Predicting Seasonal Velocity Values at Zachariæ Isstrøm

Submitted by:

Claire Jensen

Autumn 2024

ESS-569 Machine Learning in Geoscience

University of Washington

Contents

1. Introduction

The GrIS is losing mass due, in part, to the recent acceleration of many of the outlet glaciers that line the ice sheet's margins [2]. One of the most dramatic examples of outlet glacier change has been observed at Zachariæ Isstrøm (ZI), one of only three large glaciers comprising the Northeast Greenland Ice Stream (NEGIS), a notable feature of fast-flowing ice that drains 12% of the ice sheet [3].

In addition to its important role in overall GrIS mass balance, ZI exhibits marked seasonal variability, with flow speeds accelerating in the summer months. This pronounced seasonality provides the opportunity to analyze trends in ZI's velocity fluctuations, predict future trends, and relate external factors to seasonal mass variability. Understanding how seasonality has fluctuated over time and in relation to meltwater and mélange forcing is necessary for attributing ZI's changing dynamics and accurately predicting future mass loss.

This work seeks to explore machine learning techniques for predicting velocity values at ZI. While one work uses deep learning to estimate velocity based on Sentinel-2 satellite image bands [4], there are currently no machine learning models in the literature that predict time series ice velocity values, particularly on a seasonal scale. An ideal model would be able to identify and predict seasonal trends in velocity time series data, especially annual velocity peaks in July during the melt season and preceding velocity increase in the months prior. Here I provide a Github repository that provides example code in Jupyter notebooks with all steps needed to reproduce the workflow. The notebooks use small datasets with the goal of optimizing the workflow for small data before scaling up computation time and power to a larger, more representative dataset of velocity values at ZI.

2. Data

The Greenland Ice Sheet Mapping Project (GrIMP) 6/12 Day Velocity mosaic data (NSIDC-0766) provides estimates of ice velocity on the Greenland Ice Sheet (GrIS)[1]. The data were downloaded using the National Snow and Ice Data Center's (NSIDC) API. Interferometric Synthetic Aperture Radar (SAR) measurements were taken by the Sentinel-1A and Sentinel-1B satellites on separate occasions to obtain velocity estimates. Two acquisitions of the same location can be differenced to find the change in scatter and converted to meters, therefore estimating the change in magnitude of the x and y components between two images. This change in magnitude is captured in the *vv*

band of a SAR image and is representative of the velocity estimate in meters per year.

The GrIMP project data are in accordance with FAIR principles:

Findable: Metadata, including spatial and temporal coverage, resolution, format, collection method, and a metadata document with more detail.

Accessible: The data are easily accessible through `https://nsidc.org/grimp`.

Interoperable: The metadata include FAIR language.

Reusable: The data collection methods, including calculations for satellite acquisitions, are described.

I downloaded velocity estimates in the cloud-optimized geoTIFF format from January 1, 2015, to May 16, 2024, every 6 or 12 days (depending on the frequency of overlap between Sentinel-1A and Sentinel-1B). I subsetted the data to retrieve velocity values at ZI, specifically. The netCDF that stored ZI's velocity time series was 481.7 MB. To test a range of velocity values, I picked three points: upstream of the glacier, near the terminus, and in the middle of the upstream and terminus points (see Figure 1.). These points will hereby be referred to as "upstream," "middle," and "terminus."[1] Glaciers can exhibit different seasonality, including local minima and maxima, at different points on a glacier. By picking three points that vary spatially and seasonally, I can explore the performance of ML models on the same glacier and their differences.

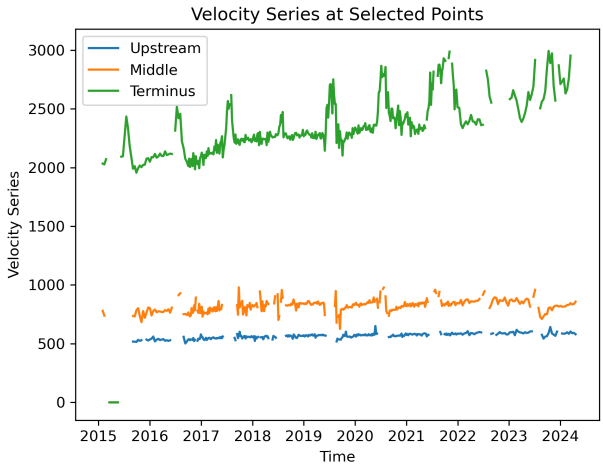

Figure 1: Raw time series data of upstream, middle, and terminus points.

[1]Points in polar stereographic coordinate system (EPSG 3413): upstream: (457794.675537, -1107613.706862), middle: (475697.184021, -1101544.002803), terminus: (488900.158671, -1098954.829477)

NaN and zero values were removed from the data (Figure 2). It is unlikely that ZI would have regions of zero or negative velocity; zero values likely represent erroneous data. Additionally, NaN values occur where no velocity estimates were able to be collected for a certain area. NaN and zero values were removed from the time series using a mask to reduce bias from the model.

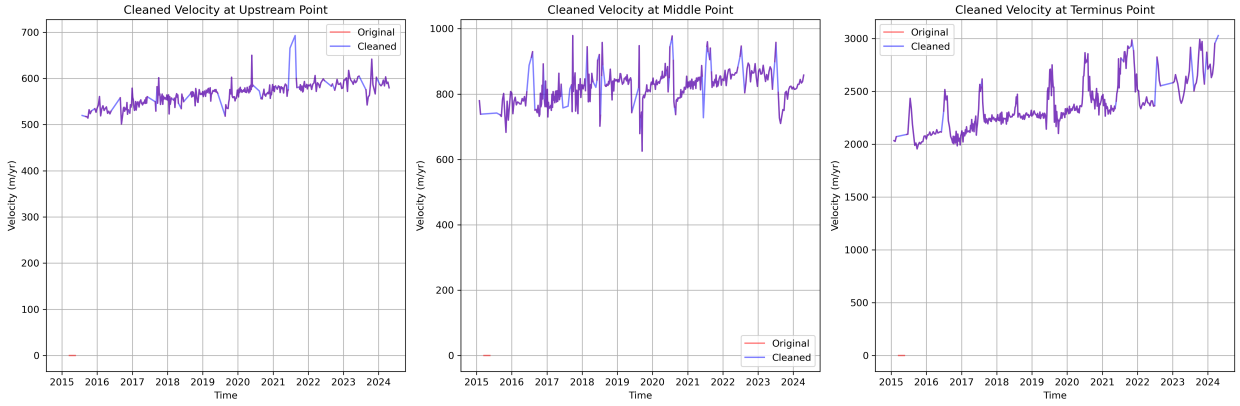

Figure 2: Cleaned velocity values of upstream, middle, and terminus points.

There were more frequent data points from 2017-2022 due to Sentinel-1A and Sentinel-1B satellites overlapping during this time frame (Fig. 3).

