Journal of *Marine Science and Engineering*

MDPI

*Article*

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

**Jingzi Zhu [1], Yu Luo [2], Tao Li [3], Yanhai Gan [4,*] and Junyu Dong [4,*]**

[1] Haide College, Ocean University of China, 238 Songling Road, Qingdao 266100, China; zhujingzi@stu.ouc.edu.cn
[2] School of Mathematical Sciences, Ocean University of China, 238 Songling Road, Qingdao 266100, China; luoyu@stu.ouc.edu.cn
[3] College of Oceanic and Atmospheric Sciences, Ocean University of China, 238 Songling Road, Qingdao 266100, China
[4] Faculty of Information Science and Engineering, Ocean University of China, 238 Songling Road, Qingdao 266100, China
* Correspondence: 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 a vortex. Marine buoys are an important tool for collecting ocean data and studying ocean dynamics, climate change, and ecosystem health. A vortex is an important ocean dynamic process. If we can predict that a buoy is about to enter a 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 a 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.

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

check for updates

Academic Editor: Chung-yen Kuo

Received: 18 April 2025
Revised: 13 May 2025
Accepted: 15 May 2025
Published: 22 May 2025

**Citation:** Zhu, J.; Luo, Y.; Li, T.; Gan, Y.; Dong, J. Prediction of the Marine Dynamic Environment for Arctic Ice-Based Buoys Using Historical Profile Data. *J. Mar. Sci. Eng.* **2025**, *13*, 1003. https://doi.org/10.3390/jmse13061003

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

The ocean buoy is an important observational 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 defense construction [1]. The study of ocean buoys and vortices 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 the 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 losses [3]. The ability to predict whether an ocean buoy is about to be inside the vortex is valuable to improve monitoring efficiency by optimizing deployment strategies (e.g., adjusting the placement and timing of the buoy). Compared with the vortex data that need to be corrected by the ALIS (Autonomous Lagrangian Instrument System) and other algorithms, the buoys in non-vortex regions exhibit more

stable and predictable trajectories. These non-vortex measurements demonstrate greater continuity, reliability, and processing efficiency due to the absence of strong rotational flow disturbances [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 rapid 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 improving monitoring accuracy, ensuring equipment safety, and optimizing resource allocation, and it is an effective solution to the problem of vortex data acquisition [6].

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 marine environments. 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 a buoy is inside a 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. If the traditional dynamic model is applied to the buoy, it needs to simulate the whole Arctic, which is too large to be deployed on the buoy, and the dynamic simulation results are different from the actual situation. Intelligent prediction based on in situ observation can quickly give prediction results according to real-time field data to support the intelligent decision of the buoy. 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. A recurrent neural network (RNN) deals with variable-length sequences 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 inference that is five times faster [15].

Therefore, based on TSMixer [15], we establish a time-series model to predict whether an ocean buoy is about to be inside a vortex. In the comparison experiment, by comparing the 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, are used for deep learning, and TSMixer is proven 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.

