Prediction of Marine Dynamic Environment for Arctic Ice-Based Buoys Using Historical Profile Data

Jingzi Zhu , Yu Luo , Tao Li , Yanhai Gan * , Junyu Dong *

Posted Date: 22 April 2025

doi: 10.20944/preprints202504.1854.v1

Keywords: ocean buoy; profile; vortex prediction; time series model; TSMixer

 doi:10.20944/preprints202504.1854.v1

Article

Prediction of Marine Dynamic Environment for Arctic Ice-Based Buoys Using Historical Profile Data

Jingzi Zhu [1], Yu Luo [2], Tao Li [3], Yanhai Gan [3,*], Junyu Dong [3,*]

[1] Haide College , Ocean University of China, 238 Songling Road, Qingdao 266100, China
[2] School of Mathematical Sciences, Ocean University of China, 238 Songling Road, Qingdao 266100, China
[3] Faculty of Information Science and Engineering, Ocean University of China, 238 Songling Road, Qingdao 266100, China
* Corresponding authors: ganyanhai@ouc.edu.cn (Y.G.); dongjunyu@ouc.edu.cn (J.D.)

Abstract: In this paper, the time series model is used to predict whether an ocean buoy is about to be inside the vortex. Marine buoys are an important tool for collecting ocean data and studying ocean dynamics, climate change, and ecosystem health. Vortex is an important ocean dynamic process. If we can predict that the buoy is about to enter the vortex, we can automatically adjust the buoy's sampling frequency to better observe the vortex's structure and development. To address this requirement, based on the profile data including latitude and longitude, temperature, and salinity collected by 56 buoys in the Arctic Ocean from 2014 to 2023, this paper uses the TSMixer time series model to predict whether an ocean buoy is about to be inside the vortex. The TSMixer model effectively captures the spatio-temporal characteristics of multivariate time series through time-mixing and feature-mixing mechanisms, and the accuracy of the model reaches 84.6%. The proposed model is computationally efficient and has a low memory footprint, which is suitable for real-time applications and provides accurate prediction support for marine monitoring. The code is available at: https://github.com/qimingfan10/Buoy_prediction.git.

Keywords: ocean buoy; profile; vortex prediction; time series model; TSMixer

1. Introduction

The ocean buoy is an important observation tool that is anchored at sea and can continuously monitor hydrological, water quality, and meteorological elements around the world, providing key data support for marine scientific research, resource development, and national defence construction [1]. The study of ocean buoy and vortex is crucial for understanding the circulation mechanism, predicting marine disasters, ensuring shipping safety, and maintaining ecosystems; its results directly address the critical needs of climate change response, ecological protection, and sustainable use of resources [2].

The vortex environment significantly compromises buoy data acquisition. The complex flow field leads to the circularity of the buoy's trajectory, which causes data discontinuity, sampling anomalies, or even loss and makes the buoy deviate from the predetermined path [3]. The ability to predict whether an ocean buoy is about to be inside the vortex is valuable to improve monitoring efficiency by optimising deployment strategies (e.g., adjusting the placement and timing of the buoy). Compared with the vortex data that needs to be corrected by ALIS and other algorithms, the buoy in the non-vortex area keeps a straight line of motion, and the data is more continuous, reliable, and easy to process [4].

Vortex prediction technology can greatly enhance how we monitor the ocean by helping us use resources better, focusing on important areas, shielding sensors from quick changes in water flow, aiding research on vortex behavior (like size and strength), and assisting in evaluating environmental effects [5]. At the same time, the optimal deployment can reduce the frequency of equipment delivery and recovery and reduce the cost. This technology is critical to improve the monitoring accuracy, ensure equipment safety, and optimize resource allocation, and it is an effective solution to solve the problem of vortex data acquisition [6].

 doi:10.20944/preprints202504.1854.v1

Marine scientific research frequently uses deep learning models. Due to their powerful nonlinear modeling ability and automatic feature extraction advantages, they can effectively deal with complex spatiotemporal data in the marine environment. Deep learning technology has shown significant application potential in seawater temperature prediction [7], buoy trajectory prediction [8] and other fields.

Currently, no technology exists to predict whether the buoy is inside the vortex. Although existing sensors are able to collect certain data on the position of the buoy as well as general environmental conditions, they lack the ability to accurately predict the state the buoy is in. Moreover, current algorithms developed to analyze such data are unable to account for the numerous interacting factors that determine whether a buoy is inside a vortex. The time-series model excels in processing data from ocean buoys because it can accurately track changes over time (like temperature and salinity at nearby depths) and how different factors work together (like the relationship between latitude, longitude, temperature, and salinity), while traditional statistical methods struggle with these complicated connections.

The time series model is a key method to deal with time series or sequential data, and its development has experienced the evolution from early statistical models to recurrent neural networks, such as RNN [9], LSTM [10], GRU [11], etc. The recurrent neural network (RNN) deals with the variable-length sequence through the recurrent structure, but it is limited by the problem of gradient disappearance [9]. Its improved versions, LSTM [10] and GRU [11], introduce a gating mechanism to achieve long-term memory and efficient training, respectively, but have high computational complexity. The Temporal Convolutional Network (TCN) uses causal dilated convolution to realize parallel computing, but its long-range dependence modeling ability is limited [12]. In 2017, Transformer broke through the limit of sequence length through the self-attention mechanism but was faced with the problem of O(n^2) computational complexity [13]. In recent years, DLinear has outperformed complex models on specific tasks by linear decomposition [14], and the latest TSMixer achieves transformer-level performance with a pure MLP architecture and 5 times faster inference [15].

Therefore, based on TSMixer [15], we establish the time series model to predict whether an ocean buoy is about to be inside the vortex. In the comparison experiment, by comparing LSTM [10], GRU [11], RNN [9], Transformer [13], TCN [12], TSMixer [15], and Dlinear [14] neural networks, the data collected by multiple buoys in the Arctic Ocean, including latitude and longitude, depth, temperature, salinity, and other data, is used for deep learning, and TSMixer is proved to be the best model. With the ability to make high-precision predictions, long-term feature capture, and multi-source data fusion as shown in Figure 1, the model provides reliable technical support for marine scientific research and resource management.

