**ADTA- 5550: Final Project**

**Sri Charan Bodduna, Student Id: 11661666**

**Part I**

1.1: YES

1.2: NO

**Part II**

I have taken the CelebA dataset. It is an abbreviated form of CelebFaces Attributes Dataset. This dataset consists of images. CelebA is a large-scale face attributes dataset with around 200,000 celebrity images, containing 40 attributes interpreting celebrity faces. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including 10,177 number of identities, 202,599 number of face images, and five landmark locations, 40 binary attributes annotations per image.

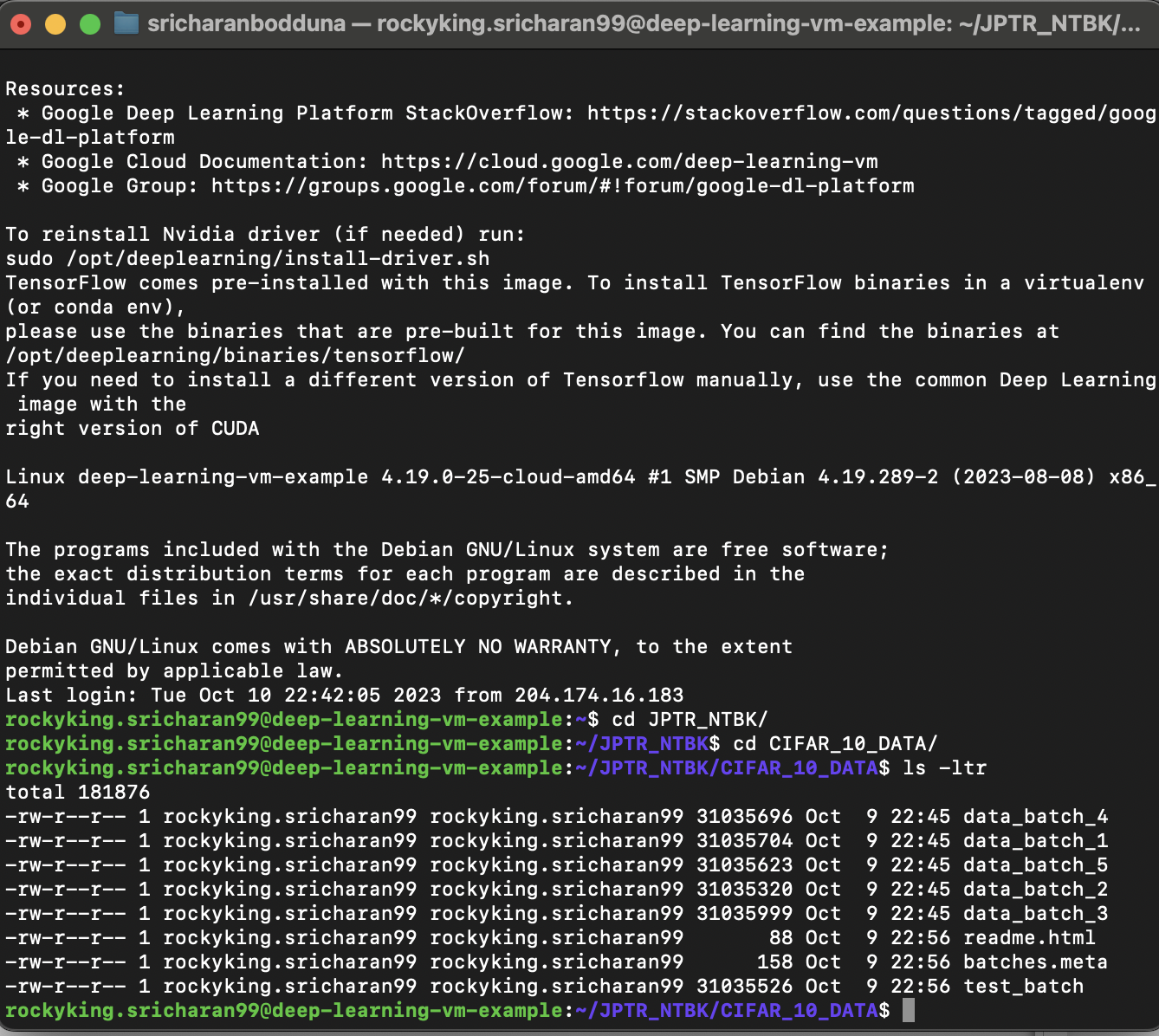
The data in CelebA dataset is properly labelled for deep learning modelling. The dataset has separate data records for training and test sets with sizes of 162,770 and 19,962 respectively. There is also a validation set of 19,867 for the following computer vision tasks: face attribute recognition, face recognition, face detection, landmark (or facial part) localization, and face editing & synthesis.

