Machine Learning, Advanced Course: Large Project - Importance Weighted Autoencoders

Group 5:
Alejandro Garcia Castellanos, algc@kth.se
Altan Senel, altans@kth.se
Christos Frantzolas, frant@kth.se
Fehmi Ayberk Uçkun, uckun@kth.se

January 9th, 2022

Abstract

Variational Autoencoders are generative models based on artificial neural networks which encode the dataset on which they are trained on using a set of latent variables. Importance Weighted Autoencoders build on VAEs by introducing an improved training objective that approximates the evidence lower bound more closely. In this project, we aim to give a general overview of how VAEs and IWAEs work, implement the two models using the Keras API and try to reproduce some of the results of the original IWAE paper by Burda et al.

1 Introduction

1.1 Variational Autoencoders.

Variational Autoencoders (or VAEs for short) are deep learning generative models, first introduced by [1]. The VAE combines a generative network (decoder) and a recognition network (encoder) into the same model. The main advantage of VAE over classical Autoencoders(AE) is that they ensure the latent space in which inputs are encoded and used for generation is continuous and easy to interpolate. Similar to Variational Inference, we assume that our dataset X is a sampled distribution generated through a random process that involves a latent variable z. The problem that often arises is that the marginal likelihood of our data X, or the posterior pdf $p_\theta(z|x)$ are intractable, i.e. impossible to compute or differentiate. Kingma and Welling corrected this issue by introducing a recognition model $q_\phi(z|x)$ which tries to approximate the posterior $p_\theta(z|x)$ (thus "encoding" the data x using a distribution over z), as well as a generative model $p_\theta(x|z)$, which is referred to as a probabilistic decoder.

The technique introduced in [1] manages to compute the Evidence Lower Bound (ELBO) by employing a reparameterization trick that allows backpropagation in neural networks, where an auxiliary noise variable ε is used to allow for a differentiable transformation to take the place of the probabilistic encoder. Thus, the authors proposed a Stochastic Gradient Variational Bayes ELBO estimator:

$$\hat{\mathscr{L}}^B(\theta,\phi,x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L}(log p_\theta(x^{(i)}|z^{(i,l)})) \tag{1}$$

where D_{KL} is the Kullback-Leibler divergence, which can be considered as a regularization of ϕ, so that the estimated posterior can be as close as possible to the prior of z.

In the case of VAEs, the prior over z is a normal multivariate Gaussian distribution, while the generative model's distribution is either a multivariate Gaussian or a Bernoulli distribution. The true posterior $q_\phi(z|x)$ is assumed to be approximately Gaussian with diagonal covariance.

1.2 Importance Weighted Autoencoders.

One issue with the way that VAEs work is that they make a lot of assumptions about the form of the posterior distribution. These assumptions can limit the expressive power of the network. The Importance Weighted Autoencoder (IWAE) was introduced by Burda et al. [2] to address this problem. Authors claim that training a recognition network which only places 20% of its samples to the VAE constrained region can still perform accurate inference. The IWAE shares the same architecture as the VAE, but its ELBO is tighter thanks to importance weighting. By acquiring k samples of the estimated posterior $q(h|x)$ from the recognition model, the following lower bound is used:

$$\mathscr{L}_k(x) = \mathbb{E}_{h_1,\ldots,h_k\ q(h|x)}[log\frac{1}{k}\sum_{i=1}^{k}\frac{p(x,h_i)}{q(h_i|x)}] \tag{2}$$

The term $\frac{p(x,h_i)}{q(h_i|x)}$ is the unnormalized importance weight. In fact, the authors proved in [2, Theorem 1] that for all k, the ELBOs satisfy:

$$log p(x) \geq \mathscr{L}_{k+1} \geq \mathscr{L}_k \tag{3}$$

The Monte Carlo estimator of the gradient of the ELBO is calculated by drawing k samples from the recognition network and calculating the weights specified in Eq. 2. When k=1 the IWAE reduces to a

standard VAE. The weighting in IWAEs gives more weight to samples that have a high probability of being reconstructed by the decoder, given the latent representation, and at the same time have a latent representation that is unlikely to be produced by the encoder, given the corresponding sample. This encourages a more "spread-out" distribution, that includes more of the latent space thus more of the possible encodings.

