Arrival Time Prediction

Advances in Artificial Intelligence for Social Good

Carnegie Mellon University
Brandon Houghton and Kenji Yonekawa
{bhoughto,kyonekaw}@andrew.cmu.edu

Abstract

In complex urban environments, daily route schedules are unable to capture the dynamic nature of traffic and as such, many users rely on their cellular devices to accurately predict the arrival of buses. We present an arrival time estimation system for informing users of public transportation in complex urban environments. Building on previous results, we layout a weighted piecewise linear mixture model that achieves a decrease of over 57% MAPE versus the currently deployed Port Authority $TrueTime^{SM}$ system evaluated over two months of route data for the 61C line. We compare our results with previous works and demonstrate the simplicity offered by our approach, allowing our model to be deployed effortlessly in place of existing prediction APIs.

Introduction

Public transportation is a vital backbone to modern cities offering reliable transportation to much of the surrounding area. An excellent public transportation system can transform a congested urban jungle into a clean and efficient one, connecting many communities and bringing as much as 1.8 billion dollars of economic value to the city (CITYLAB). A key factor in these transit systems is the accessibility and predictability of their vehicles. One approach to increasing both is to deliver a daily schedule for arrivals. The problem is, traffic is often unpredictable and using historic estimates is prone to accumulate error over time leaving riders later in the day wondering where their ride is. There has been a growing trend to publish data for social good (DATA.GOV ; pittsburghpa.gov) in the spirit of transparency and also the development of algorithms to further our predictive power. The Pittsburgh Port Authority Transit System (PAT) took this approach in the development of their recent upgrade to the Pittsburgh bus and tram system, enabling GPS tracking of every route as well as developing a real-time predictive model, $TrueTime^{SM}$. TrueTime provides a web-based API allowing app developers to query the bus system for arrival time predictions in real time and there are multiple smartphone applications that utilize this API to provide real-time bus information to users (Play b; Play c; iTunes ; Play a). These applications demonstrate the strong interest in such applications where the number of Android downloads in Pittsburgh alone exceeds 25,000 reaching nearly 10% of Pittsburghs population. Despite this real-time system, a lot of users complain about inaccurate prediction due to but not limited to; inaccurate GPS information given by PAT, delays due to weather conditions, and being unable to board the bus due to the number of passengers (Post-gazette.com ; Reddit). This deteriorates users' experience and may result in them using alternate transportation such as driving a car or calling an Uber.

Research at the University of California Berkeley pointed out eight reasons deterring people from public transportation (Forbes); which include long wait time, delay due to traffic, and missing departure due to wrong real-time information. As such, while it is nice to have a real-time prediction ecosystem, the provided information must be accurate to provide appreciable benefit to users.

In order to solve this problem, we aim to improve the accuracy of predicting bus arrival time. By using the GPS real-time API provided by the PAT bus system, we use machine learning to predict the time when buses are reaching bus stops. We first collect data provided by PAT and analyzed data to check if approaches taken by related work could also be utilized. As a result, we propose to use velocity instead of distance information in prediction. Unlike traditional approaches, we divide the bus route into meaningful segments and incorporate other information such as day of week, time of day, and weather. Our contribution is open to the public (Github.com).

The paper is organized as follows. In section Related Work, related work is presented and analyzed. Section Approach describes the proposed approach including the data used and algorithms used in the paper. We then present our evaluation method and results in Section Evaluation. Our future work is discussed in SectionFuture Work. Finally, Section Conclusion concludes the paper.

Related Work

Before the prevalence of GPS bus tracking, the majority of real-time arrival prediction was based on detecting the bus's location and estimating which route the rider was using based on cellar network signal strength and listening for the beep of a ticket collection system. While these approaches were inventive they did not directly address the accuracy of arrival-time prediction.

An example of this is Zhou et al. where they proposed arrival time prediction using passenger's cell phones. They use cellular networks, movement, and audio recordings information and used simple linear model. They used cellular networks to estimate the bus position, and movement of the phone to detect if the participant and audio recording to detect the beep of the IC card reader on buses to detect if the participant is on the bus or not. They achieved the absolute average error of approximately 80 seconds, which unfortunately is impossible to compare to other research results without knowing the prediction horizon. The research successfully utilized participatory sensing to collect data and predicted buses with more than 90% accuracy; however, their approach of using movement and sound may cause privacy issues.

