Modelling the Risk of Traffic Accidents in New York City

Springboard Capstone 2 Final Report

Gene Hopping

January 2020

Problem Statement

Road traffic accidents resulted in the death of over 37,000 people in the US in 2017 alone.[1] The cost associated with traffic accidents in 2010 (from a report published in 2015) was $242 billion, or 1.6% of the gross domestic product for that year.[2] Companies that make their living in transportation would benefit from information regarding where accidents are more likely to occur, so they can avoid those areas and decrease their risk of accidents. Rideshare companies, such as Uber and Lyft, rely on getting from A to B quickly and safely in order to gain customer satisfaction. Uber provide insurance for its drivers[3], bearing the costs of accidents so is at risk for more than loss of customer satisfaction if involved in an accident. This project will focus on the temporo-spatial prediction of the risk of accident throughout New York city. Rideshare companies could use information like this as an additional layer of data when generating routes to avoid areas of increased risk of accident.

The data we will use is the NYPD Motor Vehicle Collisions Data provided by the City of New York.

- https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95

The data is provided as a CSV and consists of the time of accident, the geographical coordinates, and other information such as cause of accident, vehicles involved, fatalities, etc. Data will be cleaned to ensure each entry contains valid coordinates and timestamps. At this stage we plan to represent the data as a raster to determine areas of increased risk of accident, at specific times during the day. A 2D matrix visualized over time is basically an image recognition problem, so we plan to develop a convolutional LSTM deep learning model. There is precedence for an approach such as this.[4]

Data Wrangling

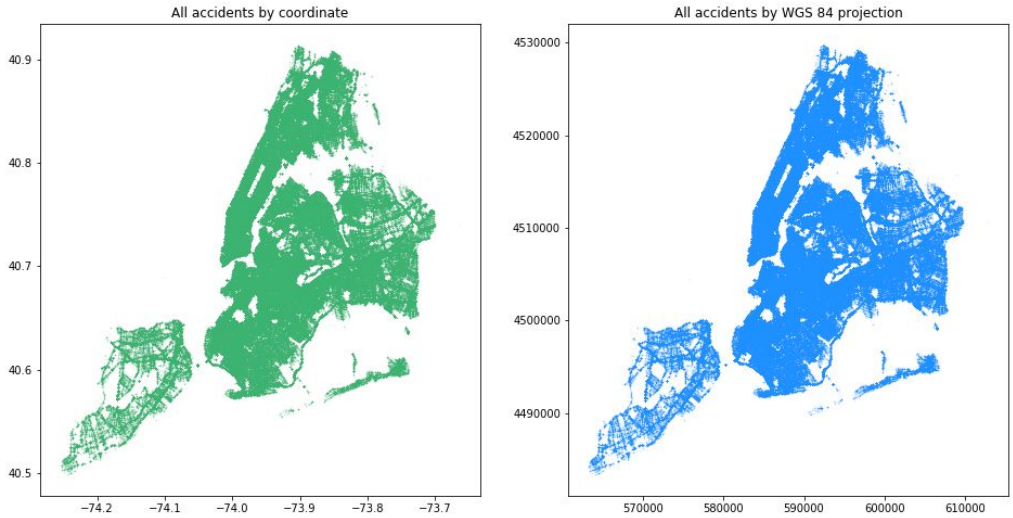
We decided to pose this as a classification problem, asking the question is there an increased risk of accident in this area, at this hour, on this day'. The dataset contains 1.58 MM rows of accidents, including date, time, longitude and latitude, as well as the number of vehicles involved, contributing factors, number of injuries, etc. In a previous project, we were broadly interested in all accidents that occurred within the city limits of New York city, so were not concerned if there were missing location values. For this project, the location of the accident and the time is important as we are attempting to build a model to predict the location of increased risk of an accident at a selected time and day.

The data were imported into a pandas dataframe from a csv file downloaded via the link above. Column titles were converted to lowercase, and any spaces in the titles were replaced

[1] https://www.nhtsa.gov/press-releases/us-dot-announces-2017-roadway-fatalities-down
[2] https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812013
[3] https://www.uber.com/drive/insurance/
[4] https://arxiv.org/pdf/1710.09543.pdf

with underscores to allow chaining. A new datetime column was created and set as the index for easy indexing using the pandas .loc accessor. Each line in the dataset represented the time and location of an accident reported to the NYPD. Upon initial inspection of the data, the number of rows of datetime data did not match the number of rows of longitude and latitude. Closer inspection revealed over 185 thousand NaN instances, and over 900 values of 0 for longitude and latitude. 0 longitude and latitude is on the equator, south of Ghana. Since the date and location of an accident is of paramount importance to this work, all NaN values were first converted to 0 and dropped together with the existing 0 values. The removal of these entries resulted in the same number of datetime values as longitude and latitude.