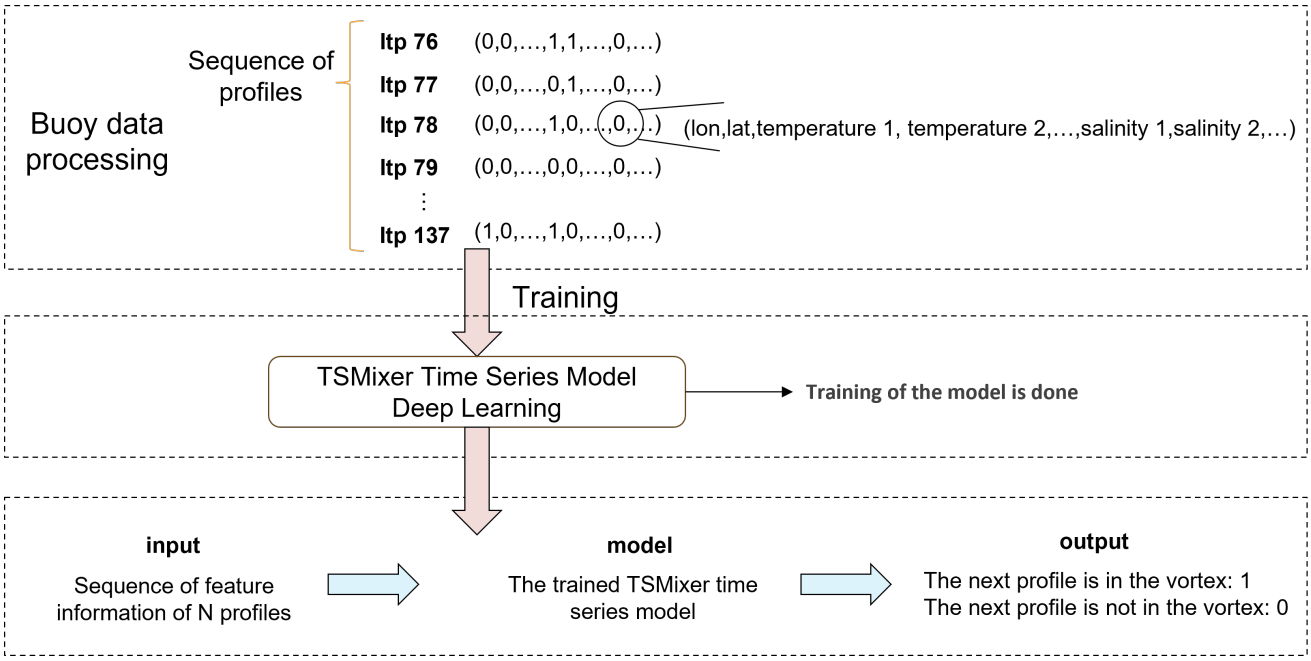



**Figure 1.** Flow chart of the model.

## 2. Dataset

### 2.1. Data Introduction

Between 2014 and 2023, 56 ice-based towed profilers (ITPs), deployed by the Woods Hole Oceanographic Institution (WHOI) in the Arctic Ocean with their drift trajectories illustrated in Figures 2 and 3, collected hundreds to thousands of profile data by floating up and down, as shown in Figure 4, forming a data set containing the 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 m 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].

**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 |

**Table 1.** *Cont.*

| Buoy | Profile Number | Buoy | Profile Number | Buoy | Profile Number |
|---|---|---|---|---|---|
| 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 |

