Scientific Research Publishing

Atmospheric and Climate Sciences, 2024, 14, 429-449
https://www.scirp.org/journal/acs
ISSN Online: 2160-0422
ISSN Print: 2160-0414

Optimizing the LSTM Deep Learning Model for Arctic Sea Ice Melting Prediction

Victoria Pegkou Christofi, Xiaodi Wang

Department of Mathematics, Western Connecticut State University, Danbury, USA
Email: victoriapc006@gmail.com, wangx@wcsu.edu

How to cite this paper: Pegkou Christofi, V. and Wang, X.D. (2024) Optimizing the LSTM Deep Learning Model for Arctic Sea Ice Melting Prediction. *Atmospheric and Climate Sciences*, **14**, 429-449.
https://doi.org/10.4236/acs.2024.144026

Received: April 7, 2024
Accepted: October 11, 2024
Published: October 14, 2024

Abstract

The National Oceanic and Atmospheric Administration reports a 95% decline in the oldest Arctic ice over the last 33 years [1], while the National Aeronautics and Space Administration states that summer Arctic Sea Ice Extent (SIE) is shrinking by 12.2% per decade since 1979 due to warmer temperatures [2]. Given the rapidly changing Arctic conditions, accurate prediction models are crucial. Deep learning models developed for Arctic forecasts primarily focus on exploring convolutional neural networks (CNN) and convolutional Long Short-Term Memory (LSTM) networks, while the exploration of the power of LSTM networks is limited. In this research, we focus on enhancing the performance of an LSTM network for predicting monthly Arctic SIE. We leverage five climate and atmospheric variables, validated for their correlation with SIE in prior studies [3]. We utilize the Spearman's rank correlation and ExtraTrees regressor to enhance our understanding of the importance of the five variables in predicting SIE. We further enhance our predictor variables with seasonal information, lagged time steps, and a linear regression simulated SIE that accounts for the influence of past SIE on current SIE. Statistical methods guide our selection of data scalers and best evaluation metrics for our model. By experimenting with hyperparameter optimization and advanced deep learning training techniques, such as batch sizes, number of neurons, early stopping, and model checkpoint, our model achieved a Mean Absolute Error (MAE) of 0.191 and R^2 of 0.996, underscoring its ability to account for nearly all the variance in our data and holds great promise for the prediction of SIE.

Keywords

Arctic, Sea Ice Extent, Deep Learning, Long Short-Term Memory Networks, Climate Change

1. Introduction

The rapid decline of Arctic Sea ice has become increasingly relevant due to its profound and far-reaching effects on the environment, climate, ecosystems, and human communities [2]. The most common measure of sea ice is extent, defined as the area covered with an ice concentration of at least 15%. Extent increases and decreases with the seasons. As shown in **Figure 1**, the median Arctic extent, as assessed over the period 1981-2010, is greatest in mid-March. The minimum median extent falls in mid-September [4].

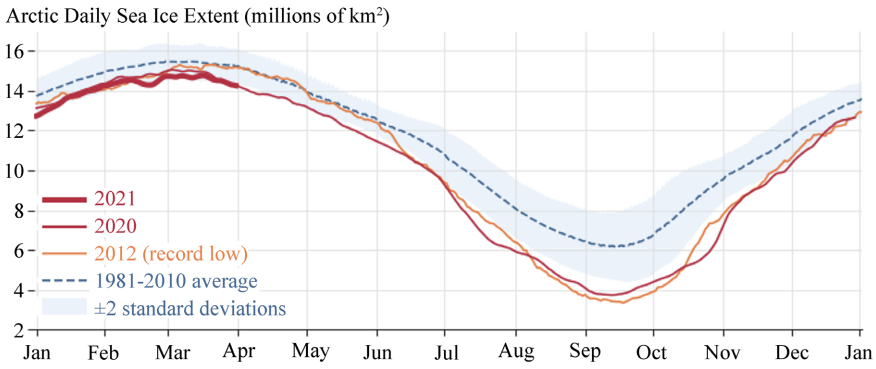

Figure 1. Daily Arctic Sea Ice Extent in millions of km² [5].

Since 2011, the decline of sea ice has persisted, and near-surface permafrost has continued to warm, signaling a transformation of the Arctic into a warmer, more humid, and less predictable environment [6]. The stability of marine ecosystems and the very survival of marine life are put at risk as sea ice continues to melt. Moreover, the alterations in the Arctic region will amplify disruptions in ocean currents, affecting global weather patterns, the viability of fishing industries, and the vulnerability of coastal cities to increased flooding risks. The consequences of melting sea ice underscore the urgent need for the development of precise predictive models that can capture the dynamic changes occurring in the Arctic. These models are essential for not only understanding the impacts of decreased sea ice on the Arctic but also for gaining invaluable insights into the broader issue of climate change.

Various approaches have been made to develop prediction models for the Arctic, however they focus on exploring convolutional neural networks and Convolutional Long Short-Term Memory networks (ConvLSTM), while the exploration of Long Short-Term Memory (LSTM) networks is limited per our literature review. It's important to understand that the melting of Arctic Sea ice is a multifaceted phenomenon, one that is influenced by a variety of climate and atmospheric variables. To address this complexity and enhance prediction accuracy, our study employs feature selection to encompass a wide array of variables, including sea surface temperature, surface pressure, total precipitation, latent heat, and sensible heat. Additionally, many predictive models underestimate the speed of melting in

the Arctic and limit the range of techniques used to enhance their LSTM models.

According to the Sea Ice Outlook by Sea Ice Prediction network (SIPN), before 2020, there were more submissions with statistical approaches and dynamical models in comparison to those utilizing machine learning [7]. In a pioneering study, Chi and Kim [8] utilized machine learning and developed a 1-month forecast LSTM model to predict sea ice concentration (SIC). Their model predicted monthly SIC with less than 9% average monthly prediction errors. Achieving accurate predictions during the melting season (June to October) was more challenging compared to the freezing season (November to May). This heightened difficulty was attributed to the impact of sea ice thinning [8].

This study branches off our previous research [9], in which we focused on exploring the combination of image and numerical data when constructing predictive models for the Arctic Sea Ice Extent (SIE). In this study, we aim to further explore the potential of LSTM models using exclusively numerical data. This approach allows us to employ a variety of techniques to optimize the accuracy of our sequential LSTM model and make informed comparisons on the most effective methods for SIE prediction.

This research not only enhances our comprehension of sea ice dynamics, but also equips us with the essential knowledge pertaining to LSTM models and their application in preserving and protecting the Earth's delicate environmental balance.

2. Model

2.1. Long Short-Term Memory Networks (LSTM)

LSTM network is a special type of Recurrent Neural Network (RNN) which was developed to overcome the limitations of traditional RNNs, particularly their inability to capture and maintain long term dependencies [10]. Consequently, LSTMs are exceptionally well-suited for tasks involving the classification and prediction of time-series data.

