Improving Variational Autoencoders with residual blocks

Piyush Jena
Columbia University
New York, NY

***Abstract*—Generative versus discriminative modeling is a major topic that divides machine learning. While discriminative modeling aims to learn a predictor based on the observations, generative modeling aims to solve the more general problem of learning a joint distribution over all the variables. It has been shown that VAE suffers from smoothening. This paper will attempt to solve it using skip connections and residual blocks.**

I. Introduction

There are two models facing neck to neck in the field of data generation: Generative Adversarial Network (GAN) [1] and Variational Autoencoder (VAE) [2]. These two models have different takes on how the models are trained. GAN is rooted in game theory, its objective is to find the Nash Equilibrium between discriminator and generator networks. VAE is rooted in Bayesian inference, i.e. it attempts to model the underlying probability distribution of data so that it could sample new data from that distribution. It attempts simulating how the data is generated in the real world.

The main reason generative modeling is attractive is because it allows visualization of the generation of the data. The process first involves using neural networks to reduce the dimensionality of the data. Then leverage the neural network to learn the means and variances of the mixture of gaussian model from which the dataset is drawn. Then use the generative model to create new samples.

As described in the NVAE paper [5], most of the research efforts on improving VAEs is dedicated to tackling the statistical challenges, such as reducing the gap between approximate and true posterior distribution, formulating tighter bounds, reducing the gradient noise or tackling posterior collapse. The role of neural network architectures for VAEs have been somewhat overlooked.

It has been shown that Variational Autoencoders (VAE) aggressively smoothens the output. The problem is alleviated by increasing the dimensionality of the latent variable. Long et al [3] suggests using skip connections in CNNs to store higher order features like edges and shapes for semantic segmentation. Here, we introduce skip connections in the inference part of the network to learn higher order features which are removed by decreasing filter size. We will also experiment using residual blocks and measure the performance.

II. Implementation Details

The implementation would involve first experimenting with the standard architectures. Most of the research have used VGG16 [4] or simpler models as the encoder. For this task we will test vanilla VAE [12] and a RESNET-18 based VAE. Next change that will be experimented is Batch Normalization. Traditionally, VAEs have avoided using Batch Normalization due to the introduction of additional noise. We will compare the 3 networks and show the results of the comparison study in the paper.

A. Datasets

- MNIST dataset [7] - This dataset is much simpler to model and will allow easy validation of the network.
- CIFAR10 dataset [8] - This dataset is more complex than MNIST and allow better visualization of the performance of the architecture. With the available hardware this dataset is relatively easy to train and test.
- ImageNet dataset [9] - A very popular dataset used for image classification
- CelebA 64 dataset [10,11] - This is the most popular dataset used to showcase the results of Variational Autoencoders. The more complex dataset will also allow us to understand how well higher order features are accounted for in the model. Due to limitations of time and computing power, we will be skipping training on the ImageNet and CelebA Dataset.

B. Expected Output

The expectation is that residual blocks and skip connections will allow the encoder network to focus on higher order features. Latent variables dedicated to this information will allow generation of sharper images. We will use Mean Squared Error to compare the described models with the above datasets.

III. Theory

Here, we try to understand the strategy to develop an algorithm to estimate the probability distribution of a dataset.

A. KL Divergence

In statistics, the Kullback-Leibler divergence denoted by $D_{KL}(P||Q)$, is a type of statistical distance: a measure of how one probability distribution (P) is different from a second (Q). While it is a distance, it is not a metric - it is not symmetric for any 2 distributions and does not satisfy triangle inequality. In our case, P is the true probability distribution and Q is the estimated probability distribution. The definition for KL Divergence between two distributions P,Q is

$$D_{KL}(Q||P) = \sum Q log \frac{Q}{P}$$

As evident, our objective is to minimize this difference and hence the KL Divergence.

B. ELBO

The outcome of the following discussion is a loss function called ELBO (Evidence Lower Bound). It comprises of two terms: Reconstruction and KL Divergence. Together, these two terms define the ELBO:

$$ELBO = E_q logp(x|z) + E_q log \frac{q(z|x)}{p(z)}$$

Minimizing KL divergence is equivalent to maximizing ELBO (with some slack). This means that ELBO is a lowerbound on KL divergence. Remember, we want to model a distribution over inputs, $p(x)$. We assume that there are factors that influence x but that we can't observe. We call those as latent variables. Lets assume z is the latent variable. $p(x, z)/p(x) = p(z|x)$. To solve this we define $q(z|x)$ to approximate $p(z|x)$. We treat this as an optimization problem which can be solved by parameterizing q using a neural network. This is called variational inference.

