VAE with VampPrior Project – Large VAE

DD2434 – Machine Learning, Advanced Course

Filip Bergentoft, Ludvig Doeser, Erik Rydving, Sara Sjöberg.

January 15, 2021

Abstract

Deep generative models have recently been very popular for reconstructing and generating data, such as images. One of the most successful models is the Variational Auto-Encoder (VAE), which usually uses a standard Gaussian prior. In [Tomczak and Welling, 2017], a new *Variational Mixture of Posteriors* Prior, or VampPrior for short, was introduced. It was shown to make the prior more flexible, allowing it to capture the latent representation to a greater extent. In this project, we have implemented the VAE with three different priors – standard Gaussian, VampPrior and VampPrior*data* – on the following datasets: *staticMNIST*, *Caltech 101 Silhouette*, and *FreyFaces* from the original paper, and *FashionMNIST* dataset, to extend the investigation. In all cases, we reproduce one of the most striking effects of using VampPrior; it always outperforms the standard Gaussian prior.

1 Introduction

The method of using generative models as an unsupervised machine learning technique has been tremendously popular lately. With the objective to capture the true data distribution one has turned to the usage of neural networks to train and find an approximate model distribution. Variational Auto-Encoders (VAEs) [Kingma and Welling, 2014], and Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] are among the most widely used and efficient approaches.

There have been multiple suggestions for improvements to these techniques. In contrast to GANs, which in general manage to generate sharper images, VAEs advantageously contain both an inference model and a generative model [Mescheder et al., 2018]. In addition, VAEs are claimed to have considerably better log-likelihoods [Wu et al., 2017]. VAEs have, however, been criticized for being too inexpressive to approximate the true posterior distribution well enough. A more expressive model for VAEs have been suggested in e.g. [Kingma et al., 2017], and an extension called adversarial variational Bayes effectively combines the advantages of VAEs and GANs [Mescheder et al., 2018].

Another successful approach is built on replacing the standard Gaussian prior that a *vanilla* VAE uses. The choice of prior distribution has namely been shown to play a vital role for the performance of the VAE [Tomczak and Welling, 2017]. As the prior works as a mediator between the variational encoder and the generative decoder, see Figure 1, it is vital to not choose a too simple prior; this may lead to, for example, over-regularization. In [Hoffman and Johnson., 2016], it is stated that multimodal priors might be a fruitful option. The use of a *variational mixture of posteriors* prior – VampPrior – was shown to result in a more powerful model [Tomczak and Welling, 2017].

This project, which has used a VAE on four common data sets of images – *staticMNIST, fashionMNIST, Caltech 101 Silhouettes*, and *Frey Faces* – to compare the effect that different priors have on the reconstruction and generation of data, is organized as follows. In section 1.1 the theory behind VAEs and the VampPrior is presented and in section 2 the implementation of the model in `TensorFlow` is described. Then, in section 3, the results with different priors are qualitatively and quantitatively compared. Lastly, in section 4, the results are compared with [Tomczak and Welling, 2017] and the use of different priors in VAEs are discussed in a broader context.

1.1 VAE

The Variational Auto-Encoder (VAE) is a method to perform approximate inference and learning with directed probabilistic models with intractable posterior distributions for either, or both, latent variables and model parameters. In the simplest form with images as the data type, the VAE takes in an image and runs it through a neural network (NN) – the encoder – to get means and variances for each dimension of a lower-dimensional latent space z, *the code*. A sample from the standard Gaussian prior distribution is then taken from the latent space with these parameters and is put through another NN – the decoder – to get back an image. The goal is to train the NNs such that the output image is as similar to the input image as possible, and further to be able to draw any random sample from the latent space to generate an image similar to the real data. The original Variational Auto-Encoder was presented in [Kingma and Welling, 2014] with the following setup. Given a dataset $X = \{x_i\}_{i=1}^N$ of i.i.d. samples it is assumed that each sample x_i is generated by a random process dependent on a latent variable Z which can be described by, firstly,

