Neural Bus Networks

by

Cooper Stokes Sloan

S.B., MIT (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author: ______________________________
Department of Electrical Engineering and Computer Science
May 22, 2018

Certified by: ______________________________
Alan Edelman
Professor of Applied Mathematics Thesis Supervisor
May 22, 2018

Accepted by: ______________________________
Katrina LaCurts
Chairman, Masters of Engineering Thesis Committee
May 22, 2018

Neural Bus Networks

by

Cooper Stokes Sloan

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2018, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

Bus schedules are unreliable, leaving passengers waiting and increasing commute times. This problem can be solved by modeling the traffic network, and delivering predicted arrival times to passengers. Research attempts to model traffic networks use historical, statistical and learning based models, with learning based models achieving the best results. This research compares several neural network architectures trained on historical data from Boston buses. Three models are trained: multilayer perceptron, convolutional neural network and recurrent neural network. Recurrent neural networks show the best performance when compared to feed forward models. This indicates that neural time series models are effective at modeling bus networks. The large amount of data available for training bus network models and the effectiveness of large neural networks at modeling this data show that great progress can be made in improving commutes for passengers.

Thesis Supervisor: Alan Edelman
Title: Professor of Applied Mathematics

Acknowledgments

Thank you to all of the MIT Julia team for the support and mentorship. Thank you to Jiahao Chen for providing the foundation and inspiration for the project. Thank you to Alan Edelman for the guidance and encouragement. And finally thank you to Allison Inanoria for always being there for me. I humbly submit this work as my contribution to the beautiful tapestry which is computer science research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Traffic and Technology

Americans spend over 40 hours stuck in traffic a year [26]. This costs the US $121 billion dollars a year. Studies from The American Journal of Preventive Medicine have shown that commuting causes a range of negative health side effects [23]. These effects include raised cholesterol, increased depression risk, increased anxiety, and decreased overall happiness. Commuting results in lower life satisfaction. Specifically, riding a bus for 30 minutes or longer is connected to the lowest level of life satisfaction compared to other commutes. All groups including bike commuters experience reduced life satisfaction proportional to the length of the commute. In addition to the detrimental health effects, road crashes result in 1.3 million deaths a year [13]. An additional 20-50 million people are injured or disabled each year in road crashes. The statistics are worse in low income countries.

In addition to health effects, traffic has a strong impact on the environment. Transportation accounts for 30% of US greenhouse gas emissions. Road traffic also contributes to reduced air quality, traffic congestion and urban sprawl.

Technology can to a great extent solve some of these problems. Public transit can largely reduce the environmental impact of transportation. A full bus is 6 times more efficient than a single driver car [34]. Additionally, buses emit a tenth of the hydrocarbons compared to single driver cars [34]. Recent years have also shown an increase in electric vehicles, which when combined with sustainable energy practices can drastically reduce emissions.

Furthermore, the advent of autonomous cars promises a future of safer roads and vastly fewer road crash deaths. Interestingly, car ownership shows signs of decline in the US, with millennials waiting longer before buying cars [33]. Ride sharing services like Uber are transforming mobility and the auto industry as a whole. An increase in sensors and connectivity of cars allows large scale optimization which can decrease travel times for users. All of these technical innovations make it cheaper, safer and more convenient for people to use modern forms of transportation. However, as mobility becomes available to more people, problems regarding traffic congestion will get worse, not better.

The problem of traffic is not going anywhere. Over the next 30 years, the US population is estimated to increase by 70 million [26]. Larger populations, and movement into urban and suburban areas exacerbates the issue. Traffic in other countries is even worse. India's transportation system is in crisis with booming population growth in urban areas. Increasing vehicle numbers are overwhelming transportation infrastructure. Clearly, transit technology needs to keep up with exploding demand.

One obvious way to reduce congestion is public transportation. However, the adoption level of public transit systems is very low. Only about 5% of US workers commute to work with public transit [30]. There are several reasons which cause people to prefer private cars to public transit. Outside of urban areas, public transit can be unavailable or impractical. Bus riders have to deal with traffic, transfers, and unreliable schedules. These factors make buses much less convenient than private autos, despite the clear environmental and economic benefits. This research focuses on improving the reliability of buses for riders by using neural networks to predict bus arrival times.

1.2 Bus Network Characteristics

Traffic networks in general are difficult to model. Part of this has to do with the stochastic nature of traffic. Traffic conditions depend on countless variables including weather, driver behavior, time of day, and construction. Although some of these variables are easy to measure, others are latent. Furthermore, the relationship between the variables is complicated. This requires the use of complex models and large amounts of data.

Besides the sheer number of variables, bus networks also exhibit some behavior which makes them different from typical car networks. For example, bus trajectories are affected by passenger demand. The amount of time a bus spends waiting at a stop depends on how many people get off and on at that stop. Furthermore, a bus may not make all of its stops along a route, and stops may move over time due to weather or construction conditions. Buses also have a schedule to follow, whereas cars do not. For this reason, some buses will slow down if they are ahead of schedule to allow other buses to catch up. These features make bus networks more complex, but they also provide structure for models to learn.

An interesting emergent property of bus networks is clumping. This term refers to the phenomenon of buses along a route to tend to clump together in groups after starting at uniform intervals. The effect is caused by the relationship between buses along a route and the respective riders. Consider the following scenario:

1. A bus misses a light, and therefore starts running behind schedule

2. At future stops, more passengers arrive due to the delay

3. The excess of passengers at future stops makes the bus slow down even more, because it takes longer to pick up more passengers

4. This produces a negative feedback loop, which causes slow buses to become slower

5. Eventually, the previous bus will catch up, causing the buses to clump together

The opposite happens to a bus when it starts running ahead of schedule, meaning fast buses get faster. This exacerbates the issue of clumping. This complex interaction among buses along the same route requires modeling at a route level rather than a bus level.

Another issue which arises studying bus networks is the natural randomness of traffic. Even with a perfect modeling scheme, there are always confounding factors. This results in a large variance which limits the theoretical best performance for prediction. Incorporating more data sources can mitigate this, because the model can exploit more patterns in the data.

1.3 Future of Travel

We now take a look to the future of urban mobility. It's difficult to talk about the future of transportation without considering the implications of autonomy and AI. Although the world has not yet seen widespread adoption of self driving cars yet, the sheer amount of capital being invested into their development indicates the world is ready for a change. This change may improve the efficiency of traffic networks, increase safety, and cut costs for urban travel. However this change will also bring unemployment and uncertainty for millions of Americans who drive for a living. Care must be taken so that this technology improves humanity as a whole, not just the lives of the urban elite.

Nevertheless, the possibilities created by autonomous cars combined with artificially intelligent traffic networks are immense. Deep learning models can be used to estimate the location and trajectory of all the vehicles in the network simultaneously. This can be combined with a historical model to predict demand and passenger behavior. Together these models can be used to dynamically allocate autonomous buses to the areas where they are needed the most. This will turn the bus network into a living, evolving graph rather than a fixed, unreliable schedule.

Rather than looking up a bus schedule to determine the fastest way to commute, passengers will simply enter their destination into their mobile phone and the fastest possible route will be computed, utilizing the various self driving car and bus fleets in the area. An electric autonomous car will pull up to their exact location. The autonomous car will find the nearest autonomous bus traveling in the correct direction and automatically link up while driving to transfer the passenger while the vehicles are still moving. Another autonomous car will make the final leg to the destination. Passengers will pay the ride directly, using a biometric based cryptocurrency. The autonomous car will then recharge and pay for any repairs using its profits.

For mid range commutes, passengers will travel via a combination of autonomous flying drones and autonomous cars. In larger cities, the number of flying drones will outnumber cars. All large urban metropolises will be connected by hyperloops or a similar technology. This will allow people to work and live in several cities seamlessly. This in turn will increase collaboration and sharing of ideas across nations, as well as improving international relations.

As connections between nations grow, there will be an increased demand for cheap, efficient inter-continental travel. Supersonic travel will return to support this demand, allowing jets to travel from New York to Paris in 3.5 hours. The decreased cost in rocket technology will for the first time allow intra-planetary rocket travel. Completely reusable rockets will make routine trips across long haul journeys, taking passengers from Houston to Sydney in a half an hour. As free travel and trade bring the world closer together, people will look to the stars, contemplating and developing solutions for interplanetary travel, furthering humanity's horizon.

By solving the technical problems of today humanity will have a bright future. This work starts with understanding and modeling the transportation networks we have now.

Chapter 2

Related Work

2.1 Theory

Significant theoretical work has been put into studying bus networks. A deeper fundamental understanding of bus networks may be the key providing accurate estimates of travel times for passengers and network operators.

Krbálek and Seba (2006) analyzed data from buses in Cuernavaca, Mexico, and found that the distribution of arrival times follows the Gaussian Unitary Ensemble (GUE)[22]. The Gaussian Unitary ensemble is a distribution stemming from random matrix theory which model Hamiltonians without time reversal symmetry. For more on random matrix theory and the GUE, see *Introduction to the Random Matrix theory: Gaussian Unitary Ensemble and Beyond*[14].

The connection between random matrix theory and bus networks is not immediately clear. However, Baik, Borodin, Deift and Suidan (2006) developed a microscopic model for analyzing the bus system in Cuernavaca, Mexico [2]. By introducing natural repulsion, Baik et al. are able to show how random matrix distributions follow.

Interestingly, the same distribution comes up in other networks as well. Jagannath and Trogdon (2017) showed that the distribution of gaps between trains in the New York City subway are captured by both the GUE and the Poisson distribution [19]. The authors explain this duality via the Coulomb potential along the route.

2.2 Modeling Bus Networks

Given the complexity of bus networks, several modeling techniques have been used to predict arrival times. Altinkaya and Zontul (2013) put together a comprehensive review of the different models that have been used [1]. Generally, the different techniques can be broken into three different classes: historical, statistical, and machine learning. Historical models are generally the most simplistic model, utilizing either average travel time between stops or average speed between stops to predict future travel times. Statistical models utilize more advanced techniques such as regression, time series analysis and Kalman filtering. Machine learning models are the most complex, and require a large amount of data, which is often difficult to obtain. A more recent review of the field by Choudhary, Khamparia and Gahier (2016) includes more recent techniques such as hybrid models and real time cell phone data[7]. Due to the highly stochastic nature of traffic networks, a combination of techniques is likely necessary to produce the most accurate results.