A typical LSTM structure is composed of distinct memory blocks known as cells. These cells act as transporters of essential information through the network's chain and thus effectively serve as the network's memory. This architecture enables the transfer of information from earlier time steps to subsequent ones, generating the capacity for long-term memory retention. An LSTM cell consists of the cell state (C), the hidden state (h) and three gates: the forget gate, the input gate, and the output gate. The cell state is the long-term memory of the LSTM whereas the hidden state is the short-term representation or output of the current time step. The gates regulate the flow of information by determining what to forget, update, and output, enabling the network to effectively capture and utilize long-term dependencies in sequential data.

Activation functions play a crucial role in controlling the flow of information through LSTM networks and learning complex patterns in sequential data. They allow LSTM to selectively update and utilize information in the cell state based on

the input, previous states, and the network's learned parameters. For the gate and cell state operations, LSTM networks traditionally employ the sigmoid and hyperbolic tangent (tanH) activation functions due to specific gate-like characteristics and suitability of these activations for handling sequential data.

The sigmoid function confines data to a range of 0 to 1; when the input is large and positive, the output approaches 1 else the output approaches 0. The tanH function maintains values between −1 and 1 to regulate data flow; when the input is a large positive number, the tanH function approaches 1, whereas it approaches −1 for large negative numbers.

Referring to Figure 2, the forget gate decides which information to keep or forget from the cell state. To make this decision, it multiplies the current input (x_t) and the previous hidden state (h_{t-1}) by weight matrices and adds a bias. The sigmoid function is then applied and produces a vector with values from 0 to 1, corresponding to each number present in the cell state. An output of 0 indicates forgetting and an output of 1 indicates keeping that piece of information. The vector output (f_t) produced by the sigmoid function is then multiplied[1] to the previous cell state (C_{t-1}) so that knowledge no longer needed is forgotten.

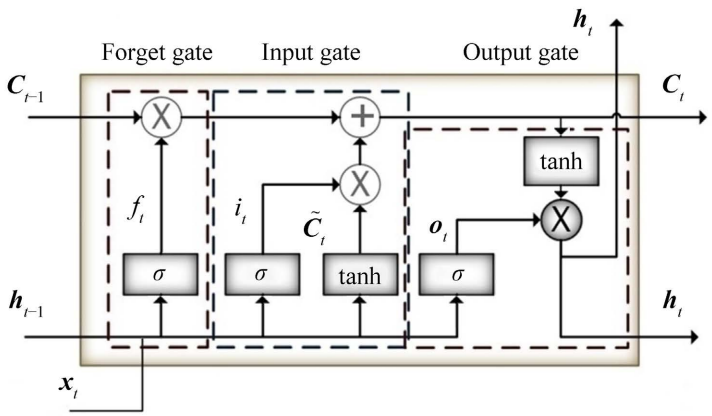

Figure 2. Basic Structure of an LSTM unit.

Then, the input gate determines the addition of new information in the current cell state. It begins by subjecting the current input (x_t) and the previous hidden state (h_{t-1}) to a sigmoid function, producing values between 0 and 1 and thus creating a regulatory filter (i_t). This process is similar to the one of the forget gate. Moreover, the current input (x_t) and the previous hidden state (h_{t-1}) also pass through a tanH function, producing a new cell state $\tilde{C}_t$. Finally, $\tilde{C}_t$. and the value of the regulatory filter (i_t) are multiplied, and this information is added to the current cell state.

The final gate, known as the output gate, utilizes relevant information from the forget and input gates to decide whether to update and store new state data. The

[1]Element-wise multiplication: this type of multiplication takes in two vectors (or more generally two matrices) of the same dimensions and returns a vector (or matrix) of the multiplied corresponding elements.

previous cell state (C_{t-1}) goes through the tanH function and the result is multiplied by the output of the sigmoid function (o_t). The resulting vector is the new hidden state (h_t) that is output by the LSTM for the current time step.

The final gate, known as the output gate, utilizes relevant information from the forget and input gates to decide whether to update and store new state data. The previous cell state (C_{t-1}) goes through the tanH function and the result is multiplied by the output of the sigmoid function (o_t). The resulting vector is the new hidden state (h_t) that is output by the LSTM for the current time step.

For our research we implemented our LSTM model with TensorFlow, a flexible and high performing end-to-end machine learning platform [11].

2.2. Research Dataset

The SIE data used for our model was obtained from the National Snow and Ice Data Centre (NSIDC) and consisted of monthly numerical data from 1989 to 2022 [12]. NSIDC monitors and provides data on various aspects of Earth's cryosphere, including sea ice, glaciers, and snow cover. They use a variety of methods, such as satellite observations and ground-based measurements, to track and analyze changes and other characteristics of sea ice. The SIE data was derived from satellite sensors that measured the presence of ice cover over large areas. From there, the satellite measurements were combined and processed to calculate the monthly extent of sea ice coverage in the Arctic available in numerical form.

To enrich our dataset with variables influencing changes in sea ice, we used ERA5 data from Copernicus [13]. ERA5 is a reanalysis dataset that offers consistent historical weather variable estimates. ERA5 integrates data from observation satellites, ground-based weather stations, and ocean buoys to create a comprehensive dataset. ERA5 data is processed by organizing it into a grid that covers the entire Earth. This grid structure process makes the data easily accessible and usable for various applications, as it provides a structured and consistent format for the data, simplifying data analysis and interpretation.

Specifically, from ERA5 we extracted the following predictor variables: sea surface temperature (SST), total precipitation (TP), surface pressure (SP), surface latent heat flux (SLHF), and surface sensible heat flux (SSHF). The selection of these variables was based on analysis from Chen [3] that confirmed correlation with SIE. For example, SST directly affects sea ice dynamics, as higher temperatures can promote further ice melt. TP, in the form of snowfall, can slow down the rate of melting through insulating ice from atmospheric heat. However, precipitation in the form of rainfall can accelerate ice melt. SP can modulate wind patterns, thus indirectly affecting sea ice. Finally, SLHF and SSHF impact the energy balance of the ice, influencing freezing and melting rate.

2.3. Data Pre-Processing

To enhance our data integrity, we applied multiple data preprocessing techniques, including normalization, dimensionality adjustments, and dataset merging.

Notably, we addressed spaces in target labels and handled missing values in the data. Specifically, we handled such values with polynomial interpolation, which is regarded as a good method for interpolation of monthly and yearly climate elements [14]. This meticulous handling is crucial to ensure the robustness and accuracy of our dataset for our machine learning model.

After verifying our SIE data, we shifted our focus onto the ERA5 climate variable data. ERA5 data is stored in specialized formats like NetCDF (Network Common Data Form) or GRIB (Gridded Binary). As TensorFlow doesn't provide built-in functionality to read ERA5 climate variable data, we extracted the data from the climate dataset. After loading the ERA5 data, we first analyzed the data structure and dimensions. Originally, our dataset contained the following dimensions: longitude, latitude, EXPVER, and time. EXPVER is a dimension representing the version or ensemble member of the experiment in climate or weather modeling. Our dataset included 2 EXPVER values, indicating that there were two ensemble members or experiment versions in the subset of the NetCDF file. In addition to EXPVER, each climate variable in the NetCDF file had its own 2D array of values representing the data for each combination of longitude and latitude. Our first step was to split the dataset based on its EXPVER. We then conducted spatial aggregation to lower data dimensions based on latitude and longitude. This effectively transformed our dataset, enabling conversion to CSV format for merging with our numerical SIE data. SIE was loaded as our target variable, while ERA5 data served as our control variables.

