DLinear Makes Efficient Long-term Predictions

Chaoqun Su
sucq@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

ABSTRACT

The variability of wind power supply can present substantial challenges to incorporating wind power into a grid system. Thus, Wind Power Forecasting (WPF) has been widely recognized as one of the most critical issues in wind power integration and operation. We are expected to accurately estimate the wind power supply of a wind farm based on Spatial Dynamic Wind Power Forecasting dataset(SDWPF).

This work addresses the wind power forecasting problem with the help of time-series decomposition techniques. In particular, after the trend and seasonal features are extracted, two one-layer FCN are adopted to transform and fuse the features for future forecasting. We extract the "Patv" column of 134 wind turbines from SDWPF and reshape it into a new numpy array with a shape of (35280,134). After dividing the dataset, we employ a SeriesDecomp module for Seasonal-Trend decompsition to get seasonal and trend series [1]. Two one-layer linear networks are employed to model these two series for the forecasting task [2].The model makes multivariate predictions and outputs 134 series of predicted values, that is, the Patvs of 134 wind turbines. Code is available at https://github.com/ChaoqunSu/kddcup.

KEYWORDS

Long-term Time Series Forecasting, Seasonal-Trend Decomposition, DLinear, Linear Network

ACM Reference Format:
Chaoqun Su. 2022. DLinear Makes Efficient Long-term Predictions. In *Proceedings of ACM Conference (Baidu KDD CUP 2022).* Baidu KDD CUP 2022, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Wind power is a kind of clean and safe source of renewable energy, but cannot be produced consistently, leading to high variability. Such variability can present substantial challenges to incorporating wind power into a grid system. To maintain the balance between electricity generation and consumption, the fluctuation of wind power requires power substitution from other sources that might not be available at short notice (for example, usually it takes at least 6 hours to fire up a coal plant). Thus, WPF has been widely recognized as one of the most critical issues in wind power integration and operation. There has been an explosion of studies on wind power forecasting problems appearing in the data mining and machine learning community.

The task of Baidu KDD CUP 2022 is Spatial Dynamic Wind Power Forecasting. As we know, this challenge task is essentially a time series forecasting(TSF) task. An illustration of a wind farm is shown in Figure 1. Over the past several decades, TSF solutions have undergone a progression from traditional statistical methods and machine learning techniques to deep learning-based solutions. Transformer is arguably the most successful sequence modeling architecture, which demonstrates unparalleled performances in various artificial intelligence applications, such as natural language processing and speech recognition. Recently, there has also been a surge of Transformer-based solutions for time series analysis. Some notable models for the TSF task include: Informer [3] (AAAI 2021 Best paper), Autoformer [1] (NeurIPS 2021) and the recent FEDformer [5] (ICML 2022). But are Transformers really effective for long-term time series forecasting? A simple model DLinear [2] outperforms existing complex Transformer-based models in most cases by a large margin. DLinear decomposes the time series into a trend and a remainder series and employs only two one-layer linear networks to model these two series with direct multi-step forecasting.

The organizer present a unique Spatial Dynamic Wind Power Forecasting dataset(SDWPF) [4] that provides the wind power data of 134 wind turbines from a wind farm over half a year with their relative positions and internal statuses. The dataset provide the information about the wind, temperature, turbine angle and historical wind power. The time range of the dataset is over half a year. There are two unique features for this competition task different from previous WPF competition settings: 1) Spatial distribution: this competition provides the relative location of all wind turbines given a wind farm for modeling the spatial correlation among wind turbines. 2) Dynamic context: the weather situations and turbine internal status detected by each wind turbine are provided to facilitate the forecasting task.

Our team employ DLinear which only has two one-layer linear networks to finish the challenge task, the remainder of this technical paper is organized as follows. Sec. 2 presents the solution overview about how to finish the challenge task. Then, we present detailed method in Sec. 3. Experimental details are then shown in Sec. 4. Finally, Sec. 5 concludes this technical paper.

2 SOLUTION OVERVIEW

As noted in the introduction, in order to be able to use DLinear well to solve this task, we first need to process the dataset. we extract the "Patv" column of 134 wind turbines from SDWPF and reshape it into a new numpy array with a shape of (35280,134). We take the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Baidu KDD CUP 2022, July 2022, Technical Paper
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

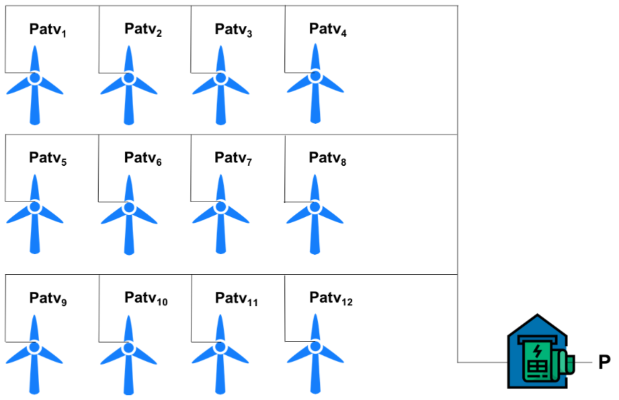

Figure 1: An illustration of a wind farm, Baidu KDD CUP 2022.[4]

data of the first 155 days as the training set and the data of 80 days from day 155 to day 235 as the validation set.

