Generative Variational Autoencoder For High Resolution Image Synthesis

Getting the best of both GANs and VAEs

Photo by Manuel Velasquez on Unsplash

This article presents our research on high resolution image generation using Generative Variational Autoencoder.

Important Points

1. Our work addresses the mode collapse issue of GANs and blurred images generated using VAEs in a single model architecture.
2. We use the encoder of VAE as it is while replacing the decoder with a discriminator.
3. The encoder is fed data from a normal distribution while the generator is fed that from a gaussian distribution.
4. The combination from both is then fed to a discriminator which tells whether the generated images are correct or not.
5. We evaluate our network on 3 different datasets: MNIST, CelebA-HQ and LSUN dataset.
6. We outperform previous state-of-the-art methods in terms of MMD, SSIM, log likelihood, reconstruction error, ELBO and KL divergence as the evaluation metrics.

Introduction

The training of deep neural networks requires hundreds or even thousands of images. Lack of labelled datasets especially for medical images often hinders the progress. Hence it becomes imperative to create additional training data. Another area which is actively researched is using generative adversarial networks for image generation. Using this technique, new images can be generated by training on the existing images present in the dataset. The new images are realistic but different from the original data. There are two main approaches of using data augmentation using GANs: image to image translation and sampling from random distribution. The main challenge with GANs is the mode collapse problem i.e. the generated images are quite similar to each other and there is not enough variety in the images generated.

Another approach for image generation uses Variational Autoencoders. This architecture contains an encoder which is also known as generative network which takes a latent encoding as input and outputs the parameters for a conditional distribution of the observation. The decoder is also known as an inference network which takes as input an observation and outputs a set of parameters for the conditional distribution of the latent representation. During training VAEs use a concept known as reparameterization trick, in which sampling is done from a gaussian distribution. The main challenge with VAEs is that they are not able to generate sharp images.

Dataset

The following datasets are used for training and evaluation:

1. MNIST—This is a large dataset of handwritten digits which has been used successfully for training image classification and image processing algorithms. It contains 60,000 training images and 10,000 test images.
2. LSUN dataset—This dataset contains millions of color images with 10 scene categories and 20 object categories. This is one of the most common datasets for training and testing GAN based neural networks.
3. CelebA-HQ dataset -This is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. This is also one of the most

common datasets for training and testing GAN based neural networks.

VAE vs Ours Network

We show how instead of inference made in the way shown in original VAE architecture, we can add the error vector to the original data and multiply by standard distribution. The new term goes to the encoder and gets converted to the latent space. In the decoder, similarly the error vector gets added to the latent vector and multiplied by standard deviation. In this manner, we use the encoder of VAE in a manner similar to that in the original VAE. While we replace the decoder with a discriminator and hence change the loss function accordingly. The comparison between model architectures of VAE and our architecture is shown in Fig 1.

