Hybrid Model: Deep learning GRU neural network and K-nearest neighbors for Wind Power Forecasting

dateteam-UCM submission for BAIDU KDD Cup 2022 (20th/2,409 teams)

Fernando Sebastián Huerta*
ferseb39@gmail.com
BCAM-Basque Center for Applied Mathematics Bilbao
Madrid, Spain

Manuel Angel Suarez*
manuesua@ucm.es
Universidad Complutense de Madrid, UCM
Madrid, Spain

Daniel Velez Serrano*
danielvelezserrano@mat.ucm.es
Universidad Complutense de Madrid, UCM
Madrid, Spain

Alejandro Carrrasco Sánchez*
acarrasco2008@gmail.com
Unspecified affiliation
Madrid, Spain

Eugenio Neira Bustamante*
eug.neira@gmail.com
Unspecified affiliation
Madrid, Spain

ABSTRACT

This paper summarizes our approach to the BAIDU KDD CUP 2022 Spatial Dynamic Wind Power Forecasting. With a data set of 245 days of historical 10-minute data for a park of 134 turbines, wind power was forecast over two days. The model combines a Recurrent Neural Networks Model with more classical time series forecasting approaches based on K-Nearest Neighbors (KNN) models.

The link to download the code corresponding to the paper can be found at GitHub:

https://github.com/ManuelAngel99/KDD_CUP_2022

KEYWORDS

Neural networks, wind power forecasting, time series, KDD Cup, GRU, KNN, ARIMA, prophet, deep learning

1 INTRODUCTION

Solar and Wind are sources of renewable energy that have high production variability. In order to accurately match electricity production and consumption, future power production prediction is critical, especially when wanting to introduce a high amount of these power sources into a country's electric grid.

In recent years, time series forecasting, based on data mining and machine learning, has provided diverse solutions to the problem of forecasting electricity demand and power production. In particular, Wind Power Forecasting (WPF) aims to accurately estimate the wind power supply of a wind farm given its historical data, it is considered one of the most critical issues for wind power integration and operation.

The 2022 KDD Cup challenge proposed a WPF challenge, providing the Spatial Dynamic Wind Power Forecasting dataset from Longyuan Power Group Corp. Ltd. This included the spatial distribution of wind turbines as well as more than 8 months of historical 10-minute data for 134 wind turbines, including power production, wind speed, direction, temperature, and other turbine internal status variables. The challenge consisted in predicting future power for a two-day interval (or 288 10-minute periods) for each of the 134 turbines with minimum RMSE and MAE. A full description of the data set and challenge can be found in [1].

*These authors contributed equally to this research.

2 SOLUTION OVERVIEW

The WPF field has been thoroughly studied over the past decades, and we will not propose here an exhaustive bibliographic study, which can be found in [1], [2], [3], and [4]. We will instead center our bibliographical citations on the papers and models that inspired our concrete solution for this particular dataset throughout the following sections.

2.1 Data exploration and pre-processing

The variables provided in the dataset were the following:

Variable Name	Specification
TurbID	Wind Turbine ID
Day	Date of the record
Tmstamp	Created time of the record
Wspd (m/s)	The wind speed recorded by the anemometer
Wdir (°)	Angle between wind direction and turbine nacelle
Etmp (℃)	Temperature of the surrounding environment
Itmp (℃)	Temperature inside the turbine nacelle
Ndir (°)	Nacelle direction, i.e., the yaw angle of the nacelle
Pab1/2/3 (°)	Pitch angle of blade 1/2/3
Prtv (kW)	Reactive power
Patv (kW)	**Active power (target variable)**

With more than eight months of 10-minute data for all of the above variables, a 2-day prediction of the target variable Patv was to be calculated.

As seen in figure 1, the highest correlation for the Patv variable is found between wind speed and Patv, therefore, the possibility of utilizing wind predictions was deemed tempting for longer-term power prediction, this matches bibliographical source's interest, as seen in [5] and [6]. However, competition rules impeded utilizing external data sources such as plant localization.

NaN values represent 1,05% of all rows in the data set, however, when considering "invalid" values (as defined in [1]) the total number is raised to approximately 30%, this is especially significant

as these invalid values are often grouped for specific turbines and periods of time.