Other more recent results used more accurate GPS tracking and focused more closely on the predictive power this offered. Choudhary et al. summarized the real-time bus prediction research done from 2004 to 2015 (Choudhary, Khamparia, and Gahier 2016) giving a good picture of the direction of recent research. It categorized models used in research into three types: historical data models, statical models, and machine learning models. They also evaluated the direction of newer methods and concluded that artificial neural networks were gaining interest in the field yet historical and regression models are still pervasive.

Some of the best performing models were remarkably simple and Wall et al. proposed using GPS and predict if a bus arrives at a bus stop within 15 minutes (Wall et al. 1999). They proved that their linear model achieved an arrival time prediction with 12% error. Some similar research followed this paper and used time over the distance in their prediction. However, we found that the use of distance versus time is not suitable in a more dynamic environment, and this model achieved poor performance on our collected dataset.

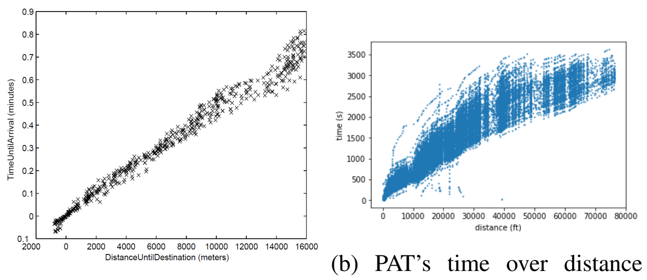
Sun et al. proposed using the velocity of the buses in predicting the arrival time (Sun et al. 2007), demonstrating that depending on the time of day, the average speed of buses differs significantly. Their GPS sensor resolution was one sample every 26 seconds and they achieved an average of around 15% mean absolute percentage error. However, their evaluation metric was based on a low-traffic, predictable environment and used bus stops to segment the whole trip leading to poor performance when considering buses that were far from arrival.

Approach

Data Set

We used a web server to collect the data from following online APIs:

- Bus API (PAT TrueTime API (Post-gazette.com)) This API provides realtime and scheduled information on PAT vehicles including bus stops, routes, vehicles locations, and prediction of bus arrival at each bus stop. Most of the smartphone applications use this API directly to predict bus arrival time and display current bus locations. We collected real-time bus location and predictions for two months (March 2018 to April 2018). We queried the API at 10 second intervals which are the minimum interval allowed. Unfortunately, while the API has the potential to update locations every 10 seconds, the actual pooling rate is much lower which resulted in obtaining each bus location every two to three minutes. For our evaluation, we selected a single route that had the most data points and ended up using: 155,398 for Training (March) and 101,504 for testing (April). When comparing other routes performance was comparable so results presented will generalize well for other routes.
- Weather API (Accuweather API (AccuWeather)) This API provides hourly weather information such as weather, temperature, and wind speed, for a specific location. We collected the weather data for the same duration explained above. When using the data in training and testing, we incorporated weather information based on the time of bus location.



(a) Wall et al.'s model (Wall et al. 1999)

(b) PAT's time over distance time data

Figure 1: Time over distance data drawn from previous research and our data

Data Representation

This section describes two data representation methods we utilized in our predictions. The first is the bus information representation and the second is observation representation.

Bus Information Representation In order to check the feasibility of existing methodologies, we first applied some of the approaches proposed by the related work.

We first investigated the distance vs. time model which Wall et al. (Wall et al. 1999) proposed. Figure 1a indicates the data used in Wall et al.'s research and figure 1b illustrates the data we are using (Mar 2018). As shown in the figure, our data is much more diverse and resulted in 36.9% Mean Absolute Percentage Error (MAPE) compared to 12% of that of Wall's. (MAPE is discussed in detail in Sec. Evaluation Method).

We analyzed the data and concluded that this approach is not suitable for two reasons: 1) error accumulates over time, 2) velocity changes drastically depending on the location on the route (e.g. bus have lower speed in downtown). In order to mitigate their influence, we decided to use distance vs. velocity instead of distance vs. time. This representation is depicted in Fig. 2. As shown in the graph, we can see that velocity is more independent of previous points which allows us to mitigate error accumulation over distance.

