Applied Energy 333 (2023) 120565

Contents lists available at ScienceDirect

Applied Energy

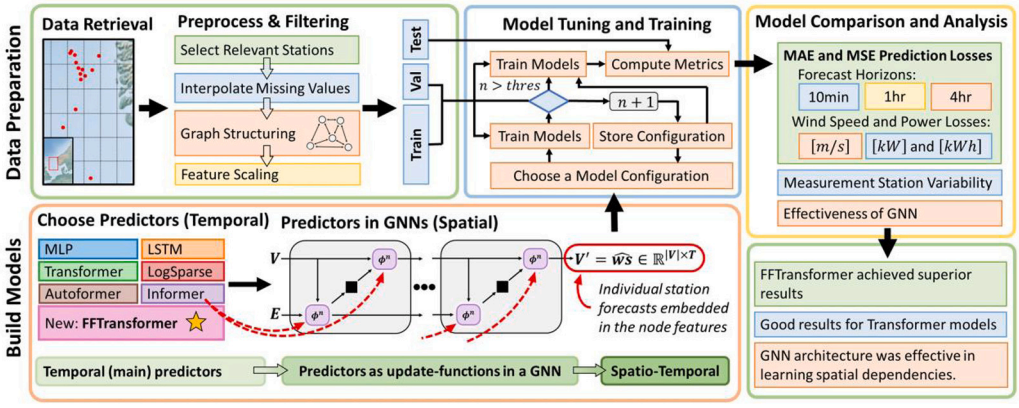
journal homepage: www.elsevier.com/locate/apenergy

Spatio-temporal wind speed forecasting using graph networks and novel Transformer architectures

Lars Ødegaard Bentsen [a,*], Narada Dilp Warakagoda [a], Roy Stenbro [b], Paal Engelstad [a]

[a] Department of Technology Systems, University of Oslo, P.O. Box 70, Kjeller, 2027, Viken, Norway
[b] Institute for Energy Technology, P.O. Box 40, Kjeller, 2027, Viken, Norway

GRAPHICAL ABSTRACT



ARTICLE INFO

Dataset link: https://frost.met.no/index.html

Keywords:
Spatio-temporal wind forecasting
Multi-step
Transformers
Graph neural networks

ABSTRACT

To improve the security and reliability of wind energy production, short-term forecasting has become of utmost importance. This study focuses on multi-step spatio-temporal wind speed forecasting for the Norwegian continental shelf. In particular, the study considers 14 offshore measurement stations and aims to leverage spatial dependencies through the relative physical location of different stations to improve local wind forecasts and simultaneously output different forecasts for each of the 14 locations. Our multi-step forecasting models produce either 10-minute, 1- or 4-hour forecasts, with 10-minute resolution, meaning that the models produce more informative time series for predicted future trends. A graph neural network (GNN) architecture was used to extract spatial dependencies, with different update functions to learn temporal correlations. These update functions were implemented using different neural network architectures. One such architecture, the Transformer, has become increasingly popular for sequence modelling in recent years. Various alterations have been proposed to better facilitate time series forecasting, of which this study focused on the Informer, LogSparse Transformer and Autoformer. This is the first time the LogSparse Transformer and Autoformer have been applied to wind forecasting and the first time any of these or the Informer have been formulated in a spatio-temporal setting for wind forecasting. By comparing against spatio-temporal Long Short-Term Memory (LSTM) and Multi-Layer Perceptron (MLP) models, the study showed that the models using the altered Transformer architectures as update functions in GNNs were able to outperform these. Furthermore, we propose the Fast Fourier Transformer (FFTransformer), which is a novel Transformer architecture based on signal decomposition and consists of two separate streams that analyse the trend and periodic components

* Corresponding author.
E-mail address: l.o.bentsen@its.uio.no (L. Bentsen).

https://doi.org/10.1016/j.apenergy.2022.120565
Received 8 September 2022; Received in revised form 6 December 2022; Accepted 19 December 2022
Available online 28 December 2022

separately. The FFTransformer and Autoformer were found to achieve superior results for the 10-minute and 1-hour ahead forecasts, with the FFTransformer significantly outperforming all other models for the 4-hour ahead forecasts. Our code to implement the different models are made publicly available at: https://github.com/LarsBentsen/FFTransformer.

1. Introduction

In the context of the global climate debate, wind has emerged as a prominent renewable energy resource to accelerate the depletion of fossil fuel-based energy production [1]. Nevertheless, in contrast to conventional power plants, wind resources are inherently intermittent in the short-term, which poses significant challenges for operators and grid planning [2]. To alleviate some of these and help facilitate large-scale adoption of wind power, accurate wind forecasting has become of critical importance.

Wind forecasting methods can be categorised as either physical or statistical. Physical models are based on detailed physical laws that model the atmosphere and typically aim to increase the resolution of coarse numerical weather prediction (NWP) models. A challenge with physical models is that they come at a very high computational cost, making them less viable for local short-term forecasting [3]. Statistical and machine learning (ML) methods, on the other hand, leverage historical data to optimise model parameters. Even though the training of ML models might be a time-exhaustive process, such models are very quick during inference, meaning that forecasts can be obtained in near real-time.

This paper focuses on spatio-temporal multi-step wind forecasting based on recent developments in deep learning (DL). With multi-step forecasting, the study aims to output more informative time series. Considering a scenario of forecasting one hour ahead, some studies give a single prediction for the average wind speed over the entire period. However, sometimes one might also be interested in the development of the wind within that hour, i.e. is the wind speed increasing or decreasing, when do the highest or lowest wind speeds occur and what are the expected peaks. Furthermore, since wind power is proportional to the wind speed cubed, $P \propto ws^3$, higher resolution wind speed forecasts are necessary to more accurately obtain the expected wind power for a particular time period. Bearing this in mind, this study decided to focus on multi-step forecasting with a 10-min time resolution for all forecasts of 10-min, 1-h and 4-h ahead.

The contributions of this paper can be summarised as:

- We show the effectiveness of a generic framework for multi-step spatio-temporal forecasting, with GNNs to capture spatial correlations and optional update functions to learn temporal dependencies, such as a Transformer or LSTM network.
- Test and compare the performance of different Transformer architectures for use in wind forecasting, namely the vanilla Transformer, LogSparse Transformer, Informer and Autoformer. This is the first paper to formulate many of these in a GNN setting and the first to apply such models to wind forecasting.
- We propose a new alteration of the Transformer architecture, namely the Fast Fourier Transformer (FFTransformer), and show its competitive performance in wind forecasting. The novel architecture is based on wavelet decomposition and an adapted Transformer architecture consisting of two streams. One stream analyses periodic components in the frequency domain with an adapted attention mechanism based on fast Fourier transform (FFT), and another stream similar to the vanilla Transformer, which learns trend components.

2. Related works

Autoregressive moving average methods, such as the autoregressive integrated moving average (ARIMA) model, are robust and easy to implement, making them popular for wind forecasting. Kavasseri et al. [4] studied *fractional*-ARIMA models to perform one- and two-day-ahead wind speed forecasts, managing to outperform both a persistence and an ARIMA model for four potential wind generation sites in North Dakota. The persistence model is a commonly used benchmark in wind-speed forecasting, where the forecasted values, $\hat{ws}_{t+1}$ are simply taken as the last recorded value ws_t, i.e. $\hat{ws}_{t+1} = ws_t$. Since wind speed time series are characterised by both long-term trends and high-frequency variation, Singh et al. [5] proposed the RWT-ARIMA model, combining the ARIMA model with wavelet transform (WT) to decompose the signal into multiple sub-series with different frequency characteristics. Various decomposition techniques have been studied for wind time series, showing the potential benefits of introducing some signal decomposition into the forecasting models, such as complete ensemble empirical mode decomposition [6], variational mode decomposition [7] and wavelet packet decomposition [8,9].

Support vector regressor (SVR) and K-nearest neighbour (KNN) algorithms have also been popular within wind forecasting [10–12]. The KNN-algorithm is based on finding similar points in the available data and can be fast in both training and testing. SVRs have been shown to yield very good forecasting performance [11], but do not scale well for larger datasets, resulting in longer computation times [10].

In this study, we focus on neural network architectures, which lie at the heart of modern ML and have become increasingly popular for wind speed forecasting, due to their ability to model non-linear relationships. Multi-Layer Perceptrons (MLP) have been successfully used both in isolation [13,14] and in combination with other methods [15,16]. Recurrent (RNN) and convolutional neural networks (CNN) represent the quintessential DL architectures for sequence modelling and are typically favoured over MLPs for wind forecasting [17]. The long short-term memory (LSTM) unit is an alteration of the vanilla RNN architecture, where gating mechanisms and skip connections are introduced to mitigate the problem of vanishing or exploding gradients [18]. Li et al. [19] proposed a hybrid architecture, using empirical wavelet transform and the regularised extreme learning machine, together with an LSTM as the main predictor. Due to the recurrent architecture, the LSTM network relies on encoding all the relevant input information into a fixed-length memory cell, which can cause information loss and limit the network's ability to retain information across longer sequences. Within the context of natural language processing (NLP), the attention mechanism was introduced by Bahdanau et al. [20], to allow the networks to directly attend to previous hidden states according to their importance. Many studies have focused on integrating attention mechanisms with LSTMs to further help the models learn long-term dependencies that can improve forecasting performance [21,22].

Oord et al. [23] proposed the WaveNet architecture for generating raw audio waveforms. The main ingredient of WaveNet is dilated causal convolution, which is a 1D convolutional operation where the causality ensures that the model cannot violate the sequence ordering, while the dilation increases the receptive field by skipping input values with a certain step. Dilated causal convolution is well suited for time series modelling and has been successfully deployed for some wind forecasting studies [24,25]. Other popular DL-based architectures used for wind forecasting also include deep belief networks [26], RNNs with Gated Recurrent Units [27] and the N-BEATS model [28].

