Error Prediction Using RNN Models for Differential Global Positioning Systems (RNN-DGPS)

[1]Mary Ann Gliefen A. Bermudo, [2]Ryan P. Himongala, [3]Kenneth D. Ligutom, [4]Rochelle F. Madulara, [5]Reymark-John A. Macapanas, [6]Melody Mae O. Maluya, [7]Jaafar J. Omar, [8]Collien Princess C. Pepito, [9]Jamaica Mae L. Pepito, [10]Nouran M. Usman, and [11]Sherwin A. Guirnaldo
Mindanao State University-Iligan Institute of Technology, Iligan City 9200, Philippines
{maryann.bermudo, ryan.himongala, kenneth.ligutom, rochelle.madulara, reymarkjohn.macapanas, melodymae.maluya, jaafar.omar, collienprincess.pepito, jamaicamae.pepito, nouran.usman, sherwin.guirnaldo}@g.msuiit.edu.ph

***Abstract*—This project aims to improve the accuracy of positioning systems by integrating Differential Global Positioning System (DGPS) technology. Traditional GPS systems often face limitations such as atmospheric disturbances and satellite geometry issues, affecting their precision. DGPS addresses these issues by using reference stations to compute and transmit correction signals, allowing receivers to correct errors and achieve accuracy at the centimeter level. The project focuses on the basic concepts of a Differential Global Positioning System (DGPS), a technique that reduces inaccuracies in GPS readings. It works by setting up a reference station network in key locations. These stations calculate real-time corrections to be transmitted to GPS receivers, which enables accurate positioning. The implementation strategy involves developing a receiver algorithm capable of processing DGPS correction data alongside standard GPS signals. Additionally, it addresses infrastructure requirements for establishing an effective reference station network, considering factors such as station placement and communication protocols.**

***Index Terms*—differential gps, RNN, real-time kinematics**

I. Introduction

The global positioning system (GPS) is a satellite-based navigation system that gives users access to precise and helpful positioning and timing data anywhere in the world via the right equipment. According to [1], GPS receivers' primary challenge is accurately estimating position, velocity, and time from noisy satellite signal measurements.

As a result, this project aims to improve the accuracy of positioning systems by integrating Differential Global Positioning System (DGPS) technology. Traditional GPS systems often face limitations such as atmospheric disturbances and satellite geometry issues, affecting their precision [2]. DGPS addresses these issues by using reference stations to compute and transmit correction signals, allowing receivers to correct errors and achieve accuracy at the centimeter level [3]. The project focuses on the basic concepts of a Differential Global Positioning System (DGPS), a technique that reduces inaccuracies in GPS readings. It works by setting up a reference station network in key locations. These stations calculate real-time corrections to be transmitted to GPS receivers, which enables accurate positioning. The implementation strategy involves developing a receiver algorithm capable of processing DGPS correction data alongside standard GPS signals.

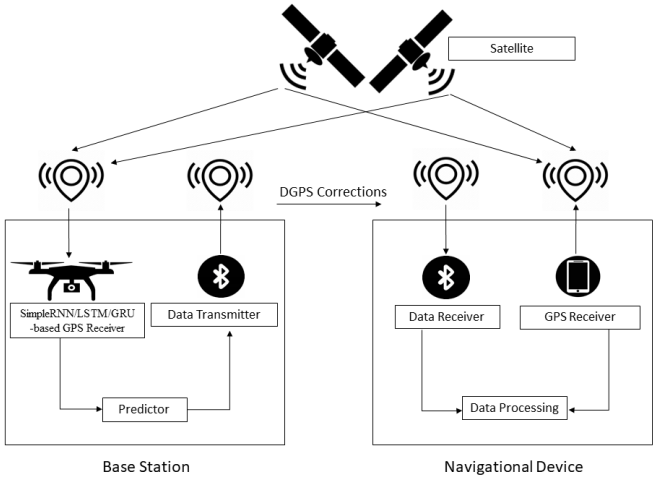

Fig. 1: Proposed positioning system structure.

Here, DGPS works by using a network of fixed reference stations with accurate receivers to compare signals from GPS satellites with the known positions at these reference stations. Discrepancies between calculated and actual positions help determine errors caused by atmospheric conditions and other factors. These error corrections are then sent to mobile DGPS receivers, allowing them to compensate for inaccuracies and provide real-time location information.

Figure 1 illustrates the proposed positioning system structure of this project. The DGPS model is integrated into the drone's onboard computer, where it can receive GPS coordinates. Then, the GPS data received by the drone is subtracted from the known reference point for error calculation.

II. Objective

This project conducted a comparative analysis of four types of RNN models: Simple RNN, LSTM, GRU, and Bi-Directional LSTM. The aim was to evaluate their performance in correcting errors in real-time positioning data, including latitude, longitude, and altitude measurements. This project aims to develop and integrate a deep learning model into the DGPS system to learn and correct errors in real-time

positioning data. In this project, each group utilizes machine learning techniques to improve the accuracy and dependability of location data provided by the DGPS system.

III. Scope and Limitations

- The scope of this project is limited to the application of Recurrent Neural Network (RNN) only in Differential Global Positioning System in line with the requirements of this subject, Artificial Intelligence.
- The correction factor used in this study is calculated from the difference of known location and raw location only.

IV. Conceptual Framework

In the development of Differential Global Positioning Systems on Figure 2, the components include the devices, data transmission system, and GPS error prediction system. This project is only limited to the development of GPS Prediction System only which include the components: Artificial intelligence-aided error prediction model using Recurrent Neural Network (RNN) and User-interface terminal that displays raw coordinates, corrected coordinates, and predicted error.

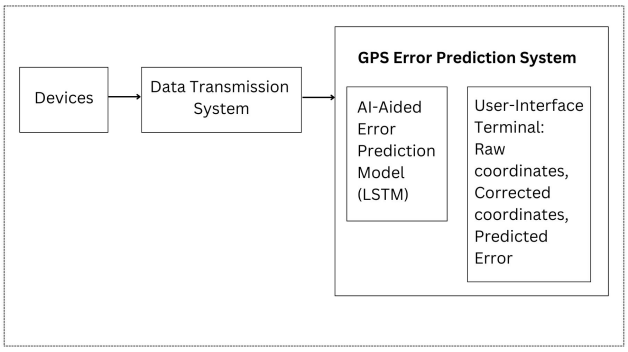

Fig. 2: Differential Global Positioning Systems.

V. Materials

A. Hardware Components

The hardware (as shown in Figure 3) used in this system are as follows:

1) UAV (Unmanned Aerial Vehicle) with Onboard Jetson Nano
   - **Description:** The drone serves as the system's central processing unit, equipped with a Jetson Nano for onboard computation. It reads latitude, longitude, and altitude data from the NEO-M8N GPS Module with Built-in Compass.
   - **Purpose:** The drone processes this data and calculates the error, which is then sent to the ground robot. It also receives error data from the ground robot to further analyze and refine the error correction process.
2) H-RTK F9P

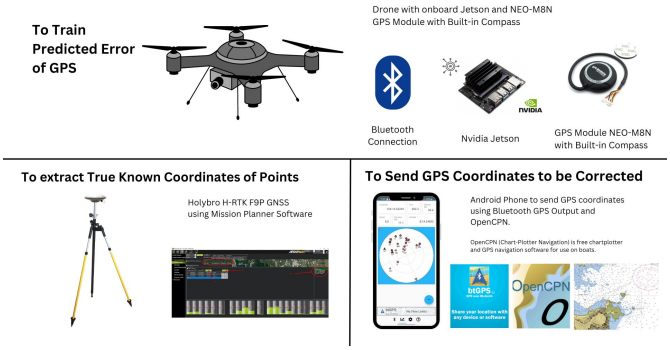

Fig. 3: Components used for the development of the system.

   - **Description**: The H-RTK F9P system gathers high-precision reference data points. It provides centimeter-level accuracy in position data, serving as a reliable reference for error correction.
   - **Purpose**: The RTK system collects reference data points that compare against the data collected by the drone and ground robot. This comparison helps identify and correct errors in the positioning data.
3) Android Phone
   - **Description:** Android phones utilize their built-in GPS (Global Positioning System) capabilities to determine and track the device's location. GPS is a satellite-based navigation system that provides location and time information anywhere on Earth where there is an unobstructed line of sight to four or more GPS satellites.
   - **Purpose:** The android phone uses BT GPS application to extract coordinates and transmit it to computer through COM and bluetooth connection to be visualized in OpenCPN software.

B. Software Components

1) Mission Planner
   - **Description:** Mission Planner retrieves data from the RTK system and assists in mission planning and monitoring. It provides a user-friendly interface for managing mission parameters and accessing real-time telemetry data.
   - **Purpose:** The Mission Planner gathers reference data points from the RTK system, which are essential for error correction. It also aids in mission planning and execution, ensuring efficient data collection and analysis.
2) Drone Kit Library
   - **Description:** The Drone Kit library is integrated into the code snippet for the model deployed on the drone. It provides APIs and libraries for communicating with the drone, accessing telemetry data, and controlling vehicle behavior.

- **Purpose:** The Drone Kit library enables developers to interact with the drone, retrieve sensor data, and implement error correction algorithms. It facilitates communication between the drone and ground robot, allowing for collaborative error analysis and refinement.

