Comparison of DCGAN and Deep CNN-based VAE for Generating Fake Faces

Jordan Hanson

Abstract

Generative models have become increasingly popular in recent years with LLMs such as GPT. Presented in this paper is a comprehensive comparative analysis of two state-of-the-art generative models: Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). This paper analysis the results of two specific architectures trained on the CelebA data-set to generate fake images. These arhitectures are deep CNN-based Variational Autoencoder and deep convolutional generative adversarial network (DCGAN).

1. Introduction

1.1. Variational Autoencoders

A Variational Autoencoder (VAE) is a type of generative model that learns the latent space of a data set to generate images. First, the model encodes an image into a latent space z where $z = Encoder(x) \approx q(z|x)$. This is done using an encoder network defined later in the paper. Then the model decodes this latent space z into a reconstructed image $x' = Decoder(z) \approx p(x|z)$. Two different losses are used to make the model generate images that resembles the input and the learned latent space resembles a Gaussian distribution. The VAE is designed to learn a latent distribution from input images and reconstruct the images using a series of encoding, sampling, and decoding steps.

1.2. Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a type of generative model that learns to generate images that resemble the input data. There are two main components of a GAN, the generator and the discriminator. The generator is a network that generates images from a random noise value called the z value. The generator tries to create images that resemble the input data. The discriminator is a network that takes an image as an input and tries to determine if the image is real or fake. The training process involves pitting these two networks against each other. The generator is trained to fool the discriminator and the discriminator is trained to determine which images are from the CelebA dataset and which are generated by the Generator. This is repeated until the generator is able to produce realistic images.

2. Data

2.1. CelebA

The data-set used to train these models is the CelebA data-set. This data-set contains over 200,000 face images with each image having a resolution of 178×218. There are over 10,000 people represented in the data-set. The data-set comes with a csv file that contains binary attributes for each image such as smiling, eyeglasses, and bald. To work with this data-set: the images were converted to 64×64, normalized, and centered. The converted images are in the shape of 3x64x64 where 3 is the number of color channels and 64 is the height and width of each channel.

Figure 1. CelebA example images after transforming and denormalizing images.

3. Architecture and Training

3.1. VAE Network Architecture

Presented in this paper is a deep CNN-based Variational Autoencoder to generate fake faces trained on the CelebA dataset. The architecture is based off figure 2 from the research paper "Deep Feature Consistent Variational Autoencoder" (Hou et al., 2016). The architecture consists of an encoder and a decoder network, which are designed to compress and reconstruct the input images.

Encoder: The Encoder is a neural network that takes an input image (represented as a tensor) and outputs the mean and log-variance of the latent distribution associated with the input. As shown in the left column of figure 2, the Encoder consists of four convolutional layers (Conv2d), followed by batch normalization (BatchNorm2d) and leaky ReLU activation functions (LeakyReLU). The output of the encoder is then flattened and passed through two fully connected layers (Linear) to produce the mean and log variance.

Decoder: The Decoder is a neural network that takes a latent space (z) reconstructs the original image from it. As shown on the right column of figure 2, the Decoder begins with a fully connected layer (Linear) and an unflatten operation, followed by a series of upsample (Upsample) and convolutional layers (Conv2d) with batch normalization (BatchNorm2d) and leaky ReLU activation functions. The final layer is a convolutional layer that produces the image.

After encoding the image we sample from the mean and log variance to generate a reparameterized z value. This done during training to allow backpropogation to function correctly. This is done in my sample function where $z = \mu + \sigma \cdot \epsilon$ where μ is the mean found by the encoder, σ is the log variance found by the encoder, and ϵ is a random noise value from a normal distribution.

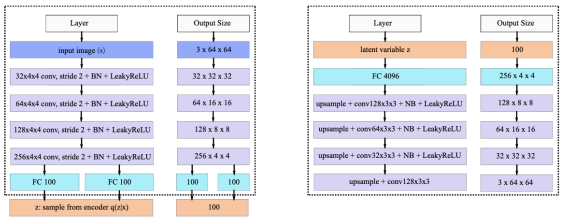

Figure 2. Architecture for deep CNN-based Variational Autoencoder. (Hou et al., 2016)

3.2. GAN Network Architecture

Presented in this paper implemented a Deep Convolutional Generative Adversarial Network (DCGAN) to generate fake faces trained on the CelebA dataset. The architecure used is shown in figure 3 and outlined in the research paper "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" (Radford et al., 2016). This GAN architecture consists of two main components: the Discriminator and the Generator. Both components are implemented as neural networks using the PyTorch library. Both the Generator and Discriminator are based off the architecture shown in figure 3.

Discriminator: The Discriminator is a neural network that takes an input image (represented as a tensor) and outputs the probability that the image is real (from CelebA dataset). The Discriminator consists of four convolutional layers (Conv2d), followed by batch normalization (BatchNorm2d) and leaky ReLU activation functions (LeakyReLU). The final layer is a convolution layer followed by a sigmoid activation function, which outputs the probability that the value is real (1 being completely real and 0 being fake).

