Final Project Report

**Name: Andrew Renninger Email: awrenninger@crimson.ua.edu**

Statement:

For this assignment's preparation, the author(s) did not use any generative AI tools.

For this assignment's preparation, the author(s) have utilized [Generative AI Tool Name], a language model created by [Generative AI Tool Provider]. Within this assignment, the [Generative AI Tool Name] was used for purposes such as [e.g., brainstorming, grammatical correction, writing paraphrasing, citation, specific sections of the assignment].”

***1. Introduction***

In this project, I trained a machine learning model on a dataset, made it ready for deployment and usage via API, then deployed it. Additionally, I made a partner page where users can draw digits online and then send to the API.

***2. Dataset Selection & Model Training***

***2.1 Dataset Description***

For this project, I selected the MNIST dataset. It is a very significant benchmark in the field of machine learning, computer vision to be exact. The MNIST dataset, comprising 28x28 grayscale images of handwritten digits, serves as a benchmark for evaluating various image classification algorithms. Its simplicity, accessibility, and well-defined nature make it an ideal choice for validating and comparing the performance of different models. The moderate size is also very valuable – large enough to train to a reasonable complexity, but small enough that it is possible to train on consumer technology. Additionally, because I had used it for a previous assignment for this class, it made sense to pick the dataset again.

2.2 Model Training and Serialization

To train the model, first I normalized the data, dividing the values by 255.0 in order to get the values between 0 and 1. I also one-hot encoded the classification labels. Next, I trained a Convolutional Neural Net (CNN) on the dataset. The design can be seen in figure 1.

A screen shot of a computer code

Description automatically generated

Figure 1. CNN Model Design

A screenshot of a computer screen

Description automatically generated

Figure 2. Model Accuracy and F1 score.

The F1 score and accuracy on the testing data can be seen in figure 2. Keras allows models to be serialized in a single step with the model.save() method. By calling this method with a filename, the model is saved to the file and can be loaded with all necessary information by calling keras.models.load\_model(). This can be seen in figure 3.

A computer code on a black background

Description automatically generated

Figure 3. Serializing and deserializing CNN using Keras

3. Deployment Preparation

3.1 Selection of Deployment Tool/Platform

For the simplest deployment possible, I chose to use Flask, a python library. It allows for quick API development and deployment. While being simple, it is also very popular in industry for its reliability and resilience.

3.2 Web/API Endpoint Development

The API should accept an image, check to make sure it is 28x28, greyscale it, normalize the pixel values, flatten the image into an array, feed it into the neural network, then interpret and return the results.

Following this, I developed the main code of the API as seen in figure 4. Note that it wraps the main code in a try catch clause, and returns the error code to the user. Depending on the authentication to the API, you would not want to do this. If open to the internet, this would be a very bad idea.

A screen shot of a computer program

Description automatically generated

Figure 4. Main API Source Code

4. Deployment and Testing

4.1 Deployment Process

4.2 Testing Methodology

4.3 Test Results and Observations

5. Reflection

5.1 Challenges and Solutions

5.2 Lessons Learned

6. Conclusion

7. References