# Crime Prediction with Graph Neural Networks and Multivariate Normal Distributions

**Selim Furkan Tekin** ( tekin@ee.bilkent.edu.tr )
Bilkent University
**Suleyman Serdar Kozat**
Bilkent University

**Research Article**

**Keywords:** crime forecasting, probabilistic graph models, deep learning

**Posted Date:** April 4th, 2022

**DOI:** https://doi.org/10.21203/rs.3.rs-1508116/v1

# Crime Prediction with Graph Neural Networks and Multivariate Normal Distributions

Selim Furkan Tekin[1*] and Suleyman Serdar Kozat[1]

[1*]Electrical and Electronics Engineering, Bilkent University, Bilkent, Ankara, 06800, Turkey.

*Corresponding author(s). E-mail(s): tekin@ee.bilkent.edu.tr;
Contributing authors: kozat@ee.bilkent.edu.tr;

**Abstract**

We study high-resolution crime prediction and introduce a new generative model applicable to any spatiotemporal data with Graph Convolutional Gated Recurrent Units (Graph-ConvGRU) and multivariate Gaussian distributions. We introduce a subdivision algorithm and create a graph representation to tackle the sparsity and complexity problem in high resolution spatiotemporal data. By leveraging the flexible structure of graph representation, we model the spatial, temporal, and categorical relations of crime events and produce state vectors for each region. We create a multivariate probability distribution from the state vectors and train the distributions by minimizing the KL-Divergence between the generated and the actual distribution of the crime events. After creating the distributions, crime can be predicted in any resolution as the first time in the literature. In our experiments on real-life and synthetic datasets, our model obtains the best score with respect to the state-of-the-art models with statistically significant improvements. Hence our model is not only generative but also precise. We also provide the source code of our algorithm for reproducibility.

**Keywords:** crime forecasting, probabilistic graph models, deep learning.

## 1 Introduction

Crimes are unlawful acts endangering public safety, leading to colossal life and economic losses if governments do not take the necessary precautions. Understanding the latent patterns of crimes in a geographical region of a city is highly important for preventing upcoming crimes events. With accurate and reliable crime prediction, the police force can be at the right place at the right time.

However, the problem of crime prediction is a highly complex and challenging task since it contains spatial, temporal, and categorical complexities with inherent interrelations [1–5]. In general, prior works on crime prediction can be classified in two main categories. The prior works in the first category [6–9] divide city area into a grid, where each cell contains the crime information for each crime category that happened in the past. The works in the second category [10–12], divide the city area into unordered regions and create a graph structure that captures the correlations among the regions with corresponding edge weights and node features. However, the regions in both of these approaches correspond to extremely wide sections of the cities. To the best of our knowledge, the minimum size of a region in grid-based methods has $4\text{km}^2$ area. For the graph-based approaches, the regions correspond to the districts of a city

and the range of sizes changes from $2km^2$ to $35km^2$. Thus, assigning a probability value to such a broad area is not expedient and decreases the feasibility in real-life situations. Note that one can increase the number of regions to enhance the detailing. However, this leads to severe problems in both of these two approaches. For the grid-based approach, as the grid size increases, the complexity and sparsity exponentially grow. On the other hand, it is difficult to define a rule to add new regions to the graph in graph-based methods.

This paper introduces a probabilistic graph-based model called *High-Resolution Crime Forecaster* (HRCF) that performs grid-based crime prediction to remedy these problems. We learn a multivariate probability distribution instead of assigning one probability value to each node in a graph and model the underlying dynamics of each region with the objective of minimizing the cross entropy. We build our graph on spatiotemporal data using a divide-and-conquer algorithm to up weight the minority class. The architecture contains graph convolutional neural networks as memory units, which capture the spatial and temporal correlations. In each time step, the model predicts Multivariate Gaussian distributions for all the regions conditioned on the previous observations, allowing us to generate predictions for any resolutions. We demonstrate significant performance gains in real life data through an extensive set of experiments compared to the state-of-the-art methods and show that our model is not only generative but also more precise.