I have downloaded the data from the Chinese University of Hong Kong website - <https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>. We can also download the data files from the same URL (<https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>).

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| image | tensor containing the 178×218 image. |
| bbox | tensor containing bounding box of their respective images. |
| keypoints | tensor to identify 63 various key points from face |
| clock\_shadow | tensor to check cloak shadow. |
| arched\_eyebrows | tensor to check arch eyebrows. |
| attractive | tensor to check if attractive or not. |
| bags\_under\_eyes | tensor to check if bags are under the eyes. |
| bald | tensor to check if bald or not. |
| bangs | tensor to check if bangs are there or not. |
| big\_lips | tensor to check if big lips are there or not. |
| big\_nose | tensor to check if big nose is there or not. |
| black\_hair | tensor to check the presence of black hair. |
| blond\_hair | tensor to check if blond hair or not. |
| blurry | tensor to check if the image is blurred. |
| brown\_hair | tensor to check the presence of brown hair. |
| bushy\_eyebrows | tensor to check the presence of bushy eyebrows. |
| chubby | tensor to check if chubby or not. |
| double\_chin | tensor to check the presence of double chin. |
| eyeglasses | tensor checks the presence of eyebrows. |
| goatee | tensor to check the presence of a goatee in a person. |
| gray\_hair | tensor to check the presence of gray hair. |
| heavy\_makeup | tensor to check the presence of heavy makeup. |
| high\_cheekbones | tensor to check the presence of high cheekbones. |
| male | tensor to check if the person is male. |
| mouth\_slightly\_open | tensor to check if the mouth is open. |
| mustache | tensor to check the presence of a mustache. |
| narrow\_eyes | tensor to check narrow eyes or not. |
| no\_beard | tensor to check if the beard is present. |
| oval\_face | tensor to check if the face is oval. |
| pale\_skin | tensor to check if the skin is pale. |
| pointy\_nose | tensor to check if the nose is pointy. |
| receding\_hairline | tensor to check if the hairline is receding. |
| rosy\_cheeks | tensor to check if the cheeks are rosy. |
| sideburns | tensor to check the presence of sideburns. |
| smiling | tensor to check if the person is smiling. |
| straight\_hair | tensor to check if the hair is straight. |
| wavy\_hair | tensor to check if the hair is wavy. |
| wearing\_earrings | tensor to check the presence of earing. |
| wearing\_hat | tensor to check the presence of the hat. |
| wearing\_lipstick | tensor to check the presence of lipstick. |
| wearing\_necklace | tensor to check the presence of the necklace. |
| wearing\_necktie | tensor to check the presence of necktie. |
| young | tensor to check if the person is young. |

**Part III**

I have downloaded the CIFAR-10 dataset from the modules (CIFAR-10 python version) in canvas. Later I have uploaded the files into the Jupyter environment on a GCP deep learning example remote server under JPTR\_NTBK/CIFAR\_10\_DATA. I have also attached the screenshot of the file and data structure in the Jupyter environment.