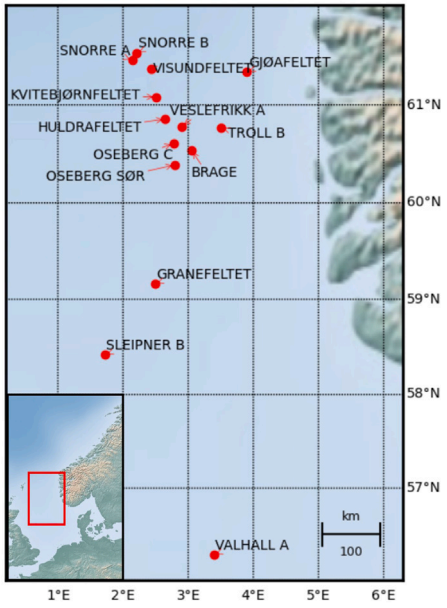


Fig. 1. The 14 measurement stations in the North Sea used to construct the dataset.

Building on the attention mechanism, the Transformer was proposed by Vaswani et al. [29], as a new sequence transduction model, particularly focused on NLP. The Transformer is fundamentally different from previous models in that it does not rely on recurrence or convolution, making it better at learning long-term dependencies. However, since the complexity scales quadratically with sequence length, various alterations of the original architecture have been proposed to alleviate the computational limitations and make the models better suited for time series forecasting, such as the Longformer [30], FEDformer [31], Temporal Fusion Transformer [32], LogSparse Transformer [33], Informer [34], Reformer [35] and Autoformer [36]. Both [37,38] managed to outperform LSTM models for wind forecasting by using a Transformer, comprised of an encoder and decoder, as in [29]. A bidirectional LSTM-Transformer model achieved superior results compared to a gated recurrent unit (GRU) and an LSTM model in [39]. Wang et al. [40] proposed a model based on the Informer together with convolutional layers that extract temporal features at different frequencies, to forecast the average wind power over the next three hours. The Spatial–Temporal Graph Transformer Network (STGTN) extends the previous research by leveraging both spatial and temporal correlations within a wind farm, to more accurately forecast wind speeds at a turbine level, 10 min - 1 h ahead [41]. Despite some efforts at employing different Transformer-based architectures for wind forecasting, namely the vanilla Transformer and Informer, the research have nevertheless been relatively scarce. In this study, we therefore aim to further research the performance of different Transformer architectures, investigating the Informer, Autoformer and LogSparse Transformer, which are yet to be thoroughly tested for wind forecasting.

Spatial dependencies can be important for improving meteorological forecasts, such as for wind. Considering the 14 measurement stations depicted in Fig. 1 used for this study, spatio-temporal forecasting aims to leverage spatial dependencies between different stations, through the physical distance between them, to improve the local forecasts for the different sites. The spatio-temporal interdependence of different physical locations can be particularly important for meteorological forecasts since physical properties such as wind fields, are non-stationary in both space and time, meaning that historical time series for different physical locations should be jointly considered to better learn global trends and propagation in both space and time. Some studies organise spatial data, i.e. the physical location of different

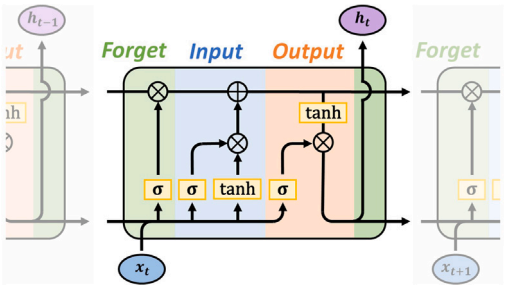


Fig. 2. Visualisation of an RNN with LSTM units having forget, input and output gating mechanisms.

measurement points, into an ordered grid, where the features for a particular location are assigned to a specific cell [42–44]. A CNN is then used to extract spatial features, while another network, such as a CNN [42] or LSTM [43], is used to learn temporal correlations. However, considering the complex topology of the different measurement stations in Fig. 1, the strict ordering of the input data required for CNNs might not be able to effectively represent the underlying spatial relationships. Graph neural networks (GNN) can better facilitate arbitrary spatial ordering and have therefore been popularly used for spatio-temporal forecasting. Khodayar et al. [45] made forecasts at different wind sites simultaneously, where each site was represented by a node in an undirected graph, using a Graph Convolutional Network (GCN) and an LSTM to extract spatial and temporal features, respectively. Similarly, Stańczyk et al. [46] also used a GCN, but with a CNN to learn temporal dependencies. Instead of using static edge features, Wang et al. [47] constructed edge features based on the time-varying spatial correlation between wind sites and used a GCN for wind farm cluster power forecasting. Finally, the M2STAN model was also proposed for spatio-temporal multi-step wind power forecasting [48], which employs a Graph Attention Network (GAT) and a Bidirectional GRU for spatial and temporal correlation modelling, respectively, along with a Transformer network for multi-modal feature fusion. This study will further build on the ideas of using GNNs for spatio-temporal forecasting, but also focus on recent advancements in Transformer-based architectures for time series analysis. Furthermore, the FFTransformer is also proposed, which is a novel Transformer architecture for time series forecasting, based on signal decomposition and learning trend and periodic components separately. Finally, even though many studies focus on single-step forecasts, predicting average wind speeds over some future interval, we also aim to improve predictions by considering multi-step forecasting, producing higher resolution time series for the forecasting windows.

3. Theory

3.1. Multi-layer perceptron

MLPs are feed-forward networks that learn weights, θ, which map the input to output, $y \approx f(x|\theta)$. A chain structure, where multiple layers are stacked, gives depth to the model, as $\hat{y} = f^{(3)}(f^{(2)}(f^{(1)}(x|\theta_1)|\theta_2)|\theta_3)$, for a model with two hidden layers. To improve the learning ability of the model, non-linearities such as the ReLU or sigmoid functions, are applied to the neuron outputs. Optimal weights are determined by minimising a differentiable loss function, using backpropagation, which will update network weights by propagating gradients of the weights with respect to the loss function, back through the network [49].

3.2. LSTM

Even though sequence analysis using DL has largely been dominated by RNNs, the original architecture is prone to exploding or vanishing gradients, resulting in significant information loss for longer

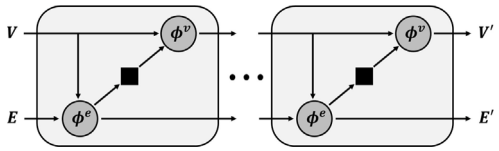


Fig. 3. Visualisation of the GNN architecture for graphs with edge and node features. $\phi^{(\cdot)}$ and ■ represent the update functions and edge-to-node aggregation, respectively.

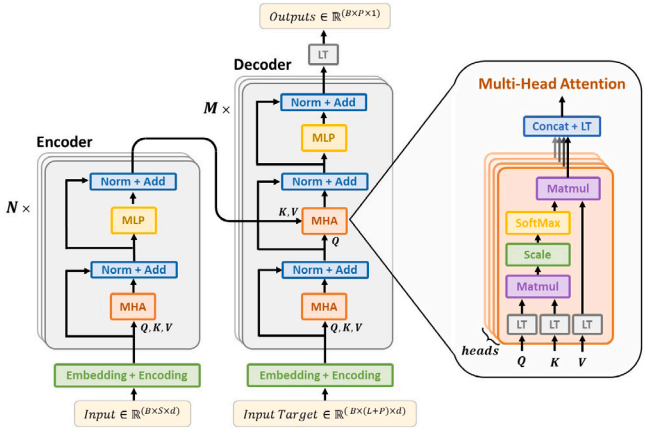


Fig. 4. The Transformer architecture [29] in an encoder–decoder setting, adapted to facilitate forecasting rather than categorical outputs, with multivariate inputs and univariate outputs. B in the inputs and outputs refer to the batch size, for a single prediction $B = 1$.

sequences. The long short-term memory (LSTM) unit introduces gating mechanisms and skip connections to alleviate some of these shortcomings [18]. An illustration showing the internal workings of an LSTM unit is given in Fig. 2, where an input gate controls the contribution of the new input, x_t, to the memory, while the forget and output gates control what information to be kept in memory and encoded in the output, h_t, respectively.

3.3. Transformer architectures

The Transformer architecture was proposed as a model for sequence analysis, without any recurrence or convolution, but instead based on the attention mechanism [29]. If we consider a time series forecasting task with d input features to predict a single output feature, the Transformer model should take an input, $x^{(e)} \in \mathbb{R}^{S\times d}$, and produce an output, $y \in \mathbb{R}^{P\times 1}$, where, S, is the look-back window and, P, the forecasting horizon. The original architecture consists of an encoder and a decoder, as shown in Fig. 4, where the encoder should encode a longer input into a hidden state representation for the decoder to decode. Inputs to the decoder, $x^{(d)} \in \mathbb{R}^{(L+P)\times d}$, are typically shorter than those to the encoder, containing the last L elements of the encoder inputs, where $L < S$, and some placeholders are used in place for the last P-indices, which correspond to the forecasting locations.

Inputs are first linearly transformed to E-dimensional space and then added with some positional encoding, so that the model can make use of the sequence order, without any recurrence. Unless otherwise is stated, it will be assumed that the positional encoding added to the encoder and decoder inputs will be the sine–cosine positional encoding proposed in the original architecture [29]. The multi-head attention (MHA) block in the encoder employs full self-attention, where each attention operation can attend to the full input sequence. Scaled dot-product attention takes Q, K and V as inputs, which represent the queries, keys, and values, respectively. Dot products between keys and queries are computed, before being passed to a softmax function to obtain the attention weights, which are then multiplied by the values to produce the final outputs. This process can be summarised as

$$\text{Attn}(Q, K, V) = \text{softmax}(\frac{(QW^Q)(KW^K)^T}{\sqrt{d_k}})(VW^V), \quad (1)$$

where $W^{(\cdot)} \in \mathbb{R}^{E\times d_{(\cdot)}}$ are weights of the different linear transformations (LT), E is the hidden dimensionality and, d_k, the dimension of keys and queries. Performing multi-head attention, with h separate projections, allows the module to simultaneously attend to different information in the input series. Outputs from the MHA are then concatenated and linearly projected to produce a single output.

A residual connection and layer normalisation are applied to the outputs before the signal is passed to an MLP, typically with a single hidden layer, which is applied to each position in the series separately. Multiple encoder layers, with different weights, are stacked to add depth to the model and hence a stronger function approximation ability.

The decoder follows almost the same architecture as the encoder, but with an additional MHA block, referred to as the cross-attention, which precedes the MLP. The cross-attention module is the same as the other MHA blocks but takes the encoder outputs as inputs to the keys and values. Additionally, the first MHA block in the decoder use masking to prevent information flow from subsequent positions.