Now we try ELBO derivation from $KL(Q||P)$

$$\begin{aligned}
D_{KL}(q(z|x)||p(z|x)) &= \sum q(z|x) log \frac{q(z|x)}{p(z|x)} \\
&= -\sum q(z|x) log \frac{p(z|x)}{q(z|x)} \\
&= -\sum q(z|x)[logp(z|x) - logq(z|x)] \\
&= -\sum q(z|x)[log \frac{p(x|z)p(z)}{p(x)} - logq(z|x)] \\
&= -\sum q(z|x)[logp(x|z) + logp(z) - logp(x) - logq(z|x)] \\
&= \sum q(z|x)[-logp(x|z) - logp(z) + logp(x) + logq(z|x)]
\end{aligned}$$

Rearranging the terms, we get

$$D_{KL}(q(z|x)||p(z|x)) - \sum q(z|x)[-logp(x|z) - logp(z) + logq(z|x)] = logp(x)$$

$$D_{KL}(q(z|x)||p(z|x)) + [E_q logp(x|z) - E_q log \frac{q(z|x)}{p(z)}] = logp(x) \tag{1}$$

Since, $logp(x)$ is a constant, and we want to minimize the D_{KL} term, we can achieve the same by maximizing the ELBO. We analytically compute the derivative of the first part and Monte Carlo estimate the 2nd part. Now that we can compute the full loss through a sequence of differentiable operations, we can use any gradient-based optimization technique to maximize the ELBO.

IV. Implementation and Results

A. Basic VAE

The most basic version of VAE was created with an encoder block with 2 3x3 convolution layers with 32 and 64 filters with padding = 1 respectively, each with stride 2, followed by a fully connected layer and a sampling layer. The decoder block had 3 transposed convolution layers with filter counts 64, 32, 3 followed by a sigmoid layer.

The training time was low - taking 10 minutes to complete 30 epochs of training on CIFAR dataset. The performance was great on MNIST dataset, with 1 epoch of training enough to show decent results. For CIFAR dataset however, it was obvious that the model capacity had been reached and all the outputs were extremely blurry.

Fig. 1. Generated images from Basic VAE after 30 epochs

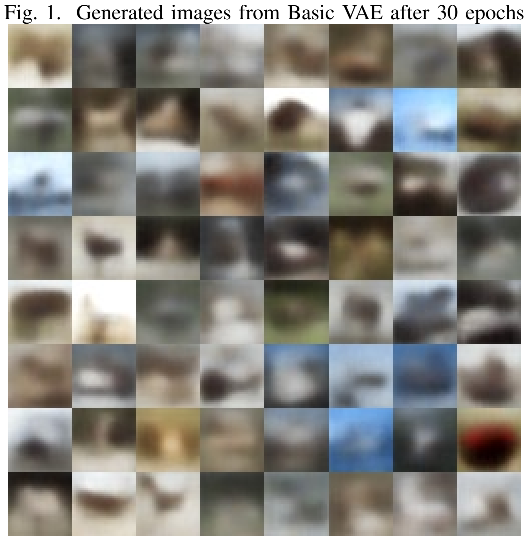
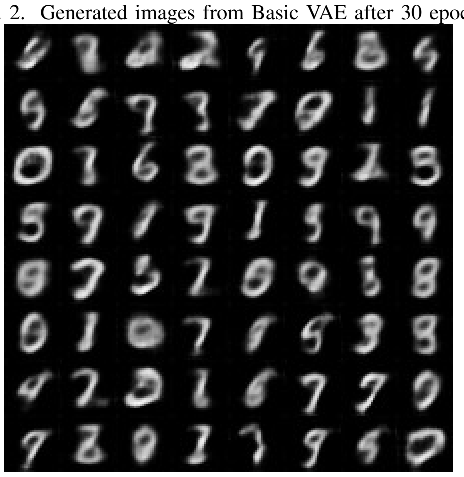


