**PROJECT REPORT**

For

Fake Image Generation using

Deep Convolutional Generative Adversarial Networks (DCGANs)



Department of Computer Science

School Of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DEHRADUN- 248007. Uttarakhand

Prepared by

|  |  |  |
| --- | --- | --- |
| **Name** | **Sapid** | **Course** |
| Udit Sharma | 500096113 | MCA (AI and ML) |

Submitted to **Dr. Pankaj Kumar**

**ASBTRACT**

Deep convolutional neural networks have been excelling continuously on various challenging visual analysis tasks. Deep models with parameter-heavy architectures have been successfully trained and deployed on a multitude of applications, and owe their success partly due to the continuous development of increasingly more powerful graphical processing units. However, the power consumption and sheer size of such models hinder their applicability in robotics applications. Thus, recent research has steered toward the optimization of deep learning architectures for deployment on devices with limited resources. This entails using modules with fewer parameters and fewer floating-point operations, as well as careful tuning of such models to train models, which are both efficient and effective.

**INTRODUCTION**

* + DCGAN is one of the popular and successful network design for GAN.
  + Deep Convolutional GAN (DCGAN) was proposed by a researcher from MIT and Facebook AI research.
  + It is widely used in many convolution-based generation-based techniques. The focus of this paper was to make training GANs stable. Hence, they proposed some architectural changes in the computer vision problems.
  + DCGANs are introduced to reduce the problem of mode collapse. Mode collapse occurs when the generator got biased towards a few outputs and can’t able to produce outputs of every variation from the dataset.

**PROJECT DESCRIPTION**

* **Dataset**

**Context**

* The CIFAR-10 dataset consists of 60000 ,32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.
* The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

**Content**

* Each image is 32 pixels in height and 32 pixels in width, for a total of 1024 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255.
* **Reference Algorithm**

**1. Convolutional Neural Network**

A convolutional neural network is a specific kind of neural network with multiple layers. It processes data that has a grid-like arrangement then extracts important features. One huge advantage of using CNNs is that you don't need to do a lot of pre-processing on images.

**Architecture of CNN**

A typical CNN has the following 4 layers

1. Input layer
2. Convolution layer
3. Pooling layer
4. Fully connected layer

**Input layer**

The input layer represents the input to the CNN. An example input, could be a 28 pixel by 28-pixel grayscale image. Unlike FNN, we do not “flatten” the input to a 1D vector, and the input is presented to the network in 2D as a 28 x 28 matrix. This makes capturing spatial relationships easier.

**Convolution layer**

The convolution layer is composed of multiple filters (also called kernels). Filters for a 2D image are also 2D. Suppose we have a 28 pixel by 28-pixel grayscale image. Each pixel is represented by a number between 0 and 255, where 0 represents the colour black, 255 represents the colour white, and the values in between represent different shades of grey

Convolution operator has the following parameters:

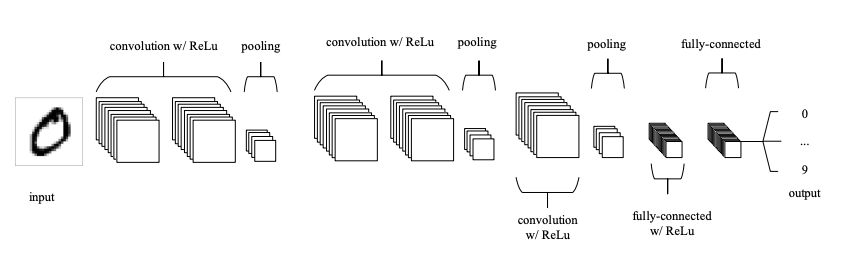
1. Filter size
2. Padding
3. Stride
4. Dilation
5. Activation function

**Pooling layer**

The pooling layer performs down sampling to reduce the spatial dimensionality of the input. This decreases the number of parameters, which in turn reduces the learning time and computation, and the likelihood of overfitting. The most popular type of pooling is *max pooling*. It’s usually a 2 by 2 filter with a stride of 2 that returns the maximum value as it slides over the input data (similar to convolution filters).