All accidents were plotted using their latitude and longitude values. This revealed multiple outlying points, that were clearly out of the New York City area. Constraints on the longitude and latitude were incrementally tightened, until by visual inspection an adequate map of New York City was obtained. Next, we converted longitude and latitude into the World Geodetic System, which is the standard for projecting the curved surface of the earth onto a planar surface, like a map. This is important, since ground distance measurements will be more accurate.

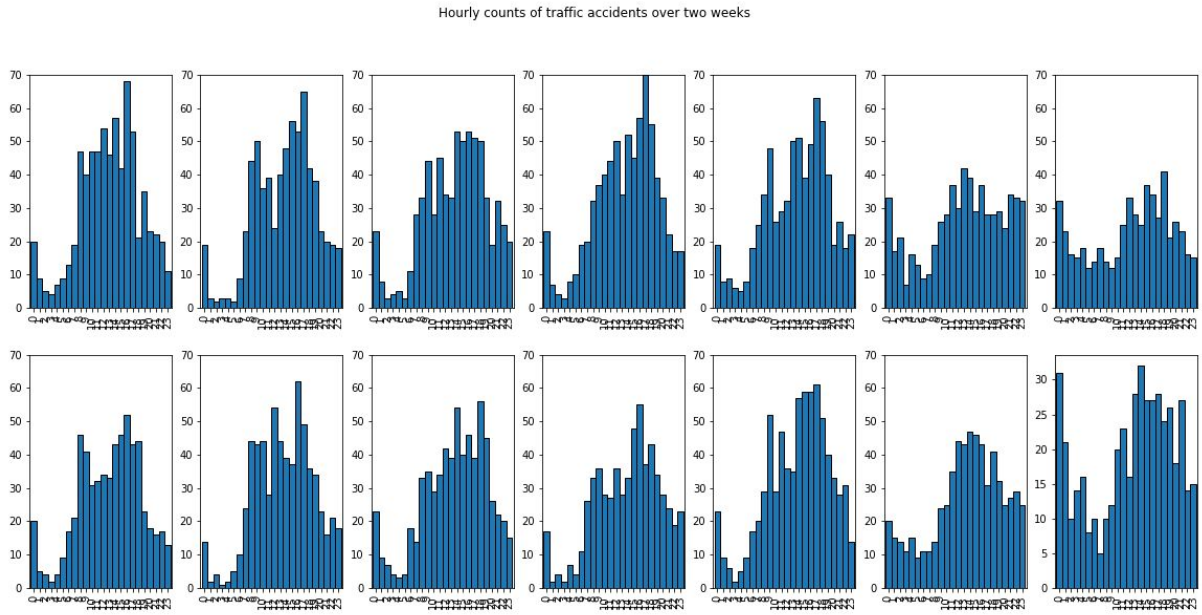


Coordinate (green) and WGS projection (blue) displayed side-by-side (above). The differences are small but would account for tens of meters in ground measurement. The east coast of the US falls into the zone 18T of the Universal Transverse Mercator coordinate system. For more information see this reference.[5] Applying this converts degrees latitude and longitude to meters easting and northing, and is a more accurate measure of ground distance. These values will be used for the remainder of this work.

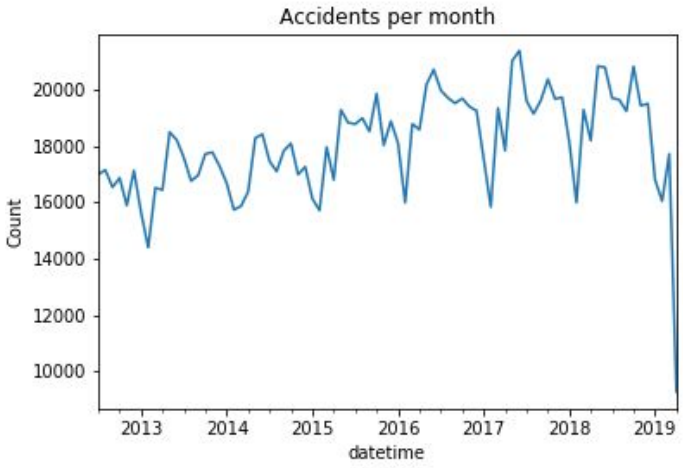
5 https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system#Locating_a_position_using_UTM_coordinates