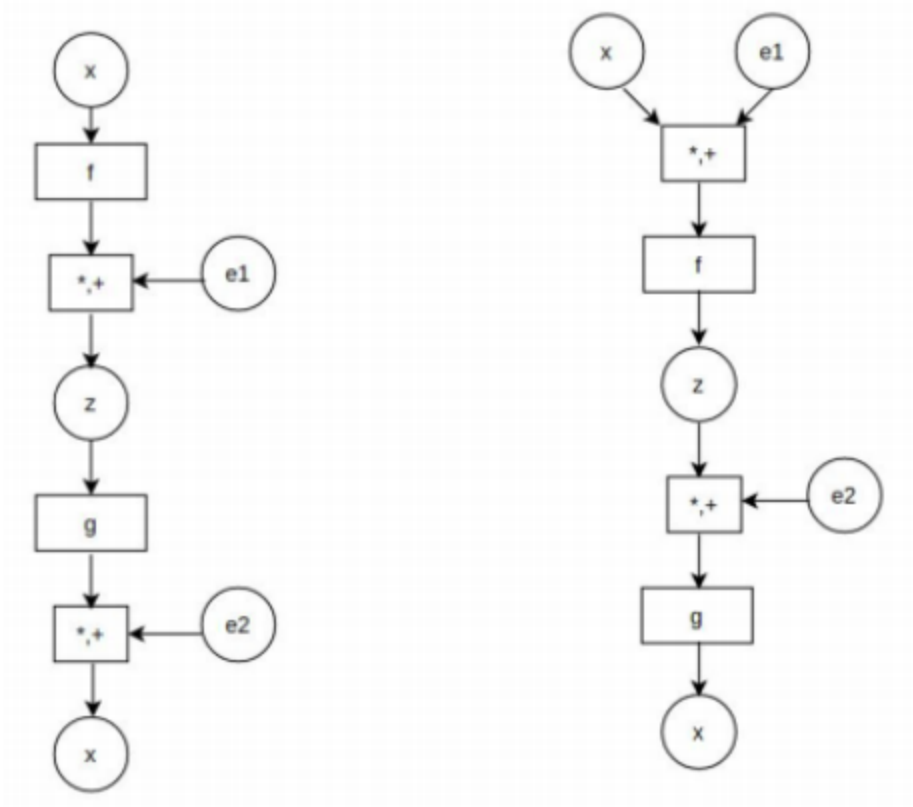

Figure 1: Comparison between standard VAE and our network where e1 and e2 denote samples from some noise distribution, x denotes image vector, z denotes latent space vector, f and g denotes encoder and decoder functions respectively and +, ∗ denotes addition and concat operators.

Our architecture can be seen both as an extension of VAE as well as that of GAN. Reasoning it as the former is easy as this requires a change in loss function for decoder, while the latter can be made by recalling the

fact that GAN essentially works on the concept of zero sum game maintaining Nash Equilibrium between the generator and discriminator. In our case, both the encoder from VAE and discriminator from GAN are playing zero sum game and are competing with each other. As the training proceeds, the loss decreases in both the cases until it stabilizes.

Network Architecture

The network architecture used in this work is explained in the below points:

1. The discriminator and encoder networks have four convolution layers, each of which uses 3×3 filters.
2. We use Batch Normalization and Leaky Rectified Linear Unit (LeakyReLU) layers after each layer.
3. In training, we found that our architecture suffers from instability during training. This was solved using WGAN loss function which measures Wasserstein distance between two distributions.
4. We used the gradient penalty term to stabilize the training.
5. Our loss function has a total for 3 terms. While training, the encoder and the generator are considered as one network. Thus, we sum up the loss functions of the two networks in the order encoder-generator, discriminator as one and train the networks.
6. Two latent vectors are sampled one from normal distribution and the other from gaussian distribution. The one from normal distribution is fed to the encoder while the one from gaussian distribution is fed to the generator.
7. The outputs from both the vectors are in turn fed to the discriminator to tell whether the generated image is real or not.

Our network architecture is shown in Fig 2.

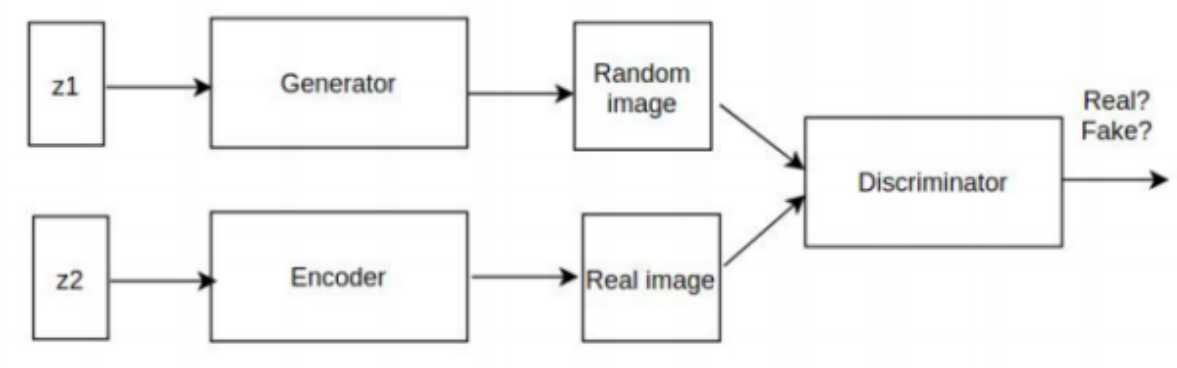

Figure 2: Our network architecture

Architecture Details

The generator and discriminator layerwise architecture details is shown in Table 1 and Table 2 respectively. We denoted ResNet block as consisting of the following layers—convolutional, max pooling layer, 30 percent dropouts in between the layers and batch normalization layer.

