A Flight Trajectory Prediction Method Based on Deep Learning with Attention Mechanism

Yuan-Li Cai* and Zi-Jian Zhou

Faculty of Electronic and Information Engineering, Xi’an Jiaotong University, Xi'an 710049, China

Abstract: This paper addresses the critical challenges in data-driven trajectory prediction for high-speed vehicles, focusing on issues such as training instability, computational inefficiency, and the mismatch between input and output sequence lengths. To overcome these challenges, we propose an attention-augmented Gated Recurrent Unit (GRU) sequence-to-sequence (Seq2Seq) framework that integrates temporal attention mechanisms to selectively emphasize informative historical states. This enhancement enables robust long-horizon trajectory predictions based on limited observational data. The proposed model synergizes the parameter efficiency and reduced complexity of GRUs with the dynamic focus capabilities provided by attention mechanisms, resulting in improved prediction accuracy without imposing significant computational burdens−thereby making the approach well-suited for real-time deployment on resource-constrained platforms. Comparative evaluations against baseline models using Long Short-Term Memory (LSTM) Seq2Seq and conventional GRU Seq2Seq architectures without attention demonstrate a substantial reduction in trajectory prediction errors. Extensive simulation results confirm kilometer-level prediction accuracy, validating both the effectiveness and practical viability of the presented method for high-speed vehicle trajectory prediction.

Keywords: Trajectory prediction, sequence-to-sequence, gated recurrent unit (GRU), attention mechanism, deep learning.

1. INTRODUCTION

The trajectory prediction of high-speed vehicles occupies a pivotal position in both theoretical research and practical applications in the fields of target detection, early warning, interception and guidance. Existing trajectory prediction methods can be broadly categorized into two types: model-based and data-driven approaches. From a methodological perspective, model-based methods predict the target states through mathematical modeling, including function approximation methods [1, 2], analytical methods [3-5], and filtering extrapolation methods [6-8]. The function approximation method estimates the trajectory curve through a combination of multiple functions, but its prediction error tends to diverge over time. Analytical methods rely heavily on prior knowledge and often struggle to obtain analytical solutions. Filtering extrapolation methods, while more practically viable, remain reliant on precise modeling frameworks. Under complex flight scenarios, constructing a unified and accurate mathematical model becomes inherently challenging. Against this backdrop, in engineering applications, multi-model filtering and data-driven methods are more commonly adopted. Although multi-model filtering can improve maneuver recognition and prediction accuracy, its model set design is contingent upon empirical knowledge, requires frequent threshold calibration, and the computational cost is high. Consequently, it remains challenging to meet the real-time and robustness requirements of trajectory prediction [9,10].

In recent years, data-driven approaches based on deep learning have demonstrated significant advantages in time-series prediction [11-14]. Recurrent Neural Networks (RNN) and LSTM have provided new perspectives for flight trajectory prediction by extracting temporal features and identifying underlying patterns [15-17]. More recently, flight trajectory prediction research has increasingly adopted encoder–decoder and attention-based architectures to enhance long-term dependency modeling and robustness under complex maneuvers [18-20]. Nevertheless, achieving a favorable trade-off among prediction accuracy, training stability, and computational efficiency for high-speed vehicles, especially for real-time deployment on resource-constrained platforms, still warrants further investigation.

GRU, as a variant of RNN, are designed to mitigate gradient vanishing and improve the modeling of long-term dependencies during backpropagation [21]. GRU functions similarly to LSTM but has a simpler structure and is easier to train, offering a viable solution to the common efficiency and training stability challenges in deep learning-based data-driven time-series prediction methods. Additionally, the Seq2Seq architecture encodes the hidden states of input sequences into semantic vectors through an encoder and then decodes these semantic vectors into outputs matching the desired output sequence length via a decoder

*Address correspondence to this author at Faculty of Electronic and Information Engineering, Xi’an Jiaotong University, Xi'an 710049, China;
E-mail: ylicai@mail.xjtu.edu.cn

[22,23]. This architecture effectively solves the problem of unequal input and output sequence lengths in neural networks, thereby better meeting practical application requirements.

In summary, to address the prevalent issues of low efficiency, poor training stability, and mismatched input-output sequence lengths in deep learning-based time-series prediction methods, this study proposes a high-speed vehicle trajectory prediction approach that leverages the strengths of the Seq2Seq architecture and GRU, further enhanced by an attention mechanism to alleviate information bottlenecks and highlight critical historical states.

2. PRELIMINARIES AND METHODS

2.1. GRU, Seq2Seq and Attention Mechanism

(1) GRU Network

The GRU network is a variant of the Recurrent Neural Network (RNN) that employs a gated mechanism to control the flow and retention of information. Compared to traditional RNNs, it better handles long-term dependency issues, and relative to LSTM, it has a simpler structure, fewer parameters, and more efficient computation. The basic structure of the GRU is illustrated in Figure **1**.

Formally, given the current input vector x_t and the previous hidden state h_{t-1}, the GRU updates its internal state via two gating units—the update gate (s_t) and the reset gate (r_t)—with the following computations:

$$\begin{cases} s_t = \sigma(W_s \cdot [h_{t-1}, x_t]) \\ r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \\ \hat{h}_t = \tanh(W \cdot [r_t \text{ e } h_{t-1}, x_t]) \\ h_t = (1 - s_t) \text{ e } h_{t-1} + s_t \text{ e } \hat{h}_t \end{cases} \tag{2.1}$$

where, $\sigma(\cdot)$ denotes the Sigmoid activation function and $\hat{h}_t$ represents the candidate hidden state. W_s, W_r and W are the related network weights. e indicates element-wise multiplication.

In this structure, the update gate s_t controls the degree to which the unit state is updated, determining how much historical information is retained, while the reset gate r_t decides how much past information is forgotten when generating the candidate state. Through the dynamic regulation of these two gating mechanisms, GRU can simultaneously model short-term and long-term dependencies without introducing a separate memory cell.

