**Image Classification Using Cifar-10**

**CPS-584 PROJECT REPORT**

****

**Project Group:**

|  |  |
| --- | --- |
| KARTHIK NIMMA  MAHEEN TANVEER | 1016505890  1016732640 |

**Project Advisor/Instructor:**

Dr. Mehdi Zargham

**April 2020**

**University Of Dayton 300 College Park, Dayton, OH 45469**

**ABSTRACT**

Image recognition is a machine learning method and it is designed to resemble the way a human brain function. With this method, the computers are taught to recognize the visual elements within an image. By relying on large databases and noticing emerging patterns, the computers can make sense of images and formulate relevant tags and categories.

This project aims towards the classification of images of ten different objects. The cifar-10 dataset is used to train and test the network. Convolutional neural network (CNN) is the most popular approach for image recognition. The network uses a Rectified linear activation function to calculate the output of each neuron. Pre-processing of the data is done to feed it to the CNN for further training. The performance of the model is compared with and without image augmentation, and network architecture is applied to another dataset to check its performance.

##### ACKNOWLEDGEMENTS

We express our immense pleasure by acknowledging the guidance of our instructor **Dr. Mehdi Zargham, Chairperson of Department of Computer Science; University of Dayton** has provided us during this project. His timely feedback and assistance throughout this project were highly appreciated. It was an honor working under his supervision.

We would also like to be grateful to our teaching assistant for his assistance and help.

Finally, we also like to express our earnest gratefulness to all friends and classmates who contributed their valuable advice.

|  |
| --- |
| **CONTENTS** |
| 1. Introduction |
| 1. Project Description |
| 1. Purpose of the Project |
| 1. Source of Data |
| 1. Survey of Current Methods |
| 1. Proposed Method |
| 1. Details of Implementation |
| 1. Results |
| 1. Conclusion and future work |
|  |

**INTRODUCTION:**

Object recognition is an important subfield in computer vision. For humans interpreting the visual world comes easy. To recognize and classify objects in images is an easy task for them, but it has proved to be a complex and challenging one for machines and therefore image classification has been an important task within the field of computer vision.

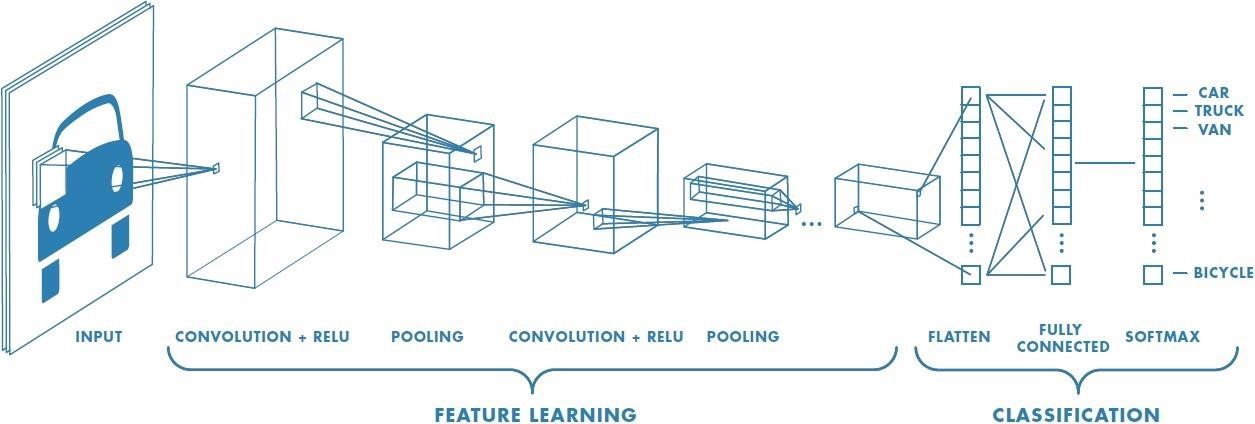
Image Recognition is the process of identifying what an image depicts. Image classification is one of the most fundamental problems in Machine Learning. With the development of deep Convolutional Neural Network (CNN), researchers have achieved good performance on the image recognition task. The branch of image processing has proved to be a very useful technology for analyzing certain behaviors.

**PROJECT DESCRIPTION:**

Our project basically classifies between the ten classes of Cifar-10. For this project, we have used the keras and tensorflow libraries and have created a neural network and trained it. Finally, we tested the output and evaluated the results that we get. The neural network which we have used in our project is CNN (Convolutional Neural Network).

**CONVOLUTIONAL NEURAL NETWORK:**

CNN gained much popularity in the past years bringing revolution in many field. It has been particularly designed for image processing and is the most popular among such technologies. The most important aspect of using CNN is that it reduces the number of parameters required for Artificial Neural network (ANN). So many researchers and developers admired this approach for solving sophisticated problems from pattern recognition, image processing, speaker recognition and many other projects. The most significant aspect of CNN is of its in-depth feature extraction and detection. A description of CNN is shown in figure.



CNN is a special type of Artificial Neural Networks that offer human-like results in image classification tasks. The representations learned by CNN are similar to how the human visual layers represent visual information: the first convolutional layers extract low-level features, such as edges and blobs, and the latest layers assign the semantic part to the image.

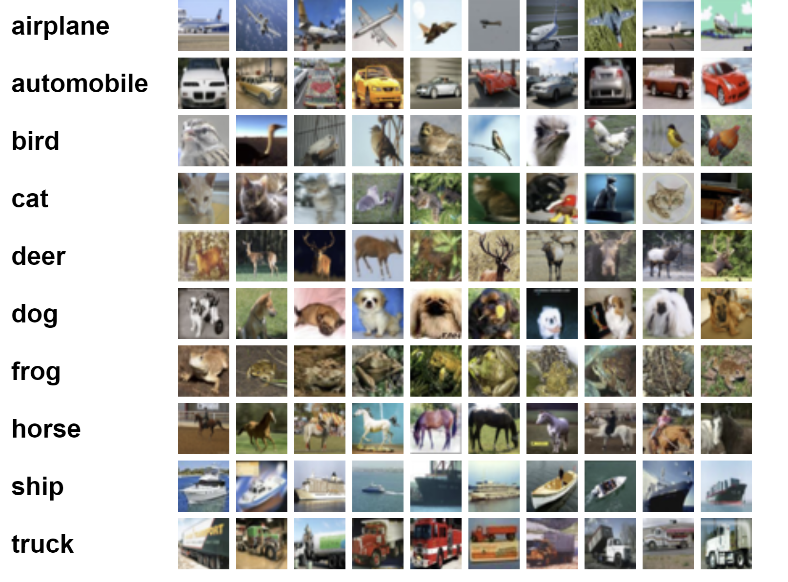
**PURPOSE OF THE PROJECT:**

The CIFAR-10 small photo classification problem is a standard dataset used in computer vision and deep learning. Although the dataset we are using for the image classification is effectively solved, but it can be used as the basis for learning and practicing of how to develop, evaluate, and use convolutional deep learning neural networks for image classification from scratch.

We have found other approaches to image classification of this particular dataset, but we wanted to explore what we learned from our deep learning course. Specifically, we used convolutional neural network model for classification. By doing this project we have done a lot of learning and this was a great chance to delve into this model further through this classification project.

