Eric's Angels: Written Evaluation

Duru Huseyni (zhuseyni), Jinho Lee (jlee812), Kamya Raman (kraman4)

Github: https://github.com/jinholee17/sea-ice-predictor

Introduction

With rising global temperatures and climate change becoming a threat to arctic ecosystems, we wanted to predict the future of sea ice melting/surface area decrease to better understand the climate emergency. Sea ice has been melting at unprecedented rates, and this should be better understood in order to create targeted solutions.

For this, we reimplemented a paper published in the Association for Computer Machinery that uses Long Short-Term Memory (LSTM) networks for sequential data analysis, and Convolutional Neural Networks (CNN) for image recognition. We implemented 2 models to predict ice melting levels. Our first model is a simple 1D LSTM on time series data, and our second model is a 2D CNN-LSTM network that uses discrete wavelet transforms as a preprocessing step, which increased our accuracy significantly.

Data & Preprocessing

1D LSTM MODEL:
We used 2 different types of time series data and combined them into one file to train our model. The sea ice extent (SIE) data we used were retrieved from the National Snow and Ice Data Center (NSIDC) [18] and consisted of monthly numerical data from 1989 to 2022. To expand upon our data and include variables that added complexity to changes in sea ice, we used the ERA5 data from Copernicus Climate Data Store, and specifically collected time series data for variables sea surface temperature, total precipitation, surface pressure, surface latent heat flux, and surface sensible heat flux. The ERA5 data was provided in a 2D array of values representing the data for each combination of longitude and latitude, so we computed the spatial mean of these to reduce the 2D data into a time series.

Once we had processed the two datasets separately, we merged them on (month, year) tuples and constructed a dataset with sea ice extent & area, as well as the ERA5 variables. To normalize the data, we applied Min-Max scaling to all numeric features, and stored both the normalized and original values, plus min/max for inverse transforms. Lastly, we prepared the data for our LSTM model by forming sequences of 12 months for training, where each sequence had shape (seq_length, num_features), and the target value was the sea ice extent for the next month. This finalized data was stored in a .pkl file.

Figure 1: Table displaying our data distribution and preprocessing values for the 1D LSTM model

Category	Metric	Value
NSIDC Data	CSV files processed	12
	NSIDC date range	1989–2022

	NSIDC entries loaded	408
ERA5 Extraction	Variables processed	5
	Total timesteps per variable	556
Final Merged Dataset	Matching (Year, Month) pairs	408
	Feature columns used	7
	LSTM input shape	(396, 12, 7)

2D CNN/LSTM MODEL:

For the 2D CNN-LSTM model, we used two complementary spatial-temporal datasets to train our model. First, we obtained sea ice concentration satellite imagery from the Copernicus Marine Environment Monitoring Service (CMEMS), which provided daily 2D data grids representing the percentage of sea ice concentration at each latitude-longitude point across the Arctic. These images were processed into 128 × 128 grids for each day, and a total of 375 frames from 1989 to 2022 were extracted and preprocessed.

To complement the sea ice data with additional environmental predictors, we incorporated reanalysis variables from the ERA5 dataset, also provided by the Copernicus Climate Data Store. These included sea surface temperature, surface pressure, total precipitation, surface latent heat flux, and surface sensible heat flux. Like the sea ice data, these were also provided as 2D spatial grids, and were remapped to match the spatial resolution of the sea ice images using bilinear interpolation.

After processing both datasets, we combined the five ERA5 variables with the sea ice concentration data to create a unified tensor of shape (6, 375, 128, 128), where each of the six channels corresponds to one variable across time. We then formed input-output training sequences using a sliding window of 5 timesteps (5 days), resulting in a training dataset where each input has shape (5, 128, 128, 1) and the corresponding target is the sea ice concentration for the following day. In total, we created 370 training examples. Both input and output data were normalized to the range [-1, 1] to match the tanh activation function used in the model.

Finally, as a final preprocessing step, we performed spatial and temporal discrete wavelet transforms to extract distinctive features from our data using the PyWavelets package. For the spatial discrete wavelet transformation, we separated the data of each 2D spatial grid into four components: approximation, horizontal detail, vertical detail, and diagonal detail. Our temporal discrete wavelet transformation was calculated across one unit on the spatial grid across all time, and was separated into approximation and detail components. The results of these two transforms were then concatenated and stored as tensors for use with the CNN-LSTM model.