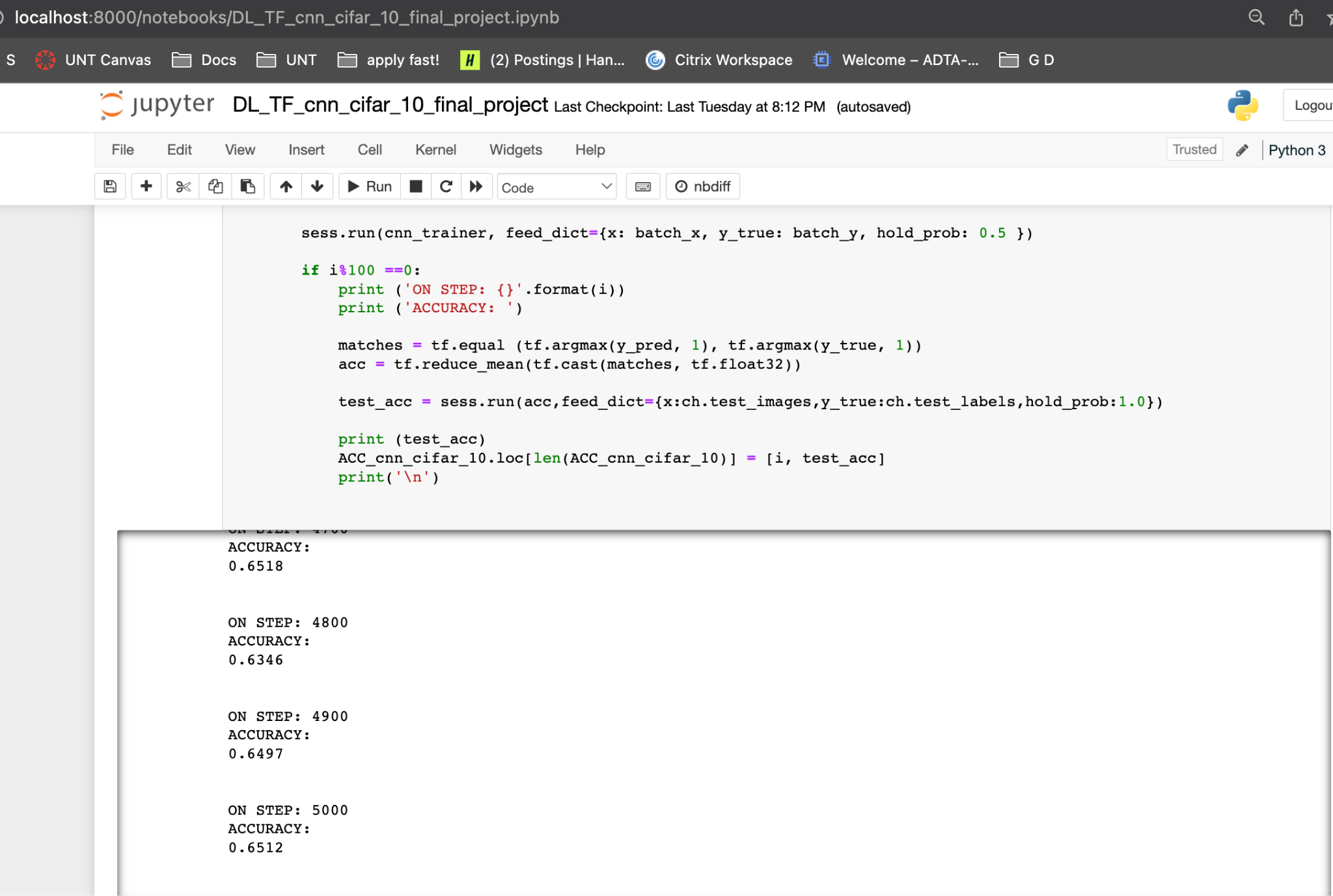


**Part IV**

Based on the lectures and previous assignments, I have built a new CNN model of two convolutional layers and a fully connected layer with a dropout on the CIFAR-10 dataset. I have first initialized weights and bias for the model. I have then created required functions for the convolutional layer and ReLU layer combined, 2x2 pooling layer (of type max-pooling), fully connected layer.

I have first applied the data of size 32x32x3 to the first convolutional layer + ReLU layer with the filter shape of 5x5, input connections as 3 and output connections as 32. The output of the convolutional layer is applied to the max pooling layer which has dropped the dimensions of the image by half. Now, the output of the max pooling layer is applied to the input of the second convolutional layer with filter shape as (3x3x32x64). Next another max pooling layer of size 2x2. Then, I flattened the output to the size 4096. Later I connected the flattened output to the 1st Fully Connected layer which gives 1024 as output size, followed by dropout layer is added and finally last fully connected layer with 10 output nodes.

Below image shows the results obtained from CNN model 1.



After running the model for 5000 steps, the results showed that the highest accuracy was found out to be **65.12%.** In this the basic model is built on CIFAR-10 dataset, hence there are many assumptions and improvements need to be performed on this model. These improvements have been discussed in Part VII.