2.4. Data Analysis

In the initial stages of our data analysis, we recognized the potential impact of extreme values, or outliers, on the performance and robustness of our LSTM model.

The distribution of data plays a significant role in outlier detection. The Shapiro-Wilk test results showed that all our variables except for SP did not follow a normal distribution. Thus, for outlier detection we decided to use the Interquartile Range (IQR) as it doesn't depend on a specific distribution and uses percentiles.

To determine the presence of outliers, we performed the following calculations: first, we solved for the first quartile value (Q1) and the third quartile (Q3). Subsequently, the IQR was derived by subtracting Q1 from Q3. We utilized the following formulas to find our outliers: Q1 − 1.5 (IQR) for the lower bounds and Q3 + 1.5 (IQR) for upper bounds. Our findings are displayed in the box plots of **Figure 3** below: all our variables, except SIE and TP, contained outliers.

Our next goal was to check for the relation between our target variable, SIE, and climate variables. We utilized Spearman's rank correlation coefficient as this method is robust against the presence of outliers and doesn't require normal distributional shape when calculating relationships between variables [15]. A Spearman coefficient is abbreviated as ρ and is calculated with the ranks of the values of each variable, instead of the actual values [16]. The value of the Spearman coefficient ranges between −1 and +1. A value of −1 indicates strong negative

correlation, meaning that as one variable increases, the other tends to decrease. A value 0 means no association, and a value of +1 indicates strong positive correlation, meaning that as values of one variable increase, the values of the other variable also increase.

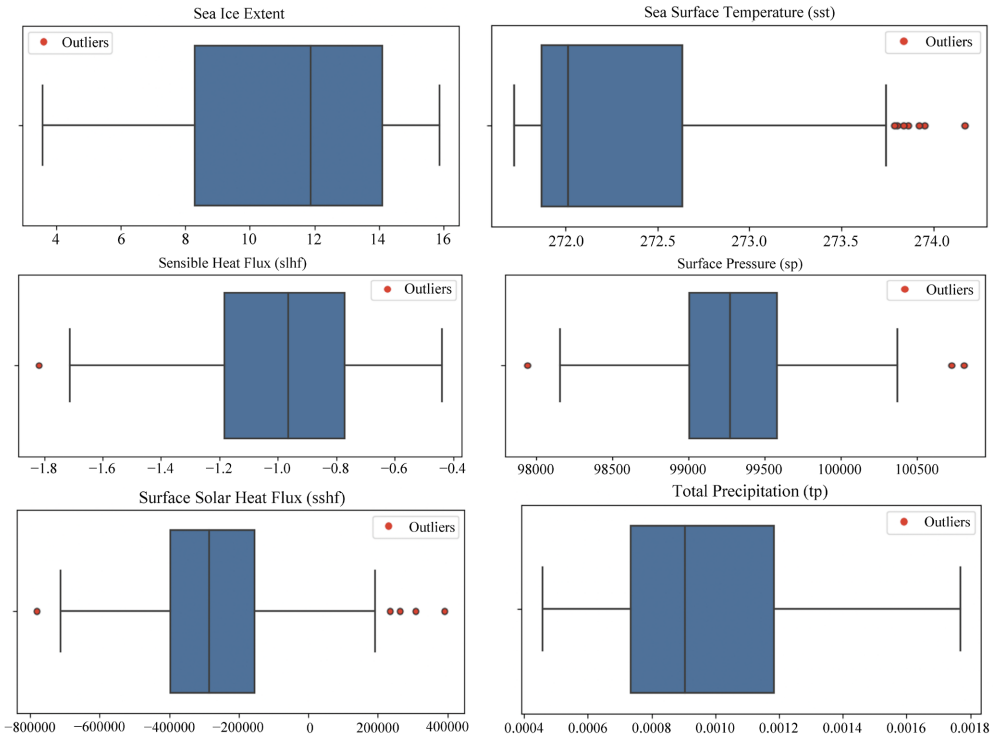

Figure 3. Box plots for the SIE and predictor variables.

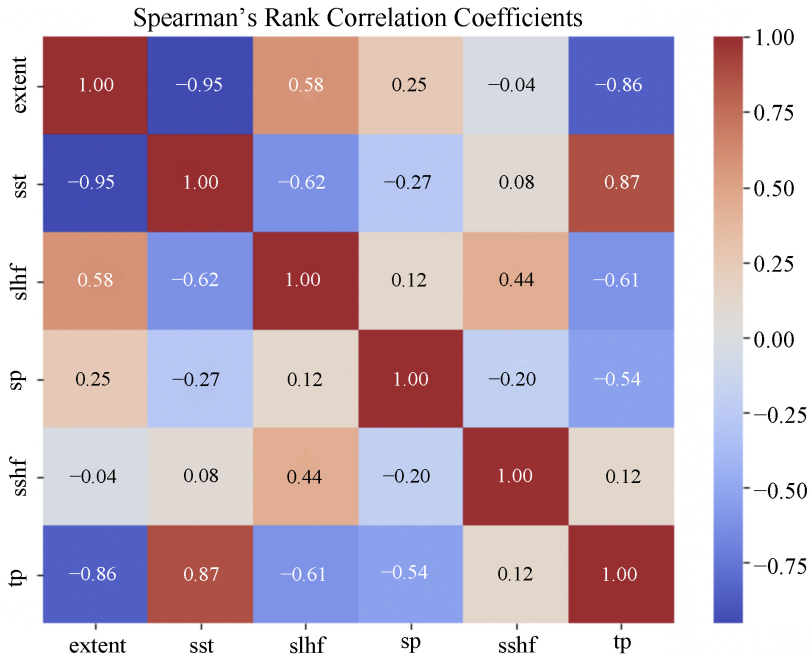

Figure 4. Heat map for SIE and predictor variables.

As shown in Figure 4, the analysis indicates a strong negative correlation between SST and TP and SIE (correlation coefficients of −0.95 and −0.86 respectively), indicating that as SST and TP increase, SIE decreases. There is a moderate positive monotonic relationship between SLHF and SIE (correlation coefficient of 0.58), and a weak positive monotonic relationship for SP and SIE (correlation coefficient of 0.25). For SSHF and SIE, the Spearman's rank correlation coefficient is −0.04, suggesting a very weak or negligible monotonic relationship between the variables.

Next, we utilized the ExtraTrees regressor, a meta-estimator that employs randomized decision trees on dataset subsets to analyze feature contributions for target variable predictions. This method assesses both linear and nonlinear relationships and enhances predictive accuracy as it averages results and helps to mitigate overfitting risks. The output is a numerical value of each variable's importance, where all values sum to 1. The results were as follows:

- Sea Surface Temperature (SST) importance: 0.600
- Latent Heat Flux (SLHF) importance: 0.075
- Surface Pressure (SP) importance: 0.012
- Sensible Heat Flux (SSHF) importance: 0.025
- Total Precipitation (TP) importance: 0.288

Our SST variable retained its position as the most influential feature for predicting SIE, despite changes in the importance of other variables. For this analysis, our SP variable resulted as the least impactful for our prediction model, followed closely by SSHF.