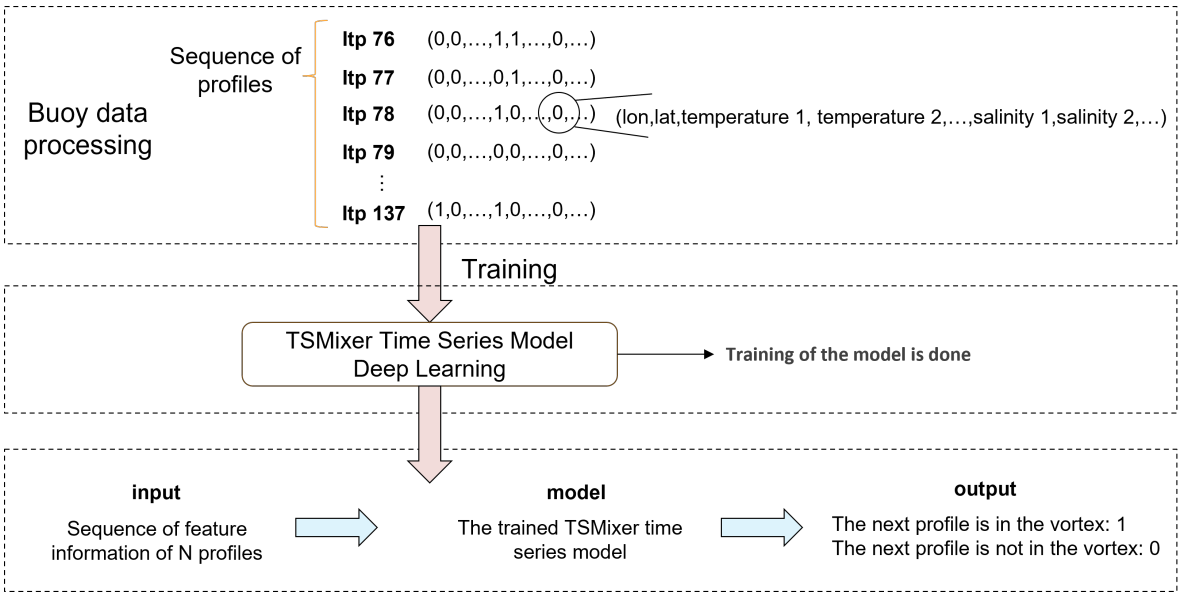


Figure 1. Flow chart of the model.

 doi:10.20944/preprints202504.1854.v1

2. Dataset

2.1. Data Introduction

Between 2014 and 2023, 56 ice-based towed profilers (ITP) deployed by the Woods Hole Oceanographic Institution (WHOI) in the Arctic Ocean collected hundreds to thousands of profile data by floating up and down as shown in Figure 2, forming a data set containing temperature and salinity corresponding to different depths (Table 1). The ITP system is composed of an ice-floating body, a coupling cable, and an underwater profiler. GPR and Iridium satellite antennas equip the ice-floating body, enabling real-time data transmission. The coupled cable and electromagnetic induction module transmit the depth data of 7-700 meters collected by the profiler (equipped with Seabird 41/41CP temperature and salt depth meter, accuracy of 0.002°C , 0.0035, and 2 dbar) to the ice control unit and acquire 1-2 profiles per day. The three-level data after quality control and 1-meter vertical resolution interpolation cover the Arctic Ocean basin from 2005 to 2024, which provides high-precision data for long-term observation of upper and middle ocean temperature and salinity and supports subsequent deep learning research [16].

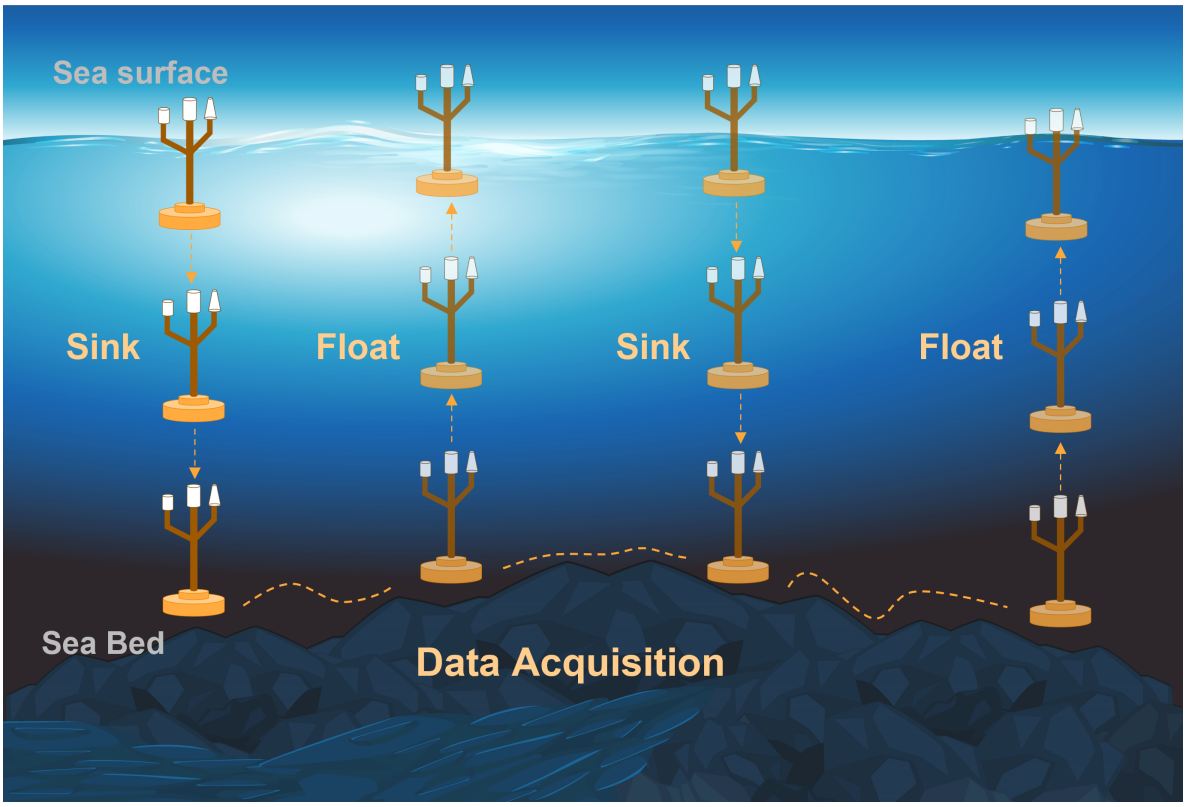


Figure 2. Buoy acquisition data diagram.

2.2. Feature Selection

While predicting whether an ocean buoy is about to be inside the vortex, selecting appropriate features is crucial for the performance of the model [17]. In this paper, we select the following features:

- **Lon**: reflects the longitude position of the trajectory
- **Lat**: reflects the latitude of the trajectory
- **Temperature**: reflects the temperature changes in the ocean environment
- **Salinity**: reflects changes in the salinity of the marine environment

 doi:10.20944/preprints202504.1854.v1

Table 1. The number of profiles for each buoy.