This large amount of invalid values can be attributed to the fact that data collection is from a real wind turbine plant, this implies a challenge when creating models but ensures the models resulting from the competition would be applicable for this real-world scenario.

Figure 1: Correlation matrix for the different dataset variables

Data pre-processing was explicitly tested for each model, as described in 4, this resulted in different techniques being applied to each model. Classic time series feature engineering methods were used to encode time and nacelle direction with sines and cosines, variables were normalized with Min-Max scaler or StandardNormal scaler, and NaN and invalid values were filled with a range of methods (forward filled, zeros...).

A distribution of median Patv values grouped by timestamp is shown in figure 2 for the full dataset, a clear tendency is identified for different times of day.

2.2 Challenges

The main challenges faced by the team in order to resolve the proposed problem include:

- Fitting models to predict short-term and long-term predictions: the prediction horizon extended to predict up to 288-time ticks. Model performance varies greatly between short-term and long-term predictions for different models, this is a common wind forecasting problem, as explained in [7].
- The forecast was evaluated per turbine basis, while creating per-turbine models were often discarded due to limitations on execution time and model size introduced by the competition organizers.
- A large number of NaN and invalid data points created a challenge when training models with low bias, these would over-fit easily on the training set.
- Offline model validation transfer to online execution was poor, even though many evaluations were performed on the 80-day validation set, as described in 4. This is attributed to:
  - Large data variability depending on wind conditions between more turbulent/calm periods that are difficult to predict.
  - When evaluating, a large number of invalid data points are eliminated for the final online result calculation, this removal of data can result in large periods of invalid data as can be seen in the training set. Also, in the online execution only a few samples were evaluated in the test period (143 samples in phase 3).

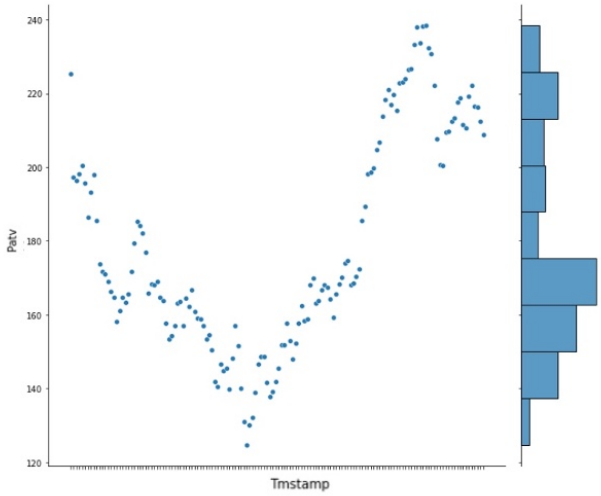

Figure 2: Median values Patv values grouped by Timestep

3 DETAILED METHOD

Our proposed recurrent neural network models were inspired by the solution given as a baseline, the model known as "PaddleSpatial WPF Baseline GRU" provided by the organizers. This model was transferred to the PyTorch framework in order to optimize execution times and combined with classical time series forecasting models based on KNN.

Model ensembles were created by analyzing errors committed for different parameters (for different prediction time horizons, different turbID and combinations) and picking the model with less error in each prediction. The experiments performed will be further explained in 4.

3.1 Recurrent Neural Network model: Baseline method paddlepaddle to PyTorch (GRU)

As the paddlepaddle framework is similar to PyTorch and opportunities for improvement were identified both from a modeling and computational efficiency perspective, a model inspired by the competition baseline was written in PyTorch.

The most remarkable differences between both models are: our PyTorch model utilizes a single model for the 134 turbines (we obtained a better score and, obviously, a shorter execution time) ; a higher dropout and higher batch size.

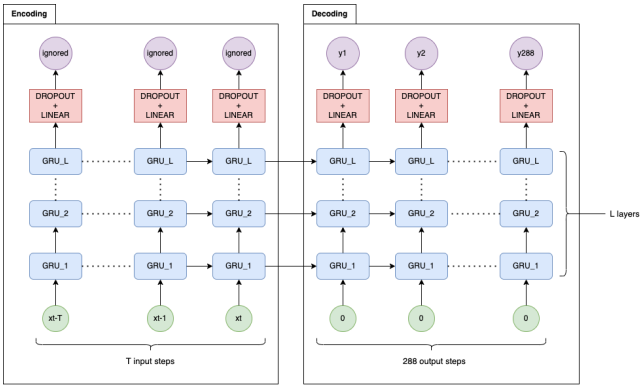

Figure 3: Diagram of the GRU model.

