B.Sc. in Computer Science and Engineering Thesis

Cross City Deep Transfer Learning Model for Crime Prediction

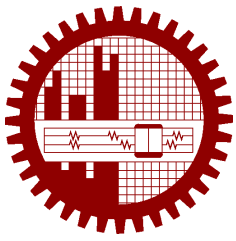
Submitted by

Nazia Afreen
201605015

Anisha Islam
201605038

Supervised by

Tanzima Hashem



Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

May 2022

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis, titled, "Cross City Deep Transfer Learning Model for Crime Prediction", is the outcome of the investigation and research carried out by us under the supervision of Tanzima Hashem.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Nazia Afreen
201605015

Anisha Islam
201605038

CERTIFICATION

This thesis titled, **"Cross City Deep Transfer Learning Model for Crime Prediction"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in May 2022.

Group Members:

Nazia Afreen

Anisha Islam

Supervisor:

Tanzima Hashem
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

ACKNOWLEDGEMENT

First and foremost, we want to convey our eternal appreciation to Almighty Allah for keeping us safe during the COVID-19 epidemic and allowing us to finish our thesis work in time.

We express our gratitude to our supervisor Dr. Tanzima Hashem who has guided us thoroughly with her invaluable direction and suggestions during our research work. We are fortunate to have such a skilled and knowledgeable supervisor for our first research work.

We want to thank Yeasir Rayhan for finding time in his busy schedule and assisting us with innovative ideas. His research on crime prediction has allowed us to pursue our work further.

We are also thankful to Fariha Tabassum Islam for providing us with the necessary dataset.

We are grateful to the Department of Computer Science and Engineering for supporting us with resources. Lastly, we would like to thank our parents, family, and friends for their constant support and encouragement.

Dhaka
May 2022

Nazia Afreen

Anisha Islam

Contents

List of Figures

List of Tables

ABSTRACT

Predicting crime ahead of time can enable law enforcement authorities to take preventive measures and save people from unwanted events. Existing research has reasonably used various statistical and deep learning models to predict crime events using historical data. However, these models are not applicable to the cities where crime data is scarce, not documented, and even not properly collected. In other domains like air pollution and human mobility prediction, transfer learning which reuses knowledge, has been shown as an effective tool to address the data scarcity issue. In this thesis, we propose a cross-city deep transfer learning model for crime prediction. For this purpose, we choose one city with available crime data as the source city and another city with no available crime data as the target city. We introduce a novel region similarity learning technique to learn the similar regions from the source city that show a similar crime pattern as a region in the target city. We first create feature embeddings for each region in the source and target cities using auxiliary data like taxi flow, road network, and points of interest (e.g., shopping centers, entertainment centers). Then we use the feature embeddings in the region similarity learning module to find the similar regions from the source city for a region in the target city. After that, we learn the crime embeddings of the regions in the source city. Finally, our model uses the weighted crime embedding of the m most similar regions from the source city and the feature embedding of the region for which crime is to be predicted, and predicts the number of crime events for a particular crime category for that region in a specific time step. We perform extensive experiments using real datasets to show the effectiveness of our model over the literature. Our model can predict crime events with reasonable accuracy for a region of the city with no crime data in a particular time period.

Chapter 1

Introduction

Criminal activities pose a significant threat to ensuring the welfare of people living in the society. The occurrences of crimes are escalating and likewise affecting the inhabitants' quality of life as each year passes [1] [2]. Predicting crime events in advance can help the security enforcing authority to take preventive measures and save people from unwanted circumstances. Researchers have developed efficient and effective crime prediction models [3–8] for regions using their historical crime data. However, the existing crime prediction models can only be applied in areas where crime data is abundant. Predicting criminal activities imposes a challenge for cities where digitization is yet to occur pervasively and historical crime data is unavailable.

Starting from statistical methods for crime predictions [9–11] that only use historical crime data, techniques involving [12] machine learning models and data mining incorporating features other than crime data are also introduced. Recent techniques have considered developing deep learning models [4–6, 8] for crime prediction. The deep learning methodologies aim to overcome the shortcomings of machine learning models, which cannot fully extract relevant features from the data effectively [13]. Similarly, the statistical machine learning models cannot fully capture crime's spatio-temporal and feature correlations. In contrast, deep learning techniques can properly encapsulate the dependencies to a certain magnitude and improve the overall accuracy of the prediction model.

However, abundant crime data is the primary basis for predicting these statistical and deep learning models. In conjunction with other embeddings (e.g., points of interest (poi), demographic, taxi flow), crime data of a region serves as an independent input feature. Existing literature depicts spatio-temporal relations of an area based on crime. Therefore, external features alone cannot serve the purpose of predicting crimes since they cannot capture the crime domain without a crime dataset.

This thesis addresses the absence of crime data by introducing a transfer learning-based crime prediction model. We use region matching as the basis for transfer learning. The intuition

behind the concept is that similar regions have similar crime patterns. Hence, we attempt to find the most similar source regions for a particular target region to predict crime occurrences in the target region before they take place.

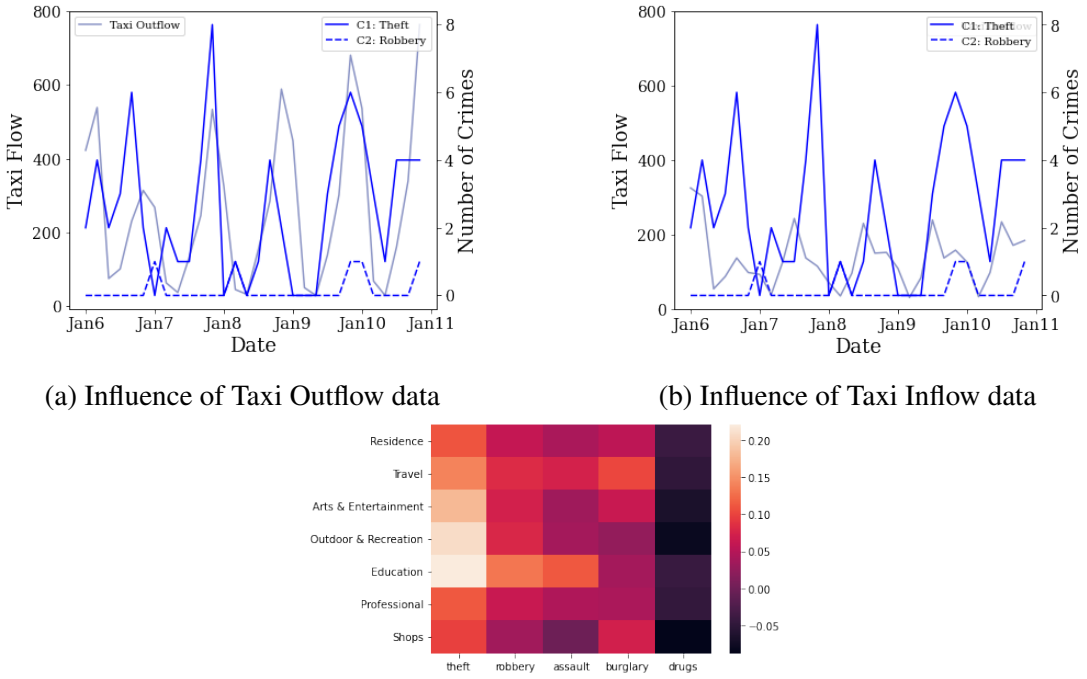
1.1 Challenges

Transfer Learning methods for crime prediction come with their own set of limitations. The challenges for a transfer learning-based crime prediction model are -

(i) Finding out the attributes that affect the crime occurrences in a region

(ii) Identifying similar regions with the same crime patterns based on external attributes

We use the following features taxi flow, poi, and road network as the external attributes that tend to affect the crime occurrences in a region. The intuition behind using the external attributes in a Chicago community area 8 is explained in Figure 1.1.



(a) Influence of Taxi Outflow data

(b) Influence of Taxi Inflow data

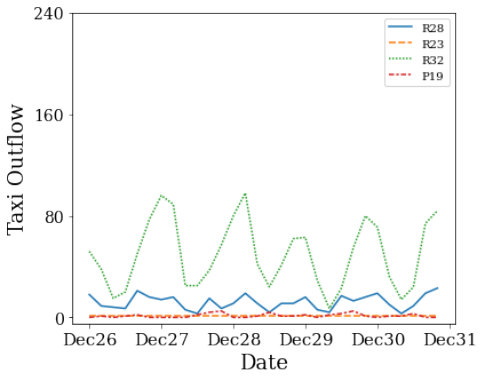
(c) Influence of POI data

Figure 1.1: Effect of external attributes on Theft

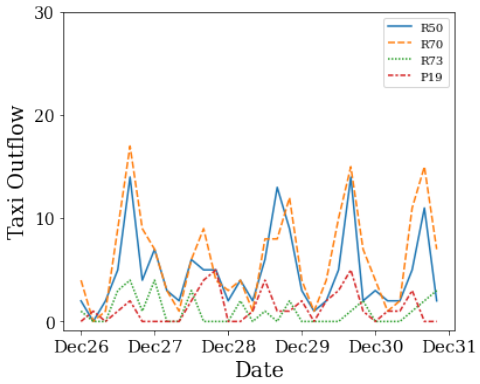
Figure 1.1a and Figure 1.1b show positive correlation between taxi flow and number of thefts. Whereas outflow exerts less influence on the number of robberies.

From Figure 1.1c, the Pearson Coefficient Correlation strongly implies that the number of burglaries happens more in areas that have more POI instances of Travel. On the other hand, narcotics have a weaker interrelation with the POI types.

These attributes that affect the crimes in a region are used to find similar regions and develop a novel deep learning model to find the matching regions. We have already established in Figure 1.1 that these attributes have a positive correlation with crimes. Therefore, it is conclusive that regions with similar external features will have similar crime distribution.

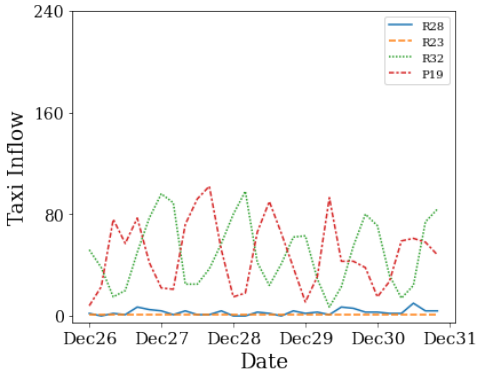


(a) Taxi outflow for Theft

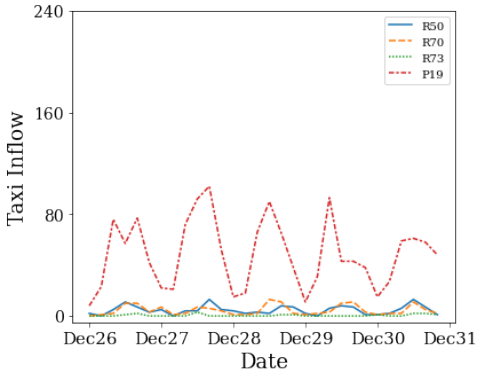


(b) Taxi outflow for Robbery

Figure 1.2: Taxi outflow in similar regions

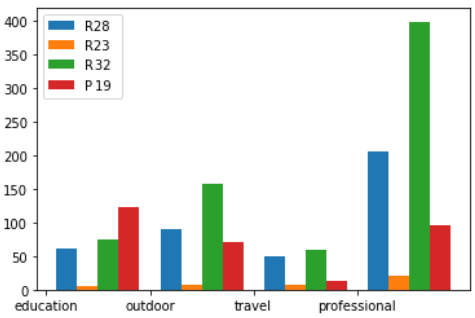


(a) Taxi inflow for Theft

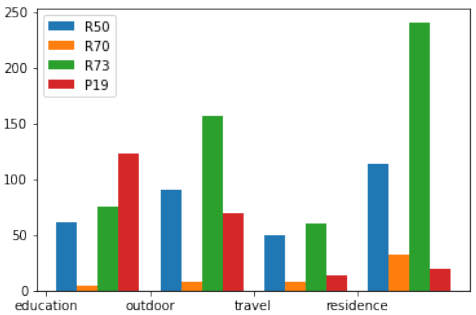


(b) Taxi inflow for Robbery

Figure 1.3: Taxi inflow in similar regions



(a) POI distribution for Theft



(b) POI distribution for Robbery

Figure 1.4: Points of interest (POI) distribution in similar regions

For crime categories of theft and robbery in precinct 19 of New York City and communities 28, 23, 32 of Chicago, taxi outflow similarity is shown in Figure 1.2, taxi inflow resemblance is

shown in Figure 1.3 and relatively similar POI distribution is illustrated in Figure 1.4. We observe that although the numbers may fluctuate, a similar trend for taxi flow and POI distribution are existent in similar regions.

1.2 Solution Overview

To address the issues mentioned in Section 1.1, we develop a cross city deep transfer learning model for crime prediction to predict the number of crimes of various categories in each time step in a region. Figure 1.5 shows that our model consists of four units -

(i) Region-representation learning unit

(ii) Region-similarity learning unit

(iii) Crime embedding learning unit

(iv) Crime prediction unit

In the first phase, our model integrates spatio-temporal and ubiquitous dependencies to achieve a recent functional embedding of a region in the region-representation learning unit. Then in the region-similarity learning unit, a region mapping function is used to find the topmost m similar regions in the source city and destination city in the time domain. The crime embedding learning module gives weight to crime data of similar regions. Finally, the crime prediction unit takes crime embedding and target region feature embedding as input and predicts crime of various categories at a particular time in the target region.

The first phase of our model is the region-representation learning unit. Crime forecasting accuracy differs with the change in the spatial representation of a region. Over the years, region embedding has been used in other spatio-temporal fields like air quality, water quality, city mobility, and thermal comfort, introducing transfer learning in their predictive models. Each region embedding was represented using only an auxiliary feature in some existing work [14, 15] or using only the corresponding domain data [16]. Our model takes novelty in the region representation as several relevant external features represent each region dynamically.

Our model learns similar feature-based region representation rather than using only grid-based or distance-based proximity. The insight behind this region learning layer is that -

(i) A direct neighborhood region connected through a densely interconnected hub with a particular region has more effect than a one-hop neighboring region with a sparse hub.

(ii) Neighborhood regions located more than one hop from a particular region in the graph structure tend to influence crime patterns of that region if they are part of a dense network.

