Project: Large VAE

Francesco Zappia [1] **Vittorio Zampinetti** [1] **Marco Schouten** [1]

Abstract

Among deep generative models, Variational Auto-Encoders (VAE) have been object of extensive research in the last few years. The choice of a prior seems to be determinant for a rich representation in the latent space. The goal of this project is to test and validate the innovative prior proposed in (Tomczak & Welling, 2018) that has been shown to outperform other standard priors in VAEs. Moreover, we also experiment the effects of having two stochastic layers in a hierarchical model. All the code is implemented from scratch using TensorFlow and Keras API and is available at https://github.com/morpheusthewhite/vae-vampprior.

[*]Equal contribution [1]Department of EECS, KTH Royal Institute of Technology, Stockholm, Sweden. Correspondence to: Francesco Zappia <zappia@kth.se>, Vittorio Zampinetti <vz@kth.se>, Marco Schouten <schouten@kth.se>.

1. Introduction

In this group project, we addressed the Variational Auto-Encoder (VAE): a state-of-the-art system used to build generative models. In machine learning literature, VAE is implemented with simple a prior, which serves to regularize the latent space. Conversely, Tomczak and Welling proposed a new innovative prior (Vamp prior) which could synthesize more complex representations in the latent space. Their work comprised the new mathematical formulation of such prior and their extensive comparison to the current literature.

Firstly, we describe the theory behind the standard Variational Auto-Encoders and the modifications regarding the Vamp Prior and the Hierarchical organization of the latent space.

Lastly, we address, compare and explain the results obtained through our implementation of:

- Standard VAE
- Vamp Prior VAE
- Hierarchical VAE

Empirical studies are drawn from two data-sets used in the reference paper (MNIST , Frey Faces) and to incrementally extend the work an additional dataset has been considered (Fashion MNIST).

2. Models

2.1. Variational Auto-Encoder (VAE)

2.1.1. Reasons to use VAE

VAE are generative models. The goal of such generative model is that one may need to generate new inputs with intentional variations, for instance adding glasses to a person in a photo, instead of random pixel variations.

2.1.2. Standard VAE

The scheme of a VAE is realized using a Neural Network whose parameters are tuned and optimized iteratively according to gradient descent. The input data point is processed by a sequence of two multilayer perceptrons (MLP) [1]. The first MLP block is defined as an encoder which maps the input point with a large original dimensionality to a much smaller latent dimensionality. The second block receives as input a point from the latent space and reverts to the dimensionality to its original value.

Having a reduced dimensionality (after encoding) intuitively implies that only the useful information components travel back and forth throughout the encoding-decoding chain (information bottleneck) [2].

To obtain meaningful outputs for any latent vector, the latent space must be organized. Conversely to auto-encoders, VAE adds regularization to avoid overfitting and to ensure that sampling in the latent space allows the generation of new meaningful content (this scheme is summarized in Figure (1)).

Therefore, VAE comprised the following modifications:

- The encoder does not map the input to an exact point, conversely, it maps it to a probability distribution.
- A point sampled from this distribution will be used as input for the decoder.

Figure 1: The key difference between a standard auto-encoder and a variational auto-encoder is represented by the presence of probability distributions as encoder's outputs. (Rocca, 2020)

The encoder's output, being it a probability distribution instead of a point, builds up a continuous latent space. The target latent space distribution is usually (but not limited to) forced being a Normal. This requirement is expressed by the Kullback-Leiber [3] divergence between the Normal standard and the latent distribution. [4]

2.1.3. Additional notes for the regularization of latent space

A regularized latent space has two properties (Rocca, 2020):

- *complete*: any point of the latent space if given as input to the decoder has a meaningful output;
- *continuous*: neighboring points in latent space must have similar outputs.

Even though the encoder produces a distribution (instead of a point) those two conditions are not automatically satisfied.

[1] A multilayer perceptron (MLP) is an example of feed-forward artificial neural network (ANN) (Noriega, 2005)

[2] Information Bottleneck (Alemi et al., 2016)

[3] "KL divergence is a fundamental equation of information theory that quantifies the proximity of two probability distribution". Intuitions behind this equation are provided in (Shlens, 2014)

[4] In case that both the latent and target distribution are to be Normals, the KL equations has a closed-form solution.

This holds because the network always wants to minimize the reconstruction error and it can be achieved by keeping variance close to zero which will force the subsequent sampling to behave equally as if the output was a point.

