Journal of Theoretical and Applied Information Technology
30th November 2025. Vol.103. No.22
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CUSTOM REGRESSION-BASED ARCHITECTURES FOR REAL-TIME TRAJECTORY PREDICTION USING SPATIO-TEMPORAL DATA

PRAVEEN KUMAR .V .S[1] , SAJIMON ABRAHAM[2] , SIJO THOMAS[3], SABU AUGUSTINE[4]

[1]Associate Professor, SAS SNDP Yogam College, Department of Computer Science , India
[2]Professor and Research Supervisor, Mahatma Gandhi University, School of Computer Science, India
[3]Research Scholar, Mahatma Gandhi University, School of Computer Science, India
[4]Professor, Amal Jyothi College of Engineering, Basic Science Department, India

E-mail: [1]praveenplavila@yahoo.co.in, [2] sajimabraham@rediffmail.com, [3]sijothakadiyel@gmail.com, [4]augsabu@gmail.com

ABSTRACT

Trajectory prediction plays a critical role in applications such as autonomous navigation, human motion forecasting, and robotic path planning. However, despite its importance, the scarcity of clean, large-scale datasets and the high bandwidth cost of continuous polling limit practical deployment in real-world intelligent systems. This study proposes a comparative evaluation framework using a hybrid dataset that integrates real and simulated user travel trajectories. The performance of seven regression models—LSTM, Bidirectional LSTM (Bi-LSTM), GRU, 1D CNN, Transformer, SVM, and XGBoost—is systematically analyzed using key metrics including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared (R^2), and Symmetric MAPE (SMAPE). The goal of this study is to benchmark these models on hybrid trajectory datasets and to test the hypothesis that sequence-aware deep models (Bi-LSTM, GRU) outperform traditional learners (SVM, XGBoost), with SMAPE providing clearer insights for low-value targets. Among these, the Bi-LSTM model demonstrates superior performance, particularly in terms of MAE and SMAPE. Additionally, the study explores hyperparameter tuning techniques, statistical validation, and the challenges of combining real and synthetic data, highlighting the effectiveness of SMAPE in low-value target scenarios. The findings not only support the use of Bi-LSTM for enhanced trajectory prediction accuracy in hybrid data environments but also underline the broader potential of predictive intelligence in reducing data transmission overhead and improving the efficiency of real-time mobility systems. These findings have direct implications for smart mobility and surveillance applications, where reducing polling frequency without compromising accuracy is essential. This study directly addresses the critical challenge of accurate real-time trajectory prediction under conditions of data scarcity by integrating both real and simulated datasets. The central goal is to design and benchmark regression-based architectures for short-term forecasting and to test the hypothesis that prediction accuracy can be significantly enhanced by incorporating temporal dependencies within spatio-temporal data.

Keywords: *Trajectory Prediction, Bidirectional LSTM (Bi-LSTM), Real And Simulated Data, Regression Models, Performance Evaluation Metrics, SMAPE*

1. INTRODUCTION

The prediction of movement trajectories has emerged as a critical capability across a broad range of applications, including the navigation of autonomous vehicles, human motion forecasting, air traffic control, and maritime navigation systems [1]. In modern intelligent systems, forecasting future positions of moving entities is essential for efficiency and safety. With the rapid advancement of sensor technologies and data collection mechanisms such as GPS, Automatic Identification Systems (AIS), and radar, vast amounts of spatio-temporal trajectory data are now available for analysis. These developments have opened new avenues for applying advanced machine learning and deep learning models to understand, interpret, and forecast dynamic movement patterns of objects and users in various domains [1]. In addition to these domain-specific uses, trajectory prediction is becoming central to smart mobility and Industry 5.0 initiatives, where accurate short-term forecasting enables real-time decision-making and resource optimization.

Despite the growing availability of real-world trajectory datasets, numerous challenges still exist in obtaining high-quality, large-scale data. Real-world data often suffer from issues such as missing values, noise due to hardware limitations, and limited coverage in certain geographical or contextual settings. To address these limitations, researchers frequently incorporate simulated data into their modeling frameworks, which helps in expanding the dataset size, introducing variability, and testing the models under a broader range of scenarios [3]. Simulated data, when properly validated, can mimic realistic movement behaviors and supplement real data, thus improving model robustness and generalizability in real-world applications [3].

The primary goal of this study is to benchmark diverse regression-based models on hybrid trajectory datasets, overcoming data scarcity while ensuring robustness under realistic mobility conditions. In this context, trajectory forecasting becomes particularly critical in **surveillance systems**, where continuous and timely monitoring of moving objects is essential. However, constant collection of precise location data from objects in motion can lead to considerable data transmission overhead, computational strain, and privacy concerns. To mitigate this, the present study aims to **predict the immediate latitude and longitude of tracked entities**, thereby reducing the frequency of real-time location polling. This not only minimizes network load and energy consumption but also enables intelligent tracking where predictive intelligence substitutes for raw sensor data. To meet the specific requirements of this application domain, we developed **customized model architectures** based on existing frameworks. These custom models are optimized to learn short-term spatial patterns while being computationally efficient and robust to both real and synthetic data inputs.

This study aims to present a comprehensive comparison of regression-based models commonly employed for trajectory prediction, particularly emphasizing their effectiveness when trained on combined real and simulated travel data. By leveraging both types of data, we aim to overcome data scarcity and ensure that the predictive models are well-equipped to handle the inherent variability and complexity of trajectory patterns. The models examined in this research include a mix of traditional machine learning techniques and modern deep learning architectures, such as Long Short-Term Memory (LSTM), Bidirectional LSTM (bi-LSTM), Gated Recurrent Units (GRU), Support Vector Machines (SVM), XGBoost, 1D Convolutional Neural Networks (CNN), and transformer models. These models were selected based on their proven capabilities in time-series forecasting and sequential data processing.

A critical aspect of this research is the evaluation of these models using diverse performance metrics that assess not only accuracy but also robustness and error behavior. Key metrics include Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared (R^2), Mean Absolute Percentage Error (MAPE), and Symmetric Mean Absolute Percentage Error (SMAPE). These metrics enable a multi-faceted evaluation of prediction quality, helping to identify the most suitable models for trajectory prediction tasks under varied data conditions. The bi-LSTM model, in particular, is highlighted for its superior performance, which is attributed to its ability to capture bidirectional temporal dependencies within trajectory sequences.

Study Goal & Hypothesis

This research hypothesizes that sequence-aware deep learning models (Bi-LSTM, GRU) will deliver higher accuracy than traditional machine learning approaches (SVM, XGBoost), and that SMAPE provides a more reliable indicator of performance when predicted values are small.

Problem Statement

Accurate short-term trajectory prediction remains a challenge due to the scarcity and noise of real-world datasets, coupled with the high cost of continuous location polling. This problem becomes critical in surveillance and mobility applications, where timely prediction is essential for reducing bandwidth overhead and improving system efficiency.

Objective of the Study

The main objective of this research is to design and evaluate regression-based models for predicting the immediate spatial coordinates of moving entities. By using a hybrid dataset that integrates real and simulated trajectories, the study aims to demonstrate that predictive intelligence can serve as a reliable alternative to raw sensor polling.

Research Questions and Hypotheses

- **RQ1:** Which regression model provides the highest accuracy for short-term trajectory prediction on hybrid datasets?

- **RQ2:** Can relative error metrics such as SMAPE provide deeper insight into model performance in low-magnitude prediction scenarios?

Hypotheses:

- **H1:** Sequence-aware deep models (e.g., Bi-LSTM, GRU) outperform traditional models (e.g., SVM, XGBoost) in capturing spatio-temporal dependencies.

- **H2:** SMAPE is more effective than absolute error metrics for distinguishing performance when target values are small.

- **H3:** GRU offers a favorable trade-off, achieving accuracy close to Bi-LSTM with reduced computational complexity.

Contributions of the Study

This work makes four major contributions:

1. Proposes customized regression-based architectures (LSTM, Bi-LSTM, GRU, 1D CNN, Transformer) optimized for short-term trajectory prediction.

2. Introduces a hybrid dataset protocol that combines real and simulated trajectories to enhance generalization under data-scarce conditions.

3. Provides a systematic multi-metric evaluation (MSE, MAE, RMSE, R^2, SMAPE) to assess both absolute and relative prediction accuracy.

4. Demonstrates the practical impact of Bi-LSTM and GRU models for deployment in real-time surveillance and continuous query environments.

5. Demonstrates practical implications by showing how next-step prediction reduces data transmission frequency, enabling efficient real-time surveillance and mobility management.

2. RELATED STUDIES

2.1 Long Short-Term Memory (LSTM) Networks

The architecture of an LSTM cell is characterized by its unique ability to handle long-term dependencies in sequential data through the incorporation of input, forget, and output gates, as well as a cell state.[39] These gates regulate the flow of information into and out of the cell state, allowing the network to selectively remember or forget information over extended sequences, a crucial feature for modeling the temporal dynamics of trajectories. LSTM networks exhibit several strengths in capturing temporal dependencies within sequential data, making them well-suited for trajectory prediction tasks.[1] Research has shown that LSTM can effectively model the dynamic dependencies between consecutive points in a trajectory sequence, leading to enhanced prediction accuracy.[8] For instance, an LSTM[1] network can analyze the speed and direction of a moving object at one time step to predict its future movement, considering the influence of past states.[8] In the aviation domain, LSTM has proven effective in capturing long-term patterns[62] in flight data, which is essential for predicting trajectories influenced by factors such as weather conditions or pre-planned flight routes over extended periods.[2] Furthermore, LSTM's capability to learn general patterns of human movement enables it to predict future trajectories, a vital feature for applications involving autonomous robots navigating in environments shared with humans.[18] Extensions of the standard LSTM, such as Social LSTM, have successfully incorporated social interactions in crowded scenarios by allowing the model to consider the influence of nearby agents on an individual's movement.[1] Despite their advantages, LSTM networks also present certain weaknesses in the context of trajectory prediction.[8] One limitation is the challenge in handling excessively long sequences due to the inherent memory constraints of the architecture.[8] While LSTM addresses the vanishing gradient problem prevalent in traditional RNNs, very long sequences can still strain its capacity to retain relevant information across the entire history. Additionally, if the training dataset is imbalanced, with a disproportionate number of

examples for certain trajectory types, the LSTM model may develop a bias towards predicting the more frequent types.[44] LSTM can also struggle with capturing periodic patterns that span very long durations unless the sequence length considered during training is sufficiently extensive.[44] Moreover, trajectories exhibiting strong trends might not be inherently well-modeled by LSTM, potentially requiring preprocessing steps such as trend removal (e.g., by differencing the data) to achieve better prediction performance.[44]
The research material provides numerous examples of LSTM's application in movement prediction across various domains.[1] These applications include flight trajectory prediction by considering the dynamic dependencies of neighboring points and combining LSTM with CNNs[49] for extracting both spatial and temporal features.[2] In maritime navigation, LSTM is used for ship trajectory prediction[32] based on AIS data.[12] For ground movement, LSTM has been applied to pedestrian trajectory prediction in crowded scenarios, taking into account social interactions [1], and to vehicle trajectory prediction[58] in autonomous driving systems.[9] Furthermore, LSTM has been used for destination prediction[64] based on travel data [16], for forecasting stock price[55] movements in financial markets [40], and for enabling robot navigation in dynamic environments.[41] Additionally, LSTM has found application in short-term traffic forecasting.[15]