Figure 2: Table displaying our data distribution and preprocessing values for the 2D CNN-LSTM model

Category	Metric	Value

CMEMS	Satellite Images Processed	375
	NSIDC date range	1989–2022
ERA5 Extraction	Variables processed	5
	Total timesteps per variable	556
Final Merged Dataset	Matching (Year, Month) pairs	375
	Feature columns used	5
	CNN input shape	(128, 128, 1)

Model Architecture

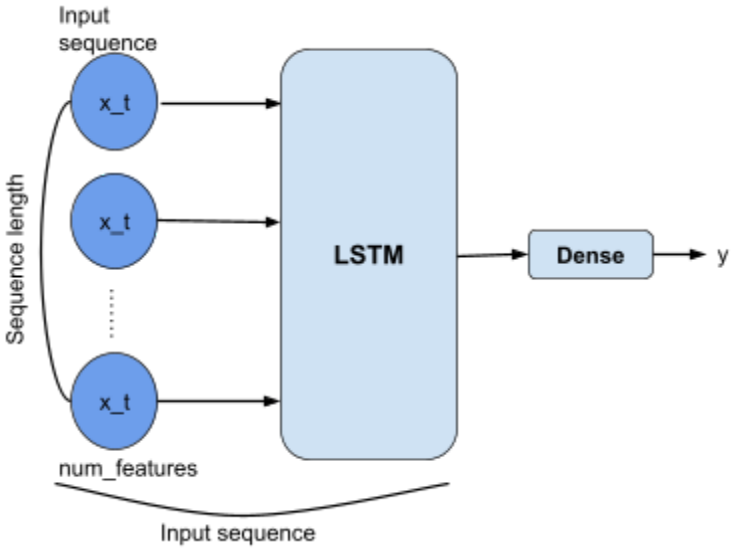
1D LSTM MODEL:

LSTMs are composed of different memory blocks, called cells, and introduce three gate mechanisms responsible for controlling the flow of information: the forget gate, the input gate, and the output gate. Our 1D model used the sigmoid and the hyperbolic tangent (tanh) as activation functions, where the sigmoid assists the network when deciding whether to keep or discard data, and tanh regulates the values flowing through the network and maintains the values between -1 and 1.

HYPERPARAMETERS:

Batch size	16
Number of neurons	64
Activation	Activation: tanh Recurrent activation: sigmoid
Learning rate	0.01, with ReduceLROnPlateau scheduler
Epochs	200
Loss Function	Mean Absolute Error (MAE)

Figure 3: Visualization of the LSTM model



For our training strategy in our LSTM, we split the data chronologically into training (70%), validation (15%), and testing (15%), and tested with and without schedular strategies (ReduceLROnPlateau, factor=0.1, patience=20, min_lr=1e-6) as well as different number of batch sizes (16, 32, 64). We used Mean Absolute

Error as our loss function mainly because the paper we adapted the model from used MAE. However, we also experimented with MSE as our loss function and that indeed provided better results.

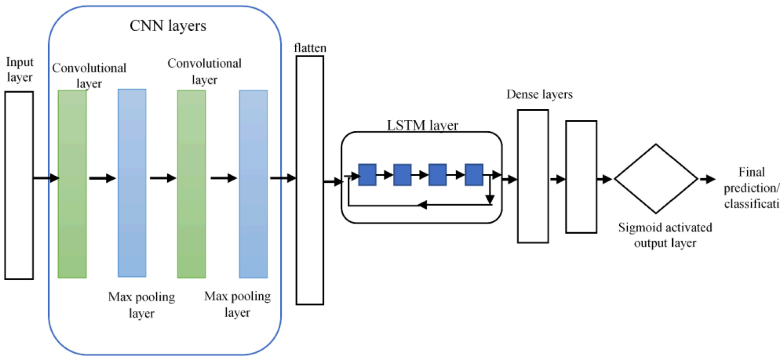
2D CNN-LSTM MODEL:

Our 2D CNN-LSTM model architecture is designed to process sequences of spatial sea ice and environmental variable grids. The model takes input tensors of shape (6, 128, 128, 1), where 6 represents the combined channels from ERA5 variables and sea ice concentration. The first stage consists of two 3D convolutional layers with 32 and 64 filters, each using a 3×3×3 kernel and tanh activation, to extract spatial-temporal features while preserving resolution through "same" padding. After each convolutional layer, a 3D max pooling operation is applied with a pool size of (1, 2, 2), reducing spatial dimensions while maintaining the temporal dimension. The extracted features are then flattened and passed through a fully connected dense layer with 1024 units, followed by a second dense layer that projects the output back to a 128×128 grid. The final output layer reshapes the output into a (128, 128, 1) map representing the predicted sea ice concentration for the next time step. The model was trained using the Adam optimizer, minimizing mean squared error (MSE) loss, with mean absolute error (MAE) tracked as an additional evaluation metric.