Table 1: Generator architecture details

Layer Output	Size	Filter
Fully Connected	$256 \times 32 \times 32$	$512 \rightarrow 256 \times 32 \times 32$
ResNet Block	$256 \times 32 \times 32$	$512 \rightarrow 256 \rightarrow 256$
ResNet Block	$256 \times 32 \times 32$	$256 \rightarrow 256 \rightarrow 256$
Upsampling	$256 \times 64 \times 64$	-
ResNet Block	$128 \times 64 \times 64$	$256 \rightarrow 128 \rightarrow 128$
ResNet Block	$128 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 128$
Upsampling	$128 \times 128 \times 128$	-
ResNet Block	$64 \times 128 \times 128$	$128 \rightarrow 64 \rightarrow 64$
ResNet Block	$64 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 64$
Conv2D	$3 \times 128 \times 128$	$64 \rightarrow 3$

Table 2: Discriminator architecture details

Layer Output	Size	Filter
Conv2D	$64 \times 128 \times 128$	$3 \rightarrow 64$
ResNet Block	$64 \times 128 \times$	$128\ 64 \rightarrow 64 \rightarrow 64$
ResNet Block	$128 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 128$
AvgPool2D	$128 \times 64 \times 64$	-
ResNet Block	$128 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 128$
ResNet Block	$256 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 256$
AvgPool2D	$256 \times 32 \times 32$	-
Fully Connected	$256 \times 32 \times 32$	$256 \times 32 \times 32 \rightarrow 1000$

Algorithm

The algorithm used in this work is trained using Stochastic Gradient Descent (SGD) as shown below:

Algorithm 1: Generative Variational Autoencoder (GVA)

i = 0

while *not converged* **do**

Sample $\{x^{(1)}, \ldots, x^{(m)}\}$ from data distribution $p_{\mathcal{D}}(x)$

Sample $\{z^{(1)}, \ldots, z^{(m)}\}$ from prior $p(z)$

Sample $\{\epsilon^{(1)}, \ldots, \epsilon^{(m)}\}$ from $\mathcal{N}(0, 1)$

$g_\theta \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\theta \log p_\theta \left(x^{(k)} | z_\phi \left(x^{(k)}, \epsilon^{(k)}\right)\right)$

$g_\phi \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\phi \log p_\phi \left(x^{(k)} | z_\theta \left(x^{(k)}, \epsilon^{(k)}\right)\right)$

Perform SGD-updates for θ, ϕ

$i = i + 1$

end

Experiments

All the generated samples are generator outputs from random latent vectors. We normalize all data into the range [-1, 1] and use two evaluation metrics to measure the performance of our network. First of them measures the distribution distance between the real and generated samples with maximum mean discrepancy (MMD) scores. The second metric evaluates the generation diversity with multi-scale structural similarity metric (MS-SSIM). Table 4. compares MMD and MS-SSIM scores with previous state of the art architectures.

Table 4: Quantitative results on MNIST

Architecture	MMD × 0.0001	MS-SSIM
WGAN-GP (Gulrajani et al., 2017)	0.327	0.996
VAE-GAN (Larsen et al., 2016)	0.075	0.972
α-GAN (Lutz et al., 2018)	0.131	0.843
Ours	0.068	0.818

We noticed the model with a small latent vector size of 100 suffers from severe mode collapse. The best results can be obtained using a moderately large latent vector size. Table 5 compares the effect of different latent variable sizes on the MMD and MS-SSIM scores respectively.

Table 5: Effect of latent vector on MMD and SSIM on MNIST

Latent variable size	MMD × 0.0001	MS-SSIM
z 100	0.104	0.856
z 500	0.085	0.821
z 1000	0.068	0.818
z 2000	0.074	0.844

As can be seen, latent variable size with value 1000 produces the best results of those being compared. Both at low and high latent variable size mode collapse is seen which is one of the main challenges faced while training GANs.

Four common evaluation metrics have been used in the literature for testing the performance of generative models. These are log-likelihood, reconstruction error, ELBO and KL divergence.

The log-likelihood is calculated by finding the parameter that maximizes the log-likelihood of the observed sample. The reconstruction error is the distance between the original data point and its projection onto a lower-dimensional subspace. The optimization problem used in our model uses KL divergence error which is intractable hence we maximize ELBO instead of minimizing the KL divergence. KL divergence is a measure of how similar the generated probability distribution is to the true probability distribution. The comparison using these evaluation metrics for our model on MNIST dataset with the original VAE architecture is shown in Table 6.

