Journal of *Marine Science and Engineering*

MDPI

Article

Contrastive Learning for Graph-Based Vessel Trajectory Similarity Computation

Sizhe Luo [1], Weiming Zeng [1,*] and Bowen Sun [2]

[1] Digital Image and Intelligent Computation Lab, Shanghai Maritime University, Shanghai 201306, China
[2] Intelligent Science and Technology Lab, Shanghai Polytechnic University, Shanghai 201206, China
* Correspondence: zengwm86@163.com

Abstract: With the increasing popularity of automatic identification system AIS devices, mining latent vessel motion patterns from AIS data has become a hot topic in water transportation research. Trajectory similarity computation is a fundamental issue to many maritime applications such as trajectory clustering, prediction, and anomaly detection. However, current non-learning-based methods face performance and efficiency issues, while learning-based methods are limited by the lack of labeled sample and explicit spatial modeling, making it difficult to achieve optimal performance. To address the above issues, we propose CLAIS, a contrastive learning framework for graph-based vessel trajectory similarity computation. A combined parameterized trajectory augmentation scheme is proposed to generate similar trajectory sample pairs and a constructed spatial graph of the study region is pretrained to help model the input trajectory graph. A graph neural network encoder is used to extract spatial dependency from the trajectory graph to learn better trajectory representations. Finally, a contrastive loss function is used to train the model in an unsupervised manner. We also propose an improved experiment and three related metrics and conduct extensive experiments to evaluate the performance of the proposed framework. The results validate the efficacy of the proposed framework in trajectory similarity calculation.

Keywords: trajectory similarity computation; graph neural network; contrastive learning

Citation: Luo, S.; Zeng, W.; Sun, B. Contrastive Learning for Graph-Based Vessel Trajectory Similarity Computation. *J. Mar. Sci. Eng.* **2023**, *11*, 1840. https://doi.org/10.3390/jmse11091840

Academic Editor: Marco Cococcioni

Received: 25 August 2023
Revised: 9 September 2023
Accepted: 20 September 2023
Published: 21 September 2023

1. Introduction

AIS devices play a crucial role as onboard navigation aids for communication and data exchange between vessels and shores. These devices utilize very-high-frequency (VHF) radio transceivers to broadcast vessel information to nearby vessels or shore-based stations while also receiving AIS data transmitted by other vessels. AIS data provide essential information such as vessel identification, characteristics, real-time position, velocity, heading, and other relevant details, which contribute to maritime traffic management and collision avoidance. In recent years, extensive research has been conducted to extract the spatiotemporal distribution patterns of vessels from historical AIS data, enhancing our understanding of waterway traffic patterns [1]. Among these studies, the analysis of AIS trajectories, derived from real-time position information in AIS messages, has garnered significant attention across various research domains. The computation of AIS trajectory similarity is a crucial task for several maritime applications, including trajectory clustering, prediction, and anomaly detection. This necessitates the development of effective methods and measures for assessing the similarity between AIS trajectories, facilitating improved insights and decision-making in maritime operations [2–4].

The current research focus on trajectory similarity computation revolves around utilizing learning-based methods to indirectly calculate trajectory similarity through trajectory representation learning. Feature-based approaches are employed to learn the feature representation of trajectory data, enabling the mapping of trajectories with varying positions, shapes, and lengths to a shared low-dimensional feature space. Subsequently, traditional

distance metrics such as Euclidean distance or cosine similarity are utilized to compute the similarity between trajectories. However, the application of learning-based trajectory similarity computation for vessel AIS data is still relatively limited, and there is a dearth of modeling approaches that explicitly capture trajectory spatial dependencies through graph learning and other techniques.

Vessel trajectories depict the continuous movement of vessels in three-dimensional physical space over time. Ideally, a complete vessel trajectory in the real world is a mathematically continuous and smooth curve, referred to as a latent path in this paper. However, in practical situations, shipborne devices are unable to capture a continuous curve that precisely represents the underlying latent path associated with a given trajectory. Instead, they can only gather discrete positions of the vessel at irregular time intervals. As a result, the challenge in trajectory representation learning lies in acquiring a representation vector that effectively captures the genuine underlying latent path, considering the sparsely and irregularly sampled trajectory data points.

In the research on learning-based trajectory similarity, there are mainly two methods: supervised learning and unsupervised learning. Supervised learning aims to utilize neural network methods to fit existing similarity measures in order to improve computational efficiency; on the other hand, unsupervised learning methods do not have existing measures as supervised signals, so each new unsupervised method develops new similarity measures. Unsupervised learning is an important approach in the field of machine learning as it can extract useful information and knowledge from unlabeled data. The automatic learning of general vessel trajectory representations from massive unlabeled AIS trajectory data is of great significance for ship trajectory similarity calculation. First, ship trajectory data often exhibit characteristics such as long time spans, high dimensionality, complexity, and noise [5–8]. Therefore, effective representation methods are needed in order to reduce the dimensionality and complexity of the data, enabling better exploration of the intrinsic structure and features of vessel trajectory data. Second, in order to enhance the model's ability to measure trajectory distances, new trajectory measurement methods need to be proposed. The development of new trajectory representation learning models can only be achieved through unsupervised learning. Finally, vessel trajectory data have a wide range of applications in practical scenarios, such as route planning, trajectory prediction, trajectory clustering, and anomaly detection. Effective general ship trajectory representations can provide strong support and assurance for these application scenarios. Therefore, learning general trajectory representations from massive unlabeled trajectory data holds significant research value and practical importance.

To address the aforementioned issues, this paper proposes a graph neural network-based framework called CLAIS for the unsupervised learning of optimal ship trajectory representations. By calculating the distances between the learned trajectory representation vectors, the framework determines ship trajectory similarity, which facilitates subsequent research tasks such as ship clustering and anomaly trajectory detection. The contributions of this paper are summarized as follows:

1. A graph-based trajectory contrastive learning framework, CLAIS, is proposed. It constructs similar trajectory samples to learn robust trajectory representation vectors and computes trajectory similarity based on the Euclidean distance between representation vectors, leading to favorable similarity results.
2. A parameterized trajectory augmentation method is introduced to enhance the robustness of the model's trajectory representation learning.
3. Improved evaluation experiments and three evaluation metrics are proposed to verify the performance of the proposed framework in learning trajectory representations and computing ship trajectory similarities.

The remaining structure of this paper is as follows: Section 2 introduces the relevant research; Section 3 provides a detailed description of the proposed model framework; Section 4 presents the proposed improvement experiments and details of the model evaluation experiments; and Section 5 concludes the work of this paper.

2. Related Work

To perform trajectory similarity computation using trajectory representations, it is crucial to obtain effective trajectory representations. Currently, many popular methods utilize deep neural networks to capture the feature representations of trajectories and map them into a feature space. Yang et al. [9] proposed a deep learning-based trajectory similarity computation model called T3S, which fits different trajectory similarity measures. By employing long short term memory (LSTM) and self-attention-based networks, T3S can retain the spatial and structural information of trajectories for similarity computation. It can automatically adjust the weights of spatial and structural information based on different similarity measures. Yang et al. [10] introduced TMN, which matches points from one trajectory to points from another trajectory using an attention mechanism that enables cross-trajectory point matching. The trajectory's spatial information is then combined, and a recurrent neural network (RNN) is used to learn the trajectory representation for fitting different similarity measures. Zhang et al. [11] proposed Traj2SimVec for scalable and robust trajectory similarity computation. Traj2SimVec acquires triplet training samples through fast trajectory compression and indexing. It further utilizes sub-trajectory similarity information as auxiliary supervision. Additionally, the framework supports point matching queries by modeling the optimal matching relationships of trajectory points under different distance metrics. Yao et al. [12] introduced TrajGAT, which constructs trajectories as quadtree structures and employs attention heads from the Transformer [13] instead of graph attention networks (GAT) to learn trajectory representations for different similarity measures. However, the aforementioned methods are supervised models based on existing similarity computation methods. While they improve computational efficiency, they cannot achieve better performance when facing the similarity computation performance bottlenecks in existing models.

In unsupervised representation learning, Yao et al. [14] use sliding windows to extract a set of spatiotemporal invariant features that capture the motion characteristics of trajectories. They convert each trajectory into a feature sequence to describe the object's movement using a feature extraction module. They further employ a seq2seq autoencoder to learn the trajectory representation. Li et al. [15] apply computer vision techniques to measure similarity in vessel trajectories. They propose a similarity measurement method based on a convolutional autoencoder, where vessel trajectories are transformed into trajectory image data, treating each grid as a pixel. They then introduce a grid-based convolutional autoencoder to extract feature vectors from trajectory data to learn the representation of the original vessel trajectories. Fu et al. [16] propose a representation learning framework called Trembr, which models trajectories and road segments separately. They designed an encoder–decoder model called Traj2Vec based on a recursive neural network. By leveraging the underlying road network and matching segments obtained using road network matching techniques, they constrain the learning process. They also introduce Road2Vec, a neural network-based approach to learn segment embeddings in the road network, capturing various relationships between road segments. However, the aforementioned models do not explicitly model the spatial structural features of trajectories using a graph, which may compromise the model's performance.

In the field of unsupervised learning, self-supervised learning is currently an important trend and considered the future of unsupervised learning [17,18]. Currently, self-supervised learning models can be mainly classified into two types: generative and contrastive learning models. Traditional self-supervised learning methods based on generative models, such as autoencoders, attempt to generate or model specific parts of input samples by using a limited number of discrete trajectory positions in the input space, thereby inferring an approximate trajectory movement curve of the input sample [19,20]. However, this approach aims to reconstruct the entire input sample by comparing the input original sample with the reconstructed sample to compute the error and train the model. For sequential data such as trajectories or time series, an autoregressive paradigm is often employed, where the prediction result of the current position point $f(x_{t-1})$ is obtained by using the fitted

result of the previous position point $f(x_t)$ as the input, i.e., $f(x_t) = f(f(x_{t-1}))$. Generating reconstructed data samples is computationally expensive, and this fine-grained approach is often unnecessary for learning models that can distinguish between different samples. In fact, it can even lead to performance degradation due to excessive focus on sample details.

Fortunately, contrastive learning provides a new approach to address this problem. The core idea of contrastive learning is to compare different trajectory samples and establish relationships among similar samples (i.e., positive samples) and differences among different samples (i.e., negative samples). Through this comparison, the model can learn the intrinsic structure and features of trajectory data, thus generating effective trajectory representations that serve as a basis for subsequent trajectory analysis and processing. As an unsupervised learning method, contrastive learning also does not require labeled data, which reduces the barriers for practical applications and avoids the cost and complexity associated with annotation. In contrast to generative methods that attempt to fully reconstruct the input original trajectories, discriminative methods learn representations through objective functions similar to supervised learning, but the input and supervisory signals during network training come from unlabeled datasets. Discriminative contrastive learning methods based on latent space have shown great potential and achieved state-of-the-art results [21,22]. Contrastive learning directly finds discriminative features that best differentiate different trajectory samples in the feature space by comparing similar and dissimilar samples within the sample set. Compared to generative methods, contrastive learning is more direct, simple, and effective for discriminative tasks.