3) BT GPS
   - **Description:** Phone application to share GPS location over Bluetooth to nearly any device or software. Used with popular marine and aeronautical charting programs with the comfort of a PC, maps on another tablet, Android Wear or even an open-source Arduino or Raspberry hardware project.
   - **Purpose:** To transmit GPS coordinates to the Jetson Nano onboarded to the drone.
4) OpenCPN
   - Description: OpenCPN is a free software maritime chart plotter and navigation software for use underway or as a planning tool. Developed by a team of active sailors and tested in real world conditions, it has multiple supported chart formats and a variety of data inputs.
   - Purpose: To visualize and receive GPS coordinates received from BT GPS application in Android phone.

VI. Environment Setup

The inclusion of five points with known locations is significant in setting up the controlled environment for developing the DGPS. RTK was set up in these five points and has achieved accuracy in the range 0.5-1.0. These five sets of coordinates from RTK serve as this paper's known locations and reference coordinates. They will aid in calculating the error between known coordinates and raw coordinates. The environmental setup of the five known reference points is depicted in Figure 4, and the error calculation formula is shown below.

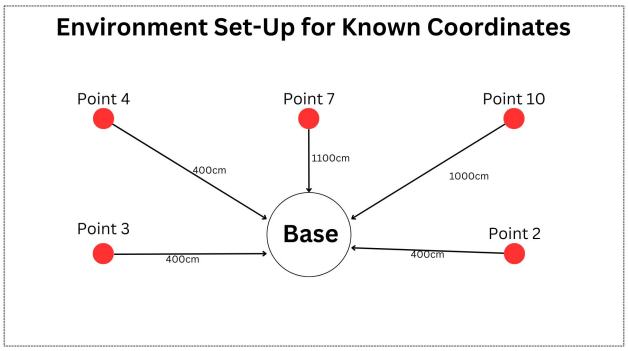

Fig. 4: Environment Setup of Five Known Reference Points.

(1)

The known location is based on data gathered in RTK, and the predicted location is generated by the devices.

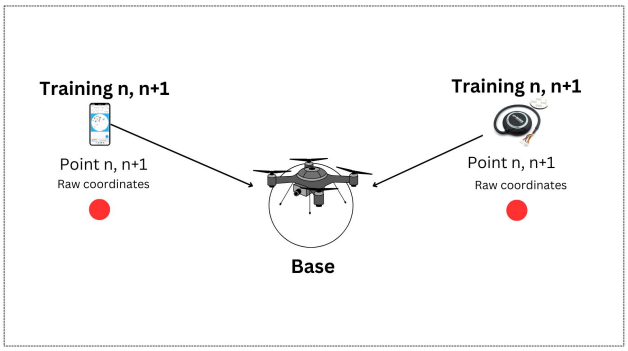

Fig. 5: Model Training Setup.

In training the model to improve the accuracy of GPS coordinates, a systematic approach is adopted to refine the model's predictive capabilities iteratively. This training procedure unfolds across multiple stages involving data collection from known coordinates, error computation, model training, and refinement. The model training setup is shown in Figure 5. Let's delve into the discussion: Firstly, the known coordinates are loaded into the model deployed on the Jetson Nano platform. The drone's GPS module then retrieves its current coordinates, allowing for the error calculation by subtracting the known coordinates from the measured ones. This error is subsequently used to train the model, generating a predicted error output.

In parallel, an Android phone, equipped with a Bluetooth-enabled GPS app, collects coordinates at the specified point. These coordinates are transmitted to the drone's Jetson platform, where they undergo the same error calculation process mentioned earlier. The resulting error is then employed to train the model further, aiming to enhance its predictive accuracy.

This process is repeated for subsequent points, ensuring comprehensive training across different locations. At each point, the known coordinates are loaded into the model, and the drone's GPS module computes the error between the known and measured coordinates. Simultaneously, the Android phone captures coordinates relayed to the drone's Jetson platform for error calculation and subsequent model training.

The overarching objective of this iterative training process is to refine the RNN model's predictive capabilities progressively. By exposing the model to diverse sets of coordinates and their associated errors, it learns to make increasingly accurate predictions. The ultimate goal is to minimize the disparity between the predicted error generated by the RNN model and the true error derived from RTK-based known coordinates.

Through this iterative training paradigm, the authors aim to harness the power of machine learning to improve the accuracy of GPS coordinates generated by the drone. By continually fine-tuning the RNN model based on real-world data, the authors strive to achieve a level of accuracy that enables the correction of GPS coordinates to match the ground truth

provided by RTK-based known coordinates closely.

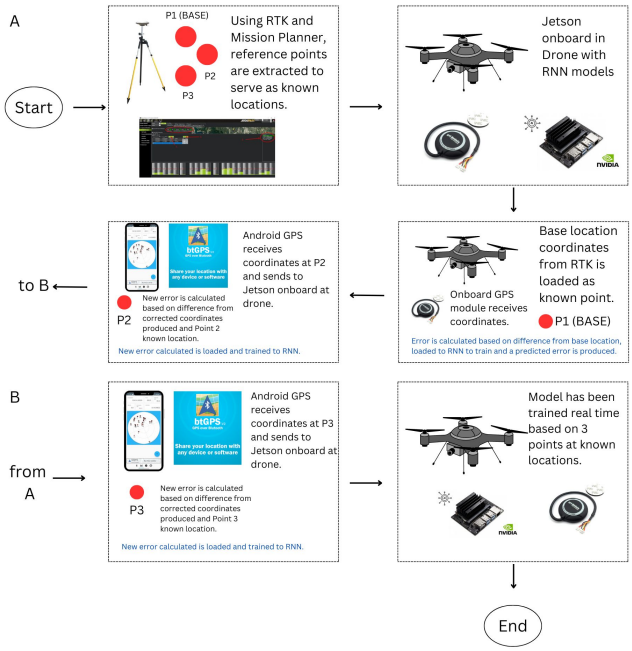

Fig. 6: Data Collection and Experiment Flow.

VII. Time Series Models (LSTM/RNN/GRU/Bi-Directional LSTM)

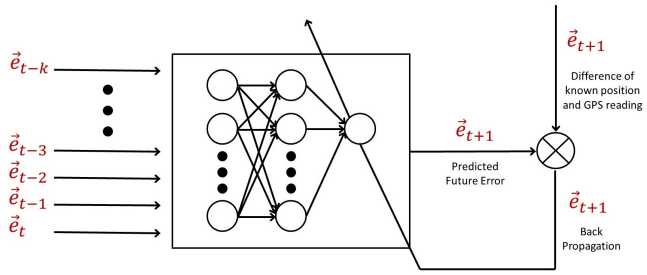

Fig. 7: Model Architecture.

Figure 5 shows the overall model architecture of the DGPS project feeding the error from devices and Figure 8 shows the hyperparameters used among the four RNN models. The RNN models used will be discussed in the next section.

A. Long Short-Term Memory (LSTM)

Long-short-term memory (LSTM) networks are a specialized type of recurrent neural network (RNN) architecture designed to overcome the limitations of traditional RNNs in capturing long-term dependencies in sequential data. Unlike standard RNNs, LSTMs incorporate memory cells that selectively retain or discard information over time, allowing them

	LSTM	Simple RNN	GRU	Bi-Directional LSTM
Input Neurons	10	5	12	8
Hidden Layers	1	1	5	2
Output Neurons	10	5	8	4
Optimizer	RMSProp	RMSProp	RMSProp	RMSProp
Batch Size	32	32	32	32
Time Steps	12	12	12	12
Learning Rate	0.0001	0.0001	0.0001	0.0001
RNN Size	128	128	128	

Fig. 8: Hyperparameters used for four RNN Models.

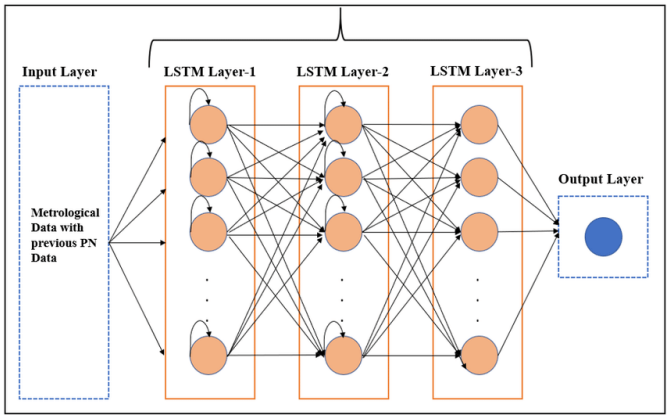

Fig. 9: LSTM Architecture.

to learn intricate patterns and relationships within temporal sequences. This makes LSTMs particularly well-suited for tasks involving time-series data, where understanding historical context is crucial for making accurate predictions about future events.

