



# Deep Learning Model for Wind Forecasting: Classification Analyses for Temporal Meteorological Data

Shubhi Harbola<sup>1,2</sup>  · Volker Coors<sup>2</sup> 

Received: 4 August 2020 / Accepted: 1 November 2021 / Published online: 19 December 2021  
© The Author(s) 2021

## Abstract

This paper proposes a multiple CNN architecture with multiple input features, combined with multiple LSTM, along with densely connected convolutional layers, for temporal wind nature analyses. The designed architecture is called Multiple features, Multiple Densely Connected Convolutional Neural Network with Multiple LSTM Architecture, i.e. MCLT. A total of 58 features in the input layers of the MCLT are designed using wind speed and direction values. These empirical features are based on percentage difference, standard deviation, correlation coefficient, eigenvalues, and entropy, for efficiently describing the wind trend. Two successive LSTM layers are used after four densely connected convolutional layers of the MCLT. Moreover, LSTM has memory units that utilise learnt features from the current as well as previous outputs of the neurons, thereby enhancing the learning of patterns in the temporal wind dataset. Densely connected convolutional layer helps to learn features of other convolutional layers as well. The MCLT is used to predict dominant speed and direction classes in the future for the wind datasets of Stuttgart and Netherlands. The maximum and minimum overall accuracies for dominant speed prediction are 99.1% and 94.9%, (for Stuttgart) and 99.9% and 97.5% (for Netherlands) and for dominant direction prediction are 99.9% and 94.4% (for Stuttgart) and 99.6% and 96.4% (for Netherlands), respectively, using MCLT with 58 features. The MCLT, therefore, with multiple features at different levels, i.e. the input layers, the convolutional layers, and LSTM layers, shows promising results for the prediction of dominant speed and direction. Thus, this work is useful for proper wind utilisation and improving environmental planning. These analyses would also help in performing Computational Fluid Dynamics (CFD) simulations using wind speed and direction measured at a nearby meteorological station, for devising a new set of appropriate inflow boundary conditions.

**Keywords** Wind flow · Prediction · Deep learning architectures · ResNet · Machine learning · Meteorological data · Energy · LSTM · Convolutional neural network · Wind forecasting

## Zusammenfassung

*Deep-Learning-Modell für Windvorhersagen: Klassifikationsanalysen für temporäre meteorologische Daten.* In diesem Beitrag wird eine Architektur von mehreren dicht verbundenen CNN in Kombination mit zwei nachgelagerten Long Short-Term Memory Verfahren zur Analyse von Windmessungen vorgeschlagen. Diese Architektur wird als “Multiple Features, Multiple Densely Connected Convolutional Neural Network with Multiple LSTM” (MCLT) bezeichnet. Insgesamt werden 58 Merkmale in den Eingabeschichten des MCLT unter Verwendung von Windgeschwindigkeits- und -richtungswerten verwendet. Diese empirischen Merkmale basieren auf der prozentualen Differenz, der Standardabweichung, dem Korrelationskoeffizienten, den Eigenwerten und der Entropie. Den CNN werden zwei aufeinanderfolgende LSTM-Schichten nachgelagert. Diese LSTM-Schichten verwenden die gelernten Merkmale aus den aktuellen und früheren Ausgaben der CNN, wodurch das Lernen von Mustern im zeitlichen Winddatensatz verbessert wird. Die MCLT-Architektur wird zur Vorhersage dominanter Geschwindigkeits- und Richtungsklassen am Beispiel von gemessenen Winddatensätze von Stuttgart

---

✉ Shubhi Harbola  
Shubhi.Harbola@ieee.org

<sup>1</sup> University of Stuttgart, Stuttgart, Germany

<sup>2</sup> University of Applied Science, Stuttgart, Germany

und den Niederlanden verwendet und evaluiert. Die maximale und minimale Gesamtgenauigkeit für die Vorhersage der vorherrschenden Geschwindigkeit beträgt 99,1 % und 94,9 % (für Stuttgart) bzw. 99,9 % und 97,5 % (für die Niederlande) und für die Vorhersage der vorherrschenden Richtung 99,9 % und 94,4 % (für Stuttgart) bzw. 99,6 % und 96,4 % (für die Niederlande). Die vorgeschlagene MCLT-Architektur zeigt vielversprechende Ergebnisse für die Vorhersage der dominanten Windgeschwindigkeit und -richtung. Damit trägt diese Arbeit dazu bei, Wind-Analysen in Umweltplanungen besser berücksichtigen zu können. Die Analysen helfen auch bei der Durchführung von CFD-Simulationen (Computational Fluid Dynamics), bei denen Windgeschwindigkeit und -richtung als Anströmungsrandbedingungen genutzt werden, um beispielsweise das Potential von Kleinwindkraftanlagen im urbanen Raum abzuschätzen.

### List of Symbols

$\mu$	Mean
$\sigma$	Standard deviation
LSTM	Long short-term memory
ADASYN	Adaptive synthetic sampling
ANN	Artificial neural network
M5P	M5 model trees
CFD	Computational fluid dynamics
ELUs	Exponential linear units
SMAPE	Symmetric mean absolute percentage error
LSSVM	Least squares support vector machine
NWP	Numerical weather prediction
CNN	Convolutional neural network
SVM	Support vector machine
ML	Machine learning
TA	Total accuracy
IDM	One-dimensional multiple convolutional neural network architecture
MCLT	Multiple features, multiple densely connected convolutional neural network with multiple LSTM architecture