2.3 Classical Models

2.3.1 Historical Models

Historical models are the most simple of all modeling techniques, but nevertheless can give reasonable accuracy in certain situations. Maiti, Pal, Pal, Chattopadhyay and Mukherjee (2014) showed that historical based models can compete with machine learning models in terms of accuracy (75.56% for historical model vs 76% for neural network)[25]. Historical data based models typically use very simple features such as travel time and average speed. This is well suited to cases where more complex features such as traffic, weather, and schedule are not available, as is the case often in developing countries. The small feature space also results in faster prediction times than larger models such as neural networks which can have huge numbers of parameters. Historical models treat traffic as static across periods such as the week, month or year depending on the amount of data. This assumption is reasonable, because traffic exhibits common patterns which repeat throughout the week, such as rush hour traffic. However in urban settings, the assumption that traffic conditions will stay the same across weeks is invalid. This means that historical models are limited in predicting more complex situations. Nevertheless, they succeed at giving good accuracy predictions in areas with more static traffic conditions such as rural areas.

2.3.2 Statistical Models

The next class of models in terms of complexity is statistical models. Travel time is determined by several variables including weather condition, time of day, intersections, network load and driver behavior. These variables can be used as independent variables to predict the trajectory of buses in a network. However many of the independent variables are latent, making modeling and prediction a difficult problem.

One statistical technique is mathematical time series analysis. In this technique, a mixture of linear or nonlinear functions is used to model the traffic. D'Angelo (1999) showed that this sort of model can produce accurate travel times for cars on a highway over short time-spans [8]. Obviously this is a limited use case. However it proves that there are certain regularities and patterns in seemingly stochastic traffic networks that can be exploited to predict travel times.

Several regression schemes have also been used to predict bus arrival times. Sun, Chen, Song and Wang (2010) used the Autoregressive Integrated Moving Average model in combination with delay models to forecast travel times of buses [35]. The study uses bus data from Tianjin in northern coastal Mainland China. Travel time for buses is broken up into three sections: free travel time, road intersection delay, and stop time delay. Each section is then independently modeled in order to forecast travel time. The underlying model has the advantage of being very simple and fast to compute. However, the model neglects several important factors which contribute to traffic congestion such as weather, seasonality, and driving conditions. Even so, Sun et al. achieved a 20% error rate for the Tianjin bus.

Another common statistical technique involves using Kalman filtering to model bus networks. Kalman filtering is an algorithm which uses several measurements to estimate latent variables by estimating the joint probability distribution over the variables. The technique is commonly used in guidance and navigation systems. Zaki, Ashour, Zorkany and Hesham (2013) used real time GPS data combined with a Kalman filter and a neural network to produce accurate real time predictions of bus arrival times[41]. The study uses data from an Egyptian bus network. The predictions produced have a mean square error in the range of one minute over the course of the route.

2.4 Machine Learning Models

Machine learning has come into prominence in recent years, revolutionizing the fields of computer vision and natural language processing. In particular, neural networks have shown a lot of recent success for their ability to tackle a wide range of problems if given enough data. Neural networks are also particularly effective at modeling bus networks. In particular, bus networks have a very large, complex feature space with many latent variables. Traditional methods of feature engineering are not effective at computing all of the these features. This may explain the effectiveness of machine learning techniques on bus networks. In particular, machine learning models are used to do feature selection and inference on bus networks. The models which are suited for this use case are kernel methods and neural networks.

2.4.1 Kernel Methods

Kernel methods are a technique for producing nonlinear classifiers from linear models via a feature transformation. A kernel function is applied to to the standard input features to produce the kernel representation. This kernel function is typically nonlinear, and represents an inner product. The kernel representation is then used to set the parameters of a model, which is typically linear. Due to the nonlinear transformation of the input data, the resulting model produces nonlinear classification with respect to the original input space. This technique is useful because it only relies on inner products, so the computation can be done very efficiently. The technique even allows for infinitely dimensional feature space depending on the choice of kernel function. The radial basis function is a popular example of a kernel function which results in an infinitely dimensional transformed feature space.

Sinn, Yoon, Calabrese and Bouillet (2012) used kernel regression to update arrival time estimates given real time GPS measurements[32]. When compared to linear regression or K-Nearest Neighbors (KNN), the kernel regression model showed the best accuracy. The study uses data from the public bus system in Dublin, Ireland. In general, the success of kernel regression suggests a complicated nonlinear feature space.

Another technique for modeling buses uses the relevance vector machine (RVM) algorithm. RVM is a variant of the support vector machine (SVM) algorithm. SVM produces a binary classifier with the maximal margin linear separator between data points of the opposite class. This gives certain guarantees about the performance of the classifier on unseen data. Additionally, SVMs can be combined with kernel methods to produce nonlinear classifiers. Furthermore, SVMs can be trained in an online fashion, which is important for machine learning problems, which may need to be trained on very large datasets. RVM is simply an extension of the SVM which gives probabilistic classification instead of deterministic classification. The RVM was developed by Tipping in 2003[36].

Yu, Wu, Chen, and Ma used the RVM algorithm to estimate predict bus headway[39]. The feature set they used included bus headway time series, travel time, and passenger demand. The study used two bus routes in Beijing, China, Compared with SVM, genetic algorithms, Kalman filters, KNN, and neural networks, the RVM model performed the best. With a confidence interval of 95%, over 95% of actual bus headways fell into the prediction interval. The success of kernel based methods also motivates the use of neural networks, which can be more powerful at learning complex features.

2.4.2 Neural Network Models

Neural networks are one of the most powerful and effective machine learning models in current usage. Neural networks work in an end to end manner, learning both the feature space and doing prediction. They work by combining many simple linear classifiers with a nonlinear activation function (called neurons) over several layers. Given enough neurons, a neural network can approximate any function [17]. Despite their incredible power, neural networks do a better job of avoiding overfitting than other powerful models. They are especially well suited for problems with vast amounts of data. For an overview of neural networks, see *Neural Network Design* [9].

Neural networks have long been used for predicting bus arrival times. As early as 1992, Faghri and Hua showed that neural networks have several applications within transportation engineering [12]. Chien, Ding, and Wei (2002) developed a neural network model for predicting bus arrival times [6]. The model contains two variants: a link based model and a stop based model. The input features for the neural network include speed, volume, delay, passenger demand and arrival times. In addition to the neural network, the model contains a adaptive algorithm which updates the predictions in real time. The neural network model with the adaptive algorithm performs well even when predicting multiple stops ahead. Chien et al. use the simulation model CORSIM, which Chien and Ding developed with real bus data from route 39 of the New Jersey Transit Corporation [10].

Jeong and Rilett (2004) compared several popular models for bus networks[20]. This included a historical data based model, a regression model and a neural network. The neural network was found to give the best performance by a large factor. The neural network showed over a 70% improvement in accuracy over the best historic based model. In all cases the neural network performed the best, with the historical model coming in second and the regression model in third. Neural network models have shown the best accuracy in general across several studies, however combining neural networks with other techniques has shown superior performance in many cases.

2.4.3 Hybrid Models

Given the complexity of modeling bus networks, a single model may not be sufficient to predict arrival times with the best accuracy. This has given rise to a number of hybrid models which combine techniques in order to produce better predictions.

One common technique is to use real time data from cell phones in order to produce real time estimates for arrival times. Zhou, Zheng and Li (2012) used Android phones to collect data from passengers on a bus to generate estimates for arrival times at various stops[42]. The system relies on user participation and uses lower energy mechanism for localizing the bus such as cell towers and movement statuses. This has the advantage of not relying on a centralized service. Additionally, they found that the performance of the system was much better than the GPS supported solution.

Khetarpaul, Gupta, Malhotra and Subramaniam combined neural networks with fuzzy logic systems to general effective real time bus arrival time predictions [21]. Fuzzy logic contrasts with boolean logic in that truth values of variables can be any number between 0 and 1, instead of just 0 or 1. Khetarpaul et al. use fuzzy clustering on the input data to model to generate inputs to a set of neural networks. In this way, Khetarpaul et al. claim the shortcomings of each model are made up for by the other. The model generates reliable predictions for bus arrival times for bus data from Dublin Ireland.

Raut and Goyal (2012) show how recurrent neural networks (RNNs) can be used to predict bus arrival times given the current weather conditions[29]. The underlying model estimates the current traffic load given historical data and current weather conditions. This can data then be used to then infer arrival times for buses at future stops. This shows the possibility of using more advanced network architectures to model buses networks.

2.5 Deep Learning Techniques

The onset of deep learning has revolutionized modern machine learning. It has allowed for self driving cars and Siri. The principle concept of deep learning is adding many layers in a neural network. The most common neural network architecture are convolutional neural networks (CNNs) and RNNs. CNNs perform the state of the art in computer vision tasks by repeatedly applying a series of convolutional filters to input images. RNNs work by passing the output of one computation to the input of the next, thereby remembering information. They have revolutionized natural language processing, to a large extent through the long short term memory (LSTM) architecture. Hochreiter and Schmidhuber (1997) introduced LSTMs, which work by combining a series of gates to store and update its internal memory[16]. In order to achieve the best accuracy on bus modeling problems, deep models may be necessary. For an overview of recent deep learning, see LeCun, Bengio and Hinton's *Deep Learning*[24].

In addition to natural language processing, deep RNNs have shown success in time series prediction, which is the same realm as predicting bus arrival times. Prasad and Prasad (2014) used deep RNNs to identify epileptic seizures from electroencephalography signals[28]. This task is typically done by a trained doctor. RNNs can be used on time series data to either perform classification or regression depending on the use case. A similar model used for regression can be used to predict bus arrival times.

Borovykh, Bohte and Oosterlee (2017) used the deep convolutional network architecture Wave-Net for conditional time series forecasting [5]. Wave-Net is a neural network derived from a family of orthonormal wavelets [3]. Borovykh et al. compared the architecture to a traditional LSTM network and found that it was able to learn effectively without the need for long term historical data.

Yang, Nguyen, San, Li and Krishnaswamy (2015) showed how deep convolutional neural networks can be applied to human activity recognition (HAR) based on time series signals from body sensors[38]. The prevalence of deep neural models in time series analysis suggests that deep architectures may work well for predicting bus arrival times.

Chapter 3

Methods

3.1 Data

3.1.1 Dataset

The dataset for this research comes from the Massachusetts Bay Transportation Authority (MBTA). The data is publicly available and consists of the following:

- GPS data for buses, consisting of latitude, longitude, route, and route direction
- Route data, describing the GPS location of stops along a route
- Bus timetable information, which shows the times when a bus arrives at each stop in a route
- Bus metadata

Buses, routes and stops are described by ids. Bus stops also have human readable names indicating their location. The dataset spans several years, and includes all of the buses and routes in Boston. This dataset is unique from previous research in its extent both in size and duration. Previous studies use several months of data, whereas this dataset allows the analysis of longer term trends, including seasonal trends.