3.2 Traditional time series forecasting models and K-nearest neighbors

Low data quality in the sense of a large percentage of NaN and invalid data points lead models to over-fit quickly, producing a high error for longer-term predictions. This led the team to explore more classic and simpler models to reduce errors in long-term predictions.

Various kinds of classic time series prediction model variations were tested: simple mean and median models per timestamp (see figure 2) and TurbID, ARIMA models, different Wavelets transform models ([8]), Fourier transform models, exponential smoothing models and KNN models.

Some examples of these can be seen in figure 4. This image was created with the Weights and Biases API, see section 4.1 for more details :

- The daily median model (seen in green) simply predicts the historic median of the training set (see figure 2 - although this figure encompasses the full train set's median, the model shown above removes the last 80 days for internal testing, as described in 4.1)
- The KNN model (in blue) is described below.
- WAVE (orange) model uses Symlet 8 wavelets, from ([8]).
- HAAR (purple) model uses Haar wavelets, from ([8]).

As mentioned in the first paragraph, and as it can be appreciated in figure 4, simpler models appeared more robust for longer-term predictions (simple green median model) in our offline testing set-up both for RMSE and MAE. However, the translation of these positive results to the final online score was not always achieved, particularly due to an increase in the measured standard deviation of scores, as described in 4.1, which implies a higher model variance.

Out of all classical models tested, the best competition score results were obtained by time-series KNN models in both offline and online testing.

The applied KNN model is similar to those described in [9]:

(1) The last *LB* data points are compared to historical sequences of length *LB*
(2) By measuring the distance using different measures, the closest K neighbors are chosen
(3) For each neighbor, the next LF timesteps are saved
(4) A prediction of length LF is produced by computing a weighted average for each neighbor's distance, to produce the final results.

Weighting introduced for each neighbor was inversely proportional to its distance (L) to the considered sequence, with a $\frac{1}{1+L}$ weight being considered.

Different distances, LB and LF lengths and K numbers were tested, and hyperparameter tuning and selection are described in section 4.3.

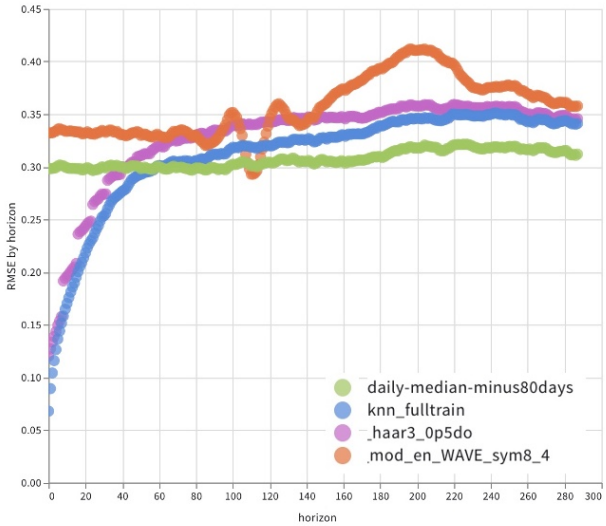

Figure 4: Comparison of various classic time series forecasting models tested and their mean RMSE error by horizon.

3.3 PyTorch GRU models for different horizons: 1 hour, 3 hours, 10 hours, and 48 hours

The model from section has been considered for different horizons. A summary of the neural network model (GRU) applied is explained below:

Hyperparameters:

(1) input_length: 144
(2) output_length: 288
(3) inputs: ["Wspd", "Wdir", "Etmp", "Itmp", "Ndir", "Pab1", "Pab2", "Pab3", "Prtv", "Patv"]
(4) targets: ["Patv"]
(5) lstm_layer: 2
(6) dropout: 0.3
(7) train_epochs: 5
(8) train_batch_size: 256
(9) val_batch_size: 512
(10) lr: 1e-4

Model:

```
BaselineGruModel(
(dropout): Dropout(p=0.3, inplace=False)
(lstm): GRU(10, 48, num_layers=2, batch_first=True)
(projection): Linear(in_features=48,out_features=1,bias=True)
```

```
)

22.8 K    Trainable params
0         Non-trainable params
22.8 K    Total params
0.091     Total estimated model params size (MB)
```