## 1 Introduction

The green energy requirement is expanding day by day with increasing population growth, and development. One of the free, clean, renewable energy source with a limitless supply that is naturally available is wind (Lawan et al. 2014; Marović et al. 2017; Tarade and Katti 2011). In today's world, mankind seeks to become more environmental friendly in its operations, and the wind is an important source of energy. To monitor, predict, and maintain weather patterns and global climate, wind speed and direction are essential components that need to be tracked (Colak et al. 2012; Vargas et al. 2010). The future wind trends are influenced by the past conditions of wind speed and direction. Moreover, to support the selection of new wind turbine installation sites, prior analysis of the wind nature, and its prediction is required (Aissou et al. 2015; Reed et al. 2011). There are four categories to group wind speed and direction prediction methods based on the time scale (Yesilbudak et al. 2013; Yesilbodak et al. 2017), *viz* (i) very short-term (these predictions cover a few seconds to 30 min ahead),

(ii) short-term (include predictions from 30 min to 6 h), (iii) medium-term (predictions for 6 h to 1 day ahead) and, (iv) long-term (from 1-day to 1-week predictions). Machine Learning (ML) (Sapronova et al. 2016), Numerical Weather Prediction (NWP) models (Aslipour and Yazdizadeh 2019; Janssens et al. 2016; Louka et al. 2008), and models incorporating both NWP and ML (Vladislavleva et al. 2013) for wind prediction are presently the focus of research and commercial applications.

The ML concepts such as fuzzy logic (Martínez-Arellano et al. 2014; Monfared et al. 2009), Artificial Neural Networks (ANN) with several hidden layers (Birenbaum and Greenspan 2017; Daraeepour and Echeverri 2014; El-Fouly et al. 2008; Vogado et al. 2018; Yesilbodak et al. 2017), and statistical models (Jursa and Rohrig 2008; Louka et al. 2008; Miranda and Dunn 2006; Yang and Chen 2019) are used to design such wind prediction frameworks. Techniques like particle swarm optimisation, wavelet transform (Liu et al. 2018; Martínez-Arellano et al. 2014; Wang et al. 2017), REP tree, M5P tree, bagging tree, K-nearest neighbour algorithm (Jursa and Rohrig 2008; Kusiak et al. 2009a; Kusiak and Zhang 2010), principal component analysis, moving average models, Markov chain (Kusiak et al. 2009b; Treiber et al. 2016; Vargas et al. 2010), combined with regression models using neural networks, have been used for wind analyses (Yang and Chen 2019). Moreover forecasting wind speed with Support Vector Machines (SVM) and its variation (Kang et al. 2017) such as Least Square Support Vector Machines (LSSVM) have also been proposed (De Giorgi et al. 2014, 2009; Harbola and Coors 2019a; Yuan et al. 2015). These works used only limited features based on wind speed, direction and power as input. The ML concept of deep learning based on Convolutional Neural Networks (CNNs) has achieved higher accuracy for classification of Two-Dimensional (2D) images and Three-Dimensional (3D) point clouds (Krizhevsky et al. 2012; Long et al. 2015; Szegedy et al. 2015). Convolutional layers in CNN learn a large number of features automatically so that they need not be designed manually (Jung et al. 2019; Kuo 2016; Qi et al. 2016). Variations of CNNs like single CNN, multiple CNN, Residual Neural Network Architecture (ResNet) (He et al. 2016; Huang et al. 2017; Xie et al. 2017) with several convolutional layers have become popular for classification. Further, One-Dimensional (1D) and 2D single CNNs have been

employed for wind power and wind speed predictions (Liu et al. 2018; Wang et al. 2017). However, these models either smooth and filter the wind dataset by applying techniques like wavelet or convert 1D wind dataset into 2D images (Liu et al. 2018; Wang et al. 2017). This leads to distortion of the original information present in temporal wind dataset. To overcome this problem, 1D single CNN (1DS) and 1D multiple CNN (1DM), working directly on the original 1D temporal wind dataset without using smoothing techniques, were proposed by (Harbola and Coors 2019b). The 1DM model showed better performance than the 1DS for prediction of the dominant class of wind speed and direction. However, only two features based on the speed and direction were included in the input layers of the 1DS and 1DM and a limited number of classes (eleven) were used for prediction.

This paper improves upon the 1DM model and proposes a deep multiple CNN architecture with multiple input features, along with multiple Long Short-Term Memory (LSTM) and densely connected convolutional layers. More number of features in CNN architecture help in learning the various properties of a sample from finer to coarser levels (de Andrade 2019). Therefore, a large number of features are used in this study. The new architecture is called Multiple features, Multiple Densely Connected Convolutional Neural Network with Multiple LSTM Architecture, *i.e.* MCLT with the following novel contributions, (a) multiple features (58 in total) are used in the input layers for better representation of the temporal wind dataset, (b) fully connected layers are replaced by LSTM layers to provide memory for a longer period and thereby improving the training of the model, (c) connecting convolutional layers similar to 2D ResNet (for images) (Duta et al. 2020) architecture so that each convolutional layer learns features of previous convolutional layers as well, and (d) a higher number of classes (21) are used for analyzing detailed trend of the temporal wind dataset. The authors are unable to find any existing work that has used these four contributions for in-depth analyses and prediction of wind nature. The remaining paper is arranged as follows; Section 2.1 describes the MCLT architecture followed by Sect. 2.2 which gives detail of the wind datasets used in the experiments. Section 3 presents the results and Sect. 4 gives conclusion and future recommendations.

## 2 Methodology

The proposed MCLT architecture is an advanced deep learning architecture, which is a combination of multiple features, multiple LSTM, and densely connected convolutional layers in a multiple CNN model for the wind nature analysis. A total of 58 features are based on the various combinations of two important temporal wind properties, *i.e.* wind speed and direction. This ensures that several details of the wind

features are learnt by the MCLT. These features are designed based on time series data from the past. The features form the input of the MCLT that has to predict a representative wind speed or direction value for a period of time immediately after the last value of the input sample in the time series. The following sections discuss the design of these multiple features, along with the MCLT framework.

