}|\mathbf{x})} \right] = \mathcal{L}^{VAE}$$

Hence \mathcal{L}_k^{IWAE} is a tighter lower bound than the ELBO for $k > 1$.

As for the VAE, the reparametrization trick is used in the IWAE and the auxiliary variables $\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_k$ are drawn independently from a standard multivariate normal distribution. The importance weights are then expressed as a deterministic function of the auxiliary variables, the data, and the parameters obtained by training the neural network. The gradient of the update rule is obtained by the following equation:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}_k^{IWAE} &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{h}_1, \dots, \mathbf{h}_k} \left[\log \frac{1}{k} \sum_{i=1}^k w_i \right] = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_k} \left[\log \frac{1}{k} \sum_{i=1}^k w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}) \right] \\ &= \mathbb{E}_{\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_k} \left[\nabla_{\boldsymbol{\theta}} \log \frac{1}{k} \sum_{i=1}^k w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}) \right] = \mathbb{E}_{\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_k} \left[\sum_{i=1}^k \tilde{w}_i \nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}) \right] \end{aligned}$$

where \tilde{w}_i denotes the normalized importance weights. Monte Carlo estimation is then used and k samples of the recognition network are drawn to obtain the estimate of the gradient.

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_k^{IWAE} \approx \sum_{i=1}^k \tilde{w}_i \nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta})$$

Method

The experimental results presented in *Results* predominantly follow the experimental setup described in the Importance Weighted Autoencoders [1]. To verify that the changes in k affect IWE, all the experiments were compared with a vanilla VAE which served as a baseline. Each experiment was run with three different k values, $k \in \{1, 5, 50\}$ to compare the results directly with the IWAE results from the paper [1]. The difference between the two training loss functions can be summarized in the following equations (see *Introduction* for derivation):

$$\mathcal{L}_k^{VAE} = \frac{1}{k} \sum_{i=1}^k \nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, h(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}), \quad \mathcal{L}_k^{IWAE} = \sum_{i=1}^k \tilde{w}_i \nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, h(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}).$$

Furthermore, all of the experiments were evaluated on two datasets: MNIST [4] and Omniglot [8]. Images from both datasets were binarized, following the technique proposed by Murray & Salakhutdinov [12], where all the pixels were set to 1 stochastically, proportionally to their intensity. The reason is that the authors argued that depending on which binarization technique is implemented, test loss can vary greatly. The images from Omniglot were resized to be equal in size to those from MNIST (28 x 28). Furthermore, to match the size of the Omniglot dataset to the one used in the paper, some of the images from the test set were randomly reallocated to the training dataset. This entails that the test set might not precisely match the one used in the original study. However, there was no clarification on how the original study determined the test set. Therefore, our attempt to replicate their results was closest to the information provided. All the random operations had a seed set to 42.

For VAE and IWAE, two architectures were used, as proposed by Burda et al. [1]. The input to both is 784, which stems from the image size. Moreover, the network weights were initialized [6] with bias equal to zero and the weights with uniform distribution in the given range:

$$W_{ij} \sim U \left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right],$$

where n is the size of the previous layer. The two architectures can be summarized as follows:

1. The encoder and the decoder are each composed of two hidden layers of size 200. In between the layers, the tanh nonlinearity activation function is used. A single stochastic layer of size 50 is used between the encoder and decoder. The stochastic layer uses Gaussian distributions and the output layer uses Bernoulli. (See Figure 3 in *Appendix A*)
2. Similar to Architecture 1, however, the main difference is that now there are two stochastic layers with sizes of 100 and 50, respectively. Thus, between the input layer and the first stochastic layer, there are now two hidden layers of size 200 each, and between the first and second stochastic layers, two hidden layers of size 100 each. Mirroring that, the two decoders have hidden block sizes of 100 and 200, respectively. (See Figure 4 in *Appendix A*)

For our training, we diverged from the paper and adjusted the batch size (explained in) which allowed us to replicate the training time to 3280 iterations over the dataset in total, in a reasonable amount of time. The learning rate was decreased every 3^i iterations, where $i \in [0, 7]$. Finally, the experiments presented in the *Results* were evaluated using the Negative Log-Likelihood. To create a stochastic lower bound on the true values during testing, k was set to 5000.

Results

After conducting experiments as outlined in the original study [1], this study assessed the generative performance of VAE and IWAE by comparing their held-out log-likelihoods on two density estimation benchmark datasets. The evaluation encompasses both qualitative and quantitative analyses of the aforementioned models.

