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

Abstract

Synthetic data has become more and more important as deep learning advances. This work attempts to investigate the importance and limitations of synthetic data in image classification using different Generative Adversarial Network (GAN) models. We start by evaluating the GANs using Frechet Inception Distance (FID) and then train a Convolutional Neural Network (CNN) with a combination of the data sets and observe the accuracies. Results from the experiments show that GANs might fool the CNN into learning the generated distribution instead, if improperly monitored.

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 [4].

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. It manages to produce an entire set of new images that mimics features in original images. In situations where real data is insufficient, synthetic data can augment the original data set to allow more training materials.

In this project, we will utilize GAN models to augment the CIFAR-10 data set with more machine-generated data. We will then perform image classification using different combinations of the data sets. We hope to experiment the significance of synthetic data on the original data distribution by comparing accuracies. Overall speaking, the quality of the generated images must be checked before passing it to the CNN. Although the images produced may be good-looking, they may still be unsuitable to use as training data.

2. Related works

2.1. Data Augmentation for GAN Training

Data Augmentation Optimized for GAN (DAG) is a framework that optimizes different GAN models and improves their learning of the original distribution. This paper [11] argues that traditional data augmentation approaches, such as flips, random crops and scales, and image jittering might mislead the generator to learn the distribution of the augmented data. It then proposes the DAG framework and evaluates the results with FID. The new models all demonstrated consistencies with the original distributions and improvements when compared to the unoptimized models. However, image classification was not performed using the DAG framework. We would also like to compare the training and test accuracies when using generated images as training data sets instead of FID in our project.

2.2. Synthetic Medical Image Augmentation

Obtaining data sets in the medical domain has been known to be a challenging task. This paper [3] compares the performance of traditional data augmentation and synthetic data augmentation when performing image classification. Images are generated using GANs based on a data set with 182 liver images. Results showed that there is roughly 10 % improvement on classification performance when training the CNN using synthetic images rather than the images generated through traditional data augmentation approaches. On top of their research, we would like to further verify this result on other commonly seen classes as well.

3. Dataset

The project hypothesizes that using generated data as training data set has a positive effect on image classification. No traditional data augmentation approaches are performed as we would like to compare the performance of the original data set and the synthetic data set on image classification. Only standardization was performed on the data sets before training.

3.1. COCO Dataset

COCO dataset is used for large-scale object detection, segmentation, and captioning dataset. However, we have encountered limitations when using this data set since the experiment is conducted under Google Colab. When the dataset is loaded into Colab for further process, both RAM allocation and disk usage has exceeded the limit. Hence, we are not able to train GAN with COCO Dataset.

3.2. CIFAR-10 data set

We then decided to use CIFAR-10 data set [8] to conduct our experiment. With its small image dimension, we can stay within Google Colab's RAM limit. The data set consists of 60000 32x32 images in 10 classes, with 4500 images for training, 500 images for validation, and 1000 images for testing in each class. The training images will be used in both the GANs and the CNN, while the testing images will be used in only the CNN. Further, 60000 32x32 images will be generated by the GAN and used in the CNN. These generated images consist of 10 classes, with same number of images for training, validation, and testing. As for the combined data set, it is obtained through merging and shuffling the CIFAR-10 and the generated data sets.

