**ADTA 5550 Final Project**

**Part-1: A Dataset of Images or Audio files**

* Name of the dataset- **‘Covid-19 Image dataset’**
* I have chosen COVID-19 image dataset which contains chest X-rays helping me to detect COVID-19 using deep learning models.

**Collection Methodology**

* It was collected from publicly released GitHub account by the [University of Montreal](https://www.umontreal.ca/en/) professors. The Pneumonia data has been taken from the [RSNA](https://www.rsna.org/) website.
* The University of Montreal released images are very useful in carrying out effective research activities in the field of deep learning.

**About Data**

* The data contains 137 cleaned images of COVID-19 and 317 in total containing viral pneumonia and normal chest X-rays structured into the test and train directories.
* The data has been retrieved form Kaggle platform. The dataset is being maintained by author named ‘Pranav Raikote’. Below is the link to the data.

[https://www.kaggle.com/pranavraikokte/covid19-image-dataset#](https://www.kaggle.com/pranavraikokte/covid19-image-dataset)

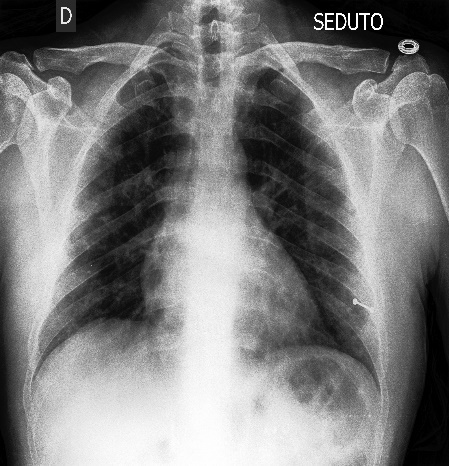
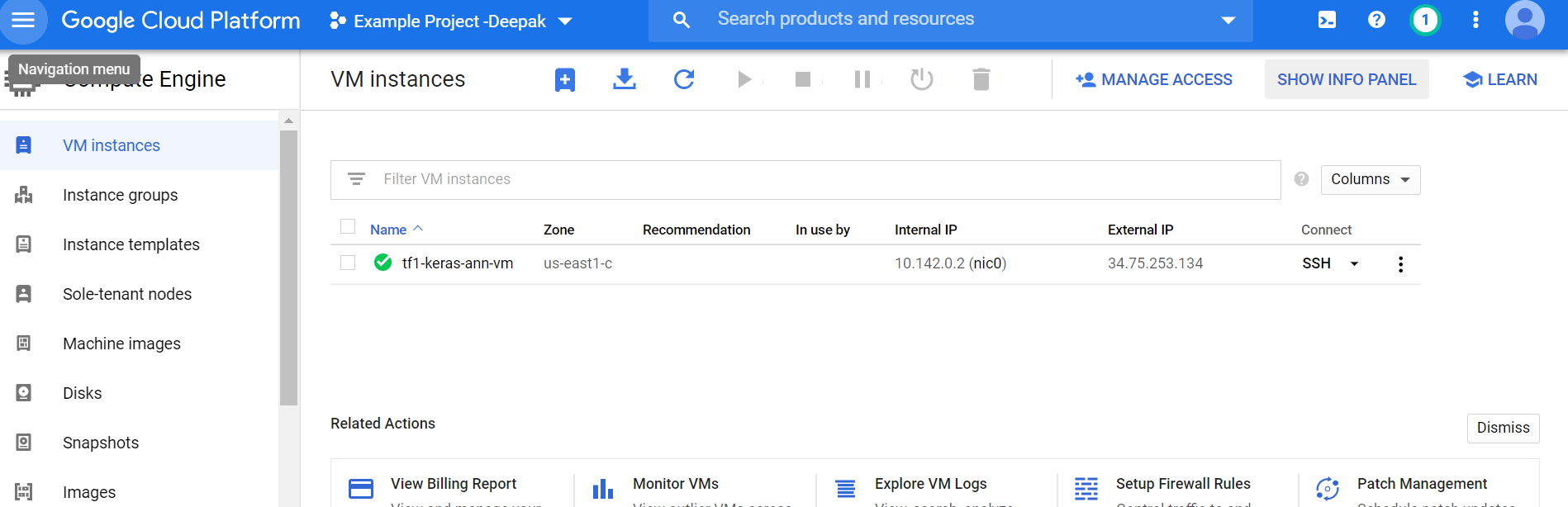


Figure 1 Sample image of the dataset retrieved. Dataset contains similar images of COVID-19 patients, Healthy people, and Pneumonia patients.

* Nearly 238 images are in JPEG (Joint Photographic Expert Group), 68 images are in JPG format and 11 images are in PNG (Portable network graphics) format.
* All the diseases effect the lungs; hence it becomes easy to detect it using an X-ray scanned at chest region. A deep learning model trained with these images helps us in easy detection of the disease.
* However, it is highly recommended to train the deep learning model with large dataset for the accurate results.