While the majority of the implementation followed the original study, we implemented slight differences. In the original study, authors used a batch size of 20, while in this study for MNIST the batch size of 48 was used, and for OMNIGLOT batch size of 45. Models were being trained on NVIDIA RTX A4000 and the number of workers was set to 4. The modifications aimed to speed up model training. With $k = 1$, the training duration was 3 hours, while with $k = 50$, it extended to 38 hours. Batch size, as well as the suggested learning rate scheduler, were implemented based on the preliminary experiments of training a VAE and remained unchanged through the experiments.

		MNIST		OMNIGLOT	
Stochastic layer	k	VAE	IWAE	VAE	IWAE
1	1	87.68	87.68	111.85	111.88
	5	87.84	85.76	110.98	102.55
	50	87.40	84.88	109.91	99.94
2	1	86.87	86.88	114.66	114.67
	5	90.94	84.78	110.10	104.71
	50	87.60	83.85	108.49	102.52

Table 1: Quantitative comparison of VAE and IWAE performance based on varied stochastic layers and altered parameter k on both datasets MNIST and OMNIGLOT. Note that loss is expressed by NLL (Negative Log-Likelihood)

Experimental results achieved in this study are displayed in Table 1, which aimed to reproduce the original results. The obtained results overall represent a successful replication of the presented methodology - IWAE. There is a minor deviation from the initial findings. However, there exists a visible pattern of improvement of the results with increased parameter k . Note that VAE and IWAE shall behave identically for $k = 1$.

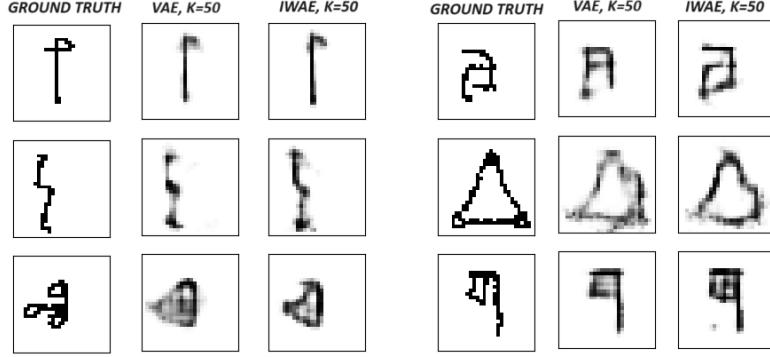


Figure 1: Train loss using parameter $k=50$ for VAE and IWAE on OMNIGLOT; (left) for stochastic layer 1 and (right) stochastic layer 2

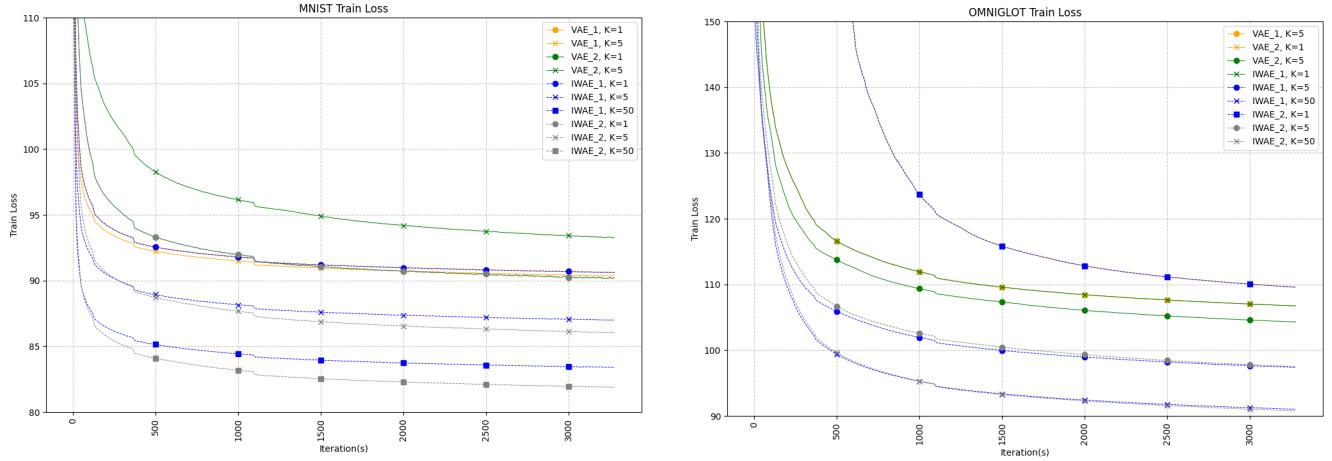


Figure 2: Train loss using different parameters for VAE and IWAE on (left) MNIST and (right) OMNIGLOT

Generally, for $k = 5$ VAE performs the worst, with the highest log-likelihood loss of 90.94. However, $k = 50$ still outperforms $k = 1$, with the lowest loss of 87.40. On the other hand, it is visible that the general improvement in IWAE with increasing parameter k exists. Moreover, in the case of MNIST, as described by Figure 2 (left), more complex architecture with 2 stochastic layers yields better both training and test results. The lowest achieved loss was generated by IWAE with two stochastic layers, 83.85.