EDA

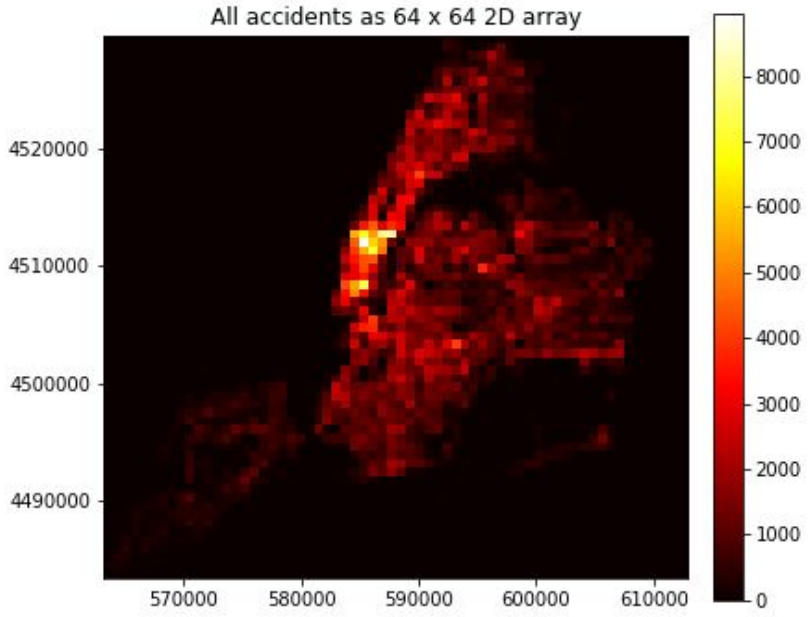
To get a feel for the consistency of the data, we first grouped accidents by hour and plotted two weeks worth of histograms. This revealed that there appears to be a bimodal distribution during the day, peaking at approximately 8am and 4pm, which is consistent with normal peak hour traffic.



Histograms plotted over two weeks, beginning with Monday, showing the number of accidents per hour (above). The bimodal distribution is apparent for many and consistently the largest number of accidents appear to be in the afternoon. There is also a period in the early hours of the morning, where there are less accidents, presumably because there is less traffic on the road. We then plotted the total number of accidents per month for the entire length of the dataset.



The number of accidents appears to be seasonal, starting lowest at the beginning of the year, peaking during summer and tailing off toward the end of the year. However, we are not just interested in predicting when the accidents are going to occur. We are interested in when and *where* the accidents are going to occur. To do this we will aggregate the accidents by hour and by location. As mentioned earlier, we will present this in a 2D array. Initially, we chose to discretize the data into a 64 x 64 grid. This returned a more pixelated view of the data, however many features of the city are still discernible.