For several reasons, the authors utilized Long Short-Term Memory (LSTM) networks in this DGPS project. Firstly, LSTM networks are exceptionally adept at handling temporal sequences, which aligns perfectly with the nature of positioning data, particularly in real-time scenarios where location updates are continuous. Through the LSTM's capability to understand and remember long-term dependencies within sequential data, the model can effectively learn from past positioning data and atmospheric conditions to make accurate predictions about future positions. This is crucial in correcting errors that may arise due to factors like atmospheric noises and other environmental variables, which can significantly impact the reliability of location data provided by the DGPS system. Secondly, the flexibility of LSTM networks in handling variable-length sequences is invaluable for the DGPS project. Since DGPS data streams can vary in length due to factors like signal loss or changes in satellite constellations, having a model that can dynamically adapt to these variations ensures robust performance under diverse conditions. Additionally, LSTM networks excel at feature extraction and representation learning, allowing the model to automatically identify

and utilize relevant features from the data, such as latitude, longitude, altitude measurements, and environmental variables. This enhances the accuracy and reliability of the location predictions by capturing the complex relationships between these factors. Moreover, the real-time capabilities of LSTM networks make them ideal for the DGPS project, where timely error correction and adaptation are important. By continuously learning and updating its internal states based on incoming data, the LSTM model can quickly respond to environmental changes and make necessary adjustments to ensure accurate positioning.

B. SimpleRNN

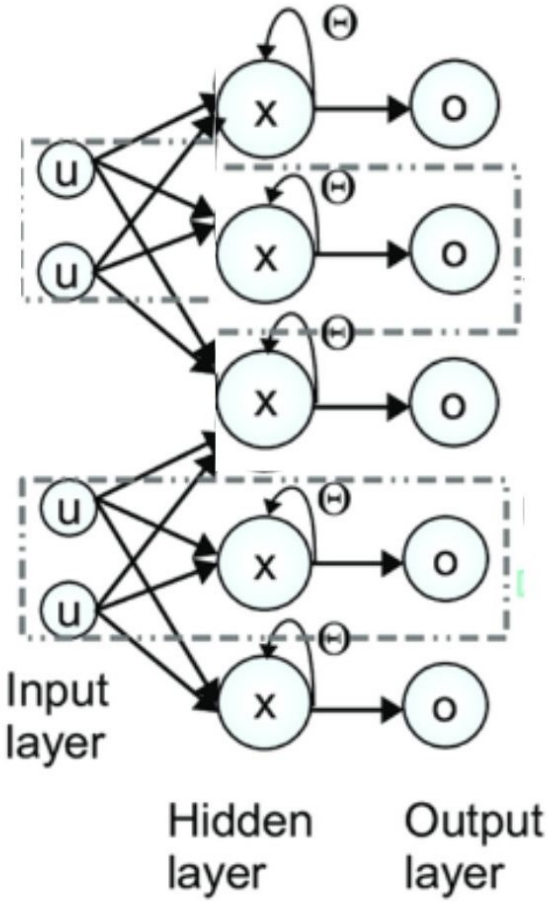

Fig. 10: Simple RNN Architecture.

RNN Recurrent Neural Networks (RNNs) are a cornerstone in modern data processing tasks, particularly in capturing long-term dependencies within sequential data. The SimpleRNN model used in this project has 4 input neurons, 1 hidden layer with 5 neurons, and 5 output neurons. The optimizer used is RMSprop (Root Mean Square Propagation).

In this DGPS project, the authors employed the original simple RNN package due to its alignment with the specific requirements and constraints. This package's simplicity and efficiency offer a straightforward implementation of recurrent neural networks without the additional complexities of more advanced variants like LSTM or GRU. Given the relatively straightforward nature of this DGPS task and the available computational resources, this simplicity fits well with this project's objectives.

By utilizing the original simple RNN package, the authors retain greater control and flexibility over the network architecture and training process. This allows us to tailor the model precisely to the intricacies of this DGPS data and application requirements, fine-tuning parameters and optimizations to achieve optimal performance. Integration into this paper's existing workflow is seamless with minimal changes to the codebase and infrastructure, facilitating rapid development and iteration.

Beyond practical considerations, leveraging the original simple RNN package provides educational value by offering insights into the fundamental principles of recurrent neural networks. It allows team members to deepen their understanding of RNN architectures, training algorithms, and practical considerations in real-world applications like DGPS.

Additionally, the original simple RNN package's resource efficiency is noteworthy. In some instances, the resource requirements of more complex RNN variants such as LSTM or GRU may exceed this DGPS project's available computational resources or constraints. By opting for the simpler package, the authors ensure that this solution remains computationally efficient while meeting the performance objectives.

Ultimately, the decision to use the original simple RNN package underscores this paper's commitment to developing a robust and effective model that enhances the accuracy and reliability of this DGPS system. While considering practical constraints and objectives, this approach allows us to leverage the strengths of recurrent neural networks in a manner that best serves this project's needs.

C. Gated Recurrent Unit

As a robust solution to the vanishing gradient problem, [4] introduced the Gated Recurrent Unit (GRU) based on the LSTM network. The GRU network, a modified version of the LSTM, shares similar design principles and consistently delivers excellent results.

The GRU uses both the update and reset gates to solve the standard RNN's vanishing gradient problem. These gates are vectors that decide what information should be passed to the output. These gates can be trained to keep past information without altering or removing it.

The figure below shows a single unit from the RNN:

- **Update Gate:** This gate allows the model to determine how many data from previous time steps should be passed along in the future. This allows the model to decide to copy all information and remove the risk of the vanishing gradient problem.
- **Reset Gate:** This gate allows the model to decide how much old information to forget.

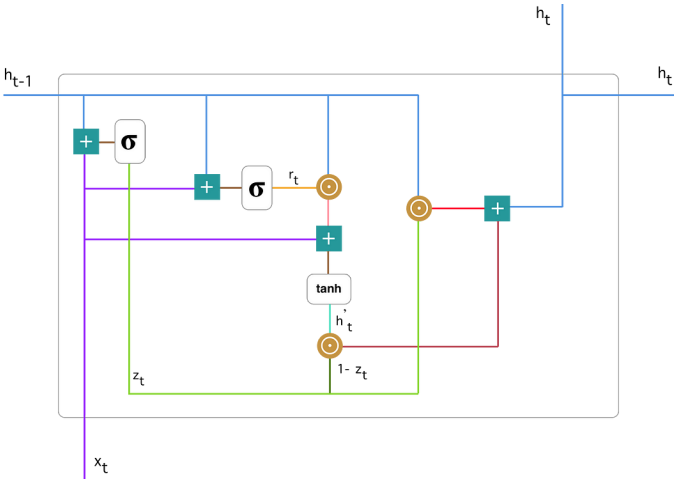

Fig. 11: GRU Architecture.

Fig. 12: Notations of the operations used in GRU.

- **Current memory content:** The current memory content allows the reset gate to store relevant past information. The model used in this project has 12 input neurons and 4 output neurons.

D. Bidirectional LSTM

A bidirectional LSTM (Long Short-Term Memory) unit combines two LSTM networks, one processing the input sequence in forward order and the other in reverse order. This allows the model to capture both past and future context information.

Below is a diagram illustrating a single unit of a bidirectional LSTM:

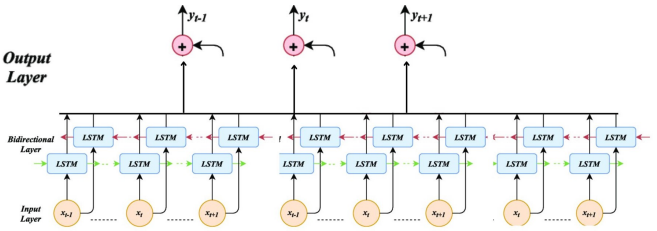

Fig. 13: Bi-Directional LSTM Architecture.

In the diagram:

- The input sequence flows into both the forward and backward LSTM units simultaneously.
- Each LSTM unit processes the input sequence in its respective direction, capturing past and future context information.
- The outputs from both LSTM units are concatenated or combined in some way to produce the final output of the bidirectional LSTM unit.

This architecture allows the model to effectively capture dependencies in both directions, making it suitable for tasks where understanding the context from both past and future information is important, such as natural language processing tasks like sentiment analysis, named entity recognition, and machine translation.

The model used in this project has 8 input neurons and 4 output neurons.

VIII. Correction Factor Equations

To calculate the correction factor of this system, the following formulas were utilized:

$$\Delta_\alpha = \alpha_{base} - \alpha_{ref} \tag{2}$$

$$\Delta_\beta = \beta_{base} - \beta_{ref} \tag{3}$$

$$\Delta_\gamma = \gamma_{base} - \gamma_{ref} \tag{4}$$

where,

- latitude: α
- longitude: β
- altitude: γ
- phone GPS Receiver: p
- Base station GPS Module: $base$
- Reference Point: ref
- Corrected Error: $Corr$

The aforementioned equations were utilized to obtain the correction factor of the raw GPS coordinates through the models investigated in this project. To calculate the error of the latitude, equation 2 is used, where α_{base} is the latitude of the base station while α_{ref} is the latitude of the reference point. Equation 3 is used to calculate the difference between the longitude in the base station β_{base} and the longitude received by the reference point β_{ref}. Equation 4 calculates the difference between the altitude of the base station γ_{base} and the altitude of the reference point γ_{ref}.

