# Predicting Chicago Traffic Accident Risk with Convolutional and LSTM Neural Networks

**Zhuoer Gu**
Stanford University
guzhuoer@stanford.edu

**Connie Hong**
Stanford University
conniehg@stanford.edu

**Isleydys Silva Torrecilla**
Stanford University
isleydys@stanford.edu

## 1 Introduction

Predicting traffic accident rates remains an ongoing challenge due to the complex factors influencing road safety, such as weather, population density, and geography. In this project, we explore the use of deep learning models—specifically Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks—to predict traffic accident rates in the Chicago Metropolitan Region. Unlike traditional regression based methods, this project's approach aims to better capture three key factors of traffic accidents: (1) their temporal nature, (2) spatial distribution, and (3) the role of non-linear and categorical environmental variables.

## 2 Related Work

Several approaches have been proposed for predicting traffic accident rates. Traditional methods based on Logistic Regression, Random Forest, and Decision Trees [7][1] have been widely studied and remain popular for their simplicity and interpretability. However, these traditional methods are not inherently designed for sequential data and fail to handle non-linearity between inputs and outputs, and this, and hence in our project context the ability to capture the dynamics of traffic flow and accident risk over time is limited. In addition, decision trees often overfit to the training data and may not predict well to imbalanced unseen data, which is the case of a traffic accident prediction problem. While random forests reduce overfitting by averaging many decision trees, they still may not be as robust at generalizing complex, non-linear interactions between inputs and outputs.

Modern deep learning approaches, such as Convolutional Neural Networks (CNNs) [5], have demonstrated promising results due to their ability to analyze spatiotemporal correlations and handle complex non-linear interactions. Extensive research in this area has led to the development of advanced models such as a Convolutional Long Short-Term Memory (ConvLSTM) neural network. This model addresses the challenge of spatial heterogeneity in accident prediction by incorporating spatial graph features and spatial model ensemble, effectively capturing spatial heterogeneity, incorporating multi-level spatial partitioning, and utilizing knowledge transfer networks [8]. More ambitious deep learning approaches aim to combine deep learning architectures such as CNNs, recurrent neural networks (RNNs), and hybrid spatiotemporal modeling (CNN-LSTM) to process the spatial and temporal characteristics of large volumes of traffic data [2].

Our work builds on existing CNN and LSTM models, while uniquely combining these neural networks with categorical environmental data. We finally focus on the specific context of Chicago, a city with distinctive and noteworthy environmental factors.

## 3 Dataset Features and Preprocessing Methods

To train our model, we used 523,219 traffic accident reports from the Chicago Data Portal ([4]), spanning from January 1, 2020, to December 2, 2024. The dataset includes both self-reported crash details and those recorded by officers, with information on crash conditions (e.g., road and weather).

### 3.1 Baseline Poisson Regression Model

Our baseline Poisson regression model aimed to predict the annual accident rate at a given time of day (in minutes from midnight). We grouped accident reports into 30-minute intervals starting from midnight, regardless of the day or month. The output variable $y$ represents the total accident count at each time-of-day (in minutes from midnight), while the input variable $x$ corresponds to the midpoint of each 30-minute interval, focusing on temporal trends within the 24-hour window.

### 3.2 CNN-LSTM Model

Our inputs to our deep learning models consists of two forms: spatio-temporal data of previous accident counts, and weather data of the current predicted time interval.

We draw inspiration from spatio-temporal data processing in deep learning techniques ([3], [6]). We define our region of interest by the minimum and maximum latitudes and longitudes of the accident records in our dataset, and partition it into a grid of $l \times p$ cells, where $S = \{s_{1,1}, s_{1,2}, \ldots, s_{l,p}\}$, with each $s_{j,k}$ representing the cell at the $j$-th latitude and $k$-th longitude intervals. We set $l = p = 11$. The time range of our dataset (from January 1, 2020, to December 3, 2024) is then divided into 1-hour intervals. At each interval $t$, the cell $s_{j,k}$ experiences $C(s_{j,k}, t)$ accidents. For binary classification, we define $A(s_{j,k}, t) = \mathbb{1}\left(C(s_{j,k}, t) > 0\right)$ to indicate whether this region $s_{j,k}$ at the time $t$ experienced any accidents.

For each $t$-th time interval, we extract the "ROADWAY-SURFACE-COND", "WEATHER-CONDITION", and "LIGHTING-CONDITION" variables from the crash dataset, and apply one-hot encoding to the categorical variables. We also append the numerical "HOUR" (ranging from 0 to 23) for the corresponding interval. These variables are combined into a $1 \times 23$ array for each $t$-th time interval. If weather data is missing for a given interval, we substitute it with data from the nearest available interval (within 1-2 hours), assuming minimal weather variation between adjacent intervals.

