Project

Doweon Kim

Nov.30 2025

Abstract

This project explores an effective approach to forecast a vehicle upcoming location, on a highway. I utilized the NGSIM US-101 dataset, which includes vehicle trajectory data gathered from actual traffic. The aim of this project was to develop a short-term prediction model with accuracy. Initially I trained an LSTM model using time-series data. Subsequently I trained an XGBoost model designed to learn from the LSTM's prediction errors and attempt to adjust them. During the experiments, I noticed that correcting both axes was not helpful. The X-direction stays stable on highways, so the LSTM already performs well. But the Y-direction changes quickly, so the boosting model helped reduce drift. Overall, the hybrid model improved the total MSE by around 11.5% compared to using only LSTM. The results show that combining two different models can make trajectory predictions more stable.

1. Introduction and Background

Forecasting vehicle trajectories plays a role in numerous contemporary transportation applications, such as autonomous vehicles, lane-change support and traffic modeling. Anticipating a car's position is challenging since drivers frequently accelerate, decelerate, switch lanes and respond to nearby traffic. Due to these maneuvers machine learning techniques are commonly employed to capture motion patterns from actual driving datasets.

LSTM (Long Short-Term Memory) networks are commonly used for time-series problems because they can remember information from previous steps. However, even a strong LSTM model can produce drifting or biased predictions, especially in the longitudinal direction (Y-axis) on highways. To reduce this problem, this project adds another model, XGBoost, which learns the remaining errors from the LSTM and corrects them. By combining the two models, the system aims to produce more stable and accurate trajectory predictions.

1.1 The problem you tried to solve

The primary challenge of this project is forecasting the Local_X and Local_Y coordinates of a moving vehicle. While an LSTM model can capture the short-term motion it tends to generate drifting inaccuracies, particularly, along the Y-axis where the vehicle's movement is rapid and highly variable. My objective was to enhance prediction precision by developing an approach in which the LSTM provides the initial prediction and a boosting model adjusts the residual errors. During testing, I found that the correction was helpful for the Y-direction but not for the X-direction, so the final model only applies boosting to Local_Y.

1.2 Results from the literature

Previous studies show that LSTM and GRU models are effective for trajectory prediction because they can learn time-dependent patterns. Some research also shows that boosting models such as XGBoost can be used to correct prediction errors in time-series tasks. These ideas motivated this project: the LSTM learns the main movement trend, and the boosting model reduces the leftover errors. Many papers also mention that the longitudinal direction on highways has a large value range, which makes prediction drift common. This supports the idea that using a second model for error correction can be helpful.

1.3 What tools and programs are already available for the problem, or for closely related ones?

For this project, I used some common tools that many people use for time-series problems. These tools helped me because I don't need to build everything by myself. TensorFlow and Keras are helpful for making LSTM models, and they are not too hard to use. They also let me train the model faster, especially when GPU is available.

For correcting the LSTM errors, I used XGBoost. This tool is good for structured data and can learn the leftover errors well. It also trains fast and does not need too much setup. I also used normal Python libraries like NumPy and Pandas for handling the data, and Scikit-learn for scaling and splitting the dataset.

I worked in Google Colab because it is easy to run code there and I can use the GPU for free. All these tools made the project easier to try many ideas and build the models step by step.

2. Overview of the architecture

In this project, I built a simple pipeline that goes from raw traffic data to a final prediction model. The architecture has a few main parts. First the data need to be cleaned because the original NGSIM data has noise and some missing or strange values. After cleaning the data is processed again to make it ready for training. Then I train the LSTM model , and later I train another model(XGBoost) to fix the errors from the LSTM. In the end, both models work together to give the finall prediction.

Step 1 : Preprocessing Module - Clean the raw NGSIM data,

Step 2 : Processing Module - Split the data, Scale it, and make sequences.

Step 3 : LSTM prediction Module - Predict the next Position.

Step 4 : Residual correction Module - use XGBoost to correct errors (Only for Y)

Step 5 : Hybrid output Module - Combine both results to get the final prediction.

2.1 Finished work: Running modules

All the main modules in my project are finished and working. They are already tested with the dataset, and each part produces the correct output for the next step. Here is a summary of the running modules:

- Data Preprocessing Module

This part loads the raw NGSIM US-101 data and removes many problems in the dataset.
It deletes tracking errors, removes jump frames, filters unrealistic speeds, and converts units from feet to meters.
At the end, it saves a clean dataset that is ready for processing.
This module runs fully without errors.

