Daniel Stevens

DATS 6203 Final Project

Using Computer Vision Methods to Predict Building Density Measurements Using Geospatial Image Classification

1. Introduction

When considering options for a topic to study for the final project of this Deep Learning class, geospatial imaging systems was an area of both mutual experience and interests of all group members. Wanting to look at something a bit outside of the box, we decided to study the ability of deep learning algorithms to accurately measure building density in geospatial images and classify them based off of that density. We knew that we wanted to limit the geographic area that we used for our project so that we knew that most of the images we used would either contain some type of structure or forested area. We chose Maryland, Virginia, and the District of Columbia (Fig. 1) because of the abundance of buildings in those areas but also because we didn't want to imbalance the training/testing data with huge forested areas and no buildings (as if our split randomly selected Alaska for training and the New York City metro area for testing for example). After narrowing the scope of our project, we were able to proceed to select which Deep Learning methods would be used to accomplish our research.

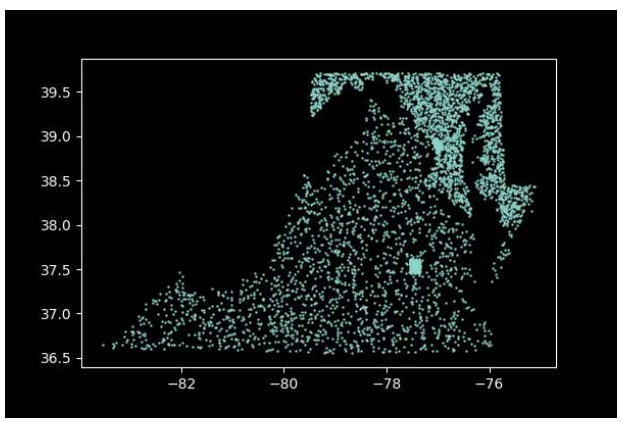

Figure 1

2. Individual Work

My contributions to the project started with a portion of the exploratory data analysis. When we limited the scope of the geographic images to be trained/tested we introduced the need to generate additional data as the scope of these three states was not enough to train a good image classifier. I wrote code to zoom in/out on our existing images and save them with the appropriate tags. The zoomed images were not used in our final analysis as they added more noise than clarity to the network without the presence of clearly defined structures in the zoomed in examples.

In the selection of and training of the models, we knew that we wanted to try multiple different algorithms to classify our images. Evan chose the Multi-Layer Perceptron network so I decided to use a Convolutional Neural Network using Keras. I chose the CNN because of how well it tends to deal with analysis using images and Keras because I gained experience with it in the class. I created the data pre-processing and splitting code for the CNN as well as designed the CNN layers and configured the output. I tried to optimize each layer of the CNN by implementing techniques such as batch normalization so that each layer is learning independently and finding the best kernel size that wasn't too small for the images being used (Fig. 2). I also created visualizations for the output of the groups' models as well as wrote the group report first draft and added changes as well as created the group presentation.

```
# %% ------------------------------------ Training Prep ------------------------------------------------------
model = Sequential()
model.add(Conv2D(32, kernel_size = (3, 3), activation='relu', input_shape=(100, 100, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
```

An example layer from the Convolutional Neural Network algorithm

3. Results

Evan chose the Multi-Layer Perceptron and was able to successfully classify geospatial images in our test set with a Mean Square Error of 0.00272 (Fig. 3). I chose to create a Convolutional Neural Network that was able to achieve a Mean Square Error of 0.00147 (Fig. 4). MSE was chosen as the validation metric because we are looking at predicting the building density of images can be viewed as a regression problem as well as a classification one. I would attribute this improvement to how much more effective CNN's tend to be as opposed to other algorithm when it comes to reading, recognizing, and analyzing image data. These results were produced with more 55,000 images (some generated by flips and rotations) that were preprocessed and normalized. I did explore fine tuning some other aspects of the CNN such as increasing the dropout rate (above the 0.20 that was included in final version) and increasing the kernel size.

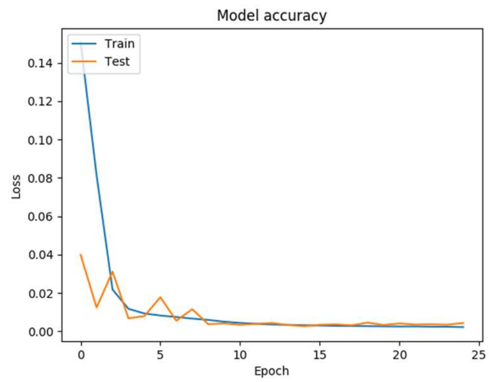

Figure 3 – MLP Loss Function

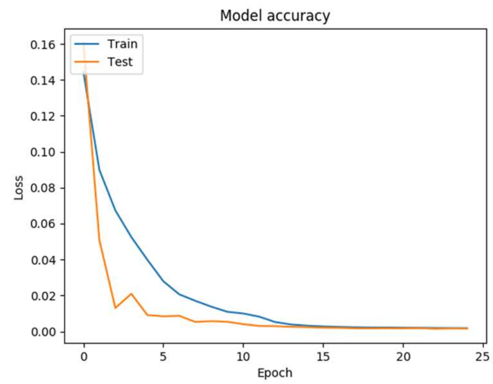

Figure 4 – CNN Loss Function

4. Summary & Conclusions

In the end, we were satisfied with what we were able to accomplish after starting from scratch with no data to finding the data set and being able to manipulate it into a usable data source. The Convolutional Neural Network proved to be a reliable predictor of building density in a given geospatial image. Some improvements that could be added in further studies would be adding more layers to the CNN to try to minimize loss even more as well as adding in some different transfer functions. Another approach that could be considered would be to pre-process the images by classifying the pixels in the image themselves as either building, forrest, etc… based off of their coloring.

Code Breakdown (excluding import statements):

57 – 21 / 57 + 18 * 100 = 0.48

5. References

1. (2019, September 16). Retrieved November 6, 2019, from https://github.com/microsoft/USBuildingFootprints.
2. SN5: Automated Road Network Extraction and Route Travel Time Estimation from Satellite Imagery. (n.d.). Retrieved November 11, 2019, from https://spacenet.ai/sn5-challenge/ .