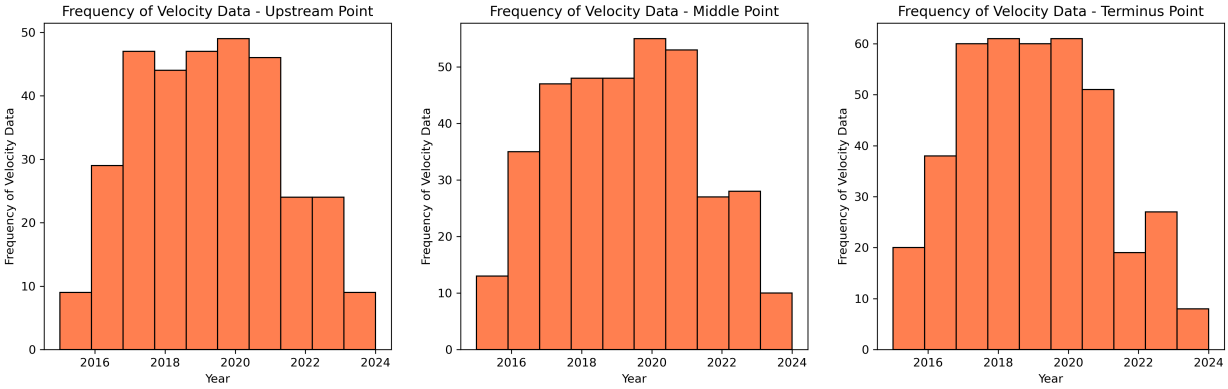

Figure 3: Cleaned velocity values of upstream, middle, and terminus points.

The upstream and terminus points' velocities were correlated (.73 and .76, respectively) with time while the middle point was not correlated (.36) (see Figure 4). Since the data have varying frequencies from 6 to 12 days, they were resampled to 6-day frequencies using interpolation in accordance with PyCaret's time series prediction library.

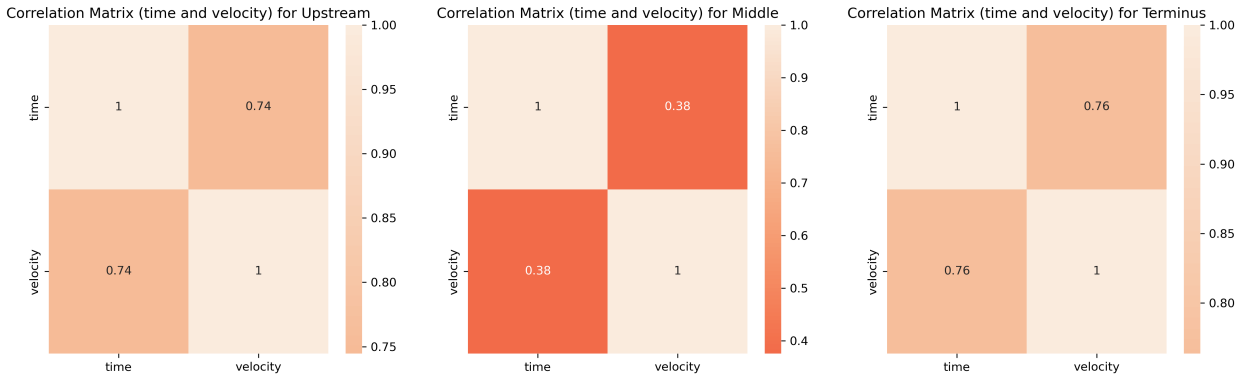

Figure 4: Cleaned velocity values of upstream, middle, and terminus points.

3. Methods and Results

I used both classic machine learning (CML) and deep learning (DL) methods for time series forecasting at ZI. See Section 3.1 for CML and Section 3.2 for DL methods. I performed this workflow on all three points, but I highlight the results for the middle point below. I chose to highlight this point because the velocity at this point is in between, both spatially and with its velocity values, the upstream and terminus points. I hypothesized that this point could be used to compare models and select a model that could perform well across data points along ZI's flowline. Additional insights about the upstream and terminus points are in the Appendix (Section 6).

3.1 Classic Machine Learning

Using the PyCaret Python package, I created a workflow for splitting the data into training and testing portions, comparing models, and tuning hyperparameters.

First, I split the AI-ready data into training and testing segments, with 75% for training and 25% for testing. Instead of specifying a percentage of the data for training and testing, I chose points before 06/01/2021 to be part of the training set and points on and after 06/01/2021 to be part of the test set. Though this creates training and testing datasets of potentially different sizes and train/test split ratios for different points on the glacier, it ensures that those points encapsulate the same dates to compare the seasonal fluctuations at each point at the same date.

I compared models using Pycaret's `compare_models` function over 10 folds, which trains and evaluates the performance of specified models on the data. I sorted by mean absolute error (MAE) and enabled cross-validation, which will split the data into smaller chunks and test each model on that chunk, k times, averaging the score of all k iterations (Fig. 13). I visually compared the

top four models' performance, particularly their abilities to identify seasonal patterns (Fig. 14). LightGBM was the only model that could accurately identify seasonal fluctuations in the dataset, though it still did not perform well.

Using a k-value of 10, the Light Gradient Boosting Machine (LightGBM) outperformed all other models using all metrics (Fig. 15; Appendix). LightGBM took the longest to run on the test data at a 0.35-second runtime, while Lars took the least amount of time at 0.25 seconds (Fig. 12; Appendix). Despite this finding, I was not concerned about the runtime for each model because my dataset is small, and a difference of 0.1 seconds between the fastest and slowest model is trivial at this scale. Despite LightGBM performing best on the training data, the ExtraTreesRegressor performed best on the test data; predicting future values is more valuable for time series forecasting, so ExtraTreesRegressor will be used moving forward.

I tuned the model using Pycaret's `tune_model` function, again optimizing using the RMSE value on 10 folds. The tuned model had a RMSE of 44.502 while the base model had a RMSE of 44.186. Since the base model outperformed the tuned model, I used the base model for predictions.

3.1.1 CML Results

I tested the model on the test data to compare to the ground truth (Fig. 5; Appendix). I found that the model captured seasonal fluctuations well, depicting the expected peak velocity in July and buildup in the preceding months. However, the predicted velocity values did not reach the extent of the local maxima and minima; instead, the model predicted smaller peaks than the ground truth. The MAE was 29.0988 and the root mean squared error (RMSE) was 39.9788. Both values indicate poor agreement with the ground truth data.

3.2 Deep Learning

I used the Keras Python package to construct two deep learning models for time series forecasting and compared these models to the functionality of Masked Encoder-based Universal Time Series Forecasting Transformer (Moirai) [5].

