Synthetic Data Generation

Leung Yiu San
Hong Kong University
of Science and Technology
ysleungad@connect.ust.hk

Leung Hon Hang Benny
Hong Kong University
of Science and Technology
hhbleung@connect.ust.hk

Yu Cheuk Hei
Hong Kong University
of Science and Technology
chyuam@connect.ust.hk

1. Introduction

Nowadays, deep learning has been widely used in different industries. Common examples are facial recognition detection systems, autonomous vehicles, and cancer detection. Although data plays a crucial role for training models in deep learning, real-world data are often scarce and limited. Synthetic data is thus proposed as an alternative to real-world data collected through sensors or measurements [1]. In situations where real data is insufficient, we can rely on synthetic data to train a model. Synthetic data is annotated information generated by computer simulations or algorithms. Unlike data augmentation, the idea of synthetic data is to use deep learning to generate quality data and feed them into other deep learning models for training. Instead of having translated, rotated, or jittered images as in data augmentation, synthetic data generation manages to produce an entire set of new images that mimics features in original images. In cases where we only have a small data set, machine-generated data can augment the original data set to allow more training materials. In this project, we will utilize a Generative Adversarial Network (GAN) model to augment the CIFAR-10 data set with more machine-generated data. We will then perform image classification using only the CIFAR-10 data set, only the synthetic data set, and a mixture of the two data sets. We hope to experiment the significance of synthetic data on small data sets by comparing accuracies when using different data sets.

2. Problem Statement

Let X and Y denote the set of training data and the set of generated samples respectively. We also denote Z as the set of random noise for further usage. Given n training images in X, our goal is to generate m sample images in Y using GAN such that X and Y come from the same distribution.

There are two major approaches for generative models, implicit and explicit density estimation. GAN adopts the implicit density estimation approach by giving up on explicit modeling density. It is a way of training generative models by leveraging two sub-models: the generator network G and the discriminator network D. The former one takes random noise, the set Z as input, learns transformation to training distribution and outputs fake images. Mathematically, it trains a mapping function $G : Z \to Y$. Its purpose is to fool the discriminator by generating real-looking images. Meanwhile, the discriminator network takes images from the generator network and the data set X as input and distinguishes whether these images are real or fake. Mathematically, it trains a mapping function $D : Y \cup X \to (0, 1)$, where the range of the function represents the likelihood that the images come from real data X rather than synthetic data Y. Using this result, we update both networks and repeat the whole process. Figure 1 summarizes the GAN model architecture.

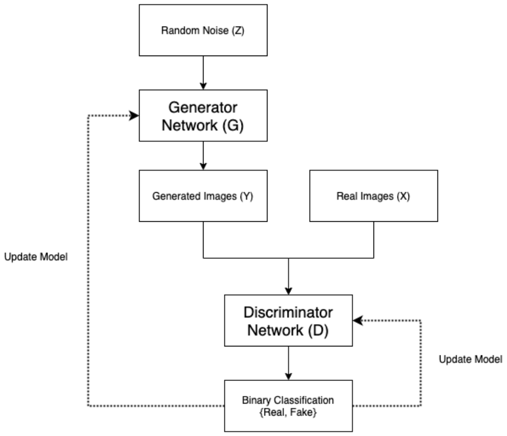

Figure 1. **GAN –** A figure summarizing the GAN architecture.

The training and testing images of this project will be obtained from the CIFAR-10 data set. The reason why we choose the CIFAR-10 data set is because its size is small. The model will be trained using the TensorFlow library on Google Colab for better performance and shorter computational time. We will evaluate our results by comparing training, validation, and test accuracies when performing image classification using only the CIFAR-10 data set, only the synthetic data set, and a mixture of the two data sets. Our target is to train a GAN model that facilitates image classification, especially on classes with small amount of real data. This also explains why we choose CIFAR-10 as our data set.