1.3 Project objective.

In this project, we aim to check whether the results reported by the authors of [2] are reproducible by implementing the VAE and IWAE models described as in [2]. We try to reproduce the paper's results with experiments on the MNIST dataset, by training different model architectures with different values of k (for sample importance weighting) and comparing the resulting negative log-likelihood (NLL) on the testing dataset with the values provided by Burda et al. We also produce the visualizations of the latent space and the reconstructed images to better evaluate the results.

2 Methods

2.1 Implementation.

For the implementation of the VAE and IWAE models specified in the paper, we used Keras [3], a deep learning Python API for the machine learning platform TensorFlow. Some of the functions used in our code are also directly part of TensorFlow and the current latest version of v2.7.0 is used. Our experiments were conducted on the MNIST dataset [4] downloaded via Keras API, using the platform Google Colab for GPU acceleration and Python 3.6.9.

The model architecture closely follows the one presented in [2]. There are 2 different architectures presented in the paper:

- Building the model using 1 stochastic layer of 50 units in the encoder, with 2 deterministic layers of 200 units.
- Using 2 stochastic layers, with 100 and 50 units respectively.

We also used the same parameters for the Adam optimizer, as well as the same learning rate for training. One significant deviation from the architecture specified in the paper is the one concerning the topology of the stochastic layers. By studying the authors' implementation of the model, we observed that the output of the first stochastic layer of the decoder is not used to generate the output of the VAE. Despite Appendix C of [2] specifying a sequential ordering of the latent variable layers, we followed the architecture as it was implemented in code by the authors.

In order to evaluate the performance of our models on the testing datasets, we calculate the negative log-likelihood of the testing data, as specified in Eq. 8 of the IWAE paper. In order to stably calculate the $\mathscr{L}_{5000}$ as an estimate of the log-likelihood, the Log-Sum-Exp trick is implemented to avoid overflow issues.

3 Results

Due to time constraints we limit the training to 1000 epochs instead of 3280 epochs that were used in the original paper [2]. Furthermore, we limit our results to the MNIST data set due to the same time constraints.

3.1 Experiments with number of samples and latent layers.

In the original paper, the authors compare two main points, how the number of hidden layers and the number of samples used for importance weighting (k) affect the results. From [2, Theorem 1] we know that the higher the number of samples the better fit we should obtain to the real NLL, therefore our results should follow the theorem.

Additionally, we will reproduce the study of the number of active units in the resulting networks. It is said in [2] that VAE tends to not use all the dimensions (units) of the latent layers. Therefore, we will study whether IWAE increases the number of active units. We will say that a unit is active if $A_u = \mathrm{Cov}_x(\mathbb{E}_{u\sim q(u|x)}[u]) > 10^{-2}$, i.e. a unit is active if the mean of its corresponding distribution changes depending on the observation [2].

		VAE		IWAE	
# stoch. layers	k	NLL	units	NLL	units
1	1	91.87	20	91.96	20
	5	88.64	20	89.57	23
	25	87.83	20	87.93	24
	50	87.45	20	87.28	25
2	1	90.64	18 + 4	90.36	18 + 4
	5	86.87	19 + 8	87.74	22 + 7
	50	85.44	19 + 9	85.60	25 + 7