Currently, existing literature has attempted to apply contrastive learning to the processing and analysis of trajectory data [23,24]. However, there is currently no existing method for calculating similarity of ship AIS trajectories based on contrastive learning. Therefore, this paper combines contrastive learning and graph learning to facilitate ship trajectory representation learning, thereby calculating the distance between ship trajectory representations to obtain vessel trajectory similarity. The CLAIS framework proposed in this article benefits from the following three aspects with respect to performance: first, by incorporating water area pretraining into trajectory graph construction, CLAIS learns the spatial structural representation of water area in advance before modeling trajectories; second, through the proposed parameterized augmentation scheme, CLAIS enhances its robustness against erroneous AIS signals by learning potential signal errors through contrastive learning with real trajectories; and third, by introducing graph neural networks to learn the representation of trajectory graphs, CLAIS strengthens its ability to learn the spatial dependency relationships of trajectories in water areas. Compared to previous research, CLAIS achieves better model performance through improvements in these three aspects.

3. Methodology

As shown in Figure 1, the CLAIS framework consists of three modules: the regional graph pretraining module in the red box; the vessel trajectory contrastive learning module in the yellow box; and the trajectory graph representation learning module in the green box. After training, the specific similarity between trajectories can be obtained by calculating the Euclidean distance of trajectory representation vectors output by the trajectory graph representation model.

The regional graph pretraining module, located in the red box, first gridizes the study area and establishes a spatial structural relationship graph based on the selected grid cells. Then, the spatial dependency of the regional structural graph is pretrained, resulting in pretraining representation vectors for each effective spatial grid.

The vessel trajectory contrastive learning module, highlighted in the yellow box, is the main part of training in the CLAIS framework. It learns trajectory representation vectors that minimize the loss through unsupervised contrastive learning. Initially, the input ship trajectories undergo various types and methods of trajectory augmentation using the proposed parameterized combination enhancement scheme, creating diverse similar

samples. The trajectory samples are then processed by the trajectory graph representation learning module to extract trajectory features and encode them into trajectory representation vectors. During the training phase, the trajectory representation vectors are further passed through a non-linear mapper. Finally, the contrastive loss component calculates the error to optimize the model.

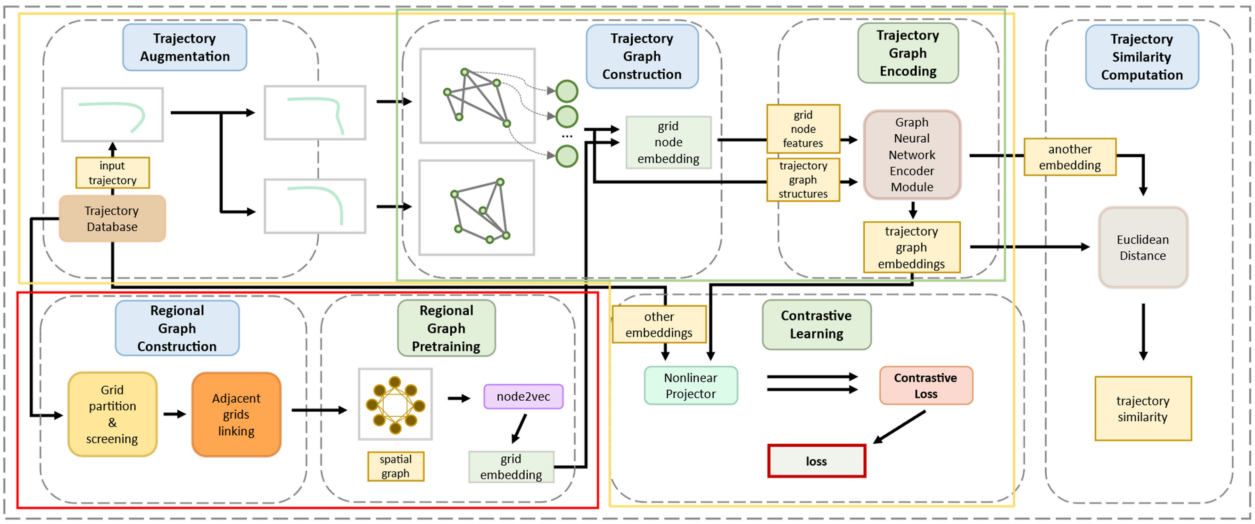

Figure 1. Overview of CLAIS framework.

The trajectory graph representation learning module, surrounded by the vessel trajectory contrastive learning module, is an independent module that includes a trajectory graph construction component and a proposed graph neural network encoder. The input trajectories are first used by the trajectory graph construction component, combined with the grid representation vectors obtained from the regional graph pretraining module, to construct a trajectory map. Then, the graph neural network encoder performs feature extraction on the trajectory graph, generating representation vectors for the trajectories.

Next, we will separately introduce the structural and methodological details of each module.

3.1. Regional Graph Pretraining Module

The regional pretraining module consists of two components: the region graph construction component and the node2vec pretraining component.

Figure 2 demonstrates the steps of the regional graph pretraining module. The region graph construction component divides the study area into adjacent and non-overlapping grids of equal size. It maps all the position points in the historical AIS database that fall within a grid to that grid. After the grid mapping, grid filtering is performed. Specifically, the ship trajectory database T is divided into square grids of equal size based on the given area range, with the grid size determined by the experimental parameter "grid_size". The grids containing the number of position points from all the points in T are then counted. Subsequently, a natural number parameter δ is manually selected, which represents the threshold for the number of historical position points contained in a grid. By removing grids in the area that do not contain position points or have a small number of position points, grids that are not suitable for navigation from a data-driven perspective (such as regions with obstacles such as islands or artificial structures) and grids that may contain noise points are filtered out. The selected valid grids make the model training more robust and construct a waterway chart that is more in line with reality, reducing unnecessary computational complexity. Figure 3 shows the heatmap of historical AIS signal position points received by onshore base stations in the gridified Shanghai Port waterway, with a fixed grid size of 0.01°. It can be observed that as the threshold δ gradually increases, the grids containing noise signals (such as grids clearly corresponding to land areas) become

fewer. However, a negative consequence is that some normal and sparsely populated water area grids are also filtered out. Additionally, to handle position points in trajectories that fall in invalid grids, an abstract grid is introduced to represent all the invalid grids when dividing and constructing the water area graph. All trajectory position points falling in invalid grids are correspondingly assigned to that grid. This grid does not contain the real-world features present in normal valid grids, such as the central position of the grid. Instead, it uses a zero vector representation to replace the normal grids for learning embedding representation vectors.

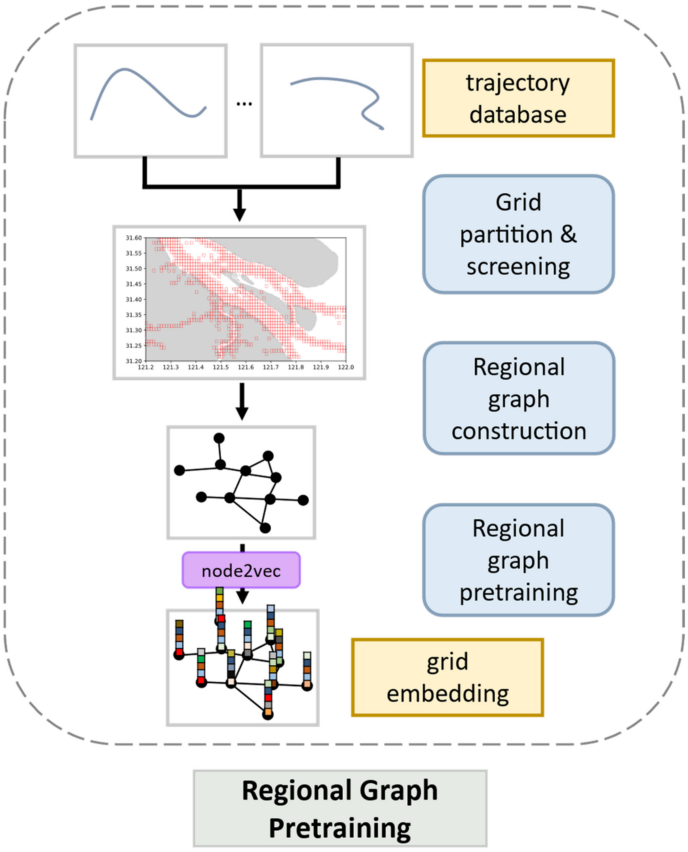

Figure 2. Overview of the regional graph pretraining module.

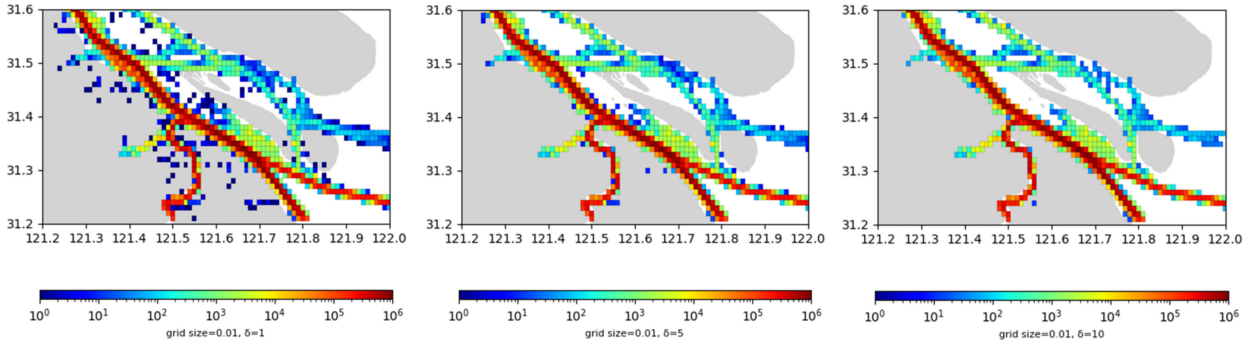

Figure 3. Gridification and historical heatmaps of the study area under different signal thresholds in the water domain. The grid size is set to 0.01°, and δ ranges from left to right as 1, 5, and 10.

After completing the spatial grid structure partitioning, CLAIS utilizes node2vec [25] to perform pretraining on the spatial graph structure. This process generates embedded representation vectors for all valid grids, which are then used to construct trajectory graph features in subsequent steps.

3.2. Vessel Trajectory Contrastive Learning Module

After completing the region pretraining, the vessel trajectory contrastive learning module takes the training set of trajectory data as input to train the model. Inspired by previous research on contrastive learning [26], this module applies the self-supervised learning paradigm to learn representations of ship trajectories by leveraging the dissimilarity between data samples as the model loss, aiming to learn the most discriminative trajectory representations.