Buoy	Profile Number	Buoy	Profile Number	Buoy	Profile Number
itp76	910	itp93	1543	itp114	4403
itp77	2367	itp95	878	itp115	261
itp78	1691	itp97	699	itp116	529
itp79	1694	itp98	179	itp117	206
itp80	3258	itp99	224	itp120	1927
itp81	671	itp100	176	itp121	1101
itp82	1087	itp101	382	itp122	1860
itp83	937	itp102	2140	itp123	1100
itp84	172	itp103	5039	itp125	151
itp85	659	itp104	6223	itp126	941
itp86	753	itp105	6061	itp127	862
itp87	647	itp107	296	itp128	408
itp88	30	itp108	673	itp129	1294
itp89	429	itp109	169	itp130	338
itp90	305	itp110	630	itp131	253
itp91	328	itp111	520	itp136	434
itp92	1855	itp113	4842	itp137	431

2.3. Data Preprocessing

The experiment collected data from April 12, 2014, to January 13, 2023, using a total of 51 buoys. Each buoy periodically sinks and floats (Figure 2), recording the latitude, longitude, temperature, and salinity with the depth changes.

Each buoy collects multiple sets of profile data, and the experiment takes all feature data in a profile as one input. We label the five days before and after the vortex center as label 1. The remaining profiles of these buoys, which are not inside the vortex, were labeled as label 0. All profile data labeled with label 0 and label 1 for each buoy are in chronological order as shown in Figure 1.

While studying the depth of the ocean, we learned that too deep is meaningless to study whether it is in the vortex. Currents in the surface layer (within 10 meters) of the Arctic are subjected to friction by sea ice, and eddy activity is reduced or even eliminated [18]. Therefore, excluding the data below 10 meters can more clearly reflect the main characteristics and seasonal variations of the Arctic. And since the depth of the collected original data is not uniform, the data values that are too deep are eliminated, and the data with a depth of 10 meters to 200 meters are retained to form a new data set for deep learning.

2.4. Data Standardization

In each profile, the data range of latitude and longitude, temperature, and salinity is not uniform, so we should choose an appropriate preprocessing method to process the data. Convert all values to numeric types. For the missing values caused by data collection, we assign the missing values to 0, which has a small impact on the value, and eliminate the duplicate data.

For latitude and longitude, temperature, depth, and salinity, to facilitate the training of deep learning models, the z-score normalization method was used to process the data and scale the data to a range of mean 0 and standard deviation 1. This method converts the raw data to its standard normal z-value [19]:

For each feature x, the standardized value x' is calculated as:

$$x' = \frac{x - \mu}{\sigma} \tag{1}$$

where:

- μ is the mean of the feature

 doi:10.20944/preprints202504.1854.v1

- σ is the standard deviation of the feature

By eliminating the dimension difference and unifying the feature scale, the model can deal with different features equally and avoid some of them dominating the training process due to their large numerical range. At the same time, the model convergence is accelerated, the training efficiency is enhanced, the stability of the model is improved, and the interference of extreme values on the training process is reduced.

2.5. Sequence Construction

To build a dataset suitable for time series prediction, we convert the data into a time series format. We normalize all the eigenvalues of each profile to form a vector. The latitude and longitude of each profile are the same. Different depths correspond to different temperatures and salinities, and all profiles recorded data from depths of 10 m to 200 m, so depth is not used as an input feature. We will be lenient with the data and discard a vector if it has more than a third of zeros. In other words, we treat abnormal data cautiously and don't arbitrarily discard any missing values.

$$X(\text{station}) = (\text{lon}, \text{lat}, \text{temperature 1}, \text{temperature 2}, \ldots, \text{salinity 1}, \text{salinity 2}, \ldots) \tag{2}$$

The dataset is divided into: training set, validation set, and test set.

- **Training set**: 60% of the data for model training
- **Validation set**: 20% of the data for hyperparameter tuning
- **Test set**: 20% of the data for final evaluation

3. Methods

In this study, we adopted the TSMixer time series model for the prediction of whether an ocean buoy is about to be inside the vortex. TSMixer is a neural network model based on time series mixture, which can effectively capture the time dependence in the profile series and the interaction between features [15]. The following is a detailed description of the construction process of the model and its key components.

3.1. Model Architecture

TSMixer is a deep learning model based on time series data, which combines time-mixing and feature-mixing mechanisms and can effectively capture the time and feature dimension information in profile series data [15]. The overall architecture of the model consists of multiple MixerLayers, and each MixerLayer contains two main parts: Time-Mixing and Feature-Mixing as shown in Figure 3.

- **Time-Mixing**: Mix features along the temporal dimension to capture temporal dependencies in the time series
- **Feature-Mixing**: Mix the data on the feature dimension to capture the correlation between different features

Input Layer:

The input data of the model has a shape of $(batch\_size, sequence\_length, num\_features)$. The input layer processes the data through a linear transformation, specifically a fully - connected layer. Let the input vector at a single time step be $\mathbf{x}_t \in \mathbb{R}^{num\_features}$, the weight matrix be $\mathbf{W} \in \mathbb{R}^{num\_features \times num\_features}$, and the bias vector be $\mathbf{b} \in \mathbb{R}^{num\_features}$. The output $\mathbf{y}_t$ of the linear transformation at this time step can be expressed as:

$$\mathbf{y}_t = \mathbf{W}\mathbf{x}_t + \mathbf{b} \tag{3}$$

This linear transformation aligns the data dimensions with the subsequent layer requirements.

MixerLayer Stacking:

 doi:10.20944/preprints202504.1854.v1

The MixerLayer is a core component of the TSMixer model. Its function is to transform the input tensor from the shape of $(batch\_size, sequence\_length, num\_features)$ to $(batch\_size, num\_features, sequence\_length)$ to prepare for subsequent operations in the temporal dimension. Suppose the input tensor is $\mathbf{X} \in \mathbb{R}^{batch\_size \times sequence\_length \times num\_features}$. After the MixerLayer operation, by transposing the second and third dimensions of $\mathbf{X}$, the output tensor $\mathbf{X}' \in \mathbb{R}^{batch\_size \times num\_features \times sequence\_length}$ is obtained.