3.2 Bidirectional LSTM (bi-LSTM) Networks

Bidirectional LSTM networks represent an extension of the standard LSTM architecture[63], designed to capture temporal dependencies in both forward and backward directions within a sequence.[3] This is achieved by employing two parallel LSTM layers: one that processes the input sequence from the beginning to the end (forward direction) and another that processes the sequence from the end to the beginning (backward direction). By considering information from both past and future contexts at each time step, bi-LSTM networks can develop a more comprehensive understanding of the temporal characteristics of trajectory data.
Bi-LSTM networks offer several advantages for trajectory prediction by capturing both past and future context, leading to improved accuracy.[3] This bidirectional processing allows for a more comprehensive understanding of the temporal characteristics of trajectory data by considering both forward and backward orders.[5] In various applications, bi-LSTM has demonstrated better performance compared to unidirectional LSTM.[3] For instance, in vessel trajectory prediction[27] using AIS data and in power consumption forecasting[65], bi-LSTM has shown superior results. The architecture enhances the ability to capture complete information about the past and future of each moment in the input sequence.[13] However, bi-LSTM networks also have certain weaknesses in the context of trajectory prediction. They may struggle with instantaneous trajectory prediction scenarios where sufficient observation time is lacking, as the model typically benefits from seeing both past and future segments of the trajectory. Additionally, in strictly causal time series prediction tasks, where future information should not influence the present, the use of backward processing might raise concerns, although techniques exist to utilize bi-LSTM for feature extraction up to the current time step without violating causality.[69]
The research material showcases numerous applications of bi-LSTM in movement prediction.[3] These include incorporating attention mechanisms for enhanced feature extraction in trajectory prediction [5], vessel trajectory prediction using AIS data[28], often outperforming other models [3], aircraft trajectory prediction by capturing long-term dependencies [5], traffic flow prediction [15], human trajectory prediction in crowded spaces [43], and prediction of stock price movements.[60] Bi-LSTM has also been used for privacy preservation in trajectory data release [13] and for moisture content prediction in drying processes.[67]

3.3 Support Vector Machines (SVM)

Support Vector Machines (SVM) are a class of supervised learning algorithms primarily known for their effectiveness in both classification and regression tasks.[12] In the context of regression, Support Vector Regression (SVR) aims to find an optimal hyperplane that best fits the data within a specified margin of tolerance. SVM operates by mapping data into a high-dimensional space and then finding a linear hyperplane that maximizes the margin, which is the distance between the hyperplane and the support vectors, the data points closest to the hyperplane.[20] For non-linear regression problems, SVM utilizes kernel functions to implicitly map the data into a higher-dimensional feature space where a linear relationship can be modeled.[20] SVM has found applications in time series data and trajectory analysis across various domains.[12] In chemometrics, SVM has been effective for both classification and regression tasks, including forecasting and studying[61] the behavior of key process parameters over time. Its ability to handle non-linear data through kernel functions makes it suitable for complex time series scenarios, where it has demonstrated superior performance compared to traditional linear techniques like ARMA models. Furthermore, SVR exhibits robustness against outliers due to its focus on support vectors, which are the most informative data points defining the hyperplane. SVM has been applied to forecast currency pair trends in financial markets, effectively capturing long-term movements and unexpected market shifts. In remote sensing, SVM regression models are utilized to identify urban area hotspots from satellite-derived nighttime light intensity data.[19] For maritime applications, SVR has been proposed for ship trajectory prediction[54] using AIS data, often with parameter optimization using heuristic algorithms to enhance prediction accuracy.[12] SVM has also been used for lane change trajectory prediction in the automotive domain and for classifying vessel trajectories based on features extracted from AIS data.[21] Additionally, SVM has been applied in biological studies for the classification of moving keratocyte cells. While SVM is not inherently designed for sequential data, it can be adapted to handle the sequential aspects of trajectory data through various techniques.[21] One common approach involves extracting relevant features from the sequential data, such as speed and acceleration, which can then be used as input to the SVM model. Another method involves treating segments of the trajectory as individual data points and using these segments as kernels within the SVM framework to establish similarity between trajectories.[21] SVM has been successfully employed for both short-range and long-range trajectory prediction by modeling trajectories as discrete time series using actual field data. Furthermore, a visual approach involves transforming plots of trajectories into images, allowing the trajectory classification problem to be tackled using SVM-based image recognition techniques. [22]

3.4 XGBoost

XGBoost, which stands for Extreme Gradient Boosting, is a highly effective and popular machine learning algorithm based on the gradient boosting framework. It is an ensemble learning method that combines the predictions of multiple weak learners, typically decision trees, sequentially, with each new tree attempting to correct the errors made by the previous ones. [2]XGBoost enhances the standard gradient boosting algorithm by incorporating regularization techniques to prevent overfitting and by implementing parallel processing to improve computational efficiency. XGBoost has found significant application in time series regression and trajectory analysis across various domains. For time series forecasting, [59]XGBoost[46] is effective when the sequential data is transformed into a supervised learning problem by using lagged values of the time series as input features to predict future values. Its ability to handle non-linear relationships and its computational efficiency have made it a popular choice for various prediction tasks. In transportation, XGBoost has been used to predict freeway travel time based on probe vehicle data, outperforming other gradient boosting methods. It has also been applied in the health domain, showing superior performance over ARIMA models in predicting COVID-19 cases. For autonomous driving, XGBoost has been successfully used for vehicle trajectory prediction in connected and autonomous vehicle environments, often outperforming traditional models like the Intelligent Driver Model (IDM). In finance, XGBoost has been applied to forecast stock prices using historical data and various technical indicators. One of the notable advantages of XGBoost is its ability to provide insights into the importance of different features in the prediction process. By analyzing the feature importance scores, researchers can gain a better understanding of which variables, such as longitudinal position, speed, or time-related features, have the greatest influence on the vehicle trajectory prediction results. This interpretability aspect is valuable for understanding the underlying dynamics of the movement data.

3.5 Gated Recurrent Units (GRU)

Gated Recurrent Units (GRU) [11] are[2] a type of recurrent neural network architecture that offer a simplified alternative to LSTMs while still being capable of capturing temporal dependencies in sequential data.[42] Similar to LSTMs, GRUs use gating mechanisms to control the flow of

information within the network. However, GRUs have a simpler architecture with only two gates: a reset gate and an update gate. The update gate determines how much of the previous hidden state to retain, while the reset gate controls how much of the past hidden state to forget. This design allows GRUs to learn long-term dependencies without the complexity of the separate cell state found in LSTMs.[2]
GRUs have been successfully applied in sequence prediction for movement data across various domains.[4] In maritime applications, GRUs have been used for ship trajectory prediction based on AIS data, often showing comparable or even better performance than LSTMs.[2] In aviation, GRUs have been employed for aircraft trajectory prediction, sometimes in combination with CNNs for enhanced feature extraction.[7] Notably, GRU-based models have demonstrated state-of-the-art performance in predicting the trajectories of Unmanned Aerial Vehicles (UAVs), even outperforming more complex architectures like LSTMs and transformers in certain scenarios. GRUs have also been used in specialized applications such as predicting the trajectory of a ping-pong ball in real-time using video data processed by the YOLOv4-Tiny object detection algorithm. Furthermore, bidirectional GRU (Bi-GRU) [47] networks have been applied to forecast network indicators in airfield operations for conflict risk assessment. Hybrid models combining GRUs with other architectures, such as transformers, have also been explored to leverage the strengths of different model types, for example, in soil moisture content forecasting.[8] Compared to LSTMs, GRUs often have fewer parameters and require less computation, which can lead to faster training times while achieving comparable or sometimes even better performance in trajectory prediction tasks. This efficiency makes GRUs a compelling alternative to LSTMs, particularly in applications where computational resources are limited or when dealing with large datasets.

3.6 1D Convolutional Neural Networks (CNN)

One-dimensional Convolutional Neural Networks (1D CNNs) [45] are a type of neural network designed to process sequential data by applying convolutional filters along one dimension of the input sequence.[2] These filters slide over the input sequence, learning to extract local patterns and features over time. 1D CNNs are particularly effective at capturing spatial hierarchies in temporal data and can be more computationally efficient than recurrent neural networks for certain tasks.
1D CNNs have been increasingly applied in sequence prediction for trajectory data across various domains.[2] Hybrid models combining 1D CNNs with LSTMs[53] have shown promise in flight trajectory prediction by leveraging CNNs for spatial feature extraction from adjacent trajectory regions and LSTMs for capturing temporal dependencies.[2] In maritime applications, 1D CNNs have been used to predict ship trajectories based on AIS data, demonstrating reduced prediction errors compared to other deep learning algorithms in some studies.[48] For pedestrian trajectory prediction, CNN-based approaches offer the advantage of parallelism and can effectively represent temporal dependencies.[51] In traffic management[37], a 1D CNN-LSTM model has shown significant improvements in traffic state prediction compared to other baseline models. 1D CNNs are also utilized for anomaly detection in time series data, such as predicting anomalies in terrestrial water storage , and in remote sensing for tasks like retrieving chlorophyll-a concentration from satellite data.[24] Furthermore, 1D CNNs[52] are being explored for hyperparameter optimization in network intrusion detection systems.
The primary strength of 1D CNNs in this context lies in their ability to efficiently extract local features from sequential data.[2] Compared to RNNs[57], 1D CNNs often exhibit faster training times due to their parallel processing capabilities and can effectively capture local patterns within the trajectory data. The combination of 1D CNNs with recurrent architectures allows for the modeling of both spatial and temporal aspects of trajectory data, potentially leading to more accurate and robust prediction models.