Figure 3-1 shows GPS data for Route 1. The route starts in Cambridge at Massachusetts Avenue and Holyoke Street, and ends in Boston at Dudley Station. The

GPS data is fairly accurate for the most part. However, in cities like Boston, GPS data suffers from an issue known as urban canyons. This occurs on relatively narrow streets with very tall skyscrapers, where a canyon effect is created. The large buildings obscure satellite signals, creating very noisy GPS signals. Certain streets in Boston suffer from this issue, however Route 1 does not touch those streets, resulting in relatively clean GPS data. All of the data points are well localized to the street.

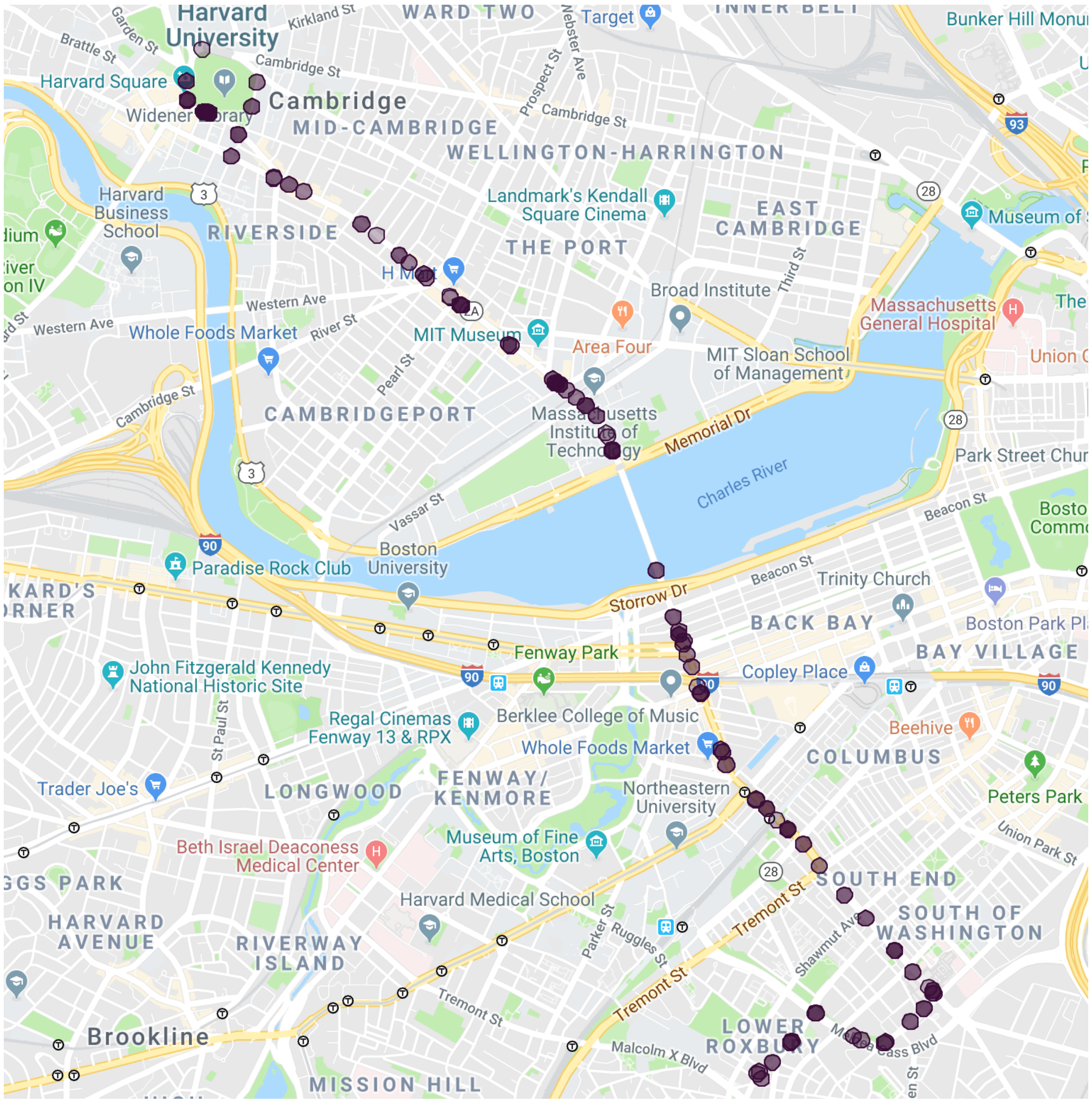
Figure 3-1: GPS Data from Route 1 serving Cambridge and West Boston

3.1.2 Interpolating Trajectories

The data provided consists of tuples of latitude, longitude, bus id, and timestamp. However, the data needed to train the model involves the arrival times at each stop, so the trajectory of the bus must be interpolated from the noisy GPS data. This is done via the following process:

1. Tuples are grouped by bus id and route, then sorted in time
2. The GPS trajectory is then converted from latitude and longitude into a 2D Cartesian projection
3. The entire trajectory of the bus is segmented into each individual trip along the route
4. The 2D coordinates are then projected onto the route using a Gaussian noise assumption, shown in Figure 3-2
5. This (x,y,t) data is then converted from a 2D position on the map to a 1D distance along the route, because all buses follow the same route
6. Stop data of the form (latitude, longitude) is then converted into the same 1D distance along the path.
7. The trajectory of the bus is then interpolated in a piecewise linear fashion, assuming a constant speed between adjacent points.
8. The trajectory is sanitized by removing any erroneous data by thresholding the velocity of the vehicle
9. This trajectory can then be used to determine the arrival time at each stop by determining the timestep when each bus was within a certain threshold of the stop. This is illustrated in Figure 3-3.

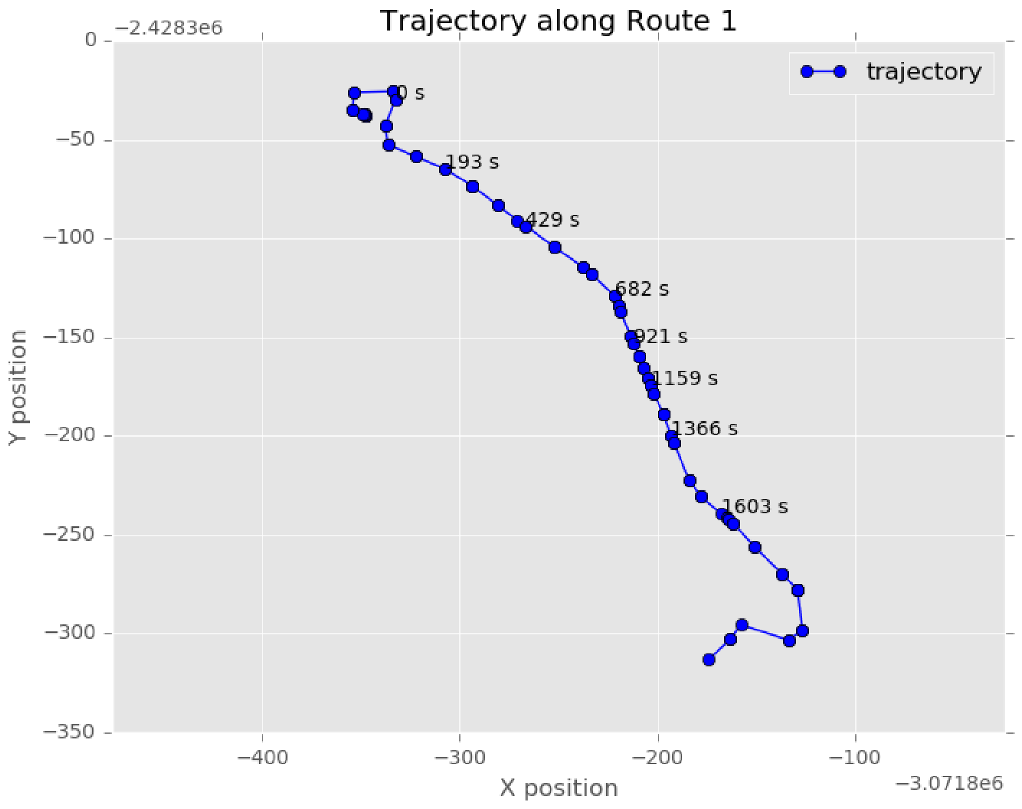

Figure 3-2: (x,y,t) trajectory of bus along route 1. The route starts on the top left and finishes on the bottom right.

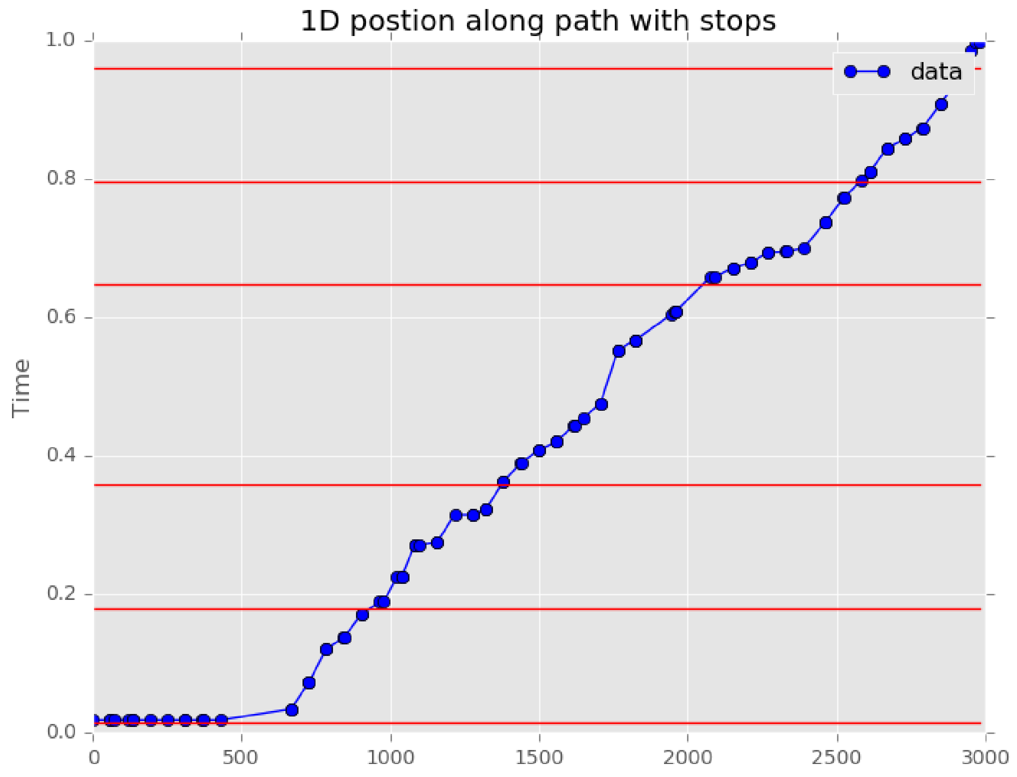

Figure 3-3: Interpolated trajectory of bus (blue) with stop locations overlain (red). Intersections indicate the bus arriving at a stop.

