Classifying Vacant Homes from Google Street Views using a Convolutional Neural Network

Robert Brasso

***Abstract*— Baltimore has upwards of 16,000 vacant homes and vacant homes outnumber homeless people 6 to 1. Vacant homes cause numerous criminal problems for a city including violence, prostitution, and drug use. Additionally, vacant homes represent lost taxes for cities, lost intergenerational wealth for families, drains on public resources, and lost home value. This research uses Google Street View images of homes in Baltimore to create a binary classification model using a convolutional neural network. The key goal of this research is to aid in building a machine learning model that can classify vacant homes in real time and provide actionable insights to community service organizations in the city of Baltimore.**

I. Introduction

In 1950, Baltimore's population was recorded at just under a million people and it ranked as the sixth largest city in the United States. Today, Baltimore's population is around 620,000 representing a 35% decline in the city's population over the last 60 years. One result of this dramatic decrease in population is that the city contains a significant amount of vacant homes. According to an article in *The Atlantic* from 2014, Baltimore has upwards of 16,000 vacant homes and vacant homes outnumber homeless people 6 to 1 (Semuels, 2018). While the city has tried for years to address this issue, most policies have fallen short of making a large-scale difference in the problem.

Vacant homes cause numerous problems for the overall health of a city. In a study by the Department of Biostatistics and Epidemiology at the University of Pennsylvania, researchers identified that despite socioeconomic factors, there was a significant relationship between aggravated assaults – particularly involving guns – and the percentage of vacant homes in census block groups in Philadelphia (Branas, Rubin and Guo, 2018). Additionally, in 2011, the New York Times wrote an article referencing the relationship between vacant homes, prostitution, and drug use in New York neighborhoods (Wilson, 2018). In combination with the criminal elements that take rise, vacant homes also represent lost taxes for cities, lost intergenerational wealth for families, drains on public resources, and lost home value (Apgar, 2018).

In previous research, I used the 2011 Housing Market Typology dataset from Baltimore Open Data to determine if a model could be created to accurately predict the market classification for a census block neighborhood based on a variety of housing market metrics such as the percent of vacant homes in a neighborhood, the amount of commercial land, average home value, etc. From that research, I concluded that many classification and clustering algorithms could classify market typology for census block neighborhoods with high accuracy, precision, and recall. Additionally, I identified that the percent of vacant homes was a key metric in classifying the market category.

The focus of this previous research was to aid in building a model that could ingest the housing market metrics, classify a neighborhood, and then provide actionable insights to community service organizations in the city of Baltimore. For instance, if a rise in one metric leads to a drop in the market category for a census block neighborhood and that drop is identified in real time, perhaps community services can be redirected to quickly stem the problem in that neighborhood before it becomes worse.

Many housing market metrics can easily be obtained in real time. For instance, home sales are quickly and readily reported on by websites like Zillow and Redfin, so we can easily apply that information to census block groups and get the average home sale price for each census block neighborhood. One metric that provides some problems for real time analysis however is the percent of vacant homes. Many cities in the country report on vacant homes, but they require people checking homes door to door or reporting from the USPS on homes where mail is not being received. The process of reporting a vacant home is thus slow and time consuming. However, as we established earlier, this metric is also one of the most important in identifying the market category of a neighborhood.

