Importance Weighted Autoencoders

Martin Iglesias Goyanes, Alexandre Loiko
martinig@kth.se, loiko@kth.se

September 16, 2022

Abstract

The goal of this project is to re-implement the paper by Burda et. al **"Importance Weighted Autoencoders".** The theory behind regular Variational Autoencoders (VAE) and Importance Weighted Autoencoders (IWAE) is presented in this research, which compares and contrasts them. This paper also presents the related work done in this field, where we analyse the results of three significant and recently released papers in the matter. In regards of the implementation, we explain the methodology used such as the evaluation on density estimation, the weight initialization, binarization, NLL, and the results of fitting a one-dimensional synthetic dataset. We can reproduce the original paper showing that we reach similar scores. We also, as in the original paper, find that IWAEs are better at utilizing their network capabilities than VAEs by learning more expressive latent representations which often results in improved log-likelihood measurements. Furthermore, IWAE reaches better results than VAE on all MNIST models, as measured by NLL and which we can generalize to all models.

1 Introduction

Generative models using an unsupervised approach have become increasingly popular [BGS]. The objective of the generative models is to capture the true data representation when the option for a supervised dataset is not available. In that way, the generative model will simulate how the data in the real world is generated [KW19].The base form of the generators is called *autoencoders* [Lan21]. This is the process that decomposes a model and then composes it back by first breaking down or encoding the data into a lower dimensional representation, then it is rebuilt from the compressed form with a decoder [Lan21].

Autoencoders have shown to be useful in a variety of application fields, notably in fraud detection and image recognition. Recent research has been expanding on the concept of autoencoders. Methods like Variational Autoencoders (VAE) [KW14] are widely used for the mentioned cases. It aims to disentangle, fetch meaningful, statistically independent factors of the data that are variational in its nature known as unsupervised representation learning [KW19].

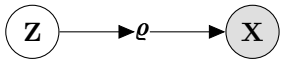
The objective of VAE is to maximize a variational lower bound on the log-likelihood of the data. While VAE brings the benefits of achieving for instance image "drawing" in a realistic manner and being successful in separating style and content [BGS], according to Burda et al. [BGS], the VAE approach has a cost, as it often assumes that the posterior distribution is roughly factorial and that the parameters may be predicted from observed points using nonlinear regression.

The VAE models are trained to maximize a variational lower bound on the log-likelihood, which then encourages the models to develop representations that satisfy the posterior assumptions. Burda et al. [BGS] illustrate that utilizing the VAE aim, an unduly simplistic representation can be generated, resulting in the failure to use the network's modeling capability. Burda et al. [BGS] instead propose a new generative model method that shares the similarity of the VAE architecture called importance weighted autoencoder (IWAE). It is trained with a tighter log-likelihood lower bound generated from importance weighting [BGS]. Multiple approximation posterior samples are generated by the recognition network, with the weight samples being averaged. The lower bound approaches the true value of the log-likelihood as the number of samples is increased [BGS]. The benefits of using multiple samples allow for flexibility for the IWAE-model to learn generative models where the posterior distributions do not fit the VAE assumption [BGS]. According to Burda et al. [BGS], the IWAE can learn richer representations with more latent dimensions than the VAE, allowing for greater log-likelihoods on density estimation benchmarks [BGS]. The question arises whether the empirical results of Burda et al. [BGS] hold, and thus the purpose of this paper is to review and implement the IWAE-model and compare it with the VAE baseline model to find distinctions between the two algorithms. The paper will also delve into the findings and compare them to papers in a more recent context.

2 Theory

Variational Autoencoder

As mentioned in section 1, a VAE is a generative probabilistic model for a dataset $\mathbf{D}$, and also a method for dimensionality reduction. The assumption of VAE is that a data point $\mathbf{X} \in \mathbf{D}$ is sampled from a parameterized family of distributions [BGS], in which the parameters are computed from a latent variable $\mathbf{Z}$. The distribution family for $\mathbf{X} \mid \mathbf{Z}$ used in [BGS] is either Gaussian or Bernoulli. The latent variable is assumed to be real-valued and to have a fixed prior $p(\mathbf{Z}) = \mathcal{N}(\mathbf{Z} \mid \mathbf{0}, \mathbf{I})$. The graphical model below shows the parameter $\boldsymbol{\varrho}$ for $\mathbf{X} \mid \mathbf{Z}$ being generated from $\mathbf{Z}$, and then used to generate $\mathbf{X}$.



The main model of [BGS]

The main example used in [BGS] and re-implemented in this work is binarized MNIST, first described in [LBBH98]. The support of $\mathbf{X}$ is $[0,1]^{28\times 28}$. The latent space is $\mathbb{R}^{50}$. The $\mathbf{X} \mid \mathbf{Z}$ parameter is $\boldsymbol{\varrho} = \boldsymbol{\mu}$ - the expected value of each pixel X_i, that is $p(\mathbf{X} \mid \mathbf{Z}) = \prod_i \text{Bernoulli}(X_i \mid \mu_i(\mathbf{Z}))$.

