SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V12I9P101

Volume 12 Issue 9, 1-10, Semtember 2025

Original Article

Human Trajectory Forecasting with RNN-Based Hybrid Models: LSTM, GRU, and SimpleRNN Combinations

Ahmad Zaki Aiman Abdul Rashid[1], Azita Laily Yusof[1*], Norsuzila Ya'acob[1]

[1]*Faculty of Electrical Engineering, Universiti Teknologi MARA (UiTM), Selangor, Malaysia.*

[1*]*Corresponding Author : azita968@uitm.edu.my*

Received: 01 July 2025 Revised: 03 August 2025 Accepted: 02 September 2025 Published: 30 September 2025

Abstract *- For 5G aerial base station (UAV-BS) networks to have a smooth and dependable handover, human trajectory prediction is crucial. Although the majority of research focuses on individual architectures rather than hybrid approaches, deep learning models such as SimpleRNN, GRU and LSTM demonstrated potential in modeling sequential data. There are currently few comparative studies of hybrid designs, especially when dropout regularization is used. The SimpleRNN-Dropout-LSTM-Dropout model produced the best results out of all of them after 50 epochs of training with Tanh activation, Adam optimizer, learning rate of 0.001, 64 hidden units, and batch size of 32. With little training and validation loss (0.0007 each), it recorded the lowest errors across all metrics: MSE (0.0003), MAE (0.0146), ADE (0.0207), FDE (0.0434), RMSE (0.0168), and MAPE (0.0157). These results demonstrate how well hybrid deep recurrent networks - in particular, SimpleRNN-LSTM combinations - perform in accurately predicting short-term human trajectories.*

Keywords *- Coordinate, Deep Learning, Hybrid, Human, Trajectory Prediction.*

1. Introduction

In a variety of applications, including intelligent tracking systems by anticipating suspicious behavior [2], reducing the frequency of user handovers in 5G networks [3], and safe navigation and collision avoidance in self-driving cars and mobile robots [1], human trajectory prediction is essential. In the context of mobility management in telecommunication, trajectory prediction is increasingly important for improving handover performance, especially in 5G networks as users move rapidly between small, densely deployed base stations. Frequent and inefficient handovers can degrade network quality and user experience. However, because human motion is nonlinear, dynamic, and even surprising, trajectory prediction is still difficult. Recurrent Neural Networks (RNNs), a recent development in deep learning, have demonstrated significant promise in modeling time-series data, including human trajectories [4]. The ability to learn temporal correlations has been demonstrated for the Long Short-Term Memory (LSTM) [5], Gated Recurrent Units (GRU) [6], and SimpleRNN designs. However, choosing the best architecture and component combination for trajectory prediction is still up for debate, particularly when it comes to striking a balance between training effectiveness and prediction accuracy. While some research has investigated more intricate designs, including attention processes [38] or graph-based models [7], others have suggested employing standalone LSTM [5] or GRU [6] models for trajectory prediction. The systematic comparison of hybrid recurrent models, which incorporate several RNN variations and dropout layers to improve generalization and robustness, has received less attention. This paper compared six hybrid deep learning architectures developed for predicting a single human trajectory consisting of different sequences of RNN layers, such as SimpleRNN, LSTM, and GRU, integrating dropout regularization. The models are trained and evaluated on normalized coordinate sequences under two epoch conditions (50 and 500), and their performances are assessed using multiple evaluation metrics, including MSE, MAE, ADE, FDE, RMSE, and MAPE. The combination of SimpleRNN-Dropout-LSTM-Dropout, which shows the novelty of this research, was trained for 50 epochs and yielded the lowest error rates across all measures, according to simulation findings. The rest of the paper is organized as follows. Section 2 describes the related works of human trajectory prediction, and Section 3 outlines the methodology used for conducting the research, including dataset generation, data normalization, sequence creation, and the evaluation metrics used. Section 4 presents experimental results and performance comparisons, and Section 5 concludes the paper with key findings and future work directions.

2. Related Works

Numerous sectors, such as smart cities and telecommunications, utilize trajectory prediction in their applications in order to understand behavioral patterns and provide real-time responses. Conventional methods such as

Hidden Markov Models (HMMs) [40], Kalman filters [41], and ARIMA [42] were among the first approaches that offered simple and effective processes for predicting trajectories. However, conventional methods commonly struggle to adapt to real-world environments, which consist of complex scenarios such as dense urban mobility or highly dynamic environments.

The conventional method of predicting trajectory was upgraded with the utilization of data-driven, machine learning approaches such as Random Forest (RF) [43], Gaussian Processes [44], and Support Vector Machines (SVM) [45] to process nonlinear patterns in both vehicle and human trajectories. However, these models commonly require features that are specifically made and are unable to scale efficiently to long sequential data. Due to this limitation, Deep Learning (DL) models were developed and adopted for predicting trajectories that are capable of learning temporal dependencies and spatial correlations directly from raw datasets.

Among DL methods, Recurrent Neural Networks (RNNs) and the upgraded versions, such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), have interested multiple researchers in predicting trajectories due to their capabilities of capturing temporal correlations in sequential data [3, 5, 38]. Multiple research studies have been conducted applying these models for predicting pedestrian trajectory, vehicle path prediction, and UAV trajectory prediction. Among the three models, LSTM-based models have shown greater accuracy in crowd trajectory prediction, while GRU-based approaches have been preferred for faster convergence and minimum computational cost applications.

In order to enhance prediction accuracy, hybrid-based deep learning designs were proposed, advancing single-model architectures. CNN-LSTM [11] was developed to predict the surrounding vehicle trajectories, Social-Grid LSTM [12] to forecast pedestrian trajectories for intelligent driving systems applications, Knowledge Graph Convolutional Network (KGCN)-LSTM [13] for predicting the trajectory of a vehicle to enhance the transportation system, and Time-to-Collision (TTC)-SLSTM [14] for forecasting human trajectories in highly dense scenarios.

