University of North Texas

ADTA 5550: Deep Learning

Final Project

Biniam Abebe

Contents

[1. Introduction 3](#_Toc172466540)

[2. Part I: Use of TensorFlow 3](#_Toc172466541)

[3. Part II: Dataset Selection 4](#_Toc172466542)

[4. Part III: CIFAR-10 Dataset Acquisition 4](#_Toc172466543)

[5. Part IV: CNN Implementation on CIFAR-10 5](#_Toc172466544)

[6. Part V: Performance Comparison 7](#_Toc172466545)

[7. Part VI: CNN Performance Improvement 8](#_Toc172466546)

[8. Conclusions 10](#_Toc172466547)

[9. References 11](#_Toc172466548)

**1. Introduction**

This project aims to apply the concepts learned in Deep Learning to practical tasks using Convolutional Neural Networks (CNNs). The project involves using TensorFlow to build, train, and test CNNs on the CIFAR-10 dataset, compare performance with previous results from the MNIST dataset and explore ways to improve model performance. This report is structured to cover each part of the project in detail, with explanations, code snippets, and results.

**2. Part I: Use of TensorFlow**

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

*Answer: Yes, I must use TensorFlow directly in coding to build, train, and test the CNN in this final project*.

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

*Answer: No, I should not use Keras in coding for this final project. The instructions explicitly state that TensorFlow (Version 1.xx) should be used directly as the AI framework to build the neural network, without using Keras APIs.*

**3. Part II: Dataset Selection**

For this project, I have selected the UrbanSound8K public domain dataset for potential use in deep learning research. This dataset contains audio files and is appropriately labeled for use in machine-learning tasks.

Key Information about the Dataset:

1. Name: UrbanSound8K
2. Official Website: [https://urbansounddataset.weebly.com/urbansound8k.html
3. Download Link: https://serv.cusp.nyu.edu/projects/urbansounddataset/urbansound8k.html
4. Dataset Size: 8732 labeled sound excerpts
5. Data Structure:
6. Organized into ten folds to support 10-fold cross-validation.
7. Audio files are stored in .wav format.
8. Each file is labeled with the sound class.
9. Classes/Categories: Air conditioner, Car horn, Children playing, Dog bark, Drilling, Engine idling, Gun shot, Jackhammer, Siren, Street music.
10. Data Split: Divided into training and test sets for cross-validation.
11. Additional Features: The dataset includes metadata files about each sound excerpt, such as start time, end time, class label, and fold assignment.

**Brief Description:**

The UrbanSound8K dataset represents a variety of urban sounds, making it suitable for tasks like environmental sound classification. Its diverse range of sounds and well-labeled data make it ideal for developing and testing audio recognition models.

**The rationale for Selection:**

I chose this dataset due to its relevance to deep learning for audio recognition. It provides a challenging yet manageable dataset for experimenting with neural networks. Unlike well-known datasets like MNIST or CIFAR-10, UrbanSound8K offers unique insights into audio data classification. This dataset provides an excellent opportunity for deep learning research due to its diverse sound classes and comprehensive labeling. Its structure supports robust evaluation through cross-validation, making it particularly suitable for audio recognition tasks.

**4. Part III: CIFAR-10 Dataset Acquisition**

For this project, I have successfully obtained the CIFAR-10 dataset as per the instructions. Here's a report on the process:

4.1 Downloading from Canvas

I accessed the Canvas module: .../DATA\_SETS and downloaded all seven available dataset files. These files constitute the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class.

Files downloaded:

1. data\_batch\_1

2. data\_batch\_2

3. data\_batch\_3

4. data\_batch\_4

5. data\_batch\_5

6. test\_batch

7. batches.meta

4.2 Transferring to Remote Virtual Machine

After downloading, I transferred all the data files to the remote virtual machine using the following steps:

1. Accessed the remote virtual machine in Google Cloud Platform (GCP) using SSH.

2. Navigate to the sub-folder JP\_NTBK in the remote VM.

3. Created a new sub-folder under ~/JP\_NTBK named "CIFAR\_10\_DATA".

4. Uploaded all seven data files from my local computer to the newly-created "CIFAR\_10\_DATA" sub-folder in the remote instance.

To upload the files, I used the file upload feature in the GCP SSH interface, following the steps outlined in the document "HOWTO\_upload\_files\_to\_remote\_server\_using\_GCP\_SSH.pdf" provided in the Canvas module: .../SW\_DOCS.

