Kungliga Tekniska Högskolan
KTH Royal Institute of Technology

# DD2434 HT20-1 Machine Learning, Advanced Course

[RE] VAE with a VampPrior

*Authors*
Frano Rajič, Ivan Stresec, Matea Španić, Adam Wiker

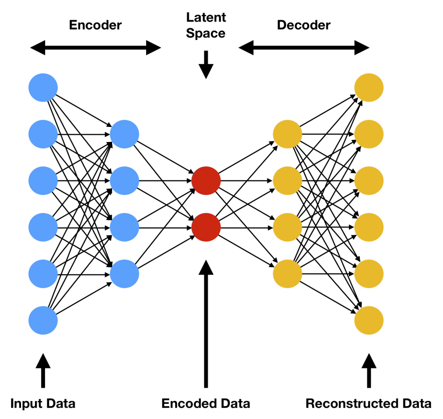
***Abstract***
We reimplement the methods and neural network architectures described in *VAE with a VampPrior* [1]. We briefly introduce and motivate variational autoencoders and the paper. To evaluate and compare the effects of different priors, we implement the standard 1-layered variational autoencoder architecture with a standard Gaussian prior and the VampPrior. Furthermore, we extend the VampPrior using the hierarchical, 2-layered architecture as described in the paper. We test both architectures with both priors on three datasets: MNIST, Omniglot, and Caltech101 (silhouettes). The results successfully re-verify the empirical findings of the original paper, showing the effectiveness of the VampPrior and the suggested hierarchical architecture.

January 15, 2021

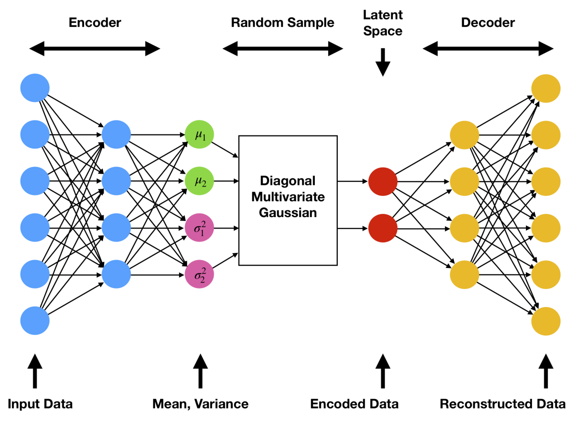
## 1 Introduction

In this project, we briefly contextualize the algorithm and neural network architecture described in *VAE with a VampPrior* [1], re-implement the paper, and reproduce some of its results.

First, a short briefing on the variational autoencoder (VAE). A VAE is a neural network architecture which builds upon the ideas of standard autoencoders. Standard autoencoders have the simple goal of crating a new representation for a set of data, which can be of a lower or higher dimension than the original data. This is done using the reconstruction error of the output and input. Meanwhile, VAEs are designed to also function as a generative model. Both autoencoder variants can be seen side-by-side in figure 1. The generative properties of the VAE are achieved by treating the latent representations as distributions, which then can be regularized and sampled. This way, we are able to create a more meaningful latent representation of data via the encoder, and a more robust decoder which can make sense of all parts of the latent space. From a more probabilistic view, perhaps more in line with the original VAE paper [2], VAE gives us a simple and differentiable lower bound which we can be optimized with a neural network. Maximizing the evidence lower bound (ELBO) is often not possible if using conventional variational inference (VI), as using the mean-field approach requires analytical solutions of the expectation w.r.t. the approximate posterior which are, in the general case, intractable.



(a)



(b)

Figure 1: Standard autoencoder (a) and variational autoencode (b) architectures (*source:* `https://www.compthree.com/blog/autoencoder/`)

There are multiple variants of the standard VAE, for example convolutional VAEs or hierarchical VAEs. Our main focus will be on VAE with a VampPrior (variational mixture of posteriors prior). The original paper for the VampPrior [1] introduces two improvements to the standard VAE architecture. Whereas the original paper on VAEs [2] proposes using a standard normal distribution as the prior, VAE with a VampPrior, as its name suggests, uses a mixture of posteriors to achieve a more flexible prior. The idea here is derived from the fact that one can find a prior to optimize the ELBO by maximizing a Lagrange function. The resulting prior is simply the aggregated posterior over all the data. However, using such a prior could easily lead to overfitting and its calculation would be very costly [1]. On the contrary, using a standard Gaussian as a prior removes costly computations but leads to overregularization. The aggregated posterior leads to overfitting and the Gaussian leads to overregularization. Finding an in-between solution that can overcome these problems will yield better results, and so the authors [1] suggest an approximation of the aggregated posteriors using pseudo-inputs, variables learned through backpropagation. Moreover, while the original VAE paper [2] suggests a simple one-layered latent architecture, the authors of VampPrior suggest a new hierarchical VAE (HVAE), using two, instead of just one layer of latent variables in both the encoder and the decoder. The newly suggested prior and architecture are shown to have state-of-the-art results on several different datasets, always outperforming the originally suggested VAE architecture.

