Large-VAE Project
Advanced Machine Learning DD2434

Eirini Stratigi, Georgios Moschovis, Ioannis Athanasiadis
Department of Electrical and Computer Engineering
KTH Royal Institute of Technology
Stockholm, SE 11428
{stratigi,geomos,iath}@kth.se

Abstract

Autoencoders are vastly being used to learn an efficient mapping between high dimensional data and their low dimensional latent representations. Although they display promising capabilities at compressing the input and thereafter decompressing the latent representations back into their original form, the generated latent representations lie in an undefined space hence the generation of novel samples is unfeasible. The variational autoencoders enhance the autoencoders by regularizing the generated latent representations. More specifically, the variational autoencoder architecture consists of an encoder and a decoder while it is trained to minimize the Reconstruction Error as well as to encode its inputs as a distribution over a latent space $\mathbf{z}$. Furthermore, a variational autoencoder is regularized and when we sample from the latent space, the generative decoder may produce data similar to the model's original inputs. In the context of this project, we follow the study presented in [1] in which a new form of prior is proposed after reformulating the variational lower bound and focuses on the minimization of the cross-entropy between the variational posterior and the prior. The aggregated posterior, which solves this minimization problem, is considered as the form of the new prior, conditioned to a number of learnable pseudo-inputs, which are trained by back-propagation.

1 Introduction

Autoencoder is a category of Neural Networks that is trained with the purpose of copying its inputs to its corresponding outputs. More specifically, there are several hidden layers $\boldsymbol{h}$ that describe a code used to represent the input. The model as presented in paper [1], employs an encoder function $\boldsymbol{h} = f(\boldsymbol{x})$, a decoder $\boldsymbol{r} = g(\boldsymbol{h})$ as well as a loss function. The encoder and the decoder are trained to compress and decompress, their inputs in a cooperative manner. Build upon this, the network grasps the relevant aspects of the inputs, encloses them into their latent representations and thereafter decomposes the latter back to their initial structure. Additionally, autoencoders are basically neural networks and thus inherit their properties. Based on that the autoencoders are trained by minimizing their loss using the feed forward and back-propagation approach.

Although Autoencoders have been proved to be quite effective at encoding high-dimensional data into lower-dimensional latent representations and then reconstructing the latter back to their initial form, they do not possess any generative capabilities. On the other hand, Variational Autoencoders (VAE) are able to generate convincing samples, similar to those that they have been trained on, by imposing a prior distribution on the latent representation space. More specifically, a VAE aims at learning the parameters that describe the probability distribution representing the input data rather than strictly mapping the high-dimensional inputs into their corresponding latent forms. Having ensured that the latent representations follow a specific probability distribution, the generative process is feasible by randomly sampling points from that distribution and feeding them to the decoder. Similarly to the Autoencoders, VAEs are trained in such way to both minimize the reconstruction error, while also regularizing the encoder's outputs to respect the prior distribution that we have declared.

From the aforementioned information, it is clear that VAEs enhance the autoencoders potentials. However, further improvement can be achieved in the design of VAE by selecting a more complex form for the prior, in the case we use a neural network for the probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ (the approximation to the posterior of the generative model $p_\theta(\mathbf{x}|\mathbf{z})$) and the generative decoder $p_\theta(\mathbf{x}|\mathbf{z})$, as it plays an important role in mediating between those two. Let the prior over the latent variables be the centered isotropic multivariate Gaussian $p_\lambda(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$. Note that in this case, the prior lacks parameters, which could lead to over-regularization that implies under-fitting and thus very poor hidden representations[2],[3]. To resolve these issues, [1] proposes a new prior that is a *variational mixture of posteriors prior* or shortly VampPrior, which takes the form of the aggregated posterior applied to a number of trainable pseudo-inputs.