A similar problem happens when means have very distant values from each other. Consequently, it will locate the various distributions very far from each other and behaving as if the output were fixed points.

The solution of this problem lies in the addition of a regularization term for the distribution parameters. In case of Normal distributions the μ and σ^2 in output from the encoder. The regularization term is given by KL divergence between the Normal with $\mu = 0$ and $\sigma^2 = 1$ and the output of the encoder. In doing so, the distributions overlap ensuring completeness and continuity (Represented in figure 2). The price is greater reconstruction error, but this trade-off can be handled.

Figure 2: Complete and continuous latent space obtained through regularization terms (Rocca, 2020). In the left figure, sampling in the white space induces results without significance (purple drawing)

2.1.4. The mathematical formulation

Inferring this model translates into maximizing the probability of the data, $p_{\lambda,\theta}(\mathbf{x})$

$$p_{\lambda,\theta}(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\lambda(\mathbf{z})\, d\mathbf{z} \tag{1}$$

As this is often not easy to solve, VI can be used to approximate $p_\theta(\mathbf{x}|\mathbf{z})$ with a family of functions $q_\phi(\mathbf{x}|\mathbf{z})$: consequently $\log p_{\lambda,\theta}(\mathbf{x})$ can be written as

$$\log p_{\lambda,\theta}(\mathbf{x}) = KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) + \mathcal{L}(\phi,\theta,\lambda) \tag{2}$$

Being

$$\mathcal{L}(\phi,\theta,\lambda) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})) \tag{3}$$

This formulations clearly fits with the previous intuition, as the first term can be recognized as the one minimizing the reconstruction error, while the second as the regularizing one. Furthermore, if estimating KL with L samples through Monte Carlo [5]

$$\overline{\mathcal{L}}(\phi,\theta,\lambda) = \frac{1}{L}\sum_{l=1}^{L}(\ln p_\theta(\mathbf{x}|\mathbf{z}_\phi^{(l)}) + \ln p_\lambda(\mathbf{z}_\phi^{(l)}) - \ln q_\phi(\mathbf{z}_\phi^{(l)}|\mathbf{x})) \tag{4}$$

A common choice for p_λ (the prior) is the standard normal distribution, i.e. $\mathcal{N}(0, I)$, while the $\ln q_\phi(\mathbf{z}_\phi^{(l)}|\mathbf{x})$ depends on the analyzed kind of data $\mathbf{x}$ (e.g. it may use Bernoulli or Normal distributions, see Section 3).

2.2. VAE with VampPrior

Although in some cases it is sufficient to use the Standard Normal prior, usually resorting to a more complicated prior allows for richer representations in latent space. VampPrior, proposed in (Tomczak & Welling, 2018), is a good example that finds a balance between over-regularization and overfitting. When the prior is too simplistic, like a standard Normal prior, the model is subject to over-regularization, which means that the prior distribution takes over the estimated posterior, which collapses, resulting in a poor representation of the input data. If we consider Equation 3 and decompose the KL-divergence into two pieces, we get (Makhzani et al., 2015):

$$\mathcal{L}(\phi,\theta,\lambda) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}[\log p_\lambda(\mathbf{z})] + \mathbb{H}[q_\phi(\mathbf{z}|\mathbf{x})] \tag{5}$$

When maximizing this function over the available data, it can be shown that a prior equal to the aggregated posterior gives the optimal solution.

$$p_\lambda^*(\mathbf{z}) = \frac{1}{N}\sum_{n=1}^{N} q_\phi(\mathbf{z}|\mathbf{x}_n) \tag{6}$$

where N is the size of the observed dataset. It is quite immediate that, in practice, this result is not convenient to be used for two reasons: the size of the dataset can be very large, leading to expensive computations; also, if the model has wide capacity in terms of parameters, it may overfit the data.

Thus, one can think of sampling a smaller number of training data and compute the aggregated posterior as the sum of a subset of the dataset as an approximation. Although it could work, the less are the samples, the less accurate the prior will become.

The VampPrior is proposed as a solution to this issue since it uses *pseudo-inputs* instead of a subset of the training data.

