Fighting Crime with Deep Learning

Muhammad Ayub[*]
ayub786@Stanford.edu

Sean Fitzgerald[†]
seantom@Stanford.edu

Crime is prevalent in metropolitan areas like Chicago and has a devastating sociological and financial impact on individuals and institutions. We show here that deep learning can be used to accurately predict at least 70% of crime in Chicago by either type or location. We expect that this predictive model will have a positive impact on many aspects of urban life, from policing to city planning. We experiment with a combination of convolutional and fully connected neural networks to efficiently handle the large amount of data necessary for predicting criminal activity.

1 Introduction

In 1995, New York City began using statistical methods to trend crime throughout the city. That year, the number of murders dropped by 1,811. Since then, cities across the United States have collected crime statistics in an effort to predict trends in criminal activity. An accurate model for predicting crime could provide significant benefit to cities, not only through targeted and more efficient policing, but also for urban planners to create city layouts that are not conducive to criminal activity. Also, home-buyers and parents could use such a model to protect family assets.

We chose to base our model on a dataset reported by the city of Chicago, which includes descriptions on all crimes occurring within city limits, beginning in 2001 and updated daily. This dataset include over 6 million crimes, much larger than the publicly available datasets from other cities in the United States. Another benefit is that Chicago also publishes myriad other datasets describing the city for public use (see Dataset and Features). Using these detailed data, we were able to create a multilayer map of Chicago which included information that we expected would be relevant to criminal activity: public parks, businesses, libraries, waterways, streets, and more. During our data collection, we interviewed a retired and a current officer; both officers agreed that crime is predictable, assuming the model captures intuition of how relevant features of a community contribute to criminal activity. We believe our models have captured some portion of this criminal intuition.

Our final deliverables are two prediction models. Both take as input a 3-dimensional dataframe with the following axes: The first and second dimensions are position, scaled from latitude and longitude. The third dimension contains layers for each of the features we have collected: elevated rail entries by station and line, interpolated life expectancy by community and year, minimum and maximum temperature by day, precipitation by day, date as a one-hot encoding of month, day, and time slot, businesses by location and category, building descriptions, waterways, major streets, public libraries, public parks, forests, school zones, and seven socioeconomic indicators assembled by the United States Census Bureau. The first deliverable model processes this input information with a Convolutional Neural Network (CNN) and a few fully connected layers to output a regression that predicts amount of crime for that location, date, and time slot. The second deliverable model processes this input information with only a CNN to output a regression that predicts the category of crime for that location, date, and time slot.

2 Related work

There exist several papers documenting machine learning approaches to predict and classify crime. In particular, methods such as Naive Bayes, Random Forests, and KNNs have been tried on a San

[*] https://www.linkedin.com/in/muhammadaliayub12/

[†] www.SeanTFitzgerald.com

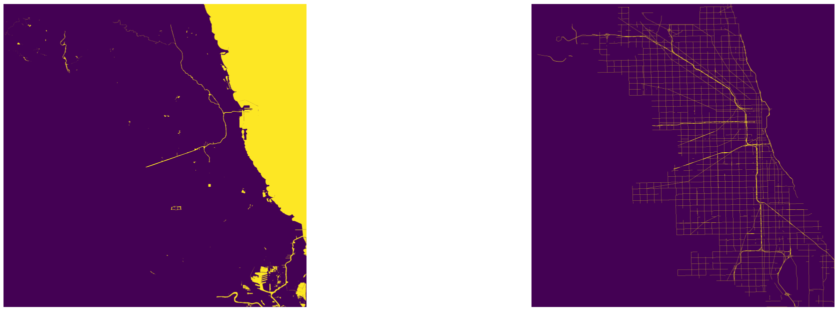
CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Francisco crime dataset [1]. Success on non-image data amounts to 20-40% accuracy. The types of data used here include crime category, latitude and longitude, and police district. Still others have used mobile data and demographic information to better predict crime in other cities with datasets (London) [4]. Other researchers have used deep learning methods on the Chicago crime dataset (as we have) to predict crime [2 , 3]. Stec & Klabjan [3] simplify the 30+ categories of crime to about 5-10 categories and are able to classify with 70% accuracy using CNNs. Still, a more powerful approach has been shown by utilizing Google Earth images in addition to the spatiotemporal datasets available on Chicago to train a CNN [2]. Here, Kang & Kang achieve a similar accuracy of 70% but with more categories.