- Data Processing Module

This module takes the cleaned dataset and prepares it for the machine learning models.
It splits the data by Vehicle_ID into train, validation, and test sets so there is no data leakage.
It also scales the features using MinMaxScaler and creates time-series sequences for LSTM.
All of these steps are completed and saved into separate files.

- LSTM Model Module

This module builds and trains the LSTM model.
It also tests different sequence lengths and uses early stopping to avoid overfitting.
The model trains without issues and can make predictions on the test data.
The output of this module is the first prediction of the next positions.

- Residual Learning (XGBoost) Module

This module learns the remaining errors from the LSTM prediction.
It takes the LSTM outputs and trains XGBoost models to fix the errors.
After testing, I found that XGBoost helps only for the Y-direction, so the final version uses correction only on Y.
This module also works correctly and improves the prediction accuracy.

- Hybrid Prediction Module

This is the final module that combines both models.
It keeps the LSTM prediction for X, and for Y it adds the residual correction from XGBoost.
This produces the final trajectory prediction.
The module runs well and shows better performance than using LSTM alone.

2.2 Work in progress: Modules I tried to plan but did not finish

During the project several concepts came to mind. I was unable to complete them due to time constraints. I intended to experiment with these concepts. I only briefly outlined them and didn't finish the implementation. One concept involved testing sequence lengths for the LSTM model. In this project I only utilized sequence lengths of 5, 8 and 10. I aimed to explore values, such as 20 or even 30 to determine if the model could capture long-term patterns more effectively. But longer sequences need more memory and training time, so I was not able to run these experiments. I also tried to think about adding more features for the sequence input, but I did not finish making a full plan. These ideas were only in the early stage, and I could not continue them during this project. These modules are not completed, but they were small ideas that I started thinking about while working on the model.

2.3 Future work: Modules I want to try in the future but not designed yet

There are some ideas that I did not design or implement in this project, but I want to try them in the future when I have more time. These ideas are only thoughts for now, and I did not make a real plan or architecture for them.

One idea is to try GRU instead of LSTM. GRU is a simpler version of LSTM and sometimes works faster. I want to see if GRU can predict the vehicle movement better or if it has less drift. I did not design the model yet, but it is something I want to explore later.

Another idea is to try a Transformer-based model. Many new papers show that Transformers can learn time patterns well, even for traffic or motion data. I think it would be interesting to test if a Transformer can handle the trajectory data better than LSTM. But I did not start the design, because Transformer models need more steps and more tuning.

I also want to try multi-step prediction, where the model predicts more future points instead of only one. This could show longer movement behavior, but it needs a different model setup. I did not begin the design for that part either.

These ideas are not finished modules, and I could not start them during this project. But they are possible future directions that I want to try when I have more time and experience

3 Data Collection

The dataset I used in this project is the NGSIM US-101 dataset. It was collected by the U.S. Department of Transportation, and it contains real vehicle trajectories from a highway in Los Angeles. The dataset includes many cars driving on the road, and each car has information recorded at high frequency. This data is used a lot in research because it is detailed and shows real traffic behavior.

Each vehicle in the dataset has several types of information. Some of the important ones are:

- Vehicle_ID to identify each car
- Frame_ID for the time step
- Local_X and Local_Y for the position on the road
- v_Vel for speed
- v_Acc for acceleration
- Lane_ID
- the IDs of the preceding and following cars.

These features help understand how a car moves and reacts to other vehicles.

In this project, I used only the US-101 section of the dataset because it is smaller and easier to work with, and it also has cleaner lane structure. I cleaned the data by removing strange values and downsampled it from 10Hz to 1Hz to make the sequence more stable. I also created extra features like speed, acceleration, and heading to help the model learn better. This processed dataset was then used to train the LSTM and the hybrid model.

4 Your methods and implementation

This section explains how I built the system step by step. The project has several parts, starting from cleaning the raw NGSIM data and ending with the hybrid model that combines LSTM and XGBoost.