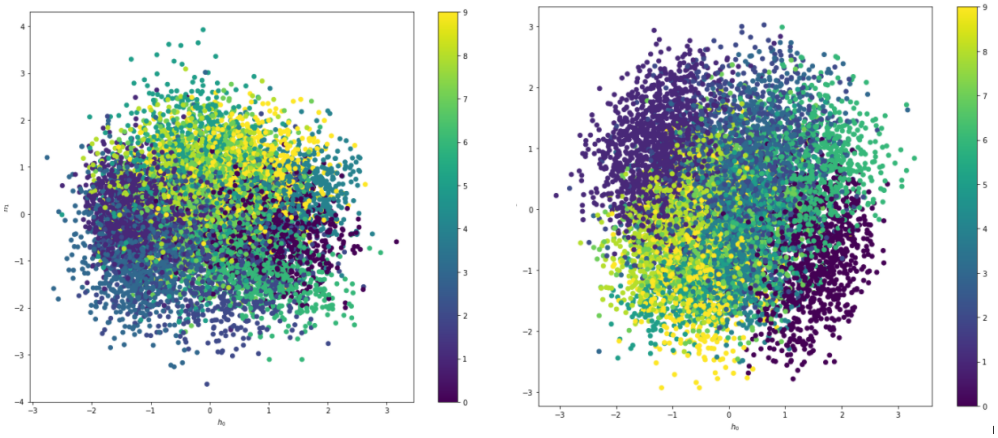
Table 1: Experiment results on MNIST dataset. For models with two stochastic layers, $x+y$ shows the active layers for the first and second layer respectively.

In Table 1 we can see the resulting NLL and active units for VAE and IWAE with different numbers of samples and for both of the model architectures over the test set of the MNIST dataset. We added an extra case of $k = 25$ for one stochastic layered models to more precisely see the effect of the number of samples. Due to time limitations, this case was not added for two stochastic layered models.

3.2 Experiment with latent layer dimension

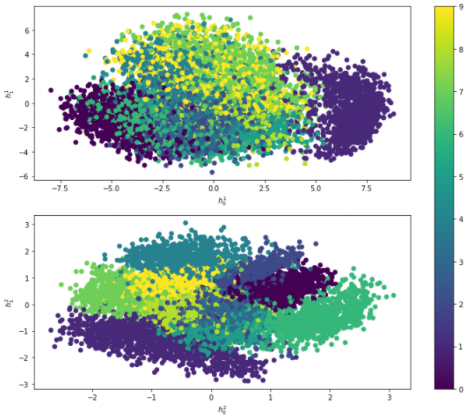
Here we trained a VAE model with one latent layer and 50 samples, but in this case, we set the dimension of the latent layer to 2 instead of 50 as mentioned in the paper. NLL turned out to be 130.42 for this case, hence can see the negative impact of having a low dimensional latent layer on the performance of the model. Both of the units in the latent layer were active.