Some of the key differentiating characteristics of our experiment have to do with the problems we explore and our generated datasets. We generate our own dataset similar to [3] with multi-layer pixelated images of Chicago but categorize on all 30+ categories instead of binning them. Additionally, we do not utilize Google Earth images to augment our dataset (unlike [2]) and therefore do not benefit from local architecture/business images. Additionally, in terms of crime volume prediction, we train on data with longer time horizons (9+ months).

3 Dataset and Features

Chicago publishes many datasets describing community geography and history, many of which might be a useful crime indicator. We collected the raw datasets ([8] - [20]) and created a data processing pipeline to format them for our neural networks. Initially, our data processing pipeline consisted of rendering every input feature as a layer in a 256 × 256 map of Chicago for every one-hour time slot from 2001 to 2018. These 256 × 256 maps of Chicago were split into 64 × 64 districts for input to a CNN. This method took days to process and, once saved, milliseconds to load. It was used to train our crime classification CNN model. The input images had 28 layers with such an approach and to add more layers required an efficient dynamic data processing pipeline.



(a) Chicago Waterways

(b) Chicago Major Streets

Figure 1: Two example renderings from our data

The first stage of our data processing pipeline optimized the size of our datasets. This primarily involved stripping all unnecessary information, which significantly reduced the dataset loading time and allowed us to more easily manipulate the tables we had collected. The second stage of our pipeline optimized lookup time by transforming the information into custom data structures. Our data generally varied in only two dimensions: time and location. For features that varied only with location between 2001 and 2018 (e.g. waterways, forests, public parks, etc.), we rendered the geographic information to create a 2D map of that feature for the city of Chicago (2048 × 2048, Figure 1). For features that varied only with time for the city of Chicago (e.g. date, weather, etc.), we created lookup arrays that could quickly provide the required parameter with very few calculations. The most difficult features to process (but arguably the most valuable features), were those which varied with both time and location (e.g. public transportation use, estimated life expectancy, and crimes). These were stored according to their sparsest dimension: Public transportation and crimes, sparsest by location (rail stations), were stored as time-based arrays with a list of occurrences (stations or crimes with associated number of entries and crime category, respectively). Estimated life expectancy was sparsest by time because it varied no more than yearly. Therefore, we rendered maps of Chicago life expectancy for each year from 2001 to 2018. The processed datasets were loaded and used to create 64 × 64 windows into our 2048 × 2048 maps of Chicago. At this scale, 64 × 64 is roughly the size of a city block, which we consider precise enough to be useful for police officer and city planners, but large enough to output acceptable model accuracies. Unfortunately, if we chose a 64×64 window uniformly at random by location and date , there was between an 80 to 90% probability that the window included no crime. To fix this, we chose 64×64 windows uniformly at random by crime, not location and date. The window was then translated a random amount such that the chosen crime no longer occurred at exactly the center of the window. Dev and Test sets were chosen first and saved before training. Mini-batch training sets were chosen and created as necessary during training. 64×64 windows within sets were allowed to overlap. However, no window overlap was allowed between Training, Dev, and Test sets. This ensured independence between datasets and made it more likely that we would notice high variance in the model.

4 Methods

We utilized two baseline methods and two complex, more powerful methods for predicting crime in Chicago. One major challenge we faced with predicting crime is that there exists no known Bayes error against which to compare our results. To reduce risk, we performed phone interviews with a retired New York City Police officer and a Washington DC Metropolitan Police officer. Both expressed confidence that crime is highly predictable. However, they also explained that predicting crime requires an intuitive knowledge of the residents, businesses, and geography of a community. That is why, for example, many police precincts adopt the "beat" model where officers are assigned to a specific neighborhood to develop an intuition for where and what types of crime can occur in that community.

We used two simple models to prove that it was possible to model at least some part of that intuition mathematically. We collected fourteen 1-dimensional parameters and trained both a support vector machine, a shallow NN, and a 10-layer fully-connected neural network (FCNN) to predict the type of crime that had occurred, given the location, weather, and limited information about the surrounding community: $\forall^{C \text{ Crime Categories}} \Pr(C|\text{crime had occurred and community information})$. These models validated that our datasets could provide at least some intuition about criminal activity in a Chicago community.The SVM performed at around 34% comparable to the 1-2 layer Keras network. The FCNN also had similar numbers.