### 1.1 Prior Art and Comparisons

Deep learning structures are extensively used in the signal processing literature to model crowd movements and provide high performance. For example, in [6], authors implemented deep learning based prediction model for spatiotemporal data (DeepST), where they model the crowd flow with the grid-based forecasting system. The authors of [8] adapted their work to predict crime distribution over the Los Angeles dataset and achieved high scores but with a very high computational cost. The authors of [9] designed multi-model units capturing spatial, temporal, and semantic information and producing predictions for each crime type for a 11km × 11km grid size but the model performance decreases as the spatial resolution increases. Due to the high sparsity and complexity of grid-based approaches, graphical models are recently used [13, 14] to remedy these problems. In crime prediction, [10] formed the graph structure by using independent Hawkes processes in each node and obtained edge weights and performed predictions for 50 regions in the Chicago crime dataset. Another study [11] implemented Gated Recurrent Network with Diffusion Convolution modules following a Multi-Layer Perceptron (MLP). Recently, [12] introduced a homophily-aware constraint on the loss function so that neighboring region nodes share similar crime patterns. Nevertheless, these works are not generative and work on the districts, which causes the complexity and sparsity problems. Our contributions are as follows:

- We introduce a subdivision algorithm to create a graph from grid-based spatiotemporal data to reduce detrimental effects of sparsity.
- We present a novel graphical architecture with Multivariate Gaussian distributions to jointly train the micro and macro probabilities and model the actual data distribution.
- With the generative structure of our model, we produce highly accurate predictions even for high spatial resolution.
- We demonstrate significant performance gains through extensive set of experiments over real life crime data.

## 2 Problem Definition

Let $A = [lat_1, lat_2]$ and $B = [lon_1, lon_2]$ define the bounds of the city region that we divide into $I \times J$ disjointed cells, which creates the set of cells $M = \{m_{1,1}, \ldots, m_{i,j}, \ldots, m_{I,J}\}$. The cells contain the longitude, $a$, and latitude, $b$, information, which is given by $m_{i,j} = (a_i, b_i)$, and $a_i \in A$ and $b_j \in B$. For a given time window $T$, we also split $T$ into non-overlapping and consequent time slots $T = \{t_1, \ldots, t_K\}$. In addition, $L = \{c_1, \ldots, c_L\}$ represents the categories of crimes. Following these definitions of sets, we define appearance of a crime, $x$, with category $c_l$ that belongs to $m_{i,j}$ at time step $t_k$ as $x^l_{i,j,k} \in \{0, 1\}$. Next, we define an event matrix that shows all the events with category $c_l$ happened at $t_k$ in all cells as $\mathbf{X}^l_k = \{x^l_{1,1,k}, \ldots, x^l_{I,1,k}, x^l_{I,2,k}, \ldots, x^l_{I,J,k}\} \in \mathbb{R}^{I \times J}$. We also define an event matrix representing any crime

event regardless of the crime type that happened in all the cells at time $t_k$ as $\mathbf{X}_k \in \mathbb{R}^{I \times J}$ by dropping the $l$ superscript.

Here, our primary goal is to predict the next event matrix representing any crime event given the previous event matrices with different categories. Thus, our goal is to learn the following mapping, $\{\mathbf{X}_1^1, \ldots, \mathbf{X}_K^L\} \xrightarrow{\text{f(.)}} \{\mathbf{X}_{K+1}\}$. Even though we produce grid-based predictions, we use a graph-based approach to model the interrelations in the data, as shown in Section 3. Furthermore, our model is generic so that we can generate predictions even for high resolutions without retraining our model, i.e., $\mathbf{X}_\text{K}$ can be $\mathbb{R}^{I' \times J'}$ where $I' \geq I$ and $J' \geq J$ so that we mitigate sparsity issues.

# 3 Methodology

## 3.1 Subdivision Algorithm and Graph Representation

To observe the distribution of events in a region, we sum all the event matrices in a period with length $T$ to create the $\mathbf{Q}$ matrix, $\mathbf{Q} = \sum_{k=1}^{T} \mathbf{X}_k$. An example $\mathbf{Q}$ matrix is shown in the left side of Fig. 1. Observe that the number of cells that contain no events is the majority, i.e., 60.42% of the cells have zero event count. Our goal is to remedy this sparsity problem by introducing new regions that have equal or close event counts. As shown in Algorithm 1, our subdivision algorithm partitions the set $M$ into $N$ disjoint sets with the divide-and-conquer approach to create the region set $R = \{r_1, \ldots, r_N\}$, where each region contains arbitrary number of cells and $M = \cup_i r_i$ and $\cap_i r_i = \emptyset$. By this algorithm, all the resulting regions contain a total event count less than a threshold value, $\tau$. Our base condition is to have a region larger than 2km $\times$ 2km or a total event count less than the threshold value. With this algorithm we drop the percentage of the regions that have zero event count to 5.97%.