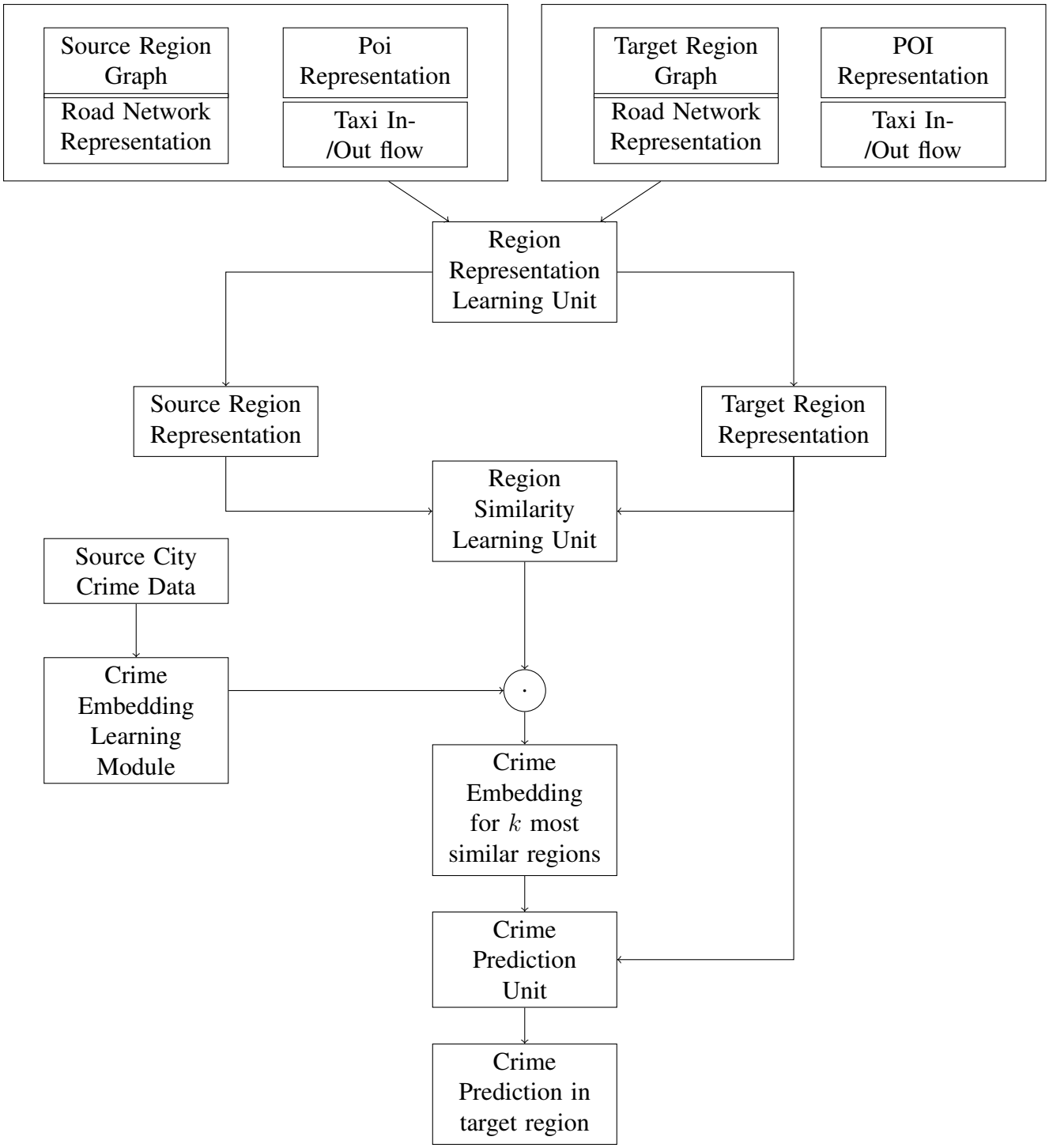


Figure 1.5: Solution overview

Our approach in the region-representation learning unit learns weighted dependencies of those relatively far regions to integrate these intuitions in our model. It incorporates them likewise if they exert sufficient effect. We consider each city region as a node where its feature vector consists of external features, for example- points of interest (POI), road or traffic junction, historical taxi inflow data, and taxi outflow data. Features like POI and road junctions are static and depict spatial views. In contrast, historical taxi trip data varies with time, illustrating

temporal factors of our region representation. The static features are passed through a graph autoencoder (GAE) [17] since it compresses the original representation and reconstructs the final representation that captures only the concentrated essential information.

This static representation is concatenated with the dynamic feature, taxi trip data, which is prepared based on a week to learn a feature embedding of each node. In this way, every node learns its embedding of the corresponding features. After that, a structural fingerprint [18] of each node was created using random walk with a restart (RWR) [19] around the region node that considers not only the adjacent regions but also the k-hop nodes.

Similar source and destination regions tend to show resemblance in the criminal activities. Thus, after learning representations of the source and the destination regions, our region-similarity learning unit requires a region mapping function to discern similar regions. A region's representation has a temporal dependency that requires dynamic embedding. Therefore, region similarity approaches will also require dynamic comparison. The existing model uses Pearson coefficients [14] and Jaccard similarity [20] to learn the similarity among the regions where the regions are based on only the target domain. These techniques do not evolve with time, resulting in a static comparison. Thus, they cannot comprehend the dynamic similarity measurement. Regional mapping of the source and target regions can be leveraged by using transformer models to integrate adaptive region similarity in each epoch during the training of the deep learning model. The main novelty of these models is attention-based mechanisms. However, transformers act poorly with large input sequences by considering all possible pairs' similarity scores and introducing redundancy. To optimize the computation time and memory requirement, sparse attention-based GAT [21] is used to compute a limited but relevant number of similarity scores from all possible pairs.

For transferring knowledge of the crime domain of the source region to the destination region, we learn the crime embedding of the source city region nodes by processing the historical crime data of the most similar source regions by using hGAT [8], a variant of GAT. Traditional graph attention networks only consider the adjacent nodes. Whereas hGAT encapsulates the architecture of GAT with parent feature information where parent means the side of the city where each region lies.

Each node's concatenated crime and feature embedding acts as an input to our crime prediction unit. Sparse attention-based LSTM (SAB-LSTM) [22] is used as our crime prediction unit since the attention mechanism searches for the most relevant data in the temporal domain. The sparse model eliminates redundant information in the long sequence and mitigates vanishing gradient problems.

1.3 Contribution

The contribution of our work is as follows:

(i) We address the crime data scarcity issue by developing a transfer learning-based deep learning model to predict crimes of various categories at a particular time instance in each target city region.

(ii) We propose a novel deep learning technique to find similar regions for crime prediction. We have used crime embeddings from these similar source regions as the alternative for historical crime data for the target city region.

(iii) We perform extensive experiments to evaluate the effectiveness of our transfer learning-based crime prediction model. Experiment results show that our model gives similar performances in both cases where we use transfer learning without crime data to train our model and use crime data to train a model. Our model also gives better performance than the case where we train a model using only features without transfer learning. In addition, transfer learning results better when region similarity is considered adaptively rather than statically.

1.4 Organization

The rest of this thesis is organized in the following way: Chapter 2 defines some commonly used terms in this thesis and gives a concrete definition of our proposed problem. After that, Chapter 3 discusses the existing literature related to our work. Chapter 4 describes our model architecture, and Chapter 5 compares the experiments' performance. Finally, Chapter 6 concludes our thesis by summarizing our work and discussing scopes for future improvements.

Chapter 2

Preliminaries and Problem Formulation

In this chapter, we explain some commonly used terminologies used in this thesis and formally define our problem. In Section 2.1, we present some terms, concepts, and notations that are required to formulate our problem and understand our methodology. Finally, in Section 2.2 we define our problem and list the notations used in this thesis in Table 2.1.

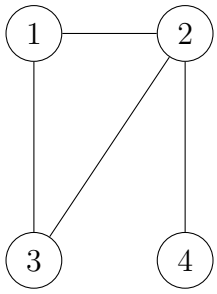
2.1 Definition

2.1.1 Graph Neural Network (GNN)

A graph with V vertices, where $|V| = n$, and E edges, where $|E| = e$, can be denoted as $G(V, E)$. In this thesis, we use vertices and nodes interchangeably. If each node of a graph has a node feature vector of length l, then the feature vector for node i can be represented as h_i, where $h_i \in \mathbb{R}^l$. Furthermore, an adjacency matrix $A \in \mathbb{R}^{n \times n}$ gives information about the connectivity of the nodes in a graph. $A_{i,j} = 1$ means that node i and node j have an edge between them and $A_{i,j} = 0$ represents that node i and node j are not connected with each other. Figure 2.1a illustrates a graph G with 4 nodes and 4 edges and Figure 2.1b shows the feature matrix of all the nodes. The adjacency matrix of this graph is also shown in Figure 2.1c.

For ease of calculation we construct a feature matrix h from the node feature vectors $h_1, h_2, \ldots, h_n$. Graph neural network [23] multiplies the node feature matrix h with a weight matrix $W \in \mathbb{R}^{l \times l'}$, where l' signifies the dimension of the transformed feature vectors, to learn a higher-dimensional feature embedding matrix $h*$ of the nodes.

$$h_i* = h_i \times W$$



(a) A graph G with 4 nodes and 4 edges

h_1					
h_2					
h_3					
h_4					

(b) Node feature matrix h of size 4×5

1	1	1	0
1	1	1	1
1	1	1	0
0	1	0	1

(c) Adjacency Matrix A

Figure 2.1: An example of a graph G, the feature matrix, and the corresponding adjacency matrix are shown above. Each node has a feature vector h_i of size $l = 5$ in this example. In the adjacency matrix, $A_{1,2} = 1$ represents there is an edge between node 1 and node 2.

where $h_i* \in \mathbb{R}^{l'}$ and transformed feature matrix $h* \in \mathbb{R}^{n \times l'}$.

To incorporate the neighbor information, for each node, we sum up the value of the transformed neighbor embeddings (including the transformed embedding of the node itself, the adjacency matrix should have all $A_{i,i} = 1$ for this reason) into one embedding and pass it through an activation function. Thus, a node's updated feature embedding h' contains information about its own features and its neighbor features, which is achieved by multiplying $h*$ with the adjacency matrix. Multiplying with the adjacency matrix means that, for each node, we are only considering the feature of the neighbor nodes and the node itself. Figure 2.2 illustrates an example of the construction of an updated feature embedding for node 1.

1	1	1	0		h_1*									h'_1								
1	1	1	1	$\times$	h_2*								$=$	h'_2								
1	1	1	0		h_3*									h'_3								
0	1	0	1		h_4*									h'_4								

Figure 2.2: The figure above illustrates the construction of the final feature embedding for node 1. We can see from Figure 2.1 that, node 1 is connected with node 2, 3 and itself. So the updated final node embedding only contains information about these three nodes only.

The updated node embedding h'_i can be represented as:

$$h'_i = \sigma(\sum_{p \in neighbor(i)} h_p \times W)$$

In this way, GNN gives a higher dimensional feature representation for each node of a graph.

2.1.2 Graph Attention Network (GAT)

Graph attention network (GAT) [24] works similarly to GNN. However, the additional feature of GAT is that it also incorporates the importance of each neighbor node while constructing the final feature embedding. The importance of node j for node i, denoted as α_{ij}, is learned through an attention mechanism, which consists of the following steps:

(i) At first, the transformed feature vector h_i^* and h_j^* are concatenated with each other and passed through a linear layer of a neural network.

$$e_{ij}^* = W_l^T \times [h_i^* \| h_j^*]$$

where W_l represents the weight matrix from the linear layer.

(ii) The output of this layer passes through the Leaky ReLU [25, 26] activation function followed by the softmax [27] function, which gives the attention co-efficient value α_{ij}.

$$\alpha_{ij} = softmax(LeakyReLU(e_{ij}^*))$$

The attention coefficient measures the importance of a node's feature in embedding another node. The final node embedding h'_i of a node learned by GAT can be represented as:

$$h'_i = \sigma(\sum_{p \in neighbor(i)} \alpha_{ip} \times (h_p \times W))$$

2.1.3 Graph Convolutional Network (GCN)

Graph Convolution Network (GCN) [28] represents a node feature vector by aggregating the feature vectors of its neighbors and passing the aggregated feature value through a single layer neural network followed by an activation function. Figure 2.3 shows the steps of a GCN layer. There can be multiple layer GCNs stacked for a particular problem. In that case, the output embedding of each GCN layer works as the feature vector for the next layer, which goes through the same aggregation, linear transformation, and activation steps mentioned above. The number of GCN layers in a network represents the hops of nodes passing the feature information for

constructing a node embedding of a particular node. GCN transforms the features of a node in a compressed dimension which we can use as a representation of the node.

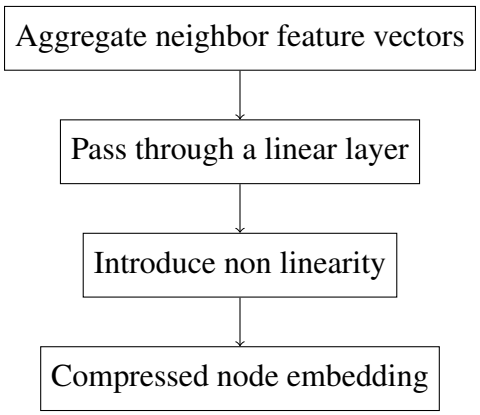


Figure 2.3: Graph Convolutional Network layer steps

2.1.4 Graph Autoencoder (GAE)

Graph Autoencoder (GAE) [17] utilizes the idea of an autoencoder and applies it to the graph domain. It consists of two different modules:

(i) Encoder module

(ii) Decoder module

Given the adjacency matrix, A of a graph, and the original feature vectors X, the encoder module learns each node's embedding or encoded representation Z_i by using GCN as the encoder. The embedding of all nodes can be represented as:

$$Z = GCN(A, X)$$

The decoder section then reconstructs the adjacency matrix from the compressed embeddings by using the inner product of the embeddings. A graph autoencoder aims to minimize the reconstruction loss between input and output, and learns an embedding of the node feature vectors in an unsupervised manner.

2.1.5 Source City and Target City

Source City For our transfer learning problem, we have to choose a data-rich city where crime data is available in abundance for transferring the learned knowledge. This city is called the **source city**.

Target City For the cross-city transfer learning problem, we have to choose a city with no available crime data to be the target city of our problem. This city is called **target city** or **destination city**. We assume that the target city is the data-scarce city.

2.1.6 Source Region and Target Region

Source Region We use the geographical or political boundaries of the source city to divide the city area into smaller parts, each of which can be considered a source region. If the source city is divided into s regions, then a source region can be expressed as S_i, where $i = 1, 2, \ldots, s$.

Source Region We use the geographical or political boundaries of the target city to divide the city area into smaller parts, each of which can be considered a target region. If the target city is divided into r regions, then a target region can be expressed as R_j, where $j = 1, 2, \ldots, r$.

2.1.7 Crime Category

We run our crime prediction model using the data of five different crime categories for comparison purposes, namely Theft, Robbery, Assault, Burglary, and Dangerous Drugs.