I split the data into 30% training and 70% testing datasets. I tested these data on a long short-term memory (LSTM) model and a convolutional neural network (CNN). A CNN learns by adjusting kernels within the layers of its architecture, optimizing for the data it is training on. A LSTM uses input, forget, and output gates to create a simulated memory unit based on the size of the `look_back` variable - the determinant of how much of the input data is used to predict the

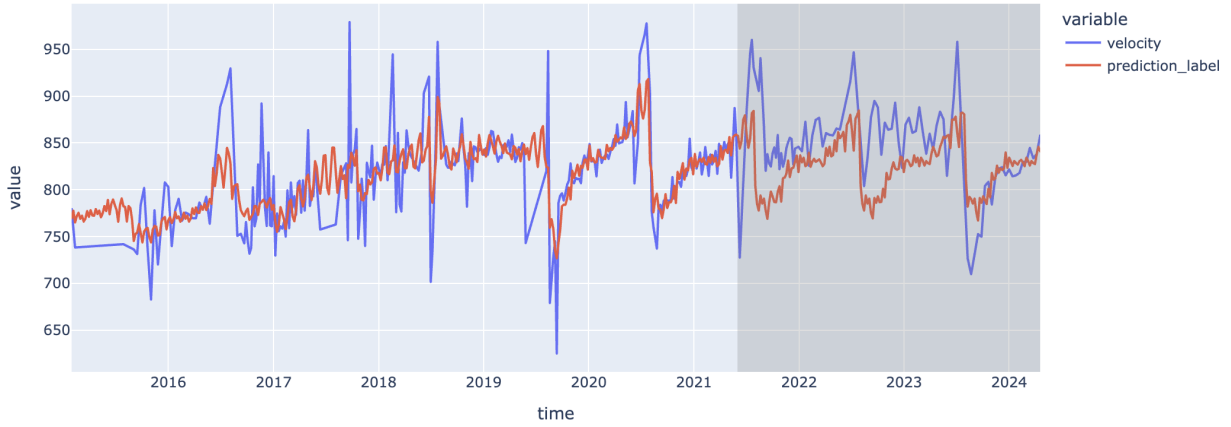

Figure 5: Predictions of velocity values on the test set of the middle point using the ExtraTreesRegressor. The ground truth velocity is blue, while the predicted velocity values are red. The test set of the data is highlighted in gray.

next value.

The CNN and LSTM were trained over 500 epochs with an Adam optimizer, a batch size of 32, and mean squared error (MSE) loss. The LSTM consisted of seven layers: three LSTM layers, three dropout layers with a rate of 0.2, and one dense layer (Fig. **??**). The CNN consisted of five layers: one convolution layer, one max pooling layer, one flattening layer, and two dense layers - one with a size of 50 and a rectified linear unit activation function and the other with a size of one (Fig. 6b).

I also tested the performance of Moirai, a self-described "universal" time series forecasting large language model (LLM) constructed by Salesforce and trained on billions of data points. Moirai seeks to close gaps in time series forecasting by learning the frequencies of test datasets using automatically varying patch sizes (a patch being a series of consecutive time points) - larger patches for higher frequency data. These patches are more representative than single data points that are fed into other models; Moirai can learn from slices of the time series. Moirai-moe, the second version of Moirai, uses a decoder-only architecture that is able to run faster than Moirai and in parallel. The model also uses a mixture-of-experts (MOE) instead of the original patch layers. MOE, instead of labeling a patch with an arbitrary frequency, is able to route the patch to the best-suited expert, which can handle differing frequencies in the same patch.

I tested Moirai and Moirai-moe with a prediction length of 20 (120 days), a context length (similar to LSTM's look back) of 30 (180 days), and a patch size of `auto`. I chose the `base` model

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 5, 100)	40,800
dropout_3 (Dropout)	(None, 5, 100)	0
lstm_4 (LSTM)	(None, 5, 100)	80,400
dropout_4 (Dropout)	(None, 5, 100)	0
lstm_5 (LSTM)	(None, 50)	30,200
dropout_5 (Dropout)	(None, 50)	0
dense_1 (Dense)	(None, 1)	51

(a) The LSTM.

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 4, 64)	192
max_pooling1d (MaxPooling1D)	(None, 2, 64)	0
flatten (Flatten)	(None, 128)	0
dense_2 (Dense)	(None, 50)	6,450
dense_3 (Dense)	(None, 1)	51

(b) The CNN.

Figure 6: Summaries of the LSTM (6a) and CNN (6b) models.

as opposed to the `small` or `large` ones.

3.2.1 DL Results

While the CNN and LSTM were both able to capture seasonal fluctuations of ZI's velocity in their training and testing predictions, their RMSE values were high. The LSTM had a train score of 35.98 RMSE and a test score of 44.89 RMSE, while the CNN had a train score of 813.48 RMSE and a test score of 834.30 RMSE. The LSTM outperformed the CNN by far, though its RMSE was still high.

The 12-month rolling average of the predictions and ground truth (Fig. 8) shows more agreement with each model, suggesting that the high variability of 6-day data might be disadvantageous to evaluating the performance of a model.

Moirai predicted values with a MAE of 67.302 and a RMSE of 85.669. Figure 17 (Appendix) depicts six predictions from Moirai based on the prediction and context lengths, with a focus on the predictions in Figure 9.

Moirai-moe predicted values with a MAE of 38.950 and a RMSE of 47.958, outperforming Moirai significantly. Figure 18 (Appendix) depicts six predictions from Moirai-moe based on the prediction and context lengths, with a focus on the predictions in Figure 10.

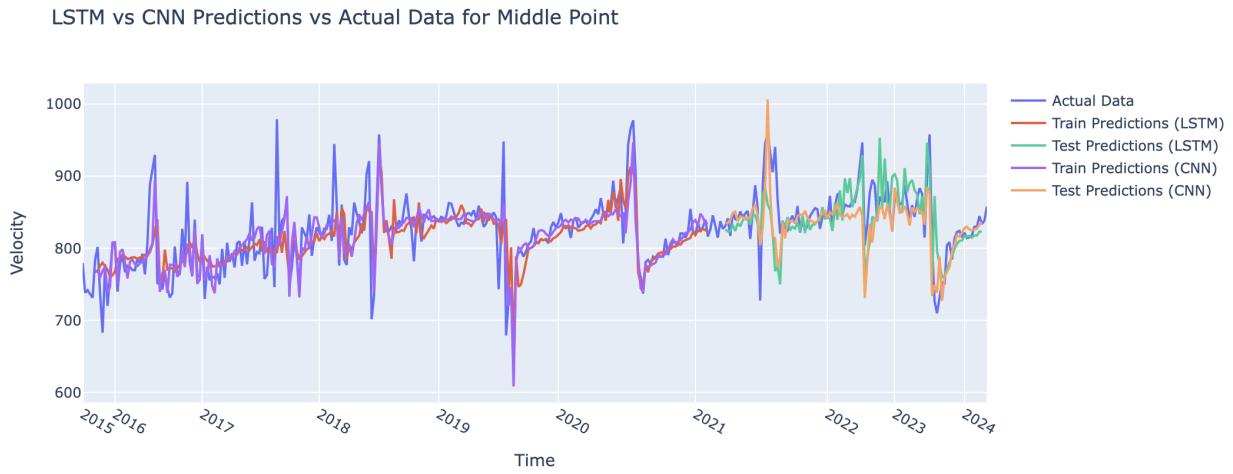

Figure 7: Train and test predictions from the CNN (train: purple, test: yellow) and LSTM (train: red, test: green) models plotted against the ground truth (blue).

