**ADTA 5550 Final Project**

**Aman Gill**

**11489675**

**Part I**

**Q1.1 - Is the student required to use TensorFlow directly in coding (build, train, and test  
CNN) in this homework assignment?** Yes, as it is explicitly stated in the assignment instructions as followed; “While working on the final project, it is required that the student should use the TensorFlow (Version 1.xx) directly as the AI framework to build the neural network”

**Q2.2 - Should the student use Keras in coding (build, train, and test CNN) in this**

**homework assignment?** No, as it is also explicitly stated in the assignment instructions as follows; “In other words, the student should not use Keras APIs, e.g., Keras sequential API, to build the neural network”

**Part II**

**Dataset of choice link (Google Speech Commands):** [**https://huggingface.co/datasets/google/speech\_commands**](https://huggingface.co/datasets/google/speech_commands)

This Google Speech Commands dataset is an amalgamation of what is essentially spoken word audio samples geared towards training and evaluating machine learning models in regard to speech recognition. Within the Dataset are short audio clips entailing content such as individuals speaking simply with commands like yes, no, stop, go etc. as well as numbers. The application of this is for deep learning capabilities in voice activated systems.

There are a total of 105,829 audio recordings and these recordings are conducted by a variety of speakers, 2,618 to be exact. This of course, encompasses a diverse range of accents and variations in speech. The dataset is structured in a way where each sample is a one second WAV file recorded at 16 kHz, and each file is labeled with the spoken word. In terms of Metadata, the speaker ID, sample rate and other audio properties are included. To further elaborate, some of the audio samples contain noise purposed for model training in real scenario environments. A screenshot of a computer code

AI-generated content may be incorrect.

A JSON example is shown above for a core word example, for the purpose of exmplifying datset structure and organization. following the File statement, the audio statement consists of the path that details the location of the WAV file, the array that details the raw waveform values reperesenting audio signals in numerical form, and the the sampling rate which is set at 16,000 in the standard frequency unit of Hertz. The label statement details the numerical identifier for the word no, and the is\_unknown statement details the boolean value showing whether the word is part of the primary command set. Finally the speaker\_id statement details the indentifier value and the utterance\_id statement is integrated to distinguish recordings from a particular speaker. This structured format ensures that each sample is labeled with explicitness in mind and categorized as well in order to facilitate machine learning processing and learning of the dataset. Furthermore, multiple spekers and background variations enables the opportunity of generalizability for the speech recognition models.

The model is overall well compromised and elaborated upon and presents solid opportunities In deep learning endeavors with voice. Real world applications may include Speech recognition efforts in training voice controlled applications, spotting keywords, or simply research in improving pre existing machine learning models in order to classify speech.

**Part III**

**Breakdown Elaboration –** I began with going to my VM instance and running the instance, and then opening a SSH browser window. To of course preceed, this I dowloaded the dataset from Canvas. (See More Below)

**A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.**

In the final steps, using the ls – l command to verify things as I went, I copied the dataset into my subfolder under my Jupyter notebook folder and I made sure to also extract the file

As shown, I used the necessary commands to create the subfolder under my Jupyter notebook folder, as well as of course, already uploaded the dataset

**Part IV**

The CNN architecture was structured to extract meaningful features from the input images through a series of convolutional and pooling layers before making final predictions using fully connected layers. The model starts with an input layer that processes 32x32 RGB images and this is followed by the first convolutional layer, which applies 32 filters of size 3x3 with the ReLU activation function. Finally, the extracted features are then reduced using a max pooling layer with a 2x2 filter and a stride of 2.

Following suit, a second convolutional layer further refined the feature extraction using 64 filters, followed by another max pooling operation. The output is flattened into a one dimensional vector to prepare it for the fully connected dense layer, which consists of 128 neurons with the ReLU activation function. Finally, the output layer consists of 10 neurons in correspondence to the 10 CIFAR classes and applies the softmax activation function to generate class probabilities.

In terms of training and performance analysis, the model was trained using the Adam optimizer with a learning rate of 0.001 and the network was trained for 5,000 epochs with evaluations occurring every 100 epochs in order to monitor the test accuracy. Initially the model boasted rapid improvement in accuracy, starting at approximately 33% after 100 epochs. As training went on, the accuracy continued to increase, stabilizing around 63.27% by epoch 5000. While this performance can be deemed as reasonable for a basic CNN on Cifar, further enhancements such as data augmentation or batch normalization could potentially improve the results.

All in all, the work done successfully implemented a convolutional neural network crafted to pertain to the Cifar dataset and demonstrated its capacity to classify images with a final test accuracy of 63.27%. Suggestions of improvement like fine tuning hyperparameters, deeper architectures, and dropout regularization to prevent overfitting as well as the employment of pre trained networks could pose a large improvement in the performance. The image below details the final results and findings.

A screenshot of a computer

AI-generated content may be incorrect.

**PART V**

The CNN trained on the MNIST dataset had a final accuracy of 98% after 5,000 training epochs, while the CNN trained on CIFAR-10 reached a final accuracy of 63.27%, which is significantly lower. Throughout training, accuracy was recorded at 50 checkpoints with intervals of every 100 epochs.