**Fully connected layer**

The last layer in a CNN is a fully connected layer. We connect all the nodes from the previous layer to this fully connected layer, which is responsible for classification of the image



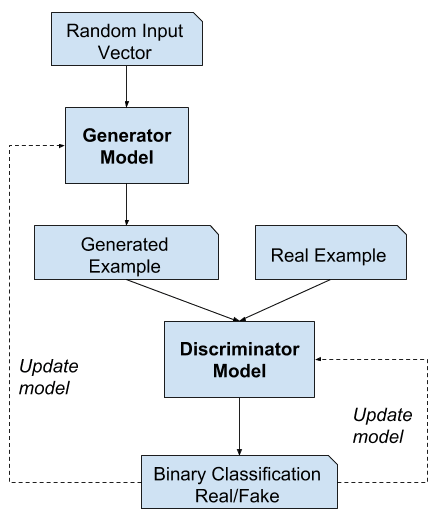
**2. Generative Adversarial Network**

Generative Adversarial Networks. Introduced in 2014 by Ian Goodfellow, GANs have shown tremendous success over the last few years in the field of Computer Science research with its ground-breaking applications. GANs were first used and lauded for generating realistic images but now they have evolved to open a new field of research itself.

It consists of two neural networks:

1. Generator - This model uses a random noise matrix as input and tries to regenerate data as convincing as possible. Its goal is to generate realistic enough images to fool the discriminator network.
2. Discriminator - This model determines whether the input image is real or fake. It gives feedback if the image is predicted to be fake and the generator model uses that information to perform better in the next epoch.

If the training of the GAN model is going well then, the Discriminator model will find it increasingly difficult to distinguish between real and fake images, predicting the fake images as real as well leading to a lot of False Positives predictions



* **SWOT Analysis**

A SWOT analysis is a planning framework that can be used to identify the strengths, weaknesses, opportunities, and threats of their strategic initiatives.

**Strength:**

* DCGAN applies stridden convolutions on the discriminator and fractional convolutions on the generator to substitute pooling layers. Features are typically extracted with CNN.
* To resolve the gradient problems DCGAN uses the Batch Standardization Algorithm. The BN algorithm fixes weak initializations, brings the gradient to each layer, and restricts the generator from collecting all samples to the equivalent stage.
* DCGAN uses various activation functions, including Adam optimization, Rectified Linear Unit (ReLU) activation function, and Leaky ReLU.
* The results show the better performance of DCGAN and confirm the capability of the GAN structure in generating samples. DCGAN is generally considered as the standard when associated with different GAN models.

**Weakness:**

* The model parameters oscillate, destabilize, and never converge.
* The generator collapses which produce limited varieties of samples, and highly sensitive to the hyperparameter selections.
* The discriminator becomes extremely successful so that the generator gradient disappears and receives nothing. Unbalance within the generator and discriminator causing overfitting

