INCREASING MARITIME DOMAIN AWARENESS USING SPATIO-TEMPORAL SEQUENTIAL DATA

Kyle Nathan Yahya (s252786), Christos Psallidas (s253542)
Dimitrios Dimitriou (s253560), Nikolaos Fergadakis (s253129)

1. ABSTRACT

Reliable vessel trajectory prediction is essential for maritime situational awareness. However, forecasting future movements using Automatic Identification System (AIS) due to irregular sampling, noise, and error prone signals, making short term trajectory forecasting a challenging task. This project models and compares four different deep learning architectures for short term trajectory prediction of cargo and tanker vessels in Danish waters. The models are RNN, LSTM, and GRU combined with a common Encoder-Decoder framework and Transformer. Results show that Encoder-Decoder GRU achieves the highest accuracy with a mean distance error prediction of 1.51 kilometers at 50 minutes into the future, statistically outperforming the baseline and the other models.

2. INTRODUCTION

Maritime transportation supports most global trade, making safe and efficient vessel movement essential. Many regions experience high traffic density, which increases the risk of congestion and accidents, highlighting the need for reliable vessel movement prediction [1]. Accurate short-term trajectory forecasting enables collision avoidance, traffic monitoring, and efficient port operations. AIS which transmits vessel kinematic data provides rich information for data-driven prediction models. However, predicting future trajectories remains challenging due to noisy data, irregular sampling rates, and diverse vessel behaviour [2]. Furthermore, vessel movements are highly likely to be non-linear due to complex environmental factors [3].

Recent advancements in deep learning have demonstrated superior performance compared to statistical or machine learning approaches [2]. Particularly RNN and its variants (LSTM, GRU) have shown to be effective in learning temporal dependencies for trajectory prediction [4]. More recently, attention based neural network, such as Transformer, is also gaining popularity because of the ability in modeling long term dependencies in parallel [5].

This project focuses on short-term cargo and tanker vessel trajectory prediction using AIS data surrounding Danish waters. We develop and compare four different deep learning models, specifically RNN, LSTM, GRU, and Transformer for this task. We standardized the experimental framework, where all models are trained, tuned, and evaluated on the same dataset and pre-processing pipeline.

3. RELATED WORK

Maritime trajectory prediction and ETA forecasting have been extensively studied in recent years. A comprehensive review by Jiang et al. [1] highlights that AIS data suffer from noisy data and irregular sampling. A recent study uses resampling and cubic spline interpolation to handle this problem while still creating dynamic movements. [6]

Another problem in deep learning trajectory prediction is the instability of training on absolute geographical coordinates. The models will have the tendency to memorize the map instead of learning patterns. To mitigate this, reviews by Zhang et al. [2] highlight that transforming the coordinates into differential features to let the model learn the movement instead of location.

Focusing on Danish waters, Forti et al. [7] implemented a Recurrent Sequence to Sequence architecture for multi step trajectory prediction. Their approach is shown to outperform traditional one to one regression methods. In addition, their work validates the use of Huber loss to handle noise and outlier better.

4. METHODOLOGY

This project is focused on supervised Sequence to Sequence regression problem. Given a historical sequence of data $X = [x_1, x_2, ..., x_N]$, the output is a predicted future sequence $Y = [y_1, y_2, ..., y_M]$. In this project, we use an observation windows of $N = 20$ steps to predict the future $M = 10$ steps, with 5 minute interval between steps. Four different deep learning architectures are used in this project. Three of the four are RNN, LSTM, and GRU combined with a common Encoder-Decoder framework to handle Sequence to Sequence task with different dimensions. The last is Transformer to compare the performance of recurrent based network and attention based network for trajectory prediction task.

4.1. Data collection and pre-Processing

The dataset used in this project consist of AIS messages by the Danish Maritime Authority [8]. Data on cargo and tanker vessels over 1.5 days period are collected for this project. This should be enough for the models to understand general pattern surrounding Danish waters as trips in this area usually last less than a day [9]. From this raw data, six features are utilized: Timestamp, MMSI (Maritime Mobile Service Identity), Longitude, Latitude, SOG (Speed Over Ground), COG (Course Over Ground).

