Individual Report

Introduction

The overall goal of this project was to propose a new method for classifying polygons based solely on their shape without incorporating other GIS-derived data. We believed this could provide value to the field of GIS because, there is a lot of mapping data for which the metadata describing the polygons is not available. We did this by converting raw shape files (relative coordinate vectors) to 2D shapes in 100x100 images and then using these images to train a CNN to predict the given classes provided in the Veer, Rein van 't et al paper. Nathan was in charge of acquiring and transforming the data. I then used the data he provided to begin to build the network architecture which both Nathan and I simultaneously tuned. Limin took the same data and trained a pretrained classifier to compare results.

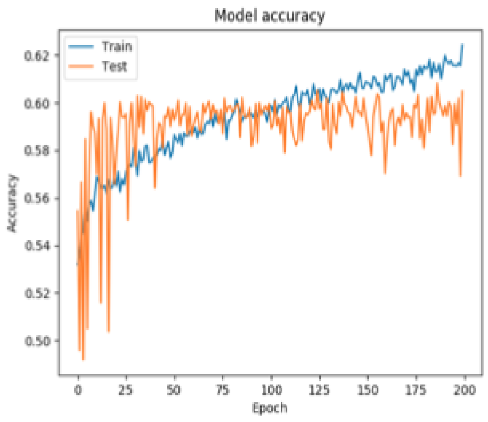
My Role

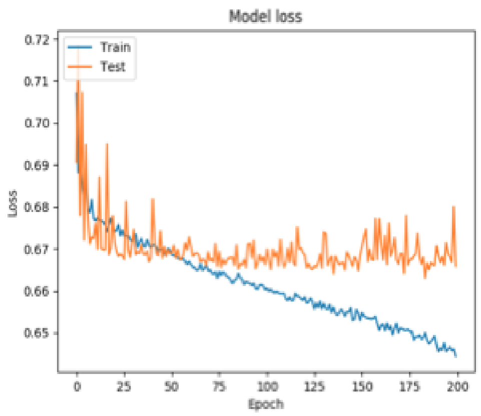
I wrote a majority of the training scripts for the “from scratch” model as well as the predict script to evaluate our results. My main training script was then copied by Nathan and used so we could simultaneously run and tune models. All of the training scripts in our final product created from the initial script I had written, only changed by doing things like adding/subtracting layers, and altering hyperparameters. I wrote all of the code which generated the train/validation figures and prediction values (classification report, loss/acc).

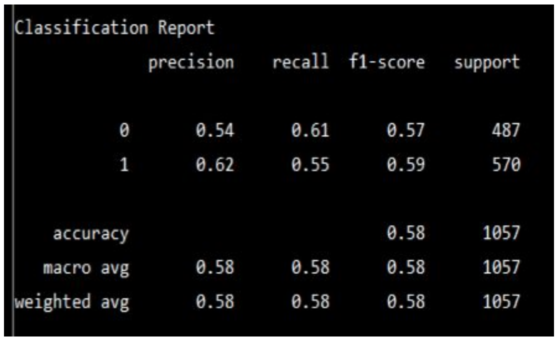
Results

"Neighborhoods"- Baseline Architecture

Using the baseline architecture that was laid out in the paper, we found a relatively unstable model that only did a bit better than random guessing (~50% for random guess). As you can see in the figures to the right, not a lot of learning occurred, and the growing train accuracy and loss indicates overfitting after around epoch 75. Our testing set results show an overall accuracy of 58%, which is quite below the accuracy found by the 1D CNN in the paper (66%). Test results shown below:



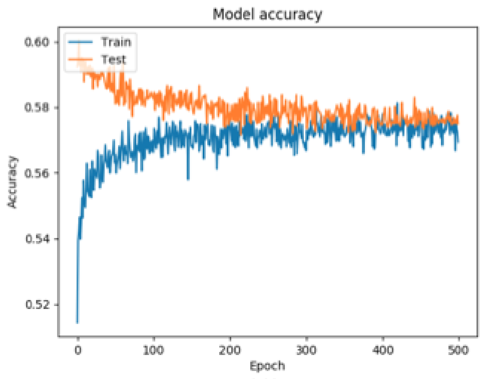


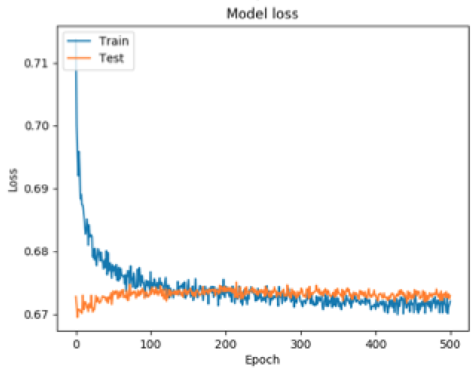


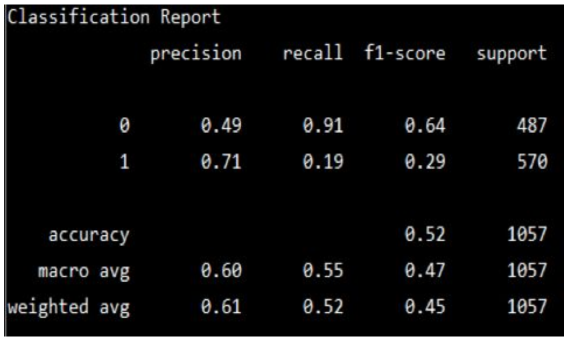
```
Classification Report
              precision    recall  f1-score   support

           0       0.54      0.61      0.57       487
           1       0.62      0.55      0.59       570

    accuracy                           0.58      1057
   macro avg       0.58      0.58      0.58      1057
weighted avg       0.58      0.58      0.58      1057
```

"Neighborhoods" More Complicated Model

Again, we found a model which did not seem to learn and floated just above null accuracy rate during training, only to give the null accuracy for testing as shown below:



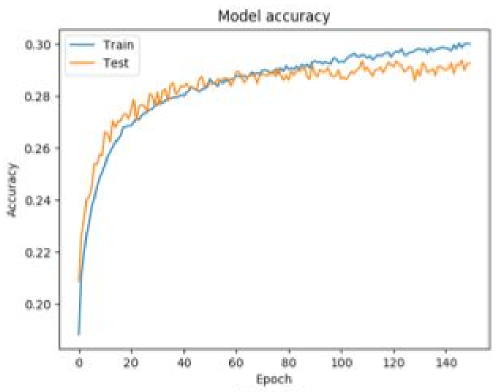


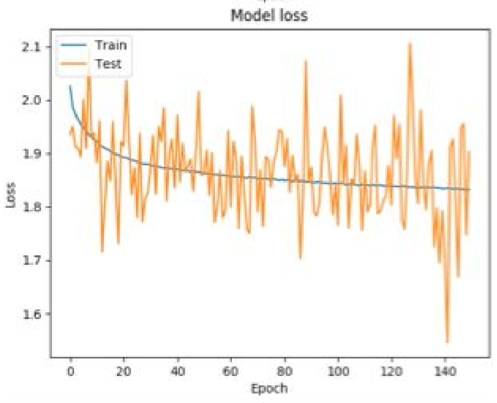


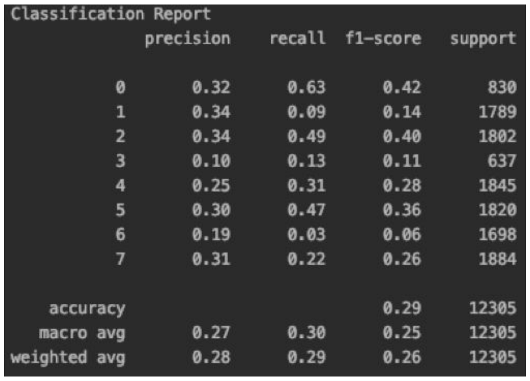
```
Classification Report
              precision    recall  f1-score   support

           0       0.49      0.91      0.64       487
           1       0.71      0.19      0.29       570

    accuracy                           0.52      1057
   macro avg       0.60      0.55      0.47      1057
weighted avg       0.61      0.52      0.45      1057
```

"Buildings" Initial Model

The initial model found our highest validation accuracy but very unstable validation loss. This indicated we had a degree of overfitting. Our testing accuracy made it to 0.29. Final accuracy on testing set was also 0.29 which is a significant step above null accuracy but a bit lower than what was found with 1D convolution which found 0.40 accuracy.



