**Final Project-5550**

**Submitted by**

**Sai Kumar Pathipati - 11600326**

**PART I- Use TensorFlow Directly in coding:**

Yes, a student may decide to create, train, and test a CNN for the homework project using TensorFlow directly rather than Keras. A student may want to utilize TensorFlow directly for a variety of reasons, including the following:

**Flexibility:** TensorFlow offers a low-level API that enables users to alter each step of the model design and training procedure. This gives students more control over their models and gives them the freedom to test out various methods and approaches.

**Performance:** Compared to utilizing a high-level API like Keras, TensorFlow offers a large variety of optimization choices for neural network training, which can boost model performance.

**Industry-standard:** TensorFlow is a widely used framework in business and research, therefore having knowledge of it might be helpful while looking for work or applying for graduate school.

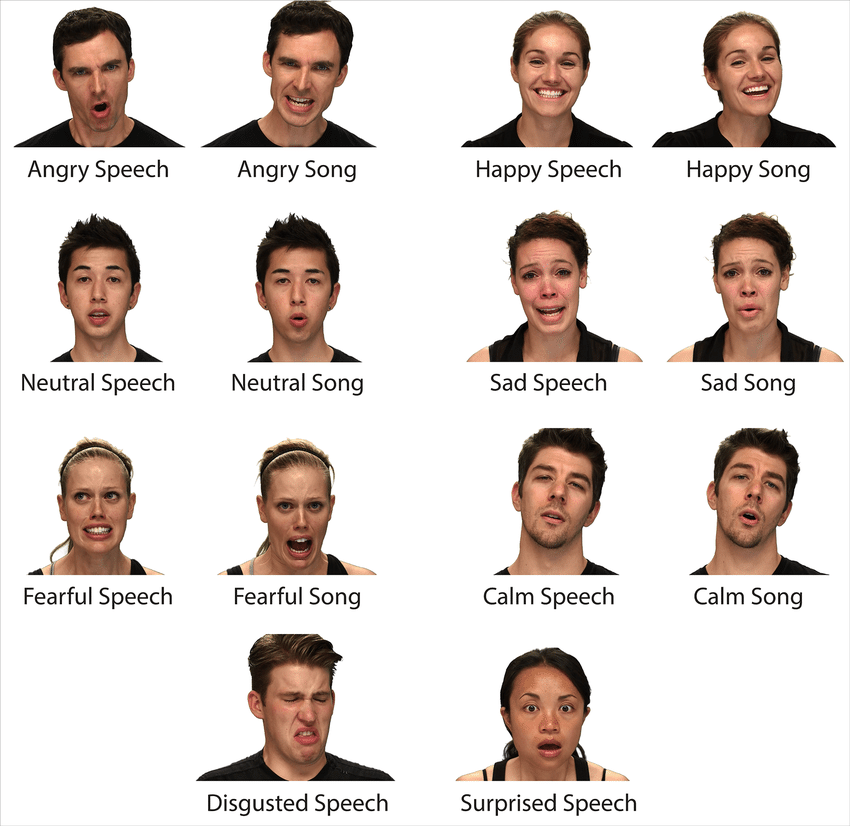
**Compatibility with other frameworks**: TensorFlow is a fantastic option for projects that call for interoperability with other platforms since it is compatible with other deep learning frameworks and tools and can be quickly integrated with them.

The use of TensorFlow directly, as opposed to a high-level API like Keras, may necessitate more coding and a steeper learning curve. The directions and criteria of the homework assignment should be carefully read by the students, who should then decide which tools and frameworks are best for their particular job and objectives.

**PART II - A Dataset of Images or Audio Files:**

**Report:** Raw data from the RAVDESS consisting only of voice sounds (16 bit, 48 kHz.wav).

These recordings are part of the RAVDESS dataset I downloaded from Kaggle. The full list of the 1440 files contained in this subset of the RAVDESS is as follows: There are a total of 1440 possible combinations, thanks to the 60 tests given to each of the 24 performers. There are a total of 24 professional actors on the RAVDESS, 12 women and 12 men. Each actor reads two phrases with a lexically matched North American neutral accent. Peace, joy, sorrow, wrath, fear, surprise, and even contempt is just few of the feelings that can be expressed verbally. Each sentence can be found in three different forms: regular, powerful, and neutral.



