Demand Forecasting Using Spectral Decomposition of Spatio-Temporal Origin-Destination Matrices

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

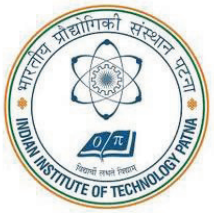
Bachelor of Technology

by

Shikhar Jaiswal
(1601CS44)

under the guidance of

Dr. Joydeep Chandra



to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY PATNA
PATNA - 801106, BIHAR

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Demand Forecasting Using Spectral Decomposition of Spatio-Temporal Origin-Destination Matrices**" is a bonafide work of **Shikhar Jaiswal** (**Roll No. 1601CS44**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Patna under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Joydeep Chandra**

Assistant Professor,

May, 2020

Department of Computer Science & Engineering,

Patna.

Indian Institute of Technology Patna, Bihar.

Acknowledgements

I'm highly grateful to my advisor Dr. Joydeep Chandra for the continuous support and guidance through out my Bachelors Project, providing me with freedom of choosing the problem that I wanted to pursue, and help with formulating the ideas and the plan of action of this work.

I'm also grateful to my mentor Manish Bhanu, Ph.D. student at IIT Patna, for his constant support during our brain storming sessions that allowed to me develop some of the key ideas presented over here. He gave me adequate space and time to complete my work at my pace, which never overburdened me and allowed to me excel in multiple dimensions.

Finally this thesis is dedicated to my parents, for their unconditional love and support through what may be known as the worst pandemic in our living memory for years to come.

Abstract

Predicting traffic from passenger pickup / drop-off demands based on historical mobility trips has been of great importance towards better vehicle distribution for the emerging on demand mobility services and allocating resources optimally. However, existing approaches for traffic flow forecast perform well in modeling and predicting, but they cause inefficiency when dealing with multi-dimensional OD data. This problem is further aggravated when additional dimensions of vehicular models, climatic conditions and so on are added. Hence, it boils down to the development of efficient tensor decomposition methods and prediction models to deal with this problem.

This thesis studies the latest state-of-the-art techniques applied to modelling passenger demands, and proposes new methods in achieving better results over the existing ideas. Furthermore, merely generating accurate predictions is not our end goal. We wish to provide a full featured interactive web application which users can use to select their source and destination points on an interactive map, and gain information on the predicted traffic from the source and destination regions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Last mile connectivity services have propped up significantly in the recent years. Ride-sharing services such as Uber, Lyft and OLA, delivery services like Swiggy and Dunzo, and e-commerce giants such as Amazon, Walmart and Snapdeal have enabled the proliferation of on-demand mobility and logistics for consumer products and services. These services also serve as a backbone to the convenience market industry, which was set for double-digit Crude Annual Growth Rate (CAGR) in several Asian countries in recent years [oGD17].

As such, it is absolutely critical for these businesses to have an in-depth understanding of the consumer demand flow patterns as they occur and change over time, to achieve demand - resource balance. A miscalculation between the demand and allocation may cause severe problems including, but not limited to, congestion, resource depletion and may drive down customer satisfaction.

For the purpose of this thesis, we study the problem of predicting mobility demands (passenger pickup and drop-off) accurately. While we focus on a single problem domain, the terminologies we develop can be easily extended into other domains mentioned. As an example, for the purpose of delivery scheduling, we can model the storage warehouses as pickup points and customer addresses as drop-offs, and using this modified input.

1.1 Origin-Destination Matrices

Origin-Destination Matrices (ODMs) are extensively used for modelling traffic movement for large cities. They reflect demand patterns of traffic networks and play important roles in traffic engineering.

Given the trip data, a geographical region for which the ODM is desired, and the time period of consideration, we begin by marking up different mutually exclusive and exhaustive territories, between which the demand characteristics are to be studied. We then define a variable $T_{i,j}$ as the number of inter-zonal trips from origin zone i to the destination zone j. Doing this for all the different ordered pairs of zones (i, j) generates the ODM for us. The diagonal values, i.e. from zone i to itself, denote the number intra-zonal trips made.

Illustrative Table			
O/D	**Zone 1**	**Zone 2**	**Zone 3**
Zone 1	$T_{1,1}$	$T_{1,2}$	$T_{1,3}$
Zone 2	$T_{2,1}$	$T_{2,2}$	$T_{2,3}$
Zone 3	$T_{3,1}$	$T_{3,2}$	$T_{3,3}$

Table 1.1: Example of an Origin-Destination Matrix

However, this two dimensional matrix merely provides us with trip statistics for a singular time interval. We can extend our ODMs by dividing our temporal duration into smaller intervals, and then stack the individual ODMs obtained for these specific intervals, to obtain a spatio-temporal three-dimensional tensor. This final OD tensor serves as the primary input and output of our modelling framework.

1.1.1 Advantages of OD Tensor based Modelling

ODMs have specific advantages which have led to their widespread popularity of use:

• First and foremost, they're extremely time-cost effective, when it comes to collection of data to the processing of the raw data into the desired matrix format. Data can be

processed directly from the raw trip logs, which is a major advantage, compared to other methods.

• They are suitable for use over a wide range of granularity pertaining to the size of the area whose characteristics they wish to represent. This allows for flexible modelling of both smaller and large areas, without much loss of robustness.

• They provide a route invariant picture of the traffic flows from one zone to another. This is helpful for models dealing with the inference of higher level map topologies (for use in delivery scheduling), than finer level flows (for example, traffic along individual routes).