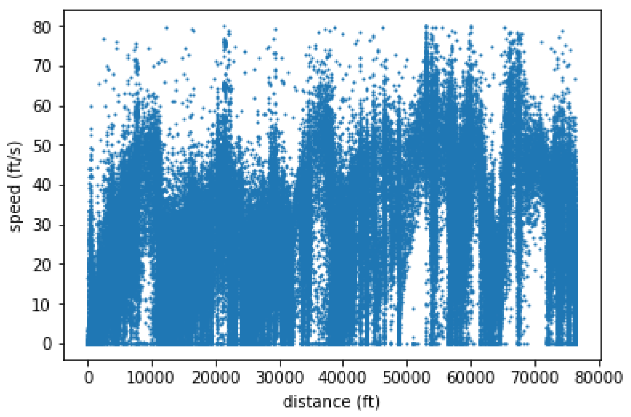


Figure 2: PAT's distance vs. velocity data

In building our model, we wanted to capture the varying terrain and conditions seen at different locations in the route, so we divided our linear model into sections. There were many ways to do this and from the papers, we originally tested using uniform segments; however, the performance fluctuated wildly based on the number of segments used with error as high as 239% MAPE. We added another type that split the data based on the number of samples which performed better but still relied on choosing the right number of bins. Finally, we decided to use a data-driven approach and allow the tree based model to divide the regions giving us a much better segmentation of the data. Fig. 3 represents the four data division methods we analyzed. Previous research (Bin, Zhongzhen, and Baozhen 2006; Sun et al. 2007; Lei et al. 2017) used bus stops on the route to divide the route. We tested some models and confirmed our models perform worse using such a method of segmentation: the MAPE (described in Sec. Evaluation Method) was 25.29% which is considerably less accurate. This is due to the fact that bus stops don't necessarily divide the routes into segments with predictable velocity, failing to capture more relevant information such as stop-signs, traffic lights, and difficult intersections.

Observation Representation In order to represent observations (e.g. time of day, day of week, weather), we leveraged the following data representation model.

- Time in Route
  We identified when a bus changes its route and defined the time the bus began a route by extrapolating linearly. Therefore, every observation we use has the time in second since the bus started that specific route. The time when the bus started the route is calculated as follows. t_0, t_1, and t_v each denotes the time before changing the route, after the route, and (predicted) time at origin, and x_0, x_1, and x_v each represents that of distance in route.

$$t_v = t_0 + \frac{(t_1 - t_0)(x_v - x_0)}{x_1 - x_0} \tag{1}$$

- Time of Day
  We used one-hot vectors to represent the time of day. The original format of this data is given as a two digit number from 1 to 24. However, the relationship between speed and time of day was highly non-linear thus by converting time to a one-hot vector we are able to account for these non-linearities at the cost of some model complexity. Later we adjust for this complexity by using the tree model to determine breaks in the time of day.
- Day of Week
  This information is generated from collected data. We believe that the day of week, especially weekday and weekend, have a large impact on the bus traffic and hence, we represented the day of week in one-hot vectors to get a better prediction performance.
- Weather
  We initially used the scale provided by the API (AccuWeather) which had 43 nominal labels for weather statuses. We soon realized our models overfit and the prediction performance degraded approximately 10% MAPE. We then checked the labels and summarized into four categories (e.g. no impact, sunny, rainy, and snow) based on our intuition of how weather affects bus traveling speed. These values are represented as one-hot vectors.
- Temperature
  We obtained this information in Fahrenheit; however, we did not use this information. We first used the number as is which significantly degraded our prediction models (around 15%). We then compiled five degrees into one and still got approximately 5% increase in error. After some analysis of the data, we decided not to use this data because it was causing overfit and weather was more related to bus velocity.

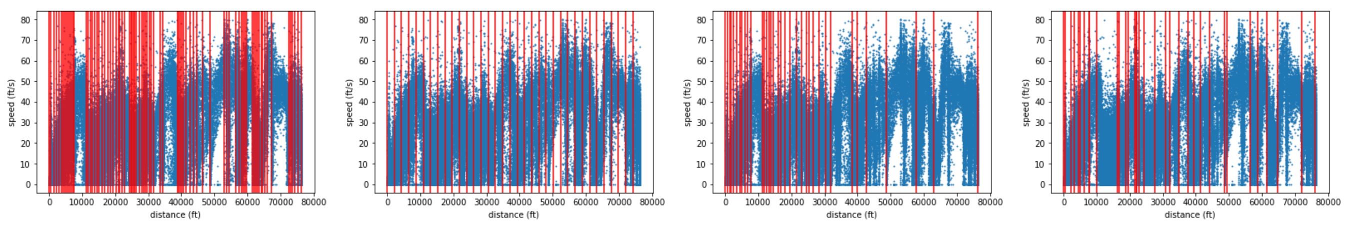
Models

Table. 1 indicates the models we designed and trained. The details are described below.