In this part of the model, We employ a SeriesDecomp module to decompose the time series data to obtain seasonal and trend series. And then two one-layer linear networks are employed to model these two series for the forecasting task. One linear network is used to model the trend series and another linear network is used to model the seasonal series. The sum of the outputs of the two linear networks serves as the final predictions.

3 DETAILED METHOD

3.1 Dataset

The raw dataset (wtbdata_245days.csv) is 245 days of data including 13 features from 134 wind turbines. At first of all, we fill missing values with zeros and replace all negative values in the "Patv" column with zeros. We only need the last feature variable (i.e. the target variable "Patv"). Then, we extract the column of "Patv" with a shape of (4727520, 1) ,where 4727520 = 134*245*144, we reshape it into a numpy array with a shape of (35280, 134) , where 35280 = 245*144.

We take the data of the first 155 days as the training set and the data of 80 days from day 155 to day 235 as the validation set. The training set is used to compute the normalized mean and variance. All the datasets are replicated in two copies, one for input and one for label. In addition, for better evaluation during model validation, we need to keep the original data corresponding to the label in order to remove bad predictions when calculating the loss. Finally, the __getitem__() method of the Dataset class has three return values, seq_x, seq_y, input_y.

It is worth mentioning that we have followed the data augment method used in the official baseline.

3.2 Model

From the experiments in previous works [1][3][5], decomposition can largely enhance the performance of Transformer-based methods in time series forecasting, which is model-agnostic and may boost other models, such as our linear model. Specifically, we use a moving average kernel on the input series to extract the trend-cyclical component of the time series. The difference between the original series and the trend component is regarded as the seasonal component.

Accordingly, DLinear is a combination of a decomposition scheme and a linear network. It first decomposes a time series data into a trend component X_t and seasonal component $X_s = X - X_t$. Then, two one-layer linear networks are applied to these two series.

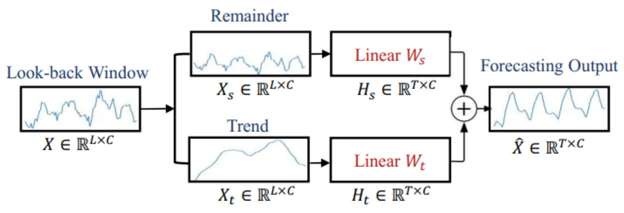
Remainder
Look-back Window
Linear W_s
Forecasting Output
$X_s \in \mathbb{R}^{L\times C}$
$H_s \in \mathbb{R}^{T\times C}$
$X \in \mathbb{R}^{L\times C}$
$\hat{X} \in \mathbb{R}^{T\times C}$
Trend
Linear W_t
$X_t \in \mathbb{R}^{L\times C}$
$H_t \in \mathbb{R}^{T\times C}$

Figure 2: The whole structure of DLinear.

The overall structure of DLinear is shown in Figure 2. The whole process is: $\hat{X} = H_s + H_t$, where $H_s = W_s X_s$ and $H_t = W_t X_t$ are the decomposed trend and remainder features. W_s and W_t are two linear layers, as illustrated in Figure 3.

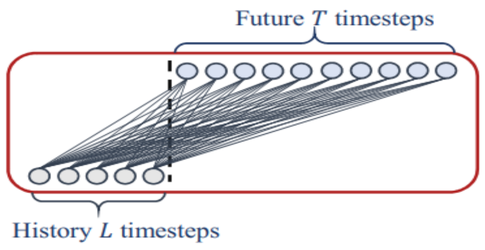

Figure 3: One Linear Layer.

4 EXPERIMENTAL DETAILS

The project has eight python files and two folders. One folder namely "checkpoints" is used to save the model and if the folder does not exist it will be created automatically. Another one folder "logs" is used to log the hyper parameters each time the model is trained. All python code files are detailed below.

4.1 prepare.py

The prepare.py file implements the prep_env() interface to prepare the experimental settings. Next we explain the meaning of each hyper parameter. See the table1 below for details.

4.2 dataset.py

There are three classes "Scaler", "WPFDataset", "TestDataset" in dataset.py.

"Scaler" is used to calculate the mean and variance of the data and save them. When we need normalization or inverse normalization, just call method "transform" or "inverse_transform" in the class.