1.2 Issues with Current Approaches

1.2.1 Performance

Despite considerable effort, traffic modelling is still an active area of research with a long way to go. New improvements are being introduced every year, and the field itself sees a paradigm shift every few years, the most recent being the shift to graph convolutional approaches from various time-series based neural networks

1.2.2 Capturing Meaningful Representations

It is difficult for neural network based approaches to extract spatial and temporal features from the input jointly. The representative ability of these networks are hindered seriously in the presence of narrow constraints. As we explain later, a number of these features can be learned through specific graph-based modelling techniques, to mitigate this issue.

1.3 Problem Statement

We aim to develop a robust and scalable end-to-end traffic visualization module, which takes as input Origin-Destination matrix based on taxi-trip data for any region over a duration of time, and is able to model and predict the demand flow through the same

region in the near future. This is broadly stated to be achieved through two smaller sub-tasks, namely the prediction of future demand through the use of our modelling framework to identify complex features that capture spatio-temporal influences, and developing an interactive web application for visualizing our generated predictions.

1.4 Chapter Conclusion

This chapter provides a rough background for the topics covered in this thesis. The next chapter gives related work that has been conducted in the domain and elucidates the contributions of the prior works. Chapter 3 provides an introduction to the dataset used in experimentation and explains the data representation strategies employed for constructing our model inputs, while also mathematically formulating the problem statement. Chapter 4 walks us through the front-end segment of our work, providing a detailed explanation of the capabilities of our interactive data visualization module. In chapter 5, we then introduce the related concepts of graph convolutions, and representation of signals. Chapter 6 provides an exposition of existing models which serve as the target baselines for our work, while chapter 7 introduces our approach to improve the modelling framework. Chapter 8 mentions the results of all our investigations. Chapter 9 contains some of the additional approaches we tried, but did not live up to their promise. Finally Chapter 10 concludes our findings and present the scope for future.

Chapter 2

Related Work

This chapter provides a brief overview of the prior works that have employed Spectral Decomposition and Deep Learning based techniques to model various OD tensor based prediction problems.

2.1 Background

Deep learning based approaches have become ubiquitous in many domains of demand modelling. By as early as 2015, Convolutional LSTM Networks introduced in [XCW+15], gained popularity as the preferred methods of choice for modelling spatio-temporal features. These models replaces regular matrix multiplication of vanilla LSTMs with convolutions. These methods led to the further rise of Convolutional Residual Networks in [ZZQ16].

2.2 Time Series Modelling

Building on [XCW+15, ZZQ16], [ZSZH18] showed one of the first points of application of these models to the domain of passenger demand modelling. Using Convolutional LSTM-based encoder-decoder network with MLP-based attention, they modelled New York TaxiNYC data from 2009 to 2015.

2.3 Graph Convolutions

A number of papers have worked on the problem of generalizing neural networks to work on arbitrarily structured graphs, with some of the most ground-breaking work presented in [KW16], [DBV16] and [BGAL19].

These works not only show the usefulness of Graph Convolutions in learning esoteric representations of node similarities, but also provide a theoretical basis behind the usage of these techniques under specific domains. GCNs currently form the state of the art in the problem domain we're pursuing.

2.4 Chapter Conclusion

This chapter provided details of the some of the existing techniques in popular use. A few results of these evaluations are also summarized in chapters 8 and 9. In next chapter, we discuss the dataset we have used for evaluating our hypothesis and formally define the passenger demand prediction problem.

Chapter 3

Dataset and Problem Definition

In this chapter, we present the dataset used for testing our methods and mention our approach in modelling the raw data. We also provide a formal definition of the problem statement in mathematical terms for further extension into other domains.

3.1 Green Taxi Trip Data

For the purpose of testing out our hypothesis and obtaining baseline scores on the previous state-of-the-art models, we make use of 2014 Green Taxi Trip Data [TT20] for the city of New York. The dataset is freely available at the NYC Open Data website for analysis and study, and consists of over 15.8 million records of the iconic NYC Green Taxi trips made in over 12 months of 2014.

Each record of a taxi trip comprises information related to the pickup and drop-off dates, times, geographical locations, and other essential information such as trip amount, distance, passenger counts and so on, totalling over 20 categories.

3.1.1 Pre-Processing

For the sake of brevity, we drop the fields unimportant to our cause and only keep the minimal essential categories of information, namely the following:

Data Fields		
Column Name	**Description**	**Type**
pickup_datetime	The date and time when the meter was engaged	Date & Time
dropoff_datetime	The date and time when the meter was disengaged	Date & Time
pickup_latitude	Latitude where the meter was engaged	Number
pickup_longitude	Longitude where the meter was engaged	Number
dropoff_latitude	Latitude where the meter was disengaged	Number
dropoff_longitude	Longitude where the meter was disengaged	Number

Table 3.1: Data Fields used from 2014 Green Taxi Trip Data

Furthermore, we found that a number of these trip records had noisy values of longitude and latitudes, which would place them outside the geographical boundary of New York City. Hence, we only considered the taxi trips occurring from a point within the city and terminating to a point within the city. This was made possible by considering a geographical extent (in terms of longitude and latitude) for the city. We considered the following values of the extent for pruning our data, available from [oCP13]:

Extent	
West: -74.257159	**East:** -73.699215
North: 40.915568	**South:** 40.495992

Table 3.2: Geographical Extent of New York City

We simply consider the trips for which all the values of latitude and longitude lie within the above mentioned ranges. After this massive cleanup, we obtain around 800,000 clean trip records, which we model in the manner described in the next section.

3.2 Problem Definition

We make use of the approach described in [ZZQ16], [ZSZH18] and [YYZ17] for modelling our OD tensor in graphical terms.