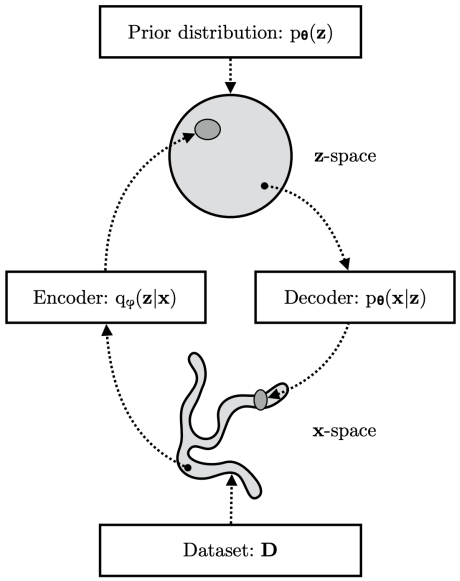

Figure 1: *Graphical explanation of a VAE.* Data points are mapped to the latent space with the encoder $q_\phi(z|x)$, and samples from the prior $p_\theta(z)$ are mapped back with the decoder $p_\theta(x|z)$. Figure 2.1 from [Kingma and Welling, 2019].

sampling z_i from a distribution $z_i \sim p_{\theta^*}(z)$, and then generating the data x_i by sampling $x_i \sim p_{\theta^*}(x|z_i)$.

It is further assumed that the prior $p_{\theta^*}(z)$ and the likelihood $p_{\theta^*}(x|z)$ are from parametric families of distributions $p_\theta(z)$ and $p_\theta(x|z)$. The prior and likelihood distributions are thus seen as known but the true parameters θ^* are unknown. The likelihood $p_\theta(x|z)$ is referred to as the decoder as it, given a z, produces a distribution over x and is modelled by a NN with parameters θ.

The posterior $p_\theta(z|x)$ is intractable and is approximated by a variational posterior $q_\phi(z|x)$ referred to as the encoder which, similarly to the decoder, is modelled by a NN with parameters ϕ. The aim is to maximize the marginal log-likelihood $\log p_\theta(x_i)$ over the parameters θ which can be formulated as a maximization problem over both θ and ϕ using that

$$\begin{aligned}\log p_\theta(x_i) &= D_{KL}\big(q_\phi(z|x_i)\|p_\theta(z|x_i)\big) + \mathcal{L}(\theta,\phi,x_i)\\ &\geq \mathcal{L}(\theta,\phi,x_i).\end{aligned} \tag{1}$$

One can thus maximize the marginal log-likelihood $\log p_\theta(x_i)$ by maximizing $\mathcal{L}(\theta,\phi,x_i)$ with respect to the parameters θ and ϕ where

$$\begin{aligned}\mathcal{L}(\theta,\phi,x_i) = &- D_{KL}\big(q_\phi(z|x_i)\|p_\theta(z)\big)\\ &+\mathbb{E}_{q_\phi(z|x_i)}\big[\log p_\theta(x|z)\big],\end{aligned} \tag{2}$$

which can be rewritten as

$$\begin{aligned}\mathcal{L}(\theta,\phi,x_i) =&\mathbb{E}_{q_\phi(z|x_i)}\big[\log p_\theta(z) - \log q_\phi(z|x_i)\big]\\ &+\mathbb{E}_{q_\phi(z|x_i)}\big[\log p_\theta(x|z)\big].\end{aligned} \tag{3}$$

Eq. (2) is used if the KL-divergence term can be computed analytically, otherwise Eq. (3) is used. The expectations in Eq. (2) and (3) can be estimated using Monte-Carlo methods.

The aim is to maximize $\mathcal{L}(\theta,\phi,x_i)$ with respect to θ and ϕ; the parameters in the decoder $p_\theta(x|z)$ and encoder $q_\phi(z|x)$ which are modelled as NNs. They are thus in need of a notion of loss. In order to achieve a loss function one can restate the optimization problem as a minimization of $-\mathcal{L}(\theta,\phi,x_i)$ with respect to θ and ϕ. The loss function $\text{Loss}(\theta,\phi,x_i)$ can thus be defined as

$$\text{Loss}(\theta,\phi,x_i) = -\mathcal{L}(\theta,\phi,x_i) \tag{4}$$

In order to allow backpropagation, the gradient of $\mathcal{L}(\theta,\phi,x_i)$ has to be computed with respect to parameters θ and ϕ. However, when estimating $\mathcal{L}(\theta,\phi,x_i)$ using the naïve Monte Carlo estimate the gradient with respect ϕ becomes troublesome by exhibiting high variance and becoming intractable for some choices of distributions (i.e. we cannot always backpropagate through a stochastic node). To overcome these problems we apply the reparameterization trick.