4.1 Architecture

After validating our project and datasets, we built two models which, together, could predict location and type of crime in Chicago. The first model is a CNN, followed by a FCNN which outputted the amount of crimes, given a two-hour window and a 64x64 multi-layered map about the size of a Chicago city block. One principle that helped guide our network design was that there should not be large changes in the size of each layer. We used the following CNN architecture:

$$\text{CONV}(4\text{X}4 \text{ FILTER, SAME}) \times 64 \rightarrow \text{MAXPOOL}(8\text{X}8 \text{ FILTER}) \rightarrow$$
$$\text{CONV}(4\text{X}4 \text{ FILTER, SAME}) \times 64 \rightarrow \text{CONV}(2\text{X}2 \text{ FILTER, SAME}) \times 64 \rightarrow$$
$$\text{MAXPOOL}(2\text{X}2 \text{ FILTER}) \rightarrow \text{CONV}(4\text{X}4 \text{ FILTER, SAME PADDING}) \times 64$$

Our goal was to slowly reduce the height and width of each intermediate layer. The output was unwound into a 1D array and concatenated with our remaining 1D data (weather, date, and time). This was fed into the following FCNN:

$$\text{RELU}(1024 \text{ HIDDEN UNITS}) \rightarrow \text{RELU}(512) \rightarrow \text{RELU}(256)$$
$$\rightarrow \text{RELU}(128) \rightarrow \text{LINEAR OR SOFTMAX}(34)$$

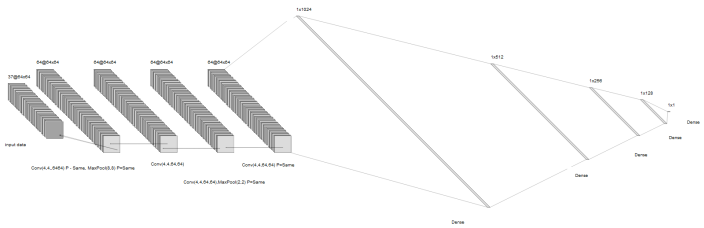


Figure 2: Crime amount regression model

The second model is a CNN which uses the following architecture: CONV(10x10 filter, s = 2, valid) × 3 → AVGPOOL(3x3, s =1, SAME) → CONV(6x6 filter, s = 2, valid)×3 → MAXPOOL(3x3 , s = 1, Valid) → SOFTMAX(34)

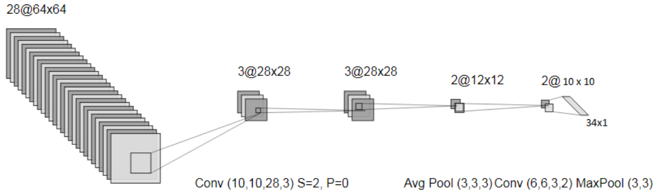


Figure 3: Crime category classification model

This model started roughly based off of the work done in (Stec & Klabjan 2018) on the CNN they trained. However their model used 1/2 layers to classify 6 categories and since our task required classification of 30+ classes, we added more layers. We decided to convolve only 3 filters on the first layer for faster training. Later on we switched to 10 filters. The reason why we chose an averaging filter (instead of max/min pooling) first was because we wanted information to flow from the first layer to later layers in the network. After convolving a second time, we opted to use max pooling arbitrarily.

4.2 Training and Results

4.2.1 Crime Volume Prediction

We experimented with two output schemes: the linear output performed a linear regression on the last layer of the FCNN. This used a mean-squared error loss function, averaged over the mini-batch and was, at best, 66% accurate. The softmax output performed a softmax regression on the last layer of the FCNN, choosing one of 20 bins for the number of crimes which had occurred. This used a logistic loss function and was, at best, 71% accurate. Although the binning was more accurate, we consider the linear regression to be a better method, since it allows the network to predict greater than the perceived maximum crimes for a given two-hour time slot and city block. Therefore, we put more effort into training and honing hyperparameters on the linear output. The most important aspect of training this network was choosing the correct learning rate. If we selected the incorrect learning rate early in training, our accuracy increased only slowly and sporadically. We chose an initially high learning rate, then manually decreased it when the accuracy plateaued. This reliably increased model accuracy above 50%. The next most sensitive hyper-parameter was mini-batch size. With the model running on an instance of p3.16xlarge, Amazon's largest computation EC2 instance, we were able to train a mini-batch size of around 3000. After expending all computing credits, we trained on our local computers. This required a reduction in mini-batch size to 512, which increased noise for cost and accuracy metrics. Model progress was therefore more difficult to track. Both models used Adam optimization, which converged much faster than regular gradient descent.