Specifically, the module learns the representation model through contrastive learning by automatically constructing similar instances (positive samples) and dissimilar instances (negative samples). This is referred to as the trajectory graph representation learning module of CLAIS. It ensures that positive samples projected in the embedding space by the representation model are close in distance, while negative samples are far apart. CLAIS follows the principles of SimCLR and employs a parameterized trajectory augmentation scheme to construct positive and negative sample pairs. The trajectory graph representation learning module encodes trajectories into feature vectors, enabling different positive samples generated from the same trajectory to have closer distances in the feature space. The positive and negative sample pairs are constructed as follows: for a batch of trajectory samples, two different augmentation techniques are randomly applied to a particular trajectory sample, resulting in two trajectory samples that are positive samples to each other, while the remaining random trajectories in the batch serve as negative samples.

The vessel trajectory contrastive learning module consists of two components: the vessel trajectory augmentation component and the contrastive learning training component. Figure 4 illustrates the schematic diagram of this module, with the vessel trajectory augmentation component highlighted in red and the contrastive learning training component highlighted in green. The following sections will provide detailed explanations of each component.

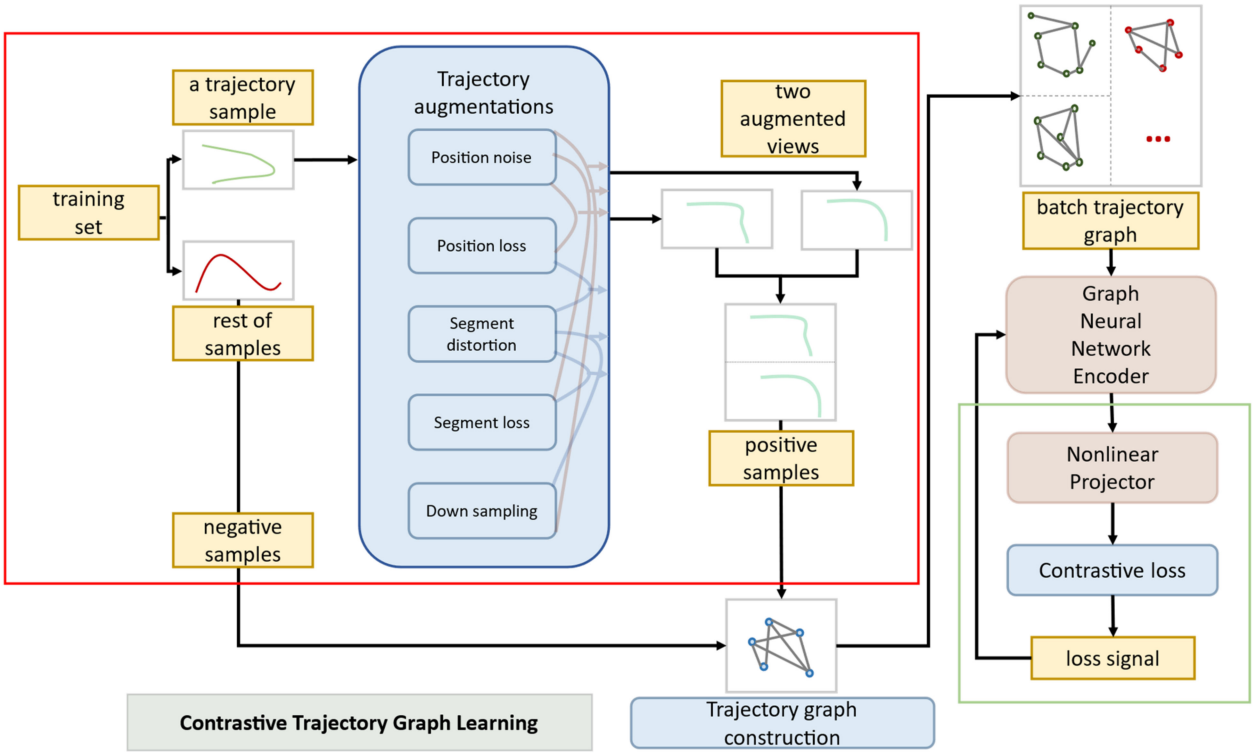

Figure 4. Trajectory contrastive learning module.

(1) Vessel trajectory augmentation component

As mentioned earlier, generating reliable augmented samples (referred to as views) from input trajectory samples to form similar positive samples is an important step. When it

comes to enhancing ship trajectory samples, it is crucial to design reasonable, effective, and interpretable augmentation schemes to facilitate effective trajectory contrastive learning.

In the field of contrastive learning, initially proposed in computer vision, image data can be augmented through techniques such as affine transformations, cropping, color variations, and noise. However, most of these augmentation techniques may not be applicable to trajectory data. Trajectory data exhibit clear sequential and geometric properties, and the aforementioned augmentation techniques either do not apply to trajectories or may disrupt the intrinsic characteristics of the data, making the learned trajectory representations unable to accurately reflect the true attributes of the trajectories.

Due to the constrained nature of ship movement within a limited sea surface, it can be simplified as two-dimensional plane motion. This study focuses solely on the geometric and sequential features of trajectories, thus excluding other information such as timestamps, heading, and speed. Referring to previous research on trajectory augmentation schemes [27,28], this study innovatively proposes a combined vessel trajectory augmentation scheme, which includes two novel augmentation methods tailored to vessel signal trajectories. The parameterized vessel trajectory augmentation scheme proposed in this study consists of two augmentation scales: random position augmentation and random segment augmentation. It also incorporates three augmentation methods: random noise, segment distortion, and downsampling, resulting in a total of five individual augmentation methods. Furthermore, to strengthen the impact of augmentation techniques on the model, this study combines these individual methods in a meaningful way, resulting in six combined augmentation methods. In total, there are eleven individual and combined augmentation methods. The following section will introduce these augmentation methods in detail.

1) Random position noise

Adding noise is the most common and effective means of augmenting data, which introduces distortions at the scale of individual position points. In the experiments conducted in this study, random positional noise is applied by randomly selecting positions to introduce noise based on the given parameter r_p (position ratio), which represents the proportion of position points in the entire trajectory that will be affected by noise. Since CLAIS adopts a grid-based approach and AIS trajectories can experience significant trajectory drift and positional errors, to enhance the model's robustness against noise, the magnitude of the noise $(d_p^{xi},\ d_p^{yi})$ is calculated by multiplying a manually set constant "base_distortion" and the experimentally controlled d_p (position distortion) as coefficients. These coefficients are then multiplied by samples drawn from a standard Gaussian distribution $N\ (0,\ 1)$. The position of a specific point $(x_i,\ y_i)$ on a trajectory after introducing noise is given by the following equation:

$$\left(x_i^{'},\ y_i^{'}\right) = (x_i,\ y_i) + \left(d_p^{xi},\ d_p^{yi}\right). \tag{1}$$

Here, $d_p^{ci} = d_p \cdot bd \cdot e_{ci}$, where bd represents the base distortion set to 0.01° in this study (both longitude and latitude), and $e_{ci} \sim N(0,1)$, $ci = xi\ or\ yi$. It can be observed that not only the magnitude of noise between longitude and latitude within the same coordinate differs, but the noise magnitudes between random positional noise are also independent of each other. This design ensures the authenticity and diversity of the learned random positional noise.

In Figure 5, (a) represents an original AIS trajectory segment without obvious positional errors or anomalies before augmentation; (b) shows the trajectory after applying random positional noise. Green dots represent the original normal position points, red dots represent the erroneous position points after introducing noise, and yellow dots represent the corresponding original normal position points before the occurrence of noise. It can be observed that the points affected by noise and the degree of noise in the augmented trajectory are random, ensuring that the generated noise follows a Gaussian distribution in each training iteration. In the figure, d_p is set to 2, indicating that the longitude and latitude noise follows a standard Gaussian distribution with a standard deviation of 0.02°. pr is set

to 0.2, indicating that approximately one-fifth of the trajectory position points are affected by noise.

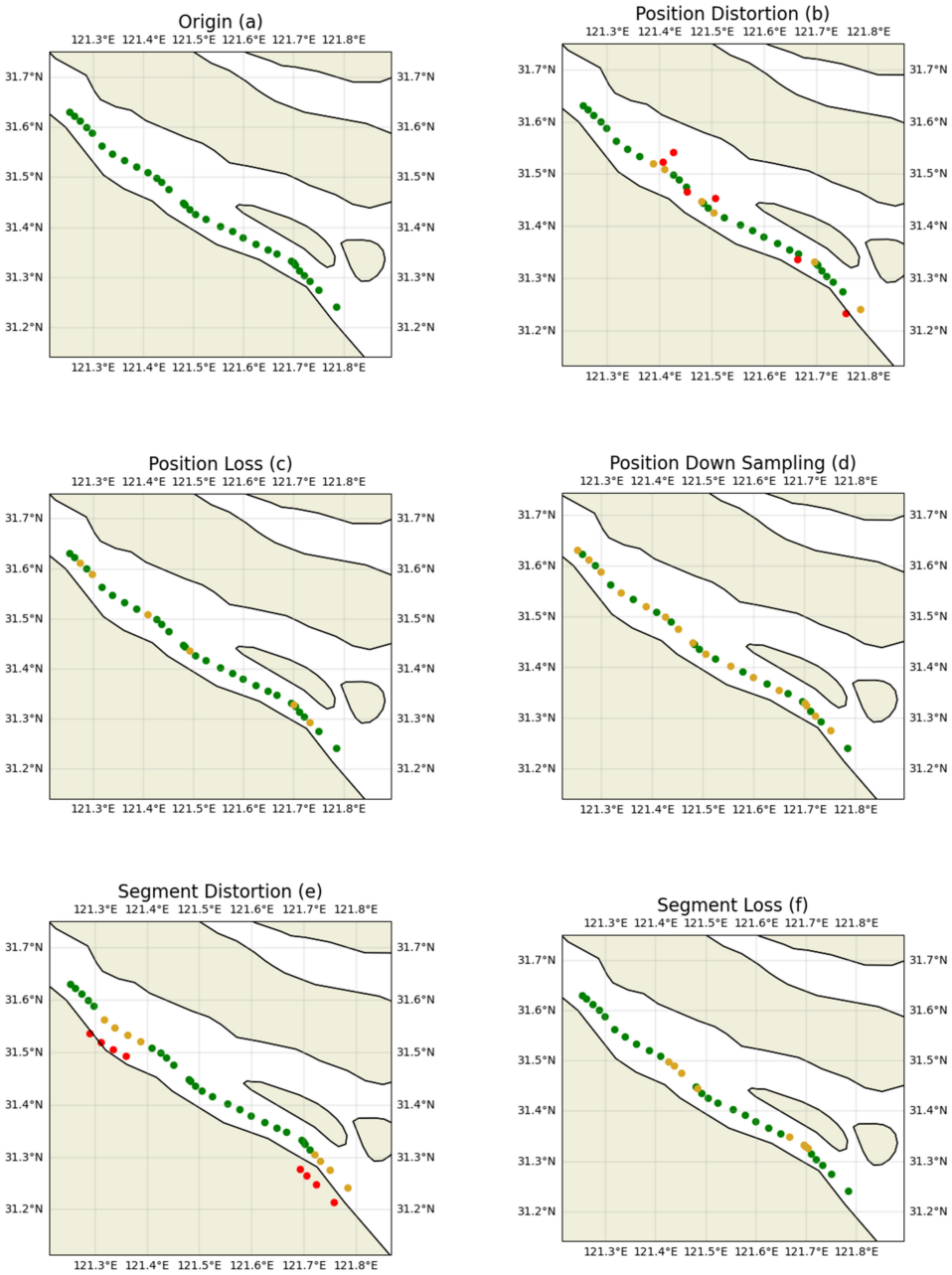

Figure 5. Trajectory enhancement methods in CLAIS: (**a**) original trajectory; (**b**) random position noise; (**c**) random position loss; (**d**) regular downsampling; (**e**) random segment distortion; (**f**) random segment loss. Green, red and yellow dots represent normal positions, error positions and missing normal positions respectively.