**SOURCE OF DATA:**

The source of our data is CIFAR-10 dataset. Cifar-10 is a standard computer vision dataset used for image recognition. It gives us natural color images. The dataset consists of 60,000 32x32 color images used for image recognition. There are 50,000 images in the training set and 10,000 in the test set. It consists of 10 object classes, with 6000 images per class. There are 50000 images in the training set and 10000 images in test set.

The **label** classes present in the dataset are namely **airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.** All these classes are mutually exclusive, and both the training and test set are labeled for training and testing.

**Survey of Current Method:**

The most popular method out there is known as **Transfer Learning.** In Transfer Learning, the knowledge of the already trained machine learning model is applied to another related problem. For example, if you have trained a simple classifier to predict whether a picture contains a backpack, you can use the knowledge that the model gained during its training to identify other objects such as sunglasses. Thanks to this, we’re basically trying to use what we’ve learned in one task to improve generalization in another. We transfer the knowledge that the network has learned in task A to a new task B.

However, Transfer Learning works only if the features learned from the first task are generic, which means that they can also be useful in other related tasks.

**What is the issue?**

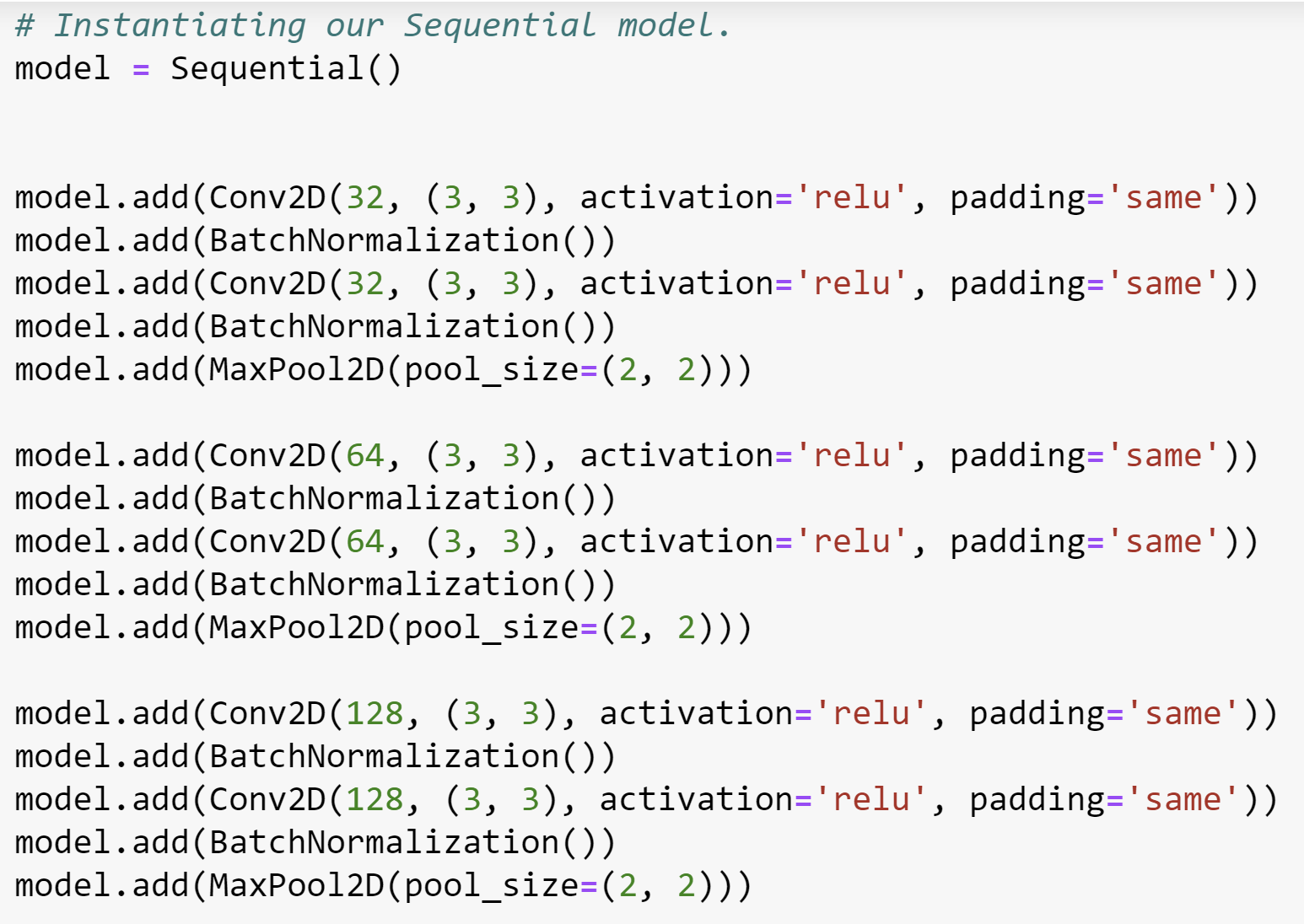
* First of all, the distribution of the training data which your pre-trained model has used should be like the data that you are going to face during test time or at least don't vary too much.
* Second, the number of training data for transfer learning should not be very less compared to the data on which you plan to use the pretrained model as this may lead to overfitting.
* Next problem is that you cannot remove layers with confidence to reduce the number of parameters. If you remove the convolutional layers from the first layers, again based on experience, you won't have good learning because of the nature of the architecture which finds low level features. Densely connected layers and deep convolutional layers can be good points for reduction, but it may take time to find how many layers and neurons to be diminished in order not to overfit.
* If the pretrained models does not have common class labels the convolutional layers after pooling layers usually keep information which may be irrelevant to your task. There is a downside for this interpretation.