Fitting a VAE model

To fit a probabilistic generative model, the most fundamental technique is maximum likelihood. That would amount to finding $\text{argmax}_{\boldsymbol{\varrho}} \prod_i p(\mathbf{x}^i; \boldsymbol{\varrho})$ over the space of functions $\boldsymbol{\varrho}(\mathbf{z})$, where for each data point $\mathbf{x}$, $p(\mathbf{x}; \boldsymbol{\varrho}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \boldsymbol{\varrho}(\mathbf{z}))p(\mathbf{z})d\mathbf{z}$. To reduce over-fitting, and make this optimization problem feasible, the possible functions $\boldsymbol{\varrho}(\mathbf{Z})$ are parametrized by $\boldsymbol{\theta}$. A naive approach is to find a local maximum of $p(\mathbf{D}; \boldsymbol{\theta}) = \prod_i p(\mathbf{x}^i; \boldsymbol{\theta})$ though gradient ascent. It requires computing $\nabla_{\boldsymbol{\theta}} p(\mathbf{x}; \boldsymbol{\theta}) = \int_{\mathbf{z}} \nabla_{\boldsymbol{\theta}} p(\mathbf{x} \mid \boldsymbol{\varrho}(\mathbf{z}; \boldsymbol{\theta}))p(\mathbf{z})d\mathbf{z}$. When $\boldsymbol{\varrho}(\mathbf{z})$ is a neural network, and $\boldsymbol{\theta}$ is its weight parameters, this integral cannot be computed analytically. In theory, it can be rewritten as $\text{E}_{\mathbf{z}}[p(\mathbf{x} \mid \boldsymbol{\varrho}(\mathbf{z}; \boldsymbol{\theta}))]$ and approximated by Monte-Carlo sampling. In practice, Monte-Carlo sampling is infeasible for this problem as it has a very slow convergence rate [BGS].

Instead of maximizing $p(\mathbf{D})$ directly, VAE introduces a new parameter $\boldsymbol{\varphi}$ and defines the *expectation lower bound* (ELBO) $\mathcal{L}(\mathbf{D}; \boldsymbol{\theta}, \boldsymbol{\varphi})$. It has the property $\log p(\mathbf{D}) \geq \mathcal{L}(\mathbf{D}; \boldsymbol{\theta}, \boldsymbol{\varphi})$. In an ideal situation, maximizing $\mathcal{L}$ over $\boldsymbol{\theta}, \boldsymbol{\varphi}$ will yield a value $\boldsymbol{\theta}$ for which $p(\boldsymbol{\theta})$ is also maximized, or at least close to a local optimum. To derive the ELBO, note that for any distribution $q(\mathbf{z})$, $\log p(\mathbf{x}) = \log \int_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})d\mathbf{z} = \log \int_{\mathbf{z}} q(\mathbf{z}) \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})} d\mathbf{z} = \log \text{E}_{\mathbf{Z}\sim q(\mathbf{Z})} \left[\frac{p(\mathbf{x},\mathbf{Z})}{q(\mathbf{Z})} \right]$. The logarithm

function is concave, therefore Jensen's inequality implies

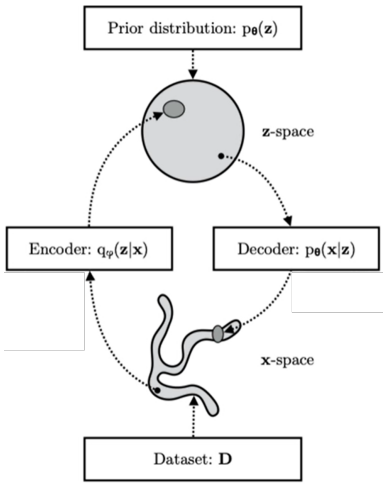
$$\log p(\mathbf{x}) \geq \mathrm{E}_{\mathbf{Z}\sim q(\mathbf{Z})}\left[\log \frac{p(\mathbf{x},\mathbf{Z})}{q(\mathbf{Z})}\right] \tag{1}$$

Equation (1) is the ELBO inequality. The method of VAE is to parametrize q by $\boldsymbol{\varphi}$ and maximize the RHS of (1) instead of the likelihood. If the family $\{q(\bullet;\boldsymbol{\varphi})\}_{\boldsymbol{\varphi}}$ is large enough to contain $\mathbf{z} \mapsto p(\mathbf{z} \mid \mathbf{x})$, the inequality in (1) simplifies to an equality. In this situation, maximizing the ELBO is equivalent to maximizing the log-likelihood. With multiple samples $\mathbf{D} = \mathbf{x}^{1:N}$, we allow $q(\mathbf{z};\boldsymbol{\varphi}) = q(\mathbf{z} \mid \mathbf{x};\boldsymbol{\varphi})$ to depend on the data point $\mathbf{x}$. Otherwise, there is no hope that $q(\mathbf{z};\boldsymbol{\varphi})$ can approximate the true posterior. We hope that after model optimization, $q(\mathbf{z} \mid \mathbf{x})$ will approximate the posterior $p(\mathbf{z} \mid \mathbf{x})$. Therefore we use a probabilistic notation $q(\mathbf{z} \mid \mathbf{x};\boldsymbol{\varphi})$ despite the fact that q is nothing more that a function that given $\mathbf{x}, \boldsymbol{\varphi}$ produces a distribution for $\mathbf{z}$. We use the notation

$$\mathcal{L}(\mathbf{D};\boldsymbol{\varphi},\boldsymbol{\theta}) = \sum_{\mathbf{x}^i\in\mathbf{D}} \mathrm{E}_{\mathbf{Z}\sim q(\mathbf{Z})}\left[\log \frac{p(\mathbf{x}^i,\mathbf{Z};\boldsymbol{\theta})}{q(\mathbf{Z},\mathbf{x}^i;\boldsymbol{\varphi})}\right] \tag{2}$$

for the ELBO.

Another way to interpret the VAE model is to use auto encoder terminology [Lan21]. Then the process that generates $\mathbf{X}$ from $\mathbf{Z}$ according to $p(\mathbf{X} \mid \mathbf{Z})$ is called the *decoder*, and the process for producing an approximate posterior Z from $q(\mathbf{Z} \mid \mathbf{X})$ is the *encoder*:



(note that in our formulation of VAE, the prior does not depend on $\boldsymbol{\theta}$ in contrast to the picture).