First, duplicate data and MMSI that are not 9 digits or do not follow MID standards [10] are removed. Since the data set is about vessels around Danish waters, data outside the region bounded by [50, 60] latitude and [0, 20] longitude are also removed. Then, the data is resampled in a 5 minute interval. To fill the NAN values, cubic spline interpolation was performed and in cases where it is not possible linear interpolation was applied. Since most vessels travel between 12 and 25 knots on average, SOG of over 40 knots is unlikely and removed. Each vessel's trajectory is then segmented into sub-trajectories whenever there is 3 hours gap between consecutive points. Short sub-trajectories with less than 35 data points are dropped to ensure enough sequence for model training. Denote a point at time t as $x_t = [Lat, Lon, SOG, COG]$, where $[Lat, Lon, SOG, COG]$ represents longitude, latitude, SOG, and COG at time t. In our experiments, we found out that the models performed significantly worse when predicting latitude and longitude. One possible reason is that it focused on memorizing the map instead of understanding the pattern of how vessels move. Therefore, the input and output vectors we use are $\tilde{x}_t$ and $\hat{y}_t$ shown in equation 1.

Lat, Lon represent the absolute location as a location context. SOG is the speed over ground in m/s. For the time-differential features $\Delta Lat, \Delta Lon$, and ΔCOG can be obtained by the equations below. Particularly for ΔCOG does not just represent the difference in COG, it needs to calculate the shortest angular difference between two points.

To prepare data for supervised learning, we create input and output sequences X_t and Y_t, using sliding window approach. As shown in Equation 1 X_t represents 20 consecutive inputs vectors while Y_t represents the subsequent 10 consecutive output vectors in one sub-trajectories.

$$\begin{aligned} \tilde{x}_t &= [Lat_t, Lon_t, \Delta Lat_t, \Delta Lon_t, SOG_t, \Delta COG_t] \\ \tilde{y}_t &= [\Delta Lat_t, \Delta Lon_t, SOG_t, \Delta COG_t] \\ X_t &= [\tilde{x}_t, \tilde{x}_{t+1}, \ldots, \tilde{x}_{t+19}] \\ Y_t &= [\tilde{y}_{t+20}, \tilde{y}_{t+21}, \ldots, \tilde{y}_{t+29}] \end{aligned} \tag{1}$$

Once we have the Sequence to Sequence pairs, the data is split into train, validation, test subsets with 64/16/20 split partitioned on the vessel MMSI. This is to ensure that the model will not train on parts of the vessel trajectories in the test set. Each feature will be standardized before being fed to the models for training.

4.2. Models

4.2.1. Baseline

For our baseline, we refer to the study by Sørensen et al.[6], which evaluated BLSTM-MDN model on cargo vessel AIS data in the region west of Norway. We will use their reported mean distance error at 25 and 50 minutes to benchmark our models.

4.2.2. Recurrent architectures with encoder-decoder

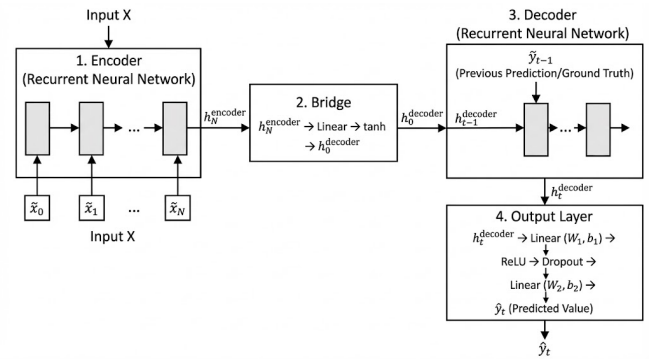

Fig. 1. Visualization of the Encoder-Decoder framework that will use either RNN, LSTM, or GRU units.

The Encoder-Decoder framework serves as a template where the core processing units in both Encoder and Decoder are one of the three recurrent neural networks (basic RNN, LSTM, GRU) which will be further discussed in the following subsections. This way we can fairly compare these recurrent architectures. The framework consists of four main components:

1. **Encoder:** A recurrent neural network that processes input sequence $X = [\tilde{x}, \tilde{x}_1, ..., \tilde{x}_N]$ step by step. The result is denoted as hidden state $h_N^{encoder} = \Phi(x_N, h_{t-1}^{encoder})$, where Φ represent the chosen recurrent architectures.
2. **Bridge:** A linear layer and tanh activation function that transforms $h_N^{encoder}$ to hidden state dimension of the decoder, $h_N^{decoder}$. We use a larger dimension of 128 for encoder hidden state to capture more information, while only 64 for decoder. The result of bridge, $h_0^{decoder} = \tanh(W_{bridge} \cdot h_N^{encoder} + b_{bridge})$, is the first hidden state of decoder.
3. **Decoder:** A recurrent neural network that uses the same recurrent architecture Φ and generates future trajectory sequence auto regressively. At each prediction step t, the hidden state is computed by $h_t^{decoder} = \Phi(\tilde{y}_t, h_{t-1}^{decoder})$.
4. **Output Layer:** The final decoder hidden state $h_f^{decoder}$ is mapped back to the physical feature space. It passes through a linear layer, a ReLU activation function, dropout, and a final linear layer, producing $\hat{y}_t = W_2 \cdot \text{ReLU}(W_1 \cdot h_f^{decoder} + b_1) + b_2$.

4.2.3. Basic recurrent neural network (RNN)

For the first model, we implemented the Encoder-Decoder framework using 2 layers of basic RNN. RNN computes hidden state h_t and predicted output $\hat{y}_t$ using these equations.

$$\begin{aligned} h_t &= \tanh(W_x \cdot x_t + b_x + W_h \cdot h_{t-1} + b_h) \\ \hat{y}_t &= \tanh(W_y \cdot h_t + b_y) \end{aligned} \tag{2}$$

Repeated multiplication on W_h causes the vanishing/exploding gradient problem. This poses a problem in maintaining information long term.

4.2.4. Long-short term memory (LSTM)

We use a 2 layers LSTM inside the Encoder-Decoder framework as our second model. LSTM is a gated RNN with a cell state that is connected recurrently to each other. This cell state serves as a memory pathway that allows model to remember or forget information across time. LSTM computes hidden state h_t with a cell state that is regulated by forget gate (f_t), input gate (i_t), and output gate (o_t). A forward pass of LSTM is expressed by the following equations.

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ \tilde{c}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{3}$$

This architecture is able to remember past information better compared to basic RNN.

4.2.5. Gated recurrent unit (GRU)

We implement a 2 layers GRU inside the Encoder-Decoder framework for the third model. GRU is similar to LSTM but computationally more efficient by merging cell and hidden states into a single state regulated by update gate (z_t) and reset gate (r_t). A forward pass of GRU is expressed by the following equations.

$$\begin{aligned} z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\ r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\ \tilde{h}_t &= \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \end{aligned} \tag{4}$$

Theoretically, GRU is also capable of remembering long-term dependencies despite having fewer parameters than LSTM, making it interesting to compare.

4.2.6. Transformer

The final model that we implement is an Sequence to Sequence Transformer. Unlike recurrent models, Transformer relies on Multi-Head Self-Attention to process input sequence in parallel. This mechanism maps the input into Queries (Q), Keys (K), and Values (V) and then computes the attention weights using these equations.

$$\begin{aligned} V_h &= \beta_{v_h} 1^T + \Omega_{v_h} X, \\ Q_h &= \beta_{q_h} 1^T + \Omega_{q_h} X, \\ K_h &= \beta_{k_h} 1^T + \Omega_{k_h} X, \\ \mathrm{Sa}_h[X] &= V_h \cdot \mathrm{Softmax}\left[\frac{K_h^T Q_h}{\sqrt{D_q}}\right]. \end{aligned} \tag{5}$$

The model we are using implement a different Encoder-Decoder framework than the one explained previously. Both the encoder and decoder consist of 2 Transformer layers with dimension 64 each. Instead of a Bridge layer, the encoder creates a memory context that is used by the decoder to generate outputs auto regressively.

4.3. Training configuration

For all models, the training is performed using Huber loss function (also known as Smooth L1 Loss) shown in the equation 6. This loss function behaves quadratically (like MSE) for small errors and linearly (like MAE) for large errors. For optimizer, we use Adam Optimizer [11] with an initial learning rate of 10^{-3}. If validation loss fails to improve after a patience of 5 epochs, learning rate will be halved to a minimum of 10^{-6}. To save computing time, training will stopped if validation loss does not improve after 15 consecutive epochs.