To calculate the error correction, the following equations are used:

$$Corr_\alpha = \alpha_p - \Delta_\alpha \tag{5}$$

$$Corr_\beta = \beta_p - \Delta_\beta \tag{6}$$

$$Corr_\gamma = \gamma_p - \Delta_\gamma \tag{7}$$

where,

- phone GPS Receiver: p
- Corrected Error: $Corr$

On the other hand, based on the error calculation equations, the error correction can then be calculated. In equation 5, the error correction can be obtained by subtracting the latitude coordinate received by the mobile phone α_p and the error

calculation of the latitude Δ_α. Similarly, the corrected longitude is achieved by calculating the difference between the navigational device β_p and the calculated longitude error Δ_β. The error correction of the altitude can also be calculated based on the difference between the phone's altitude γ_p and the error calculation of the altitude Δ_γ.

To obtain the true error, the following equations are used:

$$Terr_\alpha = Coref_\alpha - \alpha_p \tag{8}$$

$$Terr_\beta = Coref_\beta - \beta_p \tag{9}$$

$$Terror_\gamma = Coref_\gamma - \gamma_p \tag{10}$$

where,

- True error: $Terr$
- Corresponding Reference: $Coref$

To calculate the true error of the latitude, the corresponding reference will be subtracted from the latitude received by the mobile phone, as shown in equation 8. The true error for longitude can be obtained by subtracting the corresponding reference to the longitude received by the phone (equation 9). Lastly, the true error altitude is calculated based on the difference between the corresponding reference altitude and the altitude read by the mobile phone (equation 10).

IX. Drone with SimpleRNN/LSTM/GRU/Bi-Directional LSTM Model Hardware Set-Up

Shown in figure 14 is the hardware setup with the models attached to the drone.

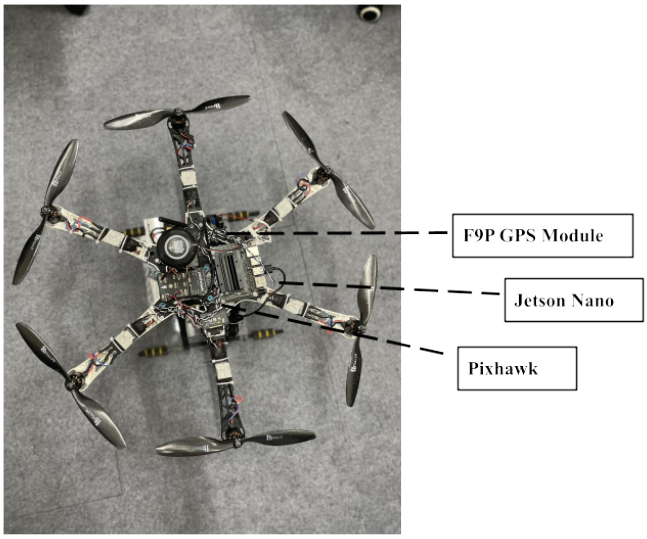

Fig. 14: Drone with Simple RNN/LSTM/GRU Model Set-Up.

X. Experimental Set-Up Flow

The experimental set-up flow starts with the real-time collection of GPS raw coordinates from devices. The collected coordinates will serve as the dataset and be loaded and trained in the RNN model in real time. The predicted error will also be generated in real time. The trained model will be saved. Data can then be plotted using raw, corrected, and known coordinates to be compared in this experimental setup. Figure 15 illustrates the flow of the experimental setup.

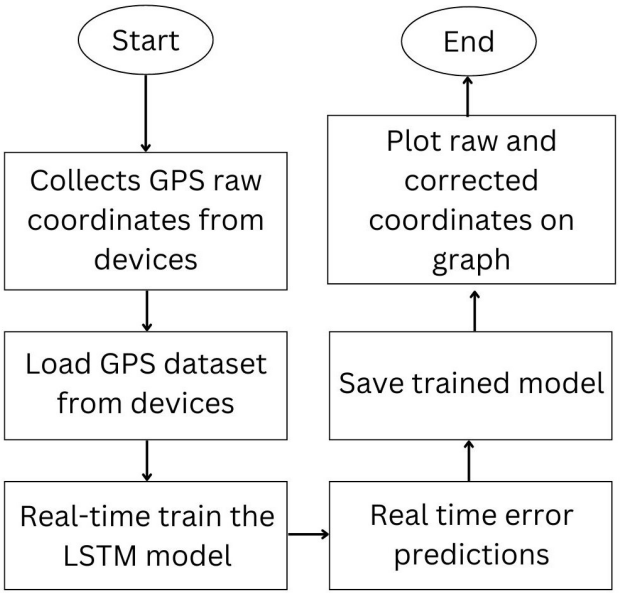

Fig. 15: Experimental Set-Up Flow.

XI. Experimental Set-up Results

The figures below show the experimental results of the five reference points gathered name labeled as ***Reference 2***, ***Reference 3***, ***Reference 4***, ***Reference 7***, and ***Reference 10***. The corrected coordinates (orange line) show a consistent, more filtered, and smoother direction of points compared to the raw coordinates (blue), which achieves the project's goal. The reference is the true label set of coordinates. The graph comparison results of both latitude and longitude from the different models were illustrated from 16 to 25.

XII. Conclusion

The developed GPS Error Prediction System has successfully achieved the objective of applying the RNN models in line with fulfilling the requirements of the Artificial Intelligence subject. Among RNN Models, the long short-term memory (LSTM) model has shown the best performance in error prediction relative to the reference points. The positioning accuracy of this model has achieved a latitude accuracy of 100%, a longitude accuracy of 99%, and an overall accuracy of 99.5%.

This is followed by the simple RNN, achieving a latitude accuracy of 87.24%, a longitude accuracy of 97.78%, and an overall accuracy of 92.51%.

Third, the gated recurrent unit (GRU) exhibited a latitude accuracy of 86.86%, a longitude accuracy of 87.1%, and an overall accuracy of 86.98%.

Lastly, the bidirectional LSTM has the lowest error prediction performance and positioning accuracy. It has an average latitude accuracy of 86%, an average longitude accuracy of 67.118%, and an overall accuracy of 76.56%.

The results of this experimental project contribute to the research efforts of enhancing the accuracy and reliability of location-based data. DGPS is indispensable from ensuring safe navigation in marine and aviation sectors to enabling precision in land surveying, agriculture, and telecommunication. Its transportation, logistics, and emergency response applications further underscore its importance in improving operational efficiency and safety.

References

[1] M. R. Mosavi, "Comparing dgps corrections prediction using neural network, fuzzy neural network, and kalman filter," *GPS Solutions*, vol. 10, no. 2, pp. 97–107, 2006.

[2] X. Meng, P. Vergados, A. Komjathy, and O. Verkhoglyadova, "Upper atmospheric responses to surface disturbances: An observational perspective," *Radio Science*, vol. 54, no. 11, pp. 1076–1098, 2019.

[3] G. Wübbena, A. Bagge, G. Seeber, V. Böder, P. Hankemeier *et al.*, "Reducing distance dependent errors for real-time precise dgps applications by establishing reference station networks," in *PROCEEDINGS OF ION GPS*, vol. 9. Institute of Navigation, 1996, pp. 1845–1852.

[4] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

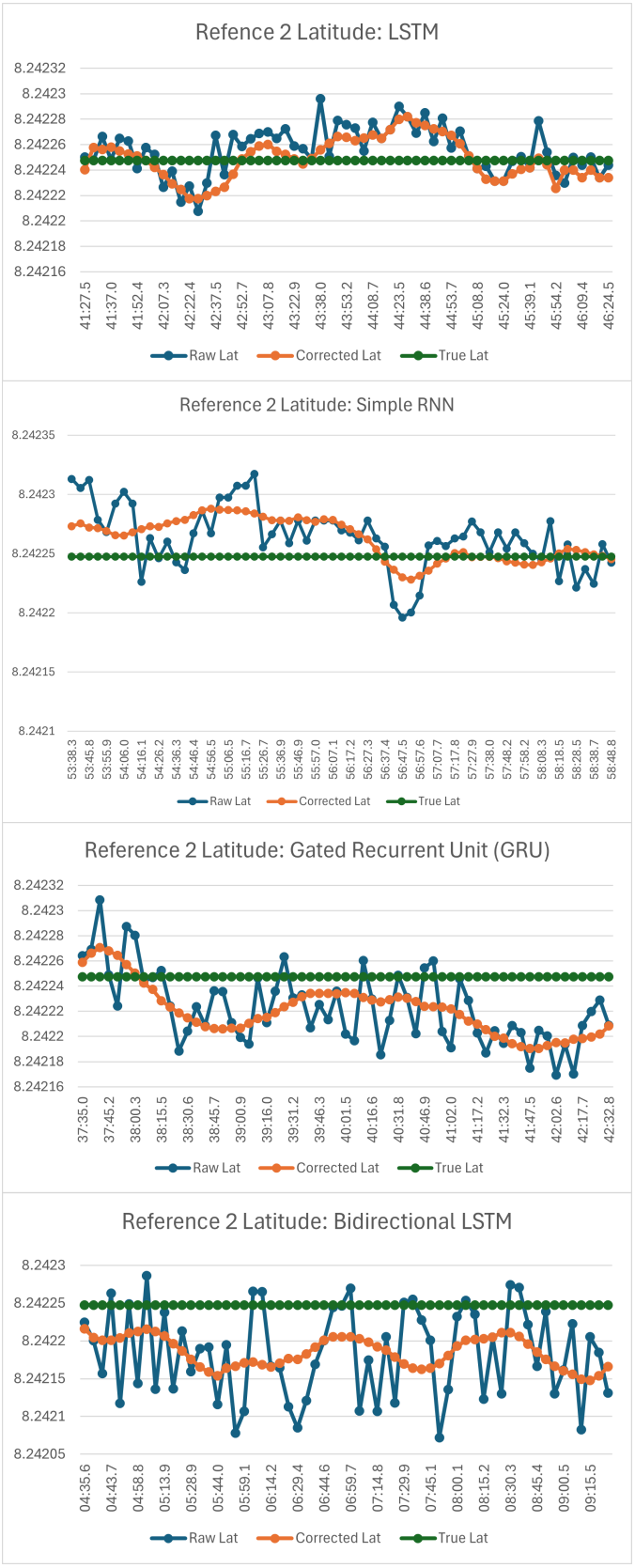

Fig. 16: Performance comparison of the LSTM, Simple RNN, GRU, Bidirectional models between Reference 2's true (known), raw, and corrected latitude points.

