NEXT LOCATION PREDICTION MODEL: A GEOHASHED BASED RECURRENT NEURAL NETWORK

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Ali Rahim-Taleqani

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

December 2020

Fargo, North Dakota

North Dakota State University
Graduate School

Title

NEXT LOCATION PREDICTION MODEL: A GEOHASHED BASED RECURRENT NEURAL NETWORK

By

Ali Rahim-Taleqani

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall Nygard
Chair

Dr. Simone Ludwig

Dr. Jill Hough

Approved:

12/23/2020
Date

Dr. Simone Ludwig
Department Chair

ABSTRACT

This work investigates the significance of choosing appropriate recurrent neural networks (RNNs) architecture for a spatiotemporal next location prediction framework. Dockless shared micro-mobility sharing programs provide spatial trajectory data that entails essential information for city planners and developers. The study compares (i) the variable-sized geohash tessellation and (ii) two common RNN architectures: Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), using bike/scooter location data for Washington DC, USA. LSTM and GRU networks are used for modeling and incorporating information from spatial neighbors into the model. The study suggests that the LSTM model yields slightly better performance than the GRU model based on the same tessellation. However, geohash size might play a significant role in model performance. The study highlights the need to explore hyperparameter tuning, multiple spatial partitioning techniques especially with the Google S2 library, and more trip data for improving the prediction performance in neural network models.

ACKNOWLEDGMENTS

I would like to thank the following people, without whom I would not have been able to complete this research, and without whom I would not have made it through my master's degree: Dr. Kendall Nygard, my academic advisor, whose insight and knowledge into the subject matter steered me through this research, my great appreciation to my committee members, Dr. Jill Hough and Dr. Simone Ludwig. They have served as both professional and personal advisors to me. Their guidance has made my graduate studies both productive and enjoyable.

And my biggest thanks to my family for all the support you have shown me through this research. For my daughter, Diana, sorry for being even grumpier than usual while I wrote this thesis! And for my wife, Ameneh, thanks for all your support, This journey would be impossible without their unconditional support, compassion, and love.

DEDICATION

For Ameneh Forouzandeh-Shaharki

In the vastness of space and the immensity of time,

it is my joy to share

a planet and an epoch with you

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS iv

DEDICATION v

LIST OF TABLES vii

LIST OF FIGURES viii

1. INTRODUCTION 1

2. RELATED WORKS 4

3. METHODOLOGY 9

3.1. Sequence to Sequence Problem 11

3.2. Tessellation Strategies 12

3.3. Problem Setting 14

4. EXPERIMENTAL STUDY 17

4.1. Experimental Setup 17

4.1.1. Data Description 17

4.1.2. Data Wrangling 18

4.1.3. Hyperparameter Tuning 21

4.1.4. Final Model and Results 28

5. CONCLUSION 30

REFERENCES 31

LIST OF TABLES

Table		Page
1.	Geohash Precision Levels	13
2.	Data Statistics	20
3.	Sample Input	20
4.	Hyperparameter Settings	28
5.	Models Performance on Training Data with 100-fold Cross-validation	29
6.	Models Performance on Test Data	29

LIST OF FIGURES

Figure		Page
1.	The diagram of LSTM and GRU cells	11
2.	Example of geohash level 7	13
3.	Trip distribution by an operator in the DC area	18
4.	A snapshot of location data	19
5.	Batch size and number of epochs (LSTM for geohash 7)	21
6.	Batch size and number of epochs (LSTM for geohash 8)	22
7.	Batch size and number of epochs (GRU for geohash 7)	22
8.	Batch size and number of epochs (GRU for geohash 8)	23
9.	Optimizers for the LSTM and GRU models	24
10.	Number of neurons for the LSTM and GRU models	25
11.	Weight constraints and the dropout rate for LSTM (geohash-7)	26
12.	Weight constraints and the dropout rate for LSTM (geohash-8)	26
13.	Weight constraints and the dropout rate for GRU (geohash-7)	27
14.	Weight constraints and the dropout rate for GRU (geohash-8)	27
15.	Activation functions for the LSTM and GRU models	28

1. INTRODUCTION

Human mobility is complex and stochastic as it entails several factors such as gender, age, activity patterns, and route choice. Understanding and predicting individual mobility provides many opportunities for city planners, location-based service providers, and micromobility operators. They are able to develop sustainable plans, offer enhanced personalized recommendations and reminders, utilize their assets, and improve operational performance [1]. Next place prediction is one of the primary tasks of spatiotemporal data mining. It refers to the prediction of where an individual human will go next based on historical data. It has several applications such as traffic forecasting, location-aware advertising, intelligent resource allocation, early warning of potential public emergencies, as well as in recommender services, including the very popular Apple Map or Google Map [2] [3] [4] .

Shared micromobility becomes one of the best options for getting around cities and enhancing human mobility. It provides sustainable transportation options to communities. These systems have two forms; 1) dock-based model; 2) dockless model. The latter offers a higher degree of flexibility to pick up and leave the bikes/scooters in any accessible locations across the operational areas. Geolocation capabilities of dockless vehicles provide valuable mobility information about commuting patterns such as visited locations, trip length, and trip time [5] [6].

The trajectory data comes from different sources: 1) sensor data like Global Positioning System (GPS) [7] [8], wireless fidelity (Wi-Fi) sensors [9] [10], and base stations [11], 2) users' check-in data on social software [2] [12], 3) vehicle traffic data recorded by city traffic bayonets [13]. The mobility pattern data can be further fed into the next place prediction models. There are two broad categories of existing next location prediction techniques: 1) pattern-based and 2) model-based. The pattern-based uses historical data to extract user mobility patterns, then predict

the next location [3] [14] [15]. Pattern-based methods can only mine explicit patterns defined apriori and cannot capture all the data's regularity. Model-based methods such as Markov models [16], matrix factorization (MF) [17], periodic mobility models [2], [17], [18], and recurrent neural networks (RNN) learn statistical models to characterize user movement regularity and make predictions with these learned models [4].

I noticed RNNs, especially long short-term memory (LSTM) comparing to other RNN variants like gated recurrent unit (GRU), are widely used as building blocks in the spatiotemporal neural networks. Irrespective of the modeling technique used, the preliminary step in modeling a location-based entity is to spatially partition the space because - from a human mobility flow perspective - it is desirable to understand aggregate flow than an individual one [19], [20], [21]. Therefore, I used geohashing as a spatial indexing technique to study aggregated flows at different aggregation levels, improve models' predictive capability, and tackle low-precision GPS readings. I also believe it is imperative to examine the impact of different types of RNN on the prediction performance of models. To the best of my knowledge, this aspect has not been studied in the transportation literature.

This gap and the works on spatiotemporal trajectory data mining motivated me to rely on shared micro-mobility data to study next location predictive models and best model configurations. I take advantage of scooter/bike-sharing program data in Washington DC and compare the variable-size geohashes with two main RNN variants, LSTM and GRU. In terms of model development, four main concerns need to be addressed: 1) choice of model, 2) anonymity of individuals, 3) variable trip length, and 4) numerical representation of geolocation data.

I summarize my contributions in the following three components:

1. First, I develop encoder-decoder Recurrent Neural Networks (RNNs) to handle variable sequence length known as many-to-many representation.
2. I also apply geohash tessellations to study the aggregated movements where data instances are much more context-sensitive to geographical features.
3. Finally, I study two primary models, LSTM and GRU, with different aggregation levels, to find the best approach.

The organization of this thesis is as follows. In Section 2, I review some related works. I describe the methodology and model development in Section 3. Then, I discuss the results in Section 4 and outline the conclusions in Section 5.

2. RELATED WORKS

Various approaches and techniques have been applied to trajectory data. Trajectory data mining, specifically next location prediction, aims to discover prominent daily temporal habits and predict future individual activities.

Content-based methods learn location transition probability, given that the current location is related to the previous position. To this end, Markov-based predictors and compressions-based predictors are two common approaches to predict locations. The former builds a transition probability from one place to another based on an object's location history. At the same time, the latter depends on the number of occurrences of location prefixes. Song et al. developed several location predictors on a two-year trajectory trace of more than 63,000 users on Dartmouth College's campus-wide Wi-Fi wireless network. Their findings show that low-order Markov methods to be more than complicated and space-consuming compression-based predictors [1].

Distribution-based approaches model geographical and temporal characteristics as two random variables and compute and rank the probability of arrival at a location. Cho et al. proposed a location prediction method based, the Periodic and Social Mobility Model (PSMM), which describes human mobility based on periodic short-range travel and social network structures. The authors determined the home distance distributions of friends and all users, the distance between 200 large cities, and the probability of friendship as a function of distance [2].

