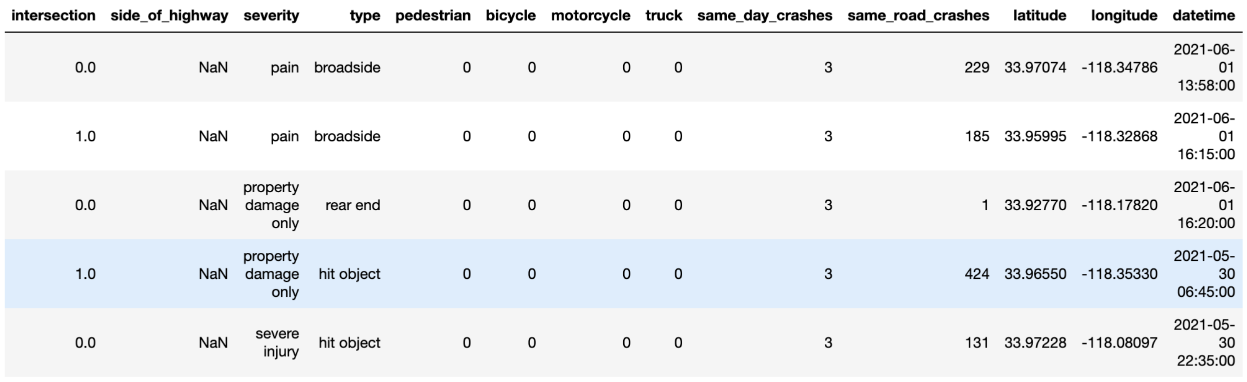
Los Angeles Traffic Collision Prediction Using Pytorch GNNs

Edward Seymour

Millions of people are on the road everyday. In fact, there are more than seven million cars registered in LA county alone. With that many cars on the road, and pedestrians and cyclists, it can be quite dangerous to commute to everyday activities like work and grocery shopping, However, some routes are more dangerous than others. In this project I created a model to return the likelihood of an accident occurring in the city of Los Angeles.

The traffic collision data for this project comes from the Los Angeles Police Department and the California Highway Patrol. Traffic volume data was obtained from the Los Angeles Department of Transportation. The traffic volume data was geocoded using the Google Geocoding API. Weather variables were obtained from the OpenWeather API. I cleaned the datasets and removed observations that were out of the time frame that included all of the data. Afterwards, I merged all of the datasets into a single dataframe. An initial look at the data revealed that most of the collisions occur near the center of the city, as can be seen from the heatmaps below.



intersection	side_of_highway	severity	type	pedestrian	bicycle	motorcycle	truck	same_day_crashes	same_road_crashes	latitude	longitude	datetime
0.0	NaN	pain	broadside	0	0	0	0	3	229	33.97074	-118.34786	2021-06-01 13:58:00
1.0	NaN	pain	broadside	0	0	0	0	3	185	33.95995	-118.32868	2021-06-01 16:15:00
0.0	NaN	property damage only	rear end	0	0	0	0	3	1	33.92770	-118.17820	2021-06-01 16:20:00
1.0	NaN	property damage only	hit object	0	0	0	0	3	424	33.96550	-118.35330	2021-05-30 06:45:00
0.0	NaN	severe injury	hit object	0	0	0	0	3	131	33.97228	-118.08097	2021-05-30 22:35:00