## 4 Methods

### 4.1 Baseline Poisson Regression Model

We fit a Poisson model in the following form:

$$y_t = exp(\beta_0 + \beta_1 x_t) \tag{1}$$

where $y_t$ corresponds to the annual count of accidents that occurred in that interval, and $x_t$ corresponds to the median of the thirty-minute interval in minutes. We will compare the performance of this baseline model with the CNN-LSTM model.

### 4.2 CNN-LSTM Model

Figure 1 illustrates the general scheme of the CNN-LSTM Model. The model takes two inputs: environmental features of the predicted hour $t$, and a sequence of $11 \times 11$ traffic accident maps from the past 24 hours.

The environmental features, represented as a $23 \times 1$ array as described in Section 3.1, are first passed through a three-layer fully connected neural network, which acts as a sub-component of our larger model. Since these features are categorical and are either label-encoded or one-hot encoded, the neural network uses the ReLU activation function to capture the non-linear relationships between the inputs and outputs. The output of this sub-network is a $1 \times 24$ vector, denoted as $E_t$.

Separately, our sequence of traffic accident maps, described in Section 3.2, are passed into the "CNN-LSTM" component of our model. First, the accident map sequence is passed into a convolutional layer. This output is then passed through a pooling layer, flattened into a 1D array, and reshaped to a $24 \times 1$ array using a fully connected layer. This intermediate output is then fed into an LSTM layer with 24 nodes, where each node represents one hour of the 24-hour interval.

Each LSTM neuron (aka cell), as shown in Table 1, takes in an input $X_t$ (with $t \in \{0, 1, ...23\}$ in our case), along with the hidden state $h_{t-1}$ and cell state $c_{t-1}$ from the previous time step. It then outputs a new hidden state $h_t$ and cell state $c_t$, which are passed as inputs to the next neuron corresponding to

the following time step in the input sequence. The hidden state $h_t$ is also passed to the next layer in the neural network. The equations in Table 1 describe the activation functions and operations performed on the neuron's inputs to produce these outputs. Namely, we see, in addition to the input parameters of $X_t, h_{t-1}, C_{t-1}$, the cells requires neural network weights (denoted as $W_f, W_i, W_c, W_o$), bias terms ($b_f, b_i, b_c$). The $\sigma$ and $\tanh$ also correspond to their respective activation functions. Overall, the "sequential" nature of the LSTM layer enables the model to capture temporal dependencies in the input data, namely, how the current accident rate at a given location may be correlated to historical accident rates at the same spot.

| Gate | Equation | Description |
|---|---|---|
| **Forget Gate ($f_t$)** | $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$ | Decides what to forget from the previous state. |
| **Input Gate ($i_t$)** | $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ | Decides new information to store in the current cell state. |
| **Candidate Cell State ($\tilde{C}_t$)** | $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$ | Generates candidate values for updating the current cell state. |
| **Cell State Update ($C_t$)** | $C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$ | Updates the cell state with new and previous information. |
| **Output Gate ($o_t$)** | $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$ | Controls the output of the cell state. |
| **Hidden State Output ($h_t$)** | $h_t = o_t \cdot \tanh(C_t)$ | Output the hidden state for the next time step. |

Table 1: Description of LSTM Gates and Their Functions

The outputs of the two sub-components, $E_t$ and $X_t$, are combined through a Hadamard product, resulting in $E_t \odot X_t$. Since ReLU is used as the activation function for all fully-connected neurons, the activations are non-linear functions of the original inputs. Applying the Hadamard product on $E_t$ and $X_t$ thus preserves the non-linearity. The resulting output is passed through a fully connected layer with 121 nodes. In a regression setting, the activation function for this layer is linear, while in a classification setting, it uses the sigmoid function. In the classification setting, the immediate output of the Fully Connected Layer shall predict the probability of having an accident in every region for the current prediction hour $t$, i.e. $\mathbb{P}(A(s_{j,k}, t) = 1 \mid x)$ with the indicators defined in Section 3.2 and $x$ a combination of input variables as discussed in Section 2. In the regression setting, this layer should output predictions $C(s_{j,k}, t)$. Finally, the output is reshaped into an $11 \times 11$ matrix.

