**PART I: Use TensorFlow Directly in Coding (5 Points)**

**TO-DO:**

Answer the following questions:

• **Question 1.1**: Is the student required to use TensorFlow directly in coding (build, train, and test CNN) in this homework assignment?

**Answer:**  Yes, as per the instructions provided, we are required to use TensorFlow for coding in this assignment.

• **Question 1.2**: Should the student use Keras in coding (build, train, and test CNN) in this homework assignment?

**Answer:** No, as per the instructions provided, we should not be using Keras for this assignment.

**PART II: A Dataset of Images or Audio Files (10 Points)**

**TO-DO:**

Search on the Internet, using Google search or any other approach, to find a dataset in the public domain, i.e., available for use without restrictions.

--) This dataset may contain either images or audio files.

--) This data of the dataset has already been appropriately labelled and ready for use in deep learning research.

--) The dataset should not be one of the datasets that are used in classwork: MNIST, CIFAR-10 (or CIFAR-100).

**Answer:** The Intel Image Classification dataset contains about 25000 images. The size of each image is 150X150. There are six categories in the dataset which are “buildings”, “forests”, “glacier”, “mountain”, “sea”, “street”.

The details of the dataset are described below.

**Name:** Intel Image Classification

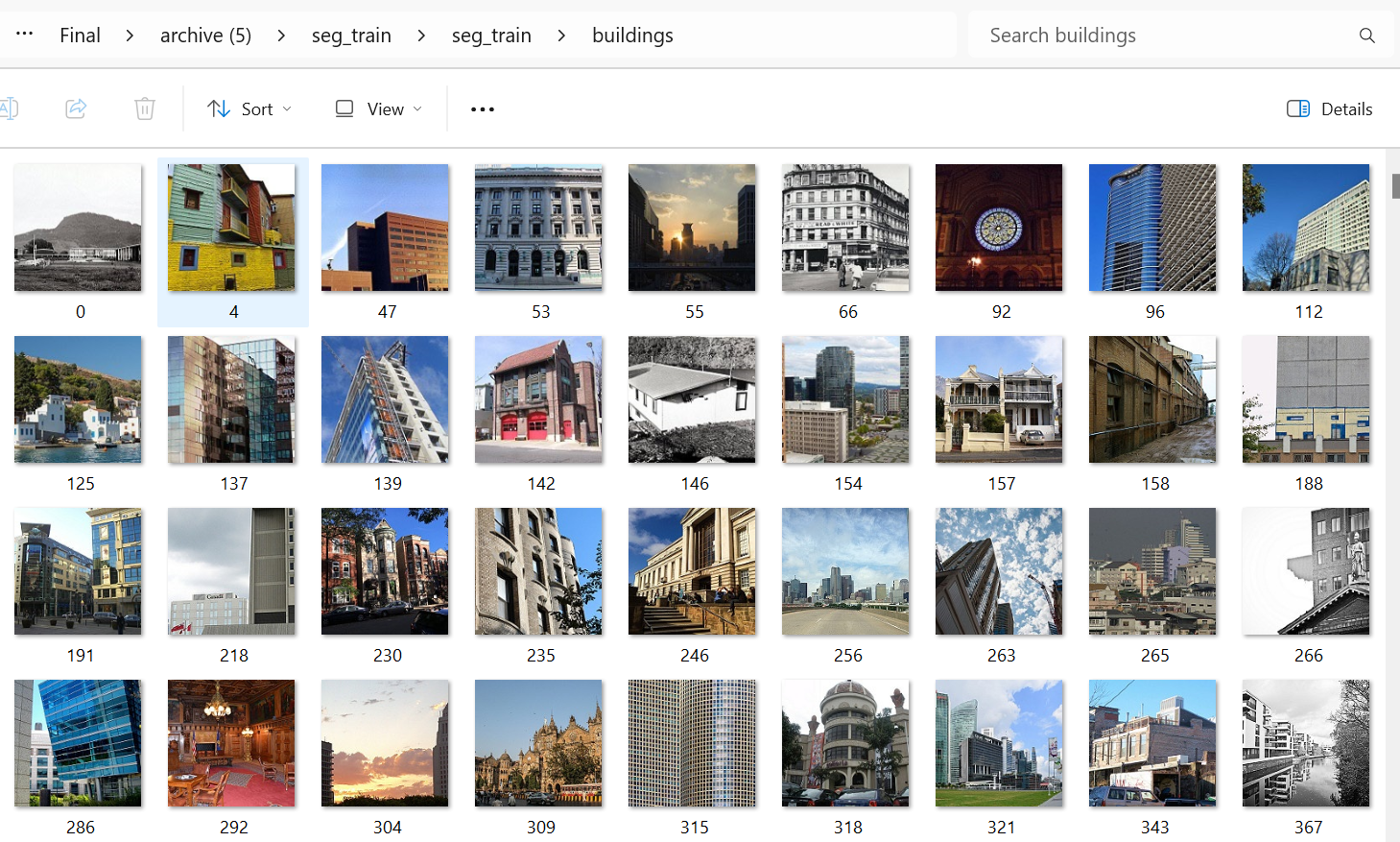
**Source:** https://www.kaggle.com

**links to download:** <https://www.kaggle.com/datasets/puneet6060/intel-image-classification>

**No of images:** 25k images of size 150x150

**Size:** 363 MB

**Screenshot of the dataset:**

****

**PART III: Obtain CIFAR-10 Dataset (5 Points)**

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 sub-datasets and one test sub-dataset, each with 10000 images. The test sub-dataset contains exactly 1000 randomly-selected images from each class. The training sub datasets contain the remaining images in random order, but some training sub-datasets may contain more images from one class than another. Between them, the training datasets contain exactly 5000 images from each class.

The classes are entirely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