Further, the input to the MCLT is a time series (or temporal) data of wind speed and direction for a certain geographic location (*i.e.* spatial location). These time series data need to be acquired at regular intervals. The time stamp in the data helps to arrange the data in the increasing order of time. More details of the data are available in Sect. 2.2. Further, several features are designed using the wind speed and direction that are explained in Sect. 2.1. The prediction of the MCLT is the class label based on the dominant wind speed and direction. The multiple wind speed values for future points in time are grouped into 21 classes using the wind speed values. Amongst these classes, the class having maximum count, *i.e.* class of the speed values that occur most (*viz.* dominant speed amongst future points) in time forms the class label of the input sample (Harbola and Coors 2019b). Similarly, 21 classes for the wind direction are designed and the class label is assigned to the sample based on the class having maximum count of the wind direction values. It may be noted that grouping into 21 classes is a process of creating the class labels of training and testing samples, while the MCLT prediction represents one class label (for a given sample) that depicts wind speed or direction value for a certain period of time immediately following the time represented by the input sample. Also, there are two trained MCLT models, one for the wind speed and another for the wind direction. The proposed method can be short term, medium term as well as long term depending on the choice of the number of future points in time that are grouped into 21 classes. This concept is discussed in detail in Sect. 2.1.

### 2.1 Designing Multiple Features

Wind speed (given in m/s) and the direction (in radians) are two input features (Harbola and Coors 2019b) to the proposed architecture. Besides these two features, 56 additional features also form part of the input. Suppose, matrix  $M_{i,j}$  has  $r$  rows and 58 columns, where  $r$  equals to the number of temporal wind values present in the dataset (each row of  $M_{i,j}$  is a time instance for wind dataset comprising speed and direction values), and  $i, j$  denote row and column number of a cell, respectively, in the matrix. Moreover, each column denotes a feature. The first feature (first column), second feature (second column) comprise the wind speed and direction values, respectively.  $M_{i,j=3}$  (third feature) is the percentage difference (*per*) between  $M_{i,j=1}$  (speed values)

Various designed features	
<p><b>Based on wind speed</b></p> <p><math>M i, j = 1, \text{Wind speed at } M i, j = 1</math></p> <p><b>Based on percentage difference (per) of wind speed</b></p> <p><math>M i, j = 3, \text{per}(M i, j = 1, M i - 1, j = 1)</math>  <math>M i, j = 4, \text{per}(M i, j = 1, M i - 2, j = 1)</math>  <math>M i, j = 5, \text{per}(M i, j = 1, M i - 3, j = 1)</math>  <math>M i, j = 6, \text{per}(M i, j = 1, M i - 4, j = 1)</math>  <math>M i, j = 7, \text{per}(M i, j = 1, M i - 5, j = 1)</math>  <math>M i, j = 8, \text{per}(M i, j = 1, M i - 6, j = 1)</math>  <math>M i, j = 9, \text{per}(M i, j = 1, M i - 7, j = 1)</math></p>	<p><b>Based on wind direction</b></p> <p><math>M i, j = 2, \text{Wind direction at } M i, j = 2</math></p> <p><b>Based on percentage difference (per) of wind direction</b></p> <p><math>M i, j = 10, \text{per}(M i, j = 2, M i - 1, j = 2)</math>  <math>M i, j = 11, \text{per}(M i, j = 2, M i - 2, j = 2)</math>  <math>M i, j = 12, \text{per}(M i, j = 2, M i - 3, j = 2)</math>  <math>M i, j = 13, \text{per}(M i, j = 2, M i - 4, j = 2)</math>  <math>M i, j = 14, \text{per}(M i, j = 2, M i - 5, j = 2)</math>  <math>M i, j = 15, \text{per}(M i, j = 2, M i - 6, j = 2)</math>  <math>M i, j = 16, \text{per}(M i, j = 2, M i - 7, j = 2)</math></p>
<p><b>Based on the standard deviation (std) of wind speed</b></p> <p><math>M i, j = 17, \text{std}(M i, j = 1, M i - 1, j = 1)</math>  <math>M i, j = 18, \text{std}(M i, j = 1, M i - 1, j = 1, M i - 2, j = 1)</math>  <math>M i, j = 19, \text{per}(M i, j = 1, M i - 1, j = 1, M i - 2, j = 1, M i - 3, j = 1)</math>  <math>M i, j = 20, \text{per}(M i, j = 1, M i - 1, j = 1, M i - 2, j = 1, M i - 3, j = 1, M i - 4, j = 1)</math>  <math>M i, j = 21, \text{per}(M i, j = 1, M i - 1, j = 1, M i - 2, j = 1, M i - 3, j = 1, M i - 4, j = 1, M i - 5, j = 1)</math>  <math>M i, j = 22, \text{per}(M i, j = 1, M i - 1, j = 1, M i - 2, j = 1, M i - 3, j = 1, M i - 4, j = 1, M i - 5, j = 1, M i - 6, j = 1)</math>  <math>M i, j = 23, \text{per}(M i, j = 1, M i - 1, j = 1, M i - 2, j = 1, M i - 3, j = 1, M i - 4, j = 1, M i - 5, j = 1, M i - 6, j = 1, M i - 7, j = 1)</math></p>	
<p><b>Based on the standard deviation (std) of wind direction</b></p> <p><math>M i, j = 24, \text{std}(M i, j = 2, M i - 1, j = 2)</math>  <math>M i, j = 25, \text{std}(M i, j = 2, M i - 1, j = 2, M i - 2, j = 2)</math>  <math>M i, j = 26, \text{per}(M i, j = 2, M i - 1, j = 2, M i - 2, j = 2, M i - 3, j = 2)</math>  <math>M i, j = 27, \text{per}(M i, j = 2, M i - 1, j = 2, M i - 2, j = 2, M i - 3, j = 2, M i - 4, j = 2)</math>  <math>M i, j = 28, \text{per}(M i, j = 2, M i - 1, j = 2, M i - 2, j = 2, M i - 3, j = 2, M i - 4, j = 2, M i - 5, j = 2)</math>  <math>M i, j = 29, \text{per}(M i, j = 2, M i - 1, j = 2, M i - 2, j = 2, M i - 3, j = 2, M i - 4, j = 2, M i - 5, j = 2, M i - 6, j = 2)</math>  <math>M i, j = 30, \text{per}(M i, j = 2, M i - 1, j = 2, M i - 2, j = 2, M i - 3, j = 2, M i - 4, j = 2, M i - 5, j = 2, M i - 6, j = 2, M i - 7, j = 2)</math></p>	
<p><b>Based on the correlation coefficient (corcoef) of wind speed and direction</b></p> <p><math>M i, j = 31, \text{corcoef}(M i, j = 1, 2, M i - 1, j = 1, 2)</math>  <math>M i, j = 32, \text{corcoef}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2)</math>  <math>M i, j = 33, \text{corcoef}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2)</math>  <math>M i, j = 34, \text{corcoef}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2)</math>  <math>M i, j = 35, \text{corcoef}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2)</math>  <math>M i, j = 36, \text{corcoef}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2, M i - 6, j = 1, 2)</math>  <math>M i, j = 37, \text{corcoef}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2, M i - 6, j = 1, 2, M i - 7, j = 1, 2)</math></p>	
<p><b>Based on the highest eigenvalue (eig1) of variance-covariance matrix formed using both speed and direction</b></p> <p><math>M i, j = 38, \text{eig1}(M i, j = 1, 2, M i - 1, j = 1, 2)</math>  <math>M i, j = 39, \text{eig1}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2)</math>  <math>M i, j = 40, \text{eig1}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2)</math>  <math>M i, j = 41, \text{eig1}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2)</math>  <math>M i, j = 42, \text{eig1}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2)</math>  <math>M i, j = 43, \text{eig1}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2, M i - 6, j = 1, 2)</math>  <math>M i, j = 44, \text{eig1}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2, M i - 6, j = 1, 2, M i - 7, j = 1, 2)</math></p>	
<p><b>Based on the lowest eigenvalue (eig2) of variance-covariance matrix formed using both speed and direction</b></p> <p><math>M i, j = 45, \text{eig2}(M i, j = 1, 2, M i - 1, j = 1, 2)</math>  <math>M i, j = 46, \text{eig2}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2)</math>  <math>M i, j = 47, \text{eig2}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2)</math>  <math>M i, j = 48, \text{eig2}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2)</math>  <math>M i, j = 49, \text{eig2}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2)</math>  <math>M i, j = 50, \text{eig2}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2, M i - 6, j = 1, 2)</math>  <math>M i, j = 51, \text{eig2}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2, M i - 6, j = 1, 2, M i - 7, j = 1, 2)</math></p>	
<p><b>Based on the entropy values (entr) calculated using eigenvalues of variance-covariance matrix formed using both speed and direction</b></p> <p><math>M i, j = 52, \text{entr}(M i, j = 1, 2, M i - 1, j = 1, 2)</math>  <math>M i, j = 53, \text{entr}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2)</math>  <math>M i, j = 54, \text{entr}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2)</math>  <math>M i, j = 55, \text{entr}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2)</math>  <math>M i, j = 56, \text{entr}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2)</math>  <math>M i, j = 57, \text{entr}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2, M i - 6, j = 1, 2)</math>  <math>M i, j = 58, \text{entr}(M i, j = 1, 2, M i - 1, j = 1, 2, M i - 2, j = 1, 2, M i - 3, j = 1, 2, M i - 4, j = 1, 2, M i - 5, j = 1, 2, M i - 6, j = 1, 2, M i - 7, j = 1, 2)</math></p>	

