Group Project DD2434

VAE with Vampprior

Nadir Ait Lahmouch - Hamza Benjelloun - Omar Benchekroun

1938 Words .
Keywords: VAE, prior, generative models, VampPrior 15th January, 2021

Abstract Variational Autoencoders (VAEs) are generative models that use variational inference to generate new content and achieve efficient dimensionality reduction. In this report, we will take a close look at VAEs, we will also experiment with different priors (standard, Mixture of Gaussians, and Vampprior) and different types of layers in the architecture, and finally compare the results on 4 datasets : staticMNIST, Omniglot, Caltech 101 and FashionMNIST.

1 Introduction

Generative models have gained significant traction in last years, the two main families of generative models being Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). Variational autoencoders achieve dimensionality reduction through neural network training, while avoiding overfitting and yielding an exploitable representation of the inputs in a smaller space.

Choosing an appropriate prior for a VAE is crucial, as it usually suffers from *over-regularization* (see 1.1.2 for more details) when using the standard normal prior. Various state-of-the-art approaches tried to overcome this issue, we will motivate the use of both the *Mixture of Gaussians* prior (MoG) and the *Variational Mixture of Posteriors* prior (VampPrior[1]) and see why they represent a good tradeoff between the quality of the generative ability and computational complexity.

1.1 Variational Autoencoders (VAEs)

1.1.1 "Vanilla" Autoencoders

To build up to the structure of a VAE, we start by defining the architecture of a what is commonly called a vanilla autoencoder, which is composed of an encoder and decoder neural networks (see fig.1). This autoencoder is usually a feed forward neural network and is trained using backpropagation to optimize the encoding-decoding scheme. The latent variable $\mathbf{z}$, lies in a smaller space (dimensionality-wise) compared to $\mathbf{x}$, this space will be referred to later as the *latent space*.

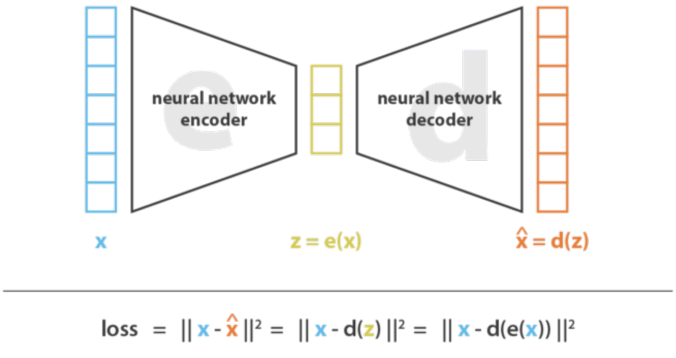

Fig. 1. Architecture of an autoencoder[2]. The loss, or reconstruction error is minimized by gradient descent over the parameters of the neural networks.

A simple autoencoder presents major limitations, mainly *overfitting* and an *under-regularized* latent space. That is because in order to minimize the reconstruction error, the neural nets need to be deeper in order to achieve the desired dimensionality reduction. The resulting latent space, achieving minimal error for the given entries (0 if the neural nets are deep enough), is

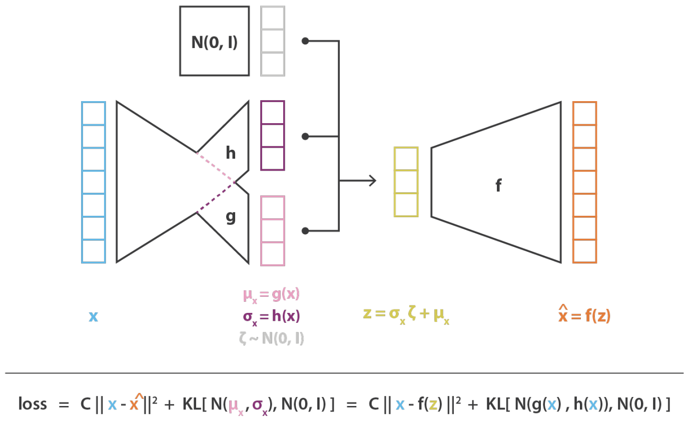
N(0, I)
h
g
f
x
μx = g(x)
σx = h(x)
ζ ~ N(0, I)
z = σx ζ + μx
x̂ = f(z)
loss = C || x - x̂ ||² + KL[N(μx , σx), N(0, I)] = C || x - f(z) ||² + KL[N(g(x) , h(x)), N(0, I)]