**Proposed Method:**

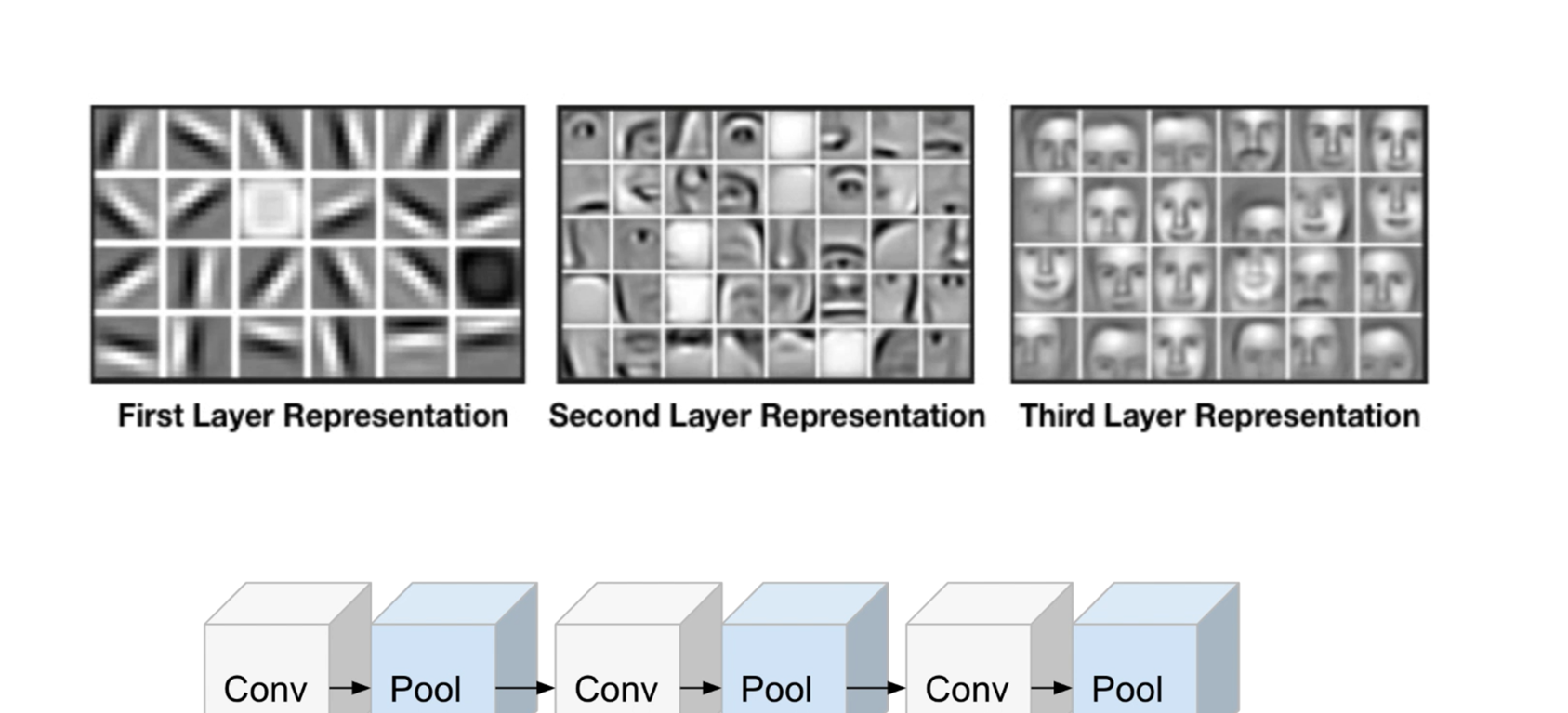
We have created a model which can classify our dataset with an accuracy of 92% and validation accuracy of 88%. The model consists of six convolutional layers with each layer followed by batch normalization function to regularize the output from each layer. Batch normalization allows us to reach higher learning rates and at the same time reduce outlier activation functions. This is possible because outliers rarely occur and in a batch the effect of outliers will be compensated with normalization. MaxPooling of 2x2 is done after each couple of convolutions. Each subsequent couple of convolutional layers have twice the number of filters than the previous. The CNN is connected to a Dense neural network (DNN) which contains 1024 neurons in the hidden layer with a Dropout probability of 0.3. Dropout operation randomly turns off a neuron in the layer to help the model generalize better and prevent overfitting. Finally, the output layer contains 10 neurons, each neuron corresponds to a class in the dataset.

**Implementation:**

1. The Keras dataset library provided convenience to load and split the dataset into training and testing sets.
2. The train and test sets are normalized to a value between 0 and 1 by dividing each pixel by of an image by 255. This is done to decrease the size of the input which enables faster processing through the network.
3. Next, we instantiate a Sequential model and then add six Convolutional layer separated by Batch normalization and Maxpooling.



The **Reason to choose** increasing size of filter is because it is proven that CNN extracts features in a hierarchical manner i.e. **low – high** level.



A screenshot of a cell phone

Description automatically generated**Kernel Size**

An even sized filter (2x2 or 4x4) isn’t used because this created distortion in resulting output image .

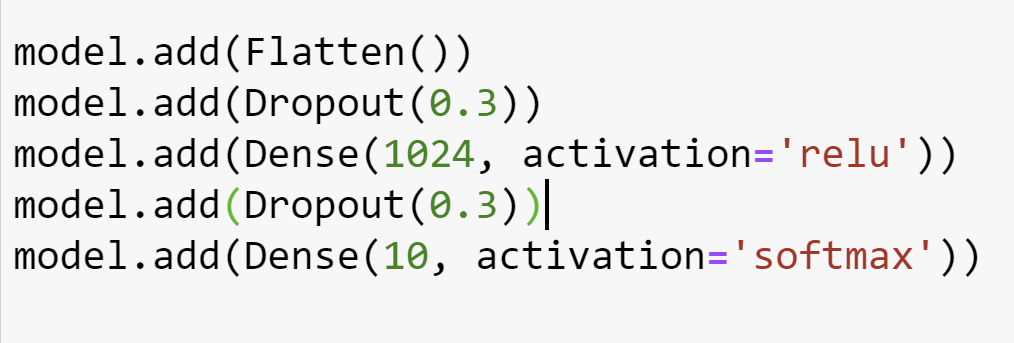
We are using a 3x3 filter for our project as our input image size is less(32x32)

**Batch Normalization:**

Generally, learning rates are kept small, such that only a small portion of gradients correct the weights, the reason is that the gradients for outlier activations should not affect already learned activations Hence higher learning rates can be used to accelerate the learning process.

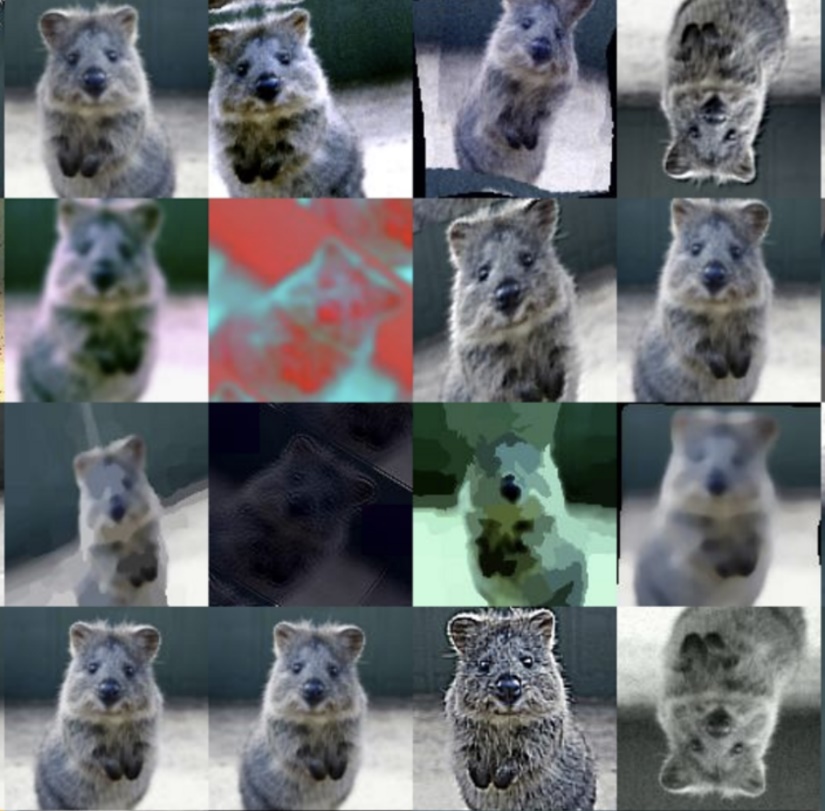
**Dense Neural Network:**

The hidden layer contains 1024 neurons which are connected to 10 neurons in the output layer. The number of neurons in the output layer equals the total number of classes.



**Dropout**

Dropout are used to prevent the model from overfitting. This is achieved by randomly turning of few neurons in a layer. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much.

