Classification of Handwritten Numbers

By: Randy Grant

Abstract

This attempts to create a method to classify handwritten numbers. The client known as FakeBank has a need to start classifying numbers which have been handwritten in various types of handwriting. This will enable the bank to start a basic neural network of recognizing digits 0-9 first, which can then be transferred to learning larger numbers commonly written on checks.

Design

Firstly, I accessed the MNIST dataset from the Keras dataset library. From there, I converted the classes to categorical class matrices, ensured the data was parsed as floats, created some initial metrics like shapes, and plotted 25 random images from the dataset. I then put each task into its own section following the lessons taught here (private course link). EDA was performed, and then a baseline logistic regression model was created noting the accuracy score. From there, a CNN model using Keras was created for which inference was done on the test set as well as a completely separate image I made from writing a number down, taking the picture, uploading it to a directory, then importing and converting the image to the needed formats for model inference.

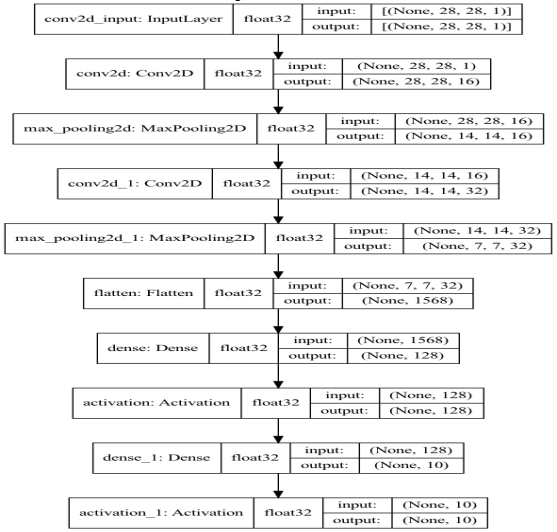
Data

The data used for this project were non-tabular data from a package in the datasets via TensorFlow's Keras. This is also located via a manual download from here. The sets were normalized with the amount of 60,000 training images and 10,000 test images with both having a 28x28 shape respectively. As stated in the design section, there was also another image for inference I created manually to test the model.

Algorithms

Logistic regression for a baseline, afterwards creating a CNN model with the following configuration:

- Uncompiled sequential model function:
  - Added two 2-dimensional convolutional layers padded as same and activated with relu
  - Included 2-dimensional max pooling
  - Units of 16 and 32 respectively with a kernel size of 3
  - Flattened to a 1-dimensional vector
  - Added one dense layer with 128 units activated with relu
  - Added a final dense layer with 10 units activated (matching the number of classes) then activated with softmax
- Compiled function:
  - Calls the uncompiled function, compiles the sequential model with:
    - Loss function of categorical entropy
    - Adam optimizer
    - Accuracy metrics



A transfer learning model is also included for comparison to CNN with:

- A Sequential model
- Pre-trained model called ResNet50 with average pooling and imagenet weights
- Addition of one dense layer with 512 units activated with relu
- Added a final dense layer with 10 units activated then activated with softmax
- ResNet50 layers set to not trainable

Tools

Python, tensorflow and keras for data manipulation, scikit-learn for scores and confusion matrix, pylab, rcParams, matplotlib, and seaborn for plots and graphs.

Communication

These slides and visualizations were presented, and this was uploaded to github.