Pattern mining techniques extract spatiotemporal patterns to predict locations, including sequential patterns, frequency patterns, integrated data patterns, and periodic patterns. Monreale et al. proposed a decision tree model called the T-pattern, a dynamic mining method for extracting GPS trajectory data. First, the authors used nearest neighbor methods to analyze the density distribution and obtain a range of interesting dense cells. Then, they constructed a tree of

temporally annotated sequences and computed its relationship to the spatiotemporal pattern within a given time tolerance. This method finds the best path on the tree (called T-pattern) that matches the given trajectory in the prediction process. Hence it computes a unique score that includes the path and punctuality scores to make a prediction [3].

Human movement is related to user preferences, activities, and spatiotemporal patterns. All can be captured in tensor and tensor factorization methods in collaborative filtering to predict locations. User location history can generate a matrix, and then matrix factorization can be used to capture user movement preferences. Bhargava et al. used tensor factorization methods to capture user movement preferences via user profiles, users' short messages in a social network, and user location and temporal information. Then they use multi-dimensional collaborative filtering to predict location [4].

In some of the literature, researchers use social information to utilize social relations to model peoples' movement and infer future visiting locations. Sadilek et al. proposed a Bayesian network for modeling the effect of a users' friend movement patterns. First, the model predicts the social relationships between user movement patterns. Then using social relationships improves the results of user location prediction. The output is also a sequence of locations that a user has visited over a given period [6].

The temporal component is a missing part in many location prediction works focused on geography or social characteristics. Some algorithms build a time-dependent model, but using a stochastic process is a better choice. Stochastic process models utilize time factors as random variables and embed location factors into the stochastic process. A typical example is the point process. Du et al. developed a Recurrent Marked Temporal Point Process (RMTPP), using a non-linear function to simultaneously model visiting time and location. The model also uses a recurrent

neural network to automatically learn a representation of influences from a user mobility history. In contrast with traditional models, time-dependent methods model time as an essential factor in location prediction. Although it improves the prediction performance, the correlation between spatial and temporal information is barely considered [22].

Traditional location prediction methods often cluster track points into regions and mine movement patterns within the regions. Such methods lose the information of points along the road and cannot meet specific services' demand. Moreover, traditional methods utilizing classic models may not perform well with long location sequences. While traditional models use a point sequence as trajectory representation, some researchers have proposed new trajectory representation methods such as the extraction of trajectory features and representation based on deep learning. Noulas et al. created a dataset composed of check-in location and time tuples and then extract features to feed into a ranking model. Prediction features can be classified as user mobility, global mobility, or temporal features. However, it ignores the spatiotemporal sequence and sparsity characteristic of check-in data [23].

Semantic-based predictors enable better reasoning and, therefore, better location prediction results. Ying et al. proposed a semantic framework for location prediction. First, they extracted frequent locations of the user's movement and the semantic information associated with these movements. Finally, they generated two tree structures to store semantic trajectory patterns. Semantic-based methods enable a better understanding of the semantic information related to visits to locations [24]. Karatzoglou et al. extended an LSTM network by applying Sequence to Sequence (Seq2Seq) learning on human semantic trajectories to improve the accuracy in a location prediction scenario. They compared their model against a semantic trajectory tree-based approach,

a probabilistic graph of first and higher-order. Sequence-to-Sequence learning shows promising results to model semantic trajectories and predict future human movement patterns [25].

Kong and Wu developed a Spatial-Temporal Long-Short Term Memory (ST-LSTM) model. They combined spatial-temporal influence into LSTM to avoid the problem of data sparsity. Further, they employed a hierarchical extension of the proposed ST-LSTM (HST-LSTM) in an encoder-decoder manner that models the contextual historic visit information to boost the prediction performance [26]. Wu et al. proposed a spatial-temporal-semantic neural network algorithm consisted of two steps. In the first step, the spatial-temporal-semantic feature extraction model (STS) converts the trajectory to location sequences with fixed and discrete points in the road networks. Then, an LSTM model is then constructed to make further predictions [27].

Besides the type of model, the spatial aggregation technique is also critical since studying mobility at the individual level is not common for many business stakeholders like city planners and marketing companies. Spatial aggregations are either performed using grids, where space is partitioned into square or rectangular grids of the fixed area [28], [29], [30], or using polygons, where space is partitioned into regular or irregular polygons of the variable area [31], [32], [33]. It is common practice in the transportation domain to apply either one of these tessellation styles for spatial partitioning.

Many applications use geohashing for the storage and efficient retrieval of geolocation data and satellite imagery. Geohashing is used to map and link multiple events together from different sources. Geohash is a geocoding system using a hierarchical spatial data structure to subdivide space into buckets of grid shape. The resultant is an alphanumeric string used as a unique identifier of a latitude/longitude pair anywhere in the world. Because of the encoding mechanism, geohash has arbitrary precision, which allows variable-size strings and flexible precision. Hence, any

nearby places will often present similar prefixes meaning the longer a shared prefix is, the closer the two places are. One such example is Microsoft's Bing Maps, which uses the Z-order curve, which uses base four numbers for indexing; hence it's also called the quad key. The maximum detail we can achieve in terms of granularity happens at a string length of 22, where each quadrant represents a GPS coordinate.

Hilbert curve based indexation is another alternative to Z-order curve like S2Geomtery developed by Google [34]. It's open-source, and many companies like Google, MongoDB, and Foursquare use this approach. It aims to solve spatial indexing problems and all sorts of operations one would find useful in the 3-dimensional world that we live in.

3. METHODOLOGY

The Deep Neural Network (DNN) is an extremely expressive learning model that can be used for highly complex vector-to-vector mappings. Recurrent Neural Network (RNN) is a DNN designed to recognize patterns in sequences of data, such as speech recognition, music generation, sentiment classification, DNA sequence analysis, video activity recognition, and many others. What differentiates RNN from other neural networks is that it takes the temporal dimension of data into account. RNN is one of the high performing neural networks that can process sequential data very well. The underlying idea behind RNN is to store relevant parts of the input (memorize important information from the past) and use this information while predicting the output in the future. It has been widely applied in many fields, such as unsegmented handwriting generation [35] and natural language processing [36]. It can process arbitrary-length sequences of inputs, especially when there are some hidden relations among different sequence elements. The sequential format of human mobility data leads me to RNN. Traditional neural networks cannot learn temporal dependencies, which was overcome by RNN. It can process arbitrary-length sequences of inputs, especially when the sequence elements are not independent.

Standard RNN suffers from the vanishing and exploding gradients. Extremely small gradients do not contribute much to learning. Hence, for very long sequences, they cannot carry information from earlier steps to later ones. The former problems were successfully addressed by Hochreiter & Schmidhuber. They developed a long short-term memory (LSTM) architecture, which is resistant to the vanishing gradient problem [37]. The latter problem turned out to be relatively easy to address by simply truncating the gradient [38], [39]. Another alternative to LSTM is the Gated Recurrent Unit (GRU) introduced by Cho et al. [40]. LSTM and GRU cells' schematic is illustrated in Figure 1. Three different gates regulate information flow in the LSTM

cell: A forget gate (Γ_f), an input gate (Γ_i), and an output gate (Γ_o). Gate mechanisms in LSTMs introduce added computational expense (higher degree of complexity) and therefore added parameterization. The forget gate evaluates what is relevant to keep from previous steps. The input gate decides what information is pertinent to add from the current step. The output gate determines what the next hidden state should be.

On the other hand, GRU does not have a cell state and uses a hidden state to transfer information. It also only has two gates, a reset gate (Γ_r), and an update gate (Γ_u). GRU has fewer tensor operations; therefore, they are a little speedier to train then LSTM's. In this work, I use the LSTM architecture similar to the one developed by Alex Graves but without peep-hole connections [35] and GRU introduced by Cho et al. [40]:

LSTM

$$\tilde{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{3.1}$$

$$\Gamma_f = \sigma\big(W_f.[h_{t-1}, x_t] + b_f\big) \tag{3.2}$$

$$\Gamma_i = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{3.3}$$

$$\Gamma_o = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{3.4}$$

$$C_t = \Gamma_f * C_{t-1} + \Gamma_i * \tilde{C}_t \tag{3.5}$$

GRU

$$\tilde{C}_t = tanh\ (W_C.[\Gamma_r * h_{t-1}, x_t] + b_C) \tag{3.6}$$

$$\Gamma_u = \sigma(W_u.[h_{t-1}, x_t] + b_u) \tag{3.7}$$

$$\Gamma_r = \sigma(W_r.[h_{t-1}, x_t] + b_r) \tag{3.8}$$

$$C_t = (1 - \Gamma_u) * C_{t-1} + \Gamma_u * \tilde{C}_t \tag{3.9}$$

In these equations, the W variables and the b variables are weight matrices and biases. The operation $*$ denotes the element-wise vector product. The LSTM's hidden state is the concatenation (h_t, c_t). Thus, the LSTM has two kinds of hidden states: a "slow" state c_t that mitigates the vanishing gradient problem and a "fast" state h_t that facilitates the LSTM to make complex decisions over short periods of time.