**Data Set Source:**

<https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio?resource=download>

**File naming convention:**The 1,440 files all have unique names according to the naming method. The filename is a seven-part number identifier (for example, 03-01-06-01-02-01-12.wav). These identifiers will detail the following aspects of the stimuli:

**Filename identifiers:**

Modality (where 0 indicates no AV, 1 indicates full AV, 2 indicates video only, and 3 indicates audio only).

Sound transmitted through the vocal cords (channel 01 for speaking, channel 02 for music).

Feeling (0 for peaceful, 1 for quiet, 2 for happy, 3 for excited, 4 for sad, 5 for angry, 6 for scared, 7 for repulsed, and 8 for shocked).

Degree of emotion (with 1 representing normal and 2 extreme). It's important to keep in mind that the 'neutral' emotion isn't extremely strong.

Children are talking by the door (Statement 01) and dogs are lounging there (Statement 02).

Iterate (starting with 01 for the first repetition and moving on to 02 for the second).

Performer who worked from 001 to 1234. Male performers are represented by odd numbers, whereas female actors and actresses are represented by even numbers.

The date 03-01-03-06-01-02-01-12 is an example. One type of file is called a "wav."

Fearful (06), but not overly so (01), speaking (03), first repetition (01), and the statement "dogs" (02) all indicate an audio-only format.

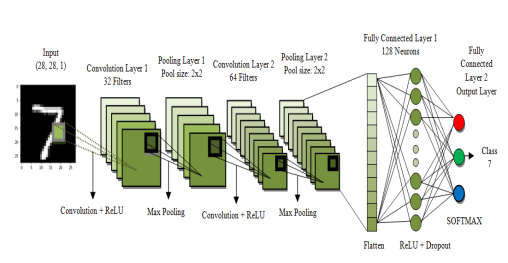
**PART III - Obtain CIFAR-10 Dataset:**

**Report:** We have been given the task of obtaining the CIFAR-10 dataset and then transferring it from our local machine to the remote instance in order to answer the question posed. To begin, we need to get the dataset down from the canvas. After getting the remote GCP virtual server up and running, we can begin an SSH session by clicking the chevron next to SSH and then selecting to open a new tab in our browser. Next, we start a GCP SSH session in our home directory and make sure we're in compliance by making a new subfolder in the JP NTBK called "CIFAR 10\_DATA" with the command mkdir CIFAR\_10\_DATA. This is necessary because we need to access the CIFAR 10\_DATA folder from within the JP NTBK. Then, we select "upload file" from the submenu, navigate to the files we want to transfer, and finally, click "Open." To continue, navigate to the location where you'd want to store the data and perform the command cp data file JP NTBK/ CIFAR 10 DATA (the command used to copy the data files and keep them in the folder). All required files have been backed up to the specified directories.

Graphical user interface, application

Description automatically generated

**PART IV - Design and diagram of the model:**



**(Convolutional Neural Network)**

The CNN's first convolutional layer consists of the following components: batch, width, height, and depth = (1, 28, 32).

Measurement of 2D information: 28 by 28

A 28x28x3 channel is the input shape, and its depth is 3.

28 bytes in width and 32 bytes in depth (depth equals 32 out-channels) is the output shape.

Size of filter/kernel/window = 5x5 Weight form for a filter is [5, 5, 3, 32].

The stroke pattern is: (1, 1, 1, 1) where 1 represents a stroke.

It's the Same for Activation and Cushioning

Function layer is represented by ReLU.

**Layer 1 of Relu:**

Layer 1 of Relu does nothing more than handle data; no filters are utilized, and no features are extracted or learned.

28 x 28 x 32 (form feed)

28x28x32 is the output shape.

Pooling Maximum pooling (Max pooling) is used at Layer 1.