**TO-DO**

**--) Access the Canvas module: …/DATA\_SETS**

**--) Download all the dataset files available there (7 files)**

**• NOTES: These data files belong to the CIFAR-10 dataset**

**--) Transfer all the data files of the dataset to the remote virtual machine**

**• Access the remote virtual machine in GCP using SSH:**

**• Open the sub-folder JP\_NTBK in the remote VM**

**• Create a new sub-folder under ~/JP\_NTBK and name it as “CIFAR\_10\_DATA”**

**• Upload all the data files of the aforementioned dataset, CIFAR-10, from the student’s local computer to the newly-created sub-folder in the remote instance.**

**Answer:** The capability to upload the datafiles to the remote virtual machine enables the files to be accessible while implementing the deep learning algorithms like MLP, CNN etc on the remote deep learning virtual machine.

* Downloaded the CIFAR-10 dataset from Canvas.

**A screenshot of a computer

Description automatically generated**

Extracted the files from the tar.gz file.

* **Accessing the remote virtual machine in GCP using SSH:** From the cloud console UI, select SSH (as shown in the below screenshot) and authenticate to see the “SSH-in-browser”.

A screenshot of a computer

Description automatically generated

**Figure:** SSH from cloud console UI

The below screenshot displays the “SSH-in-browser” to the remote virtual machine.

A screenshot of a computer screen

Description automatically generated

**Figure**: Connecting to remote virtual machine using SSH

* In the below screenshot we can see that, “**JP\_NTBK**” is selected and subfolder “**CIFAR\_10\_DATA**” is created.

A screen shot of a computer program

Description automatically generated

**Figure**: Showing folder “JP\_NTBK” and sub-folder “CIFAR\_10\_DATA”

* **Uploading the datafiles to sub-folder “CIFAR\_10\_DATA”**

We used the “upload file” option from the terminal to select the file from the local machine to upload them to the sub-folder “CIFAR\_10\_DATA” in the remote virtual machine.

The below screenshot shows the files in the upload process.

A screenshot of a computer

Description automatically generated

**Figure**: files getting uploaded to the sub-folder in the remote virtual machine.

The below screenshot shows the files in the subfolder “CIFAR\_10\_DATA” .

A screen shot of a computer

Description automatically generated

**Figure:** The screenshot shows the files successfully uploaded to the sub folder

We can verify the same when we launch the localhost. The below screenshot displays the files in the path “JP\_NTBK/CIFAR\_10\_DATA”.