Below image shows the design of the model.

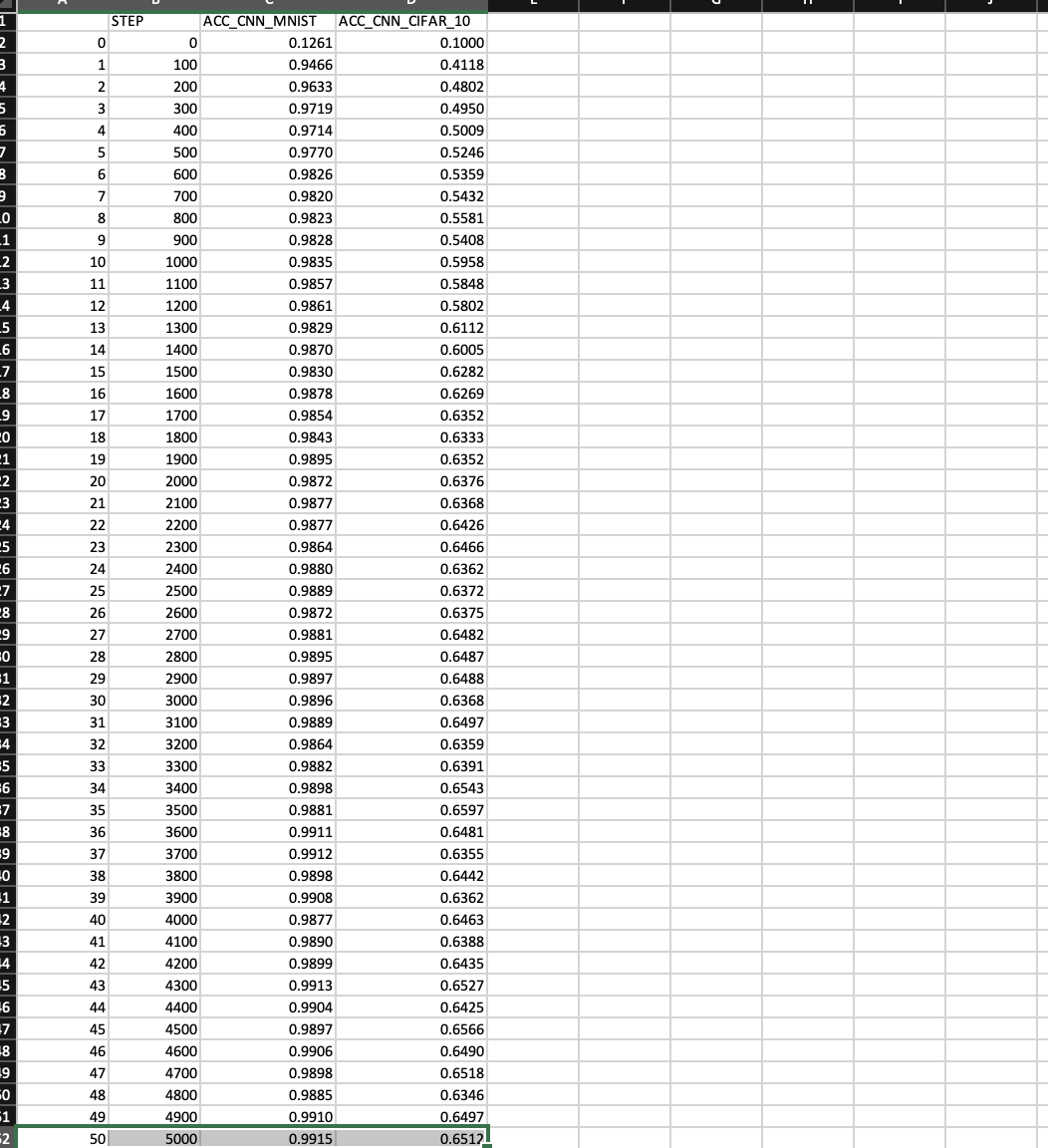
A diagram of a machine

Description automatically generated

(design created in google slides)

**Part V**

I have pulled the test accuracy results from both the models. We can see the results by comparison side by side. I have taken a total 50 samples of test accuracies and ran over 5000 steps of training the model. Here we can see the accuracy of the MNIST model is 99.15% at 5000th step and the CIFAR-10 model has 65.1%. As these both models have 2 convolutional layers followed by max pooling layers and thereby joining with fully connected layers. Hence these two models are nearly similar but having parameters as per their respective datasets.