Generator: The Generator is a neural network that takes a random noise vector (z) as input and generates an image. The noise vector is first reshaped into a 3D tensor, which is then passed through a series of transposed convolutional layers (ConvTranspose2d). Each transposed convolutional layer is followed by batch normalization (BatchNorm2d) and ReLU activation functions. The final layer is a transposed convolutional layer followed by a Tanh activation function, which produces the image.

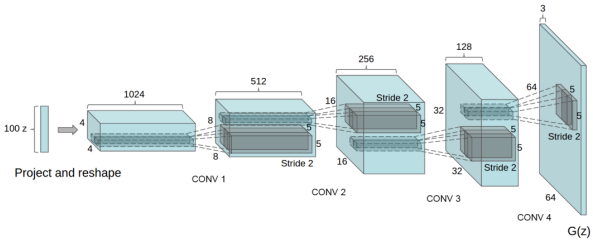

Figure 3. Architecture for Deep Convolutional Generative Adversarial Network (DCGAN). (Radford et al., 2016)

3.3. GAN Training

The training process involves making these two networks compete against each other. To train the discriminator, real images and fake images from the Generator (using random noise as input) are fed into the discriminator. The output is then fed into a Binary Cross Entropy loss (BCE) function with a label of 0 for the fake output and a label of 1 for the real output. The discriminator weights are then optimized based on the loss function using an ADAM Optimizer. The Generator is trained by generating fake images from random noise and feeding it into the discriminator. The output of the discriminator is then fed into the BCE loss function with

a label of 1 and the generator weights are optimized using ADAM. This is done for about 60 epochs, with a batch size of 128, and a latent space (z) of dimension 100.

3.4. VAE Training

The VAE was trained for about 50 epochs with a batch size of 256. For the training process, images are fed into the encoder to get the μ (mean) and σ (log variance) for that image. This is then sampled from using a sample function to generate a z value.

where $z = \mu \cdot \sigma + \epsilon$ where ϵ is random noise from a normal distribution with mean of 0 and standard deviation of 1.

This z value is then used to generated a reconstructed image. The loss functions are then applied to the ouputs. I use two different loss functions to update the weights of the network using gradient descent. These two loss functions are reconstruction loss and KL divergence. The reconstruction loss using a mean squared error function (using a sum reduction) measures the difference between the input image and the reconstructed image. The KL divergence measures how far the learned latent space differs from a normal distribution. KL diverge $= -0.5 * \sum_{batch}(1 + \sigma - \mu - e^{\sigma})$ where σ is the log variance and μ is the mean, both found using the encoder.
The KL divergence ensures that the model generates smooth and continuous data while the reconstruction loss ensures the reconstructed image resembles the input image. For each batch, these losses are added together and the weights are updated using an ADAM optimizer.

4. Results

4.1. GAN Results

After about 50 epochs, the GAN was able to generate face like images. The faces as shown in figure 4 are very good for only 50 epochs but definitely show some morphing between different images. The images were created by feeding random data drawn from a Gaussian distribution into the generator. Some images clearly resemble faces but some images are too noisy to resemble a face.

4.2. VAE Results

As shown in figure 5, the VAE was able to generate face like images from random noise. The images clearly show a general face in the middle of each image but the outside seems to morph between different attributes. Some images clearly resemble a known person with added attributes such as one of the images shows a clearly male actor with blurred long hair. In this case, the VAE has learned to encode important features of human faces, such as facial structure, eyes, nose, mouth, and hair, into its latent space. When generating new images from random noise, the VAE samples from this latent space and combines the encoded features in various ways, which can result in images that resemble known individuals but with added or altered attributes.

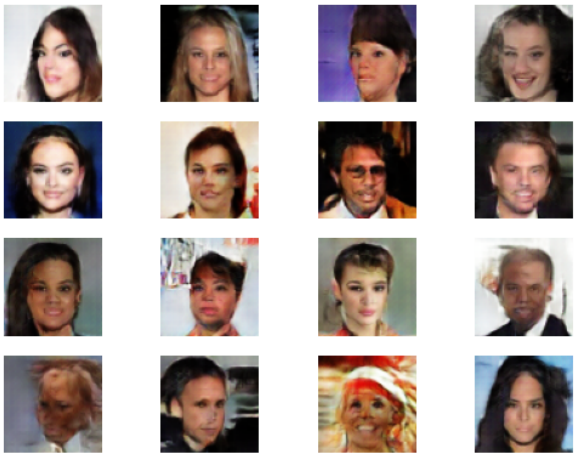

Figure 4. Fake Faces Generated by a DCGAN from Inputting Fake Noise.

The morphing effect observed in the generated images can be attributed to the fact that the VAE's latent space is a continuous space. So it represents certain images as points in the latent space and when random noise is inputted into the model the noise often lies in between known points and so the model combines features resulting in this blurring of attributes.