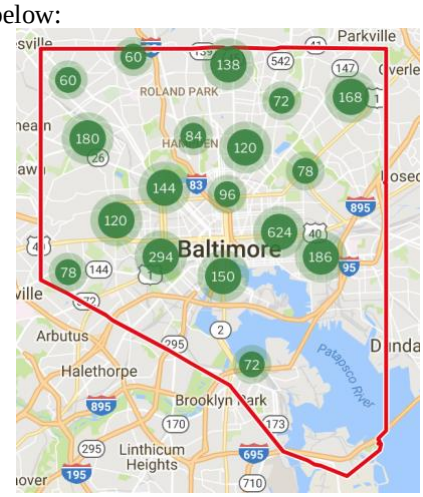
In this paper, I attempt to create a model that can quickly identify whether a home is vacant with little manual intervention. To achieve this, I used the existing list of known vacant homes from Baltimore Open Data plus I created a list of known non-vacant homes, fed the addresses from both lists into the Google Street View API to download images of the facades of the homes, then used those images to train a convolutional neural network. If the convolutional neural network could accurately classify whether a home was vacant simply by viewing the façade of a home, we could create a process such as affixing a camera to the top of a car and streaming the images into a database that checks against the neural network to determine in real time whether or not a home is vacant. That would then flag the homes that classified as vacant, so community services could be dispatched to the homes. Additionally, the model would be able to determine which homes were at risk of becoming vacant in the near future, so these homes could receive the urgent attention they need.

II. Dataset Description

The vacant buildings dataset came from Baltimore Open Data and contains a list of all known vacant buildings in

Baltimore City, with addresses and coordinates, updated bi-weekly. Finding a list of non-vacant homes proved to be considerably more problematic. Many websites offer mailing lists at a cost, but those lists are not always very accurate and the cost of getting a list for the entire city can be very high.

The first idea I had to get a list of non-vacant homes was to create a web scraper to scrape all of the non-foreclosed homes for sale on Zillow. After many failed attempts, I looked at alternative real estate sites. After browsing Redfin, I found a feature where I could export a list of up to 350 homes to a comma separated values file. I filtered Redfin to only look at Baltimore City non-foreclosed single family and townhomes. Redfin gave me results in Baltimore City that were grouped and scattered across the city as seen below:



Next, I went to each of those groupings and downloaded up to 350 homes data to a csv file. I used all of the groupings, attempting to get a reflective spread across the city. I combined all of the csv files into one master csv file and used that to define my non-vacant homes list.

The next step in getting my image dataset involved writing a program in Python to leverage the Google Street Views API. The program randomly assigned 70 percent of the data from each list to a training set and the remaining 30 percent to a testing set. The training and testing lists for both the vacant and non-vacant homes were cleaned to only have the street names and numbers. Those lists of names and numbers were pushed into a program that appended "Balitmore, MD" to the end of the address and then downloaded the images using the Google Street View API. The downloads from Google Street View API created a folder for each image and each folder contained the image and the metadata (as a json file) for the image. I wrote a program that combed through the folders and moved the images into a training and testing directory with sub directories for vacant and non-vacant homes, so the folders would only contain the images and not the metadata. The metadata was discarded.

III. Data Preprocessing

For the first batch of images that I downloaded using the Google Street View API, I passed in the coordinates hoping that those would be better at pinpointing the home I was targeting. After looking at the images, I realized that with the Google Street View API, if you use coordinates it will find the street closest to the exact coordinates and give a view of the coordinates from that street. In most instances, there was no uniformness of the images. Sometimes the images would be of the side of the home, other times the back or front.

Since this data was unusable, I proceeded to use the addresses, which provided me the façade of each home. I ran these into a very basic convolutional neural network and the accuracy of the classifier was very low. This prompted me to take a closer look at the images and I noticed the images from Google Street View API contained significant issues. The biggest issues were that most of the images contained trees, blurred sections of the image, or the viewing angle was different than that specified in the API settings.

Below is an example of an image that had too much noise from trees and thus it was very difficult to see the home:

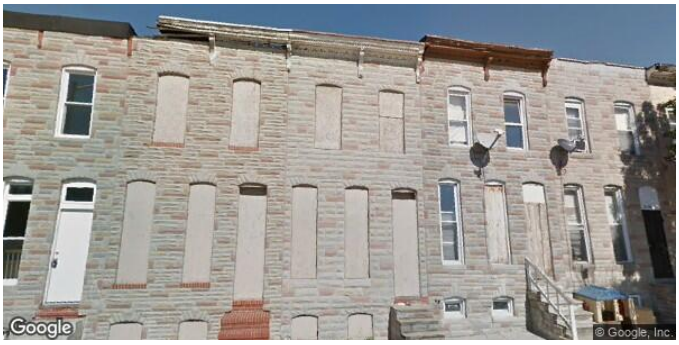
Additionally, below is an example of different image that has a viewing angle that is not normal:

The majority of images that I downloaded from the Google Street View API contained issues like this.