With this the data files are ready to be accessed by the code that we are going to write in the remote deep learning virtual machine.

A white and black line

Description automatically generated with medium confidence

**Figure:** data files in the path “JP\_NTBK/CIFAR\_10\_DATA” in localhost

**PART IV: Build, Train, and Test CNN on CIFAR-10 Dataset (30 Points)**

**TO-DO**

**--) Design the convolution neural network used for the project**

**--) Build, train, and test the convolutional neural network on the CIFAR-10 dataset using the TensorFlow AI framework and Python:**

**• Train the network with 5000 (five thousand) steps**

**• Test the network after every 100 steps of training**

**o NOTES: each time of testing, one data point of the accuracy level can be collected.**

**--) After building, training, and testing the model, copy the results of the tests into a section of the project report (ADTA5550\_final\_project.docx).**

**--) Write a report on the results of testing the model (in the same project report file)**

**Answer:**

**Model design and the layers are described as:**

The below details describe the design of the implemented model.

* **Convolution Layer 1:**
  + Convolution Shape: [batch,H,W,depth]= [1, 32, 32, 32]
  + 2D datasize: 32X32
  + Input Shape: [32X32X 3]
  + Output Shape: [32X32X 32]
  + Filter/Kernel/Window size: 5X5
  + Filter Shape= weight shape: [5,5,3,32]
  + Stride =1
  + Stride Shape: [1,1,1,1]
  + Padding: SAME
  + Activation Function Layer: ReLu
* **ReLu Layer 1:**
  + No filter (only process data, no extract/learn features)
  + Input Shape: [32X32X 32]
  + Output Shape: [32X32X32]
* **Pooling Layer 1:**
  + Pooling method: Max pooling
  + Filter/Kernal/Window size: 2X2
  + Filter/Kernal/Window Shape: [1,2,2,1]
  + Stride: 2
  + Stride Shape: [1,2,2,1]
  + Padding: SAME
  + Input channel: 32 inputs
  + Input shape: [32X32X 32]
  + Output channel: 32 outputs
  + Output shape: 16X16X32
* **Convolution Layer 2:**
  + Convolution Shape: [batch,H,W,depth]=[1,16,16,64]
  + 2D datasize: 16X16
  + Input Shape: 16X16X32 (depth=1 in channel)
  + Output Shape: 14X14X64 (depth=32 out channel)
  + Filter/Kernel/Window size: 5X5
  + Filter Shape= weight shape: [5,5,32,64]
  + Stride =1
  + Stride Shape: [1,1,1,1]
  + Padding: SAME
  + Activation Function Layer: ReLu
* **ReLu Layer 2:**
  + No filter (only process data, no extract/learn features)
  + Input Shape: 14X14X64
  + Output Shape: 14X14X64
* **Pooling Layer 2:**
  + Pooling method: Max pooling
  + Filter/Kernal/Window size: 2X2
  + Filter/Kernal/Window Shape: [1,2,2,1]
  + Stride: 2
  + Stride Shape: [1,2,2,1]
  + Padding: SAME
  + Input channel: 64 inputs
  + Input shape: 16X16X64
  + Output channel: 64 outputs
  + Output shape: 8X8X64
* **Fully Connected Layer 1(FC\_1):**
  + FC\_1 shape: [inputs (in\_channels), outputs (output\_channels)]

= [8X8X64X1024]

* **Fully Connected Layer 2(FC\_2):**
  + FC\_2 shape: [inputs (in\_channels), outputs (output\_channels)]

= [1024X1024]

* **Fully Connected Layer 3 (FC\_3):**
  + FC\_3 shape: [inputs (in\_channels), outputs (output\_channels)] = [1024, 10]

A diagram of a network of data

Description automatically generated with medium confidence

**Figure**: design of the CNN(Source: Lecture video)

The model is trained for 5000 steps and tested for every 100 steps.

**Results of testing the model:**

The below screenshot displays the result of the model. Number of steps used is 5000 and the model has he accuracy of 0.1004

Which is 10.04%.