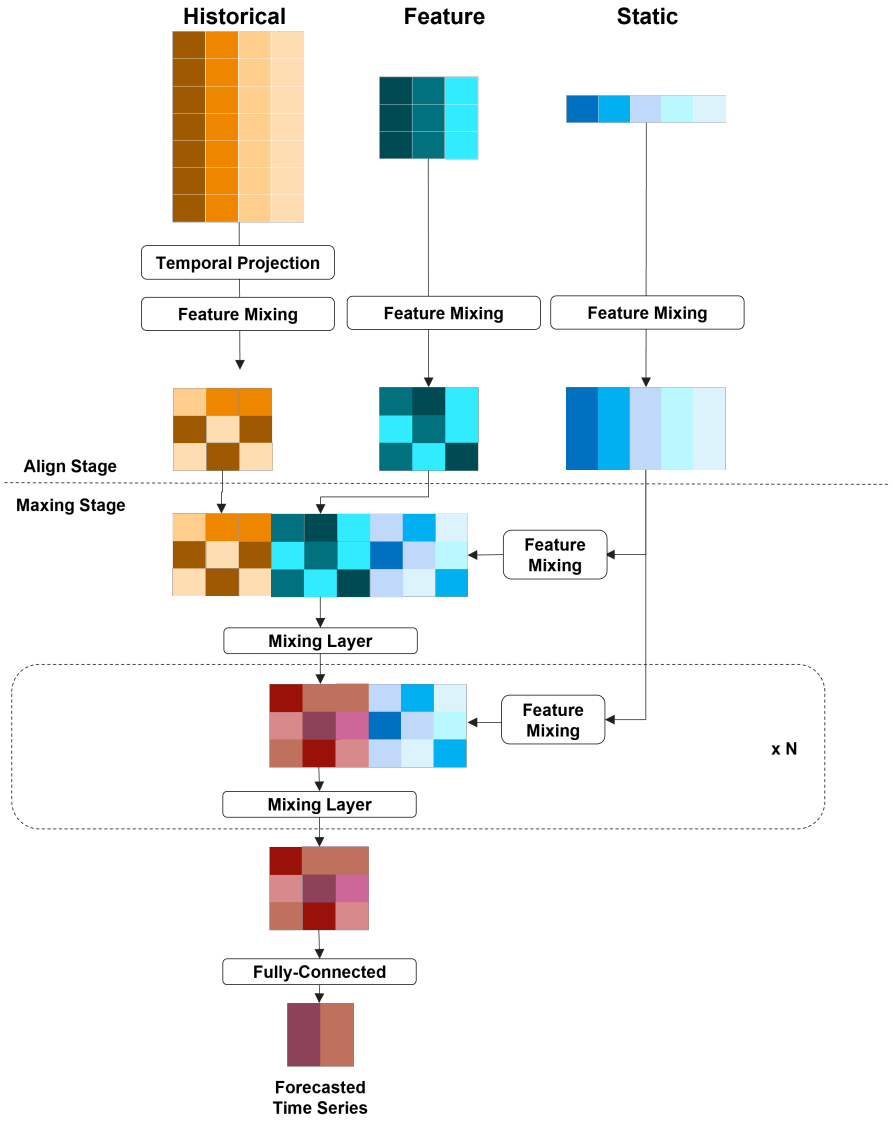


Figure 3. Tsmixer overall architecture and multi-source data fusion prediction process diagram.

Figure 4 describes how time and feature mixing of input data can be performed, including batch norm, multi-layer perceptron (MLP) application in different dimensions, and the use of residual join. Mixer Layer stacking (Mixer Layer x N) and the final Temporal Projection process are also involved, which presents the key links of the model to process time series data.

 doi:10.20944/preprints202504.1854.v1

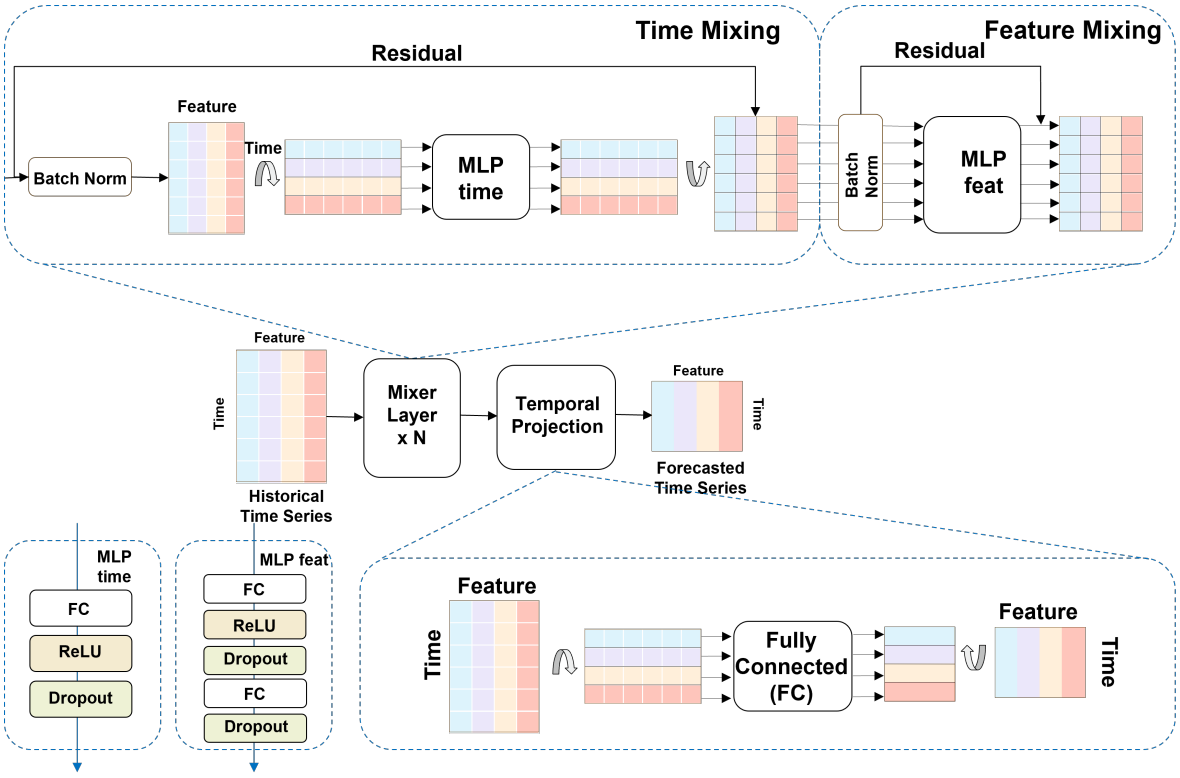


Figure 4. Schematic diagram of TSMixer single Mixer layer internal structure and processing flow.

Temporal Dimension Mixing:

For the data processed by the MixerLayer, the model uses a multi-layer perceptron (MLP) to perform mixing operations in the temporal dimension. The MLP consists of linear layers, ReLU activation functions, and Dropout layers. Assume that the input sequence of a certain batch is $\mathbf{s} = [s_1, s_2, \cdots, s_{sequence\_length}]$, where $s_i \in \mathbb{R}^{num\_features}$. In the first linear layer of the MLP, with the weight matrix $\mathbf{W}_1 \in \mathbb{R}^{hidden\_units \times sequence\_length}$ and the bias $\mathbf{b}_1 \in \mathbb{R}^{hidden\_units}$, its output $\mathbf{h}_1$ is:

$$\mathbf{h}_1 = \mathbf{W}_1 \mathbf{s} + \mathbf{b}_1 \tag{4}$$

Subsequently, the ReLU activation function $\sigma(x) = \max(0, x)$ is applied element - wise to $\mathbf{h}_1$, resulting in :

$$\mathbf{h}_2 = \sigma(\mathbf{h}_1) \tag{5}$$

If the model is equipped with a Dropout layer, some elements in $\mathbf{h}_2$ are randomly set to zero with a probability of p. Finally, after passing through another linear layer with the weight matrix $\mathbf{W}_2 \in \mathbb{R}^{sequence\_length \times hidden\_units}$ and the bias $\mathbf{b}_2 \in \mathbb{R}^{sequence\_length}$, the output result $\mathbf{o} \in \mathbb{R}^{sequence\_length}$, which has the same dimension as the input sequence (Figure 4).