2.1.8 Crime Occurrences

For a particular crime category k, the number of crime events that have taken place in the target-city region R_j in the last T timesteps can be expressed as:

$$Y_{R_j,k} = Y^1_{R_j,k}, Y^2_{R_j,k}, ..., Y^T_{R_j,k}$$

Similarly, for a particular crime category k, the number of crime events that have taken place in the source-city region S_i in the last T timesteps can be expressed as:

$$Y_{S_i,k} = Y^1_{S_i,k}, Y^2_{S_i,k}, ..., Y^T_{S_i,k}$$

2.1.9 Timestep

We divided each day into six different timesteps or time intervals of 4 hours each. We call each of these intervals a timestep, denoted with t

2.1.10 External Features

To improve the prediction accuracy of our model, we decided to fine-tune it with relevant features that affect the number of crime occurrences. We call these non-crime additional relevant features *external features*. We can classify the external features into two categories.

(i) Static Features: Points of Interest data, Road Network data

(ii) Dynamic Features: Taxi Trip data

2.1.11 Road Network Representation

Given the road network data for both the source city and the target city, we represent the road network of a city region using a vector of size d.

Source City Road Network Representation We denote the representation of the road network for a city region S_i of the source city by $road_{S_i}$, where $road_{S_i} \in \mathbb{R}^d$. Thus the road network representation of all the 77 regions of the source city is denoted as

$$road_S = (road_{S_1}, road_{S_2}, ..., road_{S_{77}})$$

where $road_S \in \mathbb{R}^{77 \times d}$.

Target City Road Network Representation Similarly, we denote the representation of the road network for a city region R_j of the target city by $road_{R_j}$, where $road_{R_j} \in \mathbb{R}^d$. Thus the road network representation of all the 77 regions of the target city is denoted as

$$road_R = (road_{R_1}, road_{R_2}, ..., road_{R_{77}})$$

where $road_R \in \mathbb{R}^{77 \times d}$.

The road network representation learning method is described in detail in Chapter 4. These representations are static representations of the road networks of both the source city and the target city.

2.1.12 Points of Interest Representation

Given the points of interest (POI) data for both the source city and the target city, we represent the POI of a city region using a vector of size d.

Source City POI Representation We denote the representation of the POI for a city region S_i of the source city by POI_{S_i}, where $POI_{S_i} \in \mathbb{R}^d$. Thus the POI representation of all the 77 regions of the source city is denoted as

$$POI_S = (POI_{S_1}, POI_{S_2}, ..., POI_{S_{77}})$$

where $POI_S \in \mathbb{R}^{77 \times d}$.

Target City POI Representation Similarly, we denote the representation of the POI for a city region R_j of the target city by POI_{R_j}, where $POI_{R_j} \in \mathbb{R}^d$. Thus the POI representation of all the 77 regions of the target city is denoted as

$$POI_R = (POI_{R_1}, POI_{R_2}, ..., POI_{R_{77}})$$

where $POI_R \in \mathbb{R}^{77 \times d}$.

The POI representation learning method is described in detail in Chapter 4. These representations are static representations of the POIs of both the source city and the target city.

2.1.13 Taxi Inflow and Outflow

We consider taxi flow data of the source and the target city regions to be the dynamic data that changes with time. The inflow of the taxi flow for a region means the total number of taxis that have entered a particular region. Similarly, taxi outflow for a region represents the number of taxis that have left that region. Taxi inflow data of 7 days of a week for a target region R_j can be denoted as $inflow_{R_j}$, where

$$inflow_{R_j} = (inflow^1_{R_j}, inflow^2_{R_j}, ..., inflow^7_{R_j})$$

Similarly, taxi outflow data of 7 days of a week for a target region R_j can be expressed as $outflow_{R_j}$, where

$$outflow_{R_j} = (outflow^1_{R_j}, outflow^2_{R_j}, ..., outflow^7_{R_j})$$

For source region S_i, the respective terms are $inflow_{S_i}$ and $outflow_{S_i}$.

2.1.14 Structural Interaction

Structural Fingerprint [18] of each node is a cluster of contextual high-order neighbor nodes which serve as the local receptive field for a node. The domain of the effective receptive field is

adaptive and weighted. Graph structural details and node features are captured in the structural fingerprint for an improved attention layer.

To construct a structural fingerprint of a center node i, a local sub-graph spanning n-hop neighbors around i is considered. The local sub-graph is denoted as (V_i, E_i) where V_i is the set of neighborhood nodes and E_i is the set of edges connecting the neighborhood nodes to node i. Each node in sub-graph is given a non-negative weight, $w_i \in \mathbb{R}^{n_i * 1}$ where n_i is the number of nodes in V_i. The structural fingerprint of a node i is denoted as $F_i = (V_i, w_i)$.

Adjusting the weights, w_i to encode local graph structures is a learning task. These weights are used to find the structural interactions between two nodes in a graph by using weighted Jaccard Similarity. Learning the structural interaction is vital for combining information about the graph structure in feature embedding. The structural interaction vector for a source city region S_i can be represented as $structure_{S_i}$, similarly, the structural interaction vector for a target city region R_j can be represented as $structure_{R_j}$. This process is explained in detail in Section 4.2.4. This structural interaction value is added to the attention value learned based on the features. This summed attention value is then used to compute the final feature embeddings for all the regions of the source city and the target city.

2.1.15 Feature Embedding

For a target region R_j, given the road network representation $road_{R_j}$, the points of interest (POI) representation POI_{R_j}, taxi inflow data $inflow_{R_j}$, and taxi outflow data $outflow_{R_j}$, we construct a feature matrix $feature_{R_j}$ by concatenating the embedding of the static external features with the dynamic external feature as:

$$feature_{R_j} = road_{R_j} \| POI_{R_j} \| inflow_{R_j} \| outflow_{R_j}$$

Similarly, the feature matrix of source city region S_i can be expressed as:

$$feature_{S_i} = road_{S_i} \| POI_{S_i} \| inflow_{S_i} \| outflow_{S_i}$$

We can express the feature embedding of a particular region as a function of the structural interaction of that region and the feature matrix of that region. For the target city region R_j, the feature embedding F_{R_j} can be expressed as:

$$F_{R_j} = f(feature_{R_j}, structure_{R_j})$$

Similarly, the feature embedding of source city region S_i can be expressed as:

$$F_{S_i} = f(feature_{S_i}, structure_{S_i})$$

The feature embedding of all the regions of the source city can be expressed as F_S.

If each embedded feature vector of a region is of size h, then, $F_{R_j} \in \mathbb{R}^h$ and $F_{S_i} \in \mathbb{R}^h$. The particular function f for feature embedding learning is described in detail in Chapter 4.

2.1.16 Similar Regions

We calculate similarities between the feature embedding f_{R_j} of the target city region R_j, and all the 77 embedded feature vectors of the source-city regions, F_S, and choose the m most similar regions of the source city for the target-city region R_j. The m most similar regions for target region R_j are denoted as $I_{R_j} = (i_{1,R_j}, i_{2,R_j}, ..., i_{m,R_j})$, and the corresponding similarity scores can be shown as $a = a_1, a_2, ..., a_m$. We describe the similarity measurement method in detail in Chapter 4.

2.1.17 Crime Embedding

Three different types of crime data of a particular crime category are considered for creating the crime embedding of a particular source region S_i. They are:

(i) the crime data of a source city region S_i for a crime category k, $crime_{S_i,k}$

(ii) the crime data of the adjacent regions of S_i for a crime category k, $adjacent_{S_i,k}$

(iii) the crime data of the parent side of the source region S_i where the region lies for a crime category k, $parent_{S_i,k}$

For simplicity, we do not show the terms involving timestep t. We define *crime embedding* of a particular category for a source city region S_i as a function of these three types of crime data. The function used to create the embedding is called hGAT, which is explained in detail in Chapter 4. The crime embedding for the source city region S_i for crime category k can be expressed as:

$$C_{S_i,k} = hGAT(crime_{S_i,k}, adjacent_{S_i,k}, parent_{S_i,k})$$

where $i = 1, 2, ..., 77$, and $k = 1, 2, ..., 5$.

Given the m most similar source-city region I_{R_j} for a target region R_j, we can express the weighted crime embedding of a target city region R_j for a particular crime category k as:

$$\begin{aligned} C_{I_{R_j},k} &= a_1 \times C_{(i_{1,R_j}),k} + a_2 \times C_{(i_{2,R_j}),k} + ... + a_m \times C_{(i_{m,R_j}),k} \\ &= \sum_{p=1}^{m} a_p \times C_{(i_{p,R_j}),k} \end{aligned}$$

2.2 Problem Definition

Given the feature embedding F_{R_j} for the region R_j of the target city, and the crime embedding $C_{I_{R_j},k}$ for crime category k and the m most similar source-city regions I_{R_j} for the last T timesteps of the target city region R_j, we aim to predict $\hat{Y}_{R_j,k}^{T+1}$, the number of crime occurrences in timestep $T+1$ for region R_j and crime category k, for the target city, where $j = 1, 2, ..., 77$ and $k = 1, 2, ..., 5$.

Table 2.1 summarizes the notations used in this thesis and their meanings.

Table 2.1: Notations used in this thesis with their meanings

Notations	Meaning
S_i, R_j, t	Source city region i, target city region j, and length of time interval
m	Number of similar source regions considered for each target region
$Y_{S_i,k}$	Number of crime events of category k that have taken place in the source-city region S_i in the last T timesteps
$\hat{Y}_{R_j,k}^{T+1}$	Number of crime occurrences in timestep $T+1$ for target region R_j and crime category k
$road_{S_i}$, $road_{R_j}$	Road network representation of the source region S_i and the target region R_j
POI_{S_i}, POI_{R_j}	Points of interest representation of the source region S_i and the target region R_j
$inflow_{S_i}$, $inflow_{R_j}$	Taxi inflow data of a week for the source region S_i and the target region R_j
$outflow_{S_i}$, $outflow_{R_j}$	Taxi outflow data of a week for the source region S_i and the target region R_j
$structure_{S_i}$, $structure_{R_j}$	Structural interaction vector of the source region S_i and the target region R_j
F_{S_i}, F_{R_j}	Feature embedding of the source region S_i and the target region R_j
I_{R_j}	m most similar source regions for a target region R_j
$crime_{S_i,k}$, $adjacent_{S_i,k}$, $parent_{S_i,k}$	Crime data of the source region S_i, crime data of the adjacent regions of S_i, and crime data of the parent side of S_i for a crime category k
$C_{S_i,k}$	Crime embedding for the source city region S_i for crime category k
$C_{I_{R_j},k}$	Weighted crime embedding of m most similar source regions for a particular crime category k

Chapter 3

Literature Review

Although crime prediction has attracted the attention of researchers for decades as the successful prediction of the occurrence of crime is paramount for the safety of the city-dwellers, using transfer learning for this purpose is a relatively unexplored area of research. Most of this research focuses on historical crime data to predict if there is a possibility of a crime incident in a particular region at a particular time. Some research focuses on predicting the number of crime occurrences as well. Furthermore, learning an appropriate node embedding for the regions of a city requires the efficient application of the graph structure. So, we can classify the existing research related to our work into three broad areas:

(i) Research focusing on solving problems using transfer learning

(ii) Research focusing on the prediction of crime using historical crime data

(iii) Research focusing on learning the graph structure for a better node embedding

The research of these categories are described in Section 3.1, Section 3.2, and Section 3.3 respectively.

3.1 Transfer Learning Related Research

Transfer learning [29] refers to applying the knowledge obtained by solving one problem to another relevant problem, especially when the training data is unavailable or limited. Transfer learning has many applications, one of which is tackling the data scarcity problem of smart-city development. Researchers have been exploring transfer learning in solving multiple problems related to urban computing [30] and spatio-temporal prediction such as taxi flow prediction, air quality prediction, and water quality prediction.

3.1.1 Medium of Transfer

One of the significant challenges of transfer learning is finding a relevant source domain from which the transfer of learned knowledge occurs, and another is finding an appropriate medium of transfer (model, instance, feature) between the source and the target domain.

Model Based Transfer

In [14] the authors developed a transfer learning framework called RegionTrans which uses bike flow data from one city to predict crowd flow in another city. They used the Pearson coefficient to find the most similar source city region to the target city region. Again, in [31], the authors proposed a general framework for transfer learning-based applications and discussed three of their projects [14, 32, 33] as applications of urban transfer learning [31] in the fields of prediction, detection, and deployment, respectively, using the proposed general framework. However, none of these works focused on predicting crime. In [32], transfer learning is used to detect ride sharing cars in a pool of vehicles by using taxi as positive instance and bus as negative instance. The proposed model, CoTrans, applies a model-based transfer learning [34] approach by training a Random Forest model on taxi and bus data to detect ride-sharing cars and focuses on utilizing shared features of similar entities for transferring knowledge rather than the similarity between regions.

Feature Based Transfer

Another approach to transfer learning is feature-based transfer learning [34], where a beneficial feature representation from one domain is applied in solving a related problem in another domain. One such application of this approach is [35] where the authors aimed to predict the thermal comfort of the residents in a building by using indoor and outdoor environmental data and human features like age and gender in similar climate-zones. The authors of [20] transferred driving trajectory-based routing knowledge between similar region pairs when there is no existing historical trajectory data in a particular region. The similarity of regions is calculated by using the Jaccard similarity function.

3.1.2 Other Applications

Predicting human mobility is a common application of transfer learning [15, 36]. In [15], the authors transferred human mobility knowledge by using points of interest embeddings of a region. In contrast, a new variety of transferring knowledge, concept matching, is explored in [36]. Activities like the intention behind movement, choosing a destination, and selecting a

path based on several criteria to reach that destination are more or less similar in all the cities. The authors of this paper exploited this similarity to predict human mobility in a new city with insufficient data from cities with abundant human mobility data.

Although transfer learning deals with the problem of data scarcity, most of the existing research uses data from one single source domain or most similar source region while transferring the knowledge [14, 20]. This method of transferring knowledge from one region or one domain is susceptible to negative transfer that can significantly reduce model performance. However, transferring knowledge from multiple regions can stabilize knowledge transfer and incorporate different relevant features from different regions. [37] explores some standard methods for assimilating data from multiple domains. In [16] the authors used data from multiple cities for traffic prediction and water quality prediction. The authors of [38] proposed a framework named FLORAL for air quality prediction in one city by using data from multiple modalities of another city.

A limited number of existing research focuses on crime prediction using transfer learning. The authors of [39] used temporal and historical crime data, spatial features, demographic and socioeconomic features, and dynamic features for crime prediction as different demographic and socioeconomic features affect the occurrences of crime in a region [40–42]. The authors also applied transfer learning by training the model on multimodal data of multiple source cities and testing model performance on data of a target city. In [43] the authors explored transfer learning for crime prediction and used relevant data from multiple modalities for this purpose. However, the authors proposed training a model in one of the disjoint 2×2 regions of NYC and using the learned parameters to predict the number of crimes in other regions. This work does not utilize the effects of geographical neighborhood regions on crime, nor does it propose any similar region mapping approach for the prediction purpose and applies parameters learned from one region to all the other regions. Thus other existing research on crime prediction has shown better performance.