3.2.1 Maps to Grids

We divide the complete geographic area into mutually exclusive and exhaustive rectangular grids, with each grid denoting a specific region or zone on the city map. We obtain the outline bounding box for New York City from the lower-left coordinate $(S_{latitude}, W_{longitude})$ and the upper right coordinate $(N_{latitude}, E_{longitude})$ values obtained from Table 3.2.

To divide the obtained bounding box into $N \times M$ grids, we simply obtain individual grid lengths and breadths as $l = \frac{E_{longitude} - W_{longitude}}{N}$ and $l = \frac{N_{latitude} - S_{latitude}}{M}$ respectively. The values of l and b intuitively define the total geographic area per grid under consideration. The larger (or smaller) these values, the larger (or smaller) is the actual geographical area considered under a particular grid, and coarser (or finer) is the analysis.

We denote a grid as g_k with $k \in [1, ..., N * M]$ for some natural ordering of individual grids (row-major or column-major order). Hence, we also obtain the bounding boxes of each individual grids, which we denote as (g_k^S, g_k^W), and (g_k^N, g_k^E).

We say a geographical location $p = (latitude_p, longitude_p)$ lies within a grid g_k ($p \in g_k$) iff $latitude_p \in [g_k^S, g_k^N]$ & $longitude_p \in [g_k^W, g_k^E]$. Using the trip record fields mentioned in Table 3.1, we generate the source grid g_{source} and destination grid $g_{destination}$ for each trip and append them to the individual trip records:

Generated Fields			
Column Name	**Description**	**Type**	**Notation**
source_grid	The grid where the meter was engaged	Number	src_grid
destination_grid	The grid where the meter was disengaged	Number	dest_grid

Table 3.3: Generated Fields from the Trip Data

Finally we obtain a representation for individual trip records as the tuple:

$$(pick\_time, drop\_time, pick\_lat, pick\_long, drop\_lat, drop\_long, src\_grid, dest\_grid)$$

3.2.2 Demand Mapping as Graphs

Let $\mathcal{T}$ to be a set of taxi trips over a grid map G. Given the t-th time interval $[start_t, end_t)$, we compute the demand map $\mathcal{M}_t = \{\delta^t_{k,l}\}_{g_k, g_l \in G}$ where:

$$\delta^t_{k,l} = \{|\tau \in \mathcal{T}| \; |\tau_{src\_grid} = g_k \wedge \tau_{dest\_grid} = g_l \wedge \tau_{pick\_time} \in [start_t, end_t)\}$$

Hence we have organized the 2D demand matrix $\mathcal{M}_t \in \mathbb{R}^{N*M \times N*M}$. Stacking these matrices, in a sequence ordered by their time intervals generates the 3D demand tensor $\mathcal{M}$ over the grid map G.

Each of these individual demand maps forms a subset of an overall demand graph $\mathcal{G}$. Here in the t-th timestep, in graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{M}_t)$, where $\mathcal{V}_t \in \mathbb{R}^{N*M \times F}$ where F is the feature dimension corresponding to the observations regarding the $N * M$ individual grids. In our work, we keep F set to 2, considering only the net outgoing and incoming traffic from an individual region.

3.2.3 Demand Prediction

Considering a grid map G, T demand matrices $\{\mathcal{M}_i \mid i = 1..., T\}$ over the previous T time intervals corresponding to $\mathcal{G}_i$ demand graphs, we try to predict U demand matrices $\{\mathcal{M}_i \mid i = T+1, ..., T+U\}$ for the next U time intervals.

3.3 Chapter Conclusion

In this chapter, we introduced the dataset we use to test our hypotheses and also described the data pre-processing and representation process using mathematical constructs. This

enabled us in obtaining a cleaned up OD tensor and vertex lists from our raw taxi trip data, and also laid the foundation of our formal problem statement. The next chapter makes use of the output prediction tensors constructed using the same principles to visualize passenger taxi demands using an interactive tool.

Chapter 4

Interactive Web Module for Visualization

The data processing pipeline presented in the previous chapter serves as the desired format for both the input and the output of our modelling framework, which is described later. We now take a slight detour and describe our web application which we have developed for the purpose of easy visualization of the predicted output.

4.1 Base Web Application

For a basic running implementation of a *React* based application, we make use of starter code provided by [Tur19]. The application makes use of *Leaflet* map API and *OpenWeatherMap* API for querying map and weather data respectively. The application allows the user to click on a specific location on a map, and the bottom carousel instantaneously portrays the latest weather predictions for the same area.

4.2 Improvised Traffic Demand Application

We revamp the application to offer a real-time map based interface for choosing the source point of the journey through a simple left click, and the destination point through a right click. The map markers appear automatically, offering information relating to latitude and longitude, and trace the path of your supposed journey through the different grids. Additionally, the traced path fires a backend query for searching the internal database, corresponding to the predicted traffic information between those two regions, which is displayed on the bottom carousel as incoming and outgoing traffic volume, along with the weather information in real-time.

This query generation is done by locating the latitude and longitude information from the source and destination markers, which in turn determines their respective grid indices. These grid indices are used in a dictionary lookup for the respective pairwise traffic information.

For a simple example, imagine being in the Carnegie Hill neighbourhood of New York city, and wishing to travel to the Bronx, which is a major tourist attraction due to its world-famous The Bronx Zoo.