4.2.2 Crime Classification

We trained with mini-batch gradient descent (MBGD) because the dataset size was too large. We used Adam to remove noise from MBGD. We did not explicitly change default values of beta1 (.9) and beta2 (.999). The cost function was the categorical cross entropy because of the multi label classification. To train the crime classification CNN, there were hyper parameters including the depth of filter layers, the encoding of the output, learning rate, and the mini-batch size. We started with a learning rate of .008, and a minibatch size of 200. Also, we initially thought that 1 hot was the correct way to encode the outputs. We later transitioned to 3 hot encoding while keeping the expected value of the outputs the same.

In terms of results, the first several models achieved +99% accuracy. After analysis, we saw that class 32 was significantly more common and resorted to F1 scores on each class since both precision is important (we don't want other crimes being mistaken as the crime being classified) and recall is important (we don't want the crime being classified to be misclassified). We wanted to discuss models 5a, 5b, 5c, and 6. In the end, the best models were 5a and 5b. Really, after training 5a for 830 epochs on 3 hot encoding, 5b bettered the existing model by training 5a's last trained parameters with learning rate .003 and new minibatch size of 400. These were the optimal hyper parameters.

We tried 1 hot encodings, 3 hot encodings, and 5 hot encodings and 3 was optimal. 5c had a 76.64% and 72.15% accuracy for dev, and test sets. And f1 scores were at .075 and .073 for dev and test sets. 5b was comparable with (77.50 Acc-Dev, .075 F1-Dev) , (70.84 Acc-Test, .072 F1-Test), and (75.12 Acc-Train, .074 F1-Train). Essentially, the F1 scores were poor because even with 3 hot encoding, certain classes were more dominant in the dataset. 5c was the 5 hot encoding with less performance. We resorted to upsampling/downsampling (USDS) for model 6 and saw (38.72 Acc-Dev, .049 F1-Dev) , (39.51 Acc-Test, .05 F1-Test), and (37.54 Acc-Train, .048 F1-Train). We learned that we would have to do more USDS to equalize the data even more. Again, these numbers are approximate since our 3 hot encoding is at most 3 values (it could be 2 or 1 if there was no 2nd or 3rd most common class in an output image). In the future, we would train to reduce the high bias and see if the variance stays small like it has.

Something important to disclose is that we tuned multiple hyper parameters together due to shortage on time. Hence, we are not sure a certain value of a hyper-parameter is strictly better than another. Also, please note we averaged F1 across all classes.

4.3 Limitations/Challenges

Some of the major hurdles we encountered were the data engineering portion of the project. It took us four iterations in total to generate the CNN data. We tried using MapBox API, Google Maps API (with a Python backend), the PIL library, and finally GIS library Shapely to generate a dynamic and static dataset. We could have performed more hyperparameter tuning if we were provided a clean dataset.

Another limitation was the bias we introduced into the model. As the team made decisions on what data was important and what data wasn't, we were implicitly applying our own biases. For example, we decided that liquor stores were a good indicator of crime, as well as socioeconomic status. This model has the potential to store and predict with this implicit bias, as well as the bias of the police officers recording crimes. This is something we would like to explore in future work.

Lastly, there were huge class imbalances in the dataset (i.e. Theft is much more common than Arson). Results from slight up-sampling/down sampling were not as promising as we had hoped. More up-sampling/down sampling is required to get equal class representation.

5 Conclusion/Future Work

We conclude that crime is highly predictable. For crime classification, the highest performing algorithms were trained on 3 hot encoded vectors with sparse data removed. Adding layers to CNN network in [3] improved capability to predict more classes, but only slightly, due to class imbalances. Lower learning rates with larger minibatches saw better results.

With more time, we would like to train models on other major cities with our methodology. We believe that there are statistical commonalities that contribute to crime between cities. Transfer learning can be used to build accurate models for cities with fewer reported crimes. Class imbalances were and continue to be a problem and we have briefly tried upsampling/downsampling data and would like to expand on that to achieve better results. Another very interesting aspect about our work is the ability to train on very small regions in a city. While we did not explore this in our work, we would like to compare crime indicators in a specific area to general crime indicators. Finally, we expect that object detection algorithms could be utilized on large city maps to very specifically pinpoint crimes before they happen.

In short, there seems to be a plethora of work remaining to understand crime with deep learning. With our work, we have provided two datasets for others to experiment and train on. One is statically generated and the other is dynamically generated (on Github/Sean): https://drive.google.com/drive/folders/1LOyrqORS4Zszm4n62siat888IIgpx4rx?usp=sharing.