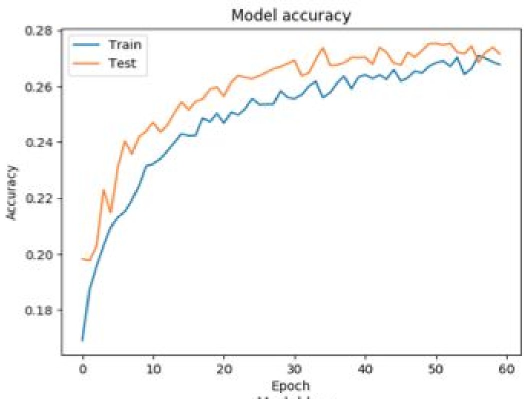


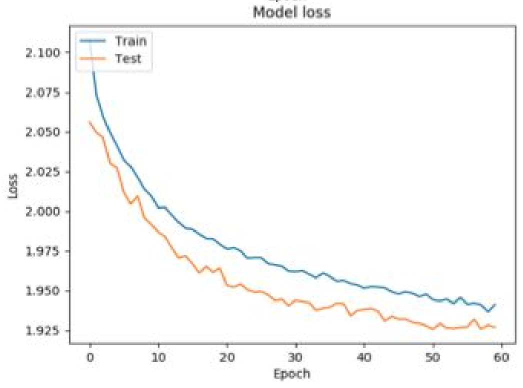


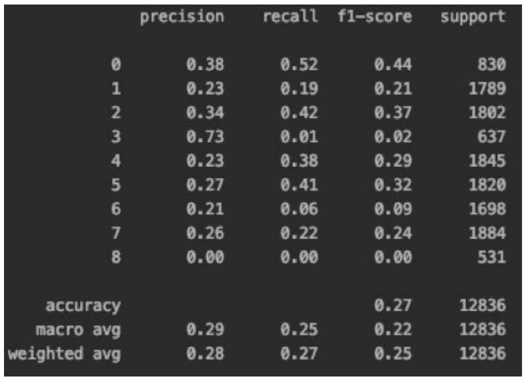
```
Classification Report
              precision    recall  f1-score   support

           0       0.32      0.63      0.42       830
           1       0.34      0.09      0.14      1789
           2       0.34      0.49      0.40      1802
           3       0.10      0.13      0.11       637
           4       0.25      0.31      0.28      1845
           5       0.30      0.47      0.36      1820
           6       0.19      0.03      0.06      1698
           7       0.31      0.22      0.26      1884

    accuracy                           0.29     12305
   macro avg       0.27      0.30      0.25     12305
weighted avg       0.28      0.29      0.26     12305
```

"Buildings" Final Model

To mitigate the overfitting in the "Buildings" initial model, we tested the effects of increasing the number of layers in the model architecture. We found a much more stable validation loss as a result, although we sacrificed some accuracy, with max validation accuracy topping out around 0.27. The final testing accuracy, shown below was also 0.27, a bit lower than the 0.40 found on this dataset with 1D CNN in the paper.







```
              precision    recall  f1-score   support

           0       0.38      0.52      0.44       830
           1       0.23      0.19      0.21      1789
           2       0.34      0.42      0.37      1802
           3       0.73      0.01      0.02       637
           4       0.23      0.38      0.29      1845
           5       0.27      0.41      0.32      1820
           6       0.21      0.06      0.09      1698
           7       0.26      0.22      0.24      1884
           8       0.00      0.00      0.00       531

    accuracy                           0.27     12836
   macro avg       0.29      0.25      0.22     12836
weighted avg       0.28      0.27      0.25     12836
```

Overall, the above results were all generated from my code, although tuning the models was a collaborative effort between Nathan and I.

Summary and Conclusions

Overall, the largest drawbacks came from not limiting the project to one specific aspect of the Veer, Rein van 't et al. paper. This caused us to not spend enough time tuning a specific model, but rather, sampling options for models for multiple datasets within the paper from which this project was inspired. I wrote the conclusions portion of the final report and will reiterate them here:

The goal of our project is to develop a deep learning framework for classifying spatial polygons, based solely on their geometry, that is more flexible, light-weight, and accurate than existing frameworks.

Overall, our method proves to be more flexible than methods which require raster or vector data because it has the ability to classify shapes for which there are no raster or vector components and only pure shapes from maps, irrespective of geolocation.

Towards our second endpoint of creating a more light-weight method, it did not entirely succeed. Ultimately, the use of images instead of relative coordinate vectors is likely not lighter-weight as the images require more memory to store.

Towards our final endpoint, we were unsuccessful in training a higher-accuracy model than the 1D CNN. We believe this is largely due to the fact that we introduce noise by putting the shape onto a white background. Overall, this introduction of noise may make it more difficult to extract features. In addition, we spent a lot of time and assessing the viability of the data used in the paper for modeling and did not make it to a point where we could continue to tune the models further for higher accuracy. The final “Buildings” model listed above seemed to have the most promise. We believe that this model could be further tuned by using SGD with decay and running for a large number of epochs to further increase accuracy.

Code Percentages

I cannot say that I used code from any specific repositories or sources to write my portion of the code for the project. I would site that I referenced code that I had helped develop for my capstone project with Binbin Wu in this repository:

https://github.com/dvitale199/plant-classification

What I used from my previous project was very little actual code aside from structure (like how to use the ImageDataGenerators in Keras etc.) but this code was sourced mostly from the Keras documentation and some from class exercises.

References.

Veer, Rein van 't et al. “Deep Learning for Classification Tasks on Geospatial Vector Polygons.” ArXiv abs/1806.03857 (2018): n. pag