[5] A sampling method commonly used for approximation (Kalos & Whitlock, 2009)

More specifically the prior will be

$$p_\lambda(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(\mathbf{z}|\mathbf{u}_k) \tag{7}$$

where K is the number of pseudo-inputs, denoted with $\mathbf{u}_k$. The peculiar difference is that the pseudo-inputs are learned, together with ϕ, with backpropagation, constituting additional parameters set. After a proper training phase, the pseudo-inputs would have captured the most important features of the actual data, summarizing in a way the whole data-set in only $K << N$ data-points.

2.3. Hierarchical VAE (HVAE)

HVAE proposes a hierarchical structure for the latent space. Basically, it involves the use of multiple latent variables whose bottom-up dependency to each other is expressed by a graphical model. In this project, we implemented a 2-level hierarchical VAE.

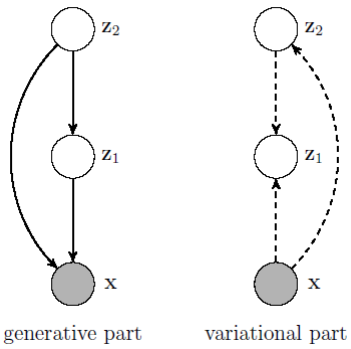


Figure 3: (Tomczak & Welling, 2018)

2.3.1. Inactive units problem

The inactive unit problems refers to the case when lower-level latent variables variational approximation does not process information coming from higher levels and, as a consequence, the original input. This is because the solution for the maximization of the ELBO finds its optimal (by avoiding extra KL costs) such that $q_\phi(\mathbf{z}_1|\mathbf{z}_2) = p(\mathbf{z}_1)$. [6]

In (Tomczak & Welling, 2018), they found out that their HVAE architecture dealt very well the inactive units problem, thus preventing the posterior to be over-regularized by the KL factor.

2.3.2. HVAE mathematical formulation

Firstly, the inference model's distributions dependencies are parameterized (ϕ, ψ) by the first MLP block of the NN (the encoder):

$$q_\phi(\mathbf{z_1}|\mathbf{x}, \mathbf{z_2}) q_\psi(\mathbf{z_2}|\mathbf{x}) \tag{8}$$

The generative model's distributions dependencies are parameterized (θ, λ) by the last MLP block of the NN (the decoder):

$$p_\theta(\mathbf{x}|\mathbf{z_1}, \mathbf{z_2}) p_\lambda(\mathbf{z_1}|\mathbf{z_2}) p(\mathbf{z_2}) \tag{9}$$

3. Experiments

3.1. Setup

The experiments involving the three models described in Section 2 were carried out on the following datasets

- Static MNIST, obtained by the MNIST dataset (a collection of handwritten digits) by setting to 1 pixel whose value was > 0.5 times the maximum value, 0 otherwise.
- Frey Faces, a collection of Brendon Frey faces.
- Fashion MNIST, a set of clothing, a dataset similar to MNIST but more complex due to larger variety in the data.

In the first case, for the binary MNIST dataset, the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ is a Bernoulli distribution, so the output consists only in the mean. Consequently the regularization loss $\ln p_\theta(\mathbf{x}|\mathbf{z})$ is simply the probability of the data given the means returned by the decoder.

With the other two datasets, the decoder returns both a mean and variance: in this case the regularization term is a *log-logistic*(Kissell & Poserina, 2017), a distribution similar to a log-normal having heavier tails and adapted to 256-discrete data.

Furthermore, both an encoder and decoder outputs $\log \sigma^2$ is generally preferred over σ and σ^2 as it reduces numerical instability when computing probabilities of samples being generated by normal distributions and permits faster computation. For the same purpose, all the probability distributions have been defined to be computed directly in log-space.

All layers are made up of 300 gated dense units (each unit returning as output the product of 2 activated neurons) and use a latent space of size 40.

While in the non-hierarchical models encoder and decoder have only a single hidden layer, in the HVAE the encoder is built as a two-step sequence: the first step is to generate parameters and sample for $\mathbf{z}_2$ (4 layers), the second step is to combine the original input and the sampled latent variable to generate upper level latent variable $\mathbf{z}_1$ (6 layers). The decoder slightly differs as two inner decoders are needed: one for $p(\mathbf{z}_1|\mathbf{z}_2)$ (4 layers), and one for $p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2)$ (5 layers).

[6] Proof in (Maaløe et al., 2017)

The number of pseudo-inputs used for the VampPrior is 500 for all dataset.

The mini-batch size is 100. The learning rate used in the Vamp model is 1e-4 while 1e-3 in the VAE and HVAE.