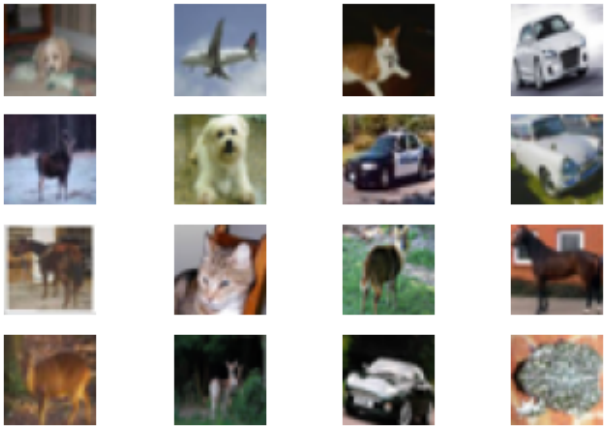

Figure 1. Sample images taken from the CIFAR-10 data set

4. Methodology

4.1. Problem Formulation

Let X and Y denote the set of training data and the set of generated samples respectively. Further, we denote Z as the set of random noise. Given n training images in X, our first task is to generate m sample images in Y using GAN such that X and Y come from the same distribution. The GAN architecture consists of a generator G and a discriminator D. The generator trains a mapping function $G : Z \rightarrow Y$ to fool the discriminator by generating real-looking images. As for the discriminator, it trains a mapping function $D : Y \cup X \rightarrow (0, 1)$ to determine the likelihood that the images come from real data X or from synthetic data Y.

Our second task is to classify images from the CIFAR-10 dataset and our generated data. This model takes a (32, 32, 3) as input and outputs a label of range [1, 10]. Let A and B denote the set of training data and the set of labels respectively. The CNN C trains a mapping function $C : A \rightarrow B$ to classify the images into different categories.

4.2. GAN Architecture

The model we are using is a traditional DCGAN. Although there are many improved versions, we still want to use DCGAN to examine its effects since it is the most commonly-used GAN.

4.2.1 Generator Architecture

The complete architecture of the Generator is summarized in Figure 10 in the Appendix. It contains nine learned layers — five convolutional up-sampling layers and four layer-normalization layers. Leaky ReLU with α = 0.2 is used throughout the whole process and the learning rate used is 0.001.

4.2.2 Discriminator Architecture

The complete architecture of the Discriminator is summarized in Figure 9 in the Appendix. It has five learned layers — four convolutional down-sampling layers and one fully-connected layer. Leaky ReLU with α = 0.2 is used throughout the whole process and the learning rate used is 0.0002. Unlike the generator model, there is no layer normalization layers within the discriminator model since we want to avoid the discriminator from getting too strong.

4.2.3 Loss Function

The objective of the Generator G is to maximize likelihood of discriminator being wrong, such that $D(G(z))$ gets closer to 1 and $D(y)$ goes to 0.5, where $y \in Y$ and $z \in Z$, while the objective of the Discriminator 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:

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

The minimax discriminator loss function is denoted as:

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

4.2.4 GAN Algorithm

When training the model, we would like to maximize both L_G and L_D. Below shows the complete GAN training algorithm: [5]

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

4.2.5 Layer Normalization

Layer Normalization computes mean and variance across different channels, capturing information over channels of the same image but not over all images. [2] Therefore, it is independent on the size of the mini-batch. Let x be the set of feature maps with dimensions $N \times C \times H \times W$, where N is the batch size, C is the number of channels, H, and W are the height and width of the feature map respectively. The equation of Layer Normalization can be defined as follows:

$$\mu_n = \frac{1}{CHW} \sum_{i=1}^{C} \sum_{j=1}^{H} \sum_{k=1}^{W} x_{nijk} \tag{4}$$

$$\sigma_n^2 = \frac{1}{CHW} \sum_{i=1}^{C} \sum_{j=1}^{H} \sum_{k=1}^{W} (x_{nijk} - \mu_n)^2 \tag{5}$$

$$\hat{x} = \frac{x - \mu_n}{\sqrt{\sigma_n^2 + \epsilon}} \tag{6}$$

$$y = \gamma \hat{x} + \beta \tag{7}$$

, where $\hat{x}$ represents the layer normalized feature maps, y represents the output after layer normalization with learnable scale and shift parameters γ and β.

4.3. CNN Architecture

The architecture of CNN is summarized in Figure 11 in the appendix. It contains nine learned layers — four convolutional down-sampling layers, four batch-normalization layers and one fully-connected layer. Leaky ReLU with α = 0.2 is used throughout the whole process and the learning rate used is 0.0001.