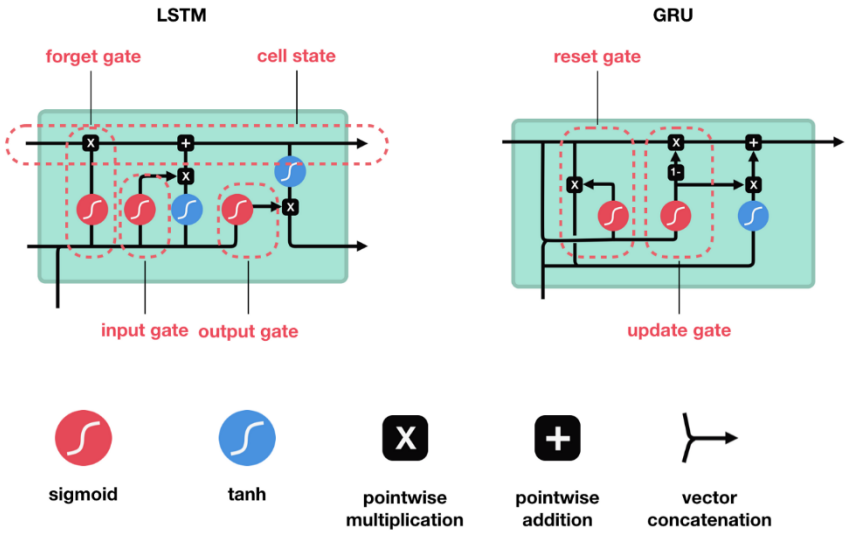

Figure 1. The diagram of LSTM and GRU cells [41]

3.1. Sequence to Sequence Problem

Generally, sequence prediction involves one-to-one or many-to-one problems representing one input time step to one output time step or multiple input time steps to one output time step, respectively. However, the length of location data varies case by case, as trip data might contain few to many locations. Given that, there are multiple input time steps and multiple-output time steps representing many-to-many type sequences. It is a more challenging type of sequence prediction problem known as sequence-to-sequence or seq-2seq for short. The sequence-to-sequence model introduced by Google [36] aims to map a fixed-length input with a fixed-length output where the input and output length may differ. The RNN encoder-decoder consists of two

RNNs acting as encoder and decoder pairs. The encoder, which includes a stack of several recurrent units (LSTM or GRU), maps a variable-length input sequence to a fixed-length vector. The decoder maps the vector representation back to a variable-length target sequence.

3.2. Tessellation Strategies

The spatial pattern of mobility is critical to both city planners and operators. For this purpose, I use the concept of geohash tessellation, which can divide a geographical area into smaller sub-areas with arbitrary precision. Geohash is a public domain geocode system with a hierarchical spatial data structure that subdivides space into buckets of grid shape invented in 2008 by Gustavo Niemeyer. The Niemeyer technique is similar to Morton encoding, a specialized instantiation of a Z-order Space-filling curve [42]. Similarly, Natural Area Codes (NAC) follow a similar encoding schema but employ a 30-bit encoding [43]. Niemeyer's technique has many useful features: rapid computation, a single-value string representation, variable precision through string truncation, proximal region detection, pattern support, and straightforward human/machine interpretation. The advantage of using geohash over transportation analysis zones, parcel data, or zip code-based areas is that I can use different resolutions (granularity level) to discover any movement pattern as small as 1 foot by 1 foot.

Geohash encodes a latitude-longitude coordinate into an alphanumeric string. A total of 32 characters, namely, 0−9 and b−z (excluding a, i, l, o), are used for the base 32 encodings. Table 1 shows the size of the geohashes at different precision levels. In geohash, each code represents a divided area. For example, a geographic point of 38.89778137, -77.02720642, (latitude, longitude) pairs encoded with precision seven will be converted to 'dqcjr1n' shown in Figure 2.

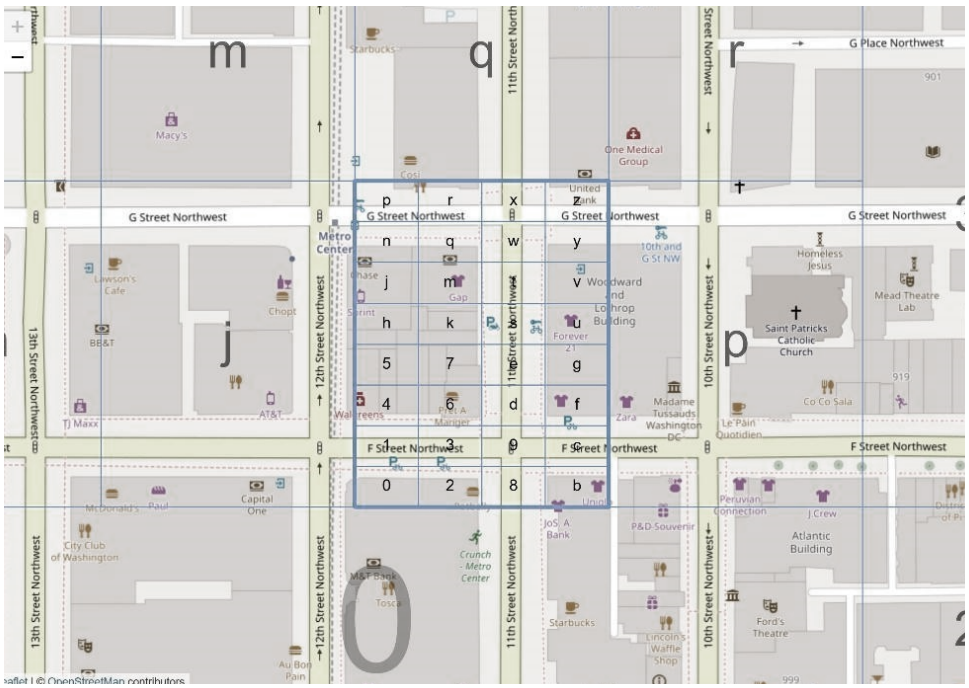

Figure 2. Example of geohash level 7

Table 1. Geohash Precision Levels

Precision		Cell Width	Cell Height
1	<=	5,000 km	5,000 km
2	<=	1,250 km	625 km
3	<=	156 km	156 km
4	<=	39.1 km	19.5 km
5	<=	4.89 km	4.89 km
6	<=	1.22 km	0.61 km
7	<=	153 m	153 m
8	<=	38.2 m	19.1 m
9	<=	4.77 m	4.77 m
10	<=	1.19 m	0.596 m
11	<=	149 mm	149 mm
12	<=	37.2 mm	18.6 mm

3.3. Problem Setting

Spatial and temporal mobility data of individual entities are generally captured by the global positioning system (GPS) signals [22]. Most micromobility operators have geo-tracking systems and capture the real-time location of bikes or scooters. Previous studies have shown that such location (trip) data can provide us with rich insights about trip distributions and how they vary from area to area during a day, week, or year [5], [44].

Trajectory data represent the spatiotemporal properties of moving objects. A location usually defines as $P = (x,\ y,\ h,\ t)$ where $x,\ y,\ h,\ t$ represents as latitude, longitude, altitude, timestamp of a given moving object, respectively. In many real-world applications, h is ignored; thus, $P = (x,\ y,\ t)$ is commonly used. Trajectory data are either recorded actively or passively. The former refers to the time when people actively (deliberately) record their locations by logging into social network platforms like Twitter, Facebook, among others. The latter group relates to instances where geolocation data are automatically recorded by a satellite-based radio navigation system like GPS. The latitude, longitude, and timestamp are collected at various frequency rates, depending upon the device's capability.

Several potential challenges are dealing with spatiotemporal trajectory data, including but not limited to the randomness of movement, sparsity problems, time sensitivity, and heterogeneous data. The randomness comes from the inherent mobility randomness of an object. Location data is context-sensitive, and features defining mobility patterns are almost fuzzy; It is challenging to predict the next locations without any historical information of an object. This problem is referred to as the cold start problem. If only a few samples are available is referred to as the sparsity problem. Both issues are common in trajectory data mining. Trajectory data is also time sensitive. In other words, as the location of an object changes by time, the relative time resolution to record

trajectory data might result in loss of information. Finally, trajectory data are from diverse resources such as vehicles, animals, and peoples. Each group has different sampling rates and movement patterns, which add to the problem's complexity [7].