Compared to LSTM, GRU has fewer parameters and a more compact structure, achieving a favorable balance between training speed and generalization performance. It is particularly well-suited for resource-constrained scenarios or tasks requiring real-time response.

(2) Sequence-to-Sequence Architecture

Seq2Seq architecture is a neural network framework designed for sequence modeling tasks, which addresses problems where both input and output are sequences of variable length. The core idea is to map the input sequence into a fixed-dimensional

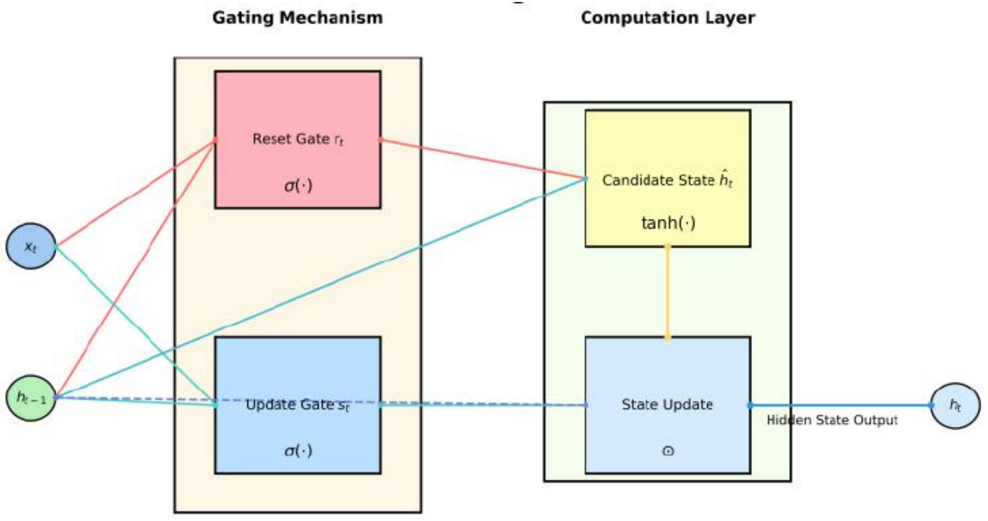

Figure 1: The basic structure of GRU.

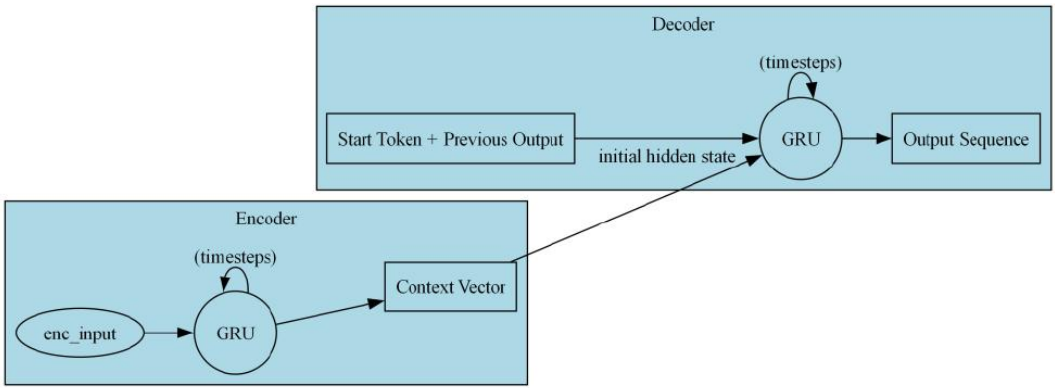

Figure 2: The basic Seq2Seq architecture.

context vector through an encoder, and then generate the output sequence step by step from this vector using a decoder, thereby achieving a mapping from input to output sequences. This architecture was initially applied to Neural Machine Translation (NMT) tasks and was quickly extended to speech recognition, text summarization, dialogue systems, and other domains. The basic Seq2Seq architecture is illustrated in Figure **2**.

Seq2Seq model typically consists of two RNNs or their variants, such as LSTM networks or GRU; in this study, GRU is selected. The encoder receives the input sequence $X = \{x_1, x_2, \ldots, x_n\}$ and iteratively updates the hidden states $h_t = f(x_t, h_{t-1})$, ultimately compressing the information of the entire sequence into the final hidden state h_n, which serves as the context vector c. The decoder uses the context vector c as its initial state and recursively generates the output sequence $Y = \{y_1, y_2, \ldots, y_n\}$, where the output at each time step depends on the previously generated output and the current hidden state, the computation is given by the following formula:

$$s_t = g(y_{t-1}, s_{t-1}, c) \tag{2.2}$$

This end-to-end training approach effectively addresses the issue of unequal input and output sequence lengths in trajectory prediction.

(3) Attention Mechanism

A fixed-length context vector can easily lead to information loss in complex dynamic scenarios. To address this, the attention mechanism is introduced in the deep learning literature, which allows the decoder to dynamically focus on different parts of the historical trajectory at each time step, thereby enhancing its ability to capture nonlinear motion characteristics.