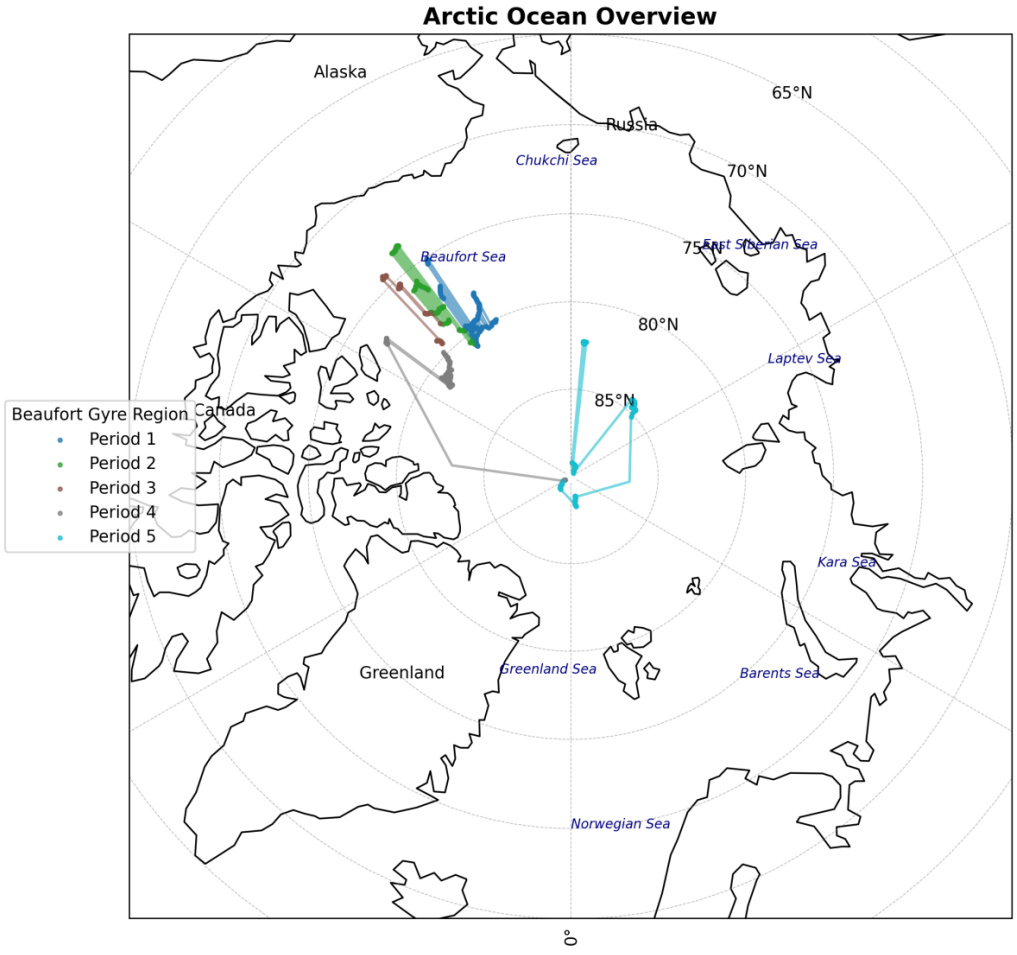



**Figure 2.** Arctic Ocean overview.

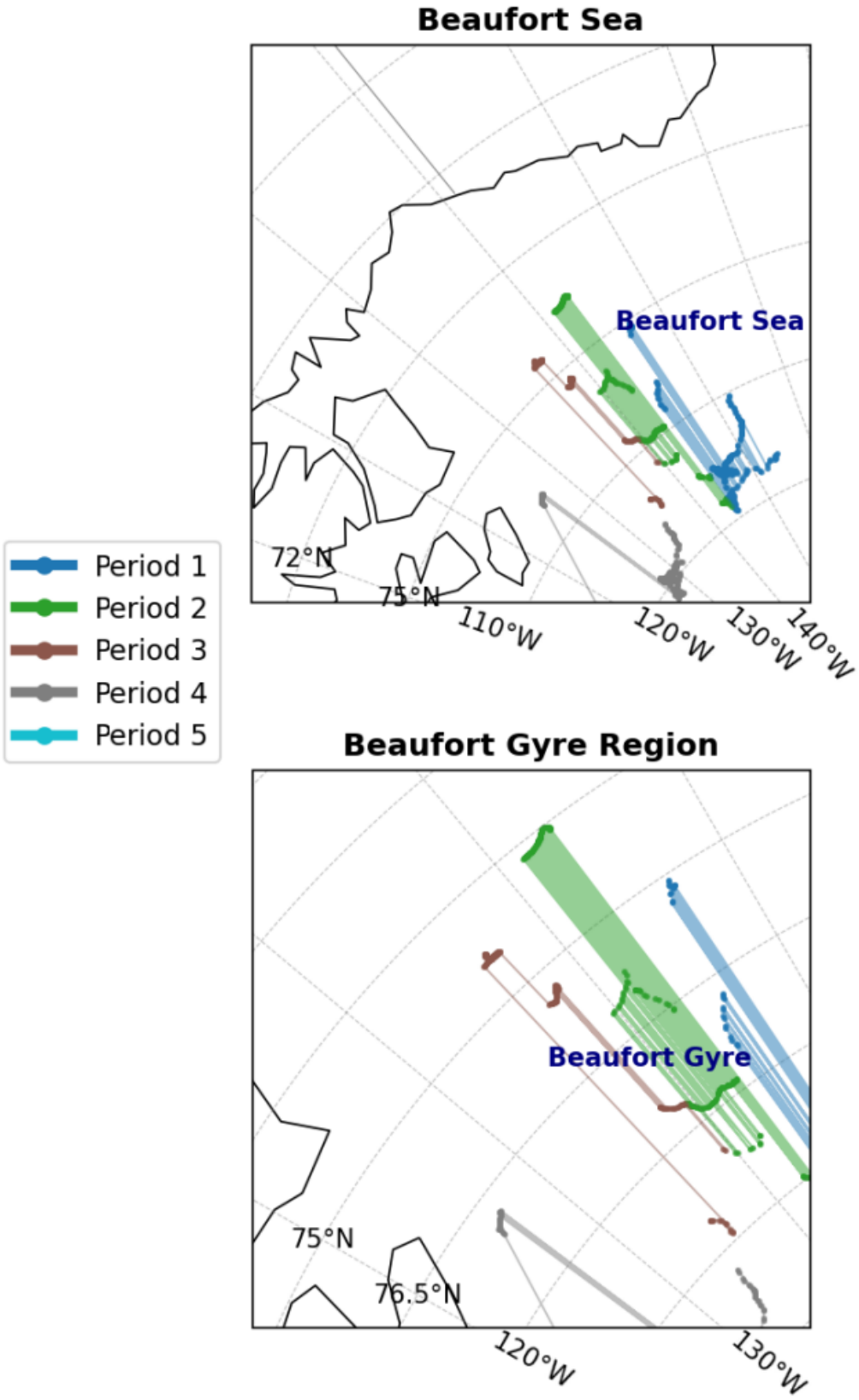



**Figure 3.** Beaufort sea and Beaufort Gyre Region.

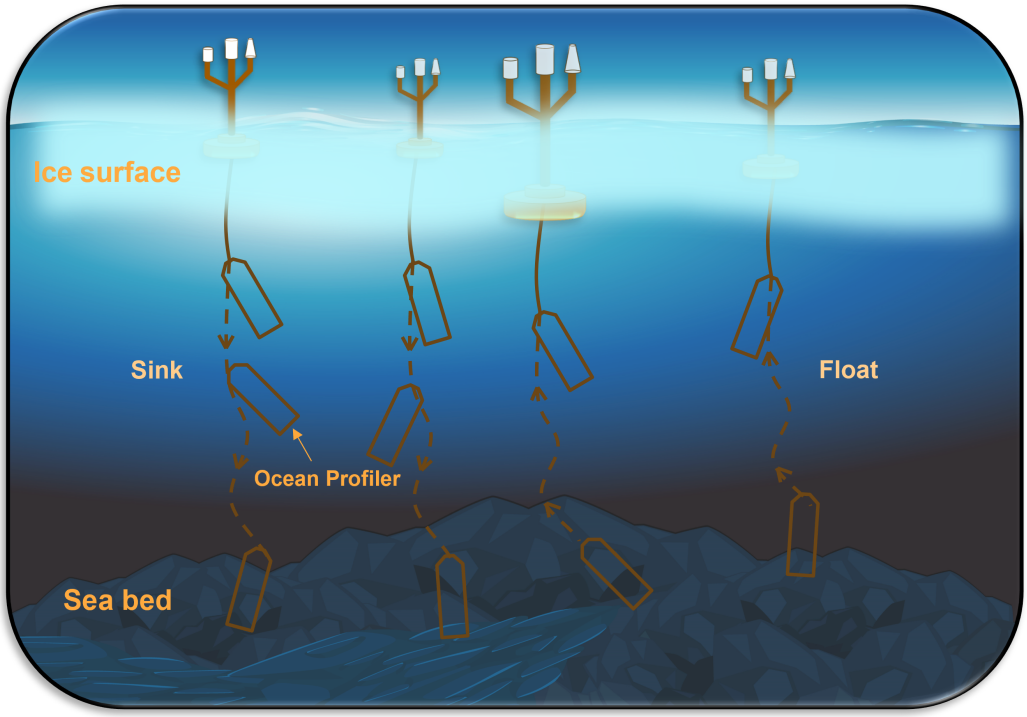



**Figure 4.** Buoy acquisition data diagram.

### 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.

### 2.3. Data Preprocessing

The experiment collected data from 12 April 2014 to 13 January 2023, using a total of 51 buoys. Each buoy periodically sinks and floats (Figure 4), 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 it is meaningless to study whether areas that are too deep are in a vortex. Currents in the surface layer (within 10 m) of the Arctic are subjected to friction by sea ice, and eddy activity is reduced or even eliminated [18]. Therefore, excluding data below 10 m 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 m to 200 m 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. We convert all values to numeric types. We assign the missing values caused by data collection 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 their standard normal z-values [19]:

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