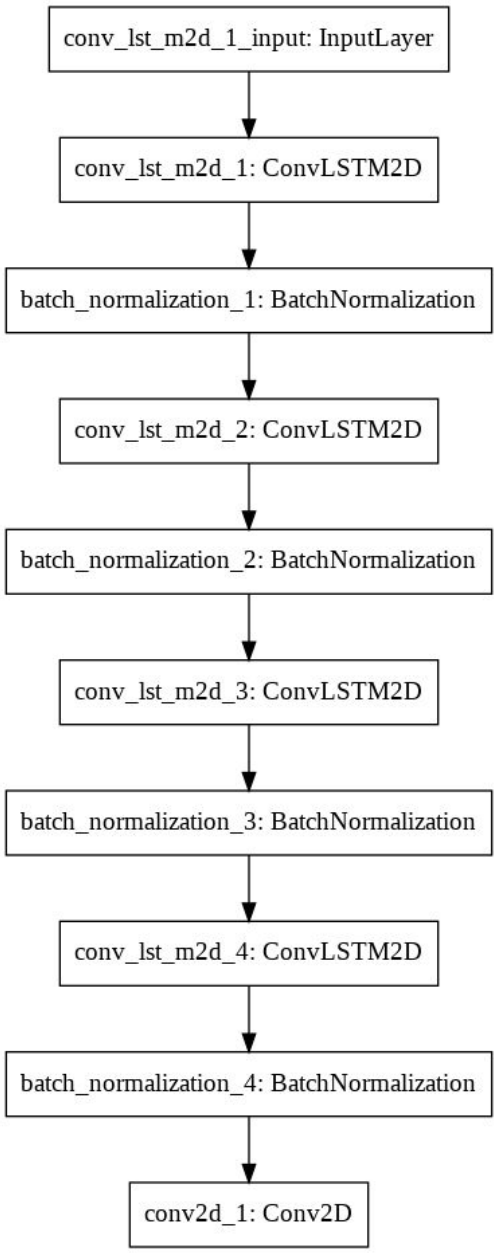


All accidents plotted in 64 x 64 bins (above). Grouping into 64 x 64 bins like this makes the width of each bin approximately 712 x 765 meters. While not the most practical distance for navigating around accidents, it will allow us to demonstrate the ability of our model to predict traffic accidents. To allow computation in a reasonable amount of time, we decided to look at the last 6 months of accident data, which for this data set consisted of the dates 16 December 2018 to April 16 2019. The data was split 70:20:10 into train:test:validate sets. The model as refined using the train and test sets, the validate set was a hold-out set and applied only to the trained model.

Model Building

We have chosen to represent the model as a 2D array of accidents. The two dimensions are defined by regular intervals of distance with the number of accidents summed in each bin. This gives us a somewhat pixelated view of accidents in NY city, but it's suitable to see if this approach works. Data arranged like this is can be thought of as a picture, or when considering

multiple time slices, frames of a movie. Given this, we can treat it like a ‘predict the next frame of a movie’ problem. This is an interesting problem, because it is a time series problem and also an image problem. Time series problems often benefit using Long Short Term Memory (LSTM) models. These are recurrent neural networks that have the ability to store previous information to apply to the current frame. This is very useful for time series problems where regular patterns over time may occur. Because ours is also an image problem, we could benefit from a Convolutional Neural Network (CNN). CNNs use a kernel function to extract important features from an image. We performed our deep learning using Keras, a high-level API that runs on top of TensorFlow. This package includes the convLSTM2D layer, which is similar to a LSTM layer but the input layer and transformations are both convolutional.

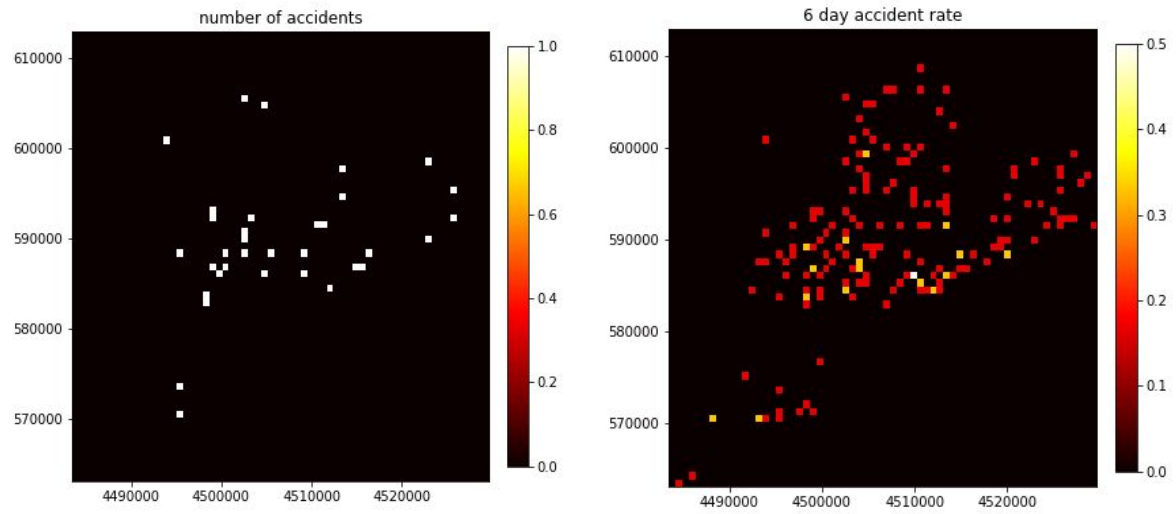


Plot of the deep learning convLSTM model we used during this work (above). After the first convolutional input, we applied 4 convLSTM2D layers with relu activation and with normalization between each layer. The final conv2D layer collapses the tensor back to a 2D image using sigmoid activation and the binary cross entropy loss function, which provides our prediction as a 64 x 64 2D array of probabilities of an accident occurring.