The attention mechanism in deep learning is inspired by the human visual and cognitive system, enabling neural networks to focus on relevant parts of the input data during processing. Its key operation is to compute the relevance weights between the encoder outputs and the decoder state, producing an attention distribution that assigns greater importance to input positions that are more relevant to the current output step. Specifically, in the context of trajectory prediction, the encoder produces a series of hidden states:

$$H = \{h_1, h_2, \ldots, h_n\} \tag{2.3}$$

where, h_n represents the temporal representation of the input trajectory at time step n. For the decoder hidden state s_i at time step i, the attention mechanism computes its relevance with all encoder states to obtain a weight distribution:

$$e_{i,t} = score(s_i, h_t) \tag{2.4}$$

$$\alpha_{i,t} = \frac{\exp(e_{i,t})}{\sum_{k=1}^{n} \exp(e_{i,k})} \tag{2.5}$$

where, $score(\cdot)$ can take the form of a dot product, an additive function, or a learnable multilayer perceptron (MLP). The context vector is then obtained by

$$c_i = \sum_{t=1}^{n} \alpha_{i,t} h_t \tag{2.6}$$

This context vector c_i dynamically aggregates the information from the input trajectory that is most relevant to the current prediction time step, thereby providing the decoder with input that better captures spatiotemporal dependencies. Finally, the decoder generates the output:

$$s_i = g(y_{i-1}, s_{i-1}, c_i) \tag{2.7}$$

In trajectory prediction tasks, the attention mechanism offers the following significant advantages: 1) The model can automatically identify the time steps in the historical trajectory that are most critical for predicting future states. For example, in the moments immediately preceding maneuvering or turning behaviors, the attention weights increase significantly, enhancing the ability of the model to perceive key dynamic changes; 2) Visualization of the attention weight matrix $\alpha_{i,t}$ can reveal the degree of focus the model places on different time steps, thereby assisting us in understanding the basis of the model predictions of the model and the underlying spatiotemporal dependencies. To isolate the contributions of the attention module and the recurrent unit selection, we will present some comparative simulation analysis results in Section 4.

2.2. Trajectory Prediction Framework

Based on the general concept of time-series prediction, the proposed Seq2Seq deep learning neural network model based on GRU and integrated with attention mechanism is shown in Figure **3**.

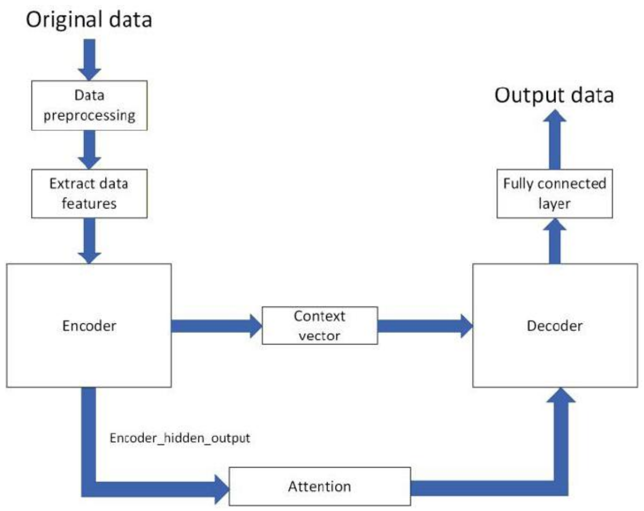

Figure 3: Seq2Seq deep learning neural network based on GRU with an attention mechanism.

In trajectory prediction tasks, the input trajectory is typically a dynamically evolving sequence over time. To capture temporal dependencies within a limited input length and support online prediction, this study employs the Sliding Window Method to segment continuous trajectories. This method uses a fixed-length window along the time axis to extract local segments of the trajectory sequence, thereby constructing multiple temporal samples for model training and prediction.

Formally, let the complete trajectory sequence be denoted as $X = \{x_1, x_2, \mathrm{L}\ , x_N\}$. Given a window length T and a sliding stride s, the input for the k^{th} sliding window is $X_k = \{x_{1+(k-1)s}, x_{2+(k-1)s}, \cdots, x_{T+(k-1)s}\}$. The corresponding prediction target can be defined as $Y_k = \{x_{T+1+(k-1)s}, x_{T+2+(k-1)s}, \mathrm{L}\ , x_{T+\tau+(k-1)s}\}$, where, τ denotes the prediction horizon, k is chosen such that $T + \tau + (k-1)s \le N$. The window slides along the time axis with stride s, thereby generating consecutive input–output sample pairs (X_k, Y_k).

Regarding sliding-window parameter settings, we set $T = 60$ to balance maneuver-context coverage and computational cost for real-time deployment; pilot experiments on a validation split indicate that shorter windows yield insufficient motion context and larger errors, while longer windows provide diminishing gains with increased computation. We consider $\tau \in \{60, 120\}$ to evaluate both moderate- and long-horizon prediction and to quantify error accumulation as the forecast length increases.

The overall trajectory prediction framework is shown in Figure **4**. In order to reduce the difficulty of model training, trajectories that are randomly distributed in space are first normalized through rotation and translation, followed by standard normalization. As shown in the online prediction part of Figure **4**, the predicted points obtained after loading the trained model are inverse-normalized to recover their actual physical scale. Finally, the predicted trajectory in real spatial coordinates is obtained through inverse rotation and translation.

3. DATA COLLECTION AND PREPROCESSING

Deep learning is a data-driven approach. To validate the effectiveness of the previously discussed framework, it is first necessary to collect flight data that reflects the characteristics of high-speed vehicles. Obviously, obtaining a large amount of real flight test data is difficult and costly. Here, we establish the required data through simulation experiments. In order to obtain high-fidelity flight data for high-speed vehicles, we first need to establish the corresponding kinematic and dynamic equations.

3.1. Kinematic and Dynamic Equations

For any high-speed vehicle, it is necessary to consider elements such as Earth's gravity, Earth's

