Kinori Rosnow (KSR43)
December 2, 2020

Traffic Forecasting

Introduction:

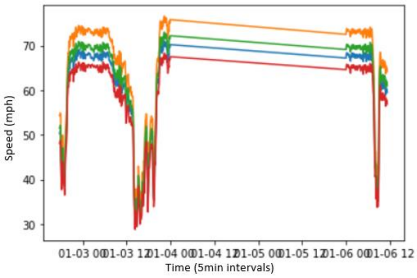
Population growth of humanity continues and as a result the stress on the infrastructure demands solutions to maintiain efficiency. Traffic jams in urban areas waste people's time while they sit idle in cars and in turn also has a wasteful economic impact. In addition cars sitting idle as they do in traffic, contribute to greenhouse gas emissions. A 2015 US study found that 30 million tons of CO2 are emitted every year from cars sitting idle [1]. A 2016 study in Canada found that if Canadian drivers spent 3 minutes a day less sitting idle in their cars, their annual emissions of CO2 would decrease by 1.4 million tons per year [2]. In addition to expanding infrastructure there is potential to approach efficiency of travel from an alternate direction such as using the existing infrastructure better.

There are groups that have researched and implemented traffic forecasting systems such as Microsoft Research with their system Smartphlow which considered events such as sports games to predict traffic [3]. Much of the work has been simulation based. These simulations have varying degrees of randomness and can either be continuous or discrete with respect to time. There are also physics guided simulations which leverage known percolation or kinematic wave models to treat traffic as a fluid.

The goal of this project is to maximize efficiency of current infrastructure by providing a model for predicting macroscopic traffic flows. Rather than modeling individual cars, I aim to use a history of lane speeds to forecast possible flows in the future. As with all forecasts there is a stochastic nature that should be conveyed. To address this, rather than just predicting lane speeds for the future, I predicted probability distributions of road speeds assuming a Gaussian distribution. From this distribution a sampling could produce possible lane speeds and the confidence of that sample based on the variance of the distribution.

Dataset:

The dataset used was provided by the Caltrans Performance Measurement System which has traffic data from highways around the greater LA area in California. The data consists of 200 sensors at various locations, each sensor measuring 4 lanes (or 3 lanes and an average of all the lanes for highways of more than 4 lanes), and measurements from each sensor at 5 minute intervals.



The dataset had many sections of missing data where there was no recorded speeds from a sensor. The sections were also no consistent across all 200 sensors so some may have recorded speeds while others did not record. For my training I handled missing data by inputting the average of the existing data in that batch.

Experiment:

Kinori Rosnow (KSR43)
December 2, 2020

Data processing:

I trained the model on sequences of 100 time steps which equates to 8.33 hours. The goal was to do longer term forecasting, but as will be explained in the future work section, I ran out of time before getting to attempt this. The data was processed and batch via a dataset class and a collate function I designed. The collate function was where the missing values were filled with averages.

Loss function:

The loss function used was the negative log likelihood assuming a Gaussian distribution. The goal was to minimize the loss by maximizing the likelihood estimate of the data given the output distribution. The functional form is:

$$L\left(X\middle|\mu,\sigma^{2}\right)\approx-\frac{n}{2}\ln\left(\sigma^{2}\right)-\frac{1}{2\sigma^{2}}\sum_{i=1}^{n}(x_{i}-\mu)^{2}$$

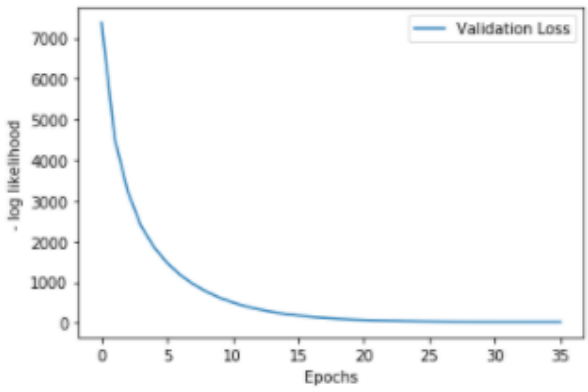
Where X denotes the observations, μ is the mean, σ^{2} is the variance and n is the counter for all datapoints. Separate negative log likelihoods were calculated for each sensor as their distributions were very different.