3.1.3 Features

The following are a number of features which are commonly used as variables to model bus networks.

- Arrival Time: Time the bus arrives at a stop
- Travel Time: Between adjacent stops
- Dwell Time: The amount of time a bus spends at a stop
- Schedule Adherence: The difference between the projected arrival time and the actual arrival time
- Headway: The difference in arrival time for two adjacent buses at a specific stop
- Direction: For a route which runs two directions
- Weather
- Traffic Congestion
- Day of Week
- Month of Year

The arrival time at each stop is computed as shown above. Travel time can them be trivially computed by taking the difference between two arrival times, minus the dwell time. Schedule adherence is computed by comparing the arrival time of the bus the closest listed arrival time. Other features are either derived from the raw data or pulled from outside sources.

3.1.4 Distribution of Arrival Times

Figure 3-4 is a histogram of the difference in arrival times between adjacent buses at a particular stop on Route 1. Three plots are overlain on the distribution. If the buses ran perfectly on schedule, the distribution should be concentrated about

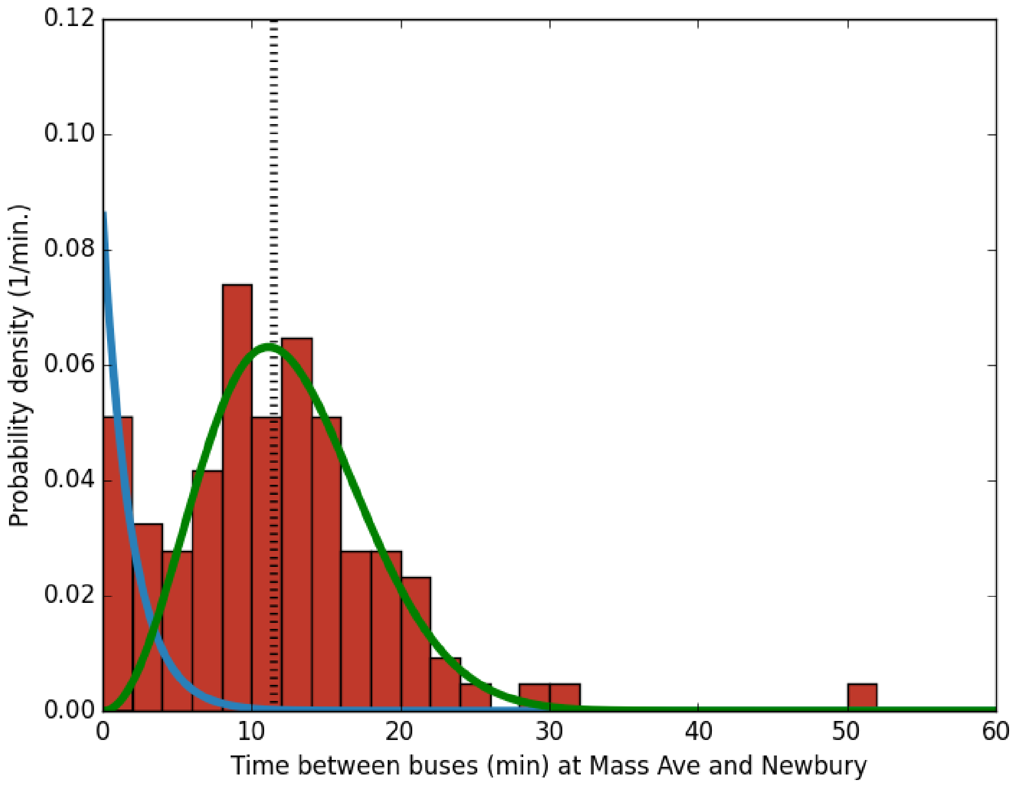

Figure 3-4: Distribution of intervals between bus arrival times at Newbury and Mass. Ave.

the mean, indicated by the dotted black line. If the buses are perfectly random and independent, the distribution should follow a Poisson distribution, indicated in green. However buses are not independent. In particular, a bus will not pass another bus, which leads to the high concentration about zero in the distribution. This effect is compounded by bus clumping. The exponential distribution captures the behavior of the buses near the origin. More complicated distribution like the GUE have been used to describe intervals between buses. A study out of Cuernavaca, Mexico [2] first identified this effect, and the same effect is observed in the New York public transit system.

3.2 Implementation

All of the models were developed in Julia [4] using Knet [40]. Knet is an imperative machine learning library which supports automatic differentiation and GPUs. Models are expressed in pure Julia code, then auto-diff is used to generate gradients which can used to train the model. This presents several advantages over libraries declarative frameworks like tensorflow. In declarative frameworks, the computation graph is generated at compile time, which creates a layer of abstraction between the computation and the implementation. This layer of abstraction can be useful for optimizing performance. However it also adds a level of complexity and makes debugging more difficult. This is because the code executing the computation is generated by the compiler instead of the implementer.

This contrasts with imperative libraries like Knet, where the code executing the computation is generated by the implementer. This makes debugging much easier, and gives the implementer the full power of the underlying language. Instead of defining operations, models are expressed explicitly, making code much more succinct. Knet also gives the advantage of allowing the developer to express the model in pure Julia code. In tensorflow, the user is limited to predefined operations, and defining a new operation requires adherring to the tensorflow mini-language.

Knet includes several optimizers including stochastic gradient descent, RMSprop and Adam, as well as many common utility operations and layers.

3.3 Models

Three model architectures were used to predict bus arrival times:

- Multilayer Perceptron (MLP)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)

All three models are applied in regression mode, taking as input knowledge of the bus network, and outputting the expected arrival time at future stops. Each of the three architectures is successful in predicting arrival times, with the RNN achieving the highest accuracy.

3.3.1 Multilayer Perceptron

Multilayer perceptrons, also known as vanilla neural networks, are the most basic neural network architecture. They consist of a series of dense linear transformations followed by element-wise non-linear transformations. Each of these steps is known as a fully connected layer, because at each layer, neurons are connected via a weight matrix to every neuron in the next layer. Popular non-linear transformations include the sigmoid, hyperbolic tangent, and rectified linear (ReLU). Figure 3-5 shows the function of a single ReLU.

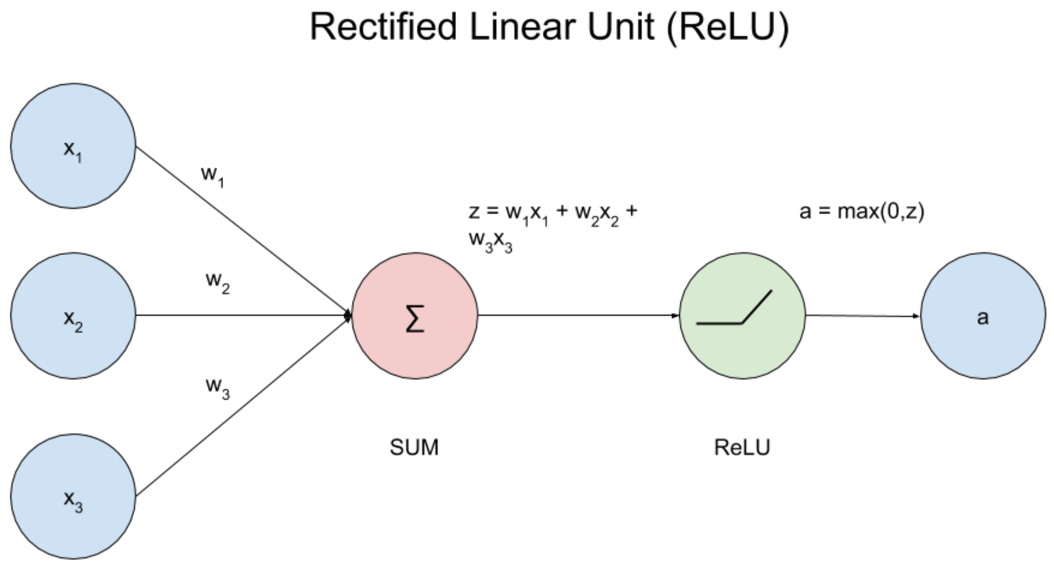

Figure 3-5: Rectified Linear Unit

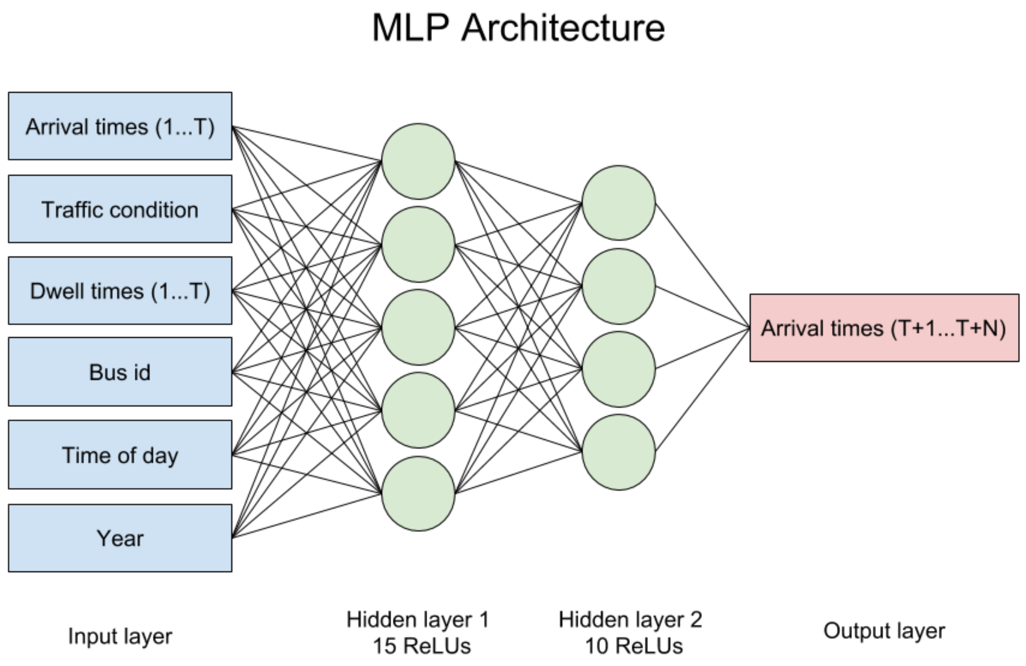

Figure 3-6: Multilayer Perceptron Architecture: Two fully connected hidden layers

The architecture used in this research uses a relatively small neural network with two hidden layers, shown in figure 3-6. The number of neurons used in each layers was 15 and 10. This small model size was used to combat overfitting. The model was trained using a mean squared error loss function for 50 epochs, with a learning rate of 0.0001. The optimizer used was stochastic gradient descent. Mean square loss was used as it is suitable for continuous valued regression tasks.