Evaluation: I have taken the average of the last ten samples of accuracies i.e., 41-50 indices. I got 99% for the MNIST model and 64.61% for the CIFAR-10 model.

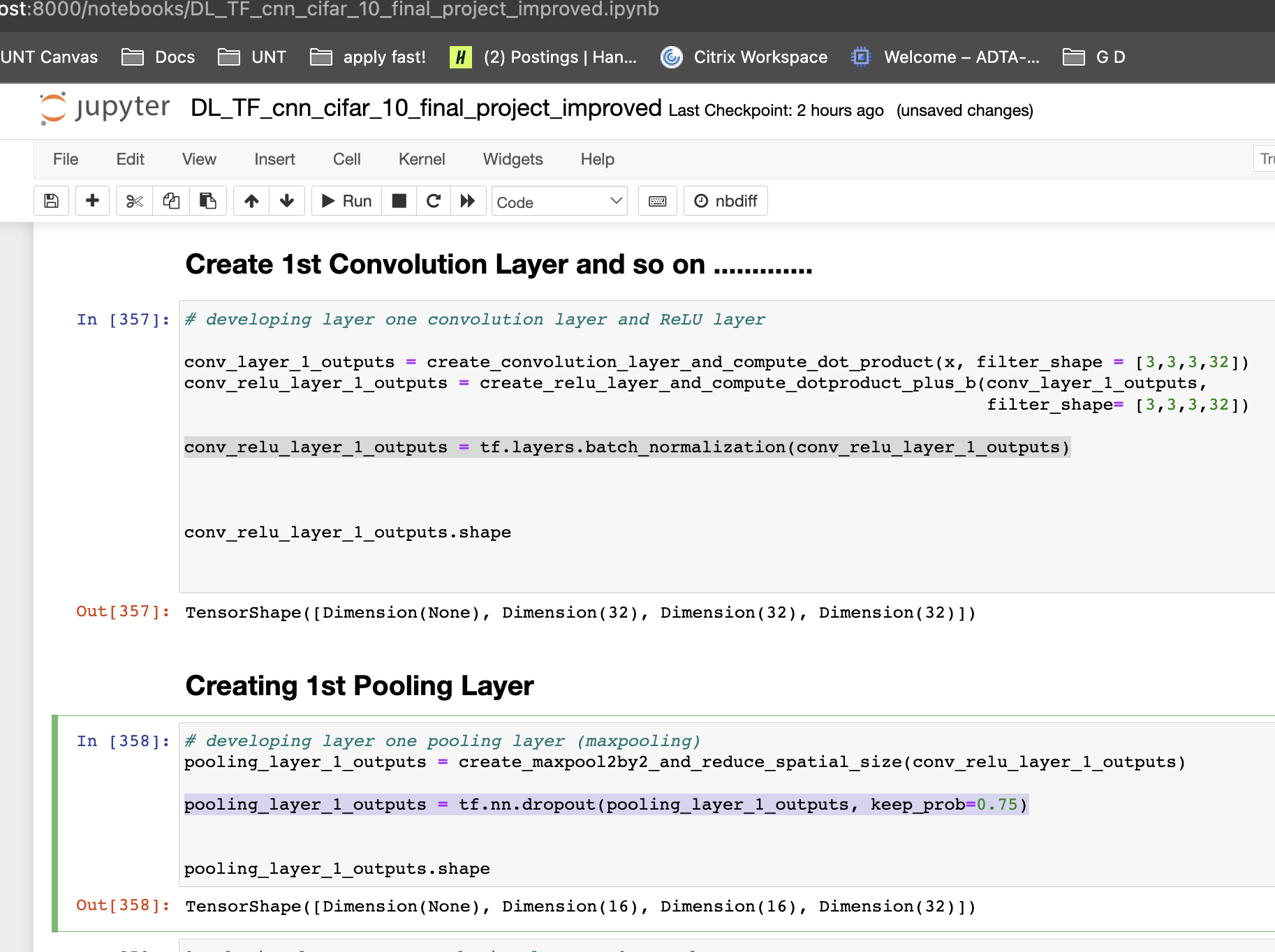
One of the possible reasons for lesser accuracy for the CIFAR-10 model is due the input data size is (32 x 32 x 3) since the data rack is of 3 dimensional (RGB colors) hence we might need a further hyper parameter tuning.

