Handwriting Recognition using Optical Character Recognition

Keller Lawson

Computer Science

University of Idaho

Coeur d'Alene, ID

Introduction

The aim of this project is to create, train, and test a model that will be able to identify English characters and numbers with high accuracy. Everyone writes letters in their own way, but with a typical resemblance to the standard English alphabet. With this in mind, this project seeks to be able to use a variance of written letters to train the model so a multitude of handwriting styles will be recognized. The project will only have support for block letters and not calligraphy or cursive writing.

Datasets

There are two datasets that I used for the training and testing. These datasets will be stacked to create one dataset.

English Alphabet (a-z): https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format

Digits (0-9): https://www.kaggle.com/oddrationale/mnist-in-csv/

Packages

The following are the packages I have used so far in the project. I installed the TensorFlow version that utilizes the GPU after attempting to train my model the first time and the estimated time was around 50 hours. After switching to my GPU, the training took around 2-3 hours.

```
import matplotlib
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import SGD
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import build_montages
import numpy as np
import argparse
import cv2
import tensorflow.keras as keras
```

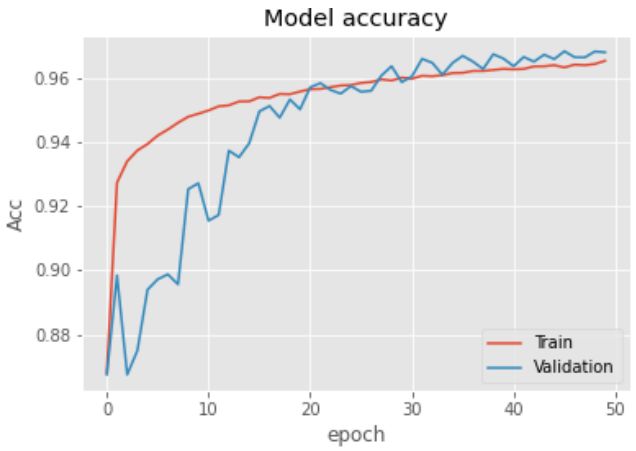
Approach

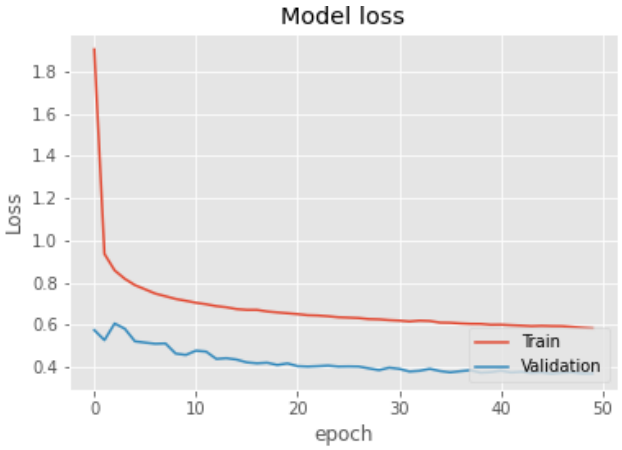
To train the model mentioned above, I chose to use Tensorflow and Keras to create a ResNet model. To train this model, I imported my two data sets. The first data set was for the English alphabet letters a-z. The second data set was for the digits 0-9. I merged these two data sets into one stacked data set and converted the image sizes from 28x28 to 32x32 to match the architecture we would be using. Before creating and training the model, I split the training dataset into a training and test data set where the test data set was 20% of the entire dataset with a random state of 13 to allow for the same data splits every time if the code needed to be ran again.

Results

Below are the results of the training and testing. The parameters I ended up using for my final training were a batch size of 128, 50 epochs and initial learning rate of 0.1. I tried running the training function 2 times before landing on these parameters. The first I trained with a batch size of 128, 20 epochs and learning rate of 0.0001. These parameters finished training with an accuracy of about 43%. Changing only the number of epochs only did not merit much better results. However, when I changed the initial learning rate, from 0.0001 to 0.1, I received much better training results which is why I decided to stop on those parameters. The plots and pictures below show the results of the training with the final parameters.

accuracy			0.97	88491
macro avg	0.96	0.96	0.96	88491
weighted avg	0.97	0.97	0.97	88491





From the training and test results shown above, the final model accuracy was around 96-97%. If we look at the two montages that display 49 test images each, we can see where the model struggles. In the second montage, it is unable to correctly identify 2 characters. The top left character that was incorrect it matched the character to as S which from it's label is incorrect. However, looking at the picture, it is hard for me to determine the character was well. As a guess, I would think it might be a 5. The lower character it missed seemed to be the most common miss. This is the difference between the digit 0 and the letter O. These two characters are so similar to each other depending on a person's handwriting, it is very simple to mistake one for the other.

Conclusion

The training of the model went better than I expected and work's great. I will be moving on to feeding it an image containing handwritten words or sentences and having the model identify the characters in the image. With the recent lecture's we had in class, I will be reviewing my neural network and applying optimizations/changes to the code. With the training time required for each new iteration of the model, it is unlikely I will be able to implement real-time character identification using the model due to time restraints.

References:

https://data-flair.training/blogs/python-deep-learning-project-handwritten-digit-recognition/

https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5

https://docs.opencv.org/master/d7/dbd/group__imgproc.html

https://github.com/jrosebr1/imutils