Even though the attention mechanism alleviates the problem of information loss for longer sequences, by allowing every position to directly attend to all other positions, its complexity scales quadratically with sequence length. Furthermore, since the Transformer was originally developed for machine translation and other NLP tasks, various modifications have emerged, which aim to both combat the complexity limitations and further adapt the architecture to facilitate time series forecasting problems. A short summary of some of these will now be given in the subsequent sections.

3.3.1. LogSparse Transformer [33]

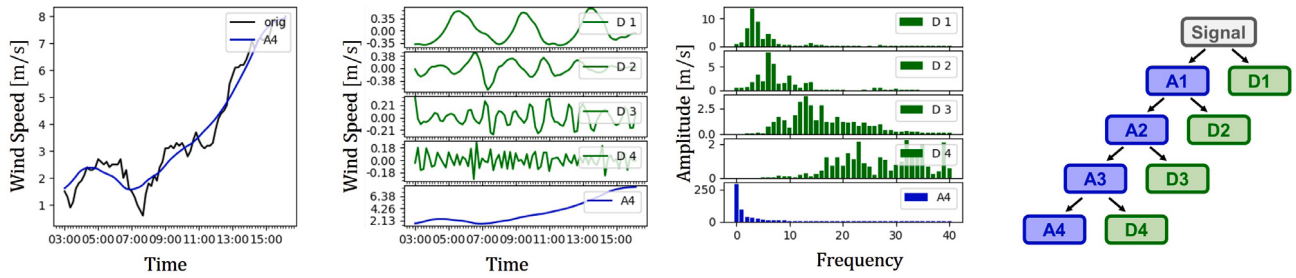
The LogSparse Transformer introduces two novel alterations. Sparse attention, where each position is only allowed to attend to other positions with an exponential step size and itself, significantly reduces the space complexity. Since the point-wise attention operation described in Section 3.3 is insensitive to local context, causal 1D-convolution was used to compute keys and queries, instead of point-wise linear transformations. The modified transformation of keys and queries, which will here be referred to as convolutional attention, might be particularly advantageous for time series forecasting, as local context could be very important for signals characterised by high-frequency fluctuations or noise.

3.3.2. Informer [34]

Instead of introducing sparsity through a fixed pattern decided by heuristic methods, the *ProbSparse* self-attention mechanism proposed for the Informer introduces sparsity by locating the most dominant queries and only allows keys to attend to these. Dominant queries are taken as those that maximise a surrogate for the KL-divergence between a uniform distribution and the query's attention probability distribution. Furthermore, self-attention distilling in the encoder highlights dominant attention by halving cascading layer inputs through 1D-convolution and MaxPooling, which makes the model much more efficient for very long sequences.

3.3.3. Autoformer [36]

Unlike the Informer and LogSparse Transformer, Wu et al. [36] proposed significant alterations to both the overall Transformer architecture and the attention module, to better facilitate time series forecasting. Instead of the scaled dot-product attention, the Autoformer introduces the Auto-Correlation mechanism, which uses keys and queries to decide on the most important time-delay similarities through autocorrelation and a time-delay aggregation, which rolls the series according



(a) Approximation and Input (b) Inverse MDWD (c) ABS(FFT(Inverse MDWD)) (d) MDWD Architecture

Fig. 5. MDWD applied to a wind speed time series. The FFT outputs clearly exhibit the different frequency characteristics of the sub-components and plotting the approximation together with the input signal visualises how this component retains the trend information.

to the selected time delays, before adding them together to produce the outputs. Different to point-wise attention, the Auto-Correlation mechanism finds dependencies based on periodicity and is specifically designed for time series forecasting. Series decomposition is applied after every Auto-Correlation and MLP module, using average pooling to decompose a signal, X, into trend and seasonal components, X_t and X_s, respectively:

$$X_t = \text{AvgPool}(\text{Padding}(X)) \tag{2}$$

$$X_s = X - X_t. \tag{3}$$

3.4. Graph neural networks

A graph can be defined as a tuple with node- and edge-specific features, $G = (V, E)$ [50]. For the spatial forecasting problem, node features, $v_i \in V$, can represent attributes associated with a particular measurement station, while edge features, $e_{ij} \in E$, contain information on the relationship between two nodes, i and j, such as the Euclidean distance between two stations. A GNN is comprised of stacked graph blocks, which perform per-edge and per-node updates in the following order:

$$e'_{ij} = \phi^e(e_{ij}, v_i, v_j) \tag{4}$$

$$v'_j = \phi^v(v_j, \bar{e}'_j), \tag{5}$$

where $\phi^{(\cdot)}$ are the update functions, which could be represented by a neural network such as an LSTM, MLP or Transformer. $(\cdot)'$ and $\bar{(\cdot)}$ represent updated and aggregated features, respectively. Aggregated edge features, $\bar{e}'_j$, are computed using an aggregation function, $\rho^{e\to v}$, as

$$\bar{e}'_j = \rho^{e\to v}(E'_j), \text{ where } E'_j = \{e'_{ij} | \forall i \in \mathcal{R}_j\}, \tag{6}$$

which could for example be a sum or mean operation. $\mathcal{R}_j$ is an index set containing all nodes sending to node j, and defines the connectivity of the graph. A visualisation of the GNN architecture is given in Fig. 3.

3.5. Multilevel wavelet decomposition

Discrete wavelet decomposition (DWD) is a method which can be used to decompose a time series signal into its trend and periodic components, referred to as the approximate and detail signals, respectively, using low- and high-pass filters. An input signal $x = x_1, x_2, \dots, x_S$ is decomposed by convolving a low pass filter $l = l_1, l_2, \dots, l_K$ and a high pass filter $h = h_1, h_2, \dots, h_K$, where $K \ll S$, over the input:

$$\text{A1}_i = \sum_{k=1}^{K} x_{2i+k-1} \cdot l_k \tag{7}$$

$$\text{D1}_i = \sum_{k=1}^{K} x_{2i+k-1} \cdot h_k, \tag{8}$$

where A1_i and D1_i are the ith elements of the approximate and detail components obtained from a single level discrete wavelet decomposition. DWD can also be stacked in multiple layers, referred to as multilevel DWD (MDWD), to extract multiple periodic components of different frequency characteristics. In a multi-layer setting, the approximate component from one layer is fed as input to the next, resulting in multiple detail and a single approximate signal, as shown in Fig. 5(d), where 'D1', 'D2', 'D3' and 'D4' are the detail components and 'A4' the approximation. The Daubechies wavelet, Db4, for the low- and high-pass coefficients, l and h in Eqs. (7) and (8), has been shown to be suitable for wind forecasting [51]. The decomposition of a wind speed time series is shown in Fig. 5(b), where inverse MDWD is applied independently to the outputs from the MDWD. By applying FFT to each time series in Fig. 5(b), it can be seen in Fig. 5(c) that detail (D1, D2, D3, D4) and approximate (A4) components yield very different frequency characteristics. The four detail components exhibited clear periodic information, with peaks at different frequencies, while the approximation, 'A4', contained most of the low-frequency trend information, clearly visualised in Fig. 5(a). As a result, we argue that MDWD is a suitable decomposition method for wind speed time series, where the special filters and multiple layers potentially allow us to extract more informative trend and periodic information than simpler decomposition methods, such as series decomposition.

4. Methodology

4.1. Dataset

Due to the future potential for offshore wind energy [52], this study decided to focus on offshore wind speed forecasting, using meteorological measurements recorded on the Norwegian continental shelf. The data was made available by the Norwegian Meteorological Institute and is openly available through the Frost API.[1] Ten-minute averaged measurements on air temperature, air pressure, dew point, relative humidity, wind direction and speed and maximum wind speed in the 10-min interval were used as input features to forecast the wind speed time series. Wind direction was decomposed into its sine and cosine components in order to fully capture the circular characteristics. For every time-step, we would therefore have a feature matrix, $f_t \in \mathbb{R}^{N\times 8}$, corresponding to the eight recorded measurements for N available stations. The forecasting problem then becomes, $\hat{V}_{(t+1)}, \hat{V}_{(t+2)}, \dots, \hat{V}_{(t+P)} = F(f_{(t)}, f_{(t-1)}, \dots, f_{(t-(S-1))})$, where F is the model, P the number of future time-steps to forecast, $\hat{V}_{(t+j)} \in \mathbb{R}^{N\times 1}$ the predicted wind speeds at time, $(t+j) \forall j \in (1, 2, \dots, P)$, and S is the look-back window, i.e. the number of previous time-steps used to make the forecasts. For the period between June 23, 2015, and February 28, 2022, 14 out of 25 available stations had some periods with measurements on all the relevant meteorological variables, and are shown in Fig. 1.

[1] https://frost.met.no/index.html.

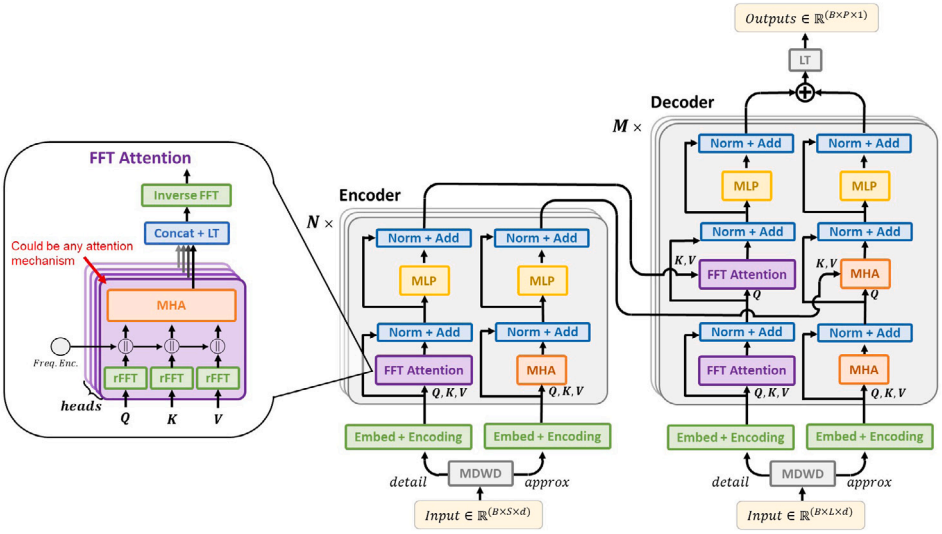


Fig. 6. FFTransformer in an encoder–decoder setting. Two streams focus on trend and periodic signal components separately. The FFT-Attention is employed for the periodic analysis in the frequency domain, using an MHA mechanism together with FFT.