3.2 Crime Prediction Related Research

Most of the existing research that targets crime prediction uses historical crime data. In [4] the authors proposed CrimeForecaster, a deep learning model, to simultaneously process spatial and temporal information to predict the probability of a crime event in a region at a particular time. To capture the spatial correlations of regions, the authors used neighborhood regions as nodes of an undirected weighted graph and used it in Gated Recurrent Unit (GRU) [44] networks for the concurrent processing of spatial and temporal data. Although CrimeForecaster outperformed existing state-of-the-art crime prediction models, it did not consider external relevant features for predicting crime and did not address the data insufficiency problem. In [3] the authors

argued that static features like historical data, geographic data, and demographic data could not incorporate the short-term variations in relevant features when a crime event occurs. For this reason, they suggested using dynamic features like region popularity and visitor count in addition to static features for predicting crime and showed that including dynamic features gives a better AUC score.

Crime prediction is a complex research area because of the interdependence of multiple factors. In [6], the authors considered the effect of time in crime prediction, along with dependency between different crime types and interrelation of crime data with points of interest and urban anomaly data. Similarly, the authors of MiST [5] only utilized the historical urban anomaly and crime data to capture the inter-region, intra-region, and cross-categorical dependency among crime types. MiST used LSTM and attention-mechanism [21] for integrating these views and predicting the probability of anomalies of each kind in a specific region at a future time slot. However, external relevant features like geographic and demographic information were not considered. In addition, MiST could not capture the geographical neighborhood region effects as the authors of these papers considered grid-based area division to construct a region.

In [8] the authors proposed an attention-based interpretable crime prediction framework named AIST. AIST uses neighbor region crime data, side region crime data, a region's historical crime data, and additional features such as taxi trip data and points of interest data to predict the number of crime occurrences of various categories. However, AIST does not apply transfer learning, nor does it consider structural information of a particular region. Most studies concentrate on the effects of geographically adjacent regions for crime prediction but do not consider the effect of regions that might be non-adjacent but have similar features or crime patterns. In [7] the authors proposed a homophily aware or node characteristic aware crime forecasting framework called HAGEN which utilizes the idea that graph neural networks (GNN) [23] seem to perform better when nodes with similar characteristics are linked with each other [45, 46]. HAGEN can capture inter-region and intra-region crime dependency and the temporal dependency in predicting crime and outperform the state-of-the-art crime prediction frameworks. However, our work is different from HAGEN as:

(i) HAGEN uses a homophily aware region graph. In contrast, we use a geographical adjacency-based region graph.

(ii) HAGEN uses historical crime data of a region for region embedding, whereas our work focuses on crime prediction when there is no historical crime data available in the target region.

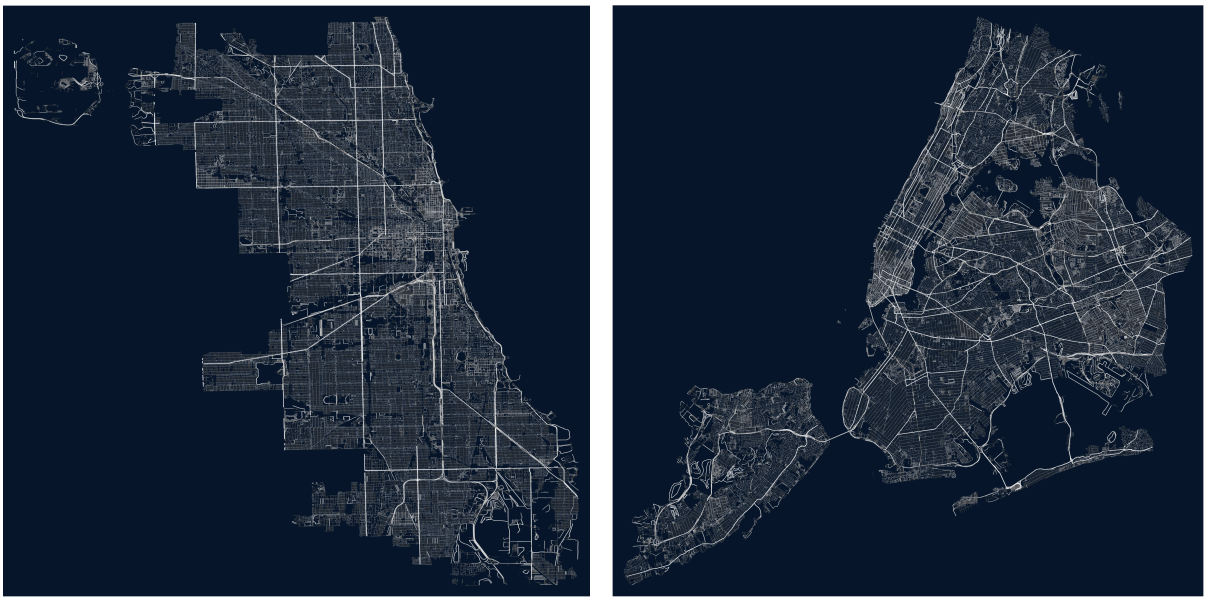
3.3 Learning Graph Structure

As regions of a city are comparable to nodes of a graph, learning appropriate node embedding can represent the most useful features of a region. Existing graph frameworks like Graph Neural Networks (GNN) [23], Graph Attention networks (GAT) [24], Graph Convolutional Networks (GCN) [28] can not represent the structural information of a graph. In addition, GAT only uses immediate adjacent neighbor nodes while learning node embeddings and assigning attention values. A recent study [18] has proposed a graph representation learning framework called ADSF that recognizes the effects of structural information of a node for learning the embedding of a particular node which results in a better representation than considering only regular feature based embedding.

Chapter 4

Methodology

This chapter gives an overview of our proposed framework and describes each of the units of our model. We choose **Chicago** to be the source city and **New York City** to be the target city for our particular problem. We refer to New York City as **NYC** in the rest of the thesis. Figure 4.1 shows the illustration of the two cities. According to the official website of the City of Chicago [47], Chicago has 77 community areas. We consider each of these 77 community areas as a region of the source city. We denote each source city region as S_i where i = 1,2,....,77. Similarly, according to the official website of the New York City Police Department [48], NYC has 77 police precincts. We consider each of these 77 police precinct areas as a region of the target city. We denote each target city region as R_j where j = 1,2,....,77.



(a) The City of Chicago (b) The City of New York

Figure 4.1: Source city Chicago and target city NYC

4.1 Overview

Figure 4.2 gives an overview of our proposed framework. The four main components of our model are: i) Region-representation learning unit, ii) Region-similarity learning unit, iii) Crime embedding learning unit, and iv) Crime prediction unit.

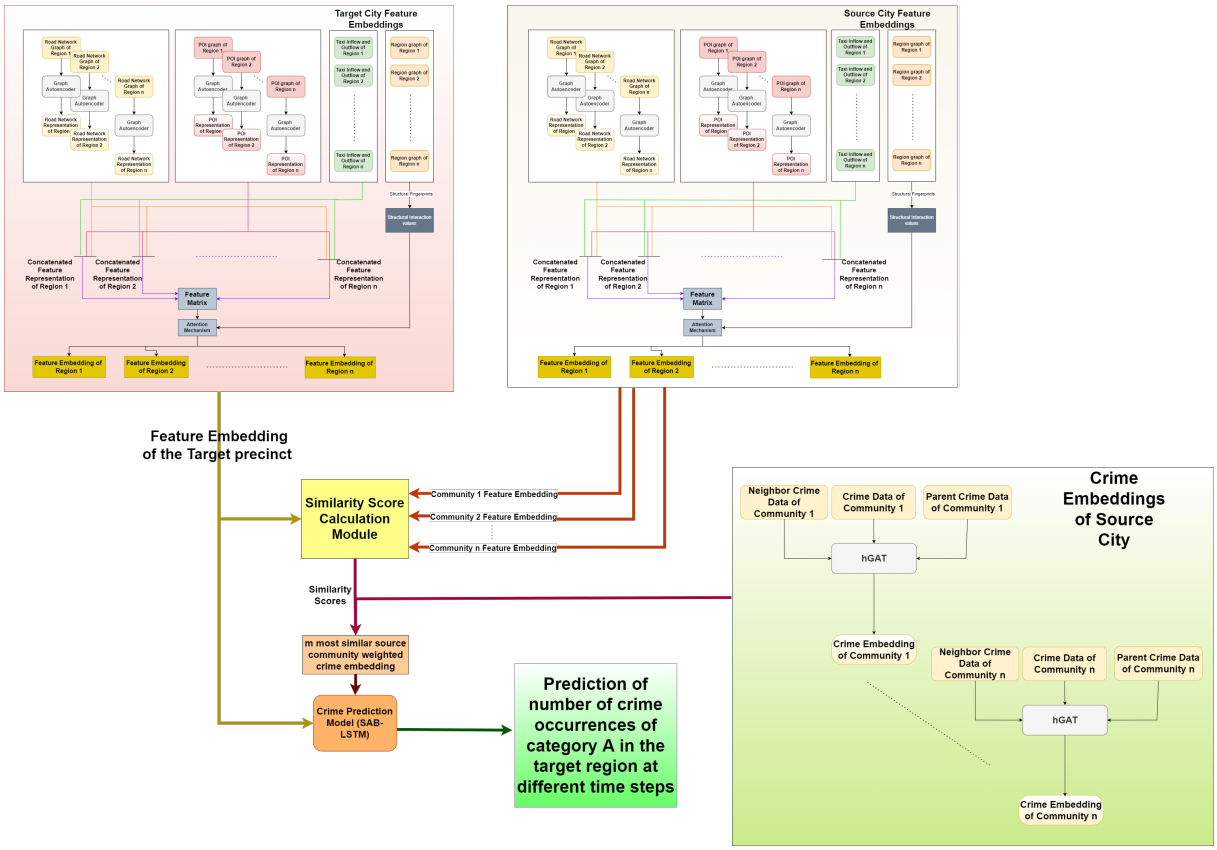


Figure 4.2: Overview of model architecture

The rest of the chapter describes each component of our proposed framework. Section 4.2 discusses the feature embedding learning module, and after that, Section 4.3 describes the region similarity learning component. We discuss the crime embedding learning unit in Section 4.4 and finally, in Section 4.5, we give a description of our crime prediction unit.

4.2 Feature Embedding Learning Unit

We learn the representation of a region by using only the relevant external features for crime prediction. We choose road network data, points of interest data, and taxi flow data to represent each region in terms of these features. We feed the source and target feature embeddings learned from this module to the region similarity learning unit for finding the most similar source regions for each target region. The overview of this unit is shown in Figure 4.6.

4.2.1 Road Network Representation

We collect the street network data, including the roads with private access, for both the source and the target cities using OSMnx [49]. The road network graphs returned by OSMnx consist of both intersections and dead-ends as graph nodes. So, for each node, we construct a feature vector consisting of the information on whether a node is an intersection or not. After that, from the latitude and longitude of the nodes of the graph, we then map each road network graph node to a region and construct road network graphs for the source and the target regions. Two such nodes are connected if the corresponding road junctions or dead-ends are connected through a road in real life. Figure 4.4 show an example of the road network graphs for both NYC and Chicago. The road network representation learning steps are shown in Figure 4.3.

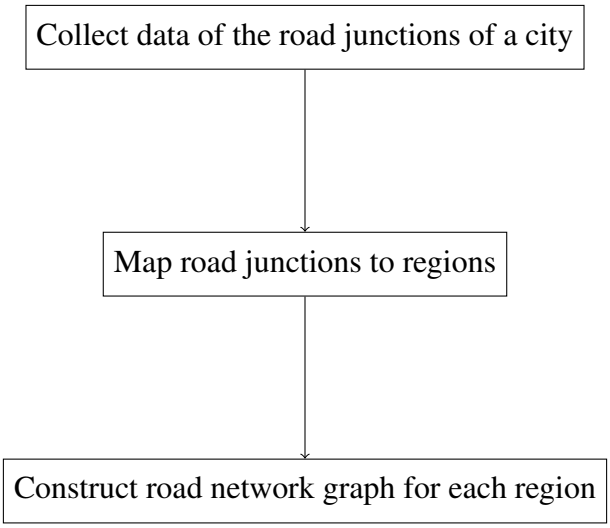
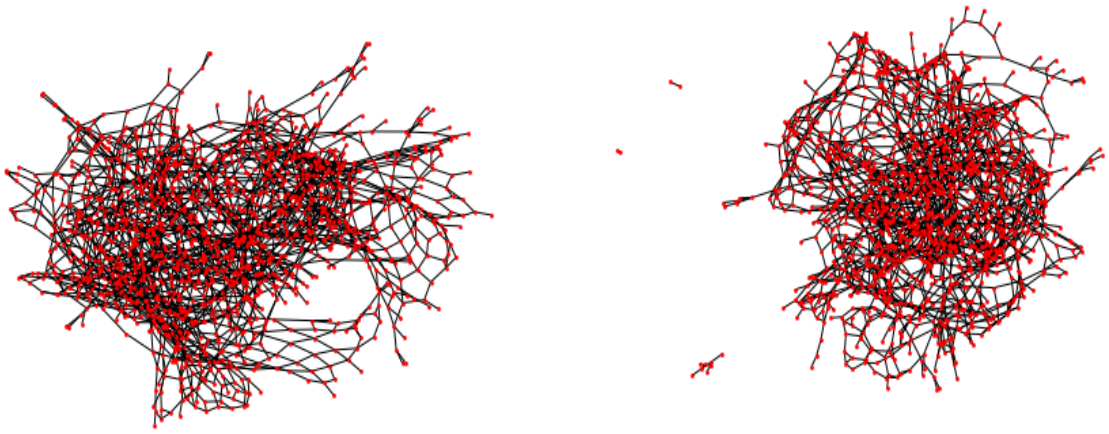


Figure 4.3: Construction steps of a road network graph

After constructing the road network graph for both the source regions and target regions, we then pass this graph through a graph autoencoder (GAE) to learn a representation of this network from the graph. GAE consists of an encoder and a decoder module. The encoder module, usually a graph convolutional network (GCN), learns an embedding of the road network graph given the adjacency matrix of the road network graph and the feature vectors of each node. The decoder module tries to reconstruct the original adjacency matrix from the given representation. This road network representation gets updated in each training step to minimize the reconstruction loss. For the target region R_j the adjacency matrix can be denoted as $A^{road}_{R_j}$, the feature vector can be denoted as $X^{road}_{R_j}$, and the embedding learned by the encoder can be denoted as $Z^{road}_{R_j}$ or $road_{R_j}$.

$$Z^{road}_{R_j} = GCN(A^{road}_{R_j}, X^{road}_{R_j})$$



(a) Road network graph for precinct 10 of NYC

(b) Road network graph of community 11, Jefferson Park, Chicago

Figure 4.4: Road network graphs for the source city and the target city

4.2.2 Points of Interest Representation

The points of interest (POI) graph is constructed from the POI data collected for the source [50] and the target cities [51]. We map each POI node to a region and connect them with edges. Then by using a GAE, as mentioned in Section 4.2.1, we get a POI embedding of each region of both the source and the target cities. For the target region R_j the adjacency matrix can be denoted as $A^{poi}_{R_j}$, the feature vector can be denoted as $X^{poi}_{R_j}$, and the embedding learned by the encoder can be denoted as $Z^{poi}_{R_j}$ or poi_{R_j}.

$$Z^{poi}_{R_j} = GCN(A^{poi}_{R_j}, X^{poi}_{R_j})$$

4.2.3 Constructing the Feature Matrix

We collect the taxi data for both the source city [52] and the target city [53]. After necessary preprocessing steps, we divide the taxi flow data into inflow and outflow and convert the inflow and the outflow data into weekly data for each region. For a target region R_j the inflow data can be represented as $inflow_{R_j}$ and the outflow data can be denoted as $outflow_{R_j}$. We concatenate the road network representation, points of interest representation, and taxi inflow outflow data and construct a feature matrix for each region. For the target region R_j, this can be denoted as $feature_{R_j}$.

$$feature_{R_j} = [road_{R_j} \| poi_{R_j} \| inflow_{R_j} \| outflow_{R_j}]$$

4.2.4 Finding the Structural Fingerprint of a Region

After constructing the feature matrix for both the source and target city region, we construct a structural fingerprint for each region to incorporate the spatial effects of crime. We construct a region graph by considering each region as nodes, and an edge connects two such nodes if they share a common boundary.