Fitting the model involves approximating $\nabla_{\boldsymbol{\varphi},\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta})$ by Monte-Carlo. It is done by re-parametrizing the expectation in (2) as $\boldsymbol{\epsilon} \sim \mathrm{Distr}, \mathbf{Z} = \mathbf{g}(\boldsymbol{\epsilon},\mathbf{x};\boldsymbol{\varphi})$ [BGS] where Distr does not depend on $\boldsymbol{\varphi}$. Then the gradient of $\mathcal{L}$ can be computed as

$$\nabla_{\boldsymbol{\varphi},\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta}) = \sum_i \nabla_{\boldsymbol{\varphi},\boldsymbol{\theta}}\,\mathrm{E}_{\boldsymbol{\epsilon}\sim\mathrm{Distr}}\left[\log \frac{p(\mathbf{x}^i,\mathbf{g}(\boldsymbol{\epsilon},\mathbf{x};\boldsymbol{\varphi});\boldsymbol{\theta})}{q(\mathbf{g}(\boldsymbol{\epsilon},\mathbf{x};\boldsymbol{\varphi}),\mathbf{x};\boldsymbol{\varphi})}\right] = \sum_i \mathrm{E}_{\boldsymbol{\epsilon}\sim\mathrm{Distr}}\left[\nabla_{\boldsymbol{\varphi},\boldsymbol{\theta}} \log \frac{p(\mathbf{x}^i,\mathbf{g}(\boldsymbol{\epsilon},\mathbf{x};\boldsymbol{\varphi});\boldsymbol{\theta})}{q(\mathbf{g}(\boldsymbol{\epsilon},\mathbf{x};\boldsymbol{\varphi}),\mathbf{x};\boldsymbol{\varphi})}\right] \tag{3}$$

We can simplify (3) by the notation $w^i(\boldsymbol{\epsilon}) = \frac{p(\mathbf{x}^i,\mathbf{g}(\boldsymbol{\epsilon},\mathbf{x};\boldsymbol{\varphi});\boldsymbol{\theta})}{q(\mathbf{g}(\boldsymbol{\epsilon},\mathbf{x};\boldsymbol{\varphi}),\mathbf{x};\boldsymbol{\varphi})}$. The expression (3) can be approximated by independently sampling $\boldsymbol{\epsilon}$ and computing $\nabla_{\boldsymbol{\varphi},\boldsymbol{\theta}} w^i(\boldsymbol{\epsilon})$. For each data point $\mathbf{x}^i$, $\boldsymbol{\epsilon}^i_1, \ldots \boldsymbol{\epsilon}^i_K$ are sampled from the stationary noise distribution. The gradient update rule for gradient ascent with a single $\mathbf{x}^i$ is then

$$\widehat{\nabla\mathcal{L}}(\boldsymbol{\varphi},\boldsymbol{\theta}) = \frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{K} \nabla_{\boldsymbol{\varphi},\boldsymbol{\theta}} \log w^i(\boldsymbol{\epsilon}^i_j) \tag{4}$$

The approximated gradient $\widehat{\nabla\mathcal{L}}$ can be summed over the training data, or computed stochastically and with mini-batches.

Importance Weighted Autoencoders

VAE can fail to find $\boldsymbol{\theta}$ that approximates a local optimum of the likelihood $p(\mathbf{D};\boldsymbol{\theta})$ [BGS]. We drop the parameters $\boldsymbol{\theta}, \boldsymbol{\varphi}$ and rewrite the ELBO for a single data point:

$$\mathcal{L}(\mathbf{x}) = \mathrm{E}_q\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\right] = \mathrm{E}_q\left[\log \frac{p(\mathbf{z} \mid \mathbf{x})p(\mathbf{x})}{q(\mathbf{z} \mid \mathbf{x})}\right] = \mathrm{E}_q\left[\log p(\mathbf{x})\right] - \mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z} \mid \mathbf{x})) = \log p(\mathbf{x};\boldsymbol{\theta}) - \mathrm{KL}(q(\mathbf{z} \mid \mathbf{x};\boldsymbol{\varphi}) \mid\mid p(\mathbf{z} \mid \mathbf{x};\boldsymbol{\theta})) \tag{5}$$

The KL term in (5) is the Kullback-Liebler divergence between the approximated posterior $q(\mathbf{z} \mid \mathbf{x})$ and the true posterior $p(\mathbf{z} \mid \mathbf{x})$. Maximizing $\mathcal{L}$ is equivalent to maximizing (5). Maximizing (5) over $\boldsymbol{\theta}$ with $\boldsymbol{\varphi}$ held constant can be interpreted as optimization under a constraint: VAE searches for $\boldsymbol{\theta}$ that gives a high likelihood, but only under the constraint that the approximated posterior is not to far from the true. [BGS] argues that VAE often does not find a goot fit for $\boldsymbol{\theta}$, because a family of functions $\{\mathbf{z} \mapsto q(\mathbf{z} \mid \mathbf{x};\boldsymbol{\varphi})\}_{\boldsymbol{\varphi}}$ is not rich enough to approximate the true posterior. In that situation, (5) trades off higher likelihood $\log p(\mathbf{x};\boldsymbol{\theta})$ for lower KL divergence, which brings the VAE loss down without achieving optimal likelihood. In section 3, we describe a one-dimensional model in which VAE is not able to model the true distribution $p(x)$ and show how it is improved by IWAE.

The solution proposed in [BGS] is to derive a different and tighter ELBO that is closer to the true log-likelihood. If the ELBO is a good approximation of the log-likelihood for all $\boldsymbol{\theta}$, then maximizing the ELBO will come close to maximizing the log-likelihood.

The VAE ELBO inequality (1) follows from $\log \mathrm{E}[Y] \geq \mathrm{E}[\log Y]$. If Y is replaced by $\overline{Y} = \frac{1}{K}Y_i$ with $Y_i \sim Y$ i.id., $\lim_{K\to\infty} \mathrm{E}\left[\log \overline{Y}\right] = \mathrm{E}[\log \mathrm{E}[Y]] = \log \mathrm{E}[Y]$ by the law of large numbers. Therefore, we can make Jensen's log-inequality tighter by averaging and re-sampling Y before taking logarithms.