Figure 1: Collision Data Sample

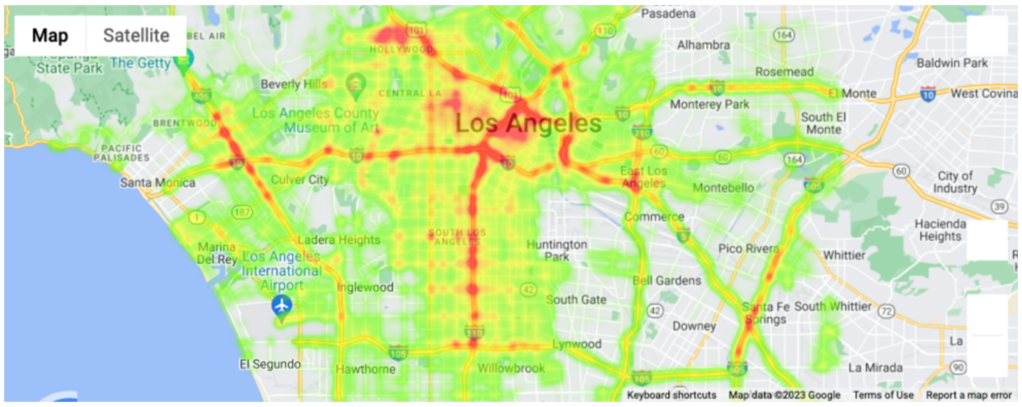


Figure 2: Heatmap of Highway Collisions

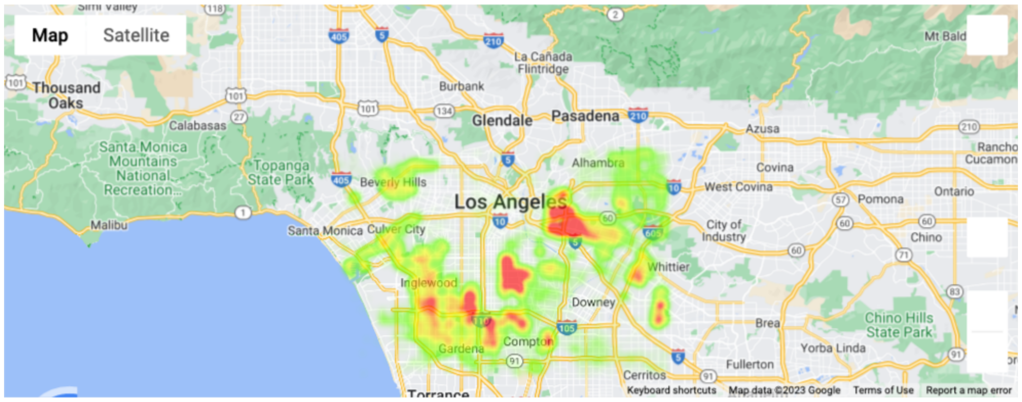
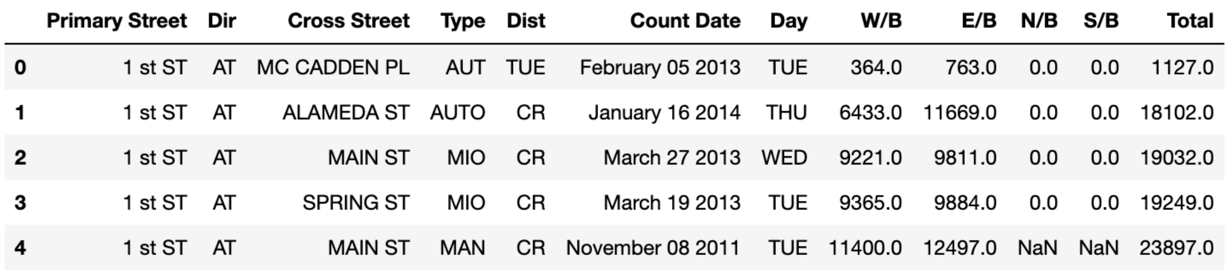


Figure 3: Heatmap of Non-Highway Collisions



	Primary Street	Dir	Cross Street	Type	Dist	Count Date	Day	W/B	E/B	N/B	S/B	Total
0	1 st ST	AT	MC CADDEN PL	AUT	TUE	February 05 2013	TUE	364.0	763.0	0.0	0.0	1127.0
1	1 st ST	AT	ALAMEDA ST	AUTO	CR	January 16 2014	THU	6433.0	11669.0	0.0	0.0	18102.0
2	1 st ST	AT	MAIN ST	MIO	CR	March 27 2013	WED	9221.0	9811.0	0.0	0.0	19032.0
3	1 st ST	AT	SPRING ST	MIO	CR	March 19 2013	TUE	9365.0	9884.0	0.0	0.0	19249.0
4	1 st ST	AT	MAIN ST	MAN	CR	November 08 2011	TUE	11400.0	12497.0	NaN	NaN	23897.0