Combining the results from the Spearman's rank correlation coefficients with ones from Extra Trees Regressor feature importance can provide valuable insights into the relationships between variables and confirm their importance in predicting SIE (see Table 1):

Table 1. Spearman's correlation and extra trees regressor results.

	Spearman's Correlation	Extra Trees Regressor
Sea Surface Temperature (SST)	−0.95: Strong negative correlation. This suggests that as SST increases, SIE decreases.	0.600: High importance, the most influential feature in our dataset.
Surface Latent Heat Flux (SLHF)	0.58: Moderate positive correlation, indicating that higher SLHF is associated with increased SIE.	0.075: Moderate importance contributing to the model's predictive power.
Surface Temperature (SP)	0.25 Positive correlation but weaker than SST and SLHF.	0.012: Very low importance but still contributes to the model
Surface Sensible Heat Flux (SSHF)	−0.04: Very weak negative correlation, indicating a negligible relationship.	0.025: Low importance. It contributes to the model but is not a major influencer.
Total Precipitation (TP)	−0.86: Strong negative correlation, indicating that higher precipitation is associated with reduced SIE.	0.288: Significant importance, the second most influential feature in our dataset.

As a summary:

- SST and TP stand out as critical factors influencing SIE, supported by both strong Spearman's correlations and high ExtraTrees importance scores.
- SLHF also plays a significant role, having a moderate Spearman's correlation and a moderate importance score.
- SP and SSHF have weak Spearman's correlations and relatively low importance scores, suggesting they contribute to the prediction but have less impact compared to other variables.

3. Building the LSTM Model

In our initial study [9] we developed a LSTM model tailored for handling numerical climate data. The model demonstrated satisfactory performance with R^2 score of 95.48, MAE of 0.58, MSE of 0.49 and RMSE of 0.70, showcasing its efficacy in SIE prediction. Experimentation showed that the best performance was achieved with 78 LSTM units and 64 batch size.

In our current research, we systematically investigate several other crucial dimensions of our model, in addition to LSTM units, batch sizes, and adjustable learning rate to evaluate the impact on its predictive capabilities.

3.1. Model Optimization

Our LSTM model's optimization journey encompasses the key areas described below.

- **Extended Historical Context**

We explore the impact of varying historical context by creating data sequences with 6 and 12 past months. Chen [3] showed that when using multiple linear regression (MLR), support vector regression (SVR), and random forest regression (RFR) techniques, a 12-month sea ice data resulted in a better forecast for SIE than using 6-month data. As shown further below, our analysis provides valuable insights into the significance of historical data length for accurate predictions in LSTM models.

- **Hyperparameter Optimization**

The number of LSTM units determines the complexity of the model. More units allow the model to learn more complex patterns in the data but may also lead to overfitting if the model becomes too complex for the given dataset. We also experiment with batch size, an important hyper-parameter in deep learning that represents the number of samples that work through the network on each iteration before updating the internal model parameters. The batch size has a direct impact on the accuracy and computational efficiency of the training process. Large batch sizes can provide a smoother gradient signal and lead to faster training times. However, it has been observed that using larger batches lead to the decline of a model's ability to generalize [17]. On the other hand, it has been demonstrated that smaller batches allow a significantly smaller memory footprint and introduce noise in the optimization, leading in improved generalization performance [18].

This yields a more stable and reliable training.

- **Data Scaling**

As presented earlier, our data doesn't follow a Normal (Gaussian) distribution and contains outliers. Standardization is suitable for data that has a normal distribution, whereas Normalization (MinMax) is preferred when data does not follow a normal distribution but is sensitive to outliers. Robust scaling is used when data has many outliers [19]-[21]. For our data scaling, we experiment with all three scalers.

- **Temporal Embedding**

To capture the temporal patterns inherent in climate data, we test the inclusion of the month variable into our data series. We aim to refine the model's understanding of seasonal variations, a critical factor in climate prediction. We also experiment with the inclusion of past SIE information to check how this could help enhance our model's ability to capture temporal dependencies in the data.

- **Advanced Training Techniques**

In addition to the adjustable learning rate, we experiment with model checkpoint and early stopping techniques. Model checkpoint ensures the preservation of the best-performing model during training, while early stopping prevents overfitting by stopping training when validation loss starts increasing, leading to a more generalizable and robust LSTM model. Overfitting occurs when a model learns to fit the training data very closely, but it performs poorly on unseen or test data.

- **Regularization Techniques**

Regularization techniques are also used to prevent overfitting and improve the generalization of a model. For example, the Elastic Net regularization combines the advantages of both L1 and L2 regularization. The L1 encourages the model to focus on a subset of the most relevant features, effectively ignoring irrelevant or redundant ones, thus reducing the complexity of the model. On the other hand, the L2 regularization prevents any one feature from dominating the model's predictions and improves the model's stability [22] [23].

- **Activation Functions**

As mentioned earlier, LSTM networks traditionally employ the sigmoid function for the gate activation and tanH function for the cell state operations, due to specific gate-like characteristics and suitability of these activations for handling sequential data. However, both sigmoid and tanH functions present the vanishing gradient problem. The Rectified Linear Unit (ReLU) activation was introduced to solve this problem. It is linear for all values greater than or equal to zero, which enables it to learn faster. However, any input value that is below or equal to zero causes the ReLU to produce an output of zero, leading to zero gradients during backpropagation, a problem known as the dying ReLU problem. The Parametric ReLU (PReLU) improves ReLU by adding a slope for negative values, thus allowing the algorithm to learn from negative data regions.

Experimenting with activation functions won't be a focus in this study.

Nevertheless, due to having both positive and negative values in our dataset, we checked our best performing model with the Parametric ReLU (PReLU) for the LSTM layer.

The primary goal is to build a model that predicts a continuous numerical value, that is of the SIE. Since the focus is on producing an output that directly represents the SIE value, we will only use the default linear function for the output layer.

3.2. Evaluation Metrics

In our study, we conduct a comprehensive evaluation of our model's performance using the following standard deterministic accuracy metrics: Mean Absolute Error (MAE), R-squared value (R^2), Mean Square Error (MSE), and Root Mean Square Error (RMSE).

MAE measures the average absolute differences between predicted values and actual values in a dataset. R^2 represents the proportion of the variance in the dependent variable that is predictable from the independent variables in a regression model. MSE measures the average of the squared differences between predicted values and actual values in a regression problem. Finally, RMSE provides a more interpretable measure by taking the square root of the MSE.

As our data analysis demonstrates, our data contains outliers. Since the squaring of errors enforces a higher importance on outliers, MSE and RMSE may be less robust for our model compared to MAE [24].