In the IWAE ELBO, $\frac{p(\mathbf{x},\mathbf{Z})}{q(\mathbf{Z})}$ has the role of Y where $\mathbf{Z}$ is drawn from $q(\mathbf{Z})$. Instead of drawing a single $\mathbf{Z} \sim q(\mathbf{Z})$, we now sample K i.id. values $\mathbf{Z}_1, \ldots, \mathbf{Z}_K \sim q(\mathbf{Z})$ and obtain the inequality

$$\log p(\mathbf{x}) \geq \mathrm{E}_{\mathbf{Z}_1,\ldots,\mathbf{Z}_K}\left[\log\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p(\mathbf{x},\mathbf{Z}_k)}{q(\mathbf{Z}_k)}\right)\right] \tag{6}$$

The RHS of (6) is the IWAE ELBO and is denoted $\mathcal{L}_K(\mathbf{x})$. To maximize $\mathcal{L}_k$, identical steps are done as when the VAE ELBO is maximized: parametrize p and q by $\boldsymbol{\theta}, \boldsymbol{\varphi}$, re-parametrize $\mathbf{Z} \sim q(\mathbf{Z} \mid \mathbf{x}; \boldsymbol{\varphi})$, generate noise $\boldsymbol{\epsilon}_1^1 \dots \boldsymbol{\epsilon}_K^1 \dots \boldsymbol{\epsilon}_1^N \dots \boldsymbol{\epsilon}_K^N$ and approximate $\nabla \mathcal{L}_K$ by

$$\widehat{\nabla \mathcal{L}_k}(\mathbf{D}; \boldsymbol{\varphi}, \boldsymbol{\theta}) = \sum_{i=1}^{N} \nabla \log \left(\frac{1}{K} \sum_{j=1}^{K} w^i(\boldsymbol{\epsilon}_j^i) \right) =$$

$$\sum_{i=1}^{N} (\nabla \overline{w}) \frac{1}{\overline{w}} = \sum_{i=1}^{N} \sum_{j=1}^{K} \frac{1}{K \overline{w}} \nabla w^i(\boldsymbol{\epsilon}_j^i) =$$

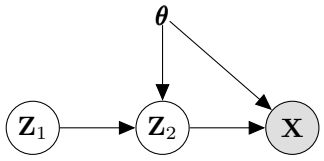
$$\sum_{i=1}^{N} \sum_{j=1}^{K} \frac{1}{K \overline{w}} w^i(\boldsymbol{\epsilon}_j^i) \nabla \log w^i(\boldsymbol{\epsilon}_j^i) =$$

$$\sum_{i=1}^{N} \sum_{j=1}^{K} \widetilde{w_j^i} \nabla \log w^i(\boldsymbol{\epsilon}_j^i) \tag{7}$$

where $\widetilde{w_j^i} = \frac{w^i(\boldsymbol{\epsilon}_j^i)}{\sum_j w^i(\boldsymbol{\epsilon}_j^i)}$ is the normalized j:th w for datapoint i. It is called the relative *importance* of $\mathbf{z}_j^i = g(\mathbf{x}, \boldsymbol{\epsilon}_j^i; \boldsymbol{\varphi})$. The expression (7) is the reason for why the model is called importance weighted autoencoder. During optimization, each $w = \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})} = \frac{p(\mathbf{z} \mid \mathbf{x})}{q(\mathbf{z} \mid \mathbf{x})} \cdot p(\mathbf{x})$. When a w^i is small relatively to other w^i from the same data point x, $p(\mathbf{z}_i \mid \mathbf{x}) > q(\mathbf{z}_i \mid \mathbf{x})$. The IWAE weights down such w^i, because for these $\mathbf{z}_i$ the approximate posterior is wrong, and using them for updating $\boldsymbol{\theta}$ would likely lead to shifting the true posterior to q.

Multiple stochastic layers

[BGS] achieved better results when having two stochastic layers $\mathbf{Z}_1, \mathbf{Z}_2$ rather than one. Introducing multiple stochastic layers is orthogonal to choosing between the VAE and IWAE loss. A generative model with two stochastic layers generates $\mathbf{X}$ has the following graphical model:



Fitting this model is similar to fitting a VAE och IWAE model with a single latent stochastic layer. The differences are that

1. $p(\mathbf{x}, \mathbf{z})$ changes to $p(\mathbf{z}_1) p_1(\mathbf{z}_2 \mid \mathbf{z}_1) p_2(\mathbf{x} \mid \mathbf{z}_2)$
2. The approximated posterior q follows a similar pattern $q(\mathbf{z} \mid \mathbf{x}) = q_2(\mathbf{z}_2 \mid \mathbf{x}) q_1(\mathbf{z}_1 \mid \mathbf{z}_2)$
3. computing the gradients (7, 4) now requires both computing $\boldsymbol{\epsilon}_2$ for $\mathbf{Z}_2$ and then using $\mathbf{Z}_2$ and $\boldsymbol{\epsilon}_1$ to compute $\boldsymbol{Z}_1$.