1.1.1 Reparametrization trick

Using the naïve Monte Carlo estimate, a part of the gradient that has to be computed is

$$\begin{aligned}&\nabla_\phi \mathbb{E}_{q_\phi(z|x_i)}\big[f(z)\big]\\ &= \mathbb{E}_{q_\phi(z|x_i)}\big[f(z)\nabla_\phi \log q_\phi(z|x_i)\big]\\ &\approx \frac{1}{L}\sum_{l=1}^{L} f(z_l)\nabla_\phi \log q_\phi(z_l|x_i), \quad z_l \sim q_\phi(z|x).\end{aligned} \tag{5}$$

This estimator exhibits high variance and is potentially intractable. This is solved by expressing the random variable Z as a deterministic mapping

$$z = g_\phi(\epsilon, x), \quad \epsilon \sim p(\epsilon),$$

where g_ϕ is an appropriately chosen function parameterized by ϕ such that $z_l = g_\phi(\epsilon_l, x) \sim q_\phi(z|x)$, $\epsilon_l \sim p(\epsilon)$. With this, $\int f(z)q_\phi(z|x)dz = \int f(z)p(\epsilon)d\epsilon = \int f(g_\phi(\epsilon,x))p(\epsilon)d\epsilon$. One can thus estimate the expectation using

$$\int f(z)q_\phi(z|x)dz \approx \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(x,\epsilon_l)), \tag{6}$$

where $\epsilon_l \sim p(\epsilon)$. This is an estimator exhibiting a lower variance than the naïve estimator (5) and has a more convenient derivative with respect to ϕ since $p(\epsilon)$ is independent of ϕ.

1.1.2 Vanilla VAE

The Vanilla VAE is the most basic example of the VAE and is based on the following. Given a dataset $X = \{x_i\}_{i=1}^N$ of i.i.d. samples and a latent variable z of a dimension chosen by the user the following assumptions are made

- x_i created by a random process involving z
- $q_\phi(z|x) = \mathcal{N}(z; \mu_\phi, I\sigma^2_\phi)$ (encoder)
- $p_\theta(z) = \mathcal{N}(z; 0, I)$ (prior)
- $p_\theta(x|z) = \mathcal{N}(x; \mu_\theta, I)$ (decoder)

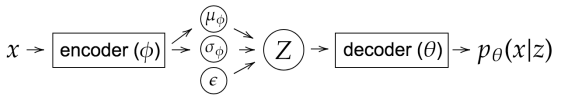

Figure 2: *Graphical model of the Vanilla VAE.*

where $p_\theta(x|z)$ and $q_\phi(z|x)$ are modelled by NNs. A graphical model of the full setup can be seen in Figure 2. An input x_i enters the encoder $q_\phi(z|x_i)$ which outputs the parameters $\mu_{\phi,i}$ and $\sigma_{\phi,i}$. One can then sample $z_{i,l} \sim q_\phi(z|x_i)$ using the result of the reparameterization trick, yielding

$$\begin{aligned} z_{i,l} &= g_\phi(x_i, \epsilon_l) \\ &= \mu_{\phi,i} + \sigma_{\phi,i} \odot \epsilon_l, \ \epsilon_l \sim \mathcal{N}(0, I) \end{aligned} \tag{7}$$

where $\odot$ denotes the element-wise product between vectors. Each $z_{i,l}$ is then passed through the decoder $p_\theta(x|z_{i,l})$ which outputs the parameters $\mu_{\theta,i,l}$; in the case of images as data input, these $\mu_{\theta,i,l}$ are exactly the reconstructed or generated images.

