Reproduction of TrAISformer:

A Transformer Network for AIS-based Vessel Trajectory Prediction

Author: Polly Chen (Master Program AI for Business) Date: 23/12/2025

Abstract

Accurate vessel trajectory prediction is a cornerstone of Maritime Situational Awareness (MSA), essential for collision avoidance and traffic management. However, traditional regression-based models (such as LSTMs) often struggle to capture the complex, multimodal nature of vessel movements, leading to physically invalid predictions at navigation waypoints. This project reproduces the experimental results of **TrAISformer**, a state-of-the-art generative Transformer model that fundamentally reframes trajectory prediction from a regression task to a discrete classification problem.

By utilizing a novel "Four-hot" encoding scheme to discretize kinematic data (position, speed, and course), the model effectively captures long-range temporal dependencies and preserves the distinct movement options available to vessels. We validated the architecture using the **Danish Maritime Authority (DMA)** dataset, strictly adhering to the original training protocols. The reproduction was highly successful: our model achieved a prediction error of **0.48 nautical miles (nmi)** at the 1-hour horizon. This result represents an **exact match** with the performance reported in the original paper, empirically confirming the reproducibility of the TrAISformer architecture and its efficacy in maritime forecasting.

Chapter 1. Introduction

1.1 Context The rapid expansion of global maritime activities has made vessel trajectory prediction a cornerstone of Maritime Situational Awareness (MSA). Accurately anticipating vessel movements is critical for applications ranging from search and rescue operations to port management and collision avoidance. While the Automatic Identification System (AIS) provides a rich stream of kinematic data, forecasting trajectories remains challenging due to the heterogeneous and multimodal nature of vessel motion.

1.2 The Challenge Traditional methods, including Kalman filters and LSTMs, typically treat trajectory prediction as a regression problem. A major failure mode of regression models occurs at waypoints/intersections: when a vessel has distinct valid options, for example, like turning left or right, regression models tend to predict an invalid "average" path that merges these modes, often leading to collisions with static obstacles. For example, if a vessel can turn left or right around an island, a regression model predicts the middle directly into the island. Furthermore, standard recurrent networks often struggle to capture the long-range dependencies required for medium-range forecasting.

1.3 TrAISformer Contribution & Objective The TrAISformer model addresses these limitations by framing trajectory prediction as a classification problem. It employs a novel "Four-hot" representation that discretizes continuous coordinates (Lat, Lon, SOG, COG) into tokens. By leveraging a Transformer architecture, it captures long-term temporal patterns to autoregressively generate future positions. The primary objective of this project is to reproduce the experimental results of the original TrAISformer paper. We validate the model on the Danish Maritime Authority (DMA) dataset to confirm its ability to outperform regression baselines. Our reproduction achieves a 1-hour prediction error of 0.48 nautical miles, exactly matching the original paper's reported result.

Chapter 2. Related Work

2.1 Traditional Methods Early approaches to vessel trajectory prediction relied heavily on state-space models like Kalman Filters and the Curvilinear Motion Model (CMM). While effective for short-term estimation, these methods assume linear or simplified kinematic priors, failing to account for complex maneuvering patterns. Additionally, filtering techniques are prone to error propagation, making them unsuitable for medium-range forecasting.

2.2 Deep Learning Approaches (RNN/LSTM) Deep learning models, particularly LSTMs and GRUs, improved upon traditional methods by learning sequential patterns. However, they suffer from vanishing gradients and often struggle to retain long-term dependencies. A critical limitation is their reliance on regression-based formulations using Mean Square Error (MSE) loss. MSE assumes a unimodal Gaussian distribution, forcing the model to predict an "average" path when a vessel faces multiple valid directions, resulting in physically invalid trajectories that may intersect with land or obstacles.

2.3 Transformers and TrAISformer's Novelty Transformers, originally popularized by models like GPT in Natural Language Processing (NLP), revolutionized sequence modeling by utilizing self-attention mechanisms. This architecture processes input sequences in parallel, allowing the model to directly retrieve information from multiple past time steps simultaneously [Vaswani et al., 2017]. This offers a critical advantage over RNNs, which process data sequentially and often struggle to retain the long-term dependencies required to understand large-scale vessel movement patterns. In maritime trajectory prediction, forecasting a vessel's direction at waypoints often requires "backtracking" to distant historical states to understand its intent , a task where standard RNNs frequently fail.

The TrAISformer model leverages this architecture with a distinct novelty: it moves from regression to classification[Nguyen et al., 2021]. While traditional regression-based models (often using LSTMs) tend to merge valid paths into a single, invalid "average" trajectory when facing multiple route options , TrAISformer discretizes data (Latitude, Longitude, Speed, Course) into "Four-hot" tokens. This enables the model to output a categorical probability distribution over a discrete grid, effectively preserving the multimodal nature of vessel trajectories at intersections. Future positions are generated autoregressively, with each predicted token conditioned on all previous outputs.

