Proceedings of the 5th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/29/20230785

TrajTransGCN: Enhancing trajectory prediction by fusing transformer and graph neural networks

Haojun Pan

Department of Information Science and Technology, Jinan University, Guangzhou, 511486, China

2499869178@qq.com

Abstract. This paper proposes a novel model named TrajTransGCN for taxi trajectory prediction, which leverages the power of both graph convolutional networks (GCNs) and Transformer. TrajTransGCN first passes the input through the GCN layer and then combines the GCN outputs with one-hot encoded categorical features as input to the transformer layer. This paper evaluates. TrajTransGCN uses real-world taxi trajectory datasets in Porto and compares it against several baselines. The experimental results show that TrajTransGCN outperforms all the other models in terms of both RMSE and MAPE. Specifically, the model achieves an RMSE of 0.0247 and a MAPE of 0.09%, which are significantly lower than those of the other models. The results demonstrate the effectiveness of the proposed model in predicting taxi trajectories, indicating the potential of leveraging both GCN and transformer layers in trajectory prediction tasks. In addition, this paper includes ablation experiments to demonstrate the effectiveness of using one-hot encodings of classification labels in complex real-time scenarios. In addition, a parameter study is carried out to examine how the TrajTransGCN's performance is impacted by the learning rate, the quantity of Transformer layers, and the size of the hidden dimension of the Transformer layer.

Keywords: trajectory prediction, deep learning, transformer, graph convolutional network.

1. Introduction

Trajectory prediction is a crucial task in many applications, such as autonomous driving, pedestrian tracking, and unmanned aerial vehicles [1-3]. The high accuracy of trajectory prediction entitles governments to contribute appropriate portions of investments in construction in different regions. Besides, predicting the trajectory of various vehicles enables autonomous driving vehicles smarter like humans, as they are provided with more valuable data, which is an essential base for smart cities.

Deep learning methods, such as Recurrent Neural Networks, Convolutional Neural Networks (CNNs), and Generative Adversarial Networks, have been extensively used in this field recently [4-5]. Recurrent Neural Networks show a great advantage in historical time series. Convolutional Neural Networks are highly capable of considering trajectory data as a two-dimensional image so that the spatial relationship of objects can be well processed and Generative Adversarial Networks can generate multiple possible trajectories and choose the most likely one.

Processing sequence coordinates through GNNs can utilize their powerful ability to extract and encode features of nodes in the sequence, thus obtaining richer feature representation. The transformer model excels at capturing the temporal correlation between nodes and specializes in processing elements in the sequence in parallel and has a fast-processing speed. It has shown excellent performance in long sequence modeling tasks with its strong parallel computing ability and powerful interpretability.

By combining these two methods, our approach aims to capture both the spatial-temporal dependencies and the relationships between objects in a scene and therefore achieve real-time trajectory prediction in various scenarios that require it. However, these methods often struggle with modeling long-term dependencies and spatial-temporal information.

One limitation of GNNs in trajectory prediction is that the performance of GNN may be affected by the structure of the graph. For example, if the graph structure is highly sparse or contains isolated nodes, GNN may not be able to adequately propagate information to nodes. Additionally, GNN is also susceptible to the influence of noise and outliers in the input data, like the loss and inaccuracy of GPS coordinates when locating the vehicles, which may lead to a decrease in model performance. Although the transformer model has shown promise in fields such as natural language processing and others, it might be restricted in how it treats time-series data. For example, if there are long-term dependencies in the time series, the transformer may not be able to capture them. Additionally, if the distribution of the input data is uneven, appropriate adjustments may need to be made to the transformer model, which requires high skills and a long time.

This paper proposes a novel approach for trajectory prediction that combines the power of Transformer networks and Graph Convolutional Networks (GCNs) for their powerful ability to handle dynamic graph data, scalability, interpretability, and flexibility, developing a Transformer-enhanced Graph Convolutional Network (TrajTransGCN) for Trajectory Prediction.

2. Related work

2.1. Trajectory prediction

Early methods for trajectory prediction were primarily based on algorithms that utilized machine learning, including linear regression, logistic regression, and decision trees[6-8]. These methods typically relied on statistical features of trajectory data, such as mean, standard deviation, maximum, minimum, and similarity between trajectories, to make predictions. Additionally, some rule-based systems, such as path planning based on traffic rules, were used for predicting pedestrian and vehicle trajectories. However, due to the limited data collection and processing capabilities available at the time, trajectory data usually only contained basic location information, such as the starting point, endpoint, and route taken, making it difficult to extract more information for prediction. If inputs of machine learning models are poorly labeled, then the algorithm’s outputs will directly reflect these inaccuracies [9]. Furthermore, the method of data processing is likely to result in multicollinearity of the input data. The existence of multicollinearity in traditional machine learning trajectory prediction can lead to inaccurate parameter estimation, overestimation or underestimation of the effects of explanatory variables, decreased predictive ability and interpretability of the model. It may prevent the model from fully utilizing the information in the trajectory data, thus affecting the accuracy and stability of the prediction results.