3. Technical Approach

The GAN model in this project will be trained according to the framework proposed in the Generative Adversarial Nets paper [2]. The network consists of a generator $G : Z \to Y$, and a discriminator $D : Y \cup X \to (0, 1)$. The higher the output of the discriminator model, the more likely that the images come from real data. The objective of G is to maximize likelihood of discriminator being wrong, such that $D(G(z)) = D(y),\ where\ y \in Y and\ z \in Z$ gets closer to 1, while the objective of D is to maximize likelihood of discriminator being correct, such that $D(x),\ where\ x \in X$ gets closer to 1. Combining both D and G, we obtain a minimax objective function $V(D, G) =$

$$\min_G \max_D \\ [\, \mathbb{E}_{x\sim X}\ log(D(x)) + \mathbb{E}_{z\sim Z}\ log(1 - D(G(z)))\,] \quad (1)$$

The minimax generator loss function is denoted as:

$$L_G = \mathbb{E}_{z\sim Z}\ log(1 - D(G(z))) \quad (2)$$

The minimax discriminator loss function is denoted as:

$$L_D = \mathbb{E}_{x\sim X}\ log(D(x)) + \mathbb{E}_{z\sim Z}\ log(1 - D(G(z))) \quad (3)$$

When training the model, we would like to maximize both L_G and L_D. Below shows the complete GAN training algorithm: [2]

Algorithm 1 GAN Training Algorithm

```
for number of training iterations do
    Sample $m$ noise samples from $Z$
    Sample $m$ real samples from $X$
    Update $D$ by ascending its stochastic gradient:
        $\nabla \frac{1}{m} \sum_{i=1}^{m} L_D$
    Sample $m$ noise samples from $Z$
    Update $G$ by ascending its stochastic gradient:
        $\nabla \frac{1}{m} \sum_{i=1}^{m} L_G$
end for
```

Further, the generator network G is an upsampling network with fractionally strided convolutions. It consists of 3 transpose convolutional layers that are used to up-sample the images back to its original size. On the other hand, the discriminator netowork D is a convolutional network. It consists of 4 convolutional layers which down-samples the input image. After the last layer, dropout is used for regularization followed by a dense layer for binary classification.

Figures 2 and 3 show the architecture of the generator and the discriminator model.

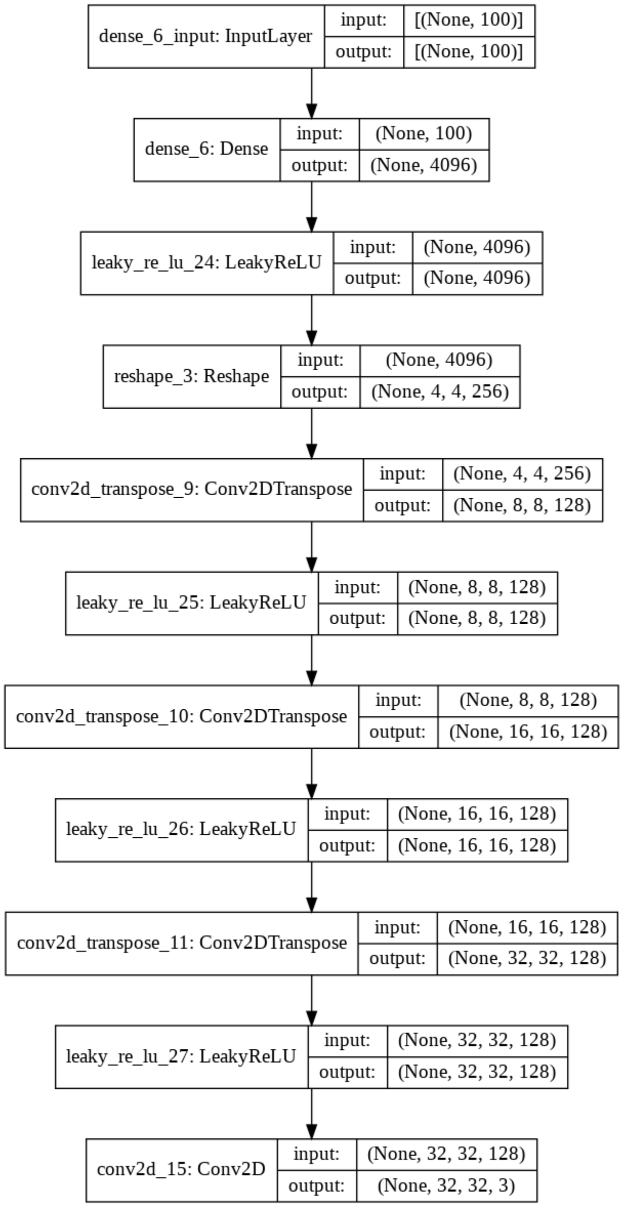
dense_6_input: InputLayer | input: [(None, 100)] | output: [(None, 100)]
dense_6: Dense | input: (None, 100) | output: (None, 4096)
leaky_re_lu_24: LeakyReLU | input: (None, 4096) | output: (None, 4096)
reshape_3: Reshape | input: (None, 4096) | output: (None, 4, 4, 256)
conv2d_transpose_9: Conv2DTranspose | input: (None, 4, 4, 256) | output: (None, 8, 8, 128)
leaky_re_lu_25: LeakyReLU | input: (None, 8, 8, 128) | output: (None, 8, 8, 128)
conv2d_transpose_10: Conv2DTranspose | input: (None, 8, 8, 128) | output: (None, 16, 16, 128)
leaky_re_lu_26: LeakyReLU | input: (None, 16, 16, 128) | output: (None, 16, 16, 128)
conv2d_transpose_11: Conv2DTranspose | input: (None, 16, 16, 128) | output: (None, 32, 32, 128)
leaky_re_lu_27: LeakyReLU | input: (None, 32, 32, 128) | output: (None, 32, 32, 128)
conv2d_15: Conv2D | input: (None, 32, 32, 128) | output: (None, 32, 32, 3)