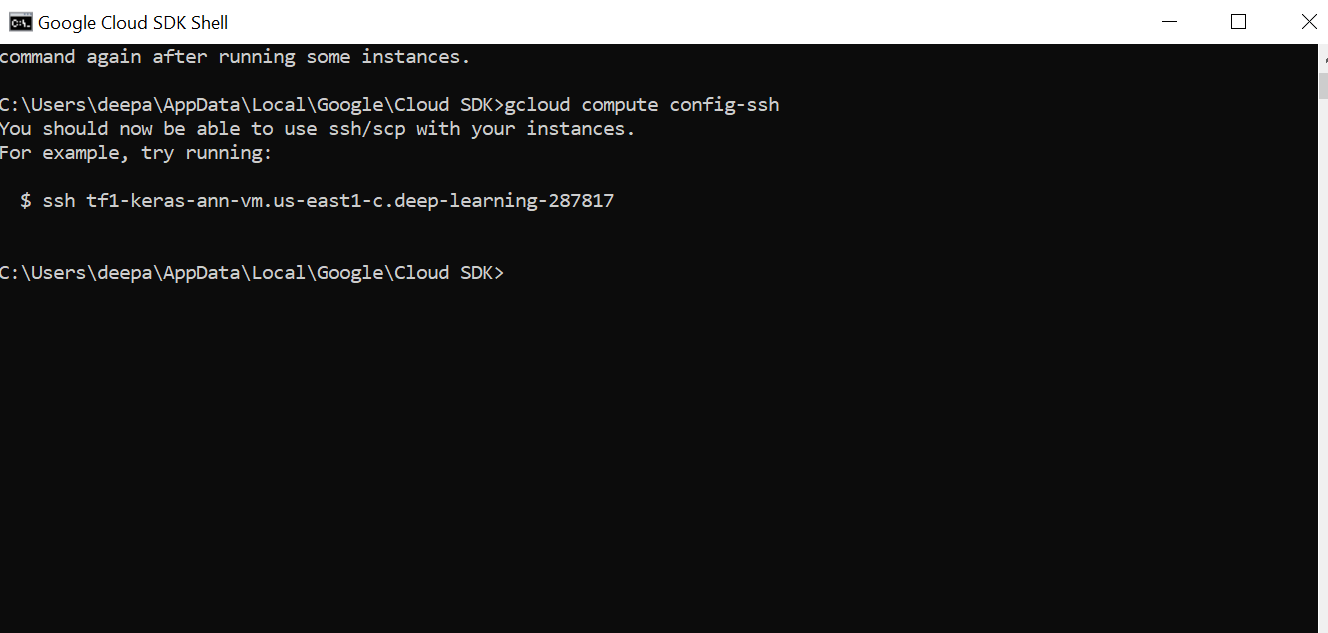
**PART II: Obtain CIFAR-10 Dataset**

**Step-1: Started VM on Google cloud Platform**



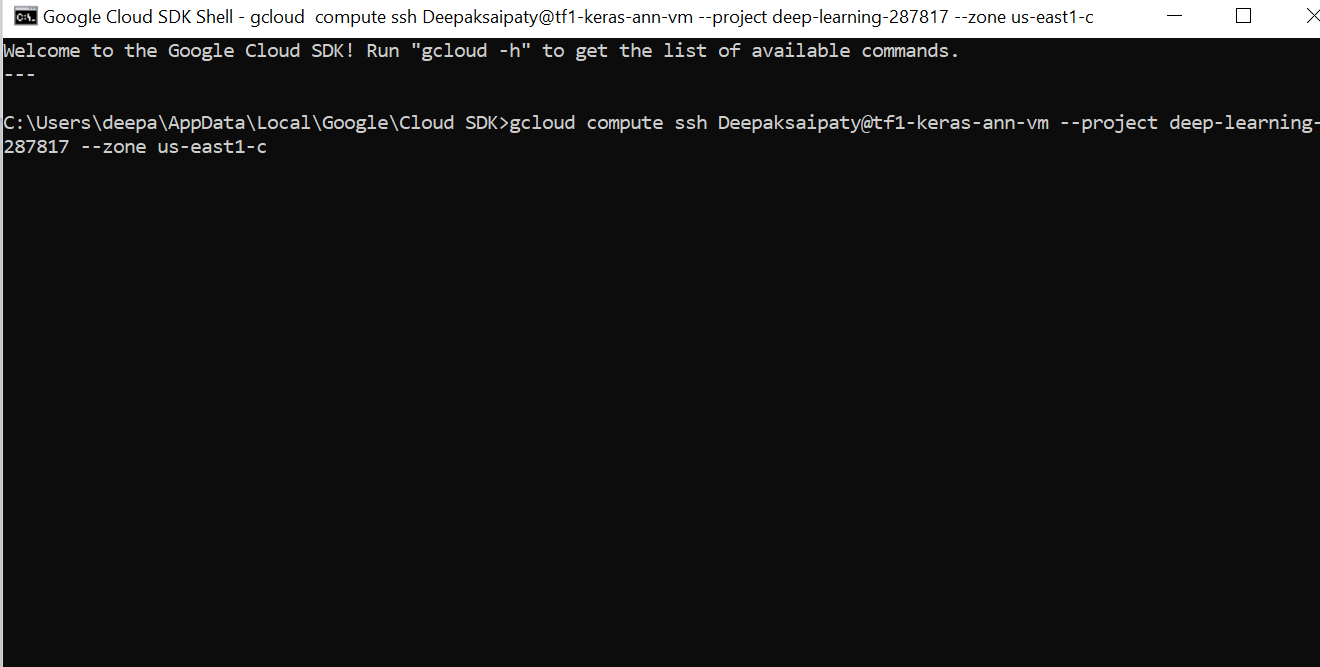
**Step -2: Configured SSH**

* Executed command ‘gcloud compute config-ssh’