Starting Point: Carnegie Hill

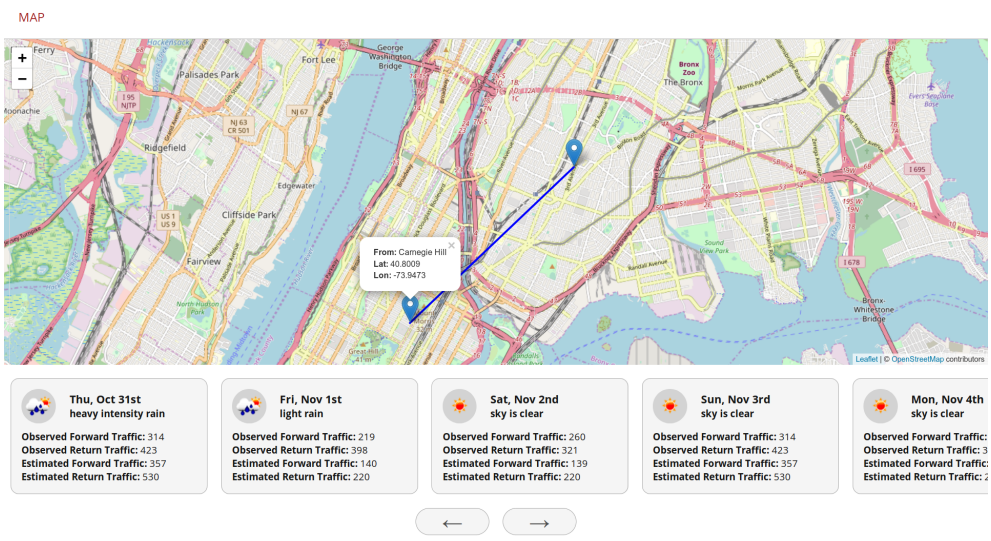


Fig. 4.1: Initial Point of the Trip

Simply, clicking on the map layout of the start and end position, will activate a couple of markers and a path tracer, which queries the database regarding the predicted traffic volume for the upcoming few days, displayed on the carousel below, along with the weather information in real-time.

Fig. 4.2: Destination Point of the Trip

4.3 Chapter Conclusion

In this chapter, we provided a tour of front-end web application, developed primarily for the purpose of providing easy access to predicted demand flow visualization. In next chapter, we provide an introduction to spectral decomposition techniques for graphs, and explain the ideas behind using those techniques for building better modelling approaches.

Chapter 5

Introduction to Graph Convolutional Networks

This chapter provides an overview of specific graph convolution techniques frequently used in the analysis of demand graphs. The material presented here follows from the seminal work presented in [KW16] and [DBV16].

5.1 Definitions

Currently, most graph neural network models have a somewhat universal architecture in common and are referred as Graph Convolutional Networks or GCNs. They are called convolutional, because filter parameters are typically shared over all locations in the graph.

For these models, the goal is then to learn a function of signals / features on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which takes as input:

• A node description list $\mathcal{V}$ containing a feature description $x_i \in \mathbb{R}^F$ for every node $i \in [1, 2, ..., N]$ such that $\mathcal{V} \in \mathbb{R}^{N \times F}$, where F is the number of input features, and

• A representative description of the graph structure in matrix form, typically in the form of an weighted adjacency matrix $\mathcal{E} \in \mathbb{R}^{N \times N}$,

and produces a node-level output $Z \in \mathbb{R}^{N \times D}$ feature matrix, where D is the number of

output features per node.

Every neural network layer can then be written as a non-linear function $H^{(l+1)} = f(H^{(l)}, \mathcal{E})$, with $H(0) = \mathcal{V}$ and $H^{(L)} = Z$, L being the number of layers. The specific models then differ only in how $f(\cdot)$ is chosen and parameterized.

5.1.1 Example

As an example, we consider the following form of a layer-wise propagation rule:

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}),$$

where $W^{(l)}$ is a weight matrix for the l-th neural network layer, A is a weighted adjacency matrix (alias for $\mathcal{E}$) and $\sigma(\cdot)$ is a non-linear activation function like the ReLU [Aga18].

5.2 Obtaining Graph Laplacian

Multiplication with A means that, for every node, we sum up all the feature vectors of all neighboring nodes but not the node itself. We fix this by enforcing self-loops in the graph by we simply add the identity matrix to A.

The second limitation is that A is typically not normalized and therefore the multiplication with A will completely change the scale of the feature vectors. Normalizing A such that all rows sum to one, i.e. $D^{-1}A$, where D is the diagonal node degree matrix such that $D_{ii} = \Sigma_j A_{i,j}$, gets rid of this problem. Multiplying with $D^{-1}A$ now corresponds to taking the average of neighboring node features. In practice, dynamics get more interesting when we use a symmetric normalization, i.e. $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. Combining these two tricks, we essentially arrive at the propagation rule introduced in [KW16]:

$$f(H^{(l)}, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

with $\hat{A} = A + I_N$, where I_N is the identity matrix of order N and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$. Expanding $\hat{A}$ in the above expression, we obtain the expression of the first-order approximation of graph Laplacian $\mathcal{L}$ as:

$$\mathcal{L} = I_N - \hat{D}^{-\frac{1}{2}} A \hat{D}^{-\frac{1}{2}}$$

5.3 Spectral Filtering

As $\mathcal{L}$ is a real symmetric positive semidefinite matrix, it has a complete set of orthonormal eigenvectors $\{u_l\}_{l=0}^{n-1} \in \mathbb{R}^n$, known as the graph Fourier modes, and their associated ordered real non-negative eigenvalues $\{\lambda_l\}_{l=0}^{n-1}$, identified as the frequencies of the graph such that $L = U\Lambda U^T$ where $\Lambda = diag([\lambda_0, ..., \lambda_{n-1}]) \in \mathbb{R}^{n \times n}$. The graph Fourier transform of a signal $x \in \mathbb{R}^n$ is then defined as $\hat{x} = U^T x \in \mathbb{R}^n$, and its inverse as $x = U\hat{x}$ [SNF+13].