After the subdivision process, we build our graph with the nodes representing the centers of each region and the edges representing the distance between the neighboring nodes. We form the region graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ represents the set of nodes with $\mid \mathcal{V} \mid = N$ where $N$ is the number of regions and $\mathcal{E}$ is the set of undirected edges

**Algorithm 1** Algorithm for Subdivision

```
procedure DivideRegions(Q, τ, r, c)
    Q* ⟵ Q[r[0] : r[1], c[0] : c[1]]    ▷ Select the sub-region
    if sum(Q*) < τ OR Q*.shape < (2, 2) then
        return [[r, c]]                                  ▷ base case
    else
        I ← SplitRegions(Q*, r, c)                       ▷ divide 4
        R ← [ ]
        for r, c ∈ I do
            R_new ← DivideRegions(Q*, τ, r, c)
            R.append(R_new)
        end for
        return R
    end if
end procedure
```

between region nodes, $\mathbf{A}$ is the weight matrix. Let $v_i \in \mathcal{V}$ denotes a node and $\epsilon_{ij} = (v_i, v_j) \in \mathcal{E}$ denotes an edge. The weight matrix $\mathbf{A}$ has dimension $N \times N$ with $A_{i,j} = (v_i - v_j)^2$ if $\epsilon_{i,j} \in \mathcal{E}$ and $A_{i,j} = 0$ if $\epsilon_{i,j} \notin \mathcal{E}$.

We define the input event in the graph with category $l$ belonging $r_i$ at time step $t_k$ as $z_{i,k}^l \in \{0, 1\}$. Hence, $\mathbf{z}_k^l = \{z_{1,k}^l, \ldots, z_{N,k}^l\} \in \mathbb{R}^N$ represents events with type $l$ happened in all the regions in this graph at time step $t_k$. Thus, the subdivision algorithm provides the following mapping $\{\mathbf{X}_1^l, \ldots, \mathbf{X}_K^L\} \xrightarrow{\text{s(.)}} \{\mathbf{z}_1^l, \ldots, \mathbf{z}_\text{K}^L\}$. Furthermore, we define the graph function, $g(.)$, which takes the output vectors of the subdivision algorithm and produces the state vectors, $\mathbf{h}_{i,k} \in \mathbb{R}^{d_h}$, for each graph node, where $d_h$ is the hidden dimension. We formulate the graph function as: $\{\mathbf{z}_1^1, \ldots, \mathbf{z}_K^L, \mathcal{G}\} \xrightarrow{\text{g(.)}} \{\mathbf{h}_{1,\text{K}+1}, \ldots, \mathbf{h}_{\text{N},\text{K}+1}\}$ where we generate the state vectors for the next time step for all the nodes. The details of this function is given in Section 3.3.

## 3.2 The Likelihood Model

Let $\mathcal{Y}$ model the random variable with the realization of $\mathbf{y}_{i,k} = \{y_1, y_2\} \in \mathbb{R}^2$, where $y_1$ and $y_2$ represent the longitude and latitude of any crime happening in $r_i$ at time step $t_k$. Let $p(\mathbf{y}_{i,k})$ be the probability density function governing this random variable. Our goal is to learn the conditional probability function $p(\mathbf{y}_{i,k} \mid \mathbf{z}_1^1, \ldots, \mathbf{z}_{k-1}^L)$. We model this probability density function with a likelihood model, $l(y_{i,k};\, \theta(\mathbf{h}_{i,k}))$, and parametrize it by the $\theta(.)$ function. This underlying model is chosen as a Multivariate Gaussian Distribution (MGD) since MGD can readily model even complex functions while avoiding overfitting [15]. Here, $\theta(.)$ corresponds to the mean vector and

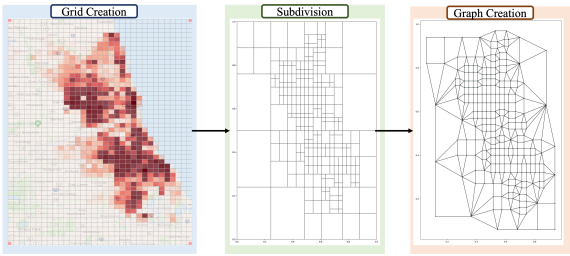



**Fig. 1**: We show the phases of creating our graph. First, we create a grid by dividing the region into $I \times J$ cells, where each cell contains the total event count. Second, with the sampling algorithm that favors the crowded regions, we obtain graph regions. Third, we create the graph with nodes representing the centers of each region and edges representing the distances between nodes.

covariance matrix, $\theta = (\mu, \mathbf{\Sigma})$, where $\mu = \begin{bmatrix}\mu_1 & \mu_2\end{bmatrix}$, and $\mathbf{\Sigma} = \begin{bmatrix} v_1^2 & 0 \\ 0 & v_2^2 \end{bmatrix}$. Note that cross terms are selected 0 to alleviate overfitting.