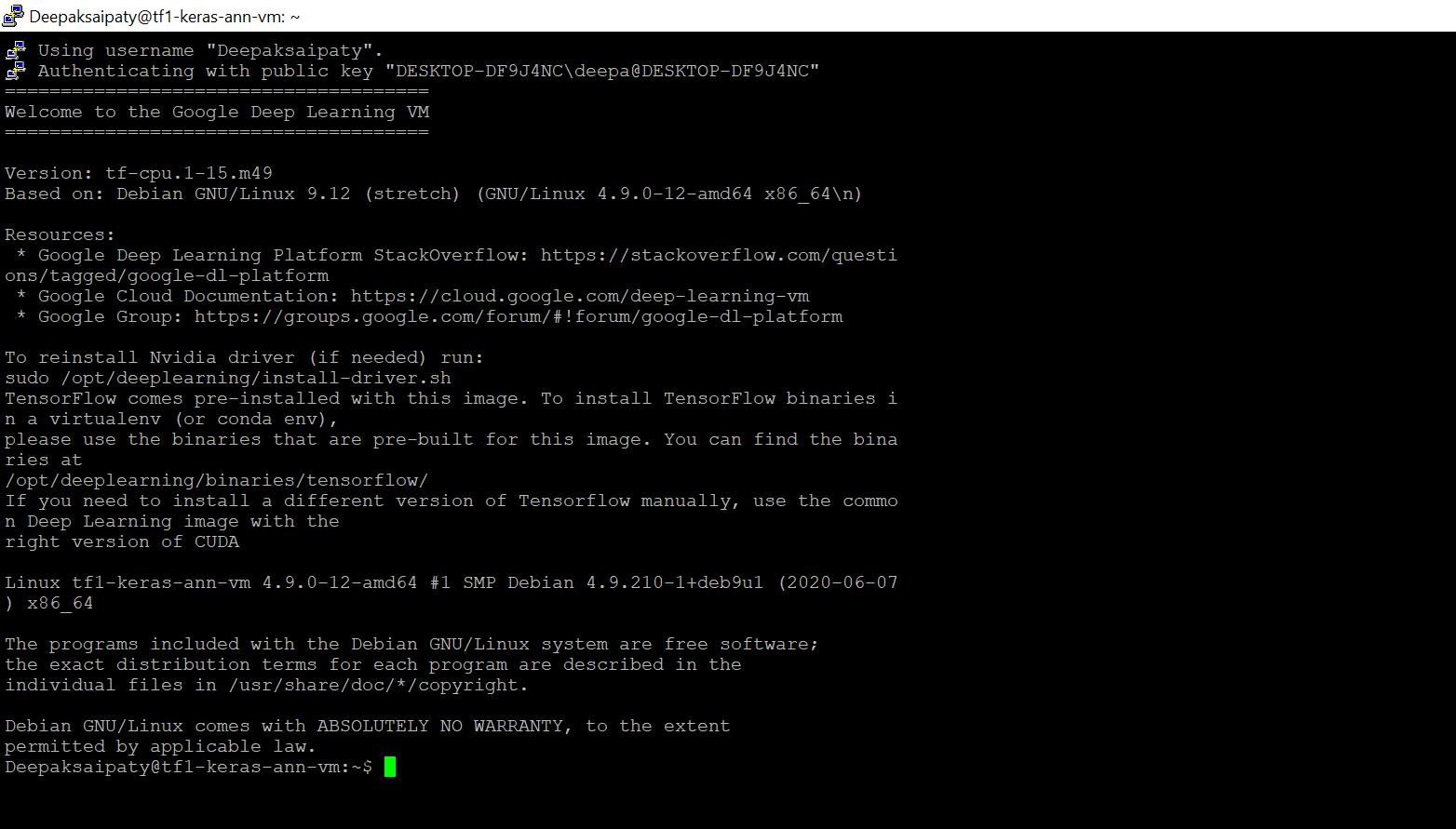


**Step-3: GCLOUD SSH command**

* Executed below command in Google shell,
* gcloud compute ssh Deepaksaipaty@tf1-keras-ann-vm –-project deep-learning-287817 –-zone us-east1-c

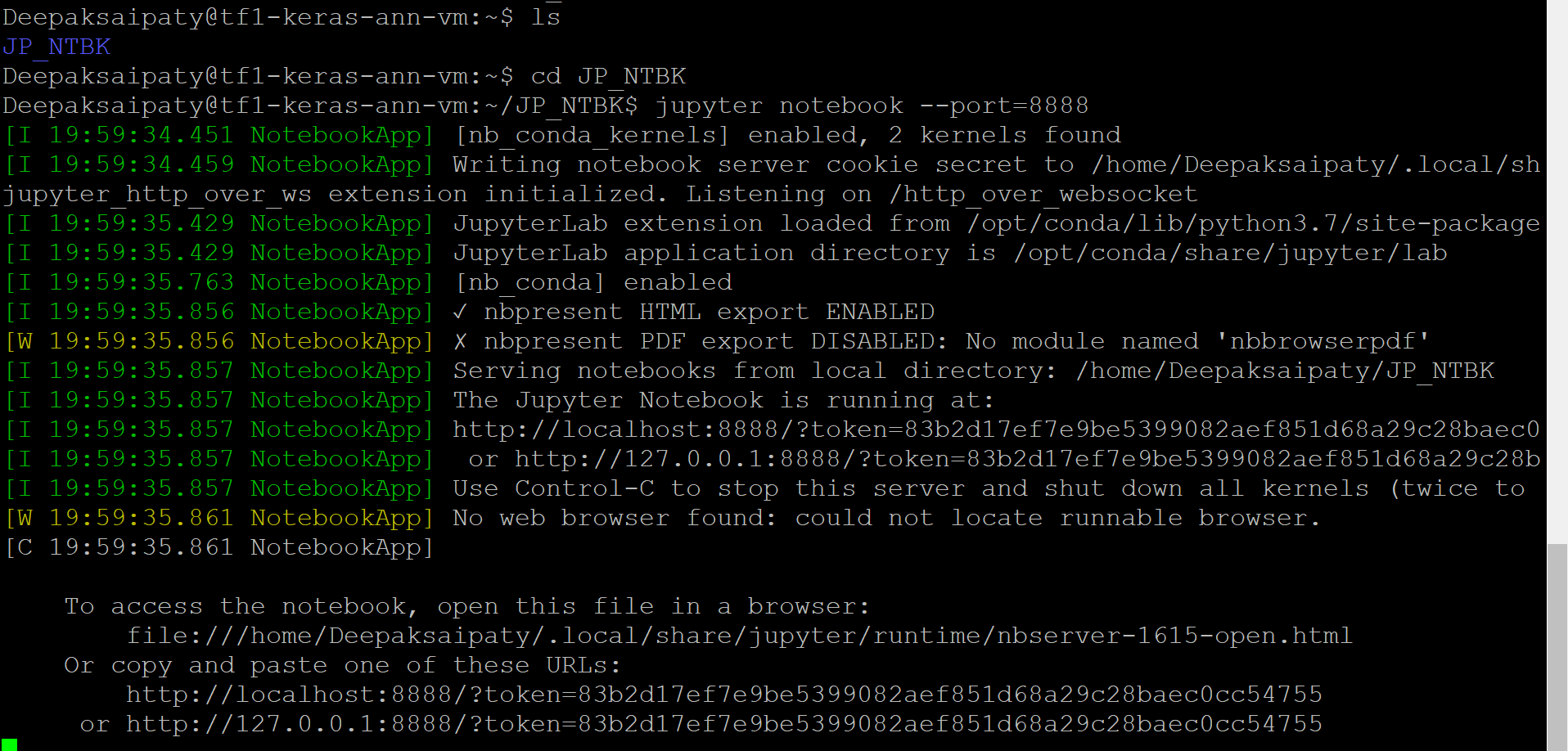


* PUTTY prompt opened showing the successful connection SSH connection to remote VM