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

where

- $\mu$ is the mean of the feature;
- $\sigma$ 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 are lenient with the data and discard a vector if more than a third of it is zeros. In other words, we treat abnormal data cautiously and do not 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 a 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 the 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: a time-mixing part and a feature-mixing part, as shown in Figure 5.

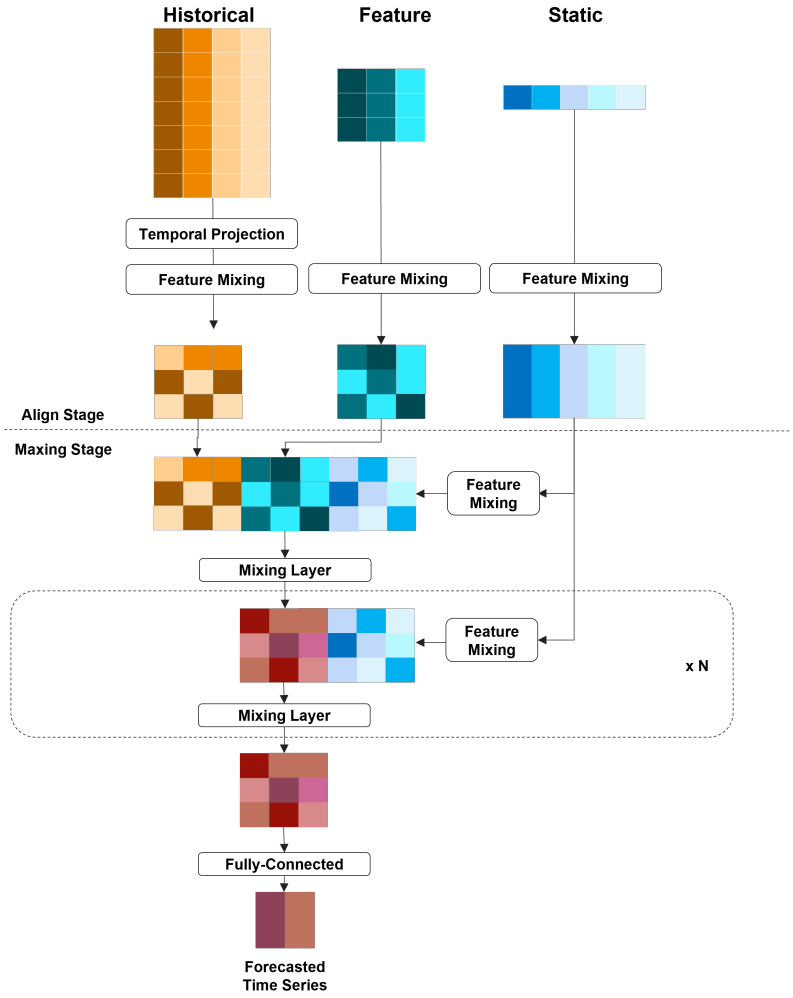



**Figure 5.** TSMixer's overall architecture and multi-source data fusion prediction process diagram.

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

#### 3.1.1. Input Layer

The input data of the model have 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.

#### 3.1.2. MixerLayer Stacking

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 6 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 joining. Mixer Layer stacking (Mixer Layer × N) and the final Temporal Projection process are also involved, which present the key links of the model to process time series data.