With the advancement of deep learning technology, trajectory prediction research is increasingly starting to turn to deep learning techniques. There has been a significant amount of research in the field of trajectory prediction. One of the most popular approaches is to use Recurrent Neural Networks, such as Long Short-Term Memory networks to predict time series data [10]. Another approach is to utilise generative adversarial networks so as to generate a multimodal trajectory prediction model [11-12]. These methods can not only handle high-dimensional, nonlinear, non-stationary, and unconventional trajectory data, but also automatically learn the feature representation and patterns of trajectory data, improving the accuracy and stability of trajectory prediction.

2.2. *Transformer in trajectory prediction*

In the area of trajectory prediction, the Transformer network has become more well-known recently. It was initially used as a translation model in 2017 in Natural Language Processing (NLP) [13]. With its encoder-decoder structure, complete reliance on self-attention, and capacity for parallel computation, it was able to solve the issues of long-term dependencies in input and output while significantly reducing the consumption of computational resources. However, it is right due to the parallel computing capability of Transformer, its ability to capture global information through self-attention, and its capacity to model long-term dependencies in sequences that it has shown excellent performance in sequence modeling tasks and it has enormous potential in various fields. As a result, people have gradually applied Transformers to other fields such as image processing and prediction Trajectory prediction is one of the typical applications of predictive modeling, and trajectory prediction based on the Transformer model has also been thriving. In 2018, Zhang et al. proposed a convolutional sub-network to control each attention head's importance to learn on Large and Spatiotemporal Graphs, which is a pioneer of transformer prediction works [14]. The Graph-based Spatial Transformer in 2022 can predict numerous paths based on a historical trajectory by simulating multiscale graph-based spatial transformers in conjunction with the trajectory smoothing algorithm "Memory Replay" which makes use of a memory graph [15]. Transformer networks have been an ascending trend in the deep learning field.

The attention mechanism has been increasingly applied in the field of trajectory prediction to improve the interpretability of the model. The attention mechanism assigns weights to different parts of the input sequence according to their importance, and the weighted sum of the sequence is then used to make predictions. In trajectory prediction, the attention mechanism permits the model to selectively concentrate on the most relevant parts of the trajectory history, such as areas with high traffic congestion or frequent changes in direction. This helps to improve the accuracy and interpretability of the model, making it easier to understand why certain predictions are made.

2.3. *GNNS in trajectory prediction*

Graph Neural Networks (GNNs) are a family of deep learning models built on graph-structured data, with origins in the PageRank algorithm in graph theory and convolutional operations in Convolutional Neural Networks (CNNs).

Early GNN models were relatively simple, such as the spectral convolution model based on the graph Laplacian matrix. The concept of GNN was first proposed by Marco Gori et al in 2005 [16]. They proposed a novel neural network model that can handle graph inputs that are cyclic, directed, undirected, or even a mixture of these. Nonetheless, how to deal with domains where the linkages, which were not known beforehand, remained to be inferred. With the advent of deep learning, GNN models have, moreover, received a great deal of study and application in recent years. In 2016, Kipf, T. N., & Welling et al simplified the Graph Convolutional Network (GCN) to extract and learn representations of graph-structured data through convolutional operations [17]. A GCN learns node representations by aggregating information from neighboring nodes in the graph, which captures the local structure of the graph. And it can operate on graphs of arbitrary sizes and shapes, making them very flexible and applicable to a wide range of problems. GAT is a type of graph neural network introduced by Veličković et al. in 2018 [18]. It employs a multi-head attention mechanism to aggregate information from neighboring nodes in a graph, allowing it to capture complex and non-linear relationships between nodes.

In the field of trajectory prediction, GNNs have also been widely applied. Socially Acceptable Trajectories with Generative Adversarial Networks (GAN) is a GAN-based trajectory prediction model that uses a GNN-based social pooling operation to model neighboring pedestrians [19]. Combining tools from sequence prediction and generative adversarial networks, it can capture the inherently multimodal human motion. Spatial-Temporal Graph Convolutional Networks is a model of dynamic skeletons based on spatiotemporal graph convolutional neural networks that can predict city traffic flows at specific time intervals [20]. It surpasses the constraints of previous methods by automatically acquiring spatial and temporal patterns from data through autonomous learning.

3. Preliminaries

3.1. Dataset

The dataset includes the trajectories of all 442 taxis operating in the Portuguese city of Porto for the entire year (from 01/07/2013 to 06/30/2014). Using mobile data terminals that have been installed in the cars, these taxis are controlled by a dispatch center for taxis. Each data sample represents a finished trip. Nine features altogether are present. Table 1 provides an illustration of a trip ID and description.

Table 1. One example of a trip ID and descriptions of features.