**Step-4: Start Jupyter notebook server in GCP remote VM**

* Executed below command,
  + jupyter notebook --port=8888



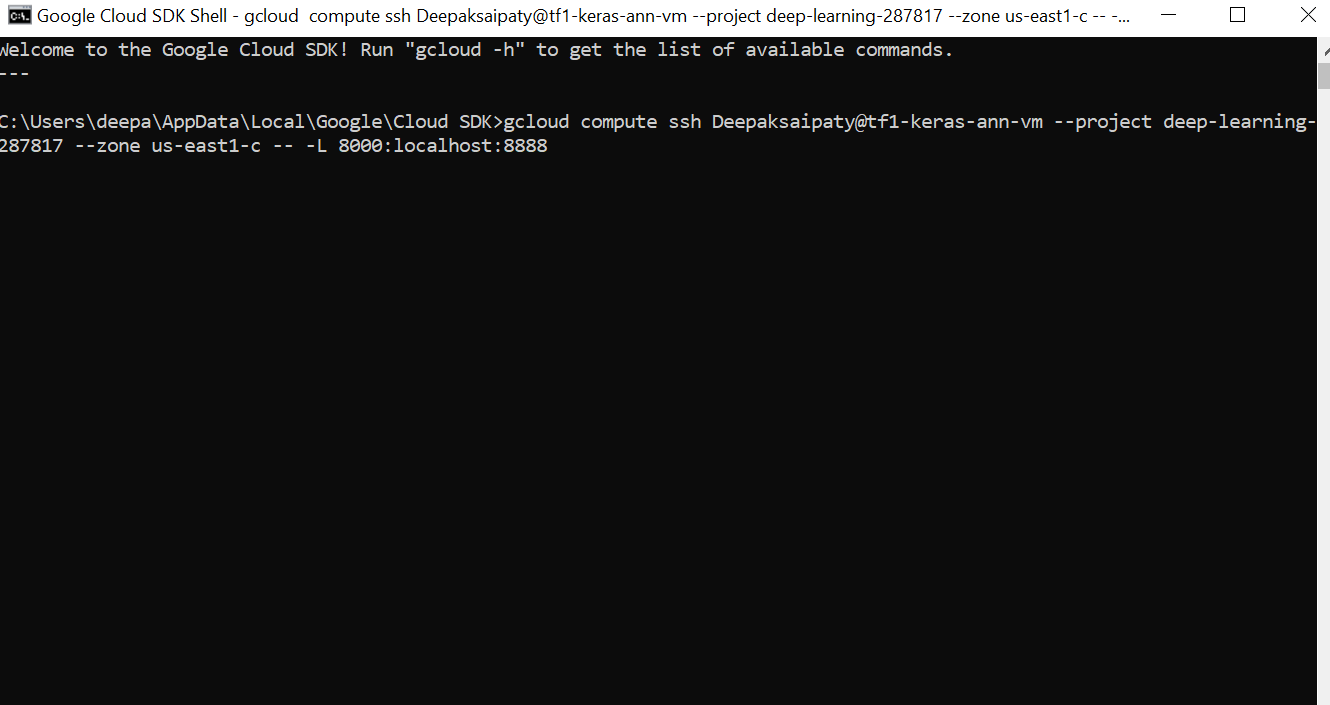
Copy pasted the URL in notebook.



**Step-5: Forward a Local Port to the Port 8888 in GCP Remote VM**

* Started a GCLOUD SDK terminal window (GCLOUD Terminal #2)
* Executed below command in terminal Gcloud terminal 2.

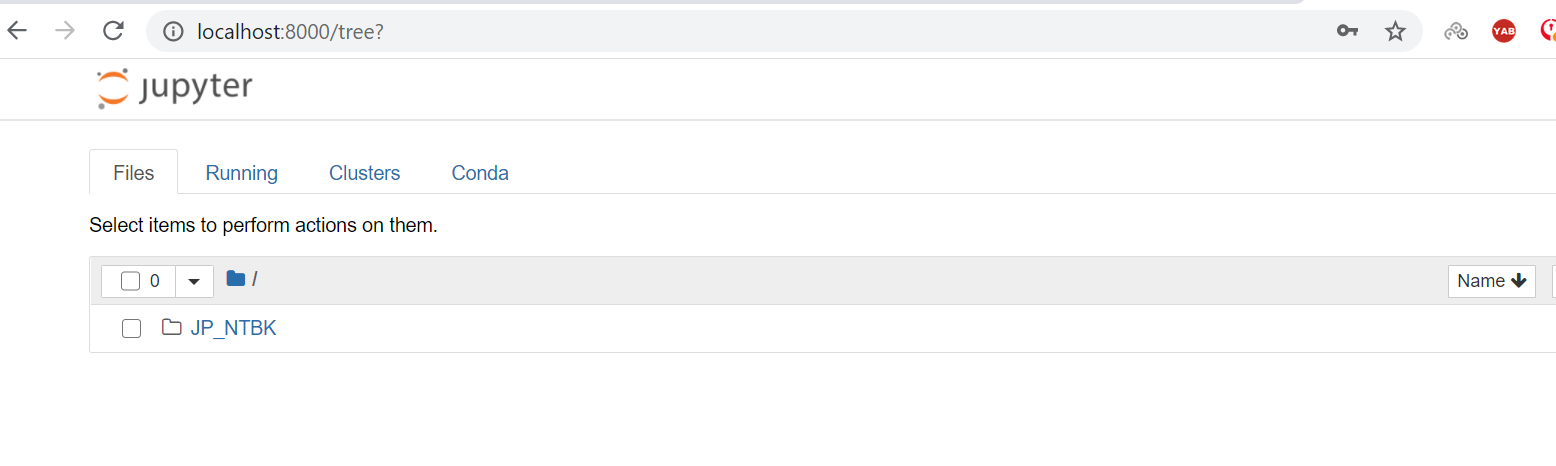
gcloud compute ssh Deepaksaipaty@tf1-keras-ann-vm –-project deep-learning-287817 –-zone us-east1-c -- -L 8000:localhost:8888