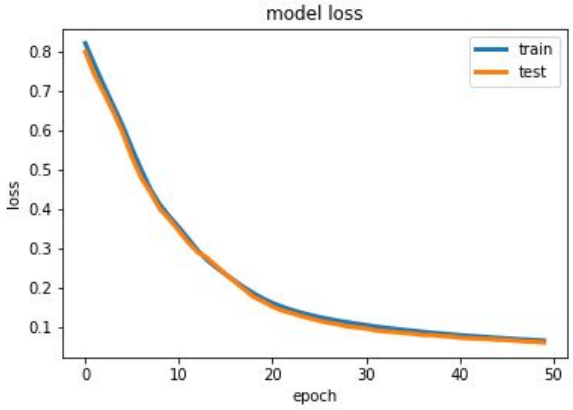
The input shape of the data required for this layer is a 5D tensor with the shape (samples, time, rows, columns, channels) which in our case is (samples, time, 64, 64, 1). Since we have a 64 x 64 matrix and it is a binary classification problem, so 1 channel. Time in this case is time steps - we have to choose how many time points we want to use to predict the hour of interest. For this work we chose to use the preceding 6 hours to predict the 7th. We chose 64 filters with a 5 x 5 kernel.

Feature Engineering

Predicting traffic accidents directly is an extremely difficult endeavor due to the incredibly complex factors involved. It would be next to impossible to include all the factors that lead to an accident, including but not limited to driver distraction, the weather conditions, traffic conditions, and road conditions. Instead of predicting accidents directly, we'll predict accident risk. Currently our accident data is temporo-spatially arranged in a tensor by hour. Each grid represents the number of accidents that occured in that grid at that hour of the day. To convert the accidents to accident risk, we find the average number of accidents per cell, over the last x number of days. For this work, we chose to find the average number of accidents over 6 days. For example, if one grid cell for the last 6 days at 8 am contained the following number of accidents: 3, 0, 6, 2, 4, 1, then the accident risk for 8 am on the day of interest would be (3 + 0 + 6 + 2 + 4 + 1)/6 = 16/6 = 2.67 accidents / hour. This transformation was applied to all the data, so the data used in modelling were 64 x 64 arrays of accident rates, based on the accidents at the same time and grid cell over the last 6 days.



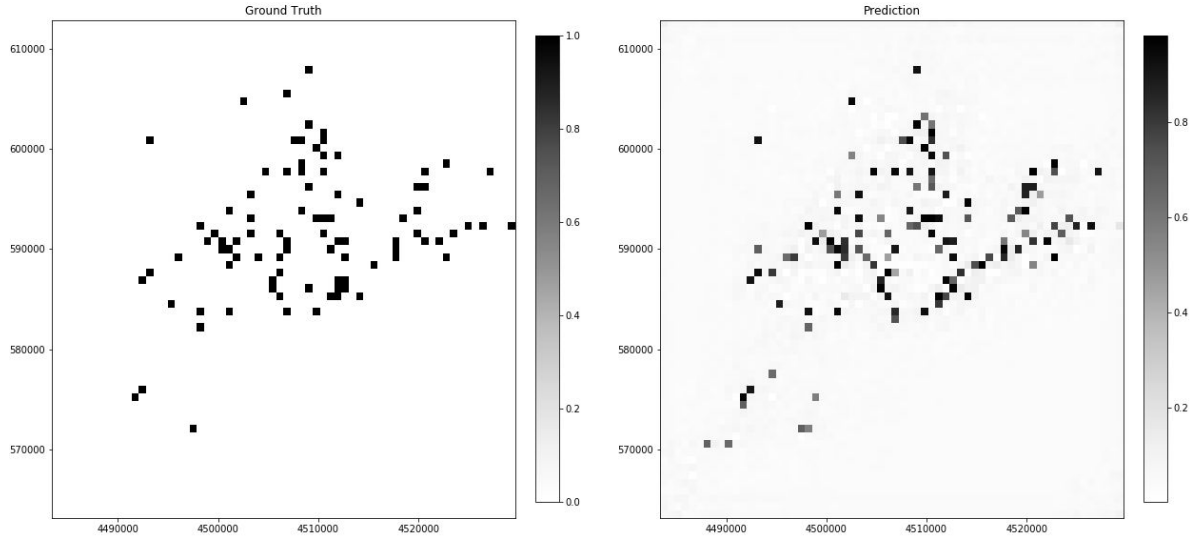
Above shows the number of accidents recorded in a single hour (left). The accident rate for the same hour was calculated using the same hour of the day, over the previous 6 days (right). The accident rate was much more informative for building our model than the accident numbers alone. Together, the final model contained 2,876,737 trainable parameters and when run in a Google Colab notebook using a GPU for 50 epochs, the total training time was approximately 6 hours.