Feature	Value	Description
TRIP_ID	1372636858620000589	A distinct identifier for each taxi travel
CALL_TYPE	C	How to request this service 'A' (from the central) 'B' (to a taxi driver) 'C' (on a stochastic street)
ORIGIN_CALL	NaN	Whether a phone call is used
ORIGIN_STAND	NaN	Whether a call stand was utilized
TAXI_ID	20000589	A distinguishing mark for the taxi driver
TIMESTAMP	1372636858	Unix Timestamp (in seconds).
DAY_TYPE	A	The daytype on which each trip began.
MISSING_DATA	False	After the GPS data stream is finished, FALSE When one (or more) locations are absent, TRUE
POLYLINE	[[-8.618643,41.141412],[-8.618499,41.141376],…,[-8.620326,41.14251],…,]	A list of GPS coordinates. Each pair of coordinates is also identified by the same brackets as [LONGITUDE, LATITUDE]. The last list item corresponds to the trip's destination while the first one represents its start. The scatters of the first polylines and the last polylines can be clearly seen in Figure 1

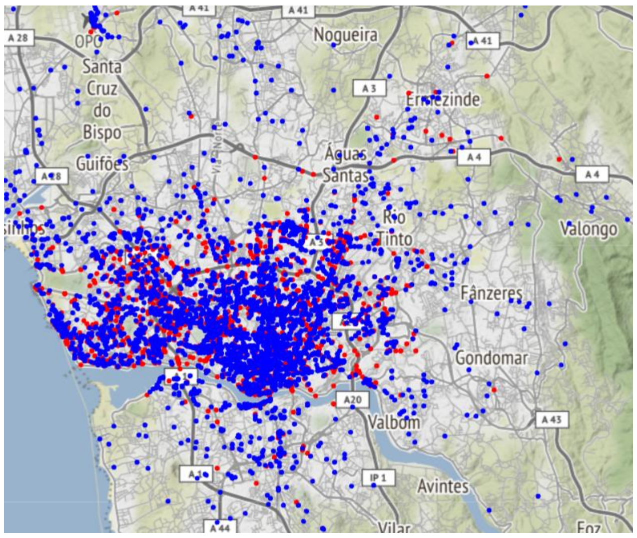

Figure 1. Scatters of the first polylines and the last polylines of each trip.

3.2. Definitions

Definition 1 (Time-series spatial coordinates of Trajectory). This paper partitions $\boldsymbol{N}$ trajectories into a series of coordinates($\boldsymbol{lon}, \boldsymbol{lat}$) for time intervals $\boldsymbol{t} = \boldsymbol{1}, \dots\dots \boldsymbol{T}, \dots, \boldsymbol{T} + \boldsymbol{4}$ and the unit of t is 15 seconds. Each coordinate describes a spatioal region of a taxi. These coordinates are represented as nodes in the graph neural networks and connected in a time-dependent graph structure.

Definition 2 (Categorical features). A set of categorical features $\boldsymbol{C} = \boldsymbol{C_1}, \boldsymbol{C_2}, \dots, \boldsymbol{C_5}$ that describe different characteristics of the geographic area under consideration. These features are processed as one-hot encoding vectors and combined with the GCN layer output of the time series GPS coordinates, and then they enter the transformer layer together.

Definition 3 (Graph). A graph is a mathematical depiction of a collection of things called nodes connected by a collection of edges. Each taxi's GPS point is treated as a node in the TrajTransGCN, and the edges between the nodes indicate their geographic and temporal interactions.

3.3. Problem definition

Taking X consisting of certain categorical features, a sequence of former longitude and latitude values as the input dataset. This paper presents each id in X as:

$$X_i = \{C_{1,i}, C_{2,i}, C_{5,i}, lon_{T,i}, lat_{T,i}, \dots, lon_{1,i}, lat_{1,i}\}, \tag{1}$$

where $C = C_{1,i}, C_{2,i}, \dots, C_{5,i}$ are categorical features which respectively refer to call, location, stand, season, and day types. The problem is to precisely predict the last 4 longitude and latitude coordinates:

$$y_{lon,lat} = \{(lon_{T+1}\ lat_{T+1}), \dots, (lon_{T+4}\ lat_{T+4})\}, \tag{2}$$

where $y_{lon,lat}$ is the predicted trajectory coordinates of all samples and each coordinate consists lon and lat, which refers to longitude and latitude, respectively.

4. Methodology

4.1. Overview

This paper aims to accurately forecast the future trajectory of taxis. Simply expressed, this paper approaches it by meticulous data preprocessing, integrating GCN with the transformer model and taking classification features as additional inputs, and combining the inputs to GCN to enter the transformer layer. This paper also lists some possible applications of the work which contains urban traffic management, ride-hailing and logistics, and so on. The whole process is depicted in Figure 2.