Trajectory prediction has also attracted researchers in wireless communication systems, especially for UAV-assisted networks and 5G handover management. By predicting User Equipment (UE) mobility patterns, handover algorithms that integrate trajectory predictions minimize handover latency, lower the likelihood of failure, and enhance Quality of Service (QoS) [8, 9]. Trajectory knowledge makes it possible to optimize resource allocation, minimize interference, and maintain uninterrupted connectivity in automotive and UAV communication systems [10]. Using Kalman filters, Markov models, or LSTM-based prediction, a good deal of research has investigated trajectory-aware handovers; however, the majority concentrates on large-scale or vehicular mobility rather than the fine-grained movement of individual humans.

Two significant gaps still exist in spite of these advancements. First, there is little comparative analysis among hybrid recurrent architectures; most of the work that has already been done focuses on either single-architecture models or more specialized designs (such as social interaction modeling). Despite the potential for more generic solutions for a variety of mobility contexts, there are few systematic assessments of hybrid RNN configurations that combine LSTM, GRU, and SimpleRNN with regularization processes like dropout. Second, although a lot of research has been done on predicting the mobility of vehicles and unmanned aerial vehicles, comparatively less has been done on forecasting the trajectory of a single human, which is crucial for applications like customized handover optimization in networks of aerial base stations.

By comparing six hybrid RNN-based designs that incorporate LSTM, GRU, and SimpleRNN layers with dropout regularization, this paper fills in these research gaps. In order to determine the best architecture for precise single-human trajectory prediction, the models are evaluated across a variety of error metrics and training settings. This has obvious implications for predictive-aware handover methods in UAV-assisted wireless networks.

3. Methodology

This Section describes the methodology utilized for producing this research, consisting of coordinate generation, where a single user trajectory in a scenario of overlapping signal coverage between two aerial base station was generated; the data normalization and sequence generation section, where the data is normalized, scaled, and inserted into a function for generating a sequence, the building of six hybrid DL models for predicting trajectory; and the evaluation metrics for evaluating the prediction accuracy.

3.1. Coordinate Generation

For predicting the pedestrian trajectory, the coordinates were generated using the Python programming language to produce a single human movement from a starting point to the end point. In this case, a scenario of two aerial base stations (UAV-BS) that produce 5G signal coverage to humans on the ground overlaps to show a handover region between the overlapping coverage.

Figure 1 below shows the illustration of the scenario depicting the serving aerial base station (UAV-BS1) in a green circle, the target aerial base station (UAV-BS2) in a red circle, the starting point of the user in a black dot, the movement of the user in a blue dot, and the endpoint of the user in a red dot. The radius of coverage at both aerial base stations (UAV-BS) is 1 km. When generating the user coordinate, the starting

point starts at (-500, 0) and ends at (550, 100) with an iteration of 1000 points using Python's Numpy linspace function. The single user coordinate generated is then saved into a CSV file using Python's Pandas library to be inserted into the hybrid DL model for the trajectory prediction process.

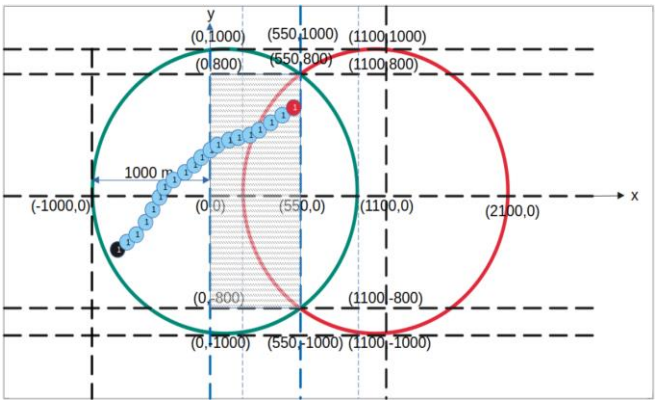

Fig. 1 Scenario of a single user movement from the starting point to the endpoint

3.2. Data Normalization and Sequence Creation

Before inserting the coordinates into the hybrid DL model, the coordinates will first go through the normalization process. The CSV file containing the single user coordinates is converted into an array using Python's Numpy module with the shape (*no. of samples*, 2). Then, using Python's Scikit-Learn *MinMaxScaler* was initialized for scaling the data to be in the range of [0,1], based on the minimum and maximum of each feature (e.g x and y coordinates).

The scaler is fit to the data and transforms it. The MinMaxScaler fit_transform function consists of fit, which computes the minimum and maximum values of the coordinates, and transform, which scales all values to the range [0,1] using the formula in Equation (1) [15].

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (1)$$

The output from the MinMaxScaler consists of coordinates that are normalized. Data normalization is important before feeding the data into machine learning models, especially those sensitive to the scale of input data, such as neural networks. There are several key reasons for implementing data normalization, such as improving the model accuracy [16], ensuring data consistency and interpretability [17], and enhancing computational efficiency [18]. Once the data has been normalized, it is then inserted into the function for creating sequences. Trajectories are time-ordered data; thus, predicting a single future point ignores how past and present positions influence future movement.

These dependencies are learned by sequence-based models like RNN, LSTM, and GRU, which produce predictions that are more realistic and accurate [19-21].

3.3. Hybrid DL Model Building

In this Section, the hybrid DL model was built using Python's Tensorflow Keras layers module, which consists of six combinations of RNN-based models such as SimpleRNN, LSTM, and GRU. Figure 2 below shows the combination of different DL models, and Table 1 shows the configuration for each of the hybrid models. Figure 2(a) consists of Model (0) and Model (1) with a combination of LSTM-Dropout-LSTM-Dropout, SGD optimizer, and a learning rate of 0.1. Figure 2(b) shows Model (2) and Model (3) with a combination of SimpleRNN-Dropout-LSTM-Dropout, the Adam optimizer, and a learning rate of 0.001. Figure 2(c) shows Model (4) and Model (5) containing SimpleRNN-Dropout-LSTM-Dropout with SGD optimizer and a learning rate of 0.001. Figure 2(d) displays Model (6) and Model (7) with a combination of GRU-Dropout-LSTM-Dropout, SGD optimizer, and a learning rate of 0.001. Figure 2(e) displays Model (8) and Model (9) with LSTM-Dropout-GRU-Dropout, SGD optimizer, and a learning rate of 0.001. Figure 2(f) displays Model (10) and Model (11) with a combination of LSTM-Dropout-SimpleRNN-Dropout, an SGD optimizer, and a learning rate of 0.001.