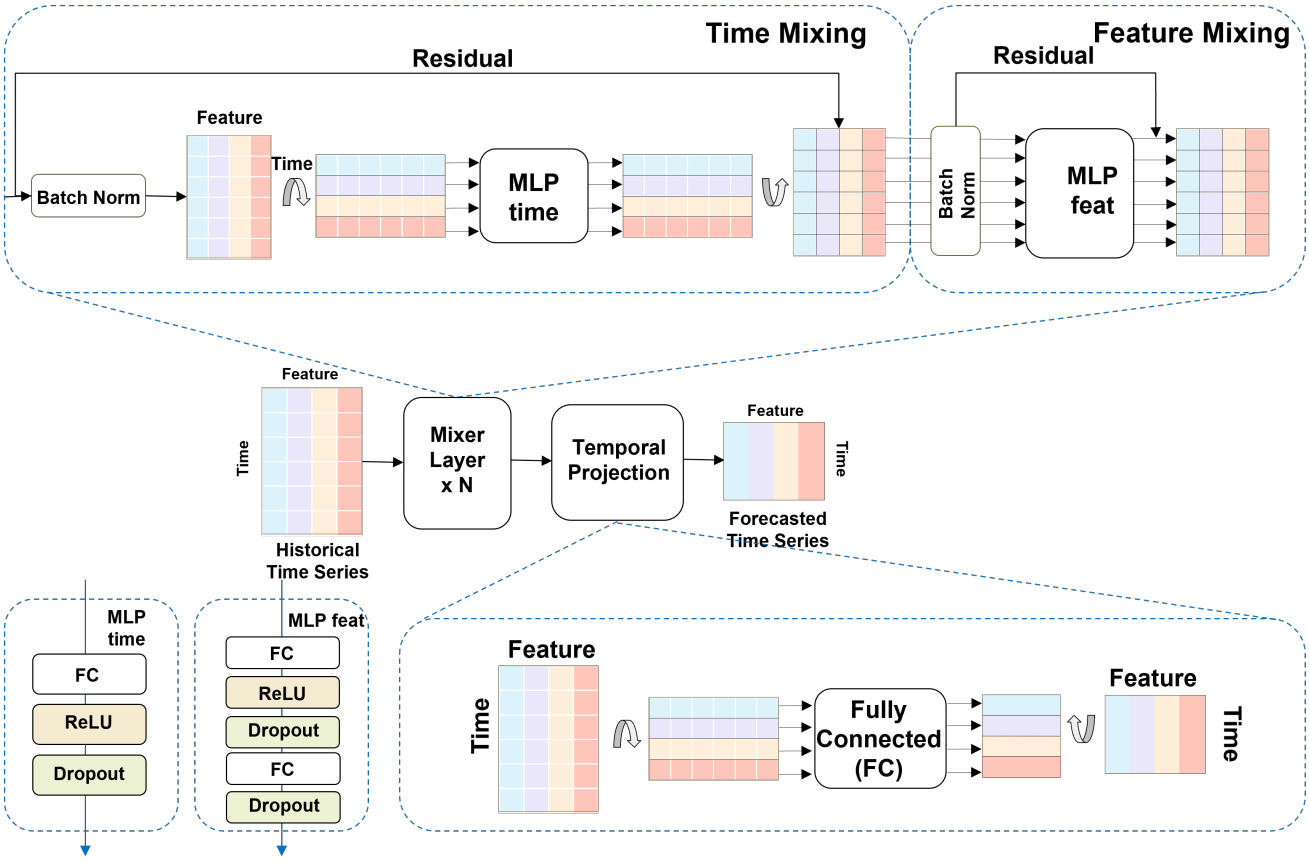



**Figure 6.** Schematic diagram of the TSMixer's single Mixer Layer internal structure and processing flow.

### 3.1.3. 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 (Rectified Linear Unit) 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 6).

#### 3.1.4. 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.

#### 3.1.5. 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,

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

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 $\mu$ and variance $\sigma^2$ 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 $\epsilon$ is a small constant to prevent division by zero, and $\gamma$ and $\beta$ are learnable parameters.

#### 3.1.6. 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 is

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

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 dimensions as the input in the feature dimension.

#### 3.1.7. 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, as shown in Algorithm 1.

**Algorithm 1** Training of TSMixer

Initialize the model parameters $\theta$ 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 $\alpha$: $\theta \leftarrow O(\theta, \nabla_\theta loss, \alpha)$.
  **end for**
  Evaluate the model on a validation set (if available) to monitor performance and potentially adjust hyperparameters.
**end for**

### 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

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

- TP (True Positive): True example, actually positive, model predicted positive.
- TN (True Negative): True negative, actually negative, model predicted negative.
- FP (False Positive): False positive, actually negative, model mispredicted positive.
- FN (False Negative): False negative, actually positive, model mispredicted negative.

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

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

## 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 24 GB 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.

**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 ($\eta$ = 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 | Equation (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; DL = Deep Learning;