Window/Filter/Kernel size: 2 by 2

Size of the filter, kernel, and window: [1, 2, 2, 1]

Repeating stroke pattern: [1, 2, 2, 1]

Identical Padding

There are 32 inputs available.

The input format is a 32 by 32 by 28 grid.

There are 32 separate outputs.

Form of Product: 14 by 14 by 32

Convolution Convolution in Layer 2 Layer 2 was given a convolutional form with the following settings: (1, 14, 64) (1, 14) (1, 14) (batch, H, W, depth)

Size of 2D Information: 14 by 14

The input is a 1 by 1 grid with a size of 14 by 14 by 32.

Filter/Kernel/Window size = 5 x 5 [5, 5, 32, 64] is the filter shape, also known as the weight shape, whereas the output form is 14 by 14 by 64 with a depth of 32 out channels.

One stroke equals one of the following patterns: [1, 1, 1, 1]

What We Mean When We Talk About "Padding" or "Activation" Function layer is represented by ReLU.

The Relu node of the convolutional layer has been reached. The input shape has dimensions of 14 by 14 by 64.

Resulting 14-by-14-by-64-grid

**Level 2 Pooling:** **Method of pooling: Method of pooling: Maximum sharing**

Size of Filter, Kernel, and Window = 2 x 2

Size of the filter, kernel, and window: [1, 2, 2, 1]

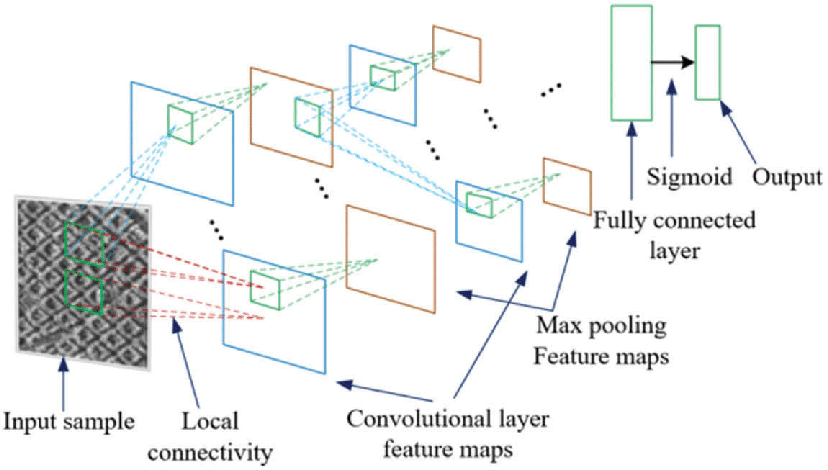
The stroke's rhythm is [1, 2, 2, 1], with a stride length of 2.

The depth input channels consist of 64 padding - SAME inputs.

Size requirements: 14x14x64 in. There are 64 channels' worth of outputs.

The final product will have the dimensions of 8x8x64 and will have fully merged layers.

The above steps are to be followed to design the convolutional neural network for RAVDESS dataset.