In general, the study presented in [1] states that too simplistic priors tend to over-regularize the model and consequently limit its potentials. Additionally it suggests that the choice of prior heavily influences the performance of VAEs and thus proposes a novel approach for learning psudo-inputs to guide the construction of a more complex prior, termed as VampPrior, which is in fact an approximation of the aggregated posterior p^*_λ. Moreover, the authors suggest that VampPrior addresses the problem of inactive latent dimensions by utilizing a more potent latent representation for the variational posterior.

Build upon these, we aim at reproducing the results and conclusions presented in [1] by conducting experiments using the standard prior and the VampPrior methods, where the prior takes the form of the centered isotropic multivariate Gaussian $p_\lambda(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ and VampPrior respectively, and prove that the likelihood is being improved with the use of the VampPrior, compared to the standard method as well as that VampPrior generates more convincing samples. There are several datasets included in [1], that are used for this implementation, among which we have chosen the MNIST and the Frey Faces for reproduction purposes. We have chosen these two datasets to evaluate the performance of VampPrior in datasets of varying difficulty.

2 Methods

In order to investigate how the proposed VampPrior influences the training process and consequently the generative capabilities of VAEs, we implemented two identical neural network architectures in which we enforced the latent representations into either following the standard or a *variational mixture of posteriors* prior. Additionally, we also experimented with different numbers of latent parameters' dimensions, ranging from 10 to 100, with the purpose of investigating their effect. In the

following sections we describe the different components of our implementation as well as provide its technical details.

2.1 Encoder/Decoder

As we stated previously, VAEs treat each D-dimensional input to be composed of D-dimensional independent probability distributions. Additionally, it is regarded that these D-dimensional inputs can be compressed into M-dimensional entities of lower dimension, which are also specified by M-dimensional independent distributions. The encoder network is responsible for transforming these D-dimensional inputs into their latent representations while the decoder network is trained to unpack these representations back to their corresponding original inputs. More specifically the encoder, termed as the variational posterior probability $q_\phi(\mathbf{z}|\mathbf{x})$, is assumed to follow the distribution $\mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \mathrm{diag}(\sigma^{\mathbf{2}}_\phi(\mathbf{x})))$ where the probability $p_\lambda(\mathbf{z})$ is the prior distribution we enforce, either $p_\lambda(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$, or $p_\lambda(\mathbf{z}) = 1/K \times \sum_{i=1}^{K} q_\phi(\mathbf{z}|\mathbf{u_k})$, with $\mathbf{u_k}$ being VampPrior's trainable pseudo-inputs, while $\mu_\phi(\mathbf{x})$ and $\sigma^{\mathbf{2}}_\phi(\mathbf{x})$ are the means and the variances of the M latent variables, which the neural network parameterizes. Based on the parameters predicted by the encoder we utilize the reparametarization trick to sample a random $\mathbf{z}_\phi$.

Thereafter we feed the sampled $\mathbf{z}_\phi$ into the decoder $p_\theta(\mathbf{x}|\mathbf{z}_\phi)$ which outputs the parameters required by the distribution we assumed for the D-dimensional inputs, based on the dataset structure. More specifically, we considered each D-dimensions of the input to have discretized logistic distribution and Bernouli distribution for Frey Faces and MNIST respectively. Similarly to the encoder, the decoder aims at predicting the parameters of the distributions giving raise to the D-dimensional input data. We have chosen to model the encoder and decoder networks using identical architectures consisted of two gated layers of 300 units size, as proposed in [4]. More specifically these gated layers operate in a self attention manner in which we pass the inputs into two linear layers $\boldsymbol{h}$ and $\boldsymbol{g}$, then apply the sigmoid activation function onto the output of $\boldsymbol{g}$, turning it into mask, and finally returning their element wise product between the gate mask and the output of $\boldsymbol{h}$.