- Linear model
  Our first approach was to generate a proof of concept where we assume that the vehicle speed is simply defined by a line from the start of the route to the end of the route. This provided a good measuring stick to ensure that we were getting better performance and that there were no errors in our model. Here our objective was simply to minimize the squared distance between observed velocity:

$$\sum_{i=1}^{N}(v_{pred} - v_{obs})^2 = \sum_{i=1}^{N}(v_{obs} - \beta_0 - \beta_1 * x)^2$$

  Where N represents the number of points in the training set. Another advantage of this model is that we have a closed form solution for β_0 and β_1 letting us use all 400GB of collected data with less than a 10 seconds of training time.
- Piecewise linear model
  When modeling the data we saw that regions of the route have a strong linear correlation in the velocity so we decided to segment the route into distinct sections and



(a) Bus Stops (b) Uniform Distance (c) Uniform Number of Data (d) Tree Based

Figure 3: Data Division Methods

#	Model name	Description
1	Linear model	Linear model over the whole pattern
2	Piecewise linear model	Linear model over segments of the pattern
3	Decision tree linear model	Decision tree with linear model over the whole pattern
4	Piecewise linear mixture model	Linear model over segments of the pattern with weighting of parameters (e.g. hour, day of week, weather)
5	Decision tree with linear mixture model	Decision tree with linear model over segments of the pattern
6	Piecewise linear model with momentum	Linear model over segments of the pattern with weighting of parameters (e.g. hour, day of week, weather) with real time data of the bus of interest. Momentum is derived from past couple of observations to compare against historic data.
7	Weighted piecewise linear model with momentum	Model choses a weighting of the momentum model and the historic mode, preferring momentum for segments close to the known location and historic for segments that are far away.

Table 1: Models Designed and Implemented in this paper

regress a linear model for each giving us the objective:

$$\sum_{j=0}^{M}\sum_{i=1}^{N}(v_{j,pred}-v_{j,obs})^2 = \sum_{j=0}^{M}\sum_{i=1}^{N}(v_{obs}-\beta_{j,0}-\beta_{j,1}*x)^2$$

Where M represents the number of segments in the model. This model greatly improved performance but required a longer training time as we selected optimal hyper-parameters by training models with up to 120 segments.

- Decision tree linear model
  To avoid searching through our hyper-parameter space, we applied a decision tree classifier to predict the velocity given features such as distance traveled in route, weather, and other observations as discussed above. This had the advantage of relatively small hyper-parameter space (depth of the tree, minimum number of examples per branch, etc.) yet required almost 5 minutes to train given the complex expressive power of this model. This model's objective maximized the entropy captured by the leaves of the tree subject to the depth constraints. While this model did perform better, we see that in predicting uniform velocity the model lost its ability to model the acceleration present in many places in the graph.

- Piecewise linear mixture model
  Here we began encoding the external observations (weather, day of the week, etc.) as one-hot vectors and used a linear mixture model to predict their impact at each segment of the route. This gave us a slightly more complex objective:

$$\sum_{j=0}^{M}\sum_{i=1}^{N}(v_{j,pred}-v_{j,obs})^2 =$$

$$\sum_{j=0}^{M}\sum_{i=1}^{N}(v_{obs}-\beta_{j,0}-\beta_{j,1}*x-\beta_{j,2}*weekday-\beta_{j,3}*obs_2-...)^2$$

  Where we regress a coefficient for each of the observational variables encoded as a one-hot vector of indicator variables. This offered a 34% decrease in MAPE, however we again have the issue of hyper-parameter search.

- Decision tree with linear mixture model
  In order to combine the results from the decision tree with the increase in performance of the additional observations using the mixture model, we use the breaks from the decision tree that define relevant sections of the route as well as the division of hours into sections roughly corresponding to rush-hour, daily traffic, and no traffic. This results in a further decrease of 15.6% MAPE. Yet these models do not take into account the current speed of the vehicle or past observations.

