Tutorial Report

Wilson Schwegler
Schwegler@oxy.edu
Occidental College

1 Introduction

1.1 Comps Project Topic

Traffic sign recognition is a critical aspect of self-driving and intelligent transportation systems. Cars with Advanced Driver Assistance Systems have become very popular and are indicating that self-driving cars are in our future. The topic for my comps project is implementing a Convolutional Neural Network (CNN) for traffic sign recognition.

1.2 Tutorial

Given my comps topic, completing the tutorial "Image Classification with Neural Networks in Python," (https://www.youtube.com/watch?v=t0EzVCvQjGE) serves as a foundational guide for understanding the basic techniques involved in training neural networks to recognize images. This tutorial is relevant to my comps topic as it deals with image classification and recognition which will be a necessity for implementing a CNN for traffic sign recognition.

1.3 Goals

Having no prior knowledge of CNNs, the goal of completing this tutorial is to learn what neural networks are and how a basic one can be implemented in Python. Another goal is to acquire the basics and necessary skills to build a robust CNN. A successful outcome of this tutorial involves obtaining an accuracy of over 60 percent during the training phase and the successful classification of various test images.

2 Methods

To complete the tutorial I used Pycharm as my IDE on my Mac computer. This tutorial was intended for Windows computers so I had to make deviations at the start as my Mac has an M1 chip.

2.1 Environment Setup

The tutorial utilizes TensorFlow which is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning. In order for me to use it on my Mac it required adjustments to my Python interpreter in Pycharm to run it using my M1 chip. I first needed to download the environment manager Conda and create an environment. I then needed to activate the environment and install TensorFlow and other imports required for this tutorial. After completing this I set my Python interpreter to the newly created Conda environment and was able to begin the tutorial.

2.2 Dataset

This tutorial utilized the cifar10 data set, which consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. In order to increase efficiency, only 20,000 training images and 4,000 testing images were selected. Cifar10 is a good data set to use for this tutorial over one like MNIST. Cifar10 is a large diverse data set that represents real-world image recognition tasks. In the same sense, it relates my my comps project by being a similar data set to one that would be used for traffic sign recognition. MNIST would not be a god fit as it is a small data set with images that have very little variability. This data set would do a poor job at representing real-world recognition tasks.

2.3 Model Architecture

The tutorial's architecture consisted of three convolutional layers followed by max-pooling layers for feature extraction. The flattened layer prepared the data for the densely connected layers, which then led to the final output layer for classification. The code utilized in the tutorial architecture is as follows:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation='
    relu', input_shape=(32,32,3)))
model.add(layers.MaxPool2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation='
    relu'))
model.add(layers.MaxPool2D((2,2)))
```

```
model.add(layers.Conv2D(64, (3,3), activation='
    relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

2.4 Training and Evaluation

The model was compiled using the Adam optimizer, which is short for Adaptive Moment Estimation. It is an iterative optimization algorithm used to minimize the loss function during the training of neural networks. The Adam optimizer was utilized as it has a fast convergence and is easy to implement. The training process spanned 10 epochs or 10 complete passes of the training data set through the algorithm. The model's performance was evaluated utilizing the testing images.

2.5 Model Loading and prediction

After the model is trained and saved, it is utilized to make predictions on a new image. The saved model is loaded to ensure the consistency of tests. The selected test image is read utilizing OpenCV and converts the BGR color space to RGB. The processed test image is than passed through the model for prediction. The index of the predicted class is extracted, corresponding to one of the ten classes in the cifar10 data set. At this point, the prediction is determined and outputted.

3 Metrics and Results

Based on my goals defined in the introduction section there are different ways to measure their success. The first two goals are to learn what neural networks are and how to implement a basic one in Python. These goals have been successfully met after completing this tutorial as it served as resource to learn the basics of what a neural network and are it showed how a basic neural network is implemented in Python. The other two goals were to obtain an accuracy of over 60 percent during the training phase and the successful classification of various test images. After the training phase the model was analyzed using:

```
loss, accuracy = model.evaluate(testing_images,
    testing_labels)
print(f"Loss: {loss}")
print(f"Accuracy: {accuracy}")
```

This was utilized to determine the accuracy of the model to be 65.70 percent which exceeded the goal of 60 percent. In order to test the successful classification of various test images I selected 10 images to cover the 10 classes. Each image was tested and successfully predicted by the model. To expand beyond the tutorial, in order to measure the accuracy of a neural network one would test how well it is at predicting outcomes or comparing it to other algorithms completing the same task. These measurements would provide insights into its comparative effectiveness against alternative methods. This evaluation would also show the validity of a model's robustness and determine its real-world applicability.

4 Reflection

4.1 Tutorial Process

Following this tutorial was very straight forward, however the setup was very difficult. Getting TensorFlow to work on my Mac took just as long as the actual tutorial itself. When looking for information to help me set up TensorFlow, I found a lot on Jupyter Notebook and had a hard time finding information relating to Pycharm. However, once I got that figured out the tutorial went very smoothly. The code was easy to learn and interpret. I do believe this tutorial provided me with the basic framework of how to build a neural network, but I still feel like I am lacking a lot of knowledge about neural networks. Reading into more articles to fully understand how neural networks work and to familiarize myself with the vocab will be helpful.

4.2 Comps Topic

After completing this tutorial I am excited to get working on my actual comps topic. I do worry that my topic may be a little too simple or not challenging enough. I may consider expanding upon the idea and compare different algorithms to determine which one is the best for traffic sign recognition. I may also consider testing different data sets to determine which one is the best suited for traffic sign recognition. It also might be interesting to do the sign recognition in real time while driving through a web camera.

4.3 Concerns

My main concern is the difficulty of the project. I am worried that I might start off too simple and then expand the project to something that may be too big for me to finish, leaving me to revert back to the simple idea. My other concern is finding a good enough data set. I am worried that I could use a poor data set that may effect my metrics and results leaving me with an unreliable conclusion.