2.2 Loss function

Consider a dataset $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^N$ where we suppose $\mathbf{x} \in \mathbb{R}^D$ is a vector of D observable variables and $\mathbf{z} \in \mathbb{R}^M$ a vector of stochastic latent variables. Our goal is to maximize the marginal log-likelihood $\log p(\mathbf{x})$ and since the probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and the generative decoder $p_\theta(\mathbf{x}|\mathbf{z})$ are neural networks with weights ϕ and θ respectively, we may use Variational Inference in order to overcome the problem of $\log p(\mathbf{x})$ being intractable. Consequently, we ought to optimize the *variational lower bound* objective given using Jensen inequality and logarithm properties:

$$\begin{aligned}
\log p(\mathbf{x}) &= \int p_\theta(\mathbf{x}|\mathbf{z}) p_\lambda(\mathbf{z})\, d\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} \\
&= \int -q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\lambda(\mathbf{z})} + q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z})\, d\mathbf{z} \\
&= -KL[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})] + \mathbb{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]
\end{aligned}$$

Furthermore, concerning the expectation over $q_\phi(\mathbf{x}|\mathbf{z})$, we consider a Monte Carlo estimate using L sample points. Under this assumption, we need to reformulate the above lower bound and starting from the Kullback-Leibler divergence, it can be approximated via Monte-Carlo (MC) estimation by $D_{MC,L}$, which correctly approximates the $D_{KL} \triangleq KL[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]$ according to the law of large numbers as L tends to infinity[5]:

$$\begin{aligned}
-D_{KL} \triangleq -KL[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})] &= \int -q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\lambda(\mathbf{z})}\, d\mathbf{z} \\
&= \int \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[-\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\lambda(\mathbf{z})} \right] d\mathbf{z}
\end{aligned}$$

$$
\begin{aligned}
-D_{MC,L} &= \frac{1}{L}\int -\log\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\lambda(\mathbf{z})}\,d\mathbf{z} \\
&= \frac{1}{L}\int -\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\lambda(\mathbf{z})\,d\mathbf{z} \\
&= \frac{1}{L}\int -\log \mathcal{N}(\mathbf{z};\mu,\sigma^2) + \log \mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})\,d\mathbf{z} \\
&= \frac{1}{2L}\sum_{i=1}^{L}\log(\sigma_i^2) + \frac{(z_\phi^{(i)}-\mu)^2}{\sigma_i^2} - (z_\phi^{(i)})^2
\end{aligned}
$$

To compute $-D_{MC,L}$ that approximates the dissimilarity between the the generative decoder $p_\theta(\mathbf{x}|\mathbf{z})$ and the prior $p_\lambda(\mathbf{z})$, we are sampling $z_\phi^{(i)}$ from $q_\phi(\mathbf{z}|\mathbf{x})$ through the reparameterization trick. As the probabilistic encoder attempts to learn the parameters of the distribution in the latent space, through adjusting its weights ϕ using back-propagation, in order to make this possible regardless of the random sampling that occurs halfway of the architecture, the trick is based on the fact that if z is a random variable following a Gaussian distribution with mean μ_ϕ and with covariance $\Sigma_\phi, \mathcal{N}(\mu_\phi, \Sigma_\phi)$, then it can be expressed as $z = \mu_\phi\zeta + \Sigma_\phi$, where ζ is sampled from the standard multivariate Gaussian $\zeta \sim \mathcal{N}(\mathbf{z}|\mathbf{0},\mathbf{I})$. This reparameterization trick stops sampling from preventing backpropagation and therefore learning.

Except for the case when we are sampling from the probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$, we also apply the reparameterization trick while generating samples from the VampPrior $p_\lambda(\mathbf{z})$. That is, when we generate new data from the non-linear trainable layer that represents $p_\lambda(\mathbf{z})$, between passing it through the respective layers of the probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and the generative decoder $p_\theta(\mathbf{x}|\mathbf{z})$. If $\{\mathbf{x}_{\text{sample}}^{(i)}\}_{i=1}^{\Delta}$ contains Δ produced generations by the VampPrior, and $(\mathbf{z}, \Sigma_{\lambda,\phi})$ are the parameters obtained by the probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ then we similarly perform the reparameterization trick as $\hat{\mathbf{z}} = \mathbf{z} \times \zeta + \Sigma_{\lambda,\phi}$, where once again $\zeta \sim \mathcal{N}(\mathbf{z}|\mathbf{0},\mathbf{I})$ and then provide $\hat{\mathbf{z}}$ as input to the decoder to produce the reproductions.