3.7 Transformer Models

Transformer models, initially a breakthrough in the field of Natural Language Processing (NLP), have demonstrated remarkable capabilities in handling sequential data and capturing long-range dependencies through their self-attention mechanisms.[8] Unlike recurrent models that process sequences step by step, transformers can process entire sequences simultaneously, allowing them to efficiently model complex interactions and dependencies across the entire trajectory. The core of the transformer architecture includes encoder and decoder structures, both heavily relying on multi-head self-attention mechanisms to weigh the importance of different parts of the input sequence when making predictions.[8] Transformer models have been successfully applied in sequence prediction for trajectory data across various domains.[8] In aviation, transformer-based models have shown high accuracy in flight trajectory prediction and exhibit superior performance in parallel sequential data processing compared to traditional recurrent networks.[8] For maritime navigation, transformers have been effective in capturing long-term dependencies in ship trajectories using AIS data.[25] In the realm of autonomous driving, transformer models are well-suited for vehicle trajectory prediction by efficiently modeling complex interactions between vehicles and the environment.[9] Scene-aware transformer models have been developed for pedestrian trajectory prediction, integrating contextual information to improve forecasting accuracy.[10] Researchers have also explored using transformers to model high-dimensional spatio[34]-temporal trajectories[26] by treating them as sequences of discrete locations, drawing parallels with language modeling techniques.[9] Furthermore, hybrid architectures integrating transformers with LSTMs have been investigated to enhance spatial and temporal feature learning in tasks like vehicle trajectory prediction , and combinations with GRUs have been used for applications like soil moisture content forecasting.[8] The attention mechanisms inherent in transformer models play a crucial role in identifying key factors and extracting diverse information from trajectory data by allowing the model to weigh the importance of different parts of the input sequence.[8] A significant advantage of transformers is their ability to handle long-range dependencies in sequential data more effectively than traditional RNNs, along with their suitability for parallel processing, which can be particularly beneficial for large trajectory datasets.

3. LITERATURE REVIEW [RESEARCH SCOPE]

Trajectory prediction is a well-researched area with applications in various fields such as autonomous driving, robotics, and air traffic management. Various approaches have been proposed, ranging from traditional methods based on kinematics and dynamics to advanced deep learning models [8]. These methods aim to predict future positions of moving objects by learning from historical trajectory data, often captured in complex and dynamic environments.

Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), have been widely used for trajectory prediction [6] due to their capability to capture temporal dependencies in sequential data [1]. Variants like Social LSTM extend this by incorporating interaction patterns between entities, such as pedestrian groups [1]. Bidirectional LSTMs (Bi-LSTMs) further enhance performance by processing input sequences in both forward and backward directions, improving context awareness [3]. These models have demonstrated their value in aviation [2], maritime navigation [12], and pedestrian movement forecasting [1].

Convolutional Neural Networks (CNNs) [29], while traditionally applied in image processing, have found a place in trajectory prediction through hybrid frameworks that combine spatial and temporal learning. These include CNN-LSTM models [50], where CNNs extract local spatial patterns and LSTMs model the sequence dynamics. Transformer models [30], originally from NLP, have shown promising results in trajectory tasks due to their self-attention mechanisms, which enable them to capture long-range dependencies and complex interactions [8].

Traditional machine learning models like Support Vector Machines (SVMs) have been employed in trajectory analysis by using feature engineering to convert time series into static input formats [12]. These models are particularly effective in low-dimensional or noise-prone scenarios. Ensemble approaches such as XGBoost [56] are also gaining traction due to their robustness, efficiency, and capacity to model nonlinear relationships in spatio-temporal data.

Recent trends in trajectory prediction have also seen the emergence of hybrid and modular models that combine the strengths of multiple architectures. For example, transformer-biLSTM hybrids and CNN-GRU pipelines leverage both long-range attention and short-term memory for improved performance. In addition, research has explored techniques like transfer learning and reinforcement learning (e.g., RC-TL [29]) to enhance generalizability in data-scarce environments. These innovations reflect the growing demand for trajectory prediction models that are not only accurate but also scalable, interpretable, and robust across domains.

The choice of model often depends on the specific application, the nature of the trajectory data, and the desired balance between accuracy, computational efficiency, and interpretability. Accordingly, comparative evaluations and systematic benchmarking—such as the one presented in this study—play a vital role in identifying the most suitable models for different deployment scenarios.

Research Scope

Building upon the diverse methodologies outlined above, this research undertakes a comprehensive comparative analysis of regression-based models tailored for immediate trajectory prediction, with a particular emphasis on real-time surveillance applications. The primary objective is to forecast the immediate latitude and longitude of moving entities using a combination of real-world and synthetically generated trajectory data. Unlike prior works that focus solely on long-term predictions or domain-specific applications, the present study addresses the need for short-term, high-precision forecasts in bandwidth-constrained environments. To achieve this, custom architectures were developed by modifying established deep learning and machine learning frameworks—including LSTM, GRU, Bi-LSTM, CNN, Transformer, XGBoost, and SVM—to suit the temporal granularity and multivariate nature of the curated dataset. The research scope also extends to evaluating these models using multiple error metrics (MSE, MAE, RMSE, R², SMAPE), thus providing a multi-dimensional perspective on predictive accuracy and robustness. By focusing on hybrid data environments and surveillance-centric constraints, this study contributes a novel experimental framework and benchmarks for selecting efficient trajectory prediction models under real-time operational scenarios.

4. METHODOLOGY

5.1 Custom LSTM Model for Trajectory Data

The proposed architecture leverages a Long Short-Term Memory (LSTM) neural network integrated within a sequential model to predict future trajectory points of moving entities. This model was particularly suitable for spatio-temporal data as it captured temporal dependencies. The architecture is compact yet efficient, allowing for real-time inference in dynamic environments.

```
model_lstm = Sequential([
  LSTM(64, activation='relu',
input_shape=(X.shape[1], X.shape[2])),
  Dense(32, activation='relu'),
  Dense(2)
])
model_lstm.compile(optimizer='adam',
loss='mse')
```

Layers Description

1. Input Layer

- Shape: (timesteps, features)
- This layer receives a multivariate time series input, where each sample corresponds to a sequence of trajectory records.
- The input format encapsulates both temporal and spatial characteristics, such as position coordinates (latitude/longitude or X/Y), velocity, or directional angle.
- The dimensionality of the input is automatically inferred from the training data, ensuring adaptability to varying feature combinations.

2. LSTM Layer

- Units: 64
- Activation Function: ReLU (Rectified Linear Unit)
- The LSTM layer serves as the temporal memory unit, designed to retain long-term dependencies across time steps.
- Using ReLU instead of tanh improves convergence, reduces vanishing-gradient issues, and enhances learning efficiency, especially with normalized dataThis layer effectively encodes temporal patterns, such as directionality, acceleration, or periodic movements in user mobility traces.

3. Dense (Fully Connected) Layer

- Units: 32
- Activation: ReLU
- This intermediate layer acts as a feature transformer. It captures abstract, non-linear combinations of the temporal features encoded by the LSTM.
- It serves to improve generalization and model capacity by refining the learned embeddings before the output layer.

4. Output Layer

- Units: 2
- Activation: Linear (default for regression)
- Outputs a pair of continuous values representing the next predicted spatial location (e.g., next X and Y coordinates).
- Linear activation is employed to support unrestricted regression values, suitable for real-valued spatial predictions.

5.2 Custom Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) networks are a streamlined variant of Long Short-Term Memory (LSTM) networks, designed to handle sequential data with temporal dependencies. In this model, the GRU processes sequences of spatio-temporal trajectory data to forecast the next probable location point (e.g., next latitude and longitude). Due to its compact structure and fewer trainable parameters, GRU offers faster training and comparable accuracy to LSTM, especially in real-time systems or edge devices.

```
model_gru = Sequential([
    GRU(64, activation='relu',
input_shape=(X.shape[1], X.shape[2])),
    Dense(32, activation='relu'),
    Dense(2)
])
model_gru.compile(optimizer='adam',
loss='mse')
```

Layer-wise Description

1. Input Layer

- Shape: (timesteps, features)
- Each input sequence represents historical trajectory segments (e.g., past GPS points or speed/angle).
- The features may include spatial coordinates, velocity, and time deltas.
- This layer serves as the data entry point for the GRU, preserving time-dependent patterns.

2. GRU Layer

- Units: 64
- Activation: ReLU
- The GRU cell replaces the LSTM's three gates (input, forget, output) with two gates: update and reset.
- The update gate controlled memory retention across time, while the reset gate determined how new input was combined with previous memory
- Benefits:
  - Reduced computational complexity compared to LSTM
  - Faster convergence and lower memory usage
  - Well-suited for real-time systems with less latency

3. Dense Layer

- Units: 32
- Activation: ReLU
- This fully connected hidden layer introduces non-linear transformations, capturing latent features from the GRU's output.
- Acts as an intermediary to refine the feature representation before prediction.

4. Output Layer

- Units: 2
- Activation: Linear (default)
- Produces two continuous values corresponding to the predicted spatial coordinates.
- The linear activation is ideal for regression-based outputs without range constraints.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5.3 Custom Bi-directional Long Short-Term Memory

Bi-directional Long Short-Term Memory (Bi-LSTM) networks are an advanced form of recurrent neural networks designed to capture temporal dependencies in both forward and backward directions. This dual-context learning makes Bi-LSTMs highly suitable for trajectory prediction tasks where the context of both past and future movements enhances predictive accuracy. The proposed model leverages Bi-LSTM to predict the next positional point of a moving object based on sequential input data.

```
model_bilstm = Sequential([
  Bidirectional(LSTM(64, activation='relu', input_shape=(X.shape[1], X.shape[2]))),
  Dense(32, activation='relu'),
  Dense(2)
])
model_bilstm.compile(optimizer='adam', loss='mse')
```

Layer-wise Description

1. Input Layer

- **Shape**: (timesteps, features)
- Represents the temporal sequence of input data, typically comprising spatial coordinates (latitude, longitude) and optionally, speed, heading, or time gaps.
- Each sequence is treated as a trajectory segment observed over a time window.

2. Bi-Directional LSTM Layer

- **Units**: 64 (applied separately in both forward and backward directions)
- **Activation**: ReLU
- **Mechanism**: The Bi-LSTM processes the sequence from both ends—past to present and future to past—by maintaining two separate LSTM cells and then concatenating their outputs.
- This dual-context modeling improved the learning of complete trajectory patterns and helped in cases where the path history alone was not sufficient for accurate prediction.

3. Dense Layer

- **Units**: 32
- **Activation**: ReLU
- Extracts higher-order representations and transforms the bidirectional outputs into a more abstract, learnable feature space.
- Acts as a nonlinear regression prior to final output projection.

4. Output Layer

- **Units**: 2
- **Activation**: Linear (suitable for regression)
- Outputs two values that represent the next predicted coordinates (e.g., next latitude and longitude).
- Unbounded activation is suitable for spatial prediction over continuous coordinate ranges.