### 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

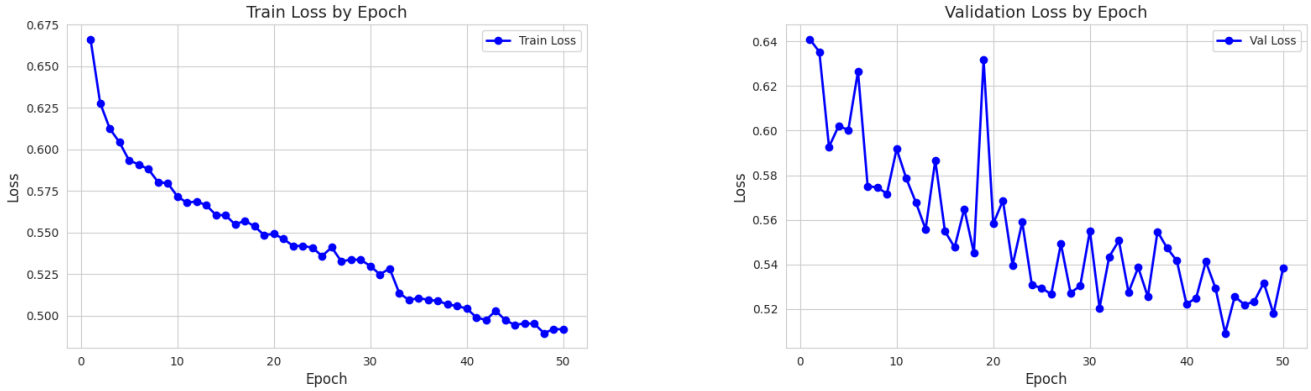
consecutive epochs. The model is trained for 50 epochs, and the number of samples in each batch is set to 32 by default, which can be changed in practice (Table 2). Although the training process is performed on the GPU to accelerate the convergence rate of the model, the inference process can be performed 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.

### *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 7.




(**a**) (**b**)

**Figure 7.** Trend of training loss and validation loss by Epoch in TEMixer. (**a**) Training loss by epoch. (**b**) Validation loss by epoch.

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 7, indicating that the generalization ability of the model on the validation set has certain instability.

As the sequence length increases from 2 to 64, the training accuracy generally increases. For short sequences, the boxes show large spreads with wide interquartile ranges and long whiskers in different folds, revealing significant fluctuations. As the sequence length grows, long-sequence boxes become narrow, with medians approaching 1.0, indicating high consistency. The validation accuracy box plots are complex; some boxes peak in height around sequence lengths of 4 to 8 and then decline, while others rise at 64, with no unified pattern. The test accuracy boxes show a trend of first rising and then falling, with a relative peak in median position around 8 to 16. Overlong sequences might cause overfitting, as suggested by lower box medians and scattered points. Different fold box plots overlap, showing inconsistent performance. The training time box plots show that for short sequences, boxes are high and spread out, while for long sequences, they are low and close together, as shown in Figure 8.

Table 3 shows the statistical results of the training, testing, and validation sets after five-fold cross-validation. The results show that the mean test set accuracy was 0.815, with a standard deviation of 0.040, and the values ranged from 0.750 to 0.846. The validation set accuracy also exhibited similar variability (standard deviation of 0.041). This variability indicates that the model's performance has a degree of sensitivity to the specific data partitioning.