Chapter 3. Methodology

3.1 Data Representation ("Four-hot Encoding") To address the heterogeneity of AIS data, which comprises position, speed, and course, we implemented the specialized "Four-hot" encoding scheme. Rather than using simple tokenization, continuous attributes (Latitude, Longitude, SOG, COG) are first discretized into N_lat, N_lon, N_SOG, N_COG bins. For each time step t, these discrete values are converted into four separate one-hot vectors, which are concatenated to form a single high-dimensional "Four-hot" vector h_t. This representation maps diverse kinematic features into a unified sparse embedding space, effectively regularizing the mapping to avoid overfitting while handling the multimodal nature of motion data.

3.2 Architecture The architecture adopts a Transformer Decoder structure that is similar to GPT models. The sequence of "Four-hot" vectors is first projected into a dense embedding space and combined with positional encodings to maintain temporal order, ensuring the model explicitly recognizes the sequential nature of the trajectory data. The network core comprises stacked self-attention layers, functioning as the encoder of information, that process the input sequence in parallel rather than sequentially. Through multi-head attention, the model captures long-term dependencies across the trajectory; this mechanism allows the model to "attend" to multiple distant past time steps simultaneously, such as previous turning points or port departures, to correctly infer the vessel's intent. Finally, the Decoder facilitates autoregressive token generation: the final output is a softmax distribution over the grid cells, representing the probability of the vessel moving to any given location. To predict a full trajectory, the model samples a position from this distribution and feeds it back into the network as the input for the next time step.

3.3 Training Strategy We framed the prediction task as a classification problem using Cross-Entropy (CE) Loss. Unlike Mean Square Error (MSE), which forces unimodal Gaussian assumptions and often merges valid paths into an invalid average (e.g., crashing into land), CE loss preserves the multimodal characteristics of the data. This allows the model to predict distinct probabilities for divergent paths at waypoints, ensuring that multiple valid potential trajectories are maintained rather than averaged out . The optimization process employed the AdamW optimizer [Loshchilov & Hutter, 2017]. AdamW improves upon the standard Adam optimizer by decoupling weight decay from the gradient update. This distinction is critical for deep Transformer architectures, as it prevents the adaptive learning rates from interfering with regularization, thereby ensuring better generalization and training stability. This optimizer was coupled with a cyclic cosine decay learning rate scheduler to further stabilize convergence. The model was trained on the Danish Maritime Authority (DMA) AIS dataset, consistent with the benchmarks established in the original study.

Chapter 4. Experimental Setup

4.1 Dataset The experiments were conducted using the Danish Maritime Authority (DMA) AIS dataset. This selection was critical to ensure direct comparability with the primary state-of-the-art benchmarks reported in Table I of the original paper [Nguyen et al., 2021]. The

data was preprocessed into trajectory sequences with fixed time intervals, matching the reference implementation provided in the repository.

4.2 Environment & Implementation Modifications Training was executed on Google Colab using a standard T4 GPU runtime. While initial code analysis was performed on a local M4 MacBook, the computational demands of the Transformer architecture necessitated cloud-based GPU acceleration. We addressed two significant technical hurdles during the reproduction process. First, the original codebase relied on the deprecated `.next()` iterator method, which caused runtime failures in the modern Python 3.12 environment; we resolved this by refactoring the data loaders to use the standard `.__next__()` syntax. Second, to mitigate data loss from Colab session timeouts and GPU quota exhaustion (which required rotating Google accounts), we modified the configuration scripts to mount Google Drive, ensuring persistent storage for model checkpoints and training logs.

4.3 Hyperparameters & Evaluation The model was trained for a maximum of 50 epochs. Crucially, we adhered to the paper's methodology by monitoring validation loss to identify the optimal model checkpoint. Consistent with the paper's strategy, we observed that Epoch 10 achieved the lowest validation loss (1.38) and selected this checkpoint for evaluation, rather than the final epoch. Performance was evaluated by computing prediction error (Haversine distance) at three time horizons: 1 hour, 2 hours, and 3 hours. These metrics were initially computed in kilometers and subsequently converted to nautical miles (nmi) to enable a precise, direct comparison against the original paper's findings.

Chapter 5. Results & Discussion

5.1 Quantitative Analysis We evaluated the optimal model checkpoint (Epoch 10) on the held-out test set comprising 1,453 trajectories. To validate reproducibility, we converted our raw output metrics from kilometers to nautical miles ($1 \text{ nmi} \approx 1.852 \text{ km}$) and compared them directly against Table I of the original paper.