2) Random position loss

Random position loss is one of the most common errors in AIS trajectories, resulting in the loss of position points. AIS messages may be lost due to occasional communication disruptions, where they fail to be received by the vessel or shore-based stations. They can also be filtered out by preprocessing programs due to obvious errors, leading to random position signal loss. In CLAIS, the parameter l_p is used to quantify the proportion of randomly lost position points in the entire trajectory, and the lost positions are generated completely randomly based on a uniform distribution. As shown in Figure 5c, green dots

represent normal position points, while yellow dots represent positions where signal loss has occurred. It can be observed that the positions where signal loss occurs are random. In this example, the value of l_p is 0.2, indicating that one-fifth of the position points experience random position loss.

3) Regular downsampling

Downsampling is a common data augmentation technique where trajectory downsampling simulates the reduction of the trajectory data sampling rate by retaining a fixed number of position points at regular intervals and discarding the remaining ones. By applying regular downsampling to trajectories, the model can learn results that are more aligned with scenarios where the data is generated with a lower sampling rate along latent paths. In the augmentation scheme of CLAIS, the downsampling rate is controlled by an integer i_p (position interval), where i_p represents selecting every i_p position point from the original trajectory to construct a new augmented trajectory. As shown in Figure 5d, green dots represent the retained position points after downsampling, while yellow dots represent the discarded position points. In this example, the sampling interval i_p is set to 2, indicating a downsampling rate of 50%.

4) Random segment distortion

Random segment distortion is an enhancement technique in CLAIS specifically designed for the characteristics of AIS trajectories at the segment scale. It introduces noise of the same direction and magnitude to trajectory segments, visually presenting the effect of distorting a single segment within the trajectory. Due to equipment and system errors, AIS may experience consecutive segments with the same noise. This continuous occurrence of fixed noise can easily mislead the recognition model into believing that the vessel trajectory indeed traverses the positions indicated by the received signals, resulting in significant performance loss. Similar to random positional noise, the random segment distortion enhancement method is controlled by two parameters: r_s (segment ratio) and d_s (segment distortion). The parameter r_s determines the proportion of positions in consecutive segments where the noise occurs. The key distinction is that the added noise is the same for all positions within a segment, rather than multiple independent noises. For a trajectory $T = [\ldots, (x_m, y_m), \ldots, (x_n, y_n), \ldots]$, let $T_S = [(x_m, y_m), \ldots, (x_n, y_n)]$, $T_S \subset T$ be a segment within T. After distortion, the distorted segment is denoted as $T_S^{'}$, where for $\forall (x_i^{'}, y_i^{'}) \in T_S^{'}$,

$$\left(x_i^{'}, y_i^{'}\right) = (x_i, y_i) + \left(d_s^x, d_s^y\right). \tag{2}$$

Here, $d_p^c = d_p \cdot bd \cdot e_c$, where bd represents the baseline error and $e_c \sim N(0, 1)$, $c = x \text{ or } y$. It is important to note that in addition to the parameters mentioned above, CLAIS also includes a parameter n_s to control the number of augmented segments. In Figure 5e, with d_s set to 2 and a distortion occurrence rate of r_s = 0.2, when n_s = 2, there are two distorted segments. The meaning represented by the green, red, and yellow colors is the same as that of random positional noise. Green indicates normal positions, while yellow represents the corresponding normal positions before the occurrence of the red noise.

5) Random Segment Loss

Random segment loss is another enhancement method in the CLAIS augmentation scheme specifically designed for AIS trajectories at the segment scale. Random segment loss is also a common anomaly in AIS data, such as consecutive signal loss due to station malfunctions or the removal of trajectory segments with abnormal vessel speeds by preprocessing programs. Incorporating this augmentation technique can greatly improve the model's ability to handle consecutive segment losses and reduce the occurrence of the model recognizing disconnected segments as multiple independent trajectories.

Similar to random positional loss, the parameter l_s (segment loss) is used to quantify the proportion of randomly lost position points within the entire trajectory, with the lost positions being completely random. Figure 5f illustrates the occurrence of consecutive segment loss when $l_s = 2$ and $n_s = 2$. The two segments of yellow position points demonstrate the consecutive loss of two trajectory segments.

6) Combined augmentations

In addition to the five individual enhancement methods mentioned above, CLAIS innovatively incorporates combined enhancement schemes. These schemes combine the five enhancement methods based on their realism, feasibility, and the purpose and significance of the enhancement operations, resulting in six combined enhancement methods. Figure 6 illustrates the visual effects of these six combined enhancement methods, which are pairwise combinations of two noise (or distortion) operations and three position loss operations: ① random positional noise + random position loss (Figure 6a); ② random positional noise + random segment loss (Figure 6b); ③ random positional noise + regular downsampling (Figure 6c); ④ random segment distortion + random position loss (Figure 6d); ⑤ random segment distortion + random segment loss (Figure 6e); ⑥ random segment distortion + regular downsampling (Figure 6f). The parameters used in the enhancement methods in Figure 6 are consistent with their respective methods in Figure 5.

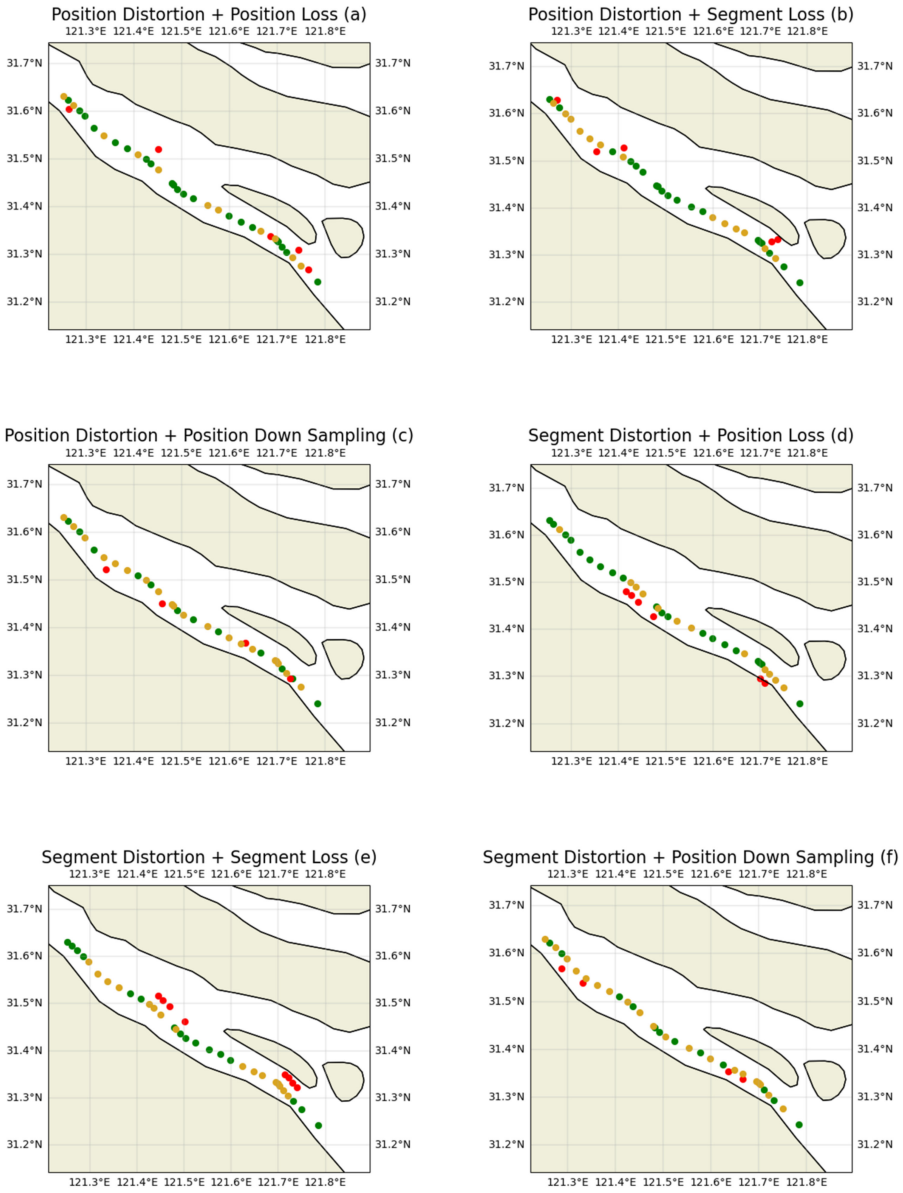

Figure 6. Combined trajectory augmentation: (**a**) position noise + position loss; (**b**) position noise + segment loss; (**c**) position noise + downsampling; (**d**) segment distortion + position loss; (**e**) segment distortion + segment loss; (**f**) segment distortion + downsampling. Green, red and yellow dots represent normal positions, error positions and missing normal positions respectively.

(2) Contrastive learning training component

CLAIS utilizes contrastive loss functions proposed in previous studies [21,29,30]. For a randomly selected mini-batch of N samples, each sample undergoes augmentation operations to generate two positive samples that are similar trajectories. A contrastive prediction task is defined, and the loss function for positive sample pairs is as follows:

$$\ell_{i,j} = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(s_{i,k}/\tau)}. \tag{3}$$

Here, $1_{[k \neq i]}$ represents the indicator function when $k \neq i$. This is the same as the approach described in the literature [31]. The remaining $2(N-1)$ augmented trajectories in the mini-batch are implicitly considered as negative samples.

3.3. Vessel Trajectory Representation Learning Module

(1) Trajectory graph construction component

The construction of the trajectory map is an essential part that explicitly transforms the input trajectories into a graph data structure with spatial dependencies. For ship trajectory coordinates, each real trajectory point is mapped to a divided grid, generating a spatial grid sequence of the same length as the trajectory's position points. Next, the grid sequence is traversed to remove consecutive repeated grids. Once all the grids representing the trajectory are obtained, the edges of the trajectory map are generated by connecting these grids.