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \le \delta \\ \delta \cdot \left(|y - \hat{y}| - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases} \tag{6}$$

Teacher forcing is applied only to all four models to prevent exposure bias in doing auto regressive prediction. The starting forcing ratio used is 0.5, meaning that 50 percent of the time ground truth is used during auto regressive step, with linear decay rate of 0.98. When validating and testing, teacher forcing ratio is set to 0 to ensuring the results is purely based on the model's performance.

5. RESULTS AND DISCUSSION

5.1. Quantitative evaluation

From Table 1, it is clear that all models statistically outperform the baseline in terms of mean distance error. At minute 5, RNN performs the best with a mean distance error of 90.5. This performance is notable given that it has the

Metric	LSTM	GRU	RNN	Transf.	Base.
Mean 5 min. (m)	98.10	92.50	90.50	117.60	–
Mean 25 min. (m)	520.60	490.80	516.70	584.70	1,750
Mean 50 min. (m)	1,539.80	1,505.10	1,564.00	1,681.60	2,530
Parameter $\times 10^3$	273.86	202.37	75.90	1,163.33	–

Table 1. Model performance comparison between three Recurrent Encoder-Decoder models and Transformer model. Mean distance error is computed in meters using Haversine distance function at a specific time steps.

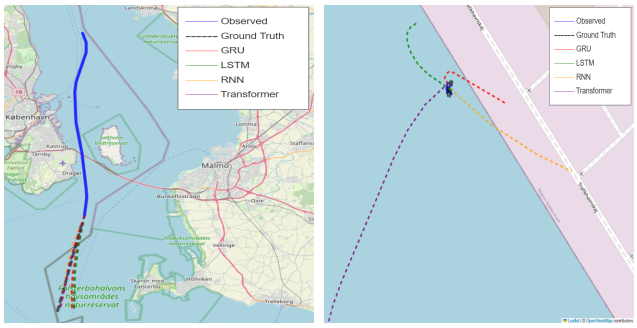

Fig. 2. Visual comparison of the predictions of four models. The left image shows a prediction from a sequence of moving vessel while the right image is from a sequence of stationary vessel. The blue line represents the input sequence (20 steps) and the dashed lines represent the output sequence (10 steps).

most lightweight model and most vessels travel straight. At longer horizons, we observe that the GRU performs the best in achieving the lowest mean distance error at minute 25 (490.80 m) and 50 (1505.10 m). Although LSTM has 35% more parameters compared to GRU, it performs worse. This is likely due to the data limitation causing the more complex model to overfit. Similarly, Transformer has the most parameters and it performs the worst out of the four models.

5.2. Qualitative evaluation

As shown in the left image of Figure 2, all four models predict moving vessel trajectories relatively accurately up to 10 steps (50 minutes) into the future. For stationary vessels, the models generally recognize this and predict low speed movements. However, since most of the sub-trajectories are not stationary, the models have learned that it is better to predict moving trajectories causing it to drift away in random directions from the harbor as shown in the right image of Figure 2. The models also do not have an understanding of land and water as some predictions are on land. When predicting more than 50 minutes into the future, we can see that the models struggle as shown in Figure 3. Because of the auto regressive prediction that lets the model feeds its slightly wrong prediction to predict the next step, the error at $t = 1$ will be compounded at $t = 30$.

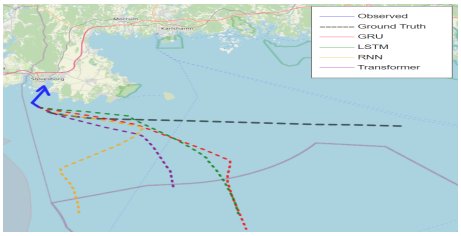

Fig. 3. Visual comparison of the predictions of four models. The blue line represents the input sequence (20 steps) and the dashed lines represent the output sequence (30 steps).

6. LIMITATIONS

Lack of time and computing resources is a big limitation in our project. Although 1.5 day dataset is reasonable because trips usually last less than a day, it was insufficient to fully train the more complex models like LSTM and Transformer. Due to the auto regressive nature of our approach, while effective for short term prediction, errors will start to compound significantly as we try to predict beyond than 50 minutes into the future. Also, the models currently are unable to differentiate land and water, leading to some predicted trajectories where the vessel cross through islands.