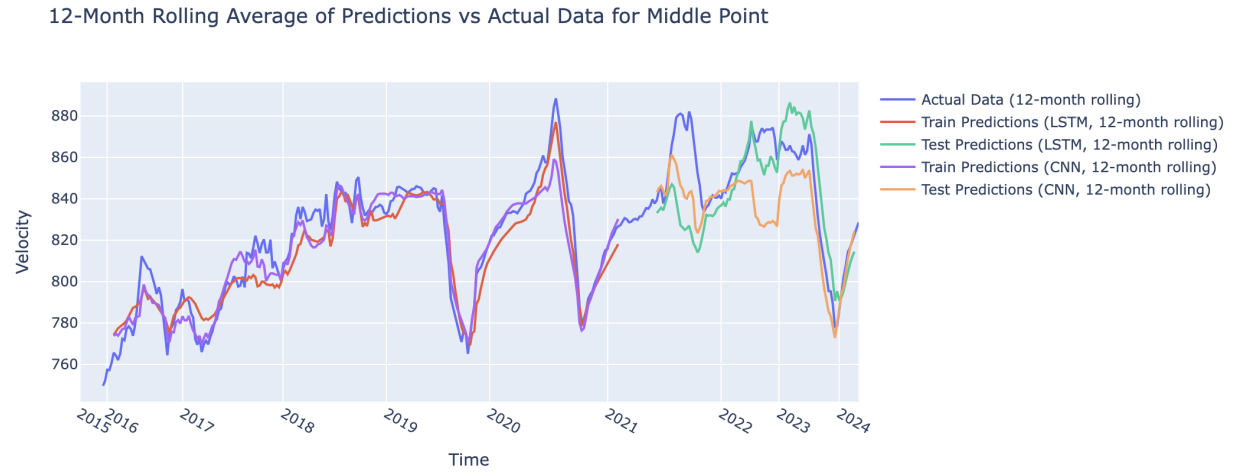

Figure 8: The 12-month rolling average of the original 6-day frequency predictions from the CNN and LSTM and the ground truth.

3.3 Model Comparison

The LightGBM performed the best out of all models tested, based on its RMSE score (Table 1). While metrics like RMSE and MAE often do not capture the granular details of the performance of models on data, despite the predictions not matching the test data, LightGBM captured the seasonal fluctuations best out of any model.

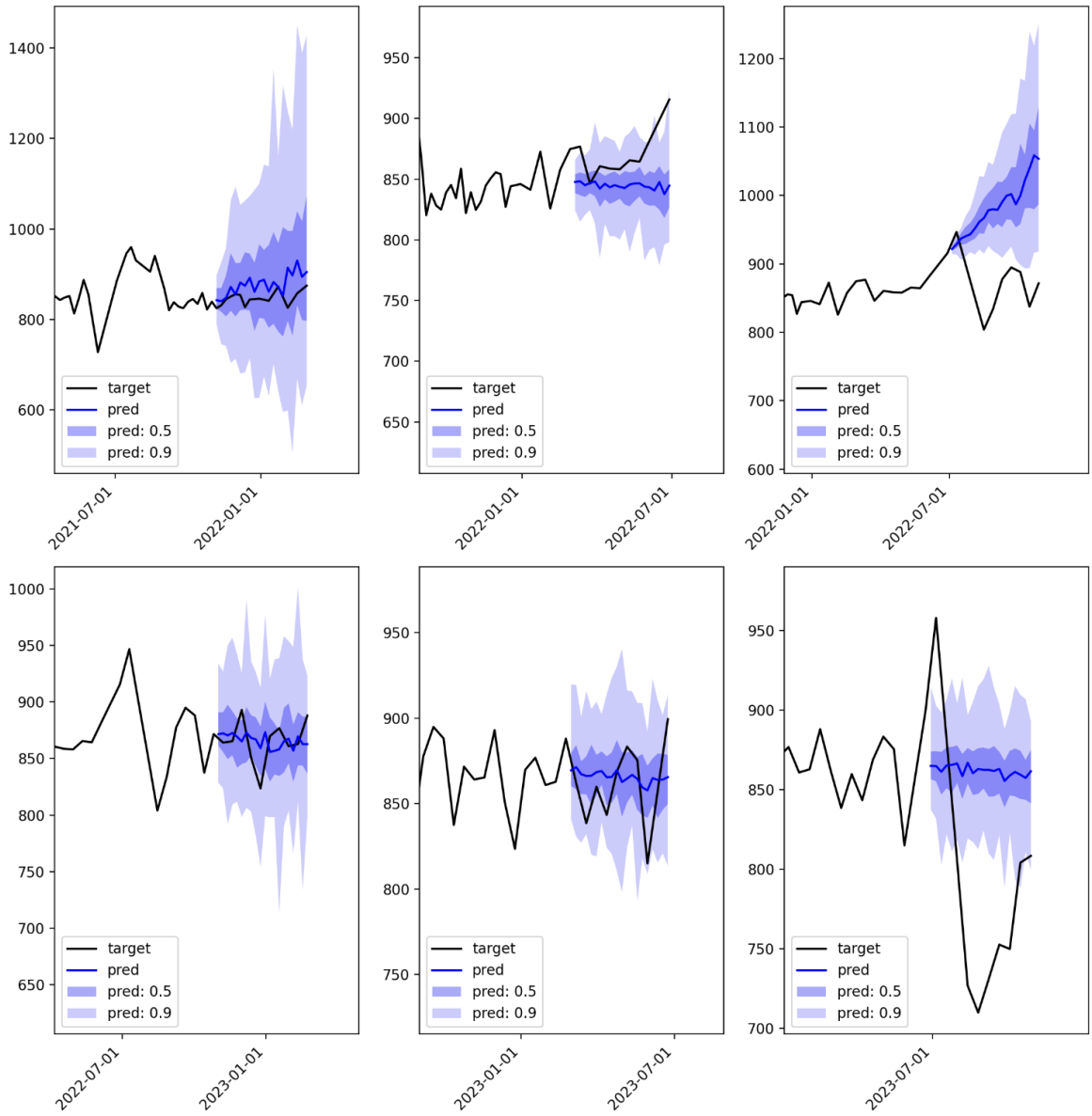

Figure 9: Predictions (blue line) from Moirai zoomed to capture the predictions and their intervals (blue highlights). Plotted against ground truth (black line)

LightGBM	CNN	LSTM	Moirai	Moirai-moe
39.9788	834.30	44.89	85.669	47.95

Table 1: RMSE errors of all models tested.