Layer (type)	Output Shape
LSTM	(None, 10, 64)
Dropout	(None, 10, 64)
LSTM	(None, 64)
Dropout	(None, 64)
Dense	(None, 2)

(a) Model (0) and Model (1)

Layer (type)	Output Shape
SimpleRNN	(None, 10, 64)
Dropout	(None, 10, 64)
LSTM	(None, 64)
Dropout	(None, 64)
Dense	(None, 2)

(b) Model (2) and Model (3)

Layer (type)	Output Shape
Simple RNN	(None, 10, 64)
Dropout	(None, 10, 64)
LSTM	(None, 64)
Dropout	(None, 64)
Dense	(None, 2)

(c) Model (4) and Model (5)

Layer (type)	Output Shape
GRU	(None, 10, 64)
Dropout	(None, 10, 64)
LSTM	(None, 64)
Dropout	(None, 64)
Dense	(None, 2)

(d) Model (6) and Model (7)

Layer (type)	Output Shape
LSTM	(None, 10, 64)
Dropout	(None, 10, 64)
GRU	(None, 64)
Dropout	(None, 64)
Dense	(None, 2)

(e) Model (8) and Model (9)

Layer (type)	Output Shape
LSTM	(None, 10, 64)
Dropout	(None, 10, 64)
Simple RNN	(None, 64)
Dropout	(None, 64)
Dense	(None, 2)

(f) Model (10) and Model (11)

Fig. 2 Combination of DL model

Table 1. Configuration for a combination of DL model

DL. No.	Epoch	Optimizer	L. Rate	Hidd. Units	Batch Size
0	50	SGD	0.1	64	32
1	500	SGD	0.1	64	32

2	50	Adam	0.001	64	32
3	500	Adam	0.001	64	32
4	50	SGD	0.001	64	32
5	500	SGD	0.001	64	32
6	50	SGD	0.001	64	32
7	500	SGD	0.001	64	32
8	50	SGD	0.001	64	32
9	500	SGD	0.001	64	32
10	50	SGD	0.001	64	32
11	500	SGD	0.001	64	32

3.4. Evaluation Metrics

This Section discusses the evaluation measures used to assess the accuracy of the prediction. Equation (2) illustrates Mean Absolute Error (MAE), a commonly used metric for evaluating the precision of prediction models by measuring the average absolute differences between expected and actual values. When compared to other error metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), it is especially valued for its interpretability [22].

Among its many uses, MAE was used to assess models that predicted rice production based on economic activity and population increase [23]. In deep learning, MAE has also been used as a loss function and has shown benefits over MSE, especially when dealing with non-Gaussian errors [24].

$$\mathrm{MAE}=\frac{1}{n}\sum_{\mathrm{i}=1}^{n}\left(y_{\mathrm{true,i}}-y_{\mathrm{pred,i}}\right) \quad (2)$$

Mean Squared Error (MSE), as shown in Equation (3), is a statistical measure that quantifies the average squared errors, which are the differences between estimated and actual values. The author(s) in [25] defined MSE as the expectation of the squared deviation of an estimator from the true parameter value. MSE measures how close an estimator is to the true value, indicating the quality of the estimation and prediction process. Predicted coordinates are denoted as y_i^{pred}, the actual coordinates are denoted as y_i^{true} and the total data points is depicted as *n*.

$$\mathrm{MSE}=\frac{1}{n}\sum_{\mathrm{i}=1}^{n}\left(y_{\mathrm{true,i}}-y_{\mathrm{pred,i}}\right)^2 \quad (3)$$

The Root Mean Squared Error (RMSE) can be measured using Equation (4) and is a widely utilized metric for quantifying the differences between predicted values generated by a model and the actual observed values. It is a popular option utilized in multiple applications, such as machine learning and structural dynamics, because it works especially well in situations where a Gaussian error distribution is expected [26, 27]. RMSE is calculated as the square root of the average of the squares of the errors, providing a measure that emphasizes larger errors due to the squaring process, and is often compared with the MAE. While RMSE is optimal for Gaussian errors, MAE is better suited for Laplacian error [28, 29]. This distinction suggests that neither metric is universally superior; their effectiveness depends on the specific error distribution.

$$\mathrm{RMSE}=\sqrt{(\mathrm{MSE})} \quad (4)$$

Mean Absolute Percentage Error (MAPE), as shown in Equation (5), is a commonly used metric to measure the accuracy of prediction or regression models. MAPE is calculated as the average of the absolute percentage errors between predicted and actual values. For each data point, the absolute error is divided by the actual value, and the results are averaged over all data points [30, 31]. Some MAPE advantages are scale-independence, which can be used to compare forecast accuracy across different datasets because it is expressed as a percentage [32], and the ease of interpretation, where decision-makers can easily understand the magnitude of errors in percentage terms [33].

$$\mathrm{MAPE}=\frac{1}{n}\sum_{\mathrm{i}=1}^{n}\left(\frac{y_{\mathrm{true,i}}-y_{\mathrm{pred,i}}}{y_{\mathrm{true,i}}}\right)*100 \quad (5)$$

The Average Displacement Error (ADE), which calculates the mean difference between projected and actual positions across all time steps or data points in a sequence or dataset, is a frequently used statistic to assess the accuracy of displacement predictions or measurements in a variety of domains. It is widely used to assess the performance of models in tasks like trajectory prediction and displacement sensing [34, 35]. For the calculation, ADE is computed as the average of the Euclidean distances between each predicted point and its corresponding ground truth point [36] as shown in Equation (6).

$$\mathrm{ADE}=\frac{1}{N}\sum_{\mathrm{i}=1}^{N}\sqrt{\left(\left(x_{\mathrm{true}}-x_{\mathrm{pred}}\right)^2+\left(y_{\mathrm{true}}-y_{\mathrm{pred}}\right)^2\right)} \quad (6)$$