- Piecewise linear model with momentum
  In order to account for the speed of the vehicle when it was measured, we trained a new model that used the past observations to predict future velocity. We tried adding the previous speed to the mixture model but observed a worse performance. Therefore, we designed a momentum

term that measured the current speed relative to the expected speed at that time and used this to predict velocity. While this worked well for points close to the stop, for far away points we saw huge errors. This lead us to a worse than linear MAPE of over 48% however as the error was systematic we used this to improve our final model.

- Weighted piecewise linear model with momentum
  Finally combining all of the traits learned from models above we use a tree based segmentation to create breaks in the route, using a linear mixture model at each of the breaks based on the observed variables. We then calculate the historic velocity using this model, as well as the momentum based velocity, and take a weighted average of the two. This gives us accurate predictions for points close to the known locations and gives a higher bias to points that are farther away. This gave us the best performance of all offering a 57% decrease in MAPE over the PAT model.

Evaluation

This section describes the evaluation method and its results along with some insights we observed

Evaluation Method

This section describes the method we used in our evaluation.

We used the real-time data collected from PAT API (Transit): the point (distance in route) and time is used to evaluate the prediction performance. In order to measure the prediction performance, we used Mean Absolute Percentage Error (MAPE) since it is a widely used metric and it is easy to compare against existing works. MAPE's equation is indicated in Eq. 2 where A and F denote actual time and predicted time, respectively.

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \quad (2)$$

To evaluate models, we estimate the true arrival time for a vehicle at a given stop by finding points where the GPS location of a bus was incident on that stop. We then gather any observations and predictions made by PAT in the past and evaluate the relative error of these against the true arrival time. As we retrieve predictions and observations at the same time we are able to match these to evaluate models more accurately.

To infer predictions using our models, we calculate arrival time using the following procedure:

1. Train the model on train data using the velocity as label and other observations (e.g. distance in route, time of day, day of week, weather, temperature, etc.) as features.
2. Predict the time between a given observation and the distance on route to predict as follows.
   (a) Divide the distance between the given point and the point to predict
   (b) For each segment, predict the velocity using given observations and calculate the time required to travel that distance
   (c) Add all predicted time for each segment and compare against the actual difference in time

#	Model name	MAPE
1	Linear model	28.48%
2	Piecewise linear model	23.84%
3	Decision tree linear model	22.70%
4	Piecewise linear mixture model	18.53%
5	Decision tree with linear mixture model	15.60%
6	Piecewise linear model with momentum	48.20%
7	Weighted piecewise linear model with momentum	12.21 %

Table 2: Evaluation Results

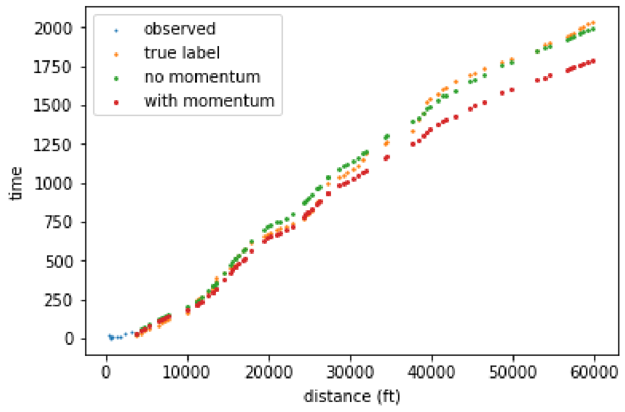


Figure 4: Prediction result with/without momentum

We used PAT real-time bus and weather information in our evaluation. We selected one of the routes (61C) in our evaluation since it had the greatest number of data points. We segment our data by month to fully separate training data vs. testing data and to better approximate a model that is updated once a month account for new data.

- Train Data
  We used the data collected in March 2018, which had 155,398 location samples.
- Test Data
  We used the data collected in the first two weeks of April 2018, which had 101,504 location samples.

Evaluation Results

Table 2 illustrates our evaluation results. #1 Linear model is the same model used for prediction in the PAT API. As shown in the table, our final prediction model outperforms existing model by 57.13 %. The improvement is significant; however, since we only have two months of data, we do not know if our models generalized to different seasons, though we expect that our model will do better given more extreme weather because of the fact that it is able to account for statistically significant indicators.