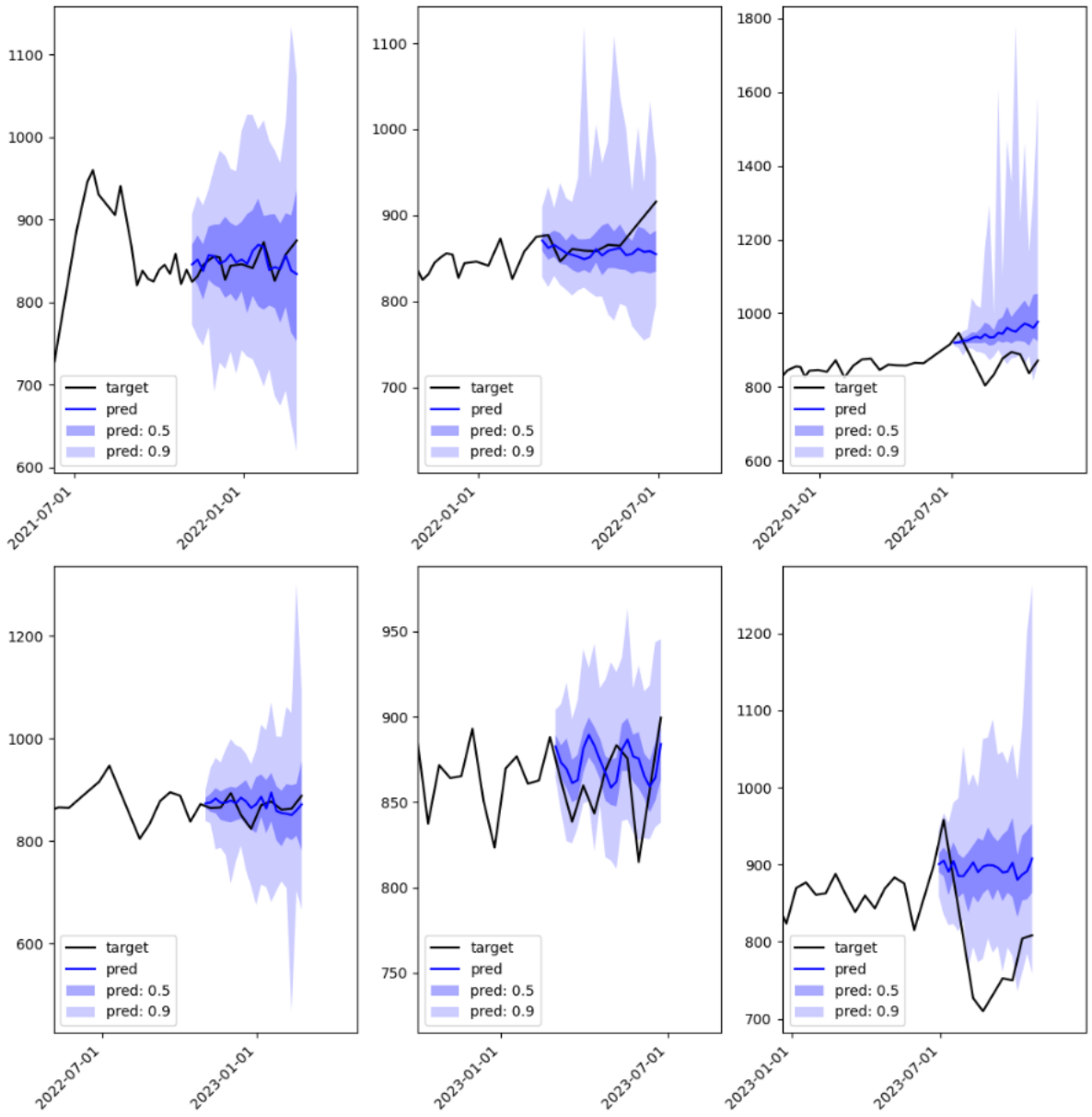

Figure 10: Predictions (blue line) from Moirai-moe zoomed to capture the predictions and their intervals (blue highlights). Plotted against ground truth (black line)

4. Reproducibility

In the Github repository, there are detailed notebooks that outline the steps to reproduce this work. All figures are generated in the notebooks, which are organized in a suggested order and the order in which they are mentioned in this report. This work can be expanded on by testing different data points and tweaking the existing models.

If I were to submit this to a scientific journal, I would likely choose to publish open access in the Journal of Glaciology or in the IEEE Journal of Remote Sensing. I would focus on insights into glacial dynamics learned from the model in the former and machine learning techniques for time series prediction of glacier velocity in the latter. Both journals can be catalysts for bridging the gap between geoscience and computer science, and are journals in which scientists from both disciplines can stand to learn more about the other.

5. Conclusion

This report and the accompanying Github repository perform and document techniques for forecasting velocity time series values at ZI. The

Based on the differing model selection, hyperparameter tuning, and performance of different models for the terminus, middle, and upstream points, it is evident that one model cannot be generalized to perform best on all types of velocity time series. At a glacier like ZI, where the velocity is greatest at the terminus and decreases upstream, velocity values differ greatly along a flowline, making it difficult to fit one model to the varying time series. Additionally, consideration of physics and exogenous variables such as mélange status, meltwater discharge, and temperature would likely aid in the prediction of velocity values since physical processes and climatic variables guide velocity over time.

However, with advances in computing power and machine learning models developing quickly, the ability to train large datasets and the availability of LLMs to use for time series forecasting are becoming readily available. Future work includes sparse sampling of velocity points at ZI and fitting a model a) to each data point, or b) to all data points sampled. Additionally, further testing of Moirai on multiple velocity time series at ZI, derived from a flowline, and expanding to other Greenland glaciers could improve the prediction accuracy of the model.

Additional work comparing the terminus and upstream points to the results of the middle point could generate interesting model comparisons, though this is out of scope for this report. The

seasonality of the terminus was captured by the CNN and LSTM (Fig. 16; Appendix), likely because seasonal peaks in velocity were exaggerated in this high-velocity area, making it easier to capture them in a model. Additionally, larger datasets of combined time series from different points on the glacier or training with data from a variety of Greenland glaciers could improve predictions, particularly with deep learning models that require more data.

Through this project, I learned about the importance of scalability; while the time series data I used to test the models were small, they still produced interesting results, especially regarding the generalization of models and metrics for evaluating performance. This project serves as a starting point for my own exploration into machine learning and directions to expand on and strengthen this work in the future.