For location-based prediction and spatial aggregation, I used geohashing as described in the previous section. While other geocoding systems are available like S2 geometry, I chose geohashing: it has a simple structure for aggregating spatial features, and I can compare the results with related studies. Hence, I divided the area of interest into a set of regions $R = r_1, \dots, r_n$. Once the areas are defined, the trip sequence is generated between each pair of regions and modeled for analysis. Choosing the appropriate geohash precision level is highly dependent on the use case.

Low precision geohashes represent cells that cover a large area, and high precision covers a small area. More location data will be located in one geohash, and trip data will be missed using the former case. On the other hand, smaller size geohash tends to act as individual points and will lose aggregation benefits. Because of the nature of micromobility trips and the average traveled distance by bike or scooter, I use only geohash 7 and 8. Needless to say that for other use case cases with different vehicle types, smaller or larger geohash sizes could also worth analyzing.

Given the geohash level, my main experiment is an extensive search to find the best RNN architecture between LSTM and GRU. I employed root mean square error (RMSE) for evaluating LSTM and GRU models. I learned that RMSE gives high weight to large errors: RMSE should be more useful when a large error is particularly undesirable – like in my study -. I used root mean square logarithm error (RMSLE) for the loss function because of the robustness to the outliers, the relative error between the Predicted and Actual Values. The RMLSE has a unique property, penalizing the underestimation of the actual value more severely than it does for the overestimation.

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2} \tag{3.10}$$

$$RMSLE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}\left(log_e(1 + \hat{y}_i) - log_e(1 + y_i)\right)^2} \tag{3.11}$$

4. EXPERIMENTAL STUDY

I provide a detailed experimental study of next location prediction modeling on the Helbiz® bike/scooter dataset in this section. I also evaluate the LSTM and the GRU models on different geohash sizes and do the hyperparameter tuning to find the best setting for this problem.

4.1. Experimental Setup

4.1.1. Data Description

Shared micromobility operators – Lime, Bird, Spin, among others – are required to share the anonymized trip and real-time location data depending upon local regulations. Washington, DC, is one of those cities in the US that asks operators to publish their data via an application programming interface (API). Given this opportunity, these data contain valuable information about most visited locations, the distribution of bikes and scooters, location of vehicles against other operators, to name a few. However, operators limit the data by showing only when bikes/scooters are available for check-out in real-time, meaning no information about the vehicles' locations between two consecutive check-outs is available. At the time of writing this thesis, only Helbiz® provides real-time location data either in-use, reserved, available for check-out, or broken. Such data offer better insights into most visited locations, route choice, curbside management, among others.

I live-streamed Helbiz® public application programming interface (API) every 30 seconds for a higher degree of time resolution and an in-depth understanding of mobility for 15 days in August 2020. Figure 3 shows a snapshot of trips in Washington, DC, for an operator over a single day. The dataset before preprocessing has almost 24 million data points.

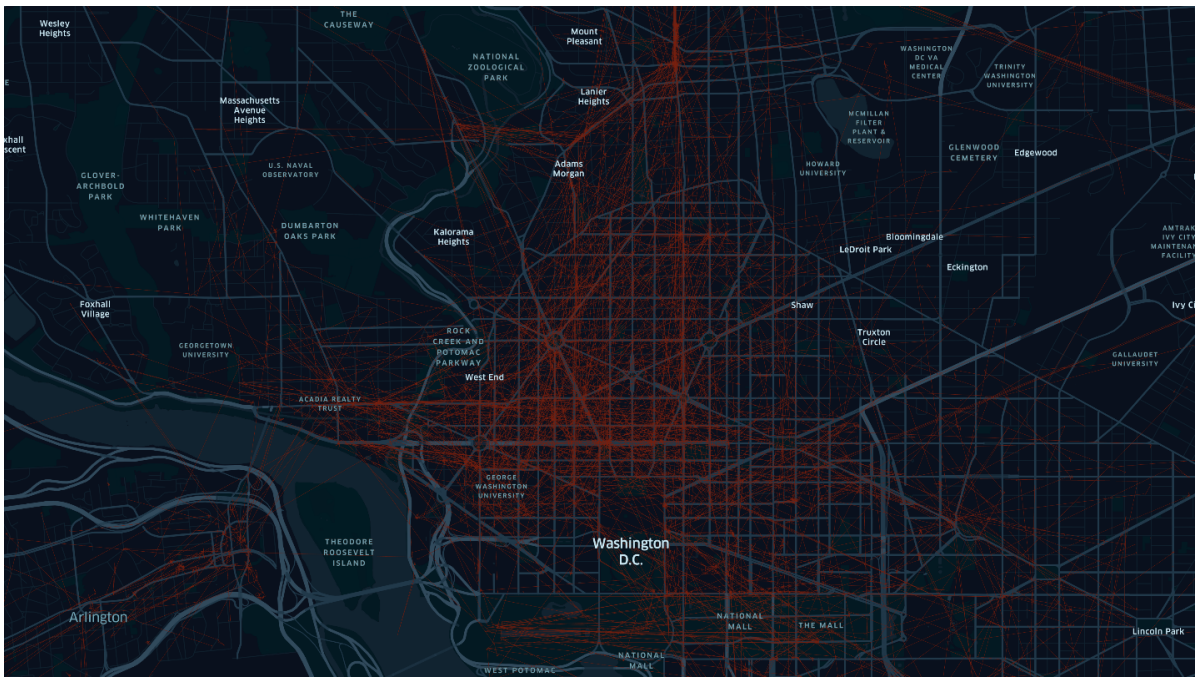

Figure 3. Trip distribution by an operator in the DC area

4.1.2. Data Wrangling

Figure 4 shows a snapshot of raw data from the Helbiz® public API, including the bike/scooter identification number, geographic coordinates (latitude and longitude), timestamp, and other related data. Since users are anonymous, I focused only on an individual vehicle's movement and used trips made by a vehicle.

Figure 4. A snapshot of location data

Further steps were necessary to convert the bike/scooter availability data into a sequence of consecutive trips. Since the vehicles are sometimes idle many reading points were not necessary for this case and were removed: only distinct GPS geo data points were retrieved for further analysis. However, if someone is interested in analyzing the number of trips, idle time, and turnover rate, it might be a promising avenue to follow. There were also some outliers, like moving vehicles from nearby cities to balance inventory and false GPS reading points not matching the area of interest: these are easily filtered out by checking the corresponding geohash. The number

of visited locations (geohashes) in a single trip – length of single trip sequence – is shown in Table 2.

Table 2. Data Statistics

The number of data points before preprocessing (GPS readings)	The number of sequences after initial preprocessing		The maximum length of a sequence		The minimum length of a sequence	
	Geohash 7	Geohash 8	Geohash 7	Geohash 8	Geohash 7	Geohash 8
23,797,853	2,321	4,744	146	231	2	2

Moreover, I extracted the sine and cosine of days of a week and hours of a day to capture the temporal periodicity. The central part of feature engineering was how to present the geohash alphanumeric string and the corresponding loss function for training and metrics for evaluation. Since handling numeric values is much easier for deep learning models, I used geohash centroid to represent features and target values' locations. The final number of features is six for every trip, as shown in Table 3. Finally, I made a 1-step forecast using 2-step prior trip information for a given trip because of the variable trip length. A sample of given input and output are shown in Table 3.

Table 3. Sample Input

Input	Output
[0.5, 0.8660254, 0.43388374, -0.90096887, 39.0, -77.00523376], [0.5, 0.8660254, 0.43388374, -0.90096887, 39.0, -77.00798035]	[39., -77.00935364]

4.1.3. Hyperparameter Tuning

Hyperparameters such as batch size, number of epochs, and learning rate refer to an external configuration that could not be estimated from data. Because of limited computational power, I iteratively tune one parameter while holding other parameters constant. The goal is to find the best hyperparameters for a pair of geohash level and RNN model.

4.1.3.1. Batch size and number of epochs

There are some trends between RMSE of LSTM and GRU with different geohash levels. While RMSE for LSTM with geohash 8 is on average lower than LSTM with geohash 7, it is the opposite for GRU models. As shown in Figure 5, Figure 6, Figure 7, and Figure 8, LSTM is slightly sensitive to batch size and epochs more than GRU. It might be because of a higher degree of complexity in LSTM architecture. In general, lower values of batch size yields consistent results over the different number of epochs.