The first 60% of the data was used for training, whereas the following 20% and the remainder were used for validation and testing, respectively. If measurements for a single time-step were missing for a particular station, linear interpolation was used to fill the missing entries. However, if there were consecutive time-steps missing, the station would not be considered for these periods. For different periods, there would be a variable number of stations that had available data, meaning that the models should be able to take any subset of the 14 stations as input.

4.2. Fast Fourier Transformer

Since many time series, such as wind measurements, are characterised by both trend and periodic components, we propose the Fast Fourier Transformer (FFTransformer), based on signal decomposition and an adapted attention mechanism, named FFT-Attention. Given the different frequency characteristics for wind speed time series discussed in Section 3.5, the FFTransformer is comprised of two streams, one which analyses signals with clear periodicity and another that should learn trend components, i.e. the detail and approximate signals in Fig. 5(b), respectively. Here, we used MDWD to decompose the input signals into its periodic and trend components, due to its capability to clearly extract different level frequencies using multiple levels of low- and high-pass filters. Nevertheless, the architecture discussed is not limited to MDWD and other methods for signal decomposition could be used, such as the 'Series Decomposition' in the Autoformer.

As shown in Fig. 6, the right-hand stream in both the encoder and decoder were exactly the same as for the original Transformer in Fig. 4. However, to better facilitate the analysis of periodic signals in the left-hand stream, we introduce the FFT-Attention mechanism, which performs attention in the frequency domain. In particular, as the first operation of FFT-Attention, FFT is applied to the key, query and value inputs. Real and imaginary components of the FFT outputs are concatenated with the frequency values, to provide information on the corresponding frequency for the values in a particular position, similar to the motivation behind positional encoding, before being fed to an MHA block. Outputs from the FFT-Attention module are then concatenated and projected, as for the other attention mechanisms, and finally, inverse FFT transform values back to the time domain. Any attention mechanism could be used in place of all the MHA blocks in Fig. 6, such as the *ProbSparse* or convolutional attention. The final network outputs from both streams (FFT Attention and Trend) are at last added and linearly transformed to produce the predictions.

After some experimentation, it was found that not adding temporal or positional encoding to the detail components (i.e. inputs to the FFT-Attention) yielded the best results. This seemed sensible, as adding positional encoding in the time domain would not translate to the frequency domain. Instead, the concatenation of frequency values to the FFT outputs, served somewhat the same purpose as the positional encoding did for the trend components. Nevertheless, since this method was fairly trivial, further research studying better encoding strategies for the frequency domain could be desirable.

4.3. Spatio-temporal framework

All models were constructed in an encoder-only setting, i.e. without decoders, and used as update functions, $\phi^{(\cdot)}$, in two-layer GNNs. To facilitate the GNN framework, the dataset was constructed as graphs. Measurement stations were represented by nodes, with the historical time series of the eight meteorological variables assigned to the input node features. Since there would be a variable number of available measurement stations at different times, the input graphs would not be the same at different times, but with a variable number of nodes. This was desirable, as it meant that we did not have to discard the data for all stations or interpolate missing values for a particular time interval where a few stations had missing data, as would be the typical case for a CNN. Considering a node, i, its input features were $v_i \in \mathbb{R}^{1\times(S+P)\times 8}$, where S is the historical look-back window, i.e. the number of previous time-steps used to predict the next P time-steps. Placeholders were used for the last P indices of the node input features to the first graph block, since we do not have information on recorded values in the future. These values were set to either the last available measurements, meaning the values at position S, or the mean of the input series, i.e. the mean of each feature over positions $(1, 2, \ldots, S)$. Zero-values were used as placeholders for the seasonal inputs to the Autoformer, as well as for the detail components in the FFTransformer model. It should be noted that placeholders were added subsequent to the 'MDWD' and prior to the 'Embed + Encoding' block in Fig. 6, meaning that inputs to the MDWD were $v_i \in \mathbb{R}^{1\times S\times 8}$. Differences in latitude and longitude between stations were assigned to the input edge features as $e_{ij} = [(\text{lon}_j - \text{lon}_i), (\text{lat}_j - \text{lat}_i)]$, for two stations, i and j. A pseudocode summarising the Spatio-Temporal forecasting framework is given in Algorithm 1.

Algorithm 1 Pseudocode for the spatio-temporal forecasting framework. Slicing of tensors follow a Python syntax.

Input:
$V \in \mathbb{R}^{N\times S\times 8}$ ▷ recorded values for the 8 features for previous S times
$T \in \mathbb{R}^{1\times(S+P)\times 4}$ ▷ time stamps for previous (S) and future (P) times, 4 → {minute, hour, day, week}
$E \in \mathbb{R}^{M\times 2}$ ▷ distance in latitude and longitude between neighbouring stations
$\mathcal{R} \in \mathbb{Z}^{M\times 2}$ ▷ index set containing sender and receiver indices
ϕ^v ▷ node update function, e.g. FFTransformer or LSTM
ϕ^e ▷ edge update function
Output:
$\hat{V} \in \mathbb{R}^{N\times P\times 1}$ ▷ predicted wind speeds for future P time steps

if ϕ^v is FFTransformer **then**
  for $d_i = 1, ..., 8$ **do**
    $V_{d_i}^{(A1)}, V_{d_i}^{(D1)}, V_{d_i}^{(D2)}, V_{d_i}^{(D3)}, V_{d_i}^{(D4)} \leftarrow \text{MDWD}(V[:,:,d_i])$ ▷ for a four-layer MDWD (see Section 3.5)
  end for
  $V^{trend} \leftarrow \text{concat}(V_{1:8}^{(A1)})$
  $V^{freq} \leftarrow \text{concat}(V_{1:8}^{(D1:D4)})$
  $V^{trend} \leftarrow \text{concat}(V^{trend}, V^{trend}[:,S,:].\text{repeat}(1,P,1))$ ▷ persistence placeholders to forecast locations
  $V^{freq} \leftarrow \text{concat}(V^{freq}, \text{zeros}(N,P,8))$ ▷ zero-value placeholders to forecast locations
  $V^{freq} \leftarrow NodeEmbed^{freq}(V^{freq})$ ▷ learned linear projection as node and temporal embedding
  $V^{trend} \leftarrow NodeEmbed^{trend}(V^{trend}) + TempEmbed(T) + PosEmbed(S+P)$ ▷ positional embedding as in [29]
  $V^{(0)} \leftarrow [V^{freq}, V^{trend}]$
else
  $V \leftarrow \text{concat}(V, V[:,S,:].repeat(1,P,1))$ ▷ persistence placeholders to forecast locations
  $V^{(0)} \leftarrow NodeEmbed(V) + TempEmbed(T) + PosEmbed(S+P)$ ▷ PosEmbed not used for ST-MLP/LSTM
end if
$E \leftarrow E.repeat(1, S+P, 1)$ ▷ extend edge features to series length
$E^{(0)} \leftarrow EdgeEmbed(E) + TempEmbed(T) + PosEmbed(S+P)$ ▷ PosEmbed not used for ST-MLP/LSTM
for $l = 1, ..., L$ **do** ▷ for L number of stacked graph blocks
  $E^{(l)} \leftarrow \phi^e(E^{(l-1)}, V^{(l-1)})$ ▷ update edge features between stations
  $\overline{e_j^{(l)}} \leftarrow \rho^{e\to v}(e_{ij}^{(l)}) \quad \forall i \in \mathcal{R}_j$ **for all** $j \in (1,2,...,N)$ ▷ aggregate edge features, assumed to be mean
  $V^{(l)} \leftarrow \phi^v(V^{(l-1)}, \overline{E^{(l)}})$ ▷ update node features for each station
end for
Return: $\hat{V} \leftarrow V^{(L)}[:,-P:,:]$

Table 1
Final model parameters after tuning.

Parameter	ST-MLP	ST-LSTM	ST-Transformer	ST-LogSparse	ST-Informer	ST-Autoformer	ST-FFTransformer	Explanation
d	(64, 64, 32)[a]	(16, 32, 64)[a]	(64, 64, 32)[a]	(64, 64, 32)[a]	(64, 64, 32)[a]	(64, 64, 32)[a]	(64, 64, 32)[a]	Input/Output dimension for every GNN layer
d_{hidden}	(256, 256, 128)[a]	(64, 128, 256)[a]	(256, 256, 128)[a]	(256, 256, 128)[a]	(256, 256, 128)[a]	(256, 256, 128)[a]	(256, 256, 128)[a]	Hidden dimensionality of MLP modules
GNN layers	2	2	2	2	2	2	2	Number of graph blocks
Activation	ReLu	ReLu	ReLu	GELU	GELU	GELU	GELU	Activation function for MLP modules
LR	0.001	0.001	0.001	0.001	0.001	0.001	0.001	Learning rate
LR decay	0.8	0.8	0.8	0.8	0.8	0.8	0.8	Learning rate decay after each epoch
BS	32	32	32	32	32	32	32	Batch size
Dropout	0.05	(0.05, 0.05, 0.15)[a]	0.05	0.05	0.05	0.05	0.05	Dropout rate
LSTM/MLP layers	2	(1, 1, 2)[a]	–	–	–	–	–	LSTM/MLP layers in each update function
Heads	–	–	8	8	8	8	8	Number of attention heads
Attn Kernel	–	–	–	6	–	–	3	Kernel for convolutional attention, see: [33]
Local Attn	–	–	–	4	–	–	–	Local attention length, see: [33]
Restart length	–	–	–	16	–	–	–	Restart attention length, see: [33]
Sparse Attn	–	–	No	Yes	Yes	Yes	Yes	Some sparse attention pattern [33,34,36]
c	–	–	–	–	3	3	3	Sampling factor, see: [34,36]
mvavg kernel	–	–	–	–	–	25	–	Moving average for series decomposition [36]
num decomp	–	–	–	–	–	–	4	Number of layers in MDWD, see Section 3.5
Time (s)	0.0021	0.0051	0.0047	0.0054	0.0063	0.0096	0.0172	Average compute time for a 6-step forecast

[a]Parameter values used for the 1, 6 and 24 step forecast models. Values not shown in parenthesis were kept the same for all horizons.