* * *

To reiterate, our goal for this project is to re-implement the VAE architecture described in the VampPrior paper, describe the methods used in our implementation and, finally, verify the re-implementation through the replication of some results shown in the original paper. This will give insights to the effect of different priors and other modifications to the standard VAE.

## 2 Methods

### 2.1 Data

Three datasets of differing complexity were used in the experiments. The first one is the MNIST dataset [3] which features ten different types of handwritten digits. Next we have the Omniglot dataset [4] which features features 1623 different handwritten characters from 50 different alphabets. And lastly we have the Caltech101 dataset [5] which features images of objects belonging to 101 different categories. More specifically the Caltech101 silhouettes dataset was used, which features the silhouettes of caltech101 images in black and white. These three datasets have an increasing complexity to them, with MNIST being the least complex, and therefore give insight to how the different methods scale with the dataset used. All of the datasets used were binarized, meaning that the data was transformed into a binary set containing only ones and zeros. This was done using a random binomial with the initial value, scaled to the range zero to one, as the probability of that node being active, or set to one.

### 2.2 Model

The re-implementation of the VampPrior method was written in Python using the TensorFlow (TF) library. The functionality of the TF library was exploited with the help of TF tutorials. For this project, the decision was made to implement the standard VAE architecture with the standard Gaussian prior and the VampPrior, as well as the HVAE architecture with the same priors. The difference between the VAE and HVAE is rooted in the network structure. The HVAE is a two-layered version of the VAE, as visualized in figure 2. This allows for demonstrative experiments and comparisons. Fully-connected layers with gating were used in all networks for the encoding and decoding layers, like in the original implementation. Gated layers have been shown to maintain non-linearity while reducing the vanishing gradient problem [6]. Architectures using convolutional layers were omitted for simplicity.

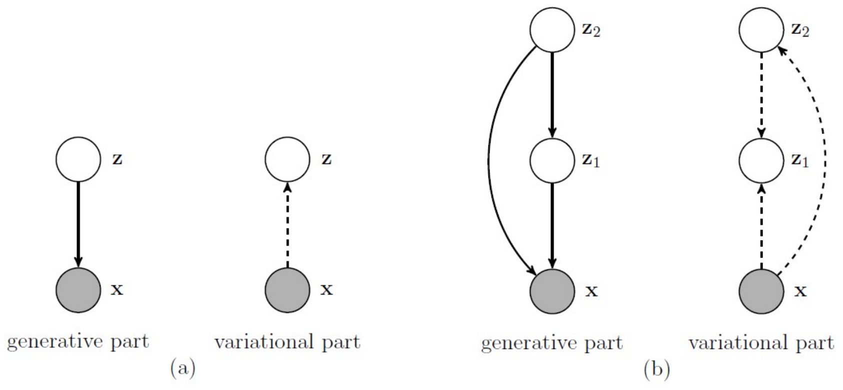


Figure 2: Stochastical dependencies in: (a) a one-layered VAE (b) a two-layered HVAE

### 2.3 Training

Regardless of which architecture or prior is used, the core procedure remains more-or-less the same. This procedure of training the network will now be explained step-by-step.

- First, the encoder is given an image in order to obtain the parameters of the approximate posterior distribution $q(z|x)$. This distribution is modelled as a multivariate normal distribution with a diagonal covariance matrix. Output of the encoder network will be parameters of those distributions: means and variances. To be more specific, we will output log-variances to improve numerical stability. The prior for z is p(z) and is modelled as a standard Gaussian (SG) in one case and as a Variational Mixture of Posteriors prior (VampPrior) in the other case.

- Next, we want to generate sample latent variables for the decoder. We will sample from the distribution defined by the parameters (means, log-variances) obtained by the encoder network. Backpropagation cannot flow through the random node, which arises in the process of sampling. In order to backpropagate through a random node, we will thus apply the reparameterization trick. We will introduce a new parameter $\epsilon$ that is a random sample from the standard normal distribution and will act as a random noise to maintain the randomness of z, as z's can be computed as $z = \mu + \sigma\epsilon$. The procedure can be seen in figure 3.