Figure 2. **Generator Model –** A figure showing the architecture of our generator model.

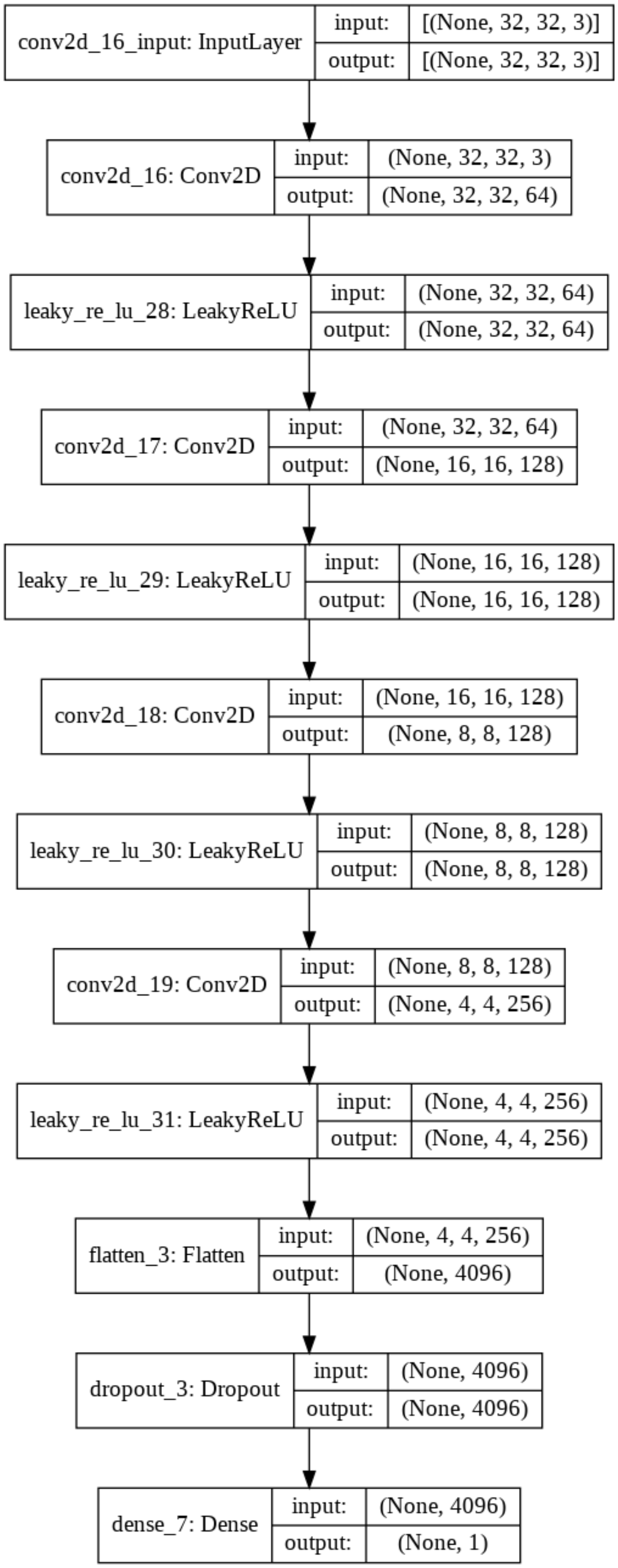
conv2d_16_input: InputLayer
input: [(None, 32, 32, 3)]
output: [(None, 32, 32, 3)]
conv2d_16: Conv2D
input: (None, 32, 32, 3)
output: (None, 32, 32, 64)
leaky_re_lu_28: LeakyReLU
input: (None, 32, 32, 64)
output: (None, 32, 32, 64)
conv2d_17: Conv2D
input: (None, 32, 32, 64)
output: (None, 16, 16, 128)
leaky_re_lu_29: LeakyReLU
input: (None, 16, 16, 128)
output: (None, 16, 16, 128)
conv2d_18: Conv2D
input: (None, 16, 16, 128)
output: (None, 8, 8, 128)
leaky_re_lu_30: LeakyReLU
input: (None, 8, 8, 128)
output: (None, 8, 8, 128)
conv2d_19: Conv2D
input: (None, 8, 8, 128)
output: (None, 4, 4, 256)
leaky_re_lu_31: LeakyReLU
input: (None, 4, 4, 256)
output: (None, 4, 4, 256)
flatten_3: Flatten
input: (None, 4, 4, 256)
output: (None, 4096)
dropout_3: Dropout
input: (None, 4096)
output: (None, 4096)
dense_7: Dense
input: (None, 4096)
output: (None, 1)