If the number of n-hop neighbors around a particular target region R_j is denoted as n_{R_j}, the structural fingerprint of a target region is constructed by giving weights, $w_{R_j} \in \mathbb{R}^{n_{R_j} \times 1}$, to the n-hop neighbor regions around that region, using Random Walk with Restart (RWR) [19] to learn the node weights.

Suppose, for a target region R_j, each iteration starts from the center node R_j, randomly walking to its neighboring nodes in N_{R_j} where the number of neighbor nodes in n-hop distance is n_{R_j}, with the weight of the node acting as the probability of choosing the node. Each step is defined by:

$$w_{R_j}^{t+1} = c.A_{R_j} w_{R_j}^{t} + (1-c).e_{R_j}$$

where A_{R_j} is the normalized adjacency matrix of center node R_j, c is the decaying parameter in range of $[0, 1]$ that controls the option of restart or random walk and $e_{R_j} \in \mathbb{R}^{n_{R_j} \times 1}$ is a vector where only the entry corresponding to node R_j is 1.

These weights are then used to compute the structural interaction $structure_{R_i,R_j}$ between the structural fingerprint of node R_i and R_j.

$$structure_{R_i,R_j} = s_{R_i,R_j} = \frac{\sum_{R_p \in (V_{R_i} \cup V_{R_j})} \min(w_{R_i R_p}, w_{R_j R_p})}{\sum_{R_p \in (V_{R_i} \cup V_{R_j})} \max(w_{R_i R_p}, w_{R_j R_p})}$$

where w_{R_i}, w_{R_j} are n-hop neighbor node weights for R_i and R_j and V_{R_i}, V_{R_j} are sets of neighborhood nodes of the corresponding structural fingerprints of R_i and R_j.

4.2.5 Feature Embedding of a Region

Given the structural interaction vector for a region $structure_{R_j}$ the feature vector of that region $feature_{R_j}$, the feature embedding is learned by using a modified version of the graph attention network (GAT). Instead of using only the attention values based on the features of the regions, we use the structural interaction values of the neighbors to incorporate the structural information of a node. We sum these two attention values to get the final attention value used to learn the feature embedding of a region.

Given the structural interaction values and the feature matrix of a city, we can think of the rest of the feature embedding learning procedure as the regular working procedure of a GAT. At first,

we pass our feature matrix through a linear layer to get a transformed representation $feature^*$. This process is illustrated in Figure 4.5.

$$feature^* = feature \times W$$

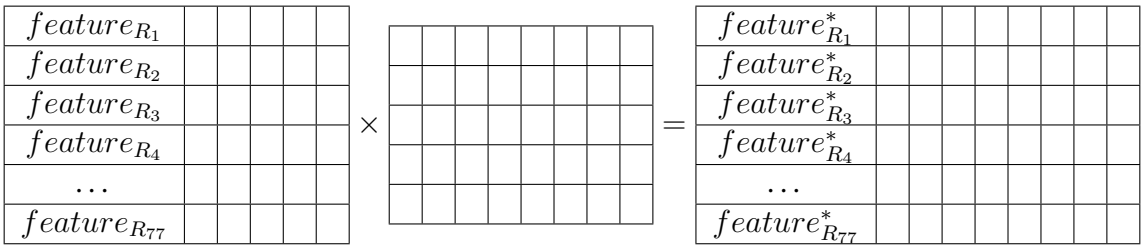


Figure 4.5: The figure shows the transformation of target city node feature vectors of size d to size d'. The number of regions for the target city is $n = 77$. We multiply the feature matrix $feature \in \mathbb{R}^{n \times d}$ with a learnable weight matrix $W \in \mathbb{R}^{d \times d'}$ and get a transformed feature matrix $feature^* \in \mathbb{R}^{n \times d'}$. Here $d = 5$ and $d' = 8$.

Then we pass the transformed feature matrix through a GAT layer. But instead of using the generated attention only, we add the structural interaction value of the neighbors to the attention value. This step ensures that we combine the weighted values of the features of the structurally strongly connected regions in the new feature embedding of a target region. Furthermore, reducing the effect of the structurally weakly connected regions. The intuition is that GAT only calculates attention values for the immediate neighbors based on the feature vectors and combines weighted features of the immediate neighbors to find the desired embedding. On the other hand, we combine the weighted features of the neighbors along with their structural interaction values. The formula for the feature embedding of the target region R_j can be represented as:

$$F_{R_j} = \sigma\Big(\sum_{R_p \in V_{R_j}} (structure_{R_j,R_p} + \alpha_{R_j,R_p}) \times (feature_{R_p} \times W)\Big)$$

4.3 Similarity Measurement Learning Unit

After learning the feature embeddings of the source and the target city regions, we aim to find the m most similar source city regions for the target region for transfer learning purposes. For this reason, we experiment with two different techniques for measuring the similarity score between regions: graph attention network (GAT) based similarity learning function and Pearson Coefficient based similarity learning function. Section 4.3.1 and Section 4.3.2 describe these two methods respectively.

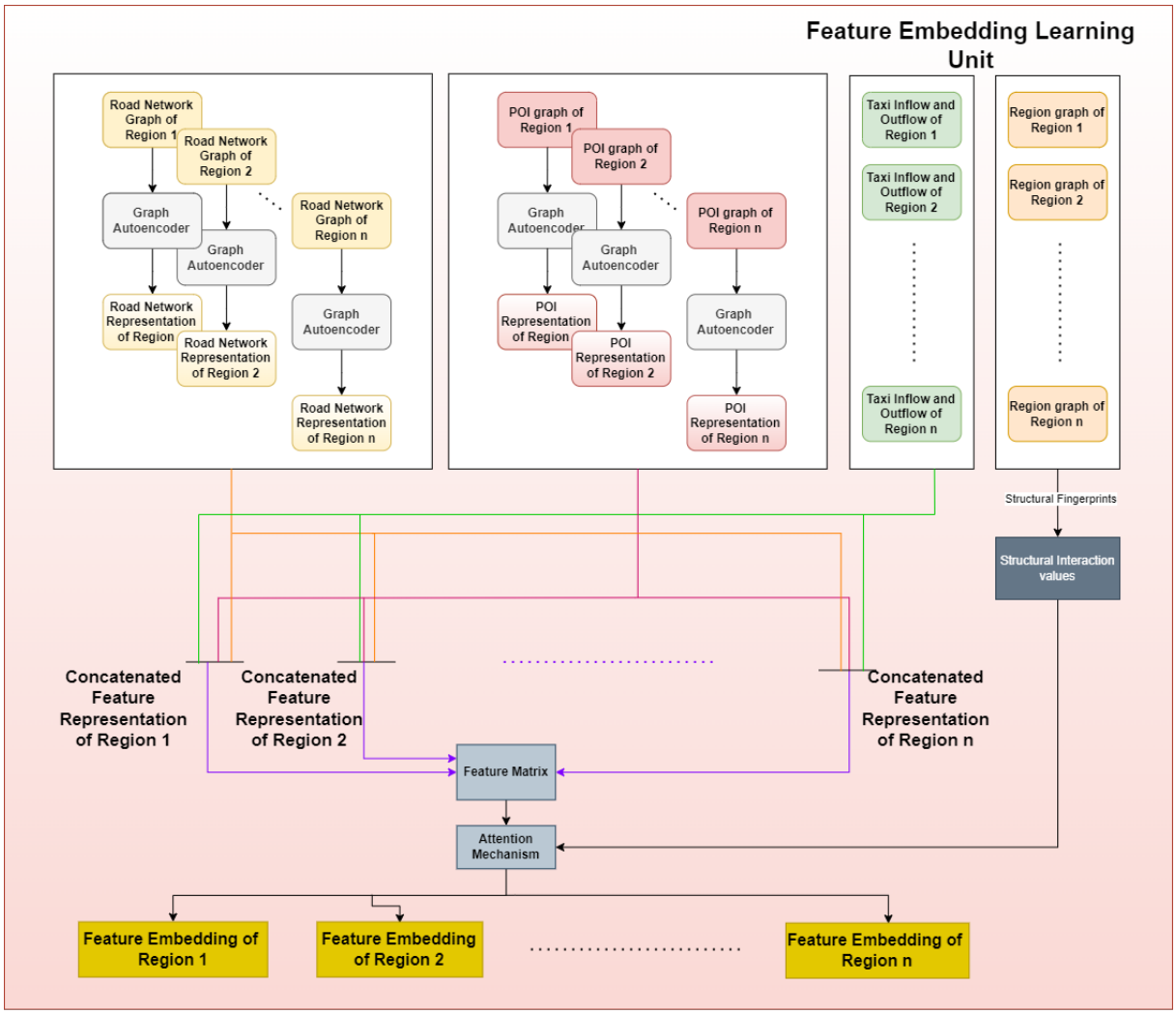


Figure 4.6: Feature embedding learning unit

4.3.1 Graph Attention Network Based Similarity Learning Module

We can formulate the similarity score learning technique as a graph attention learning problem. We can think of a graph having $n + 1$ nodes, where n represents the number of regions in the source city. n of the nodes represent the n regions of the source city. The remaining node represents the target region for which we want to find the m most similar source regions. This graph can be represented as $G'(V, E, F)$ where V is the set of all $n + 1$ vertices, F is the corresponding node feature embeddings, and E is the set of edges. The graph is shown in Figure 4.7.

The remainder of this module is implemented by feeding this graph through a graph attention network. For the target region R_j, the GAT gives an attention value for each of the neighbor regions (in this case, all the regions of the source city).

$$a = GAT(G') = (a_{R_j,S_1}, a_{R_j,S_{1=2}}, \ldots, a_{R_j,S_{77}})$$

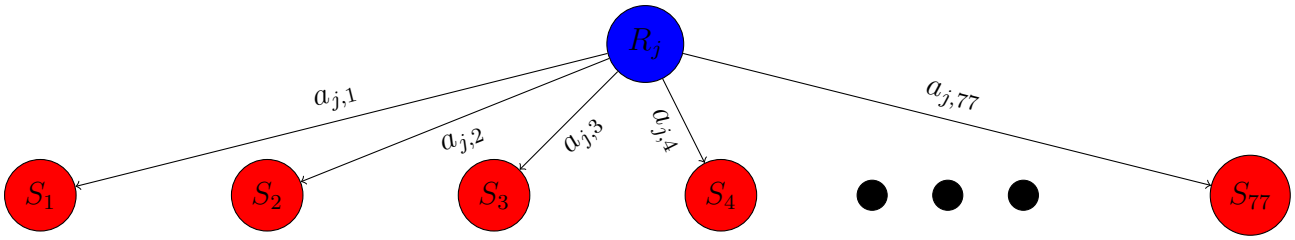


Figure 4.7: Construction of a graph for similarity score learning

These attention values signify the importance of each neighbor node in the features of the target region node. We use these attention scores as the similarity scores. We assume that the higher the attention value, the more similar a source, and a target region are. We find the m most similar regions by taking the m regions whose attention values are highest. For target region R_j, this can be represented as:

$$I_{R_j} = (\underset{i}{\text{argmax}} \forall_{S_i} a_{R_j,S_i})[:m]$$

where a_{R_j,S_i} is the attention value of source region S_i on region R_j. The complete steps of this process is shown in Figure 4.8 using simpler notations for the attention values.