5.4 Custom 1D CNN

Convolutional Neural Networks (CNNs) are well-known for their success in image and signal processing, but their 1D variant is also highly effective for sequential data. The 1D CNN model is used here to process spatio-temporal trajectory data, treating each sequence as a temporal signal where local movement patterns (e.g., direction changes, speed bursts) can be captured through convolutional filters. This architecture emphasizes local feature extraction and is computationally faster than RNN-based models like LSTM and GRU.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
model_cnn = Sequential([
    Conv1D(64, kernel_size=3, activation='relu', input_shape=(X.shape[1], X.shape[2])),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(2),
])
model_cnn.compile(optimizer='adam', loss='mse')
```

Layer-wise Description

1. Input Layer

- Shape: (timesteps, features)
- Accepts a multivariate time-series input representing movement histories.
- Each sequence includes timesteps worth of data points with multiple features (e.g., position, speed, angle).

2. Convolutional Layer (1D)

- Filters: 64
- Kernel Size: 3
- Activation: ReLU
  This layer slid over the sequence dimension to detect local temporal features such as short-term shifts in position or velocity
- Convolutions allow the model to learn translationally invariant movement patterns (e.g., sharp turns or direction changes) across time.

3. Flatten Layer

- Transforms the multidimensional convolution output into a single-dimensional vector.
- Prepares the data for fully connected (dense) layers by collapsing temporal and feature dimensions.

4. Dense Layer

- Units: 32
- Activation: ReLU
- Acts as a non-linear transformation unit to refine the feature map obtained from convolution.
- Facilitates interaction between extracted features and contributes to high-level learning.

5. Output Layer

- Units: 2
- Activation: Linear
- Predicts two real-valued outputs representing the next spatial coordinates (e.g., latitude and longitude).
- The use of a linear activation function makes it suitable for continuous trajectory prediction.

5.5 Custom Transformer architecture

The Transformer architecture, originally proposed for natural language processing, has recently been adapted for time-series and sequential data modeling. Its self-attention mechanism allows the model to weigh the importance of all positions in the input sequence simultaneously, making it ideal for capturing long-range dependencies and complex interactions within trajectory data. This model offers a parallelizable and scalable alternative to RNNs (LSTM/GRU) for mobility forecasting.

```
input_layer = Input(shape=(X.shape[1], X.shape[2]))
attention_output = MultiHeadAttention(num_heads=4, key_dim=X.shape[2])(input_layer, input_layer)
attention_output = LayerNormalization()(attention_output)
dense_output = Dense(64, activation='relu')(attention_output)
flatten_output = Flatten()(dense_output)
final_output = Dense(2)(flatten_output)

model_transformer = Model(inputs=input_layer, outputs=final_output)
model_transformer.compile(optimizer='adam', loss='mse')
```

1. Input Layer

- Shape: (timesteps, features)
- Accepts sequential trajectory input such as position, velocity, and timestamp features.

  It acted as input for the self-attention mechanism, enabling interactions across time steps2. Multi-Head Attention Layer

- Heads: 4
- Key Dimension: Equal to feature size
- This layer computes scaled dot-product attention in parallel across multiple "heads".
- Enables the model to capture diverse movement patterns and dependencies between all positions in the trajectory window.

3. Layer Normalization

- Normalizes the output of the attention layer to stabilize training and accelerate convergence.
- Helps mitigate issues like vanishing/exploding gradients, commonly seen in deep architectures.

4. Dense Layer

- Units: 64
- Activation: ReLU
- Applies nonlinear transformation to refine and combine the attention-encoded features.
- Enhances model capacity for learning high-level abstractions of spatial-temporal dependencies.

5. Flatten Layer

- Converts the 3D attention-dense output into a 1D vector.
- Acts as a bridge between the encoder-like part and the final output projection.

6. Output Layer

- Units: 2
- Activation: Linear
- Produces two continuous values representing predicted coordinates (e.g., next latitude and longitude).

5.6 Custom XGBoost

Extreme Gradient Boosting (XGBoost) is a high-performance ensemble learning method based on decision tree algorithms. While primarily designed for tabular and structured data, XGBoost can be adapted for time-series and trajectory prediction by flattening temporal data into a 2D representation. This approach is non-sequential but powerful for learning non-linear spatial relationships across timesteps. In this context, the model predicts spatial coordinates by treating each trajectory segment as a fixed-length feature vector.

Model Design Description

Input Representation

- Flattened Format: Trajectory data with shape (timesteps, features) is reshaped into a 2D vector of size (timesteps × features) to suit XGBoost's tabular format.
- This transformation allows the model to learn correlations across all time steps simultaneously, but at the cost of losing inherent sequential dependencies.

XGBoost Regressor

- Algorithm: Gradient-boosted decision trees
- Hyperparameters:
    - n_estimators=100: 100 boosting rounds
    - max_depth=4: Shallow trees for better generalization
- Mechanism:
    - Trees are added sequentially, each one improving upon the residual error of the previous.
    - Uses a gradient descent approach to minimize the objective function (here, mean squared error).
    -

Output

- Target: A 2D continuous output (e.g., predicted latitude and longitude).
- XGBoost outputs this as a vector from its final regressor leaf nodes.

5.7 Custom Support Vector Regression

To explore the feasibility of classical machine learning approaches in trajectory prediction, a Support Vector Regression (SVR) model was designed and implemented to predict the future spatial coordinates of a moving object. This model leverages a multivariate input representation that includes geospatial and kinematic features derived from historical observations. Specifically, the model incorporates latitude, longitude, instantaneous speed, and movement direction over a fixed temporal window as input features.

```
# Reshape input to flatten temporal dimension
X_flat = X.reshape(X.shape[0], -1)

# Train/test split
X_train_flat, X_test_flat = X_flat[X_train.shape[0]*-1:], X_flat[:X_test.shape[0]]

# XGBoost Regressor
xgb = XGBRegressor(n_estimators=100, max_depth=4)
xgb.fit(X_train_flat, y_train)

# Prediction & Evaluation
pred_xgb = xgb.predict(X_test_flat)
mse_xgb = mean_squared_error(y_test, pred_xgb)
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
# Train the SVR separately for latitude and longitude
svr_lat = SVR(kernel='rbf')
svr_lon = SVR(kernel='rbf')

svr_lat.fit(X_train_flat, y_train[:, 0]) # Train on latitude
svr_lon.fit(X_train_flat, y_train[:, 1]) # Train on longitude