Figure 4: Traffic Volume Sample

The data was now sufficiently cleaned, so the next step was to determine the best spatial and temporal resolution to use for modeling. As for 'best', I was looking for a resolution that would maximize accidents per timeframe, minimize size of the data, and retain usefulness. The best spatial resolution was determined to be 1 mi. x 1 mi., or a 50 x 30 grid. As for temporal resolution, the weather and collision data was already in an hourly format, so that is the frequency that I decided to use.

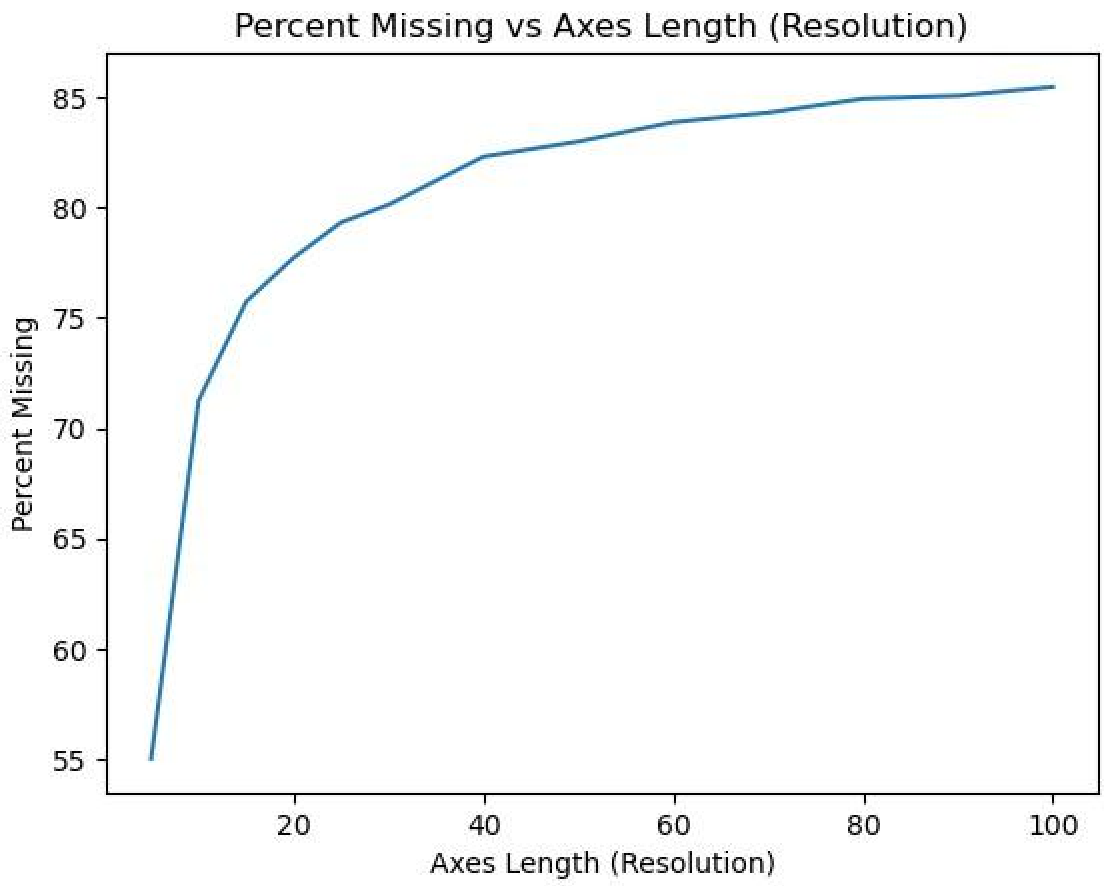


Figure 5: Spatial Resolution vs Missing Cells

After deciding on the resolutions that I was going to use, I created the n-dimensional array that would be used for in the modeling notebook. This was a fairly simple process of looping through the collision, weather, and traffic volume datasets and filling the n-dimensional array with the values. The collision layer of the n dimensional array had either a 1 or a 0 depending on whether an accident occurred in that region at that time period. I split this array in two ways, since I would be testing out two different models. The first was a single 75/25 split to retain the temporal aspect of the data so that this could be exploited during modeling. The other was an sklearn test train split of the same size. Finally, I created a windowed version, with a window of 6, of the test and train datasets that were just created.