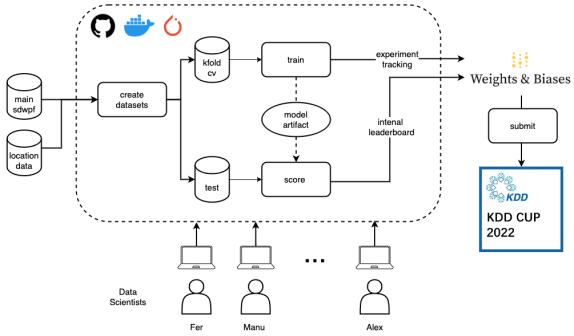

Figure 5: Diagram of team workflow and development sandbox.

4 EXPERIMENT PARTS

4.1 Offline testing methodology

As described in section 2.2, offline testing to validate which model to upload to the competition webpage was challenging. Different testing strategies were put in place by the team, finally the Weights And Biases API (see [10]) was deemed the best solution for collaboration and offline model comparison.

The last 80 days of the train set were used as internal offline testing for our models, i.e., the interval of days encompassing days [166, 245] was separated when training models and a final model train, including the whole train set, was performed before the final model upload to the competition web page.

The evaluation script was used to evaluate 2,048 predictions over these 80 days, these predictions were then averaged and the results uploaded to the weights and biases API, where different error metrics were calculated and compared for each model, mainly: competition score (according to [1]), RMSE and MAE committed per turbine and RMSE and MAE per time horizon for each of the 288 predictions.

The **standard deviation** of these errors was also calculated, this measure proved to be an essential measure to consider. As described in section 2.2, model transfer from offline to online scoring was not very good. The team found a good result in both the model's offline score and the standard deviation of the 2,048 model scores indicated a better online performance than considering the model score alone.

Error metrics per TurbID and per time horizon enabled easy model ensemble creation. However, due to the high number of invalid values (not considered in the evaluation) and high data variability of the training, validation and testing sets, the team found that better results in these graphical comparisons often did not necessarily result in better online model performance when ensembling the models.

4.2 Recurrent neural network tests

For the GRU model described in section 3.3, a cross-validation strategy with 5 kfolds was employed over the [1, 165] day interval. After these, final offline tests were performed as described in section 3.3.

The following table shows the partitioning of the 5 kfolds:

Kfold Number	Train	Validation
1	[34,165]	[1,33]
2	[1,33],[67,165]	[34,66]
3	[1,66],[100,165]	[67,99]
4	[1,99],[133,165]	[100,132]
5	[1,132]	[133,165]

Over the test period, we randomly selected several starting points. We considered 2,048 test sets (test_x and test_y with the nomenclature of the BAIDU KDD 2022 competition). This allows us a trade-off between computational time and the reliability of the error metric. For the train period, we have considered all possible predictions shifted every 10 minutes.

For example, the first and last five test subsets are presented below:

Nº test	Maximum Day	Maximum Tmstamp
1	180	0:00
2	180	0:10
3	180	0:40
4	180	0:50
5	180	11:40
...	...	...
2,044	242	6:50
2,045	242	7:00
2,046	242	7:50
2,047	242	8:20
2,048	242	9:20

In addition, we replicate the test scheme as in the web, i.e., with a history of only 14 days prior to the prediction.

Evaluation score

The following table shows the scores of the GRU model in PyTorch for each kfold and the mean of the kfold in phase 2:

Model	Offline score	Online score
KFOLD 1	-43.409	**-44.92653**
KFOLD 2	-43.413	-44.94513
KFOLD 3	-43.141	-45.06132
KFOLD 4	-43.164	-45.29073
KFOLD 5	-42.987	-45.48204
MEAN KFOLD	-43.41	**-44.98518**

One of the conclusions from the scores of each of the kfolds is that the Online score is strongly dependent on the period of days

considered (within the 245 days). Also, note the deviation from this using the same model. Note that the offline score is more robust as it has more test sets (2,048 vs. 219). The table below shows the score of the best models in phase 3:

N°	Model	Offline score	Online score
1	MEAN KFOLD	-43.41	-46.20274
2	MEAN KFOLD + GRU288	-	-46.04143
3	Model 2 + Haar (nlevels=4)	-	-46.05027
4	GRU288 MAX KFOLD	-43.708	-45.76524
5	FINAL (HYBRID) MODEL. GRU288, GRU60, GRU18, GRU6 MAX KFOLD and KNN	43.883	**-45.56335**

4.3 KNN experimental test

KNN hyper-parameters were also chosen by the validation method described in section 4.1 .