In the case of reconstruction error the equation that we have to resolve is $\mathbb{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$. In order to calculate this mean we have to distinguish two different cases for each dataset, MNIST and Frey Faces respectively. In the case of MNIST all we have to do is to calculate the mean thought Monte Carlo estimation. The distribution that is used for this case is the Bernoulli and the log of the Reconstruction Error can be described as:

$$
\begin{aligned}
R_{loss} &= \sum_x x\log(\mathbf{y}) + (1-x)\log(1-\mathbf{y}) \\
&= RE,
\end{aligned}
$$

where x are the data points of the dataset and $\mathbf{y}$ the outputs of the generative decoder.

In case of Frey Faces dataset, our goal was to generate images by taking into account the pixel values between the range 0 to 255. The problem with this approach is that we need to calculate also the mathematics between two continuous pixels like 220 and 221 and the distribution that is used to describe the Reconstruction Loss is the Discretized Logistic Mixture Likelihood[6]. Having in mind that the CDF of the standard logistic distribution is nothing else than the sigmoid function itself, we eventually can compute the probability per pixel as follows:

$$
p(x_i|\mu_i, s_i) = \text{CDF}\left(x_i + \frac{1}{256}\middle|\mu_i, s_i\right) - \text{CDF}\left(x_i|\mu_i, s_i\right),
$$

where the μ_i is the output of the decoder and $\log s_i$ which is a logarithmic scaling factor. Following this formulation we calculate the R_{loss} as follows:

$$
R_{loss} = -\sum \log\left[p(x_i|\mu_i, s_i) - 10^{-7}\right],
$$

where the factor 10^{-7} is used only as a number that will keep the argument of the logarithm as non zero. Following the above steps, we can finally calculate the RE for the Frey Faces dataset as the average of the R_{loss}.

Combining all the above, we define the overall loss as $L_{\text{overall}} = -\beta \times D_{MC,L} + RE$, where $\beta \in \mathbb{R}$ is a parameter, which we are annealing during the training epochs.

2.3 Arithmetic stability and the logsumexp trick

In order to increase arithmetic stability of our algorithm, we use the logsumexp trick, which helps us avoid `NaN` values by preventing divisions with very small or very large numbers, which are considered zero or `Inf` in computers' floating-point arithmetic respectively. For this purpose, when we aim to compute some small number $x \in \mathbb{R}$, we may rewrite it as $x = \log e^x = \log e^{x+M-M} = \log(e^M e^{x-M}) = \log e^M + \log(e^M e^{x-M}) = M + \log(e^M e^{x-M})$, where $M \in \mathbb{R}$ is a large number that we picked arbitarily.

Similarly, if we consider $\log \sum_{k=1}^{K} f_k$, where $f_1, f_2, ..., f_K \in \mathbb{R}$ are either extremely small or very large numbers, we may find the maximum among them $M = \max(f_1, f_2, ..., f_K)$ and then similarly apply the logsumexp trick. By computing similar steps, we derive a more stable equivalent numerical expression, which equals:

$$\log \sum_{k=1}^{K} e^{f_k} = \log \left(\sum_{k=1}^{K} e^{f_k + M - M} \right) = \log \left(e^M \sum_{k=1}^{K} e^{f_k - M} \right) = M + \log \left(\sum_{k=1}^{K} e^{f_k - M} \right)$$

Precisely, we perform the trick once when computing the outputs of the VampPrior $p_\lambda(\mathbf{z})$. After we have calculated the latent representation for the computed pseudo-inputs means and expanded the calculations, we take the max element M in $\log\left(\mathcal{N}(\mu_\phi, \Sigma_\phi)/K\right) = \log(\mathcal{N}(\mu_\phi, \Sigma_\phi)) - \log K$, where K is the number of pseudo-inputs and compute logsumexp to increase numerical stability. In addition, we also apply the logsumexp trick when calculating the likelihood of the dataset $\mathcal{L}(\phi, \theta, \lambda)$. That is, we take into account all the overall losses considering both image reconstruction and respect to prior, for all N samples.