Fig. 1 Various designed features in MCLT

**Table 1** Classes formed using the mean and standard deviation of the wind data

Class	Lower range	Upper range
1	$\mu - k_1\sigma$	$\mu + k_1\sigma$
2	$\mu + k_1\sigma$	$\mu + k_2\sigma$
3	$\mu + k_2\sigma$	$\mu + k_3\sigma$
4	$\mu + k_3\sigma$	$\mu + k_4\sigma$
5	$\mu + k_4\sigma$	$\mu + k_5\sigma$
6	$\mu + k_5\sigma$	$\mu + k_6\sigma$
7	$\mu + k_6\sigma$	$\mu + k_7\sigma$
8	$\mu + k_7\sigma$	$\mu + k_8\sigma$
9	$\mu + k_8\sigma$	$\mu + k_9\sigma$
10	$\mu + k_9\sigma$	$\mu + k_{10}\sigma$
11	$\mu + k_{10}\sigma$	$+\infty$
12	$\mu - k_2\sigma$	$\mu - k_1\sigma$
13	$\mu - k_3\sigma$	$\mu - k_2\sigma$
14	$\mu - k_4\sigma$	$\mu - k_3\sigma$
15	$\mu - k_5\sigma$	$\mu - k_4\sigma$
16	$\mu - k_6\sigma$	$\mu - k_5\sigma$
17	$\mu - k_7\sigma$	$\mu - k_6\sigma$
18	$\mu - k_8\sigma$	$\mu - k_7\sigma$
19	$\mu - k_9\sigma$	$\mu - k_8\sigma$
20	$\mu - k_{10}\sigma$	$\mu - k_9\sigma$
21	$-\infty$	$\mu - k_{10}\sigma$