Verification:

To ensure successful transfer, I used the following command in the SSH terminal:

*ls -l ~/JP\_NTBK/CIFAR\_10\_DATA*

This command listed all the transferred files, confirming that all seven CIFAR-10 dataset files were successfully uploaded to the correct location on the remote server.

With the CIFAR-10 dataset now available on the remote virtual machine, I am ready to proceed with the project's next steps, including building, training, and testing the Convolutional Neural Network (CNN) using this dataset.

.

5. Part IV: CNN Implementation on CIFAR-10

In this section, I will detail the process of designing, building, training, and testing a Convolutional Neural Network (CNN) using TensorFlow on the CIFAR-10 dataset.

5.1 Network Design

Here's the architecture of the CNN designed for this project:

[Insert a diagram of the network architecture here]

Layer-wise details:

1. Input Layer: 32x32x3 (RGB image)

2. Convolutional Layer 1: [number of filters, filter size, stride, activation function]

3. Pooling Layer 1: [type of pooling, pool size, stride]

4. Convolutional Layer 2: [number of filters, filter size, stride, activation function]

5. Pooling Layer 2: [type of pooling, pool size, stride]

6. Flatten Layer

7. Fully Connected Layer 1: [number of neurons, activation function]

8. Dropout Layer: [dropout rate]

9. Fully Connected Layer 2 (Output): 10 neurons (for 10 classes), softmax activation

5.2 Implementation Details

The CNN was implemented using TensorFlow 1.xx, as required. Key points of the implementation include:

- Direct use of TensorFlow, without Keras APIs

- Use of "STEP" instead of "EPOCH" for training

- Training for 5000 steps

- Testing performed after every 100 steps of training

5.3 Training and Testing Results

The model was trained for 5000 steps, with testing performed every 100 steps. Here are the key results:

[Insert a table or graph showing accuracy and loss over training steps]

Final Test Accuracy: [X.XX%]

5.4 Analysis of Results

[Provide a brief analysis of the training and testing results. Discuss any patterns observed in the accuracy and loss over time, and comment on the final performance of the model.]

5.5 Code Implementation

The complete code for building, training, and testing this CNN model can be found in the accompanying Jupyter Notebook: [Notebook\_Name.ipynb]

Key sections of the code include:

- Data loading and preprocessing

- Model architecture definition

- Training loop implementation

- Testing and evaluation

[You may include a few key code snippets here if desired]

6. Part V: Performance Comparison

In this section, we will compare the performance of two Convolutional Neural Networks (CNNs): one trained on the MNIST dataset from HW4, and the other trained on the CIFAR-10 dataset in this project.

6.1 Data Collection

MNIST Dataset (HW4):

Here are the 50 data points of accuracy levels produced by the CNN with the MNIST dataset:

[Insert table or list of 50 accuracy data points for MNIST]

CIFAR-10 Dataset (Current Project):

Here are the 50 data points of accuracy levels produced by the CNN with the CIFAR-10 dataset:

[Insert table or list of 50 accuracy data points for CIFAR-10]

6.2 Comparison of Results

To visually compare the performance, I've plotted the accuracy levels for both datasets:

[Insert a line graph showing accuracy over steps for both MNIST and CIFAR-10 on the same plot]

Key observations:

1. [Observation about initial accuracy levels]

2. [Observation about rate of improvement]

3. [Observation about final accuracy levels]

4. [Any other notable patterns or differences]

6.3 Analysis of Performance Gap

Based on the results, we can observe that [state which dataset achieved higher accuracy]. The reasons for this performance gap likely include:

1. Dataset Complexity:

[Explain how the complexity of CIFAR-10 differs from MNIST, e.g., color vs. grayscale, object types]

2. Image Resolution:

[Discuss the impact of image size and detail on model performance]

3. Number of Classes:

[Compare the number of classes in each dataset and how this affects the task difficulty]

4. Model Architecture:

[Discuss any differences in the CNN architectures used for each dataset and their potential impact]

5. Training Process:

[Analyze any differences in the training process, such as number of steps, batch size, etc.]

6.4 Conclusions from Comparison

[Summarize 2-3 key learnings from this comparison. What does this tell us about the nature of these datasets and the challenges in image classification tasks?]