A screenshot of a computer

Description automatically generated

**Figure**: Result of the model

**Report on the results of the test:**

The model runs for 5000 steps and tested for every 100th step.

In the below screenshot we can see that, the accuracy for the 0th step is 10.0% but there is an increase on 100th step where the accuracy is increased to 12.17%. The accuracy again comes down to 10.04% at the step number 5000.

A screenshot of a computer

Description automatically generated

**Figure**: Result of the model at the initial steps

A screenshot of a computer

Description automatically generated

**Figure**: Result of the model at the final steps

**Jupyter Notebook:**  ADTA5550\_Final\_Project\_Part\_IV.ipynb is submitted as part of the assignment for this section.

**PART V:** **Compare Convolutional Neural Network Performance (10 Points)**

**TO-DO**

**In HW 4, when the student runs the code (in a Jupyter Notebook document) to train and test the CNN on the data set MNIST, the accuracy level is printed 50 times (one test for every 100 steps of training).**

**--) Collect the data (50 data points) of accuracy levels produced by the CNN with the MNIST dataset that the student worked on in HW 4. Let’s name the collected dataset as “ACC\_cnn\_mnist.”**

**--) Collect the data (50 data points) of accuracy levels produced by the CNN with the CIFAR-10 dataset. Let’s name the collected dataset as “ACC\_cnn\_cifar\_10.”**

**--) Compare the performance of the two CNN’s that have been used on the datasets – MNIST and CIFAR-10**

**--) Using critical thinking, write a report on the results of comparing their performance, i.e., the accuracy levels, in which the student is expected to provide possible reasons to explain the gap in the performance of the two CNN’s used on these two datasets if such a gap exists.**

**Answer:**

**Accuracy levels produced by the CNN with the MNIST** dataset are saved in file “ACC\_cnn\_mnist.docx”. This file will be submitted as part of this assignment in the canvas.

**Accuracy levels produced by the CNN with the CIFAR-10** datasetare saved in file “ACC\_cnn\_cifar\_10.docx”. This file will be submitted as part of this assignment in the canvas.

**Comparing the performance of the two CNN’s that have been used on the datasets:** The below screenshot shows the accuracy for both MNIST and CIFAR-10 dataset.

* We can see that there are steep fluctuations in accuracy for MNIST dataset. The accuracy of CIFAR-10 dataset does not fluctuate much.
* Final accuracy for MNIST dataset is 94.72% which is very good.
* Final accuracy for CIFAR-10 dataset is 10.04% which is a low accuracy.

**A white rectangular object with black text

Description automatically generated**

**Figure:** The accuracy for MNIST and CIFAR-10 datasets

**Explanation of the gap in the performance of the two CNN’s used on these two datasets:**

The difference in accuracy levels for MNIST and CIFAR-10 can be explained with the below mentioned points.

* **Complexity:** 
  + MNIST is a comparatively simpler dataset which consists of 70,000 handwritten images of number 0-9 in greyscale which means no colours. The size of each image is 28x28. The digits are centred and standardized which makes the processing less challenging and lesser complex.
  + On the other hand, CIFAR-10 is a much complex dataset having 60,000 colourful images of size 32X32 with different classes. The images are not positioned in the centre and have varied background. This makes it much more complex to process by the algorithm.
* **Architecture of the Model:**
  + The architecture of the model used has two convolutional layers, two ReLu layers, two pooling layers and three FC layers. This model can handle the MNIST dataset which is simpler.
  + CIFAR-10 dataset needs a much deeper network with more convolutional layers and complex architecture.

**Conclusion:** With these observations we can conclude that the possible reasons which explain the performance gap for the two datasets is the difference in the complexity of the dataset and the architecture of the Convolutional Neural Network used to process them.