3.3. Data Preparation

Data splitting into training, validation, and test sets is essential for assessing and fine-tuning machine learning models, preventing overfitting, choosing the best model, and ensuring they can make accurate predictions on new, unseen data. It's a fundamental practice for robust and reliable model development. Since our data is a time series, we used a temporal split: allocated the earlier portion of the data for training (80%), a middle portion for validation (10%), and the most recent portion for testing (10%).

We organize our data in sequences, with each sequence containing a series of past data points of the SIE and predicting variables, and the corresponding target representing the next data point for the SIE. The inclusion of the SIE from the previous month was important to account for temporal dependencies in the data.

For example, in the case of sequences considering three lagged time steps, the structure of the inputs and outputs would be as follows:

X [[SIE(t-3), var1(t-3), var2(t-3) …]
[SIE(t-2), var1(t-2), var2(t-2) …]
[SIE(t-1), var1(t-1), var2(t-1) …]]

Y SIE(t)

4. LSTM Model

In this section we present the steps we take for exploring performance

improvements to our supervised regression learning problem: predicting the SIE at the current month (t) using the SIE and environmental variables values from prior time steps. We will only highlight the configurations that yield the best results. Our baseline, stateful LSTM model was built with two dense layers and utilized the default activation methods. The first is the LSTM layer, using sigmoid and tanH activations, followed by an output layer with a single unit that uses the linear activation function.

```
model = Sequential()
model.add(LSTM(lstm_units, input_shape=(n_past_months, n_features)))
model.add(Dense(1))    # Output layer for regression
model.compile(loss='mae', optimizer='adam')
```

We also included the ReduceLROnPlateau scheduler, a learning rate adjustment technique that operates by monitoring a specific metric, in our case the validation loss. If the validation loss does not improve after the number of epochs defined by the "patience" variable, the scheduler reduces the learning rate to 10% of its current value (factor parameter). The learning rate will never go below min_lr.

```
ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=20, verbose=1, min_lr=1e-6)
```

The first improvement we wanted to make was to train our model with an extended historical context of 6 and 12 months (*i.e.*, 6 and 12 lagged time steps). We also experimented with different scalers to optimize our model's performance by tailoring the data preprocessing to our specific dataset. The Robust Scaler scales the data in such a way that makes it more robust to outliers by using the median and interquartile range (IQR) rather than the mean and standard deviation. Standard and MinMax scalers are both not robust to outliers.

Table 2 below shows the best performance results for 6 and 12 lagged time steps for all scalers we tested. We obtained best MAE for 12 lagged time steps, indicating better predictive performance. This observation is consistent with the findings of Chen et all for their support vector, random forest, and multiple linear regression models. Out of the two scalers that are not robust to outliers, Standard scaler had superior results. Moreover, considering all performance indicators, scaling with Robust scaler provided better results than with MinMax scaler. Thus, we decided to check performance with both Standard and Robust scalers for the next tests.

Table 2. Best performance results with learning rate adjustment technique.

Data Scaler	Months	LSTM Units	Batch Size	MAE	R^2	MSE	RMSE
Standard	6	64	32	0.485	0.98	0.235	0.484
Standard	12	96	64	0.334	0.99	0.181	0.425
MinMax	6	78	64	0.743	0.95	0.575	0.758
MinMax	12	78	32	0.467	0.98	0.273	0.522
Robust	6	64	32	0.571	0.97	0.330	0.574
Robust	12	64	64	0.480	0.98	0.232	0.482

LSTM networks are well-suited for sequence prediction tasks where the order and timing of events matter. To enhance the model's performance and provide additional context regarding how data changes over time, we decided to conduct further experiments with the incorporation of seasonal information. Seasonal information can offer valuable insights into recurring patterns within the data.

To achieve this, we included month information using cyclical encoding. Cyclical encoding is a methodology used to represent cyclical patterns, such as those related to time of day, months, or seasons. This technique ensures that the values "wrap around" in a circular fashion, preserving the inherent cyclical nature of the data. By encoding the months in this manner, we aimed to provide our LSTM model with a deeper understanding of how seasonal changes influence the data, ultimately testing its impact on prediction accuracy. Cyclical encoding can be done using the following transformations [25].

$$x_{\sin} = \sin\left(2\pi * x / \max(x)\right) \tag{1}$$

$$x_{\cos} = \cos\left(2\pi * x / \max(x)\right) \tag{2}$$

For our monthly data, let $M \equiv \text{month}$, where $M = 1, \cdots, 12$, and

$$M_{\sin} = \sin\left(\pi * M / 6\right) \tag{3}$$

$$M_{\cos} = \cos\left(\pi * M / 6\right) \tag{4}$$

For a 12-month historical context, our model performed best with 96 LSTM units and 64 batch size (see **Figure 5** and **Figure 6**). Specifically, we obtained the following results:

Standard Scaler: MAE: 0.399 || R^2: 0.993 || MSE: 0.086 || RMSE: 0.294

Robust Scaler: MAE: 0.230 || R^2: 0.993 || MSE: 0.088 || RMSE: 0.296

We also experimented with the inclusion of yearly information. However, this did not improve our results and therefore we excluded this from our dataset.

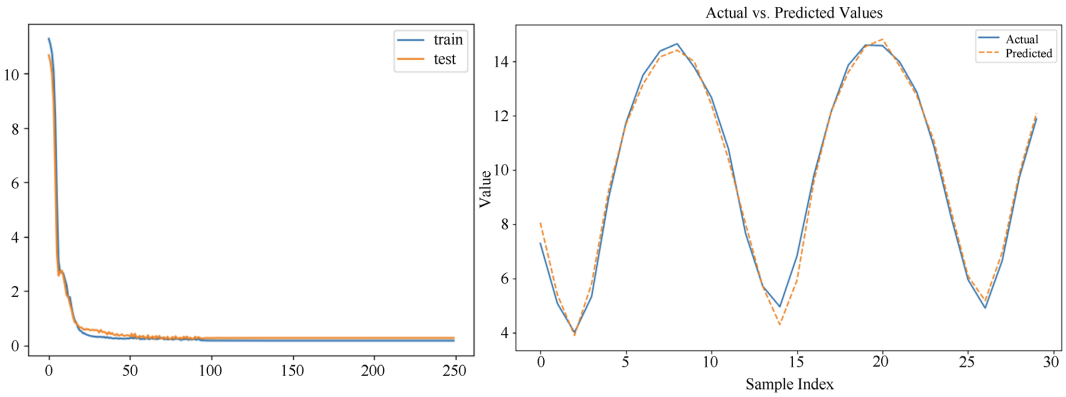

Figure 5. Inclusion of Month in Dataset, performance for 12-month historical context—Standard Scaler.

The next step was to utilize the early stopping technique. Early stopping monitors the model's performance on a validation dataset during training and stops training when the performance starts to degrade, preventing overfitting. If the validation loss

stops improving or begins to worsen, considering the "patience", training is halted. Therefore, it helps in preventing the model from learning noise in the training data, saving time and resources, and often results in better generalization to unseen data.