4.3.1 Loss Function

The loss function of the CNN [9] is denoted as:

$$\frac{1}{N} \sum_{i=1}^{N} -log(y_i * log(p_i) + (1 - y_i * log(1 - p_i)) \tag{8}$$

4.3.2 Batch Normalization

Batch Normalization is used as Layer Normalization is unfavourable for CNNs. [2] Unlike Layer Normalization, Batch Normalization computes mean and variance across images within the mini-batch. [7] It adds regularization effect on the network to avoid over-fitting and makes the network more easier to train. Let x be the set of feature maps with dimensions same as that in Layer Normalization. The equation of Batch Normalization can be defined as follows:

$$\mu_c = \frac{1}{NHW} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} x_{icjk} \tag{9}$$

$$\sigma_c^2 = \frac{1}{NHW} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} (x_{icjk} - \mu_c)^2 \tag{10}$$

$$\hat{x} = \frac{x - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} \tag{11}$$

$$y = \gamma \hat{x} + \beta \tag{12}$$

, where $\hat{x}$ represents the batch normalized feature maps, y represents the output after batch normalization with learnable scale and shift parameters γ and β.

4.4. Choosing Hyperparameters

Correct choice of learning rates plays an important role in training GANs. Since the loss function of GAN is based on a min-max function, it is hard to balance both the generator and the discriminator. Our choice of hyperparameters

should facilitate the GANs to optimize themselves without causing gradients to vanish.

Learning rates 0.001, 0.0005, and 0.0002 have been tested and we have finally chosen 0.001 for the generator and 0.0002 for the discriminator. When using 0.001 as learning rate, there is a high chance for the gradients to vanish since the discriminator is too strong. This also explains why we choose different learning rates for the discriminator and the generator.

5. Evaluation and Results

5.1. Generating Images

We have trained 10 seperate GANs, all with 100 epochs, to generate images for the 10 different classes. At epoch 50, the shape of the object starts to form but the details are still in development. There are also some noise remaining in the images. At epoch 100, the shape of the object becomes a lot more obvious and details can be seen.

Figure 2. Sample images of the horse class at epoch 50 (top 2 rows), and at epoch 100 (bottom 2 rows)

5.2. Evaluting Images

To evaluate the quality of the generated images, we use Frechet Inception Distance (FID). The inception score estimates the quality of a collection of synthetic images. Each FID score category is obtained by performing image classification with an Inception v3 model. Let R and G denote the set of real sample and the set of generated samples respectively. The FID score L is denoted by:

$$L = || \mu_R - \mu_G ||^2 + Tr(\sigma_R + \sigma_G - 2 * \sqrt{\sigma_R * \sigma_G}) \quad (13)$$

Below shows the FID scores for the generated categories, which are relatively low. This means that the distribution of the generated images is close to the original distribution.

Categories	FID score
Airplane	65.113
Automobile	38.060
Bird	73.156
Cat	52.456
Deer	48.013
Dog	37.978
Frog	40.173
Horse	51.377
Ship	32.909
Truck	27.774

Table 1. FID scores for each class

5.3. Image Classification

In this section, we present the image classification results achieved by the described CNN architecture on the three different data sets. 3 planned experiments and 2 additional experiments were performed.

- Train and test using the CIFAR-10 data set.
- Train and test using the synthetic data set.
- Train and test using the mixed data set.
- Train using the CIFAR-10 data set and test using the synthetic data set
- Train using the synthetic data set and test using the CIFAR-10 data set

A brief summary of the training accuracy at the last epoch and the test accuracy can be seen in the table below:

Training Data set	Testing Data set	Training Accuracy	Testing Accuracy
CIFAR-10	CIFAR-10	0.8573	0.7166
CIFAR-10	Synthetic	0.8573	0.5011
Synthetic	Synthetic	1.0000	1.0000
Synthetic	CIFAR-10	1.0000	0.1359
Mixed	Mixed	0.9613	0.8546

Table 2. Summary of the training and test accuracies

5.3.1 Experiment 1 – Train and Test using CIFAR-10

The first experiment is to train and test a CNN model with images from CIFAR-10 only. Although 20 epochs were performed, we have chosen the model at the 10^{th} epoch since we observed over-fitting afterwards. We finally achieved 71.66% accuracy on the test set. The confusion matrix also showed that the classes with the most missed prediction are dog and cat, meaning that the two classes are extremely alike.