2.4 Experimental Setup[1]

With the purpose of evaluating how the more complex VampPrior affects the VAEs generative performance, compared to the simple Standard Gaussian, we conducted experiments using the Frey Faces and the MNIST datasets. More specifically, we split the datasets in ratios of $[0.8, 0.1, 0.1]$ for train, validation and test purposes while training for 1000 and 830 epochs for MNIST and Frey Faces datasets respectively. Additionally, we used warm up training for 100 epochs in which we gradually increase, up to 1 the β coefficient, which determinate the contribution of the prior regularization in the loss $L_{\text{overall}} = -\beta \times D_{MC,L} + RE$, $\beta \in \mathbb{R}$.

In addition, we experimented by modifying the Neural-Network architecture, precisely doubled the amount of units in the hidden layer and therefore used 600 units instead of 300. Another factor of the architecture we experimented with was the number of pseudo-inputs K that were used, for computing the VampPrior $p_\lambda(\mathbf{z}) = 1/K \times \sum_{i=1}^{K} q_\phi(\mathbf{z}|\mathbf{u_k})$. Precisely, apart from the default value for $K = 40$, we experimented with $K = 10$ and $K = 100$. Our aim was to investigate how these parameters affect the latent representations produced by the encoder and generative capabilities of the decoder, while coupling the prior with the posterior. Finally, we trained using the Adam optimizer with normalized gradients, mini batch size of 100 samples, learning rate $lr = 5 \times 10^{-4}$.

3 Results

From our experiments, it is evident that VampPrior outperforms the standard centered isotropic multivariate Gaussian $p_\lambda(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ prior, as the authors of the original paper claim, for different values of the latent size $M \in \{10, 40, 100\}$ thus by learning representations $\mathbf{z} \in \mathbb{R}^M$ of different dimensions. When it comes to modifying the network architecture, thus the hidden units size and the number of pseudo-inputs, again, VampPrior outperformed the centered isotropic multivariate Gaussian prior $p_\lambda(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ as it is demonstrated on Table 3, keeping latent size constant and equal to 40. However, considering Tables 1 and 3, the results of different architectures are similar to each other. There is only one case where Standard prior outperforms VampPrior in MNIST dataset,

[1] The code for the experiments is available on Github. Please inform us to add you as a collaborator.

for $M = 100$, 500 pseudo-inputs and hidden unit size equal to 300.

Table 1: **Bits per dimension**, **Frey Faces**

Methods		
Standard	VampPrior	Latent Size
4.569	**4.555**	10
4.620	**4.532**	40
4.562	**4.537**	100

Table 2: **Likelihood** for **MNIST**

Methods		
Standard	VampPrior	Latent Size
93.917	**92.297**	10
89.703	**89.015**	40
90.042	90.780	100

Table 3: **Bits per dimension** results for **Fray Faces** dataset trying different architectures

Methods				
Standard	VampPrior	Latent Size	Hidden unit size	Number of pseudo-inputs
4.620	**4.536**	40	600	500
4.617	**4.532**	40	300	1000
4.617	**4.528**	40	300	100

What is more, apart from the likelihood and bits per dimension results for Fray Faces and MNIST datasets, we include several figures as well. Indicative results with latent size 10 and the standard method at epoch 829 with Frey Faces dataset are the following:

Figure 1: Generated Image

Figure 2: Test Reconstruction

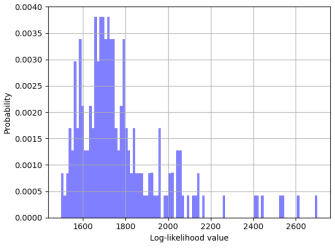

Figure 3: Likelihood

Results with with latent size 10, VampPrior and with Frey Faces dataset are the following:

Figure 4: Generated Image

Figure 5: Test Reconstruction

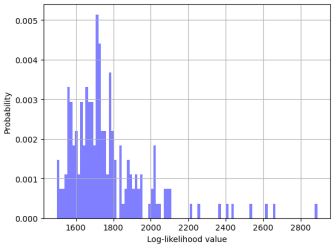

Figure 6: Likelihood

The examples below illustrate results with Fray Faces data set, latent size 40 for the standard prior:

Figure 7: Generated Image

Figure 8: Test Reconstruction

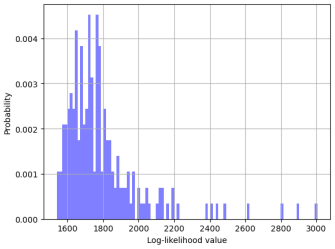

Figure 9: Likelihood

Now follow the results with the same dataset and parameters for the VampPrior:

Figure 10: Generated Image

Figure 11: Test Reconstruction

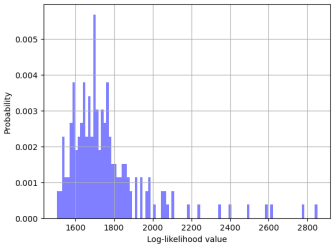

Figure 12: Likelihood

The examples below illustrate results with MNIST data set, latent size 40 for the standard prior:

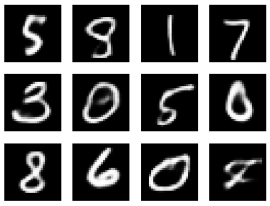

Figure 13: Generated Image

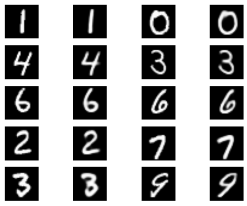

Figure 14: Test Reconstruction

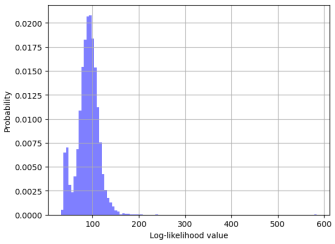

Figure 15: Likelihood

Based on the qualitative results presented above it is evident that both standard and VampPrior reconstruct their inputs effectively while their generated data are quite convincing in all four setups. When investigating the likelihood histograms, their heavy-tailed structure indicates the presence of multiple relatively easy to reconstruct samples as well as the existence of a few hard one. Additionally, it can be concluded that the VampPrior approach results in less heavy-tailed histograms, compared to the standard one.

Below, we depict some indicative pseudoinputs and as well as the loss graph for a VampPrior setup:

Figure 16: Generated Image

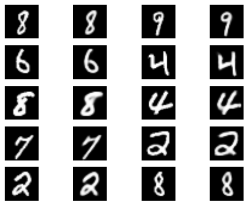

Figure 17: Test Reconstruction

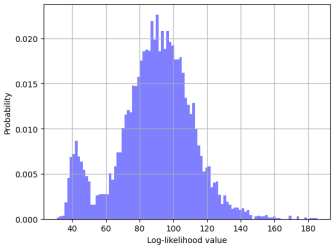

Figure 18: Likelihood

Regarding the qualitative results referring to the MNIST dataset, the conclusion are similar to make in the Frey Faces case. Comparing the performances between the Frey Face and the MNIST datasets, it is clear that VAE perform better as suggested by both the qualitative and the quantitative, in terms of likelihood results. This conclusion perfectly aligns with our intuition, since learning a face is vastly harder than learning hand-written digits.