# Predict both coordinates
pred_lat = svr_lat.predict(X_test_flat)
pred_lon = svr_lon.predict(X_test_flat)

# Combine predictions
pred_combined = np.column_stack((pred_lat, pred_lon))
```

Feature Representation

Each trajectory sequence is represented by a flattened vector that concatenates several time steps of the following features:

- Latitude
- Longitude
- Instantaneous Speed (km/h)
- Movement Direction (degrees)

For a time window of T timesteps, the input vector to the model has a dimensionality of $T \times 4$. This design enables the SVM to model the influence of movement behavior over time without explicitly encoding temporal recurrence.

Model Configuration

The model utilizes two separate SVR regressors with the following setup:

- **SVR Kernel**: Radial Basis Function (RBF)
- **Target Outputs**:
  - SVR_1 → Predicts next **latitude**
  - SVR_2 → Predicts next **longitude**
- The final predicted coordinate is obtained by combining outputs from both models.

- Together, these models provided a comprehensive comparison of both deep learning and traditional regression approaches under identical experimental conditions

5. EXPERIMENTAL EVALUATIONS

5.1 Performance Measures

5.1.1 Mean Squared Error (MSE)

The Mean Squared Error (MSE) is a widely used metric for evaluating the performance of regression models. It is calculated as the average of the squared differences between the predicted values and the actual values.[8] The formula for MSE is given by: MSE = (1/n) * Σ(actual – forecast)^2, where *n* is the number of observations, 'actual' represents the true values, and 'forecast' represents the predicted values. A lower MSE value indicates that the model's predictions are closer to the actual values, signifying better accuracy. However, MSE is sensitive to outliers because it squares the errors, which means that large errors have a disproportionately greater impact on the MSE value. This property makes MSE particularly useful when large errors are undesirable and should be heavily penalized.

5.1.2 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is another common metric for evaluating regression model performance. It measures the average of the absolute differences between the predicted values and the actual values.[8] The formula for MAE is: MAE = (1/n) * Σ|actual – forecast|. Similar to MSE, a lower MAE value indicates better accuracy, as it signifies that the predictions are, on average, closer to the true values. However, MAE is less sensitive to outliers compared to MSE because it uses the absolute difference rather than the squared difference, treating all errors equally regardless of their magnitude. An important advantage of MAE is that it provides a straightforward measure of the model's accuracy in the same units as the target variable, making it easily interpretable.

5.1.3 Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) is derived from the MSE and is calculated as the square root of the mean of the squared differences between predicted and actual values.[8] The formula for RMSE is: RMSE = sqrt((1/n) * Σ(actual –

forecast)^2). Like MSE, a lower RMSE indicates better accuracy. RMSE shares the sensitivity to outliers with MSE due to the squaring of errors. A key benefit of RMSE is that it provides an error metric in the same units as the original data, making it easier to interpret the magnitude of the prediction error in the context of the problem. For this reason, RMSE is often preferred over MSE for reporting the performance of regression models.

5.1.4 R-squared

R-squared (R^2) is a statistical metric that represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s) in a regression model.[8] The formula for R-squared is: R^2 = 1 – (SSres / SStot), where SSres is the sum of squared residuals and SStot is the total sum of squares. R-squared values range from 0 to 1, with higher values indicating a better fit of the model to the observed data. It measures the goodness of fit, indicating how well the model's predictions match the actual data points. Unlike the error-based metrics, R-squared is independent of the scale of the data, providing a relative measure of the model's explanatory power.

5.1.5 Symmetric Mean Absolute Percentage Error (SMAPE)

The Symmetric Mean Absolute Percentage Error (SMAPE) is a variation of MAPE that aims to address the asymmetry issue by using the average of the absolute actual and forecast values in the denominator.[3] The formula for SMAPE is: SMAPE = (1/n) * Σ(2 * |forecast – actual| / (|actual| + |forecast|)) * . Similar to MAPE, a lower SMAPE value indicates better accuracy. SMAPE has both lower and upper bounds, typically ranging from 0% to % or 0% to % depending on the specific formula used. Compared to MAPE, SMAPE is less prone to issues with zero or near-zero actual values, making it a more stable metric in such cases.

Comparative Analysis Measures

Metric	Formula	Strengths	Weaknesses
Mean Squared Error (MSE)	$MSE = (1/n) * \Sigma (y\_i - ŷ\_i)^2$	Penalizes large errors more; sensitive to outliers.	Sensitive to outliers; not easily interpretable due to squared units.
Mean Absolute Error (MAE)	$MAE = (1/n) * \Sigma \|y\_i - ŷ\_i\|$	Easy to interpret; less sensitive to outliers.	Ignores direction of error; may underrepresent large deviations.
Root Mean Squared Error (RMSE)	$RMSE = sqrt((1/n) * \Sigma (y\_i - ŷ\_i)^2)$	Same units as original data; balances MSE interpretation.	Still influenced by outliers; less robust for skewed distributions.
R-squared (R^2)	$R^2 = 1 - [\Sigma(y\_i - ŷ\_i)^2 / \Sigma(y\_i - ȳ)^2]$	Explains variance captured by the model; intuitive performance indicator.	Can be misleading for non-linear or biased models; depends on data spread.
Symmetric Mean Absolute Percentage Error (SMAPE)	$SMAPE = (100/n) * \Sigma [\|y\_i - ŷ\_i\| / (\|y\_i\| + \|ŷ\_i\|)/2]$	Scale-independent; effective for percentage-based error comparison.	Undefined if actual or predicted values are zero; can exaggerate error on small magnitudes.

5.2 DATASET

To support the evaluation of trajectory prediction models under scalable and controlled conditions, a synthetic dataset was developed to mimic the spatio-temporal movement patterns found in the Beijing T-Drive dataset [70] . The T-Drive dataset, which records the GPS trajectories of over 10,000 taxis in Beijing with high temporal fidelity, has been widely used in mobility research for modeling urban dynamics and traffic prediction. However, due to licensing restrictions and practical limitations in data volume and preprocessing, a custom dataset generator was implemented to simulate similar behavior with enhanced control over temporal resolution and user variability.

The synthetic dataset was generated using a Python-based simulation script that creates user-specific movement data with a fixed 5-minute granularity. Each simulated user begins at a randomly initialized geographic coordinate and is assigned a unique base speed and direction. For every user, a sequence of 1,000 records is generated using a dynamic location update mechanism. This mechanism calculates the subsequent position using a geospatial transformation based on spherical geometry. The angular displacement in latitude and longitude is computed from the user's current speed and movement direction, incorporating Earth's radius and bearing angle adjustments to produce geographically valid coordinates. Small randomized variations in both speed and direction are added at each step to inject stochastic behavior, replicating real-world travel irregularities such as acceleration, deceleration, or minor route deviations.

Each record includes the following fields: User ID, Latitude, Longitude, Timestamp, Speed (in km/h), and Direction (in degrees). The timestamp is incremented in fixed 5-minute intervals to maintain temporal consistency across all users. Upon completion, the system exported data for 3,000 users, resulting in a total of 3 million trajectory points saved in structured JSON format. This synthetic dataset retains key characteristics of realistic urban mobility—namely, spatial continuity, directional variation, and velocity diversity—while offering full control over parameters such as data density, geographical scope, and user count. It serves as a reliable surrogate for real-world data in model training, evaluation, and ablation studies.