An interesting observation we faced during the evaluation is the significant performance degradation from #5 Decision tree with linear mixture model to #6 Piecewise linear model with momentum. Fig. 4 illustrates the true label, prediction results of #5 and that of #6 when the first eight data of a specific bus was given for momentum calculation. As indicated in the figure, the momentum model outperforms non-momentum model when predicting somewhere close; however, its prediction significantly degrades when predicting somewhere further. This clearly indicates that the momentum is better in predicting close distance while the historical model is more suitable for predicting further distance. After finding this problem out, we came up with a new model, #7 Weighted piecewise linear model with momentum. In this model, we introduced a new idea of weighing how much momentum we take into prediction into consideration. Unfortunately, we have not had enough time to analyze the suitable weighting mechanism. In this paper, we introduce a proof of concept for this model as described in Eq. 3 and Eq. 4. p_h and p_m indicates prediction based on historical data and momentum respectively, and x_p and x_l are predicting position and current position. As indicated in Table 2, our proof of concept proved that the use of weighting successfully improved the prediction accuracy. The weighted model is not based on proven data and hence, we need further investigation on this.

$$prediction = p_h * (1 - weight) + p_m * weight \tag{3}$$

$$weight = \frac{1}{\sqrt{x_p - x_l}} \tag{4}$$

Although we cannot compare our models against others objectively due to lack of publicly available data for evaluation, we believe our models are superior to others due to two reasons: 1) we leverage tree-based algorithm to divide the route into segments with similar traits as opposed to previous research which uses bus stops or uniform distance to segment the route, and 2) we leverage momentum (real-time observations of the past) to improve the prediction accuracy. Although we use data from different locations, our prediction model achieves high prediction accuracy (approx 12% MAPE) with low precision GPS data (every 120-180 seconds) compared to previous research. For example, Agafonov et al. used adaptive model (similar to regression tree) and achieved MAPE of around 10% to 27% with 30 seconds interval GPS data (Agafonov and Myasnikov 2015) in Samara, Russia; however, their result shows that as the prediction horizon increases, the accuracy also increases which implies that bus velocity is stable. In the meantime, Treethidtaphat et al. evaluated deep neural network against the linear model and concluded that neural network achieves 55% better (approximately 25% MAPE) using five seconds interval GPS data in Bangkok, Thailand (Treethidtaphat, Pattara-Atikom, and Khaimook 2017). Note that we use data in a different city and hence the performance may vary if we use the same data; however, our models achieve higher prediction accuracy with less frequent data.

Future Work

Our future work is as follows.

- Design and evaluate weighted momentum model
  As explained in Sec. Evaluation Results, although our weighted momentum model outperformed other models, the model is still a proof of concept and we believe it has a room for improvement. One possible way of improving is to analyze true labels, historical predictions, and momentum based predictions and design a model. We believe this can be done through some machine learning algorithms; however, we first need to collect more data and further analyze before deciding how to train and derive the best weighting model.
- Collect and train on a larger set of data
  We only used data from March to April 2018; however, we project that we would observe different behaviors in different seasons, such as an increase in delays when there is heavy snow. Therefore, collecting and evaluating on longer duration of time is one of our future goals. This was unfortunately beyond our ability as we only had a couple of months to collect data but we will continue to collect this information and make it publicly available.
- Add more observation data to improve the performance.
  We have only used bus real-time information and weather information. However, there are a lot of available information that would improve the performance of prediction such as real-time traffic information, school class schedules, and holidays and events. These can help explain the more extreme fluctuations in transit use, especially as dramatically fewer students use the PAT transportation system during holidays and on break. We also wish to add Google's real-time traffic information to get an accurate estimate of traffic to help inform our model further.
- Define publicly available evaluation method
  There is no publicly available benchmark for arrival time prediction and hence, it is difficult to compare different proposed models. We think that developing a comprehensive benchmark containing both training and a withheld evaluation set would better allow us to compare prediction models using the same data. We believe each model have strength and weakness, and hence, it is important for us to have such platform we can evaluate different models and share contributions with others.
- Provide our contribution as an API
  We implemented our models and evaluated in a public repository where everyone can access; however, users have to collect data and train on them before start predicting. One of our future goals for this project is to publish our contribution as an API so that various application developers can utilize our model to improve their prediction performance. This is especially important as our lightweight model was developed with the intention of being deployed on a server somewhere.

Conclusion

We presented a novel approach to capturing the underlying velocity model for PAT public transportation vehicles achieving an increase of over 57% MAPE in predicting the arrival times. This result allows for much more precise estimation of wait times and the authors hope to deploy this system in order to better the accessibility and predictability of public transportation in Pittsburgh. The authors would like to thank Professor Fei Fang for her support in developing this project and for the PAT Developer Program which provided us with enough API keys to collect this large amount of transit data.