Fig. 2. Architecture of a variational autoencoder[2] using the reparametrization trick, the prior is a standard normal distribution.

however said to be lacking regularity and interpretability. Sampling a random point from this latent space and decoding it would give meaningless results.
Conclusion : It is necessary to find a trade-off between minimizing the reconstruction error and having a regularized latent space.

1.1.2 Variational Autoencoders (VAEs)

Variational autoencoders rely on variational inference to avoid the problem of overfitting and having an under-regularized latent space, i.e. ensuring the generative ability. The encoding-decoding scheme is modified : we encode an input as a distribution over the latent space, instead of a point in the latent space. We then sample a point from this distribution to decode, compute the reconstruction error and use backpropagation through the network.

The loss in this case is a lower bound (ELBO) of the marginal likelihood. It has two components: The reconstruction error and the regularization term using the KL divergence, the latter is used to drive the encoder to match the prior $p(\mathbf{z})$ (see fig.2). Additionally, to be able to use backpropagation through the network even though we're sampling **z**, we use the *reparametrisation trick*[3], as show in fig.2.

The choice of the prior $p(\mathbf{z})$ is important, and is the subject of our experiments. It turns out that using a standard normal distribution as a prior tends to result in an over-regularized VAE with few active latent dimensions[4].

As seen in Tomczak's paper[1] we have :

$$\begin{aligned}\mathbf{E}_{x\sim q(x)}(\ln(p(x)) \geq \mathbf{E}_{x\sim q(x)}[\mathbf{E}_{q_\Phi(z/x)}[\ln(p_\theta(x/z))]]\\ + \mathbf{E}_{x\sim q(x)}[\mathbf{H}(q_\Phi(z/x))]\\ - \mathbf{E}_{z\sim q(z)}[-ln(p_\lambda(z))]\end{aligned}$$

Then to find the optimal prior that maximize the average marginal loglikelihood we should maximize the following Lagrange function with the Lagrange multiplier β :

$$\max_{p_\lambda(z)} -\mathbf{E}_{z\sim q(z)}[-\ln(p_\lambda(z)) + \beta(\int p_\lambda(z)dz - 1)]$$

As mentioned in the Vampprior paper[1] the solution of this problem is simply the aggregated posterior: $p_\lambda(z) = \frac{1}{N}\sum_{k=1}^{N} q_\Phi(z/x_n)$ where N is the number of data points and q_Φ is the s the variational posterior(the encoder). As we see the optimal prior formula is computationally expensive since we sum over all training point. Two more appropriate alternatives exist:

- Mixture of Gaussians (MoG) : where $p(\mathbf{z}) = \frac{1}{K}\sum_{i=1}^{K}\mathcal{N}(\mu_k, diag(\sigma_k^2))$, these hyperparameters are trained by backpropagation.
- The *Variational Mixture of Posteriors* Prior (Vampprior): where $p(\mathbf{z}) = \frac{1}{K}\sum_{i=1}^{K} q_\Phi(z/\mu_k)$ as referred to in the VampPrior paper[1], q_Φ is the variational posterior and μ_k is a D-dimensional vector we refer to as a pseudo-input learned also through backpropagation.

