FINAL REPORT

- ADVANCEMENTS IN CNN ARCHITECTURES -

By: Tanvi Gandhi

➢ PROPOSAL

• MOTIVATION

In recent years, we have witnessed the explosion of Deep Neural Networks, especially Convolutional Neural Networks (CNNs) due to their ability in recognizing visual patterns directly from pixel images with minimal preprocessing. CNNs have been a part of the machine learning family since 1989 and were considered breakthrough due to their exemplary performance. This has inspired several ideas to bring advancements in CNNs have been explored, such as the use of different activation and loss functions, parameter optimization, regularization, and architectural innovations. Thus, we want to further explore the current state of the art practices in Deep Learning, relating to image recognition. These architectures include LeNet, Alexnet, ResNet, VGG-16 and many more. It would be interesting to tweak and play with the architectural components to try and come up with an improved model for the handwritten digits.

• WHO MIGHT CARE?

The domain of deep learning has still a lot to explore and it would be wise to understand the intrinsic details of the different dimensions of a Deep Neural Network so that we could come up with an improved version. Even the mere understanding of the basic principles would be fruitful and pave a way to dive into it. Further, it would be very beneficial for the organizations who are working in the image recognition field to have the comprehended architectures. The goal of this study is not only to have a deep understanding of the working of the NNs but also demonstrate myself and my knowledge and skills in this vast domain.

• DATA

The data comprises the MNIST dataset and Dog-Cat classification dataset which is kind of a benchmark dataset for trivial image processing systems. Due to the limited GPU availability, we would be using the mentioned datasets although better datasets are available like ImageNet, CIFAR10 etc. The MNIST dataset contains 60,000 training images and 10,000 testing images. For the dog-cat classification, the training set comprises 25,000 images and test set contain 12,500 images.

- APPROACH

As an initial step, it seems apt to decide that we will implement the LeNet and AlexNet architecture. We will tweak and play with the parameters and compare their accuracy on the MNIST and dog-cat dataset.

- DELIVERABLES

  - **Jupyter Notebook:** A python notebook which will contain all the code involved in the process.
  - **Final Report:** A document to highlight the entire process followed and key takeaways.
  - **Presentation and/or Paper:** A presentation for the audience and/or a research paper to highlight the significant insights.

➢ DATASET

- MNIST-10

The MNIST (Modified National Institute of Standards and Technology) is a large database of handwritten digits and is commonly used for training various image processing systems. The images are black and white and fit into a 28 * 28 pixel bounding box. It consists of 60,000 training images and 10,000 testing images.

- DOG-CAT

The Dog-Cat dataset was provided by the Microsoft Research team and is taken from Kaggle website (link: https://www.kaggle.com/c/dogs-vs-cats/data). The training data comprises 25,000 colorful images of dogs and cats and the test data consists of 12,500 images to test upon.

➢ IMPLEMENTATION

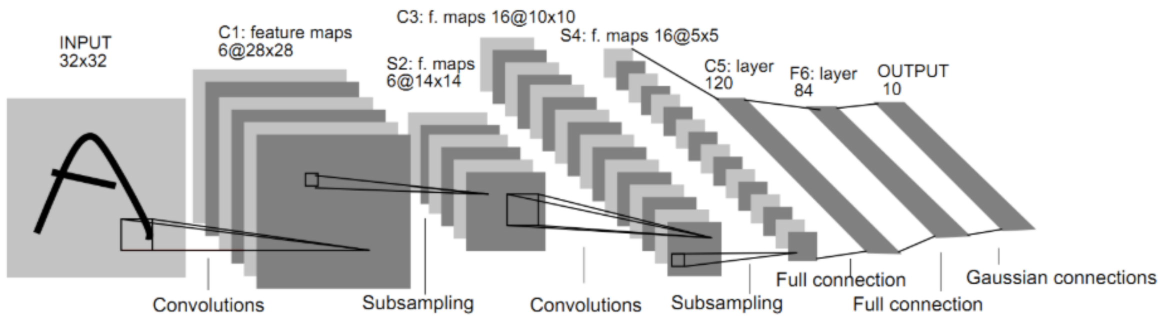
• HARDWARE

The implementation of LeNet and AlexNet was done on my local system. The details are:

- Processor : Intel(R) Core(TM) i3-6006U CPU @2.00GHz 1.99 GHz
- Installed Memory (RAM) : 4.00 GB
- System type : 64-bit Windows 10 Operating System, x64-based processor

• LENET-5 ON MNIST DATASET

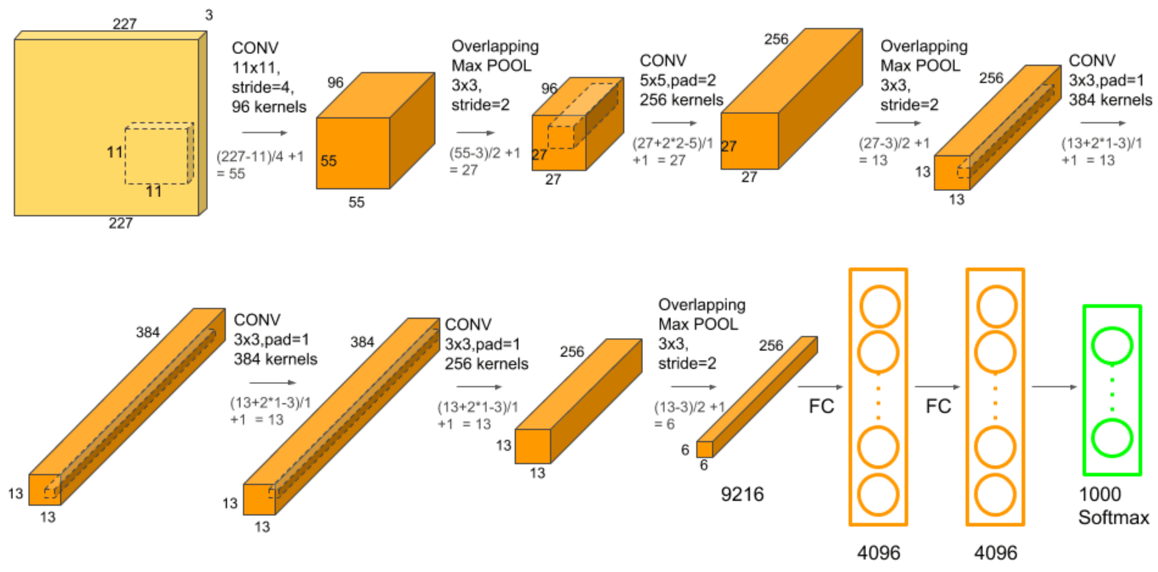
The LeNet architecture comprises total 7 layers, excluding the input layer, with 3 convolutional layers, 2 pooling layers and 2 fully connected layers. I have used 'Relu' as an activation function for the convolutional layers and 'softmax' at the output layer. Below is the architecture used:



The architecture was trained for 60000 samples and tested on 10000 samples. The total time was almost half hour since the network was really simple for 20 epochs with 128 batch size. The loss function used was categorical_crossentropy along with Adam optimizer. The Loss was 0.035 and Accuracy was 0.989 for the testing data.

• ALEXNET ON MNIST DATASET

Alexnet is quite a deeper network and came out in 2012 as the winner of the ImageNet competition. It consists of eight layers: five convolutional layers and three fully-connected layers. The features which were not standard at the time when the architecture came out were usage of Relu function which reduces the training time a lot, the overlapping pooling instead traditional "pool" outputs of neighboring groups of neurons with no overlapping and Dropout i.e. turning off neurons with a predetermined probability (e.g. 50%) which means that every iteration uses a different sample of the model's parameters, which forces each neuron to have more robust features that can be used with other random neurons.