**PART VI: Improve Convolutional Neural Network Performance (20 Points)**

**TO-DO**

**--) It is assumed that there is a gap in the performance of the two CNN’s.**

**--) Based on the student’s observations while working on PART IV, he/she is asked to think of some ways to improve the performance of the CNN that has worse performance.**

**--) Using critical thinking and the experiences of working with the MNIST dataset, the student proposes some changes to the network or the network training process with which the network performance may be improved.**

**--) Write a proposal to make changes to improve the CNN performance (in the above project report file)**

**--) Make the changes to the network or the network training process as proposed in coding**

**--) Build, train, and test the updated convolutional neural network on the CIFAR-10 dataset using the TensorFlow AI framework and Python with the proposed changes implemented in another Jupyter Notebook document.**

**--) Write a report on the results of testing the model (in the above project report file)**

**Answer:**

**Proposal to make changes to improve the CNN performance:** We can improve the architecture of the CNN to increase the accuracy for CIFAR-10 dataset.

A few enhancements are proposed below to achieve higher performance.

* **Adding More Convolutional Layers:** The model used in part IV has two convolutional layers. A deeper network with more convolutional layers can capture more details from this complex dataset.
* **Adding Dropout:** it will help to reduce overfitting in the model.
* **Increased Filter Size:** larger filter size will help to capture more details from the dataset.

**Design of the newly created CNN for better performance**

The redesigned convolutional neural network has four convolutional layers, a dropout layer after the first convolutional layer and increased filter size. The below details describe the architecture of the redesigned CNN.

* **Convolution Layer 1:**
  + Convolution Shape: [batch, H, W, depth] = [?, 32, 32, 64]
  + 2D datasize: 32x32
  + Input Shape: 32x32x3 (depth=3 in channel)
  + Output Shape: 32x32x64 (depth=64 out channel)
  + Filter/Kernel/Window size: 3x3
  + Filter Shape: [3, 3, 3, 64]
  + Stride: 1
  + Stride Shape: [1, 1, 1, 1]
  + Padding: SAME
  + Activation Function Layer: ReLu
* **ReLu Layer 1:**
  + No filter (only process data, no extract/learn features)
  + Input Shape: 32x32x64
  + Output Shape: 32x32x64
* **Pooling Layer 1:**
  + Pooling method: Max pooling
  + Filter/Kernel/Window size: 2x2
  + Filter/Kernel/Window Shape: [1, 2, 2, 1]
  + Stride: 2
  + Stride Shape: [1, 2, 2, 1]
  + Padding: SAME
  + Input channel: 64 inputs
  + Input shape: 32x32x64
  + Output channel: 64 outputs
  + Output shape: 16x16x64

**Convolution Layer 2:**

* + Convolution Shape: [batch, H, W, depth] = [?, 16, 16, 128]
  + 2D datasize: 16x16
  + Input Shape: 16x16x64 (depth=64 in channel)
  + Output Shape: 16x16x128 (depth=128 out channel)
  + Filter/Kernel/Window size: 3x3
  + Filter Shape: [3, 3, 64, 128]
  + Stride: 1
  + Stride Shape: [1, 1, 1, 1]
  + Padding: SAME
  + Activation Function Layer: ReLu

**ReLu Layer 2:**

* + No filter (only process data, no extract/learn features)
  + Input Shape: 16x16x128
  + Output Shape: 16x16x128