In training, we could directly maximize the log-likelihood (MLE) where we put the locations of each crime event into the likelihood model, $\mathcal{L} = \sum_{i=1}^{N} \log(l(\mathbf{y}_{i,k};\ \ \theta(\mathbf{h}_{i,k})))$, and sum all the log-likelihoods in each region. However, the MLE loss does not directly suffer from negative predictions, which, as we observe in our experiments, causes the model to get stuck on the local minima. Therefore, we minimize the Kullback-Leibler divergence [16], between the actual distribution and the likelihood model, which is equivalent to minimizing the cross-entropy $\mathcal{L} = -\sum_{i=1}^{N} \sum_{\mathbf{y} \in \mathcal{Y}_{i,k}} p(\mathbf{y}) l^*(\mathbf{y};\ \ \theta(\mathbf{h}_{i,k}))$, where $l^* = \log(l(.))$, but the term $p(\mathbf{y})$ is still unknown and note that $\mathbf{y}$ contains continuous values. Recall that we quantize each region with the grid-based approach where each cell takes 1 if a crime happened, 0 otherwise. Hence, we can rewrite our loss as binary-cross entropy:

$$\mathcal{L} = -\frac{1}{IJ} \sum_{i=1}^{IJ} \sum_{\mathbf{y} \in \mathcal{Y}_{i,k}} q_i l^* + (1 - q_i) \log(1 - l), \quad (1)$$

where $q(.)$ maps the longitude and latitude to the crime event value:

$$q(y_1, y_2) = \begin{cases} 1, & \text{if an event is observed at } (y_1, y_2), \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

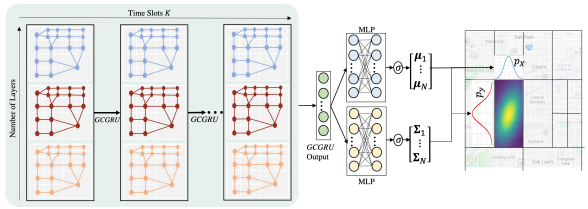



**Fig. 2**: We show HRCF architecture. In each time slot $t_k$, we extract features with the Graph-ConvGRU operations. We pass these features to two different MLP heads, $\mu$ and $\Sigma$. Each head produces the $\mu_i$ and $\mathbf{\Sigma}_i$ parameters of distribution in each node.

The parameters of the model is optimized by minimizing (1) with the Adam optimizer using batches of training data.

### 3.3 High Resolution Crime Forecaster (HRCF)

A crime event correlates with abnormal events in nearby locations and an after-effect of a crime can trigger another crime in nearby areas. We use graph convolution to learn such relations due to its success in modelling spatialtemporal series[14]. The graph convolution is similar to the traditional 2D convolution. An image pixel is similar to a node in a graph, where the size of a filter in 2D convolution determines the number of neighbors. The 2D convolution takes the weighted average of pixel values of the central node along with its neighbors. Similarly, graph convolution takes the weighted average of the central node along with its unordered and variable in size neighbors. Hence, each node uses the information coming from its neighbors. Therefore, we use convolution operations to propagate crime information at each node to the other nearby nodes. We assume that structurally similar nodes will show close behavior. Since graph convolutional networks allow parameter sharing across the nodes in the graph, we can capture the spatial dependencies for different patterns observed in other locations. To capture the temporal relations, we use graph convolutions with deep memory units [17]. Graph Convolutional Gated Recurrent Unit (Graph-ConvGRU) generalizes the Convolutional Gated Recurrent Unit (ConvGRU) model [18] to graphs by replacing the Euclidian 2D convolution $*$ by the graph

convolution $*_{\mathcal{G}}$:

$$\mathbf{i} = \sigma(\mathbf{W}_{zi} *_{\mathcal{G}} \mathbf{z}_k + \mathbf{W}_{hi} *_{\mathcal{G}} \mathbf{h}_{k-1}),$$
$$\mathbf{r} = \sigma(\mathbf{W}_{zr} *_{\mathcal{G}} \mathbf{z}_k + \mathbf{W}_{hr} *_{\mathcal{G}} \mathbf{h}_{k-1}),$$
$$\tilde{\mathbf{h}} = \tanh(\mathbf{W}_{zh} *_{\mathcal{G}} + \mathbf{W}_{hh} *_{\mathcal{G}} (\mathbf{r} \odot \mathbf{h}_{k-1})),$$
$$\mathbf{h}_k = \mathbf{i} \odot \mathbf{h}_{k-1} + (1 - \mathbf{i}) \odot \tilde{\mathbf{h}},$$

where $\mathbf{W}_{h.} \in \mathbb{R}^{K \times d_h \times d_h}$ and $\mathbf{W}_{z.} \in \mathbb{R}^{K \times d_h \times d_x}$ are the parameters that we learn. The hidden dimension, $d_h$, the input dimension, $d_x$, and $K$ determine the number of parameters, which is independent of the number of nodes $N$. The gates $\mathbf{i}$, $\mathbf{r}$, and $\tilde{\mathbf{h}}$ enable resetting and updating the stored information.

As show in Fig. 2, we stack Graph-ConvGRU units and use as an encoder to encode the input sequence. At each time slot $t_k$, each unit takes the previous hidden state, $\mathbf{h}_{k-1}$ and input node feature, $\mathbf{z}_k$, to produce the next state $\mathbf{h}_k$. Note that we apply these operations for every node on the graph. Thus, in each time step, $t_k$, we feed the Graph-ConvGRU unit at each layer with $N$ different node features, $\mathbf{z}_{i,k}$, where the subscript $i$ denotes the node index. Similarly, at each time step, $t_k$, Graph-ConvGRU unit outputs the hidden state, $\mathbf{h}_{i,k}$, for each node. As given in the Section 2, $\mathbf{z}_{i,k} \in \mathbb{R}^L$ are the event vectors and $\mathbf{h}_{i,k} \in \mathbb{R}^{d_h}$ are the state vectors that parametrize the likelihood model, $l(\mathbf{y}_{i,k};\ \theta(\mathbf{h_{i,k}}))$. Each Graph-ConvGRU output, $\mathbf{h}_{i,k}$, passes to the MLP heads as shown in Fig. 2. Each MLP head is responsible for generating Multivariate Gaussian parameters for each node, $\mu_1 = \sigma(\mathbf{W}_{\mu}\mathbf{h}_{1,k} + b_{\mu}), \ldots, \mu_{\mathrm{N}} = \sigma(\mathbf{W}_{\mu}\mathbf{h}_{\mathrm{N},k} + b_{\mu})$ and $\mathbf{v}_1 = \sigma(\mathbf{W}_{v}\mathbf{h}_{1,k} + b_{v}), \ldots, \mathbf{v}_{\mathrm{N}} = \sigma(\mathbf{W}_{v}\mathbf{h}_{\mathrm{N},k} + b_{v})$, where $\mathbf{W}_{\mu} \in \mathbb{R}^{d_h \times 2}$, $\mathbf{W}_{v} \in \mathbb{R}^{d_h \times 2}$ and $b_{.}$ are the weights we learn, $\sigma(.)$ is the sigmoid activation, and $\mu_i = [\mu_{i,1}, \mu_{i,2}]$, $\mathbf{v}_i = [v_{i,1}, v_{i,2}]$ are the parameter vectors. Since the multivariate distribution is uncorrelated, we write the likelihood model as follows:

$$l(y_1, y_2 \mid \mu, \mathbf{\Sigma}) = \frac{\exp(\frac{-1}{2}(\frac{(y_1-\mu_1)^2}{v_1} + \frac{(y_2-\mu_2)^2}{v_2}))}{2\pi(v_1 v_2)^{\frac{1}{2}}},$$

which allows us to generate likelihoods for any location under the distribution, as we show in the right side of Fig. 2.

# 4 Experiments

This section compares our model performance with the baseline models on a real life and a synthetic datasets. We first describe the datasets then the models, and finally, discuss the results.