3.3 Visualizing the results.

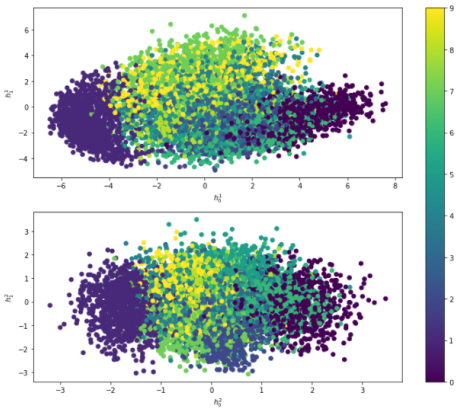


(a) VAE-1, k=50

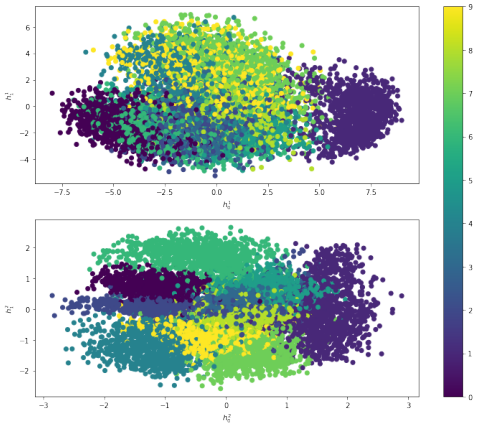
(b) IWAE-1, k=50



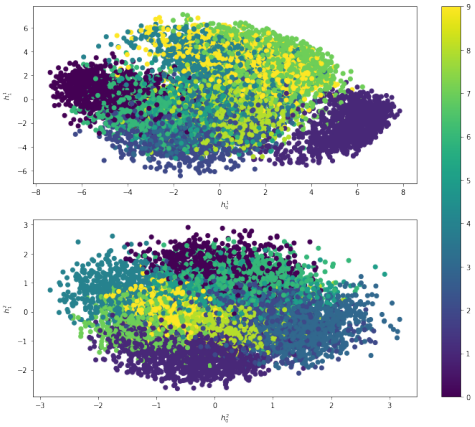
(c) VAE-2, k=1



(d) VAE-2, k=50



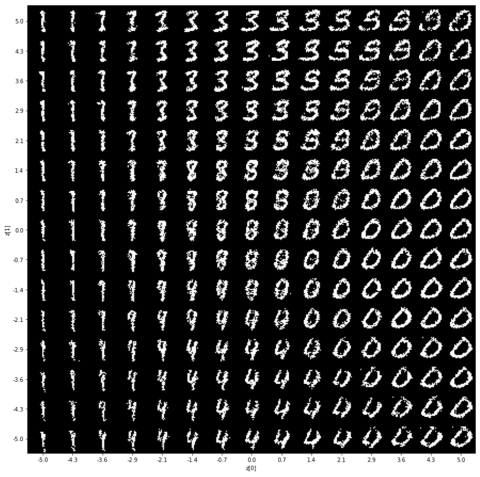
(e) IWAE-2, k=1



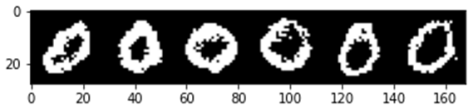
(f) IWAE-2, k=50

Figure 1: Latent layer representations after PCA for 2 dimensions. Colors are representing the labels of the images. For the models with two stochastic layers, both latent representations are given.

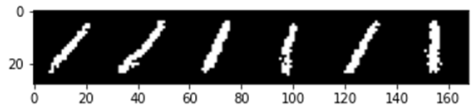
After training the model, the output of the encoder for the test data is visualized in Figure 1. To visualize the latent space of 50 dimensions, we used PCA and reduced the dimension to 2. The reason why directly using 2-dimensional latent space is not optimal is showcased in the previous section.



(a) Latent space visualization with numbers.



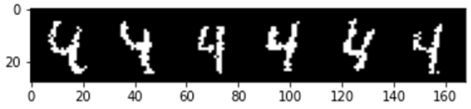
(b) Label = 0



(c) Label = 1

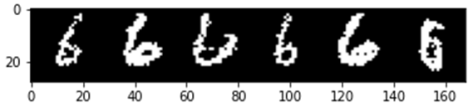
(d) Label = 2

(e) Label = 3

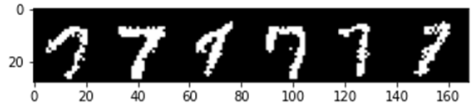


(f) Label = 4

(g) Label = 5

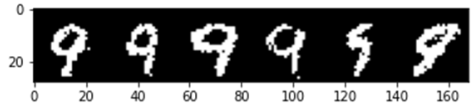


(h) Label = 6



(i) Label = 7

(j) Label = 8



(k) Label = 9

Figure 2: Reconstructions of the test images with IWAE-1 k=50.

In Figure 2, we show the reconstructions of test images. By checking the latent space distribution of the numbers we can understand the mistakes our decoder makes like putting an "8" instead of "3" and putting a "5" instead of "9".

4 Discussion

We can see in Table 1 that we get higher NLL than the ones presented on the original paper. Furthermore, we observe that in our results there are some cases where VAE obtains a lower NLL than IWAE, which does not happen in the original paper. We believe that these two issues come from the fact that we trained the models for only 1000 epochs. In this situation, we cannot confirm that IWAE achieves better results than VAE within the 1000 epoch limit using the MNIST dataset however results are comparable. In the original paper, the highest reported difference between the likelihoods of VAE and IWAE is 1.88 for the MNIST dataset. Such a small difference can be explained via best case scenario reporting or via that IWAE needs more training.