**Pooling Layer 2:**

* + Pooling method: Max pooling
  + Filter/Kernel/Window size: 2x2
  + Filter/Kernel/Window Shape: [1, 2, 2, 1]
  + Stride: 2
  + Stride Shape: [1, 2, 2, 1]
  + Padding: SAME
  + Input channel: 128 inputs
  + Input shape: 16x16x128
  + Output channel: 128 outputs
  + Output shape: 8x8x128
* **Convolution Layer 3:**
  + Convolution Shape: [batch, H, W, depth] = [?, 8, 8, 256]
  + 2D datasize: 8x8
  + Input Shape: 8x8x128 (depth=128 in channel)
  + Output Shape: 8x8x256 (depth=256 out channel)
  + Filter/Kernel/Window size: 3x3
  + Filter Shape: [3, 3, 128, 256]
  + Stride: 1
  + Stride Shape: [1, 1, 1, 1]
  + Padding: SAME
  + Activation Function Layer: ReLu
* **ReLu Layer 3:**
  + No filter (only process data, no extract/learn features)
  + Input Shape: 8x8x256
  + Output Shape: 8x8x256
* **Pooling Layer 3:**
  + Pooling method: Max pooling
  + Filter/Kernel/Window size: 2x2
  + Filter/Kernel/Window Shape: [1, 2, 2, 1]
  + Stride: 2
  + Stride Shape: [1, 2, 2, 1]
  + Padding: SAME
  + Input channel: 256 inputs
  + Input shape: 8x8x256
  + Output channel: 256 outputs
  + Output shape: 4x4x256
* **Convolution Layer 4:**
  + Convolution Shape: [batch, H, W, depth] = [?, 4, 4, 512]
  + 2D datasize: 4x4
  + Input Shape: 4x4x256 (depth=256 in channel)
  + Output Shape: 4x4x512 (depth=512 out channel)
  + Filter/Kernel/Window size: 3x3
  + Filter Shape: [3, 3, 256, 512]
  + Stride: 1
  + Stride Shape: [1, 1, 1, 1]
  + Padding: SAME
  + Activation Function Layer: ReLu
* **ReLu Layer 4:**
  + No filter (only process data, no extract/learn features)
  + Input Shape: 4x4x512
  + Output Shape: 4x4x512
* **Pooling Layer 4:**
  + Pooling method: Max pooling
  + Filter/Kernel/Window size: 2x2
  + Filter/Kernel/Window Shape: [1, 2, 2, 1]
  + Stride: 2
  + Stride Shape: [1, 2, 2, 1]
  + Padding: SAME
  + Input channel: 512 inputs
  + Input shape: 4x4x512
  + Output channel: 512 outputs
  + Output shape: 2x2x512
* **Fully Connected Layer 1 (FC\_1):**
  + FC\_1 shape: [inputs (in\_channels), outputs (output\_channels)] = [2x2x512, 1024]
* **Fully Connected Layer 2 (FC\_2):**
  + FC\_2 shape: [inputs (in\_channels), outputs (output\_channels)] = [1024, 1024]
* **Fully Connected Layer 3 (FC\_3):**
  + FC\_3 shape: [inputs (in\_channels), outputs (output\_channels)] = [1024, 10]

**Report on the results of testing the model:** The model shows a significant increase in the accuracy with redesigned network which is **70.96%.** With this we learn that increasing the number of convolutional layers may increase the accuracy of the model in case of a complex dataset.

* **Redesigned Model Accuracy: 0.7096999883651733**
* **Part IV Model Accuracy: 0.1004**

The below screenshot shows the accuracy attained by the redesigned model.

A screenshot of a computer

Description automatically generated

**Figure:** Accuracy of the redesigned model

**Jupyter Notebook:**  ADTA5550\_Final\_Project\_Part\_VI.ipynb is submitted as part of the assignment for this section.

**PART VII: Project Report (20 Points)**

**TO-DO**

**--) Complete the project report (ADTA5550\_final\_project.docx)**

**Answer:**

**Introduction to the project:** Part IV, V, VI are important parts of the project where we have built, trained and tested CNN on the CIFAR-10 dataset, compared the accuracy of the CIFAR-10 dataset with the MNIST dataset. We also redesigned the CNN to improve the performance of the model for CIFAR-10 dataset.

**Description of Part IV, V, VI:** The below points describe the activities done on the different sections of the project.

* **Part IV: Build, Train, and Test CNN on CIFAR-10 Dataset**
  + In this section we built a convolutional neural network, trained and tested it on CIFAR-10 dataset.
  + CIFAR-10 is a complex dataset with colourful non-centred images which makes it a comparatively complex dataset.
  + The CNN had two convolutional layers, two ReLu layers, two Pooling layers and three Fully Connected layers.
  + This network had an accuracy of 10.04% which is on the lower side.

The below diagram shows the CNN implemented in part IV.