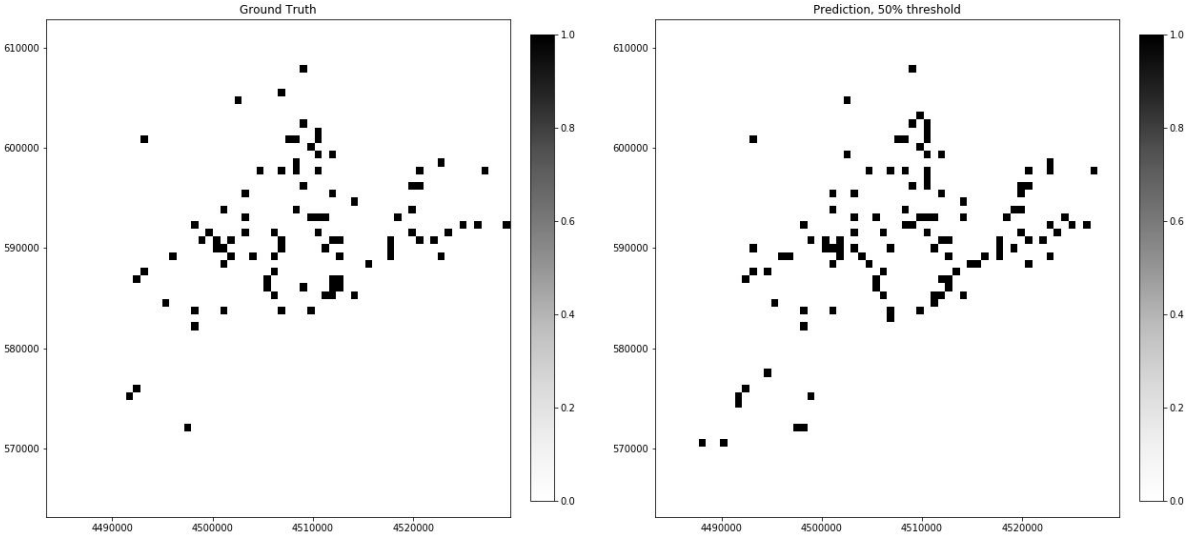
The model appear to converge by the 50th epoch, so we did not extend the training time past this (above).

Assessing the Model

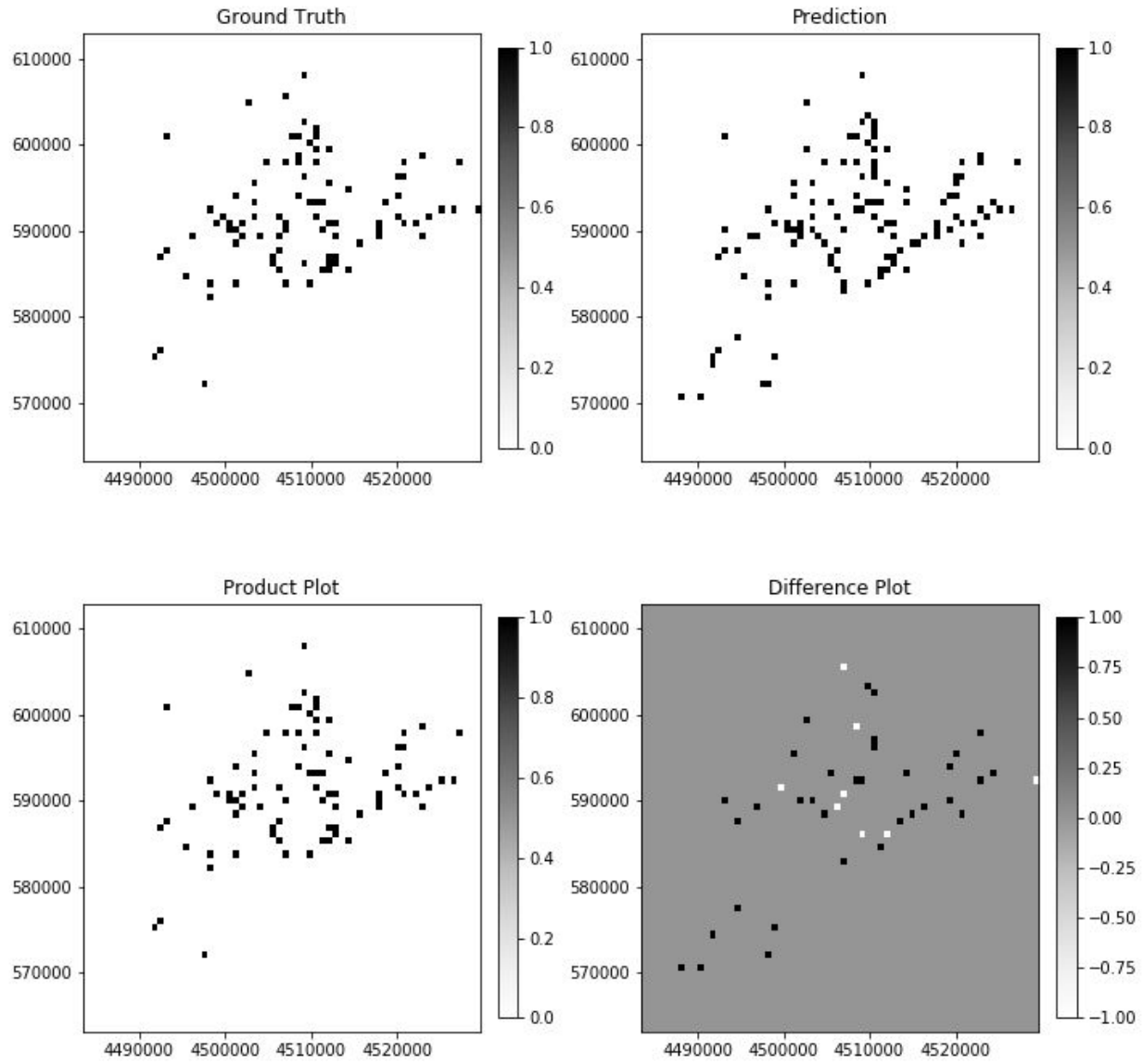
While Keras provides many built-in analyses, for simplicity we calculated our own confusion matrix. As mentioned earlier the model outputs a 64 x 64 array of *probabilities* of the risk of an accident.