HYPERPARAMETERS:

Batch size	16
Number of neurons	64
Optimizer	Adam
Activation	Activation: tanh
Learning rate	0.001
Epochs	10
Loss Function	Mean Squared Error (MSE)
Metrics Function	Mean Absolute Error (MAE)

Figure 4: Visualization of a CNN-LSTM model



Results

1D LSTM MODEL:

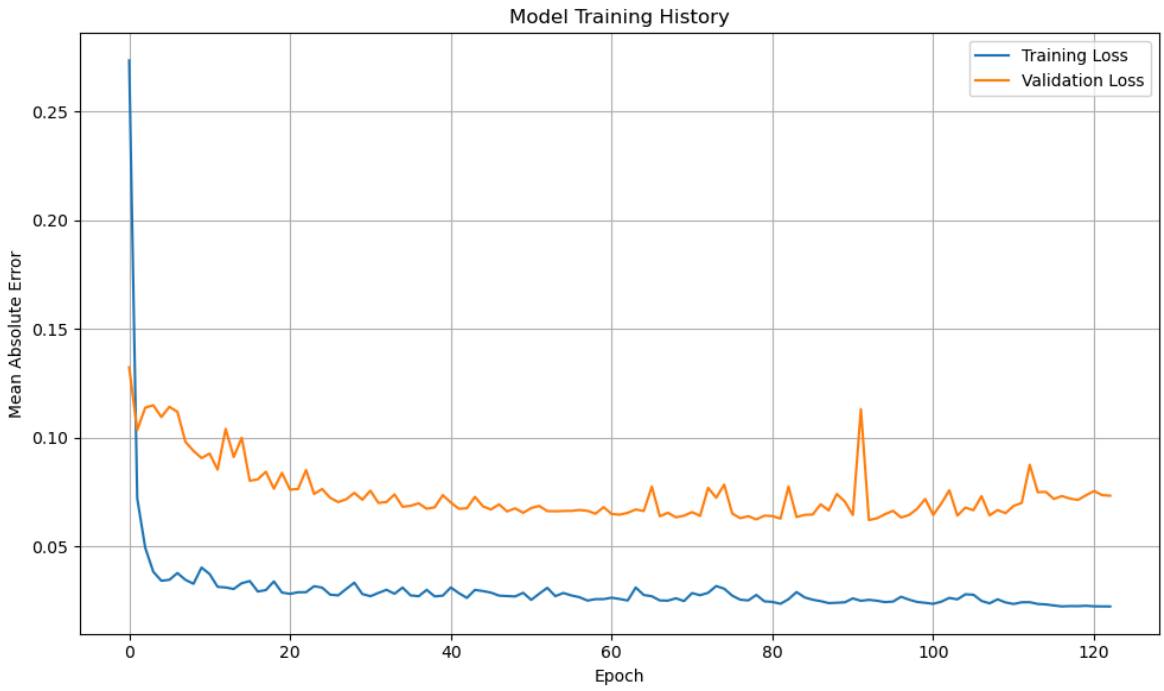
After experimenting with different combinations of batch sizes, neurons, learning rates, and LR schedulers, below are the top 10 results after hyperparameter tuning. It should be noted that the paper we are reimplementing used MAE as their loss function; however, in our model, MSE gave the best result! Best model used 64 neurons, learning rate 0.01, MSE loss, batch size 16, and learning rate scheduler enabled. MSE loss slightly edged out MAE for best MAE in this case, even though most other top models used MAE.

Since all of our data was normalized around [-1, 1], an MAE of about 0.0292 represents about 1.46% of our data range. This is a wonderful result and indicates that our regression model fits the data extremely well.