3.2. Results

The performance of the model over a dataset is evaluated on a validation set computing the loss, which corresponds to the negative lower-bound (Equation 3). Unfortunately, due to the limited time and available resources, the quantitative results represent single experiments and not averages over more training sessions. However, it is enough to show how the reconstruction and generative parts of the network behave and to draw conclusions.

3.2.1. Standard VAE

The simplest of the developed models is still able to achieve good results under certain conditions.

Being trained for 300 epochs with the fashion MNIST and 200 with the other two datasets (always using a warmup of 100 epochs) the model can reconstruct the data while capturing the most important features (see Figure 4)

In the generation phase instead the model shows some limitations, especially for MNIST (Figure 5a), while it achieves good results both on the fashion MNIST and Frey datasets (Figure 5c and Figure 5b).

3.2.2. VampPrior VAE

To evaluate the effect of using a VampPrior, we show here the results obtained with the three datasets. The first dataset we analyze is binary MNIST. It corresponds to the easiest task as also reported in Table 1, it reaches a smaller LL concerning the other two datasets. The experiment has been performed by running the training for 200 epochs with a warm-up of 60 epochs and 200 pseudo-inputs.

The number of pseudo-inputs is much less than the size of the dataset (which is 60000) and this allows to have a prior that approximates the aggregated posterior, that in theory would be the optimal prior for maximizing the lower-bound, but in practice will lead to overfitting. The result is that after the training phase, the pseudo-inputs show some representative patterns that describe the dataset features, but do not represent single samples from the dataset (see Figure 7). Although reconstruction is quite good with all the three datasets, we notice some noise in the MNIST reconstruction in Figure 4a, but this is probably due to the binary structure of the first dataset, opposed to the other two that are continuous; more on this in section 4.
Generation results with VampPrior (Figure 8) show better latent representations, in particular concerning the standard VAE, confirming the hypothesis about the richness of the innovative prior.

Dataset	Model		
	standard VAE	VampPrior	HVAE
staticMNIST	-276.18	-544.73	-458.72
Frey Faces	-2344.96	-2279.22	-1975.06
Fashion MNIST	-3190.18	-3348.91	-2765.41

Table 1: Summary table showing the log-likelihood obtained on the test set after training

3.2.3. HVAE

Concerning the reconstruction, HVAE behaved better than the previous models, especially with Frey Faces (Figure 9a).

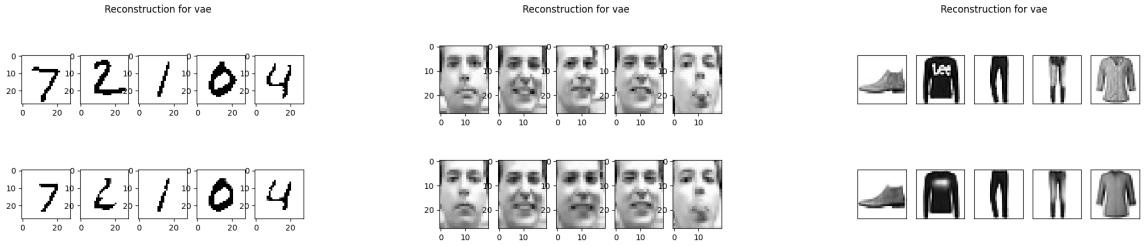
On the other hand, when vectors in the latent space were sampled as input for the generation phase, HVAE produced unintelligible pictures for MNIST (Figure 10a). A plausible cause might be that as the NN is significantly larger, more computational time was needed. However, for Frey Faces (Figure 10b (Figure 10) results were far better than the Standard VAE and Vamp Prior VAE

The number of epochs needed for the MNIST, Frey and Fashion MNIST was, respectively 300 (100 of warmup), 400 (50) and 100 (30).