4.4. Experimental set-up

All features were scaled separately using a standard scaler, with zero mean and unit variance. Three forecasting horizons were considered for the multi-step forecasting problem, namely 1-, 6- and 24-step forecasts, corresponding to 10-min, 1-h and 4-h periods. Since the study considered multi-step forecasting, a prediction was made for every 10-min interval also in the 1- and 4-h ahead settings, instead of simply average wind speed forecasts over the respective periods. Models were trained to minimise the mean squared error (MSE) and hyperparameter tuning was conducted in Optuna [53]. All experiments were conducted in PyTorch on a single Nvidia RTX 2070 GPU. Every model was trained for 30 epochs using an Adam optimiser. Both look-back windows and model-specific parameters were treated as hyperparameters and decided based on the tuning procedure and computational constraints, with final model-parameters given in Table 1. From the tuning, it was found that a look-back window of 32 time-steps was suitable for the 10-min and 1-h ahead forecasts, increased to 64 for the 4-h forecasts, which seemed reasonable, as longer contexts might be useful to better predict trends further into the future. Some explanation of the different parameters are given in Table 1, though the authors refer the interested reader to the provided references in the 'Explanation' column for more detailed descriptions of model-specific parameters. In Table 1, d are the model dimensionalities, while d_{hidden} and 'Activation', refer to the dimensionality and activation function for hidden layers in the ST-MLP model and MLP modules in Transformer-based models. All Transformer-based models employed the same architecture for the edge-update functions, $\phi^{(e)}$, namely a single vanilla Transformer encoder layer [29].

For the ST-MLP and ST-LSTM models, both autoregressive and vector-output models were studied, where different training strategies such as mixed teacher forcing was studied for the autoregressive architectures. However, vector-based architectures were found to yield the best results for both the ST-MLP and ST-LSTM models, where the multiple forecasting steps are mapped directly using a linear projection with P outputs corresponding to the forecast locations, as

$$h_t = \text{LSTM}(X_t, X_{(t-1)}, \dots, X_{(t-(S-1))}) \tag{9}$$

$$\hat{V} = LT(h_t). \tag{10}$$

X_t are the input features at time, t, respectively. LT represents the linear projection that projects the last hidden features, $h_t \in \mathrm{R}^{N\times d}$, to the P number of predicted outputs, $\hat{V} \in \mathrm{R}^{N\times P}$, as $LT : \mathbb{R}^d \to \mathbb{R}^P$, where d is the hidden dimensionality of the model.

The ST-LSTM model was quite sensitive to model parameters, explaining the different dimensionalities, d in Table 1, used for this model. Instead of a simple linear projection, it was found beneficial to use a two-layer MLP as the final head for the ST-LSTM model, with parameters given by d, d_{hidden} and 'Activation' in Table 1.

Considering the ST-FFTransformer, the MHA modules in the trend (right-hand) stream in Fig. 6, employed convolutional *ProbSparse* attention, but different to the original *ProbSparse Attention*, which focuses on dominant queries, it was found marginally beneficial to instead locate dominant keys. Convolutional *ProbSparse Attention* was also used for the MHA module in the FFT-Attention, but finding top queries, as in the original formulation. For non-dominant query locations, the original *ProbSparse Attention* formulation assigns mean values to the outputs [34]. However, when using the *ProbSparse Attention* for the FFT-Attention, we instead set the outputs for non-dominant locations to zero to introduce sparsity and only select a subset of the possible frequencies. This was thought desirable, as to avoid outputs with a very large number of frequencies of small or similar amplitudes, which could be considered noisy.

Considering the computational time for each model, it can be seen from the last row in Table 1, that the ST-FFTransformer model was slower than the other models. This was mainly due to the MDWD layer, which was not optimised to run in parallel over the different input features. Furthermore, both the ST-Autoformer and ST-FFTransformer models rely on computing FFTs, which are fairly slow and explain why these took longer to compute forecasts. However, since the time to compute any forecasts was much smaller than one second, and considerably less than the 10-min sampling rate of the data, this was not considered a significant problem for this study and is therefore left for future work where computational speed might be more important.

The persistence model was used as a benchmark against which to compare all the other models. For the spatio-temporal setting, the persistence model will use the last recorded wind speed for a station as the forecast over the entire horizon, as $\hat{V}_{(t+1)} = \hat{V}_{(t+2)} = \cdots = \hat{V}_{(t+P)} = V_t$, using the notation in Section 4.1. Even though this is quite a trivial method for making forecasts, the model can achieve fairly accurate results in the short-term and is therefore used as an important baseline to outperform.

5. Results and discussion

5.1. Forecasting error

To evaluate the predictive performance of the different models, we start by comparing the mean absolute (MAE) and squared (MSE) errors, given by the following equations:

$$\text{MAE} = \frac{1}{n}\sum_{i=0}^{n} |y_i - \hat{y}_i| \tag{11}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n} (y_i - \hat{y}_i)^2, \tag{12}$$

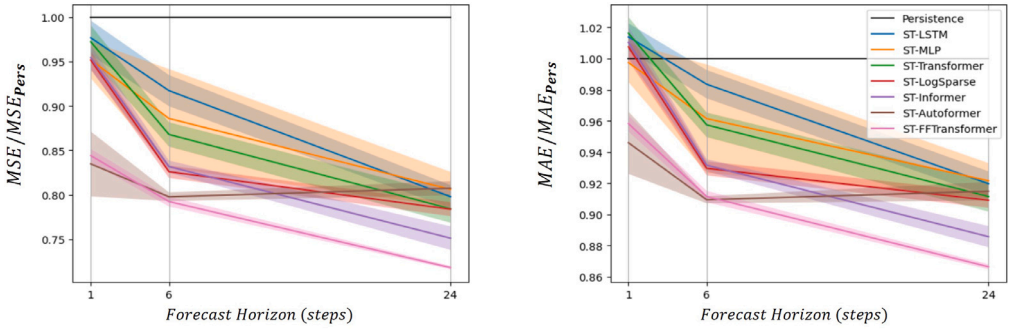
where n is the total number of samples and, $\hat{y}$, the model predictions, which should be close to the targets y. For each forecasting horizon, every model was trained five times, with the average errors on the test set given in Table 2. Before computing the metrics, the predictions and labels were transformed back to a meters-per-second scale for better interpretability of the results. The percentage improvement values in Table 2 are relative to the persistence model and are provided in order to highlight the relative forecasting performances. These are also visualised in Fig. 7, where the shaded regions show the variability from the five training iterations of each model as ± one standard deviation.

Looking at the ST-Transformer model in Table 2, it was evident that the model did not report any remarkable advancements. Compared to the ST-MLP model, the ST-Transformer only showed marginal improvements for the 6- and 24-step forecasts, while being on-par with the persistence and ST-LSTM models for the single-step setting. For the 1- and 6-step forecasts, the ST-MLP model outperformed the ST-LSTM model, while the ST-LSTM only yielded slightly better results than the ST-MLP model for the longer 24-step forecasts. This might indicate that for the immediate short-term forecasts, all the previous time-steps might be relevant, resulting in the fully connected architecture of the ST-MLP achieving better accuracies than the ST-LSTM. However, for longer input sequences and forecast horizons, the LSTM architecture might be better at encoding the useful information, resulting in the improving performance of the ST-LSTM model, compared to the ST-MLP, for the longer forecasting settings. Furthermore, looking at Fig. 7, the ST-MLP model showed significantly more variability between training iterations, compared to the other models. Since this variability was uncontrollable, it meant that one could not reliably conclude on the ST-MLP model's exact performance, which was undesirable.

The ST-LogSparse and ST-Informer performed consistently better than the ST-Transformer model across all forecasting horizons in terms of both MSE and MAE, which showed the potential improvements brought by the *ProbSparse* and convolutional attention mechanisms for wind forecasting. Even though the ST-LogSparse and ST-Informer models reported slightly inferior forecasting performance in terms of

Table 2
Test losses in m/s for different models and forecast horizons.

Model	MSE (% Improvement)						MAE (% Improvement)					
	1-step (10 min)		6-steps (1 h)		24-steps (4 h)		1-step (10 min)		6-steps (1 h)		24-steps (4 h)	
Persistence	0.4138	0.0%	1.0282	0.0%	2.9712	0.0%	0.4277	0.0%	0.6852	0.0%	1.1922	0.0%
ST-MLP	0.3938	4.8%	0.9108	11.4%	2.3951	19.4%	0.4265	0.3%	0.6588	3.9%	1.0987	7.8%
ST-LSTM	0.4040	2.3%	0.9430	8.3%	2.3707	20.2%	0.4336	−1.4%	0.6739	1.7%	1.0964	8.0%
ST-Transformer	0.4022	2.8%	0.8923	13.2%	2.3302	21.6%	0.4346	−1.6%	0.6561	4.3%	1.0866	8.9%
ST-LogSparse	0.3937	4.9%	0.8492	17.4%	2.3290	21.6%	0.4309	−0.7%	0.6369	7.0%	1.0838	9.1%
ST-Informer	0.3948	4.6%	0.8552	16.8%	2.2321	24.9%	0.4321	−1.0%	0.6383	6.8%	1.0559	11.4%
ST-Autoformer	**0.3454**	**16.5%**	0.8203	20.2%	2.3990	19.3%	**0.4046**	**5.4%**	**0.6231**	**9.1%**	1.0907	8.5%
ST-FFTransformer	0.3492	15.6%	**0.8147**	**20.8%**	**2.1336**	**28.2%**	0.4098	4.2%	0.6245	8.9%	**1.0329**	**13.4%**



(a) MSE divided by persistence MSE (b) MAE divided by persistence MAE

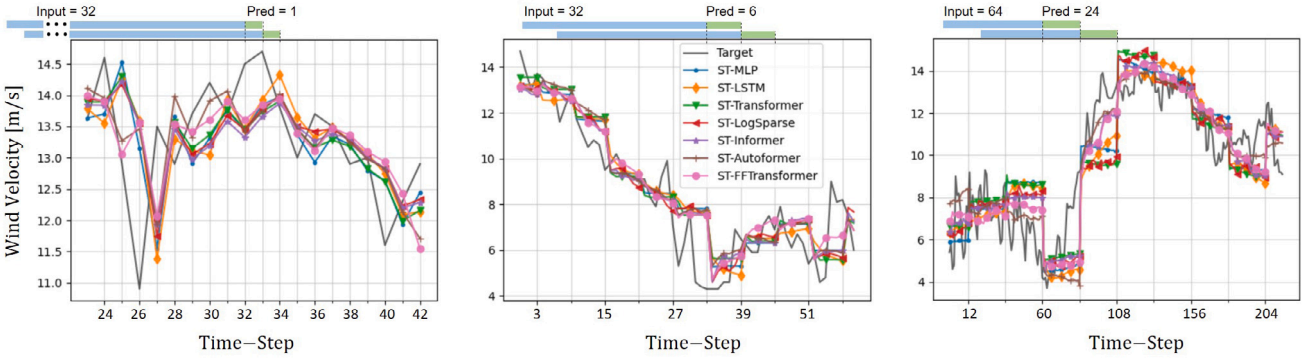
Fig. 7. Visualisation of the test errors for the different forecast horizons (1, 6 or 24 steps) divided by the respective persistence errors. Shaded areas show the ± one standard deviation from the five training iterations of each model.