The pseudoinputs for the MNIST and Frey Frame data sets are the following:

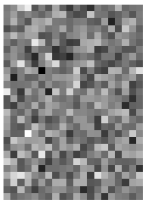

Figure 19: Pseudo-inputs Frey Face dataset

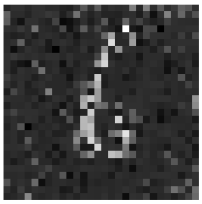

Figure 20: Pseudo-inputs MNIST dataset

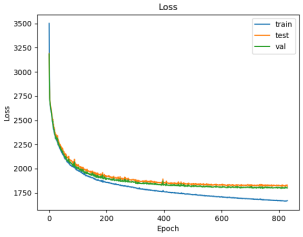

Figure 21: Loss for Standard Method with Latent size 10

In their work, J Tomczak and Max Welling argued that enforcing simplistic prior distribution in VAE networks, results in overregularization which leads to lacking latent representations. Additionally, the authors suggest that the optimal prior distribution can be obtained by minimizing the loss with respect to the prior. More specifically, it is shown that the aggregated posterior $p^*_\lambda(\mathbf{z}) = \frac{1}{N}\sum_{n=1}^{N} q_\phi(\mathbf{z}|\mathbf{x}_n)$ is theoretically the optimal choice. However, in practice, computing the aggregated posterior is computational expensive while it also turns to produce rather poor results due to overfitting. Build upon these, the authors proposed a method addressing both the overegularization and the overfitting caused by a too simplistic and the aggregated priors respectively, via approximating the latter through a mixture of variational posteriors with pseudo-inputs.

4 Discussion

As stated in the **Loss function** section, the authors in [1], address the intractability of the input's marginal likelihood by considering its lower bound (ELBO). Although minimizing the ELBO instead of the intractable marginal likelihood feels like a reasonable abstraction, Alexander Alemi et al. in their work[7] showed that this is not always a good practise. More specifically the authors provide a framework showing that models with identical ELBO's can result in different qualitative and quantitative performances. In order to mitigate this issue, the writers suggest that Reconstruction Error and KL divergence should they be reported independently, since solely using likelihood-based loss fails to deliver qualitative insights for some category of models.

Finally, in [8], the author indicated that using the convenient Gaussian-based parameterization, used in [1], is not appropriate for modeling latent hyperspherical structure. In order to effectively resolve

this issue, Tim R. Davidson et al. proposed a novel von Mises-Fisher distribution capable of modeling hyperspherical spaces.

References

[1] Tomczak, J.M. & Welling, M. (2018) VAE with a VampPrior. *AISTATS 2018*.

[2] Kingma, D.P. & Welling, M. (2014) Auto-Encoding Variational Bayes. *CoRP*, abs/1312.6114.

[3] Hoffmann, M.D. & Johnson, M.J. (2016) ELBO surgery: yet another way to carve up the variational evidence lower bound. *NIPS Workshop: Advances in Approximate Bayesian Inference*.

[4] Dauphin, Y.N., Fan A., Auli, M. & Grangier, D. (2016) Language Modeling with Gated Convolutional Networks, arXiv:1612.08083, 2016.

[5] Durrieu, J., Thiran, J. & Kelly, F. (2012) Lower and upper bounds for approximation of the Kullback-Leibler divergence between Gaussian Mixture Models. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, pp. 4833-4836, doi: 10.1109/ICASSP.2012.6289001.

[6] Kingma, D.P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever & Welling, M. (2016). Improved variational inference with inverse autoregressive flow. *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, pp. 4743–4751.

[7] Alemi, A.A., Poole, B., Fischer, I.S., Dillon, J.V., Saurous, R.A. & Murphy, K. (2018). Fixing a Broken ELBO. *ICML*.

[8] Davidson, T., Falorsi, L., Cao, N.D., Kipf, T., Tomczak, J. & (2018). Hyperspherical Variational Auto-Encoders. *UAI*.