The convolution operator on graph $\star\mathcal{G}$ is defined in the Fourier domain such that $x \star \mathcal{G} y = U((U^T x) \odot (U^T y))$, where $\odot$ is the element-wise Hadamard product. It follows that a signal x is filtered by g_θ as

$$y = g_\theta(\mathcal{L})x = g_\theta(U\Lambda U^T)x = U g_\theta(\Lambda) U^T x$$

A non-parametric filter, i.e. a filter whose parameters are all free, would be defined as $g_\theta(\Lambda) = diag(\theta)$, where the parameter $\theta \in \mathbb{R}^n$ is a vector of Fourier coefficients.

5.4 Using Chebyshev Approximation

Using the above representation of g_θ, we can develop a parameterized polynomial version of the above filter as,

$$g_\theta(\Lambda) = \Sigma_{k=0}^{K-1} \theta_k \Lambda^k$$

However, the above approximation is computationally expensive with $\mathcal{O}(n^2)$ operations because of the Λ being in $\mathbb{R}^{n \times n}$. A solution to this problem is to parameterize $g_\theta(\mathcal{L})$ as

a polynomial function that can be computed recursively using $\mathcal{L}$ itself. By taking the Chebyshev approximation,

$$g_\theta(\Lambda) = \Sigma_{k=0}^{K-1}\theta_k T_k(\widetilde{\Lambda})$$

the filtering operation can then be written as $y = g_\theta(\mathcal{L})x = \Sigma_{k=0}^{K-1}\theta_k T_k(\widetilde{\mathcal{L}})x$, where $T_k(\widetilde{\mathcal{L}}) \in \mathbb{R}^{n\times n}$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\widetilde{\mathcal{L}} = 2\mathcal{L}/\lambda_{max} - I_N$. Chebyshev polynomial $T_k(x)$ of order k may be computed by the stable recurrence relation,

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

with $T_0 = 1$ and $T_1 = x$. Using the above two equations, and denoting $\overline{x}_k = T_k(\widetilde{\mathcal{L}})x \in \mathbb{R}^n$, we can use the recurrence relation to compute $\overline{x}_k = 2\widetilde{\mathcal{L}}\overline{x}_{k-1} - \overline{x}_{k-2}$ with $\overline{x}_0 = x$ and $\overline{x}_1 = \widetilde{\mathcal{L}}x$. The entire filtering operation $y = g_\theta(\mathcal{L})x = [\overline{x}_0, ..., \overline{x}_{K-1}]\theta$ then costs $\mathcal{O}(K|\mathcal{E}|)$ operations which is suitable for sparse $\mathcal{L}$.

5.5 Chapter Conclusion

In this chapter, we introduced some foundational concepts of Graph Convolutions and Spectral Filtering techniques, and explained the proposed benefit of specific filters. The next chapter presents some of the previously implemented ideas in this domain, which serve as the baseline models for comparison with our approaches.

Chapter 6

Baseline Models

This chapter provides an exposition of the various baseline models we have worked with. To the best of our knowledge, the baseline models form the current state-of-the-art while considering graph convolutional models for passenger demand forecasting.

6.1 Baseline Graph Convolution Network with CGRNNs (STM-GCN)

Following the architecture given in [GLW+19], the authors argue that two important aspects are largely are non-Euclidean pair-wise correlations between regions, and processing regions only based on local information.

Hence they present different aspects of correlations between regions as graphs, whose vertices represent regions and edges encode the pair-wise relationship among regions. First, they use the proposed Contextual Gated Recurrent Neural Network (CGRNN) to aggregate observations in different times considering the global contextual information. After that, multi-graph convolution is applied to capture different types of correlations between regions. Finally, a fully connected neural network is used to transform features into the prediction. The following three features are used:

- **Neighborhood:** Neighborhood of a region is defined based on the spatial proximity.

They construct the graph by connecting a region to its 8 adjacent regions in a 3×3 grid.

• **Function Similarity:** Region functionality could be characterized using its surrounding points of interest for each category, and the edge between two vertices (regions) is defined as the POI similarity:

$$A_{S,i,j} = sim(P_{v_i}, P_{v_j}) \in [0, 1]$$

where P_{v_i}, P_{v_j} are the POI vectors of regions v_i and v_j.

• **Transport Connectivity:** Here, they define regions that are directly connected by these roads as "connected" and the corresponding edge is defined as:

$$A_{C,i,j} = max(0, conn(v_i, v_j) - A_{N,i,j}) \in \{0, 1\}$$

where $conn(u, v)$ is the indicator function of the connectivity between v_i and v_j.

6.2 Baseline Grid-Embedding based Multi-Task Learning (GEML)

Following the architecture given in [WYC+19], we build our next baseline model which broadly makes use of two grid-embedding features, due to the limitation of GCNs on grids with low-scale demands:

• **Geographical Neighborhood:** For a grid g_i, its geographical neighbor set is formulated as:

$$\Phi_i = \{g_j \mid dis(g_i,\ g_j) \leq L\}$$

where $dis(g_i,\ g_j)$ denotes the spatial distance of these two grids' centers, the straight length from g_i's center to g_j's geographically, and L is a threshold of the distance which can determine the range of neighborhood.

• **Semantic Neighbourhood:** For grid g_i, within an arbitrary time slot $t^{'} = 1, 2, ..., t$,

we can obtain a set of its semantic neighbors via:

$$\Omega^i_{t^{'}} = \{g_j \mid m_{i,\,j} > 0 \;||\; m_{j,\,i} > 0,\; m_{i,\,j} \;\in\; M_{t^{'}},\; m_{j,i} \;\in\; M_{t^{'}}\}$$

where $m_{i,j}$ is the demand between grid i and grid j and $M_{t^{'}}$ is the demand tensor at timestep $t^{'}$.