References

[1] I. Joughin. *MEaSUREs Greenland 6 and 12 Day Ice Sheet Velocity Mosaics from SAR, Version 2*. National Snow and Ice Data Center. 2022. DOI: 10.5067/1AMEDB6VJ1NZ.

[2] Michalea D. King et al. "Dynamic Ice Loss from the Greenland Ice Sheet Driven by Sustained Glacier Retreat". In: *Communications Earth & Environment* 1.1 (Aug. 2020), p. 1. ISSN: 2662-4435. DOI: 10.1038/s43247-020-0001-2.

[3] E. Rignot and J. Mouginot. "Ice Flow in Greenland for the International Polar Year 2008–2009". In: *Geophysical Research Letters* 39.11 (2012). ISSN: 1944-8007. DOI: 10.1029/2012GL051634.

[4] James B. Tlhomole, Matthew Piggott, and Graham Hughes. "Deep Glacier Image Velocimetry: Mapping Glacier Velocities from Sentinel-2 Imagery with Deep Learning". In: *Climate Change AI*. NeurIPS 2023 Workshop on Tackling Climate Change with Machine Learning. Climate Change AI, Dec. 16, 2023.

[5] Gerald Woo et al. *Unified Training of Universal Time Series Forecasting Transformers*. May 22, 2024. DOI: 10.48550/arXiv.2402.02592. arXiv: 2402.02592 [cs]. Pre-published.

6. Appendix

	Description	Value
0	Session id	123
1	Target	velocity
2	Target type	Regression
3	Original data shape	(364, 3)
4	Transformed data shape	(364, 5)
5	Transformed train set shape	(272, 5)
6	Transformed test set shape	(92, 5)
7	Numeric features	2
8	Date features	1
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Transform target	True
14	Transform target method	yeo-johnson
15	Fold Generator	TimeSeriesSplit
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	reg-default-name
21	USI	2ad1

Figure 11: The output of Pycaret's setup function which specifies the model's parameters.

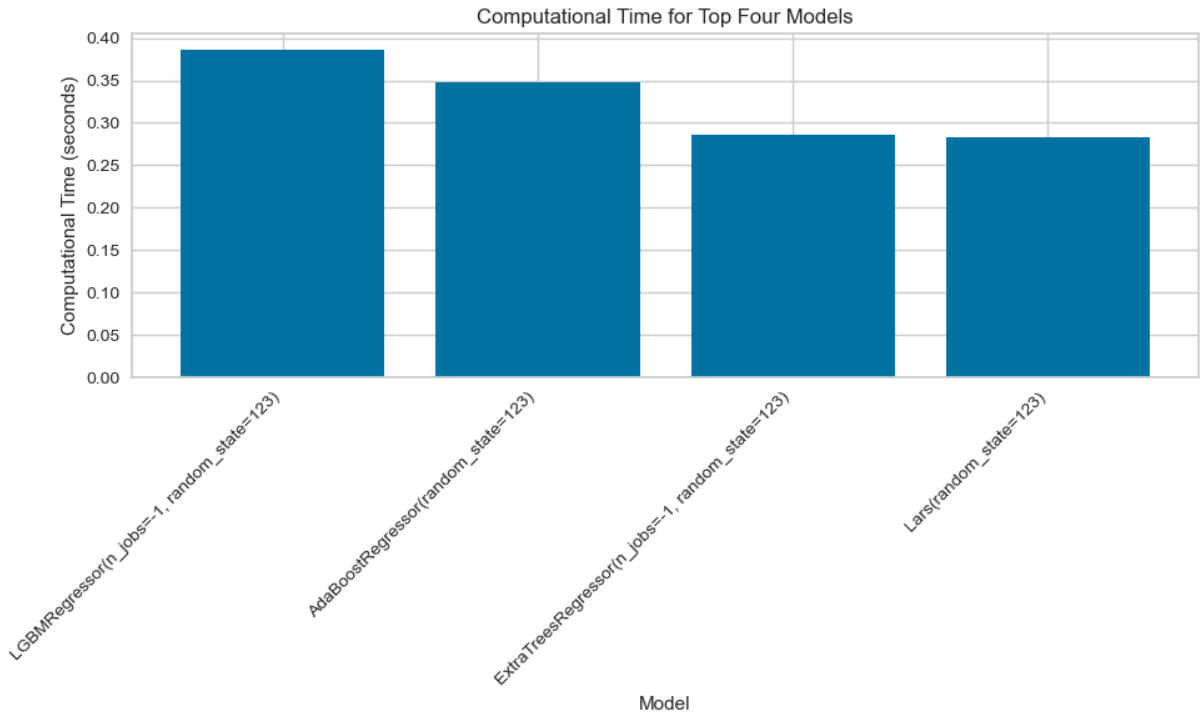

Figure 12: The time in seconds that it took to run the top four models on the test data.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lightgbm	Light Gradient Boosting Machine	31.4361	2225.9147	44.1856	-0.7503	0.0537	0.0384	0.0720
ada	AdaBoost Regressor	35.6793	2561.4955	48.2397	-1.3645	0.0588	0.0435	0.0130
et	Extra Trees Regressor	36.0432	2735.0325	48.7077	-1.3405	0.0594	0.0440	0.0250
lar	Least Angle Regression	36.3794	2826.4439	47.9333	-1.1297	0.0583	0.0455	0.0090
ridge	Ridge Regression	37.0945	2978.1135	48.9743	-1.2873	0.0595	0.0464	0.3700
omp	Orthogonal Matching Pursuit	37.9754	3007.0358	49.4812	-1.4924	0.0601	0.0476	0.0090
dummy	Dummy Regressor	38.4435	2601.1221	47.9499	-1.3542	0.0582	0.0460	0.0090
rf	Random Forest Regressor	38.6190	2783.5259	49.6966	-2.7909	0.0610	0.0475	0.0250
gbr	Gradient Boosting Regressor	39.0398	2934.1325	50.6737	-2.5415	0.0621	0.0474	0.0170
en	Elastic Net	39.3257	2628.6182	48.7408	-1.6600	0.0591	0.0477	0.2040
lasso	Lasso Regression	39.3877	2637.5243	48.7967	-1.6749	0.0592	0.0477	0.2890
llar	Lasso Least Angle Regression	39.3877	2637.5243	48.7967	-1.6749	0.0592	0.0477	0.0090
lr	Linear Regression	39.4083	3488.9702	51.5371	-1.6044	0.0624	0.0494	0.0870
br	Bayesian Ridge	39.4392	3335.3365	51.3380	-1.6860	0.0622	0.0495	0.0090
knn	K Neighbors Regressor	43.6963	3463.9950	54.8524	-3.7836	0.0672	0.0540	0.0120
dt	Decision Tree Regressor	44.4903	3948.2351	59.2573	-4.6039	0.0706	0.0540	0.0080
par	Passive Aggressive Regressor	515.5132	400546.0839	518.4047	-393.6579	3.3757	0.6324	0.0120

Figure 13: The output of Pycaret's compare_models function sorted by MAE.