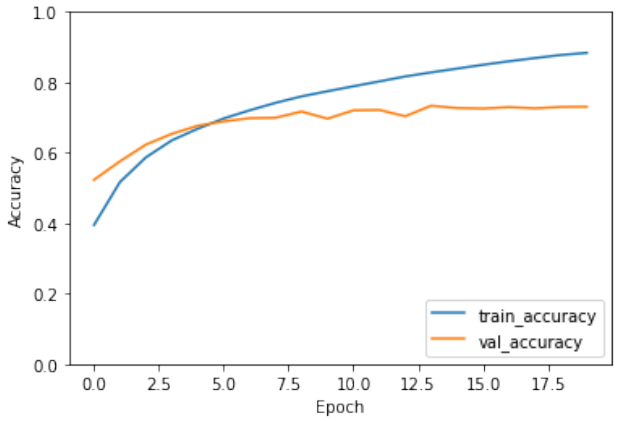

Figure 3. The training and validation accuracies of experiment 1, overfitting observed starting from the 7^{th} epoch

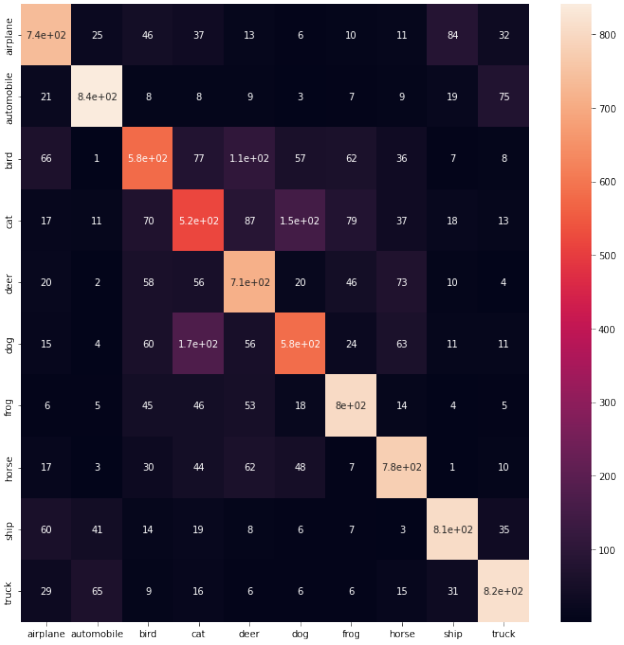

Figure 4. The confusion matrix of experiment 1

5.3.2 Experiment 2 – Train and Test using Synthetic

The second experiment is to train and test the CNN model with images from the GAN output, which should also be having the same distribution as the CIFAR-10 dataset. Unexpectedly, both training and validation accuracy reached 100% at the very beginning. There is no exact reason to explain this phenomenon but our hypothesis is that, the generated images are from the modes of the original distribution. This occurs when the generator over-optimizes for a specific discriminator in each iteration, which limits the final output images of the GANs within a small set of output types. [6] Therefore, the 6000 generated images of each class may only represent modes of the original distribution instead, which makes it a lot easier to learn.