The full KNN model description is included in section 3.2.

KNN models were tested separately by performing neighbor search on the whole train set and only on the 14 days previous to each prediction, the later producing worse results.

The different hyper-parameter tests performed included (value retained in bold type):

- Distances: **Euclidean**, cosine,
- LB: 2, **6**, 12, 15, 30, 144
- k neighbor number: 30, 60,100, 150, 300, 500, **1500**, 2000

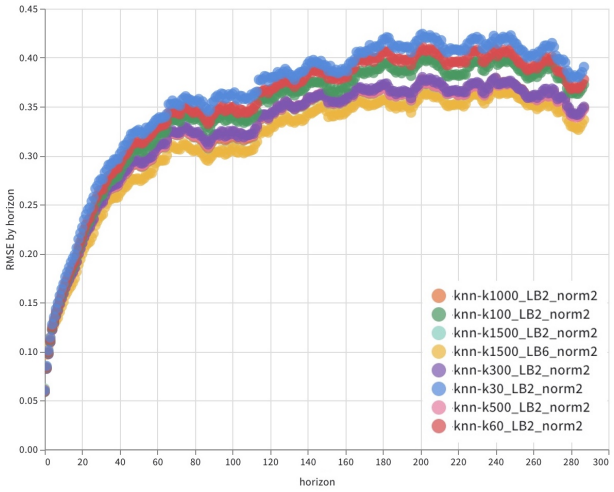

Figure 6: MEAN RMSE for different KNN models per prediction horizon.

In Figure 6, different k-neighbor number models are compared per prediction time horizon for an LB of 2 and 6 and the euclidean distance. The $k = 1500$ model has the least RMSE error of all LB=2 models and by changing LB to 6 (1 hour) the error can be seen to decrease further. The tests in the figure were performed offline with only 300 evaluations (compared to the full set of 2,048 evaluations described in 4.1) for rapid testing. This figure was made with the Weights and Biases API as discussed in section 4.1.

4.4 Visual comparison between models (test samples)

The predictions of the different models (GRU288, KNN, GRU60, GRU18, GRU6, hybrid model) for one turbine are shown below:

Figure 7: Comparison forecasting for TurbID=7 and Sample=1. 288 horizons.

Figure 8: Comparison forecasting for TurbID=7 and Sample=1. 60 horizons.

5 FINAL MODEL. HYBRID MODEL (KNN AND GRU)

The methodology described in section 4.1 enabled model comparison in order to create the best offline ensemble models, these models were then uploaded to the competition web page to study their online performance.

The final prediction with the best online score is obtained as indicated in the following lines.

(1) $(\hat{y}_1^{GRU288}, ..., \hat{y}_{288}^{GRU288})$ the vector of predictions resulting from taking the maximum for each component of the vector for each kfold from GRU model horizon=288

(2) $(\hat{y}_1^{GRU60}, ..., \hat{y}_{60}^{GRU60})$the vector of predictions resulting from taking the maximum for each component of the vector for each kfold from GRU model horizon=60

(3) $(\hat{y}_1^{GRU18}, ..., \hat{y}_{18}^{GRU18})$ the vector of predictions resulting from taking the maximum for each component of the vector for each kfold form GRU model for horizon=1
(4) $(\hat{y}_1^{GRU6}, ..., \hat{y}_6^{GRU6})$ the vector of predictions resulting from taking the maximum for each component of the vector for each kfold form GRU model for horizon= 6
(5) $(\hat{y}_1^{KNN288}, ..., \hat{y}^{KNN288})$ the vector of predictions resulting from KNN model (view table for optimal parameters).
(6) $(\hat{y}_1^{max(GRU288,KNN288)}, ..., \hat{y}^{max(GRU288,KNN288)})$ the max vector obtained from (1) and (5)

