Journal Report 7

10/14/19-10/25/19
Odin Woitek
Computer Systems Research Lab
Period 1, White

Daily Log

Tuesday October 15

My partner found an article about glob.glob not being sorted consistently, which could be an issue, since we load the the training and testing sets from glob.glob each time we run the model. I decided to use pickle to dump the sets into a file so we can test the model on a consistent test set.

Thursday October 17

I decided to save the model with pickle after fitting it on the training set, so I don't have to run it again once I get something working. I tried running the model with the exact same architecture as the MNIST model just to see what would happen, and immediately found an issue with my output sets. They were just really long one-dimensional arrays of strings of an authors serial number. After some googling and reading the tutorial more carefully, I found that I needed to make the outputs a class matrix with the number of dimensions equal to the number of possible authors.

Monday October 21

I decided to use scikit-learn's label encoder to normalize the labels in my output sets and convert them to numerical labels, so my output format can be as similar as possible to the format from the MNIST tutorial so I can still use it as a guide to find out what I need to do.

Tuesday October 22

I fixed the glob.glob issue by pickling the inputs and outputs before loading them into training, testing, and validation sets. Doing this also reduced the startup time before the model runs, since getting those inputs and outputs with glob.glob every time took a while. I added the encoder to assign numerical labels and used np_utils.to_categorical to convert the output dimensions as I found I needed to do last Thursday. Once I had adjusted the architecture so the model would be able to train, I started running it, leaving everything else the same as the MNIST architecture. Since the images I am using are much larger than the MNIST images, it will take significantly longer to run. In about 2 hours, I'll just look at the results to see how well this preliminary test works.

Thursday October 25

At the end of last class, I decided to run the model that I had just to see how well it worked. By lunchtime, the model had finished running and had a loss of 2.728440862317759 and an accuracy of 0.27062204480171204. This was about what I expected, since the model was designed for the MNIST problem. When running this, I realized we pretty much got to that accuracy and loss level by the second epoch, and that the model took way to long to make very small improvements in accuracy. Before I started looking for how to change the architecture, I tried to find out if there were issues with how we were feeding in data. After looking at other models for analyzing handwriting, I think we have an unnecessarily large amount of input data, with over 23,000 113x113 images. There also isn't an even amount of contributions between authors, which could also present a problem since the first author has a significantly more contributions than any of the other authors. I was concerned that having the structure of the output being dependent on the number of authors being fed into the network could be a problem, but other projects have done similar things. I'll keep note of this potential issue for later, but for now I'll work on improving the model itself.

Timeline

Date	Goal	Met
September 30 - October 4	Set up Tensorflow for GPU for higher performance	No, Tensorflow GPU only works with NVIDIA graphics cards. I also tried optimizing for Intel chips but ran into problems there as well.
October 7-11	Load input and output data into training and testing sets	Yes, Used sci-kit learn to split data into training, testing, and validation sets.
October 14-25	Run and evaluate model	Yes, using the architecture of the MNIST tutorial model, I got an accuracy of about .27 and a loss of about 2.73
October 28 - November 1	Continue research on CNNs and improve accuracy to 0.3	
November 4-8	Improve accuracy to 0.5	

Reflection

These two weeks were mostly just dealing with handling data and what the input and outputs of the model were going to look like. I was not concerned with the actual model itself yet because I wanted to make sure everything around it was correct before I started working on it, in order to avoid having my work become useless when I find out that I should have handled my data some other way. I decided to pickle the training and testing sets so they would be consistent each time I ran the model, so I wouldn't accidentally train on a previous testing set or, more importantly, I wouldn't test on a previous training set, which would defeat the purpose of a testing set. Once I had done this, I thought I was ready to run the model, but came upon an error with my output sets, which I was able to fix by making the output a class matrix, using np_utils.to_categorical. I also encoded the output data using scikit-learn's LableEncoder just to make things simpler by having integers represent each author rather than strings of their serial numbers. My original idea for pickling the training and testing sets was to do so after splitting them, but I found that it was much simpler and faster to just do it before splitting them, so I made that alteration as well. With this done, I was ready to run the model. It ended up getting stuck at a loss of 2.728440862317759 and an accuracy of 0.27062204480171204 pretty quickly, and didn't make many improvements off of this for the next two hours of running. I knew that I needed to adjust the architecture to better fit my problem, since I hadn't touched it at all since the MNIST tutorial, but I also looked for other possible issues with handling the data. Specifically, I realized that the reason this was taking so long was that I just had so much data. After a lack of improvements after the initial epoch, I started to think that I need to make my dataset smaller for the sake of time. I also decided that I needed to address some of my other concerns about the data I was feeding into the model, specifically the uneven contributions by different authors, meaning one result was severely over represented in the data when there is no practical reason for that to be the case.

Here is how I handled the data before running the model. The model didn't use a validation set, which is why I just created it and didn't do any restructuring.

```
import pickle as pkl
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import numpy as np
import glob
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D
from keras.utils import np_utils

np.random.seed(123)

x = pkl.load(open("inputs.pkl", 'rb'))
y = pkl.load(open("outputs.pkl", 'rb'))

encoder = LabelEncoder()
encoder.fit(y)
encoded_y = encoder.transform(y)

X_train, X_test, y_train, y_test = train_test_split(x, encoded_y, test_size=0.167,

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
print(X_val.shape, y_val.shape)
plt.imshow(x[0])

X_train = X_train.reshape(X_train.shape[0], 1, 113, 113)
X_test = X_test.reshape(X_test.shape[0], 1, 113, 113)
print(X_train.shape)

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

y_train = np_utils.to_categorical(y_train, num_authors)
y_test = np_utils.to_categorical(y_test, num_authors)
print(y_train.shape)
```