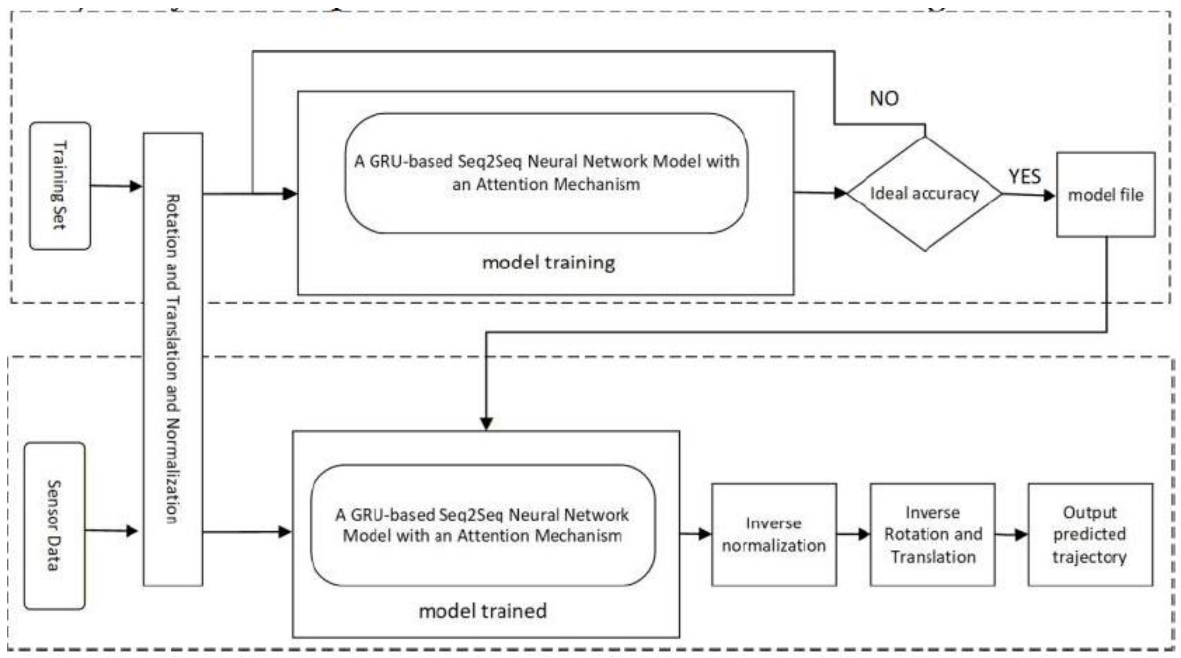
Training Set
Rotation and Translation and Normalization
A GRU-based Seq2Seq Neural Network Model with an Attention Mechanism
model training
NO
Ideal accuracy
YES
model file
Sensor Data
A GRU-based Seq2Seq Neural Network Model with an Attention Mechanism
model trained
Inverse normalization
Inverse Rotation and Translation
Output predicted trajectory

Figure 4: The overall trajectory prediction framework.

rotation, and aerodynamics [16,23,24]. According to the principles of flight mechanics, the kinematic equations of the vehicle can be described as

$$\frac{\mathrm{d}r}{\mathrm{d}t} = V\sin\theta \tag{3.1}$$

$$\frac{\mathrm{d}\lambda}{\mathrm{d}t} = \frac{V\cos\theta\sin\psi}{r\cos\phi} \tag{3.2}$$

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = \frac{V\cos\theta\cos\psi}{r} \tag{3.3}$$

where, r denotes the distance from the vehicle to the Earth's center; λ is the longitude; ϕ is the latitude; V represents the flight speed; θ is the flight path angle; ψ is the heading angle.

If only the unpowered flight segment, i.e. the gliding phase, is considered, the dynamics of the vehicle can be described as

$$\frac{\mathrm{d}V}{\mathrm{d}t} = -\frac{D}{m} - \frac{\mu}{r^2}\sin\theta + \omega_E^2 r\cos\phi \\ (\sin\theta\cos\phi - \cos\theta\sin\phi\sin\psi) \tag{3.4}$$

$$V\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{L\cos\sigma}{m} + \left(\frac{V^2}{r} - \frac{\mu}{r^2}\right)\cos\theta + 2\omega_E\cos\phi \\ \cos\psi + \omega_E^2 r\cos\phi(\cos\theta\cos\phi + \sin\theta\sin\phi\sin\psi) \tag{3.5}$$

$$V\frac{\mathrm{d}\psi}{\mathrm{d}t} = \frac{L\sin\sigma}{m\cos\theta} - \frac{V^2\cos\theta\sin\psi\tan\phi}{r} + \\ 2\omega_E(\tan\theta\cos\phi\sin\psi - \sin\phi) \\ -\frac{\omega_E^2 r}{\cos\theta}\sin\phi\sin\phi\cos\psi \tag{3.6}$$

where, m denotes the vehicle mass; μ represents the gravitational constant; ω_E is the Earth's rotational angular velocity; σ is the roll angle; L and D correspond to the aerodynamic lift and drag, respectively, whose expressions are given by

$$L = \frac{1}{2}\rho V^2 S C_L(\alpha, Ma) \tag{3.7}$$

$$D = \frac{1}{2}\rho V^2 S C_D(\alpha, Ma) \tag{3.8}$$

where, ρ denotes the atmospheric density; S is the reference area of the vehicle; C_L and C_D are the lift and drag coefficients, respectively, both of which are functions of the Mach number Ma and the angle of attack α.

According to Eq. (3.1) to Eq. (3.6), the flight trajectory data of a specific vehicle under specific flight conditions can be simulated and generated. This study focuses on the balanced longitudinal dynamics, with Earth's rotational and Coriolis effects intentionally neglected to ensure simulation tractability and enable efficient generation of large-scale training datasets. However, in operational scenarios where rotational dynamics become significant—including long-duration flights, extended downrange trajectories, high-latitude missions, or maneuvers involving substantial heading changes—the simplified model may produce trajectories that diverge from high-fidelity dynamics, potentially compromising the predictor's generalizability. Future work will therefore prioritize higher-fidelity validation by incorporating Earth rotation and Coriolis effects into the simulation framework, followed by validation against real flight-test data as it becomes available.

When the lift, gravity, and centrifugal force reach equilibrium in the longitudinal plane, the rate of change of the flight-path will be zero. From this, the control equations for equilibrium gliding flight can be derived as follows [16, 25]:

$$\frac{L\cos\sigma}{Vm}+\frac{1}{V}\left(\frac{V^2}{r}-\frac{\mu}{r^2}\right)\cos\theta=0 \tag{3.9}$$

With this equation, given a specific range of initial flight conditions and considering the possible deviations in the overall parameters of the vehicle, as well as potential random disturbances, a large amount of flight trajectory data can be generated through Monte Carlo simulation.