The following is the code which implements the MLP.

```
# w is a vector of weight matrices
# x is the input vector
function mlp(w,x)
    for i=1:2:length(w)
        # Apply linear transformation
        x = w[i]*x .+ w[i+1]
        if i<length(w)-1
            # Apply ReLU non-linearity
            x = max.(0,x)
        end
    end
    return x
end
```

3.3.2 Convolutional Neural Network

Convolutional neural networks are used primarily for image classification. Convolutional filters are applied iteratively to an image to detect low and high level features in the image. However they are also applicable to regression tasks. For time series prediction, 1D convolutions are used, as opposed to the 2D convolutions applied to image tasks. 1D convolutions are therefore suitable for the time series task of predicting bus arrival times.

The architecture used contains two convolutional layers followed by two fully connected layers, shown in figure 3-7. The first convolutional layers consists of 20 filters with 5 neurons each, and the second filter consists of 5 filters with 5 neurons each. The number of neurons in the fully connected layers depends on the number of input features and number of stops being predicted. Again, mean squared error loss was used to train the model.

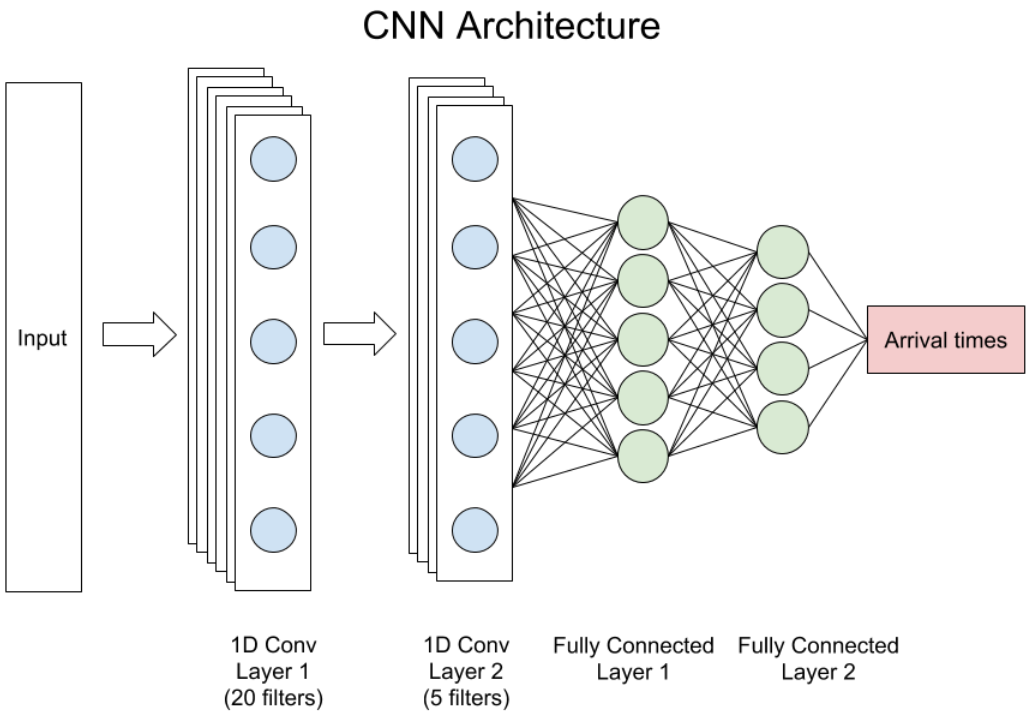

Figure 3-7: Convolutional Neural Network Architecture: Two convolutional layers and two fully connected hidden layers

The following is the code which implements the CNN.

```
# w is a vector of weight matrices
# x is the input vector
function cnn(w,x)
  x = reshape(x,(length(x),1,1,1))
  n = length(w)-4
  for i=1:2:n
    # Apply a convolution with max pooling
    x = pool(max.(0, conv4(w[i],x) .+ w[i+1]))
  end
  # Flattens the vector before the fully connected layers
  x = mat(x)
  for i=n+1:2:length(w)-2
    # Apply linear transformation and ReLU activation
    x = max.(0, w[i] * x .+ w[i+1])
  end
  return w[end-1] * x .+ w[end]
end
```

3.3.3 Recurrent Neural Network

Recurrent neural networks (RNNs) extend feed forward neural networks such as CNNs and MLPs by adding persistent state. Typically RNNs are used to model sequences, with the internal state of the network being updated as the sequence is fed through it. This makes them particularly well suited for time series analysis.

This research uses an Elman network with a hyperbolic tangent activation[11]. The network used consists of a single hidden layer with 10 recurrent neurons. In each time step, a single element of the input sequence is fed into the network, producing an output and a hidden state. The output is used to train the network, and the hidden state is fed into the network on the next timestep. This is illustrated in figure 3-8. The network is fed some empty initial state, in this case a zero vector.

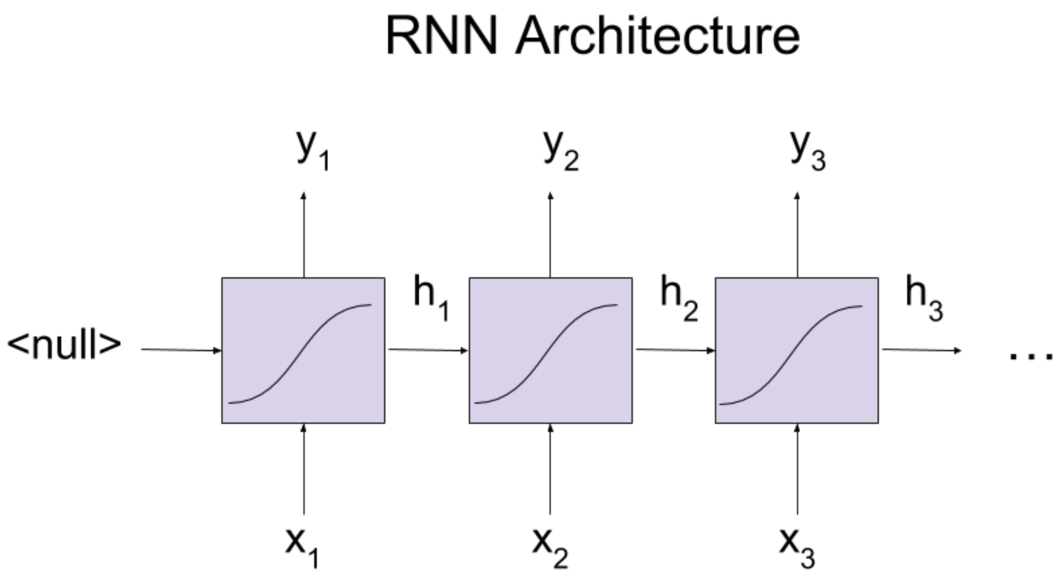

Figure 3-8: Recurrent Neural Network Architecture: One hidden layer with 10 neurons

The network was trained via back propagation through time (BPTT)[37]. This allows for training the network across an entire sequence, and reinforcing the correct outputs at each step of the sequence. Again, root mean square error across the predictions was used as the loss function.

The following is the code which implements the RNN in Julia.

```julia
# w is a vector of weight matrices
# x is the input vector
# h is the state
function rnn(w,x,h)
    # Generate the new state from the previous state and the input
    h = tanh.(w[1]*vcat(x,h) .+ w[2])
    # Generate prediction from state
    y = w[3]*h .+ w[4]
    return (y,h)
end
```

Chapter 4

Results

4.1 Linear Baseline Model

First a simple linear model was trained to establish a baseline accuracy. Figure 4-1 shows the test and training loss for the linear model.

The loss curve is smooth, and follows our expectation for the training loss to be slightly better than the test loss. Furthermore, this model allows us to establish the baseline accuracy of 60 seconds for test RMSE. The further models will be compared against the linear model.

4.2 MLP

The next model trained was the multilayer perceptron. The MLP has much more expressive power than the linear model, so one would expect that the training loss will be significantly lower. However the high variance of the model can lead to poor performance on the test set. Figure 4-2 shows the loss curves for the MLP model. As expected, the training loss is significantly lower, with a RMSE of around 30 seconds. The test loss is significantly higher, but still better than the linear model. This shows that the model was able to capture more of the variability in the data.

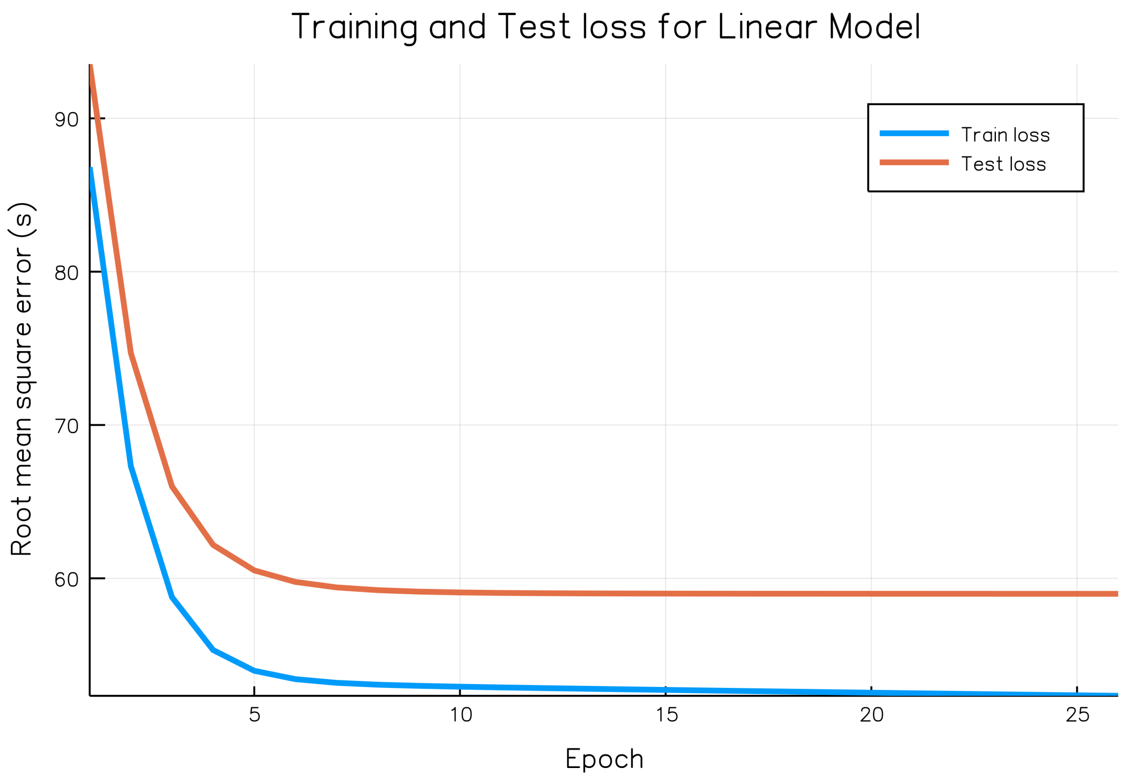

Figure 4-1: Loss for linear model (lower is better). The plot shows characteristic overfitting, with a test loss baseline just under 60 seconds.