Dimension Restoration:

After completing the temporal dimension mixing, the tensor needs to be restored to its original shape $(batch\_size, sequence\_length, num\_features)$. Let the tensor after temporal dimension mixing be $\mathbf{Y} \in \mathbb{R}^{batch\_size \times num\_features \times sequence\_length}$. By transposing the second and third dimensions again, $\mathbf{Y}' \in \mathbb{R}^{batch\_size \times sequence\_length \times num\_features}$ is obtained.

Residual Connection and Layer Normalization:

To ensure the stability of model training, the output $\mathbf{Y}'$ of the temporal dimension mixing and the original input $\mathbf{X}$ of the input layer are added through a residual connection, that is,

$$\mathbf{Z}_1 = \mathbf{X} + \mathbf{Y}' \tag{6}$$

 doi:10.20944/preprints202504.1854.v1

. Then, layer normalization is performed on $\mathbf{Z}_1 \in \mathbb{R}^{batch\_size \times sequence\_length \times num\_features}$. For each element in the batch, the mean μ and variance σ^2 are calculated along the $num\_features$ dimension:

$$\mu = \frac{1}{num\_features} \sum_{i=1}^{num\_features} z_{1ij} \tag{7}$$

$$\sigma^2 = \frac{1}{num\_features} \sum_{i=1}^{num\_features} (z_{1ij} - \mu)^2 \tag{8}$$

After layer normalization, the output is

$$z_{2ij} = \frac{z_{1ij} - \mu}{\sqrt{\sigma^2 + \epsilon}} \gamma + \beta \tag{9}$$

where ϵ is a small constant to prevent division by zero, and γ and β are learnable parameters.

Feature Dimension Mixing:

To capture the correlations between features, the model conducts mixing operations in the feature dimension, which is also achieved with the help of an MLP. Let the input tensor after layer normalization be $\mathbf{Z}_2 \in \mathbb{R}^{batch\_size \times sequence\_length \times num\_features}$. If necessary, it is reshaped to obtain $\mathbf{Z}_2'$. In the first linear layer of the MLP for feature - dimension mixing, with the weight matrix $\mathbf{W}_3 \in \mathbb{R}^{hidden\_units \times num\_features}$ and the bias $\mathbf{b}_3 \in \mathbb{R}^{hidden\_units}$, the output

$$\mathbf{h}_3 = \mathbf{W}_3 \mathbf{Z}_2' + \mathbf{b}_3 \tag{10}$$

After ReLU activation and Dropout operations similar to those in the MLP for temporal dimension mixing, and then passing through another linear layer with appropriate weight matrices and biases, the output has the same dimension as the input in the feature dimension.

Second Residual Connection and Layer Normalization:

The output $\mathbf{F}$ of the feature - dimension mixing and the output $\mathbf{Z}_2$ of the temporal - dimension mixing after layer normalization are added through a residual connection, that is:

$$\mathbf{Z}_3 = \mathbf{Z}_2 + \mathbf{F} \tag{11}$$

Subsequently, layer normalization is applied again. The resulting output tensor will be used for subsequent computational tasks of the model.

The algorithm pseudocode is shown in the figure.

Algorithm 1 Training of TSMixer

Initialize the model parameters θ randomly.
for $epoch = 1$ to E **do**
  Shuffle the training data D_{train}.
  for each batch of data B in D_{train} **do**
    Extract features X and labels Y from the batch B.
    Make predictions $\hat{Y} = M(X; \theta)$.
    Calculate the loss $loss = L(\hat{Y}, Y)$.
    Calculate the gradients of the loss with respect to the model parameters: $\nabla_\theta loss$.
    Update the model parameters using the optimization algorithm O and the learning rate α: $\theta \leftarrow O(\theta, \nabla_\theta loss, \alpha)$.
  end for
  Evaluate the model on a validation set (if available) to monitor performance and potentially adjust hyperparameters.
end for

 doi:10.20944/preprints202504.1854.v1

3.2. Loss Function

The loss function used is Binary Cross-Entropy Loss (BCELoss), whose formula is:

$$\text{BCELoss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)] \tag{12}$$

3.3. Evaluation Index

To comprehensively evaluate the model's performance, we employed multiple evaluation metrics including accuracy, precision, recall, F1-score, MSE, and R^2.

Accuracy measures the proportion of correctly predicted samples out of the total samples, calculated as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{13}$$

Precision and recall evaluate the model's correctness and coverage in predicting positive classes, respectively:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{14}$$

The F1-score is the harmonic mean of precision and recall, providing a comprehensive assessment of classification performance:

$$F1\text{-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{15}$$

Mean Squared Error (MSE) calculates the average squared difference between predicted and true values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{16}$$

MSE is sensitive to outliers and reflects the overall distribution of prediction errors.

The R^2 (coefficient of determination) measures the model's explanatory power for the target variable, ranging between $[0, 1]$. Its formula is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{17}$$

4. Experiments

4.1. Experimental Setup

The experiment utilizes high-performance hardware, including the multithread-capable Intel Core i7-12700K CPU and the NVIDIA GeForce RTX 3090 GPU with 24GB of VRAM, which supports large-scale deep learning training. On the software side, Python 3.9 and PyTorch 1.12.1 are employed. The dataset consists of 5847 multivariate time-series samples with geophysical features, where the temporal sequence length (seq_len) is 4 and the number of features per timestep (num_feat) is 492. For model parameters, the time-mixing layer dimension (time_dim) is set to 256, and the feature-mixing layer dimension (feature_dim) is 2048. During training, the Adam optimizer with a learning rate $\eta = 0.001$ and momentum parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ is used. It combines momentum and an adaptive learning rate mechanism for rapid convergence, and the loss function is BCELoss (Binary Cross-Entropy Loss). The evaluation metrics include classification metrics such as Acc (Accuracy), Prec (Precision), Rec (Recall), and F1, and regression metrics such as MSE (Mean Squared Error) and R^2, as shown in Table 2.

 doi:10.20944/preprints202504.1854.v1

Table 2. Model and Experimental Setup Parameters.