The edges in the trajectory map connect two types of neighboring nodes: sequence neighbors and spatial neighbors. For sequence neighbors, N_{seq} neighboring grids in the sequence that are adjacent to the current grid node's front or back are added to the sequence neighbor set based on the obtained grid sequence. As for spatial neighbors, all spatial neighbors of the current grid in the region map are set as spatial neighbors. The union of the aforementioned sequence neighbor set and spatial neighbor set yields the complete neighbor set of the grid nodes. Bidirectional edges are established between each node and its neighbors, along with self-loops, creating the adjacency relationships between all the edges of the nodes.

Once the trajectory map structure is obtained, the grid embedding representation vectors generated by the region map pretraining are retrieved based on the grid indices. These vectors serve as the node features for the corresponding grids. After obtaining the pretraining embeddings for all the nodes, the trajectory map is constructed, and the next step of trajectory representation learning begins.

(2) Trajectory graph representation learning component

After the previous component enhances trajectory modeling into a trajectory graph structure, it is inputted to the trajectory graph representation learning component in this section.

The structure of the trajectory graph representation learning component in CLAIS consists of two layers of graph convolutional networks (GCN), one layer of GAT, and a two-layer bidirectional gate recurrent unit (GRU) as the readout function.

GCN is responsible for learning the relationships between grid nodes and the spatial local context information of the grids. The first graph convolution layer takes the input trajectory graph and performs information propagation and aggregation through its own node embedding and the embedding of neighboring nodes, resulting in a layer of node representations. The second graph convolution layer takes the node representations from the first layer as input and further propagates and aggregates information to obtain higher-level node representations. Multiple layers of GCN can learn more complex graph structure features. Each GCN layer is followed by a batch normalization layer [32,33] and a ReLU function that generates non-linearities [34,35]. Graph convolution can be formalized as follows:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right), \tag{4}$$

where $\tilde{A} = A + I_N$, A is the neighbor matrix of constructed trajectory graph, I_N is the identity matrix representing self-loop, and D is the degree matrix of $\tilde{A}$. $H(l)$ is the input feature matrix in the l-th layer, and $H(0) = X$, i.e., the original input is the pretrained embedding of grids. $W(l)$ is the parameter matrix in the l-th layer, and σ represents the activation function, which is ReLU in our case. Graph convolution can be seen as an effective extension of the Laplacian operator applied to the graph domain, analogous to its application on images. Further detail is in [36].

After the two GCN layers, a GAT layer is used to measure the importance weights between nodes. GAT [37] is a graph neural network that uses attention mechanisms. It can adaptively assign different weights to each neighboring node, dynamically selecting and focusing on important neighbor nodes to better capture the relationships between nodes and enhance the representation capability of graph data. GAT first performs a linear transformation on the features of each node to generate node representation vectors. Then, it learns the correlations between nodes by calculating attention weights between them. In this case, GAT also uses multiple attention heads, each generating a set of different attention weights to capture information between nodes from different perspectives. In this way, the model can selectively aggregate node features from different attention heads to better capture the relationships between nodes. The graph attention mechanism in GAT is formalized as follows:

$$a_{ij} = \frac{exp(LeakyReLU(\vec{a}^T [Wh_i || Wh_j]))}{\sum_{k \in N_i} exp(LeakyReLU(\vec{a}^T [Wh_i || Wh_k]))}, \tag{5}$$

where a_{ij} is attention weight between node i and j, h_i and h_j are representations of node i and j, which is the output of GCN layers, W is a shared parameter matrix that transforms the input features, $[\cdot||\cdot]$ concatenates transformed features, and $\vec{a}$ is a linear weight, which is realized here with a single-layer feed-forward neural network that maps the concatenated feature into a real number. Thus, the relevance of node i and j can be learned by W and $\vec{a}$. $LeakyReLU$ is a nonlinearity activation function derived empirically which, in fact, is $ReLU$ with leak. It is formalized as follows:

$$\text{LeakyReLU}(x) = \begin{cases} x, x > 0 \\ \alpha x, x \leq 0 \end{cases}. \tag{6}$$

Then, $softmax$ normalizes features into the output attention score a_{ij}. After obtaining a_{ij}, the final representation from multi-head attention is calculated as follows:

$$\vec{h}'_i = \|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathrm{W}^k \vec{h}_j \right), \tag{7}$$

where $\|$ represents concatenation.

GRU is a type of recurrent neural network used for processing sequential data [38]. Two-layer bidirectional GRU refers to using two layers of GRU and concatenating the results of forward and backward propagation. This allows for the capturing of the contextual information of nodes at different time steps and acquisition of richer node representations. The final GRU layer serves as the readout function, aggregating and extracting graph-level information from the node-level representations to generate the final representation of the trajectory graph.

4. Experiment

In this section, experiments were conducted to validate the effectiveness of the proposed CLAIS framework. The experimental design included various comparative experiments from different perspectives, aiming to verify the performance of CLAIS and observe the effects of different parameter settings on CLAIS and the comparison models.

4.1. Data & Preprocess

The experiments in this paper utilized AIS data from Shanghai Port between August and October 2022, resulting in a total of 46,079 trajectories after preprocessing. For the experiments, a subset of 31,000 trajectories was selected from the dataset. The statistical information of the dataset and the parameters of the preprocessing methods are shown in Tables 1 and 2, respectively.

Table 1. Statistical information of the dataset.

Statistic Information	Value
Longitude range	[121.167° E, 122.000° E]
Latitude range	[31.215° N, 31.632° N]
Number of recorded positions	37,406,189

Table 2. Preprocess parameter.

Parameter	Value
Maximum signal time interval	10 min
Minimum velocity	0 knots
Maximum velocity	50 knots
Minimum trajectory length	100
Maximum trajectory length	4000

4.2. Experiment Metrics

To evaluate the model's ability to measure trajectory similarity, appropriate metrics are needed in order to quantify the results generated by the model. Since CLAIS is based on unsupervised learning, there are no real labels or matching samples available to guide the model's error calculation. Inspired by previous research [27,39,40], this paper introduces a new metric called trajectory augmentation invariance based on self-similarity experiments. This metric quantifies the model's ability to recognize similar trajectories.

First, the concept of self-similarity experiments to evaluate unsupervised learning are introduced, followed by the introduction of the proposed evaluation metric, i.e., trajectory augmentation invariance.

As mentioned earlier, a well-performing vessel trajectory similarity calculation model should ideally be able to accurately distinguish different trajectory sequences sampled from the same underlying path. For this purpose, the idea behind self-similarity experiments suggests simulating two different trajectories sampled from the same underlying path by splitting a sampled trajectory into two sub-trajectories. Specifically, for the self-similarity task, given a test trajectory dataset referred to as the trajectory database set D, and an empty set Q, the following steps are performed: We randomly select q trajectories T_q from D and split the trajectories in T_q into two sub-trajectories by alternating the order of their internal position points (e.g., extracting odd-indexed points and retaining even-indexed points, or vice versa). These two sub-trajectories are considered to be twin sub-trajectories of each other. Next, the q odd-indexed sub-trajectories $T_{q \cdot odd}$ are added to the empty set Q, while the other half of q even-indexed sub-trajectories $T_{q \cdot even}$ are returned to the database set D. This process creates a database set that includes the other half of the trajectories corresponding to the set Q, which is referred to as the query set. Now the model can compute the similarity between each trajectory T_i in Q and the trajectories in D. To avoid interference from the distribution of trajectory lengths, the trajectories in D that do not belong to T_q undergo an operation of selecting every other half of the trajectory points. Since the q trajectories $T_{q \cdot even}$ in Q correspond to trajectories generated from the same underlying path in D, a prior assumption is made that these trajectories should be the closest to each other, indicating the highest similarity. Therefore, in the self-similarity experiment, the similarity rankings should place these trajectories at the top. Based on this,

the model's ability to capture trajectory similarity can be evaluated using the following three metrics:

1. Precision P represents the proportion of queries where the corresponding twin sub-trajectory is ranked first (ordinal number 0 in computer indexing). For $\forall T_i \in Q$, if the rank calculated by the model is denoted as r_i, then the precision P can be calculated as follows:

$$P = \frac{\sum_{T_i \in Q} 1_{(r_i=0)}}{|Q|}, \tag{8}$$

where $1_{(r_i=0)} = 1$ if $r_i = 0$ otherwise 0, and $|Q|$ represents element number of Q, i.e., $|Q| = q$.

2. The mean rank R_μ is the mean rank of T_i representing the twin sub-trajectory in the ranking, denoted as follows:

$$R_\mu = \frac{\sum_{T_i \in Q} r_i}{|Q|}. \tag{9}$$

3. The rank standard deviation represents the sample standard deviation of the ranks r_i. It is denoted by R_σ, and the formula is as follows:

$$R_\sigma = \sqrt{\frac{\sum_{T_i \in Q} (r_i - R_\mu)^2}{|Q| - 1}}. \tag{10}$$

R_σ is a newly proposed evaluation metric in this paper, aiming to observe the stability of the model's ability in similarity computation. If the model demonstrates good robustness in perceiving trajectory similarity in self-similarity experiments, the results of the query trajectory ranking should be relatively stable, indicating a smaller rank standard deviation R_σ.

However, the aforementioned self-similarity experiments have some limitations. Under the above-mentioned self-similarity experiment approach, whether or not augmentation operations are performed, the method only evaluates the ability to detect the twin sub-trajectories of the original trajectory before augmentation or the twin sub-trajectories of the variant trajectory after augmentation. In this case, the evaluation of the model's ability in self-similarity experiments may lead to misunderstandings, as the model might mistakenly identify variant trajectories as another trajectory. The model's ability to recognize trajectory similarity is merely based on its capability to detect twin sub-trajectories between the original and variant trajectories, i.e., without evaluating whether the model retrieves the original trajectory through the variant trajectory.

Based on the above, this paper argues that a trajectory similarity computation model should possess the ability to perceive the corresponding original trajectory even in the presence of noise and positional loss in a given trajectory, thereby identifying the ownership of the original trajectory. This requirement implies that a model with good similarity computation performance should not only be able to recognize twin sub-trajectories of the original trajectory but also the twin sub-trajectories of corresponding variant trajectories of the original, alongside its the capability to identify the original trajectory through the variant trajectories. Therefore, the self-similarity experiment metric needs to evaluate the model's ability to find both the corresponding variant twin sub-trajectory and the original twin sub-trajectory before augmentation. Thus, this paper introduces the concept of trajectory augmentation invariance: a trajectory similarity computation model should be capable of simultaneously finding the original twin sub-trajectory and the variant twin sub-trajectory before and after augmentation, respectively, for a trajectory in a given trajectory set.