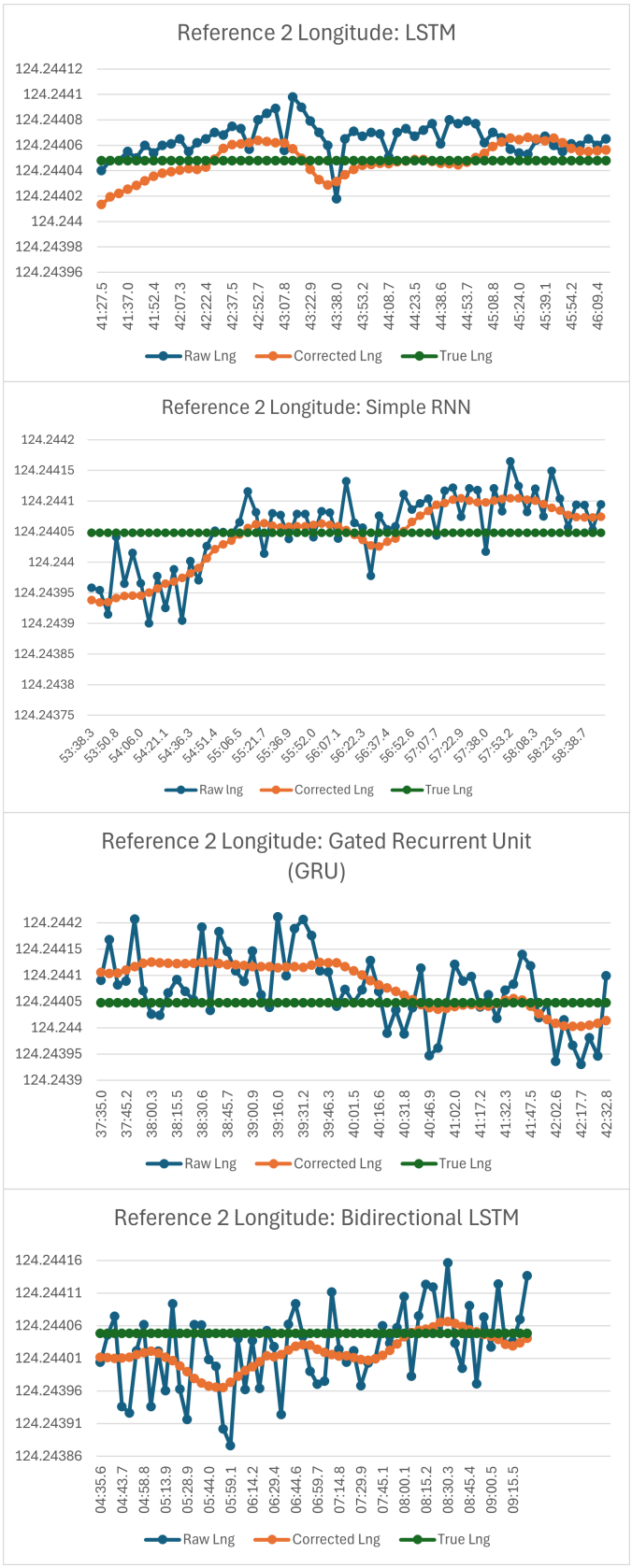

Fig. 17: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 2's true (known), raw, and corrected longitude points.

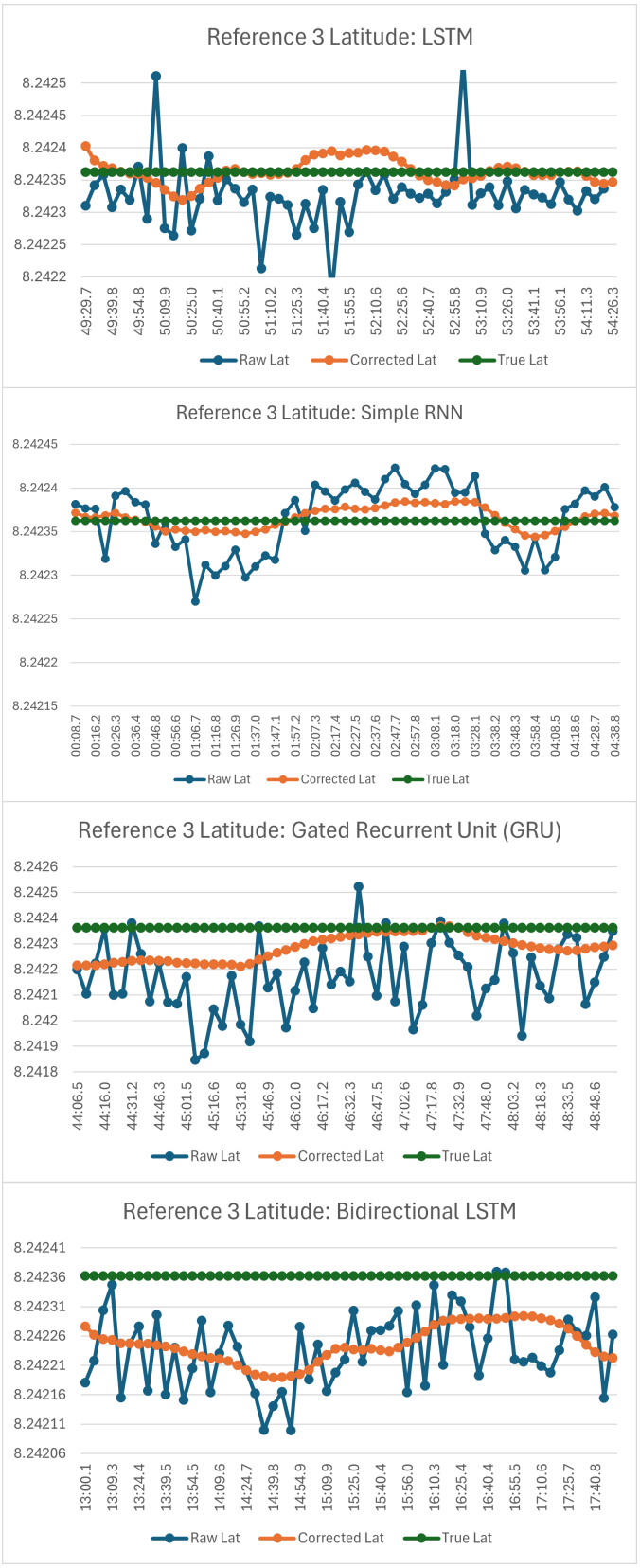

Fig. 18: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 3's true (known), raw, and corrected latitude points.