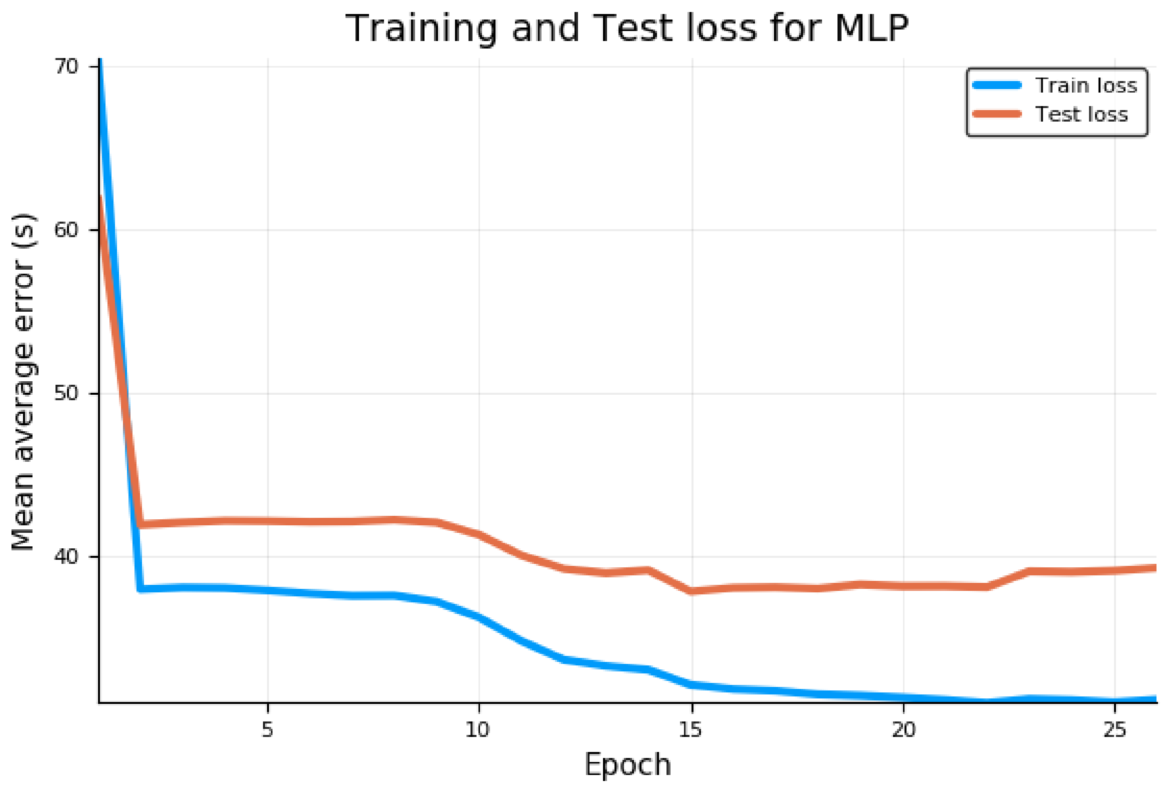

Figure 4-2: Loss for MLP model (lower is better). The MLP can capture more of the variation in outputs than the linear model.

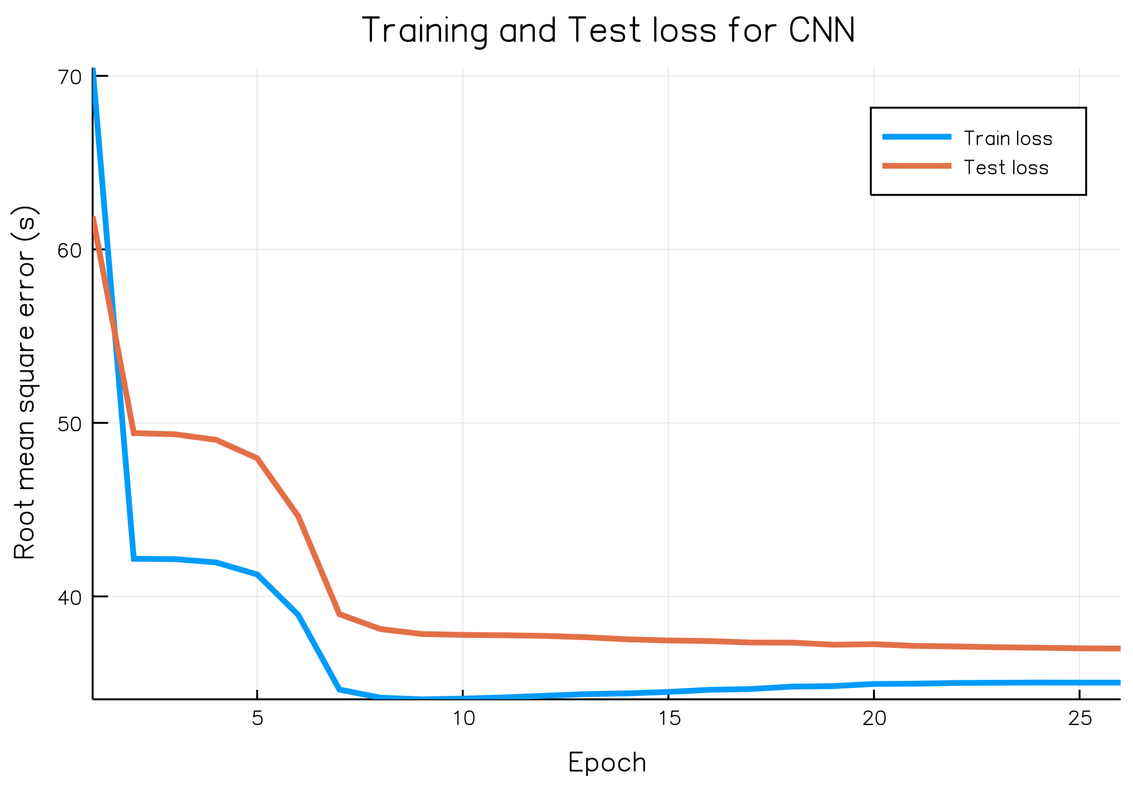

Figure 4-3: Loss for CNN model (lower is better). The CNN improves upon the accuracy of the MLP.

4.3 CNN

Figure 4-3 shows the loss curves for the CNN model. The CNN model improves slightly on the MLP model. It achieves a test RMSE of around 37 seconds. The CNN model is significantly larger, and takes longer to train than the MLP. An interesting observation is that the loss curve for training starts increasing after around 10 iterations. The larger model seems to have some difficulty with convergence. The model is trained until the validation accuracy stops decreasing, which occurs at around 25 epochs.

4.4 RNN

The final model trained was an RNN. RNNs take significantly longer to train due to the back propagation through time algorithm requiring an entire sequence to be feed through the network for each update. In this case, the prediction optimization for future stops was optimized rather than the entire sequence.

RNNs are considered more difficult to train than other forms of models because of the exploding/vanishing gradient problem [27]. During back propagation through time, the signal to train the network passes through the neural network several times. If the magnitude of the gradient does not stay relatively constant between time steps, the gradient can exponentially blow up or shrink. This reduces the quality of gradients and makes training the networks more difficult. The same effect occurs in large feed forward networks with many layers, such at deep CNNs. In feed forward networks, this effect is mitigated with batch normalization[18]. RNNs also have an issue with exploding and vanishing gradient, because unrolled RNNs because each timestep of training adds another layer. However for regression problems, the signal is only received at the end, and then has to be propagated back over the network several times. For this reason, the size of RNNs for regression tasks is limited.

Figure 4-4 shows the loss curves for training this model. The loss curves are much noisier than the other models. However, the model converges eventually to just under 30 seconds for training and just over 30 seconds for test. This represents the best overall loss across all of the models.

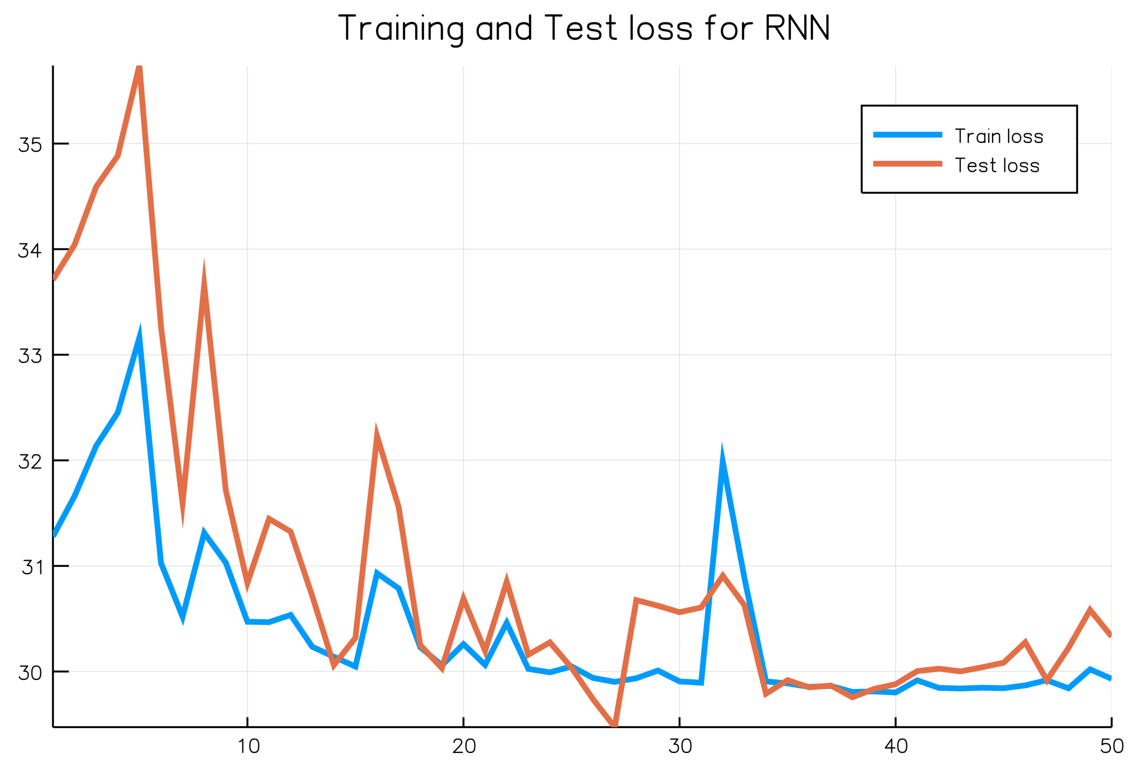

Figure 4-4: Loss for RNN model (lower is better). The RNN achieves the lowest overall loss.