* SSH terminal popped up

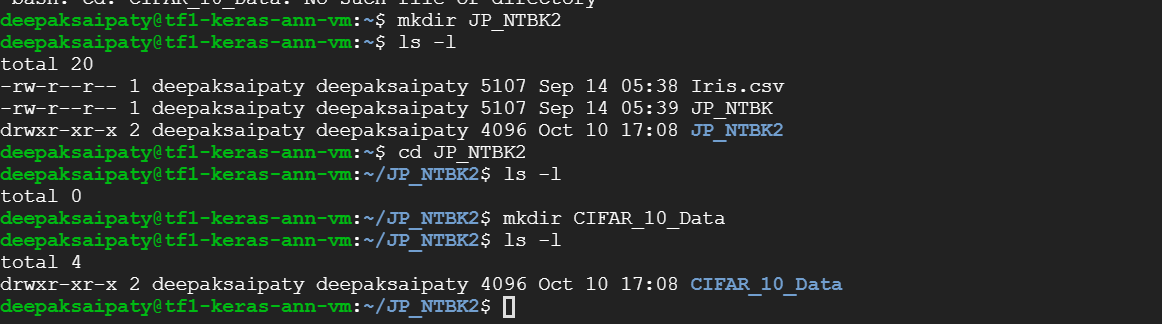
**Step-6: Run Jupyter Notebook in a local browser**

* Entered below link in browser
  + <http://localhost:8000>
  + Logged in with the Notepad copied token in Step-4.

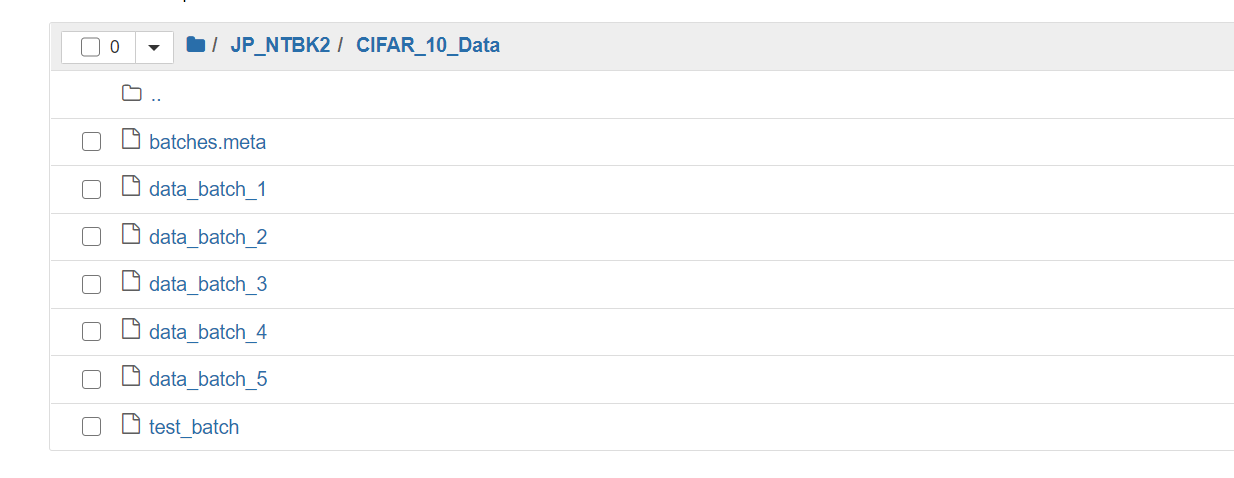


**Step-7: Uploaded CIFAR Dataset**

* The CIFAR-10 dataset consists of 60000 32x32 color 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.
* Initially downloaded the CIFAR files from Canvas /datasets as a Zip folder.
* Later created a new directory JP\_NTBK2 on my VM instance and then create a sub-folder CIFAR\_10\_Data.



* Later uploaded CIFAR files (7 files) onto /JP\_NTBK2/CIFAR\_10\_Data/ subfolder through SSH.



* As seen in above picture, the dataset is broken into batches to **prevent** machine from running **out of memory**. The CIFAR-10 dataset consists of 5 batches, named data\_batch\_1, data\_batch\_2, etc. As stated in the official web site, **each file** packs the data using **pickle** module in python.

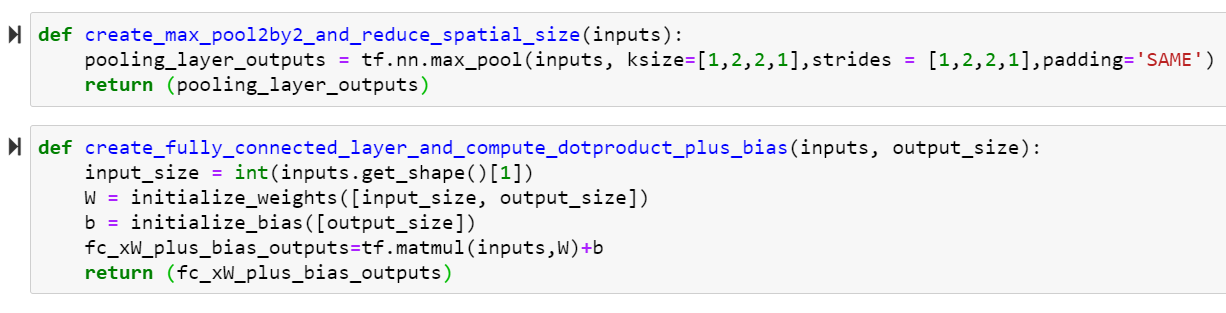
**Part-3 Build, Train, and Test CNN on CIFAR-10 Dataset**

* ‘[DL\_TF\_cnn\_cifar\_10\_final\_project.ipynb](http://localhost:8000/notebooks/JP_NTBK2/DL_TF_cnn_cifar_10_final_project.ipynb)’ File is downloaded from canvas and uploaded on to my VM to work on CIFAR dataset.