MAE for the single-step forecasts, compared to the persistence model, both showed approximately a five percent improvement in MSE. Since MSE penalise large errors more heavily than the MAE metric, it meant that for the single-step forecasts, the persistence model had on average fewer slightly smaller errors, but a larger number of drastically wrong predictions than the ST-LogSparse and ST-Informer models.

In general, all Transformer-based models attained better results than the ST-MLP and ST-LSTM models for the multi-step forecasts. The ST-Autoformer and ST-FFTransformer performed remarkably well for the 1- and 6-step forecasts, achieving more than three times the improvement reported for the third best model in terms of MSE for the 1-step forecasts, and nearly a doubling compared to the ST-MLP for the 6-step forecasts. Even though the ST-Autoformer model performed very well for the 1- and 6-step forecasts, its performance was seen to degrade for the 24-step setting, where it was inferior to all other Transformer-based models, clearly visualised in Fig. 7. The ST-FFTransformer on the other hand, continued to improve, achieving superior results compared to all other models also for the 4-h forecasts. Both the ST-FFTransformer and ST-Autoformer achieved the best accuracy for three settings each, likely because of data decomposition. For the 1- and 6-step ahead forecasts, the difference between the two models was very small, while the ST-FFTransformer substantially outperformed the ST-Autoformer for the longer forecasting horizon of four hours. In the short term, wind time series fluctuates rapidly, while for longer time periods, it might become more important to learn trend components as well as oscillations. One reason for the superior performance of the FFTransformer architecture for the longer forecasting setting might be that it separately processes the frequency and trend components, using the two streams described in Section 4.2, with attention and feed forward modules applied to both. On the other hand, the Autoformer model is mainly focused on processing periodic components using the Auto-Correlation module and does not perform significant processing on extracted trend components after series decomposition, making it potentially struggle to learn longer-term trends that lack periodicity. Such a hypothesis was strengthened by the fact that all other models seemed to perform comparatively better for the longer forecasting setting, compared to the ST-Autoformer. Because of the two streams used in the ST-FFTransformer, it is able to perform well for both short-term forecasts, where periodic behaviour is thought more important, as well as for the four-hour forecasts, where non-oscillating trend components become increasingly important. Even though the ST-FFTransformer achieved good results across different horizons, it did not outperform the ST-Autoformer in the short-term. This was despite more advanced decomposition using MDWD, which was initially thought better at extracting trend and periodic components at different frequencies, compared to the simple moving average operation used in the Autoformer. In terms of decomposition, this could indicate that the MDWD is not necessarily superior to the 'Series Decomposition', or more likely, that the repeated use of decomposition in the Autoformer might work slightly better than the single decomposition performed on the inputs in the ST-FFTransformer. For future research it would be interesting to see how the FFTransformer architecture would perform with Auto-Correlation modules in place of the FFT Attention blocks in Fig. 6 and experimenting with some data decomposition applied to every encoder or decoder layer.

Considering the 1-step forecasts, the ST-Autoformer model exhibited considerable variability between training iterations, compared to the ST-FFTransformer, seen by the shaded regions in Fig. 7, which was undesirable. We therefore argue for the competitive forecasting performance of our proposed FFTransformer architecture, which performed consistently well across all forecasting horizons and with limited variability.

Fig. 8 shows some predictions under the different forecast horizons, for some randomly chosen time periods for the Kvitebjørnfeltet measurement station in Fig. 1. Overall, it was difficult to conclude on major differences between the models based on the plotted time series. Nevertheless, it was observed that for the 6-step setting, the ST-MLP and ST-Transformer models often gave near-constant predictions for all six time-steps, while the ST-LSTM, ST-LogSparse, ST-Informer, ST-FFTransformer and ST-Autoformer were able to produce slightly more diverse time series. For the 24-step (4 h) setting, all models showed more near-constant predictions, as seen in Fig. 8(c), where



(a) One-Step Forecasts (10-minute) (b) Six-Step Forecasts (1-hour) (c) 24-Step Forecasts (4-hour)

Fig. 8. Prediction examples of the different models under the 1-, 6- and 24-step forecasts. The vertical grid line spacings correspond to the prediction horizons, i.e. in (b), 6-step predictions are shown with vertical grid spacing 6.

Table 3
Estimated power errors for different models and forecast horizons.

Model	MAE [kW] (% Improvement)						Interval MAE [kWh] (% Improvement)					
	1-step (10 min)		6-steps (1 h)		24-steps (4 h)		1-step (10 min)		6-steps (1 h)		24-steps (4 h)	
Persistence	166.1	0.0%	265.4	0.0%	459.2	0.0%	27.684	0.0%	228.8	0.0%	1604.2	0.0%
ST-MLP	165.1	0.6%	253.2	4.6%	422.0	8.1%	27.509	0.6%	209.4	8.5%	1387.0	13.5%
ST-LSTM	166.0	0.1%	258.5	2.6%	418.9	8.8%	27.669	0.1%	212.0	7.3%	1386.0	13.6%
ST-Transformer	166.4	−0.2%	250.7	5.5%	417.7	9.1%	27.729	−0.2%	206.9	9.5%	1373.0	14.4%
ST-LogSparse	164.7	0.8%	245.5	7.5%	417.4	9.1%	27.453	0.8%	203.5	11.0%	1371.1	14.5%
ST-Informer	165.1	0.6%	245.7	7.4%	405.7	11.7%	27.519	0.6%	204.4	10.7%	1347.3	16.0%
ST-Autoformer	**156.3**	**5.9%**	**239.8**	**9.7%**	416.9	9.2%	**26.046**	**5.9%**	**196.3**	**14.2%**	1383.0	13.8%
ST-FFTransformer	158.5	4.6%	239.9	9.6%	**396.5**	**13.7%**	26.409	4.6%	197.4	13.7%	**1300.6**	**18.9%**

some forecasts exhibit step-like changes at every prediction interval (i.e. 24 time-steps). Forecasting accurate time series for longer horizons is challenging and the near-constant predictions indicate that the models sometimes struggled to confidently predict sharp increases or decreases in wind speeds. Having additional features and measurement stations or much deeper networks together with larger datasets, could potentially enable the models to more accurately learn these long-term changes. However, some of the near-constant predictions might also reflect the inherent stochasticity associated with wind time series, meaning that in some scenarios it might be inconceivable to attain accurate multi-step forecasts for longer horizons, based solely on previous time series and no physical information. Nevertheless, for other time-steps, the ST-LSTM, ST-FFTransformer, ST-Autoformer and ST-Informer models seemed much better than the ST-MLP, ST-Transformer and ST-LogSparse at capturing the overall trends within the intervals in Fig. 8(c).

To investigate the physical interpretation of the forecasting results in relation to wind energy production, two additional MAE metrics were computed and provided in Table 3, which correspond to the estimated errors in kW and kWh. Powers were estimated based on the NREL 5 MW reference wind turbine for offshore system development [54]. By transforming the true and predicted wind speeds to powers using the reference turbine's power curve and then calculating the MAEs, the results show crude estimates for the average power errors using the different models. For the first metric in Table 3, results were fairly similar to those discussed in Table 2, but arguably more interpretable, in terms of understanding the consequence of differing predictive performances and potential risks associated with the proposed models.

Instead of looking at the point-wise power difference between the true and predicted values, an operator might be primarily concerned with the total overall energy for a future time interval. For the Interval MAE, predictions were again transformed to power values, before the values associated with a particular forecast interval were summed. Finally, since each step was associated with a 10-min interval, summed values were divided by six, to convert them to the total energy estimates in kWh. Therefore, the Interval MAE provides an estimate of the difference between the predicted and true total energies produced for the relevant forecast horizon. The performance of different models was similar to previous results, but with all models showing greater improvements over the persistence model. Considering the ST-FFTransformer, it managed to reduce the error by 300 kWh for the 4-h forecasts, corresponding to a 19% improvement over the persistence model. This was fairly notable, as it also meant an additional ≈5% improvement compared to most other models, and illustrated the potential cost savings and benefits of more accurate forecasting models.

5.2. Station predictability

Since the models made forecasts for all 14 stations in Fig. 1, simultaneously, it was thought interesting to inspect the average errors for each station independently. Fig. 9, shows the average MAEs for every station under the 6-step forecasting setting, along with error bars of $\pm 2\sigma$, where σ is the standard deviation computed based on five separately trained models. It was seen that there was significant variability in how well the models were able to make forecasts for the different stations, with MAEs ranging from 0.53 m/s for some stations to 0.74 m/s for others. Even though there were distinct differences in the performance of different models, as discussed in the previous section, stations associated with higher or lower MAEs seemed consistent across all models. The data only contained historic information on the specific measurement stations in Fig. 1, with a fixed physical layout, meaning that spatial features would not change and that the meteorological data was likely to follow a particular distribution, specific to the area considered. As a result, a station might be inherently easier to forecast than others, due to its location relative to surrounding stations and due to the dominant wind fields for this specific geographical area potentially being preferable for forecasting at a particular location.