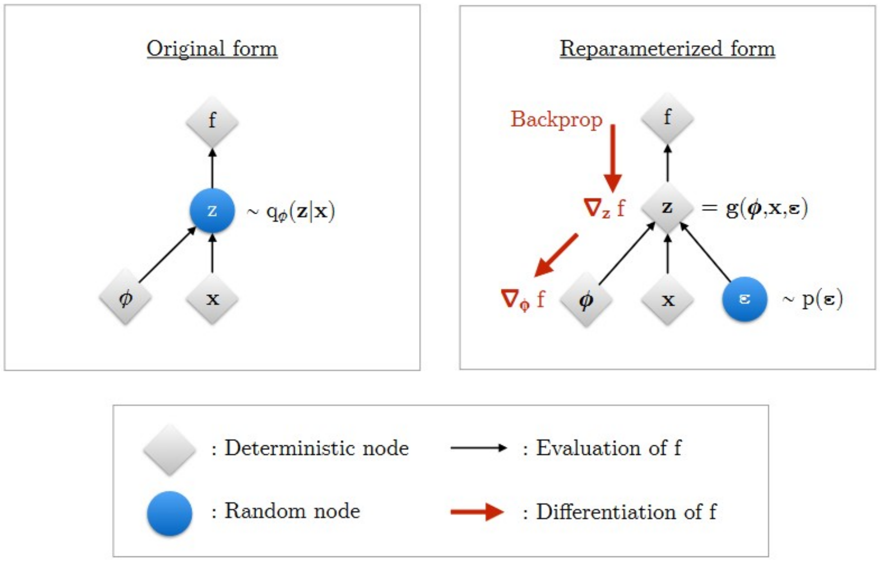


Figure 3: Reparameterization trick (*source:* `https://arxiv.org/pdf/1906.02691.pdf`)

- We then give reparameterized samples to the decoder. As output we get a conditional distribution of the observation $p(x|z)$.

- During training, we want to maximize ELBO on the marginal log-likelihood. Second expectation from the equation:

$$\mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[\ln p(\mathbf{x})] \geq \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p_\theta(\mathbf{x} \mid \mathbf{z}) + p_\lambda(\mathbf{z}) - \ln q_\phi(\mathbf{z} \mid \mathbf{x})\right]\right] \tag{1}$$

  can be estimated by its Monte Carlo estimate. Loss can be computed as the mean of $\log p(x|z) + \log p(z) - \log q(z|x)$.

- In order to minimize the loss function, we record operations during the forward pass using TF's GradientTape which are then used in TF's automatic differentiation to calculate the gradient. The weights are updated with the respect to the negative direction of the gradient, minimizing the loss function and, conversely, maximizing the ELBO.

Once the variational autoencoders are trained we can test them, calculate the average ELBO and the log-likelihood, as well as reconstruct the actual data, i.e. images. We estimate the test marginal log-likelihood by using importance sampling with 5000 sample points, as described in [1].

## 3 Results

Tables 1 and 2 show the values of the ELBO and the log-likelihood for all of the three datasets as well as the VAE and HVAE architectures, with the standard Gaussian prior and the VampPrior. The networks the MNIST dataset were trained for 300 epochs, while the other datasets were trained for 400 epochs. The reason MNIST was trained for less epochs was that it has a lot more datapoints, and takes a lot longer to train on. The choice was made to accept the trade-off in order to spend less time on training.