**Supporting Functions to build, Train and Test CNN model**

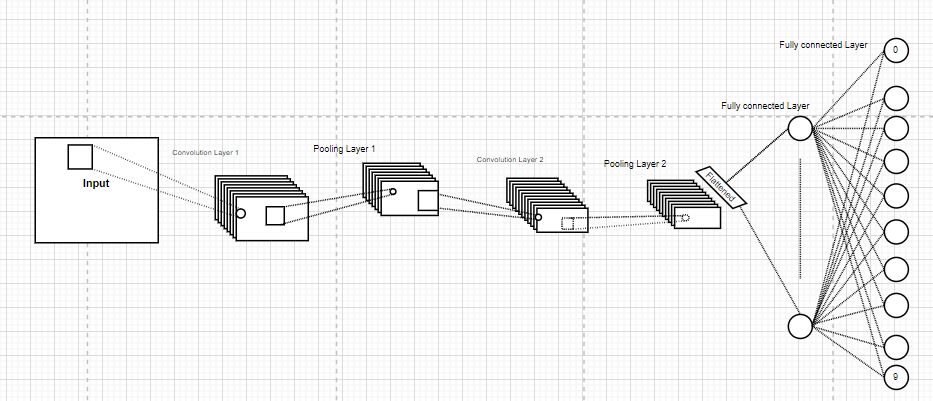




* All the defined functions are used to build the convolutional neural networks.

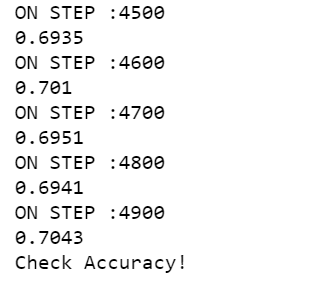
**Design of the Model**

* Here first convolution layer has been built using ReLU as activation function.
* Created first pooling layer that works along with convolution layer.
* Above steps are repeated, making 2 convolution and 2 pooling layers in total.

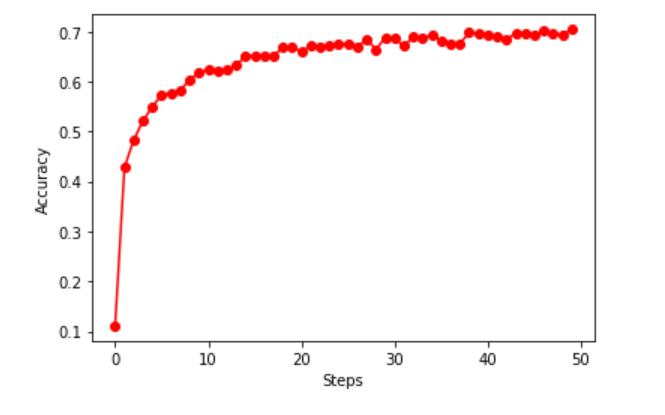


* Flattening layer has been created to reshape output data from pooling layer 2.
* Created a fully connected layer that accepts output from flattening layer.
* Above step is repeated to add few more fully connected layers.
* A dropout layer has been created to drop some of the inputs the model received.
* A fully connected final output layer is created that accepts the outputs from dropout layer as its inputs. This layer gives us the final output.
* Loss function is created to compute SoftMax cross entropy between the labels and the predicted outputs.
* An optimizer is created to optimize the network. An AI trainer is created to train the network.

**Results of Testing the Model**



* Accuracy gained by the model on the final step is 70.43 %



* We can clearly observe from the above graph that there is an increase in accuracy for each step of testing.

**Report on the results of the Test**

* As per the requirement, the model is tested with 5000 steps (On every 100th step).
* For each step a batch of 100 images are trained on the model and for every 100 steps the model is tested with test example.
* As we can see the model increased its accuracy or the prediction level for every next step of training. We can say that with increase in training the prediction accuracies can be improved depending upon the data.

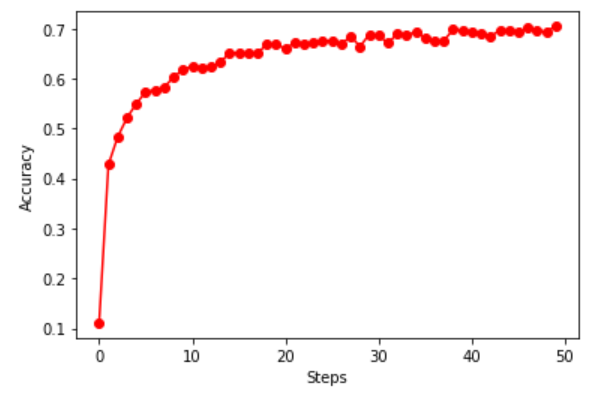
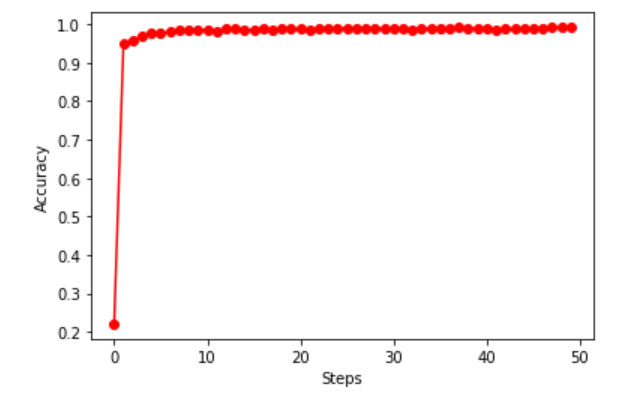
**Part-4 Convolutional Neural Network Performance**