and  $M_{i-1, j=1}$ .  $M_{i, j=4}$  (fourth feature) is the percentage difference between  $M_{i, j=1}$  and  $M_{i-2, j=1}$ . Similarly, the features from  $M_{i, j=5}$  to  $M_{i, j=58}$  are based on the percentage difference (*per*), standard deviation (*std*), correlation coefficient (*cor-coef*), eigenvalues (Weinmann et al. 2015) (*eig1*, *eig2*) and entropy (*entr*) of wind speed and direction. These are shown in detail in Fig. 1, where values up to  $M_{i-7, j}$  are used only due to hardware constraints in the present study, it could be decreased or increased as per available hardware. Thus, each row of  $M_{i, j}$  has column (or feature) values that are dependent on the current and previous rows, *i.e.*  $i$  to  $i - 7$ . In Fig. 1, for example *std* ( $M_{i, j=2}$ ,  $M_{i-1, j=2}$ ,  $M_{i-2, j=2}$ ) means standard deviation of three quantities inside the brackets. The explanation of other features in Fig. 1 is similar. These features are calculated using adjacent temporal values of wind speed and direction and help in describing trends like increase, decrease, stationary, deviation from the mean. The features can be varied depending on the available hardware for training the MCLT. This is discussed in more detail in section 4.3.2. The above constructed  $M_{i, j}$  matrix is further rescaled by dividing each cell’s value by the maximum value amongst all the cells. This rescaling helps in resizing values to a smaller range for better learning of the MCLT. This rescaled  $M_{i, j}$  matrix is used in below concepts.

Samples for training and testing the proposed architecture are designed using  $M_{i, j}$ . A sample consists of input values and a corresponding class label. This class label is predicted by the MCLT. The sample’s input consists of a matrix of dimension  $K_B * 58$  using values from  $M_{i, j=1..58}$  to  $M_{i+K_B, j=1..58}$ , where  $K_B$  is a scalar quantity that depends on the user. Therefore, rows from  $i$  to  $i + K_B$  (and all columns of these rows) of  $M_{i, j}$  form the input of the sample. The columns of matrix  $K_B * 58$  are treated as separate features, each of one dimension in the input layers of the MCLT as discussed in the next section.

The corresponding class label of the sample is a class reflecting the wind speed or the wind direction value for the future  $K_F$  (a scalar value) time values immediately after the last time value ( $i + K_B$ ) in the sample’s input. The class label of the sample is designed using values of speed from  $M_{i+K_B+1, j=1}$  to  $M_{i+K_B+K_F, j=1}$ , where  $K_F$  is a scalar. Here, first column of  $M_{i, j}$  is used that is based on the wind speed values.

For this, mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of the given historical temporal wind dataset are calculated, separately for speed and direction. Then, 21 classes are designed using ( $\mu$ ) and ( $\sigma$ ), of wind speed values as shown in Table 1. The  $\mu$  and  $\sigma$  concepts provide statistical segregation of classes (Ghilani 2010).  $k_i$ , where  $i \rightarrow 1-10$  as shown in Table 1, is decided empirically. Speed values from  $M_{i+K_B+1, j=1}$  to  $M_{i+K_B+K_F, j=1}$  (these speed values without rescaling are used for classes construction) are grouped into these 21 classes, and count of values in each class is found. The class having maximum count is assigned to the class label of the sample. This maximum count represents the dominant speed amongst  $K_F$  future points in time (*i.e.* class of the speed values that occur most) (Harbola and Coors 2019b). Likewise, the class label of the sample based on the direction is determined by finding the maximum count of direction values from  $M_{i+K_B+1, j=2}$  to  $M_{i+K_B+K_F, j=2}$  (these direction values without rescaling are used for classes construction) among these 21 classes. The ( $\mu$ ) and ( $\sigma$ ) based on the wind direction values are used for designing these 21 classes of the wind direction. Here, the second column of  $M_{i, j}$  is used that is based on the wind direction values. As stated earlier, the grouping into 21 classes is a method of creating the class labels of training and testing samples, while the MCLT prediction represents one class label (for a given sample) that depicts wind speed or direction value for  $K_F$  time period immediately after the last time value ( $i + K_B$ ) in the sample’s input. Based on the definition of a sample, from a dataset consisting of matrix  $M_{i, j}$  with  $r$  rows, training samples can be generated by varying  $i$  from 1 to  $r - K_F$  with an increment of 1. This helps in performing the temporal wind data analysis over wind speed and direction.