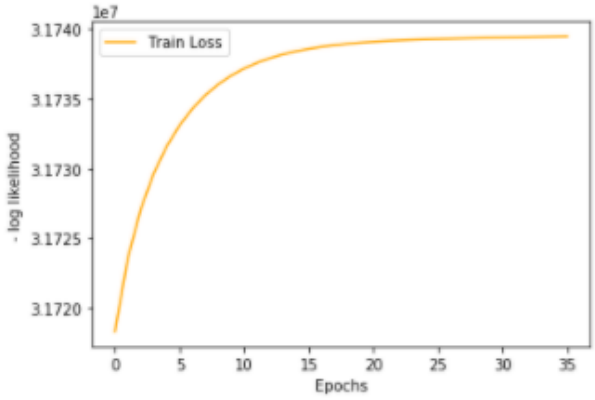
Model:

In order to model the traffic data I used the pytorch framework to build a recurrent neural network variant. The first layer was a 1D convolutional layer that created feature maps of the 200 sensor's 4 lane speeds. I fed each time step into the convolutional layer as a flattened vector of 800 features (input channel = 800 = 200 sensors x 4 features). The convolutional layer had 512 out channels embedding each timestep. There was no need to make the embeddings of roads independent as their lane speeds are not independent. In addition, the number of timesteps were preserved as the kernel size was 1 and the stride was 1. The goal was to allow the network to model these dependencies itself. The feature maps were passed to a single long/short term memory recurrent neural network variant (LSTM). Finally the LSTM layer passed its outputs to 2 linear layers. One linear layer aimed to compute the mean while the other was there for the standard deviation. I did not have the network compute the variance directly to allow the network the flexibility to produce negative values. Although standard deviations are never negative, the loss only takes the variance into account and as a result the sign will not matter.

Kinori Rosnow (KSR43)
December 2, 2020

The model was trained for 36 epochs total. The 36th epoch model was used for the visualizations in the conclusion. The training and validation loss plots are below. I am not familiar with training losses increasing in the manner it did in the plot. Further investigation revealed that the training loss actually started very high and reduced in the first epoch.



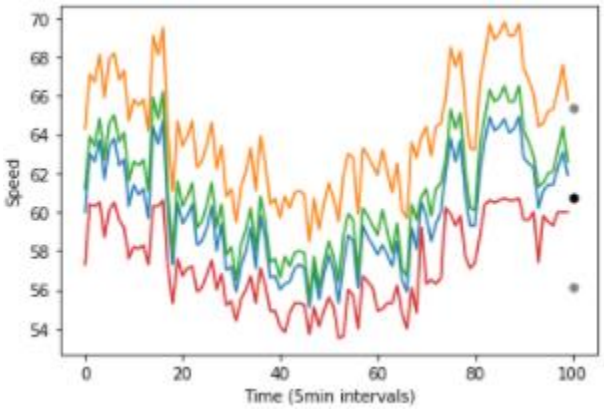


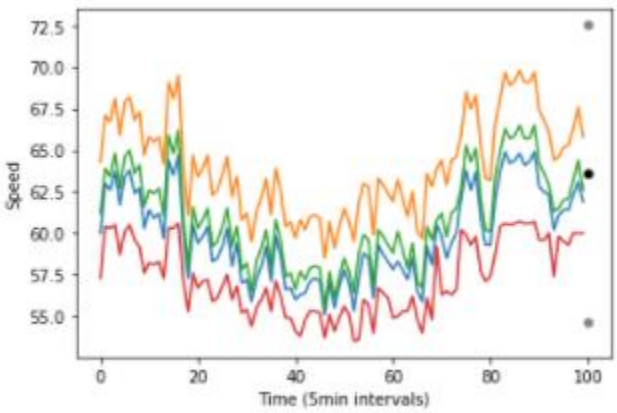
System:

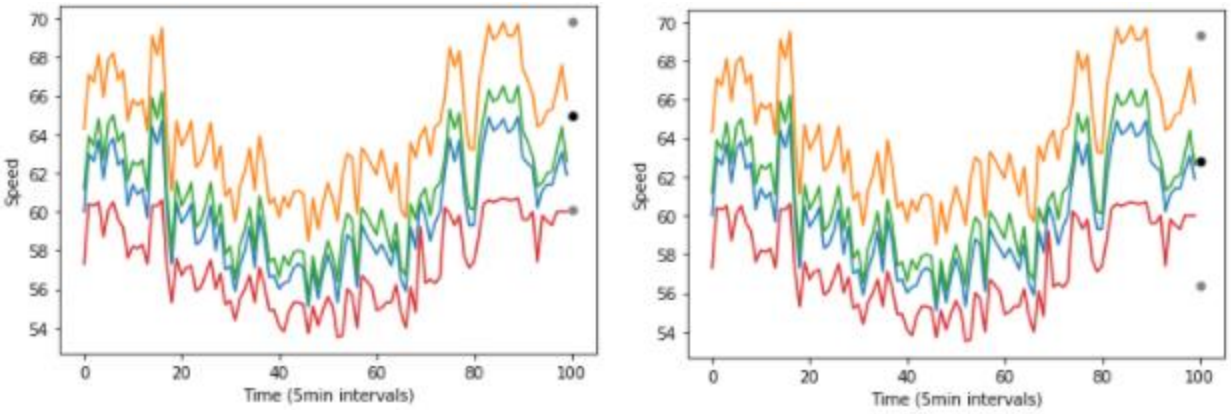
The data was modeled on an AWS gd4n.xlarge EC2 instance where I used the GPU to accelerate the training. Python 3.7 was used with Pytorch 1.7, Numpy 1.18.1 and 3.1.3.

Conclusion:

The model showed signs of convergences looking at the negative log-likelihood on the validation set which was checked after every epoch. I only had time to predict a single following time step, a major limitation of my evaluations. The visualizations of the next step show preceding time series of speeds (the 4 leading colored lines) of a single sensor leading up to a predicted mean (black dot) and a standard deviation above and below that mean (grey dot). Certain output distributions model their sensors better than others; however the output distributions did seem to capture much of not only a mean, but also the uncertainty. The following visualizations are of sensors 0 (top left), 2 (top right), 100 (bottom left), and 199 (bottom right).







The model's forecasts for the above sensors are below. The values are generated assuming a Gaussian distribution.

Sensor	Mean (mph)	Standard Deviation (mph)
0	60.7	4.6
2	63.6	9.0
100	65.0	4.9
199	63.0	6.4

It seems that following the next timestep's speed was not just wild guesses as for many of the sensors the mean was somewhere in the middle of the actual 4 lane speeds. What is also exciting is for sensors 0 and 100, the standard deviations aren't so wide that the model is just creating large error bars to account for every possibility. It seems there is support for the idea that the model is picking up on something. However, sensor 2 does have relatively wide error bars and sensor 0's distribution seems to be skewed off of center. Despite this, the result did show promise for future modeling especially with better usage and augmentation of data.

Future Work:

There are many desired improvements I would make to the current data processing and modeling. Listed below are potential avenues to build off of the work of this project.

1. Other distributions or time series models

It would be interesting to see the effects of using other distributions to model the lane speeds. In particular I would be curious to try non-symmetric distributions to see if they perform better. It is possible that not all sensors should use the same distribution.

2. Graph NNs (GCNN)

The dataset did come with a graphical representation of the highways. Using this as an input could provide the model with more information to allow it to better understand the dependencies of the road speeds.

3. Strategies for missing data

Better handling of the missing data would be ideal. I have two ideas that have potential to improve performance but they both required too much time for this semester. The first is to get rid of missing sections of data entirely. The issue is there are almost no sections where all sensors are dead and many sections where at least one of them is. The best use of the data with this strategy would be to create datasets of the sensors separately then to segment out the missing data. This is not only tedious but also proves a challenge to then incorporate the lane speeds of all of the sensors as additional information to model any of the highways. The more exciting option is to generate the missing data with another model such as a bidirectional LSTM or some sort of regression model. This has potential to be much better than my averaging strategy given that the model generating the data performs well.

4. Shuffle the lane speeds

One of the major issues I ran into when designing a way to generate multiple timesteps in the future was how to create a possible input to the model based on my predicted output. The simplest answer I came up with was to sample the output distribution randomly four times then use that as the input to the model. This is problematic because many of the lane speeds have specific relationships (ex. Left lane is fastest) so I may need to order them to get good performance from the generation. Generation can have compounding errors with off performance early in the generation. I realized too late that I should have tried training the model with shuffled lane speeds for each sensor. This way the lane speed order may not matter. This could be similar to rotating images for image recognition.

Citations:

- Dataset: http://pems.dot.ca.gov/
- https://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-243.pdf
- https://www.microsoft.com/en-us/research/project/predictive-analytics-for-traffic/
- https://afdc.energy.gov/files/u/publication/idling_personal_vehicles.pdf
- https://www.nrcan.gc.ca/energy/efficiency/communities-infrastructure/transportation/cars-light-trucks/idling/4415
- https://www.statlect.com/glossary/log-likelihood