Category	Setting/Parameter	Value	Description
Hardware	CPU	Intel Core i7-12700K	High perf multithreaded CPU
	GPU	NVIDIA GeForce RTX 3090	24 GB VRAM, supports large-scale DL training
Software	Python Version	3.9	Multi - thread high - perf CPU
	PyTorch Version	1.12.1	Multi - thread high - perf CPU
Dataset	Samples	5847	Multivariate time series
Model Parameters	Sequence Length	n	Input time series length
	Num_Features	492	Features per timestep
	Time-Mix Dim	256	Time-Mixing MLP dim
	Feature-Mix Dim	2048	Feature-Mixing MLP dim
	Dropout Rate	0.1	Anti - overfitting regularization
	Batch Size	32	Training mini-batch size
	Epochs	30	Total training iterations
	Learning Rate	Adam	Optimizer configuration (η=0.001)
	Loss Function	BCELoss	Binary cross-entropy loss metric
	optimizer	Adam	For model parameter update
	Device	cuda	GPU acceleration enabled
Training	Adam	$\eta = 0.001$	Optimizer with learning rate
	BCELoss	Equations (12)	Binary cross-entropy loss function
Metrics	Acc/Prec/Rec/F1	Equations (13)–(15)	Classification metrics
	MSE/R^2	Equations (16) and (17)	Regression metrics

* MLP = Multilayer Perceptron; lon/lat = longitude/latitude; DL = Deep Learning; Temp = Temperature; Lon/Lat = Longitude/Latitude

4.2. Training Process

In the process of model training, we use Binary Cross-Entropy Loss (BCELoss) as the loss function and the Adam optimizer for parameter updates. To prevent overfitting and improve the efficiency of the training process, we employ the early-stopping method. A validation dataset is used during training, and the training process will be terminated early if the performance on the validation set does not improve for a certain number of consecutive epochs. The model is trained for 50 epochs, and the number of samples in each batch is set to 32 by default, which can be changed in practice (Table 2). Although the training process is performed on the GPU to accelerate the convergence rate of the model, the inference process can be performed entirely on the CPU, which makes it more flexible and widely applicable, especially in resource-constrained environments (Table 2).

In each epoch, the model is first set to the training mode and then traverses the training set data. We move the data to the CPU for each batch, empty the gradient, and then perform forward propagation to compute the output. We update the model parameters by calculating the loss values and backpropagating them. During training, we recorded the training loss and accuracy for each epoch to monitor the fitting ability of the model.

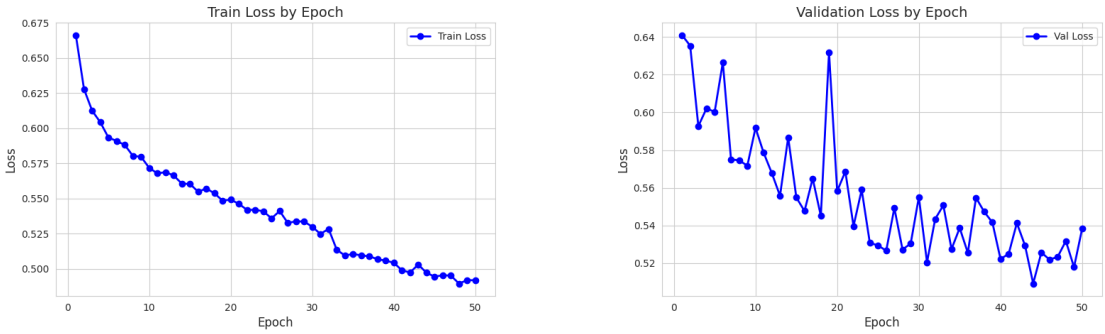
After each epoch, the model is set to the evaluation mode, and the loss and accuracy are calculated on the validation set. By saving the model weights with the lowest validation set loss, we indirectly achieve an effect similar to the early stopping strategy, avoiding overfitting.

 doi:10.20944/preprints202504.1854.v1

4.3. Experimental Results

The train loss gradually decreases from 0.6663 in the first epoch to 0.4917 in the 50th epoch, indicating that the fitting ability of the model on the training set is progressively enhanced, as shown in Figure 5.

The Val Loss decreases from 0.6411 in the first epoch to 0.5091 in the 44th epoch, showing an overall downward trend, but it fluctuates in some epochs, such as 0.6319 in the 19th epoch, as shown in Figure 5, indicating that the generalization ability of the model on the validation set has certain instability.



(**a**) Train loss by epoch (**b**) Validation loss by epoch

Figure 5. Trend of training loss and validation loss by Epoch in TEMixer.

With the increase of sequence length from 2 to 64, the training accuracy generally increased. For short sequences, each Fold fluctuates greatly and the difference is obvious, and for long sequences, it tends to be consistent and close to 1.0. The verification accuracy changed complicated, and there was no uniform rule in each Fold. Some of them decreased after reaching the peak around 4 to 8, and some increased at 64. The test accuracy first increased and then decreased, reaching a relative peak around 8 to 16. The over-fitting of the over-length sequence may reduce the accuracy, and the curves of different folds cross. The training time decreased overall, the length of short sequences and different folds varied greatly, and the length of long sequences was greatly shortened and tended to be close, as shown in Figure 6.

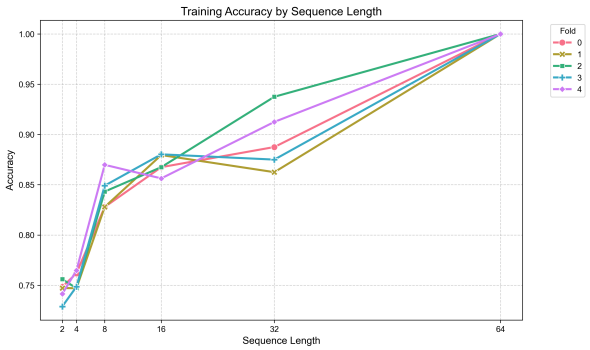
Through five-fold cross validation, it is found that when the sequence length is 16, the model has the best effect, and the train_accuracy reaches 0.880239521, val_accuracy reaches 0.853658537, and test_accuracy reaches 0.846153846, as shown in Table 3.

Table 3. Model result with varying Fold Size.

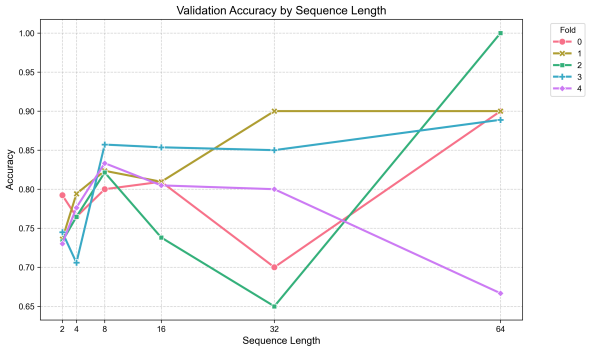
n	Sequence length	fold	Accuracy			Training Time (s)
			Train	Val	Test	
5487	16	0	0.86746988	0.80952381	0.826923077	1.795639753
5487	16	1	0.879518072	0.80952381	0.807692308	2.921166658
5487	16	2	0.86746988	0.738095238	0.846153846	1.3659904
5487	16	3	0.880239521	0.853658537	0.846153846	1.690137386
5487	16	4	0.856287425	0.804878049	0.75	1.039544582

* All values are normalized measurements (assuming Accuracy is a normalized value between 0 and 1).