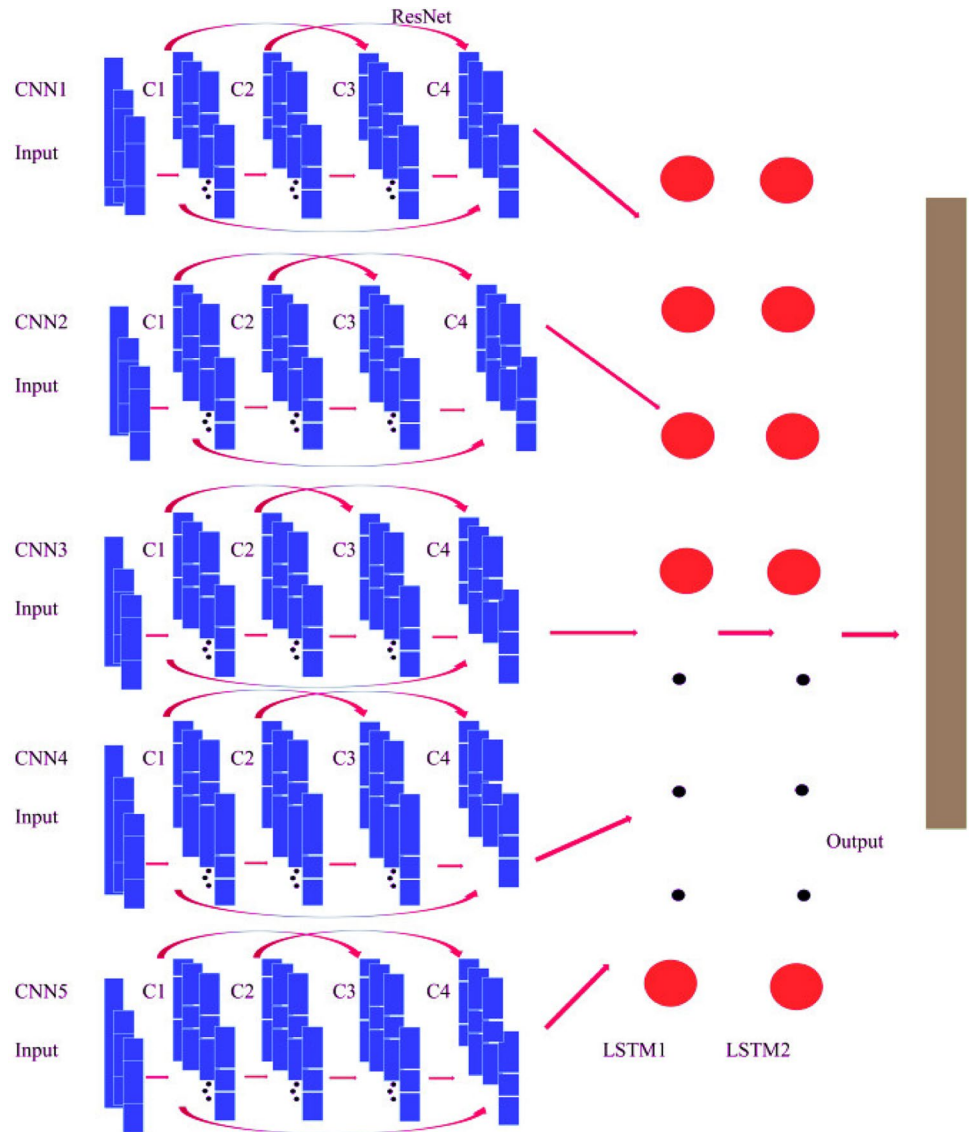
### 2.2 MCLT Architecture

The MCLT architecture is shown in Fig. 2. There are five input layers corresponding to each view  $CNN_i$  ( $CNN_1, CNN_2, CNN_3, CNN_4,$  and  $CNN_5$ ) as in the 1DM. For a given sample's input, five views corresponding to each input layer in the MCLT are formed as follows: (a) first view takes all  $K_B$  values of the sample's input, *i.e.* rows from  $i$  to  $i + K_B$  (and all columns of these rows) of  $M_{i,j}$ , (b) second view takes half of  $K_B$  values of the sample's input from rows  $i$  to  $i + K_B$  at an interval of two (and all columns of these rows) of  $M_{i,j}$ , (c) third view also takes half of  $K_B$  values of the sample's input but from rows  $i + 1$  to  $i + K_B$  at an interval of two (and all columns of these rows) of  $M_{i,j}$ , (d) fourth view takes one-third of  $K_B$  values of the sample's input but from rows  $i$  to  $i + K_B$  at an interval of three (and all columns of these rows) of  $M_{i,j}$ , and (e) fifth view again takes one-third of  $K_B$  values of the sample's input but from rows  $i + 1$  to  $i + K_B$  at

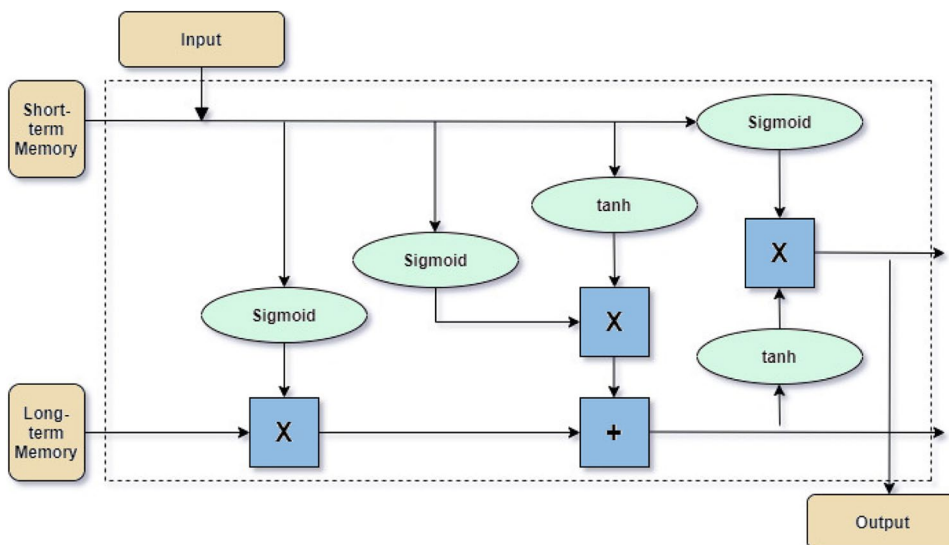
an interval of three (and all columns of these rows) of  $M_{i,j}$ , (Harbola and Coors 2019b). The input layer of each view is followed by four successive convolutional layers ( $C_1, C_2, C_3, C_4$ ). The densely connected convolutional layers similar to ResNet are realised as follows, (a)  $C_3$  directly takes as input features from both  $C_2$  and  $C_1$  (while in the 1DM model,  $C_3$  took input only from previous layer  $C_2$ ), and (b)  $C_4$  directly takes input features from  $C_3, C_2$  and  $C_1$  (while in traditional CNN models,  $C_4$  takes input only from  $C_3$ ) (Zhao et al. 2019).