In order to train the encoder and decoder, the loss function introduced in Eq. (4) is used. With the assumptions on $p_\theta(z)$, $p_\theta(x|z)$ and $q_\phi(z|x)$ the loss function can be expressed as

$$\begin{aligned} &\text{Loss}(\theta, \phi, x_i) = -\mathcal{L}(\theta, \phi, x_i) \\ &= D_{KL}\big(q_\phi(z|x_i) \| p_\theta(z)\big) - \mathbb{E}_{q_\phi(z|x_i)}\big[\log p_\theta(x|z)\big] \\ &\approx D_{KL}\big(q_\phi(z|x_i) \| p_\theta(z)\big) - \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x_i|z_{i,l}), \end{aligned} \tag{8}$$

since given the choice of prior $p_\theta(z)$ and encoder $q_\phi(z|x_i)$ the KL-divergence can be differentiated and computed analytically. With the use of the reparameterization trick the loss function $\text{Loss}(\theta, \phi, x_i)$ is now differentiable with respect to θ and ϕ and can thus be optimized using backpropagation that propagates through both the encoder and decoder.

An issue with choosing the standard Gaussian prior – a single multivariate Gaussian – is that it will act as an anchor to the approximate posterior $q_\phi(z|x)$ due to the regularization term $D_{KL}\big(q_\phi(z|x_i) \| p_\theta(z)\big)$ and its monomodal property. This causes the approximate posterior to remain close to the mode of the prior which may hamper the performance. This effect can be reduced by instead introducing a multimodal prior.

1.1.3 VampPrior VAE

The *Variational Mixture of Posteriors Prior* (VampPrior) introduced in [Tomczak and Welling, 2017] proposes a new choice of prior stemming from the methods of Empirical Bayes. In Empirical Bayes, given a prior $p_\lambda(z|\lambda)$ with hyperparameters λ the probability of the data

$$p(x|\lambda) = \int_Z p_\theta(x|z) p_\lambda(z|\lambda) dz \tag{9}$$

is maximized with respect to the prior and hyperparameters λ.

In this case of VAE $p(x|\lambda)$ is intractable, but using Eq. (1) one can instead maximize $\mathcal{L}(\theta, \phi, x_i)$ with respect to $p_\lambda(z|\lambda)$, yielding the following optimization problem

$$\max_{p_\lambda(z|\lambda)} \mathbb{E}_{q_\phi(z|x_i)}\big[\log p_\lambda(z|\lambda)\big] - \beta\Big(\int p_\lambda(z|\lambda) dz - 1\Big),$$

where the second term comes from the method of Lagrangian multipliers and ensures that $p_\lambda(z|\lambda)$ is a proper density. The solution of this problem is found in [Hoffman and Johnson., 2016] and is given by

$$p^*_\lambda(z|\lambda) = \frac{1}{N}\sum_{i=1}^{N} q_\phi(z|x_i). \tag{10}$$

However, this choice is computationally demanding and due to being dependent on the data $X = \{x_i\}_{i=1}^N$ may lead to overfitting. In order to overcome these issues but still retain a prior with a multimodal property (such that it does not over-regularize the posterior), the optimal solution (10) can be approximated by a variational mixture of posteriors with *pseudo-inputs* u_k as

$$p_\lambda(z) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(z|u_k). \tag{11}$$

The pseudo-inputs are generated by an additional NN referred to as *means*, whose input is simply an identity matrix $I \in \mathbb{R}^{K \times K}$, and whose output u_k, which could be seen as a pseudo-image (see Figure 7), is fed into the encoder as seen in Figure 3. The *means* NN is then trained by a continued backpropagation from the encoder originating from the training of the encoder and decoder.

An alternative prior called VampPrior-*data* works similarly, but without the *means*-network. Instead of the pseudo-inputs u_k in Eq. (11), K random images from the training-data are used.

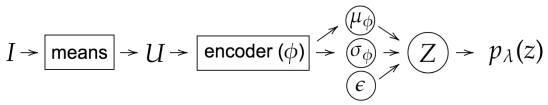

Figure 3: *Graphical model of the VampPrior VAE.* The decoder is unchanged from Figure 2.