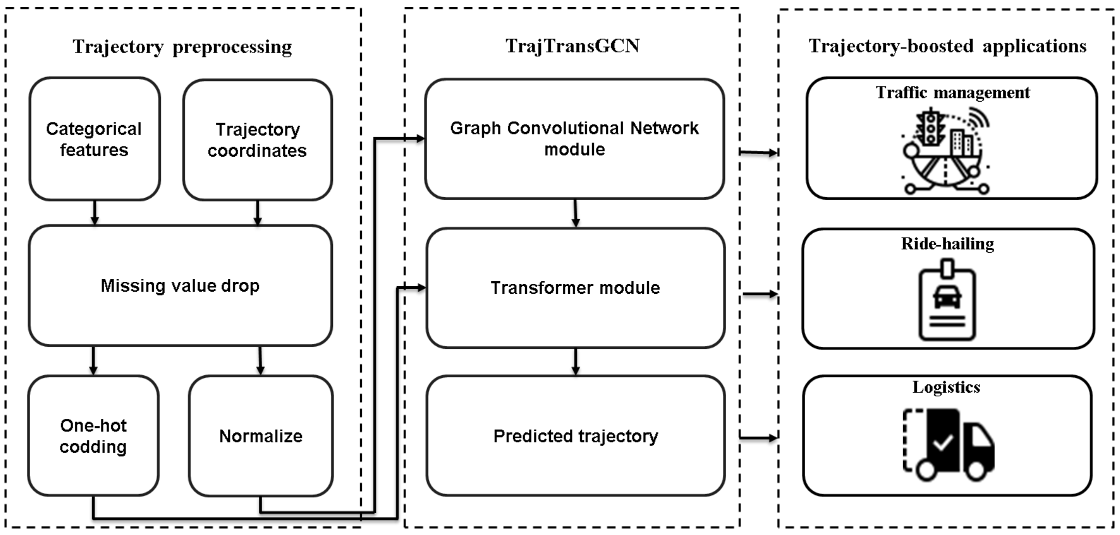

Figure 2. Overall framework.

4.2. Trajectory preprocessing

This paper takes several steps to ensure the quality and integrity of our dataset. First of all, this paper starts with the data cleaning process, where this paper drops the samples with missing values or abnormal values during data cleaning to make the initial quality and the reliability of the data. As for categorical features, this paper categorizes whether a phone call was used, whether a call stand was utilized, and whether data is missing into two categories with values of 0 and 1. Additionally, this paper categorizes call types into three categories with values of 1, 2, and 3, and extracts month and hour information from timestamps. Months (1-12) are classified into four categories based on seasons: spring (1-3), summer (4-6), autumn (7-9), and winter (10-12), while hours (1-24) are categorized into four groups based on time of day: early morning (1-6), morning (7-12), afternoon (13-18), and night (19-24). The features after categorization are summarized in Table 2

Table 2. Features after categorization.

Feature	Value
ORIGIN_CALL	'0' (yes) '1' (no)
ORIGIN_STAND	'0' (yes) '1'' (no)
CALL_TYPE	'1' (from the central) '2' (to a taxi driver) '3' (on a random street)
SEASON	'1' (spring) '2' (summer) '3' (autumn) '4' (winter)
DAY	'1' (before dawn) '2' (morning) '3' (afternoon) '4' (night)

After categorizing the indicators, this paper performs the one-hot encoding on them, whose importance lies in converting categorical features into fixed-length vectors where only one element is 1 and the rest are 0s and ensuring that the distances between different values are equal. These 0-or-1 values are abundant categorical information about the real-time situation, which will later be one kind of input data of transformer layers.

In terms of time-series values, this paper extracts the coordinates (longitudes, latitudes) in the condition of taxis driving for 11 minutes, which is the average travel time for all samples, resulting in 63191 samples of dataset. Due to the GPS positioning interval of taxi trajectory data being 15 seconds, this paper obtains 44 time-series longitude and latitude coordinates in each sample. Then this paper takes out the last 4 time series of longitude and latitude coordinates among these 44 as the prediction values and uses the remaining 40 as a part of the input data.

To construct the adjacency matrix for our graph-based trajectory prediction models, this paper created edges between all pairs of time steps within each sample. Specifically, this paper iterated over each sample, represented by 40-time steps, and created edges between each time step and the next four consecutive time steps, resulting in a total of 1560 edges per sample. The resulting edge list was then used to construct the adjacency matrix for our graph convolutional network (GCN).

Overall, this paper has rich input data: each sample has 40 time series of latitude and longitude coordinates, an adjacency matrix, and one-hot encoded values for the classification indicators, laying a solid foundation for future model training and evaluation.

4.3. TrajTransGCN

This paper proposes a neoteric model in taxi trajectory prediction field using a combination of Transformer and Graph Neural Networks (GNN).

The spatial connections between the GPS locations are recorded using the GNN module. This paper chooses a Graph Convolution Network (GCN) model. By representing the data as a graph, with the locations as nodes and the taxis' movements as edges, GCNs can learn to capture the spatial patterns and dependencies between different locations. Specifically, this paper uses a two-layer GCN to encode the graph structure of the GPS coordinates. The first layer takes the time series GPS coordinates and the related adjacency matrix as input and outputs a hidden representation of size 16, while the second layer further refines the hidden representation to size 2. The output of the GCN is then concatenated with the one-hot encoded values of classification features and fed into the Transformer module.

This paper further processes the trajectory encoding using a Transformer model which is used to capture the temporal dependencies among the GPS coordinates. The Transformer model consists of 1 Transformer Encoder layer where this paper uses one Multi-head Attention layer, two fully connected layers, as well as Layer Normalization and Dropout operations. Combining the result from the GCN layer and the one-hot coding values of categorical features as input, this paper fully leverages the advantages of the transformer model. Finally, this paper uses a fully connected layer to transform the Encoder output into the final prediction result with a dimension of 8.