Therefore, the self-similarity experiments in this paper are improved as follows. Given a trajectory database collection D, an empty augmented trajectory database collection $D^{'}$, and a query trajectory collection Q, the following steps are performed:

1. Randomly select q trajectories, T_q, from D. Divide T_q into two sub-trajectories, $T_{q \cdot odd}$ and $T_{q \cdot even}$, based on the order of their internal positional points.
2. Apply selected augmentation operations to the trajectories in T_q to create variant trajectories, $T_q^{'}$. Split $T_q^{'}$ into $T_{q \cdot odd}^{'}$ and $T_{q \cdot even}^{'}$.
3. Add $T_{q \cdot odd}^{'}$ to the empty set Q. Place $T_{q \cdot even}$ back into the database collection D and add $T_{q \cdot even}^{'}$ to $D^{'}$.
4. Subsequently, randomly downsample half of the trajectories in D that do not belong to T_q and place them back into D. Downsample the augmented trajectories and add them to $D^{'}$.

By following this procedure, the rankings of the corresponding counterpart trajectories in D and $D^{'}$ can be obtained through the query collection Q.

The aforementioned three evaluation metrics can be improved as follows:

1. Augmentation invariance precision (P) represents the proportion of queries where the corresponding trajectory is ranked highest in both D and $D^{'}$. Let us denote the rankings of $\forall T_i \in Q$ in D and $D^{'}$ as r_i and $r_i^{'}$, respectively. Then, the enhanced invariant precision can be calculated as follows:

$$P = \frac{\sum_{T_i \in Q} 1_{(r_i = 0, r_i^{'} = 0)}}{|Q|}. \tag{11}$$

2. Augmentation invariance mean rank (R_μ) represents the average rank of the corresponding twin sub-trajectory T_i in both D and $D^{'}$. It can be calculated as the average of the ranks in D and $D^{'}$ for each $T_i \in Q$. Mathematically, it can be expressed as follows:

$$R_\mu = \frac{\sum_{T_i \in Q} \frac{\left(r_i + r_i^{'}\right)}{2}}{|Q|}. \tag{12}$$

3. Augmentation invariance rank standard deviation (rank std), denoted as R_σ, represents the sample standard deviation of the average ranks $\frac{\left(r_i + r_i^{'}\right)}{2}$. It measures the variability in the average ranks of the corresponding twin sub-trajectories in D and $D^{'}$. The formula for calculating R_σ is as follows:

$$R_\sigma = \sqrt{\frac{\sum_{T_i \in Q} (\frac{(r_i + r_i^{'})}{2} - R_\mu)^2}{|Q| - 1}}. \tag{13}$$

Based on the aforementioned three metrics, we conducts enhanced invariant self-similarity experiments to evaluate the proposed CLAIS framework and compare its performance and robustness with the control models.

4.3. Comparative Baselines and Parameter Setting

To ensure the comprehensiveness and representativeness of the control models, the experiments compare five trajectory similarity calculation methods, including CLAIS. Three of these methods are classical trajectory similarity metrics or computation methods: Fréchet distance, Hausdorff distance, and dynamic time warping (DTW) distance. Additionally, a representative sequence neural network model, LSTM [41], is included. To ensure fair comparisons in the experiments, the LSTM-based method, which is also based on learning, is combined with CLAIS's spatial pretraining method, augmentation scheme, and contrastive learning method. This means that the sequential LSTM is used to replace the encoder in CLAIS, participating in the model comparison experiments. In the following experiments, CLAIS refers to the complete CLAIS framework, with the graph neural

network encoder proposed in this paper. Table 3 provides important parameter settings for the training of both learning models.

Table 3. Training parameter setting.

Model Parameter	Value	Augment Parameter	Value
Train set size	1000	position ratio	0.2
Batch size	1024	position distort	2
Pretrain epoch	10	position loss	0.2
Train epoch	200	position interval	2
Hidden size	128	segment ratio	0.2
Output size	128	segment distort	2
Grid size	0.01°	segment loss	0.2
		segment num	2

4.4. Model Comparison Experiment

To validate the effectiveness of the CLAIS framework, model comparison experiments were conducted. Using the WSK dataset, a database collection of 2 k to 10 k trajectories with a maximum length of 4000 was selected. The query trajectory collection consisted of 100 trajectories. The performance of different models was tested under different database sizes, using the metrics P, R_μ, and R_σ. The results of the experiments are shown in Table 4, with the best-performing metric indicated in bold.

Table 4. Performance comparison of different database sizes, bold indicates best performance.

	Database Size	2 k	4 k	6 k	8 k	10 k
P	Fréchet	0.16	0.14	0.12	0.10	0.10
	Hausdorff	0.12	0.08	0.07	0.06	0.06
	DTW	0.44	0.41	0.40	0.39	0.39
	LSTM Encoder	**0.79**	0.67	0.66	**0.66**	**0.64**
	CLAIS	0.78	**0.72**	**0.68**	0.62	0.62
R_μ	Fréchet	35.43	70.32	105.85	141.67	176.04
	Hausdorff	52.91	104.19	156.00	208.20	258.84
	DTW	11.21	21.76	33.54	44.57	55.46
	LSTM Encoder	0.81	1.71	2.39	3.11	3.79
	CLAIS	**0.40**	**0.81**	**1.20**	**1.60**	**1.89**
R_σ	Fréchet	51.73	103.01	154.11	204.48	251.07
	Hausdorff	77.06	152.45	227.31	302.33	371.82
	DTW	18.72	35.33	54.13	71.9	89.71
	LSTM Encoder	3.41	6.758	9.38	12.17	14.45
	CLAIS	**1.85**	**3.91**	**5.55**	**7.421**	**8.695**

From Table 4, it can be observed that among the three traditional methods, DTW, as the most effective and commonly used distance measure, achieves the best results in terms of precision, mean rank, and rank standard deviation, outperforming the other two methods. However, both learning methods based on the CLAIS framework significantly outperform the traditional methods represented by DTW. In terms of precision, both the LSTM encoder and CLAIS achieve their best results at different database sizes, demonstrating the good performance of the CLAIS framework with both types of encoders. In terms of mean rank, CLAIS consistently outperforms the LSTM encoder, maintaining an error of less than two ranks under the enhanced invariant experimental conditions even at the 10 k data level. Examining the standard deviation further shows that the differences in ranks are minimal, indicating that the CLAIS model remains stable when dealing with a large amount of error in vessel trajectories at the 10 k data level.

4.5. Robustness Experiment

The robustness of the models in the face of parameter variations plays a crucial role in the comprehensive evaluation of model performance. In this experiment, the robustness of the models in measuring trajectory similarity when facing trajectory errors was observed by fixing other parameters and varying a single parameter in the setting where the query set size was 100 and the database size was 1 k. The fixed parameter values were the same as the enhancement parameters in Table 3. Due to space limitations, the variations of five parameters are presented.

Figure 7 illustrates the variations in different trajectory similarity calculation methods with respect to the pos_ratio parameter, which controls the proportion of random position noise in the trajectories. In terms of precision, the traditional calculation methods generally maintain a relatively stable performance as the proportion of noise positions increases. On the other hand, the learning-based methods exhibit a relatively larger performance decline, but still outperform the non-learning methods. Regarding the mean rank metric, the traditional calculation methods also maintain a relatively stable level. The method based on the LSTM encoder experiences a significant performance decline after a pos_ratio value of 0.5, while the method based on the GNN encoder shows a slight increase but still maintains the best performance among all methods.

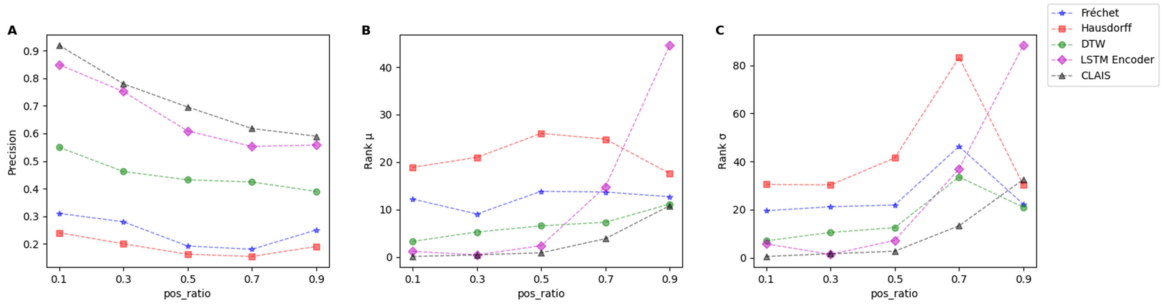

Figure 7. Variation of model performance w.r.t. pos_ratio (**A**–**C**) are P, R_μ, and R_σ w.r.t pos_ratio respecticely.

Figure 8 demonstrates the variations of the models with increasing pos_distort. The parameter pos_distort controls the magnitude of random position noise in the points. It can be observed that both methods based on CLAIS exhibit high robustness and are minimally affected by large variations in pos_distort. On the other hand, traditional methods may experience significant performance loss due to their own definitions and other factors. For example, the definition of the Hausdorff distance as the maximum of the minimum distances between sets may not remain stable when the pos_distort increases.

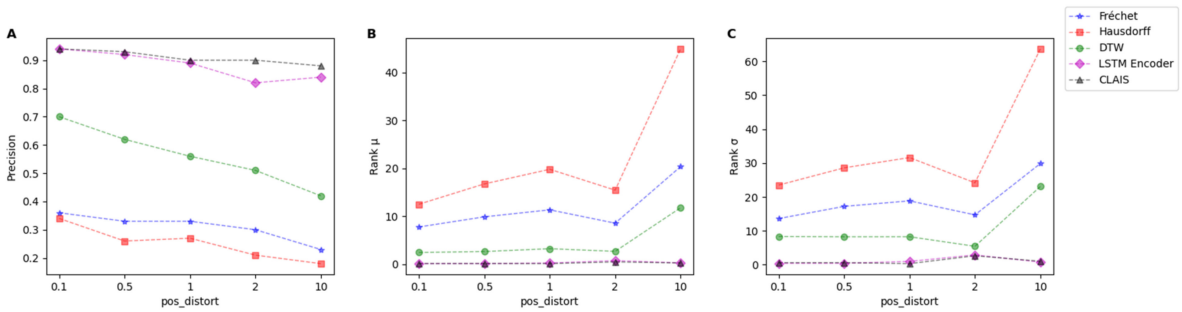

Figure 8. Variation of model performance w.r.t. pos_distort. (**A**–**C**) are P, R_μ, and R_σ w.r.t pos_distort respecticely.

Parameter pos_loss controls the proportion of randomly lost positions. In Figure 9, the traditional methods maintain a relatively stable performance, and there is even a phenomenon where the average rank decreases as the loss increases. The analysis suggests

that this may be due to the loss of excessive noisy positions, resulting in the measurement of only the truly valid trajectory points. As a result, the performance of finding similar trajectories slightly improves.