Final Displacement Error (FDE) in Equation 33 shows a calculation that determines the distance between the actual endpoint and the prediction endpoint at a particular predicted time. It is widely used to evaluate the accuracy of models that predict movement or paths, such as in autonomous vehicles or pedestrian tracking [37].

$$\mathrm{FDE}=\sqrt{\left(\left(X_{\mathrm{true}}-X_{\mathrm{pred}}\right)^2+\left(y_{\mathrm{true}}-y_{\mathrm{pred}}\right)^2\right)} \quad (7)$$

4. Results and Discussion

4.1. Error Performance and Losses For Different Combinations of DL Models

In this Section, the error performance and losses for different combinations of DL models are discussed thoroughly. The full results for all six hybrid DL models are shown in Table 2.

Table 2. Error performance and losses for hybrid DL models with a linearly generated dataset

Model No.	Epoch	Val_Loss	Train_loss	MSE	MAE	ADE	FDE	RMSE	MAPE
0	50	0.0888	0.0203	0.1895	0.4326	0.6124	0.7208	0.4353	0.4787
1	500	0.0295	0.0077	0.004	0.063	0.0896	0.1282	0.0651	0.0696
2	50	0.0007	0.0007	0.0003	0.0146	0.0207	0.0434	0.0168	0.0157
3	500	0.0002	0.0002	0.0005	0.0202	0.0285	0.0518	0.0219	0.0219
4	50	0.0107	0.0048	0.0497	0.2183	0.3093	0.4185	0.2229	0.2401
5	500	0.0039	0.0023	0.0205	0.1377	0.1982	0.2754	0.1434	0.1512
6	50	0.1075	0.0244	0.2015	0.4465	0.6315	0.7436	0.4489	0.4941
7	500	0.0032	0.0021	0.0057	0.0741	0.1049	0.1441	0.0757	0.0815
8	50	0.0992	0.0242	0.2036	0.4478	0.6347	0.7468	0.4512	0.4955
9	500	0.0029	0.0021	0.0090	0.0916	0.1323	0.1704	0.0947	0.1010
10	50	0.0087	0.0107	0.0264	0.1546	0.2271	0.2899	0.1625	0.1705
11	500	0.0014	0.0030	0.0064	0.0785	0.1111	0.1539	0.0803	0.0863