4.3. Comparison of Models

While both models were able to generate images that represented, both models clearly have advantages and disadvantages. In terms of training speed, the VAE clearly had an advantage here. After only a couple epochs, the VAE was able to generate faces even if they resembled the original images and the GAN took a couple epochs to show faces. The GAN also took significantly longer to run for 50 epochs, the GAN took around 6.5 hours while the VAE took about 4.5 hours for 50 epochs. The results clearly show that the VAE is able to generate better looking faces but the images look like they just take images it has already seen and add attributes to it. The GAN creates more original images but the images look less like faces with some being really bad and some being pretty good. The VAE also has the advantage of being able to isolate features and manipulate the latent space with more control.

The VAE and GAN have different advantages and disadvantages and depends on the use case for which model is best.

Figure 5. Fake Faces Generated by VAE from Inputting Fake Noise.

The GAN is able to generate more original images but takes more training to get good results. This is likely due to the fact that two models are trained from scratch to generate and determine which image is correct. The VAE shows more realistic results but seems to generate more blurred images where attributes are mixed together.

4.4. Exploring VAE Latent Space

The nature of the VAE allows more finetuned choice of latent space (z) for image reconstruction. By feeding certain images into the encoder, we are able to determine the latent space of different attributes. This is done by feeding all images into the encode and sample functions to find all the z values. We then can seperate all the z values into two groups those with the attribute and those without the attribute. We then can find the mean of the z values for each group. By finding the difference of these two groups we can determine the latent space for a specific attribute ($z_att = zMean_{attribute} - zMean_{nonAttribute}$). The attributes are defined in a csv file included with the CelebA dataset.

In figure 6, the reconstruction of the latent space of a smile is shown. Clearly the smile is the most prominent property of the image but it seems that long hair as well as a couple other attributes are included as well. THis is likely due to the data-set having more people with long hair and those attributes smiling. In figure 7, the smile z value is added to the z value of other non smiling images. This is done using an intensity factor of 2.5 being applied to the smile z value. $z = z_{nonattimage} + intentsity \cdot z_{smile}$. This shows the power of the VAE where attributes can be added to images.

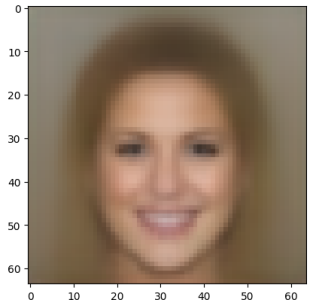

Figure 6. Reconstruction of Latent space of a smile.

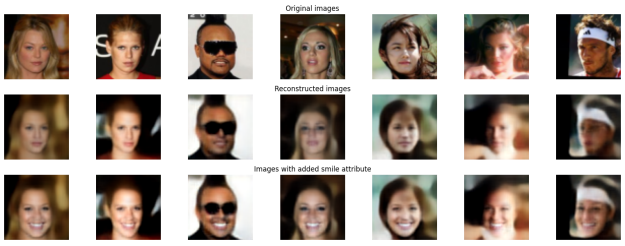

Figure 7. Applying Smile Attribute to Non Smiling Faces.

Figure 8, shows the same process this time with eyeglasses. The same process of determining the z value of eyeglasses are added to the z value of non eyeglasses pictures. Both figure 7 and 8, show good results where non attributes images are altered to display the attribute.

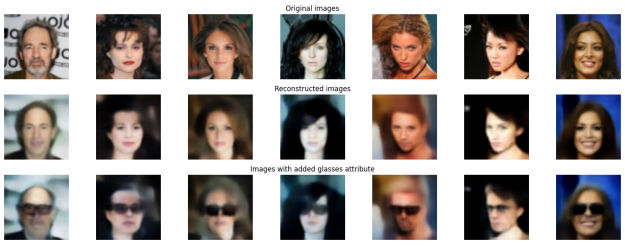

Figure 8. Applying Eyeglasses Attribute to Faces without Eyeglasses.

5. Future Work

This experimentation could be expanded by increasing the epochs and including other face data-sets to improve generated image quality. These architectures could also be

applied to other types of images to see what they are able to uncover and generate. Further experimentation could try to solely generate faces by combining different attributes together using their z values. This could be interesting to see how much control researchers can apply to the VAE by picking the z value that is used to reconstruct images. More experimentation into the GAN's latent space and choice of noise going into the generator could expand its capabilities. Researchers may be able to apply a similar process to the GAN that was done in this paper with attribute isolation in the VAE. With more control over z values the GAN may be able to generate better images with less training time.

6. Conclusion

In this paper, two different generative models are compared based on the quality of their results. These two models are deep CNN-based Variational Autoencoder and deep convolutional generative adversarial network (DCGAN). Both of these models have advantages and disadvantaged that may make them better or worse for different use cases. The deep CNN-based VAE gives more control over the latent space of an image while the DCGAN allows potentially better results and more original images.

7. Citations and References

References

Hou, X., Shen, L., Sun, K., and Qiu, G. Deep feature consistent variational autoencoder, 2016.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.