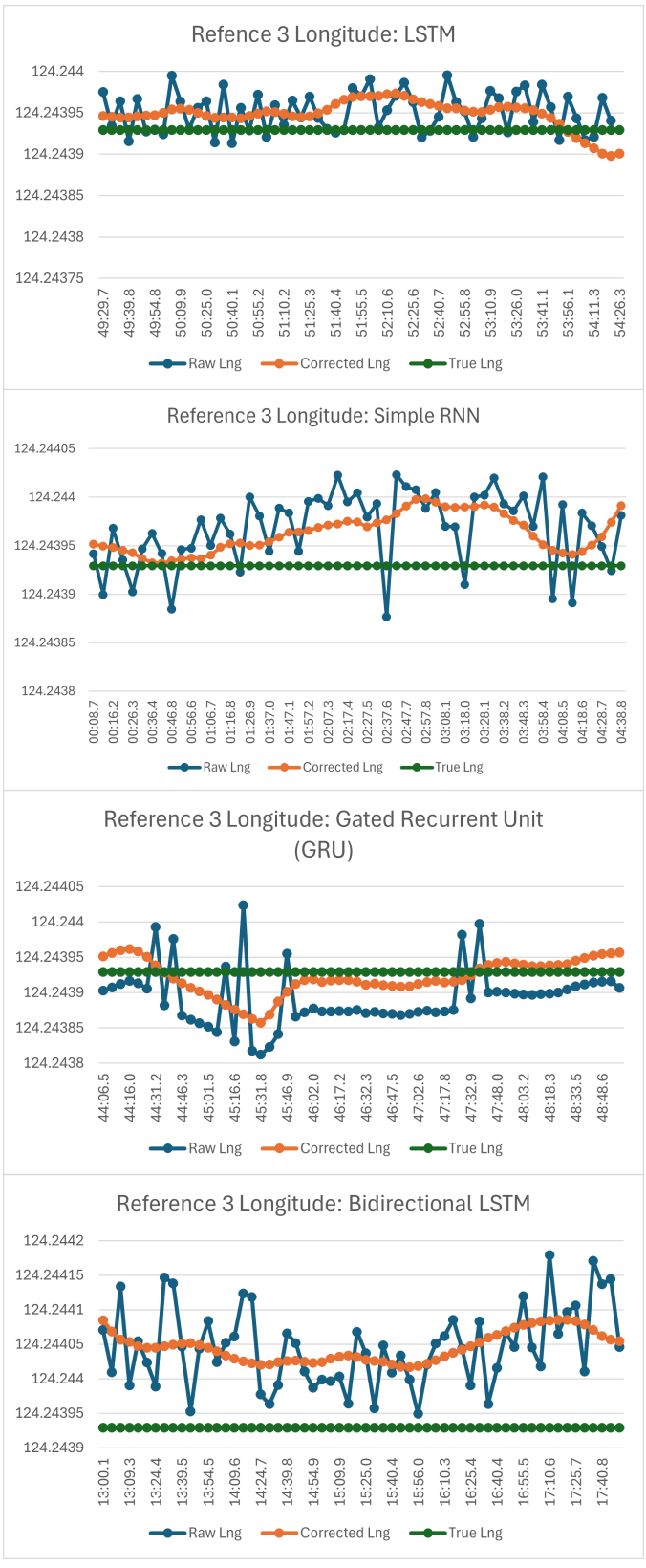

Fig. 19: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 3's true (known), raw, and corrected longitude points.

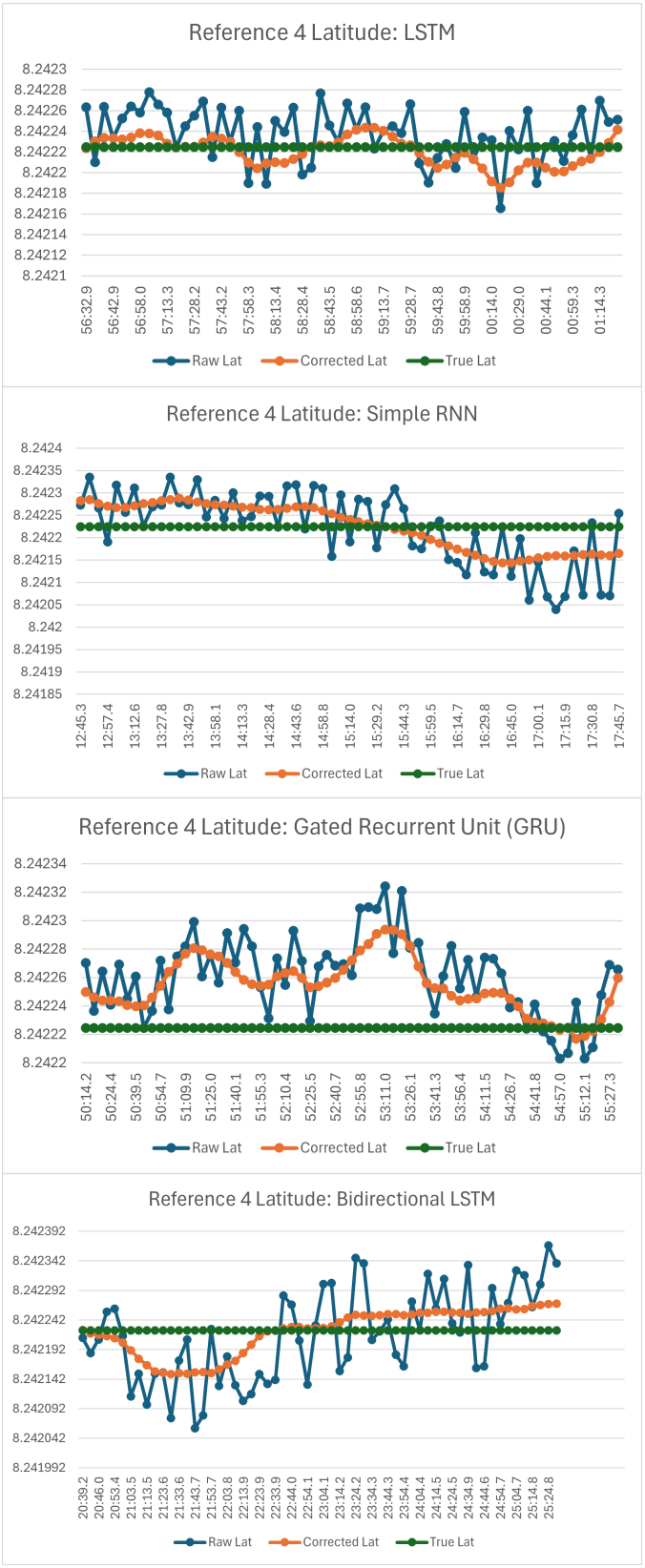

Fig. 20: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 4's true (known), raw, and corrected latitude points.

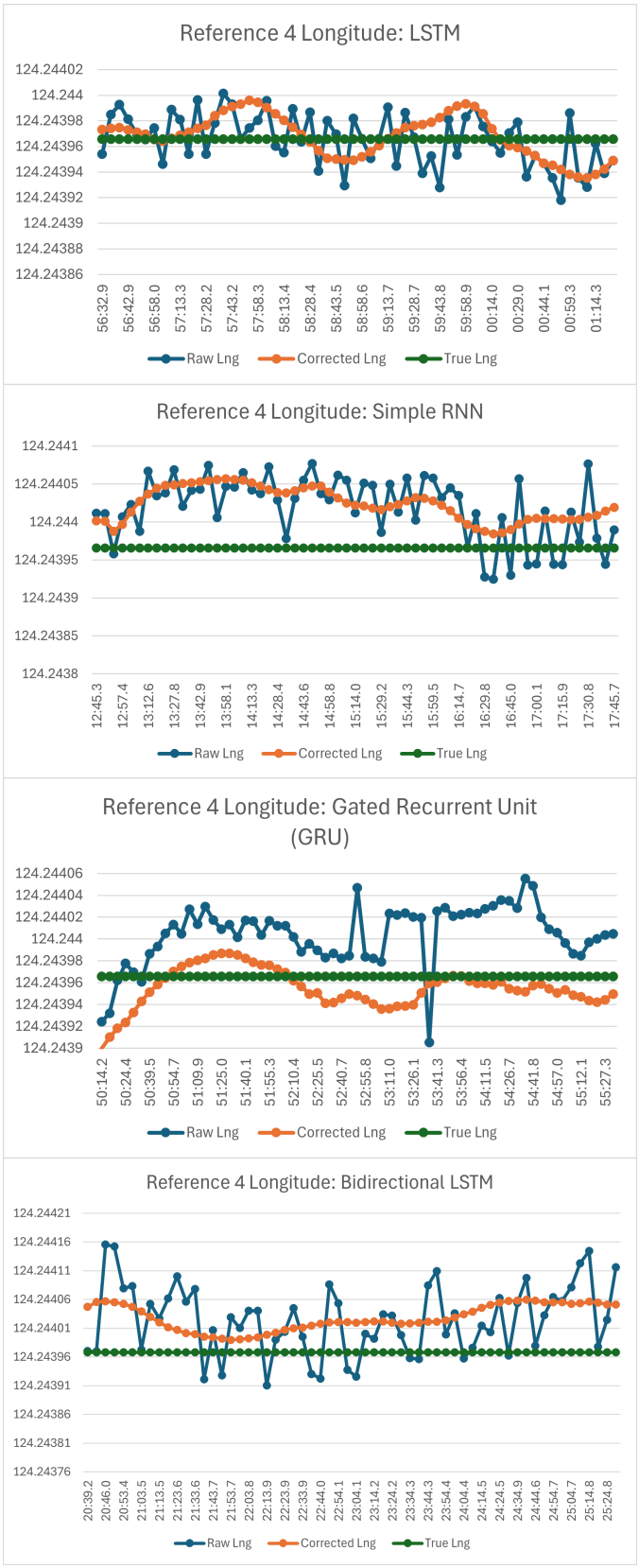

Fig. 21: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 4's true (known), raw, and corrected longitude points.