Figure 5: Top 10 best LSTM models across 144 experiments

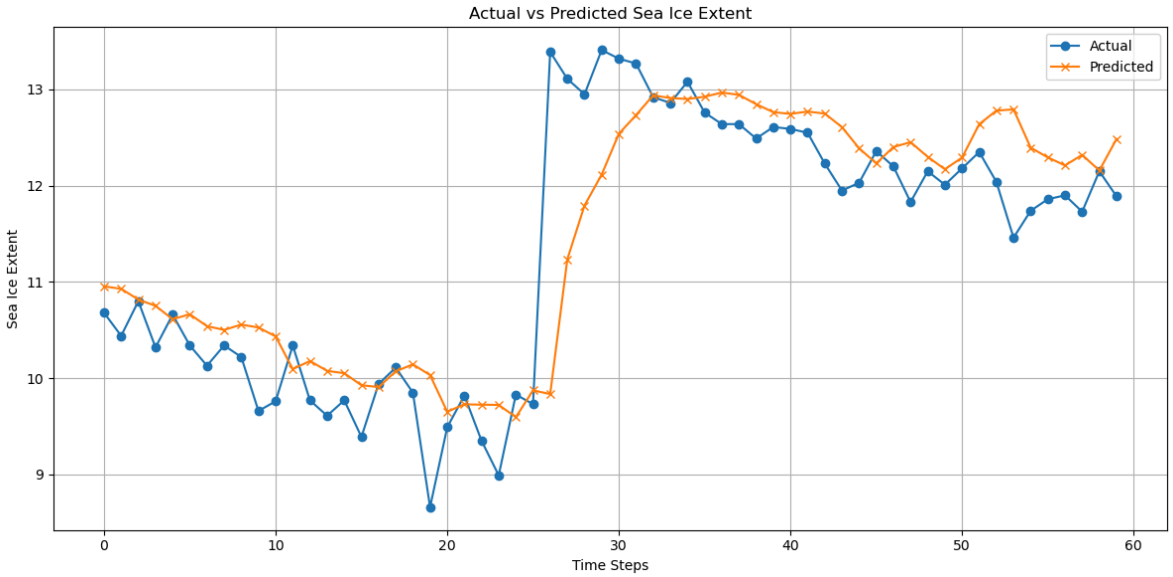
Rank	Neurons	Learning Rate	Loss Function	Batch Size	LR Scheduler	MAE	RMSE	R^2	Notes
1	64	0.01	MSE	16	True	0.0292	0.0499	0.7923	**Best overall**
2	50	0.01	MAE	32	True	0.0293	0.0487	0.8020	Highest R^2
3	100	0.01	MAE	32	False	0.0296	0.0504	0.7880	
4	64	0.01	MAE	16	True	0.0296	0.0489	0.8003	
5	32	0.01	MAE	64	True	0.0297	0.0502	0.7891	
6	100	0.01	MAE	16	True	0.0298	0.0486	0.8024	
7	50	0.01	MAE	64	False	0.0298	0.0508	0.7846	
8	50	0.01	MAE	32	False	0.0299	0.0501	0.7901	
9	100	0.01	MAE	16	False	0.0300	0.0495	0.7956	
10	50	0.01	MAE	64	True	0.0301	0.0510	0.7828	

Figure 6: Plot showing the training history of the LSTM model



From the training history plot, it can be observed that training MAE quickly drops below 0.05 within the first ~20 epochs. Validation loss decreases steadily and then stabilizes with minor fluctuations, indicating a healthy training process with no clear overfitting. This progress plateaus after ~40 epochs, showing effective learning rate reduction and early stopping.

Figure 7: Plot showing actual vs. predicted values of sea ice extent



Overall, the LSTM model was able to successfully capture general trends and seasonal patterns of sea ice extent. However, it can be seen that sharp transitions like abrupt increases or decreases are smoothed by the model, reflecting a common limitation of LSTMs in capturing sharp spikes in time series. The predictions align well with the overall trajectory, although slight temporal lags or amplitude dampening are present during steep changes. The quantitative measures on the model on the test set were as follows:

Figure 8: Quantitative evaluation of the LSTM model on the test set

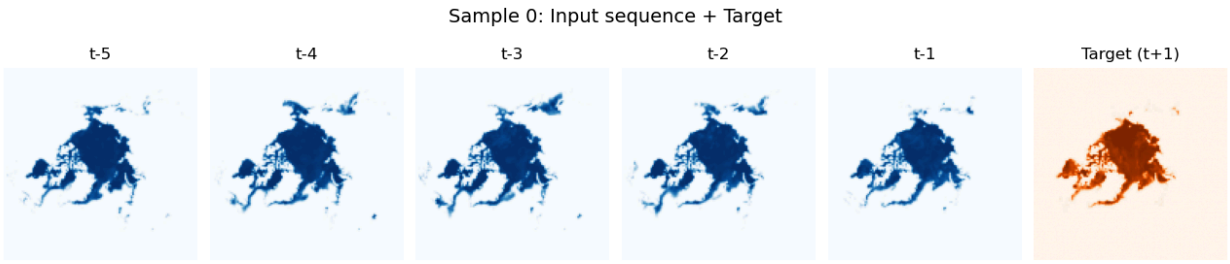
Metric	**Best Value with MAE loss**	**Best Value with MSE loss**
MAE	0.0292	0.0293
MSE	0.00249	0.00237
RMSE	0.0499	0.0487
R^2	0.7923	0.8020

These results indicate good predictive power, with over 80% of the variance explained ($R^2 = 0.802$). The low MAE and RMSE also suggest the model has accurate short-term forecasting ability. We think that the use of learning rate scheduling and early stopping played a key role in optimizing training and avoiding overfitting.

2D CNN-LSTM Model:

Our 2D CNN-LSTM model achieved strong performance in predicting sea ice concentration across the Arctic region. After 10 training epochs, the model reached a final training loss of 0.0518 and a mean absolute error (MAE) of 0.0620. On the validation set, the model achieved a loss of 0.0589 and a validation MAE of 0.0647, indicating good generalization to unseen data. These results suggest that the model effectively captures both the spatial structure and short-term temporal evolution of sea ice concentration using combined sea ice and environmental reanalysis data.

Figure 9: 5 timesteps within our 2D CNN-LSTM model followed by the predicted target timestep



To visualize the input and target data used in our 2D CNN-LSTM model, we display a sequence of six images representing five input timesteps followed by the predicted target timestep. Each image is a 128 × 128 grid showing the spatial distribution of sea ice concentration, where lighter colors indicate higher concentrations. The first five frames illustrate the model's input: a sliding window of past sea ice conditions over five consecutive days. The final image in the sequence shows the ground truth sea ice concentration for the next day, which the model is trained to predict. This side-by-side visualization highlights how the model uses spatial-temporal context to anticipate short-term changes in sea ice coverage.

Figure 10: 10 Different Hyperparameter Experiments Using Our CNN-LSTM model

Rank	Neurons	Learning Rate	Loss Function	Batch Size	MAE	MSE	Validation MAE	Validation MSE	Notes
1	64	0.001	MSE	16	0.0518	0.0620	0.0589	0.0647	**Best overall**
2	64	0.0005	MSE	16	0.0525	0.0628	0.0593	0.0652	
3	128	0.001	MSE	16	0.0541	0.0645	0.0610	0.0678	
4	32	0.001	MSE	32	0.0568	0.0682	0.0635	0.0701	
5	64	0.005	MSE	16	0.0582	0.0705	0.0651	0.0724	
6	64	0.001	MAE	16	0.0583	0.0712	0.0694	0.0756	

7	128	0.001	MSE	32	0.0585	0.0718	0.0701	0.0785	
8	32	0.001	MAE	16	0.0591	0.0701	0.079	0.0788	
9	64	0.005	MAE	16	0.0595	0.0723	0.0792	0.0801	
10	64	0.0001	MAE	32	0.0601	0.0701	0.0794	0.0803	

Across different model configurations, we observed that the 2D CNN-LSTM architecture performed consistently well, with mean absolute errors (MAE) ranging from 0.0507 to 0.0601. As with our LSTM model, our data was normalized between [-1, 1], indicating that the error was approximately in 2.5% of our full data range. The best overall performance was achieved with 64 neurons, a learning rate of 0.001, and the mean squared error (MSE) loss function, resulting in a training MAE of 0.0620 and a validation MAE of 0.0647. Models using a lower learning rate (0.0005) demonstrated slightly slower convergence but comparable final validation performance, while models with a higher learning rate (0.005) exhibited signs of instability and higher validation errors, suggesting potential overfitting or poor generalization. Increasing the number of neurons to 128 slightly increased model capacity but did not significantly improve validation metrics, indicating that a moderately sized network was sufficient for this task. Alternative loss functions like MAE also produced competitive results. Overall, these results suggest that our model architecture generalizes well across different hyperparameter settings, with performance most sensitive to learning rate and batch size tuning.