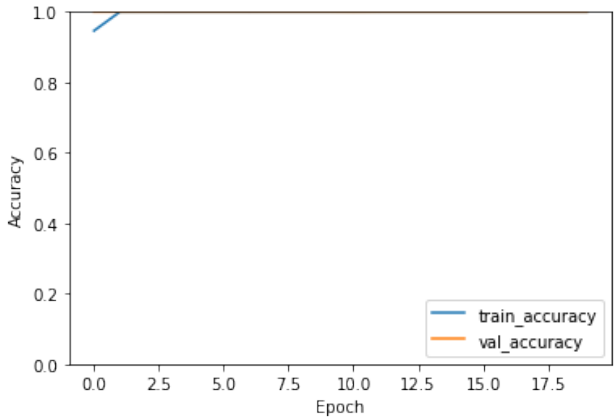

Figure 5. The training and validation accuracies of experiment 2

5.3.3 Experiment 3 – Train and Test using Mixed

The third experiment is to train and test the CNN model with images from both real and generated data. It shows improvement on the performance of the CNN model as it yields an accuracy of 85.46% which is much better than that of experiment 1. The confusion matrix is also similar to that of experiment 1 with same mostly misclassified predictions. Since the generated images also come from the same distribution as the real images, it is reasonable that the most wrongly predicted classes also remains the same.

5.3.4 Experiment 4 & 5 – Addtional Experiments

As we observed unexpected results in experiment 2, we decided to perform 2 additional experiments to further verify our results.

The fourth experiment trains using the CIFAR-10 data set but tests using the synthetic data set. It only yields an accuracy of 50.11%, much lower than that of experiment 1 (71.66%). The confusion matrix is still similar to that of experiment 1 and 3, with cats and dogs being mostly misclassified. However, the number of misclassified images increased a lot.

The fifth experiment trains using the synthetic set but tests using the CIFAR-10 data set. This time, the accuracy obtained is even lower, with only 13.59%. From the confusion matrix, we can observe that nearly all images are classified as airplane, automobile, ship, and truck, which is incorrect.

The results obtained in this section are not as expected. Since the synthetic data set comes from the CIFAR-10 data set, we expect them to have similar distributions, but results