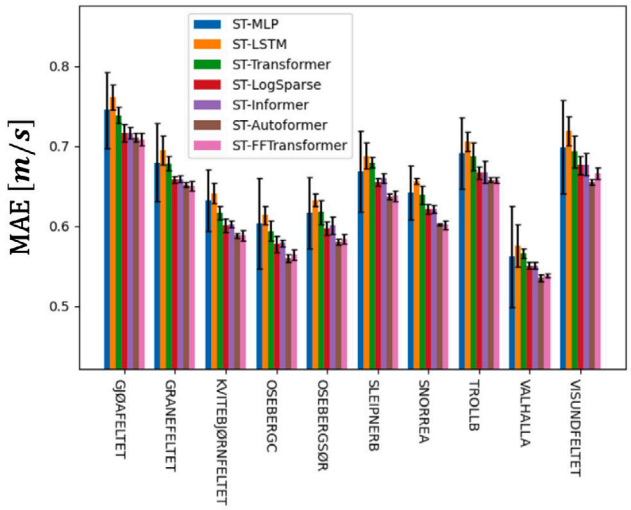


Fig. 9. MAEs for the different available measurement stations in the test set under the 6-step forecasting setting. The error bars are the ±2 standard deviations from the five training iterations of each model.

5.3. Graph connectivity

For the results discussed so far, all input data was constructed as complete graphs, meaning that all nodes had edge features sending to all other nodes in the network and itself. This trivial method for formulating the graphs could result in very large inputs, significantly increasing the computational and memory costs. For instance, if a graph had 14 nodes, it would result in $14^2 = 196$ edges. Some of the edges might be redundant, meaning that the relevant spatial information could be provided by a subset of the edges, in addition to excess information potentially making training more challenging. Even though this study did not focus on discussing better connectivity strategies for wind forecasting or using learnable adjacency matrices, we conduct a brief investigation into whether some of the connections could potentially be removed. Fig. 10 plots the different MAEs on the test data, as we increase the number of connections in the input graphs. 'n_closest' refers to the number of closest neighbours to which we allow a node to connect. Looking at Fig. 1, this meant that if 'n_closest' was set to three, Valhall A would only have edges sending to it from Sleipner B, Granefeltet and Oseberg Sør. The first row in Fig. 10, shows the MAEs for different 'n_closest' values, while the second row contains the same information, but normalised by MAE_0, which was the MAE when 'n_closest' = 0, i.e. the MAE for predictions made without any spatial information. While the first row provides information on the relative differences between models, the second row visualises the percentage improvements gained from including additional edges for a particular model.

First, MAEs were seen to rapidly decrease as we increased the number of edges, before converging to constant values when more than around five neighbours were considered for the edges. The sharp decrease indicated that the models were able to successfully leverage spatial correlations to improve forecasts, proving the effectiveness of the proposed GNN architectures. Nevertheless, since MAEs converged to constant values for non-complete graphs, it indicated that a number of connections could potentially be removed without impairing predictive performance. Further work would therefore be desirable to investigate better methods for which to construct the graphs or learn optimal connectivity for spatio-temporal wind forecasting.

Looking at the second row of Fig. 10, it was seen that the percentage reduction in MAEs was greater for the longer forecasting horizons. For the immediate short-term (i.e. prediction length of 1), spatial correlations were thought less important than for the 1- and 4-h forecasts, due to the large physical distances between nodes resulting in wind fields not having time to propagate in the immediate short-term. The added benefit of having long-range connections between nodes far apart, was therefore greater for the 6- and 24-step settings, than for the 1-step ahead forecasts. Comparing the 6- and 24-step forecasts, the latter also converged slightly later, which was likely due to the longer-term forecasts taking advantage of information from nodes further away from the target. The percentage change in MAEs was also greater for the 24-step setting than for the 6-step, even though the difference was not as big as between the 1- and 6-step settings, which indicated that the 24-step forecasts might have benefited from even larger geographical information than was available. Overall, it was concluded that all models were able to take advantage of spatial information in making forecasts, with the Transformer-based architectures generally showing slightly larger improvements than the ST-MLP model.

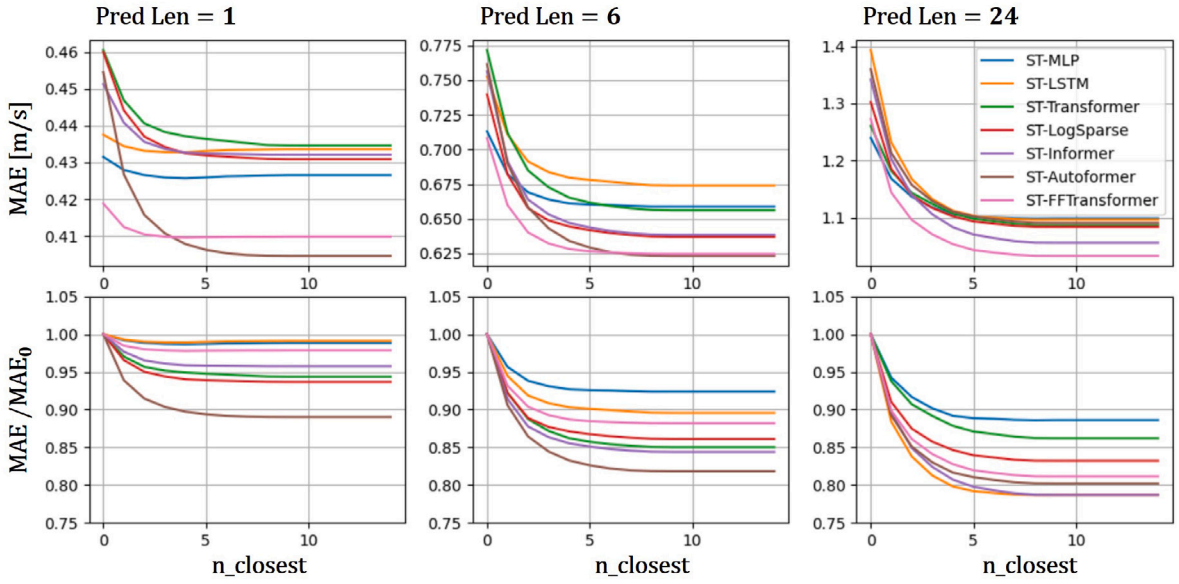


Fig. 10. Test MAEs for different degrees of graph connectivity, with 'n_closest' referring to the maximum number of edges sending to a particular node. In the bottom row are the MAEs normalised by the MAEs for 'n_closest' = 0.

6. Conclusion

In recent years, Transformer-based models have presided over sequence-based deep learning, often superseding recurrent or convolutional models. Nevertheless, research employing these architectures for wind forecasting has been scarce. This study considered different Transformer architectures as the main predictor for spatio-temporal multi-step forecasting, focusing on the LogSparse Transformer, Informer and Autoformer. This is the first time many of these have been applied to wind forecasting and placed in a spatio-temporal setting using GNNs. Additionally, the novel FFTransformer was proposed, which is based on signal decomposition using wavelet transform and an adapted attention mechanism in the frequency domain. Results show that the FFTransformer architecture was very competitive, achieving results on par with the Autoformer-based model for the 1- and 6-step forecasts, while significantly outperforming all other models for the longer 24-step forecasts. Even though the vanilla Transformer architecture generally did not yield significant improvements over an MLP model, it was seen that the convolutional attention in the LogSparse Transformer and the *ProbSparse Attention* of the Informer, were able to slightly improve prediction performance. By estimating the associated prediction errors in kW and kWh, we showed the potential physical effects of different forecasting performances with regards to the power grid, with the FFTransformer model showing an additional 5% improvement over all other models for the 4-h forecasts. Nevertheless, obtaining the powers based on the NREL 5 MW reference turbine, the method was fairly trivial and it would be desirable to further test the different models on real wind power datasets. By removing graph connections in the input data, we showed that the proposed GNN architectures were successful in leveraging spatial correlations to improve local forecasts. However, it was also seen that some connections in the input data might be redundant, calling for additional research into more efficient approaches for graph connectivity in the context of wind forecasting. Since the FFTransformer model is not restricted to a particular signal decomposition technique or attention mechanism, slight alterations from the particular set-up used in this study might also be relevant and could be easily implemented and tested to facilitate different applications. We therefore hope that this study will spark further research into modifications and other applications of the FFTransformer, as well as investigation into the applicability of different Transformer-based architectures for use in wind forecasting.

CRediT authorship contribution statement

Lars Ødegaard Bentsen: Project administration, Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Validation, Visualization, Writing – original draft. **Narada Dilp Warakagoda:** Supervision, Writing – review & editing. **Roy Stenbro:** Supervision, Writing – review & editing. **Paal Engelstad:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Lars Odegaard Bentsen reports financial support was provided by Research Council of Norway.

Data availability

The author does not have permission to distribute the data, but anyone can obtain access and download the data using the Frost API: https://frost.met.no/index.html.

Acknowledgements

This work was in part financed by the research project ELOGOW (Electrification of Oil and Gas Installation by offshore Wind), Norway. ELOGOW (project nr: 308838) is funded by the Research Council of Norway under the PETROMAKS2 program with financial support of Equinor Energy AS, ConocoPhillips Skandinavia AS and Aibel AS. The authors would also like to thank The Norwegian Meteorological institute for enabling the open access data used for this study.