**Opportunities:**

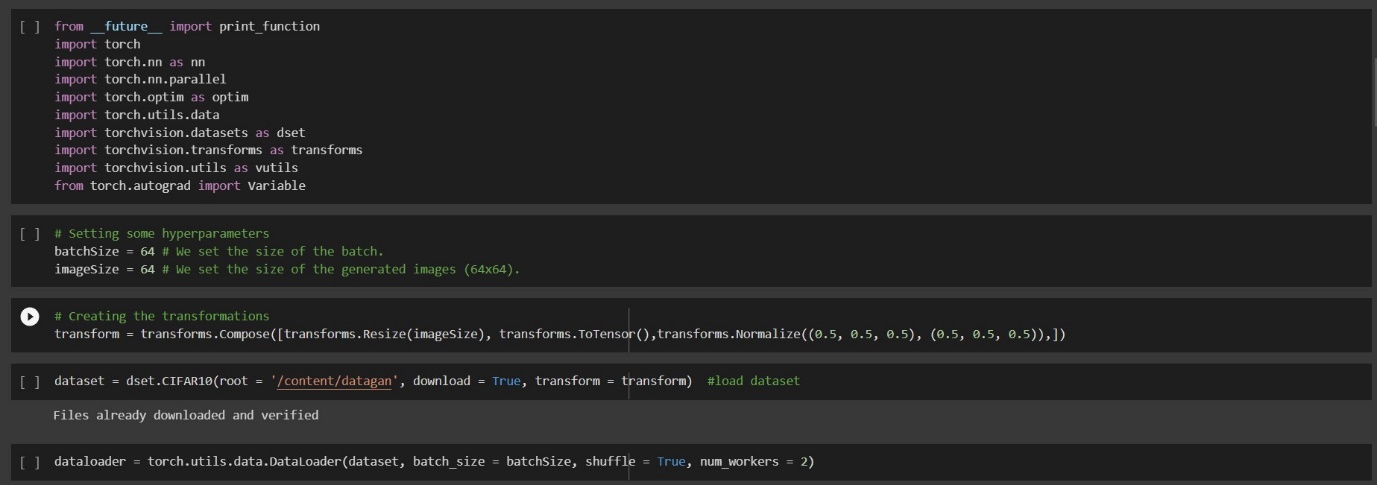
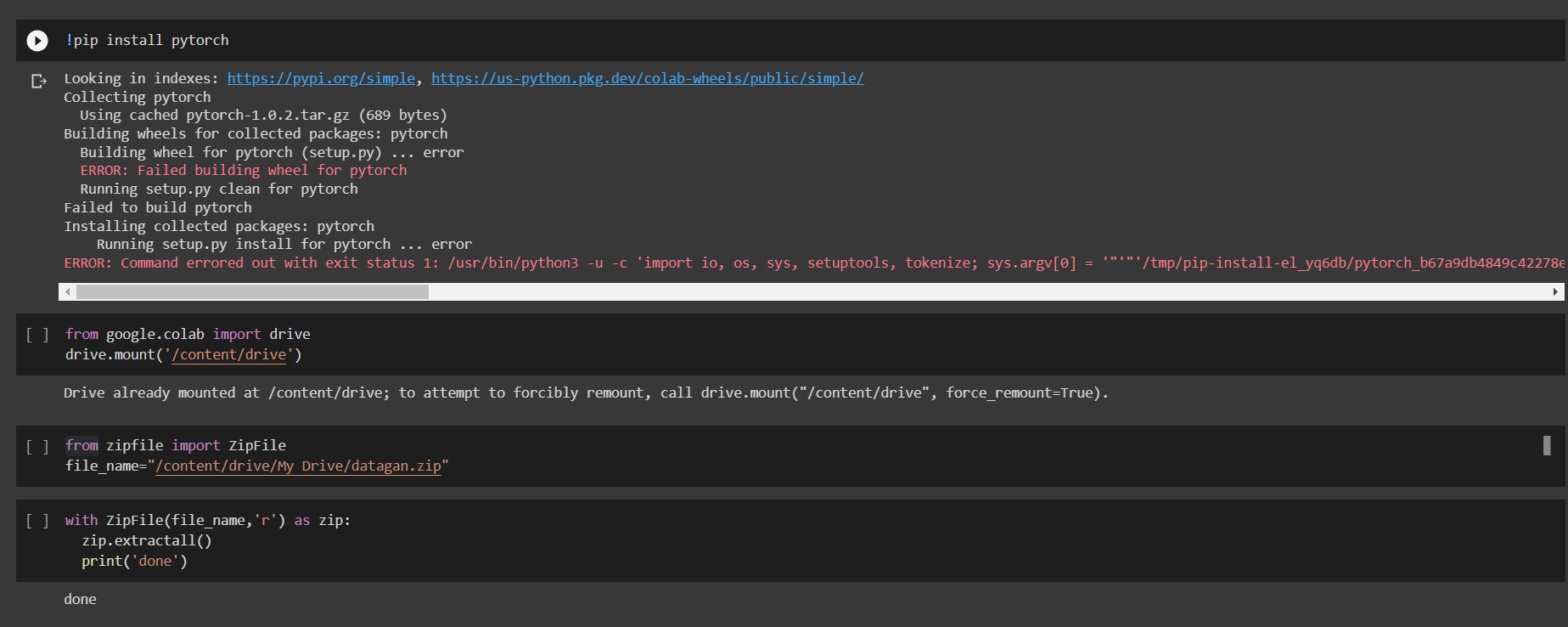
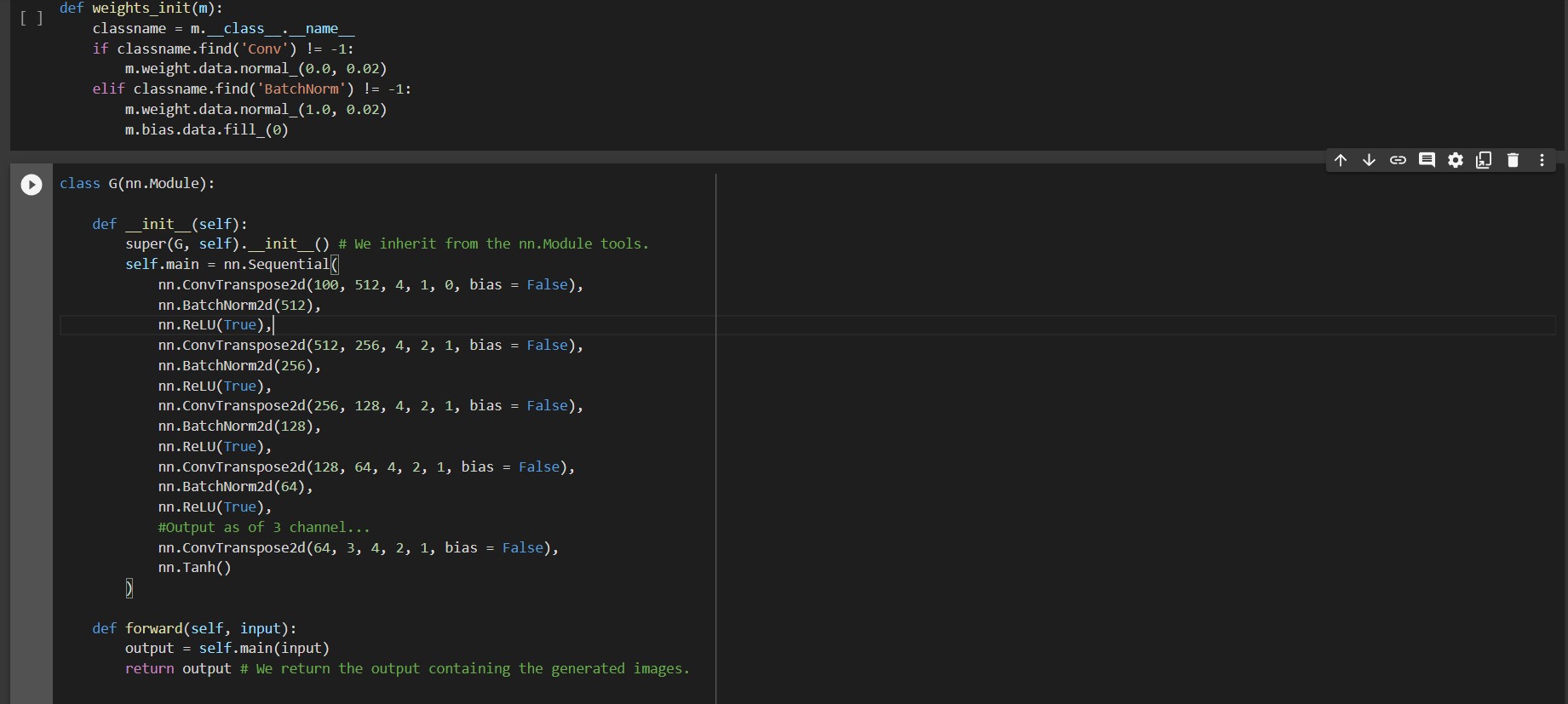
* Generative modeling involves using a model to generate new examples that plausibly come from an existing distribution of samples, such as generating new photographs that are similar but specifically different from a dataset of existing photographs.
* The generative model can then be used to create new plausible samples on demand.