Table 6: Comparison of results in original VAE vs our architecture on MNIST

Evaluation Metrics	VAE (Kingma and Welling, 2013)	Ours
log-likelihood	-1.568	-1.353
reconstruction error	88.5 × 0.001	4.27 × 0.001
ELBO	-1.697	-1.404
KL divergence	0.165	0.046

We compare our log probability distribution value with those obtained by previous state of the art methods which is shown in Table 7. The log probability distribution is an important evaluation metric in the sense that it shows the diversity of the samples generated.

Table 7: Results for independent samples for a model trained on MNIST

Method	$logp(x) \geq$
VAE + NF (T=80) (Rezende and Mohamed, 2015)	-85.1
VAE + HVI (T=16) (Salimans et al., 2015)	-88.3
conv VAE + HVI (T=16) (Salimans et al., 2015)	-84.1
VAE + VGP (2hl) (Tran et al., 2015)	-81.3
DRAW + VGP (Tran et al., 2015)	-79.9
VAE + IAF (Kingma et al., 2016)	-80.8
Ours	-82.2

Results

We present the generated images on all the 3 datasets used for testing. The images were trained for 1000 iterations. The images generated using the CELEBA-HQ dataset is shown in Fig 3.

Figure 3: 1024 × 1024 images generated using the CELEBA-HQ dataset.

The images generated using the LSUN BEDROOM dataset is shown in Fig 4.

Figure 4: 256 × 256 images generated using LSUN BEDROOM dataset

The images generated from different LSUN categories is shown in Fig 5.

Figure 5: Sample 256 × 256 images generated from different LSUN categories

We compare our results with previous state of the art networks on MNIST dataset in Fig 6.

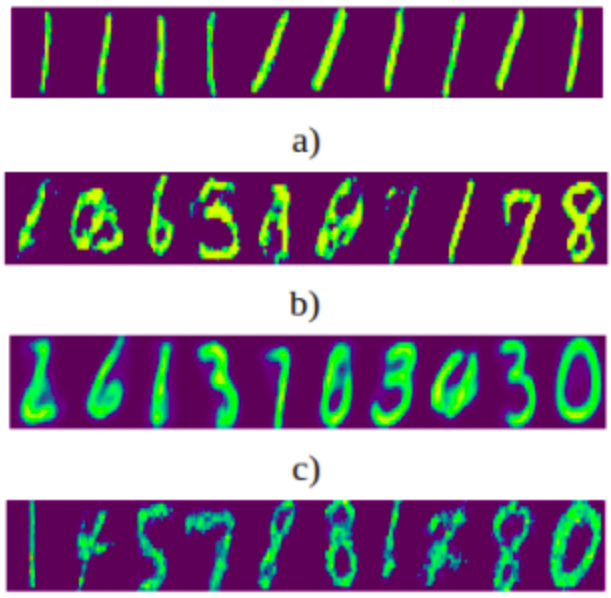

Figure 6: Generated MNIST images a) GAN b) WGAN c) VAE d) GVAE

Conclusions

In this blog, we presented a new training procedure for Variational Autoencoders based on generative models. This allows us to make the inference model much more flexible, allowing it to represent almost any posterior distributions over the latent variables. Our network was trained and tested on 3 publicly available datasets. On evaluating using MMD, SSIM, log likelihood, reconstruction error, ELBO and KL divergence as the evaluation metrics, our network beats the previous state of the art algorithms. Using generative model approaches to generate additional training data especially in fields like medical imaging could be revolutionary as there is a shortage of medical data for training deep convolutional neural network architectures.

References

S. U. Dar, M. Yurt, L. Karacan, A. Erdem, E. Erdem, and T. Çukur. Image synthesis in multi-contrast mri with conditional generative adversarial networks. IEEE transactions on medical imaging, 38 (10):2375–2388, 2019.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In Advances in neural information processing systems, pages 5767–5777, 2017.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

Before You Go

Research Paper: https://arxiv.org/pdf/2008.10399.pdf

Code: https://github.com/abhinavsagar/gvae

By Abhinav Sagar on August 1, 2020.

Canonical link

Exported from Medium on April 28, 2021.