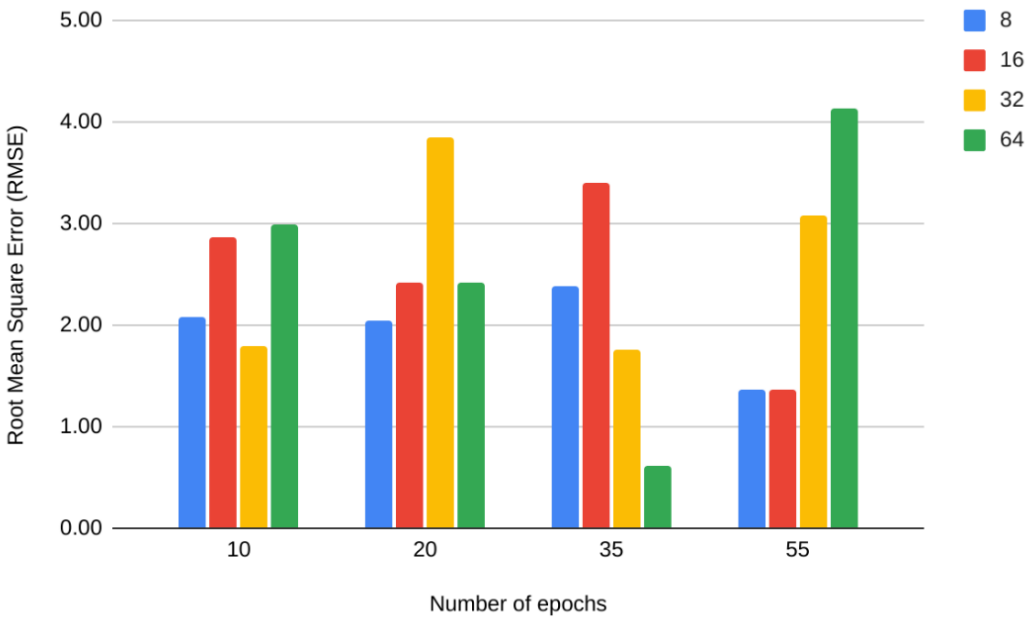

Figure 5. Batch size and number of epochs (LSTM for geohash 7)

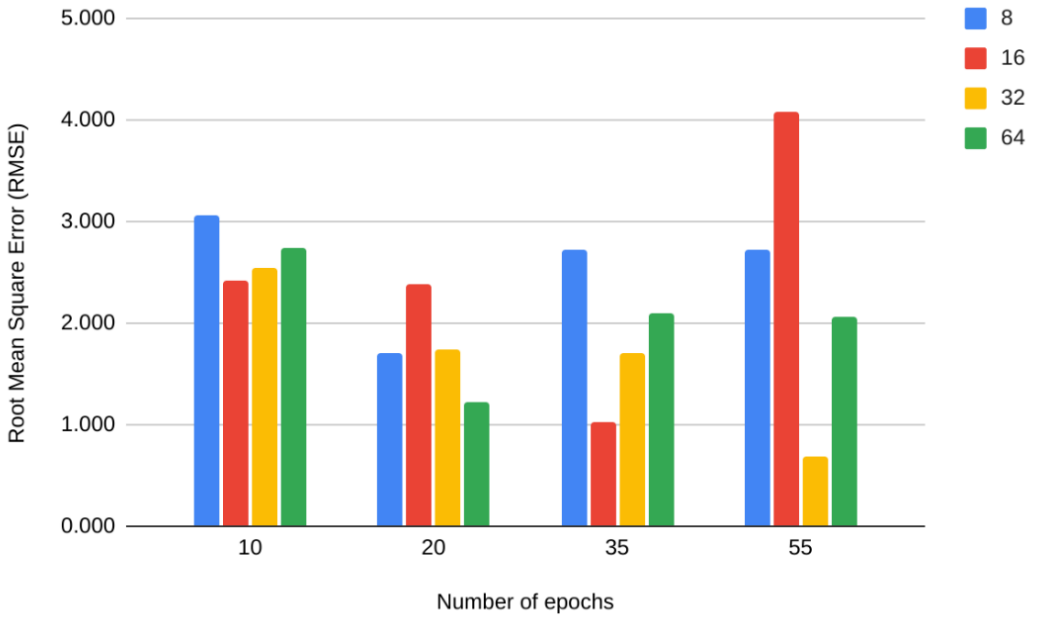

Figure 6. Batch size and number of epochs (LSTM for geohash 8)

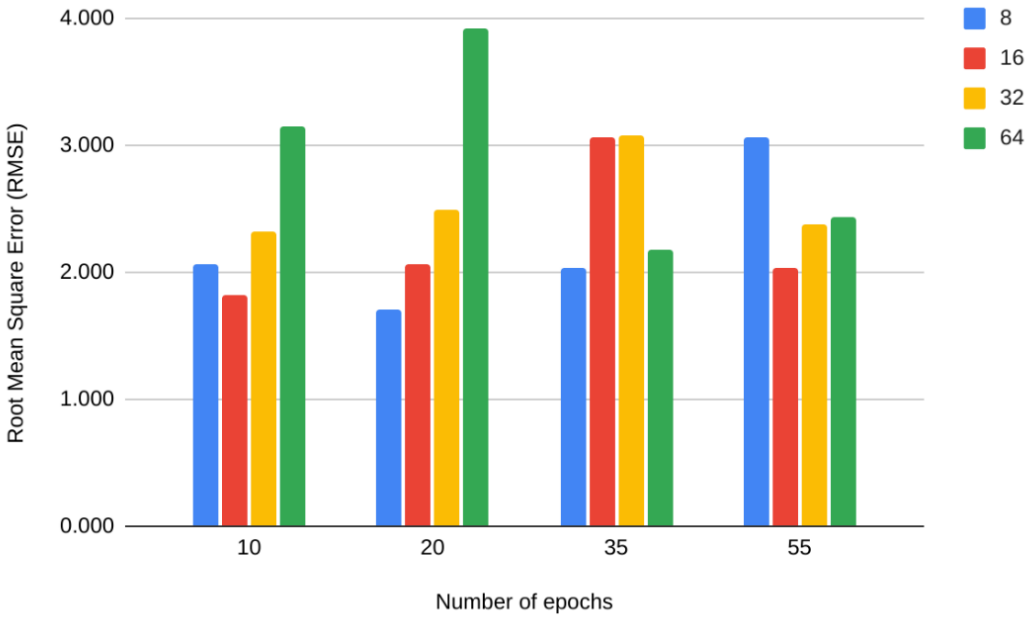

Figure 7. Batch size and number of epochs (GRU for geohash 7)

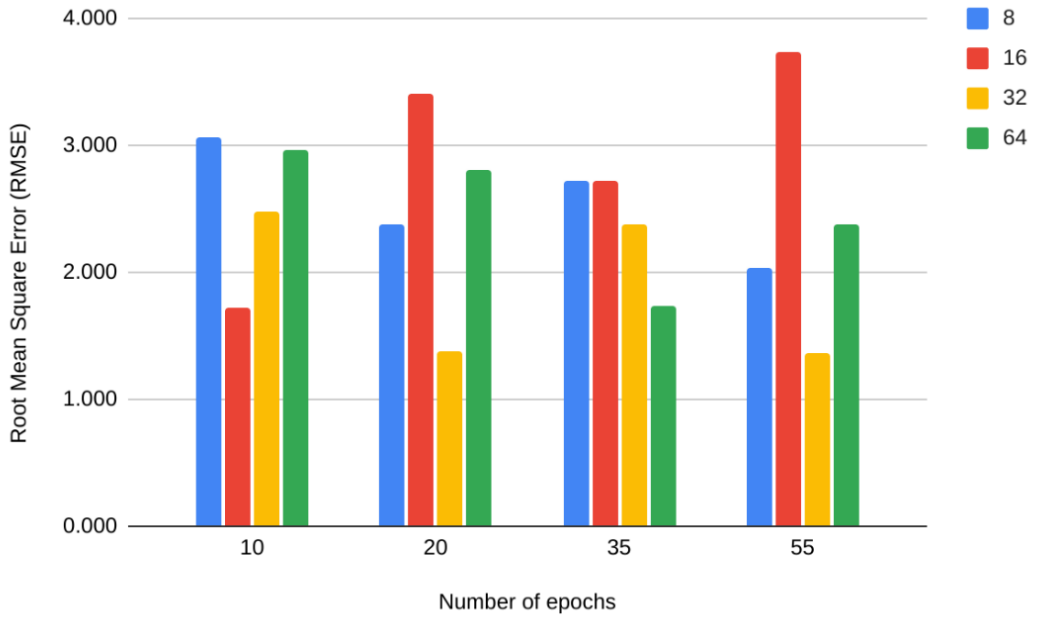

Figure 8. Batch size and number of epochs (GRU for geohash 8)

4.1.3.2. Optimizer

The optimization algorithm's appropriate choice is critical in achieving a good result in a reasonable time. Both models suggest AdaMax and RMSprop as effective optimizers for this problem, as shown in Figure 9. The AdaMax delivers slightly better performance over the RMSprop. Also, geohash level 7 has a lower value of RMSE for both models. The AdaMax is much less sensitive to the hyper-parameters' choice (e.g., the learning rate).