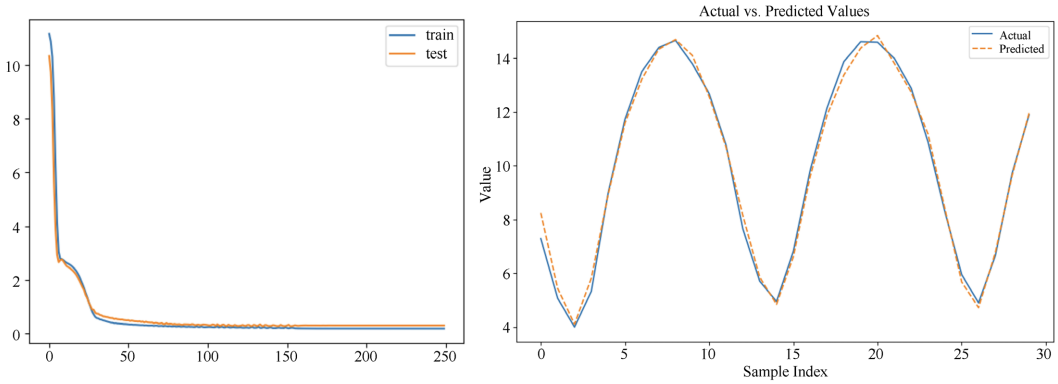

Figure 6. Inclusion of Month in Dataset, performance for 12-month historical context—Robust Scaler.

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, re-
store_best_weights=True)
history = model.fit(train_X, train_y, epochs=250, batch_size=64,
                    validation_data=(validation_X, validation_y),
                    callbacks=[early_stopping, reduce_lr], verbose=0,
                    shuffle=False)
```

We tested three different values for patience: 10, 15, and 20 (see Table 3). For Standard scaler we obtained better performance results for patience equal to 20. For Robust scaler, the performance results for 15 and 20 patience were very close. There was a slight improvement on MAE with patience 15, which indicated that our model was able to achieve better performance in terms of absolute prediction accuracy on the validation data.

Thus, for our subsequent testing we decided to use patience 20 for Standard scaler and patience 15 for Robust scaler.

Table 3. Inclusion of early stopping—Performance results for patience 10, 15, 20.

Data Scaler	Early Stopping Patience	MAE	R^2	MSE	RMSE
Standard	10	0.272	0.991	0.109	0.33
Standard	15	0.305	0.989	0.133	0.364
Standard	20	0.260	0.992	0.095	0.309
Robust	10	0.396	0.979	0.267	0.517
Robust	15	0.236	0.994	0.076	0.275
Robust	20	0.242	0.994	0.074	0.272

The next step was to further explore our model's performance with the

inclusion of the model checkpoint technique. Model checkpoint saves the model's weights and architecture periodically during training, typically after each epoch or after a certain number of training steps. This feature enables the ability to resume training from a specific point if the training process is interrupted or to select the best performing model for later use. Therefore, this provides access to the model's best state during training, even if the training process is interrupted. Additionally, it allows for the evaluation of the model's performance on different datasets or at different points in time.

Setting the monitor equal to "val_loss" allowed the callback to save the weights when it observed an improvement in validation loss. The callback only saved the model weights if the validation loss had improved compared to the previous best value. We then loaded the best performing weights to make the predictions.

The order in which we utilized early_stopping, reduce_lr, and checkpoint callbacks is important. First, we wanted to monitor the model's performance with early_stopping. If training continues without early stopping, our model could consider reducing the learning rate with reduce_lr, when needed. Finally, we placed checkpoint last to save the model's weights after the other callbacks run.