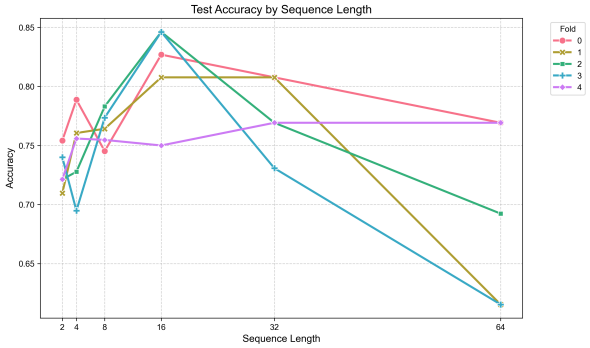
 doi:10.20944/preprints202504.1854.v1



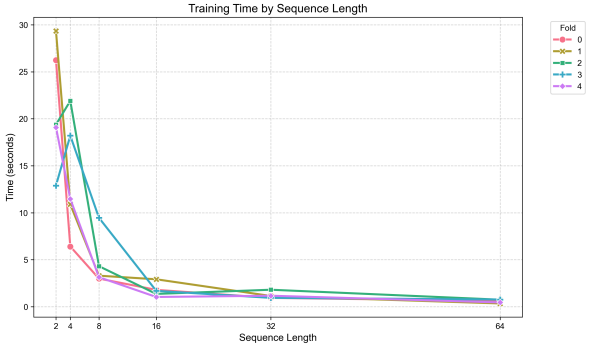
(**a**) Training Accuracy by Sequence Length



(**b**) Validation Accuracy by Sequence Length



(**c**) Test Accuracy by Sequence Length



(**d**) Training Time by Sequence Length

Figure 6. Comparison of TSMixer model training accuracy, validation accuracy, test accuracy and training time under different sequence lengths.

 doi:10.20944/preprints202504.1854.v1

4.4. Comparative Experiments

To verify the performance of the TSMixer model [15], we choose various classical deep learning models as comparison models, namely GRU [11], TCN [12], Dlinear [14], LSTM [10], RNN [9], and Transformer [13]. These models perform well in time series prediction and classification tasks and are able to provide a strong benchmark against which to compare our research.

We recorded the training loss (Train Loss), validation loss (Val Loss), validation set accuracy (Val Acc), mean squared error (MSE), and coefficient of determination (R^2) for each epoch. And each model uses the early stopping method to avoid the waste of computing resources. We use the validation accuracy (Val Acc) as the main evaluation metric to monitor the performance of the models and select the best model. We find that still the TSMixer model works best as shown in Table 4:

Table 4. Performance comparison results of different models.

Model	Train Loss	Val Loss	Val Acc	MSE	R^2
TSMixer [15]	0.4918	0.5178	0.7446	0.1759	0.2965
GRU [11]	0.4744	0.5299	0.7420	0.1747	0.3013
TCN [12]	0.5474	0.5187	0.7387	0.1753	0.2986
Dlinear [14]	0.5171	0.5397	0.7227	0.1841	0.2636
LSTM [10]	0.5537	0.5526	0.6913	0.1909	0.2366
RNN [9]	0.5543	0.5903	0.6769	0.2033	0.1869
Transformer [13]	0.6590	0.6628	0.6076	0.2352	0.0593

4.5. Ablation Study

To further evaluate the impact of sequence length on model performance, a five-fold cross-validation was conducted, and the results are illustrated in Figure 6. This figure demonstrates the changes in accuracy on the training, validation, and test sets, as well as the training time, across different sequence lengths (2, 4, 8, 16, 32, 64).

Observing the training accuracy (Figure 6a), it can be seen that the model's ability to fit the training data gradually improves with increasing sequence length, reaching near-perfect accuracy for all folds when the sequence length is 64. However, the trends in validation accuracy (Figure 6b) and test accuracy (Figure 6c) are more complex. We observed that for most folds, the model tends to achieve relatively high validation and test accuracy when the sequence length is 8 or 16. As the sequence length further increases to 32 and 64, the validation and test accuracy for some folds show a decreasing trend, which may indicate the onset of overfitting, where the model performs well on the training data but its ability to generalize to unseen data diminishes.

Furthermore, as shown in Figure 6d, the training time increases significantly with the sequence length. While shorter sequence lengths (such as 2 and 4) result in fast training, the training time grows exponentially when the sequence length is increased to 32 and 64, which necessitates a consideration of computational resource limitations in practical applications.

Overall, the results of the five-fold cross-validation suggest that a sequence length between 8 and 16 appears to be a suitable choice for the atrial fibrillation detection task. This range strikes a good balance between achieving favorable average validation and test accuracy and maintaining a reasonable training time. Although there is some variability in the results across different folds, the general trend indicates that both too short and too long sequence lengths can negatively impact the model's performance. These findings are consistent with the conclusions drawn from our previous single-validation experiment, further emphasizing the importance of selecting an appropriate sequence length.

 doi:10.20944/preprints202504.1854.v1

5. Discussion

5.1. Model Advantages

In the field of processing data from ocean buoys, TSMixer demonstrates significant advantages. Its unique architecture combines Time-mixing and Feature-mixing mechanisms; the former captures temporal dependencies, and the latter mines feature correlations. The two mechanisms work together to break the limitations of traditional single processing methods and capture dependencies comprehensively and deeply [15]. At the same time, for ocean buoy data containing multiple features, TSMixer effectively fuses multivariate information through feature mixing, which overcomes the shortcomings of traditional statistical methods that require high data stationarity and are difficult to capture nonlinear relationships. Additionally, the multi-layer MixerLayer stack provides the model with powerful modeling capabilities, allowing for automatic parameter adjustments during training. In the face of different ocean environment data, the model can adaptively learn the law and accurately capture complex and changing dependencies, which provides more reliable support for predicting whether an ocean buoy is about to be inside the vortex.

5.2. Model Performance Analysis

TSMixer performs well on the validation set's accuracy. In the process of increasing the number of training epochs, the accuracy of the validation set is gradually improved, which offers obvious advantages compared with other models. Compared with other models, the MSE value is lower and the R^2 value is higher (Table 4), indicating that TSMixer has a stronger ability to explain the data , can better capture the law in the data, reveal the relationship between the independent variable and the dependent variable, and provide more powerful support for predicting the state of the buoy. The model can more accurately approximate the real situation when predicting whether the buoy is inside the vortex. The prediction accuracy of the model is high.

6. Conclusion

This paper innovatively proposes using the time series model TSMixer to predict whether an ocean buoy is about to be inside the vortex.