7. FUTURE WORK

Future research should prioritize increasing the dataset to the full summer seasons of multiple years, as maritime activity in Danish waters peaks during these months [12]. Only then we might be able to observe the generalization capabilities of complex models like Transformer. To enforce physical constraints, Geographic Raster Images (GRI) [13] should be implemented to penalize predictions that intersect with land. Furthermore, to avoid the error accumulation due to auto regressive decoding in long term prediction, non auto regressive (parallel) decoding strategies[14] should be experimented. Lastly, adding environmental variables to the dataset would likely improve prediction accuracy.

8. CONCLUSION

In this project, we modeled and evaluated different Sequence to Sequence deep learning architectures for short term maritime trajectory prediction. Among the models, the Encoder-Decoder GRU proved to offer the optimal balance of model complexity and performance. It outperforms the other three models and baseline with a mean distance error of 1.51 kilometers at a 50 minutes prediction.

9. REFERENCES

[1] Shuo Jiang, Lei Liu, Peng Peng, Mengqiao Xu, and Ran Yan, "Prediction of vessel arrival time to port: A review of current studies," *Maritime Policy & Management*, 2025.

[2] Xiaocai Zhang, Xiuju Fu, Zhe Xiao, Haiyan Xu, and Zheng Qin, "Vessel trajectory prediction in maritime transportation: Current approaches and beyond," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 19980–20003, 2022.

[3] Xian Ding, Hongwei Bian, Heng Ma, and Rongying Wang, "Ship trajectory generator under the interference of wind, current and waves," *Sensors*, vol. 22, no. 23, 2022.

[4] Yongfeng Suo, Wenke Chen, Christophe Claramunt, and Shenhua Yang, "A ship trajectory prediction framework based on a recurrent neural network," *Sensors*, vol. 20, no. 18, 2020.

[5] S. Thundiyil, S.S. Shalamzari, J. Picone, and S. McKenzie, "Transformers for modeling long-term dependencies in time series data: A review," in *2023 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, 2023, pp. 1–5.

[6] Kristian Aalling Sørensen, Peder Heiselberg, and Henning Heiselberg, "Probabilistic maritime trajectory prediction in complex scenarios using deep learning," *Sensors*, vol. 22, no. 5, Mar 2022.

[7] Nicola Forti, Lauren M Millefiori, Paolo Braca, and Peter Willett, "Uncertainty-aware recurrent encoder-decoder networks for vessel trajectory prediction," in *2021 IEEE 24th International Conference on Information Fusion (FUSION)*. IEEE, 2021, pp. 1–8.

[8] Danish Maritime Authority, "Ais data portal," `https://aisdata.ais.dk`, 2025, Accessed: 2025-11-29.

[9] DanPilot, "Transit pilotage and route distances in Danish waters," 2025, Accessed: 2025-12-05.

[10] International Telecommunication Union, "Recommendation ITU-R M.585-9: Assignment and use of identities in the maritime mobile service," `https://www.itu.int/`, 2022, Defines the 9-digit MMSI structure and Maritime Identification Digits (MID).

[11] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] Mikkel Handberg and Others, "AIS Trajectories from Danish Waters for Abnormal Behavior Detection," 2023, Accessed via DTU Data. States that normal traffic in the region has high seasonality related to fishing and leisure.

[13] Y. Ma and Others, "Vessel trajectory prediction with physical constraints and geographic information using geographic raster images," *Ocean Engineering*, vol. 303, pp. 117–129, 2024, Adjust author/title based on the exact PDF from your link.

[14] X. Chen, H. Zhang, and F. Zuo, "Vehicle trajectory prediction based on intention-aware non-autoregressive transformer with multi-attention learning for internet of vehicles," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2022.

DECLARATION OF USE OF GENERATIVE AI

This declaration **must** be filled out and included as the **final page** of the document. The questions apply to all parts of the work, including research, project writing, and coding.

- I/we have used generative AI tools: yes

If you answered *yes*, please complete the following sections. List the generative AI tools you have used:

- Gemini
- ChatGPT
- GitHub Copilot

Describe how the tools were used:

What did you use the tool(s) for? Used for debugging process of the coding part. Also to finalize the report and check any grammar mistakes in the report.

At what stage(s) of the process did you use the tool(s)? Coding and final check of the report

How did you use or incorporate the generated output? For coding, check whether it is finding the correct problems and use the proposed solution if it does what we want to do. For the report, after we found the mistakes, either use their proposed changes it it makes sense for us or just fix it manually.

GITHUB LINK

`https://github.com/lelegod/dl_maritime.git`