```
checkpoint=ModelCheckpoint(filepath='best_model_weights.h5', monitor='val_loss', save_best_only=True)
history = model.fit(train_X, train_y, epochs=250, batch_size=64,
                    validation_data=(validation_X, validation_y),
                    callbacks=[early_stopping, reduce_lr, checkpoint], verbose=0, shuffle=False)
# Load the best model weights
model.load_weights('best_model_weights.h5')
```

The combination of model checkpoint and early stopping enabled us to balance two important but somewhat conflicting objectives: preventing overfitting and ensuring that we save the best model. Therefore, we expected that it would not only ensure the stability and reliability of our model but would also optimize the use of computational resources, as unnecessary iterations would be prevented.

The performance results for standard scaler deteriorated when we combined early stopping and model checkpoint. Both the MAE and MSE increased by approximately 8.46% and 25.26% respectively. In contrast, the results with Robust scaler improved; MAE and MSE decreased by approximately 10% and 6.7%. The difference in model performance with different scalers alongside model checkpoint and early stopping is something worth investigating more.

Standard Scaler: MAE: 0.282 || R^2: 0.99 || MSE: 0.119 || RMSE: 0.345

Robust Scaler: MAE: 0.212 || R^2: 0.994 || MSE: 0.069 || RMSE: 0.263

Overall, with Robust scaler, combined with early stopping and model checkpoint, the model's performance results prove that the predictions are very close to the actual values (MAE: 0.212, MSE: 0.069) and the model captures almost all the underlying patterns in the data (R^2: 0.994). The low RMSE (0.263) also indicates a tight fit of our model to the data. Figure 7 below shows the train and test loss and

the actual versus predicted values SIE graph. Table 4 also shows the actual (test dataset) versus predicted SIE values.

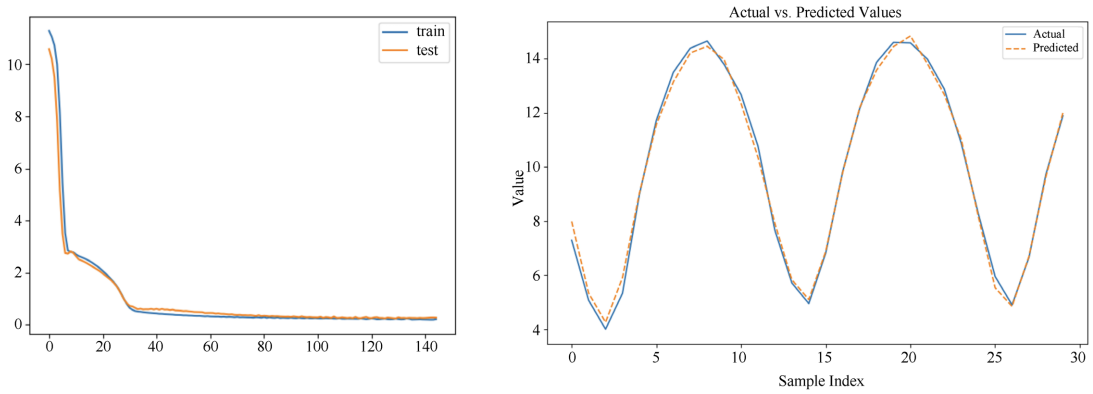

Figure 7. Robust scaler: Inclusion of Month in Dataset—Performance for 12 month window, Early Stopping and Model checkpoint.

Table 4. Actual versus Predicted values for Robust scaler.

Actual	Predicted	Actual	Predicted
7.29	7.98	6.82	6.88
5.07	5.34	9.83	9.79
4	4.26	12.15	12.19
5.33	5.91	13.87	13.58
8.99	9.04	14.61	14.45
11.73	11.57	14.59	14.84
13.5	13.14	13.99	13.82
14.39	14.21	12.88	12.68
14.66	14.46	10.88	11.03
13.79	13.97	8.29	8.19
12.68	12.35	5.95	5.53
10.77	10.35	4.9	4.85
7.65	7.91	6.66	6.70
5.71	5.83	9.73	9.64
4.95	5.10	11.89	12.00
7.29	7.98	6.82	6.88

Since the robust scaler is designed to handle outliers and due to the existence of outliers in our data, we believe that the use of Robust scaler resulted in a more stable performance. Model checkpoint saves the model weights when the validation metric improves. Based on our observations, we believe that Robust Scaler leads to more stable training, thus making easier for early stopping to identify the optimal stopping point and for model checkpoint to capture more robust model

snapshots.

Similar to past research results [8], the model with Robust scaler also achieved more accurate predictions during freezing season compared to the ones during melting season, with July having the least accurate prediction. Table 5 below includes the performance results for each season and Table 6 includes the performance results per month.

Table 5. Evaluation Metrics for Melting and Freezing season (96 LSTM units, 64 batch size, Robust scaler, adjustable learning rate, early stopping with patience 15 and model checkpoint).

Melting Season (June to October)		Freezing Season (November to May)	
MAE	0.254	MAE	0.174
R^2	0.975	R^2	0.988
MSE	0.103	MSE	0.039
RMSE	0.321	RMSE	0.199

Table 6. Evaluation Metrics per month (96 LSTM units, 64 batch size, Robust scaler, adjustable learning rate, early stopping with patience 15 and model checkpoint).

1989-2002	MAE	MSE
Jul	0.349	0.185
Jan	0.323	0.115
Jun	0.281	0.106
Aug	0.266	0.097
May	0.265	0.086
Oct	0.228	0.074
Mar	0.224	0.051
Apr	0.175	0.031
Feb	0.168	0.030
Sept	0.153	0.028
Dec	0.102	0.013
Nov	0.057	0.004

As a next step, we wanted to test the impact of adding additional past SIE as a feature in our model's input sequence. Our hypothesis was that inclusion of more past SIE values could enhance capturing temporal dependencies in the data. If there were patterns or trends in the historical SIE that influence future values, the model could learn from these patterns. Also, if there was a sudden increase or decrease in SIE, it might affect the subsequent month's extent, and the model could learn to recognize such patterns.

Specifically, we first tried including the SIE of the corresponding month in the previous year, which did not improve the model's performance. So, instead, we

incorporated a simple linear regression simulated SIE of the previous year.

We build a linear regression model to predict SIE based on past SIE. We then extracted the slope coefficient which we multiplied to the SIE of the corresponding month of the previous year. This new feature captures the influence of past SIE on current SIE, as learned by the model.

$$\mathrm{SIE}_{\mathrm{SIM}}(t) = \mathrm{LinRegSlopeCoeff} * \mathrm{SIE}(t-12) \tag{5}$$

where $t = M\ of\ Y$

$$\mathrm{SIE}_{\mathrm{SIM}}(t) = \mathrm{Slope} * \mathrm{SIE}(t-12) \tag{6}$$

where $t = M\ of\ Y$.

As shown in Figure 8 and Table 7, MAE improved by 9.91% and MSE by 20.3%. Moreover, the predictions for melting season improved and outperformed the predictions for freezing season, overcoming the limitations of the previous model configuration and those in past research results that we are aware of.

Robust Scaler: MAE: 0.191 || R^2: 0.996 || MSE: 0.055 || RMSE: 0.234

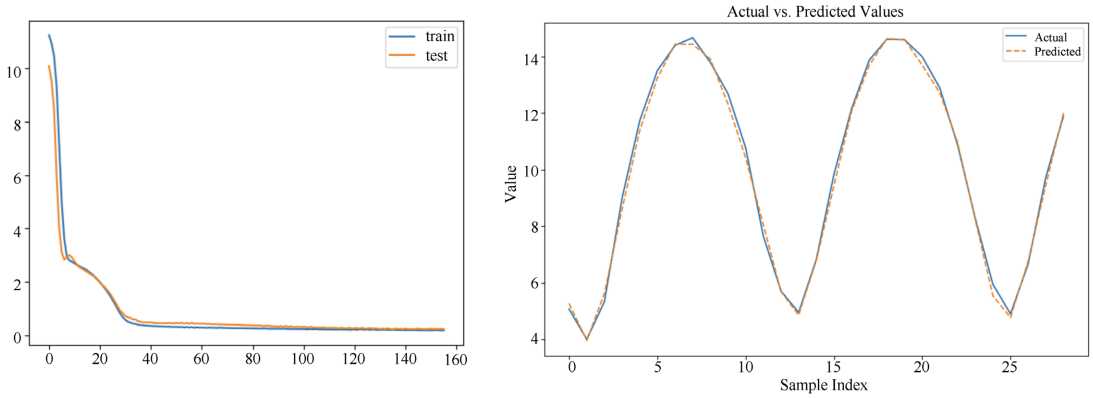

Figure 8. Robust scaler: Inclusion of SIE in Dataset—Performance for 12 month window, early stopping and model checkpoint.

Table 7. Evaluation Metrics for Melting and Freezing season with inclusion of past year SIE (96 LSTM units, 64 batch size, Robust scaler, adjustable learning rate, early stopping with patience 15 and model checkpoint).

Melting Season (June to October)		Freezing Season (November to May)	
MAE	0.170	MAE	0.207
R^2	0.989	R^2	0.981
MSE	0.048	MSE	0.060
RMSE	0.220	RMSE	0.245

We also tested the impact on performance when including the L1 and L2 regularization. L1 regularization (Lasso) adds a penalty term based on the absolute values of the model's coefficients and can help with feature selection by driving some coefficients to exactly zero. L2 regularization (Ridge) adds a penalty based

on the squared values of the coefficients, which helps prevent overfitting and can improve generalization. We iteratively tested different combinations of L1 and L2 values and identified that the best performing values for our model were 0.001 and 0.001 respectively. The inclusion of L1 and L2 did not have any positive impact on our model's performance. Although on average MSE remained the same, MAE slightly deteriorated by 3%. Since L1 did not lead to an improvement in the model performance, it may suggest that most of our features are contributing to our model and thus there aren't many irrelevant features to eliminate. The fact that L2 did not help improve performance may suggest that early stopping is effectively controlling our model's complexity leading to no overfitting.

Lastly, we tested PReLU activation function both at the LSTM layer and as a separate layer. Both had a significant negative impact on the performance. In the first case, MAE increased by approximately 120% and in the second it increased by 50%.