4.3.1. Graph convolutional network. GCNs are a type of neural network that can handle graph-structured data. They perform convolution operations on graphs by transforming the graph structure into matrix operations. Specifically, GCN's convolutional operation includes

1. Information propagation: according to the graph structure, the features of each node are weighted and summed with the features of its neighboring nodes, i.e., the aggregation of the features of the neighboring nodes. The aggregation method can be a simple weighted sum or a transformation of the neighboring node features followed by summing.

2. Feature transformation: by transforming the aggregated features, each node obtains a new feature representation. The transformation method usually adopts linear transformation, i.e., multiplying the aggregated features with a learnable weight matrix to obtain the new feature representation of the node.

The GCN formulation is rooted on the graph Laplacian matrix, which is defined as:

$$L = D - A, \tag{3}$$

where A is the adjacency matrix of the graph and D is the degree matrix. The GCN layer takes as input the feature vectors of nodes in the graph and propagates them to their neighbors in a message-passing manner. The output of each GCN layer is a new set of feature vectors that captures the updated information of each node in the graph. Since the input data in our GCN model is a time series of GPS coordinates, this paper can stack multiple GCN layers to capture the temporal dependencies of the data. The forward pass of a single GCN layer can be defined as follows:

$$H^{(l+1)} = \sigma\left(\widehat{D}^{-\frac{1}{2}}\hat{A}\widehat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right), \tag{4}$$

where $H^{(l)}$ denotes the feature matrix of the graph at layer l, $\hat{A}$ is the adjacency matrix with self-connections, $\widehat{D}$ is the degree matrix of $\hat{A}$, $W^{(l)}$ is the weight matrix of the l-th layer and $\sigma(\cdot)$ is the activation function.

4.3.2. Transformer. The transformer module of TrajTransGCN consists of TransformerEncoder and a linear layer.

TransformerEncoderLayer is an encoder framework, whose purpose is to encode input features by applying a sequence of operations that includes a Multi-Head Self-Attention mechanism and a Feed-Forward Network. Each TransformerEncoderLayer is made up of a feed-forward network and a multi-head self-attention mechanism. The Multi-Head Self-Attention mechanism, in particular, enables the model to attend to diverse points in the input sequence and capture relationships between them, whereas the Feed-Forward Network transforms the attention output non-linearly.

The self-attention mechanism can be defined as follows:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{5}$$

where $Q, K, and\ V$ are the query, key, and value matrices respectively, and d_k is the dimensionality of the key vectors. This mechanism helps TrajTransGCN to pay attention to various sections of the input sequence, based on their relevance to the current prediction.

The feed-forward function can be defined as:

$$\boldsymbol{FFN}(x) = max(0, xW_1 + b_1)W_2 + b_2, \tag{6}$$

where x is the input vector with features, W_1, b_1, W_2, b_2 are learnable parameters. The overall operation of the TransformerEncoder can be expressed as:

$$TransformerEncoder(x) = LayerNorm(x + Dropout(Attention(x) + x)) + \\ Dropout\ (FFN\ (LayerNorm\ (x + Dropout\ (Attention\ (x) + x)))), \tag{7}$$

where $\mathrm{LayerNorm}(\cdot)$ and and $\mathrm{Dropout}(\cdot)$ are normalization and dropout functions respectively. Transformer model then applies a linear layer to the final encoded features to generate the output. W ithout a specific decoder module, TrajTransGCN is a generative model based on autoregression, where the previous trajectory sequence elements are generated first, and the next element is predicted based on the previous trajectory coordinates, allowing TrajTransGCN to efficiently learn complex dependencies in the data and generate accurate predictions.

5. Experiments

5.1. Dataset

This paper evaluates the performance of TrajTransGCN on a series of experiments over a taxi-trajectory dataset in Porto. After processing, the dataset used in this article contains abundant spatiotemporal information, including time-series coordinate values obtained through GPS positioning and categorical indicator values encoded through one-hot encoding. Table 3 shows the overall descriptions of the processed dataset.

Table 3. Statistics about the final dataset.

Trajectories	Record time (min)	Taxi ID numbers	Avg traj length (km)	Time features	Spatial features
63190	11	440	4.038	15	80

In this study, the training set is randomly chosen from the dataset at 80%, and the test set is chosen from the remaining 20%.

5.2. Baselines

This paper compares TrajTransGCN with the 4 baselines to see the performance.

MLP [21]: a classic neural network model used to solve regression problems.

LSTM [22]: a common type of recurrent neural network model that addresses the vanishing gradient problem in traditional recurrent neural networks by using specialized neurons.