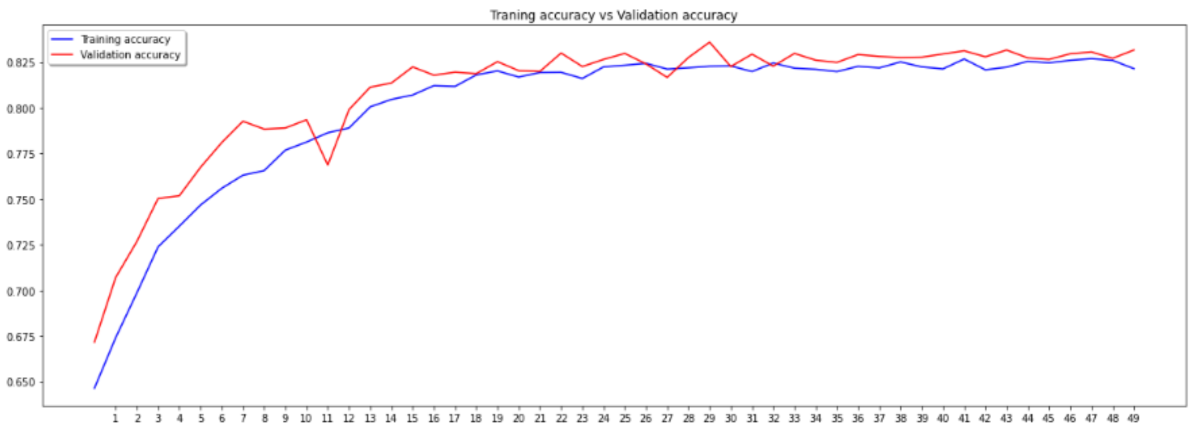
The network was trained on 60,000 samples and tested on 10,000 samples. The total time was almost 3 days for 20 epochs with batch size of 128. The loss function used was categorical_crossentropy along with Adam optimizer. The loss was 0.054 and accuracy was 0.9893 for the testing data.