References

[1] Okumus Inci, Dinler Ali. Current status of wind energy forecasting and a hybrid method for hourly predictions. Energy Convers Manage 2016;123:362–71.
[2] Van Kuik GAM, Peinke Joachim, Nijssen Rogier, Lekou Denja, Mann Jakob, Sørensen Jens Nørkær, Ferreira Carlos, van Wingerden Jan-Willem, Schlipf David, Gebraad Pieter, et al. Long-term research challenges in wind energy–a research agenda by the European academy of wind energy. Wind Energy Sci 2016;1(1):1–39.
[3] Chang Wen-Yeau, et al. A literature review of wind forecasting methods. J Power Energy Eng 2014;2(04):161.
[4] Kavasseri Rajesh G, Seetharaman Krithika. Day-ahead wind speed forecasting using f-ARIMA models. Renew Energy 2009;34(5):1388–93.
[5] Singh SN, Mohapatra Abheejeet, et al. Repeated wavelet transform based ARIMA model for very short-term wind speed forecasting. Renew Energy 2019;136:758–68.
[6] da Silva Ramon Gomes, Ribeiro Matheus Henrique Dal Molin, Moreno Sinvaldo Rodrigues, Mariani Viviana Cocco, dos Santos Coelho Leandro. A novel decomposition-ensemble learning framework for multi-step ahead wind energy forecasting. Energy 2021;216:119174.
[7] Wu Binrong, Wang Lin, Zeng Yu-Rong. Interpretable wind speed prediction with multivariate time series and temporal fusion transformers. Energy 2022;252:123990.
[8] Meng Anbo, Ge Jiafei, Yin Hao, Chen Sizhe. Wind speed forecasting based on wavelet packet decomposition and artificial neural networks trained by crisscross optimization algorithm. Energy Convers Manage 2016;114:75–88.
[9] Liu Hui, Tian Hong-qi, Pan Di-fu, Li Yan-fei. Forecasting models for wind speed using wavelet, wavelet packet, time series and artificial neural networks. Appl Energy 2013;107:191–208.
[10] Jørgensen Kathrine Lau, Shaker Hamid Reza. Wind power forecasting using machine learning: State of the art, trends and challenges. In: 2020 IEEE 8th international conference on smart energy grid engineering. SEGE, IEEE; 2020, p. 44–50.
[11] Zendehboudi Alireza, Baseer M Abdul, Saidur R. Application of support vector machine models for forecasting solar and wind energy resources: A review. J Clean Prod 2018;199:272–85.
[12] Colak Ilhami, Sagiroglu Seref, Yesilbudak Mehmet. Data mining and wind power prediction: A literature review. Renew Energy 2012;46:241–7.
[13] Sfetsos A. A novel approach for the forecasting of mean hourly wind speed time series. Renew Energy 2002;27(2):163–74.
[14] More Anurag, Deo MC. Forecasting wind with neural networks. Mar Struct 2003;16(1):35–49.
[15] Shi Jing, Guo Jinmei, Zheng Songtao. Evaluation of hybrid forecasting approaches for wind speed and power generation time series. Renew Sustain Energy Rev 2012;16(5):3471–80.
[16] Guo Zhen-hai, Wu Jie, Lu Hai-yan, Wang Jian-zhou. A case study on a hybrid wind speed forecasting method using BP neural network. Knowl-Based Syst 2011;24(7):1048–56.
[17] Alkhayat Ghadah, Mehmood Rashid. A review and taxonomy of wind and solar energy forecasting methods based on deep learning. Energy AI 2021;4:100060.
[18] Schmidhuber Jürgen, Hochreiter Sepp, et al. Long short-term memory. Neural Comput 1997;9(8):1735–80.
[19] Li Yanfei, Wu Haiping, Liu Hui. Multi-step wind speed forecasting using EWT decomposition, LSTM principal computing, RELM subordinate computing and IEWT reconstruction. Energy Convers Manage 2018;167:203–19.
[20] Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural machine translation by jointly learning to align and translate. 2014, arXiv preprint arXiv:1409.0473.
[21] Li Pengbo, Wang Xiangwen, Yang Junjie. Short-term wind power forecasting based on two-stage attention mechanism. IET Renew Power Gener 2020;14(2):297–304.
[22] Huang Bin, Liang Yuying, Qiu Xiaolin. Wind power forecasting using attention-based recurrent neural networks: a comparative study. IEEE Access 2021;9:40432–44.

[23] Oord Aaron van den, Dieleman Sander, Zen Heiga, Simonyan Karen, Vinyals Oriol, Graves Alex, Kalchbrenner Nal, Senior Andrew, Kavukcuoglu Koray. Wavenet: A generative model for raw audio. 2016, arXiv preprint arXiv:1609.03499.

[24] Dong Xiaochong, Sun Yingyun, Li Ye, Wang Xinying, Pu Tianjiao. Spatio-temporal convolutional network based power forecasting of multiple wind farms. J Mod Power Syst Clean Energy 2021;10(2):388–98.

[25] Shivam Kumar, Tzou Jong-Chyuan, Wu Shang-Chen. Multi-step short-term wind speed prediction using a residual dilated causal convolutional network with nonlinear attention. Energies 2020;13(7):1772.

[26] Wang HZ, Wang GB, Li GQ, Peng JC, Liu YT. Deep belief network based deterministic and probabilistic wind speed forecasting approach. Appl Energy 2016;182:80–93.

[27] Niu Zhewen, Yu Zeyuan, Tang Wenhu, Wu Qinghua, Reformat Marek. Wind power forecasting using attention-based gated recurrent unit network. Energy 2020;196:117081.

[28] Putz Dominik, Gumhalter Michael, Auer Hans. A novel approach to multi-horizon wind power forecasting based on deep neural architecture. Renew Energy 2021;178:494–505.

[29] Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Łukasz, Polosukhin Illia. Attention is all you need. Adv Neural Inf Process Syst 2017;30.

[30] Beltagy Iz, Peters Matthew E, Cohan Arman. Longformer: The long-document transformer. 2020, arXiv preprint arXiv:2004.05150.

[31] Zhou Tian, Ma Ziqing, Wen Qingsong, Wang Xue, Sun Liang, Jin Rong. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. 2022, arXiv preprint arXiv:2201.12740.

[32] Lim Bryan, Arık Sercan Ö, Loeff Nicolas, Pfister Tomas. Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int J Forecast 2021;37(4):1748–64.

[33] Li Shiyang, Jin Xiaoyong, Xuan Yao, Zhou Xiyou, Chen Wenhu, Wang Yu-Xiang, Yan Xifeng. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Adv Neural Inf Process Syst 2019;32.

[34] Zhou Haoyi, Zhang Shanghang, Peng Jieqi, Zhang Shuai, Li Jianxin, Xiong Hui, Zhang Wancai. Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 35. 2021, p. 11106–15.

[35] Kitaev Nikita, Kaiser Łukasz, Levskaya Anselm. Reformer: The efficient transformer. 2020, arXiv preprint arXiv:2001.04451.

[36] Wu Haixu, Xu Jiehui, Wang Jianmin, Long Mingsheng. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Adv Neural Inf Process Syst 2021;34:22419–30.

[37] Wang Lei, He Yigang, Li Lie, Liu Xiaoyan, Zhao Yingying. A novel approach to ultra-short-term multi-step wind power predictions based on encoder–decoder architecture in natural language processing. J Clean Prod 2022;354:131723.

[38] Qu Kai, Si Gangquan, Shan Zihan, Kong XiangGuang, Yang Xin. Short-term forecasting for multiple wind farms based on transformer model. Energy Rep 2022;8:483–90.

[39] Liu Zhifeng, Ding Feng, Wan Fujing. Wind speed forecasting method based on deep learning strategy using long short term memory neural network and transformer model. In: 2021 IEEE 23rd int conf on high performance computing & communications; 7th int conf on data science & systems; 19th int conf on smart city; 7th int conf on dependability in sensor, cloud & big data systems & application (HPCC/DSS/SmartCity/DependSys). IEEE; 2021, p. 2288–93.

[40] Wang Hai-Kun, Song Ke, Cheng Yi. A hybrid forecasting model based on CNN and informer for short-term wind power. Front Energy Res 2022;1041.

[41] Pan Xiaoxin, Wang Long, Wang Zhongju, Huang Chao. Short-term wind speed forecasting based on spatial-temporal graph transformer networks. Energy 2022;253:124095.

[42] Hu Tianyu, Wu Wenchuan, Guo Qinglai, Sun Hongbin, Shi Libao, Shen Xinwei. Very short-term spatial and temporal wind power forecasting: A deep learning approach. CSEE J Power Energy Syst 2019;6(2):434–43.

[43] Zhu Qiaomu, Chen Jinfu, Shi Dongyuan, Zhu Lin, Bai Xiang, Duan Xianzhong, Liu Yilu. Learning temporal and spatial correlations jointly: A unified framework for wind speed prediction. IEEE Trans Sustain Energy 2019;11(1):509–23.

[44] Wang Yun, Zou Runmin, Liu Fang, Zhang Lingjun, Liu Qianyi. A review of wind speed and wind power forecasting with deep neural networks. Appl Energy 2021;304:117766.

[45] Khodayar Mahdi, Wang Jianhui. Spatio-temporal graph deep neural network for short-term wind speed forecasting. IEEE Trans Sustain Energy 2018;10(2):670–81.

[46] Stańczyk Tomasz, Mehrkanoon Siamak. Deep graph convolutional networks for wind speed prediction. 2021, arXiv preprint arXiv:2101.10041.

[47] Wang Fei, Chen Peng, Zhen Zhao, Yin Rui, Cao Chunmei, Zhang Yagang, Duić Neven. Dynamic spatio-temporal correlation and hierarchical directed graph structure based ultra-short-term wind farm cluster power forecasting method. Appl Energy 2022;323:119579.

[48] Wang Lei, He Yigang. M2STAN: Multi-modal multi-task spatiotemporal attention network for multi-location ultra-short-term wind power multi-step predictions. Appl Energy 2022;324:119672.

[49] Goodfellow Ian, Bengio Yoshua, Courville Aaron, Bengio Yoshua. Deep learning, Vol. 1. Cambridge: MIT Press; 2016.

[50] Battaglia Peter W, Hamrick Jessica B, Bapst Victor, Sanchez-Gonzalez Alvaro, Zambaldi Vinicius, Malinowski Mateusz, Tacchetti Andrea, Raposo David, Santoro Adam, Faulkner Ryan, et al. Relational inductive biases, deep learning, and graph networks. 2018, arXiv preprint arXiv:1806.01261.

[51] Catalão JP da S, Pousinho Hugo Miguel Inácio, Mendes Vıctor Manuel Fernandes. Short-term wind power forecasting in Portugal by neural networks and wavelet transform. Renew Energy 2011;36(4):1245–51.

[52] Díaz H, Soares C Guedes. Review of the current status, technology and future trends of offshore wind farms. Ocean Eng 2020;209:107381.

[53] Akiba Takuya, Sano Shotaro, Yanase Toshihiko, Ohta Takeru, Koyama Masanori. Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019, p. 2623–31.

[54] Jonkman Jason, Butterfield Sandy, Musial Walter, Scott George. Definition of a 5-MW reference wind turbine for offshore system development. Technical report, Golden, CO (United States): National Renewable Energy Lab.(NREL); 2009.