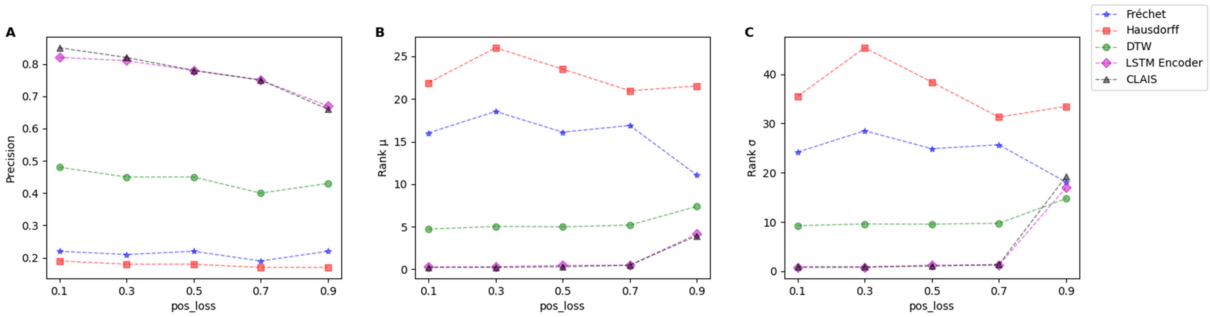

Figure 9. Variation of model performance w.r.t. pos_loss. (**A**–**C**) are P, R_μ, and R_σ w.r.t pos_loss respecticely.

The parameter seg_ratio controls the proportion of position points involved in segment distortion. As shown in Figure 10, it can be observed that the LSTM experiences a significant performance decline similar to pos_ratio when a large amount of distortion occurs. In contrast, CLAIS still maintains the best robustness.

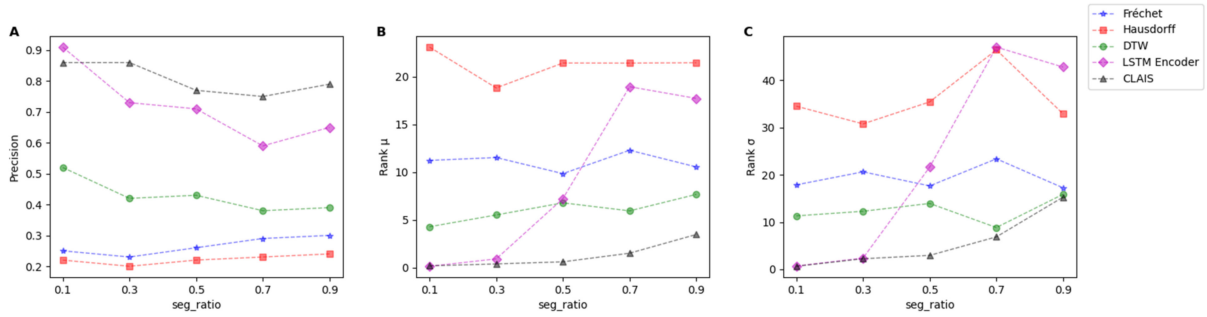

Figure 10. Variation of model performance w.r.t. seg_ratio. (**A**–**C**) are P, R_μ, and R_σ w.r.t seg_ratio respecticely.

The parameter seg_distort controls the proportion of position points involved in segment distortion. As shown in Figure 11, it can be observed that the model performance exhibits a similar trend to pos_distort, maintaining overall stability as the seg_distort parameter varies.

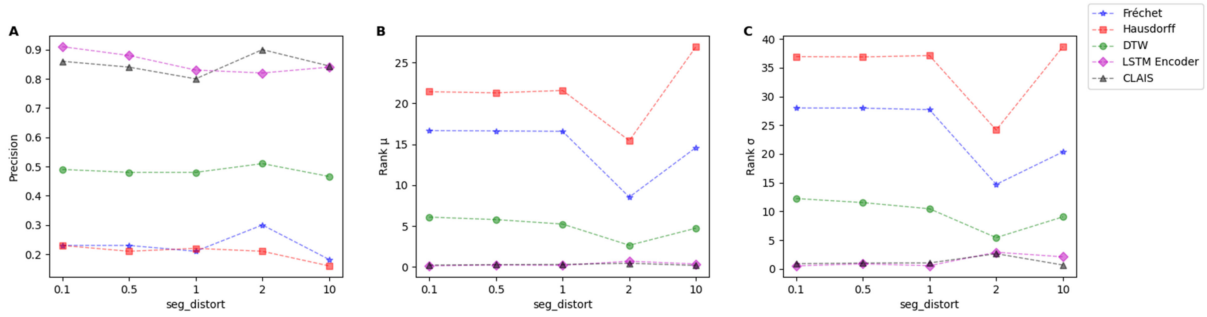

Figure 11. Variation of model performance w.r.t. seg_distort. (**A**–**C**) are P, R_μ, and R_σ w.r.t seg_distort respecticely.

4.6. Grid Size Experiment

The size of the grid is crucial to CLAIS, as it is a common challenge in grid-based approaches. For a given waterway, the width of the channel affects the density of naviga-

tion, which, in turn influences the extent to which the grid granularity can preserve the geometric features of the trajectories. Finer grids can represent more detailed trajectory shapes so that the proposed model can identify different underlying paths. However, due to computational costs, the grid size cannot be infinitely reduced. Therefore, different waterways theoretically have different optimal grid sizes to balance computational complexity and performance. To illustrate how the grid size affects the ability of the proposed model to identify different trajectories, we conducted experiments on augmentation invariance self-similarity under various grid sizes. The experiment took the grid size as a variable, query size and database size were fixed to 1 k and 10 k, respectively, and the remaining model parameters were kept the same as those in Table 3. The average results of metrics for five runs of experiments can be seen in Table 5.

Table 5. Experiment on grid size.

Grid Size	P	R_μ	R_σ
0.0010°	0.7286	2.2438	34.5396
0.0015°	0.6976	1.4097	7.6423
0.0020°	0.6736	2.2986	19.8756
0.0025°	0.6258	3.1892	16.2722
0.0030°	0.6060	3.7766	30.2616
0.0040°	0.5246	5.2674	25.2150
0.0050°	0.4070	8.4100	22.8003

From Table 5, it can be observed that when the grid size is smaller than 0.002, although there is a slight decrease in top-1 precision, it remains relatively stable. The proposed model is able to accurately identify the underlying real trajectories corresponding to the variant trajectories; even when the grid size increases to 0.003, the top-1 precision is still above 60%. Furthermore, by examining the average ranking, it can be noted that this metric performs even better than the top-1 precision. Within the range of under 0.003, the model consistently ranks the original trajectory among the top 4 out of 10,000 trajectories when searching for 1000 variant trajectories on average. However, when the grid size increases to 0.005, the top-1 precision drops to 40.7%, and the average ranking decreases to 8.41. Additionally, the standard deviation of the average ranking shows an increasing trend. This indicates that the grid size parameter indeed has an impact on the model, but despite this influence, the model can still maintain good performance within an appropriate trajectory size range. Moreover, depending on the specific requirements for application, a trade-off can be made between model performance and computational complexity.

4.7. Visualization

In the previous experiments, the performance of different comparison models was quantitatively evaluated, but it is difficult to intuitively understand the differences between various distances. In this section, visual experiments were conducted to observe whether the results of ship trajectory similarity calculation using CLAIS align with intuition, i.e., if visually similar trajectories exhibit a higher spatial proximity. The experiment randomly selected 500 trajectories from the dataset and chose two different target trajectories as query trajectories. For all comparison methods, pairwise similarity scores were computed between the query trajectories and the remaining trajectories (excluding themselves) in the dataset. The top 1% (i.e., the top 5) most similar trajectories were visualized. The visualization results, as shown in Figure 12, depict the query trajectory in red and the top-k nearest trajectories in blue.

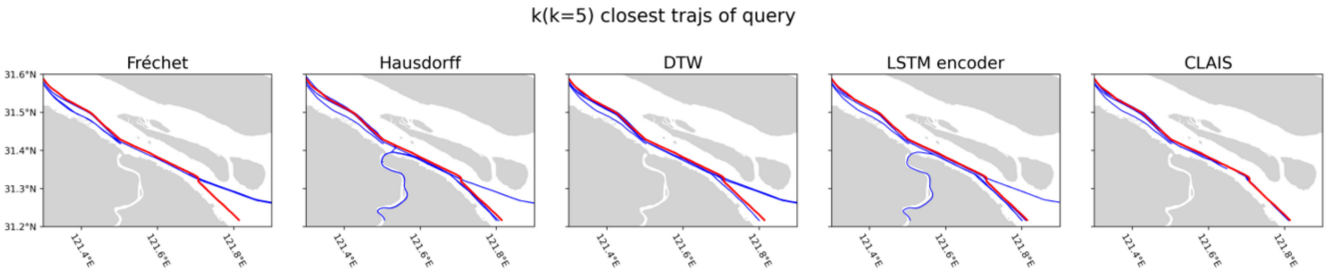

Figure 12. Top five similar trajectories of query trajectory.

From the figure, it can be observed that due to the dataset being located in a harbor area, the trajectories are very close to each other, posing a significant challenge for similarity measurement. All methods obtained trajectory sets with tight spatial distances. Among the traditional methods, DTW achieved the best results, identifying the characteristics of trajectories such as the entrance to the harbor area and turning points at the mouth of the Huangpu River. The learning-based method based on the LSTM encoder successfully recognized the trajectories in the harbor area but had misjudgments at the turning point of the Huangpu River. On the other hand, the proposed CLAIS demonstrated the strongest ability to identify similar trajectories and successfully recognized the entrance direction and turning points, visually demonstrating the effectiveness of the proposed framework.

4.8. Discussion

In this section, three important experiments were conducted. The first experiment, the model comparison experiment, tested the ability of different models to retrieve the closest trajectories by calculating AIS trajectory similarity under different database sizes. The second robustness experiment examined the performance degradation of each comparison model under different variations. The third nearest neighbor experiment visualized the distribution of the closest trajectories obtained by different models.

In the robustness experiment, the pos_ratio parameter controlled one aspect. In terms of accuracy, both learning-based methods experienced a certain degree of decline when the random position noise increased significantly. The analysis suggests that these methods adopt an average pooling approach to obtain the final representation vector, making it difficult to maintain stable feature vectors in the presence of a high proportion of position noise. In terms of average rank, the significant performance drop of the LSTM encoder may be attributed to its modeling of the trajectory within a grid while neglecting the spatial features. As the positions with noise increase, the LSTM tends to misidentify positions that fall into other grids as belonging to other trajectories, leading to a substantial decrease in identification accuracy. On the other hand, CLAIS, based on the GNN encoder, demonstrates the advantage of incorporating spatial modeling. By performing message passing on neighboring grids' features, it can identify erroneous positions within a close range and recognize them as part of its own trajectory, exhibiting relatively better performance.

It is worth noting that observing the changes in pos_ratio and seg_ratio reveals similar patterns. A performance increase exceeding the expected trend occurs when both distortion values are set to 2. This can be attributed to setting the distortion value to 2 during the training phase, which enables the model to fit better to the scenario with a distortion value of 2 compared to other parameters during inference.