Challenges

One of the most challenging parts of this project we have encountered has been finding a pre-process the data into a usable format. For one, there are many hoops to go through to secure the data in the first place, such as submitting a request to download the data and not being able to push datasets of a large size to GitHub. After we had secured the data, we ran into an issue of the sparsity of the data that we needed to augment with other sources. For our 1D LSTM, in particular, we ended up needing to compile data from many different sources to ensure more reliable results. We ran into a few challenges when pre-processing the data, to ensure the varying data formats could all be parsed the same way. All in all, finding and working with datasets has been the most challenging part of our project.

As for acquiring and pre-processing 2D data, we similarly ran into issues. Accessing sea ice and environmental reanalysis datasets required navigating different satellite data repositories and APIs like the Copernicus Marine Toolbox and CDO. Handling authentication errors, incomplete downloads, and large amounts of data were all early obstacles in the process. Preparing the 2D data involved multiple complex steps: extracting specific variables from GRIB files, normalizing variable scales, and aligning temporal sequences across datasets with different time resolutions. Ensuring ERA5 and OSISAF data were properly matched was particularly challenging.

Specifically, when working with the CNN-LSTM model, we found that conceptually understanding wavelet transforms was a challenge in itself. Understanding how discrete wavelet transforms decompose spatial

and temporal patterns, and integrating the resulting multi-resolution features into a deep learning model, required additional research and experimentation.

A more general challenge was that balancing model complexity and training time was difficult. Large convolutional layers and dense layers introduced heavy memory and computational demands, often leading to long training times or out-of-memory issues.

Discussion & Future Work

There were many limitations in our implementation, including not being able to include all geographical areas where sea ice melting can be predicted, as we had to limit our area to Northern hemisphere specifically for feasible compute power, struggling to find *enough* data for some time series, and way too big data for other datasets, and overall having a limited amount of information on how the paper actually implemented these models. Further explorations could work on training models with data from the Southern hemisphere as well (so that we can actually talk more about penguins! 🐧), and also making sure to get the as much as data as possible, assuming we would have more compute power.

In terms of our 1D LSTM model, the main limitations included having a limited amount of data and not having enough time and resources to tune many hyperparameters. Future improvements could involve using stacked LSTM layers, attention mechanisms, or including additional predictors and variables from the Copernicus dataset, like atmospheric and oceanic indices to better capture extreme variations. Again, we realized that our LSTM model was generally good at predicting trends but lacked specificity in plotting steep transitions, so that could definitely be an area of work.

Reflection

Overall, this was definitely a challenging project and with difficulties heavily lying in data collection and the preprocessing step, as well as understanding the graphical and mathematical implementation of discrete wavelet transforms. Nevertheless, we thoroughly enjoyed the process of choosing, understanding, and applying everything we learned all throughout the semester in a final project. As a group, we learned a lot on the importance of understanding the data we are working with. This part was definitely challenging and different from all of the other assignments, where we were very gratefully given and explained the data we were working with.

Implementing two different models also allowed us to approach the data we collected in different ways, and made us think about preprocessing what we have into different shapes, depending on the model it was going to be fed into. We also had a lot of fun with the ability to compare the accuracy of our two models and evaluate their positive and negative aspects in terms of accuracy, overfitting, and computational power. We think that implementing two models helped us look at our semester of DL from a wider perspective, and reflecting on all of the different models we learned.

Lastly, we were also very passionate about the topic we chose, and therefore quite excited about our findings. It felt especially fulfilling to be able to do a project on climate change and sea ice melting

predictions, a topic that we are interested in. This experience once again showed how deeply integrated deep learning can be to real-life issues, and the depth of fields we could explore using deep learning. As three students interested in the intersection of CS with different areas, this was a great learning experience. Our team greatly enjoyed the whole process and feels like we have gained a lot from it!

Sources

Original Paper: Victoria Pegkou Christofi, Andrew Li, and Audrey Lin. 2024. Wavelet Based Multiscale Deep Learning Algorithms for Arctic Sea Ice Melting Prediction. In Proceedings of the 2024 6th International Symposium on Signal Processing Systems (SSPS '24). Association for Computing Machinery, New York, NY, USA, 29–39. https://doi.org/10.1145/3665053.3665054.