5.3 Experimental Set up

All trajectory prediction experiments—including model training, evaluation, and performance benchmarking—were carried out on a personal computing system configured for machine learning research. The hardware and software environments were selected to balance computational efficiency with accessibility for academic experimentation.

5.4 7.2 Hardware Configuration

- Processor: Intel® Core™ i7-10750H CPU @ 2.60GHz (12 logical cores)
- Memory (RAM): 16 GB DDR4
- Storage: 512 GB NVMe Solid State Drive (SSD)
- Graphics Processing Unit (GPU): NVIDIA GeForce GTX 1650 (4 GB GDDR6, CUDA-enabled)
- Display & Cooling: Full HD display with dual-fan thermal cooling (sustained training capability)

5.5 Operating System and Development Environment

- Operating System: Ubuntu 22.04 LTS (64-bit)
- Programming Language: Python 3.10.12 (installed via Miniconda for virtual environment isolation)
- Code Editor / IDE: Visual Studio Code (v1.85) with Python and Jupyter extensions
- Interactive Environment: Jupyter Notebook and JupyterLab

5.6 Python Libraries and Frameworks

Function	Tool/Library Used
Deep Learning	TensorFlow 2.13.0 (Keras API)
Machine Learning	XGBoost (v1.7.6), Scikit-learn (v1.3.1)
Data Handling & Numerical Ops	NumPy (v1.24.4), Pandas (v2.0.3)
Visualization	Matplotlib (v3.7.1), Seaborn (v0.12.2)
Attention Mechanism	MultiHeadAttention (Keras layer)
Utilities	SciPy (v1.11.1), tqdm, joblib

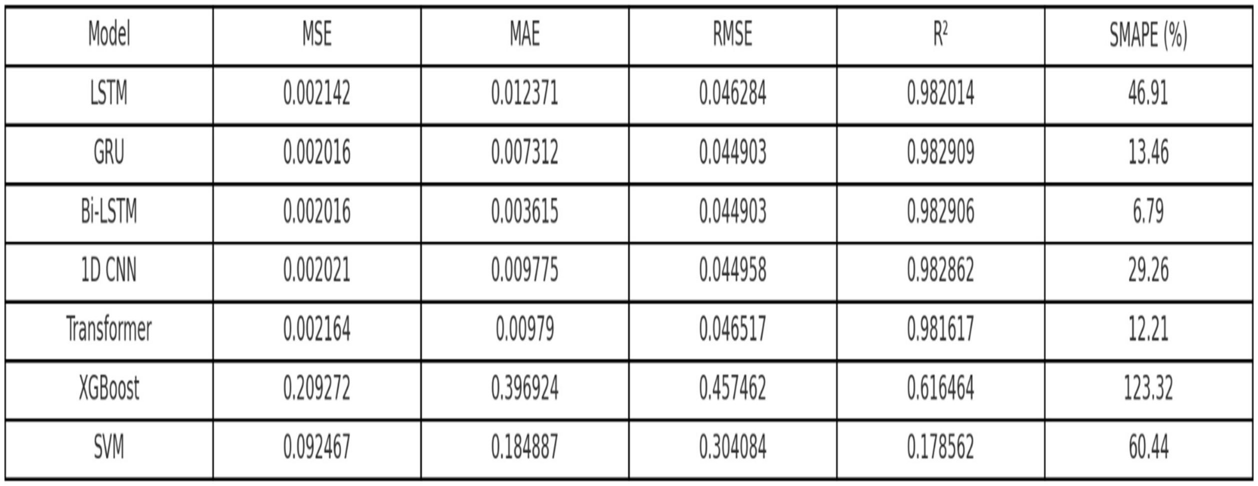
5.7 Experiment Controls and Reproducibility

To ensure consistency and repeatability: GPU acceleration (CUDA) was enabled during training. Output metrics were logged via in-notebook visualization and TensorBoard tracking

This environment was found suitable for training RNN-based models (LSTM, GRU, Bi-LSTM), convolutional models (1D CNN), transformer-based attention models, and tree-based learners like XGBoost. The setup supported end-to-end execution from data preprocessing to model evaluation in real time.

5.8 RESULT ANALYSIS

5.8.1 Individual Model Performance

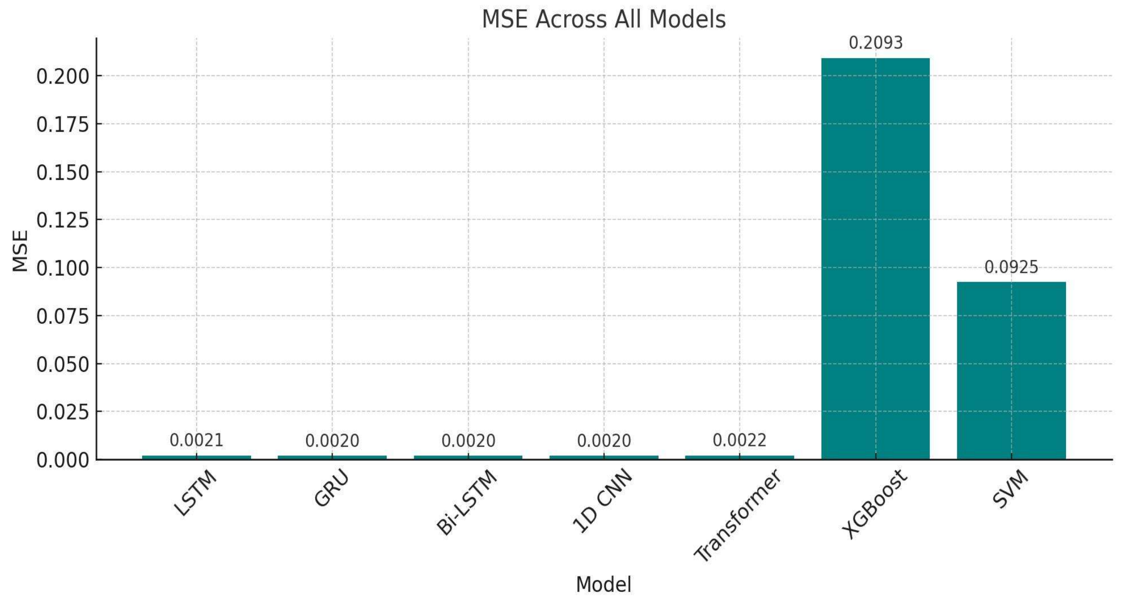


Model	MSE	MAE	RMSE	R²	SMAPE (%)
LSTM	0.002142	0.012371	0.046284	0.982014	46.91
GRU	0.002016	0.007312	0.044903	0.982909	13.46
Bi-LSTM	0.002016	0.003615	0.044903	0.982906	6.79
1D CNN	0.002021	0.009775	0.044958	0.982862	29.26
Transformer	0.002164	0.00979	0.046517	0.981617	12.21
XGBoost	0.209272	0.396924	0.457462	0.616464	123.32
SVM	0.092467	0.184887	0.304084	0.178562	60.44

5.8.2 Mean Squared Error (MSE)

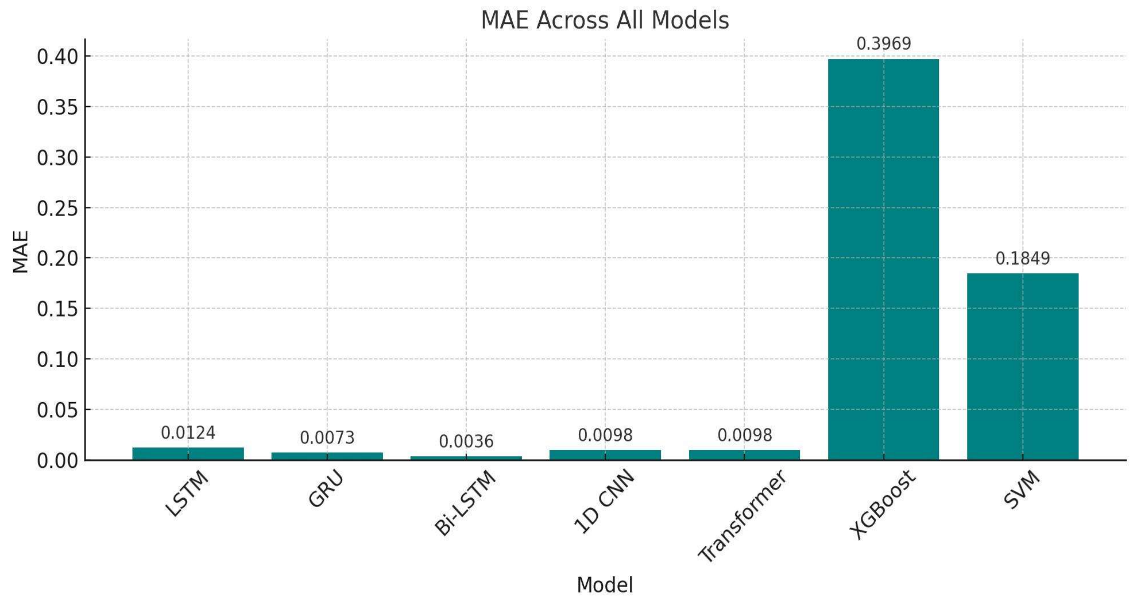
The MSE metric, which penalizes larger errors more heavily, shows that GRU and Bi-LSTM achieve the lowest error values (0.002016), followed closely by 1D CNN and LSTM. The Transformer model is marginally inferior, recording an MSE of 0.002164. In contrast, XGBoost and SVM demonstrate significantly higher error, with XGBoost recorded the highest MSE at 0.209272, indicating limited generalization capacity under continuous spatial prediction tasks.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



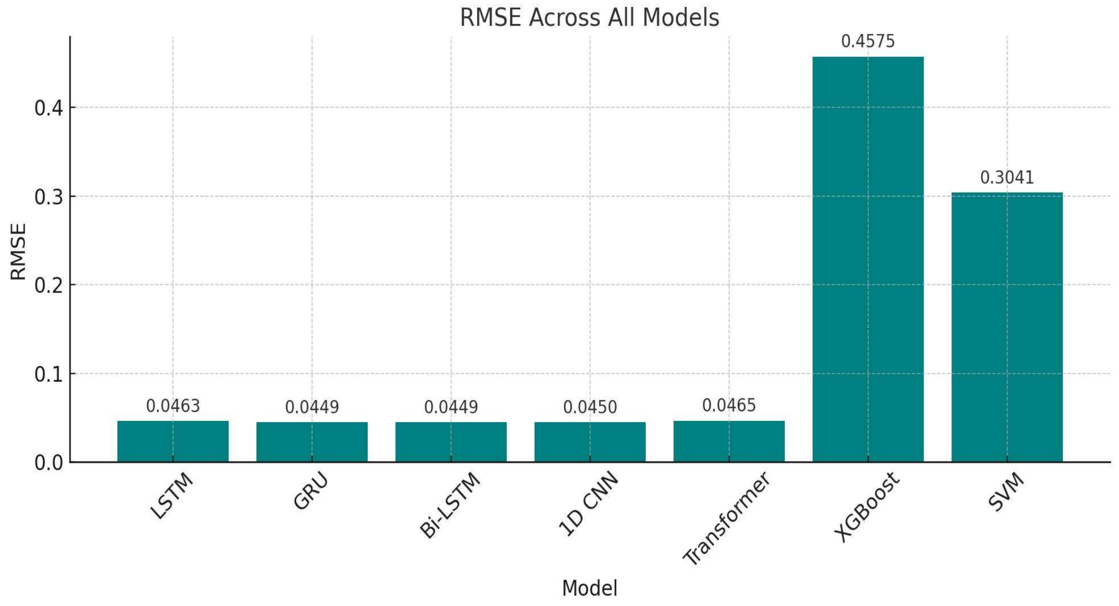
5.8.3 Mean Absolute Error (MAE)

Bi-LSTM clearly outperforms all other models in minimizing average deviation (0.003615), followed by GRU (0.007312). This demonstrates Bi-LSTM's high fidelity in capturing both the direction and scale of movement. XGBoost and SVM again show elevated errors, suggesting that traditional models struggle to match the precision of deep learning methods for sequential data.



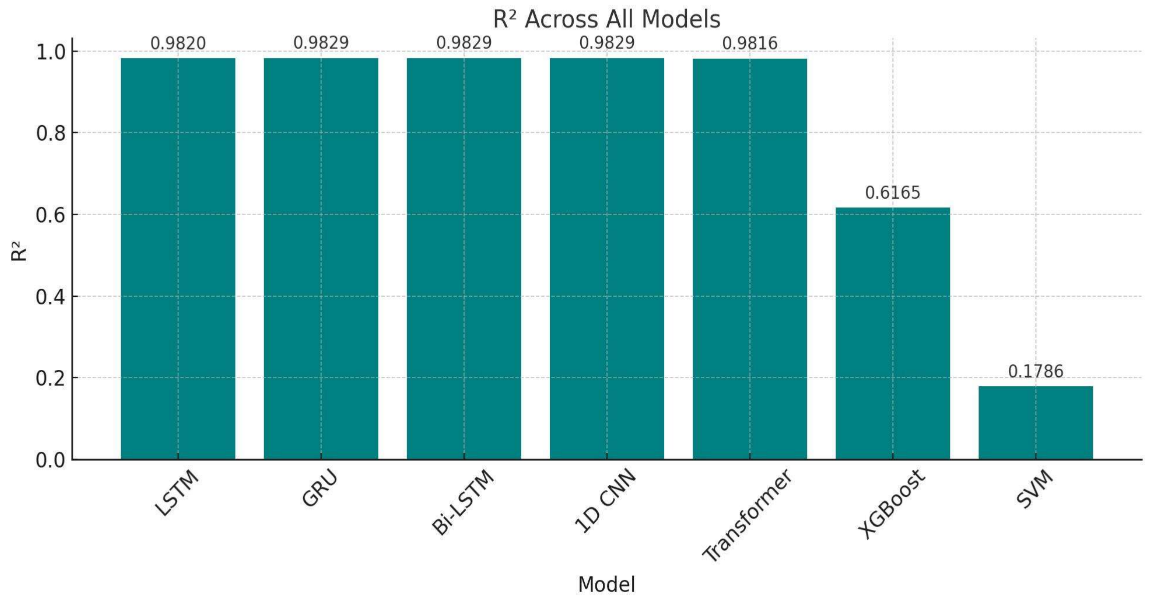
5.8.4 Root Mean Square Error (RMSE)

Similar trends are observed with RMSE, where GRU and Bi-LSTM again achieve the best performance (0.044903), and LSTM, 1D CNN, and Transformer follow closely. XGBoost (0.457462) and SVM (0.304084) exhibit significantly higher RMSE values, reflecting both large and frequent deviations from actual trajectories.



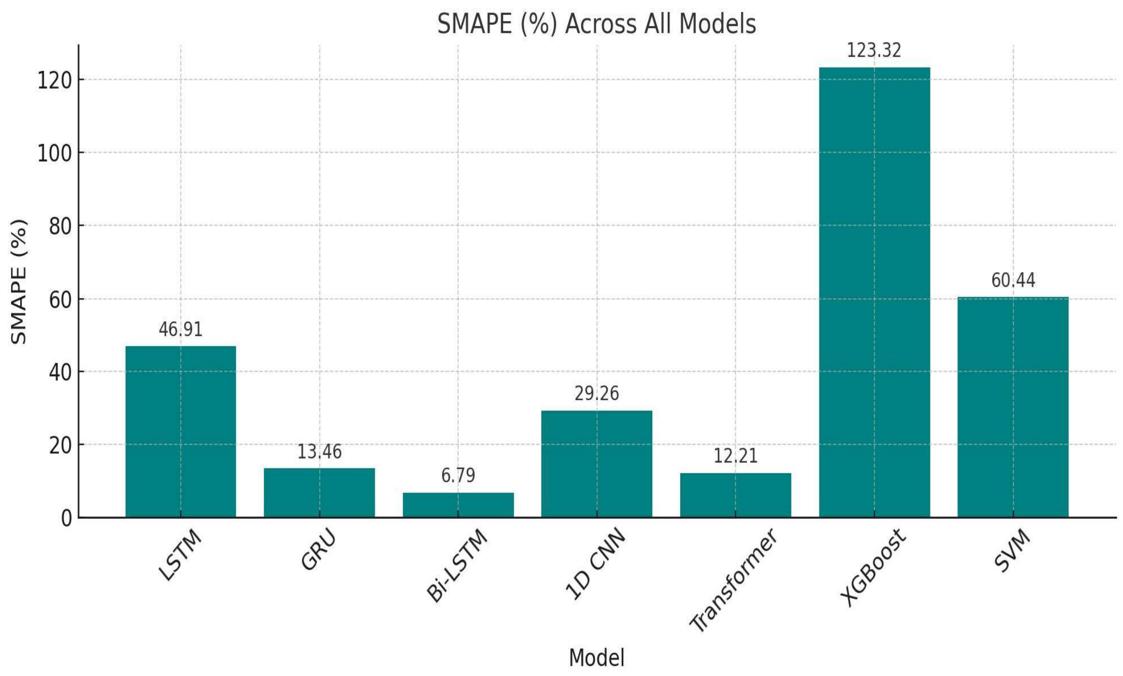
5.8.5 Coefficient of Determination (R²)

R² values reinforce model quality. Deep learning models—GRU, Bi-LSTM, 1D CNN, and LSTM—achieve near-perfect values (≈ 0.982), indicating excellent model fit. Transformer shows a slightly lower but still strong R² (0.981617). In stark contrast, XGBoost (0.616464) and especially SVM (0.178562) perform poorly, indicating that these models capture only limited variability in the data.



Symmetric Mean Absolute Percentage Error (SMAPE)

SMAPE measures relative error and shows that Bi-LSTM delivers outstanding performance (6.79%), followed by Transformer (12.21%) and GRU (13.46%). These models not only predict accurately but also scale well across varying trajectory magnitudes. XGBoost, with a SMAPE of 123.32%, and SVM (60.44%) illustrate severe performance instability when normalized against actual values.

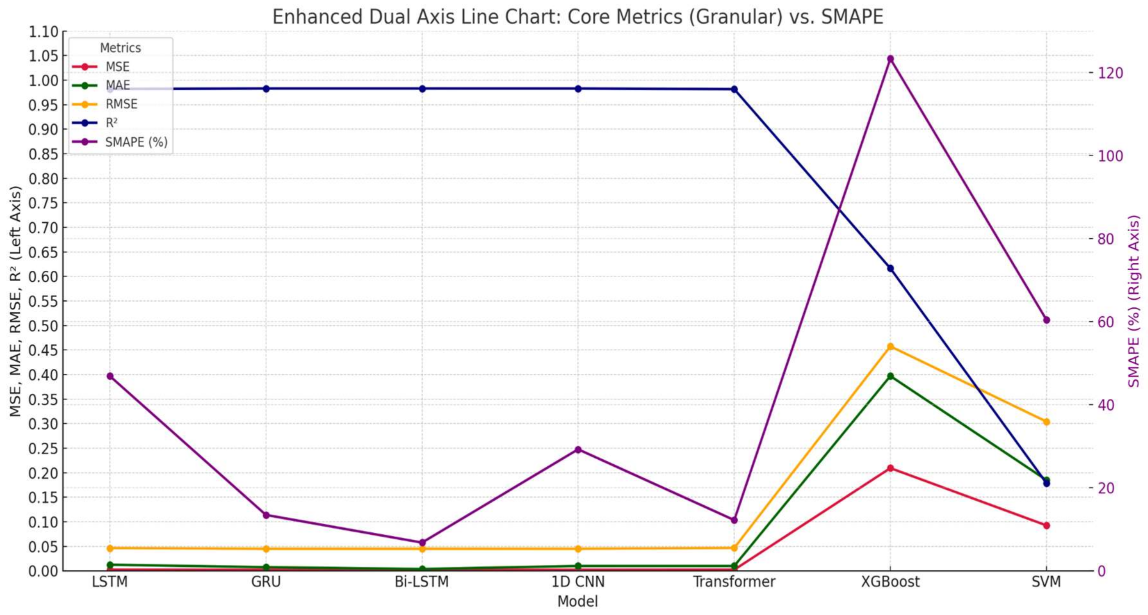


5.8.6 Comparative Trend Analysis

From a comparative standpoint, Bi-LSTM emerges as the most robust mode, offering the lowest MAE and SMAPE while matching GRU in RMSE and MSE. GRU itself strikes an excellent balance between accuracy and computational efficiency, with top-tier RMSE and second-best MAE. 1D CNN and Transformer follow closely, with Transformer having a relatively better SMAPE despite slightly higher absolute errors.

LSTM, while historically reliable, underperforms compared to its gated and bidirectional variants, especially in terms of MAE and SMAPE. This reinforces the need for enhanced memory dynamics in capturing subtle trajectory transitions.

In contrast, XGBoost and SVM are not suitable for high-fidelity trajectory prediction in this configuration. XGBoost's inability to capture temporal dependencies and SVM's restriction to a single spatial dimension significantly hinder their predictive capabilities.



6.9 Discussions

The comparative evaluation highlights several critical insights into the suitability of different regression-based models for trajectory prediction. Sequence-aware deep architectures, particularly Bi-LSTM and GRU, consistently outperformed traditional methods in both absolute (MSE, MAE, RMSE) and relative (SMAPE) error metrics. This confirms the hypothesis that temporal context is essential for capturing the dynamics of mobility patterns. Furthermore, while CNN and Transformer architectures demonstrated competitive performance, their effectiveness depended on the ability to capture localized versus long-range dependencies. Traditional models such as SVM and XGBoost, although computationally efficient, failed to adequately model sequential dependencies, making them less appropriate for real-time applications. Another important observation is that SMAPE provided clearer differentiation between models when predicting low-magnitude values, reaffirming its value as a supplementary evaluation metric. These results collectively suggest that deep learning architectures, especially Bi-LSTM and GRU, represent the most practical and accurate solutions for deployment in bandwidth-constrained, real-time trajectory monitoring systems.

6. SUMMARY AND CONCLUSION

In this study, a comprehensive framework was established for trajectory prediction by evaluating a range of machine learning and deep learning models on both real and simulated data. A synthetically curated dataset was developed to mirror the behavior of real-world urban mobility, particularly inspired by the characteristics of the Beijing traffic dataset. The synthetic dataset included realistic temporal granularity at 5-minute intervals and incorporated multivariate spatio-temporal features such as latitude, longitude, speed, and direction. This augmentation enabled robust experimentation and model validation at scale.

Multiple regression-based models were trained and assessed using standard evaluation metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Coefficient of Determination (R^2), and Symmetric Mean Absolute Percentage Error (SMAPE). The experimental results demonstrated a clear performance distinction between traditional machine learning methods and modern deep learning architectures.

