ECEN 649: Pattern Recognition - Project Report

Samyuktha Sankaran (UIN: 327006025)
VCS Kaushik(UIN: 827004640)

Abstract

In this project, we trained a CNN model to classify the images in the CIFAR10 dataset. We attempted to show what the neural network learns through these classification using General Adversarial Networks(GAN).

1 Introduction

The key concept behind this project is on **Deconvolutional Layer**. The DeConv net actually performs a convolutional transpose operation on the image. This term is quite ambiguous as there is actually a "Deconvolutional layer", that exactly reverses the operation of convolution layer. When we use neural nets to generate images, we typically perform an Up-sampling to enhance the image from lower to higher resolution. We use a transposed convolution layer to perform the Up-sampling operation.

$$f = \sum_{i=0}^{n} w_i * x_i + b \tag{1}$$

- w - weights of the kernel
- x - pixels of the input image
- b - bias term

In a convolution operation, we pass a filter across the image and perform a dot product and add a bias term. This generates the activation map, which is passed to the subsequent layers. A convolution operation can be viewed as a **Many to one relation**. Going backwards on a convolution operation is transposed convolution. It is a **one to many relation.** It forms the same connectivity as a convolution layer but in the reverse direction.

General Adversarial Networks(GAN) was introduced by Ian Goodfellow in his paper. A discriminative algorithm tries to classify the data instances given its features and predicts a label where it belongs. Common example is spam filters applied on e-mails. A generative algorithm works in the opposite manner. Given a label or class, they attempt to predict it's feature. Our aim in this project is to classify the images of CIFAR 10 dataset and generate the image for a given class that would maximize the score of the respective class. We make attempt to study what the neural network tries to learn at each stage using GAN and transpose convolutions.

1.1 Data set

We worked with the CIFAR 10 data set in this project. It consists of 60,000 images in 10 classes, split into train and test data in ratio 5:1. The test set consists of about 1000 randomly selected images from each class and the train set consists of about 5000 images per class. This is a standard data sets, popularly used for various machine learning problems. It doesn't require much of preprocessing and there are no missing components to it, which makes it easy and reliable to work with. The classes in the data set are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

1.2 Software and Packages

Platform: Google Colaboratory.
Packages: Tensorflow, Keras.
Language: Python.

2 Procedure

The project can be fairly divided into three parts.

2.1 Training a CNN

The keras package has a library to download the cifar-10 dataset. We load the train and test data onto the pandas data-frame. The first step in any machine learning problem is to pre-process the data. Here, the predictors are 32x32x3 pixels which are converted to float and normalized to have the values within the range[0,1]. One hot encoding is applied to the labels of the data set.