2 Method

A VAE has been implemented and trained on four image sets *staticMNIST*, *FashionMNIST*, *Caltech 101 Silhouette*, and *Frey Faces*, presented in Figure 4. All except *FashionMNIST* were studied in [Tomczak and Welling, 2017], while this dataset was included to further empirically test if the VampPrior outperforms the standard normal prior. The *staticMNIST* and *FashionMNIST* datasets were available to import in `TensorFlow` while the other two were downloaded from their respective websites[1].

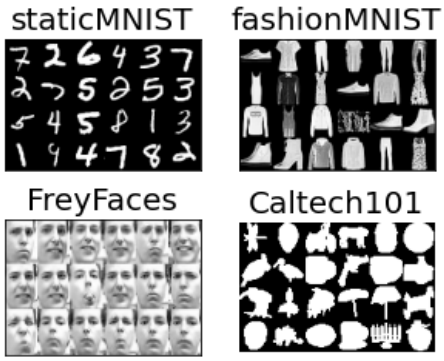

Figure 4: *Example images from the four datasets.*

The VAE model was implemented and trained in `TensorFlow` using the optimizer `keras.optimizers.Adam()`. The encoder and decoder networks were both given two layers with 300 hidden units each, and the dimension of the latent space was set to 40. The weights of the networks were initialized according to Xavier [Glorot and Bengio, 2010].

For the VampPrior, an additional network with a single layer, called *means*, was inserted before the encoder as described in section 1.1.3. The weights were initialized through a standard Gaussian distribution with mean= 0 and standard deviation= 0.01. In all cases, 500 pseudo-inputs were utilized (see motivation in Figure 5).

To quantitatively evaluate the performance of the different priors, we compute the log-likelihood with the use of importance sampling on the test-data. Generally, we have

$$\begin{aligned}\log p(x) &= \log \mathbb{E}_p[p(x|z)] = \log\left(\int p(x|z)p(z)dz\right)\\ &= \log\left(\int p(x|z)\frac{p(z)}{q(z|x)}q(z|x)dz\right)\\ &= \log \mathbb{E}_q\left[p(x|z)\frac{p(z)}{q(z|x)}\right],\end{aligned}$$

and the sampling is then performed through

$$\begin{aligned}\log p(x) &=\simeq \log\left(\frac{1}{n}\sum_{i=1}^{N_s} p(x|z_i)\frac{p(z)}{q(z|x)}\right)\\ &= \log\left(\frac{1}{n}\sum_i e^{\log p(x|z_i)+\log p(z_i)-\log q(z_i|x)}\right)\\ &= \texttt{logsumexp}(\text{Loss}) - \log(n),\end{aligned} \tag{12}$$

where we choose to use $N_s = 500$ number of samples as in the original article, and where Loss is the loss-function defined in Eq. (4) and with the use of $L = 1$ from Eq. (6).

A VAE was trained for 1000 epochs for each dataset and prior choice, and the marginalized log-likelihood was calculated for each trained model.

3 Results

To choose the number of pseudo-inputs to use for VampPrior, models with a different number of pseudo-inputs were trained on *staticMNIST* for 100 epochs to see how the log-likelihood would be affected. The results can be seen in Figure 5.

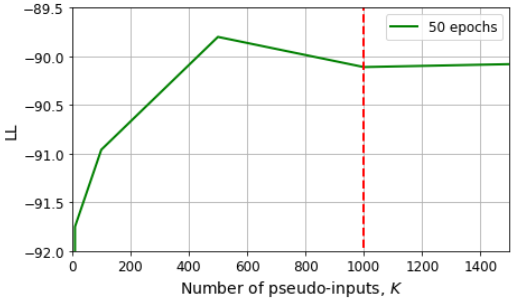

Figure 5: *Test marginal log-likelihood as a function of number of pseudoinputs.* The model was trained 100 epochs with staticMNIST. The red line indicates where [Tomczak and Welling, 2017] stopped.