Of the two models that I tested, the first was a temporal GNN (Graph Neural Network) using the windowed dataset created previously. This data only included the temporal progression of accidents without weather and traffic data. I used the Pytorch Geometric Temporal library to build the graph data and construct the GNN. This model did not perform as well as hoped. The precision was fairly steady throughout training, and the BCE loss function only slightly decreases, as can be seen in the following graph.

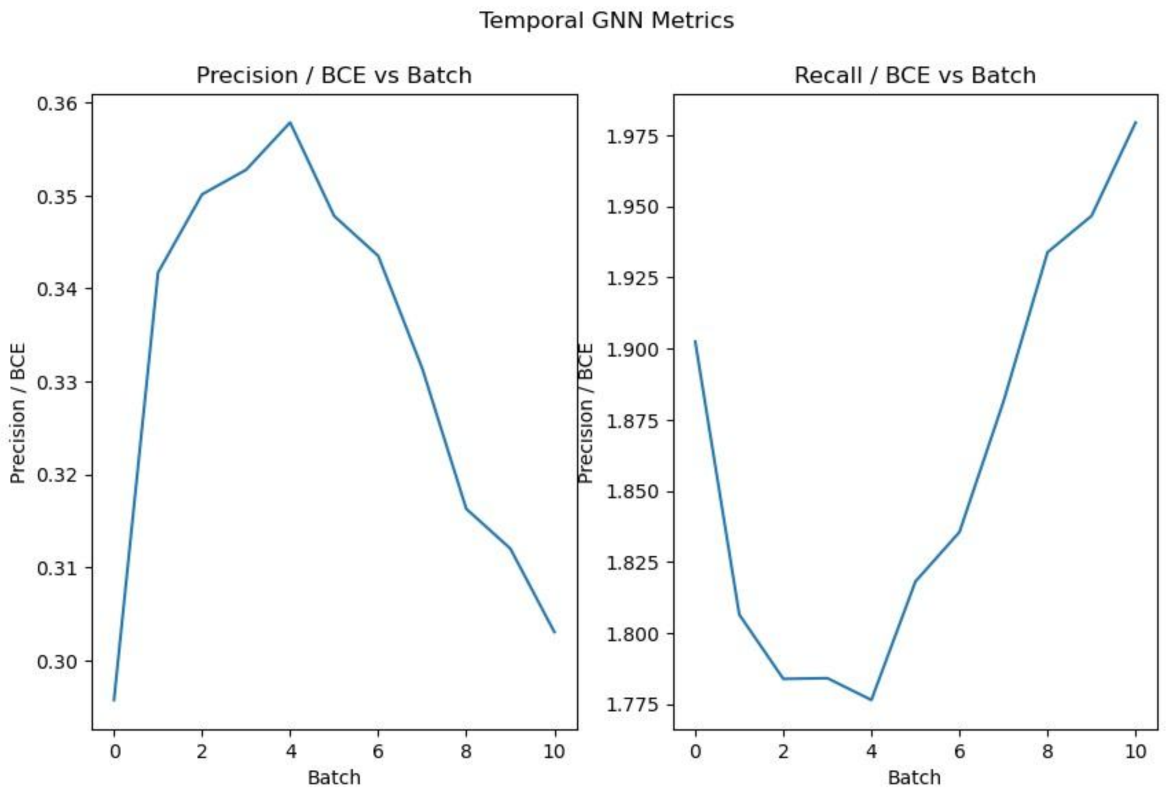


Figure 6: Precision/BCE vs Batch Number for Temporal GNN

As can be seen, the precision gets progressively worse, while the recall gets progressively better. This means false positives are quite a large source of error for our model. In all, this model is not good enough to be considered successful.