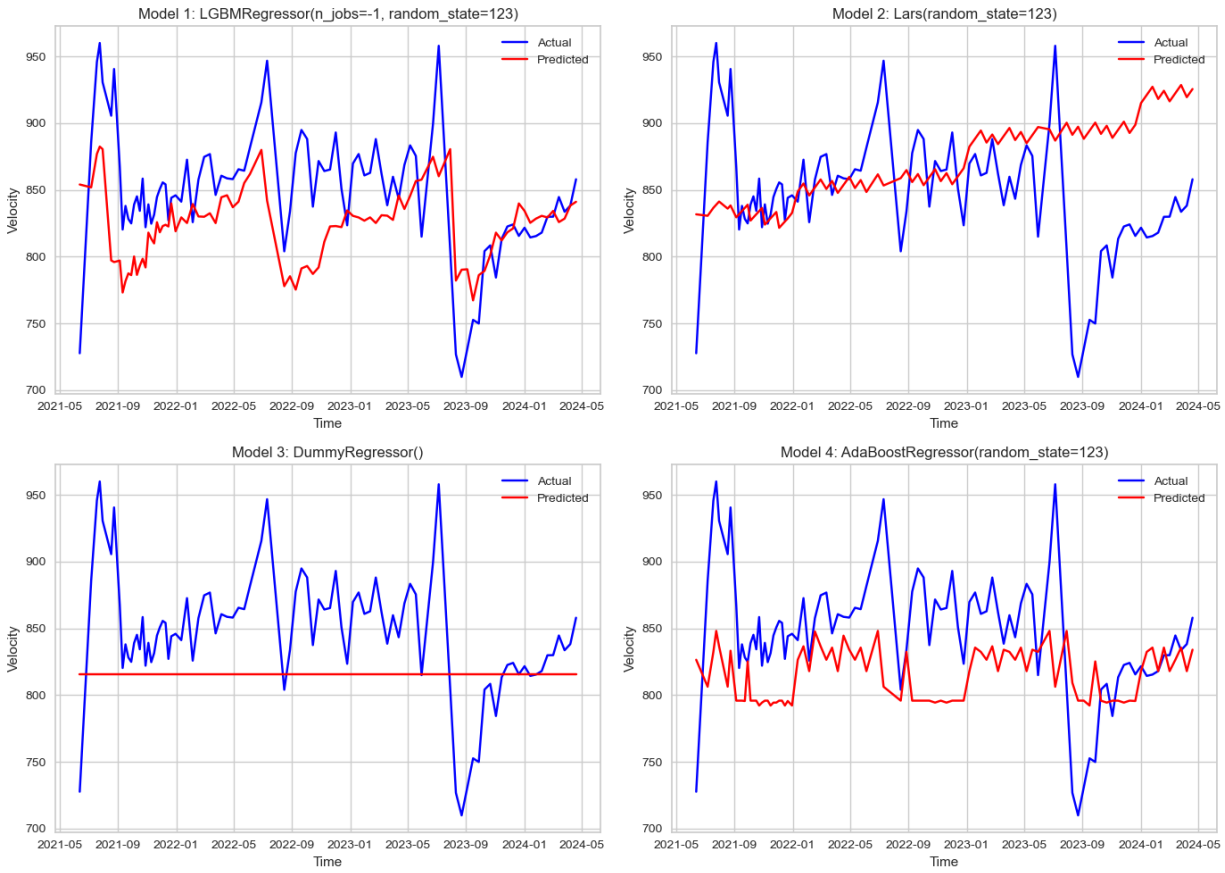

Figure 14: The top four models based on their RMSE score on the training data.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Light Gradient Boosting Machine	38.1687	2370.2136	48.6848	-0.1064	0.0580	0.0447

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Least Angle Regression	47.9550	4259.5550	65.2653	-0.9883	0.0771	0.0581

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Dummy Regressor	44.2082	3153.1685	56.1531	-0.4718	0.0664	0.0514

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	AdaBoost Regressor	45.5108	3118.1159	55.8401	-0.4555	0.0663	0.0531

Figure 15: The output of Pycaret's predict_models function evaluating performance on the test data.

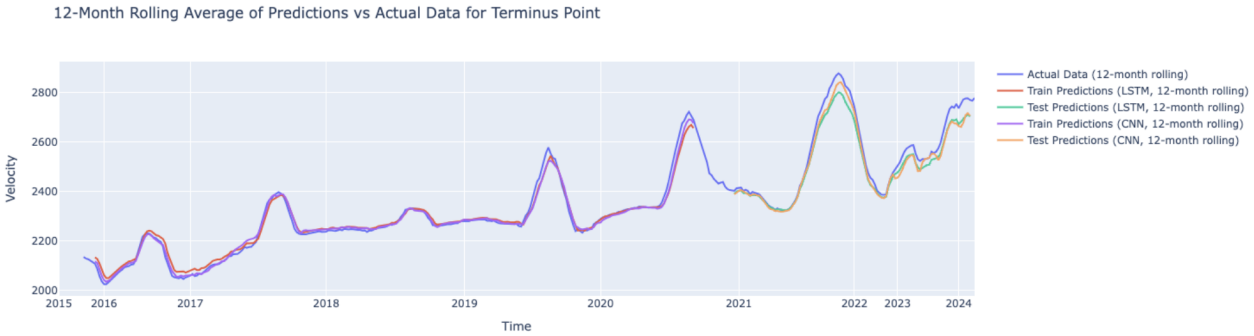

Figure 16: The 12-month rolling average of the original 6-day frequency predictions from the CNN and LSTM and the ground truth of the terminus point.