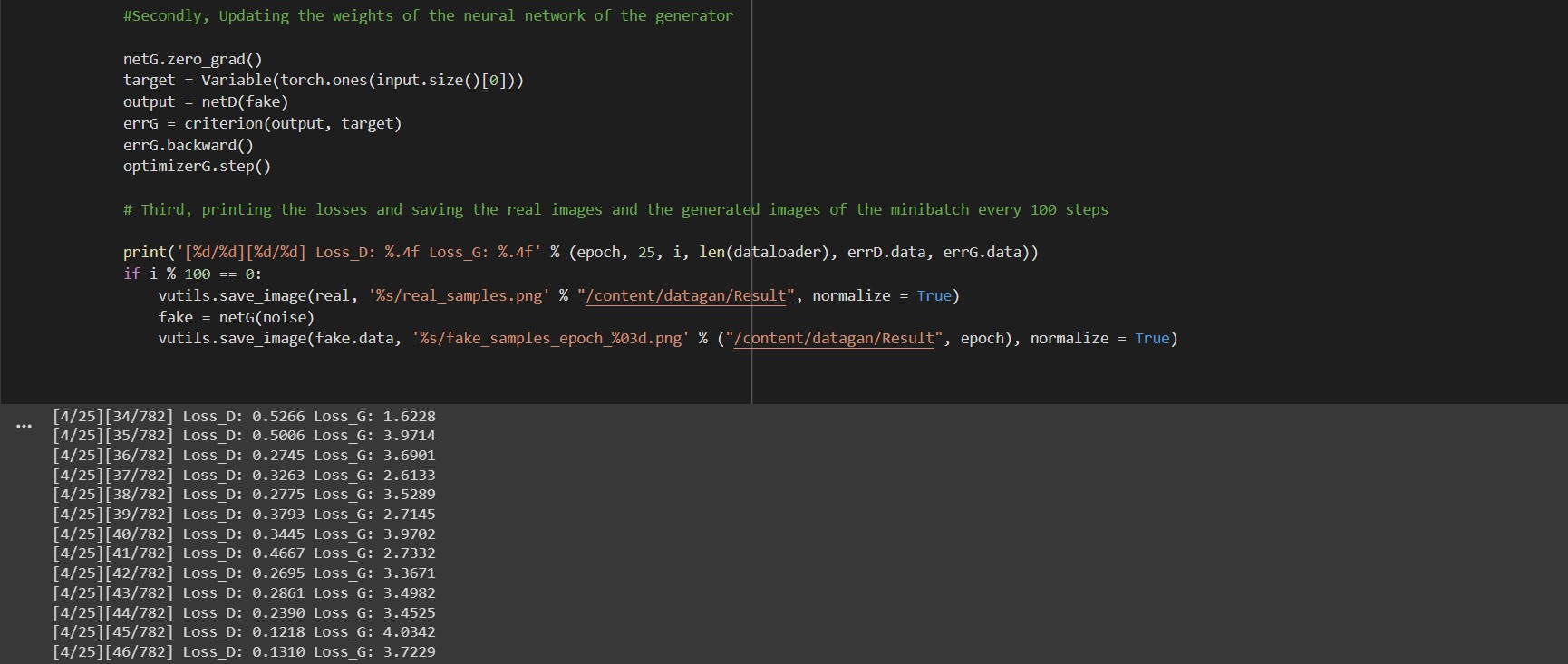
**Threats:**

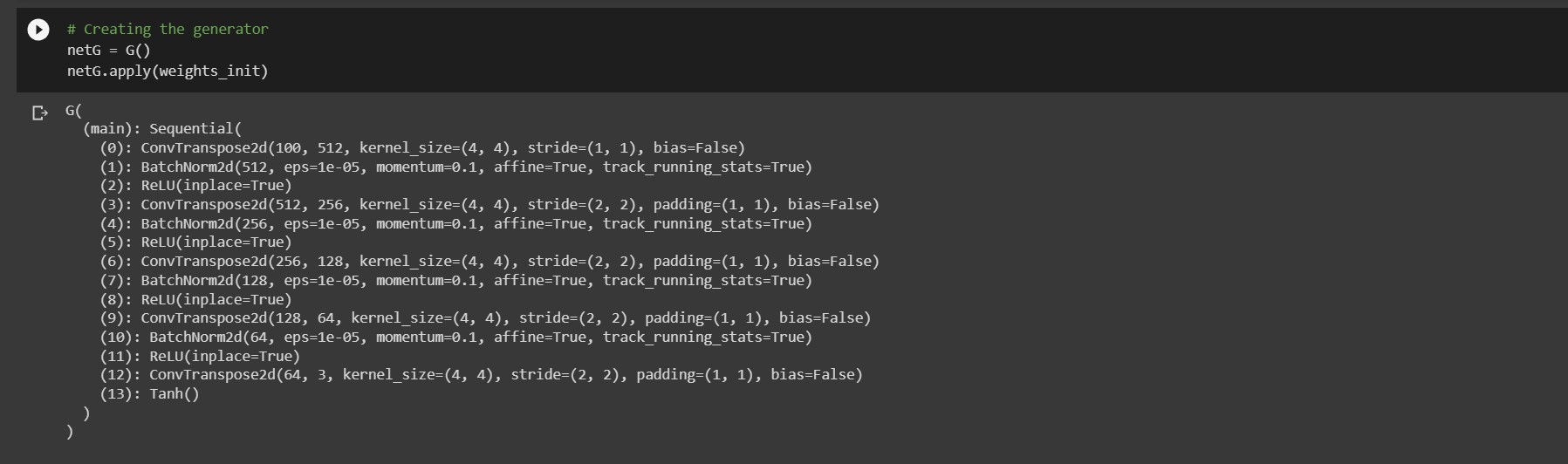