2 Methods and setup

In this experiment we aim at verifying empirically whether the MoG helps the VAE to train a representation that better reflects variations in data than the standard normal. Our implementation of the VampPrior[1] was not effective enough to include its results in this paper, the model utilizing it only worked when we entered a single pseudo-input. But as seen in the introduction, the MoG is a good alternative that shouldn't give results that far from the VampPrior[1].

Latent variables are represented with 16 stochastic hidden units, which is a common size used in VAEs. In the learning phase we use ADAM algorithm and normalized gradients with a learning rate of 0.001, and a batch of size 32 which is here again a popular size often used in deep learning models. We did not take a large batch as mentioned in the original article[1] (batch_size = 100) because the training was taking a long time at each run, so we had to simplify the model a notch. Finally, the number of the mixture components for our MoG model is 10.

We conducted the experiment on 3 among 6 datasets given by the author of the original article[1] which are MNIST[5], OMNIGLOT[6] and Caltech 101[7] and to strengthen our results we added another dataset called FashionMNIST[8]. The non binary datasets (i.e. FashionMNIST, Omniglot) were binarized. We also applied the Bernoulli distribution on all of them.

3 Results

3.1 Quantitative results

Like in the original paper[1], we quantitatively evaluate our implementation using the test marginal log-likelihood. Additionally, we added the **loss** even if they're related. You can see in

Table 1 a comparison between models with the standard prior and the MoG for the previously named four datasets.

We can from the start notice that the use of MoG is outperforming the VAE with only the standard prior. Indeed, the values of Log-likelihood (LL) and the loss are respectively higher and lower in all the tests (the best performance is in bold for each dataset, see Table 1). Furthermore, we notice that the results given by the Convolutional model are largely better than the MLP one, outperforming it in each run as well.

Now if we compare our results to the results of the original paper[1], we find some differences. As already discussed before, we didn't compute the results for a VampPrior[1] model. However, the MoG solution is an alternative approximating the optimal solution as well so comparing it with the paper's results on the VampPrior[1] is feasible. In Figure 3, we present a comparison between the results of using the standard prior in our implementation versus the original paper's, and a comparison between our MoG and their VampPrior's[1] solution in the second Figure 4 (we only compared our models including a Convolutional network as they perform better). We can see that the results are slightly off. In the MNIST data for example, our standard gaussian model yields a -76.24 in the likelihood while theirs gives -82.41. Another example is the Omniglot dataset where this time, their model is better with -97.65 versus -102.06 in ours. We think that this difference can be due to multiple reasons:

- The structure of the network is different, as we can't perfectly land the exact same model just from reading the paper.
- Some hyperparameters are different. For example the batch size (32 vs 100[1]), or the "activation function", we are using the "leaky relu" activation function in our CNN model.
- To compute our Log-likelihood we take a number of samples to use in the encoding and then decode them. The Loglikelihood is then the mean of these values, $LL = E_{z \sim P}(p(x|z))$. Therefore, changing the parameter "number of samples" can modify the results. In our implementation, it is equal to 16 to have a good visualisation of the generated images at each step.

Moreover, as done in the paper[1], we tested whether increasing the number of components in the Mixture of Gaussians ameliorates the results or not. And to do that, we plotted the log-likelihood for multiple values of "the number of the mixture components", on the staticMNIST data, and the result in is Figure 5. We can see that we reach the same result in the original paper[1]: considering more components doesn't seem to improve the results as the likelihood is fluctuating.

log-likelihood/loss				
	MLP		Conv	
Dataset	Standard	MoG	Standard	MoG
staticMNIST	-104.21/118.85	**-102.97/117.98**	-76.24/99.75	**-73.59/97.60**
Omniglot	-140.61/149.18	**-140.19/149.15**	-102.06/124.87	**-101.45/124.41**
Caltech 101	-131.14/150.92	**-130.72/150.04**	-97.93/127.45	**-93.24/123.21**
FashionMNIST	-132.19/146.83	**-131.98/145.67**	-104.83/127.74	**-103.19/125.59**

Table 1
Test log-likelihood (LL) and loss between different models with the standard normal prior (standard) and the Mixture of Gaussian (MoG).