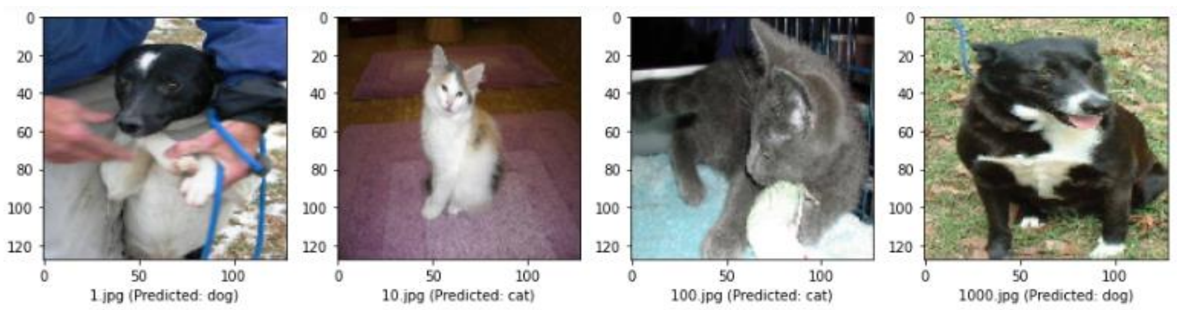
• LENET ON DOG-CAT DATASET

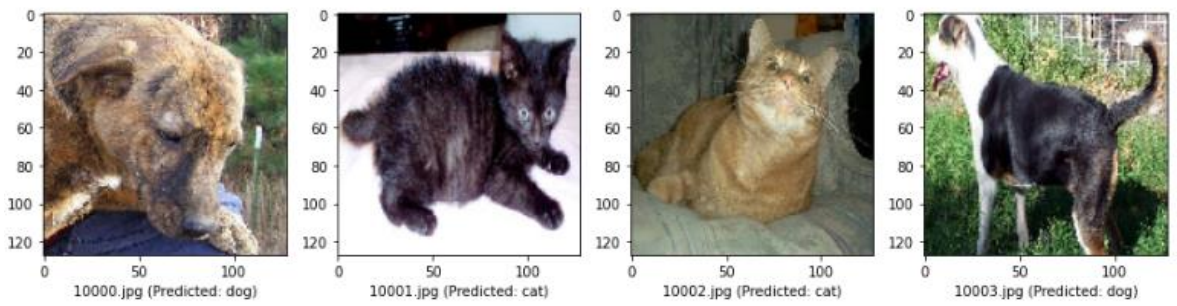
The LeNet architecture we opted for Dog-Cat classification is similar to the MNIST implementation. The model summary is shown below. There were 25,000 training samples which were split into 4:1 training and validation dataset. The testing data was separate, consisting 10,000 samples. We had to first extract the labels from the image names. The Relu activation function was used to provide optimal results. The input image size was fixed to 128 * 128 with 3 channels. We have used ImageDataGenerator to augment the training samples, by zooming, flipping, rotating etc., so that model is able to learn better. The ReduceLROnPlateau was used on 'val_loss' as a callback with a patience level of 2 so that learning rate can be reduced if no improvement is seen for 2 epochs. The loss function used was categorical_crossentropy along with Adam optimizer. The network was trained for 50 epochs with batch size of 128. The final validation accuracy was 83.17%.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_5 (Conv2D)            (None, 124, 124, 6)       456       