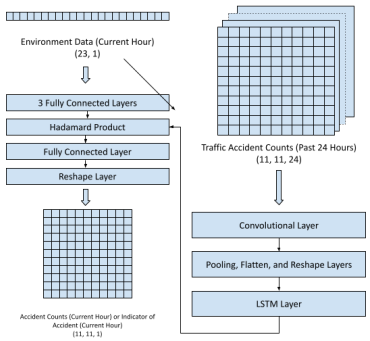


Figure 1: CNN-LSTM General Layout

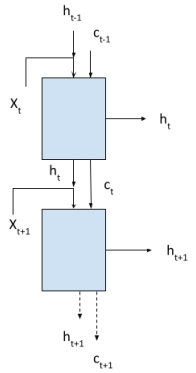


Figure 2: LSTM Layer

## 5 Experiments and Results

### 5.1 Dataset Inputs

The Poisson regression model was trained and tested first on the total dataset of 523,219 accident reports. Next, the accident reports were also mapped to one-hour time intervals and assigned to one of the possible $j \times k$ grids to prepare the data to be inputted into the CNN-LSTM models. This preprocessing step resulted in 43,131 input-output pairs, with predicted hours being incremented by values of 1 hour. Finally, all models employed an 80%-20% train-test split for their respective input datasets. The neural networks were then trained for 50 epochs using stochastic gradient descent.

### 5.2 CNN-LSTM Regression

#### 5.2.1 Experiment Settings

The output layer of this neural network consists of an $11 \times 11$ grid, where each cell represents the predicted counts $\hat{C}(s_{j,k}, t)$ of accidents occurring at that specific cell location $s_{j,k}$ and hour $t$. $C(s_{j,k}, t)$ corresponds to the *true* counts of accidents occur at that cell location and hour. As is typical in regression problems, we train our model using the mean-squared-error (MSE) loss. The MSE loss, adapted to dimensions of the output, is defined as follows:

$$\text{MSE} = \frac{1}{n} \sum_{k=1}^{11} \sum_{j=1}^{11} \sum_{t=1}^{n} \Big( C(s_{j,k}, t) - \hat{C}(s_{j,k}, t) \Big)^2 \tag{2}$$

Namely, in addition to summing our losses over the dataset, we also "flatten" the output array and sum the losses over the cells of the output grid.

#### 5.2.2 Results

| Model | MSE (Train, Test) | MAE (Train, Test) |
|---|---|---|
| *Baseline Poisson Regression* | 193897, 78448 | 266, 220 |
| *CNN-LSTM Regression* | 0.099, 0.112 | 0.127, 0.132 |

Table 2: Model Performance: MSE and MAE for Baseline and ConvLSTM Regression

We observe, in Table 2, that the CNN-LSTM regression model outperforms the baseline Poisson regression model. We see that the baseline model fails to account for the various outliers in the input data.

Furthermore, the baseline model only considers the distribution of accident counts over time, while we hypothesize that the CNN-LSTM model, which also incorporates spatial and weather information, benefits from these additional features, providing more predictive power and improving performance.

### 5.3 CNN-LSTM Binary Classification

#### 5.3.1 Experiment Settings

The output layer of this neural network consists of an $11 \times 11$ grid, where each cell represents the probability of an accident occurring at that specific cell location and hour. The network was trained using the log-loss function, which measures the difference between the predicted and true values for each cell. The loss function (i.e. the negative log-likelihood for binary classification) we aim to minimize, adapted for our $11 \times 11$ output grids, is shown described in the below formula:here, the index $k$ refers to the row, and the index $j$ refers to the column of the grid. $y_{j,k}^{(i)}$ represents the true label, $A(s_{j,k}, i)$, of the $(j,k)$-th cell in the $i$-th output image. The notation $\gamma_{j,k}^{(i)}$ refers to the predicted probability of a positive classification (i.e., the probability that an accident occurs) at the $(j,k)$-th grid location in the $i$-th output image given the inputted data. Note that $\gamma_{j,k}^{(i)}$ is equivalent to $\mathbb{P}(A(s_{j,k}, i) = 1 \mid x)$, with $A$ being the indicators and $x$ representing the combination of input variables used, as discussed in Section 2 and Section 3. A logistic regression threshold of probability 0.5 was used for our model analysis, meaning that predictions with probabilities greater than or equal to 0.5 were labeled as 1, and those with probabilities below 0.5 were labeled as 0.

$$\log[L(\theta)] = \sum_{k=1}^{11} \sum_{j=1}^{11} \sum_{i=1}^{n} y_{j,k}^{(i)} \log \left[ \gamma_{j,k}^{(i)} \right] + (1 - y_{j,k}^{(i)}) \log \left( 1 - \gamma_{j,k}^{(i)} \right) \tag{3}$$

where $n$ is the number of hours recorded in the dataset.

### 5.3.2 Results

The model's results highlight key challenges in performing binary classification on an imbalanced dataset. Most cells in the $11 \times 11$ output "images" correspond to locations without accidents, with far fewer cells representing accident locations. We believe this imbalance may explain some of the observed model outcomes. Table 3 shows the training and testing performance of our model. The test

| Train/Test | Precision | Recall | Accuracy | AUC |
|---|---|---|---|---|
| *Train* | 0.606 | 0.166 | 0.926 | 0.92 |
| *Test* | 0.54 | 0.151 | 0.923 | 0.91 |

Table 3: Model Performance: CNN-LSTM - Classifier

set exhibits optimal precision, recall, accuracy, and AUC(Area under Curve) values of 0.54, 0.151, 0.923, and 0.91, respectively. While the model's accuracy and AUC values appear reasonable, the precision and recall values remain relatively low.