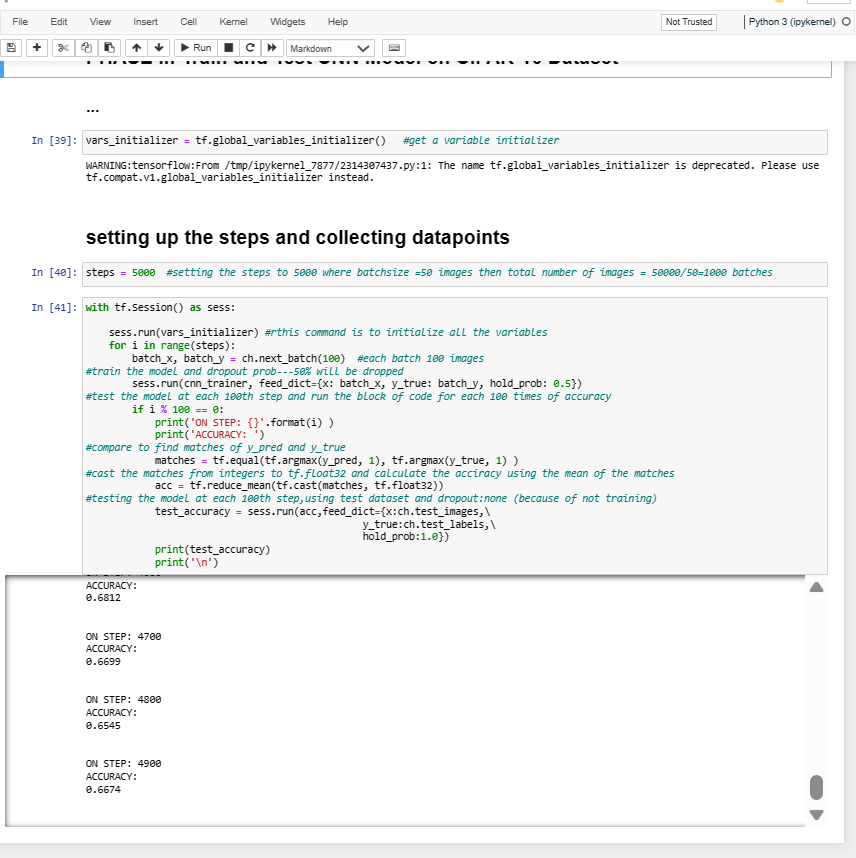


**The Cifar-10 Dataset:**

**Report:** I began by making the necessary folders and then importing the data into the virtual machine. Once I knew where I was going, I unpicked the data in my dataset and began working with it. Seven metadata files, five training batches, and one test batch are the final tally. I proceeded after declaring the required helper functions. As the initial step in developing a CNN, providing placeholders for the untransformed incoming data is a good place to start. First, a fully connected layer is created, followed by a convolution layer, and finally a layer that combines relu and pooling layers for a second convolution. After the dropout layer is established, the last completely linked layer is constructed, and outputs are produced.

In the second phase, we train and test the model, which entails establishing steps, collecting data points, testing the model at the hundredth step, running the block of code 100 times to determine accuracy, comparing to identify matches between y false and y true, converting the matches from integers to tf. float32 to determine accuracy, and so on.

**Provide a comprehensive report of your findings:**  Our analysis leads us to believe that the classification will be about right. Not as bad as either of those separately, but not as nice as both. After inputting steps=5000, running the algorithm, and gathering 50 data points at the 100th step, we found that the two convolution layers and the two pooling layers gave an accuracy of 66.74% at step 5000.

****

**PART V - Compare Convolutional Neural Network Performance:**

The information given demonstrates the accuracy of two separate CNN models (trained on the MNIST and CIFAR-10 datasets).