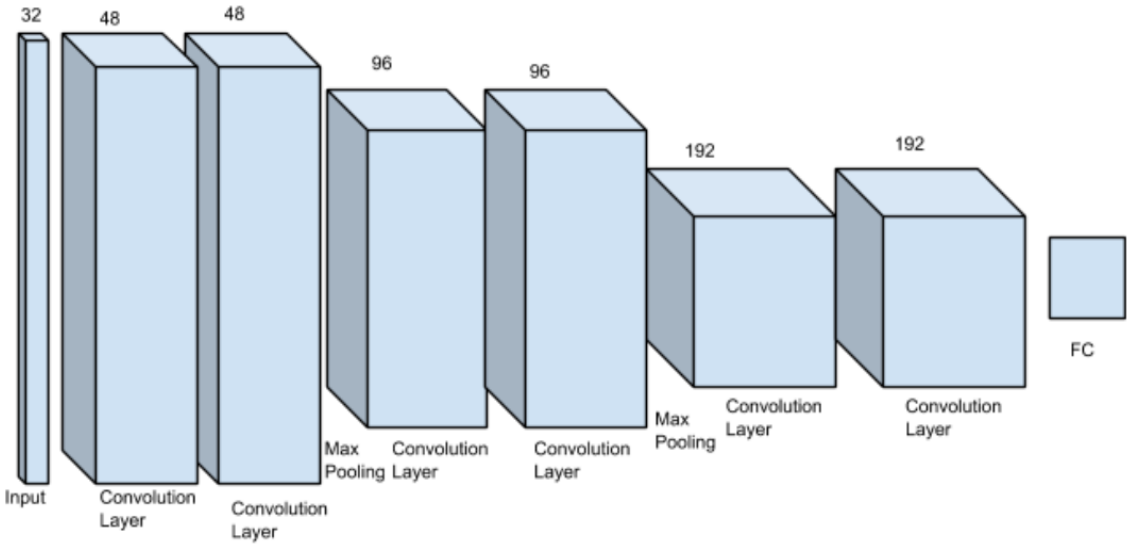

Figure 1: CNN to perform classification

The model is trained on the datasets with 20% of validation set and yield an accuracy of 83.96%

2.2 GAN to generate images

General Adversarial networks comprises of two components, namely the generator and discriminator.

2.2.1 Generator

The generator takes in random noise, in this case **Uniform random noise** and transform it to an image of (32x32x3) volume. The generator learns from the discriminator and generates images that looks closer to the images in the data sets. The configuration of the generator is as follows:

- It is initialized by the Xavier uniform initializer.

- Kernel size = 5X5
- stride = 2
- Typically 'ReLu' activation is used in CNN architectures. However here we have used Leaky ReLu activation with a slope of 0.2 We apply Leaky Relu activation for all the layers except the last layer.
- Batch normalization is applied to every layer.

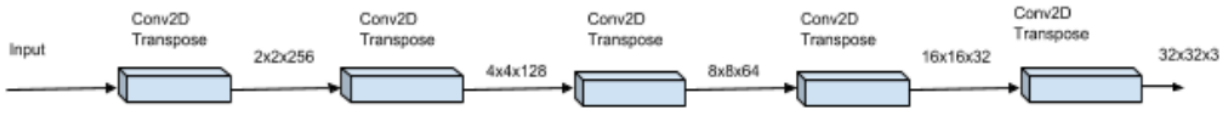

Figure 2: Generator layout

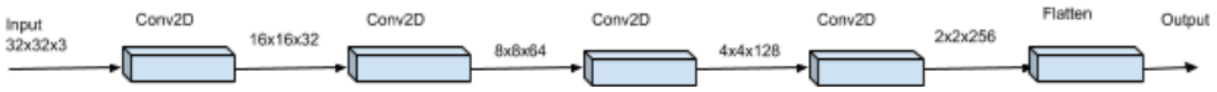

Figure 3: Discriminator layout

2.2.2 Discriminator

The goal of the discriminator is to determine if the image is a real image or from the generator. There are no pooling layers here.

- Kernel = 5x5; Stride = 2
- Batch normalization is applied for each layer except the first layer.
- Leaky Relu activation is used except for the last layer.
- The sigmoid function is used in the last layer to determine the probability that the image is a generated image.