3.2. Data Preprocessing

To reduce the learning difficulty of the prediction network and improve data processing efficiency, the preprocessing of the flight trajectory data is divided into two steps: rotation and translation, followed by normalization.

(1) Spatial Normalization

To address the random spatial distribution of the original trajectories on the ground, this study applies a spatial coordinate transformation to align the samples to a common reference frame. This eliminates directional randomness and provides spatially consistent standardized inputs, thereby reducing model complexity and improving training stability and generalization [26]. The specific implementation steps are as follows:

First, a reference trajectory $\mathbf{x}^0$ is selected, and its geometric center coordinates $\mathbf{x}^0_{center}$ are determined. Along with the geometric center coordinates $\mathbf{x}^i_{center}$ of the i^{th} trajectory in the total *M* trajectories, the covariance matrix and its singular value decomposition (SVD) are computed as follows

$$H_i=\mathbf{x}^i_{center}(\mathbf{x}^0_{center})^{\mathrm{T}}=U_iS_iV_i^{\mathrm{T}} \tag{3.10}$$

where, U_i and V_i are orthogonal matrices, and S_i is a diagonal matrix. Both the rotation matrix R_i and the translation vector l_i for i^{th} trajectory are defined by

$$R_i=U_iV_i^{\mathrm{T}} \tag{3.11}$$

$$l_i=\mathbf{x}^i_{center}(-R_i)+\mathbf{x}^0_{center} \tag{3.12}$$

Finally, the spatially normalized trajectory is obtained as

$$\mathbf{x}^i_{tf}=\mathbf{x}^iR_i+\mathrm{repmat}(l_i,size(\mathbf{x}^i,1),1) \tag{3.13}$$

where, $\mathrm{repmat}(l_i,size(\mathbf{x}^i,1),1)$ denotes the translation vector l_i replicated along the number of columns of $\mathbf{x}^i$.

(2) Data Normalization

Due to the high speed and long flight range, significant numerical differences exist among different feature dimensions. If the raw state variables are directly input to the model, features with larger magnitudes may dominate the training process, diminishing the influence of other features and reducing prediction accuracy. To improve training effectiveness and prediction reliability, this study applies min–max normalization to standardize the sample features. Each trajectory in the dataset is a 6-dimensional time series, and the trajectory set is defined as

$$U=\{\mathbf{x}^i(t)\,|\,i\in[1,M]\cap\text{¢},t\in[1,T]\cap\text{¢}\} \tag{3.14}$$

where, $\mathbf{x}^i(t)=[x^i_E(t),y^i_E(t),z^i_E(t),v^i_x(t),v^i_y(t),v^i_z(t)]$ represents the three-dimensional position and velocity components of the flight vehicle in the Earth-centered and Earth-fixed (ECEF) coordinate system. Each component in $\mathbf{x}^i(t)$ needs to be normalized [16,23]. For example,

$$\bar{x}^i_E(t)=\frac{x^i_E(t)-x^{\min}_E}{x^{\max}_E-x^{\min}_E} \tag{3.15}$$

Table 1: Configuration of Network Hyperparameters

Hyperparameter	Value/Setting
Learning Rate	5e-5
Encoder & Decoder Layers	2 recurrent layers each
Hidden Layer Dimension	128
Optimizer	Adam
Loss Function	Mean Absolute Error (MAE)
Batch Size	64
Training Epochs	100
Sliding Stride (s)	1
Input Sequence Length (T)	60
Output Sequence Length (τ)	60 (Case 1 & 3) / 120 (Case 2 & 4)

where, $x_E^{\min} = \min_{i\in[1,M]\cap¢} \min_{t\in[1,T]\cap¢} x_E^i(t)$, $x_E^{\max} = \max_{i\in[1,M]\cap¢} \max_{t\in[1,T]\cap¢} x_E^i(t)$. Finally, we get the normalized trajectory data as follow

$$\mathbf{y}^i(t) = \bar{\mathbf{x}}^i(t) = [\bar{x}_E^i(t), \bar{y}_E^i(t), \bar{z}_E^i(t), \bar{v}_x^i(t), \bar{v}_y^i(t), \bar{v}_z^i(t)] \tag{3.16}$$

4. SIMULATION RESULTS AND ANALYSIS

The above algorithm and model are implemented in Python 3.7, and the prediction network is built with the PyTorch framework. Both the encoder and the decoder of the proposed deep learning neural network consist of 2 GRU layers, and with a hidden layer dimension of 128. In the training phase, Adam optimizer is adopted, which combines adaptive learning rates with first-moment estimates to effectively enhance convergence speed and generalization capability. Mean Absolute Error (MAE) is defined as the loss function, whose continuity, smoothness, and global differentiability contribute to stable convergence during optimization.

To comprehensively assess the proposed trajectory prediction algorithm, simulation experiments were conducted across four distinct scenarios. For comparative evaluation, two attention-free Seq2Seq baselines with identical encoder–decoder architectures were employed: an LSTM–Seq2Seq model and a conventional GRU–Seq2Seq model. All models underwent training and evaluation under strictly identical experimental conditions, including input–output configurations, data normalization protocols, training/validation splits, and optimization parameters. Unless otherwise specified, the baseline models followed the same training procedure as the proposed model, differing only in recurrent cell type and the absence of attention mechanisms. Hyperparameter specifications are summarized in Table **1**.

4.1. Case 1

In this case, both the input sequence length and the output sequence length are 60. Totally 122,590 samples are stochastically generated, and the dataset is split into training, validation, and test set in 8:1:1 ratio. Three separate models are trained, with each model outputting a single dimension corresponding to the position states x_E, y_E, and z_E. The models are separately trained for 100 epochs with a batch size of 64. The training and validation loss curves of x_E-model are depicted in Figure **5**. It can be seen that the convergence rate is quite fast, and the performance on the training set and the validation set is almost identical.