The detailed pseudo code of MCLT implementation is given in Algorithm 1. All the feature maps from the last convolutional layer  $C_4$  of each view (total 5 views) are first flattened to 1D form (step 13 in Algorithm 1) and then appended one after another (step 14 Algorithm 1). This appended feature vector is then passed to a common LSTM layer called  $LSTM_1$  (step 16 Algorithm 1), which in turn is followed by the second LSTM layer called  $LSTM_2$ . In the

**Fig. 2** MCLT architecture. Red arrows denote connections between different convolutional layers and LSTM layers. All the feature maps from  $C_4$  of  $CNN_1, CNN_2, CNN_3, CNN_4$  and  $CNN_5$  are appended to form a vector and passed into  $LSTM_1$ . Multiple blue boxes in Input,  $C_1, C_2, C_3$  and  $C_4$  represent multiple features in that layer. Red circles in  $LSTM_1$  and  $LSTM_2$  represent neurons



**Fig. 3** LSTM unit. Black arrows denote connections between different intermediates layers, where blue color square box and cyan color ellipse represent point-wise operations and layers, respectively (Chollet 2017)



1DM model, fully connected layers were present in the place of  $LSTM_1$  and  $LSTM_2$ . The output layer (which is dense or fully connected layer) comes after  $LSTM_2$ . The output layer uses softmax function for classification, and the number of neurons in this layer would be the same as the number of classes in the dataset, *i.e.* 21 neurons corresponding to 21 classes (step 18 Algorithm 1).

operation, that takes values such as number of features, stride (amount by which 1D kernel shifts), input from a CNN layer, activation function and dropout (Srivastava et al. 2014) value. *Concatenate* in Algorithm 1 means that  $C_1$  and  $C_2$  (step 9),  $C_1$ ,  $C_2$  and  $C_3$  (step 11), are joined together one after another and then treated as input for the next step *i.e.* making the densely connected convolutional layers. *LSTM*

**Algorithm 1** Architecture Pseudo-code

```

1: procedure MCLT(Input, Output)
2:                                     ▷ Input ← MCLT multiple views
3:                                     ▷ Output ← MCLT output layer
4:   Merged ← [ ]
5:   for i ← 1 to 5 do
6:     CNNiprocessing
7:      $C_1 \leftarrow Conv1D(features, stride, input = CNN_i Input, ELU, dropout)$ 
8:      $C_2 \leftarrow Conv1D(features, stride, input = C_1, ELU, dropout)$ 
9:      $C_{2concat} \leftarrow Concatenate(C_1, C_2)$ 
10:     $C_3 \leftarrow Conv1D(features, stride, input = C_{2concat}, ELU, dropout)$ 
11:     $C_{3concat} \leftarrow Concatenate(C_1, C_2, C_3)$ 
12:     $C_4 \leftarrow Conv1D(features, stride, input = C_{3concat}, ELU, dropout)$ 
13:     $C_4 \leftarrow flatten(C_4)$ 
14:    Merged.append( $C_4$ )
15:   end for
16:    $LSTM_1 \leftarrow LSTM(neurons, input = Merged, dropout)$ 
17:    $LSTM_2 \leftarrow LSTM(neurons, input = LSTM_1, dropout)$ 
18:   Output ← Dense(neurons, input =  $LSTM_2$ , softmax)
19: end procedure
    
```

Further, *Merged* in Algorithm 1, is initially defined as an empty list (step 4) and for each iteration inside for loop, flattened  $C_4$  is appended to it (step 14).  $CNN_i Input$  in step 7 means input corresponding to  $CNN_i$ . *Conv1D* in Algorithm 1 denotes a function representing 1D convolutional

and *Dense* (steps 16–18 in Algorithm 1) denote LSTM and fully connected layers, respectively. LSTM units include a memory element that can maintain information in memory for long periods of time. Figure 3 shows the LSTM architecture in detail as available in (Chollet 2017; Hochreiter

and Schmidhuber 1997). A set of gates (input, output, forget (memory element)) is used to control when information enters LSTM units, when it leaves, and when it is forgotten. Thus, these memory units aid in learning longer-term dependencies. The densely connected convolutional layers help  $C_3$  directly learn features from both  $C_1$  and  $C_2$ , unlike in 1DM, where  $C_3$  learnt features from  $C_2$  only. Likewise,  $C_4$  directly learns features from  $C_1$ ,  $C_2$ , and  $C_3$ , unlike traditional CNN where  $C_4$  considers input only from  $C_3$ .

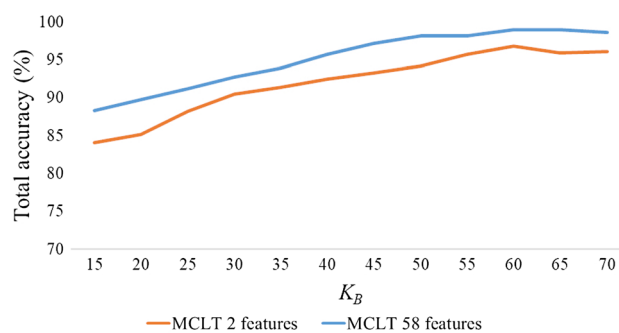
Each input layer of the MCLT, thus, takes multiple 1D features. In this study, there are 58 features in each input layer. A higher number of features in CNN architecture help in learning the various properties of a sample from finer to coarser levels. Therefore many features are used in this study. Thus, for a sample having input values from  $i$  to  $i + K_B$  of  $M_{i,j}$ , each column of these rows form a 1D feature of the input layer. Thus, the MCLT incorporates multiple features and multiple views in the input layers, as well as each convolutional layer takes input from several previous layers, with the presence of memory units in the LSTM layers. The output layer of the MCLT uses the sample's class label, either based on the wind speed or direction, for training and testing the architecture. The sample's class label is designed using  $M_{i+K_B+1}$  to  $M_{i+K_B+K_F}$  values as discussed in the above section. Accordingly, there are two trained MCLT models, one for the wind speed and another for the wind direction. The samples' inputs to these two models remain the same but the the class labels are based on the wind speed (when the model is trained to predict the wind speed) or the wind direction (when the model is trained to predict the wind direction). Further, the parameters determined in training comprise the weights and biases of neurons of convolutional and LSTM layers as well as the output layer.

### 3 Dataset

Historical temporal wind datasets of about more than 30 years are considered as test cases for the proposed MCLT. The first case is the climate and air measuring station located in the corner of Hauptstaetter Strasse 70173 Stuttgart,<sup>1</sup> Germany, which is one of the sources for the wind data collected from 1987 to 2017 in Stuttgart. The temporal resolution of this dataset is thirty minutes as wind speed and direction values are measured at an interval of thirty minutes. The second case is the dataset of Netherlands from the station 210 Valkenburg<sup>2</sup> with 37 years of historical data from 1981 to 2018. The datasets are split into subsets, each of them

<sup>1</sup> <https://www.stadtklima-stuttgart.de>.