**Comparison on MNIST and CIFAR**

* CIFAR10 is an image classification dataset which consists of 50,000 images which includes 45,000 training images and 5,000 test images. The images are 32\*32 size.
* MNIST is a handwritten digits’ dataset which consists of 60,000 images, including 50,000 training images and 10,000 test images. The input images are of size 28\*28.
* The accuracies obtained i.e 50 (5000 steps/ 100 (test on every 100th step)=50) on MNIST is considered as ‘ACC\_cnn\_mnist and on CIFAR is considered as ‘ACC\_cnn\_cifar\_10’.
* All these accuracies are collected for comparing the performance of CNN and CIFAR.

|  |  |  |
| --- | --- | --- |
| Step Number | ACC\_cnn\_mnist | ACC\_cnn\_cifar\_10 |
| 0 | 0.2208 | 0.111 |
| 100 | 0.9484 | 0.4277 |
| 200 | 0.958 | 0.4835 |
| 300 | 0.97 | 0.5216 |
| 400 | 0.9757 | 0.5498 |
| 500 | 0.9757 | 0.5734 |
| 600 | 0.981 | 0.5771 |
| 700 | 0.9835 | 0.5832 |
| 800 | 0.9839 | 0.6025 |
| 900 | 0.9837 | 0.6181 |
| 1000 | 0.9863 | 0.6245 |
| 1100 | 0.9818 | 0.6224 |
| 1200 | 0.9873 | 0.6231 |
| 1300 | 0.9874 | 0.6325 |
| 1400 | 0.9845 | 0.6507 |
| 1500 | 0.9859 | 0.6511 |
| 1600 | 0.9886 | 0.6503 |
| 1700 | 0.9855 | 0.6502 |
| 1800 | 0.9875 | 0.6687 |
| 1900 | 0.9873 | 0.6687 |
| 2000 | 0.9871 | 0.6592 |
| 2100 | 0.986 | 0.6722 |
| 2200 | 0.9879 | 0.6692 |
| 2300 | 0.9897 | 0.6709 |
| 2400 | 0.9894 | 0.6756 |
| 2500 | 0.9894 | 0.6751 |
| 2600 | 0.99 | 0.6689 |
| 2700 | 0.9895 | 0.6842 |
| 2800 | 0.9872 | 0.6642 |
| 2900 | 0.9894 | 0.6881 |
| 3000 | 0.9887 | 0.686 |
| 3100 | 0.9898 | 0.6707 |
| 3200 | 0.9866 | 0.6892 |
| 3300 | 0.9884 | 0.6877 |
| 3400 | 0.99 | 0.6926 |
| 3500 | 0.9889 | 0.6817 |
| 3600 | 0.99 | 0.6755 |
| 3700 | 0.9909 | 0.6761 |
| 3800 | 0.9902 | 0.6976 |
| 3900 | 0.9892 | 0.6968 |
| 4000 | 0.9887 | 0.6927 |
| 4100 | 0.9867 | 0.6897 |
| 4200 | 0.987 | 0.684 |
| 4300 | 0.9875 | 0.6968 |
| 4400 | 0.9893 | 0.6971 |
| 4500 | 0.9887 | 0.6935 |
| 4600 | 0.9898 | 0.701 |
| 4700 | 0.9907 | 0.6951 |
| 4800 | 0.9908 | 0.6941 |
| 4900 | 0.9916 | 0.7043 |

* **Number of convolutional networks used on both datasets**: 2. There is no difference between the CNN model design for calculating the accuracies on different datasets (CIFAR, MNSIT).
* On the final step we can see there is a difference of nearly 29% between the final accuracies.
* CIFAR dataset consists of color images and MNIST consists of handwritten digit images. It is easy for the deep learning model to predict a handwritten image instead of a color one.
* I have performed a trial an error method on checking the accuracies by differing its filter size.
* Highest error rate was noted with 1\*1 filter whereas the least was on 6\*6.
* However, we can say that final accuracies completely depend on the type of data, size of dataset, and level of tuning we performed on the DL model. We need to tune the hyper parameters such that model does not run through overfitting or underfitting.



* Model was able to predict the image accurately from initial steps in MNIST dataset. Whereas it in case of CIFAR, it started to increase its level of prediction after some consistent level of training.

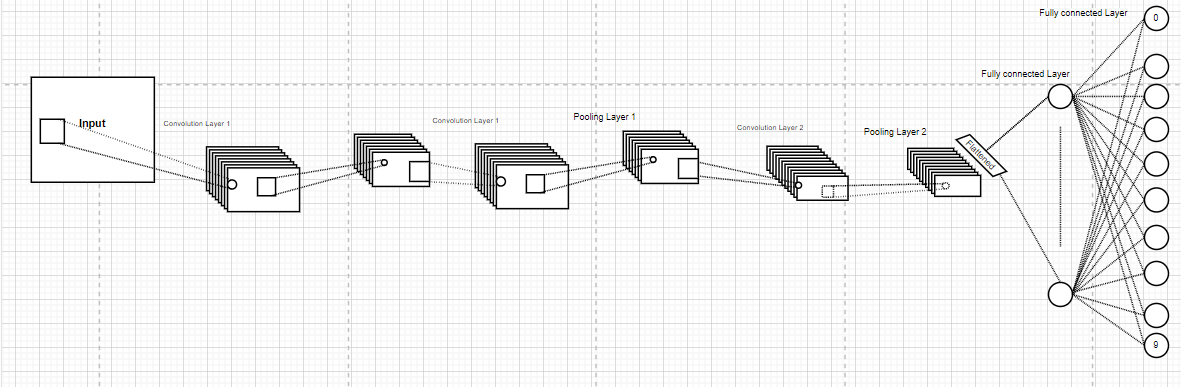
**Part-5 Improve Convolutional Neural Network Performance**

* I tried to gain the accuracy with different parameters, i.e by changing steps, number of CNN layers, Filter sizes, Number of training images in each step etc.
* I was managed to increase a minor percent of accuracy with different parameters as mentioned in next steps.