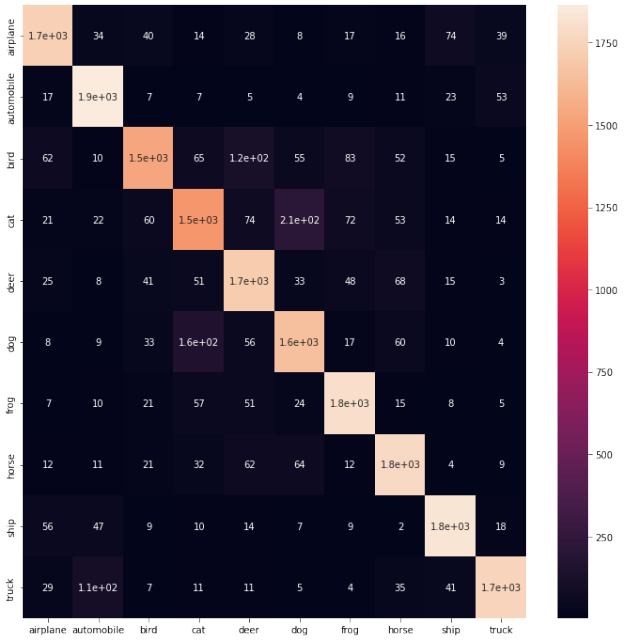

Figure 6. The confusion matrix of experiment 3

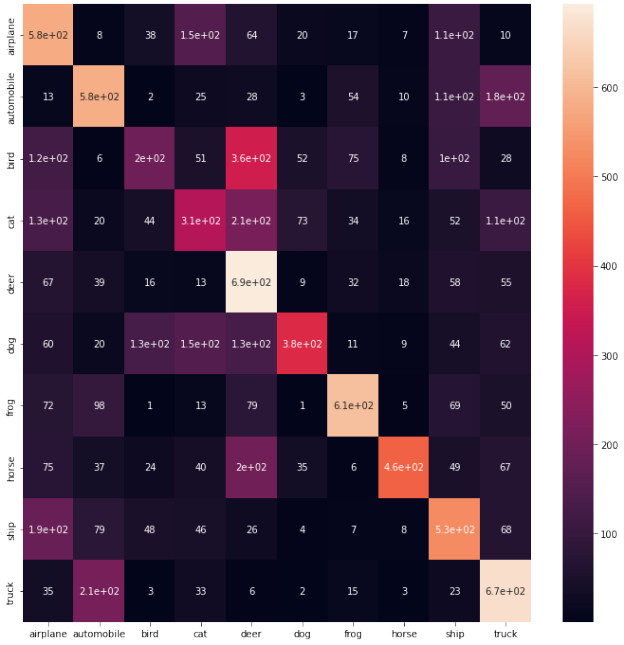

Figure 7. The confusion matrix of experiment 4

show that they do not. This further verifies our hypothesis that the generated images represents only the modes of the original distribution, and thus lead to different distributions.

Figure 8. The confusion matrix of experiment 5

6. Conclusion

In this work we evaluated the effect of synthetic data on image classification. It was demonstrated that the inclusion of generated data may not be advantageous for the classification accuracy. Since GAN is based on a two-person zero sum non-cooperative game, it is hard to train a stable GAN that produces decent images across the whole distribution. Therefore, image produced by GANs may often collapse to modes and mislead the CNN model, causing it to learn the distribution of the generated images instead of the original one. When passing generated images to CNNs, the quality of the images must be checked in advance. Although the images produced may seem good-looking and have low FID scores, they may still be unsuitable to use as training data since it represents a different distribution from the original one.

In the future, different GANs can be tested to see if they produce better generated images for training CNNs. WGAN [1] uses a new algorithm and another loss function to improve learning stability and avoid mode collapse while unrolled GANs [10] considers previous discriminator versions as well to prevent the generator from over-optimizing of a single discriminator. Further, the COCO data set is also suggested since it contains images with higher resolution. Similar experiments, such as generating male and female faces for binary classification, can also be done.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017. 6

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 3

[3] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331, Dec 2018. 1

[4] 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

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 3

[6] Google. Common problems - mode collapse. https://developers.google.com/machine-learning/gan/problems#mode-collapse. 5

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 3

[8] Alex Krizhevsky. The cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html. 2

[9] Vlastimil Martinek. Cross-entropy for classification. https://towardsdatascience.com/cross-entropy-for-classification-d98e7f974451. 3

