Generative Models for Image Generation

Group Members:
Vishal Vijayvargiya
vvijayva@sfu.ca
Mahya Sadeghi
mahy.sadeghi@gmail.com
Amineh Dadsetan
adadseta@sfu.ca
Kiana Mostaghasi
kiana.mostaghasi@gmail.com
Amirmasoud Toudeshki
ghasemit@sfu.ca

Abstract

This report presents two approaches for artificial image generation using generative models, namely, Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN). VAEs are appealing because they are built on top of standard function approximators (neural networks), and can be trained with stochastic gradient descent. In GAN, we train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. Experiments demonstrate potential of both the approaches and an analysis is performed to study the advantages and disadvantages of them.

1 Introduction

Generative image algorithms are being in interest of researchers and have two main categories: parametric and non-parametric [1]. The non-parametric models mostly deal with matching from a database of existing images and have been used in texture synthesis [2], super-resolution [3] and in-painting [4].

On the other hand, parametric models for image generation has been studied deeply (for example on MNIST digits or for texture synthesis [5]). However, generating natural images of real world is still a big issue and had not been successful until recently. A recurrent network approach [6] and a deconvolution network approach [7] have recently had some success with generating natural images. Another approach generates images using an iterative forward diffusion process [8].

A variational sampling approach to generating images [9] has had some success. In just three years, Variational Autoencoders (VAEs) have showed up as one of the most popular approaches to unsupervised learning. VAEs are appealing because they are built on top of standard function approximators (neural networks), and can be trained with stochastic gradient descent.

VAEs have already shown progress in generating several types of complicated data, including handwritten digits, faces, segmentation, and predicting the future from static images.

Another well known algorithm in this area is Generative Adversarial Networks. GAN attempts to train an image generator by simultaneously training a discriminator to challenge it to improve.

(Generative Adversarial Nets) compares challenges in generative models. It mentions challenges in training, inference, sampling and model design for four generative model: deep directed graphical models, deep undirected graphical models, generative autoencoders and adversarial models. In our project we try to test Variational Autoencoder (VAN) and Generative Adversarial Networks (GAN) for two data set: faces and a landscape time lapse. We want to show how this two major algorithms could generate new images based on training data of real world image.

2 Approaches

We used three different approaches to generate new images from a given dataset of images. Our first approach is to use an auto-encoder that is to encode an image to a reduced set of features and then construct the image from the reduced feature set. In that way first we train an encoder neural network to encode our images and with the encoded data we train a decoder neural network that can generate pictures from a small set of features. Using this approach we construct an application that can generate a picture that is similar to two given picture from our dataset with defined percentage. Our approach to do that is to use the set of encoded features of each image and generate new set of encoded features. With the new generated encoded feature set, using the decoder part we generate the picture.

After using this approach, we were interested to implement other kinds of generative models to generate new images given a dataset of images. Those models are variational auto-encoder and generative adversarial nets. In the following sections, we are going to discuss these structures.

2.1 Variational autoencoder(VAE)

Variational Autoencoder is one of the methods that we used to generate new data from our given data.

The idea is to use variational inference. In varational inference schema, instead of computing posterior distribution over the latent variable, we assume that latent variable is from a specific probability distribution function like $q_\phi(z)$ in which z is our latent variable and ϕ is the varitional parameter of the distribution that we can change through optimization. From this prior distribution and with the assumption that the posterior distribution $q_\phi(z|x)$ is a gaussian, we can approximate this distribution. To do that, we set the $q_\phi(z|x)$ to be the out put of encoder part. Thus, the logarithm of posterior distribution function is:

$$log\ q_\phi(z|x^{(i)}) = log\mathcal{N}(z; \mu^{(i)}, \sigma^{2(i)}I) \tag{1}$$

Now to measure how how much our approximation is close to the real answer, we need to compute two things: 1- The latent loss and 2- the generation loss. The first term in computing error is to do the difference between the distribution of $q_\phi(z|x)$ and $q_\phi(z)$ which we aim to minimize it(works like a regulizer) and we can comput it with KL-divergence. The second term is due to the different between our generated image and the main image. As we see the first term try to prevent the output to be very close to input and instead try to minimize the differenc between $q_\phi(z|x)$ and $q_\phi(z)$(which is a unit gaussian).