**Image Augmentation:**

In order to combat the high expense of collecting thousands of training images, image augmentation has been developed in order to generate training data from an existing dataset. Image Augmentation is the process of taking images that are already in a training dataset and manipulating them to create many altered versions of the same image. This both provides more images to train on, but can also help expose our classifier to a wider variety of positions and orientation situations so as to make our classifier more robust.

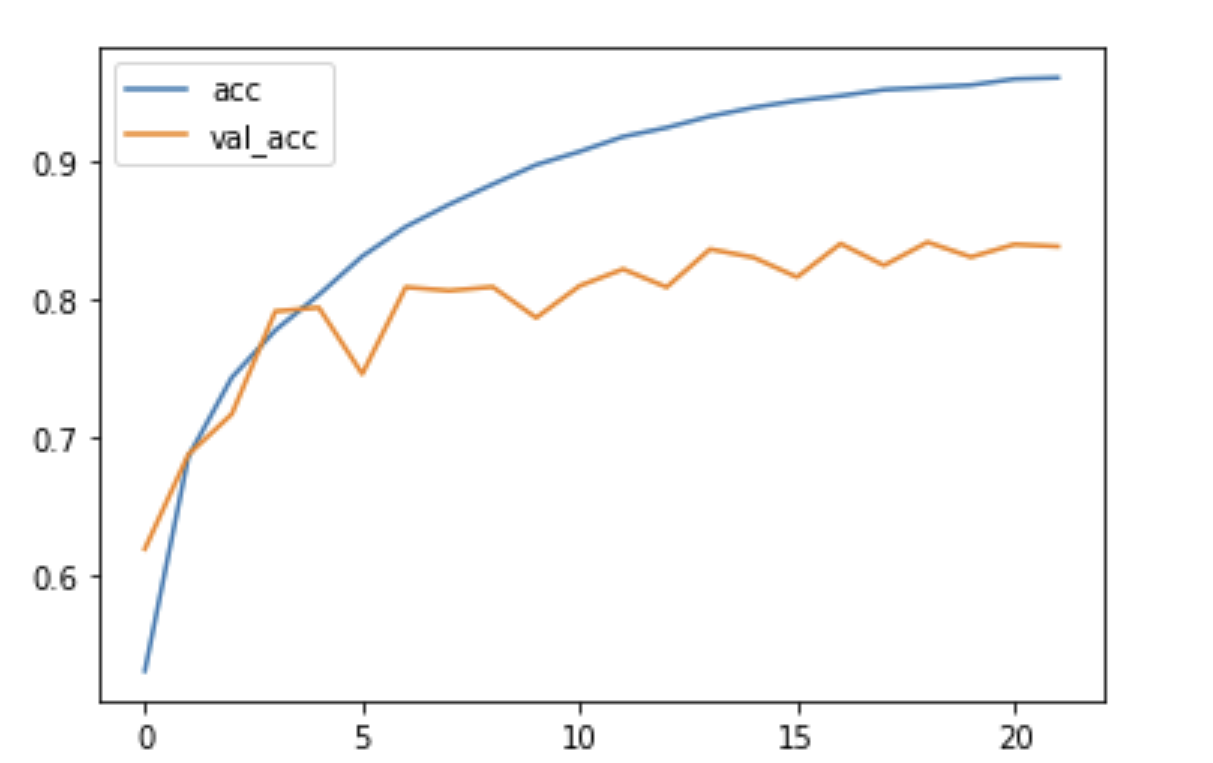
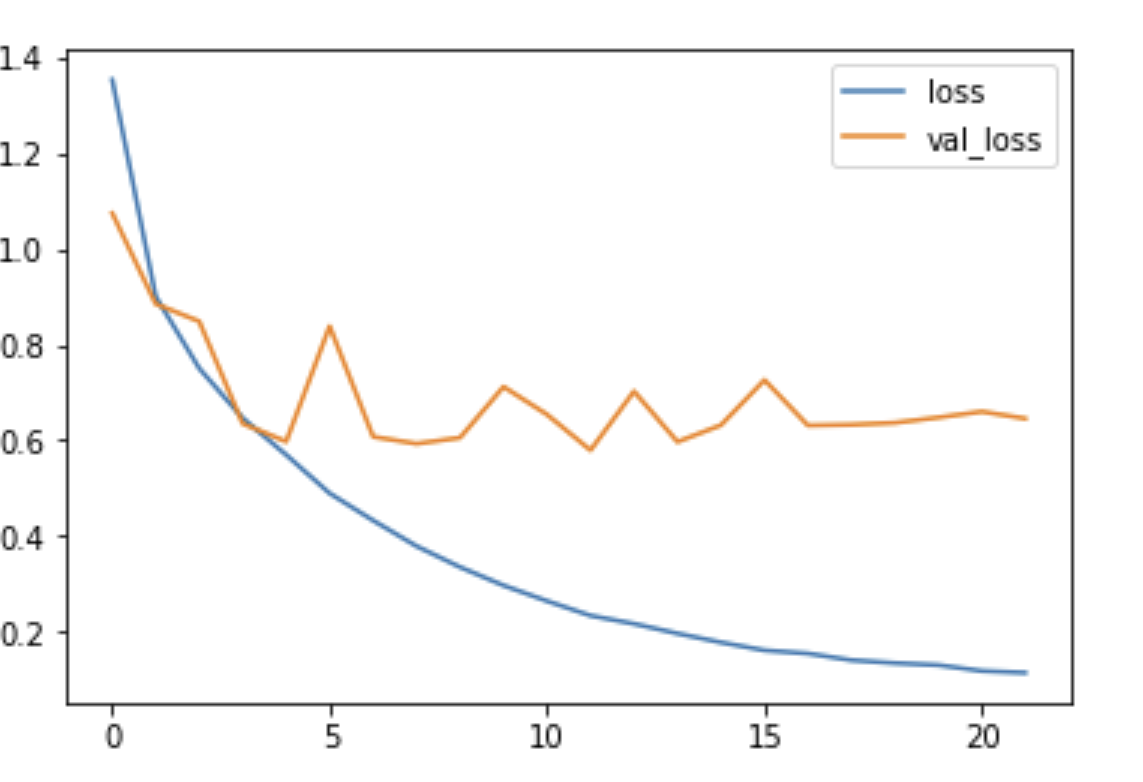
The image is just for demonstration, it doesn’t represent the operations performed on our data.