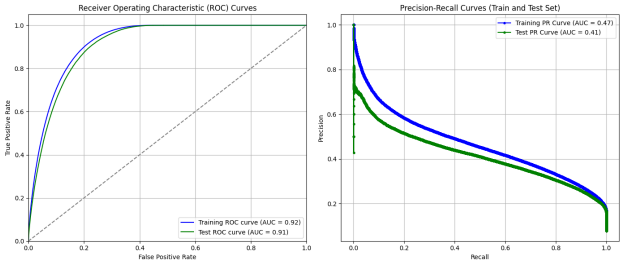


Figure 3: ROC (L) and Precision-Recall (R) Curve

| Metric | Value |
|---|---|
| TN | 951084 |
| FP | 10580 |
| N | 69783 |
| TP | 12420 |

Figure 4: Test Data Metrics

As shown in Figure 4, our dataset exhibits a significant concentration of true negatives, reflecting the imbalance between the negative and positive classes. This results in the model achieving high accuracy by predominantly predicting negative outcomes, but it does not indicate strong performance in identifying positive cases. The over-prediction of the negative class leads to a large number of false negatives, which in turn reduces recall. Although this approach yields a favorable ROC curve, it lowers both recall and precision, as the model struggles to correctly identify accidents.

## 6 Conclusion / Future Work

This work develops predictive deep learning models for the Chicago Metropolitan Region, leveraging CNN and LSTM techniques alongside custom categorical environmental features and one-hot encoding to improve accident rate predictions. Our CNN-LSTM regression model outperforms the baseline Poisson regression model in terms of Test MSE and MAE, highlighting the value of spatiotemporal and environmental data. Additionally, we find that a regression model is more suitable than a binary classification model for this problem, due to the imbalance between accident and non-accident occurrences. Given the multi-faceted nature of traffic accident rate prediction, future work should explore explainable AI methods, such as LIME and SHAP, to gain a deeper understanding of the relationship between predictors and predicted outcomes.

## 7 Contributions

Zhuoer contributed to brainstorming model architectures by researching existing literature and developing techniques to preprocess the data according to its nature, specifically handling categorical inputs.

Connie implemented the code for the preprocessing pipelines, training and evaluating the models using TensorFlow's Keras library, and conducted a literature review focused specifically on CNN-LSTM networks.

Isleydys contributed to brainstorming on initial ideas for developing the project, as well as researching on existing literature on models for accident rate predictions including deep learning models.

All authors contributed to this manuscript.

Our project mentor Ryan Chi provided valuable feedback and extensive help on the baseline model designs, the interpretability of Poisson model, and the usage of LSTM. We are also very grateful for the guidance from other CA's in this class. We received tremendous help from CA's Roshni Sahoo, Paris Zhang and Edward Chen who inspired us to consider the categorical and continuous properties of time-based data in our design, draw comparisons with random forests and gradient boosted trees, and comprehending the motivations behind the designs in cited literature.

All code for this work can be found in this Github repository.

## References

[1] Pires Abdullah and Tibor Sipos. Drivers' behavior and traffic accident analysis using decision tree method. *Sustainability*, 14(18), 2022.

[2] Mohammad Tamim Kashifi, Mohammed Al-Turki, and Abdul Wakil Sharify. Deep hybrid learning framework for spatiotemporal crash prediction using big traffic data. *International Journal of Transportation Science and Technology*, 12(3):793–808, 2023.

[3] Kun-Yu Lin, Pei-Yi Liu, Po-Kai Wang, Chih-Lin Hu, and Ying Cai. Predicting road traffic risks with cnn-and-lstm learning over spatio-temporal and multi-feature traffic data. In *2023 IEEE International Conference on Software Services Engineering (SSE)*, pages 305–311, 2023.

[4] City of Chicago. Traffic crashes (crashes), 2024. Accessed: 2024-12-05.

[5] Honglei Ren, You Song, Jingwen Wang, Yucheng Hu, and Jinzhi Lei. A deep learning approach to the citywide traffic accident risk prediction. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3346–3351, 2018.

[6] Honglei Ren, You Song, Jingwen Wang, Yucheng Hu, and Jinzhi Lei. A deep learning approach to the citywide traffic accident risk prediction, 2018.

[7] R. Vanitha and Swedha M. Prediction of road accidents using machine learning algorithms. *Middle East Journal of Applied Science and Technology (MEJAST)*, 6, 2023.

[8] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, page 984–992, New York, NY, USA, 2018. Association for Computing Machinery.