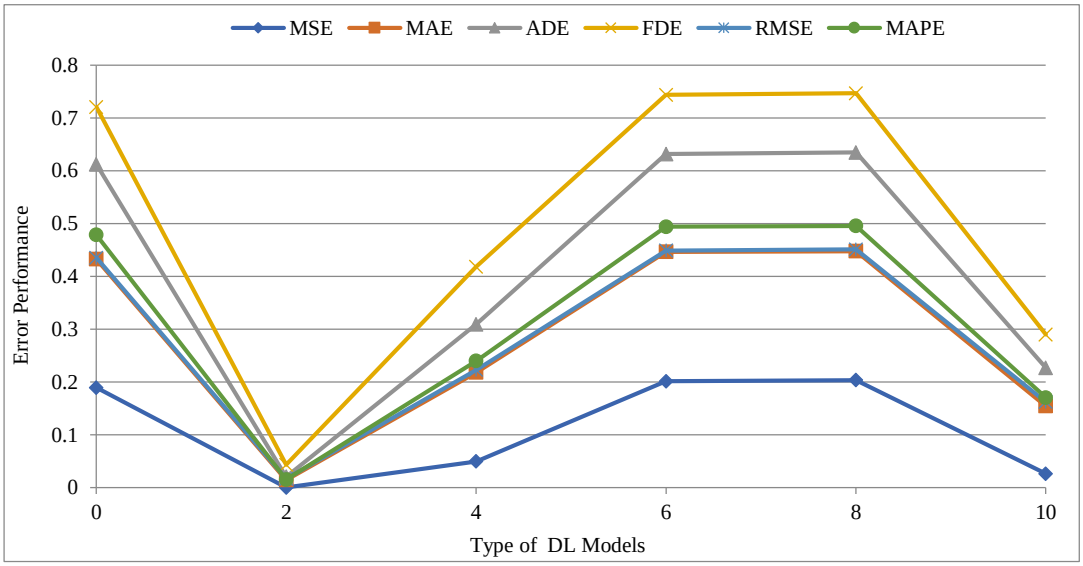

Fig. 3 Error performance for DL models with 50 epochs

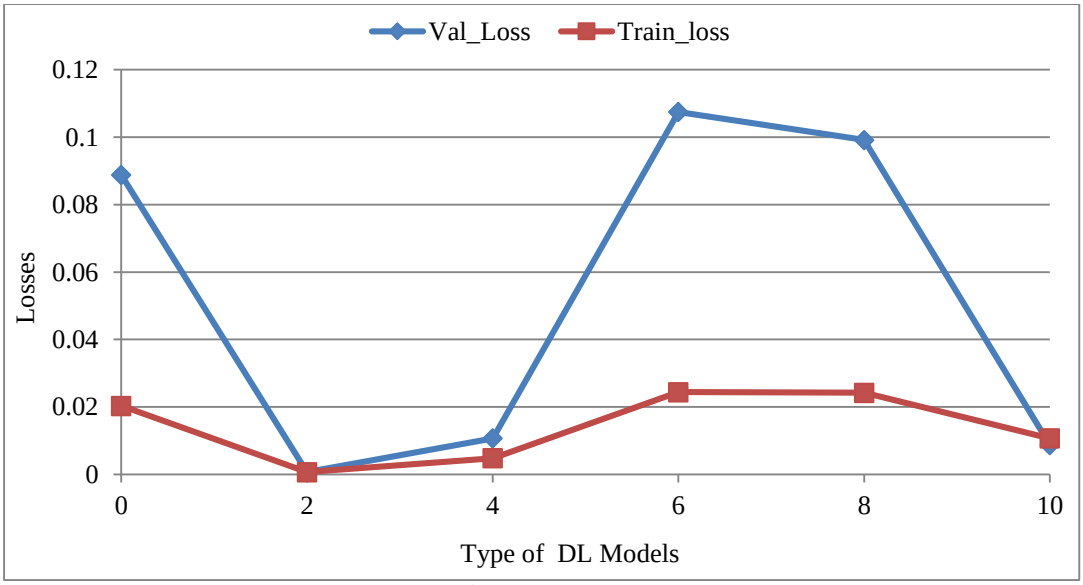

Fig. 4 Losses for DL models with 50 epochs

The error performance and loss performance for 50 epochs were plotted using the scatter-line plotting method, as shown in Figure 3 for error performance and Figure 4 for loss performance. For 50 Epochs, the best model with the lowest error in terms of MSE, MAE, RMSE, ADE, FDE, and MAPE is the 2nd model *(SimpleRNN-Dropout-LSTM-Dropout)* with RMSE of 0.0168, MAE of 0.0146, MSE of 0.003, ADE of 0.0207, FDE of 0.0434, and MAPE of 0.0157, while, the poorest model performance is the 8th model *(LSTM-Dropout-GRU-Dropout)* with RMSE of 0.4512, MAE of 0.4478, MSE of 0.2036, ADE of 0.6347, FDE of 0.7468, and MAPE of 0.4955, as shown in Figure 3 below. For the evaluation of losses, the lowest training loss and validation loss go to the 2nd model (*SimpleRNN-Dropout-LSTM-Dropout*) with training loss of 0.00069401 and validation loss of 0.000062976, while the highest loss goes to the 6th model *(GRU-Dropout-LSTM-Dropout)* with training loss of 0.0244 and validation loss of 0.1075. In conclusion, the best model in terms of minimum error and losses goes to the 2nd model *(SimpleRNN-Dropout-LSTM-Dropout).*

The error performance and loss performance for 500 epochs were plotted using the scatter-line plotting method, as shown in Figure 5 for error performance and Figure 6 for loss performance. For 500 epochs, the lowest error in terms of MSE, MAE, RMSE, ADE, FDE, and MAPE is the 3nd model *(SimpleRNN-Dropout-LSTM-Dropout)* with MSE of 0.0005, MAE of 0.0202, RMSE of 0.0219, ADE of 0.0285, FDE of 0.0518, and MAPE of 0.0219, while, the poorest model performance is the 5th model *(SimpleRNN-Dropout-LSTM-Dropout)* with MSE of 0.0205, MAE of 0.1377, RMSE of 0.1434, ADE of 0.1982, FDE of 0.2754, and MAPE of 0.1512, as shown in Figure 5 below.