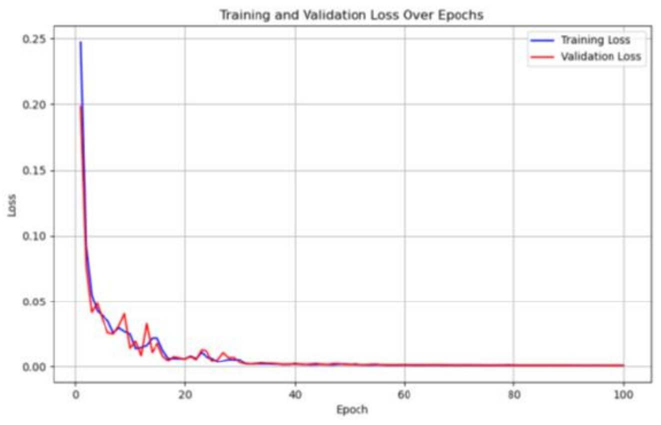

Figure 5: Loss curves of the x_E-mode (Case 1).

For a randomly selected sample from the test set, the prediction results are shown in Figure **6**. It can be seen that the predicted values and the actual values are almost consistent at a macro level. The performance in the other two coordinate directions is comparable, but due to space limitations, it will not be presented here.

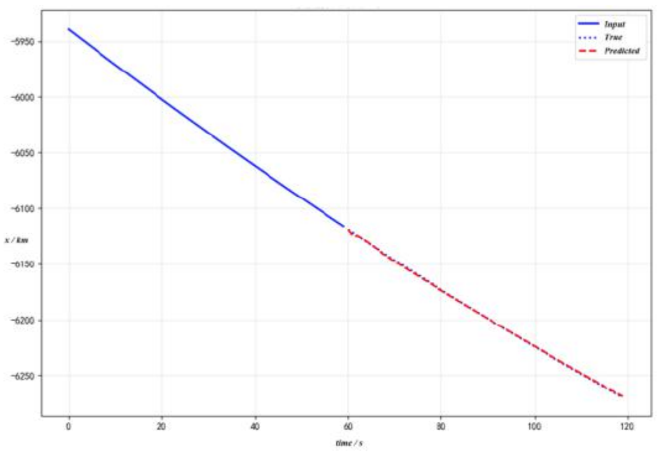

Figure 6: Comparison of predicted x-axis results for a random sample (Case 1).

Table **2** presents the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for Case 1, including comparative results from the two attention-free Seq2Seq baselines. These findings demonstrate that the integration of the attention mechanism enhances prediction accuracy.

Table 2: Prediction Errors on the Test Sets for Case 1

Model	x-MAE/m	y-MAE/m	x-RMSE/m	y-RMSE/m
LSTM–Seq2Seq	2520.75	935.28	3347.56	1577.82
GRU–Seq2Seq	2620.18	972.45	3479.60	1640.62
GRU–Seq2Seq + Attn. (Ours)	2286.48	849.54	3037.66	1433.52

4.2. Case 2

Compared with Case 1, the input sequence length in this case remains 60, while the output sequence length has been extended to 120. In other words, with the same input, a longer trajectory is forecasted. Totally 116,590 samples are similarly generated. The training and validation loss curves of x_E - model are shown in Figure **7**. A comparison of the results from a random test sample is presented in Figure **8**. Clearly, performance declines as the prediction duration extends.

The prediction errors for this case are given in Table **3**, further validating that the proposed method outperforms baseline models. As anticipated, extending the prediction horizon from 60 to 120 time-steps increases errors; however, kilometer-level positional accuracy is consistently maintained.

4.3. Case 3

Unlike Case 1 and Case 2, in this case, a single unified network is trained to predict trajectories simultaneously in three coordinate directions. Both the input and output sequences remain the same as in Case 1, with a length of 60. The total number of samples generated through Monte Carlo simulation amounts to 122,590. Across all test set data, the average prediction error for each point is at kilometer-level. Moreover, simultaneously predicting

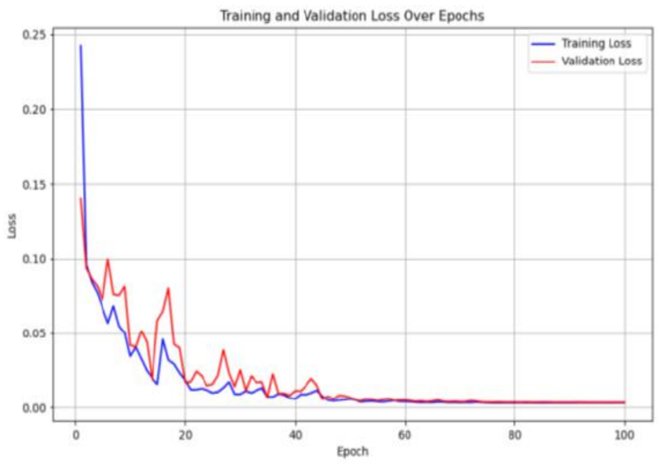

Figure 7: Loss curves of the x_E - mode (case 2).

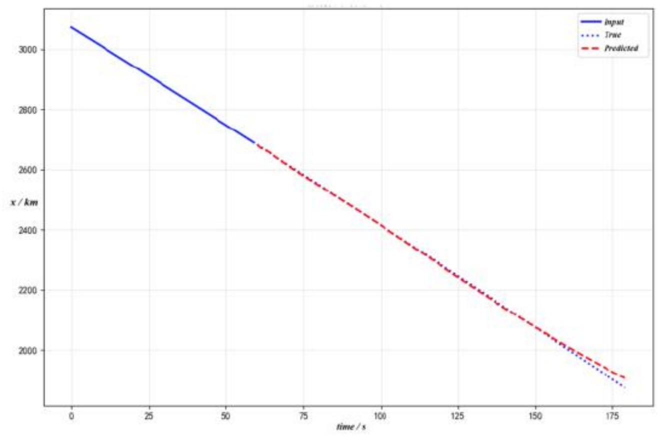

Figure 8: Comparison of predicted x-axis results for a random sample (Case 2).