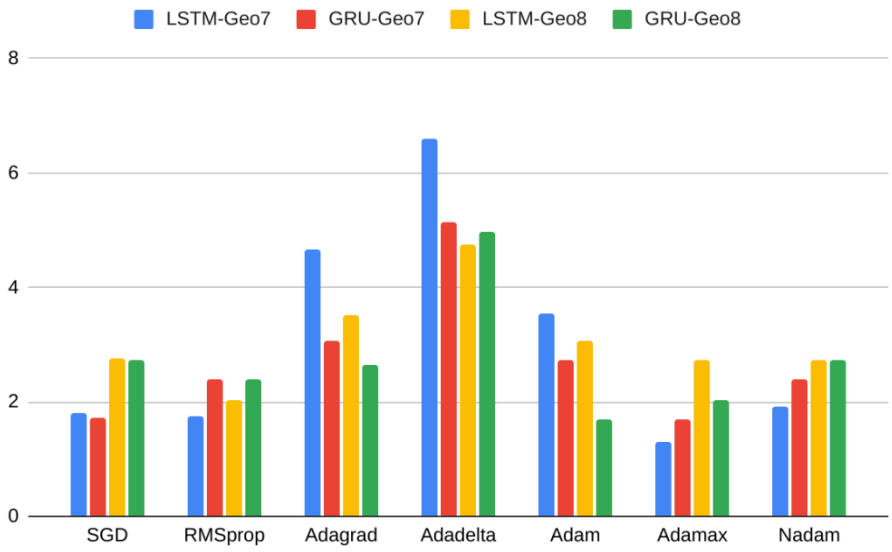

Figure 9. Optimizers for the LSTM and GRU models

4.1.3.3. Number of neurons

The number of neurons in the layer shows the representational capacity of the network. The larger number of neurons, the longer the model needs to be trained. As shown in Figure 10, in general, geohash level 8 yields better results than level 7. Comparing the LSTM with the GRU model, the GRU model shows stable and better performance in the range of 20 and 45 neurons. However, it is relatively smaller for the GRU model with a range of 25 and 40.

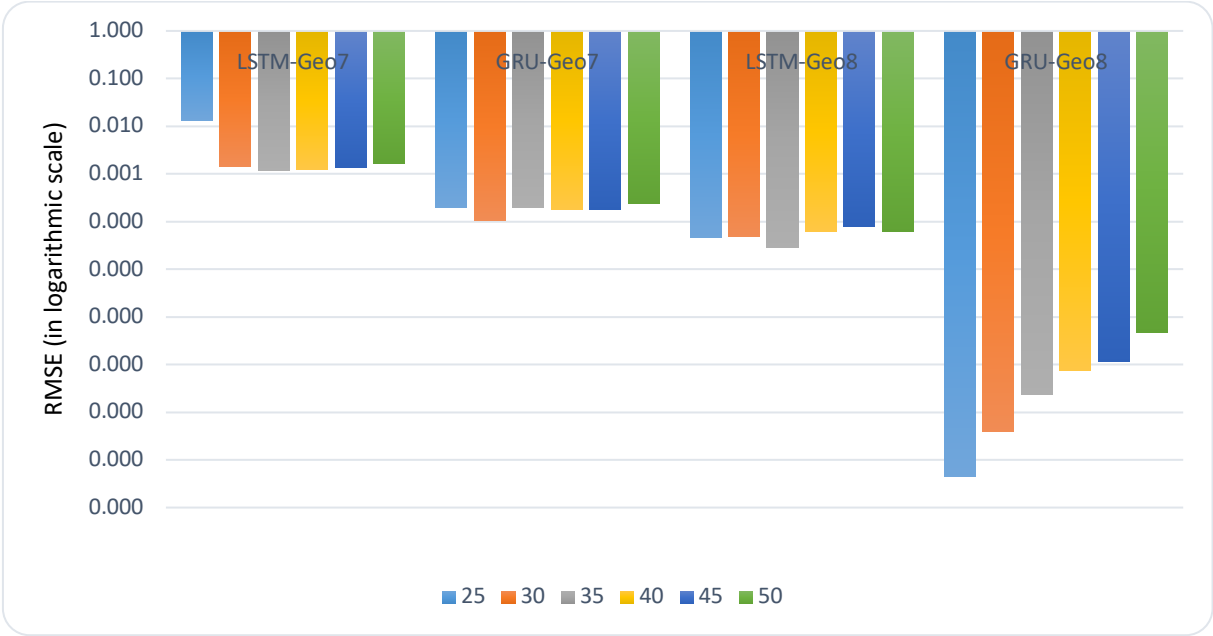

Figure 10. Number of neurons for the LSTM and GRU models

4.1.3.4. Regularization

Regularization is a technique to reduce the complexity of the model. It helps to solve the overfitting problem. There are many methods to do the regularization, and here I applied two techniques; adding weight constraints and drop out to neural network layers. Weight constraints check the size of weights at each layer. If the size exceeds a predefined threshold, they are rescaled to make sure the weight is below the limit or between a range. Weight constraints offer an approach to reduce the overfitting of a deep learning neural network model on the training data and improve the model's performance on new data. Dropout is also a regularization method that approximates training many neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or "dropped out." It makes the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. Figure 11, Figure 12, Figure 13, and Figure 15 illustrate RMSE in logarithmic scale by various dropout rates (0.1, 0.3, and 0.5) and weights constraints values (1, 2, 3, 4, and 5). In general, RMSE is

much better with geohash level 8 than level 7 for both models. Also, changing weight constraints does not produce any significant positive change for almost all models. However, increasing the dropout rate yields a better RMSE value for nearly all models.

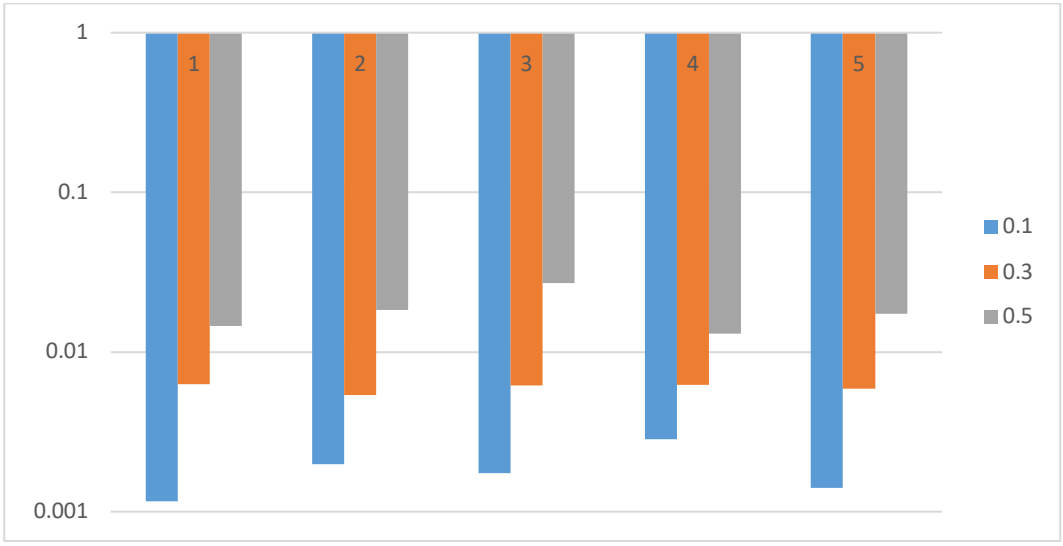

Figure 11. Weight constraints and the dropout rate for LSTM (geohash-7)