SVR [23]: a support vector machine regression approach based on historical trajectory data to predict future positions

GBRT [24]: a gradient boosting regression algorithm to fit non-linear relationships between historical trajectory data and future trajectory predictions

5.3. Evaluation metrics

Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) are two commonly utilized metrics in assessments to assess the efficiency of models in trajectory prediction accuracy. The average deviation between the values that were anticipated and those that were actually obtained is measured by RMSE, which is easily interpreted. Often used to assess the relative inaccuracy between expected and actual data, MAPE measures the average percentage difference between the predicted and actual values. More prediction accuracy and decreased RMSE are indicators of lesser prediction mistakes while more prediction stability and a smaller relative error are both indicated by a lower MAPE. Hence, the predictive power and stability of TrajTransGCN may be thoroughly assessed in trajectory prediction experiments utilizing both RMSE and MAPE. The equations are shown below:

$$\boldsymbol{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}} \tag{8}$$

$$\boldsymbol{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{9}$$

where $\hat{y}_i$ is each predicted coordinate in time series and y_i is each actual trajectory coordinate.

5.4. Experimental results

The experimental results presented in Table 4 reveal that the proposed TrajTransGCN achieves the best performance among all the evaluated models in terms of both RMSE and MAPE. Specifically, the RMSE and MAPE values of our model are 0.0247 and 0.09%, respectively, which are significantly lower than those of the other models.

The results demonstrate the effectiveness of the proposed TrajTransGCN in predicting taxi trajectories. Compared to the LSTM, MLP, SVR, and GBRT models, our model achieves more accurate predictions. The subpar result of the MLP model is due to its inability to capture temporal dependencies. The SVR and GBRT models are based on regression techniques and perform reasonably well, but they are still outperformed by TrajTransGCN. The LSTM model, which is a popular choice for sequential prediction tasks, also lags behind our model in terms of accuracy.

The success of our model can be attributed to the combination of GCN and Transformer layers, which capture both spatial and temporal dependencies in the input data. The GCN layer is used to model the spatial relationships among the taxi trajectories, while the Transformer layer leverages the temporal dependencies of the trajectories. The joint use of these two layers enables our model to capture both local and global patterns in the data, leading to more accurate predictions.

Table 4. Experimental results.

Model	RMSE	MAPE
LSTM	0.0255	0.13%
MLP	1.14	3.35%
SVR	0.04	0.22%
GBRT	0.027	0.153%
TrajTransGCN	0.0247	0.09%

5.5. Ablation study

The ablation study conducted in this paper aims to further investigate the impact of removing specific modules from the proposed TrajTransGCNon its performance:

No one-hot coding does not perform one-hot coding on classification features and only uses formal values of classification features.

GCN-only simply uses the time-series coordinates as the structure of the graph and does not adopt transformer layers.

Transformer-only does not adopt GCN layers and makes all features as input at the same time.

Table 5 reports the performances of three different models that are compared to the TrajTransGCN

Table 5. Ablation study results.

Model	RMSE	MAPE
No one-hot coding	28.2523	2.03%
Transformer-only	0.0275	0.12%
GCN-only	0.0567	0.43%
TrajTransGCN	0.0247	0.09%

As shown in Table 5, “No one-hot coding” that does not use one-hot encoding for the classification features achieved the lowest performance with an MAPE of 2.03% and a RMSE of 28.2523, which are both dramatically higher than others. This indicates that the one-hot encoding of the classification features is an essential component that helps improve the prediction accuracy. "Transformer-only" that uses only transformer layers achieves better performance than "GCN-only" that uses only the time-series coordinates as the structure of the graph. "Transformer-only" model achieves an RMSE of 0.0275 and a MAPE of 0.12%, while "GCN-only" model achieves an RMSE of 0.0567 and a MAPE of 0.43%. It shows the importance of the diversity of the spatial and temporal information. However, the proposed TrajTransGCN that combines both GCN and transformer layers and incorporates one-hot encoding for the classification features obtains the finest performance among all models. The TrajTransGCN achieves an RMSE of 0.0247 and a MAPE of 0.09%, which outperforms all other models in terms of prediction accuracy. These results suggest that combining both GCN and transformer layers and incorporating one-hot encoding for the classification features can help to effectively leverage the spatial-temporal details in the data as well as improve the prediction accuracy.

5.6. Parameter study

The purpose of this parameter study is to examine how various hyperparameters may affect the effectiveness of the proposed TrajTransGCN. This paper especially emphasis on the impact of 3 hyperparameters: learning rate, number of transformer layers, and transformer hidden dimension size. To achieve this, this paper varies each hyperparameter separately while fixing the other two hyperparameters, comparing training loss over 50 training epochs to see the performances of different parameters. For each hyperparameter setting, this paper calculates the training loss every 10 epochs. The training loss results are recoreded in the form of a list and later transformed into graphs to see the overall tendancy. The parameter study is conducted by performing the experiments whose range of each parameter is depicted in Table 6

Table 6. Ranges of different parameters.

Hyperparameter	range
Learning rate	[0.1,0.05,0.025,0.01,0.001,0.0001,0.00001,0.000001,0.0000001,0.00000001]
Transformer layers	[1,2,3,4,5,6,7,8,9,10]
Transformer hidden dims	[16,32,64,128,256,512]

5.6.1. Effect of learning rates. The learning rate controls the size of the step taken in each update of the model's parameters during the optimization process. It is one of the most important hyperparameters in deep learning, as it directly impacts the convergence speed and performance of the model. Figure 3 shows the effect of different learning rates while training, where we can pinpoint that the learning rate of 0.0001 performs best in training, and training loss explodes with the learning rate descending, manifesting that a lower learning rate can lead to slow convergence or even failure to converge.