4.1 Data Preprocessing

The first step was to clean the raw US-101 dataset. The original data has many problems, such as missing values, sudden jumps, and vehicles that do not follow normal frame order. I removed these issues so the model can learn smoother motion patterns.

I used only the US-101 section because it is smaller and has clearer lane structure. I removed vehicles with wrong tracking (for example, their Frame_ID does not increase by 1), and I deleted unrealistic points like very high speeds or strange acceleration values. Jump frames, where the vehicle suddenly teleports, were also removed.

The dataset originally had 10Hz sampling, so I downsampled it to 1Hz to make the data more stable and reduce noise. I also created new features such as speed, acceleration, and heading using the position changes. After these steps, I kept only vehicles with at least 30 frames so the model has enough sequence to learn from.

4.2 Data Processing

I changed the units from feet to meters. Then I selected the features I wanted to use: Local_X, Local_Y, v_Vel, v_Acc, speed, and acceleration. The target was the next Local_X and Local_Y. To avoid data leakage, I split the dataset by Vehicle_ID into train, validation, and test sets. This makes sure the same vehicle does not appear in two different sets. I used MinMaxScaler and fitted it only on the training set. After scaling, I made time-series sequences using a sliding-window method. Each sequence uses several past steps (seq_len) to predict the next position.

4.3 LSTM Model

The next step was to train the LSTM model. LSTM is good for time-series data because it remembers the previous steps. I tested different sequence lengths: 5, 8, and 10. For each seq_len, I trained an LSTM with 64 units, a Dense layer of 32 units, and a final output layer with 2 values (the next X and Y).

I used MSE as the loss function and Adam as the optimizer. I also used early stopping so the model stops training when the validation loss does not improve. After testing the three seq_len values, I picked the one with the best validation performance and trained the final LSTM model with that setting.

4.4 XGBoost Residual Model

Even though the LSTM worked well, it still had some drifting problems, especially in the Y-direction. To reduce this error, I added another model: XGBoost. This model does not predict the position directly. Instead, it predicts the residual, which is the difference between the true value and the LSTM prediction.

The input for XGBoost was the last step’s features plus the LSTM prediction for that step. I trained two XGBoost models at first: one for X-residual and one for Y-residual. But after testing, I found that correcting the X-direction did not help, because X stays very stable on highways. The Y-direction changed fast, so XGBoost helped only for Y. Because of this, the final model uses the XGBoost correction only for the Y axis.

4.5 Hybrid Model

The final model is a combination of the LSTM prediction and the residual correction. The idea is simple:

- For X, I keep the original LSTM prediction because XGBoost did not improve it.
- For Y, I add the residual from XGBoost to the LSTM prediction to fix the drifting problem.

This gives the final hybrid prediction:

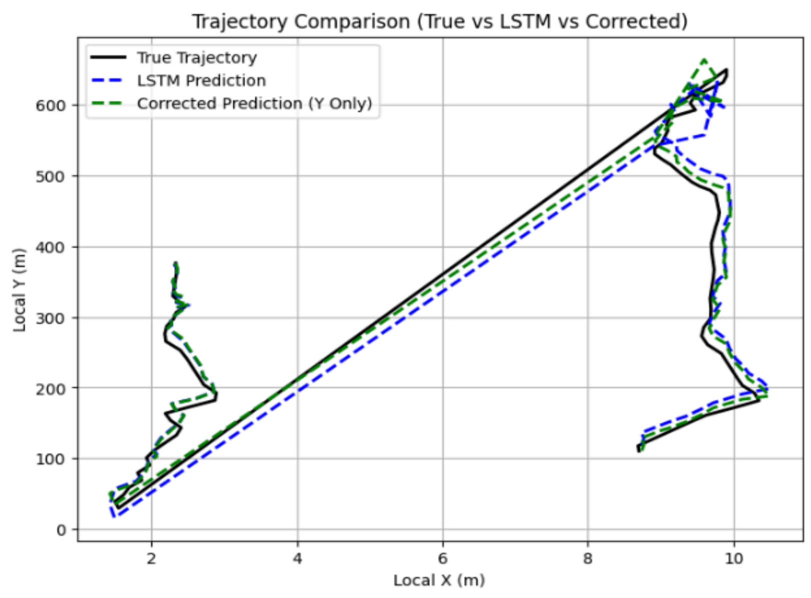
- X_final = X_LSTM
- Y_final = Y_LSTM + correction

This hybrid method made the prediction more stable, especially in the Y-direction where most of the drift happened. It improved the overall MSE and reduced the error compared to using only the LSTM model.