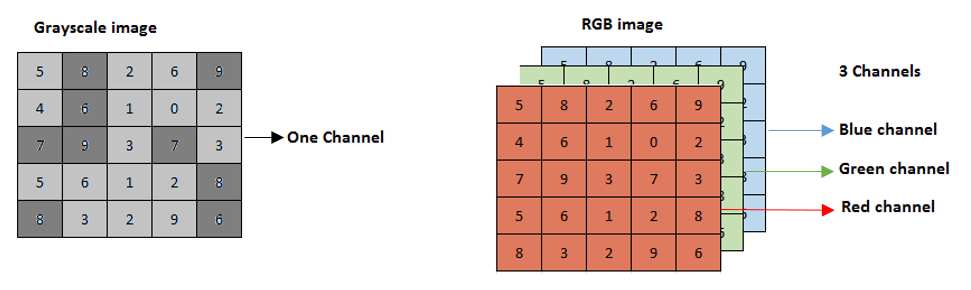
One reason that sharply stands out for this performance gap is the complexity of the datasets themselves as MNIST images are simple, consisting of 28x28 grayscale pixels with well-defined shapes, making it easier for the CNN to identify patterns. On the contrary, CIFAR images are more detailed, with 32x32 pixel RGB values representing realistic objects such as animals, vehicles, and household items. The additional complex nature in object shapes, textures, and backgrounds etc. makes feature extraction much more more challenging for the CNN.

In addition, another contributing factor is the difference in feature distinction since the digits in MNIST have clearly displayed forms. The Cifar categories however share in similar visual characteristics. As an example, differentiating between a cat and a dog necessitates deeper feature learning than distinguishing between a handwritten pair of numbers such as 3 or 8. Since the same CNN architecture was used for both datasets, it may not have been the best option for handling the greater variation present in Cifar.

As further reinforcement of aforementioned, color channels are also to be considered as MNIST images are grayscale, so the model only needs to analyze intensity variations. The Cifar images have the color channels of red green and blue so consequently there are more data points for the model to process thus requiring the CNN to extract meaningful features across multiple dimensions and boosting the difficulty curve.

The amount of training data available is another factor to consider. While both datasets contain 50,000 training images, Cifar exhibits much higher variation between classes, making it more demanding in terms of learning capacity. With a simple CNN, the model might not be extracting the most useful representations, leading to lower performance. Data augmentation techniques, such as flipping, rotating, or color shifting images, could help improve the Cifar model’s accuracy by providing more diverse training examples. Optimization techniques can also influence accuracy, because while both models employed Adam Optimizer, the Cifar model might benefit more from tuning the hyperparameters like the learning rate or introducing weight decay and dropout to enhance performance. These modifications could prevent overfitting and improve accuracy over more training iterations.

In summary, the comparison between the two CNNs serves to highlight the importance of dataset complexity when training deep learning models. The MNIST model achieved nearly perfect accuracy due to its simpler structure, whereas the CIFAR-10 model struggled to surpass 63%. This outcome demonstrates that the same CNN architecture does not necessarily generalize well across different datasets. If one was to Improve the Cifar performance, it would require deeper architectures, as well as more extensive data preprocessing and higher regularization techniques.



**A screenshot of a computer

AI-generated content may be incorrect.Part VI**

Throughout the updated model training, the loss gradually decreased while the test accuracy steadily improved, At various checkpoints, the accuracy was monitored to track progress. Around epoch 3300, the model reached a test accuracy of approximately 62.15%, which continued to increase as training progressed. By epoch 4000, the test accuracy had improved to 63.77%, and further renovations resulted in a final test accuracy of 64.65% at the completion of training. These results overall confirm that the adjustments made to the network architecture, including additional convolutional layers and optimized hyperparameters, had a measurable positive impact on classification performance.

The improvements shown could be described as minimal but they do reveal the effectiveness of tuning CNN models for better feature extraction and generalization. The updated model outperformed the previous iteration by achieving a higher accuracy, which can show that deeper architectures and enhanced regularization techniques can behold improvements when working with complex image datasets.

**PART VII**

The objective of this project was to develop, train, and evaluate convolutional neural networks using the Cifar dataset. The project was divided into multiple phases, each focusing on different aspects of CNN implementation. Initially, a baseline CNN model was built and trained to establish a performance benchmark. This was followed by a comparative analysis of different CNN architectures to understand variations in accuracy. Finally, improvements were proposed and implemented to optimize the network’s performance. The insights gained from these experiments contribute to a broader understanding of how CNNs process image data and what factors influence their classification accuracy.

Throughout the project, several key steps were executed to analyze and improve CNN performance. In this phase, a CNN model was designed using TensorFlow and trained on the Cifar dataset. The architecture consisted of convolutional, pooling, and dense layers to process and classify images into ten categories. The model was trained for 5000 epochs, with accuracy monitored at regular intervals. The final test accuracy reached approximately 63.14%, establishing a performance baseline for subsequent improvements. A comparative study was conducted to evaluate the CNN’s performance on the Cufar dataset in relation to its previous application on the MNIST dataset. The Cifar dataset, being more complex and containing colored images, posed a greater challenge compared to the grayscale MNIST dataset. As expected, the test accuracy for Cifar was lower than what had been achieved with MNIST. This performance gap was analyzed, highlighting the impact of dataset complexity, network depth, and feature extraction on classification accuracy. Based on the insights from the comparative analysis, modifications were proposed to enhance CNN accuracy. The revised architecture incorporated additional convolutional layers and an optimized learning rate to improve feature extraction and training stability. The updated model was trained for another 5000 epochs, leading to an improved final test accuracy of 64.65%. While the gains were incremental, they demonstrated the effectiveness of deeper networks in learning complex image representations.

The project provided several valuable insights into CNN behavior. First, dataset complexity plays a significant role in network performance, as seen in the accuracy differences between MNIST and Cifar Second, increasing network depth and adjusting hyperparameters can enhance accuracy, though improvements may be gradual. Finally, training CNNs requires careful balancing between model complexity and computational efficiency, as deeper architectures demand more training time and resources.