For the OMNIGLOT dataset, IWAE slightly outperforms the results achieved in the original study. IWAE shows a significant decrease of loss with increasing k , similarly, VAE slightly benefits from increased k . The lowest achieved loss with VAE is 110.10, whereas the best-performing IWAE has a log-likelihood of 99.94. Training loss across different parameters k and architectures is shown by Figure 2.

Discussion

This study replicated the importance-weighted autoencoder, a variation on the regular VAE trained by maximizing a tighter log-likelihood lower bound derived from importance weighting. According to Burda et al. [1], the VAE results are comparable to those previously documented in the literature, with slight improvement with increasing k . In the case of VAE, a similar pattern can be observed in these experiments, however, in the case of MNIST VAE does not follow the expected outcome. Replicating the VAE result for MNIST, with two stochastic layers and $k = 5$ was not entirely successful. This could be due to some changes in parameters that might have happened during the training process and were not explicitly detailed in the original paper. However, other than that, we showed successful reproducibility of the results from the original study, i.e. the evidence shows the efficiency of IWAE. Results in Table 1 clearly state that in both cases, MNIST and OMNIGLOT, increasing k yields greater results - lower loss. Moreover, diagrams displayed in Figure 2 display increased performance (lower loss) with IWAE models

and increased $k = \{5, 50\}$.

Figure 1 showcases the generated samples from OMNIGLOT using $k = 50$ for both VAE and IWAE $k = 50$ for stochastic layers 1 and 2, respectively. In both cases, it is visible that images generated using the IWAE methodology generate better samples, in terms of their sharpness and details. For instance, the first sample from the top on the right-hand side (stoch. layer 2), in the case of VAE-generated structure differs from the ground truth sample, while the IWAE-generated sample mostly matches the outline of the symbol with slight errors.

The positive correlation between increasing parameter k and improved performance in IWAE models, as evidenced by lower loss in both MNIST and OMNIGLOT datasets, adds credibility to the importance of this methodology. The visualizations provided by Figures 1, 2 provide visual and quantitative support for this argument. In conclusion, these findings align with the theoretical understanding that importance-weighted VAE allows for better posterior approximation. However, it is essential to critically evaluate whether the trade-off in improvement in performance, with increasing k , is practically affordable. While lower loss values are indicative of better performance, the computational cost associated with higher values of k may limit the real-world application of such an approach.

Recent and Future Work

The proposed method of Importance weighted autoencoders, and more specifically the tighter log-likelihood lower bound, has been reinterpreted in an article by Cremer et al. [2]. The authors introduce the idea that the IWAE in fact optimizes the ELBO but that it is using a more complex distribution when doing so.

The lower bound was later reinterpreted once more by Nowozin [9]. The article describes the bound as a biased estimator of the true marginal likelihood and shows that the bias is of order $\mathcal{O}(k^{-1})$ for a k -sample importance weighting estimator. Furthermore, the author introduces the jackknife variational inference (JVI) method as a way of reducing the bias of the estimator to $\mathcal{O}(k^{-(m+1)})$ for any $m < k$. The generalized jackknife technique is here applied to the IWAE estimator and bias is reduced without compromising computational efficiency. It is worth noting however that the resulting estimator is not a lower bound on the log-marginal likelihood and that the method can result in a an increased variance compared to the VAE and IWAE estimators.

In an article by Daudel et. al. [3] the VR-IWAE framework is introduced. This extended formulation generalizes the importance weighted autoencoder (IWAE) bound by combining it with the Variational Rényi (VR) bound. The paper provides a theoretical foundation and proposed further research to investigate practical applications of the VR-IWAE approach.

Some of the negative aspects of the IWAE algorithm are discussed in an article by Finke and Thiery [5]. The authors first point out that the availability of a reparametrization is crucial for the algorithm to work. Secondly, the gradient used in IWAE suffers from noisiness as the number of samples, k , grows. It is also the case that the gradient will never achieve zero variance, even when the estimated posterior is equal to the true posterior. To combat these issues the authors introduce adaptive importance sampling for learning (AISLE) in which the sticking-the-landing IWAE gradient presented by Roeder et al.[11] and the doubly-reparametrised IWAE gradient presented by Tucker et al. [13] are special cases.

To summarize, the IWAE algorithm improves on the VAE approach. However, advances have since been made and there is still room for future research on the topic.