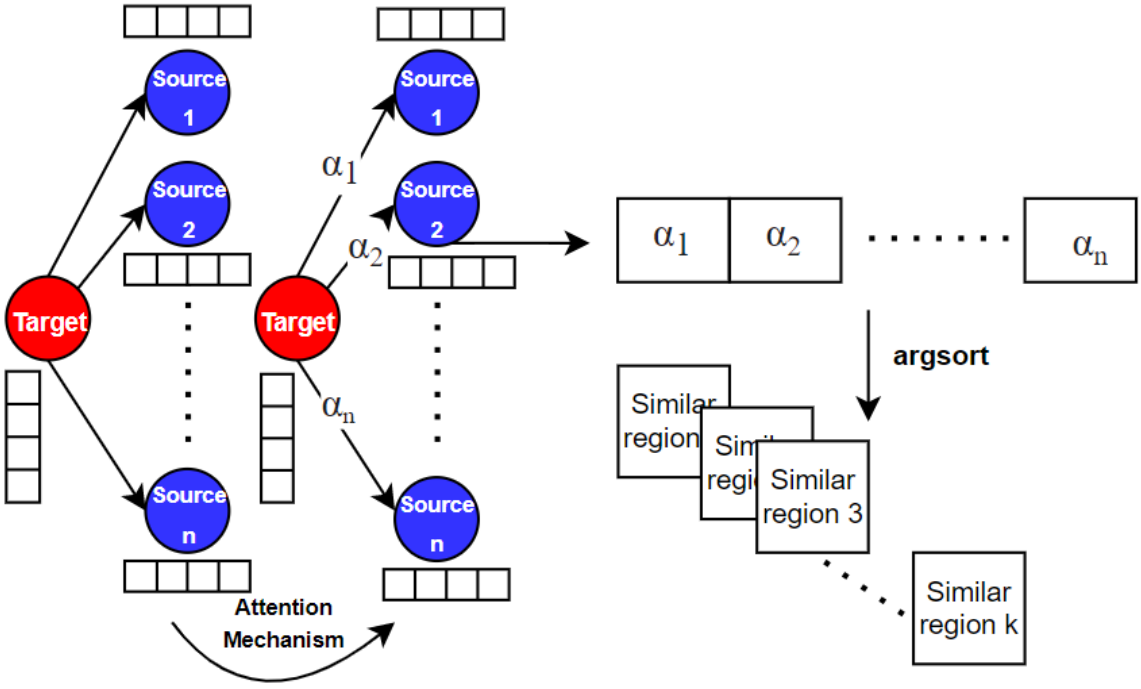


Figure 4.8: Attention based similarity

4.3.2 Pearson Coefficient Based Similarity Learning Module

The second method of learning the similarity between two regions uses the Pearson Correlation Coefficient to find the correlation between the source region and the target region feature embeddings. The overview of this method is shown in Figure 4.9 using simpler notations for the correlation coefficients. Pearson Correlation computes the degree of linear correlation between two vectors X and Y. The value of this coefficient is usually in $[-1, 1]$ where a value close to 1 indicates a strong positive correlation, a value close to -1 indicates a negative correlation, and a correlation value 0 signifies that there is no correlation between the vectors in question. The formula of this correlation coefficient is as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

We denote the source region feature embedding as F_{S_i} and the target region feature embedding as F_{R_j}. The formula mentioned above can be applied to find the linear correlation between the two embeddings as:

$$r_{R_j,S_i} = \frac{\sum_{k=1}^{d}(F_{R_j}^k - \bar{F_{R_j}})(F_{S_i}^k - \bar{F_{S_i}^k})}{\sqrt{\sum_{k=1}^{d}(F_{R_j}^k - \bar{F_{R_j}})^2}\sqrt{\sum_{k=1}^{d}(F_{S_i}^k - \bar{F_{S_i}})^2}}$$

where d is the dimension of the feature embeddings, and $\bar{F_{R_j}}$ and $\bar{F_{S_i}}$ represent the mean value of the feature embedding of the target city and source city respectively .

For a target region, we compute this for all the other source regions and choose m regions with the highest correlation values as the m most similar source regions for each target region. The correlation values can be expressed as:

$$r = (r_{R_j,S_1}, r_{R_j,S_{1=2}}, \ldots, r_{R_j,S_{77}})$$

For a target region R_j the m most similar source regions can be represented as:

$$I_{R_j} = (\underset{i}{\operatorname{argmax}} \forall_{S_i} r_{R_j,S_i})[:m]$$

4.4 Crime Embedding Learning Unit

In the next step, after learning the feature embeddings of the source city and the target city regions and calculating the similarity scores, we focus on learning the crime embedding of the source city regions. We calculate the crime embedding for each region of the source city

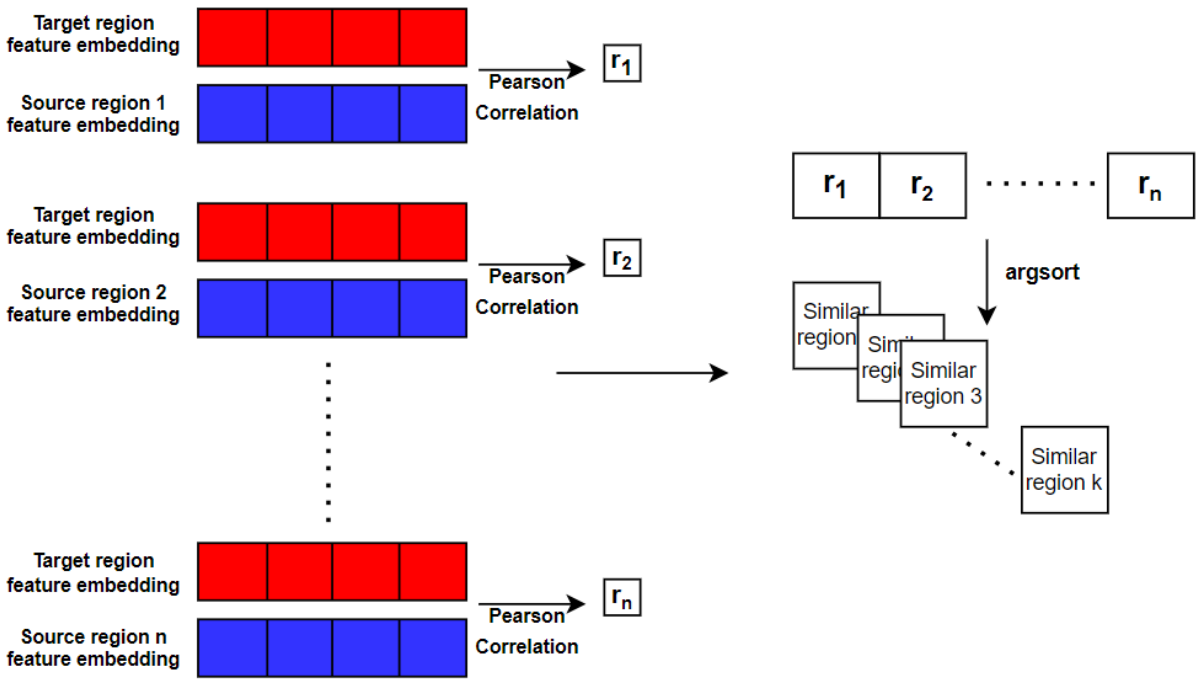


Figure 4.9: Pearson correlation based similarity

offline and save the embeddings for later use. We assume our source city has abundant crime data. Thus, transferring the similar region crime embeddings from the source city can work as a replacement for the target city crime data and assist the target region crime prediction model in performing better.

We use hGAT, a variant of GAT, to learn the crime embedding of the source regions. In addition to considering the crime data of the neighbor regions while constructing the crime embedding of a region, hGAT takes into account the crime data of those regions that fall on the same side of the city as the region whose crime embedding is constructed. Section 4.4.1 describes this module in detail and Figure 4.11 illustrates the overview of this unit.

4.4.1 hGAT

Figure 4.10 shows an overview of this framework. hGAT works similarly to a graph attention network. We calculate the attention values for a region in two steps: Firstly, we use the crime data of the source city region S_i, the region for which we want to calculate the attention values of other regions, and its one-hop neighbor $S_{i'}$. The crime data for category k in timestep t in the region S_i can be represented as $crime^t_{S_i,k}$. For simplicity, we can omit the symbol t and express it as $crime_{S_i,k}$. Similarly, the crime data for region $S_{i'}$ can be expressed as $crime_{S_{i'},k}$. We transform the crime data of these regions by multiplying them with a weight matrix.

$$crime^*_{S_i,k} = w_1 \times crime_{S_i,k}$$

$$crime^*_{S_{i'},k} = w_1 \times crime_{S_{i'},k}$$

Then we concatenate these two representations and pass it through another linear neural network layer. We introduce non linearity to the output of this linear layer and get the unnormalized attention value learned for region S_i and $S_{i'}$ represented as $e^1_{S_i,S_{i'}}$.

$$e^1_{S_i,S_{i'}} = LeakyReLU(w_2 \times [crime^*_{S_i,k} \| crime^*_{S_{i'},k}])$$

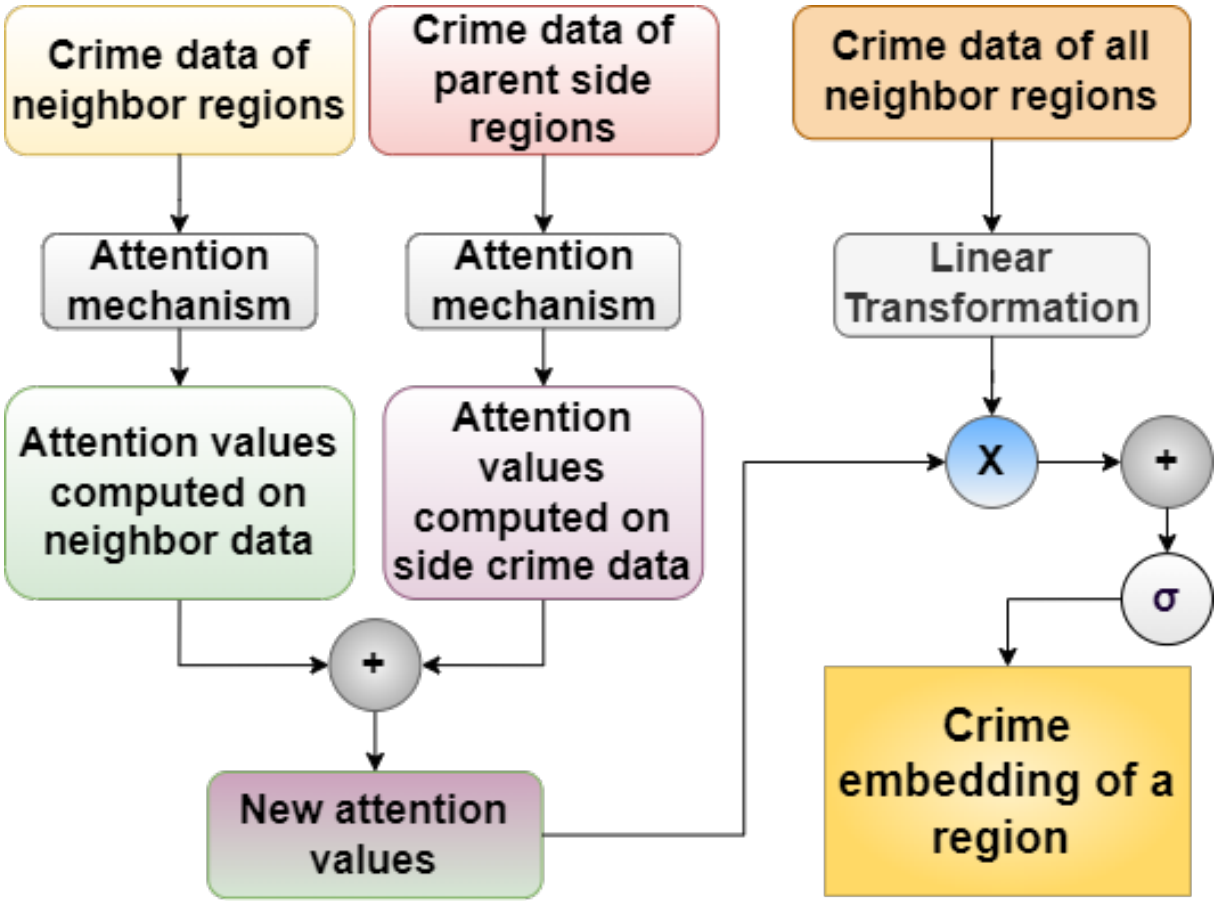


Figure 4.10: hGAT

In the second step, we utilize the crime data of all the regions that lie on the same side of the city as S_i. We compute it by adding the crime data of all such regions and constructing a parent crime vector for category k, $parent_{S_i,k}$.

$$parent_{S_i,k} = \sum_{S_p \in side_{S_i}} crime_{S_p,k}$$

Then like the steps mentioned previously, we transform the parent crime data of region S_i and region $S_{i'}$ by multiplying them with a weight matrix.

$$parent^*_{S_i,k} = w_3 \times parent_{S_i,k}$$

$$parent^*_{S_{i'},k} = w_3 \times parent_{S_{i'},k}$$

Then we concatenate these two representations and pass it through another linear neural network layer. We introduce non linearity to the output of this linear layer and get the un-normalized attention value learned for region S_i and $S_{i'}$ represented as $e^2_{S_i,S_{i'}}$.

$$e^2_{S_i,S_{i'}} = LeakyReLU(w_4 \times [parent^*_{S_i,k} \| parent^*_{S_{i'},k}])$$

After that, we add these two attention values and normalize the result to get the final attention value.

$$\alpha_{S_i,S_{i'}} = softmax(e^1_{S_i,S_{i'}} + e^2_{S_i,S_{i'}})$$

4.4.2 Final Crime Embedding of a Region

The final crime embedding of a source city region is learned by using a graph attention layer where we learn the attention values from Section 4.4.1. The formula can be represented as:

$$C_{S_i,k} = \sigma(\sum_{S_{i'} \in neighbor(S_i)} (\alpha_{S_i,S_{i'}} \times w_1 \times crime_{S_{i'},k}))$$

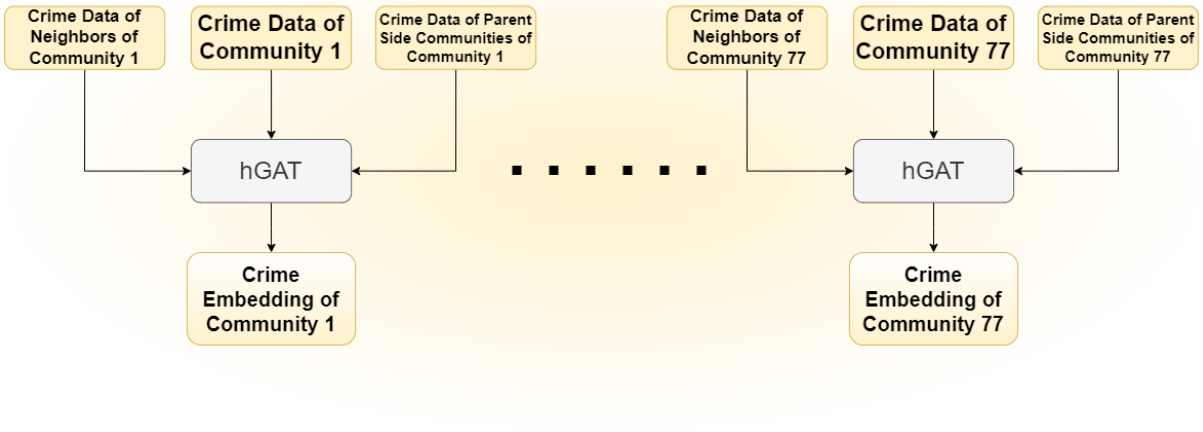


Figure 4.11: Crime embedding learning unit

4.5 Crime Prediction Unit

After learning the feature embeddings of both the source city and the target city regions, and the crime embeddings of the source city region, we advance to the crime prediction unit. The steps of this unit are shown in Figure 4.12.