6 Contributions

Sean data engineered and built the ad-hoc dataset that was used to generate mini-batches at runtime. Sean also designed and trained the FCNN on the tabular data architecture as well as the crime volume prediction model. Ali data engineered the static dataset and trained the support vector machine and smaller NNs at the beginning and different CNNs for the multi-class crime model. Both partners worked on the paper, poster, milestone, and proposal.

We would like to thank Daniel Kunin for his continued support and direction of the crime classification problem. He encouraged us to train CNNs and provided critical feedback to help the project along.

Github Repo: https://github.com/mrplants/crime_prediction

References

[1] Shama, N. (2017). A machine learning approach to predict crime using time and location data (Doctoral dissertation, BRAC University).

[2] Kang, H. W., & Kang, H. B. (2017). Prediction of crime occurrence from multi-modal data using deep learning. PloS one, 12(4), e0176244.

[3] Stec, A., & Klabjan, D. (2018). Forecasting Crime with Deep Learning. arXiv preprint arXiv:1806.01486.

[4] Bogomolov, A., Lepri, B., Staiano, J., Oliver, N., Pianesi, F., & Pentland, A. (2014, November). Once upon a crime: towards crime prediction from demographics and mobile data. In Proceedings of the 16th international conference on multimodal interaction (pp. 427-434). ACM.

[5] Crawford, K. (2016). Artificial intelligence's white guy problem. The New York Times, 25.

[6] Duan, L., Hu, T., Cheng, E., Zhu, J., & Gao, C. (2017). Deep Convolutional Neural Networks for Spatiotemporal Crime Prediction. In 2017 International Conference on Information and Knowledge Engineering (IKE) (pp. 61-67).

[7] Convolutional Neural Networks **https://www.coursera.org/learn/convolutional-neural-networks**

[8] Chicago Police Department. (2018). Crimes - 2001 to present. Retrieved from https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2

[9] U.S. Census Bureau. (2012). Census Data - Selected socioeconomic indicators in Chicago, 2008 – 2012. Retrieved from https://data.cityofchicago.org/Health-Human-Services/Census-Data-Selected-socioeconomic-indicators-in-C/kn9c-c2s2

[10] City of Chicago. (2018). Business Licenses. Retrieved from https://data.cityofchicago.org/Community-Economic-Development/Business-Licenses/r5kz-chrr

[11] Illinois Department of Public Health (IDPH). (2010). Public Health Statistics- Life Expectancy By Community Area. Retrieved from https://data.cityofchicago.org/Health-Human-Services/Public-Health-Statistics-Life-Expectancy-By-Commun/qjr3-bm53

[12] Chicago Transit Authority. (2018). CTA - Ridership - 'L' Station Entries - Daily Totals. Retrieved from https://data.cityofchicago.org/Transportation/CTA-Ridership-L-Station-Entries-Daily-Totals/5neh-572f

[13] NOAA. (2018). Climate Data Online Search. Retrieved from https://www.ncdc.noaa.gov/cdo-web/search

[14] Chicago Park District. (2018). Parks - Locations. Retrieved from https://data.cityofchicago.org/Parks-Recreation/Parks-Locations-deprecated-November-2016-/wwy2-k7b3

[15] Chicago Public Library. (2018). Libraries - Locations, Hours and Contact Information. Retrieved from https://data.cityofchicago.org/Education/Libraries-Locations-Hours-and-Contact-Information/x8fc-8rcq

[16] City of Chicago. (2018). Waterways. Retrieved from https://data.cityofchicago.org/Parks-Recreation/Waterways/eg9f-z3t6

[17] City of Chicago. (2018). Major Streets. Retrieved from https://data.cityofchicago.org/Education/Libraries-Locations-Hours-and-Contact-Information/x8fc-8rcq

[18] City of Chicago. (2018). School Grounds. Retrieved from https://data.cityofchicago.org/Education/Libraries-Locations-Hours-and-Contact-Information/x8fc-8rcq

[19] City of Chicago. (2018). Forest Preserves. Retrieved from https://data.cityofchicago.org/Education/Libraries-Locations-Hours-and-Contact-Information/x8fc-8rcq

[20] City of Chicago. (2018). Boundaries - Buildings. Retrieved from https://data.cityofchicago.org/Education/Libraries-Locations-Hours-and-Contact-Information/x8fc-8rcq