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 62, 62, 6)         0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 58, 58, 16)        2416      
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 29, 29, 16)        0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 13456)             0         
_________________________________________________________________
dense_7 (Dense)              (None, 120)               1614840   
_________________________________________________________________
dense_8 (Dense)              (None, 84)                10164     
_________________________________________________________________
dense_9 (Dense)              (None, 2)                 170       
=================================================================
Total params: 1,628,046
Trainable params: 1,628,046
Non-trainable params: 0
```

Below is the Training vs. Accuracy graph for 50 epochs. Since the testing labels were not given, I've also shown some of the images with the predicted labels so that we can get an idea of the accuracy level.

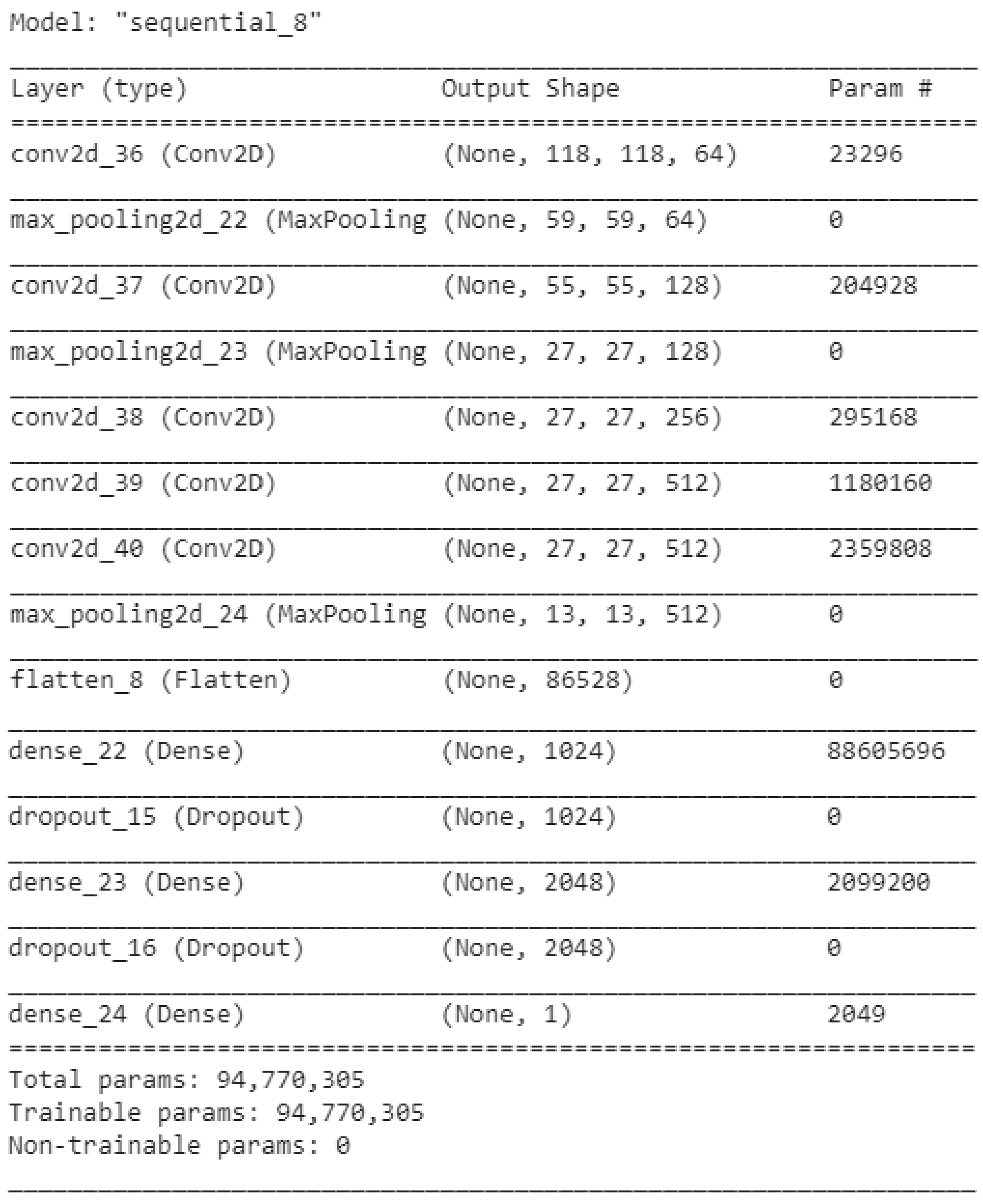





10000.jpg (Predicted: dog)
10001.jpg (Predicted: cat)
10002.jpg (Predicted: cat)
10003.jpg (Predicted: dog)

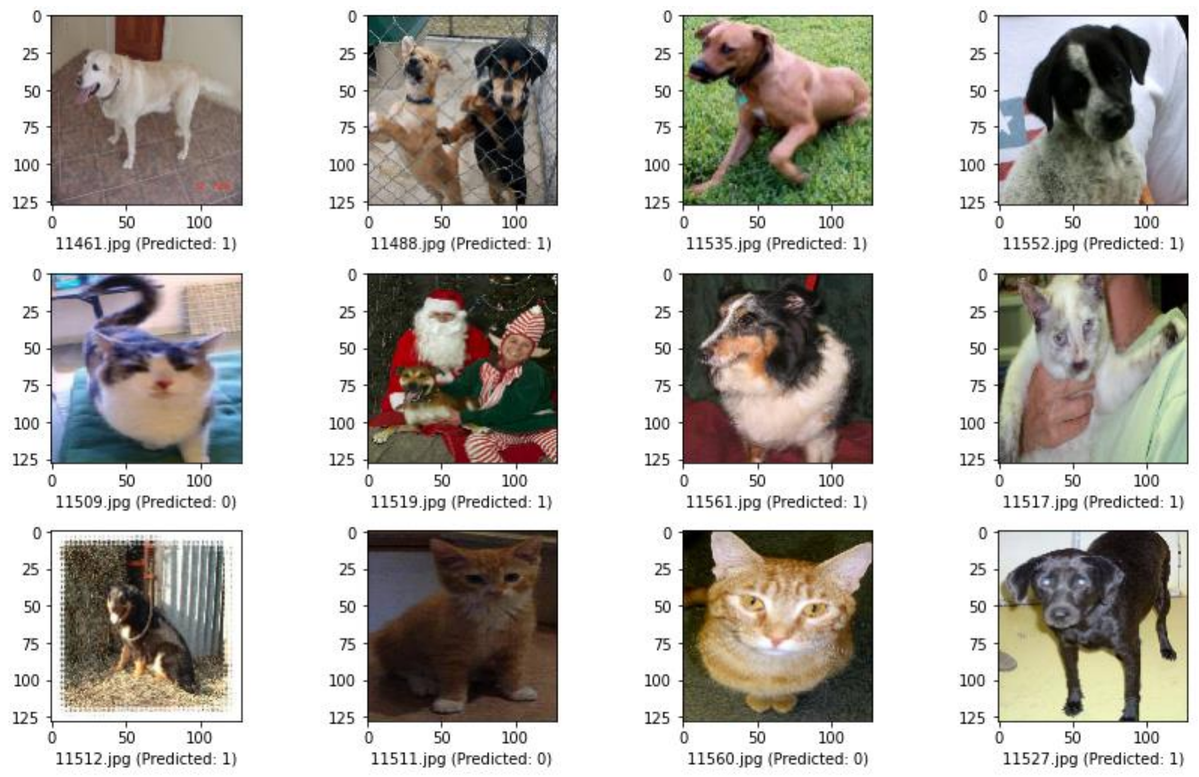
- ALEXNET ON DOG-CAT DATASET

When AlexNet came out, it introduced many features which were not standard at that time including usage of Relu function which reduces the training time a lot, the overlapping pooling instead traditional “pool” outputs of neighboring groups of neurons with no overlapping and Dropout i.e. turning off neurons with a predetermined probability (e.g. 50%) which means that every iteration uses a different and random layer characteristics. In our, Dog-Cat classification, we tried to tweak the architecture by changing the number of kernels in each of the 5 convolutional layers. The model summary is shown below. The training, validation and testing set was same as the LeNet implementation. ImageDataGenerator was used to train on augmented images. The ReduceLROnPlateau was used on ‘val_accuracy’ as a callback with a patience level of 2 so that learning rate can be reduced if the accuracy does not improve for 2 epochs. The network was trained for 50 epochs with batch size of 12 on Google Colab to utilize the GPU. The loss function used is binary_crossentropy along with the SGD optimizer. The final training accuracy was 91% and final validation accuracy was 90.66%.