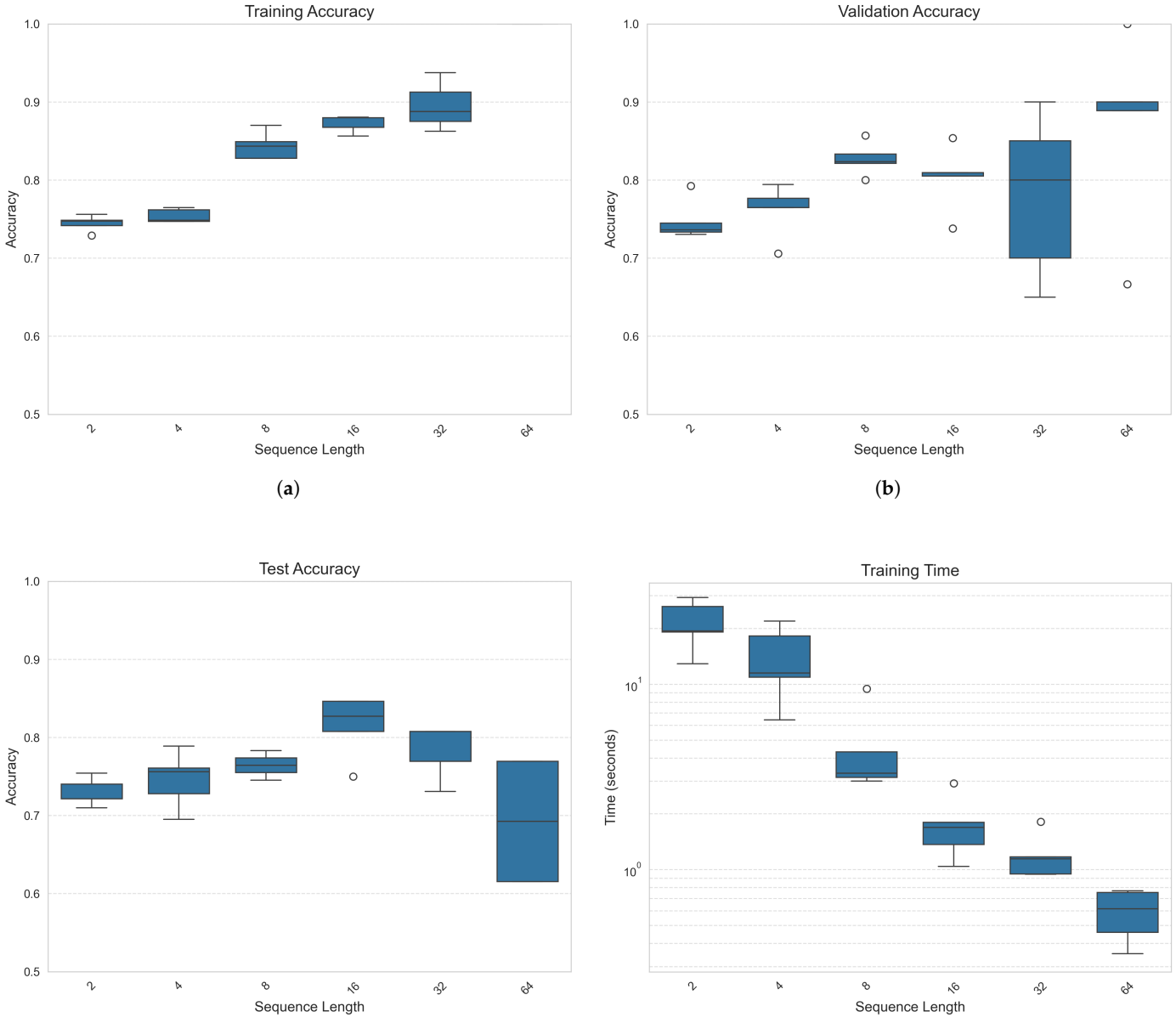



**Figure 8.** Comparison of TSMixer model training accuracy, validation accuracy, test accuracy, and training time under different sequence lengths. (**a**) Training accuracy by sequence length. (**b**) Validation accuracy by sequence length. (**c**) Testing accuracy by sequence length. (**d**) Training time by sequence length.

**Table 3.** Summary of 5-fold cross-validation results (n = 5487, sequence length = 16). Metrics are reported as mean ± standard deviation.

| Metric | Value |
|---|---|
| Train Accuracy | 0.8702 ± 0.0099 |
| Validation Accuracy | 0.8031 ± 0.0414 |
| Test Accuracy | 0.8154 ± 0.0399 |
| Test Accuracy 95% CI | [0.7659, 0.8649] |
| Avg. Training Time (s) | 1.76 ± 0.71 |

All accuracy values are normalized measurements. Training time is in seconds.

### *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.

Although GRU and TCN achieved slightly higher validation accuracy than TSMixer—0.8013 and 0.7994, respectively, compared to 0.7994 for TSMixer—this difference is not statistically significant according to the paired $t$-test results ($p = 0.1199$, greater than 0.05). Moreover, GRU requires GPU acceleration during inference, which limits its applicability on hardware without such computational capabilities. In contrast, TSMixer performs inference efficiently on general-purpose hardware without the need for GPU resources. Considering our deployment environment, where GPU availability is limited, TSMixer offers a more practical solution despite the marginally lower accuracy.

Furthermore, although TCN shows comparable validation accuracy to TSMixer, the statistical analysis reveals no significant difference between them ($p = 0.1199$). However, TCN is a relatively older architecture introduced in 2018, whereas TSMixer, proposed in 2023, incorporates more recent advances in time-series modeling. Its modern architectural design offers better scalability and aligns with the current research trends. Therefore, given their similar performance and the lack of statistically significant differences, we selected TSMixer as the preferred model, balancing accuracy, deployment feasibility, and alignment with the state of the art.

### *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 8. 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, and 64).

Observing the box plots of training accuracy (Figure 8a), the model's ability to fit training data improves as the sequence length increases. For short sequences, boxes of different folds have large spreads and significant differences. When the sequence length is 64, the boxes are narrow with medians close to 1.0, showing excellent fitting. The trends of box plots for validation accuracy (Figure 8b) and test accuracy (Figure 8c) are complex. For most folds, boxes at sequence lengths of 8 or 16 have high medians, indicating good accuracy. However, when it increases to 32 and 64, some fold-specific box medians decline, potentially indicating overfitting.

The box plots of training time (Figure 8d) show that short sequences have short and stable training times. When the sequence length reaches 32 and 64, the training time increases. But at a sequence length of 64, the model has high accuracy. If computational resources permit, its performance advantages are worth considering.

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.

## 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 capture temporal dependencies, and the latter mine 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 have difficulty capturing 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.

**Table 4.** Model performance summary with TSMixer Val Acc statistical comparison.

| Model | Val Acc | MSE | $R^2$ |
|---|---|---|---|
| Mamba | 0.7581 ± 0.0104 | 0.1985 ± 0.0194 | 0.2057 ± 0.0748 |
| LSTM | 0.7248 ± 0.0360 | 0.2109 ± 0.0249 | 0.1559 ± 0.1022 |
| TCN | 0.7994 ± 0.0049 | 0.1476 ± 0.0045 | 0.4091 ± 0.0197 |
| RNN | 0.6471 ± 0.0064 | 0.2208 ± 0.0028 | 0.1163 ± 0.0110 |
| GRU (needs GPU) | 0.8013 ± 0.0070 | 0.1459 ± 0.0053 | 0.4162 ± 0.0220 |
| Transformer | 0.6309 ± 0.0053 | 0.2259 ± 0.0046 | 0.0960 ± 0.0184 |
| DLinear | 0.6495 ± 0.0111 | 0.2174 ± 0.0038 | 0.1298 ± 0.0138 |
| iTransformer | 0.6796 ± 0.0138 | 0.2036 ± 0.0064 | 0.1852 ± 0.0257 |
| TSMixer (Ours) | 0.7994 ± 0.0066 | 0.1591 ± 0.0041 | 0.3632 ± 0.0171 |