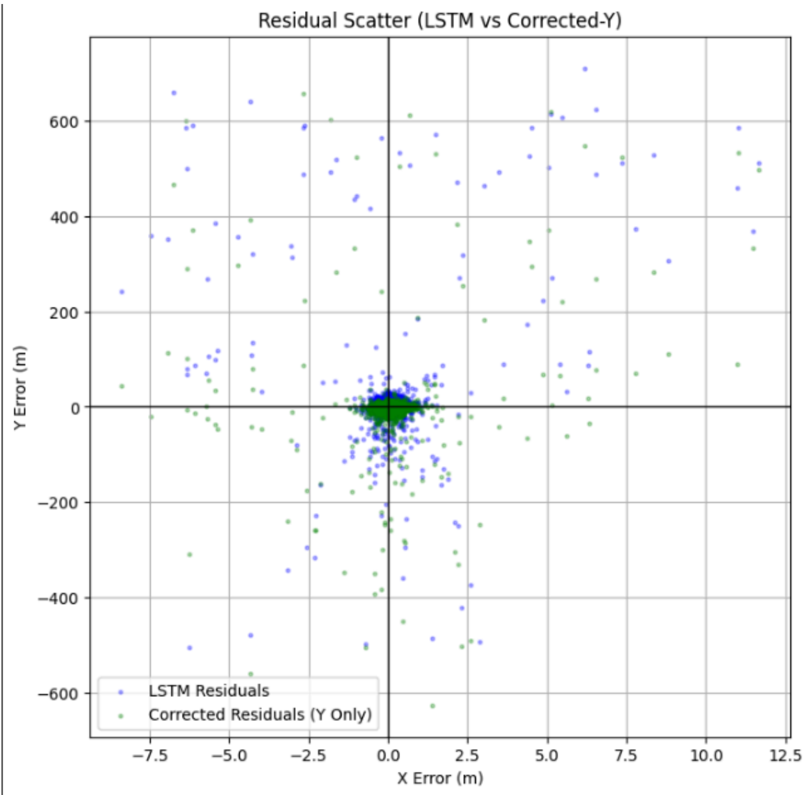
5 Results and Evaluation

This section shows the final results of the experiments and compares the LSTM model with the hybrid model. I first checked which sequence length works best, and then I measured the error on the test set. I also used some plots to see how the predictions and residuals look in a more visual way.

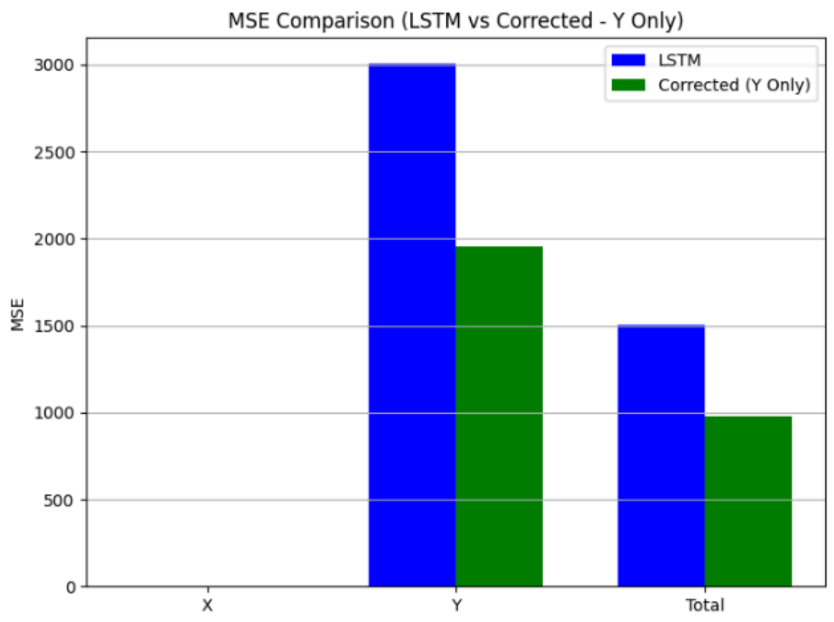
[Figure.1]



[Figure.2]



[Figure.3]



5.1 Validation Results for Different Sequence Lengths

I trained LSTM models with three different sequence lengths: 5, 8, and 10.
The validation losses were:

- seq_len = 5 → 0.003871
- seq_len = 8 → 0.004734
- seq_len = 10 → 0.003510

The sequence length 10 gives the lowest validation loss, so I chose seq_len = 10 for the final LSTM model. This means the model works better when it can see 10 past steps of motion history.