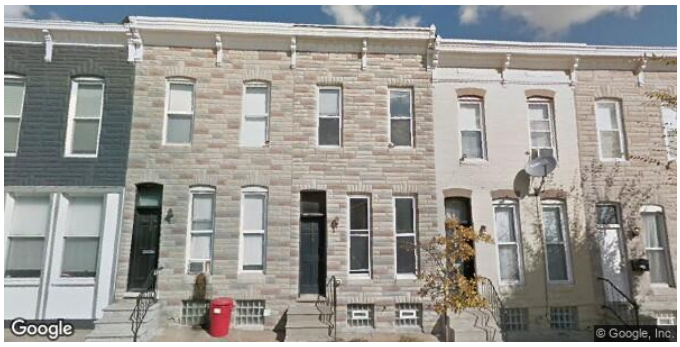
The quality of the images I was left with was very strong, however the image sets ended up quite small since about 85% of the images were unusable.

Below are sample images that were used in the neural network.

Vacant:



Non-vacant:



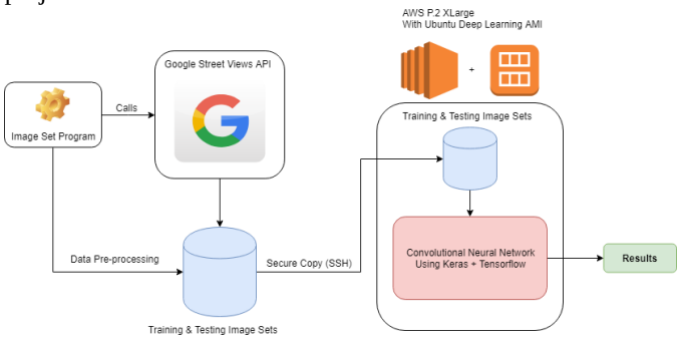
Due to the manual work of creating each image set, I first started out with smaller image sets and once I confirmed that the images with better quality resulted in a more accurate model, I increased the quantity of images in the image sets. Since the image sets are randomly created, I needed to repeat the process of checking all of the images every time I increased the size of the image set.

IV. Architecture

Convolutional neural networks, like the one used in this research require high amounts of CPU cores to process jobs in a time effective manner. Current quad core CPUs are too underpowered to provide timely results from convolutional neural network models. The key to success with convolutional neural networks is to leverage video cards that have several thousand CPU cores, such as Nvidia cards with CUDA cores (Shi et al., 2018).

The research in this paper was completed in early 2018 during a time of high demand for video cards with CUDA cores. The prices of video cards were currently double their normal price at this time due to a massive shortage caused by the rush on crypto currency mining.

The alternative solution I decided upon was to run my code leveraging an AWS EC2 instance. The design diagram for this project is shown below:



I choose the p2.xlarge instance and used the deep learning Ubuntu image. The p2.xlarge instance has 12gb of GPU memory and 2496 parallel processing cores (1 Telsa K80 GPU). Using spot pricing, I was able to get leverage this system at .42 cents per hour, while it would typically cost around $1.20 per hour. The major downfall to spot pricing is that the only way to stop your instance is to terminate it, thus losing all of the data on your instance.

Once my instance was set up, I connected over SSH and created directories, installed python packages, and setup Jupyter Notebooks. I used Secure Copy to copy all of the images I downloaded locally to my EC2 instance. Next, I launched my EC2 instance so that local host from my instance would be passed as my computers local host, so when I launched Jupyter Notebooks from my instance, it would load locally on my computer.