They make use the vanilla LSTM framework to aggregate observations in different timesteps into a suitable representation. After that, they take the grid embedding vector sequence $\{v^i_1, v^i_2, ..., v^i_t\}$ as inputs and further convert their learned input representation into a periodic-skip LSTM which skips irrelevant sequential patterns. Finally they make use of multi-task learning paradigm to train the framework on different loss objectives, namely outbound traffic flow, inbound traffic flow and overall traffic flow. Rest of the details follow from the paper.

6.3 Chapter Conclusion

In this chapter we have provided a brief overview of the different baseline models used in our work. Working upon these models, we present our approaches in the next chapter.

Chapter 7

Proposed Methodology

We now present our own model Iterative-GEML on top of the previous baseline models, and also explain the motives behind pursuing our approaches.

7.1 Issues with Baseline GEML

7.1.1 Use of Grid-Embeddings

Grid embeddings are particularly inspired by the message passing schema of the GCNs. While being powerful methods for learning features from spatio-temporal data, Grid Embeddings are particularly tricky to train in conditions of dense inputs. The original GEML model only learns a joint representation of embedding outputs to generate the prediction for the next timestep. To mitigate this effect, we condition the individual grid embedding modules to generate the prediction for the next timestep using the same LSTM-based encoder-decoder architecture which is used to learn the joint representations.

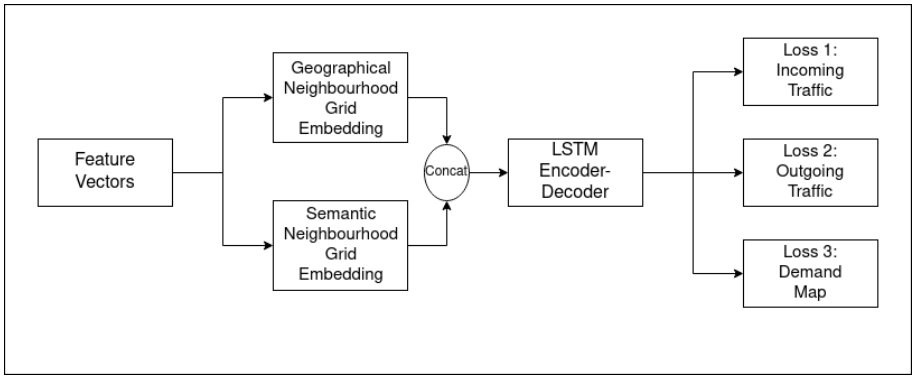


Fig. 7.1: Original GEML Model

7.1.2 Multi-Task Learning

In the baseline GEML model, the authors made use of multi-task learning approach for modelling different aspects of the expected output separately. While this technique holds promise for sparse inputs, in our case, this situation leads to an overlap in the loss objectives, which leads to model noise. Hence we only make use of singular objective functions to train our model.

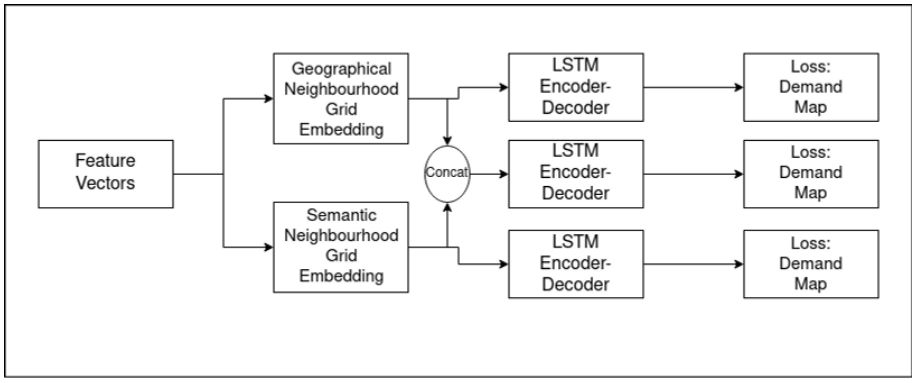


Fig. 7.2: Iterative-GEML Model

7.1.3 Chebyshev Filtering

As a final variation of our model, we generate the Chebyshev approximation representation of the Grid Embedding input, and pass them in place of the original input. We propose that this technique would help the model to learn better signals from the original input to train the model.

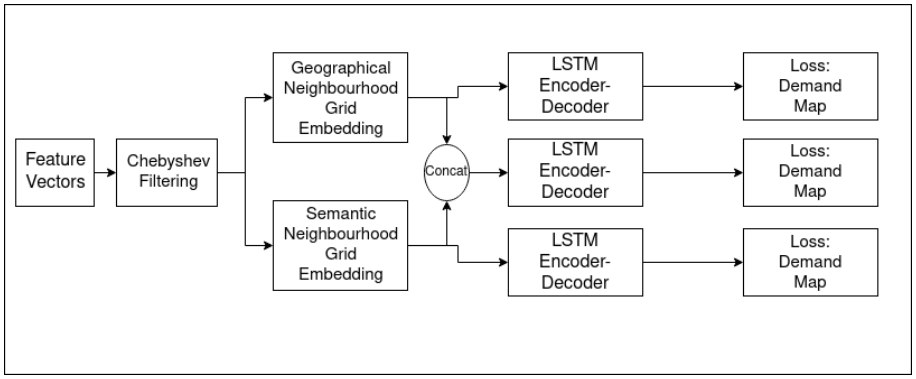


Fig. 7.3: Iterative-GEML Model with Chebyshev Filter

7.2 Chapter Conclusion

In this chapter, we have proposed our improved Iterative-GEML model for rectifying some of the shortcomings of the original GEML model. The next chapter presents our evaluation study on all of the introduced models and portrays the empirical promise of our approach.