**Result:**

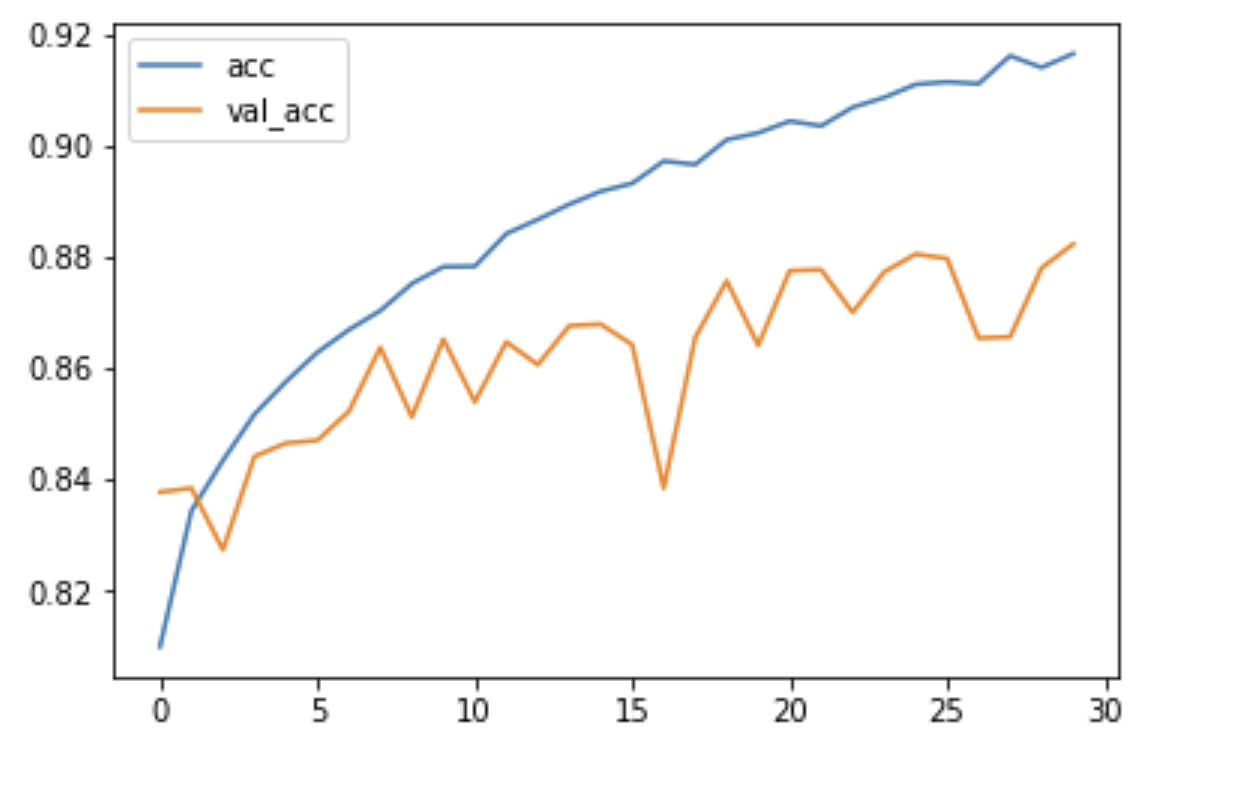
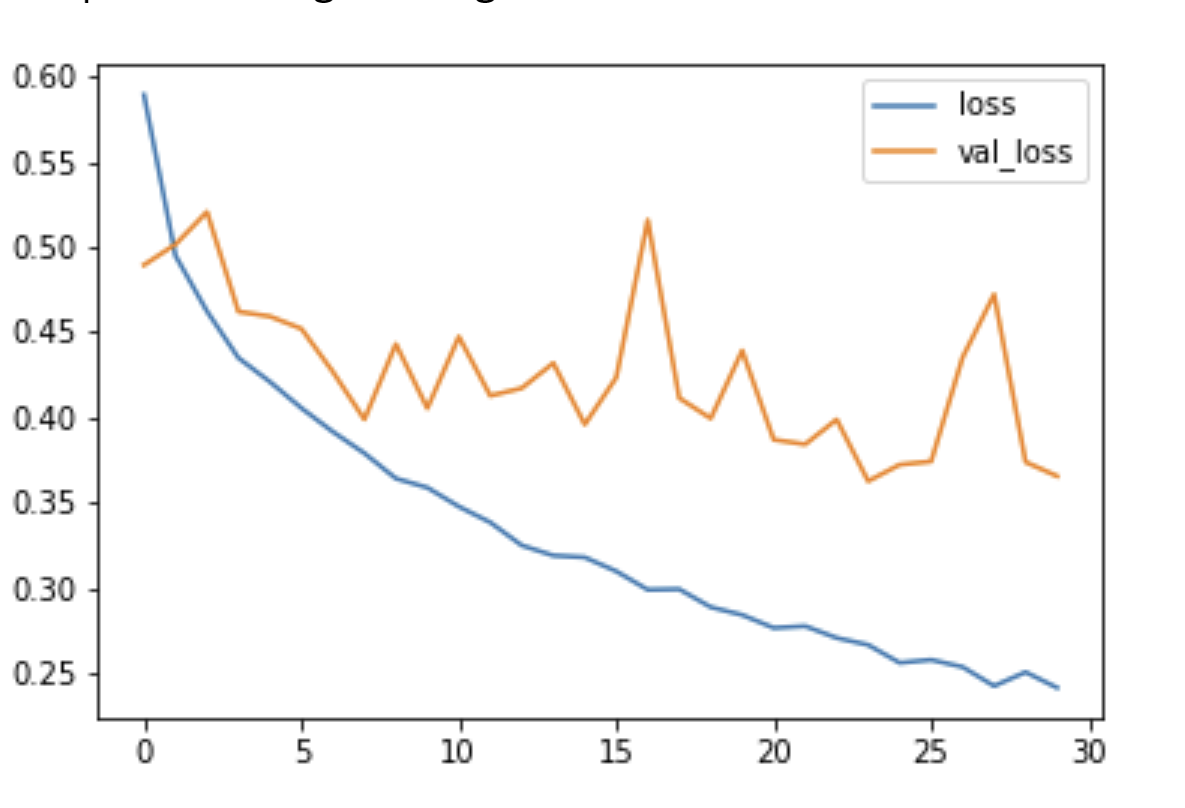
**Without augmentation**

Loss – 0.18 Val\_loss – 0.75 Acc- 97.2% Val\_Acc - 81.78%



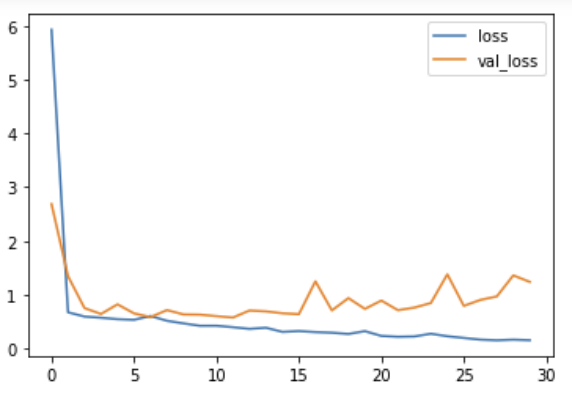
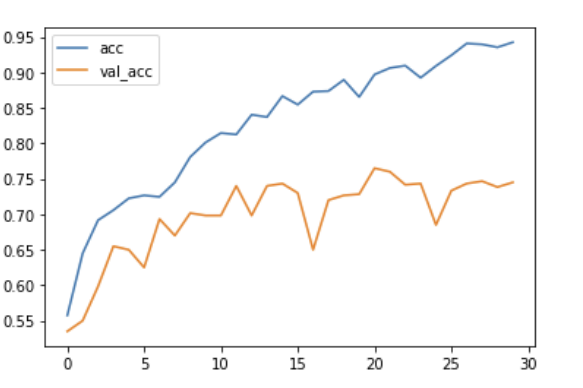
**With Augmentation**