Table 4.1: Root mean square error for all models

	Linear	MLP	CNN	RNN
Train	54.3	31.2	35.0	29.8
Test	59.2	39.9	37.5	30.4

Table 4.2: R^2 score for all models

	Linear	MLP	CNN	RNN
Train	0.636	0.8798	0.8488	0.8904
Test	0.5673	0.8035	0.8264	0.8859

Table 4.1 summarizes the final loss values for all of the models, and table 4.2 summarizes the R^2 score for each of the models.

Chapter 5

Evaluation

5.1 Model Comparison

Figure 5-1 shows the root mean square error across all of the models. Linear model serves as a baseline for comparison. All of the neural network models improve upon the accuracy of the linear model, as is expected [20].

The MLP and CNN architectures achieve similar accuracy. Both models fall under the class of feed forward models, meaning data is feed from input to output without any feedback loops. However, the MLP model is symmetric about the inputs, meaning that no temporal aspects can be extracted from the input. This is by nature of the network being fully connected, so each input is treated equally. However in the case of the CNN, the model is not symmetric about the inputs. Specifically, the 1D convolutions act in order on the input sequence. In image processing, this ordering is used to capture translational invariance. However for time series prediction, convolutions capture high level patterns in sequences which are then used as features for prediction in the fully connected layers. This temporal aspect is possibly what gives the CNN advantage over the MLP. Additionally the CNN is a larger model, with many more parameters than the MLP. This increases the variance of the model and allows it to capture more patterns in the data.

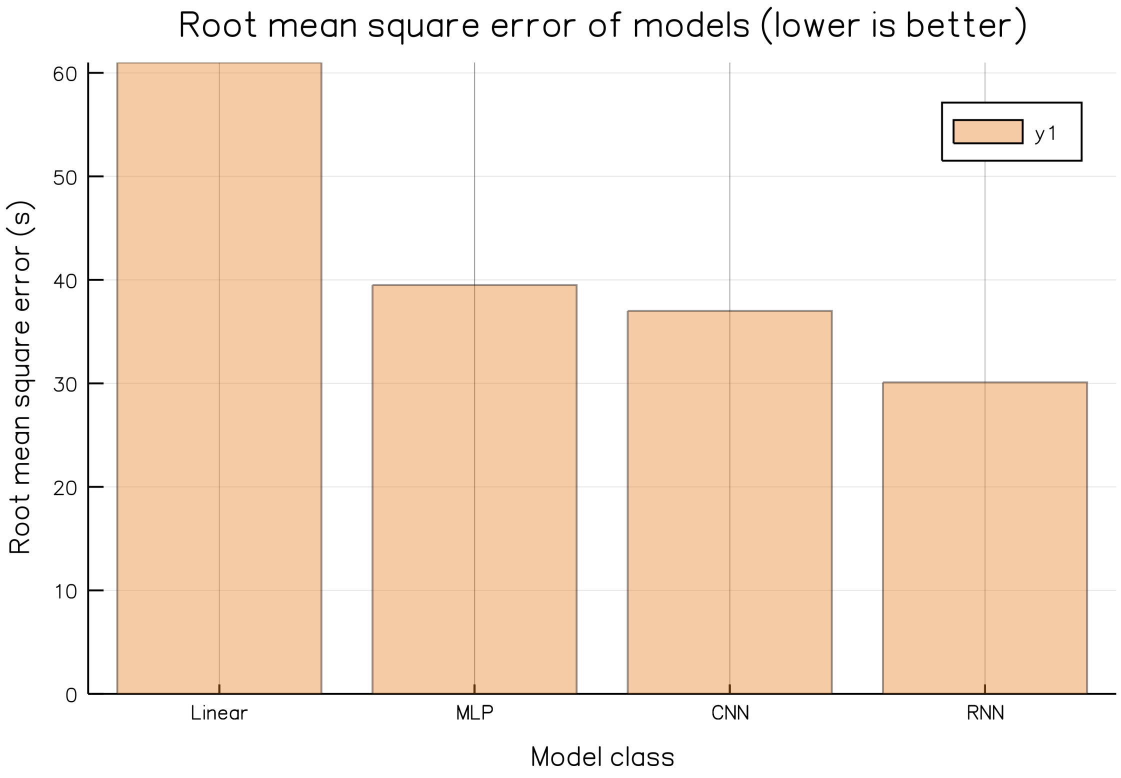

Figure 5-1: Root mean square error across all models

Figure 5-2 shows the coefficient of determination (R^2) score for each of the models. The R squared score shows how much in the variance of the data the model is able to capture. Zero indicates that the model is as good as random guessing, while a score of 1 indicates the model perfectly captures the data. The linear model achieves an R^2 score of around 0.56. This shows that the linear model does not do a great job of modeling the data, as is expected. The linear model does not have enough expressivity to capture some of the more subtle patterns in the data.

The best coefficient of determination comes from the RNN, which achieves around 0.89. This represents much better performance than the linear model. However, there is still a lot of variance in the underlying data which the model cannot capture. This gets back to the issue of the stochastic nature of traffic. Large deviations can occur in traffic networks for various reasons which makes prediction difficult. For example traffic accidents and construction are not included as features to the model, and these factors make a drastic impact on the underlying network. To some extent, incorporating more data into the feature set will increase the amount of variance the model can capture. However, this also requires increasing the size of the model, which may lead to overfitting, and decreased inference performance.

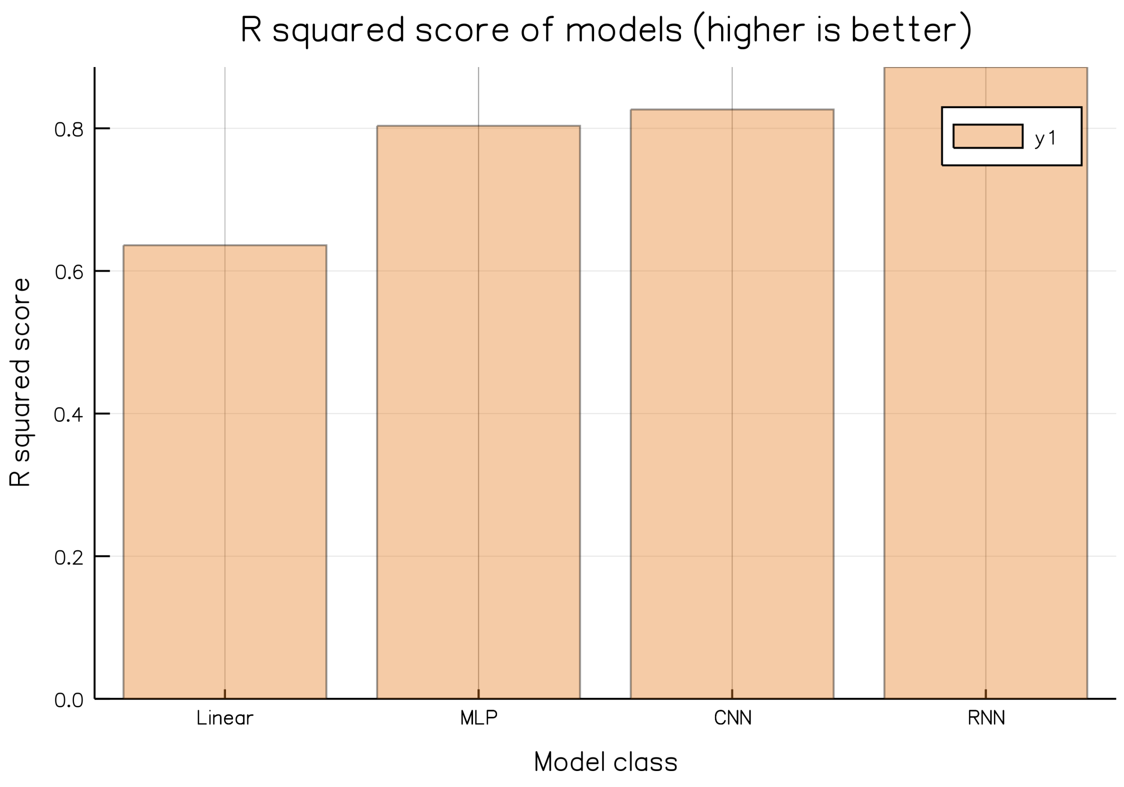

Figure 5-2: R^2 score across all models

5.2 Analysis of Strengths and Weaknesses

In order to fully model a traffic network, the following categories of information must be captured.

1. Spatially local, temporally local: Information about the bus at the current timepoint, including its location, number of passengers, speed etc.

2. Spatially global, temporally local: Information about the overall traffic network currently, including traffic conditions, number of buses in the network, construction, weather etc.

3. Spatially local, temporally global: Information about each bus which create trends over time, including the bus id, and driver

4. Spatially global, temporally global: Information about the topology of the network overall, including distances between stops, number of intersections, relative location of routes etc.

The linear model can only reason globally, because each input parameter gets one weight. Therefore no temporal aspects can be captured.

The MLP can do a good job of capturing global data and global trends, so it will be good at modeling numbers 3 and 4. However it is not really designed for modeling numbers 1 and 2. Although the input data is inherently a sequence, the MLP treats all inputs equally, so the sequential nature of the data is immediately lost.

Convolutional nets are also able to model number 3 and 4 well, and may be able to capture certain patterns of numbers 1 and 2. The convolutions respect the sequential nature of the input data, so patterns over time can be captured. Additionally, CNNs are good at capturing patterns in very high dimensional spaces. For example, Alpha Go [31] uses a convolutional network to read the Go board and detect similarities between different board positions to create strategies. Despite the incredible state space of the game of Go, with over 10^170 possible board positions, the convolutional net can detect which positions are equivalent or similar. Given access to the entire state of the traffic network, a CNN would do a good job of determining which factors are important for determining travel time. However this would require an immense amount of training data, because the network has to be able to recognize an immense number of states. One possible application of CNNs for traffic analysis would be an autoencoder, which takes in the entire state of a traffic network, and extracts the features which are relevant for the prediction task at hand. This helps reduce the dimensionality of the problem. However, to truly capture sequential data, RNNs are more suitable.