[1] Caltech 101: https://people.cs.umass.edu/~marlin/data/caltech101_silhouettes_28_split1.mat
Frey Faces: http://www.cs.nyu.edu/~roweis/data/frey_rawface.mat

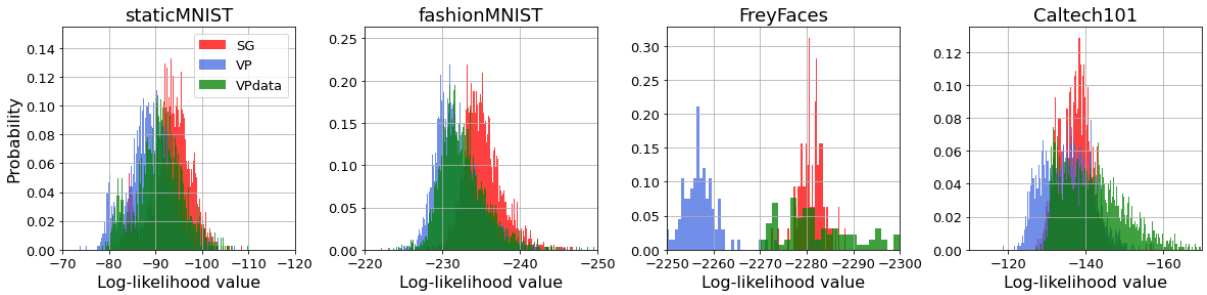

Figure 6: *Histograms of test marginal log-likelihoods for all data-sets.* In all cases, VampPrior (VP) results in the best log-likelihood values.

In Figure 6, histograms for the test marginal log-likelihood (LL) are shown for all datasets and all priors used. It is evident that the VampPrior yields the best log-likelihood in all cases, which is also shown in Table 1, where the mean values for $\log p(x)$ are presented. We also note that VampPrior-*data* is better than standard Gaussian for the *MNIST* datasets, but not for *Frey Faces* and *Caltech 101*.

DATASET	SG	VP	VP *data*
staticMNIST	-93.08	**-88.72**	-90.85
fashionMNIST	-235.32	**-231.94**	-232.79
Frey Faces	-2281.16	**-2257.76**	-2286.52
Caltech 101	-137.32	**-135.15**	-142.02

Table 1: *Test marginal log-likelihood after training VAEs for all combinations of datasets and priors.* VampPrior gives better results for all datasets.

Next, we present a subset of pseudo-inputs generated by the trained *means*-NN in Figure 7, and a qualitative visual comparison of the reconstructed and generated images using VP in Figure 8.

4 Discussion

We have reproduced some of the results in [Tomczak and Welling, 2017] by using a VAE with the VampPrior on some common datasets. Looking at Table 1 we see that using the VampPrior gives the higher log-likelihood of the three tested prior for all four datasets, in accordance with the original VampPrior paper. However VampPrior data had the worst performance out of all three priors on *Caltech101* and *FreyFaces*. This is most likely due to its overfitting nature briefly mentioned in [Tomczak and Welling, 2017] and the low number of samples in those two datasets.

In addition we reproduced the log-likelihood plot over the number of pseudo-inputs used, which can be seen in Figure 5. Our results are once again aligned with those of [Tomczak and Welling, 2017], showing that around 500 pseudo-inputs achieves the best performance. Worth mentioning is also that the training time scales with the number of pseudo-inputs. Compared to $K = 500$ pseudo-inputs, $K = 1000$ takes approximately three times as long to train and $K = 1500$ takes five times as long.

Furthermore, it is interesting to note that the pseudo-inputs in Figure 7 are visibly *prototypical* objects in almost all cases; we can identify numbers in *staticMNIST*, clothes in *fashion-MNIST*, silhouettes in *Caltech 101*. That *Frey*