```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_36 (Conv2D)           (None, 118, 118, 64)      23296     
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 59, 59, 64)        0         
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 55, 55, 128)       204928    
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 27, 27, 128)       0         
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 27, 27, 256)       295168    
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 27, 27, 512)       1180160   
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 27, 27, 512)       2359808   
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 13, 13, 512)       0         
_________________________________________________________________
flatten_8 (Flatten)          (None, 86528)             0         
_________________________________________________________________
dense_22 (Dense)             (None, 1024)              88605696  
_________________________________________________________________
dropout_15 (Dropout)         (None, 1024)              0         
_________________________________________________________________
dense_23 (Dense)             (None, 2048)              2099200   
_________________________________________________________________
dropout_16 (Dropout)         (None, 2048)              0         
_________________________________________________________________
dense_24 (Dense)             (None, 1)                 2049      
=================================================================
Total params: 94,770,305
Trainable params: 94,770,305
Non-trainable params: 0
_________________________________________________________________
```

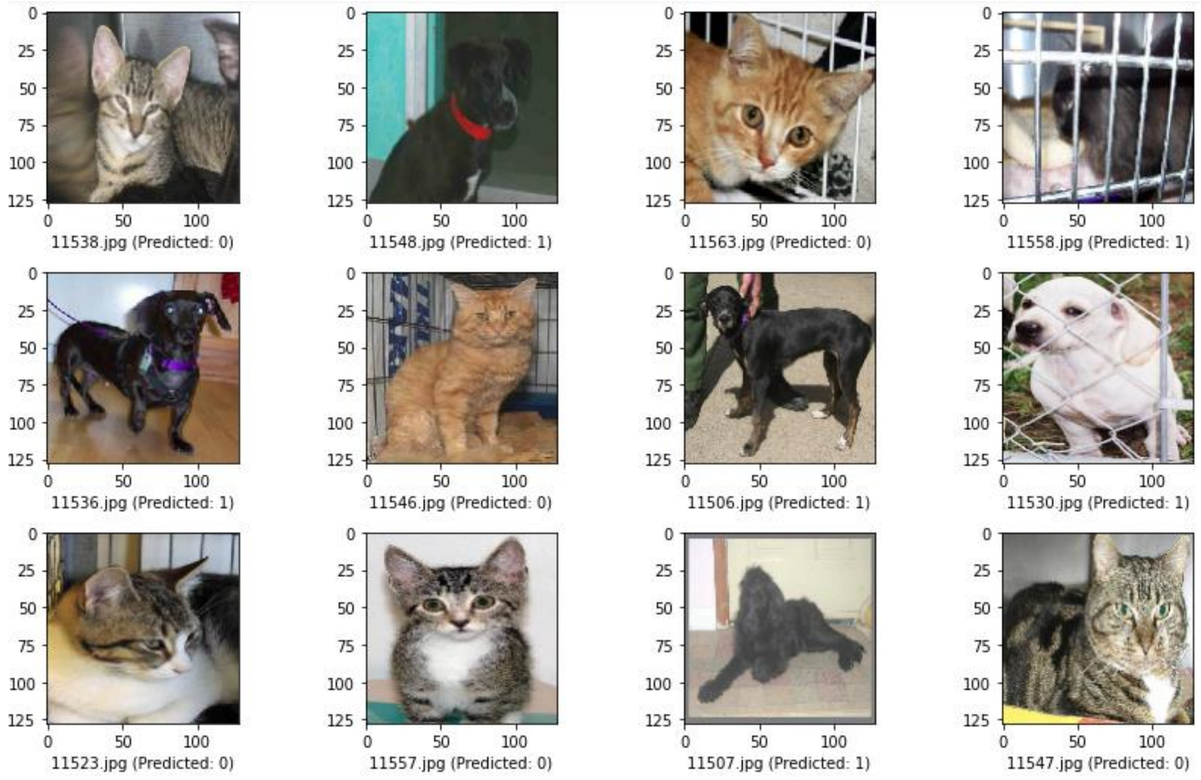
Below is the Training vs. Accuracy graph for 50 epochs. Since the testing labels were not given, I’ve also shown some of the images with the predicted labels so that we can get an idea of the accuracy level. The predicted label 0 denotes a cat and 1 denotes a dog.



11461.jpg (Predicted: 1) 11488.jpg (Predicted: 1) 11535.jpg (Predicted: 1) 11552.jpg (Predicted: 1)

11509.jpg (Predicted: 0) 11519.jpg (Predicted: 1) 11561.jpg (Predicted: 1) 11517.jpg (Predicted: 1)

11512.jpg (Predicted: 1) 11511.jpg (Predicted: 0) 11560.jpg (Predicted: 0) 11527.jpg (Predicted: 1)



11538.jpg (Predicted: 0) 11548.jpg (Predicted: 1) 11563.jpg (Predicted: 0) 11558.jpg (Predicted: 1)

11536.jpg (Predicted: 1) 11546.jpg (Predicted: 0) 11506.jpg (Predicted: 1) 11530.jpg (Predicted: 1)

11523.jpg (Predicted: 0) 11557.jpg (Predicted: 0) 11507.jpg (Predicted: 1) 11547.jpg (Predicted: 0)