References

[AccuWeather] AccuWeather. Apis. `https://developer.accuweather.com`. Accessed: 2018-2-11.

[Agafonov and Myasnikov 2015] Agafonov, A., and Myasnikov, V. 2015. An adaptive algorithm for public transport arrival time prediction based on hierarhical regression. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*.

[Bin, Zhongzhen, and Baozhen 2006] Bin, Y.; Zhongzhen, Y.; and Baozhen, Y. 2006. Bus arrival time prediction using support vector machines. *Journal of Intelligent Transportation Systems* 10(4):151–158.

[Choudhary, Khamparia, and Gahier 2016] Choudhary, R.; Khamparia, A.; and Gahier, A. K. 2016. Real time prediction of bus arrival time a review. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT-2016) Dehradun, India 14-16 October 2016*.

[CITYLAB] CITYLAB. Public transit is worth way more to a city than you might think. `https://www.citylab.com/life/2013/08/public-transit-worth-way-more-city-you-think/6532/`. Accessed: 2018-4-28.

[DATA.GOV] DATA.GOV. Open government. `https://www.data.gov/open-gov/`. Accessed: 2018-04-27.

[Forbes] Forbes. Top eight reasons people give up on public transit. `https://www.forbes.com/sites/jeffmcmahon/2013/03/06/top-eight-reasons-people-give-up-on-public-transit/#10232203fd84`. Accessed: 2018-4-27.

[Github.com] Github.com. Pat_prediction. `https://github.com/kenzht/PAT_prediction`. Accessed: 2018-4-27.

[iTunes] iTunes. Busgazer: Realtime pittsburgh transit app. `https://itunes.apple.com/us/app/busgazer-realtime-pittsburgh-transit-app/id936235643?mt=8`. Accessed: 2018-2-11.

[Lei et al. 2017] Lei, J.; Chen, D.; Li, F.; Han, Q.; Chen, S.; Zeng, L.; and Chen, M. 2017. A bus arrival time prediction method based on gps position and real-time traffic flow. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress*.

[pittsburghpa.gov] pittsburghpa.gov. Open data for all of pittsburgh. `http://pittsburghpa.gov/open-data/index.html`. Accessed: 2018-04-27.

[Play a] Play, G. Busgazer. `https://play.google.com/store/apps/details?id=com.busgazer.oba&hl=en`. Accessed: 2018-2-11.

[Play b] Play, G. Maps - navigation & transit. `https://play.google.com/store/apps/details?id=com.google.android.apps.maps&hl=en`. Accessed: 2018-2-11.

[Play c] Play, G. Pat track. `https://play.google.com/store/apps/details?id=rectangledbmi.com.pittsburghrealtimetracker&hl=en`. Accessed: 2018-2-11.

[Post-gazette.com] Post-gazette.com. Real-time port authority bus tracking system not always real. `http://www.post-gazette.com/news/transportation/2014/10/16/Real-time-Port-Authority-tracking-not-always-real/stories/201410160155`. Accessed: 2018-2-11.

[Reddit] Reddit. Why is the bus gps tracking accuracy so bad? `https://www.reddit.com/r/pittsburgh/comments/2sxf9o/why_is_the_bus_gps_tracking_accuracy_so_bad/`. Accessed: 2018-2-11.

[Sun et al. 2007] Sun, D.; Luo, H.; Fu, L.; Liu, W.; Liao, X.; and Zhao, M. 2007. Predicting bus arrival time on the basis of global positioning system data. *Transportation Research Record: Journal of the Transportation Research Board* 2034:62–72.

[Transit] Transit, P. A. Welcome to port authority of allegheny county's truetime system. `http://truetime.portauthority.org/bustime/home.jsp`. Accessed: 2018-2-11.

[Treethidtaphat, Pattara-Atikom, and Khaimook 2017] Treethidtaphat, W.; Pattara-Atikom, W.; and Khaimook, S. 2017. Bus arrival time prediction at any distance of bus route using deep neural network model. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC): Workshop*.

[Wall et al. 1999] Wall, Z.; Dailey, D. J.; Wall, Z.; and Dailey, D. J. 1999. An algorithm for predicting the arrival time of mass transit vehicles using automatic vehicle location data. In *in 78th Annual Meeting of the Transportation Research Board, National Research Council, Washington D.C.*