**Part VI**

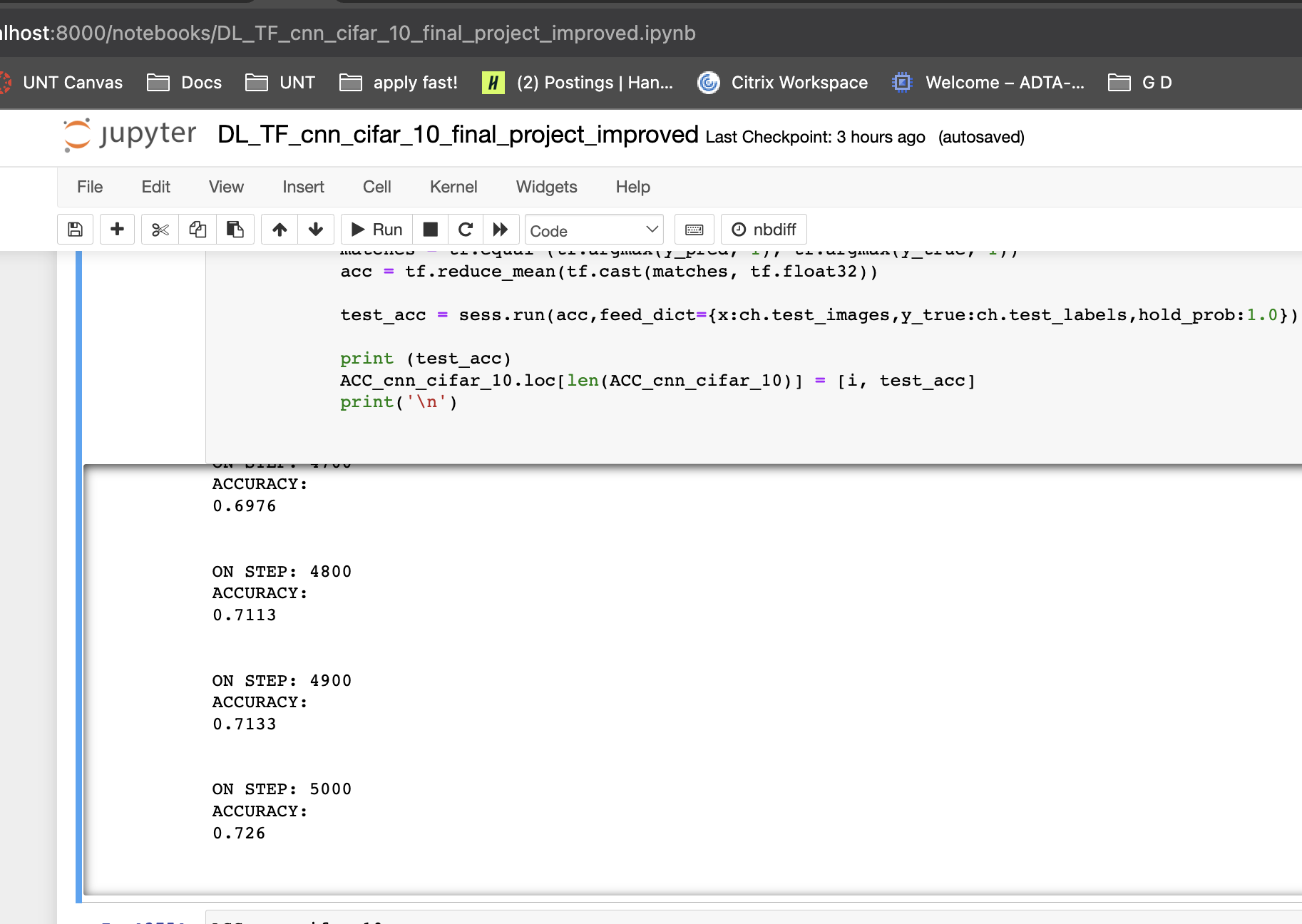
In this section, we will discuss more about improvements made on the model 1 of the CIFAR-10 dataset. Since we are performing modeling on 3d data which means each image contains 3 layers of color (RGB). hence the size of the interim processing will be high. So, I have used batch normalization to reduce dependency and improve learning rate of the model, Hence I have added batch normalization layer between convolutional and pooling layers. I have also added a dropout of 0.25 after each max pooling layer. Adding dropout layers will regularize the outputs produced from previous layers.