Chapter 8

Experiments and Results

This chapter describes the various experiments we tried out with model variations described in the previous chapter. We ran all models under the same experimental settings, and obtained results for the baselines as well as our own variations.

8.1 Experimental Setup

We pre-processed the dataset to generate demand graphs as described in Chapter 3. For quicker testing, we have only made use of the first 3 months of the data. By default, we set time interval to 1 hour, while training the model on the latest 24 hour data, before generating predictions. For the Green Taxi Trip dataset, we used 20×20 grids to partition the whole city of New York, where each grid covers 2.5km $\times$ 2.5km area. This gives us over 400 individual grids covering the entire city of New York.

Since New York city is itself located alongside the Atlantic Ocean, we find that certain portions of the map include water bodies and canals. While this issue might be less of a problem in cases of grids covering two major islands, certain grids are almost entirely covered by water bodies. This leads to unnecessary complexity in the STM-GCN and GEML models, where we feed grid information in the form of weighted adjacency matrices. As such, to reduce this complexity, we simply drop all the grids where the average traffic

over a time interval considered is less than a specific threshold (30 cars in our case). This gives us 55 land-connected grids to train our baseline and improved models.

The entire set of observations is processed into a 80-20 split for training and testing subsets. Each individual model is trained for 10 epochs until convergence on the cross-validation split of 0.20. The predictions are made in a multi-step forecasting manner with rolling window approach. Hence, for every forecast on timestep t to $t + k$, we make use of the past $t - 1$ to $t - 24$ timesteps, where $k \in [0, 1, 2, ...11]$.

8.2 Results

For each predicted demand graph $\mathcal{G}_t^{pred}$, we calculate the difference with the true demand graph $\mathcal{G}_t^{true}$, and further use this sequence of differences to calculate the grid-wise RMSE error as:

$$RMSE_{graph} = \sqrt{\frac{\Sigma_{(true,pred)\in G_{test}}(\mathcal{G}_t^{true}.\mathcal{M}_t - \mathcal{G}_t^{pred}.\mathcal{M}_t)^2}{|G_{test}|}}$$

Taking an average of the grid-wise RMSE values gives us the Aggregate Mean Grid RMSE over the entire time window of consideration. Dividing by the size of the time-window gives us the Average Mean Grid RMSE per hour.

RMSE Scores: STM-GCN (with Chebyshev Filter)		
Time Window Forecast (hours)	**Aggregate Mean Grid RMSE (over entire time window)**	**Average Mean Grid RMSE (per hour)**
1	4.2480	4.2480
2	10.6932	5.3466
3	16.5633	5.5211
4	21.9328	5.4832
5	27.9115	5.5823
6	33.9786	5.6631
7	40.0603	5.7229
8	46.4328	5.8041
9	52.5006	5.8334
10	58.4	5.8400
11	64.3907	5.8537
12	70.6188	5.8849

Table 8.1: RMSE Scores Obtained from Baseline STM-GCN Model with Chebyshev Filter

RMSE Scores: GEML		
Time Window (hours)	**Aggregate Mean Grid RMSE (over entire time window)**	**Average Mean Grid RMSE (per hour)**
1	4.4934	4.4934
2	8.9604	4.4802
3	13.3278	4.4426
4	17.1028	4.2757
5	23.0640	4.6128
6	29.3772	4.8962
7	34.8796	4.9828
8	40.04	5.0050
9	46.0476	5.1164
10	52.067	5.2067
11	58.4529	5.3139
12	64.4088	5.3674

Table 8.2: RMSE Scores Obtained from Baseline GEML Model

RMSE Scores: Iterative-GEML		
Time Window (hours)	**Aggregate Mean Grid RMSE (over entire time window)**	**Average Mean Grid RMSE (per hour)**
1	4.0374	4.0374
2	7.4940	3.7470
3	11.1495	3.7165
4	14.8464	3.7116
5	18.5975	3.7195
6	22.3176	3.7196
7	26.0092	3.7156
8	29.7288	3.7161
9	33.4638	3.7182
10	37.137	3.7137
11	40.8452	3.7132
12	44.5404	3.7117

Table 8.3: RMSE Scores Obtained from our Iterative-GEML Model

The Iterative-GEML approach provides us with a decent performance boost of 11% across time windows. This needs to be further verified over bigger and varied datasets.

RMSE Scores: Iterative-GEML with Chebyshev Filter		
Time Window (hours)	**Aggregate Mean Grid RMSE (over entire time window)**	**Average Mean Grid RMSE (per hour)**
1	4.3775	4.3775
2	7.5054	3.7527
3	11.1315	3.7105
4	14.8376	3.7094
5	18.5565	3.7113
6	22.2624	3.7104
7	25.9770	3.7110
8	29.6880	3.7110
9	33.39	3.7100
10	37.080	3.7080
11	40.7627	3.7057
12	44.5104	3.7092

Table 8.4: RMSE Scores Obtained from our Iterative-GEML model with Chebyshev Filter

The further addition of Chebyshev filter adds an additional performance gain over long time windows, however, might lead to a slight degradation in performance under shorter time windows.

8.3 Chapter Conclusion

In this chapter, we have illustrated the results obtained from the various models we exposited in the previous chapter. These results provide empirical evidence regarding the robustness of our proposed approaches. Next chapter presents some additional approaches that we developed over the course of our work.

Chapter 9

Additional Approaches

This chapter describes the various experiments and model variations we tried out during the course of this thesis, that didn't meet the objectives that they were intended for. While some models performed well compared under certain conditions, overall, they did not perform well on evaluation metrics.