Table 1: The Meanings of Hyper Parameters in prepare.py

Hyper Parameter	Meanings
path_to_test_x	Online test set path
data_path	Folder path to store data
filename	data file name
input_len	Length of sequence input to model
output_len	Length of sequence output from model(288)
label_len	Length of overlap between input and output
start_col	Start column of data acquisition
capacity	Number of wind turbine
patient	Determine earlystopping
day_len	Sequence length in one day(144)
train_size	Days to use as training set
val_size	Days to use as validation set
is_debug	Whether the model is being trained
checkpoints	Folder path is used to save the model
logs_path	Folder path is used to save the logs
num_workers	Number of threads to load data (batch)
train_epochs	Number of epochs
batch_size	Size of each batch
log_per_steps	Steps to record logs
lr	Learning rate
lr_adjust	Patterns of lr changes, default "type1"
gpu	Gpu id to use
name	Loss function, default "FilterMSELoss"
pred_file	Python file to predict
framework	PaddlePaddle

"WPFDataset" is used to process the raw data and divide the training set and validation set. The hyper parameters "data_path", "filename", "flag", "size", "train_days", "val_days"need to be determined when instantiating the class. Operation when "TestDataset" is instantiated can refer to "WPFDataset" .

4.3 DLinear.py

The model is defined in DLinear.py. There are two classes "SeriesDecomp", "WPFModel" in it. "SeriesDecomp" implements the function of series decomposition to get seasonal and trend component. Hyper parameter "DECOMP"(i.e."kernel_size", 18 by default) is required when instantiating "SeriesDecomp". The class "WPFModel" is very simple, since there are only two linear layers that make separate predictions for seasons and trends from the class "SeriesDecomp".The sum of the outputs of the two linear networks serves as the final predictions

4.4 common.py

Two functions "Adjust learning rate", "EarlyStopping" are implemented in common.py. When adjusting the learning rate, there are two modes to choose from according to the specific environment. The class "EarlyStopping" implements the functions of saving the model and judging whether to stop training.

4.5 loss.py

In loss.py file, the bad input is filtered out on the basis of MSE to achieve a unique loss function "FilterMSELoss". Filtering out inputs that meet the following conditions does not count as loss:

- Pab1 > 89° or Pab2 > 89° or Pab3 > 89°
- Ndir > 720° or Ndir < -720°
- Widr > 180° or Widr < -180
- Patv < 0 and Wspd > 2.5

The above conditions are implemented by paddle. logical_or() and logical_and().

4.6 utils.py

The function _create_if_not_() is implemented in utils.py to create a folder, which is convenient to call when saving models and logs.

4.7 optimization.py

The optimizer Adam is defined in optimization.py, which is convenient to call when training the model.

4.8 train.py

Create train_loader, val_loader from train_dataset and val_dataset, and augment the data before feeding it into the model. The method of data augment follows from the baseline, which is Regression SMOTE.

The data fed to the model is batch_x with a shape of(batch_size, 134, input_len). Similarly, the shape of the output of the model is (batch_size, 134, output_len). Calculate loss using outputs and labels and gradient descent with optimizer Adam.

On the other hand, during validation, the output of the model is evaluated using function regressor_detailed_scores()(metrics.py).The validation score determines the quality of the model. When the score does not increase in the patience epoch, it will stop early and save the best model.

4.9 predict.py

When making inference, we need to first load the model from the checkpoints folder. Load the online test data from "path_to_test_x", and then We use the normalization parameters of the training set to normalize the online test set. Feed test data into the model to get predictions.

5 HOW TO RETRAIN OR INFER?

Code is available at https://github.com/ChaoqunSu/kddcup. Before retraining or inferring, please install the libraries as follows:

- PaddlePaddle-gpu==2.3.0
- Pandas
- Numpy

You can finish it by "pip install -r requirements.txt".

If you want to retrain the model, please make the corresponding changes to the hyper parameters of "prepare.py" according to your needs first. And then "python train.py".

After training the model, you can view some information such as hyper parameters during training and loss per epoch through the .txt file in the "logs" folder.

If you want to make predictions, please specify the hyper parameter "path_to_test_x " and then "python predict.py".

REFERENCES

[1] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Advances in Neural Information Processing Systems, 34, 2021* (June 2021).

[2] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2022. Are transformers effective for time series forecasting? (May 2022). https://doi.org/10.48550/arXiv.2205.13504

[3] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. *In The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference* 35 (2021).

[4] Jingbo Zhou, Xinjiang Lu, Yixiong Xiao, Jiantao Su, Junfu Lyu, Yanjun Ma, and Dejing Dou. 2022. SDWPF: A Dataset for Spatial Dynamic Wind Power Forecasting. *In Proceedings of The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (Baidu KDD Cup 2022)* (March 2022).

[5] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. *In International Conference on Machine Learning* (Feb. 2022).