Loss – 0.26 Val\_loss – 0.38 Acc – 91.8% Val\_acc – 88.1%



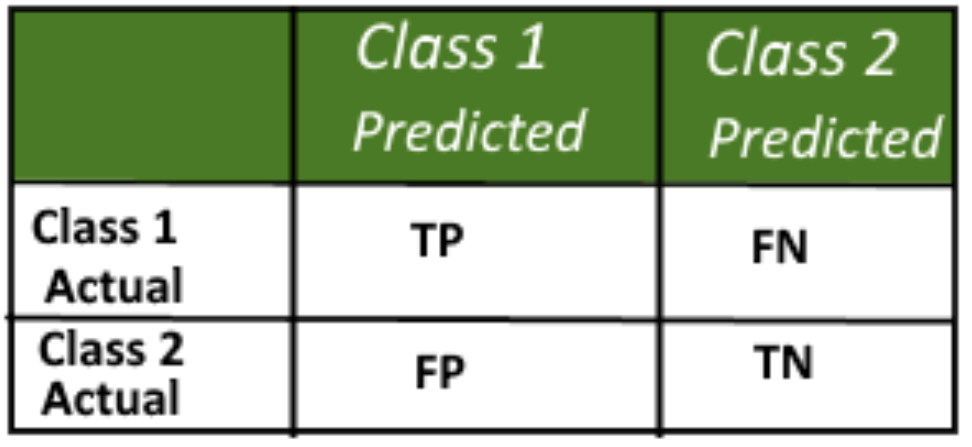
**Using the above model on Cats and dogs Dataset.**

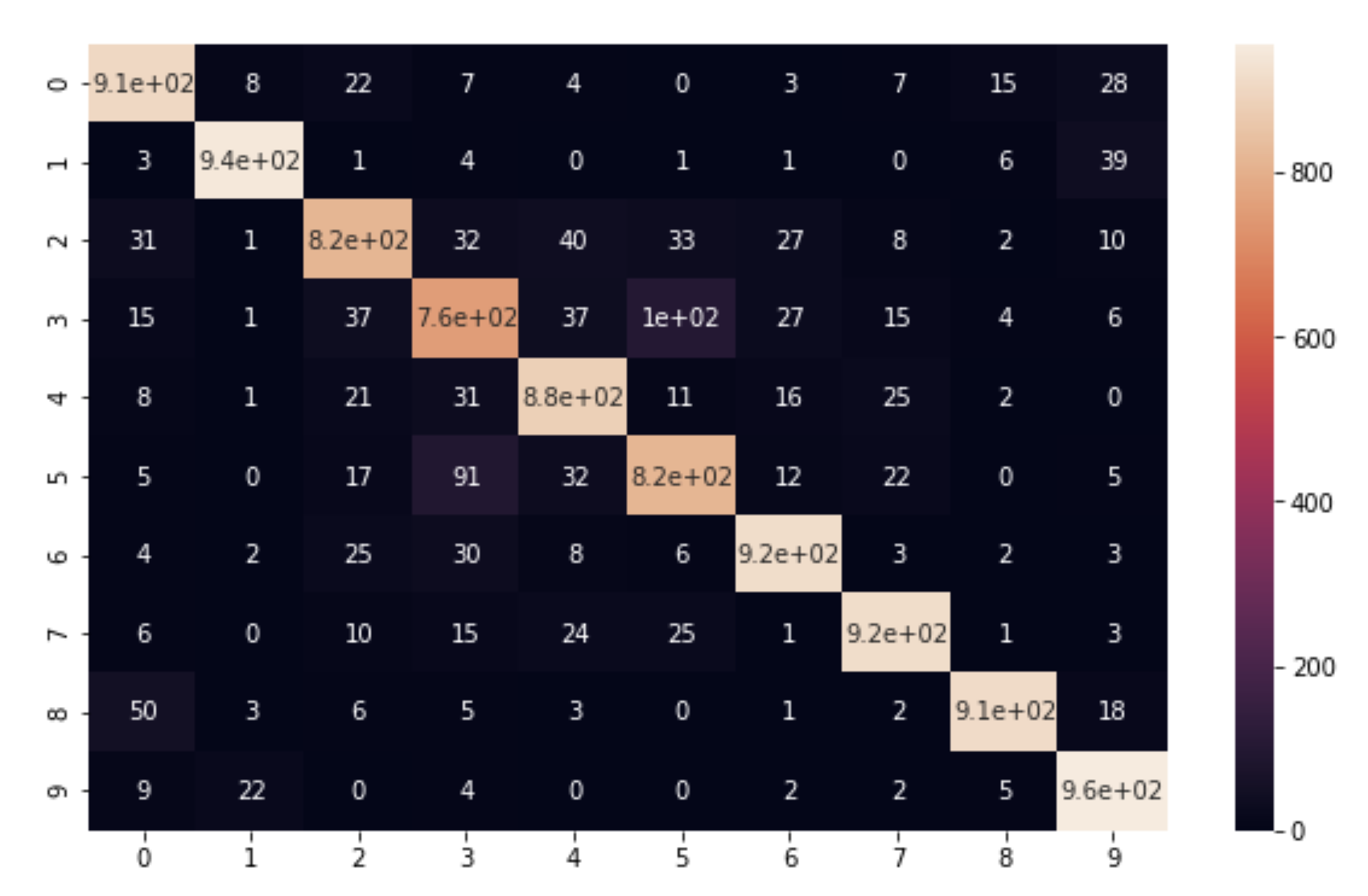
Loss – 0.45 Val\_loss – 1.3 Acc – 92% Val\_acc – 72%