V. Neural Network Model

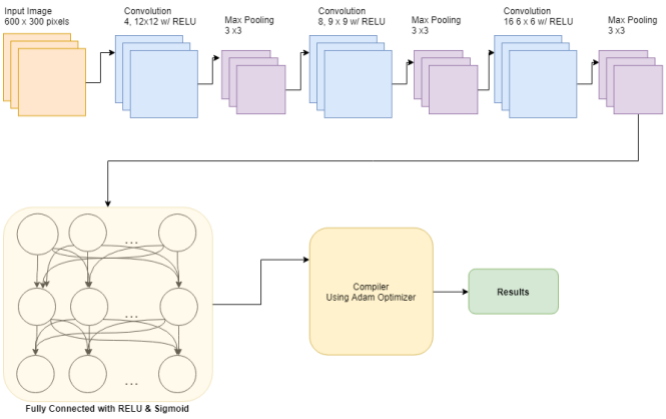
The model for this project was built using Keras with Tensorflow handling the backend processing. For all of my model iterations, I used RELU and Sigmoid for the fully connected layers. Additionally, I compiled the results using the Adam optimizer (Kingma and Ba, 2018), RELU activation (Agarap, 2018), and binary cross-entropy for loss.

For the first version of the model I used two convolutional layers. The first and second layers used a batch size of 32 with a 3x3 sliding window and pooling window of 2x2. This version used 400 steps per epoch, had 2 epochs, and 80 validation steps.

The second version of my model used a batch size of 4 with a 11x11 sliding window and 3x3 pooling window for the first layer, a batch size of 16 with a 6x6 sliding window and 3x3 pooling window for the second layer, and a batch size of 32 with a 3x3 sliding window and 2x2 pooling window for the third layer. I kept the steps per epoch at 400, used 5 epochs, and 100 validation steps.

The final version of my model used a batch size of 4 with a 12x12 sliding window and 3x3 pooling window for the first layer, a batch size of 8 with a 9x9 sliding window and 3x3 pooling window for the second layer, and a batch size of 32 with a 6x6 sliding window and a 3x3 pooling window for the third layer.

Below is a diagram of the final iteration of the network used for my research:



VI. Results

The results from my first model yielded very poor results. After further consideration, it was apparent that the sliding window I was using was much too small and could not pick out the subtle details that are unique to the vacant homes. After the first two model runs, I filtered through the pictures and noticed some minor gains with the accuracy getting up to 52.68%. After that, I ran the model again with new images with the second model. The initial results of this model were

better, and it achieved an accuracy of 56.38%. After parsing out the noisy images, the accuracy increased to 65.15% with a validation accuracy of 70.45%. In my final model run, I used another brand-new image set, but parsed through the images very strictly to remove unusable images. Combined with the parameters of the new model, the results for this run had an accuracy of 67.60% and a validation accuracy of 73.68%. The table below shows the results for each of the model runs and each epoch.

Model Run	Epoch	Accuracy	Validation Accuracy	FOV
1	1	0.4991	0.5008	40
1	2	0.5003	0.4992	40
2	1	0.4985	0.4993	40
2	2	0.5026	0.4997	40
3	1	0.5268	0.5007	60
3	2	0.5227	0.4976	60
4	1	0.5626	0.6415	80
4	2	0.5616	0.6396	80
4	3	0.5638	0.6428	80
4	4	0.5598	0.6412	80
4	5	0.5634	0.6441	80
5	1	0.6496	0.7048	80
5	2	0.646	0.7035	80
5	3	0.649	0.7065	80
5	4	0.6515	0.7045	80
5	5	0.6513	0.7052	80
6	1	0.6742	0.7362	90
6	2	0.676	0.7343	90
6	3	0.6742	0.7349	90
6	4	0.6746	0.7368	90
6	5	0.6754	0.7357	90
Yellow = Model 1, Green = Model 2, Blue = Model 3				

VII. Conclusion

This research provides some evidence that using a convolutional neural network to classify vacant homes may be a viable option, however there are serious caveats.