Figure 3. **Discriminator Model –** A figure showing the architecture of our discriminator model.

4. Preliminary Results

We have successfully generated some images using our architecture. Yet, these images are quite blurry and there is still room for improvement. Figure 4 shows the image generated by the GAN model.

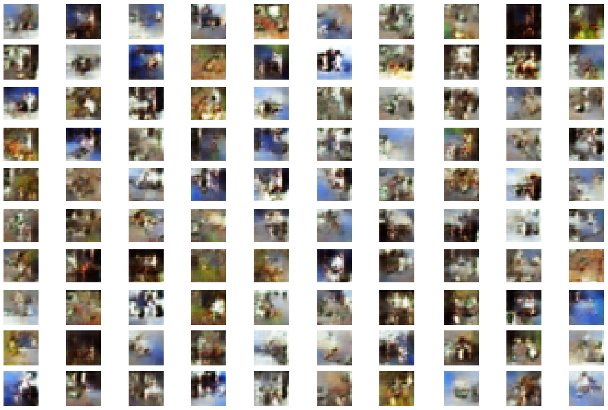

Figure 4. **Results of the Generated Images –** Images generated using the GAN model.

In the coming weeks, we will try to improve the GAN model by modifying the architectures of the generator and the discriminator model. We will also try to use some advanced techniques for training GANs, such as minibatch discrimination and one-sided label smoothing. [3]

References

[1] GERARD ANDREWS. What Is Synthetic Data? https://blogs.nvidia.com/blog/2021/06/08/what-is-synthetic-data/, 2021. Online; accessed 8 June 2021. 1

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 2

[3] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. 3