**Observations drawn from the data include:**

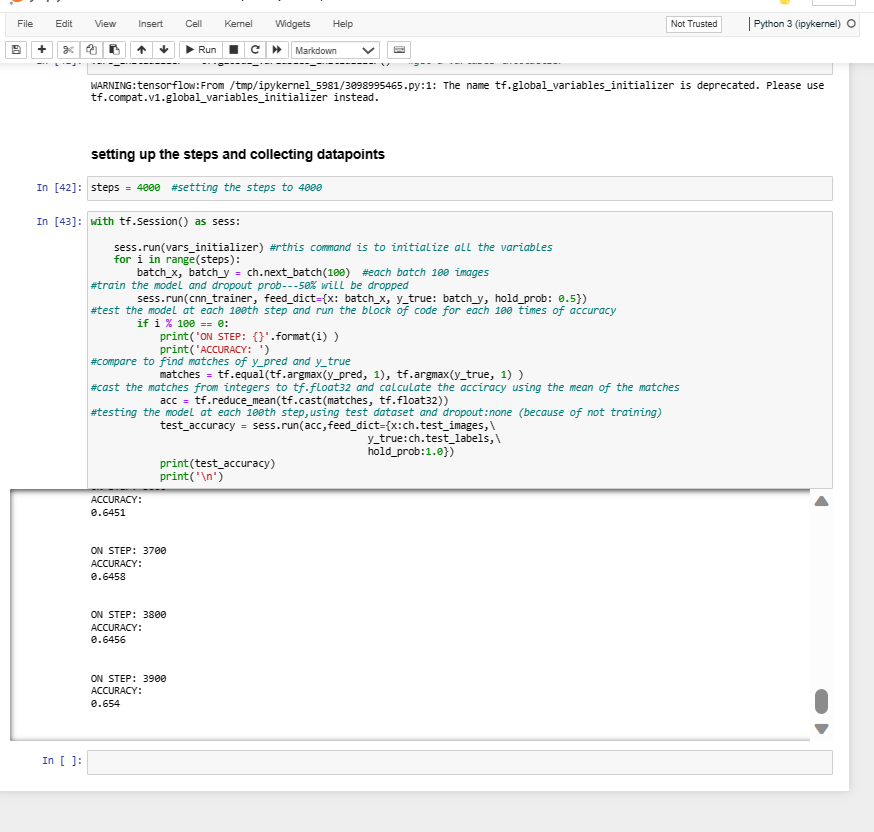
* Both models exhibit gradual improvement as the number of training steps rises, starting out with poor accuracy.
* Throughout the training phase, the MNIST model outperforms the CIFAR-10 model in terms of accuracy.
* After 3000 training steps, the MNIST model reaches an accuracy of 98.95%, while the CIFAR-10 model does so after 3100 training steps.
* The rate of accuracy improvement varies depending on the training step interval. For instance, between steps 100 and 500 of the CIFAR-10 model, accuracy significantly improves, but after that, the pace of improvement declines.
* After a specific number of training steps, both models reach high accuracy and might not need any more training. After 2000 training steps, the MNIST model reaches accuracy of 99.07%, while the CIFAR-10 model does so after 3500 training steps and accuracy of 67.08%.
* In conclusion, the data demonstrates that a CNN model's accuracy grows with the number of training steps, although the pace of progress and the highest accuracy that can be achieved might vary depending on the dataset and the model's design.

The following are examples of strategies for optimizing hyper-parameters to boost performance:

* **Identifying the Optimal Rate of Learning (L):** Starting with a high LR and decreasing it over time is recommended. The Quantity of Goods in a Batch Comprising: The amount of RAM in a GPU is a good indicator of the maximum batch size at which it can operate, so typically you'll want to go with that. The next step is to increase the model's height and width so that it can handle more weight. This is represented by the number of filters in each convolution layer.
* **Methods for maximizing output by restructuring data structures:** To improve the image quality, we can raise the image size from its present 32323 pixels to at least 64643 pixels, a process called as progressive enlargement. Then Randomly switching pictures: Rotating the camera in a different direction will change the shot's perspective. A random reordering of the images: The strategy works best when applied to topics that are off-center.
* **Optimization strategies for better model performance:** A subset of the data is used, and certain samples from heavily sampled data classes are ignored, in order to fine-tune the model. Classification weights: Class weights assign the same importance to each class during training regardless of whether the database is balanced or biased. Modifying the model based on data from the trains: It is feasible to retrain a model after it has been used to make predictions by applying new training data to fix wrongly predicted images.

**PART VI: Improve Convolutional Neural Network Performance:** Improvements to the CNN's performance on the cifar-10 dataset are suggested. In this case, I've added a brand-new third convolutional layer with the parameters: The number of steps was decreased from 5000 to 4000 by using a ReLu activation and pooling layer with a filter/kernel/window size of 5 x 5, filter shape = weight shape: [5, 5, 64, 128].