5.2 LSTM Performance on the Test Set

With seq_len = 10, I evaluated the LSTM model on the test set.
The MSE values were:

- LSTM MSE (X) = 0.491678
- LSTM MSE (Y) = 3004.093151
- LSTM Total MSE (2D) = 1502.292414

The 2D RMSE of the LSTM prediction was:

- LSTM 2D RMSE = 38.76

These numbers show a big difference between X and Y.
The X error is very small, but the Y error is very large. This means the LSTM can predict the lateral position (Local X) quite well, but it has strong drift in the longitudinal direction (Local Y), where the values grow quickly along the highway.

In **Figure 1** (Trajectory Comparison), the blue dashed line (LSTM prediction) slowly drifts away from the black line (true trajectory), especially in the Y direction. This matches the large MSE in Y.

5.3 Hybrid Model Performance (LSTM + XGBoost)

I added the XGBoost residual model to correct the LSTM errors.
The final hybrid model keeps X from the LSTM and only corrects Y:

- X final = X LSTM
- Y final = Y LSTM + residual Y from XGBoost

The test results for the hybrid model were:

- Corrected MSE (X) = 0.491678 (same as LSTM, no change)
- Corrected MSE (Y) = 1953.511029
- Corrected Total MSE (2D) = 977.001353

The 2D RMSE decreased to:

- Corrected 2D RMSE = 31.26

So the hybrid model reduces the total error compared to the pure LSTM.

In **Figure 1**, the green dashed line (corrected prediction) follows the black true trajectory more closely than the blue LSTM line, especially in the long straight part and the turning region. This shows that the residual correction helps the Y movement become less drifted.

5.4 Improvement Summary

The improvement can be summarized as:

- X Improvement: 0% (no change, X uses LSTM only)
- Y Improvement: 34.97%
- Total Improvement: 34.97%

So the hybrid model reduces the Y-direction MSE by about one-third and also lowers the total 2D error by the same rate. This confirms that the error-correction model is effective mainly for the longitudinal direction.

5.5 Residual and Error Distribution

To understand the behavior better, I looked at the residual distribution.

- In **Figure 2 (Residual Scatter Plot)**, each point is an error in X and Y.
    - The blue points are the LSTM residuals.
    - The green points are the corrected residuals.
      After correction, many green points are closer to the origin, especially along the Y axis. This means the hybrid model reduces large Y errors and makes the error cloud more concentrated near (0, 0).
- In **Figure 3 (MSE Bar Chart)**, the bars compare LSTM and correct MSE for X, Y, and Total.
    - The X bar is the same for both models,
    - but the Y and Total bars are much lower for the corrected model.
      This gives a clear visual confirmation of the numerical improvement.

6 Achievements and Observations

During this project, I learned many things about time-series prediction and also about how different models behave with real trajectory data. One interesting observation was about the hybrid model. At first, I expected that adding XGBoost corrections to both X and Y would make the whole prediction better. But the result was different than I imagined. The X direction actually became worse when I tried to correct it, and the LSTM alone was already the best choice for X.

This was surprising to me, and it helped me understand that not every correction method works for every feature.

Another thing I learned is how LSTM and XGBoost work in time-series data. Before doing this project, I did not have much experience with these models. By training them myself and checking the results, I understood more about how LSTM remembers past steps and how XGBoost learns errors that LSTM cannot capture. I also saw that LSTM can drift in the Y direction because the values grow fast on highways.

I also observed that sequence length is very important. As I expected, when the model sees more past steps, the performance becomes better. In my experiment, seq_len = 10 worked the best. This showed me that giving enough history to the model helps it understand the movement pattern more clearly.

Overall, these observations helped me understand not only the final model but also the behavior of different time-series methods in real data.