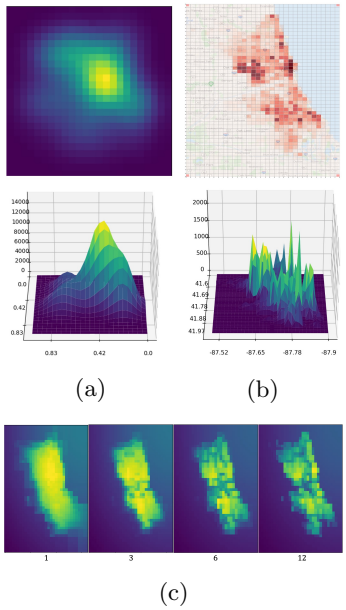
## 4.1 Data Description

### 4.1.1 Synthetic Dataset

First, we form an AR(1) process to produce correlated event counts in each time step. Second, we create multiple Multivariate Gaussian distributions belonging to 4 different categories, and each category has 10 different distributions. We randomly select the distributions centers between $[0.2, 0.8]$ and set the distribution variance values from $0.02, 0.01, 0.005, 0.003$. We sample the crime events, where the AR(1) process determines the total number of samples in each time step. Third, to create self and cross relations, we form 4 different temporal and categorical patterns and assign them to each distribution. At even time steps, distributions in the first category can generate events. Likewise, the distributions in the second category can generate at odd time steps. At time steps divisible by 4, the distributions in the third category can generate events, and the fourth category is the opposite of the third. Moreover, after the 10 time steps, we switch the behaviors of the categories, where the first category behaves like the second and vice-versa. Hence, our data has both temporal, spatial, and categorical interrelationships. An example data is shown in Fig. 3a.

### 4.1.2 The Chicago Crime Dataset

We collect the Chicago crime data between 2015 and 2019 [19]. The working area has a longitude range of $[41.60, 42.05]$ and a latitude range of $[-87.9, -87.5]$, creating a rectangle with height 50km and width 33km. The rectangle is non-Euclidean due to the shape of the Earth, yet, we assume an Euclidean rectangle. We partition the region into $50 \times 33$ disjoint geographical regions yielding 1km $\times$ 1km size squares. As described in Section 2, we map crime events to each square region. In Fig. 3b, we show all the crimes on these squares as a heatmap without considering crime type, where the dense red color represents high



(a) (b)

(c)

**Fig. 3**: We show the distribution of events for the Chicago crime data (a) and the synthetic data (b). We show the predictions of HRCF model in different epochs at (c).

event count. We also select the time resolution as 24 hours. Thus, our target is to predict the locations of the crime events regardless of their type in the partitioned regions for the next 24 hours. Moreover, we perform an exploratory data analysis [1] to investigate spatial, temporal, and categorical covariance. We filter the data by selecting crime types that forms the %90 percent of the crime data. We perform 4 experiments where each experiment consists of 1 year of data. We split the data into 10 months of training, 1 month of validation, and 1 month of testing in each experiment. We report the validation and test scores in Section 4.3.

## 4.2 HRCF Configurations

For the subdivision algorithm, we select the threshold value as 1000 and minimum region size as 2km $\times$ 2km and obtain 203 regions, which is also the number of nodes in our graph. Our model uses 16 input features total, 8 features representing the crime counts for each crime type, 2 features for the locations of the nodes, and 6 features for the calendar features such as weekends and holidays. We set the Graph-ConvGRU layers as 3, where hidden dimensions of each layers are, $50, 20, 10$, respectively. The $K$ value is 3 in each layer, and we set the bias flag "true" with the symmetric normalization. The number of layers in the MLP is 2, and the hidden layer contains 64 nodes. We set the input sequence length as 10 with a batch size of 5. For the training, we use a 0.003 learning rate, with an early stop tolerance of 5. Lastly, we use 0.001 as L2 regularization to regularize the model parameters.

## 4.3 Baseline Models

We compare our algorithm with the well known sequential learning models: Autoregressive Integrated Moving Average (ARIMA), Support Vector Regression (SVR), Random Forest (RF), Gaussian Process Regression (GPR) and ConvLSTM[18]. Note that ARIMA, SVR, RF, GPR are regression models; however, we make classification whether a crime event happens or not. Thus, after we provide the regression output, we select a threshold value for classification. We select the threshold value that maximizes the difference between True Positive Rate (TPR) and False Positive Rate (FPR). Since ARIMA, SVR, and RF are only applicable to single time series, we create a separate model for each cell. For the ConvLSTM, SVR, GPR, and RF, we feed the previous 10 time step values of the target series as input features to be fair with the graph model.