However, we can see that the results satisfy [2, Theorem 1] since for all cases the NLL decreases as we increase the number of samples used. In addition, we observe that IWAE uses more of the hidden units than VAE, so even if we observe that VAE has a lower NLL in our results we can argue that IWAE obtains a richer latent representation as the authors argued.

Furthermore, we observed that if we only use a two-dimensional latent layer the model does not have enough expressiveness to capture all the relevant data characteristics. Therefore, we believe that if we use less than the active units that we have obtained in Table 1, then we might get worse results.

Finally, we can validate more our hypothesis that IWAEs learn a richer latent representation than VAEs seen in the results in Figure 1. Here, we observe that IWAE tends to generate more distinct areas for each of the digits. Also, we observe that for the case of two latent layers the second layer has a simpler representation of the latent space than the first one and that the second layer for $k = 1$ is simpler than the one produced with $k = 50$. The reason for the latter statement is that we have four active units in the case of $k = 1$, and 9 and 7 active units for VAE and IWAE respectively in the case of $k = 50$. Therefore, we would expect that the PCA technique that we are using to obtain the results would have an easier job to encode a 4-dimensional space into a 2 two dimensional space rather than the 9 and 7-dimensional spaces, which could explain the apparent simpler representations in the case of $k = 1$.

4.1 State-of-the-art work built on IWAEs.

Variants of this lower bound presented in [2] are utilized in recent papers to derive tighter lower bounds by making use of bias reduction and variance reduction techniques. Sebastian Nowozin summarizes Jackknife and generalized Jackknife used for reducing the biases of estimators [5]. In the paper, the estimator is evaluated on n samples and is denoted as $\hat{T}_n$. When an estimator is consistent and a smooth function of linear statistics, Jackknife bias-correction achieves the following:

$$\mathbb{E}[n\hat{T}_n - (n-1)\hat{T}_{n-1}] = T + O(n^{-2}) \tag{4}$$

therefore the Jackknife bias-corrected estimator reduces the bias by $O(n^{-2})$.

Generalized jackknife introduced by Schucany [6], that makes use of estimators $\hat{T}_G$ of order m, defined as:

$$\hat{T}_G^{(m)} = \sum_{j=0}^{m} c(n,m,j)\hat{T}_{n-j} \tag{5}$$

where $c(n,m,j)$ are Sharot coefficients.The generalized version achieves a bias order of $O(m^{-(j+1)})$ and is used to reduce the bias of the IWAE estimator in the paper.

Harshil, David, and Aleksandar propose overdispersed variational autoencoders OVAE and OIWAE [7], which are variants of VAE and IWAE. Rather than drawing samples from the variational distribution; their scheme draws samples from an overdispersed (i.e. heavier-tailed) proposal that is in the same family as the variational distribution. Experimenting on two different datasets, they show in both cases their techniques provide a lower variance grading estimate than VAE and IWAE; and reach a higher log-likelihood lower bound.

On the other hand, the study of Pierre-Alexandre and Jes investigate variance reduction techniques that can be utilized to derive tighter bounds [8]. Their paper mentions a branch of variance reduction techniques based on negative dependence. It is argued that the lower bound gets tighter when the weights get more negatively dependent. Another study builds on these ideas and introduces a joint sampling scheme with a hierarchical structure, and their analysis shows that learning dependency among the joint samples can induce negative correlation and reduce the variance of the estimator [9].

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[2] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[3] Francois Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[4] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[5] Sebastian Nowozin. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. 2018.

[6] WR Schucany, HL Gray, and DB Owen. On bias reduction in estimation. *Journal of the American Statistical Association*, 66(335):524–533, 1971.

[7] Harshil Shah, David Barber, and Aleksandar Botev. Overdispersed variational autoencoders. pages 1109–1116, 2017.

[8] Pierre-Alexandre Mattei and Jes Frellsen. Negative dependence tightens variational bounds. 2020.

[9] Chin-Wei Huang, Kris Sankaran, Eeshan Dhekane, Alexandre Lacoste, and Aaron Courville. Hierarchical importance weighted autoencoders. pages 2869–2878, 2019.