**Extensive analysis of the test results:** Our analysis leads us to believe that the classification will be about right. Not as bad as either of those separately, but not as nice as both. Incorporating an extra convolution layer and decreasing the step size from 5000 to 4000 improved accuracy from 64.2% to 65.4%. The model did not much improve when a number of crucial improvements were made, including the addition of a layer and an increase in the step size (both of which have a small effect)

****

**PART VII: Project Report**

**Introduction:**

The project overview is divided into six sections, including this one. Finding a data set that contains media files of any kind is the first order of business. The audio files I utilized were sourced from RAVDESS (the Ryerson Audio-Visual Database of Emotional Speech and Song), which can be found on the platform Kaggle. Second, we'll get our hands on the CIFAR-10 dataset and transfer its seven individual files to a distant virtual machine. Finally, we focus on developing, training, and evaluating a CNN model using the CIFAR-10 dataset in Section III. The MNIST dataset and the CIFAR-10 dataset are compared to one another in the fourth part. Here in the fifth and last section, we'll talk about several ways that the model we've built up to this point could use some tweaking.

The student's contributions to the project should be detailed in Sections III, IV, and V. As I was mentioning previously, we have access to the CIFAR-10 dataset.

Images having a resolution of 32 by 32 by 3 pixels are used in this data set. Various dictionaries are updated with the results of combing through these images. The column containing the image data will be referred to as "data," and the corresponding variable will be denoted by "X" for the sake of brevity and clarity. After it has been scaled and reshaped, it is ready to be fed into the network's first convolution layer. One-hot coding is crucial for streamlining the data so that the network can understand it because each class is represented by an integer. The first stage of the project, model construction, can now get begin with the availability of the image data. To reach this goal, supplementary capabilities must be made available. We need to come up with a function to represent the relative importance of each filter. It is the job of two functions—initialize weights and initialize bias—to consider the filter's overall form and the bias individually. In response to filter shape and bias shape parameters, return the contents of tf.variables. The convolution layer has now been defined to accept picture data and filter forms as input parameters. The preceding function is called here to calculate the weights. The desired result of keeping the same size can be achieved by setting the stroke to 1 and the padding to 1. In the second stage, the model is trained and tested by first establishing a sequence of steps, then gathering data points, and finally evaluating the model at the 100th step by iteratively running a block of code 100 times to evaluate its correctness. We look for a correlation between y pred and y true, and if we discover one, we transform the associated values into tf.float32 format. The values of y predicted and y true must coincide. Following the stated guidelines, we subjected our model to a total of 100 iterations on the test dataset. We ensured that no data was lost due to insufficient training and evaluated the final fifty data points from the cifar-10 dataset for precision. Comparable to Chapters 3 and 5 Adding a new convolution layer and decreasing the total number of steps resulted in a vastly superior final output. In this series' last installment, we compared data from MNIST and CIFAR to determine whether there was any discrepancy and we proposed some potential ways for improving the model's accuracy.

Document all the student has learned about CNN model testing from doing PART III, PART IV, and PART V of the Final Project.

It is suggested that while attempting to label images, you employ a convolutional neural network with a ReLU activation layer. Multi-class classification uses loss functions other than the standard binary-cross entropy, such as SoftMax. Models are trained with one output layer, two fully linked layers, and two convolutional layers in the early stages. It is anticipated that 68% of the time, it will be correct. The model's hyperparameters are fine-tuned through trial and error by adding three additional convolution layers, adjusting the step size, playing about with the FC layer's values, and so on. No matter what was changed in the original model, the modified ones generated nearly identical results. The final percentage after making all the necessary adjustments was close to 67%. The final model enhancement was a three-layer conv that was 4,000 steps in size. A maximum of 66% accuracy was found during the testing process.

Researchers found that the input picture resolution had no effect on the test's accuracy whether a 2- or 3-layer convolution layered network was used. Consequently, methods like L2 regularization are required to fully appreciate the significance of network effects. The majority of the findings (68% overall) were correct, thus fixing this will require making changes to the code. The model is probably not perfect, but it's also not horrible. The potential exists to construct more accurate models.

**PART VIII: Final Presentation Videos: YouTube Links**

* **Final\_project\_presentation\_video\_Sai Kumar Pathipati\_1.mp4**
* [**https://youtu.be/pMLO29lMw9Y**](https://youtu.be/pMLO29lMw9Y)
* **Final\_project\_presentation\_video\_Sai Kumar Pathipati\_2.mp4**
* [**https://youtu.be/UfVz3Tv\_oVk**](https://youtu.be/UfVz3Tv_oVk)