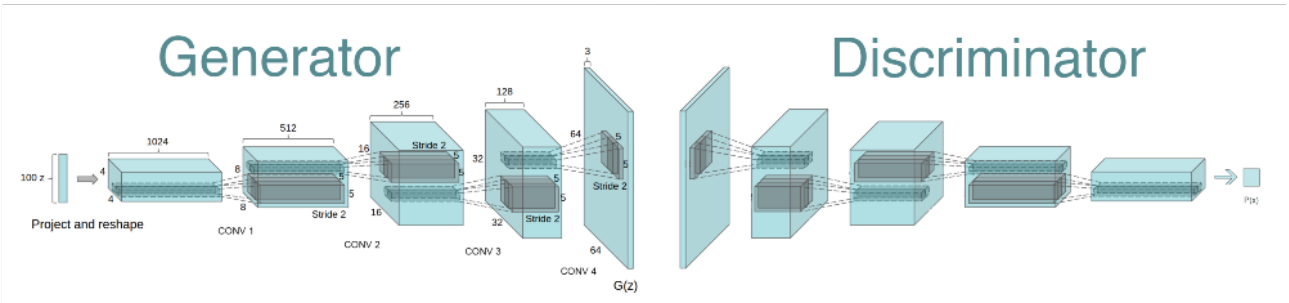

Figure 4: GAN Architecture

2.2.3 Procedure

- We generate the images using the generator and pass it through the discriminator and obtain the probability of the generated image.
- We next pass the real images from the data set and obtain the probability for real images.
- The generator and discriminator losses are computed in the optimizer function.
- The generator loss is sigmoid cross entropy between the probability of the generated images and ones. The discriminator loss has two components - one is the cross entropy of the probability of the generated images and ones and second is the probability of real images and zeros.
- The purpose of the generator and discriminator optimizer is to minimize the entropy between the real images and positive label. Also the discriminator tries to minimize the entropy between the fake images and negative labels. The Adam optimizer was used here. The network is trained for 100 epochs with batch size as 128.

2.2.4 Results

The results of the images through the GAN network after every 10 network is as follows. We managed to generate one image per class.

Figure 5: Epoch 10

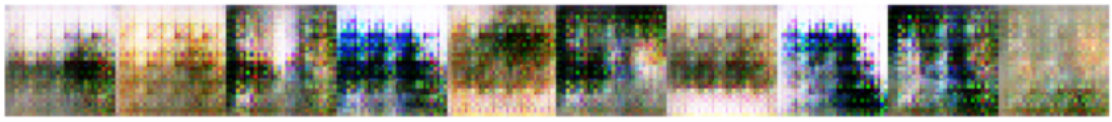

Figure 6: Epoch 20

Figure 7: Epoch 30

Figure 8: Epoch 40

Figure 9: Epoch 50

Figure 10: Epoch 60

Figure 11: Epoch 70

Figure 12: Epoch 80

Figure 13: Epoch 90

Figure 14: Epoch 100

The corresponding adversarial and discriminator losses are as follows:

Epochs	**Discriminator Loss**	**Adversarial Loss**
10	3.356	0.197
20	3.781	0.092
30	4.599	0.125
40	2.797	0.137
50	6.337	0.068
60	1.488	0.383
70	5.692	0.014
80	5.851	0.011
90	5.931	0.015
100	8.177	0.029

Inference : Initially a uniform random is noise is passed to the Generator, and the image generated from the generator and the image from the dataset are passed to the discriminator. After multiple iterations(or epochs) the generator starts to generate images that are closer to the real image.

2.3 Integrating the classifier and GAN networks

In order to meet our second goal which is to generate images that results in maximum score to a particular class of the network, we opted to combine the previous two models. In this model the Discriminator of the GAN is replaced with a classifier network. We train the generator to create images which pertain to one class of the classifier, hoping to get an image which maximizes the class score of that class. Unlike a conventional GAN where the desired output is an image which, the discriminator detects as a real image, the desired output in this case is for the classifier to detect the generated image to belong to a particular class. We opted for this approach as GAN is well known for their image generation capabilities, and we wanted to check if we could train a GAN to generate images of a specific class. We also hoped that by further training we can enable the GAN to generate images which maximize the class score.

We pass the dataset through the classifier and then through the GAN. Here an extra parameter which contains the labels of the dataset is added. A 1-D variable which has all its entries as zero except for the class predicted by the classifier. The labels are fed to the generator along with input noise. The generator generates an image, which ideally when passed through the classifier should have a probability of 1 for the said class. However we are able to maximize the score of that particular class, but not achieve full probability. These are the images obtained for the airplane class.