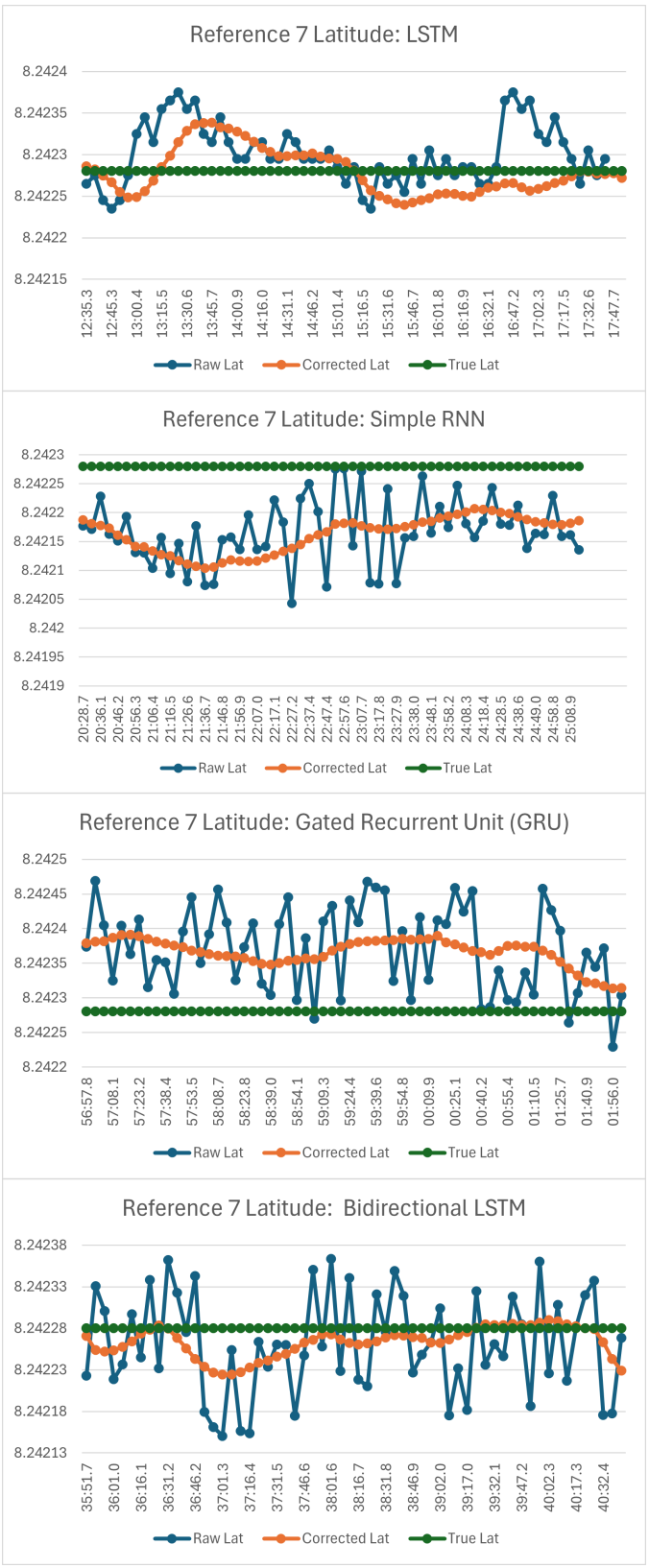

Fig. 22: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 7's true (known), raw, and corrected latitude points.

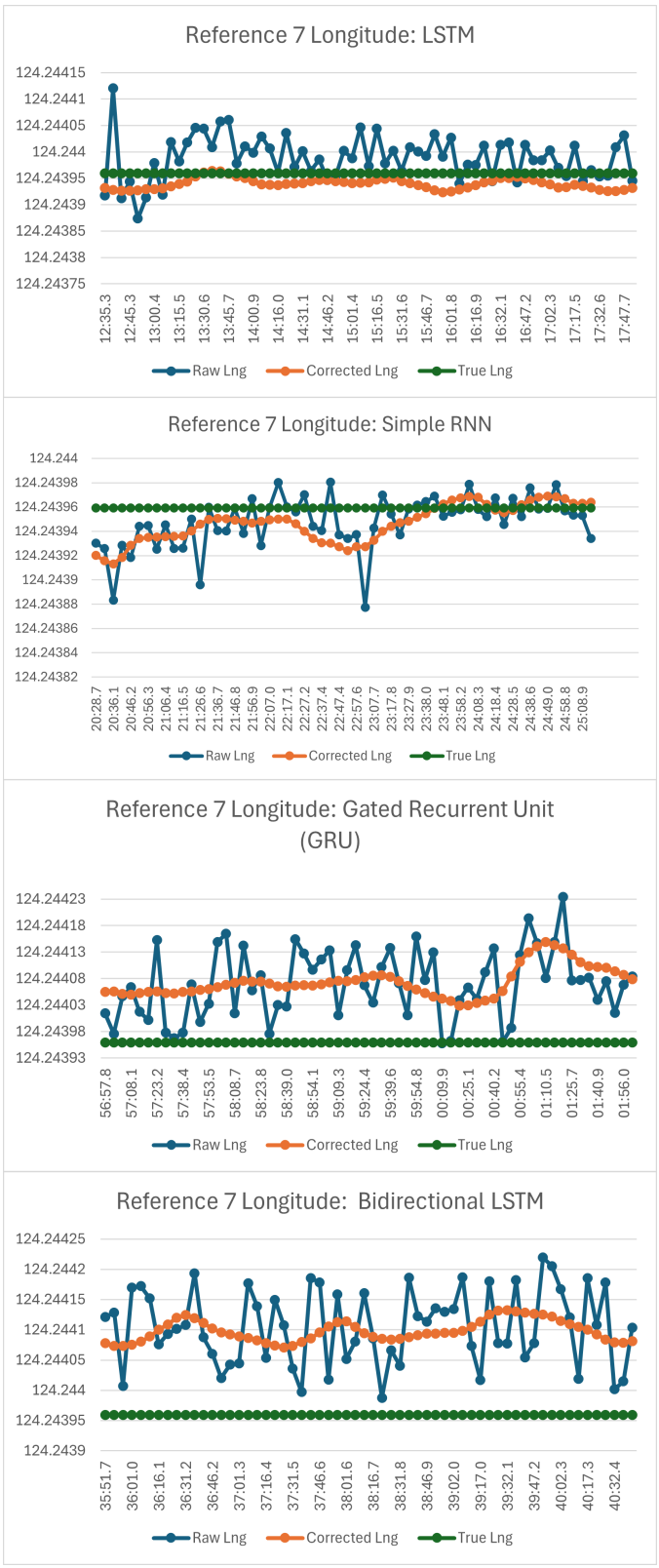

Fig. 23: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 7's true (known), raw, and corrected longitude points.

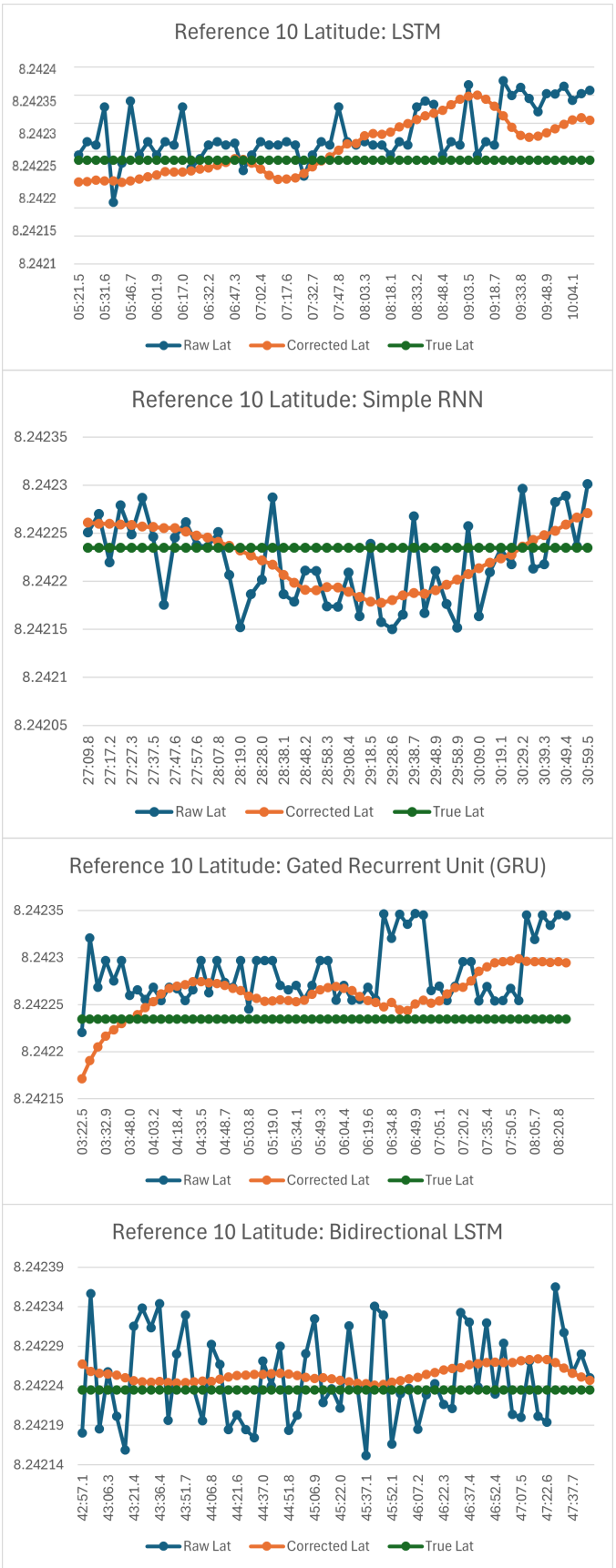

Fig. 24: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 10's true (known), raw, and corrected latitude points.