At first, we construct a weighted crime embedding where we learn the weights by the similarity score learning methods mentioned in Section 4.3. We take the crime embedding of the m most similar source regions $\in I_{R_j}$ and multiply them with the corresponding similarity score to get the weighted crime embedding for category k.

$$C_{I_{R_j},k} = \sum_{p \in I_{R_j}} (a_p \times C_{p,k})$$

For comprehensibility, we omit the timestep related terms. We can assume that, $C_{I_{R_j},k}$ is the weighted crime embedding of all T previous timesteps. We then concatenate the feature embedding of the target region F_{R_j} and the weighted crime embedding $C_{I_{R_j},k}$, and pass it through a SAB-LSTM layer and get a hidden layer representation for timestep $T+1$.

$$h_{R_j}^{T+1} = SAB - LSTM([F_{R_j} \| C_{I_{R_j},k}])$$

We then pass the output of the SAB-LSTM through a linear layer with one output node which predicts the number of crime occurrences. We then apply non-linearity in the output of this linear layer and get the prediction for timestep $T+1$.

$$\hat{Y}_{R_j,k}^{T+1} = \tanh\left(w \times h_{R_j}^{T+1}\right)$$

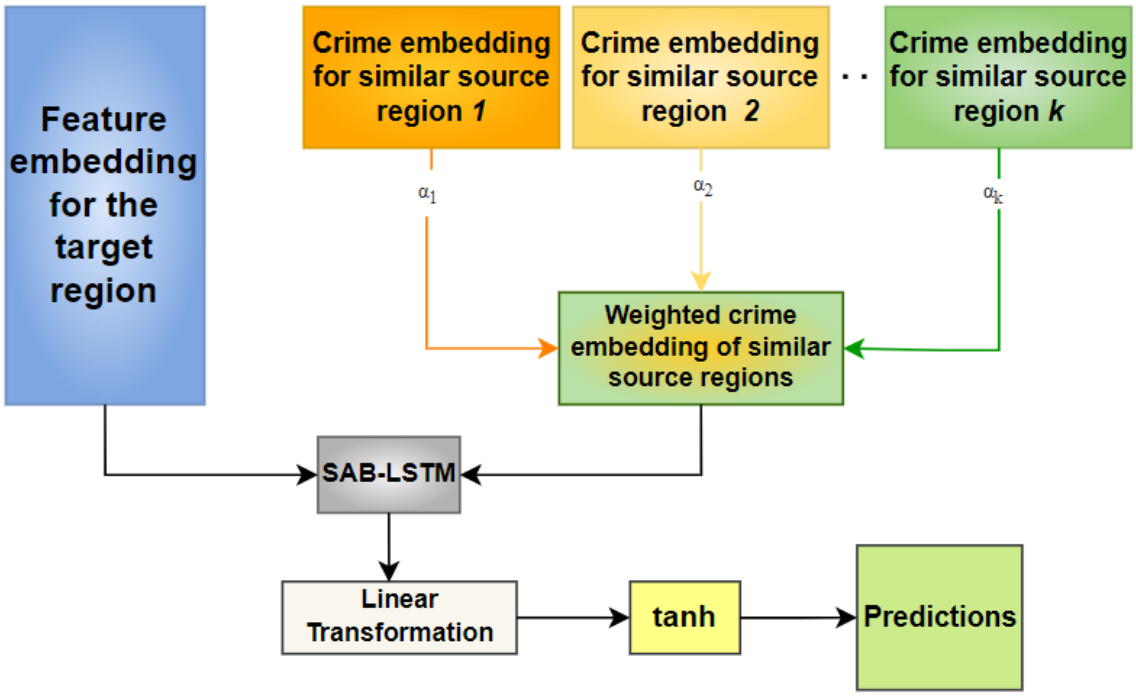


Figure 4.12: Crime prediction unit

Chapter 5

Experiments

This chapter describes the datasets used in our thesis and discusses the performance comparison between different models for comparison purposes. We show our model's superior performance in crime prediction using transfer learning and describe the outcome of using the similarity learning techniques mentioned in Section 4.3. Finally, we show the effects of changing the number of similar regions in our model. We organize the chapter in the following way: Section 5.1 describes the datasets, Section 5.2 gives details of the model parameters and the preprocessing steps that we follow. After that, Section 5.3 explains the evaluation metric used in our thesis. Section 5.4 shows the performance comparison between different models. Finally, Section 5.5 compares the performance of the two similarity learning methods, and Section 5.6 explains the effects of varying the number of similar regions in our model.

5.1 Datasets

For the source city, Chicago, to learn the crime and the feature embeddings of the regions, we have collected crime data [54], road network data using OSMnx, points of interest (poi) data [50], and taxi flow data [52]. Similarly, we collected this information for the target city NYC from the publicly available datasets [48, 51, 53] and OSMnx. Table 5.1 gives a detail explanation on the datasets. The train-test ratio while training the source city model is $67 : 30$ following [8]. We do not use any crime data from the target city while training the transfer learning model. However, we test the transfer learning model using the crime data from NYC.

Table 5.1: Datasets

Type of data	Details
Chicago crime data (2019) and NYC crime data (2019)	We collect crime data for five different crime categories, namely Theft, Robbery, Assault, Burglary, and Narcotics, from 01 January 2019 to 31 December 2019 for both the source city and the target city. We divide the data into 6 timesteps for each day, each time step consisting of 4 hours each starting from 12am everyday.
Chicago POI data and NYC POI data	We collect $20,500$ points of interest data for NYC and $18,943$ points of interest data for Chicago of 7 different categories, namely residence, travel, arts and entertainment, outdoors and recreation, education, professional, shops. After that, we use this data to construct POI graphs for all the regions of the source and the target cities.
Chicago Taxi data (2019) and NYC Taxi data (2019)	We collect $34,738,922$ records of green and yellow taxi data for NYC from 01 January 2019 to 31 December 2019. Similarly, we collect $29,110,097$ records of taxi data from the same time frame for Chicago. We convert the data into weekly data for all regions and use the pick-up and drop-off locations to convert the taxi data into inflow and outflow data.
Chicago road network data and NYC road network data	We collected the road network data using a publicly available python library, OSMnx. We collected the road network data, extracted information about the nodes, and assigned each node a feature of whether a node is a road junction or a dead end. Finally, we use this data to construct road network graphs for all the regions of the source and the target cities.

5.2 Parameters and Preprocessing

We choose the number of similar regions for our transfer learning model, $m = 5$. We normalize the concatenated crime and feature embeddings before passing them to the SAB-LSTM layer. We also scale the crime data of Chicago to the [-1,1] range and scale the output of the transfer learning model likewise. Before comparing the performance on the test set, we inversely transform the test data and output of the model to get the correct prediction. In addition, we add a 40 percent dropout to the output of the SAB-LSTM layer to reduce overfitting. We also introduce a 30 percent dropout to the region similarity learning unit and region representation learning unit for similar reasons. For the dimension of the feature vectors of each region learned from the region representation learning unit, we choose $d = 100$. We consider $T = 20$ as the number of time steps considered for predicting crime occurrences at the next time step. Finally, we use Adam [55] as our optimizer.

5.3 Evaluation Metric

We use mean squared error (MSE) as our evaluation metric for our transfer learning model. Given the crime data of the m most similar source city regions, we construct the training data by adding the crime data of these regions for a particular crime category. We then take 67 percent of this data for training. If we represent the summed crime data of the source city m similar regions I_{R_j} for a target region R_j as $y_{I_{R_j}}$ and the predictions from the SAB-LSTM layer as $\hat{y}_{R_j}$, then the loss function during the training can be described as:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y^i_{I_{R_j}} - \hat{y}^i_{R_j})^2$$

where N = number of predictions

5.4 Performance Comparison

This section describes the models used in our thesis for comparison purposes and shows the results of our experiments in Section 5.4.1 and Section 5.4.2 respectively.

5.4.1 Models

We consider two different classes of models for comparing the performance of different models for crime prediction.

(i) With Transfer Learning

(ii) Without Transfer Learning

Table 5.2 gives an overview of the models used in our thesis. We train two models, M1 and M2, using transfer learning for predicting crime. Both of these models do not use the crime data of the target city during the training phase. The difference between the two models is that the first uses an attention-based similarity learning technique, and the second uses a correlation-based similarity learning technique.

Furthermore, for the non-transfer learning models, M3 is the non-transfer learning version of M1 and M2. We train this model using the target city NYC feature embedding and NYC data as train labels. The final model, M4, is an existing literature called AIST [8], which predicts crime by learning the feature embedding and crime embedding of the city where we want to predict crime and uses historical crime data as train labels. We use the target city crime data to evaluate all four models.

5.4.2 Results

Table 5.3 summarizes the experiment results run on different variations of the models mentioned in Section 5.4.1. We show our experiment results for two different precincts of the target city, 19 and 22. We can see that our proposed transfer learning model M1 with no crime data of the target city performs as well as the models M3 and M4 which are trained using the target city crime data for precinct 19 and crime category Theft. For all other crime categories for this precinct, we can see that our model performance reduces slightly in the range of 0.05% - 0.26% compared to the performance of the models M3 and M4, which is negligible.

For precinct 22, we can see that our model outperforms the models trained with crime data by 0.49% for the crime category Robbery and by 0.94% for the crime category Burglary. This proves that our model has the potential to perform as well as the traditional models trained with crime data. Figure 5.1a, Figure 5.1b, Figure 5.1c shows the performance of the models in barplots for the three crime categories for precinct 19.

5.5 Comparison between Similarity Learning Techniques

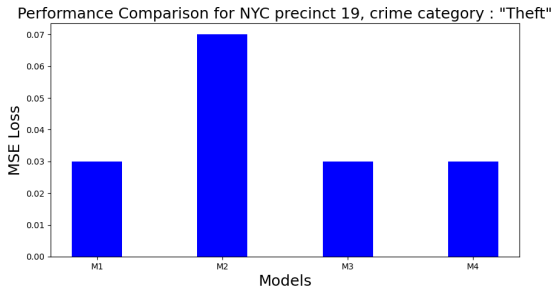
From Figure 5.1a, Figure 5.1b and Figure 5.1c, we see that between the two transfer learning models, M1 and M2, the one that learned the most similar source city regions using attention-based similarity tends to perform better than the one that learned the similar regions using Pearson Correlation-based region similarity. For precinct 19, our model M1 with the attention-based

Table 5.2: Model details

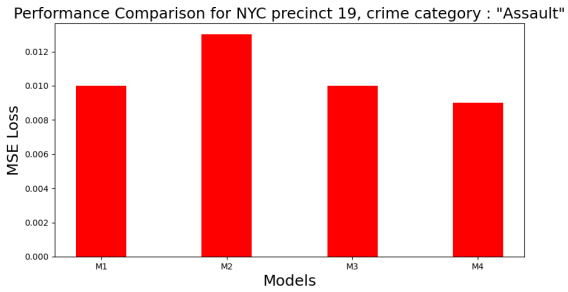
With Transfer Learning			
Model	Train data and label	Test data	Similarity Module
M1	• Train Data: NYC feature embedding, Chicago crime embeddings • Train Label: Chicago crime data	NYC crime data	Attention based similarity
M2	• Train Data: NYC feature embedding, Chicago crime embeddings • Train Label: Chicago crime data	NYC crime data	Pearson Correlation based similarity
Without Transfer Learning			
M3	• Train Data: NYC feature embedding • Train Label: NYC crime data	NYC crime data	Not applicable
M4 (AIST)	• Train Data: NYC feature embedding, NYC Crime embedding • Train Label: NYC crime data	NYC crime data	Not applicable

Table 5.3: Performance comparison

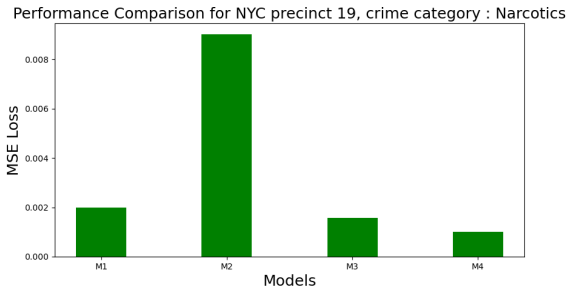
Precinct	Category	M1		M2		M3	M4 (AIST)
		MSE Loss	Similar Regions	MSE Loss	Similar Regions	MSE Loss	MSE Loss
19	Theft	0.03	28, 33, 7, 32, 23	0.07	55, 38, 54, 18, 29	0.03	0.03
	Robbery	0.0001	40, 50, 16, 70, 73	0.001	55, 38, 54, 18, 29	7.64e-7	5.23e-8
	Assault	0.01	70, 50, 16, 14, 40	0.013	55, 38, 54, 18, 29	0.00948	0.009
	Burglary	0.005	33, 28, 43, 24, 59	0.007	55, 38, 54, 18, 29	0.0047	0.004
	Narcotics	0.002	33, 28, 68, 60, 54	0.009	55, 38, 54, 18, 29	0.00158	0.001
22	Theft	0.0001	50, 40, 61, 70, 14	0.007	55, 38, 54, 18, 29	1.71e-6	5.18e-8
	Robbery	8.52e-7	50, 40, 70, 61, 14	0.0003	55, 38, 54, 18, 29	1.69e-6	5.24e-8
	Assault	0.007	50, 61, 14, 1, 70	0.003	55, 38, 54, 18, 29	0.0006	5.25e-8
	Burglary	3.13e-5	70, 61, 16, 14, 73	0.0002	55, 38, 54, 18, 29	0.0006	5.24e-8
	Narcotics	0.008	33, 60, 68, 54, 29	0.007	55, 38, 54, 18, 29	0.0006	5.28e-8



(a) NYC Precinct 19, Crime Category: Theft



(b) NYC Precinct 19, Crime Category: Assault

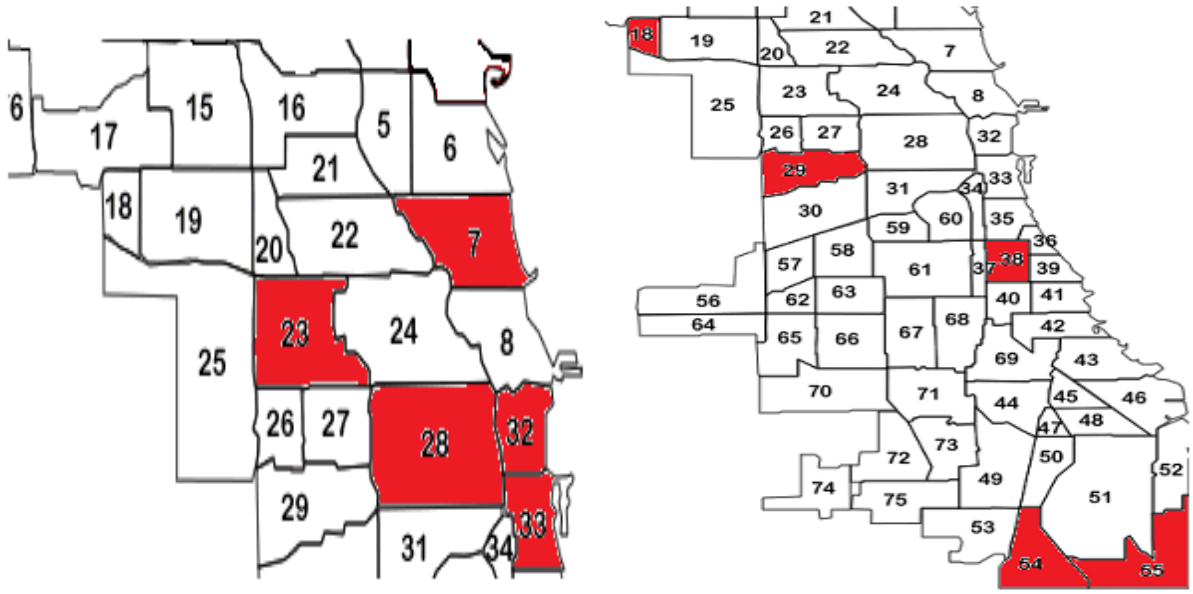


(c) NYC Precinct 19, Crime Category: Narcotics

Figure 5.1: Performance comparisons for models M1, M2, M3, M4 for precinct 19 and crime categories Theft, Assault, Narcotics

similarity learning module outperforms the Pearson Correlation-based model M2 by 1.33% in Theft prediction, 9% in Robbery prediction, 0.3% while predicting Assault, 0.4% in predicting Burglary, and 3.5% while predicting Narcotics. This result is also reflected in the predictions for precinct 22 for all the crime categories except Narcotics, where M2 outperforms M1 by 0.14%. Figure 5.2 shows the similar regions learned for precinct 19 of the target city by using the two different similarity learning techniques.