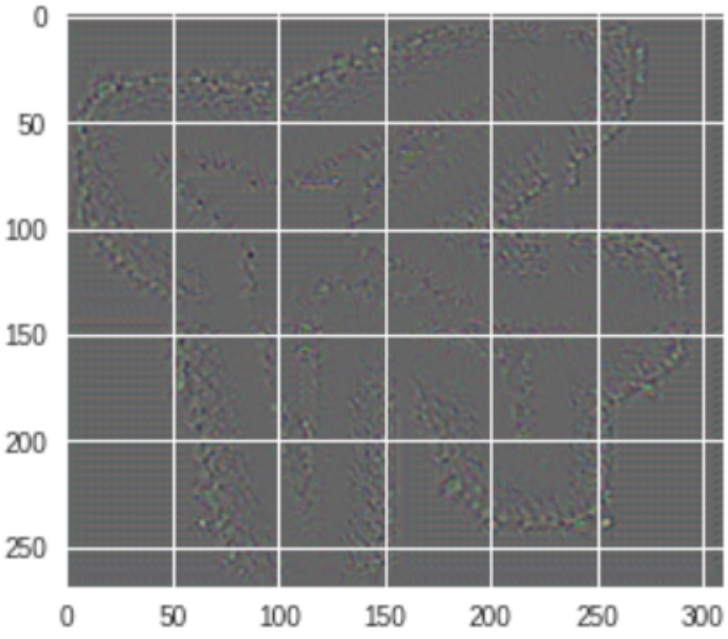

Figure 15: Class: airplane; Epoch : 10

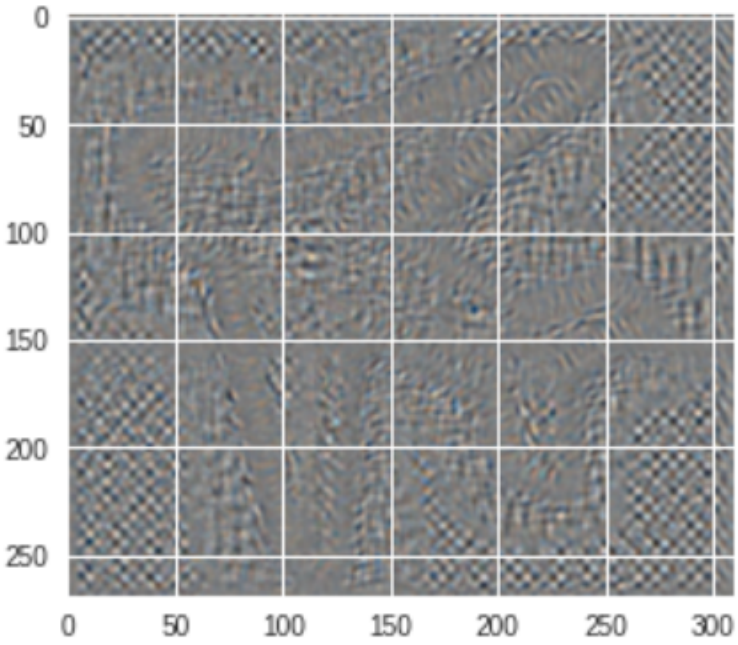

Figure 16: Class: airplane; Epoch : 30

3 Future work

Although we tried to demonstrate our understanding of the neural network layers, we could not succeed in generalizing it for all classes of the data set. Currently, we are able to obtain the images for a single class (here airplane), but we could not generalize for the entire data set.

Google's **Deep Dream** algorithm is a good approach to our problem statement. It is a way to visually represent what their neural network, Google Net thinks of the image. Using GAN, they generate hallucinogenic images from the real image.

Brief description on Deep dream : An image is fed to the deep dream and it starts to look for every objects it has been trained till now, like dogs, jellyfish, etc. The process is repeated until it is fully classified as a dog or any other chosen object. Say earlier the model was 50% sure that the image is a dog, now it will be 70% sure.

We attempted to do back-propagation based on this idea. But some concepts were beyond our scope and using GAN, we were able to accomplish the same for one class for a few samples. The difficulty lie in applying back-propagation to the CNN layers. We used the keras and tensorflow packages which has APIs for CNN and back-propagation is carried on internally through these packages. Hence it was hard for us to unravel the function. Also we were not able to determine an apt loss function to achieve the results. These are some of the results obtained while trying to generate images for all the classes.