9.1 Iterative-GEML with Multi-Task Learning

While we had discarded the original Multi-Task Learning paradigm present in the original GEML baseline model, we had done so because of the overlap among the different tasks. Here we present the Iterative-GEML with the original Multi-Task Learning approach. As expected, the results are nothing short of underwhelming.

RMSE Scores: Iterative-GEML with Multi-Task Learning		
Time Window (hours)	**Aggregate Mean Grid RMSE (over entire time window)**	**Average Mean Grid RMSE (per hour)**
1	4.5123	4.5123
2	9.0178	4.5089
3	12.9762	4.3254
4	18.1092	4.5273
5	22.43	4.4860
6	26.60400	4.4340
7	33.1919	4.7417
8	40.2256	5.0282
9	42.3513	4.7057
10	46.674	4.6674
11	56.3651	5.1241
12	55.6860	4.6405

Table 9.1: RMSE Scores Obtained from our Iterative-GEML Model with Multi-Task Learning

9.2 Iterative-GEML with ARMA Filter

ARMA spectral filtering technique introduced recently in [BGAL19] has recently gained traction as one of the computationally efficient techniques. Inspired by popular statistical technique of Auto Regressive Moving Average (ARMA) models, these filters are shown to be robust to perturbations in underlying graphs, while being stable enough to be computed efficiently. While a detailed analysis of these filters is out of the scope of this thesis, we tried running our Iterative-GEML model with ARMA filters.

RMSE Scores: Iterative-GEML with ARMA Filter		
Time Window (hours)	**Aggregate Mean Grid RMSE (over entire time window)**	**Average Mean Grid RMSE (per hour)**
1	4.5348	4.5348
2	9.0164	4.5082
3	13.4991	4.4997
4	18.8588	4.7147
5	22.2360	4.4472
6	28.7490	4.7915
7	34.3959	4.9137
8	34.8024	4.3503
9	43.6122	4.8458
10	45.044	4.5044
11	49.5913	4.5083
12	60.7236	5.0603

Table 9.2: RMSE Scores Obtained from our Iterative-GEML Model with ARMA Filter

9.3 Chapter Conclusion

In this chapter, we have presented experimental results from additional variations that we tried on our proposed models. We leave the analysis of these results as a thought for the future. The next chapter offers a conclusion for the entire thesis, and mentions the scope of future work.

Chapter 10

Conclusion and Future Work

In this work, we studied the problem of predicting multi-step city-wide traffic volume and developed a robust traffic prediction framework. On the user interface side of things, we implement an interactive map-based web application for querying our results.

On the theoretical side, we attempt to solve the problem by implementing recent advances in deep learning based approaches. We build models employing Graph Convolutional Networks, Spectral Decomposition and Approximation techniques to emphasize the effects of representative city-wide traffic patterns on each-step prediction during the decoding phase.

In this work, we have implemented and obtained RMSE scores for NYC Green Taxi Trip Data for the months of January to March 2014 for two different baseline models. We have experimented with a hybrid approach to fuse the features of two baseline models in order to achieve better performance, and obtained empirical results. There are a few further directions this project could take.

10.1 Testing on Varied Datasets

Currently, we only train for the first three months of 2014 in order to validate our hypothesis quickly. Hence, a natural next step for this work would be to validate our findings on larger

and diverse sets of data. One possible way of achieving this objective could be to use the Green Taxi data from all years (2013 - 2019). Another possible alternative would be to use datasets for other cities around the world.

10.2 Making Use of Attention-based Transformer Networks

Recent trends in attention-based transformer networks have shown immense potential in terms of better empirical performance [ZSZH18, YWK+18]. One of the most seminal works by [VSP+17] have shown that given a more natural approach to modelling long target sequences as weighted sums of smaller input subsequences, we can obtain better evaluation scores. Hence, the next possible strategy would be to make use of Transformer Networks to model this problem.

References

[Aga18] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[BGAL19] Filippo Maria Bianchi, Daniele Grattarola, Cesare Alippi, and Lorenzo Livi. Graph neural networks with convolutional arma filters. *arXiv preprint arXiv:1901.01343*, 2019.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc., 2016.

[GLW+19] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3656–3663, 2019.

[KW16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[oCP13] Department of City Planning. New york city borough boundary metadata. `https://www1.nyc.gov/assets/planning/download/pdf/data-maps/open-data/nybb_metadata.pdf?ver=19c`, 2013.

[oGD17] The Institute of Grocery Distribution. Vietnam tipped to be asia's fastest growing convenience market. https://www.igd.com/articles/article-viewer/t/igd-vietnam-tipped-to-be-asias-fastest-growing-convenience-market/i/16565, 2017.

[SNF+13] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[TT20] Taxi and Limousine Commission (TLC). 2014 green taxi trip data — nyc open data. https://data.cityofnewyork.us/Transportation/2014-Green-Taxi-Trip-Data/2np7-5jsg, 2020.

[Tur19] Ivan Turashov. React's weather forecast application using openweathermap api. https://github.com/IvanTurashov/react-weather-forecast, 2019.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[WYC+19] Yuandong Wang, Hongzhi Yin, Hongxu Chen, Tianyu Wo, Jie Xu, and Kai Zheng. Origin-destination matrix prediction via graph convolution: A new perspective of passenger demand modeling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1227–1235, New York, NY, USA, 2019. Association for Computing Machinery.

[XCW+15] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach

for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[YWK+18] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[YYZ17] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[ZSZH18] Xian Zhou, Yanyan Shen, Yanmin Zhu, and Linpeng 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*, WSDM '18, page 736–744, New York, NY, USA, 2018. Association for Computing Machinery.

[ZZQ16] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceeding of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, November 2016.