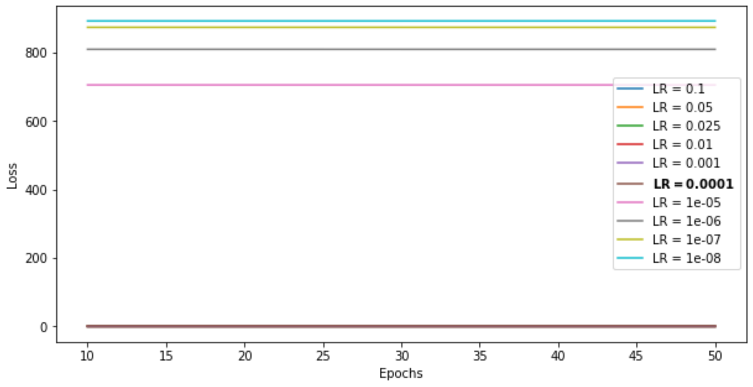

Figure 3. Training loss of all learning rates 0.1 to 0.0001.

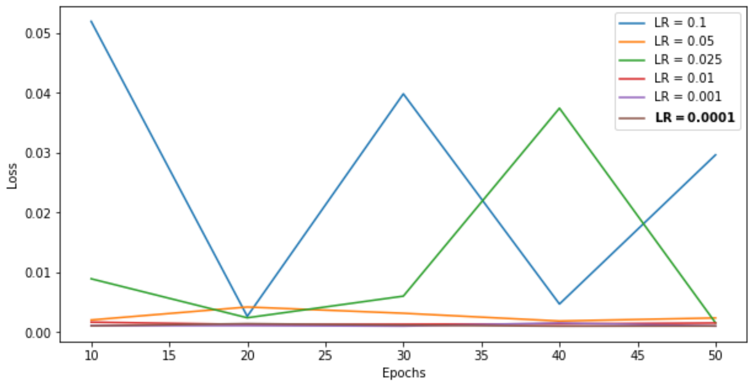

Figure 4. Training loss of learning rates ranging from 0.1 to 0.0001.

Figure 4 illustrates that, in an appropriate range, the optimization process of TrajTransGCN may oscillate, resulting in fluctuations during training epochs under the condition of a higher learning rate. The line of 0.0001 learning rate is the most smooth and has a down tendency, revealing the improvement of weights to have a better performance.

5.6.2. Effect of transformer layers. A deeper transformer design can capture more complicated patterns and dependencies in the input data, but it also raises the danger of overfitting and slows down training. The number of transformer layers is another crucial hyperparameter in models. The notation "NL" is used in the legend of Figure 5 to denote the "number of transformer layers" and displays the training loss of TrajTransGCNs with various transformer layer counts.

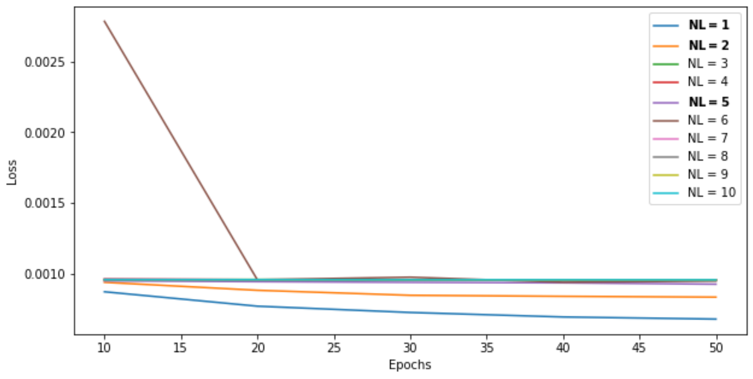

Figure 5. Training loss of transformer layers.

In figure 5, it is worth mentioning that TrajTransGCN with 1 transformer layer performs best when training, which can help decrease the complexity of the module and the possibility of overfitting. TrajTransGCNs with transformer layers 2 and 5 perform relatively better than others, which may be effective in some different situations.

5.6.3. Effect of transformer hidden dimension size. The hidden dimension size is another hyperparameter that is able to have a significant impact on the performance of a model. The hidden dimension size determines the number of features that the model can learn, and it directly affects the model's ability to represent the input data. A hidden dimension size which is deeper can increase the model's capacity to learn complicated patterns, but it is also likely to lead to overfitting and slow down the training. A smaller hidden dimension size can result in a simpler model with better generalization, but it may not be able to capture all relevant features in the input data. This paper adjusts the transformer hidden dimension sizes to see whether TrajTransGCN can well handle the spatial-temporal information as one-hot coding of categorical features is new inputs of transformer layers. Figure 6 shows the training loss of TrajTransGCNs with different sizes of transformer hidden dimension where the abbreviation "HD" is used in the legend to represent "hidden dimension".