First, it appears that image quality plays a huge part in the results of the neural network. The general image quality using google street views was very poor. My best results required me to manually comb through every image and in total remove about 85% of the total image set. This was a major hurdle to overcome because it required manual effort and judgement, making it was difficult to get a large enough sample size to effectively train a model. The result was often underfitting the model, which resulted in the training accuracy being less than the validation accuracy. Also, the major hope from this research was to make a model that could classify images of homes in real time, but since trees and other objects would be in the image of the façade of the home, it is difficult to imagine a practical application of this technology.

The second major problem is that by manually parsing out the images in the image set, I am introducing bias. For instance, I removed images where a large portion of the image contained a tree, but perhaps the existence of trees in the image is an important feature in the classification of these homes. Even though the accuracy of the model increased, it is impossible to say if I did not subconsciously remove images in another pattern.

The final problem is that due to the surge in GPU prices, training a neural network like this can be costly. While spot pricing on AWS made it possible to do this at a reasonable cost, if this model was to be used in a production environment the cost of computation may be greatly higher.

With all of those caveats, if future research was able to leverage a platform that produces better image quality for the facades of homes, this model for classifying vacant homes would be valid. For instance, if someone went door to door with a high definition camera taking pictures of the facades of homes and those were the images used for the model, I feel strongly that this model could produce strong results. Additionally, if a better image source was available and the model produced better results, you could potentially feed images from that source of all addresses within a city and create a reasonably accurate database of all of the homes within a city. This would be a valuable resource for cities and towns that have formerly been hit by the loss of industry and thus have a high percentage of vacant homes. Future research on this topic should experiment with different types of activation and optimization algorithms to see if any additional efficiencies can be gained.

References

Semuels, A. (2018). Could Baltimore's 16,000 Vacant Houses Shelter the City's Homeless?. [online] The Atlantic. Available at: https://www.theatlantic.com/business/archive/2014/10/can-homeless-people-move-into-baltimores-abandoned-houses/381647/

Branas, C., Rubin, D. and Guo, W. (2018). Vacant Properties and Violence in Neighborhoods.

Wilson, M. (2018). Foreclosures Empty Homes, and Criminals Fill Them Up. [online] Nytimes.com. Available at: https://www.nytimes.com/2011/10/15/nyregion/foreclosures-empty-homes-and-criminals-fill-them-up.html

Apgar, W. (2018). The Municipal Cost of Foreclosures: A Chicago Case Study. [online] Available at: https://www.issuelab.org/resources/1772/1772.pdf

Kingma, D. and Ba, J. (2018). Adam: A Method for Stochastic Optimization. [online] Available at: https://arxiv.org/abs/1412.6980

Agarap, A. (2018). Deep Learning using Rectified Linear Units (ReLU). [online] Available at: https://arxiv.org/pdf/1803.08375.pdf

Shi, S., Wang, Q., Xu, P. and Chu, X. (2018). Benchmarking State-of-the-Art Deep Learning Software Tools. [online] Available at: https://arxiv.org/pdf/1608.07249.pdf

Krizhevsky, A., Sutskever, I. and Hinton, G. (2018). ImageNet Classification with Deep Convolutional Neural Networks. [online] Papers.nips.cc. Available at: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

Goodfellow, I., Bulatov, Y., Ibarz, J., Arnoud, S. and Shet, V. (2018). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. [online] Arxiv.org. Available at: https://arxiv.org/abs/1312.6082

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2018). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. [online] Available at: http://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf?utm_content=buffer79b43&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer

https://data.baltimorecity.gov/Housing-Development/Vacant-Buildings/qqcv-ihn5