[10] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks, 2017. 6

[11] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021. 1

Appendix

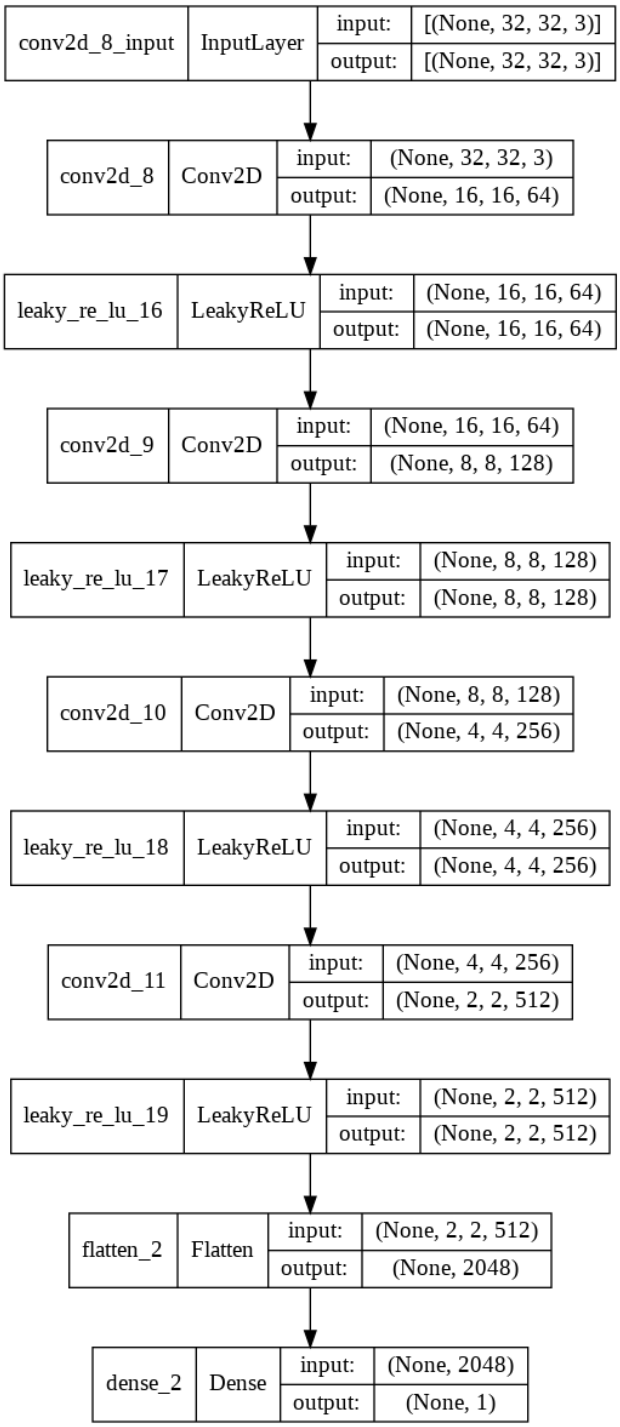