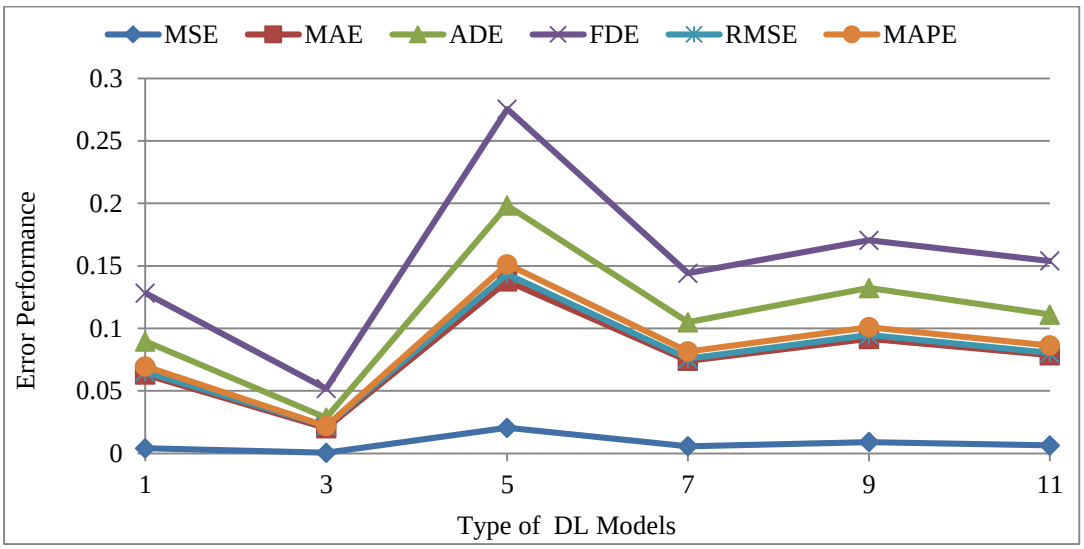

Fig. 5 Error performance for DL models with 500 epochs

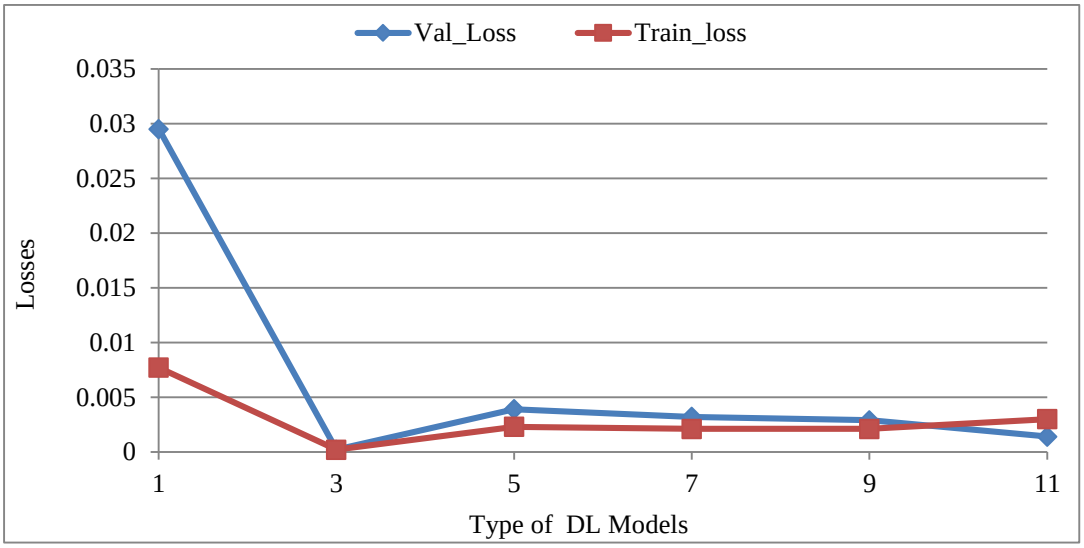

Fig. 6 Losses for DL models with 500 epochs

For the evaluation of losses, the lowest training loss and validation loss go to the 3rd model *(SimpleRNN-Dropout-LSTM-Dropout)* with training loss of 0.00019618 and validation loss of 0.00022697, while the highest loss goes to the 1st model *(LSTM-Dropout-LSTM-Dropout)* with training loss of 0.0077 and validation loss of 0.0295. In conclusion, the best model in terms of minimum error and losses goes to the 3rd model *(SimpleRNN-Dropout-LSTM-Dropout).*

Table 3 shows the hybrid models that have the lowest error and losses, showing the 2nd model *(SimpleRNN-Dropout-LSTM-Dropout)* at 50 epochs and the 3rd model *(SimpleRNN-Dropout-LSTM-Dropout)* at 500 epochs. When comparing the two models, it can be seen that the 2nd model has the lowest error in terms of MSE of 0.0003, MAE of 0.0146, ADE of 0.0207, FDE of 0.0434, RMSE of 0.0168, and MAPE of 0.0157.

Table 3. Best combination of DL models in terms of error performance and losses

Model No.	Epoch	Val_Loss	Train_loss	MSE	MAE	ADE	FDE	RMSE	MAPE
2	50	0.0007	0.0007	0.0003	0.0146	0.0207	0.0434	0.0168	0.0157
3	500	0.0002	0.0002	0.0005	0.0202	0.0285	0.0518	0.0219	0.0219

Table 4. Error performance and losses for the 2nd model with the UCY-ZARA02 dataset

Model No.	Epoch	Val_Loss	Train_loss	MSE	MAE	ADE	FDE	RMSE	MAPE
2	50	0.0015	0.0006	0.0029	0.0425	0.0711	0.1236	0.0537	0.0452

Table 5. Comparison of ADE and FDE between the 2nd model (SimpleRNN-dropout-LSTM-dropout) and [38]

	2nd Model (*(SimpleRNN-Dropout-LSTM-Dropout)*	[38]
ADE	0.0711	0.4827
FDE	0.1236	0.7099

4.2. Comparative Analysis with State-of-the-Art Methods

The 2nd model *(SimpleRNN-Dropout-LSTM-Dropout)* is then compared with other researchers' models using a similar dataset to that used by the researcher. The comparison research is based on [38], which uses Interaction-Aware LSTM (IA-LSTM) to predict pedestrian trajectory, focusing on accurately recreating complex human-human interactions in crowded environments. The study proposes a new correntropy-based method to measure the relative importance of human-human interactions and depict each individual's personal space. The utilization of the interaction module with LSTM allows each pedestrian LSTM to receive interaction information from others in the scene. The disadvantage of this method is that the performance of the IA-LSTM model is highly dependent on the Gaussian kernel size (σ), which requires careful adaptation for different datasets or situations. This might be a useful limitation. The research in [38] utilizes the UCY-ZARA02 dataset [39, 40].

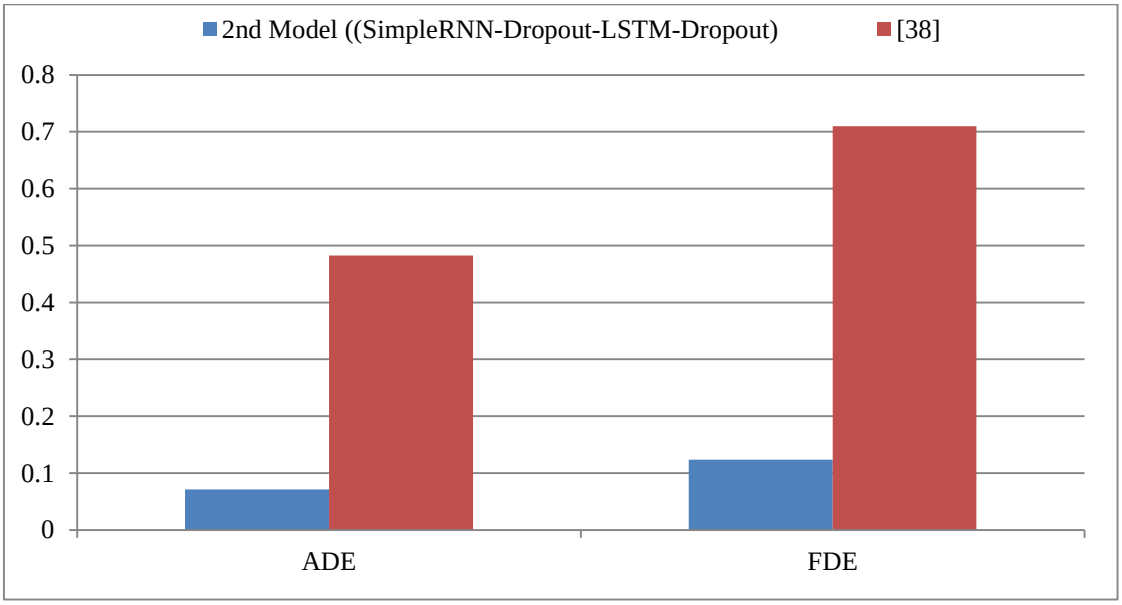

Fig. 7 Comparison of ADE and FDE between the 2nd model and [38]

For standardization purposes, this research will use the UCY-ZARA02 dataset for comparing the proposed model with the model in [38]. Table 4 shows the error performance from the 2nd model resulting from predicting the trajectory using the UCY-ZARA02 dataset. The comparison of prediction accuracy between the 2nd model and the model in [38] was made in terms of ADE and FDE, as shown in Table 5 and Figure 7. As seen in Figure 7, the ADE and FDE for the 2nd model are lower than those of the model in [38]. This suggests that, compared to the model in [38], the second model predicts trajectories with the least error and the highest accuracy. I

t can also be concluded that the hybrid-based DL model displays better prediction accuracy than a single-based model or a model consisting of an interaction module. The primary reasons why Interaction-Aware LSTM (IA-LSTM) produced lower results in terms of ADE and FDE when compared to the proposed hybrid-based models are due to the interaction-aware models' limitations, which tend to indiscriminately incorporate all agents within a predetermined proximity when modeling interactions [46].