**Table 4.** *Cont.*

| Paired *t*-Test *p*-Values for Val Acc (TSMixer vs. Others) | | |
|---|---|---|
| **Comparison** | ***t*-Statistic** | ***p*-Value** |
| TSMixer vs. Mamba | 5.8232 | 0.0043 |
| TSMixer vs. LSTM | 4.1660 | 0.0141 |
| TSMixer vs. TCN | −1.9722 | 0.1199 |
| TSMixer vs. RNN | 5.3506 | 0.0059 |
| TSMixer vs. GRU | −4.3121 | 0.0125 |
| TSMixer vs. Transformer | 4.8380 | 0.0084 |
| TSMixer vs. DLinear | 5.3194 | 0.0060 |
| TSMixer vs. iTransformer | 3.5243 | 0.0242 |

## 6. Conclusions

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 training epochs, when the TSMixer model takes all the profiles collected before and after the center of the vortex over five consecutive days as the vortex dataset, the average validation accuracy reaches 80.31%, and the average test accuracy reaches 81.54%, under the configuration of a data scale of 5487 samples and a sequence length of 16 (Table 3). The corresponding Mean Square Error (MSE) is 0.1591, and the coefficient of determination ($R^2$) is 0.3632, based on statistical results from multiple experimental runs (Table 4).Additionally, the model demonstrates high computational efficiency, achieving an average total training time of 25.68 s for five-fold cross-validation, making it highly suitable for time-sensitive application scenarios.These statistical results confirm that TSMixer effectively captures the spatiotemporal patterns in buoy data and reliably predicts whether the buoy is located within a vortex.

Predicting whether the buoy is about to be inside the vortex is of great significance to practice. 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. and Y.L.; Methodology, J.Z. and Y.L.; Code, Y.L.; Writing—original draft preparation, J.Z.; Buoy observation defense, data acquisition and processing, vortex identification and labeling, T.L.; Project administration, Y.G. and J.D. All authors have read and agreed to the published version of the manuscript.

**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 used in this study are publicly available at https://www2.whoi.edu/site/itp/ (accessed on 10 December 2024). However, data annotations are not publicly accessible. Data can be shared upon request. The code is available at: https://github.com/qimingfan10/Buoy_prediction.git (accessed on 18 April 2025).

**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.

## 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 |
| Lon/Lat | Longitude/Latitude |
| MLP | Multi-layer Perceptron |
| ReLU | Rectified Linear Unit |

## References


1. Lin, M.; Yang, C. Ocean Observation Technologies: A Review. *Chin. J. Mech. Eng.* **2020**, *33*, 32. [CrossRef]
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), Honolulu, HI, USA, 5–8 November 2001; Volume 4, pp. 2470–2472. [CrossRef]
3. Song, D.L.; Wang, H.J.; Zhou, L.Q.; Zang, S.P. Kinematic and dynamic analysis of a lowered ocean microstructure turbulence profiler. *Period. Ocean. Univ. China (Nat. Sci. Ed.)* **2019**, *49*, 145–152. [CrossRef]
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. *J. Mar. Sci. Eng.* **2023**, *11*, 1985. [CrossRef]
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. *J.-Geo-Inf. Sci.* **2015**, *17*, 1136–1142. [CrossRef]
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. *J. Atmos. Ocean. Technol.* **2022**, *39*, 595–617. [CrossRef]
7. Zhang, Q.; Wang, H.; Dong, J.; Zhong, G.; Sun, X. Prediction of Sea Surface Temperature Using Long Short-Term Memory. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 1745–1749. [CrossRef]
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. *J. Mar. Sci. Eng.* **2024**, *12*, 958. [CrossRef]
9. Zaremba, W.; Sutskever, I.; Vinyals, O. Recurrent Neural Network Regularization. *arXiv* **2015**, arXiv:1409.2329.
10. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
11. Chung, J.; Gülçehre, Ç.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
12. Lea, C.; Vidal, R.; Reiter, A.; Hager, G.D. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. *arXiv* **2016**, arXiv:1608.08242.
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; 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; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
14. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are Transformers Effective for Time Series Forecasting? *Proc. AAAI Conf. Artif. Intell.* **2022**, *37*, 11121–11128. [CrossRef]
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 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '23, Long Beach, CA USA, 6–10 August 2023; pp. 459–469. [CrossRef]
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 Trans. Am. Geophys. Union* **2006**, *87*, 434–438. [CrossRef]
17. ITP. Ice-Tethered Profiler Observational Dataset. 2023. Available online: https://www2.whoi.edu/site/itp/ (accessed on 21 April 2025).

18. Rhines, P.B. Slow oscillations in an ocean of varying depth Part 1. Abrupt topography. *J. Fluid Mech.* **1969**, *37*, 161–189. [CrossRef]
19. Singh, D.; Singh, B. Investigating the impact of data normalization on classification performance. *Appl. Soft Comput.* **2020**, *97*, 105524. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.