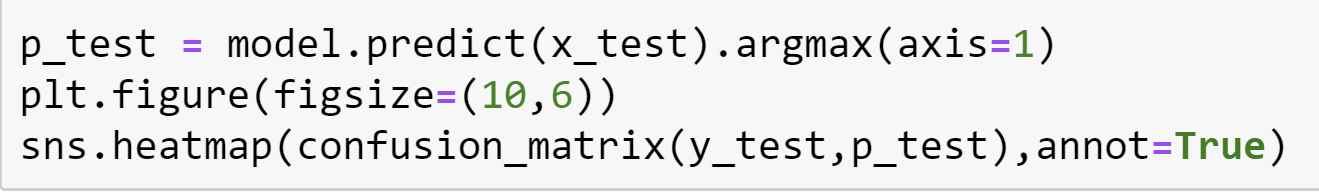
 

**Confusion Matrix:**

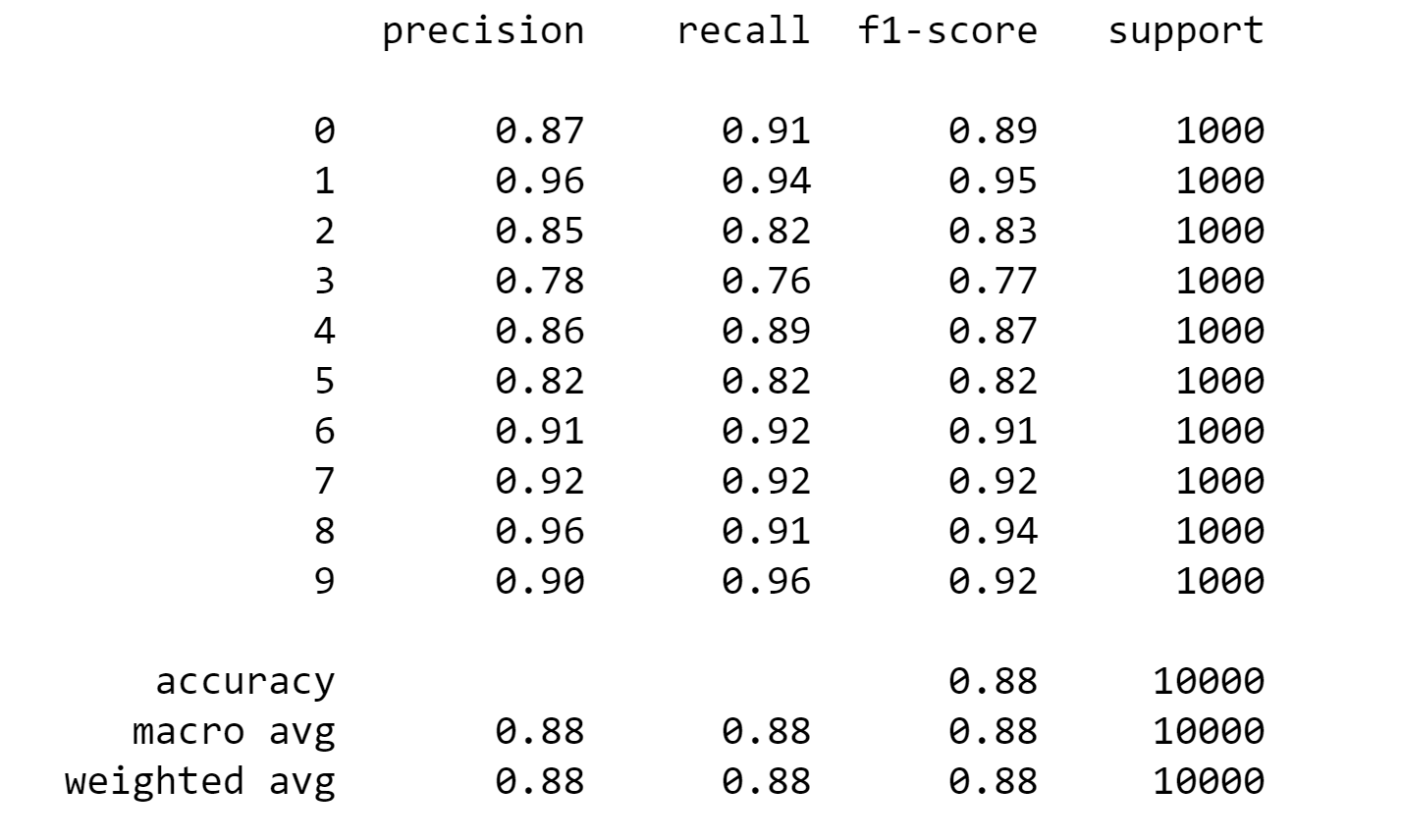
A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.







**Classification Report:**



There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative**: the case was negative and predicted negative
2. **TP / True Positive**: the case was positive and predicted positive
3. **FN / False Negative**: the case was positive but predicted negative
4. **FP / False Positive**: the case was negative but predicted positive

**Precision — *What percent of your predictions were correct?***

Precision is the ability of a classifier not to label an instance positive that is actually negative.

Precision: Accuracy of positive predictions.

Precision = TP/(TP + FP)

**Recall — *What percent of the positive cases did you catch?***

Recall is the ability of a classifier to find all positive instances.

Recall: Fraction of positives that were correctly identified.

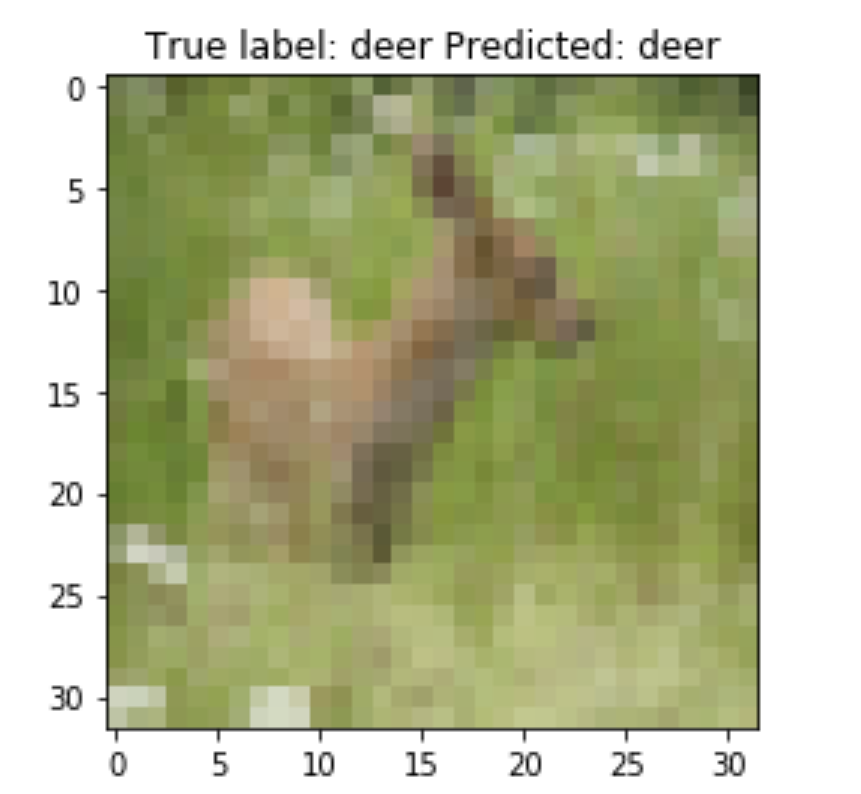
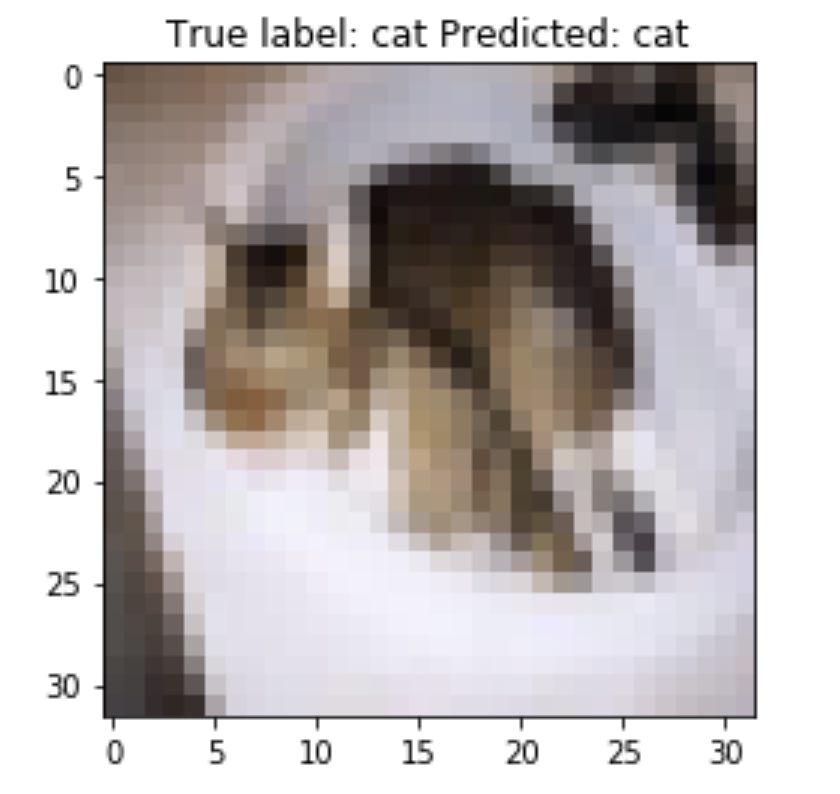
Recall = TP/(TP+FN)