Related work

Christopher et al. [UB21] implemented a deep learning algorithm for high-dimensional exploratory item factor analysis. Because of the Marginal Maximum Likelihood (MML) estimator's consistency, normalcy, and efficiency as the sample size approaches infinity, it is the favored strategy for fitting item response theory models in psychometrics [UB21]. However, when the sample size and several latent components are large, state-of-the-art MML estimation processes like the Metropolis-Hastings Robbins-Monro (MH-RM) algorithm, as well as approximate MML estimation procedures like variational inference (VI), are computationally time-consuming [UB21]. That is why they propose IWAE, the IWAE uses an importance sampling strategy to approximate the MML estimator, in which increasing the number of importance-weighted (IW) samples generated during fitting improves the approximation at the cost of decreasing computational efficiency [UB21]. Christopher et al. [UB21] IW-ELBO is defined as:

$$\nabla_\delta E_{X_{1:R}} \left[\log \frac{1}{R} \sum_{r=1}^{R} w_r \right] = E_{\epsilon_{1:R}} \left[\sum_{r=1}^{R} \widetilde{w_r} \nabla_\delta \log w_r \right] \approx \tag{8}$$

$$\frac{1}{S} \sum_{s=1}^{S} \left[\sum_{r=1}^{R} \widetilde{w_{r,s}} \nabla_\delta \log w_{r,s} \right] \tag{9}$$

Where $\epsilon_{1:R} \sim \prod_{r=1}^{R} N(\boldsymbol{\epsilon}_r)$ and $\widetilde{w}_r = {}^{w_r} / \sum_{r'=1}^{R} w_r$, are normalized importance weights. The problem of this ELBO is that increasing the R reduces the performance of the above gradient estimator of the inference model parameters. They resolve this issue by using Tucker et al.'s [TLGM18] doubly reparametrized gradient estimator:

$$\nabla_\Psi E_{X_{1:R}} \left[\log \frac{1}{R} \sum_{r=1}^{R} w_r \right] = E_{\epsilon_{1:R}} \left[\sum_{r=1}^{R} \widetilde{w}_r^2 \frac{\partial \log w_r}{\partial \boldsymbol{x}_r} \frac{\partial \boldsymbol{x}_r}{\partial \psi} \right] \approx \tag{10}$$

$$\frac{1}{S} \sum_{s=1}^{S} \left[\sum_{r=1}^{R} \widetilde{w}_{r,s}^2 \frac{\partial \log w_{r,s}}{\partial \boldsymbol{x}_{r,s}} \frac{\partial \boldsymbol{x}_{r,s}}{\partial \psi} \right] \tag{11}$$

According to Christopher et al. [UB21], the double estimator is unbiased, has a higher SNR as R increases, and has lower variance than alternative estimators. Because the IW-ELBO gradient and doubly reparametrized gradient estimators may be successfully estimated with a single Monte Carlo sample in practice [UB21]. Christopher et al. [UB21] concluded, that the amortized importance-weighted variational estimator they proposed, unlike other estimators, performs comparably to state-of-the-art estimators in both the setting where the number of observations increases and the setting where the number of items and the number of observations both increase at the same time, according to their simulations [UB21]. Although, they managed to find several limitations in their proposed approach. Though, they did have several limitations with their proposed approach, mainly tuning

the number of latent factors. The methods of tuning the latent factors failed for large sample sizes, in which they possibly identified that the log-likelihood approximation error may exceed the sampling error.

Lim et al. [LROI21] proposed IWAE as a way of handling non-ignorable missing features in electronic health records data. Electronic health records are commonly used to investigate the relationships between patient health information and outcomes [LROI21]. There are no published algorithms that can manage missing not at random (MNAR) patterns of missingness in the VAE environment, according to Lim et al. [LROI21], and this presents a significant obstacle to their applicability to electronic health records and other modern biological datasets. Hence the proposal of using IWAE. Their specific proposal is NIMIWAE which is IWAE with nonignorable misses [LROI21]. Unlike the traditional IWAE, they introduce two latent variables Z and X^m. Like the traditional IWAE, they draw K samples of Z however, they additionally draw M samples of $\boldsymbol{X}^m$ for each sample of $\boldsymbol{Z}$ in the same manner. The ELBO is derived as follows [LROI21]:

$$\sum_{i=1}^{n} E_{\{\boldsymbol{Z},\boldsymbol{X}^m\}\sim q_\theta(\boldsymbol{Z},\boldsymbol{X}^m)}\Bigg[\log \frac{1}{K}\frac{1}{M}\sum_{k=1}^{K}\sum_{l=1}^{M} \frac{p_\psi\left(\boldsymbol{x}_i \mid \mathbf{z}_{ik}\right) p\left(\mathbf{z}_{ik}\right) p_\phi\left(\boldsymbol{r}_i \mid \boldsymbol{x}_i^o, \boldsymbol{x}_{ikl}^m, \mathbf{z}_{ik}\right)}{q_{\theta_1}\left(\mathbf{z}_{ik} \mid \boldsymbol{x}_{\boldsymbol{i}}^o, \boldsymbol{r}_{\boldsymbol{i}}\right) q_{\boldsymbol{\theta_2}}\left(\boldsymbol{x}_{ikl}^m \mid \mathbf{z}_{ik}, \boldsymbol{x}_{\boldsymbol{i}}^o, \boldsymbol{r}_{\boldsymbol{i}}\right)} \Bigg] \tag{12}$$

The samples of $\{\boldsymbol{Z}, \boldsymbol{X}^m]$ are drawn from ancestral sampling, and Adam optimizer is used for the lower bound. Lim et al. [LROI21] show through statistical simulations that the proposed model performs well in handling missing futures under MNAR.

In the paper [LX18] the authors demonstrate how VAE could be used for anomaly detection in skin disease images. This work seems to be the first applied study of deep generative models for skin anomalies detection showing solid results being able to detect all diseases with 0.78 AUC, and more specific ones like melanoma with 0.86 AUC.