After multiple epochs of training, when the TSMixer model takes all the profiles before and after the center of the vortex for five days as the vortex data, the accuracy of the validation set reaches 0.853658537 (85.37%), the accuracy of the test set is 0.846153846 (84.62%) under the configuration of the data scale of 5487 and the sequence length of 16 (Table 3). And the Mean Square Error (MSE) is 0.1759. The coefficient of determination (R^2) is 0.2965 (Table 4). In addition, the model shows high efficiency in the training process, with the training time of only 25.68 seconds and the data loading time as low as 0.034 seconds, making it very suitable for time-sensitive application scenarios. These results indicate that TSMixer can effectively capture the regularizations in the buoy data and accurately predict whether the buoy is inside the vortex.

Predicting whether the buoy is about to be inside the vortex is of great significance to reality. This technology helps to optimize the allocation of Marine monitoring resources, protect sensors, promote the study of vortex behavior and environmental impacts, and reduce costs through reasonable equipment deployment to provide strong support for Marine environmental research.

Author Contributions: Conceptualization, J.Z. Zhu, Y.L.; methodology, J.Z. Zhu, Y.L.; Code Y.L.; writing, J.Z. Zhu; Buoy defense, data acquisition and processing, vortex identification and labeling, T.L.

Funding: This work was partially supported by the National Science and Technology Major Project of China (Grant No. 2022ZD0117201), and the Natural Science Foundation of China (Grant No. 42394130).

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to (Data are available from the WHOI ITP program upon request. Derived data are available from the authors.).

 doi:10.20944/preprints202504.1854.v1

Acknowledgments: This work was supported by China Ocean University and Woods Hole Oceanographic Institution (WHOI).

Conflicts of Interest: The authors declare no conflicts of interest. Data can be shared on the request.

Abbreviations

The following abbreviations are used in this manuscript:

batch_size	Number of samples input into the model each time
sequence_length	Length of the time series, that is, the number of time steps
num_features	Number of features contained in each time step
DL	Deep Learning; Temp = Temperature
Temp	Temperature
Lon/Lat	Longitude/Latitude
MLP	Multi-layer Perceptron

References

1. Lin, M.; Yang, C. Ocean Observation Technologies: A Review. *Chinese Journal of Mechanical Engineering* **2020**, *33*, 32. https://doi.org/10.1186/s10033-020-00449-z.
2. Soreide, N.; Woody, C.; Holt, S. Overview of ocean based buoys and drifters: present applications and future needs. In Proceedings of the MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295), 2001, Vol. 4, pp. 2470–2472 vol.4. https://doi.org/10.1109/OCEANS.2001.968388.
3. Song, D.L.; Wang, H.J.; Zhou, L.Q.; et al.. 下放式海洋微结构湍流剖面仪运动学与动力学分析. 中国海洋大学学报（自科版） **2019**, *49*, 145–152. https://doi.org/10.16441/j.cnki.hdxb.20170102.
4. Li, Y.; Yang, F.; Li, S.; Tang, X.; Sun, X.; Qi, S.; Gao, Z. Influence of Six-Degree-of-Freedom Motion of a Large Marine Data Buoy on Wind Speed Monitoring Accuracy. *Journal of Marine Science and Engineering* **2023**, *11*. https://doi.org/10.3390/jmse11101985.
5. Mou, N.X.; Zhang, H.C.; Chen, J.; Zhang, L.X.; Dai, H.L. A Review on the Application Research of Trajectory Data Mining in Urban Cities. *Journal of Geo-information Science* **2015**, *17*, 1136–1142. https://doi.org/10.3724/SP.J.1047.2015.01136.
6. Wang, J.; Fu, L.L.; Haines, B.; Lankhorst, M.; Lucas, A.J.; Farrar, J.T.; Send, U.; Meinig, C.; Schofield, O.; Ray, R.; et al. On the Development of SWOT In Situ Calibration/Validation for Short-Wavelength Ocean Topography. *Journal of Atmospheric and Oceanic Technology* **2022**, *39*, 595 – 617. https://doi.org/10.1175/JTECH-D-21-0039.1.
7. Zhang, Q.; Wang, H.; Dong, J.; Zhong, G.; Sun, X. Prediction of Sea Surface Temperature Using Long Short-Term Memory. *IEEE Geoscience and Remote Sensing Letters* **2017**, *14*, 1745–1749. https://doi.org/10.1109/LGRS.2017.2733548.
8. Song, M.; Hu, W.; Liu, S.; Chen, S.; Fu, X.; Zhang, J.; Li, W.; Xu, Y. Developing an Artificial Intelligence-Based Method for Predicting the Trajectory of Surface Drifting Buoys Using a Hybrid Multi-Layer Neural Network Model. *Journal of Marine Science and Engineering* **2024**, *12*. https://doi.org/10.3390/jmse12060958.
9. Zaremba, W.; Sutskever, I.; Vinyals, O. Recurrent Neural Network Regularization, 2015, [arXiv:cs.NE/1409.2329].
10. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **1997**, *9*, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
11. Chung, J.; Çağlar Gülçehre.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555* **2014**. https://doi.org/10.48550/arXiv.1412.3555.
12. Lea, C.; Vidal, R.; Reiter, A.; Hager, G.D. Temporal Convolutional Networks: A Unified Approach to Action Segmentation, 2016.
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems; Guyon, I.; Luxburg, U.V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R., Eds. Curran Associates, Inc., 2017, Vol. 30.
14. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence* **2022**, *37*, 11121–11128. https://doi.org/10.1609/aaai.v37i9.26317.

 doi:10.20944/preprints202504.1854.v1

15. Ekambaram, V.; Jati, A.; Nguyen, N.; Sinthong, P.; Kalagnanam, J. TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting. In Proceedings of the Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2023; KDD '23, pp. 459–469. https://doi.org/10.1145/3580305.3599533.
16. Toole, J.; Krishfield, R.; Proshutinsky, A.; Ashjian, C.; Doherty, K.; Frye, D.; Hammar, T.; Kemp, J.; Peters, D.; Timmermans, M.L.; et al. Ice-tethered profilers sample the upper Arctic Ocean. *Eos, Transactions American Geophysical Union* **2006**, *87*, 434–438. https://doi.org/10.1029/2006EO410003.
17. ITP, W. Ice-Tethered Profiler Observational Dataset, 2023. [Online]. Available: https://www2.whoi.edu/site/itp/. (Data can be shared upon request).
18. Rhines, P.B. Slow oscillations in an ocean of varying depth Part 1. Abrupt topography. *Journal of Fluid Mechanics* **1969**, *37*, 161–189. https://doi.org/10.1017/S0022112069000474.
19. Singh, D.; Singh, B. Investigating the impact of data normalization on classification performance. *Applied Soft Computing* **2020**, *97*, 105524. https://doi.org/https://doi.org/10.1016/j.asoc.2019.105524.