5. Conclusions

Similarity calculation between vessel trajectories is one of the fundamental and crucial problems in vessel trajectory analysis. It directly impacts applications such as vessel trajectory prediction, route planning, and anomaly detection. Therefore, it holds great significance in domains such as maritime safety and maritime regulation, where vessel trajectory analysis plays a vital role. There is still significant room for improvement in the trajectory representation learning capability of existing ship trajectory similarity calculation

methods, especially in terms of explicitly modeling the spatial structure of trajectories. Additionally, with the accumulation of AIS data, a large amount of unlabeled data remains underutilized, resulting in significant waste. There is an urgent need for a method in AIS trajectory similarity calculation that possesses good ship trajectory representation learning capability and similarity calculation under unsupervised conditions.

This work makes contributions in the following three respects. First, to achieve good performance in region and trajectory learning models, this paper proposes utilizing a historical trajectory database to model water areas as regional graphs and leverages pretraining to learn the spatial dependency relationships of the regional grid. By modeling regional graphs and pretraining, the model has already learned spatial features such as the habitual routes and adjacency relationships of the spatial grid before modeling the trajectory graph. This approach reduces training costs and achieves better performance. Second, the proposed CLAIS framework in this paper enhances the model's ability to capture the potential real trajectories of trajectories under erroneous AIS signals, such as noise or position loss, through a contrastive learning approach. By employing different augmentation techniques on trajectories, the model generates similar augmented trajectory sample pairs using the proposed parameterized augmentation scheme. These augmented trajectories are then modeled as trajectory graphs, and graph neural networks are used to learn the spatial dependency relationships of the trajectory graphs and improve the model's capability to extract the spatial features of trajectories.

Lastly, this paper proposes three metrics for trajectory similarity experiments under contrastive learning and conducts extensive experiments to validate the model's performance. The experiments are conducted five times with different random seeds, and the average results demonstrate the effectiveness of CLAIS. When the database size is 10 k, CLAIS achieves a high level of accuracy in the most similar trajectory accuracy metric with a value of 62%, an average rank of 1.89, and a low standard deviation of only 8.695. In comparison, the best-performing traditional method with the DTW metric achieves values of 39%, 55.46, and 89.71, respectively. Even replacing the CLAIS encoder with an LSTM that only models sequential features yields performance of only 64%, 3.79, and 14.45 for the corresponding metrics.

Although this paper effectively contributes to the research on AIS trajectory similarity computation, there are still aspects that require further investigation. First, this paper only considers the geometric features, directional features, and sequential features of vessel trajectories, while neglecting important information such as speed, heading, timestamps, and seasons, as well as vessel type and tonnage. This may not fully reflect real-world scenarios and can have a significant impact on trajectory analysis. Second, the contrastive learning method in this paper compares trajectory graphs constructed based on the grid features of water areas, overlooking trajectory-level features and water area-level features. It is worth considering future research that incorporates joint contrastive learning for features at different levels to enhance representation learning capabilities. Furthermore, the augmentation scheme in this paper does not take into account the specific AIS signal error conditions in a particular real-world water area. It will be necessary to conduct research in the future that adapts the augmentation scheme parameters based on statistical information about signal errors, making the model more suitable for handling specific erroneous signals in the studied water area.

Author Contributions: Methodology, S.L.; Software, S.L.; Validation, S.L. and B.S.; Writing—original draft, S.L.; Writing—review & editing, W.Z. and B.S.; Supervision, W.Z.; Funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this research is provided by hifleet.com.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shelmerdine, R.L. Teasing out the Detail: How Our Understanding of Marine AIS Data Can Better Inform Industries, Developments, and Planning. *Mar. Policy* **2015**, *54*, 17–25. [CrossRef]
2. Tao, Y.; Both, A.; Silveira, R.I.; Buchin, K.; Sijben, S.; Purves, R.S.; Laube, P.; Peng, D.; Toohey, K.; Duckham, M. A Comparative Analysis of Trajectory Similarity Measures. *GISci. Remote Sens.* **2021**, *58*, 643–669. [CrossRef]
3. Zhao, L.; Shi, G. A Novel Similarity Measure for Clustering Vessel Trajectories Based on Dynamic Time Warping. *J. Navig.* **2019**, *72*, 290–306. [CrossRef]
4. Zhao, L.; Shi, G. Maritime Anomaly Detection Using Density-Based Clustering and Recurrent Neural Network. *J. Navig.* **2019**, *72*, 894–916. [CrossRef]
5. Sang, L.; Wall, A.; Mao, Z.; Yan, X.; Wang, J. A Novel Method for Restoring the Trajectory of the Inland Waterway Ship by Using AIS Data. *Ocean Eng.* **2015**, *110*, 183–194. [CrossRef]
6. Zhao, L.; Shi, G.; Yang, J. Ship Trajectories Pre-Processing Based on AIS Data. *J. Navig.* **2018**, *71*, 1210–1230. [CrossRef]
7. Yan, R.; Mo, H.; Yang, D.; Wang, S. Development of Denoising and Compression Algorithms for AIS-Based Vessel Trajectories. *Ocean Eng.* **2022**, *252*, 111207. [CrossRef]
8. Lee, W.; Cho, S.-W. AIS Trajectories Simplification Algorithm Considering Topographic Information. *Sensors* **2022**, *22*, 7036. [CrossRef]
9. Yang, P.; Wang, H.; Zhang, Y.; Qin, L.; Zhang, W.; Lin, X. T3S: Effective Representation Learning for Trajectory Similarity Computation. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 2183–2188.
10. Yang, P.; Wang, H.; Lian, D.; Zhang, Y.; Qin, L.; Zhang, W. TMN: Trajectory Matching Networks for Predicting Similarity. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia, 9–12 May 2022; pp. 1700–1713.
11. Zhang, H.; Zhang, X.; Jiang, Q.; Zheng, B.; Sun, Z.; Sun, W.; Wang, C. Trajectory Similarity Learning with Auxiliary Supervision and Optimal Matching. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan, 7–15 January 2021; pp. 3209–3215.
12. Yao, D.; Hu, H.; Du, L.; Cong, G.; Han, S.; Bi, J. TrajGAT: A Graph-Based Long-Term Dependency Modeling Approach for Trajectory Similarity Computation. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; ACM: New York, NY, USA, 2022; pp. 2275–2285.
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Association for Computing Machinery: New York, NY, USA, 2017; Volume 30.
14. Yao, D.; Zhang, C.; Zhu, Z.; Hu, Q.; Wang, Z.; Huang, J.; Bi, J. Learning Deep Representation for Trajectory Clustering. *Expert Syst.* **2018**, *35*, e12252. [CrossRef]
15. Li, S.; Liang, M.; Liu, R.W. Vessel Trajectory Similarity Measure Based on Deep Convolutional Autoencoder. In Proceedings of the 2020 5th IEEE International Conference on Big Data Analytics (ICBDA), Xiamen, China, 8–11 May 2020; pp. 333–338.
16. Fu, T.-Y.; Lee, W.-C. Trembr: Exploring Road Networks for Trajectory Representation Learning. *ACM Trans. Intell. Syst. Technol.* **2020**, *11*, 1–25. [CrossRef]
17. Balestriero, R.; Ibrahim, M.; Sobal, V.; Morcos, A.; Shekhar, S.; Goldstein, T.; Bordes, F.; Bardes, A.; Mialon, G.; Tian, Y.; et al. A Cookbook of Self-Supervised Learning. *arXiv* **2023**, arXiv:2304.12210.
18. Gui, J.; Chen, T.; Zhang, J.; Cao, Q.; Sun, Z.; Luo, H.; Tao, D. A Survey of Self-Supervised Learning from Multiple Perspectives: Algorithms, Applications and Future Trends. *arXiv* **2023**, arXiv:2301.05712.
19. Chen, X.; Xu, J.; Zhou, R.; Chen, W.; Fang, J.; Liu, C. TrajVAE: A Variational AutoEncoder Model for Trajectory Generation. *Neurocomputing* **2021**, *428*, 332–339. [CrossRef]
20. Miguel, M.Á.D.; Armingol, J.M.; García, F. Vehicles Trajectory Prediction Using Recurrent VAE Network. *IEEE Access* **2022**, *10*, 32742–32749. [CrossRef]
21. Wu, Z.; Xiong, Y.; Yu, S.X.; Lin, D. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 19–23 June 2018; pp. 3733–3742.
22. Chen, T.; Kornblith, S.; Swersky, K.; Norouzi, M.; Hinton, G. Big Self-Supervised Models Are Strong Semi-Supervised Learners. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020; Curran Associates Inc.: Red Hook, NY, USA, 2020; pp. 22243–22255.
23. Liu, X.; Tan, X.; Guo, Y.; Chen, Y.; Zhang, Z. CSTRM: Contrastive Self-Supervised Trajectory Representation Model for Trajectory Similarity Computation. *Comput. Commun.* **2022**, *185*, 159–167. [CrossRef]
24. Jing, Q.; Yao, D.; Gong, C.; Fan, X.; Wang, B.; Tan, H.; Bi, J. TrajCross: Trajecotry Cross-Modal Retrieval with Contrastive Learning. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 344–349.

25. Grover, A.; Leskovec, J. Node2vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 855–864.
26. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the 37th International Conference on Machine Learning, Online, 13–18 July 2020; Volume 119, pp. 1597–1607.
27. Li, X.; Zhao, K.; Cong, G.; Jensen, C.S.; Wei, W. Deep Representation Learning for Trajectory Similarity Computation. In Proceedings of the 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, 16–19 April 2018; pp. 617–628.
28. Deng, L.; Zhao, Y.; Fu, Z.; Sun, H.; Liu, S.; Zheng, K. Efficient Trajectory Similarity Computation with Contrastive Learning. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, 17–21 October 2022; ACM: New York, NY, USA, 2022; pp. 365–374.
29. Sohn, K. Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 1857–1865.
30. Van den Oord, A.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. *arXiv* **2018**, arXiv:1807.03748. [CrossRef]
31. Chen, T.; Sun, Y.; Shi, Y.; Hong, L. On Sampling Strategies for Neural Network-Based Collaborative Filtering. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 767–776.
32. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 448–456.
33. Santurkar, S.; Tsipras, D.; Ilyas, A.; Mądry, A. How Does Batch Normalization Help Optimization? In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 2488–2498.
34. Hahnloser, R.H.R.; Sarpeshkar, R.; Mahowald, M.A.; Douglas, R.J.; Seung, H.S. Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit. *Nature* **2000**, *405*, 947–951. [CrossRef]
35. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
36. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2017**, arXiv:1609.02907.
37. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2018**, arXiv:1710.10903.
38. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734.
39. Ranu, S.; Deepak, P.; Telang, A.D.; Deshpande, P.; Raghavan, S. Indexing and Matching Trajectories under Inconsistent Sampling Rates. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; pp. 999–1010.
40. Su, H.; Zheng, K.; Wang, H.; Huang, J.; Zhou, X. Calibrating Trajectory Data for Similarity-Based Analysis. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22–27 June 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 833–844.
41. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]