Table 1: ELBO values of the trained models

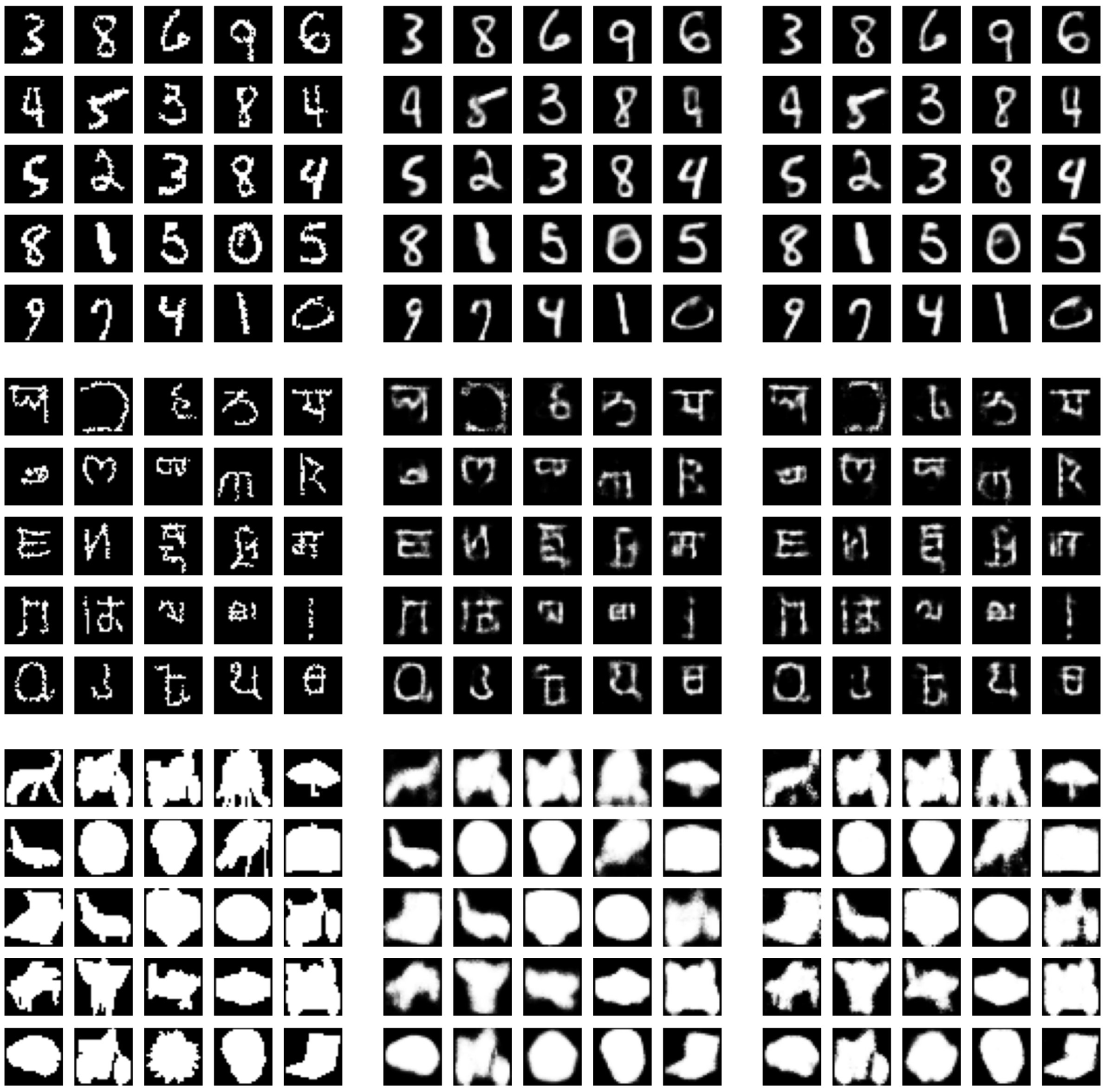
| | VAE | | HVAE | |
|---|---|---|---|---|
| Dataset | Standard | VampPrior | Standard | VampPrior |
| MNIST | -94.33 | -90.78 | -92.96 | -90.34 |
| Omniglot | -124.8 | -124.4 | -123.4 | -123 |
| Caltech101 | -136.6 | -133.3 | -134.2 | -130.6 |

Table 2: Log-likelihoods of the trained models

| | VAE | | HVAE | |
|---|---|---|---|---|
| Dataset | Standard | VampPrior | Standard | VampPrior |
| MNIST | -89.80 | -86.60 | -88.39 | -86.09 |
| Omniglot | -116.92 | -116.29 | -114.76 | -114.48 |
| Caltech101 | -124.88 | -121.78 | -117.53 | -116.92 |

Generally, we can see the ELBO and the log-likelihood values acting as we would have expected and as was shown in the original paper. The architectures using VampPrior always outperform those using the standard Gaussian prior, and HVAE outperforms the VAE when using the same priors. HVAE with the VampPrior always shows the best results, meaning the highest ELBO and the highest log-likelihood.

In figure 4 we see a sample of the raw test set images as well as the reconstructions from two different experiments, the regular VAE with a Gaussian prior and the HVAE with a VampPrior. These are the worst and best performing models for each dataset, showing how the prior and structure's performance boost affects the generating part of the VAE, and the ability to reconstruct images in a way that they're recognizable for humans.