## 4.4 Performance Analysis and Results

In Table 1, we show the validation and test scores of models for both real life and synthetic datasets. We use the F1 metric, which measures the overall model performance of how precise and sensitive a model is. We also present the accuracy of our models since it is essential in crime prediction. We observe that our model outperforms the baseline models up to 3.10% for both validation and test scores for both datasets. We perform a statistical significance test with a p-value of 0.05 comparing baseline results with the HRCF and observe that the HRCF performs significantly better in all metrics. HRCF is not only sensitive to crime events

[1] You can find all of our codes to perform analysis and experiments at github.com/sftekin/high-res-crime-forecasting

| Validation Scores for Chicago Dataset | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HRCF | | ConvLSTM | | ARIMA | | RF | | SVR | | GPR | |
| Date | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc |
| 2015 | **0.724** | **0.895** | 0.705 | 0.865 | 0.708 | 0.866 | 0.700 | 0.860 | 0.701 | 0.872 | 0.695 | 0.868 |
| 2016 | **0.728** | **0.891** | 0.714 | 0.864 | 0.717 | 0.865 | 0.727 | 0.873 | 0.716 | 0.875 | 0.703 | 0.865 |
| 2017 | **0.731** | **0.895** | 0.709 | 0.865 | 0.716 | 0.868 | 0.713 | 0.865 | 0.718 | 0.879 | 0.703 | 0.869 |
| 2018 | **0.720** | **0.894** | 0.704 | 0.868 | 0.705 | 0.867 | 0.707 | 0.869 | 0.698 | 0.873 | 0.692 | 0.869 |
| Test Scores for Chicago Dataset | | | | | | | | | | | | |
| | HRCF | | ConvLSTM | | ARIMA | | RF | | SVR | | GPR | |
| Date | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc |
| 2015 | **0.723** | **0.888** | 0.713 | 0.871 | 0.705 | 0.863 | 0.718 | 0.874 | 0.709 | 0.876 | 0.694 | 0.867 |
| 2016 | **0.709** | **0.881** | 0.698 | 0.863 | 0.718 | 0.876 | 0.693 | 0.855 | 0.697 | 0.870 | 0.684 | 0.864 |
| 2017 | **0.708** | **0.877** | 0.699 | 0.865 | 0.686 | 0.853 | 0.697 | 0.862 | 0.697 | 0.872 | 0.686 | 0.866 |
| 2018 | **0.720** | **0.884** | 0.708 | 0.870 | 0.712 | 0.869 | 0.716 | 0.872 | 0.708 | 0.876 | 0.693 | 0.869 |
| Scores for the Synthetic Dataset | | | | | | | | | | | | |
| | HRCF | | ConvLSTM | | ARIMA | | RF | | SVR | | GPR | |
| Set | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc | F1 | Acc |
| Val | **0.484** | **0.775** | 0.470 | 0.738 | 0.344 | 0.611 | 0.387 | 0.677 | 0.341 | 0.784 | 0.300 | 0.660 |
| Test | **0.491** | **0.781** | 0.475 | 0.742 | 0.345 | 0.585 | 0.398 | 0.692 | 0.356 | 0.780 | 0.308 | 0.679 |

**Table 1**: We show model's validation and test scores on Chicago crime and synthetic dataset. Bold scores are the best. Date column shows the years the experiment belongs.

but also accurate. When we look at the score differences between years, we observe that HRCF is more adaptive to the changes in the dataset.

In the synthetic dataset results shown in Table 1, we expect to see higher performance from HRCF since we sampled the data from multiple Gaussian distributions. However, the performance of the GPR model is low, and GPR also assumes the data is coming from a Gaussian distribution. The reason is that the GPR model is not adaptive to temporal changes and can only capture the spatial covariates. From this perspective, we can say that HRCF behaves as a GPR model that can capture both spatial and temporal covariates.

In Fig. 3c, we show how the output of HRCF evolves between epochs. The model learns generic bounds of crime locations with the close score values on the high crime rate regions in the first epochs. Then the score values on the predictions start to distinguish in the 6th epoch. After the 12th epoch, the model makes a detailed forecast. We observe that model binarizes the output predictions, which means the probability distribution generates likelihoods close to 0 or 1. We expect this behavior since we use the BCE loss that makes negative predictions closer to 0 and positive predictions to 1. Furthermore, this causes the PDF to assign close values in responsible regions. To assess the generative property of our model on high resolution, we obtain predictions for a $100 \times 66$ resolution grid from the HRCF model trained on $50 \times 33$. We achieved 36.67 validation and 36.02 F1 scores on test sets, which are 4.77% relatively higher than the trained ConvLSTM model for the resolution of $100 \times 66$.