The VAE is trained by maximizing the evidence of the lower bound of the data distribution. When trained on only normal data, the resulting model is able to perform efficient inference and to determine if a test image is normal or not. [LX18]

The authors proposed VAE Based Score and Importance Weighted Autoencoder (IWAE) Based Score. Focusing on the IWAE based score, the authors propose a tighter lower bound on $\log p(x)$, which is

$$\log p(x) \geq E_{q(z|x)}\left[\log \frac{1}{K}\sum_{i=1}^{K}\frac{p\left(x \mid z_i\right) p\left(z_i\right)}{q\left(z_i \mid x\right)}\right] \tag{13}$$

Where VAE suggest to use the negative of maximizing the evidence lower bound as an anomaly scores like

$$s_{vae}(x) = KL(q(z \mid x)\|p(z)) - \frac{1}{L}\sum_{i=1}^{L}\log p\left(x \mid z_i\right) \tag{14}$$

$$s_{\text{vae}}^{kl} = KL(q(z \mid x)\|p(z)) \tag{15}$$

$$s_{\text{vae}}^{\text{reconst}} = -\frac{1}{L}\sum_{i=1}^{L}\log p\left(x \mid z_i\right) \tag{16}$$

IWAE suggest a similar thing using the negative of Eq.16 to compute the scores as

$$s_{\text{iwae}}\left(x\right) = -\log\left(\frac{1}{L}\sum_{i=1}^{L}\frac{p\left(x \mid z_i\right) p\left(z_i\right)}{q\left(z_i \mid x\right)}\right) \tag{17}$$

$$s_{\text{iwae}}^{kl}\left(x\right) = -\log\left(\frac{1}{L}\sum_{i=1}^{L}\frac{p\left(z_i\right)}{q\left(z_i \mid x\right)}\right) \tag{18}$$

$$s_{\text{iwae}}^{\text{reconst}}\left(x\right) = -\log\left(\frac{1}{L}\sum_{i=1}^{L}p\left(x \mid z_i\right)\right) \tag{19}$$

In this particular study, the authors state that IWAE doesn't introduce a significant difference in the outlier detection.

3 Method

3.1 Evaluation on density estimation

The models were tested on two benchmark datasets: MNIST, which is a dataset including photographs of handwritten digits provided by LeCun et al. [LBBH98]. In addition, the VAE and IWAE were used to fit a one-dimensional synthetic distribution with a one-dimensional latent variable.

In both data sets, the observations are binarized 28×28 images, with the normal splits of MNIST being 60,000 training and 10,000 test instances. Two architectures were used to train the models. The first architecture, with 50 units and a single stochastic layer $\mathbf{z}^1$. There were two deterministic layers with 200 units each between the observations and the stochastic layer. The second architecture contains two stochastic layers, $\mathbf{z}_1$ and $\mathbf{z}_2$, each of which has 100 and 50 units. There were two deterministic layers of 200 units each between $\mathbf{x}$ and $\mathbf{z}_1$. There were two deterministic layers with 100 units each between $\mathbf{z}_1$ and $\mathbf{z}_2$.

For all deterministic hidden units, the tanh non-linearity activation function was utilized, it is used to find exact solutions to nonlinear cases [EWA07]. With the exception of the visible layer, which employed Bernoulli distributions, all stochastic layers used Gaussian distributions with diagonal covariance. An exp non-linearity activation function was used to apply the Gaussian distributions' estimated variances. Glorot and Bengio [GB10] presented a heuristic for initializing all models. For optimizations, Adam [KB17] was used with the values $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-4}$, and mini-batches of size 20. For $i = 0 \ldots 7$, the training proceeded for 3^i runs over the

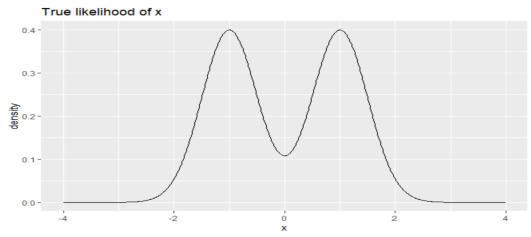


Figure 1: The distribution of the 'camel' data set

data, with a learning rate of $0.001 \cdot 10^{-i/7}$. Preliminary experiments on MNIST with a VAE with one stochastic layer led to the learning rate schedule.

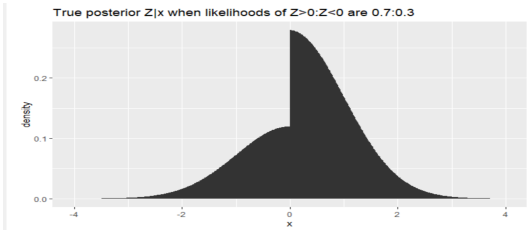
The main way of evaluation was to estimate the expected log-likelihood. Since the ELBO $\mathcal{L}_k$ is an increasingly accurate estimation of the log-likelihood, it was estimated by $\frac{1}{|\mathbf{D}_{\text{test}}|}\mathcal{L}_{5000}(\mathbf{D}_{\text{test}})$. Another evaluation method was to count *active units*. [BGS] discovered that the performance of the model seems higher when more latent space elements are active. A component Z_j is active, if it has $\text{Var}[\text{E}[Z_j \mid \mathbf{X}]] \neq 0$, the motivation being that otherwise Z_j has the same posterior expectation for every data point and can't contain information that predicts $\mathbf{X}$. [BGS] discovered that the histogram of $\{\widehat{\text{Var}}[\text{E}[Z_j \mid \mathbf{X}]] \mid j = 1 \dots \dim(\mathbf{Z})\}$ consists of two tight clusters: one very close to zero, and another one well separated and much higher. We re-create this histogram in Figure 3.

3.2 One-dimensional camel dataset

We used VAE and IWAE to recover the density $p(x) = \frac{1}{2}\mathcal{N}(\mu = -1, \sigma^2 = 1/2^2) + \frac{1}{2}\mathcal{N}(\mu = 1, \sigma^2 = 1/2^2)$. The distribution is plotted in Figure 1.