(a) Raw data (b) VAE with std Gaussian (SG) (c) HVAE with VampPrior

Figure 4: Reconstructed test data for different datasets: *MNIST* ($1^{st}$ row), *omniglot*($2^{nd}$ row) and *Caltech101* ($3^{rd}$ row)

## 4 Discussion

From the results in table 1 and 2 we see the same trends which were shown in the original paper [1]. What we do not see are numbers as good as those in the original paper. We were, however, not surprised to find that this was the case as we did not have full information on the training and testing used in the original experiments, meaning we were not able to fully replicate the process. For example we did not find how many epochs they originally trained for, and had to adapt our experiments to the resources we had available. The code we wrote is also not identical to the code used in the original paper and a different library was used for the part of training the networks. This likely lead to small differences, and combined with the fact that we did not include fine-tuning techniques during training, such as early stopping, the slightly worse results were to be expected.

One point of interest related to this is that the results differ less for us when comparing the Gaussian prior with the VampPrior than in the original paper [1], especially for the omniglot dataset. This indicates that the VampPrior, while effective, does not outperform the Gaussian prior as clearly when the training is performed with less careful consideration of surrounding factors and hyperparameters. This could partly be seen when adding the warmup factor during training, as it had a greater effect on the HVAE, especially for the caltech101 dataset where the increase in estimated log-likelihood was more than double for the HVAE compared to the VAE, providing more of a benefit to the models that were harder to train, than the ones that were simpler to train.

From the generated images shown in figure 4 we can first note that for the MNIST dataset neither model struggles to reconstruct the given data. There are a few differences that can be seen such as the zero in the fourth row and the fourth column having less of the unneeded line when reconstructed by the HVAE with a VampPrior. Another example is the four in the last column of the second row, for which the regular VAE with Gaussian prior does not appear too certain if it should be a four or a nine.

For the omniglot dataset there are slightly more differences. The HVAE with a VampPrior generally provides clearer reconstructions with sharper lines and edges, though it is far from perfect. This can also be seen in the original paper [1], although their images were even sharper, which makes sense as we know that their models were trained to a point that achieved a better log-likelihood and ELBO.

We can see a similar effect in the reconstructions of the caltech101 images, with the HVAE with VampPrior reconstructing sharper, more detailed, images. This allows it to maintain some of the features that get lost in the reconstruction from the regular VAE. This also goes in line with the idea that the suggested solutions from the original paper [1] (at least in part) help with the problem of overregularization that is noted to happen with simple priors.

In the paper *Variational Autoencoders with Implicit Optimal Priors* [7] it is mentioned that the VampPrior has sensitive hyperparameters such as the number of mixtures and requires heavy computational cost, making it difficult to tune them. They instead suggest to estimate the KL divergence without modeling the aggregated posterior explicitly, making it cheaper and easier to tune hyperparameters. A disadvantage of their approach being the fact that the implicit prior can not be sampled from. They also note that the pseudo inputs have initial value dependence which warm-up can help with. Their tests show that the new suggested approach performed better than VampPrior with a lighter computational cost.

This more recent paper, which builds upon the VampPrior, shows how problems with the VampPrior can be mitigated through a somewhat different approach. These problems also go well in line with what we have noticed during our experiments, such as it being difficult to train due to sensitivity and high computational cost. But the paper [7] also reaffirms the strengths of the VampPrior, showing that it performs well compared to a standard VAE.

## References

[1] Jakub M. Tomczak and Max Welling. *VAE with a VampPrior*. 2018. arXiv: `1705.07120 [cs.LG]`.

[2] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: `1312.6114 [stat.ML]`.

[3] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: `http://yann.lecun.com/exdb/mnist/`.

[4] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science* 350.6266 (2015), pp. 1332–1338. ISSN: 0036-8075. DOI: `10.1126/science.aab3050`. eprint: `https://science.sciencemag.org/content/350/6266/1332.full.pdf`. URL: `https://science.sciencemag.org/content/350/6266/1332`.

[5] Li Fei-Fei, R. Fergus, and P. Perona. "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories". In: *2004 Conference on Computer Vision and Pattern Recognition Workshop*. 2004, pp. 178–178. DOI: `10.1109/CVPR.2004.383`.

[6] Yann N. Dauphin et al. *Language Modeling with Gated Convolutional Networks*. 2017. arXiv: `1612.08083 [cs.CL]`.

[7] Hiroshi Takahashi et al. *Variational Autoencoder with Implicit Optimal Priors*. 2019. arXiv: `1809.05284 [stat.ML]`.