## 5 Conclusion

We studied the problem of high-resolution crime forecasting and introduced a novel generative graph neural network architecture, HRCF. Our new subdivision algorithm performs balanced sampling on the data to create representative regions. We built our graph according to formed regions, parameterized the problem using a likelihood model, and obtained probability density functions belonging to each region. We evaluated HRCF on a real-world and synthetic datasets, and results showed that our model achieves statistically significant performance gains over the state-of-the-art baseline models. For the future work we plan to apply our model to any spatiotemporal data. We also provide the source code of our algorithm for reproducibility.

## References

[1] Wang, H., *et al.*: Crime rate inference with big data. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining **13-17-Augu**, 635–644 (2016). https://doi.org/10.1145/2939672.2939736

[2] Khan, M.H., *et al.*: Spatiotemporal features of human motion for gait recognition. Signal, Image and Video Processing

**13**(2), 369–377 (2019). https://doi.org/10.1007/s11760-018-1365-y

[3] Patil, A.R., *et al.*: A spatiotemporal approach for vision-based hand gesture recognition using Hough transform and neural network. Signal, Image and Video Processing **13**(2), 413–421 (2019). https://doi.org/10.1007/s11760-018-1370-1

[4] Huang, Q., *et al.*: View transform graph attention recurrent networks for skeleton-based action recognition. Signal, Image and Video Processing **15**(3), 599–606 (2021). https://doi.org/10.1007/s11760-020-01781-6

[5] Deepak, K., *et al.*: Residual spatiotemporal autoencoder for unsupervised video anomaly detection. Signal, Image and Video Processing **15**(1), 215–222 (2021). https://doi.org/10.1007/s11760-020-01740-1

[6] Zhang, J., *et al.*: Dnn-based prediction model for spatio-temporal data. SIGSPACIAL '16. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2996913.2997016

[7] Zhang, J., Zheng, Y., Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. 31st AAAI Conference on Artificial Intelligence, AAAI 2017, 1655–1661 (2017) https://arxiv.org/abs/1610.00081

[8] Wang, B., et al.: Deep Learning for Real Time Crime Forecasting, 33–36 (2017) https://arxiv.org/abs/1707.03340

[9] Huang, C., et al.: MIST: A multiview and multimodal spatial-temporal learning framework for citywide abnormal event forecasting. WWW 2019, 717–728 (2019). https://doi.org/10.1145/3308558.3313730

[10] Wang, B., Luo, X., Zhang, F., et al.: Graph-Based Deep Modeling and Real Time Forecasting of Sparse Spatio-Temporal Data (2018) https://arxiv.org/abs/1804.00684. https://doi.org/10.475/123

[11] Sun, J., *et al.*: CrimeForecaster: Crime Prediction by Exploiting the Geographical Neighborhoods' Spatiotemporal Dependencies. Lecture Notes in Computer Science **12461 LNAI**, 52–67 (2021). https://doi.org/10.1007/978-3-030-67670-4_4

[12] Wang, C., Lin, Z., Yang, X., Sun, J., Yue, M., Shahabi, C.: HAGEN: Homophily-Aware Graph Convolutional Recurrent Network for Crime Forecasting (2021) https://arxiv.org/abs/2109.12846

[13] Wu, Z., *et al.*: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)

[14] Jain, A., *et al.*: Structural-RNN: Deep learning on spatio-temporal graphs. CVPR **2016-Decem**, 5308–5317 (2016). https://doi.org/10.1109/CVPR.2016.573

[15] Salinas, D., *et al.*: DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting **36**(3), 1181–1191 (2020) https://arxiv.org/abs/1704.04110. https://doi.org/10.1016/j.ijforecast.2019.07.001

[16] Kullback, S.: Information Theory and Statistics. Dover Publications Inc., Mineola, New York (1968)

[17] Seo, Y., *et al.*: Structured sequence modeling with graph convolutional recurrent networks. Lecture Notes in Computer Science **11301 LNCS**(2013), 362–373 (2018). https://doi.org/10.1007/978-3-030-04167-0_33

[18] Shi, X., *et al.*: Convolutional lstm network: A machine learning approach for precipitation nowcasting. NIPS'15, pp. 802–810. MIT Press, Cambridge, MA, USA (2015)

[19] Department, C.I..P.: Chicago crimes, 2001-2018. Technical Report ICPSR37256 - v1, Inter-university Consortium for Political and Social Research, Ann Arbor , MI (2019). https://doi.org/10.3886/ICPSR37256.v1