4. Conclusion

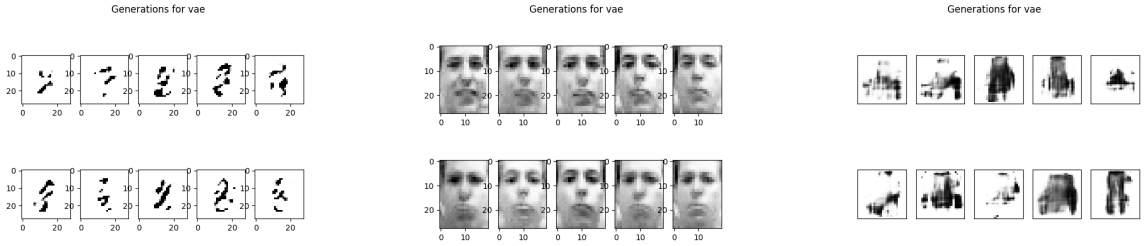
Different metrics are inspected: log-likelihoods (summary available in table 1), time performance, reconstruction and generation pictures.

- By analyzing reconstruction pictures, MNIST dataset had the worst results irrespective of the chosen model. HVAE and Vamp Prior VAE yielded the best results with Frey Faces and Fashion MNIST. This can be probably explained by the fundamental difference that the static MNIST has concerning the other two datasets: it is binary data (where each pixel is either 0 or 1): the input data contains less smooth distributions which are captured with many more difficulties by this model. For example, by having only two discrete values, if the output value is slightly smaller than 0.5 it will not exceed the threshold, resulting in very different output pixel (Figure 6a). Conversely, by having a wider range of discrete output values (like in Frey Faces) the resulting output when in error is not constrained to choose a very distant value (1 when it should be 0, or vice versa), but may assume a closer one (0.6 when it should be 0.4). It is worth noticing that during reconstruction with the Fashion MNIST dataset, some details, such as brand prints, are cut out and only the general piece of



(a) MNIST (b) Frey Faces (c) Fashion MNIST

Figure 4: Reconstruction results with Standard VAE



(a) MNIST (b) Frey Faces (c) Fashion MNIST

Figure 5: Generation results with Standard VAE

clothing is generated (see Figure 6c); this behavior is expected and appreciated since it means that the information retained by the encoder is strictly enough for the task. One might notice a correlation between overfitting and reconstructions showing excessive details (we could apply this reasoning looking at Figures 11a and 4c).

- By analyzing generation pictures, Standard VAE generated the least accurate pictures; this is due to its intrinsically simple prior, which forces heavily the regularization in the latent space. Such results were to be expected considering the theoretical model's implication discussed in subsection 2.1. Vamp Prior VAE turned out to be much better for MNIST, perhaps due to the help of pseudo-inputs which captured relevant patterns.

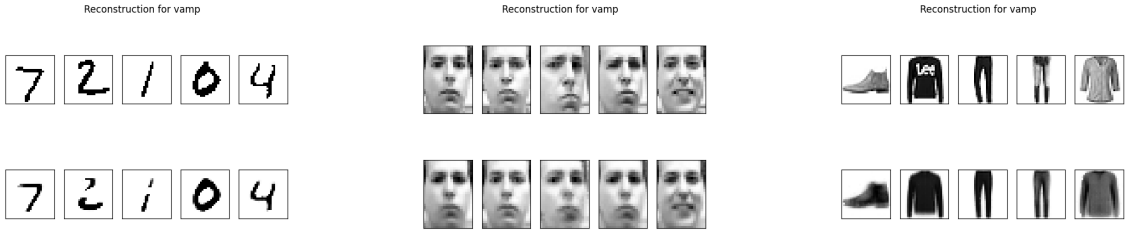
- By analyzing the log-likelihood table one would expect the Standard VAE model to perform better than the VampPrior, but by looking at the result this is not the case. This is because the regularization losses in the 2 cases are different: VampPrior mainly attempts to increase the complexity and richness of the latent space to model more complex data.

- By analyzing time performance, the fastest was the Standard VAE due to its simple network topology. Vamp Prior is slightly slower due to the addition of new layers for Pseudo-inputs; HVAE is the slowest due to the large number of NN layers comprising both the encoder and the decoder, that is due to the requirements of latent variables dependencies which explicitly influences lower-level latent variables.

- Warm-up throughout experiments showed being significant for avoiding too much regularization of the result, which causes the model to be lacking in reconstruction abilities. This might be because the regularization task (with backpropagation starting from the latent encoding) updates the weights in its advantage faster than the reconstruction task, which has to backpropagate from the end of the network: the regularization loss is slowed down so that both contributions become equally relevant.