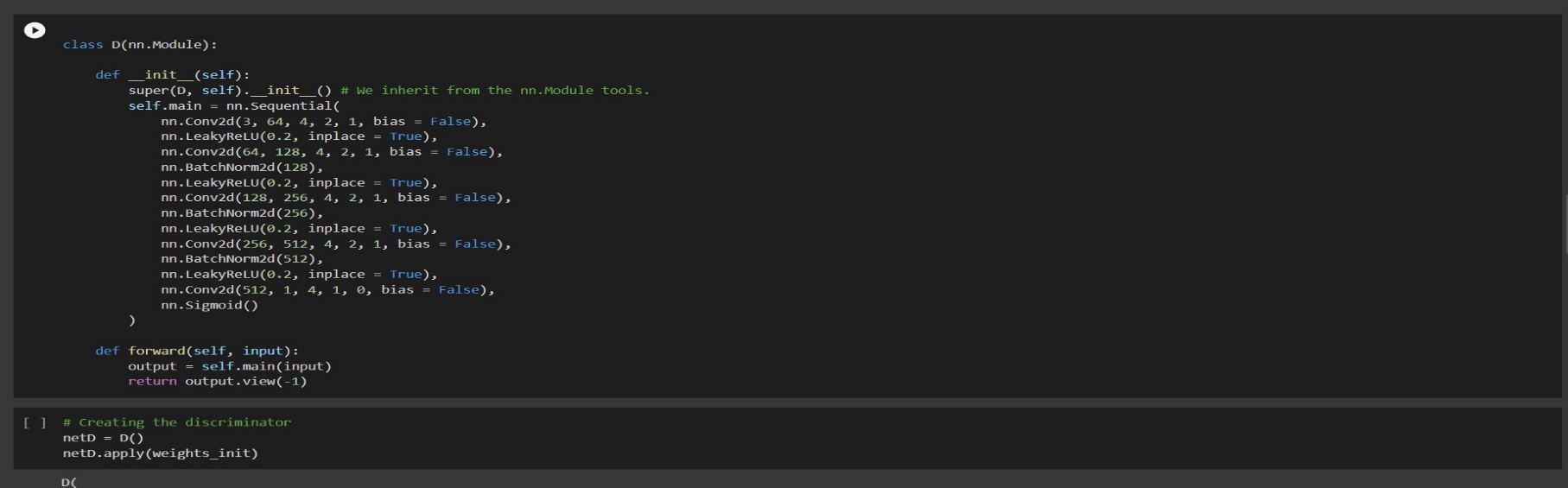
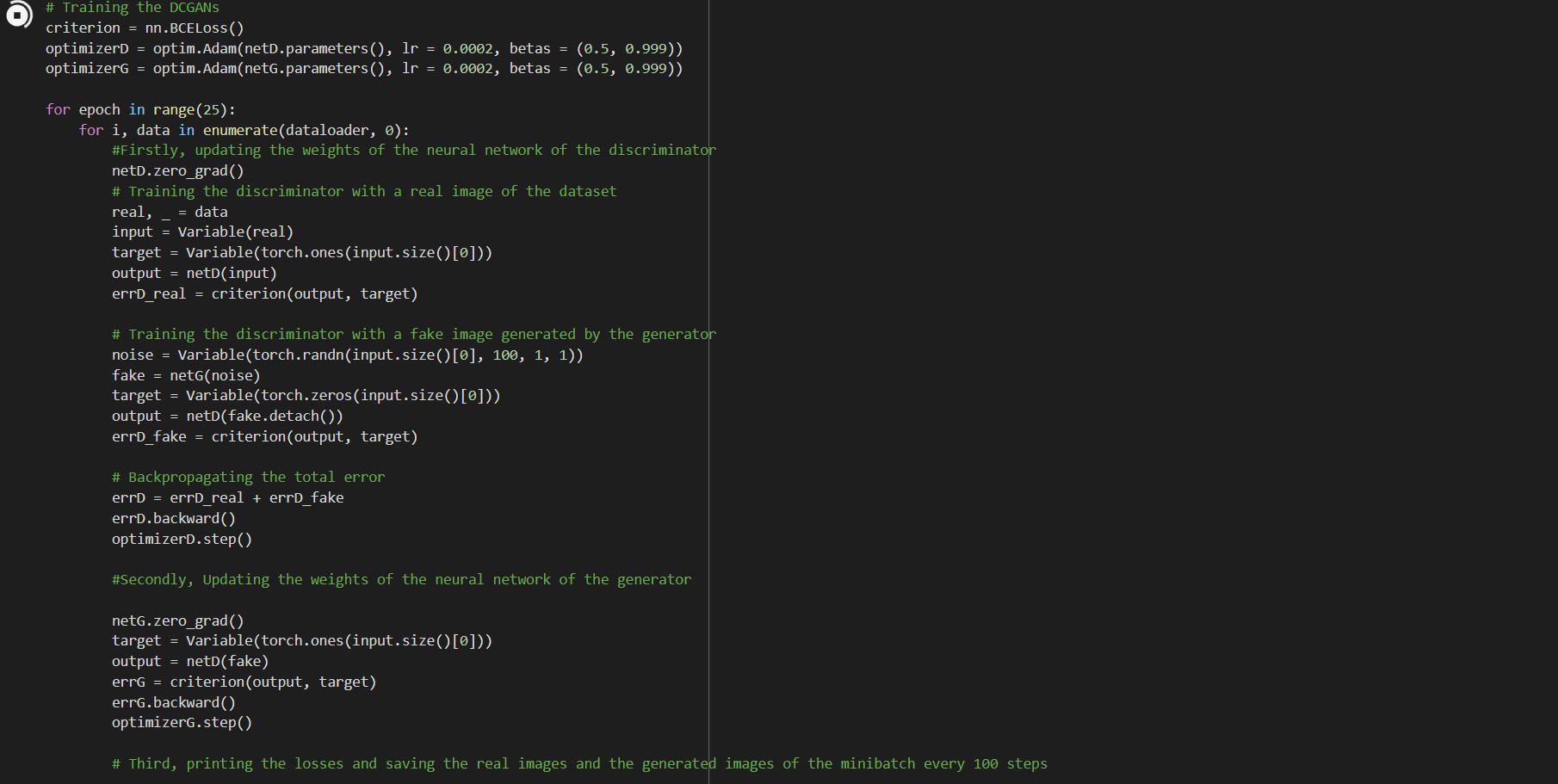
* Training of models is time consuming
* Need high configured machine to train models