Ground truth (left) and prediction (right) of the probability of the location of an accident occurring in New York city (above). Because we want a binary classification and not a probability, we need to stratify the results into 0 or 1. We do this using the predict_classes function, which does the stratification for us. Any probability 0.5 or above becomes a 1, the rest 0.



Ground truth (left) and prediction (right) of the location of a traffic accident in New York City (above). Each prediction and ground truth 64 x 64 array was converted into a pandas dataframe which allowed element-wise multiplication and subtraction. When the two were multiplied together, since they were both binary matrices, only correctly identified, or true positive values would remain. When we subtract the ground truth matrix from the prediction, true positive values become zero (1 -1), incorrectly predicted accidents or false positive values become 1 (1 - 0) and missed predictions or false negative values are -1 (0 - 1). True negative values can then be calculated by subtracting the sum of the true positive, false positive and false negative values from the size of the matrix (64 x 64 = 4096).

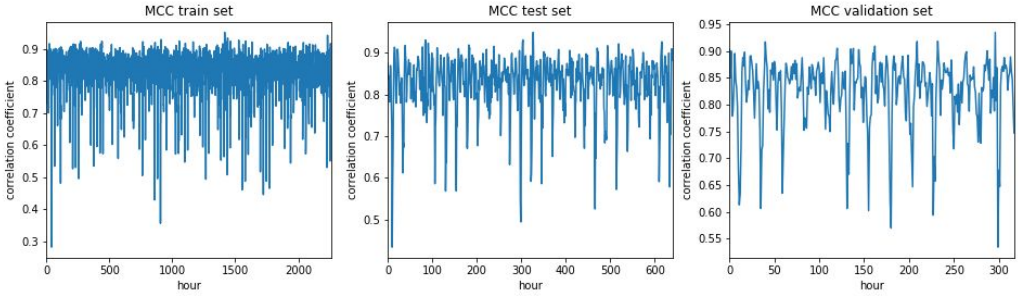


Above are four images showing the ground truth for increased risk of accidents (left top), the predicted matrix (right top), the product matrix (bottom left) which displays the correctly predicted cells, and the difference plot (bottom right) which returns -1 (white) if a cell was falsely predicted or +1 (black) if a cell was not predicted, but existed in the ground truth (right). Owing to the imbalance between the two classes (vastly more negative values than positive, which is a good thing!) reporting the accuracy of the model is not a suitable metric. Instead we can use the Matthew's correlation coefficient, or phi coefficient as a measure of the quality of a binary prediction.[6]

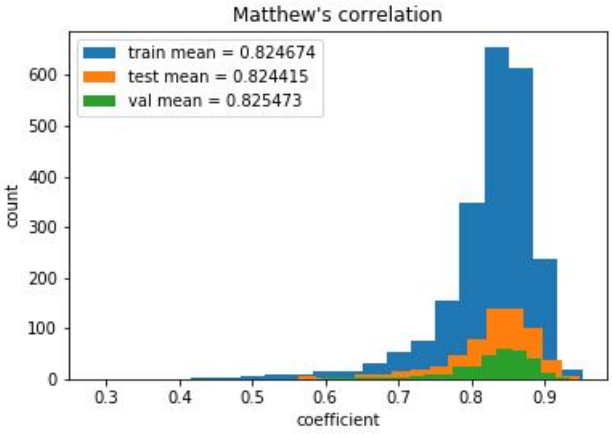
[6] https://en.wikipedia.org/wiki/Matthews_correlation_coefficient