Among the models tested, the Bi-Directional Long Short-Term Memory (Bi-LSTM) network achieved the most consistent and accurate results. It recorded the lowest MAE (0.0036), tied for lowest MSE and RMSE (0.0020 and 0.0449 respectively), and achieved the highest R^2 (0.9829) with the lowest SMAPE (6.79%). These outcomes indicate superior learning of sequential dependencies and temporal context, making Bi-LSTM the most suitable model for trajectory prediction in this experimental setup.

In contrast, traditional models like XGBoost and SVM, while faster and more interpretable, exhibited significantly higher errors and lower R^2 values, reaffirming the limitations of non-sequential models in capturing dynamic mobility patterns.

Overall, this investigation confirms the critical role of sequence-aware deep learning models, particularly Bi-LSTM, in achieving high-accuracy forecasting in trajectory prediction tasks involving spatio-temporal data.

The core regression-based trajectory prediction framework developed in this study serves not only as a standalone forecasting module but also as a foundational component for more advanced real-time analytics systems, such as continuous query clustering frameworks. By accurately estimating the immediate future locations of moving entities, these predictive models reduce reliance on high-frequency raw data collection, thus enabling efficient spatio-temporal stream management. This predictive capability can be seamlessly integrated into a **continuous query processing environment**, where incoming location data streams are dynamically grouped and clustered based on predicted trajectory segments. Such clustering facilitates intelligent grouping of entities with similar movement intents, enabling optimized resource allocation, anomaly detection, and contextual response in surveillance and mobility monitoring systems. Therefore, the predictive regression models described herein not only fulfill their primary purpose but also act as enablers for scalable, low-latency, and adaptive continuous query clustering architectures discussed in chapter 4.

REFERENCES

[1]. IA-LSTM: Interaction-Aware LSTM for Pedestrian Trajectory Prediction - arXiv, accessed on April 21, 2025, https://arxiv.org/html/2311.15193v2
[2]. LSTM-based Flight Trajectory Prediction - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/publication/328399562_LSTM-based_Flight_Trajectory_Prediction
[3]. AIS-Based Intelligent Vessel Trajectory Prediction using Bi-LSTM - ResearchGate, accessed on April 21, 2025, https://www.researchgate.net/publication/358879212_AIS-Based_Intelligent_Vessel_Trajectory_Prediction_using_Bi-LSTM
[4]. Multidynamic Graph Convolutional Networks for Vessel Trajectory Prediction | ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A, accessed on April 18, 2025, https://ascelibrary.org/doi/10.1061/AJRUA6.RUENG-1531
[5]. A Novel Trajectory Prediction Method Based on CNN, BiLSTM, and Multi-Head Attention Mechanism - MDPI, accessed on April 21, 2025, https://www.mdpi.com/2226-4310/11/10/822
[6]. Trajectory Prediction for Autonomous Driving: Progress, Limitations, and Future Directions, accessed on April 18, 2025, https://arxiv.org/html/2503.03262v1
[7]. GRU-Based Deep Learning Framework for Real-Time, Accurate, and Scalable UAV Trajectory Prediction - MDPI, accessed on April 18, 2025, https://www.mdpi.com/2504-446X/9/2/
[8]. Research on flight trajectory prediction method based on transformer - SPIE Digital Library, accessed on April 21, 2025, https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13018/1301854/Research-on-flight-trajectory-prediction-method-based-on-transformer/10.1117/12.3024772.full
[9]. Exploring Transformer-Augmented LSTM for Temporal and Spatial Feature Learning in Trajectory Prediction - arXiv, accessed on April 18, 2025, https://arxiv.org/html/2412.13419v1
[10]. Transformer-based model for predicting trajectories in autonomous vehicle–pedestrian conflicts: a proactive approach to road safety - Canadian Science Publishing, accessed on April 18, 2025, https://cdnsciencepub.com/doi/10.1139/cjce-2024-0137
[11]. VECTOR: Velocity-Enhanced GRU Neural Network for Real-Time 3D UAV Trajectory Prediction - MDPI, accessed on April 18, 2025, https://www.mdpi.com/2504-446X/9/1/8
[12]. Using LSTM with Trajectory Point Correlation and Temporal Pattern Attention for Ship Trajectory Prediction - MDPI, accessed on April 18, 2025,

https://www.mdpi.com/2079-9292/13/23/4705

[13]. Bidirectional LSTM-Based Privacy Preserving Method for Trajectory Generation - STEMM Institute Press, accessed on April 21, 2025, http://www.stemmpress.com/uploadfile/202407/5804ddf94111207.pdf

[14]. Ship Trajectory Prediction Model Based on Improved Bi-LSTM | ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A, accessed on April 18, 2025, https://ascelibrary.org/doi/10.1061/AJRUA6.RUENG-1234

[15]. Development and evaluation of bidirectional LSTM freeway traffic forecasting models using simulation data, accessed on April 18, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC8668885/

[16]. Enhancing Next Destination Prediction: A Novel LSTM Approach Using Real-World Airline Data - arXiv, accessed on April 18, 2025, https://arxiv.org/html/2401.12830v2

[17]. Object Location Prediction in Real-time using LSTM Neural Network and Polynomial Regression - arXiv, accessed on April 18, 2025, https://arxiv.org/pdf/2311.13950

[18]. Social LSTM: Human Trajectory Prediction in Crowded Spaces - Stanford Computational Vision and Geometry Lab, accessed on April 18, 2025, https://cvgl.stanford.edu/papers/CVPR16_Social_LSTM.pdf

[19]. Monitoring the Spatiotemporal Trajectory of Urban Area Hotspots Using the SVM Regression Method Based on NPP-VIIRS Imagery - MDPI, accessed on April 18, 2025, https://www.mdpi.com/2220-9964/10/6/415

[20]. Vessel Trajectory Prediction Model Based on AIS Sensor Data and Adaptive Chaos Differential Evolution Support Vector Regression (ACDE-SVR) - MDPI, accessed on April 18, 2025, https://www.mdpi.com/2076-3417/9/15/2983

[21]. Vessel trajectory classification via transfer learning with Deep Convolutional Neural Networks - PMC, accessed on April 18, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11346933/

[22]. Movement trajectory classification using supervised machine learning - DiVA portal, accessed on April 18, 2025, http://www.diva-portal.org/smash/get/diva2:1376904/FULLTEXT01.pdf

[23]. Data-Driven 4D Trajectory Prediction Model Using Attention-TCN-GRU - MDPI, accessed on April 18, 2025, https://www.mdpi.com/2226-4310/11/4/313

[24]. 1D Convolutional Neural Network-based Chlorophyll-a Retrieval Algorithm for Sentinel-2 MultiSpectral Instrument in Various Trophic States - Sensors and Materials, accessed on April 18, 2025, https://sensors.myu-group.co.jp/sm_pdf/SM3448.pdf

[25]. TPTrans: Vessel Trajectory Prediction Model Based on Transformer Using AIS Data - MDPI, accessed on April 21, 2025, https://www.mdpi.com/2220-9964/13/11/400

[26]. Encoding Agent Trajectories as Representations with Sequence Transformers - arXiv, accessed on April 18, 2025, https://arxiv.org/html/2410.09204v1

[27]. A TDV attention-based BiGRU network for AIS-based vessel trajectory prediction - PMC, accessed on April 18, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10090245/

[28]. A CNN-LSTM Architecture for Marine Vessel Track Association Using Automatic Identification System (AIS) Data - MDPI, accessed on April 18, 2025, https://www.mdpi.com/1424-8220/23/14/6400?type=check_update&version=1

[29]. RC-TL: Reinforcement Convolutional Transfer Learning for Large-scale Trajectory Prediction - BORIS, accessed on April 18, 2025, https://boris.unibe.ch/166954/14/Emami_RC_TL_preprint.pdf

[30]. VT-Former: A Transformer-based Vehicle Trajectory Prediction Approach For Intelligent Highway Transportation Systems - arXiv, accessed on April 18, 2025, https://arxiv.org/html/2311.06623v2

[31]. A Ship Trajectory Prediction Method Based on an Optuna–BILSTM Model - MDPI, accessed on April 18, 2025, https://www.mdpi.com/2076-3417/14/9/3719

[32]. (PDF) A Ship Trajectory Prediction Method Based on an Optuna–BILSTM Model, accessed on April 18, 2025, https://www.researchgate.net/publication/38

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