The distribution was chosen because for most parameterizations, it has a non-gaussian posterior. It was one-dimensional in order to be able to plot the fitted density and approximated posterior, does not require GPU resources to fit, that the true entropy $\text{E}[\log p(X)]$ can computed with arbitrary precision. The model used a one-dimensional latent stochastic variable z, and two deterministic layers with 5 units each. All non-linearities were tanh, and the network was used to fit $\mu(z), \log \sigma(z)$.

$X \sim p(X)$ can be generated as $p(X|Z) = \mathcal{N}(X \mid \mu = \texttt{sgn}(Z), \sigma^2 = 1/2^2)$, the posterior for x slightly above 0 will look as the two truncated or folded Gaussians below:



This is highly non-Guassian. We therefore expected VAE to struggle with fitting this model. X can also be generated as $X = F_x^{-1}(\Phi(Z))$ deterministically with Φ being the CDF of $\mathcal{N}(0, 1)$, and F_x the CDF of X. Then both $\text{Var}[X \mid Z] = \text{Var}[Z \mid X] = 0$ which are both Gaussian with zero variance.

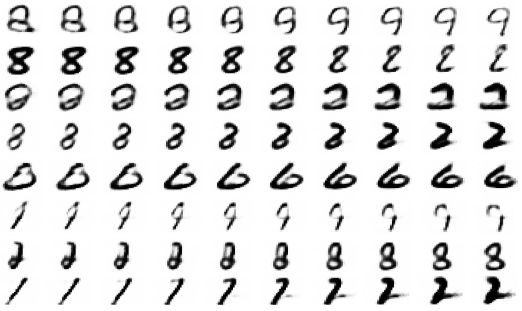


Figure 2: Interpolation in Z-space

4 Results

Visually, it's difficult to discern differences between the VAE, IWAE and different choices of K. Figure 2 shows $\mathbf{X}$ sampled from the VAE model with $K = 50$ in (4). For each row, $\mathbf{Z}_1, \mathbf{Z}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are sampled independently. Every cell in the row is generated by interpolating between $\mathbf{Z}_1$ and $\mathbf{Z}_2$. We do not draw X_i from the Bernoulli distribution. Instead we plot $\text{E}[X_i]$ in gray scale.

We trained with architectures described in section 3 and obtained the following results table. It is structured the same way as **Table 1** in [BGS]. The table shows the attained average negative log-likelihood at the end of training, and also the number of training epochs each model did run for. Our ambition was to run for the full 3280 epochs as our main source, but due to time and GPU constraints our models trained for 400 to 1600 epochs.

		MNIST			
		VAE		IWAE	
# stochastic layers	k	NLL	active units	NLL	active units
1	1	87.5	15	87.5	15
	5	88.3†	14	87.4‡	20
	50	88.7†	13	85.9†	24
2	1	99.7‡	7+0	99.7‡	7+0
	5	88.5	13+1	87.5†	17+1
	50	88.3	15+1	85.4‡	31+3

Table 1: Reproduction attempt of Table 1 of [BGS]. No symbol = trained for 1600 epochs, † = 800 epochs, and ‡ = 400 epochs

We found that $\text{Var}[\text{E}[Z_i \mid \mathbf{X}]]$ was distributed similar to [BGS]. Below is a histogram of a model with 11 active units out of 50.

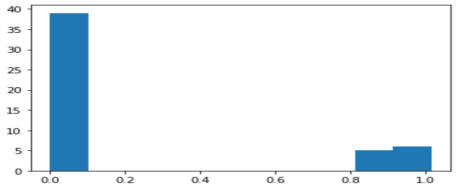


Figure 3: Histogram of Var[E[$Z_i \mid \mathbf{X}$]] for our VAE MNIST model

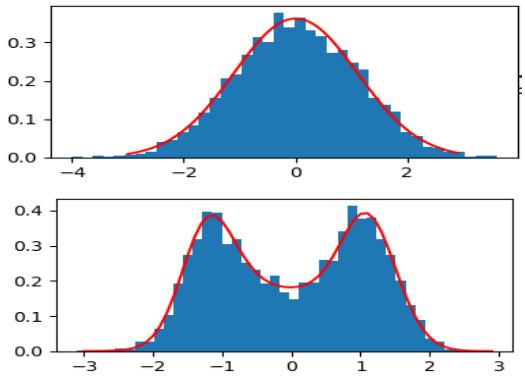


Figure 4: VAE (first) and IWAE model (second) for the camel data set

4.1 One-dimensional camel dataset

Training the VAE on the camel-hump-distributed data produced the first distribution in Figure 4 for $\mathbf{X}$. The distribution is computed as $p(x) = \int p(z)\mathcal{N}(\mu(z \mid x), \sigma^2(z \mid x))dz$. Increasing the synthetic dataset to 10000 samples did not change the the result visibly. Training the model for 3 epochs on 1000 data points as an IWAE under $\mathcal{L}_{20}$ produced the second distribution of Figure 4. The true entropy can be computed by numerical integration to $\mathrm{E}[\ln p(X)] = -1.358$. The average $\mathcal{L}_{5000}$ ELBO of the IWAE model on a large test set is estimated to -1.37. For the VAE model that fails to fit the two humps, the average $\mathcal{L}_{5000}$ is -1.52.

5 Discussion

Weight initialization

We discovered that one has to be careful in the initialization to avoid numerical issues when the output decoder layer is Gaussian. In [BGS], every linear layer weight is initialized as Unif$(-a, a)$, where a is $\sqrt{\frac{6}{\text{n\_in}+\text{n\_out}}}$, and n_in, n_out are the input and output size of that layer. We found that this initialization can be numerically unstable for a Gaussian X at the start of training. The following situation can lead to underflows: for a large x, $\sigma_{enc}(x) = \texttt{exp}(\text{nn}(x))$ can be large, and the sampled z can be even large larger. $\sigma_{dec}(z) = \texttt{exp}(\text{nn'}(z))$ can be very small, and therefore $\log p(x \mid z)$ may underflow. We changed the initialization to start with constant σ^2, which solved issues with Gaussian $\mathbf{x}$, but made training get stuck for Bernoulli $\mathbf{x}$.