Compared with the linear, MLP, and CNN models, the RNN is the only one which has state. This allows it to capture certain properties which the other models cannot. Recurrent neural networks are well suited to model numbers 1 and 2, and certain types of RNNs can capture numbers 3 and 4. More specifically, long short term memory networks (LSTMs) do a good job of capturing long term and short term dependencies [16]. This makes RNNs the most general model for modeling traffic data, and may explain why they perform the best. This is because RNNs have memory which is selectively updated based on new data. In the case of predicting bus arrival times, there is a lot of state which updates over the course of a route. The number of passengers on the bus, for example, is a quantity which changes over time and is very important in determining how long the bus will stay at each stop. A feed forward net will not be able to represent this. This may explain why the RNN performs much better than its feed forward counterparts. Furthermore, there is a high degree of uncertainty in traffic networks which requires updating your belief of the state based on new information. RNNs are very well suited for doing this. In a scenario with free access to all data related to the traffic network, a CNN could be used to distill the entire state into a lower dimensional space. Then an RNN can be used on the sequential data from each bus to generate accurate predictions based both on the global and local behavior.

Chapter 6

Further Work

To improve upon the accuracy or traffic models, more data sources need to be incorporated. Traffic networks are extremely complex, which makes modeling them difficult. Good models require a lot of data, which forces researchers to use simulations instead of real data [15]. This is one current limitation of the field. One potential avenue is incorporating more nuanced traffic data such as volume, delay, and average speed of all cars in the network. This data is easily available in traffic simulators, but real high resolution traffic data is less available.

Another strategy to improving the accuracy of predictions is increasing the expressiveness of the model. Given that RNNs generated the best results, other RNN architectures may also be successful. Long short term memory networks (LSTM) would be a good avenue of pursuit because they model dependencies at various time scales. Additionally, a hybrid model may do a good job a solving the different tasks necessary to predict traffic. Specifically, a CNN can be used to model the current state of the overall network, while and RNN can be used to generate real time predictions of trajectories for each of the buses in the network. In this way work can be reused because all buses share the same state embedding.

To fully model traffic networks, a holistic approach needs to be taken. Traffic conditions are determined not only by local behavior, but by the complex interactions of all the vehicles in the network. Bus routes cannot be treated as independent. Therefore good models will need to reason about the network as a whole. A good path to follow is generating some global state embedding with a large convolutional neural network, then using this embedding to update beliefs of a reccurent network in an online fashion. This allows for real time updating of predictions.

Chapter 7

Conclusion

7.1 Implications

7.1.1 Neural networks can handle the high volume of data generated by traffic networks

One issue with using larger models is overfitting. Linear models limit the variance of the estimator class, improving the generalization of the model. However this research shows that neural networks models can combine several different data sources and pull out relevant features, with little overfitting. The models in this research do overfit to some extent, but their generalization scores are better than the linear model, despite the models having an extremely high variance. In particular, the CNN trained in this research improved almost two-fold compared to the linear model, with around a thousand-fold increase in the number of parameters. This may indicate that even larger models may be effective at traffic prediction. Traffic networks generate huge amounts of data, especially given the advent of mobile apps such as Google Maps and Waze. All of this data can be leveraged to improve the operating efficiency of the network.

7.1.2 Recurrent neural networks are effective at modeling bus time series data

Traffic networks generate time series data, which are well modeled by RNNs. Feed forward models like MLPs and CNNs do a good job of feature extraction and pattern recognition, however their ability to model trends is limited. CNNs can do 1D convolutions which model time series data to some extent, however they cannot remember long term patterns. This is where RNNs can shine. This research shows that RNNs outperform feed forward networks on traffic modeling tasks. Traffic networks have a lot of latent state which cannot be directly measured. For example the data used in this research does not have any passenger counts, which are vital to estimating dwell time at each stop. However RNNs can model latent features like this over long and short time periods. This behavior underlies their effectiveness at predicting bus arrival times.

7.2 Contributions

In summary, in this thesis I:

1. Used MBTA GPS data from buses to model their trajectories over a 3 year period
2. Processed the GPS data to extract high level features such as arrival times at stops and wait times at each stop
3. Gathered more data related to modeling the traffic network such as traffic estimates and bus metadata
4. Used all of the processed data to train a series of models including linear, MLP, CNN and RNN
5. Generated predictions with these models and showed that neural networks can generate accurate predictions of travel times

6. Showed that large feed forward models can generate highly accurate predictions without overfitting

7. Showed that RNNs outperform feed forward models

8. Gave an explanation for this improvement based on time series analysis and latent variables

Traffic networks are notoriously difficult to model, and classical methods struggle to capture high levels of variation in the network. Most techniques for predicting arrival times are moving from deterministic to learning based algorithms. The explosion in the amount of data available in traffic networks now allows larger, more accurate models of traffic. These models can be used to generate predictions and deliver these predictions to passengers, decreasing wait times and increasing the overall efficiency of the network. Additionally, the increase in available data necessitates larger models and new architectures. This research analyzed the effectiveness of the three most common neural network architectures and found that recurrent neural networks show the best accuracy. These models can be applied to reduce passenger wait times and increase the reliability of bus networks.

Bibliography

[1] Mehmet Altinkaya and Metin Zontul. Urban bus arrival time prediction: A review of computational models. *International Journal of Recent Technology and Engineering (IJRTE)*, 2(4):164–169, 2013.

[2] Jinho Baik, Alexei Borodin, Percy Deift, and Toufic Suidan. A model for the bus system in cuernavaca (mexico). *Journal of Physics A: Mathematical and General*, 39(28):8965, 2006.

[3] Bhavik R Bakshi and George Stephanopoulos. Wave-net: A multiresolution, hierarchical neural network with localized learning. *AIChE Journal*, 39(1):57–81, 1993.

[4] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[5] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[6] Steven I-Jy Chien, Yuqing Ding, and Chienhung Wei. Dynamic bus arrival time prediction with artificial neural networks. *Journal of Transportation Engineering*, 128(5):429–438, 2002.

[7] Rubina Choudhary, Aditya Khamparia, and Amandeep Kaur Gahier. Real time prediction of bus arrival time: A review. In *Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on*, pages 25–29. IEEE, 2016.

[8] Matthew D'Angelo, Haitham Al-Deek, and Morgan Wang. Travel-time prediction for freeway corridors. *Transportation Research Record: Journal of the Transportation Research Board*, (1676):184–191, 1999.

[9] Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. *Neural network design*. Martin Hagan, 2014.

[10] Yuqing Ding, Steven Chien, and Noreen Zayas. Simulating bus operations with enhanced corridor simulator: Case study of new jersey transit bus route 39.

Transportation Research Record: Journal of the Transportation Research Board, (1731):104–111, 2000.

[11] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[12] Ardeshir Faghri and Jiuyi Hua. Evaluation of artificial neural network applications in transportation engineering. *Transportation Research Record*, 1358:71, 1992.

[13] Association for Safe International Road Travel. Annual Global Road Crash Statistics, 2018.

[14] Yan V Fyodorov. Introduction to the random matrix theory: Gaussian unitary ensemble and beyond. *London Mathematical Society Lecture Note Series*, 322:31, 2005.

[15] Abolhassan Halati, Henry Lieu, and Susan Walker. Corsim-corridor traffic simulation model. In *Traffic Congestion and Traffic Safety in the 21st Century: Challenges, Innovations, and OpportunitiesUrban Transportation Division, ASCE; Highway Division, ASCE; Federal Highway Administration, USDOT; and National Highway Traffic Safety Administration, USDOT.*, 1997.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[19] Aukosh Jagannath and Thomas Trogdon. Random matrices and the new york city subway system. *arXiv preprint arXiv:1703.02537*, 2017.

[20] Ranhee Jeong and R Rilett. Bus arrival time prediction using artificial neural network model. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 988–993. IEEE, 2004.

[21] Sonia Khetarpaul, SK Gupta, Shikhar Malhotra, and L Venkata Subramaniam. Bus arrival time prediction using a modified amalgamation of fuzzy clustering and neural network on spatio-temporal data. In *Australasian Database Conference*, pages 142–154. Springer, 2015.

[22] Milan Krbálek and Petr Seba. The statistical properties of the city transport in cuernavaca (mexico) and random matrix ensembles. *Journal of Physics A: Mathematical and General*, 33(26):L229, 2000.

[23] Carolyn Kylstra. 10 Things Your Commute Does to Your Body, 2014.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[25] Santa Maiti, Arpan Pal, Arindam Pal, Tanushyam Chattopadhyay, and Arijit Mukherjee. Historical data based real time prediction of vehicle arrival time. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1837–1842. IEEE, 2014.

[26] US Department of Transportation. Beyond Traffic: 2045 Final Report, 2017.

[27] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[28] Sharat C Prasad and Piyush Prasad. Deep recurrent neural networks for time series prediction. *arXiv preprint arXiv:1407.5949*, 2014.

[29] Ranjana Dinakar Raut and Vineet Kumar Goyal. Public transport bus arrival time prediction with seasonal and special emphasis on weather compensation changes using rnn. *International Journal of Advanced Research in Computer and Communication Engineering*, 1(6):378–382, 2012.

[30] Chase Saw. What Percentage of Americans Use Public Transit to Get to Work?, 2017.

[31] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[32] Mathieu Sinn, Ji Won Yoon, Francesco Calabrese, and Eric Bouillet. Predicting arrival times of buses using real-time gps measurements. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 1227–1232. IEEE, 2012.

[33] DART First State. Millennials and car ownership? It's complicated, 2016.

[34] DART First State. The Environmental Benefits of Riding Public Transit, 2018.

[35] Wenxia Sun, Pei Chen, Ti Song, and Qiulan Wang. Bus arrival time prediction model study in apts. In *ICCTP 2010: Integrated Transportation Systems: Green, Intelligent, Reliable*, pages 2597–2605. 2010.

[36] Michael Tipping. Relevance vector machine, October 14 2003. US Patent 6,633,857.

[37] Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339 – 356, 1988.

[38] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI*, pages 3995–4001, 2015.

[39] Haiyang Yu, Zhihai Wu, Dongwei Chen, and Xiaolei Ma. Probabilistic prediction of bus headway using relevance vector machine regression. *IEEE Transactions on Intelligent Transportation Systems*, 18(7):1772–1781, 2017.

[40] Deniz Yuret. Knet: beginning deep learning with 100 lines of julia. In *Machine Learning Systems Workshop at NIPS*, volume 2016, page 5, 2016.

[41] M Zaki, I Ashour, M Zorkany, and B Hesham. Online bus arrival time prediction using hybrid neural network and kalman filter techniques. *International Journal of Modern Engineering Research*, 3(4):2035–2041, 2013.

[42] Pengfei Zhou, Yuanqing Zheng, and Mo Li. How long to wait?: predicting bus arrival time with mobile phone based participatory sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 379–392. ACM, 2012.