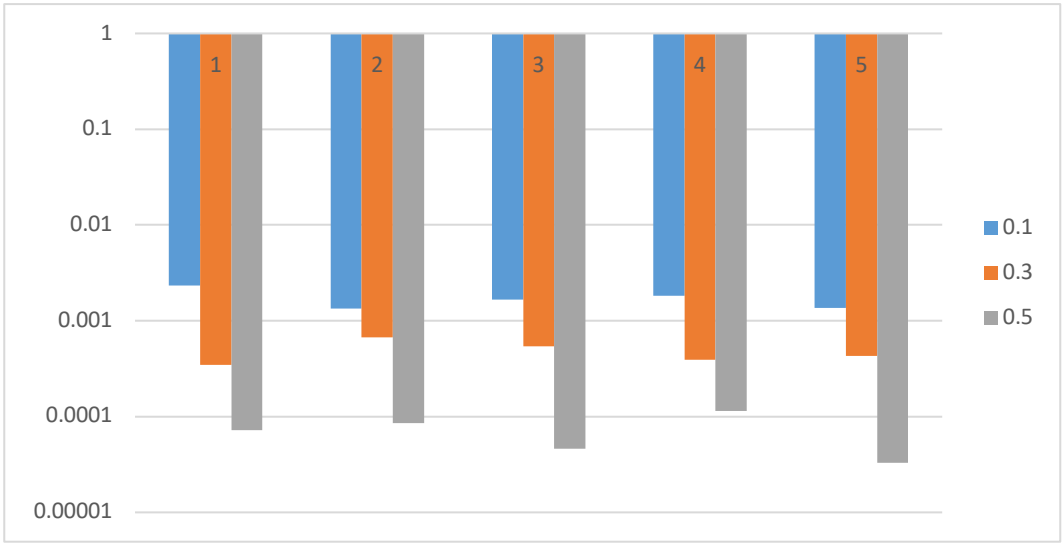

Figure 12. Weight constraints and the dropout rate for LSTM (geohash-8)

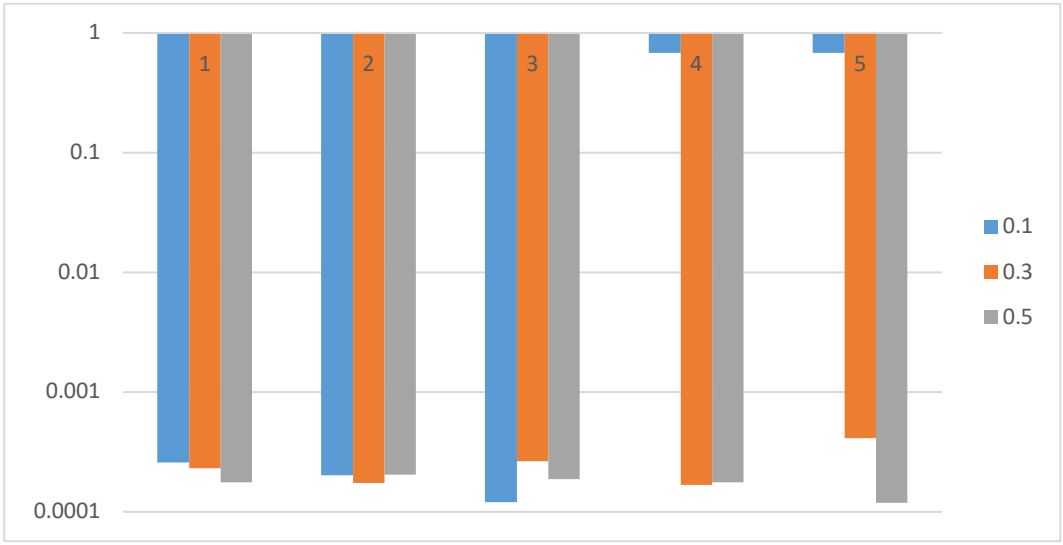

Figure 13. Weight constraints and the dropout rate for GRU (geohash-7)

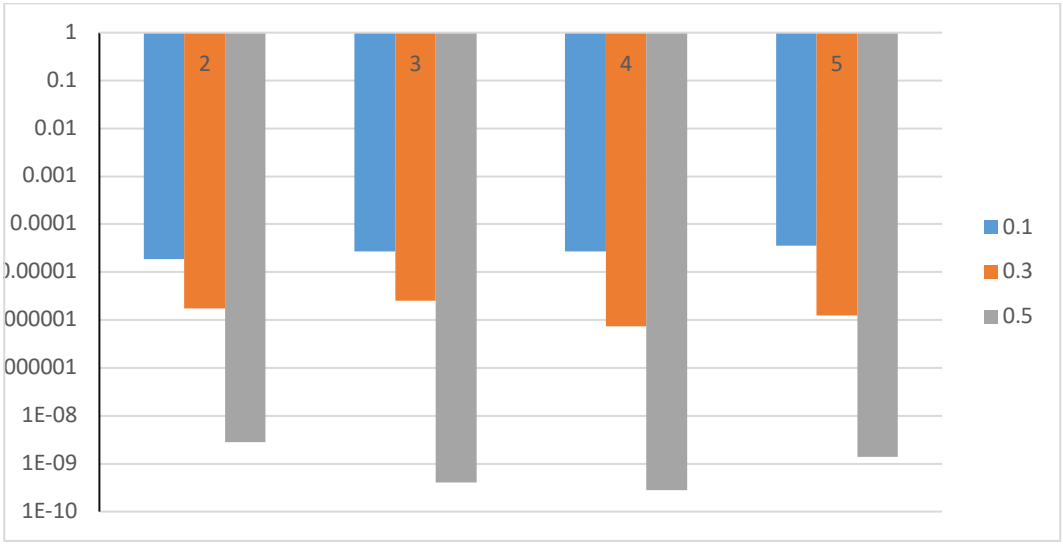

Figure 14. Weight constraints and the dropout rate for GRU (geohash-8)

4.1.3.5. Activation

Basically, activation functions introduce non-linearity into neural networks meaning that the neural networks can successfully approximate functions (up-to a specific error). LSTM models deliver lower RMSE with Softplus, very similar to Relu, enticingly smooth, and differentiable (Figure 15). In comparison, GRU provides better performance with S-shaped functions Sigmoid and Softsign.

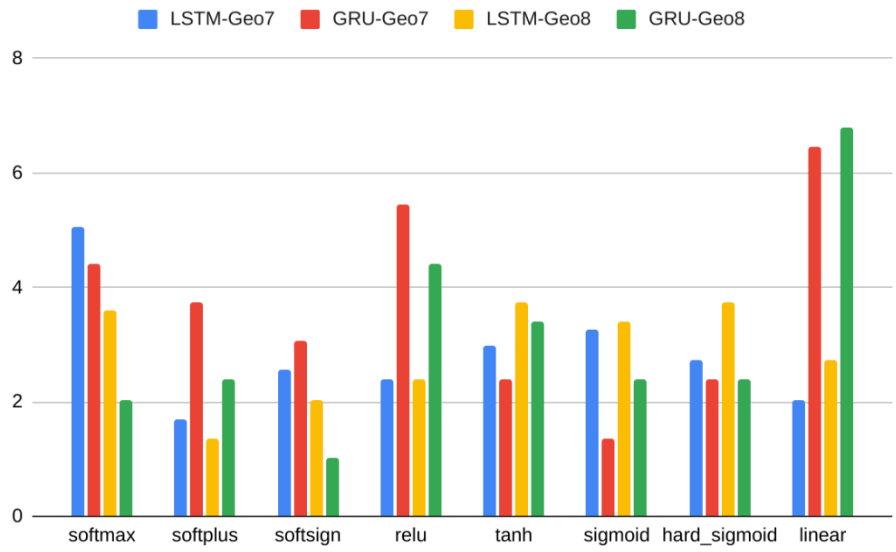

Figure 15. Activation functions for the LSTM and GRU models

4.1.4. Final Model and Results

The final tuned hyperparameter settings are described in Table 4.

Table 4. Hyperparameter Settings

	LSTM-GEO7	GRU-GEO7	LSTM – GEO8	GRU-GEO8
Optimizer	AdaMax	AdaMax	RMSprop	Adam
Batch Size	64	8	32	32
Epochs	35	20	55	55
Number of Neurons	35	30	35	10
Learning Rate	0.001	0.3	0.001	0.2
Activation function	Softplus	Sigmoid	Softplus	Softsign
Kernel Initializer	Lecun uniform	Normal	Glorot uniform	Glorot uniform
Weight Constraint	5	5	5	4
Dropout	0.1	0.5	0.5	0.5

Once the parameters are set, we split the dataset 99%, 1% between training and testing stages, and then feed to the LSTM and GRU. Because of the small sample size, I used 100-fold cross-validation, and the average results are shown in Table 5.