Binarization and NLL

The log-likelihoods in 1 differ substantially from Table 1 in [BGS]. We investigated the possibility that our negative log likelihood computation differed from that of 1. The model assumes pixels are binary, but MNIST contains grayscale values. Therefore, results depend on the method of binarization. Both we and [BGS] used a grayscale value p of a pixel to independently draw $X \sim \text{Bernoulli}(p)$. A new binary image was generated from the grayscale image at every epoch. An alternative method is to set a cut off value, e.g. $X = \mathbb{1}(p \geq 1/2)$. We found that this method gives higher likelihood.

We saw that [Bur] detects inactive units, and somehow switches them off, disregarding their influence, before estimating the NLL. How it was done was not described in [BGS], and we were not able to implement this method. We would expect that the inactive Z_i should be frozen at 0 or at E[E[$Z_i \mid \mathbf{X}$]], the prior for Z_i be 1, however [Bur] does something different.

Apart from disregarding inactive units, we believe that our NLL computation is identical to that of [BGS]. All differences are due to reproduction failure, or differences in training, model setup, model initialization or shorter training time.

Reproduction of results

As seen in table 1, we reproduced the basic trends of [BGS]. Our main reproduction failure is that we did not achieve better results with 2 layer VAE than with 1 layer VAE. Also, the results of our main source were several nats better across all models. The number of active units on the 2nd stochastic layer was typically stuck at 1 or 2 during the first 100s of epochs for both VAE and IWAE. We tested our implementation for simple low-dimensional data sets, and found that models with 2 stochastic layer could be trained successfully. We do believe that our 2-layer implementation is correct. However, we are uncertain whether our training rate decay rates are optimally tuned for 2 stochastic layers.

We believe that this work reproduced most of the results of [BGS]. Our likelihood metrics are lower than [BGS]. We believe that it is because we were not able to keep training for as long as [BGS], possibly because we did not eliminate the effect of non-active units. We believe that our network initialization, training schedule and batch size matches those of [BGS]. In particular, the IWAE scores are clearly better than the VAE scores. A difference from [BGS] is that the VAE networks are not able to benefit from 2 stochastic layers. However, the IWAE networks do perform better with 2 layers compared to single-layer networks. As expected, the performance of IWAE with $K = 5$ is between VAE and IWAE with $K = 50$. Models with better performance have more active units than worse models. We did not finish our reproduction of the results with the Omniglot data set; therefore they are not

presented.

Camel dataset

The training target is expected to be a little higher (although we cannot have significant overfitting, because the number of training samples is orders of magnitude above the number of parameters). The estimated expected log-likelihood is as expected a little lower than the true value; it can be seen from the distribution that the right hump is a little higher, and density around 0 is higher than in the true distribution.

6 Conclusion

We believe that we have conclusively reproduced the following results from [BGS]: IWAE reaches better results than VAE on all MNIST models, as measured by NLL. We believe that this result generalizes across different models, and that IWAE is better at utilizing their network capabilities by learning more expressive latent representations which leads to better results in log-likelihood measurements. We found that training is highly sensitive to weight initialization, despite the fact that our neural networks had only two nonlinear layers. We reproduced the claim that model performance on the same model architecture is strongly correlated with the number of active units. Our findings for $K = 1, 5, 50, 5000$ in IWAE support the theoretic claim that the K interpolates between VAE loss and true negative log-likelihood. We found that in all tested cases results for IWAE improve with K.

We would like to see whether these results hold for other data sets. We would also like to perform further experiments with different batch sizes, initializations and learning rate schedules. We are interested in whether convolutional networks would improve the NLL scores.

Because of the small batch size, we learned that training this model on GPU is very inefficient. In our implementation, the bottleneck seems to be CPU/GPU sync, and the GPU is heavily under-utilized. Increasing the batch size decreases training time by a large factor, but in doing so it risks distorting the loss landscape [Cha]. But since the images are re-binarized every epoch, we think that the end result found with e.g. batch size of 512 should not differ much from the results in Table 1 found by batch size 20. We would like to investigate that, and to see whether we can increase the batch size gradually during the training as suggested in [SKYL18].

References

[BGS] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *ICLR 2016*.

[Bur] Yuri Burda. Iwae implementation by [BGS] authors. `https://github.com/yburda/iwae/`.

[Cha] D. Chang. Effect of batch size on training dynamics. *https://medium.com*.

[EWA07] Salwa El-Wakil and M. Abdou. New exact traveling wave solutions using modified extended tanh function method. 31:840–852, 2007.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 2010.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[KW14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[KW19] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12:307–392, 2019.

[Lan21] Micheal Lanham. *Generating a New Reality: From Autoencoders and Adversarial Networks to Deepfakes*. Apress, 2021.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[LROI21] D. K. Lim, N. U. Rashid, J. B. Oliva, and J. G. Ibrahim. Handling non-ignorably missing features in electronic health records data using importance-weighted autoencoders, 2021.

[LX18] Yuchen Lu and Peng Xu. Anomaly detection for skin disease images using variational autoencoder, 2018.

[SKYL18] S. L. Smith, P.J. Kindermans, C. Ying, and Q. V. Le. Don't decay the learning rate, increase the batch size, arxiv, 2018.

[TLGM18] G. Tucker, D. Lawson, S. Gu, and C. J. Maddison. Doubly reparameterized gradient estimators for monte carlo objectives, 2018.

[UB21] C. J. Urban and D. J. Bauer. A deep learning algorithm for high-dimensional exploratory item factor analysis, 2021.