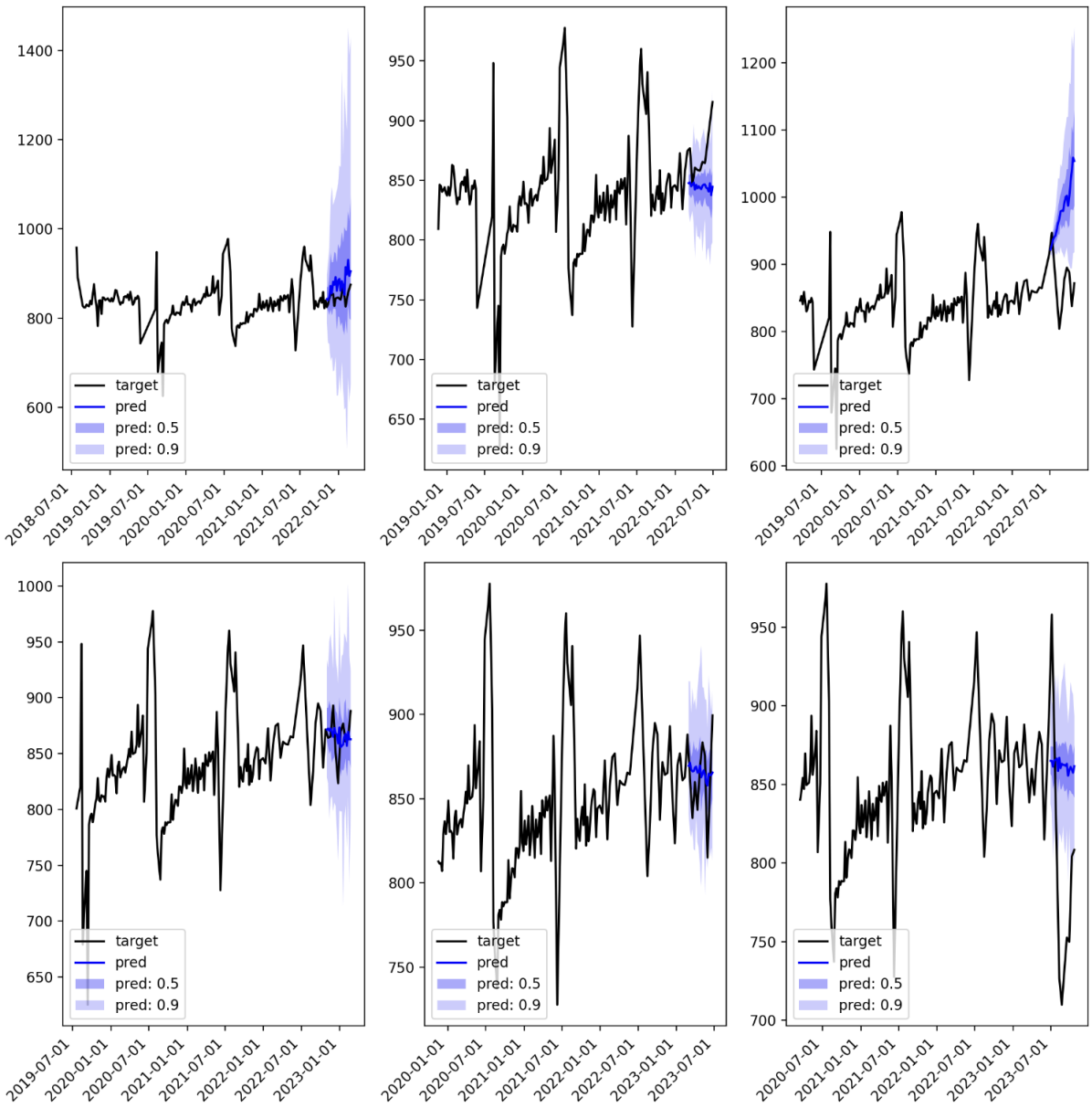

Figure 17: Predictions from Moirai and extent of the context length of ground truth data.

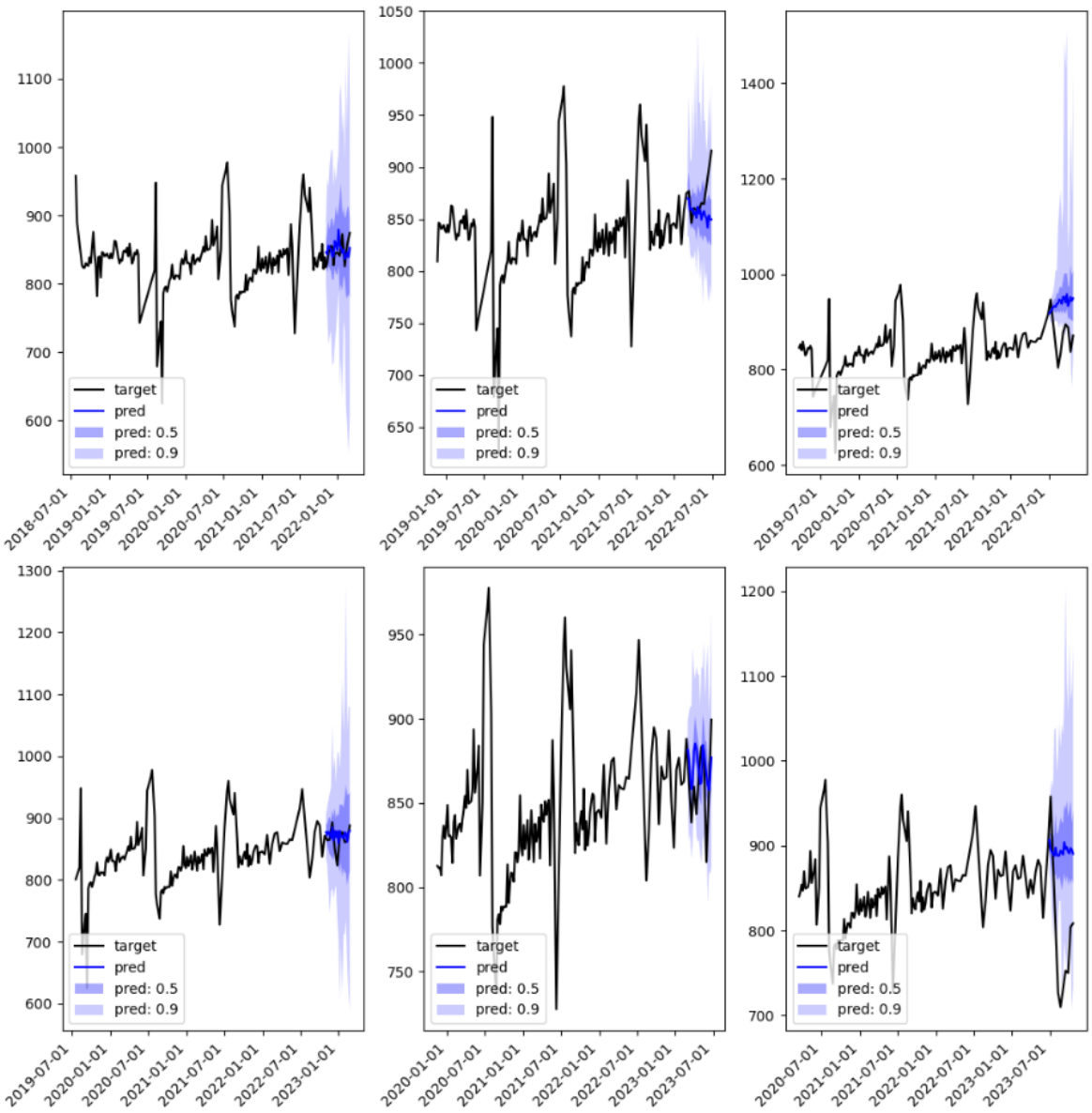

Figure 18: Predictions from Moirai-moe and extent of the context length of ground truth data.