Fig. 2. Generated images from Basic VAE after 30 epochs

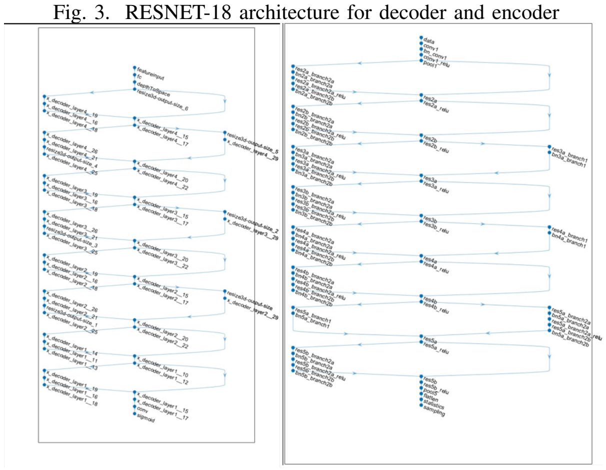


B. RESNET-18 VAE

1) Encoder Architecture: The encoder architecture uses a RESNET-18 architecture with the final fully connected layer size of 2*(latent variables count). A sampling layer was added that will generate the mean and variance of the distribution. We use 2 implementations of Encoder i.e. with and without batch normalization.

2) Decoder Architecture: The decoder architecture uses a reverse of the encoder with interpolation layers in between to increase the size of the output. The final layer in the decoder uses a sigmoid or tanh activation function.

Fig. 3. RESNET-18 architecture for decoder and encoder



The training time was high - especially in MATLAB despite using GPU. Also, despite using the same settings as pytorch the convergence was slow making it seem like it was stuck in a local minima. We preprocessed the data to be centered around 0 and changed the activation function to tanh. This significantly helped in convergence. On pytorch however we got outstanding results with hazy but identifiable figures. VAEs tendency to low pass filter the expected output is very well documented. We also attempted the use of RESNET-50 but we didn't receive any significant improvement in the output indicating that the model capacity was yet to be reached.

Fig. 4. RESNET-18 output with batchnorm and tanh for CIFAR after 30 epochs

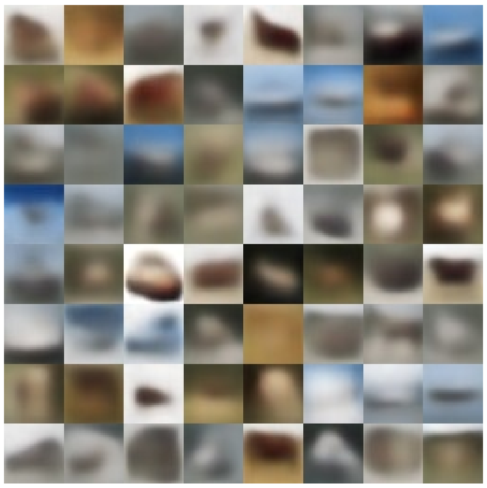


Fig. 5. RESNET-18 output without batchnorm and tanh for CIFAR after 30 epochs

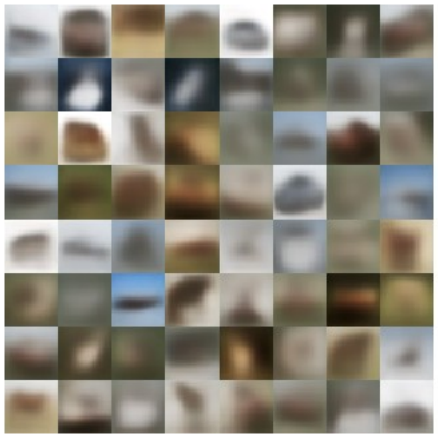
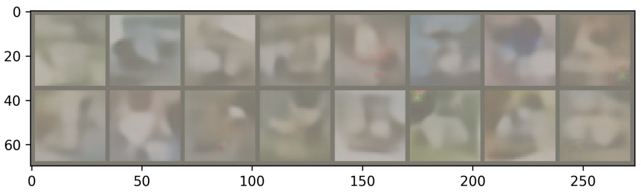


Fig. 6. RESNET-50 output for CIFAR after 30 epochs



V. Conclusion

From the experiments we found out that increasing model capacity using deeper networks allowed better generative models. However, using RESNETs didn't solve the problem of low pass filtering in VAEs significantly. This is the reason deep learning scientists have preferred GANs over VAEs as the gold standard for generative models. We found that preprocessing the data (for example - centering around 0) did help in convergence. Better pre-processing steps may help in the generating better outputs. We also found out that removing batch normalization layers also helped in training time and a lower loss value.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144.

[2] Kingma, D. P., Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[3] Long, J., Shelhamer, E., Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

[4] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[5] Vahdat, A., Kautz, J. (2020). NVAE: A deep hierarchical variational autoencoder. Advances in Neural Information Processing Systems, 33, 19667-19679.

[6] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[7] Yann LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

[8] Alex Krizhevsky et al. Learning multiple layers of features from tiny images, 2009.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition (CVPR), 2009.

[10] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

[11] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In International conference on machine learning. PMLR, 2016.