This comparison highlights [key point about deep learning or CNN performance on different datasets]. It underscores the importance of [critical factor in CNN design or training] when dealing with more complex image classification tasks.

7. Part VI: CNN Performance Improvement

Based on the observations from Parts IV and V, this section focuses on improving the performance of the CNN model on the CIFAR-10 dataset.

7.1 Proposal for Improvements

After careful consideration of the current model's performance and the nature of the CIFAR-10 dataset, I propose the following changes to potentially improve the CNN's performance:

1. Increase Model Complexity:

[Explain how you plan to add more layers or increase the number of filters in existing layers]

2. Data Augmentation:

[Describe techniques like random rotations, flips, or color jittering to artificially expand the dataset]

3. Regularization Techniques:

[Discuss the introduction or adjustment of dropout, L2 regularization, or batch normalization]

4. Learning Rate Schedule:

[Explain the implementation of a learning rate decay strategy]

5. Batch Size Adjustment:

[Discuss any changes to the batch size and the rationale behind it]

Rationale for Proposed Changes:

[Provide a brief explanation of why you believe these changes will lead to improved performance]

7.2 Implementation of Changes

The proposed changes were implemented in a new Jupyter Notebook: [Updated\_Notebook\_Name.ipynb]. Key modifications to the code include:

[List 2-3 key code changes, possibly with brief code snippets]

7.3 Results of Updated Model

The updated CNN was trained and tested using the same procedure as before (5000 steps, testing every 100 steps). Here are the results:

[Insert a table or graph showing accuracy and loss over training steps for the updated model]

Final Test Accuracy of Updated Model: [Y.YY%]

7.4 Comparison with Original Model

To visualize the impact of our changes, here's a comparison of the original and updated models:

[Insert a line graph showing accuracy over steps for both the original and updated models on the same plot]

Key Observations:

1. [Observation about changes in initial accuracy]

2. [Observation about differences in learning rate/speed]

3. [Observation about final accuracy comparison]

4. [Any other notable differences in performance]

7.5 Analysis of Results

[Provide a detailed analysis of how the changes affected the model's performance. Discuss which changes seemed most impactful and why. If some changes didn't yield the expected improvements, hypothesize why.]

7.6 Conclusions and Further Improvements

[Summarize the effectiveness of the implemented changes. Discuss what was learned from this process and suggest any further improvements that could be made based on these results.]

This experiment in improving the CNN's performance demonstrates [key learning about model optimization]. It highlights the importance of [critical factor in deep learning] and provides valuable insights for future work with complex image classification tasks.

8. Conclusions

This final project for ADTA 5550: Deep Learning has provided valuable insights into the practical application of Convolutional Neural Networks (CNNs) using TensorFlow. Here are the key conclusions drawn from this study:

8.1 Dataset Exploration and Preparation

[Summarize key learnings from working with the public domain dataset and CIFAR-10, including data handling and preprocessing]

8.2 CNN Implementation

[Discuss main takeaways from building and training a CNN using TensorFlow, highlighting any challenges faced and overcome]

8.3 Performance Analysis

[Summarize insights gained from comparing CNN performance on MNIST and CIFAR-10 datasets]

8.4 Model Optimization

[Reflect on the process of improving the CNN's performance, discussing successful strategies and lessons learned from less successful attempts]

8.5 Overall Learning Outcomes

1. [Key learning point about deep learning theory or practice]

2. [Important insight about working with complex datasets]

3. [Significant understanding gained about model optimization]

4. [Any other major takeaway from the project]

8.6 Future Work

Based on the experiences and results from this project, potential areas for future exploration include:

- [Suggestion for further model improvement or different architecture]

- [Idea for applying similar techniques to different types of data]

- [Proposal for exploring more advanced deep learning concepts]

In conclusion, this project has demonstrated [overall statement about the power/challenges of deep learning]. It has provided practical experience in [key skills developed] and deepened understanding of [important concepts]. The skills and knowledge gained through this project will be valuable in future deep learning endeavors and real-world applications.

9. References

[List all references used throughout the project, including but not limited to:]

1. [Reference for TensorFlow documentation]

2. [Reference for CIFAR-10 dataset]

3. [Reference for chosen public domain dataset]

4. [Any academic papers or books consulted]

5. [References for deep learning concepts and techniques used]

6. [Any other relevant sources of information]

[Ensure all references are properly formatted according to the required citation style]