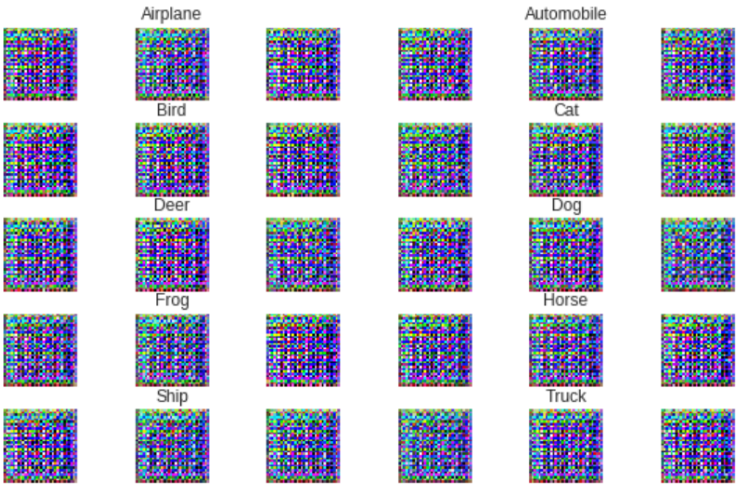

Figure 17: Epoch : 10

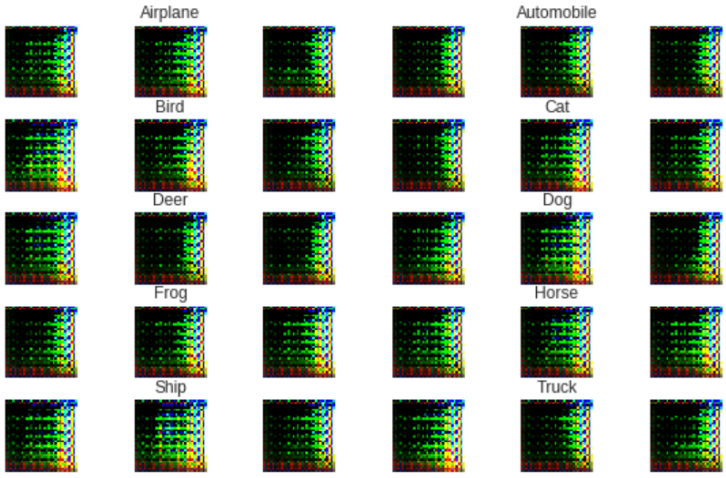

Figure 18: Epoch : 30

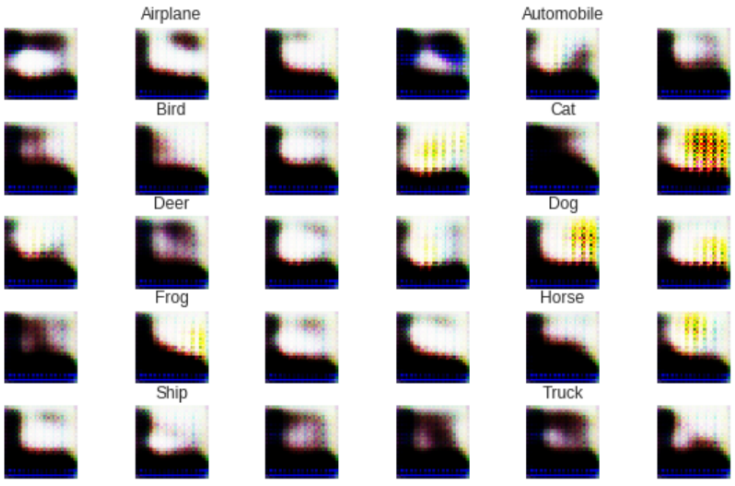

Figure 19: Epoch : 30

4 Conclusion

To summarize, we built a CNN model to classify the CIFAR 10 dataset, with an accuracy of 83.96%. Following which we demonstrated what the neural net learns by implementing a GAN architecture. We generated images for each class with discriminator loss of 8.177 and adversarial loss of 0.029 obtained over 100 epochs. We then passed the data to the classifier and GAN and generated images for a single class that can increase the weight of that class.

References

[1] http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes

[2] https://people.eecs.berkeley.edu/ jonlong/long_shelhamer_fcn.pdf

[3] https://papers.nips.cc/paper/5485-deep-convolutional-neural-network-for-image-deconvolution.pdf

[4] An Introduction to Statistical Learning with Applications in R by Gareth James and Daniela Witten.

[5] Understanding Machine Learning: From Theory to Algorithms by Shai Ben-David and Shai Shalev-Shwartz.