Prediction Horizon	**Our Result (km)**	**Our Result (nmi)**	**Paper Table I (nmi)**	**Status**
1 Hour	0.89 km	**0.48 nmi**	**0.48 nmi**	**Exact Match**
2 Hours	1.70 km	**0.92 nmi**	0.94 nmi	Successful
3 Hours	2.79 km	**1.51 nmi**	1.64 nmi	Successful

The analysis confirms a highly successful reproduction. Most notably, the 1-hour prediction error of 0.48 nmi matches the original paper's reported result exactly. This validates that the implementation of the "Four-hot" encoding and Transformer architecture functions precisely as described in the literature. As illustrated in our error plots (prediction_error.png), the displacement error increases approximately linearly with prediction time, which is consistent with trajectory prediction physics.

5.2 Training Dynamics and Overfitting The training process exhibited a sharp divergence between training and validation performance after the initial convergence phase. The model achieved its optimal validation loss (1.38) at Epoch 10.

Beyond Epoch 10, clear signs of overfitting emerged:

1. **Training Loss** continued to decrease significantly (dropping from 0.75 to -1.68).
2. **Validation Loss** began to deteriorate, rising from 1.38 to 3.91 by Epoch 50.

This divergence is a classic indicator of overfitting. Notably, the Cross-Entropy loss function can produce negative values when the model becomes overconfident on training samples, assigning extremely high probabilities to correct labels.

This behavior indicates that the model's capacity (≈ 57 million parameters) is large relative to the dataset size, leading to the memorization of training samples if left unchecked. This observation empirically validates the paper's methodology of utilizing Early Stopping. By selecting the checkpoint at Epoch 10 rather than the final epoch, we ensured the model retained its generalization capability. Consequently, the optimal model (Epoch 10) achieved state-of-the-art performance on the test set, as detailed in Section 5.1.

5.3 Limitations While the reproduction was successful, three limitations persist. First, the evaluation relies solely on the DMA dataset. Second, the model relies exclusively on AIS kinematics, ignoring external factors like weather or ocean currents which influence vessel motion. Finally, the architecture is computationally expensive, necessitating high-end GPUs (T4 or better) for training.

Chapter 6. Conclusion

This project successfully reproduced the TrAISformer architecture and its experimental results using the Danish Maritime Authority (DMA) dataset. By implementing the novel "Four-hot" encoding and a GPT-style Transformer decoder, we validated the efficacy of framing trajectory prediction as a classification task rather than a regression problem.

Our experimental evaluation yielded a prediction error of **0.48 nautical miles (nmi)** at the 1-hour horizon, matching the state-of-the-art results reported in the original paper exactly. This finding confirms that the model’s discrete grid-based approach effectively captures multimodal vessel dynamics—such as distinct turning options at waypoints—where traditional regression baselines (e.g., LSTMs, Kalman Filters) typically fail by averaging potential paths.

In conclusion, the successful replication of these metrics validates the TrAISformer as a robust solution for medium-term maritime forecasting. The study highlights the superiority of self-attention mechanisms in modeling long-range dependencies, provided that early stopping is applied to mitigate the overfitting inherent in large-capacity models.

Chapter 7. References

[1] D. L. Nguyen, R. Vadaine, G. Hajduch, R. Gribonval, and R. Fablet, "TrAISformer—A Generative Transformer for AIS Trajectory Prediction," *arXiv preprint arXiv:2109.03958*, 2021.

[2] Danish Maritime Authority (DMA), "Historical AIS Data," [Online]. Available: https://dma.dk/safety-at-sea/navigational-information/ais-data.

[3] A. Vaswani et al., "Attention is All You Need," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5998–6008. (Foundational Transformer paper)

[4] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *arXiv preprint arXiv:1711.05101*, 2017. (AdamW Optimizer)

[5] A. Radford et al., "Improving Language Understanding by Generative Pre-Training," OpenAI, 2018. (GPT Architecture)

Appendix: Complete Reproduction Script

The following code was executed on Google Colab to reproduce the TrAISformer results:

```
# Mount Google Drive

from google.colab import drive
drive.mount('/content/drive')

# Clone repo
!git clone https://github.com/CIA-Oceanix/TrAISformer.git
%cd TrAISformer

# Fix Python 3.12 compatibility issue
!sed -i 's/.next()/.__next__()/g' trainers.py

# Save directly to Drive
!sed -i 's|./results|/content/drive/MyDrive/TrAISformer_results|g' config_trAISformer.py

# Create results folder
!mkdir -p /content/drive/MyDrive/TrAISformer_results

# Run training
!python trAISformer.py
```