Table 5. Models Performance on Training Data with 100-fold Cross-validation

	LSTM-Geo7	GRU-Geo7	LSTM-Geo8	GRU-Geo8
RMSE	38.57	48.96	47.46	47.32
Loss	0.0011783709	0.0000711143	0.0000531201	0.0000000001

Overall, the experimental results show that the LSTM models yield better results than the GRU models. Moreover, the two models show contrasting results as the geohash level increases. Both models are then evaluated against test data, and corresponding RMSE are described in Table 6**.**

Table 6. Models Performance on Test Data

	LSTM-Geo7	GRU-Geo7	LSTM-Geo8	GRU-Geo8
RMSE	41.979336	44.6361	45.71772	45.337414

5. CONCLUSION

Next-visiting location problem has been studied from different perspectives. With the introduction of new micromobility options, a new data source is available to develop and examine new methods. I examined two known RNN architectures in conjunction with the geohashing technique to find the best model for these problems.

My goal is to compare the two common RNNs combined with the spatial aggregation approach to improve the prediction performance. The proposed geohash based sequence learning model predicts a bike/scooter user's next location in different areas of a city. For training purposes, I used a dataset that consists of 15 days of bike/scooter location data recorded every 30 seconds in Washington, DC. Experimental results show that the LSTM predictor slightly outperforms GRU models. However, GRU uses fewer training parameters, hence use less memory and perform faster than LSTM.

There are several avenues for future research. First and foremost, more data improve the performance of the models. It requires at least a full year of location data from various operators to capture seasonality and distribution variation by operators. Other source data like weather data, points of interest, elevation, and distance to other transportation facilities are useful depending upon the problem's context.

Other tessellations strategies, especially S2 by Google and H3 by Uber, are the two top choices for exploring. The former uses a variable size hierarchal spatial indexing technique based on the Hilbert space-filling curve, while the latter uses a hexagonal indexing method. The main problem that needs to be addressed is how two neighboring geo-points are being treated when assigned to two different spatial buckets. The researcher could also extend the current work by analyzing a new deep learning model, temporal convolutional networks, for sequential data.

REFERENCES

[1] L. Song, D. Kotz, R. Jain and X. He, "Evaluating location predictors with extensive Wi-Fi mobility data," in *Ieee Infocom 2004*, 2004.

[2] C. Eunjoon, S. A. Myers and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

[3] A. Monreale, F. Pinelli, R. Trasarti and F. Giannotti, "Wherenext: a location predictor on trajectory pattern mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[4] P. Bhargava, T. Phan, J. Zhou and J. Lee, "Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data," in *Proceedings of the 24th international conference on world wide web*, 2015.

[5] A. Rahim Taleqani, C. Vogiatzis and J. Hough, "Maximum Closeness Centrality-Clubs: A Study of Dock-Less Bike Sharing," *Journal of Advanced Transportation,* vol. 2020, 2020.

[6] A. Sadilek, H. Kautz and J. P. Bigham, "Finding your friends and following them to where you are," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012.

[7] J. Yuan, Y. Zheng, L. Zhang, X. Xie and G. Sun, "Where to find my next passenger," in *Proceedings of the 13th international conference on Ubiquitous computing*, 2011.

[8] L. X. Pang, S. Chawla, W. Liu and Y. Zheng, "On mining anomalous patterns in road traffic streams," in *International conference on advanced data mining and applications*, 2011.

[9] D. Kotz and T. Henderson, "Crawdad: A community resource for archiving wireless data at dartmouth," *IEEE Pervasive Computing,* vol. 4, no. 4, pp. 12-14, 2005.

[10] S. Scellato, M. Musolesi, C. Mascolo, V. Latora and A. T. Campbell, "Nextplace: a spatio-temporal prediction framework for pervasive systems," in *International Conference on Pervasive Computing*, 2011.

[11] N. Eagle and A. S. Pentland, "Reality mining: sensing complex social systems," *Personal and ubiquitous computing,* vol. 10, no. 4, pp. 255-268, 2006.

[12] L.-Y. Wei, Y. Zheng and W.-C. Peng, "Constructing popular routes from uncertain trajectories," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012.

[13] M. Chen, Y. Liu and X. Yu, "Nlpmm: A next location predictor with markov modeling," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2014.

[14] Z. Li, B. Ding, J. Han, R. Kays and P. Nye, "Mining periodic behaviors for moving objects," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010.

[15] C. Zhang, J. Han, L. Shou, J. Lu and T. La Porta, "Splitter: Mining fine-grained sequential patterns in semantic trajectories," *Proceedings of the VLDB Endowment,* vol. 7, no. 9, pp. 769-780, 2014.

[16] W. Mathew, R. Raposo and B. Martins, "Predicting future locations with hidden Markov models," in *Proceedings of the 2012 ACM conference on ubiquitous computing*, 2012.

[17] N. Duong-Trung, N. Schilling and L. Schmidt-Thieme, "Near real-time geolocation prediction in twitter streams via matrix factorization based regression," in *Proceedings of the 25th ACM international on conference on information and knowledge management*, 2016.

[18] Q. Liu, S. Wu, L. Wang and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[19] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong and W.-c. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems,* vol. 28, pp. 802-810, 2015.

[20] X. Zhou, Y. Shen, Y. Zhu and L. Huang, "Predicting multi-step citywide passenger demands using attention-based neural networks," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

[21] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[22] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[23] A. Noulas, S. Scellato, N. Lathia and C. Mascolo, "Mining user mobility features for next place prediction in location-based services," in *2012 IEEE 12th international conference on data mining*, 2012.

[24] J. J.-C. Ying, W.-C. Lee, T.-C. Weng and V. S. Tseng, "Semantic trajectory mining for location prediction," in *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2011.

[25] A. Karatzoglou, A. Jablonski and M. Beigl, "A Seq2Seq learning approach for modeling semantic trajectories and predicting the next location}," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2018.

[26] D. Kong and F. Wu, "HST-LSTM: A Hierarchical Spatial-Temporal Long-Short Term Memory Network for Location Prediction.," in *IJCAI*, 2018.

[27] F. Wu, K. Fu, Y. Wang, Z. Xiao and X. Fu, "A spatial-temporal-semantic neural network algorithm for location prediction on moving objects," *Algorithms,* vol. 10, no. 2, p. 37, 2017.

[28] X. Li, G. Pan, Z. Wu, G. Qi, S. Li, D. Zhang, W. Zhang and Z. Wang, "Prediction of urban human mobility using large-scale taxi traces and its applications," *Frontiers of Computer Science,* vol. 6, no. 1, pp. 111-121, 2012.

[29] J. Xu, R. Rahmatizadeh, L. Bölöni and D. Turgut, "Real-time prediction of taxi demand using recurrent neural networks," *IEEE Transactions on Intelligent Transportation Systems,* vol. 19, no. 8, pp. 2572-2581, 2017.

[30] S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman and C. Ratti, "Taxi-aware map: Identifying and predicting vacant taxis in the city," in *International Joint Conference on Ambient Intelligence*, 2010.

[31] Y. Ding, Y. Li, K. Deng, H. Tan, M. Yuan and L. M. Ni, "Dissecting regional weather-traffic sensitivity throughout a city," in *2015 IEEE International Conference on Data Mining*, 2015.

[32] J. Shen, X. Liu and M. Chen, "Discovering spatial and temporal patterns from taxi-based Floating Car Data: A case study from Nanjing," *GIScience \& Remote Sensing,* vol. 54, no. 5, pp. 617-638, 2017.

[33] Y. Zhang, B. Li and K. Ramayya, "Learning individual behavior using sensor data: The case of gps traces and taxi drivers," *{SSRN} Electronic Journal,* 2016.

[34] C. S. Perone, "Google's S2, geometry on the sphere, cells and Hilbert curve," *Terra Incognita.[Online]. Available: http://CRAN. R-project. org/package= raster,* 2015.

[35] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850,* 2013.

[36] S. Ilya, O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems,* vol. 27, pp. 3104-3112, 2014.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[38] T. Mikolov, "Statistical language models based on neural networks," *Presentation at Google,* vol. 80, p. 26, 2 April 2012.

[39] R. Pascanu, T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013.

[40] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078,* 2014.

[41] M. Phi, "Illustrated Guide to LSTM's and GRU's: A step by step explanation," Medium, 28 June 2020. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. [Accessed 04 August 2020].

[42] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," 1966.

[43] www.nacgeo.com, "The Natural Area Coding System," [Online]. Available: http://www.nacgeo.com/nacsite/documents/nac.asp. [Accessed 2020 August 4].

[44] J. Xu, R. Rahmatizadeh, L. Bölöni and D. Turgut, "A sequence learning model with recurrent neural networks for taxi demand prediction," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, 2017.