Table 3: Prediction Errors on the Test Sets for Case 2

Model	x-MAE (m)	y-MAE (m)	x-RMSE (m)	y-RMSE (m)
LSTM–Seq2Seq	3950.62	2130.49	5931.83	3481.22
GRU–Seq2Seq	4150.24	2345.67	6231.66	3832.82
GRU–Seq2Seq + Attn. (Ours)	3633.97	1922.51	5458.91	3140.80

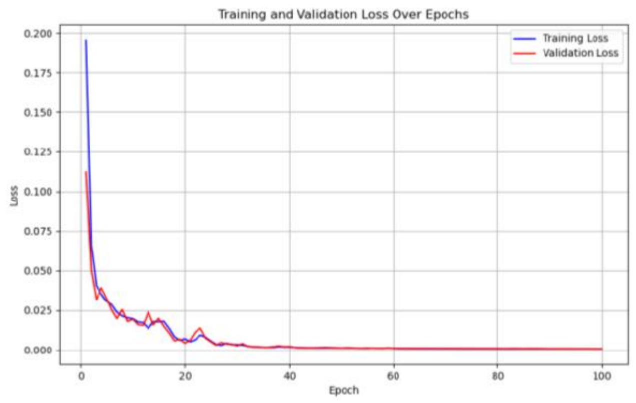

Figure 9: Loss curve of the prediction model (Case 3).

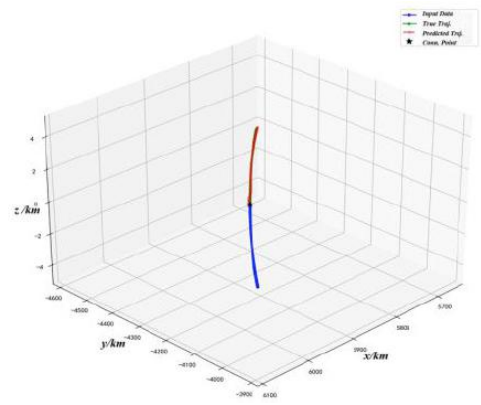

Figure 10: 3D trajectory prediction results (Case 3).

Table 4: Prediction Errors on the Test Sets for Case 3

Model	x-MAE /m	y-MAE /m	z-MAE /m	Average-MAE /m
LSTM–Seq2Seq	1248.90	805.59	0.39e-2	1027.25
GRU–Seq2Seq	1298.85	838.20	0.41e-2	1068.53
GRU–Seq2Seq + Attn. (Ours)	1135.36	732.35	0.35e-2	933.86

$x_E(t), y_E(t)$, and $z_E(t)$ yields better results than predicting them separately (Case 1), with smaller overall prediction errors. The training and validation loss curves are given in Figure **9**, and a random test of three-dimensional trajectory prediction is shown in Figure **10**.

The experimental data of the test set show that the flight trajectories in the three coordinate directions are correlated. At the same time, joint (collaborative) forecasting achieves much higher accuracy than forecasting each coordinate separately (Table **4**). This table further demonstrates that the proposed method is superior to the classic LSTM-Seq2Seq and GRU-Seq2Seq models.

4.4. Case 4

This case serves as an extension of Case 3, involving the prediction of trajectory data with an output sequence length of 120, based on input data with a sequence length of 60. The same as Case 2, the total number of samples is 116,590. A comparison of prediction errors between Case 3 and Case 4 is shown in Table **5**.

The results indicate that, in this long-term trajectory prediction case, the prediction errors are relatively larger. This may be due to the originally set training parameters being insufficient for the demands of this scenario. Adjusting the network structure (e.g., the

Table 5: Comparison of Prediction Errors on the Test Sets for Case 3 and Case 4

	x-axis /m	y-axis /m	z-axis /m	Average/m
MAE (Case 3)	1135.3619	732.3527	0.0035	976.6438
MAE (Case 4)	2165.3720	4312.0507	0.00071	3042.8695
RMSE (Case 3)	2915.5412	1425.5350	0.0071	2327.8198
RMSE (Case 4)	5120.3315	10325.6360	0.0035	8854.8114

Table 6: Comparison of Model Sizes and Training Costs

Model	Params (trainable)	Training time (relative)
LSTM–Seq2Seq	400,259	1.15×
GRU–Seq2Seq	300,291	0.85×
GRU–Seq2Seq + Attention (Ours)	333,315	1.00×

number of GRU layers) or training parameters (e.g., batch size, hidden layer dimension, or number of epochs) could potentially address this issue. For the baseline models, we obtained similar results. Due to space constraints, further discussion is omitted.

Finally, Table **6** presents a comparative analysis of model sizes and training costs between the proposed framework and the typical baseline models, illustrating its practical deployability through computational efficiency metrics.

5. CONCLUSIONS

This paper presents an intelligent prediction framework for high-speed flight trajectories. Simulation results validate that the proposed method consistently captures the inherent characteristics of flight trajectories across diverse scenarios, delivering high prediction accuracy. This performance stems from the integration of a Seq2Seq architecture with attention-augmented GRU neural networks, which enables effective modeling of long-term dependencies and mitigates error accumulation during prediction. Notably, the method achieves these results with a compact network design (≈333k trainable parameters) and only marginal training-cost overhead compared to the baseline GRU–Seq2Seq architecture. These attributes—combined with its computational efficiency—support real-time deployment on resource-constrained platforms such as embedded systems.