This approach substantially escalates computational demands, particularly in scenarios that simultaneously present various pedestrians or vehicles. Another reason would be the interpretability challenges faced by the attention mechanisms and neural network architectures utilized in the IA-LSTM model [46]. This creates a propensity for the model to allocate unreliable correlation coefficients to certain agents, adversely impacting trajectory prediction accuracy.

5. Conclusion

This study investigated the efficiency of hybrid-based deep learning architectures combining LSTM, GRU and SimpleRNN units for the task of single human trajectory prediction. Simulations consist of two training durations (e.g. 50 and 500 epochs) showing that SimpleRNN-Dropout-LSTM-Dropout outperformed other hybrid models. The model demonstrated improved accuracy and reported the lowest error rates in terms of MSE, MAE, ADE, FDE, RMSE, and MAPE after 50 epochs of training with Tanh activation and the Adam optimizer at a learning rate of 0.001. The fact that it demonstrated the lowest training and validation losses further supported its generalization ability. The results suggest that combining both SimpleRNN and LSTM components with dropout regularization can increase forecast accuracy while maintaining processing efficiency. The proposed hybrid model also shows greater prediction accuracy in terms of ADE and FDE when compared to other researchers' models that integrated an interaction module with a DL model. This indicates that hybrid model achieved greater accuracy in forecasting trajectory than single-based or models integrated with an interaction module due to its minimum error. Future work will explore trajectory prediction in more complex environments and extend the methodology to multi-agent scenarios.

Acknowledgment

This paper is part of research work supported by FRGS Grant File no.: FRGS/1/2024/TK07/UITM/02/6 and Faculty of Electrical Engineering, Universiti Teknologi MARA, Shah Alam.