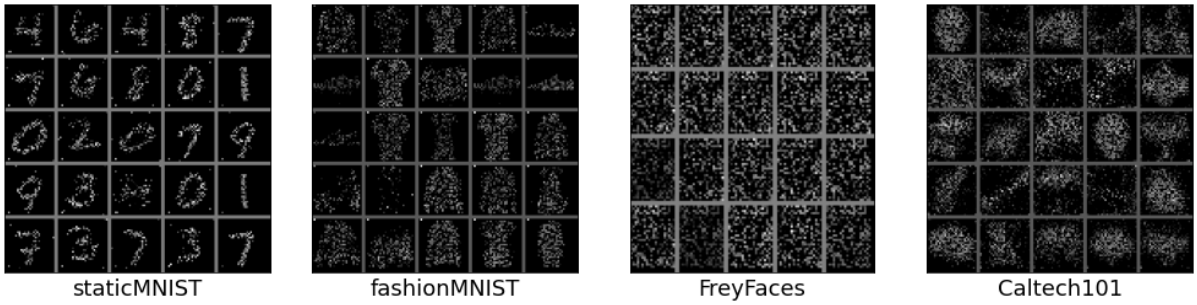

Figure 7: *Subset of trained pseudo-inputs for all datasets.*

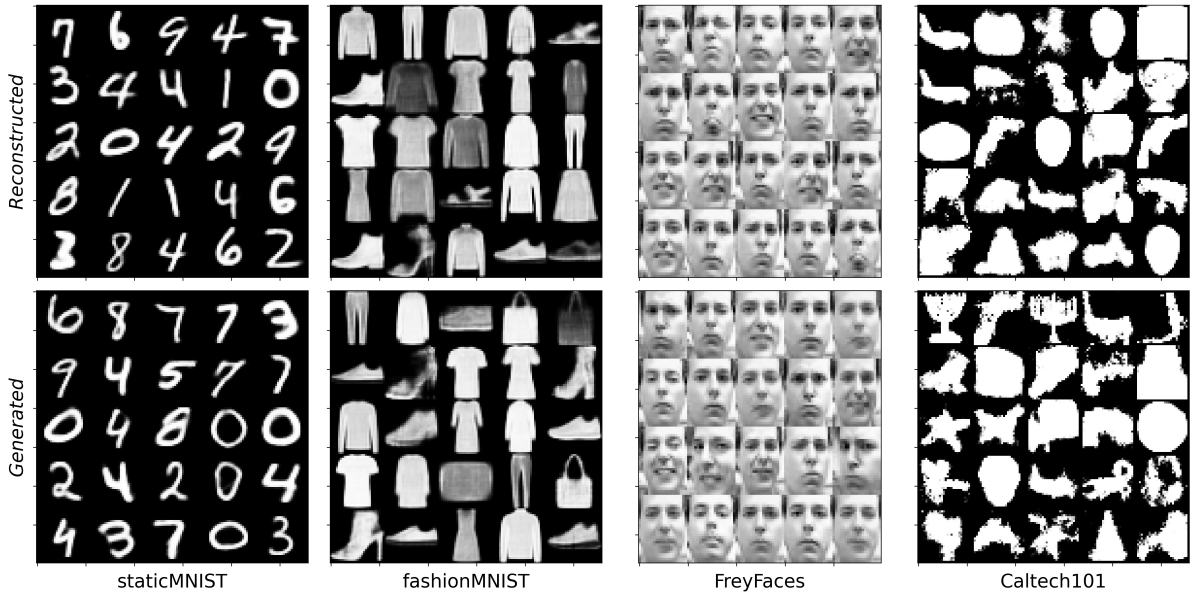

Figure 8: *Comparison between reconstructed and generated data for all datasets.* After 1000 epochs, 40 dimension in latent space, batch_size = 100, number of pseudo inputs are 500.

Faces fails to produce prototypical objects is probably an effect of the low number of data points in *Frey Faces* (only 1900 training points compared to e.g. *staticMNIST* that has 60000 training points), which also explains why the *Frey Faces*-plot in Figure 6 yields larger bins. Another reason could be that the images have mostly bright pixels, and no black background as the other datasets, making it harder to recognize the pseudo-inputs as faces. Lastly, as discussed in [Larsen et al., 2016, Tomczak and Welling, 2017] we show that the VAE occasionally reconstructs and generates blurry images, see especially *Caltech 101* in Figure 8.