**PROJECT IMPLEMENTATION**

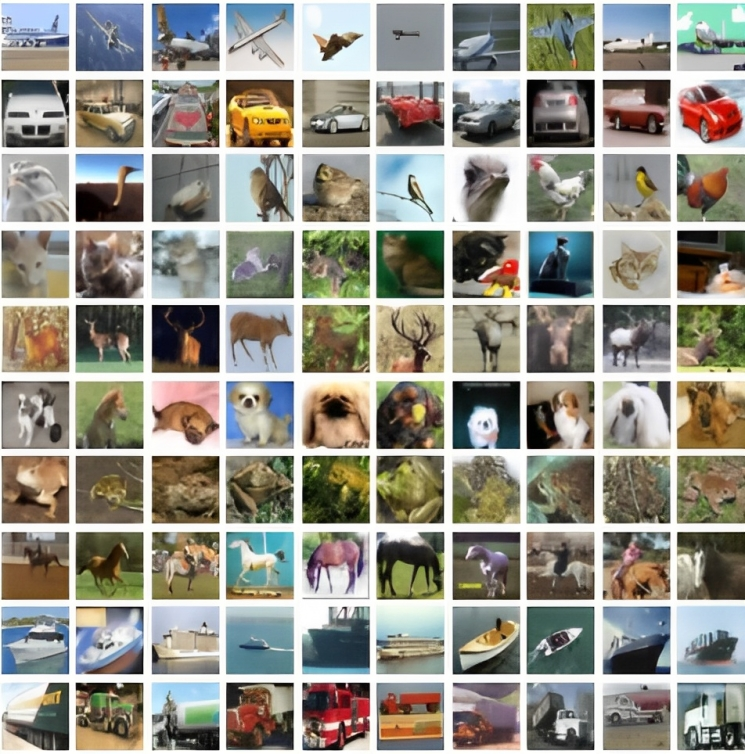
* In this project, we have used deep convolutional generative adversarial neural network (DCGANs).
* We load the dataset which is CIFAR-10 that has 60000 images of resolution 32x32. It has 50000 training image and 10000 test images.
* In DCGANs, it mainly composes of convolution layers without max pooling or fully connected layers. It uses convolutional stride and transposed convolution for the down sampling and the up sampling.
* Firstly, we create generator and we propagate forward with an inversed convolution called deconvolution. We normalize all the features along the dimension of the batch. We apply a ReLU rectification to break the linearity. And further, we add inversed convolutional layer twice and again perform same steps and after four times complete steps, we use Tanh rectification to break the linearity and stay between -1 and +1.
* Secondly, we create discriminator and we propagate forward with convolution. We normalize all the features along the dimension of the batch. We apply a leaky ReLU rectification. And further, we add convolutional layer and again perform same steps and after four times complete steps, we use sigmoid rectification to break the linearity and stay between 0 and 1.
* We measure the error between the prediction and the target and update the weights and back propagate the model
* We run 25 epochs with 782 mini batch.
* Finally, we store the resultant sample image in folder after every epoch.

**PROJECT CODE**





**Resultant Image**



**REFERENCES**

Bergstra, James and Bengio, Yoshua. Random search for hyper-parameter optimization. JMLR, 2012.

Coates, Adam and Ng, Andrew. Selecting receptive fields in deep networks. NIPS, 2011.

Coates, Adam and Ng, Andrew Y. Learning feature representations with k-means. In Neural Networks: Tricks of the Trade, pp. 561–580. Springer, 2012.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009.

IEEE Conference on, pp. 248–255. IEEE, 2009. Denton, Emily, Chintala, Soumith, Szlam, Arthur, and Fergus, Rob. Deep generative image models using a laplacian pyramid of adversarial networks. arXiv preprint arXiv:1506.05751, 2015.