We see from Figure 5.2a that the regions which are marked as similar $(7, 23, 38, 32, 33)$ by the attention-based similarity learning module for precinct 19 of the target city are either neighbors to each other or situated one or two hops away. Therefore, these regions have a higher chance of sharing similar external feature patterns and crime patterns. On the other hand, Pearson Correlation only learns linear correlation between two vectors. Thus this method can not capture the spatial or temporal effects while learning the similar regions. We can see this in Figure 5.2b. For precinct 19, this method's regions marked as similar regions are situated in different parts of the Chicago city. Therefore have a lower chance of capturing the spatial effects of crime. For these reasons, we conclude that the attention-based transfer learning model performs better than the Pearson Correlation method.

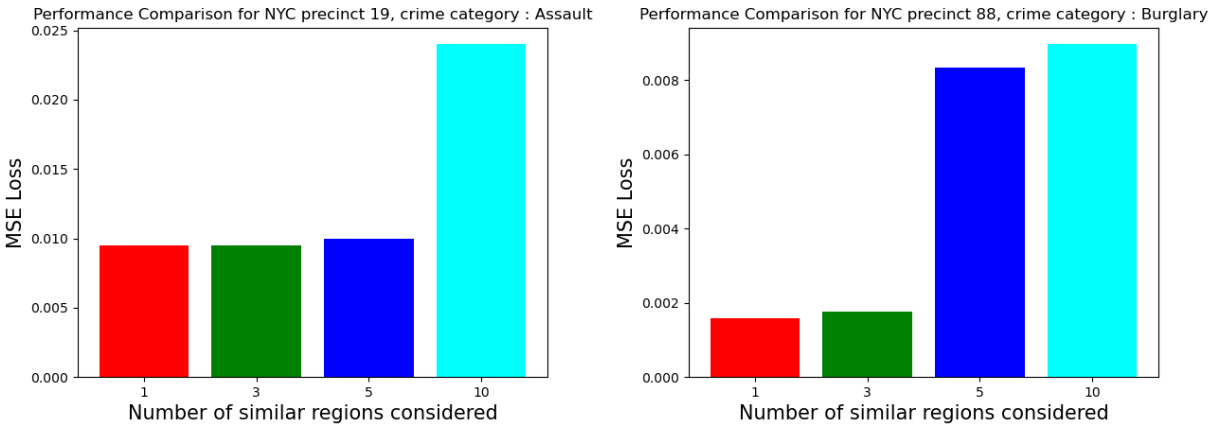


(a) Attention based similar regions (b) Pearson Correlation based similar regions

Figure 5.2: Similar Chicago regions for NYC precinct: 19, and crime category: Theft based on attention based similarity and Pearson Correlation based similarity

5.6 Effect of Number of Similar Regions

We also experiment with different numbers of similar regions. We choose 1,3,5 and 10 as the number of similar regions and show our model's performance using a bar plot for precinct 19 and precinct 88 of the target city and crime category Assault and Burglary respectively in Figure 5.3a and Figure 5.3b.



(a) Effect of number of similar regions for NYC precinct 19, crime category: Assault (b) Effect of number of similar regions for NYC precinct 88, crime category: Burglary

Figure 5.3: Effect of varying the number of similar regions

We see that increasing the number of similar regions negatively affects the model's performance. We consider redundant and unnecessary information while learning the crime embeddings and

finding similar regions while using many similar regions having low similarity values. For our experiments we choose $m = 5$ as the number of similar regions as this value lies between the two extreme options $m = 1$ and $m = 10$.

Chapter 6

Conclusion

We present a transfer learning-based deep learning crime prediction model to deal with the absence of crime data. Existing literature cannot predict crime in regions where crime data is unavailable. To alleviate this shortcoming, we introduce transfer learning in our model. Our model uses a region similarity learning unit to find the most similar source city regions for the region in the target city. We predict the number of crime occurrences of a particular category in a target city region at a particular time step. For this purpose, we propose four units in our model. Firstly, the region representation learning unit captures all the dependencies with crime occurrences in a region and learns a feature embedding of the regions that encapsulate those dependencies. After that, our region similarity learning unit adaptively finds similar source city regions for a target city-region based on those dependencies. The third unit is the crime embedding unit that transfers the knowledge from m similar source city regions to a region in the target city where we want to predict crime. Finally, the crime prediction unit gives the desired prediction in the target city region. We conduct experiments on real publicly available datasets. Our evaluation of the transfer learning model shows that it performs almost as well as the trained models using historical crime data.

For our future work, we will explore cross-categorical crime prediction, where we use crime data of one category to predict the number of crime occurrences in another category. We also consider the multi-city source regions as another possible field to explore in the future. Furthermore, we plan on extending our work to other spatio-temporal domains. Our thesis can bridge the gap between transfer learning and crime prediction.

References

[1] O. Jonathan, A. Olusola, T. Bernadin, and T. Inoussa, "Impacts of crime on socio-economic development," *Mediterranean Journal of Social Sciences*, vol. 12, p. 71, 09 2021.

[2] X. Zhao and J. Tang, "Crime in urban areas: A data mining perspective," *ACM SIGKDD Explorations Newsletter*, vol. 20, 04 2018.

[3] S. K. Rumi, K. Deng, and F. D. Salim, "Crime event prediction with dynamic features," *EPJ Data Science*, vol. 7, no. 1, p. 43, 2018.

[4] J. Sun, M. Yue, Z. Lin, X. Yang, L. Nocera, G. Kahn, and C. Shahabi, "Crimeforecaster: Crime prediction by exploiting the geographical neighborhoods' spatiotemporal dependencies," in *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part V*, pp. 52–67, Springer International Publishing, 2021.

[5] C. Huang, C. Zhang, J. Zhao, X. Wu, D. Yin, and N. Chawla, "Mist: A multiview and multimodal spatial-temporal learning framework for citywide abnormal event forecasting," in *The World Wide Web Conference*, pp. 717–728, 2019.

[6] C. Huang, J. Zhang, Y. Zheng, and N. V. Chawla, "Deepcrime: Attentive hierarchical recurrent networks for crime prediction," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1423–1432, 2018.

[7] C. Wang, Z. Lin, X. Yang, J. Sun, M. Yue, and C. Shahabi, "Hagen: Homophily-aware graph convolutional recurrent network for crime forecasting," *arXiv preprint arXiv:2109.12846*, 2021.

[8] Y. Rayhan and T. Hashem, "Aist: An interpretable attention-based deep learning model for crime prediction," *arXiv preprint arXiv:2012.08713*, 2020.

[9] P. Chen, H. Yuan, and X. Shu, "Forecasting crime using the arima model," in *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 5, pp. 627–630, 2008.

[10] L. McClendon and N. Meghanathan, "Using machine learning algorithms to analyze crime data," *Machine Learning and Applications: An International Journal (MLAIJ)*, vol. 2, no. 1, pp. 1–12, 2015.

[11] S. R. Bandekar and C. Vijayalakshmi, "Design and analysis of machine learning algorithms for the reduction of crime rates in india," *Procedia Computer Science*, vol. 172, pp. 122–127, 2020.

[12] C. Tabedzki, A. Thirumalaiswamy, P. van Vliet, S. Agarwal, and S. Sun, "Yo home to bel-air: predicting crime on the streets of philadelphia," *University of Pennsylvania, CIS*, vol. 520, 2018.

[13] P. Stalidis, T. Semertzidis, and P. Daras, "Examining deep learning architectures for crime classification and prediction," *Forecasting*, vol. 3, pp. 741–762, 10 2021.

[14] L. Wang, X. Geng, X. Ma, F. Liu, and Q. Yang, "Cross-city transfer learning for deep spatio-temporal prediction,"

[15] R. Jiang, X. Song, Z. Fan, T. Xia, Z. Wang, Q. Chen, Z. Cai, and R. Shibasaki, "Transfer urban human mobility via poi embedding over multiple cities," *ACM Transactions on Data Science*, vol. 2, no. 1, pp. 1–26, 2021.

[16] H. Yao, Y. Liu, Y. Wei, X. Tang, and Z. Li, "Learning from multiple cities: A meta-learning approach for spatial-temporal prediction," in *The World Wide Web Conference*, pp. 2181–2191, 2019.

[17] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[18] K. Zhang, Y. Zhu, J. Wang, and J. Zhang, "Adaptive structural fingerprints for graph attention networks," in *International Conference on Learning Representations*, 2019.

[19] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Sixth international conference on data mining (ICDM'06)*, pp. 613–622, IEEE, 2006.

[20] C. Guo, B. Yang, J. Hu, and C. Jensen, "Learning to route with sparse trajectory sets," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 1073–1084, IEEE, 2018.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] N. R. Ke, A. G. ALIAS PARTH GOYAL, O. Bilaniuk, J. Binas, M. C. Mozer, C. Pal, and Y. Bengio, "Sparse attentive backtracking: Temporal credit assignment through reminding," *Advances in neural information processing systems*, vol. 31, 2018.

[23] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE international joint conference on neural networks*, vol. 2, pp. 729–734, 2005.

[24] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017.

[25] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, Citeseer, 2013.

[26] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[27] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," *Advances in neural information processing systems*, vol. 2, 1989.

[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[29] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[30] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, pp. 1–55, 2014.

[31] L. Wang, B. Guo, and Q. Yang, "Smart city development with urban transfer learning," *Computer*, vol. 51, no. 12, pp. 32–41, 2018.

[32] L. Wang, X. Geng, X. Ma, D. Zhang, and Q. Yang, "Ridesharing car detection by transfer learning," *Artificial Intelligence*, vol. 273, pp. 1–18, 2019.

[33] B. Guo, J. Li, V. W. Zheng, Z. Wang, and Z. Yu, "Citytransfer: Transferring inter-and intra-city knowledge for chain store site recommendation based on multi-source urban data," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–23, 2018.

[34] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[35] N. Gao, W. Shao, M. S. Rahaman, J. Zhai, K. David, and F. D. Salim, "Transfer learning for thermal comfort prediction in multiple cities," *Building and Environment*, p. 107725, 2021.

[36] T. He, J. Bao, R. Li, S. Ruan, Y. Li, L. Song, H. He, and Y. Zheng, "What is the human mobility in a new city: Transfer mobility knowledge across cities," in *Proceedings of The Web Conference 2020*, pp. 1355–1365, 2020.

[37] Y. Zheng, "Methodologies for cross-domain data fusion: An overview," *IEEE transactions on big data*, vol. 1, no. 1, pp. 16–34, 2015.

[38] Y. Wei, Y. Zheng, and Q. Yang, "Transfer knowledge between cities," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1905–1914, 2016.

[39] F. K. Bappee, A. Soares, L. M. Petry, and S. Matwin, "Examining the impact of cross-domain learning on crime prediction," *Journal of big data*, vol. 8, no. 1, pp. 1–27, 2021.

[40] A. Bogomolov, B. Lepri, J. Staiano, N. Oliver, F. Pianesi, and A. Pentland, "Once upon a crime: towards crime prediction from demographics and mobile data," in *Proceedings of the 16th international conference on multimodal interaction*, pp. 427–434, 2014.

[41] X. Wang, D. E. Brown, and M. S. Gerber, "Spatio-temporal modeling of criminal incidents using geographic, demographic, and twitter-derived information," in *2012 IEEE International Conference on Intelligence and Security Informatics*, pp. 36–41, IEEE, 2012.

[42] M. De Nadai, Y. Xu, E. Letouzé, M. C. González, and B. Lepri, "Socio-economic, built environment, and mobility conditions associated with crime: a study of multiple cities," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[43] X. Zhao and J. Tang, "Exploring transfer learning for crime prediction,"

[44] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[45] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7793–7804, 2020.

[46] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," *arXiv preprint arXiv:2006.07988*, 2020.

[47] C. of Chicago, "An official website of the city of chicago." https://www.chicago.gov/city/en/depts/dgs/supp_info/citywide_maps.html, 2010-2022.

[48] C. of New York, "New york city police department." https://www1.nyc.gov/site/nypd/about/about-nypd/about-nypd-landing.page, 2022.

[49] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017.

[50] D. YANG, "Foursquare dataset." https://sites.google.com/site/yangdingqi/home/foursquare-dataset?authuser=0.

[51] C. of New York, "Nyc opendata." https://data.cityofnewyork.us/widgets/rxuy-2muj?mobile_redirect=true, 2017.

[52] C. T. Trips, "City of chicago data portal." https://data.cityofchicago.org/Transportation/Taxi-Trips-2019/h4cq-z3dy, 2019.

[53] C. of New York, "Nyc opendata." https://data.cityofnewyork.us/browse?q=taxi%20trips%202019&sortBy=relevance, 2017.

[54] C. Crimes, "City of chicago data portal." https://data.cityofchicago.org/Public-Safety/Crimes-2019/w98m-zvie, 2019.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Generated using Undegraduate Thesis LaTeX Template, Version 1.4. Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

This thesis was generated on Monday 16th May, 2022 at 9:53am.