4.1 Improvements

To mitigate overfitting, the original VampPrior paper implemented *early-stopping* with a look-ahead of 50 iterations. This would use some of the data for validation to check regularly if the model is starting to become overfitted, and in that case cut the learning short. As we did not implement early-stopping, our results might be overfitted in some cases. However, since the loss reached at the end of learning and the marginal log-likelihood calculated afterwards were not too dissimilar, this does not seem to have been a big problem.

The original VampPrior paper also implemented a Hierarchical VAE. To prevent the problem of inactive latent variables, they used a two-layered VAE with the variational part given by

$$q_\phi(z_1|x, z_2)q_\psi(z_2|x), \tag{13}$$

and the generative part given by

$$p_\theta(x|z_1, z_2)p_\lambda(z_1|z_2)p(z_2), \tag{14}$$

with $p(z_2)$ being the VampPrior. Their results were better for the HVAE than a one-layered VAE for all cases, but since we focused on showing the effect of VampPrior as opposed to a Gaussian prior we decided not to implement the HVAE.

4.2 Comparison with prior of Mixture of Gaussians

Looking back at the approximation of optimal prior we did to get the VampPrior in section 1.1.3, another possible approximation is to let

$$p_\lambda(z|\lambda) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}(\mu_k, \sigma_k^2), \tag{15}$$

which is a Mixture of Gaussians (MoG). Instead of using pseudoinputs, this prior would have trainable parameters $\lambda = \{\mu_k, \sigma_k \ : k \in [K]\}$ that would be trained to approximate Eq. (10) as well as possible. Similarly to the VampPrior, the MoG has the multimodal property that the standard Gaussian prior lacks which reduces the risk of overregularizing the variational posterior. However, the authors of [Tomczak and Welling, 2017]

argue that the VampPrior has two advantages over the MoG.

Firstly, the VampPrior is taken as the mean over the posterior with pseudo-inputs (see Eq. (11)), resulting in a coupled prior and posterior. This results in a reduced number of trainable parameters since $\lambda = \{\mu_k, \sigma_k \quad : k \in [K]\}$ does not exists for the Vamprior and the prior and posterior are trained simultaneously. Instead, when using the VampPrior we only have one $\mu - \sigma$ pair in the latent space.

Secondly MoG and VampPrior have different effects on the resulting gradient in backpropagation. The MoG prior has a similar gradient to the standard Gaussian prior whilst the VampPrior has a term (See equation (20) and (21) in Supplementary material to [Tomczak and Welling, 2017]) which is close to zero when $q_\phi(z|x) \approx q_\phi(z|u_k)$. This means that if the latent distribution produced by the pseudo-inputs is dissimilar to the latent distribution produced by the real inputs the gradient will point in a direction where the approximative posterior has higher variance. This should in theory result in a model that generalizes better.

4.3 Other works using VampPrior

In other works, the VampPrior has for instance been used to implement a Variational Fair Auto-Encode, trying to create a less biased deep generative model [Botros and Tomczak, 2018]. The VampPrior has also been used to show that using the ELBO in training neural networks does not necessarily lead to good qualitative results [Alemi et al., 2018].

Appendix – Code used for the project

Colab_TensorFlow

References

[Alemi et al., 2018] Alemi, A. A., Poole, B., Fischer, I., Dillon, J. V., Saurous, R. A., and Murphy, K. (2018). Fixing a broken elbo.

[Botros and Tomczak, 2018] Botros, P. and Tomczak, J. M. (2018). Hierarchical vampprior variational fair auto-encoder.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. JMLR Workshop and Conference Proceedings.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc.

[Hoffman and Johnson., 2016] Hoffman, M. and Johnson., M. (2016). Elbo surgery: yet another way to carve up the variational evidence lower bound.

[Kingma et al., 2017] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2017). Improving variational inference with inverse autoregressive flow.

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes.

[Kingma and Welling, 2019] Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

[Larsen et al., 2016] Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric.

[Mescheder et al., 2018] Mescheder, L., Nowozin, S., and Geiger, A. (2018). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks.

[Tomczak and Welling, 2017] Tomczak, J. M. and Welling, M. (2017). VAE with a vampprior. *CoRR*, abs/1705.07120.

[Wu et al., 2017] Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. (2017). On the quantitative analysis of decoder-based generative models.