I have also tried various hyper parameter tuning measures like adding 3rd and 4th layers, increasing depth of convolutional layers to 128, 256 etc. but I did not see any improvements in it.

Below image shows improvements made to the model in highlighted.



The highest test accuracy I got for the improved model is **72.6%** at 5000th step. That is a good overall improvement in the model based on the hyperparameter tuning.



**Part VII**

Introduction

In this final project, I have developed a neural network model on the CIFAR-10 dataset. CIFAR (Canadian Institute for Advanced Research) have developed this dataset which contains images of various vehicles, birds and animals. There are total 10 classes in the dataset. The data have 60000 observations in total.

Convolutional Neural Network (CNN) is a neural network model which consists of one or many convolutional layers which means convolution operation of input with the kernel. The resultant dot product will be applied to the ReLU layer which acts as an activation function. These two layers work in tandem. We can have any number of these combined layers in CNN model based on the parameters and data size. Later, we apply the result to the pooling layer which compresses the previous output dimensions by half. We can have pooling layer after convolutional layer. Later, we flatten the result from the pooling layer and apply it to a fully connected layer. Which can be any number of layers based on hardware and parameters in the dataset. Finally, we have the output nodes which are equal to the number of classes in the dataset.

Project description

In the model 1 as part of the preliminary model, I have developed the model with CNN containing 2 convolutional layers which are also followed by max pooling layers. Then connected a fully connected layer by applying a flatten version of pooled output. The model has produced an accuracy of 59-61% on testing data when run for 5000 steps. Since this model is simple and not hyper parameters tuned, I have made a few assumptions which helps the model to improve its accuracy. My assumptions are normalizing the data between convolutional layers. My first assumption was to normalize results from the convolutional layers. I have added batch normalization in between the convolutional layer and max pooling layer. Batch normalization improves gradient flow in the network and reduces the need for regularization.

Then I compared the model with the results of the MNIST CNN model. CNN models on MNIST dataset have produced the results of accuracies greater than 94%. The models developed on CIFAR-10 have produced only 60% with 2 convolutional layers.

From the assumptions made on model 1 of CNN on CIFAR-10 dataset, I have the necessary improvements and built model 2. There is an improvement in model 2 after doing hyper parameter tuning. As explained during class session, batch normalization helps in improving gradient flow through the network. Which has been worked out in my case.

Conclusions from the results

1. As we generalize the results from model 1 of CIFAR-10 CNN, we can see the accuracy found is low since the model is very simple, not tuned, and not regularized.
2. I have added batch normalization and dropout layers in convolutional layers as I felt these are essential to have better results since we are performing predictive modeling on 3d images.
3. Adding these improvements which helped the model to improve test accuracies by 7%. So, the final test accuracy I got for the improvised model is 72.6%. Which is a significant improvement. The average accuracy for the last 10 samples of test accuracies is 71.3%

Conclusion

Finally, as we saw the basic model with lesser accuracy which led to drawing assumptions to improve the model capacity. Thereby, building the model 2 based on the improvements I understood from model 1. Model 2 have resulted in better results and accuracies when running through 5000th steps of training data and testing the model at 50 distinguished steps.

In this modeling project using CNN on CIFAR-10. I have learned various hyper tuning parameters that drive the model and improve overall performance and predicting the results more accurately. It was a major setback to strengthen the model in terms of accuracy.

Reference:

1. Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep Learning Face Attributes in the Wild. In Proceedings of the International Conference on Computer Vision (ICCV), December 2015.
2. Dataset link for part II: <https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>
3. CelebA dataset attributes taken from: <https://datasets.activeloop.ai/docs/ml/datasets/celeba-dataset/>