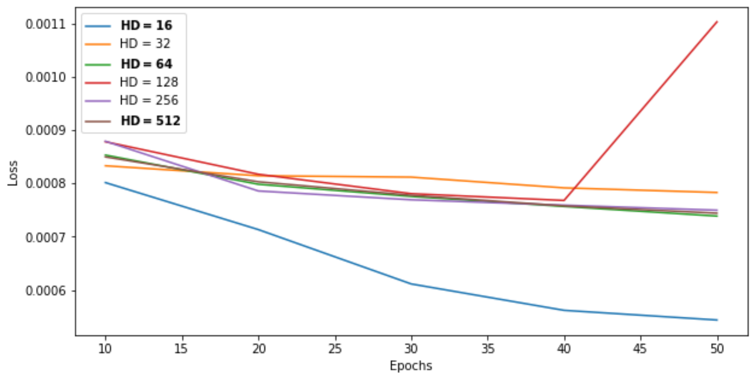

Figure 6. Training loss of transformer hidden dimension sizes.

Figure 6 reveals that TrajTransGCN performs best when transformer hidden dimension sizes are 16, which shows predominance over others in training.

6. Conclusion

This paper proposes a model named TrajTransGCN for trajectory prediction with abundant spatial-temporal information, which combines GCN and Transformer layers. TrajTransGCN takes the output of the GCN layer and the one-hot coding of the classification label as the input of the Transformer layer. The experiments reveal that TrajTransGCN provides better performance than the baseline model in terms of both RMSE and MAPE, indicating the effectiveness of our proposed model. This paper conducts an ablation study, indicating the contribution of the one-hot coding of the classification features, and highlighting the importance of incorporating time information in trajectory prediction. A parameter study is performed to investigate the effects of the learning rate, the number of transformer layers, and the hidden dimension size. The optimal parameter settings are found to be a learning rate of 0.0001, one transformer layer, and a hidden dimension size of 16.

References

[1] Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 2012, pp. 3354-3361, doi: 10.1109/CVPR.2012.6248074.

[2] D. -E. Kim and D. -S. Kwon, "Pedestrian detection and tracking in thermal images using shape features," 2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Goyangi, Korea (South), 2015, pp. 22-25, doi: 10.1109/URAI.2015.7358920.

[3] Y. Zeng, R. Zhang and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," in IEEE Communications Magazine, vol. 54, no. 5, pp. 36-42, May 2016, doi: 10.1109/MCOM.2016.7470933.

[4] H. Shafienya and A. Regan, "4D Flight Trajectory Prediction based on ADS-B data: A comparison of CNN-GRU models," 2022 IEEE Aerospace Conference (AERO), Big Sky, MT, USA, 2022, pp. 01-12, doi: 10.1109/AERO53065.2022.9843822.

[5] S. Liu, H. Liu, H. Bi and T. Mao, "CoL-GAN: Plausible and Collision-Less Trajectory Prediction by Attention-Based GAN," in IEEE Access, vol. 8, pp. 101662-101671, 2020, doi: 10.1109/ACCESS.2020.2987072.

[6] Gaffney, S., & Smyth, P. (1999, August). Trajectory clustering with mixtures of regression models. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 63-72).

[7] Oh, C., & Kim, T. (2010). Estimation of rear-end crash potential using vehicle trajectory data. *Accident Analysis & Prevention*, *42*(6), 1888-1893.

[8] Türkcan, S., & Masson, J. B. (2013). Bayesian decision tree for the classification of the mode of motion in single-molecule trajectories. *PloS one*, *8*(12), e82799.

[9] Chan, S., Reddy, V., Myers, B., Thibodeaux, Q., Brownstone, N., & Liao, W. (2020). Machine learning in dermatology: current applications, opportunities, and limitations. *Dermatology and therapy*, *10*, 365-386.

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

[11] Goodfellow IJ, Pouget-Abadie J, Mirza M (2014) Generative adversarial networks. Adv Neural Inf Process Syst 3(11):2672–2680

[12] Sadeghian A, Kosaraju V, Sadeghian A, Hirose N, Rezatofighi H, Savarese S (2019) Sophie: an attentive gan for predicting paths compliant to social and physical constraints. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

[13] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[14] Zhang, J., Shi, X., Xie, J., Ma, H., King, I., & Yeung, D. Y. (2018). GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*.

[15] Li, L., Pagnucco, M., & Song, Y. (2022). Graph-based spatial transformer with memory replay for multi-future pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2231-2241).

[16] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, *20*(1), 61-80.

[17] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[18] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[19] Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., & Alahi, A. (2018). Social gan: Socially acceptable trajectories with generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2255-2264).

[20] Yan, S., Xiong, Y., & Lin, D. (2018, April). Spatial temporal graph convolutional networks for skeleton-based action recognition. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

[21] Karlik, B., & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems, 1*(4), 111-122.

[22] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation, 12*(10), 2451-2471.

[23] Castro-Neto, M., Jeong, Y. S., Jeong, M. K., & Han, L. D. (2009). Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications, 36*(3), 6164-6173.

[24] Li, Y., Zheng, Y., Zhang, H., & Chen, L. (2015, November). Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems* (pp. 1-10).