Now, the question is how to compute the different between our generated image and the main image while we havn't generated image yet. The answer is to sample the z from the posterior distribution function $q_\phi(z|x)$. To do this sampling we need a reparametrization. Assuming $z \sim q_\phi(z|x)$ we will have

$$z = \mu + \sigma^2 \times \epsilon,\ \epsilon \sim \mathcal{N}(0, I) \tag{2}$$

It means that to sample z from the mean vector and variance vector, we just need to sample ϵ from a unit guassian distribution. After sampling this function we can compute a mean squared error for the generation loss term.

Finally the structure of the network that we are using is described in Figure 1.

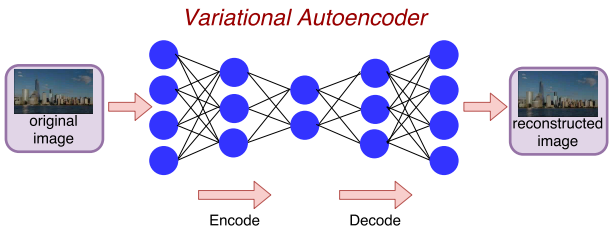

Figure 1: Structure of neural network used in VAE.

2.2 Generative adversarial nets

So far, the most striking successes in deep learning have involved discriminative models, and these striking successes have primarily been based on the backpropagation and dropout algorithms, using piecewise linear units which have a particularly well-behaved gradient. Deep generative models have had less of an impact, due to the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies, and due to difficulty of leveraging the benefits of piecewise linear units in the generative context. We used a new generative model estimation procedure that sidesteps these difficulties.

GAN (or Generative Adversarial Nets) is a framework for estimating generative models via an adversarial process. In this framework we simultaneously train two models:

- A generative model G that captures the data distribution,
- A discriminative model D that estimates the probability that a sample came from the training data rather than G.

In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 1/2 everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with back-propagation.

We consider the special case (adversarial nets) when the generative model generates samples by passing random noise through a multilayer perceptron. The discriminative model is also a multilayer perceptron. In this case, we can train both models using only the highly successful back-propagation and dropout algorithms and sample from the generative model using only forward propagation.

Figure 2 shows the structure of the network and the algorithm in figure 3 describes the algorithm.

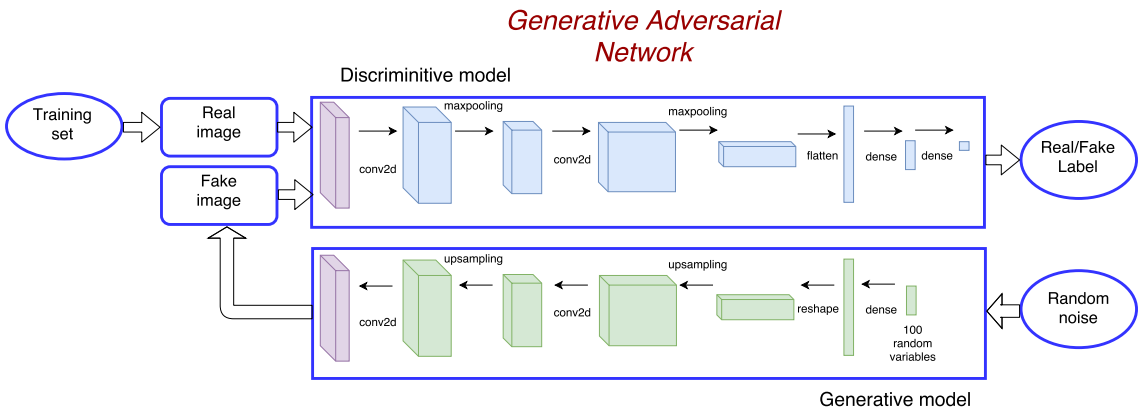

Figure 2: Diagram for GAN

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 3: GAN algorithm

3 Experiments

We tested our implementation of VAE and GAN on multifarious datasets. The most interesting datasets were Celebrity Face Dataset (CFD) [10] and New York Skyline Dataset (NYSD) [11]. CFD contained colored images of size 64×64, with training set size 9000 and test set size 1000. NYSD was created using frames of a video, partitioned into 1200 training images and 125 test images, each of size 64×64. Our experiment setting is as follows:

- batch size = 100
- original dimension = 12288 (this is $64 \times 64 \times 3$i)
- intermediate dimension (hidden layer nodes) = 512
- latent dimension (encoded dimension) = 2
- number of epoch = 250
- epsilon_std = 1.0

Selecting a good model architecture for GAN is very crucial for faster and better results. We tried various modifications, this includes experimenting with different activation functions like ReLU and tanh, using 2-3 layers of convolution, droput layer in discriminator and batch normalization in discriminator and generator. Different combinations gave different results, finally concluding with following architecture:

Generator Model(We are showing each layer with its dimension and its activation fucntion):

input dimension = 100 $\rightarrow$ dense(1024) $\rightarrow$ dense($128 \times 16 \times 16$) $\rightarrow$ BatchNormalization() $\rightarrow$ Activation('tanh') $\rightarrow$ Reshape(128, 16, 16) $\rightarrow$ UpSampling2D(size=(2, 2)) $\rightarrow$ Convolution2D(64, 5, 5, border_mode='same') $\rightarrow$ Activation('tanh') $\rightarrow$ UpSampling2D(size=(2, 2)) $\rightarrow$ Convolution2D(3, 5, 5, border_mode='same') $\rightarrow$ Activation('tanh') $\rightarrow$ Output Image

Discriminator model:

Image $\rightarrow$ Convolution2D(64, 5, 5, border_mode='same') $\rightarrow$ Activation('tanh') $\rightarrow$ MaxPooling2D(pool_size=(2, 2)) $\rightarrow$ Convolution2D(128, 5, 5) $\rightarrow$ Activation('tanh') $\rightarrow$ MaxPooling2D(pool_size=(2, 2)) $\rightarrow$ Flatten() $\rightarrow$ Dense(1024) $\rightarrow$ Activation('tanh') $\rightarrow$ Dense(1) $\rightarrow$ Activation('sigmoid')

4 Results

Figures show the results we obtained from VAE and GAN. VAE on skyline dataset generated most interesting results, with colorful sky in images. It should be noted that the original dataset didn't contain sky with red color, but VAE generated it. VAE for face dataset provided new faces but with some blur.

For GAN the results were sharp but with some irregularities. This could be attributed to less number of epochs, as training GAN takes time. Figure also shows results obtained on various epochs.

4.1 Analysis and Conclusion

Since both the models are "generative" with completely different approach, it is difficult to come up with a common comparison criteria. We provide benefits and drawbacks of both the techniques as part of analysis.

Pros:

- VAE - We can compare generated images directly to the originals, which is not possible when using a GAN
- VAE - Gives decent results in less training time
- GAN - Can generate the sharpest images

Cons:

- VAE - Network tends to produce more blurry images
- GAN - No way of determining which initial noise values would produce a particular picture

- GAN - Can lead to results where theres no actual object in a generated image, but the style just looks like picture

With our project we can conclude that the problem of generating realistic-looking images is complex. Both the approaches generate good artificial images, thus providing a novel way for data augmentation and image encoding. As a future work, it is possible to get better results if we combine VAE and GAN. Using the same encoder-decoder set-up, but using an adversarial network as a metric for training the decoder.

Figure 4: Generated images with GAN

Figure 5: Generated images with VAE

Figure 6: Generated Images with VAE and Skyline dataset

5 Contribution

Amineh - Worked on VAE model selection and implementation and implemented simple interface
Vishal - Worked on GAN model selection and implementation

Kiana - Worked on improving traditional autoencoder and poster
Mahya and Amirmasoud - Worked on possible techniques to improve VAE and dataset search

Refrences

[1] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[2] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[3] William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *IEEE Computer graphics and Applications*, 22(2):56–65, 2002.

[4] James Hays and Alexei A Efros. Scene completion using millions of photographs. *Communications of the ACM*, 51(10):87–94, 2008.

[5] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000.

[6] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[7] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[8] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. Celebrity face dataset.

[11] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. New york skyline video.