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Where TP, TN, FP, FN = true positive, true negative, false positive and false negative respectively. Applying this formula returns a value between 1 and -1 where 1 is complete agreement between the ground truth and prediction, 0 is equivalent to random chance, and -1 indicates complete disagreement between the model and ground truth.

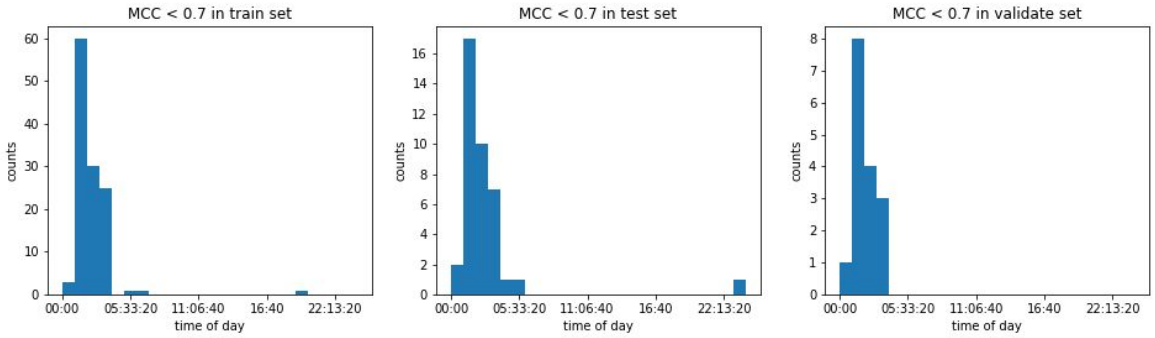


Calculating the Michael's correlation coefficient across our three sets we see generally very good scores, with the average for the train set, the test set and the validation set is 0.82, 0.82 and 0.83, respectively. It is pleasing to see the hold out set also scored similarly as the train and test set, since these data were not included in the model training at all. However, we can see from the repeating patterns of lower values in the plots above, and the negative skew in the histograms below that there is some heterogeneity in our predictions.

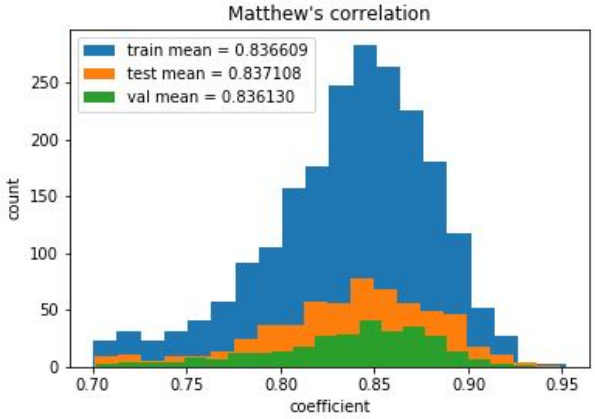


Ideally the MCC histogram would be a normal distribution around the average value. This indicates that our model is weaker at predicting the risk of accidents at certain hours of the day. If it was weak at predicting a certain region, we wouldn't see this in an acute decrease in MCC score.

First we added a datetime index back to our confusion data frames. We then set a threshold of 0.7 and filtered out the index values for hour where these occurred. Plotting histograms of these values, we see that the vast majority of values are found between 1 and 3 am.



If we recalculate the mean MCC values for our model if we exclude these data points, our mean values for the train set, test set and validation set all increase slightly to 0.84.



Conclusion

We generated a deep learning model for predicting the risk of traffic accidents in New York City. This model was based on the keras convLSTM2D architecture, which accepted 2D arrays of accident rate which were averaged over the last 6 days per hour, per grid cell.

We demonstrated that the model was predictive for future unseen data with a Michael's correlation coefficient of 0.84.

Attempts to train the same model with the raw accident counts, and not accident rates failed to converge, resulting in gross overestimation of the number and locations of accidents.

To be a truly useful model, the resolution of each grid would ideally be that of a city block. It would also be interesting to increase the time resolution to more frequent periods than 1 hour. This, of course, would dramatically increase the computation time.