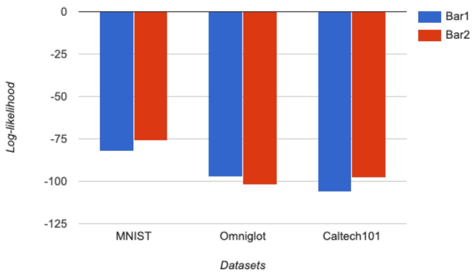

Fig. 3. Comparison of the LL between our results (bar2) and the paper's results (bar1) for model using standard prior.

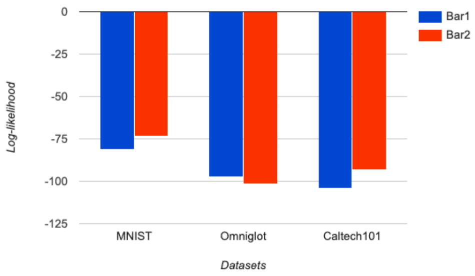

Fig. 4. Comparison of the LL between our results (bar2, MoG) and the paper's results (bar1, VampPrior).

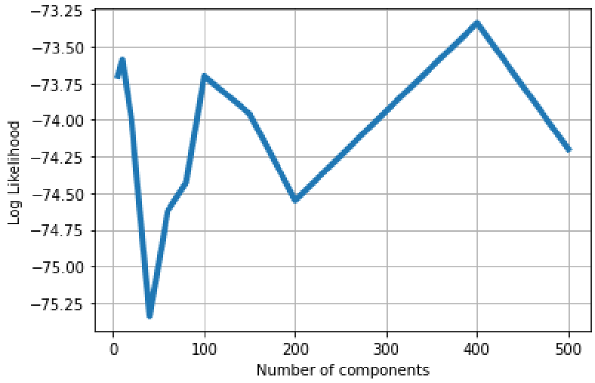

Fig. 5. VAE with MoG with varying the number of the mixture components for the staticMNIST data.

3.2 Qualitative results

We generated images after each run for the model using a MoG mixture with 10 components. We did the same with the standard prior but the images weren't as sharp as the MoG ones. You can see these images in Figure 6.

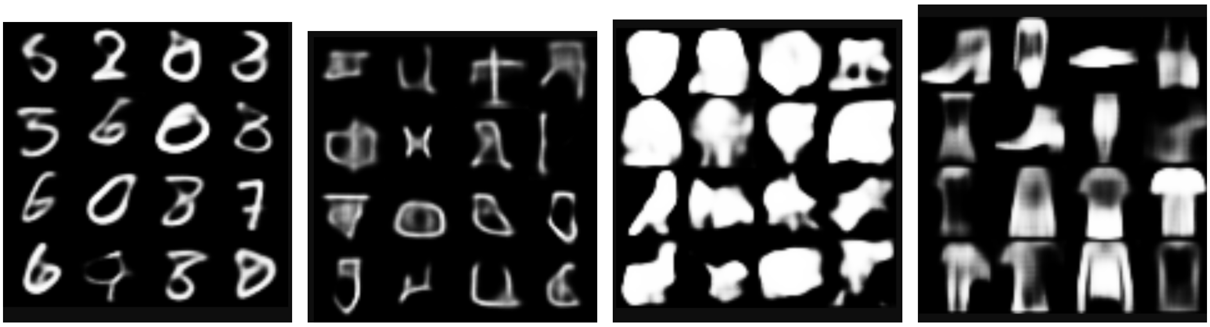

Fig. 6. VAE with MoG with varying the number of the mixture components for the staticMNIST data. From left to right, MNIST, Omniglot, Caltech101, FashionMNIST.