7 Discussion and Conclusions

Discussion

In this project I focused on forecasting the location of a vehicle utilizing authentic highway data from the NGSIM US-101 dataset. Initially the data contained a lot of noise so much of the effort went into cleaning. Structuring the trajectories. I eliminated tracking inaccuracies, frame jumps, improbable motions and also reduced the data sampling rate to 1Hz. Following preprocessing I generated features such as velocity, acceleration and direction which improved the models grasp of the movement patterns.

The primary model employed was an LSTM as LSTMs are capable of capturing time-series patterns from steps. I experimented with sequence lengths and discovered that utilizing 10 steps yielded optimal results. Although the LSTM predicted the X direction accurately it exhibited drift in the Y direction due to growth of highway movement along that axis. This drift led to errors in the Y dimension.

To address this issue I incorporated a model, XGBoost, to adjust the residual errors left by the LSTM. Initially I attempted to correct both X and Y. Discovered that fixing X actually degraded the results. Since the LSTM was already precise for X no adjustment was necessary in that dimension. On the other hand, correcting the Y residual significantly enhanced the outcomes. The combined model lowered the Y MSE by 35% and the overall 2D error was reduced by a similar margin. This demonstrated that the two models can work together: LSTM captures the pattern while XGBoost corrects the remaining errors.

Through these experiments, I learned how different models behave with time-series data. LSTM performs well with smooth motion but struggles with large range values. XGBoost is good at modeling small corrections but does not always help every feature. I also saw the importance of sequence length and how giving more history can improve prediction quality.

Conclusions

This project successfully built a hybrid trajectory prediction model using LSTM and XGBoost. The preprocessing pipeline produced clean and stable data, and the sequence-based LSTM model learned the general movement of the vehicles. By adding the XGBoost residual model only to the Y direction, the final hybrid system achieved much better accuracy than the LSTM alone. The results show that combining different types of models can reduce drift and produce more stable predictions in real traffic data.

The project also helped me understand more about time-series modeling, residual learning, and highway trajectory characteristics. The improvements were significant, especially in the Y direction, which is known to be difficult in highway prediction.

For future work, I want to try other models such as GRU or Transformer-based models. Transformers are becoming popular in many sequence problems, and they may capture long-range patterns better than LSTM. I also want to explore multi-step prediction and include information from nearby vehicles. These extensions could make the model even stronger and more realistic for real-world applications.

8 References

[REF1]

Deo, N., & Trivedi, M. M. (2018, May 15). Convolutional social pooling for vehicle trajectory prediction. arXiv.org. https://arxiv.org/abs/1805.06771

The paper predicts vehicle trajectories using LSTM and social pooling on NGSIM data. Used in **Section 1.2** when discussing past research on trajectory prediction.

[REF2]

Jo, E., Sunwoo, M., & Lee, M. (2021, August 9). *Vehicle trajectory prediction using hierarchical graph neural network for considering interaction among multimodal maneuvers*. MDPI. https://www.mdpi.com/1424-8220/21/16/5354

The paper predicts vehicle paths by modeling interactions between nearby cars. Used in **Section 5** when mentioning advanced future models that consider vehicle interaction.

[REF3]

Amin, F., Gharami, K., & Sen, B. (2024, April 8). *Trajectoformer: Transformer-based trajectory prediction of autonomous vehicles with spatio-temporal neighborhood considerations - International Journal of Computational Intelligence Systems*. SpringerLink. https://link.springer.com/article/10.1007/s44196-024-00410-1

The paper applies Transformer models to vehicle trajectory forecasting and shows strong performance by capturing long-range temporal patterns. Used in **Conclusions** when mentioning future plans to try Transformer-based models.

Appendices

A. Individual stories

Since I worked on this project. I did all parts of the work by myself. I started by studying the NGSIM dataset and cleaning the data, because the raw data had many errors like jump frames and unrealistic movements. I also made new features such as speed and acceleration to help the model understand the movement better.

After preprocessing, I trained the LSTM model and tested different sequence lengths. Then I added the XGBoost residual model to reduce the drift in the Y direction. I learned that the correction helps Y but not X, which was an interesting result for me. I also made the plots, evaluated the models, and wrote the whole report. Through this project, I learned more about time-series data, LSTM behavior, and how to combine models to improve predictions.