References

[1] Yusheng Peng et al., "MRGTraj: A Novel Non-Autoregressive Approach for Human Trajectory Prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 4, pp. 2318-2331, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Parth Kothari, Sven Kreiss, and Alexandre Alahi, "Human Trajectory Forecasting in Crowds: A Deep Learning Perspective," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7386-7400, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Renata K.G. Dos Reis, Maria G.L. Damasceno, and Jussif J.A. Arnez, "Trajectory-Based Handover Cell Selection Algorithm using GRU Model in 5G Networks," *2024 20th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Paris, France, pp. 603-606, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jianjing Zhang et al., "Recurrent Neural Network for Motion Trajectory Prediction in Human-Robot Collaborative Assembly," *CIRP Annals*, vol. 69, no. 1, pp. 9-12, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Xin Xu, "Context-based Trajectory Prediction with LSTM Networks," *CIIS '20: Proceedings of the 2020 3rd International Conference on Computational Intelligence and Intelligent Systems*, Tokyo, Japan, pp. 100-104, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Yanbo Zhang, and Liying Zheng, "Pedestrian Trajectory Prediction with MLP-Social-GRU," *ICMLC '21: Proceedings of the 2021 13th International Conference on Machine Learning and Computing*, Shenzhen, China, pp. 368-372, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Xin Li, Xiaowen Ying, and Mooi Choo Chuah, "GRIP: Graph-based Interaction-Aware Trajectory Prediction," *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, pp. 3960-3966, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[8] Lei Zhang, Xuefei Chen, and Yuxiang Ma, "A Handover Scheme Based on Mobility Prediction for Autonomous Moving Platforms in 5G Networks," *2022 International Wireless Communications and Mobile Computing, (IWCMC)*, Dubrovnik, Croatia, pp. 104-109, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Anna Prado, Hansini Vijayaraghavan, and Wolfgang Kellerer, "ECHO: Enhanced Conditional Handover Boosted by Trajectory Prediction," *2015 IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, pp. 1-6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Zhongliang Zhao et al., "Mobility Management with Transferable Reinforcement Learning Trajectory Prediction," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2102-2116, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Guo Xie et al., "Motion Trajectory Prediction Based on a CNN-LSTM Sequential Model," *Science China Information Sciences*, vol. 63, no. 11, pp. 1-21, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[12] Bang Cheng et al., "Pedestrian Trajectory Prediction Via the Social-Grid LSTM Model," *The Journal of Engineering*, vol. 2018, no. 16, pp. 1468-1474, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[13] Juan Chen et al., "KGCN-LSTM: A Graph Convolutional Network Considering Knowledge Fusion of Point of Interest for Vehicle Trajectory Prediction," *IET Intelligent Transport Systems*, vol. 17, no. 6, pp. 1087-1103, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Huu-Tu Dang et al., "TTC-SLSTM: Human Trajectory Prediction Using Time-to-Collision Interaction Energy," *2023 15th International Conference on Knowledge and Systems Engineering (KSE)*, Hanoi, Vietnam, pp. 1-6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Lucas B.V. de Amorim, George D.C. Cavalcanti, and Rafael M.O. Cruz, "The Choice of Scaling Technique Matters for Classification Performance," *Applied Soft Computing*, vol. 133, pp. 1-37, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Basilio Gregori Chepino et al., "Effect of MinMax Normalization on ORB Data for Improved ANN Accuracy," *Journal of Electrical Engineering Energy and Information Technology (J3EIT)*, vol. 11, no. 2, pp. 29-35, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Joel Wolfrath, and Abhishek Chandra, "A Biased Estimator for MinMax Sampling and Distributed Aggregation," *arXiv Preprint*, pp. 1-4, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Jida Huang, and Tsz-Ho Kwok, "Normalization and Dimension Reduction for Machine Learning in Advanced Manufacturing," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE)*, St. Louis, Missouri, USA, pp. 1-8, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Shuo Zhao et al., "An Integrated Framework for Accurate Trajectory Prediction Based on Deep Learning," *Applied Intelligence*, vol. 54, no. 20, pp. 10161-10175, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Chujie Wang et al., "Exploring Trajectory Prediction through Machine Learning Methods," *IEEE Access*, vol. 7, pp. 101441-101452, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[21] Yang Li et al., "Pedestrian Trajectory Prediction Combining Probabilistic Reasoning and Sequence Learning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 3, pp. 461-474, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[22] Scott M. Robeson, and Cort J. Willmott, "Decomposition of the Mean Absolute Error (MAE) into Systematic and Unsystematic Components," *PLOS ONE*, vol. 18, no. 2, pp. 1-8, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Andik Adi Suryanto, and Asfan Muqtadir "Application of the Mean Absolute Error (MEA) Method in the Linear Regression Algorithm for Rice Production Prediction," *Saintekbu Journal of Science and Technology*, vol. 11, no. 1, pp. 78-83, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[24] Jun Qi et al., "On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression," *IEEE Signal Processing Letters*, vol. 27, pp. 1485-1489, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[25] Mark D. Schluchter, "Mean Square Error," *Encyclopedia of Biostatistics*, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[26] Sean Reiter, and Steffen W.R. Werner, "Interpolatory Model Reduction of Dynamical Systems with Root Mean Squared Error," *IFAC-PapersOnLine*, vol. 59, no. 1, pp. 385-390, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] T. Chai, and R.R. Draxler, "Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)? - Arguments Against Avoiding RMSE in the Literature," *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247-1250, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[28] Timothy O. Hodson, "Root-Mean-Square Error (RMSE) or Mean Absolute Error (MAE): When to use Them or Not," *Geoscientific Model Development*, vol. 15, no. 14, pp. 5481-5487, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[29] Robert Gilmore Pontius Jr, Olufunmilayo Thontteh, and Hao Chen, "Components of Information for Multiple Resolution Comparison Between Maps that Share A Real Variable," *Environmental and Ecological Statistics*, vol. 15, no. 2, pp. 111-142, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[30] Arnaud de Myttenaere et al., "Mean Absolute Percentage Error for Regression Models," *Neurocomputing*, vol. 192, pp. 38-48, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[31] Sungil Kim, and Heeyong Kim, "A New Metric of Absolute Percentage Error for Intermittent Demand Forecasts," *International Journal of Forecasting*, vol. 32, no. 3, pp. 669-679, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[32] S.K. Morley, T.V. Brito, and D.T. Welling, "Measures of Model Performance Based on the Log Accuracy Ratio," *Space Weather*, vol. 16, no. 1, pp. 69-88, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[33] Ummul Khair et al., "Forecasting Error Calculation with Mean Absolute Deviation and Mean Absolute Percentage Error," *Journal of Physics: Conference Series*, vol. 930, pp. 1-6, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[34] Bohao Shen, and Jianzhi Li, "Displacement-Sensing Method Based on Residual Scaling for One-Shot MMF Specklegram Prediction," *Sensors*, vol. 25, no. 5, no. 1-18, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[35] Yang Zhang, Peng Liu, and Xuefeng Zhao, "Structural Displacement Monitoring Based on Mask Regions with Convolutional Neural Network," *Construction and Building Materials*, vol. 267, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[36] Qiaode Jeffrey Ge et al., "On the Computation of the Average of Spatial Displacements," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, St. Louis, Missouri, USA, vol. 7, no. 46, pp. 1-33, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[37] Abduallah Mohamedet al., "Social-Implicit: Rethinking Trajectory Prediction Evaluation and the Effectiveness of Implicit Maximum Likelihood Estimation," *Computer Vision - European Conference on Computer Vision 2022*, Tel Aviv, Israel, pp. 463-479, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[38] Jing Yang et al., "IA-LSTM: Interaction-Aware LSTM for Pedestrian Trajectory Prediction," *IEEE Transactions on Cybernetics*, vol. 54, no. 7, pp. 3904-3917, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[39] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski, "Crowds by Example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655-664, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[40] Ning Ye et al., "Vehicle Trajectory Prediction Based on Hidden Markov Model," *KSII Transactions on Internet and Information Systems*, vol. 10, no. 7, pp. 3150-3170, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[41] Ling-Xiao Li, Guang-Li Sun, and Jiang-Peng Song, "Target Trajectory Prediction Based on Neural Network and Kalman Filtering," *Proceedings Volume 11342 Applied Optics and Photonics China (Aopc2019), AOPC 2019: AI in Optics and Photonics*, Beijing, China, pp. 127-135, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[42] Zhixian Yan, "Traj-ARIMA: A Spatial-Time Series Model for Network-Constrained Trajectory," *IWCTS '10: Proceedings of the Third International Workshop on Computational Transportation Science*, San Jose, California, pp. 11-16, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[43] Dingyi Yin, "Road Traffic Prediction Based on Base Station Location data by Random Forest," *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, pp. 264-270, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[44] Qun Lim, Kritika Johari, and U-Xuan Tan, "Gaussian Process Auto Regression for Vehicle Center Coordinates Trajectory Prediction," *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, Kochi, India, pp. 25-30, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[45] Ranjeet Singh Tomar, Shekhar Verma, and Geetam Singh Tomar, "SVM Based Trajectory Predictions of Lane Changing Vehicles," *2011 International Conference on Computational Intelligence and Communication Networks*, Gwalior, India, pp. 716-721, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[46] Shiji Huang et al., "Interpretable Interaction Modeling for Trajectory Prediction Via Agent Selection and Physical Coefficient," *arXiv Preprint*, pp. 1-8, 2024. [CrossRef] [Google Scholar] [Publisher Link]