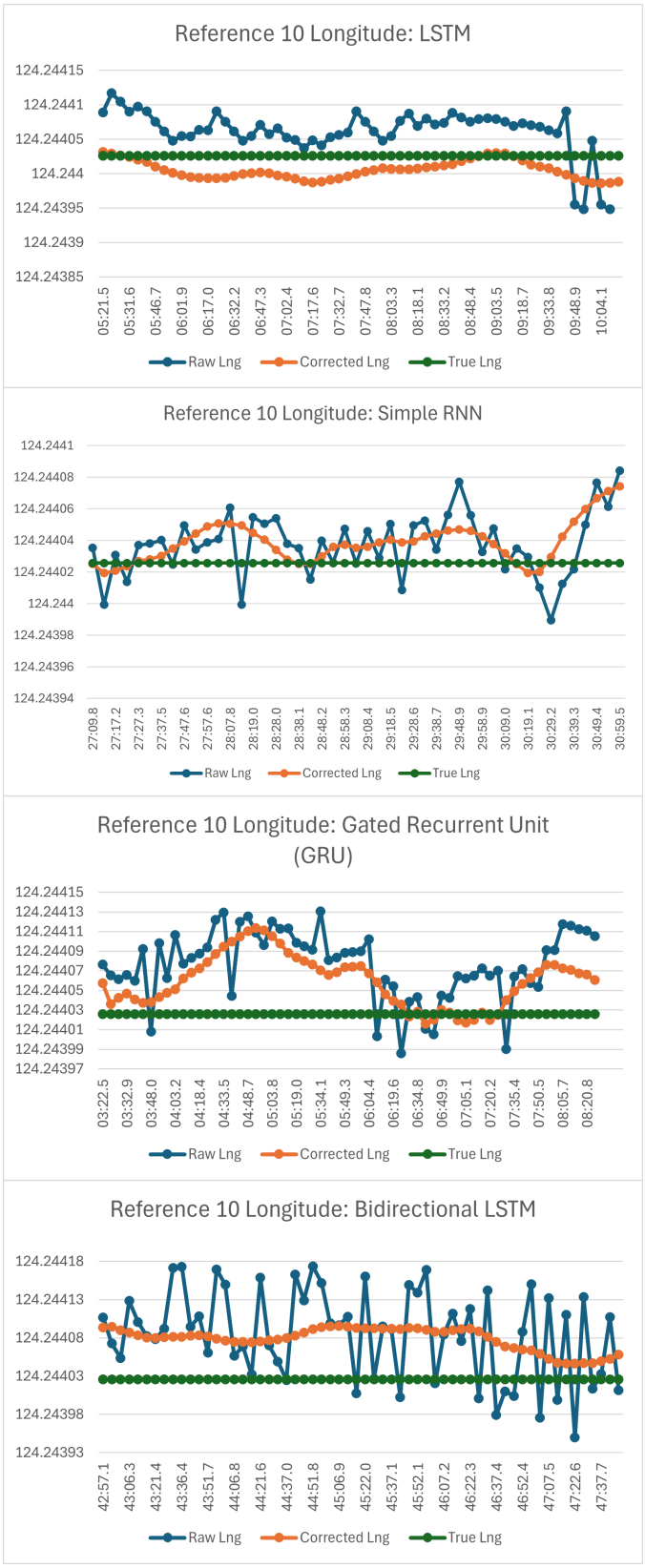

Fig. 25: Performance comparison of the LSTM, simple RNN, GRU, and bidirectional LSTM models between Reference 10's true (known), raw, and corrected longitude points.

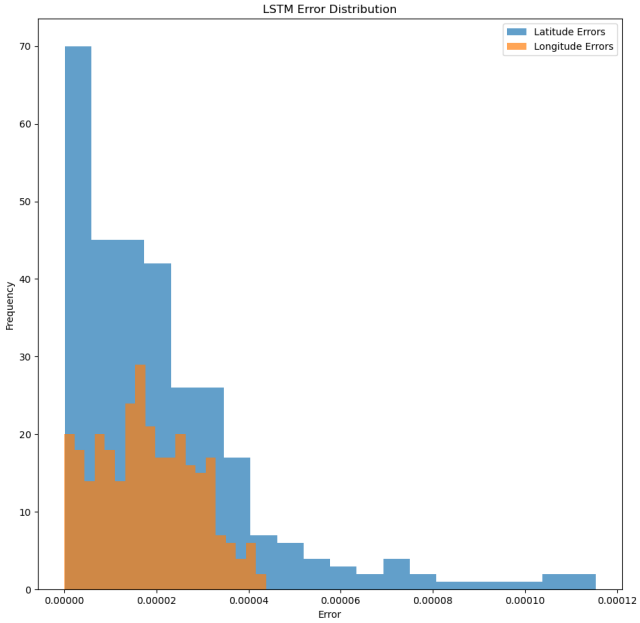

Fig. 26: Error distribution of the LSTM model.

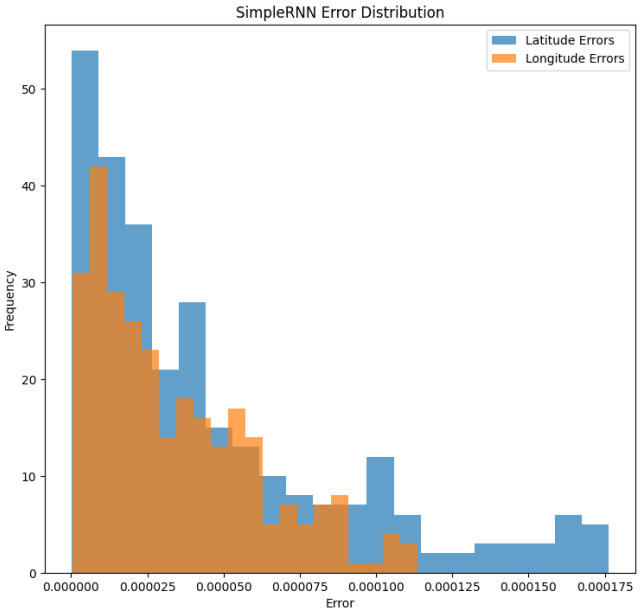

Fig. 27: Error distribution of the simple RNN model.

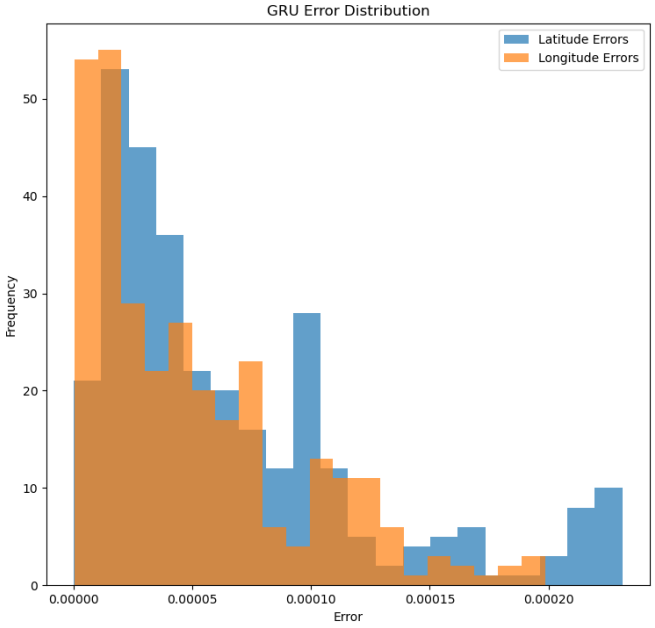

Fig. 28: Error distribution of the GRU model.

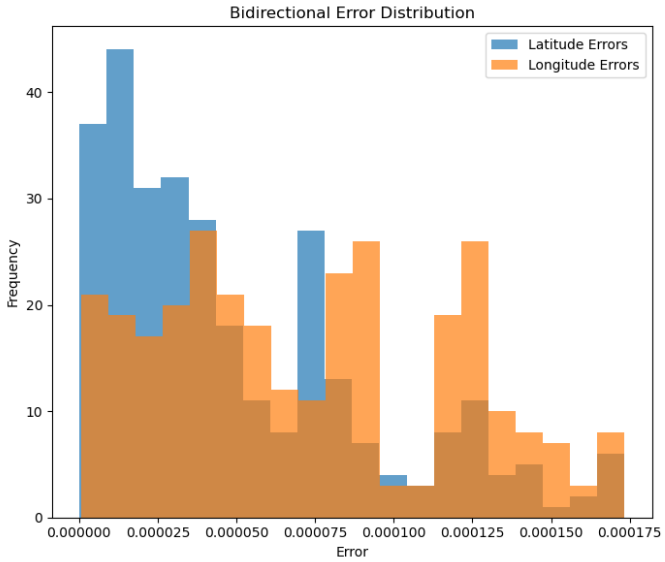

Fig. 29: Error distribution of the bidirectional LSTM model.