REFERENCES

[1] Cheng GZ, Fu XN. Tracking algorithm of hypersonic vehicle based on two-factor enhancement. Dynam Syst Control 2020; 9(2): 81-98. https://doi.org/10.12677/DSC.2020.92008

[2] Luo Y, Tian X, Wang H, *et al*. Trajectory prediction of hypersonic vehicles based on control quantity prediction. In: Proc IEEE Int Conf Information Technology, Networking, Electronic and Automation Control (ITNEC); 2020 Jun 12-14; Chongqing, China. https://doi.org/10.1109/ITNEC48623.2020.9084956

[3] Hu JC, Zhang J, Chen WC. Analytical solutions of steady glide trajectory for hypersonic vehicle and planning application. J Beijing Univ Aeronaut Astronaut 2016; 42: 961-968.

[4] Yu WB, Chen WC. Entry guidance with real-time planning of reference based on analytical solutions. Adv Space Res 2015; 55(9): 2325-2345. https://doi.org/10.1016/j.asr.2015.02.002

[5] Yu W, Yang J, Chen W, *et al*. Analytical trajectory prediction for near-first-cosmic-velocity atmospheric gliding using a perturbation method. Acta Astronaut 2021; 187: 79-88. https://doi.org/10.1016/j.actaastro.2021.06.030

[6] Zeng L, Zhang HB, Zheng W. A three-dimensional predictor-corrector entry guidance based on reduced-order motion equations. Aerosp Sci Technol 2018; 73: 223-231. https://doi.org/10.1016/j.ast.2017.12.009

[7] Mostafa K, Abdel MS, Ahmed MY. Flight simulation and drag prediction for a pitching-accelerating hypersonic reentry vehicle. J Spacecr Rockets 2023; 60(1): 230-238. https://doi.org/10.2514/1.A35441

[8] Xue X, Huang S, Wei D. An adaptive multivariate Student's t-process recursive method for hypersonic glide vehicle trajectory prediction[J]. IET Radar Sonar & Navigation, 2023, 17(6): 1061-1077. https://doi.org/10.1049/rsn2.12400

[9] Xiao CH, Li J, Lei HM, *et al*. Near-space target tracking based on improved IMM-UKF algorithm. J Detect Control 2018; 3: 108-113.

[10] Li JL, Guo J, Tang SJ. Trajectory prediction based on multi-model and multi-intent fusion for hypersonic gliding targets. J Astronaut 2024; 45(2): 167-180.

[11] Kuai JW, Zhao KX, Sun LG, *et al*. A method of space debris reentry time prediction using LSTM neural network based on ballistic coefficient pre-estimation. J Astronaut 2022; 43(12): 1731-1738.

[12] Zuo X, Zhu R, Zhou YK. Online tracking and prediction of slip ring degradation using chaos theory-based LSTM neural network. Meas Sci Technol 2023; 34(5): 055012. https://doi.org/10.1088/1361-6501/acb5b6

[13] Khan M, Wang H, Riaz A, *et al*. Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification. J Supercomput 2021; 77: 7021-7045. https://doi.org/10.1007/s11227-020-03560-z

[14] Florent A, Arnaud LF. An LSTM network for highway trajectory prediction. In: Proc IEEE Intell Transp Syst Conf (ITSC); 2017 Oct 16-19; Yokohama, Japan.

[15] Sun LH, Yang BQ, Ma J. Trajectory prediction in pipeline form for intercepting hypersonic gliding vehicles based on LSTM. Chin J Aeronaut 2023; 36(5): 421-433. https://doi.org/10.1016/j.cja.2023.02.017

[16] Cai YL, Deng YF, Su YH. LSTM-based trajectory classification and prediction method for hypersonic vehicles. In: Proc 21st China Conf System Simulation Technology and Its Applications; 2020. p.303-307.

[17] Li MJ, Zhou CJ, Lei HM, *et al*. An intelligent trajectory prediction algorithm of reentry glide target based on control parameter estimation. Syst Eng Electron 2023; 43(1): 221-233.

[18] Dong X, Tian Y, Dai L, Li J, Wan L. A new accurate aircraft trajectory prediction in terminal airspace based on spatio-temporal attention mechanism. Aerospace 2024; 11(9): 718. https://doi.org/10.3390/aerospace11090718

[19] Xu Y, Pan Q, Wang ZF, Hu BQ. A novel trajectory prediction method based on CNN, BiLSTM, and multi-head attention mechanism. Aerospace 2024; 11(10): 822. https://doi.org/10.3390/aerospace11100822

[20] Zhang Z, Guo D, Zhou S, *et al*. Flight trajectory prediction enabled by time-frequency wavelet transform. Nat Commun 2023; 14: 5258. https://doi.org/10.1038/s41467-023-40903-9

[21] Cho K, van Merriënboer B, Gulcehre C, *et al*. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proc Conf Empir Methods Nat Lang Process (EMNLP); 2014. p.1724-1734. https://doi.org/10.3115/v1/D14-1179

[22] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Adv Neural Inf Process Syst 2014; 27: 3104-3112.

[23] Zhang ZC, Cai YL, Fang YZ. An attention-based Seq2Seq model for predicting the boost-phase trajectory of ballistic missiles. In: Proc 40th Chinese Control Conf (CCC); 2021. p.553-558.

[24] Bao CY, Zhou X, Wang P, *et al*. A deep reinforcement learning-based approach to onboard trajectory generation for hypersonic vehicles. Aeronaut J 2023; 127: 1638-1658. https://doi.org/10.1017/aer.2023.4

[25] Zhang J, Xiong J, Li L, *et al*. Motion state recognition and trajectory prediction of hypersonic glide vehicle based on deep learning. IEEE Access 2022; 10: 21095-21108.

https://doi.org/10.1109/ACCESS.2022.3150830

[26] Berg M de, van Kreveld M, Overmars M, Schwarzkopf O. Computational geometry: algorithms and applications. 3rd ed. Santa Clara: Springer; 2008.

Received on 05-11-2025 Accepted on 03-12-2025 Published on 22-12-2025