Figure 9. Summary of Discriminator model in GAN

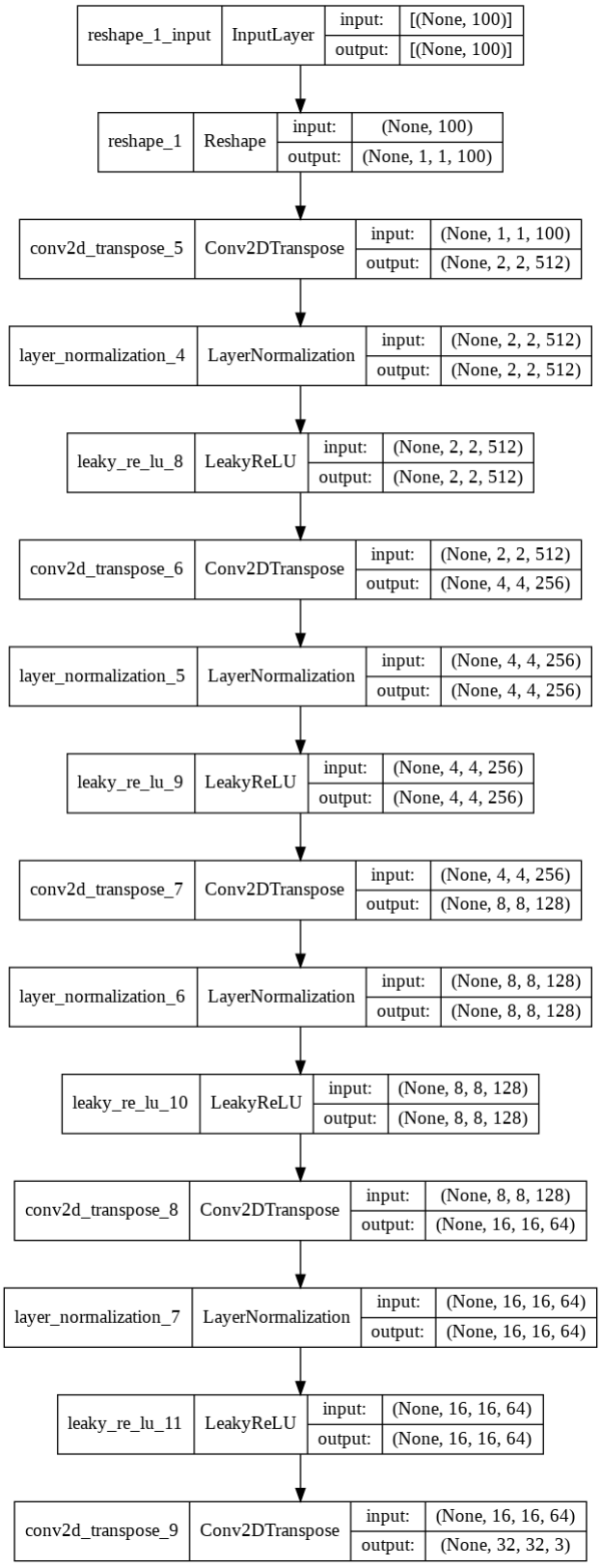

Figure 10. Summary of Generator model in GAN

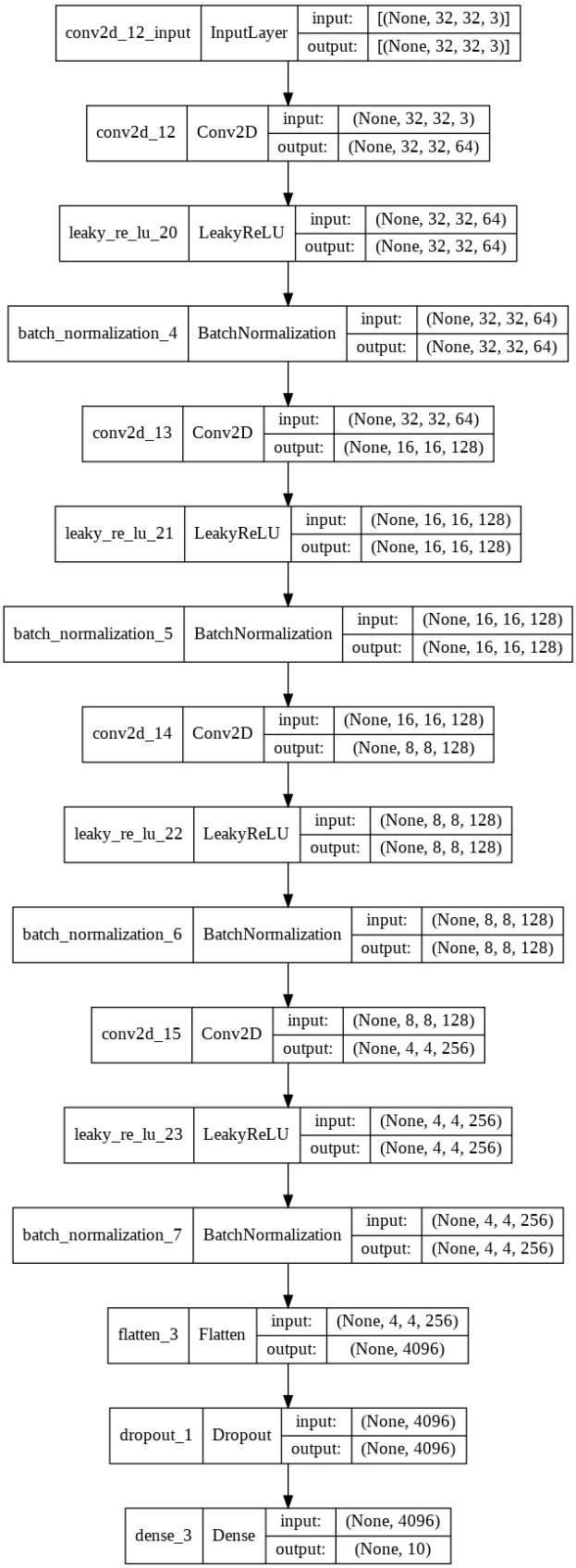

Figure 11. Summary of CNN model