The final model (hybrid model) is obtained by the following vector

$$\Big(\hat{y}_1^{GRU6}, ..., \hat{y}_6^{GRU6}, \hat{y}_7^{GRU18}, ..., \hat{y}_{18}^{GRU18}, \hat{y}_{19}^{GRU60}, ..., \hat{y}_{60}^{GRU60}, \hat{y}_{61}^{max(GRU288,KNN288)}, ..., \hat{y}_{288}^{max(GRU288,KNN288)}\Big)$$

The plot below shows the different scores per TurbID:

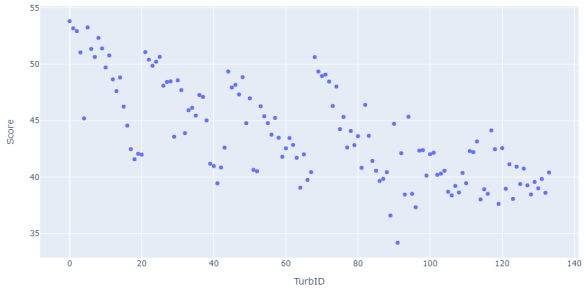

Figure 9: Score by TurbID. hybrid model

6 CONCLUSIONS

This paper presents the procedure applied by our team to resolve the challenge of the Baidu KDD cup 2022.

The competition has been specially challenging regarding data quality (missing, invalid and outlier) values have been common as the provided data set is formed by real data from a wind turbine park. In particular, invalid data values encompass up to 30% of the training set. This has meant that model learning and translation between different test sets has been inconsistent.

With this in mind, the team developed a full internal testing methodology based on the Weights and Biases API that created more reliable and faster internal predictions, as well as facilitating model comparisons for better model ensemble creation.

The result from this methodology is our final hybrid model, that combines models traditional and simple methods (KNN) with more complex methods (GRU recurrent neural networks) as well as models optimized for different horizons and custom models.

REFERENCES

[1] Jingbo Zhou et al. "SDWPF: A Dataset for Spatial Dynamic Wind Power Forecasting Challenge at KDD Cup 2022". In: *Techincal Report* (2022).
[2] Shahram Hanifi et al. "A Critical Review of Wind Power Forecasting Methods—Past, Present and Future". In: *Energies* 13.15 (2020). ISSN: 1996-1073. DOI: 10.3390/en13153764. URL: https://www.mdpi.com/1996-1073/13/15/3764.
[3] M. Lydia and G. Edwin Prem Kumar. "Deep Learning Algorithms for Wind Forecasting: An Overview". In: *Artificial Intelligence for Renewable Energy Systems*. John Wiley Sons, Ltd, 2022. Chap. 6, pp. 129–145. ISBN: 9781119761686. DOI: https://doi.org/10.1002/9781119761686.ch6. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119761686.ch6. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119761686.ch6.
[4] Yun Wang et al. "A review of wind speed and wind power forecasting with deep neural networks". In: *Applied Energy* 304 (2021), p. 117766. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2021.117766. URL: https://www.sciencedirect.com/science/article/pii/S0306261921011053.
[5] Sue Haupt and William Mahoney. "Taming wind power with better forecasts". In: *IEEE Spectrum* 52 (Nov. 2015), pp. 47–52. DOI: 10.1109/MSPEC.2015.7335902.
[6] Ekaterina (Katya) Vladislavleva et al. "Predicting the Energy Output of Wind Farms Based on Weather Data: Important Variables and their Correlation". In: *Renewable Energy* 50 (Feb. 2013), pp. 236–243. DOI: 10.1016/j.renene.2012.06.036.
[7] Jannik Schütz Roungkvist and Peter Enevoldsen. "Timescale classification in wind forecasting: A review of the state-of-the-art". In: *Journal of Forecasting* 39.5 (2020), pp. 757–768. DOI: https://doi.org/10.1002/for.2657. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/for.2657. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/for.2657.
[8] Gregory R. Lee et al. "PyWavelets: A Python package for wavelet analysis". In: *Journal of Open Source Software* 4.36 (2019), p. 1237. DOI: 10.21105/joss.01237. URL: https://doi.org/10.21105/joss.01237.
[9] Samya Tajmouati et al. "Applying k-nearest neighbors to time series forecasting: two new approaches". In: (Mar. 2021).
[10] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.