4 Discussion

Overall, we can say that the MoG is a better approximation than the standard normal prior. So we think that the application of the VampPrior[1] (as proven in the original paper) will yield better performances of the VAE in general. And this latter is both qualitative and quantitative. A VAE using the standard prior generates blurry images often hard to interpret (especially for MNIST and Omniglot), while the MoG one gives sharper images (see Figure 6) and so does the VampPrior one[1].

The choice of MoG as a prior was an intuitive extension of standard normal prior. With MoG we segregate the latent space to distinct classes. It is like if we suppose that our observed data is inferred from a mixture of Gaussian. Inferring the class of a data point is equivalent to inferring which mode of the latent distribution the data point was generated from. The idea behind MoG was to assess whether adding more Gaussian singles will improve the loss. But after the experiments, we notice that both the log-likelihood and the loss were improved.

Let's start with the positive side of the work carried out by the two researchers in their article[1]. Theoretically it appears that VampPrior[1] outperforms both standard normal and MoG prior[1]. Additionally, the advantages of this method over MoG are that we reduce the number of parameters required to learn by our model[1], which can reduce the execution time. Furthermore, the prior and posterior will cooperate during the training[1].

Now for the relevance of the discoveries in this paper, it's hard to tell whether it's ground breaking or not. First of all, it's still recent (2018) and we didn't find any relevant papers taking it as a base to build on it more effective VAE models. Furthermore, there have been some recent discoveries on some new priors constructions yielding good results. For instance, Goyal et al.[9] learn a tree structured non-parametric Bayesian prior for capturing the hierarchy of semantics presented in the data or Hui-po wang, wen-hsiao peng and wei-jan ko who propose the notion of code generators for learning a prior from data for AAE[10]. The fact is, research in image generation is so lucrative and a lot of big labs are racing to find the most effective models to generate images. GANs (Generative Adversial Networks) are for now considered for the state-of-the-art algorithms when it comes to image generation, e.g. BigGan of DeepMind.

And last year, they even announced VQ-VAE 2 who is supposedly yielding competitive results with GAN models. That is to say, it's hard to position Tomczak and Welling VampPriors inside all these powerful methods or maybe we just have to wait for further searches based on it.

5 Conclusion

In this article we have theoretically proved that the prior plays a crucial role in the improvement of the VAE. We saw that theoretically, the VampPrior outperforms both standard normal and MoG as it approximates the best the optimal solution maximizing the ELBO. But we only empirically compared the MoG and the classical standard normal prior. The result was that generations and reconstructions obtained from the MoG VAE are of better quality than the ones achieved by the standard VAE. We therefore claimed that the VampPrior does the same as well and even better, as already proven by Tomczak and Welling in their paper[1].

6 References

1. Tomczak, J. M. & Welling, M. VAE with a VampPrior (February 2018).
2. Rocca, J. *Understanding Variational Autoencoders (VAEs)* July 2020.
3. Kingma, D. P. & Welling, M. An Introduction to Variational Autoencoders (2019).
4. Burda, Y., Grosse, R. & Salakhutdinov, R. *Importance Weighted Autoencoders* 2016. arXiv: 1509.00519 [cs.LG].
5. *MNIST dataset of handwritten digits* http://yann.lecun.com/exdb/mnist/.
6. *OMNIGLOT dataset* https://github.com/yburda/iwae/blob/master/datasets/OMNIGLOT/chardata.mat.
7. *Caltech 101 dataset* https://people.cs.umass.edu/~marlin/data/caltech101_silhouettes_28_split1.mat.
8. *FashionMNIST dataset* https://github.com/zalandoresearch/fashion-mnist.
9. Goyal, P., Hu, Z., Liang, X., Wang, C. & Xing, E. Nonparametric variational auto-encoders for hierarchical representation learning. *Proc.IEEE Int. Conf. Computer Vision (ICCV* (2017).
10. Hui-po wang, wen-hsiao peng & wei-jan ko. Learning priors for adversarial autoencoders (May 2019).