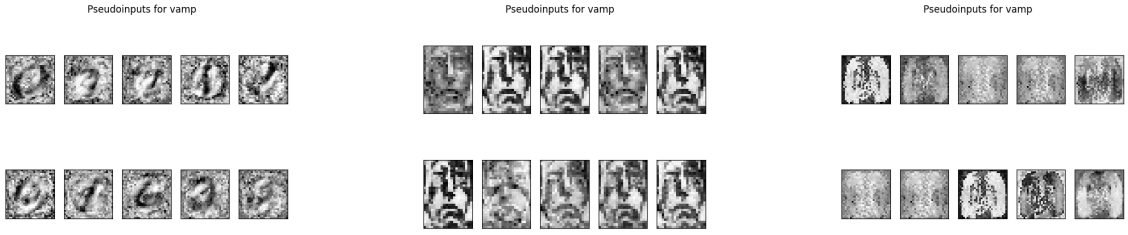
- Regarding the learning rate, we noticed that for a successful training with VampPrior that was to be set to a lower value than with standard Normal prior (by a factor of 10); perhaps this comes from the fact that double backpropagation and updates on the encoder parameters are performed while training (once with actual data and once with pseudo-inputs), resulting in larger updates in one single step concerning the other priors.

- Finally it needs to be noted that obtained loss is different to the original authors' loss (Tomczak & Welling,



(a) MNIST (b) Frey Faces (c) Fashion MNIST

Figure 6: Reconstruction results with VampPrior



(a) MNIST (b) Frey Faces (c) Fashion MNIST

Figure 7: Pseudoinputs learnt by VampPrior

2018) due to different parameters which were required to run the algorithm in a reasonable time, that perhaps were critical, nevertheless, results were comparable. Furthermore, it is important to notice that our implementation differs from one of the authors of the original paper (Tomczak & Welling, 2018) in the choice of the initializers of the networks which, even if maybe negligible on a long run, may still be influential due to the reduced number of epochs of training.

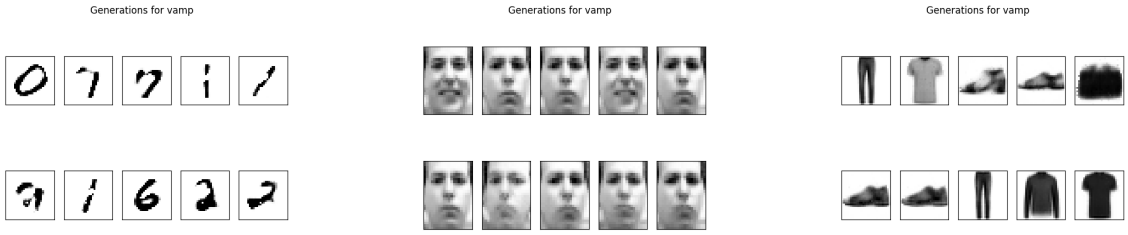
It would be interesting to elaborate on the use of L, as the paper provided few and unclear information. Some tests showed that increasing the value of L did not improve the results; this is probably because lower values of L introduce a random factor that avoids the model being more easily stuck on local minima, and such effect is reduced when increasing L and averaging the result (see Equation 4).

4.0.1. Effect of pseudo-input learning

It is worth to briefly analyze the structure of the pseudo-inputs, shown in subsubsection 3.2.2. As already mentioned they seem to represent some features of the original datapoints, but without preserving the same structure; in other words, they keep some patterns and yet do not copy the samples. This might be due to the cooperation of the simultaneous training of both the encoder and the pseudo-inputs themselves. Moreover, what we see in the pseudo-inputs, is linked with what the encoder is saving from the data since there's no reason for the pseudo-inputs to become useless for the network. This can be easily seen in the case of MNIST, where we recognize noisy clouds around the center which make us realize that the encoder discards those area when encoding the samples.