5. Conclusions

In this research, we optimized an LSTM model for the prediction of Arctic Sea ice extent on a monthly scale by carefully choosing several key hyperparameters and then properly tuning them through feature re-engineering and the inclusion of advanced training techniques. The development of accurate Arctic SIE predictive models is crucial due to the rapid decline of sea ice in recent years and provides essential insights for understanding and mitigating the impact of climate change on polar regions.

In the initial stages of our data analysis, we focused on identifying outliers present in our data and exploring the relationship between SIE and our five climate variables: sea surface temperature, surface pressure, total precipitation, latent heat, and sensible heat. Our experiments revealed the presence of outliers in the data, influencing our choice of evaluation metrics and data scaling techniques. Feature importance analysis also confirmed that all our features significantly contributed to SIE, a finding further supported by the results of L1 Regularization.

Our original model configuration included 96 LSTM units, 64 batch size, one-month time lag, and an adjustable learning rate. Our model delivered predictions with a MAE of 0.480, R^2 of 0.98 and MSE of 0.232.

We trained our model with an extended historical context of 6 and 12 months (*i.e.*, 6 and 12 lagged time steps). We experimented with three different scalers: Standard, MinMax, and Robust Scaler. The model's performance with Standard and Robust Scaler was best for the 12 lagged time steps, with Standard outperforming Robust. We incorporated month information with cyclical encoding to introduce temporal and seasonal information. After these enhancements, the model's performance with Robust Scaler was superior in terms of MAE. The configuration with Robust scaling continued to outperform after the incorporation of advanced training techniques like early stopping and model checkpoint, resulting in more accurate predictions for the freezing season (November to May) compared

to the melting season (June to October).

However, our results changed when we included one more feature: specifically, a linear regression simulated SIE that accounted for the influence of past SIE on current SIE. This increased the accuracy of predictions for the melting season compared to the freezing season and overcame limitations in not only model configuration but also those in past research. Further, our observations following L2 Regularization suggested to us that our early stopping was effectively controlling our model's complexity and prevented overfitting.

Overall, following meticulous hyperparameter tuning, feature re-engineering and inclusion of the advanced training techniques of early stopping and model checkpoint, MAE improved by 60% (0.191) and MSE by 76% (0.055). R^2 also reached an impressive value of 0.996, underscoring the model's remarkable ability to explain and account for nearly all the variance in the data.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] The National Oceanic and Atmospheric Administration (2018) Arctic Report Card: Update for 2018. https://arctic.noaa.gov/report-card/report-card-2018/executive-summary-5/

[2] The National Aeronautics and Space Administration (2022) Arctic Sea Ice Minimum Extent Vital Signs—Climate Change: Vital Signs of the Planet. NASA Climate Change. https://climate.nasa.gov/vital-signs/arctic-sea-ice/

[3] Chen, S., Li, K., Fu, H., Wu, Y.C. and Huang, Y. (2023) Sea Ice Extent Prediction with Machine Learning Methods and Subregional Analysis in the Arctic. *Atmosphere*, **14**, Article 1023.

[4] Serreze, M.C. and Meier, W.N. (2019) The Arctic's Sea Ice Cover: Trends, Variability, Predictability, and Comparisons to the Antarctic. *Annals of the New York Academy of Sciences*, **1436**, 36-53.

[5] NASA Earth Observatory (2021) World of Change: Arctic Sea Ice. Earth Observatory. https://earthobservatory.nasa.gov/world-of-change/sea-ice-arctic

[6] Arctic Monitoring and Assessment Programme (2017) Snow, Water, Ice and Permafrost in the Arctic (SWIPA). Assessment Summary for Policy-Makers. https://swipa.amap.no

[7] Mu, B., Luo, X., Yuan, S. and Liang, X. (2023) IceTFT v1.0.0: Interpretable Long-Term Prediction of Arctic Sea Ice Extent with Deep Learning. *Geoscientific Model Development*, **16**, 4677-4697.

[8] Chi, J. and Kim, H. (2017) Prediction of Arctic Sea Ice Concentration Using a Fully Data Driven Deep Neural Network. *Remote Sensing*, **9**, Article 1305.

[9] Pegkou Christofi, V., Li, A. and Lin, A. (2024) Wavelet Based Multiscale Deep Learning Algorithms for Arctic Sea Ice Melting Prediction. *Proceedings of the* 2024 6*th International Symposium on Signal Processing Systems*, Xi'an, 22-24 March 2024, 29-39. https://doi.org/10.1145/3665053.3665054

[10] Sherstinsky, A. (2020) Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *Physica D: Nonlinear Phenomena*, **404**,

Article 132306. https://doi.org/10.1016/j.physd.2019.132306

[11] TensorFlow. https://www.tensorflow.org/

[12] Fetterer, F., Knowles, K., Meier, W.N., Savoie, M. and Windnagel, A.K. (2017) Sea Ice Index, Version 3 [Data Set]. Boulder, Colorado USA. National Snow and Ice Data Center.

[13] Copernicus Climate Change Service (2023) Reanalysis ERA5 Single Levels Monthly Means. https://cds.climate.copernicus.eu/datasets/reanalysis-era5-single-levels-monthly-means?tab=download

[14] Sluiter, R. (2009) Interpolation Methods for Climate Data: Literature Review. KNMI (Royal Netherlands Meteorological Institute), R&D Information and Observation Technology. https://uaf-snap.org/wp-content/uploads/2020/08/Interpolation_methods_for_climate_data.pdf

[15] National Center for Atmospheric Research, Dennis, S. and National Center for Atmospheric Research Staff (2014) The Climate Data Guide: Statistical & Diagnostic Methods Overview. https://climatedataguide.ucar.edu/climate-tools/statistical-diagnostic-methods-overview

[16] Schober, P., Boer, C. and Schwarte, L.A. (2018) Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia & Analgesia*, **126**, 1763-1768. https://doi.org/10.1213/ane.0000000000002864

[17] Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M. and Tang, P. (2017) On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.

[18] Masters, D. and Luschi, C. (2018) Revisiting Small Batch Training for Deep Neural Networks.

[19] Huilgol, P. (2024) Analytics Vidhya. Feature Transformation and Scaling Techniques to Boost Your Model Performance. https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/

[20] de Dieu Nyandwi, J. (2021) The Ultimate and Practical Guide on Feature Scaling. https://jeande.medium.com/the-ultimate-and-practical-guide-on-feature-scaling-d03fbe2cb25e

[21] Scikit-Learn Developers (2023) Plot Different Scalers on the Same Data. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#plot-all-scaling-quantile-transformer-section

[22] Neptune Labs (2023) Fighting Overfitting with L1 or L2 Regularization: Which One Is Better? https://neptune.ai/blog/fighting-overfitting-with-l1-or-l2-regularization

[23] Van Otten, N. (2023) L1 And L2 Regularization Explained, When to Use Them & Practical How to Examples. https://spotintelligence.com/2023/05/26/l1-l2-regularization/

[24] Towards Data Science: Comparing Robustness of MAE, MSE, and RMSE. https://towardsdatascience.com/comparing-robustness-of-mae-mse-and-rmse-6d69da870828

[25] Van Wyk, A. (2022) Encoding Cyclical Features for Deep Learning. https://www.kaggle.com/code/avanwyk/encoding-cyclical-features-for-deep-learning