References

- [1] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. *Importance Weighted Autoencoders*. 2016. arXiv: 1509.00519 [cs.LG].
- [2] Chris Cremer, Quaid Morris, and David Duvenaud. *Reinterpreting Importance-Weighted Autoencoders*. 2017. arXiv: 1704.02916 [stat.ML].
- [3] Kamélia Daudel et al. “Alpha-divergence Variational Inference Meets Importance Weighted Auto-Encoders: Methodology and Asymptotics”. In: *Journal of Machine Learning Research* 24.243 (2023), pp. 1–83. URL: <http://jmlr.org/papers/v24/22-1160.html>.
- [4] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.
- [5] Axel Finke and Alexandre H. Thiery. *On importance-weighted autoencoders*. 2020. URL: <https://openreview.net/forum?id=ryg7jhEtPB>.
- [6] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

- [7] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].
- [8] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. “Human-level concept learning through probabilistic program induction”. In: *Science* 350.6266 (2015), pp. 1332–1338.
- [9] Sebastian Nowozin. “Debiasing Evidence Approximations: On Importance-weighted Autoencoders and Jackknife Variational Inference”. In: *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=HyZoi-WRb>.
- [10] Massimiliano Patacchiola. *Estimating the gradient of the ELBO*. 2021. URL: <https://mpatacchiola.github.io/blog/2021/02/08/intro-variational-inference-2.html>.
- [11] Geoffrey Roeder, Yuhuai Wu, and David Duvenaud. *Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference*. 2017. arXiv: 1703.09194 [stat.ML].
- [12] Ruslan Salakhutdinov and Iain Murray. “On the quantitative analysis of deep belief networks”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 872–879.
- [13] George Tucker et al. *Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives*. 2018. arXiv: 1810.04152 [cs.LG].

Appendix A - VAE Architectures

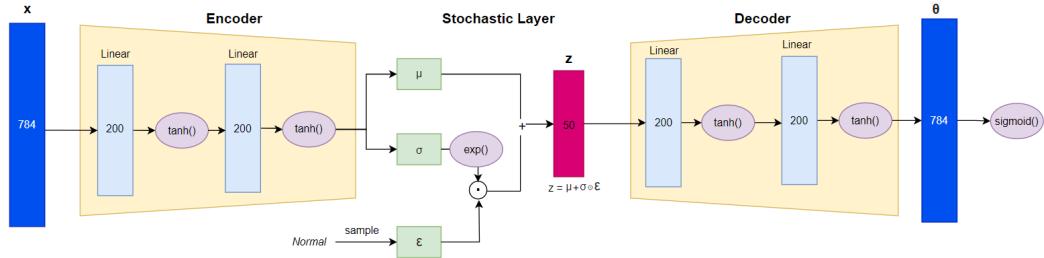


Figure 3: Architecture 1 Diagram

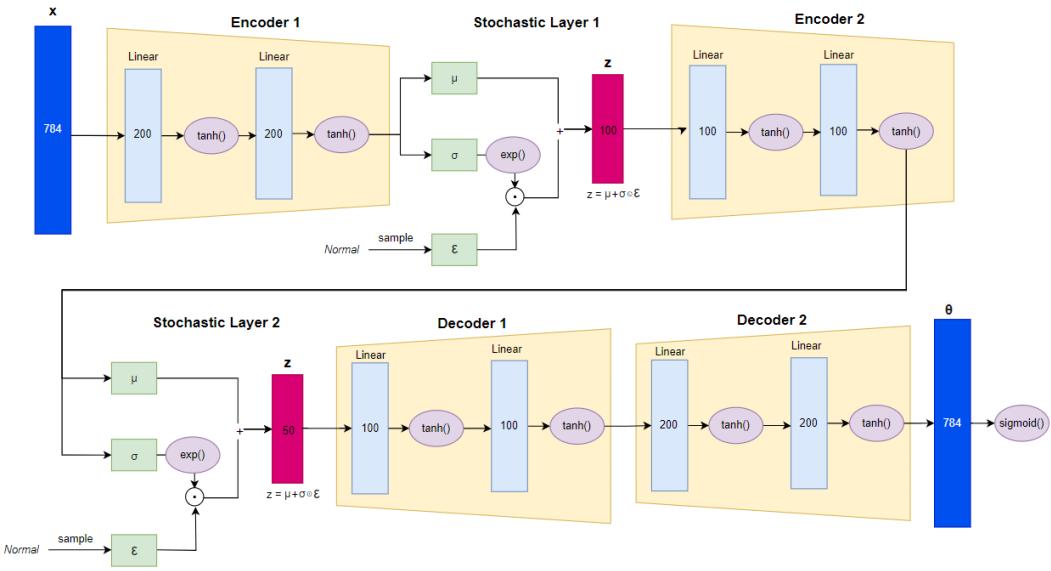


Figure 4: Architecture 2 Diagram