**Proposal of New Model**

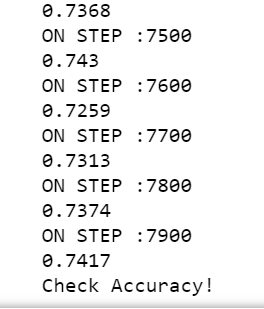
|  |  |  |
| --- | --- | --- |
| **Parameters** | **Designed model on CIFAR** | **RE-designed Model on CIFAR** |
| Number of steps | **5000** | **8000** |
| Number of CNN layers | **2** | **3** |
| Number of training images in each step | **100** | **100** |
| Final Accuracy | **70.43** | **72.90** |

**Redesigned Model**

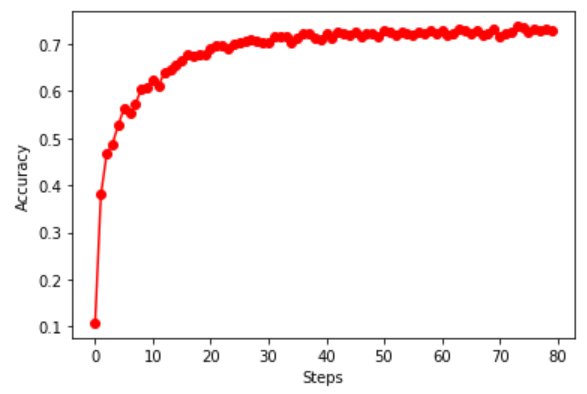


* Here first convolution layer has been built using ReLU as activation function.
* Created first pooling layer that works along with convolution layer.
* Above steps are repeated twice, making 3 convolution and 3 pooling layers in total.
* Flattening layer has been created to reshape output data from pooling layer 3.
* Created a fully connected layer that accepts output from flattening layer.
* Above step is repeated to add few more fully connected layers.
* A dropout layer has been created to drop some of the inputs the model received.
* A fully connected final output layer is created that accepts the outputs from dropout layer as its inputs. This layer gives us the final output.
* Loss function is created to compute SoftMax cross entropy between the labels and the predicted outputs.
* An optimizer is created to optimize the network. An AI trainer is created to train the network.

**Results of testing the model**



* Accuracy gained on the final step is 72.90 %.



* Picture showing the accuracy levels of redesigned model on every 100Th step.

**Report on the results**

* To improve the accuracy the model is trained with 8000 steps (On each step 100 images are used for training).
* For each step a batch of 100 images are trained on the model and for every 100 steps the model is tested with test example.
* The only aim of redesigning the model is to increase accuracy and it completely depends on the type of data we are working on.
* With an increased amount of training (Through number of steps) and an increasing in number of layers of CNN might be the causes for gained accuracy.
* However, we might not be able to guess the exact parameter to be changed, to get the desired accuracy. As observed from a research, few algorithms may not generate high accuracy no matter at what level we tune the parameters.

**Part-6 Project Report**

1. **Introduction**

In this project I have worked on designing convolutional neural networks on different datasets. A two layered CNN has been created initially. Later to improve its accuracy additional layers have been added with some level of tuning. The main aim of the project is focused on exploring the datasets needed for Deep learning models on platforms like Kaggle, GitHub etc. and effectively designing the CNN models depending upon the datasets. The DL models designed in this project are completely implemented in Python language (TensorFlow library). Comparisons have been made by running the similarly designed CNN on different datasets such as MNSIT and CIFAR. All the observations and conclusions about the project are mentioned below.

1. **Description of work done**

Step-1:

A two layered CNN has been designed along with their respective pooling layers. ReLU activation function is used in each convolutional layer. Flattening layer has been created to reshape the output data from the final pooling layer. A fully connected layer is created which will accept output of flattening layer. A dropout layer has been created to drop some of the inputs received. Loss functions and optimizers are used along. This design is executed on CIFAR dataset which gave me nearly 70 % accuracy. Further research to improve the accuracy of the model is documented in next steps.

Step-2:

The design mentioned in step-1 is executed on MNIST dataset in previous works. Accuracy gained on the MNIST dataset is 99.16% By comparing the two results we can say that accuracy or the prediction levels completely depends on the data type we have. In MNIST dataset the image is handwritten digit coded and in CIFAR they are colored. So, there is a difference in the type of data even though there is no change in the model design.

Step-3

Research has been conducted to improve the accuracy of the model mentioned in step 1. A CNN layer has been added to the existing model. And the number of steps for training the model has been increased to 8000 (Previously it was 5000). I was able to gain nearly 73% with the new design on the same dataset. A difference of 3% is observed. Apart from this no further changes were made to the model.

**3.Conclusions**

**Step:1: Initial design on CIFAR**

Model designed gained an accuracy of 70%

**Step:2: Comparing with MNIST**

Similar model of step 1 gained 99% accuracy on MNIST dataset.

**Step:3: Redesign of model on CIFAR**

The model has been redesigned adding a CNN with some extent of tuning, it resulted in increased accuracy of nearly 3% (73% gained).

**4. Conclusions on the Project**

We can conclude that accuracy of the model increases on increasing the CNN layers (depends on the data we have considered) and certain level of hyper tuning the model will also result in increased accuracy. Sometimes no matter how many layers we may add and how much level of tuning we perform the accuracy might not change.

**Part-7 YouTube Video**

Link to the video presentation- <https://youtu.be/X4kVxQ60-80>

**References:**

[Towards Data Science published paper](https://towardsdatascience.com/cifar-10-image-classification-in-tensorflow-5b501f7dc77c)

[Research Paper on Classification using CNN](http://athena.ecs.csus.edu/~hoangkh/Image%20Classification%20with%20Fashion-MNIST%20and%20CIFAR-10%20-%20Final%20Report.pdf)