A diagram of a network of data

Description automatically generated with medium confidence

Figure: design of the CNN(Source: Lecture video)

* **Part V: Compare Convolutional Neural Network Performance**
  + In this section we compared the performance of the CNN for CIFAR-10 dataset and MNIST dataset.
  + We have used the similar CNN for both the datasets but there is a significant difference in the accuracy. CIFAR-10 has an accuracy of 10.04% but MNIST has an accuracy of 94.72%.
  + After studying the datasets in details, we found out that the MNIST dataset is a greyscale dataset and have 70,000 hand written centred images of numbers 0-9 which makes it a simpler dataset and the CNN with two convolutional layers could successfully process it and resulted higher accuracy.
  + On the other hand, we observed that, CIFAR-10 dataset has colourful non-centred images of different classes. The CNN with two convolutional layers was not sophisticated enough to capture the details of this dataset resulting in lower accuracy.
* **VI: Improve Convolutional Neural Network Performance**
  + In this section we proposed changes to improve the performance of the CNN. Built, trained and tested the new CNN on the CIFAR-10 dataset.
  + As we understood from part V, that CIFAR-10 dataset needs a sophisticated CNN with deeper network so we built the new CNN with four Convolutional layers, four ReLu layers, four Pooling layers and three FC layers.
  + We also added a dropout layer after first convolutional layer to reduce the overfitting of the model and increased the filter size to capture more details.
  + The new CNN showed a significant increase in the performance and the accuracy increased to 70.96% from 10.04%.

**Conclusions for part IV, V and VI**

**Part IV Conclusions:** The conclusion for part IV where we used the CNN with two convolutional layers for dataset CIFAR can be described as below:

* The CNN is run for 5000 steps and tested at every 100 steps, but it does not provide appropriate performance and records the accuracy of **10.04%**. This is low accuracy to attain from a CNN.
* This CNN needs to be redesigned to attain higher accuracy.

A screenshot of a computer

Description automatically generated

**Figure**: Accuracy of the CNN with two convolution layers (part IV)

**Part V Conclusions:** The conclusion for Part V where we compared the results for the CNN with MNIST dataset and CIFAR-10 dataset can be described as below:

* The MNIST dataset has higher accuracy of **94.72%** with the CNN having two convolutional layers.
* The CIFAR-10 dataset has accuracy of **10.04%** when processed with the similar CNN having two convolutional layers.
* The datasets have different complexity. MNIST dataset is a simpler dataset with only greyscale, centred images. CIFAR-10 dataset has colourful non-centred images which makes it a complex dataset to process.
* The CIFAR-10 dataset needs a deeper CNN which can capture all the details of this complex dataset.

A white background with black text

Description automatically generatedA white background with black text

Description automatically generated

**Figure:** Accuracy of CIFAR-10 and MNIST datasets

**Part VI Conclusions:** The conclusion for part VI where we worked to improve the performance of the CNN ca be described as below:

* We added more convolutional layers to the CNN to make it deeper which enabled it to capture more details from the dataset.
* The new CNN has four convolutional layers and a dropout layer after the first convolutional layer to reduce overfitting.
* The new CNN improved the accuracy of from **10.04%** to **70.96%**.
  + **Redesigned Model Accuracy: 0.7096999883651733**
  + **Part IV Model Accuracy: 0.1004**
* From this result we learn that, for a complex dataset, we need a deeper network with more convolutional layers to attain a higher performance/ accuracy.

A screenshot of a computer

Description automatically generated

**Figure:** Accuracy of the redesigned CNN for improving performance

**Conclusion:**

* From the above findings, we can state that a simple dataset can have a high accuracy when processed with a simple CNN having one or two convolutional layers.
* On the other hand, a complex dataset may not provide appropriate performance for a simple CNN. We need a deep CNN with multiple convolutional layers to attain appropriate accuracy.
* Analysing a dataset prior to constructing a CNN allows a better understanding of its complexity and helps to design and build a CNN for higher performance.

**Note:** The following documents are submitted as part of this assignment

* ACC\_cnn\_cifar\_10.docx
* ACC\_cnn\_mnist.docx
* ADTA5550\_Final\_Project\_Part\_IV.ipynb
* ADTA5550\_Final\_Project\_Part\_VI.ipynb
* ADTA5550\_final\_project.docx