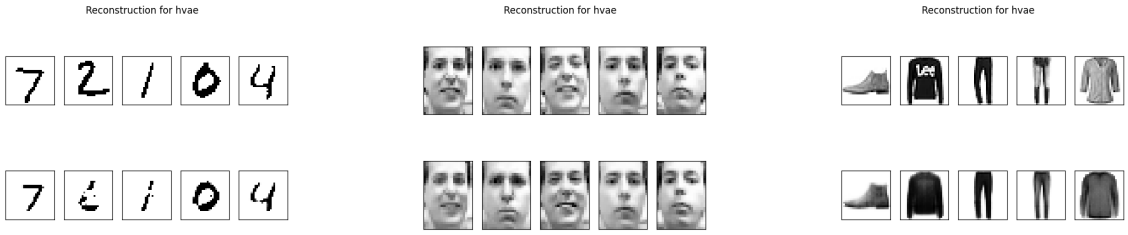
4.0.2. Future developments

This research showed the great potential of variational auto-encoders as generative models. It would be very interesting to compare alternative priors and discriminate which induces the best results in general or for specific problem instances. For example, the Mixture of Gaussian's prior may be critical in developing a model where each mixture component (for example, in the case of MNIST) corresponds to a digit. Even more compelling would be to test how this system performs for much larger datasets, to test the scaling of the system. Additionally testing on different sizes of the latent space could yield critical insights about the general behaviors of this system and more specifically for the inactive units problem (subsubsection 2.3.1). Future development could also provide insights about deeper hierarchical dependencies in HVAE by adding many more levels.



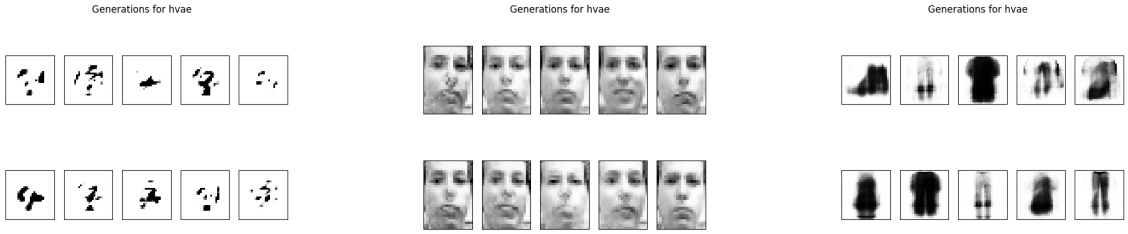
(a) MNIST (b) Frey Faces (c) Fashion MNIST

Figure 8: Generation results with VampPrior



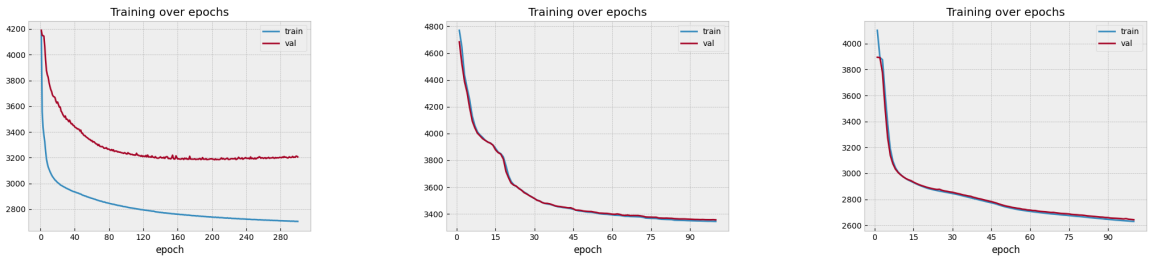
(a) MNIST (b) Frey Faces (c) Fashion MNIST

Figure 9: Reconstruction results with HVAE



(a) MNIST (b) Frey Faces (c) Fashion MNIST

Figure 10: Generation results with HVAE



(a) Standard VAE (b) Vamp Prior VAE (c) HVAE

Figure 11: Losses results for Fashion Mnist

References

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Kalos, M. H. and Whitlock, P. A. *Monte carlo methods*. John Wiley & Sons, 2009.

Kissell, R. and Poserina, J. Chapter 4 - advanced math and statistics. In Kissell, R. and Poserina, J. (eds.), *Optimal Sports Math, Statistics, and Fantasy*, pp. 103 – 135. Academic Press, 2017. ISBN 978-0-12-805163-4. doi: https://doi.org/10.1016/B978-0-12-805163-4.00004-9. URL http://www.sciencedirect.com/science/article/pii/B9780128051634000049.

Maaløe, L., Fraccaro, M., and Winther, O. Semi-supervised generation with cluster-aware generative models. *arXiv preprint arXiv:1704.00637*, 2017.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Noriega, L. Multilayer perceptron tutorial. *School of Computing. Staffordshire University*, 2005.

Rocca, J. Understanding variational autoencoders (vaes), Jul 2020. URL https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

Shlens, J. Notes on kullback-leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*, 2014.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223. PMLR, 2018.