**F1 score — *What percent of positive predictions were correct?***

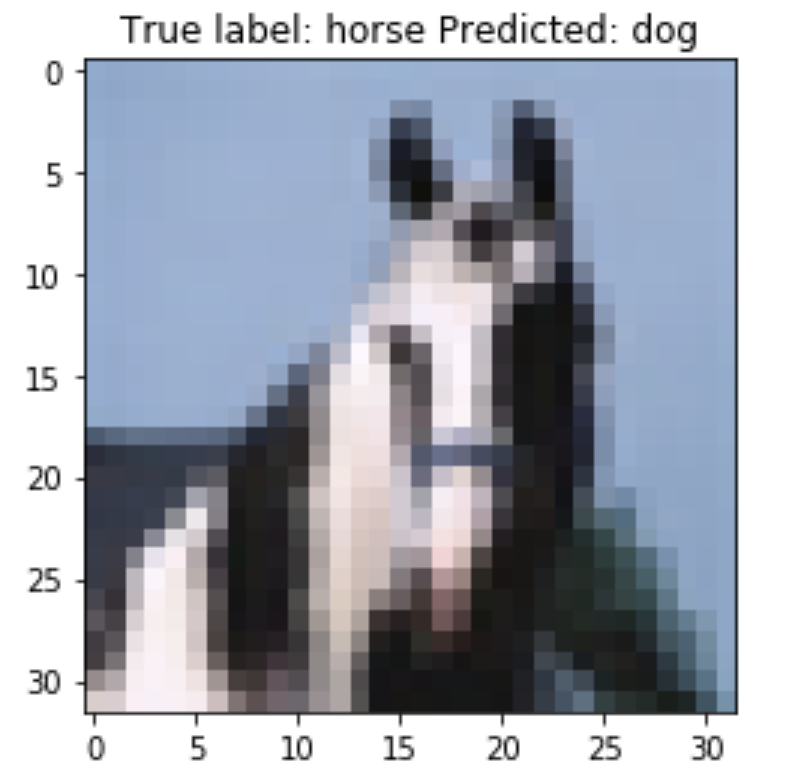
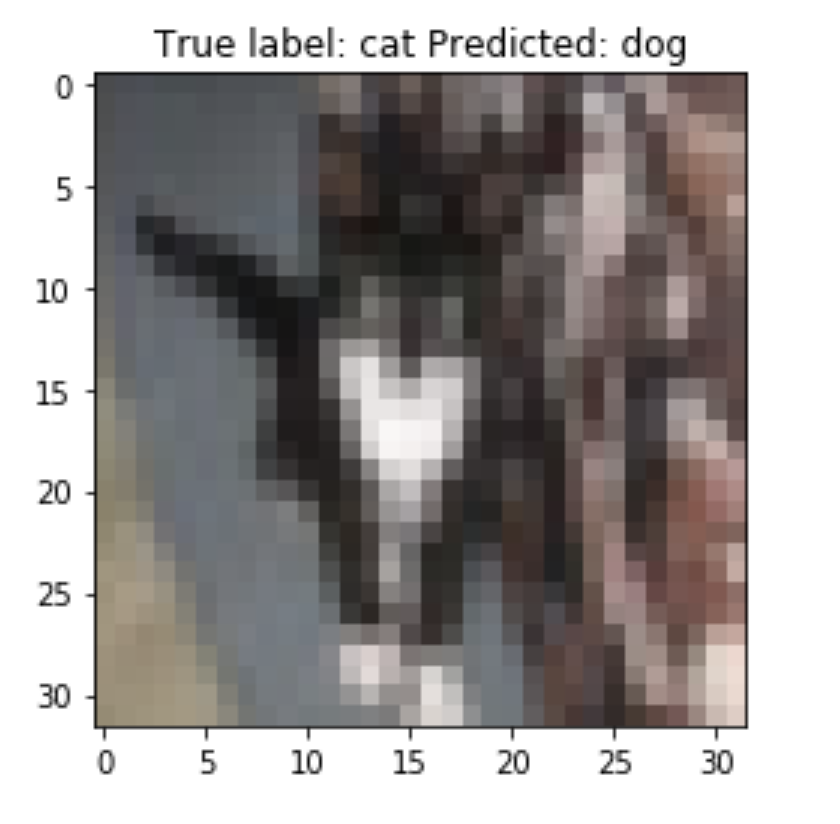
The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation.

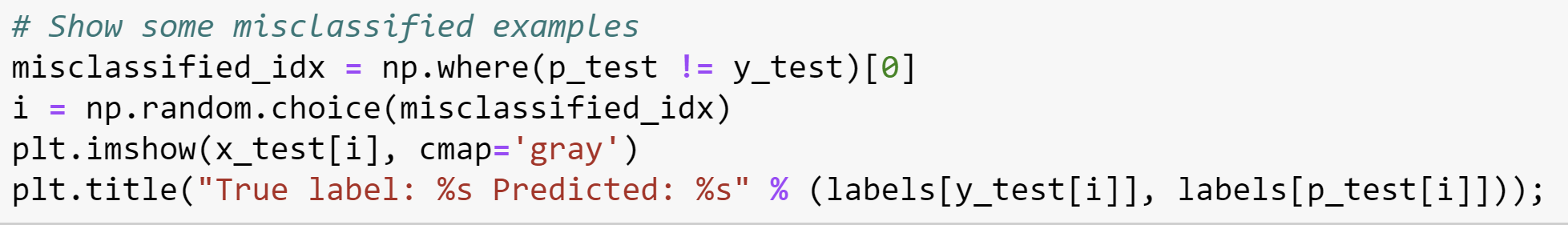
F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

**Correctly Classified**



**Misclassification:** Our model has few instances of misclassification. This is seen more in animal classes. We think the reason for the misclassification is high similarity among the classes because the image size is less and have similar background





**Conclusion**

We conclude that overfitting is a major problem in deep learning. We have overcome this problem by following various techniques such as dropouts, image augmentation and callbacks. Each of these techniques are context dependent so there is no universal way of doing things to prevent overfitting. These techniques helped to reach very low validation loss and a higher validation accuracy. The usage of our network architecture on the cats and dogs dataset did not give satisfying results and in fact caused overfitting. This proved the fact that there is no universal approach to the problem and each task should be looked at differently.

**Future Work**

The major challenge of running such models is the requirement of hardware resources. We have run this model on a powerful GPU but it still took a considerable amount of time. Moreover, not many notebooks house a decent GPU. Hence, we plan to use certain techniques which allow efficient use of the CPU alone. Pipeline parallelism extends on simple task parallelism, breaking the task into a sequence of processing stages. Pipelining is a common technique found in CPUs to increase throughput processing. We think this will help to achieve the performance, if not comparable to a GPU, at least better than the current.