0198215_A_Ship_Trajectory_Prediction_Method_Based_on_an_Optuna-BILSTM_Model

[33]. The challenges for multi-physics validation experiments for trajectory simulations of high-speed vehicles, accessed on April 18, 2025, https://arc.aiaa.org/doi/pdfplus/10.2514/6.2025-0503

[34]. Challenges of Spatio-Temporal Trajectory Data Use: Focus Group Findings from the 1st International Summer School on Data Science for Mobility - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/publication/371099579_Challenges_of_Spatio-Temporal_Trajectory_Data_Use_Focus_Group_Findings_from_the_1st_International_Summer_School_on_Data_Science_for_Mobility

[35]. FHWA-HRT-21-071: Trajectory Investigation for Enhanced Calibration of Microsimulation Models, accessed on April 18, 2025, https://www.fhwa.dot.gov/publications/research/operations/21071/21071.pdf

[36]. Efficient Trajectory Extraction and Parameter Learning for Data-Driven Crowd Simulation - GAMMA - The University of North Carolina at Chapel Hill, accessed on April 18, 2025, http://gamma.cs.unc.edu/REACH/GI/GI15Bera.pdf

[37]. Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns | Request PDF - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/publication/280230023_Trajectory_data_reconstruction_and_simulation-based_validation_against_macroscopic_traffic_patterns

[38]. Bias in Data: What Embeddings Reveal About Real vs Synthetic Data Distribution - Voxel51, accessed on April 18, 2025, https://voxel51.com/blog/bias-in-data-what-embeddings-reveal-about-real-vs-synthetic-data-distribution/

[39]. Spatial-Temporal Attentive LSTM for Vehicle-Trajectory Prediction - MDPI, accessed on April 21, 2025, https://www.mdpi.com/2220-9964/11/7/354

[40]. Forecasting S&P 500 Using LSTM Models - arXiv, accessed on April 18, 2025, https://arxiv.org/html/2501.17366v1

[41]. Sequential Learning of Movement Prediction in Dynamic Environments using LSTM Autoencoder - arXiv, accessed on April 18, 2025, https://arxiv.org/pdf/1810.05394

[42]. Learning PyTorch by Examples (5): Sequence Prediction with LSTM and GRU, accessed on April 18, 2025, https://jinli.io/en/p/learning-pytorch-by-examples-5-sequence-prediction-with-lstm-and-gru/

[43]. (PDF) Human trajectory prediction using LSTM with Attention ..., accessed on April 18, 2025, https://www.researchgate.net/publication/373641901_Human_trajectory_prediction_using_LSTM_with_Attention_mechanism

[44]. Trajectory prediction with LSTM : r/learnmachinelearning - Reddit, accessed on April 18, 2025, https://www.reddit.com/r/learnmachinelearning/comments/1ei9dud/trajectory_prediction_with_lstm/

[45]. Long Short-Term Memory Neural Network for Financial Time Series - arXiv, accessed on April 18, 2025, https://arxiv.org/pdf/2201.08218

[46]. Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models - arXiv, accessed on April 18, 2025, https://arxiv.org/pdf/2009.10819

[47]. Real-time Short- Term Trajectory Prediction Based on GRU Neural Network - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/publication/341077180_Real-time_Short-_Term_Trajectory_Prediction_Based_on_GRU_Neural_Network

[48]. Deep learning innovations in South Korean maritime navigation: Enhancing vessel trajectories prediction with AIS data | PLOS One, accessed on April 18, 2025, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0310385

[49]. Prediction of Defensive Player Trajectories in NFL Games with Defender CNN-LSTM Model - Amazon Science, accessed on April 18, 2025, https://assets.amazon.science/8f/31/de231564410aa55e346d02c34c12/prediction-of-defensive-player-trajectories-in-nfl-games-with-defender-cnn-lstm-model.pdf

[50]. The proposed 1D CNN-LSTM architecture. The proposed hybrid architecture... | Download Scientific Diagram - ResearchGate, accessed on April 18, 2025,

https://www.researchgate.net/figure/The-proposed-1D-CNN-LSTM-architecture-The-proposed-hybrid-architecture-takes-an-input_fig3_372375110
[51]. Convolutional Neural Network for Trajectory Prediction - CVF Open Access, accessed on April 18, 2025, https://openaccess.thecvf.com/content_ECCVW_2018/papers/11131/Nikhil_Convolutional_Neural_Network_for_Trajectory_Prediction_ECCVW_2018_paper.pdf
[52]. Architecture of the one dimensional convolutional neural networks ..., accessed on April 18, 2025, https://www.researchgate.net/figure/Architecture-of-the-one-dimensional-convolutional-neural-networks-1D-CNN-model-for_fig3_343341551
[53]. Hyperparameter tuning for the hybrid 1D CNN LSTM architecture. - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/figure/Hyperparameter-tuning-for-the-hybrid-1D-CNN-LSTM-architecture_tbl3_372375110
[54]. Optimization of LSTM Ship Trajectory Prediction Based on Hybrid Genetic Algorithm, accessed on April 18, 2025, https://www.sciopen.com/article/10.11947/j.JGGS.2024.0306
[55]. Hyperparameter Tuning of the LSTM model for Stock Price Prediction - International Journal of Intelligent Systems and Applications in Engineering, accessed on April 18, 2025, https://ijisae.org/index.php/IJISAE/article/download/6274/5074/11397
[56]. Hyper parameter tuning LSTM network on time series data : r/learnmachinelearning - Reddit, accessed on April 18, 2025, https://www.reddit.com/r/learnmachinelearning/comments/1fbav2b/hyper_parameter_tuning_lstm_network_on_time/
[57]. MarlonCajamarca/Keras-LSTM-Trajectory-Prediction: A Keras multi-input multi-output LSTM-based RNN for object trajectory forecasting - GitHub, accessed on April 18, 2025, https://github.com/MarlonCajamarca/Keras-LSTM-Trajectory-Prediction
[58]. Vehicle Trajectory Prediction Using LSTMs With Spatial–Temporal Attention Mechanisms | Request PDF - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/publication/349128017_Vehicle_Trajectory_Prediction_Using_LSTMs_with_Spatial-Temporal_Attention_Mechanisms
[59]. Application of an ANN and LSTM-based Ensemble Model for Stock Market Prediction - arXiv, accessed on April 18, 2025, https://arxiv.org/html/2410.20253v3
[60]. A Stock Prediction Method Based on Heterogeneous Bidirectional LSTM - MDPI, accessed on April 18, 2025, https://www.mdpi.com/2076-3417/14/20/9158
[61]. A Novel Multivariate Bi-LSTM model for Short-Term Equity Price Forecasting - arXiv, accessed on April 18, 2025, https://arxiv.org/html/2409.14693v1
[62]. Long term Trajectory prediction using Bi-LSTM - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/figure/Long-term-Trajectory-prediction-using-Bi-LSTM_fig11_359435217
[63]. What's the difference between a bidirectional LSTM and an LSTM? - Stack Overflow, accessed on April 18, 2025, https://stackoverflow.com/questions/43035827/whats-the-difference-between-a-bidirectional-lstm-and-an-lstm
[64]. Vehicle Destination Prediction Using Bidirectional LSTM with Attention Mechanism - MDPI, accessed on April 18, 2025, https://www.mdpi.com/1424-8220/21/24/8443
[65]. Comparing Long Short-Term Memory (LSTM) and bidirectional LSTM deep neural networks for power consumption prediction - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/publication/374618094_Comparing_Long_Short-Term_Memory_LSTM_and_bidirectional_LSTM_deep_neural_networks_for_power_consumption_prediction
[66]. Visualization of trajectory prediction by Bi-LSTM and comparison models - ResearchGate, accessed on April 18, 2025, https://www.researchgate.net/figure/Visualization-of-trajectory-prediction-by-Bi-LSTM-and-comparison-models_fig8_359435217
[67]. Full article: A novel spatio-temporal attention-based bidirectional LSTM model for moisture content prediction in drying process - Taylor & Francis Online, accessed on April 18, 2025, https://www.tandfonline.com/doi/full/10.1080/07373937.2024.2406441?af=R

[68]. A Destination Prediction Algorithm using Spatial Temporal Bidirectional LSTM Networks - CEUR-WS, accessed on April 18, 2025, https://ceur-ws.org/Vol-2498/short57.pdf

[69]. [R] [D] Sanity Check on use of biLSTM for time series prediction : r/MachineLearning - Reddit, accessed on April 18, 2025, https://www.reddit.com/r/MachineLearning/comments/1dl3lui/r_d_sanity_check_on_use_of_bilstm_for_time_series/

[70]. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., & Huang, Y. (2010). **T-drive: Driving directions based on taxi trajectories**. *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 99–108. https://doi.org/10.1145/1869790.1869807