The next model was a GNN with weather variables as the explanatory features and collisions as the target. The results for this model were not much better. The recall for this model, for instance, was notably worse. However, when looking at precision, the model does seem to make a constant improvement. Although, because of the recall results, it can be assumed that this model is also not very effective.

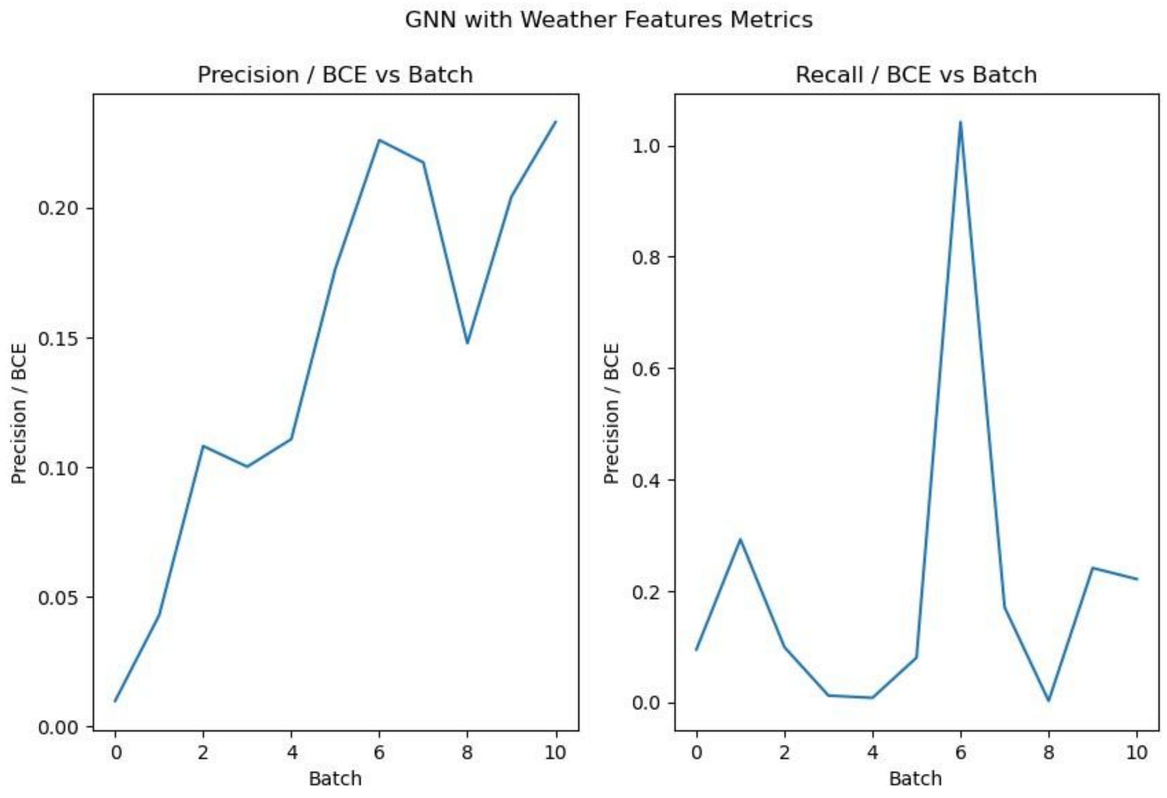


Figure 7: Precision/BCE vs Batch Number for GNN with Weather Features

Neither of the models tested works exceptionally well. This is not too unexpected, as traffic accidents are dependent on much more than weather, traffic counts, or other nearby or previous accidents. The data was simply not enough to explain the accidents that occurred and, therefore, the model could only do so much. Both models are able to correctly predict the presence of accidents more often than a zero rate or weighted rate classifier. However, the large rate of false positives and false negatives certainly makes the model too inaccurate to be used in a real life setting.