<sup>2</sup> [https://www.projects.knmi.nl/klimatologie/onderzoeksgegevens/potentiele\\_wind/](https://www.projects.knmi.nl/klimatologie/onderzoeksgegevens/potentiele_wind/).



**Fig. 4** Total accuracy comparison of 2 and 58 features MCLT, for different values of  $K_B$

corresponding to the data for one month. This allows for an analysis of the data on a monthly basis. One matrix  $M_{i,j}$  (Sect. 2.1) is generated for each of these subsets.

## 4 Experiments and Results

This section explains the results of MCLT for Stuttgart and Netherlands datasets. Section 4.1 provides the details of the hardware and software configuration along with the organisation of the training and testing samples. Section 4.2 presents the obtained accuracies for different datasets and features. Subsection 4.3 represents the qualitative discussion of the obtained results and comparison with other existing methods.

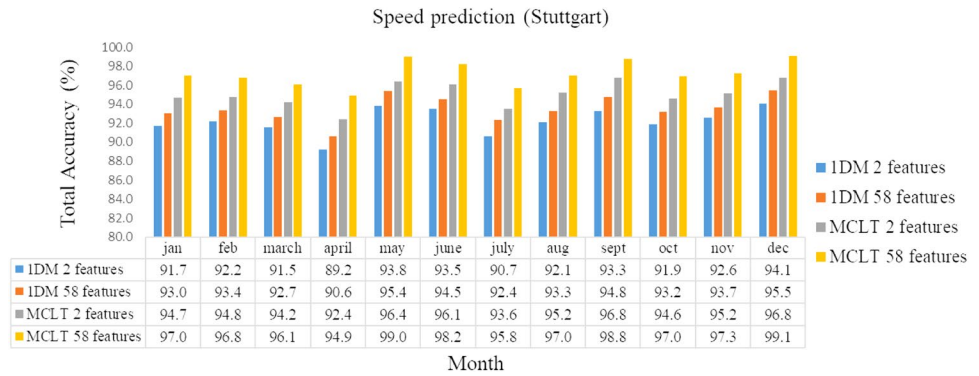
### 4.1 Test Setup

The proposed MCLT architecture has been coded in Python language using Keras library (Chollet 2017) with TensorFlow in the backend and executed on Intel® Core™ i7- 4770 CPU @3.40 GHz having four cores. The total samples for a month were randomly divided into training and testing samples, with 30% of the total samples as the testing samples. This procedure of random division of the total samples into training and testing samples, followed by the training and testing of the MCLT was repeated 20 times in order to determine the mean accuracy values. This procedure, thus, accounted for the randomness in splitting into training and testing. Moreover, the splitting technique was applied by ensuring that the input values of each testing sample should not overlap (*i.e.* disjoint) with the input values of the training samples.

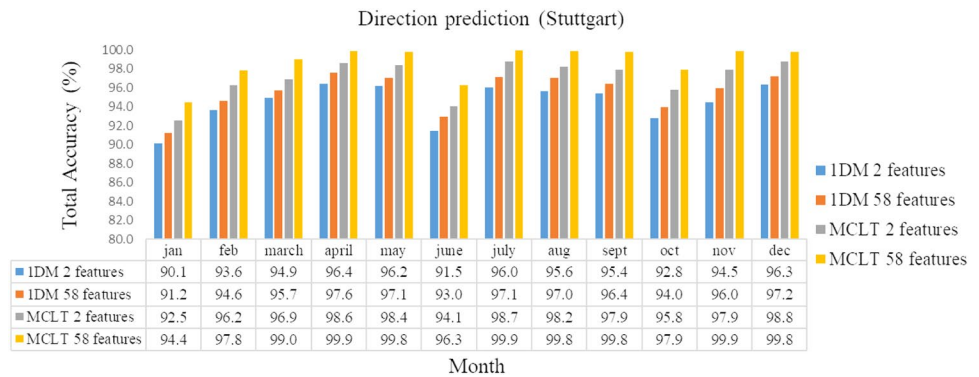
Further, Adaptive Synthetic Sampling (ADASYN) technique (He et al. 2008) was used to enhance the number of training samples for better learning of the MCLT. ADASYN generates samples of the minority class according to their density distributions and avoids over-sampling. The number



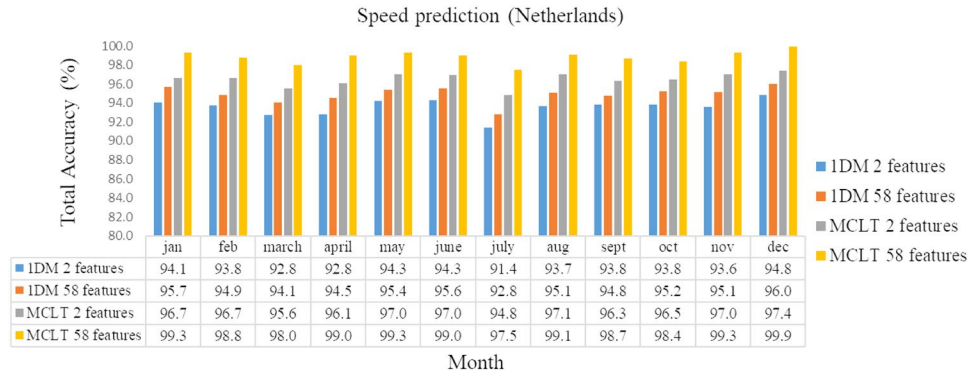
**Fig. 5** Total accuracies in percentage for different months of Stuttgart for dominant speed prediction



**Fig. 6** Total accuracies in percentage for different months of Stuttgart for dominant direction prediction



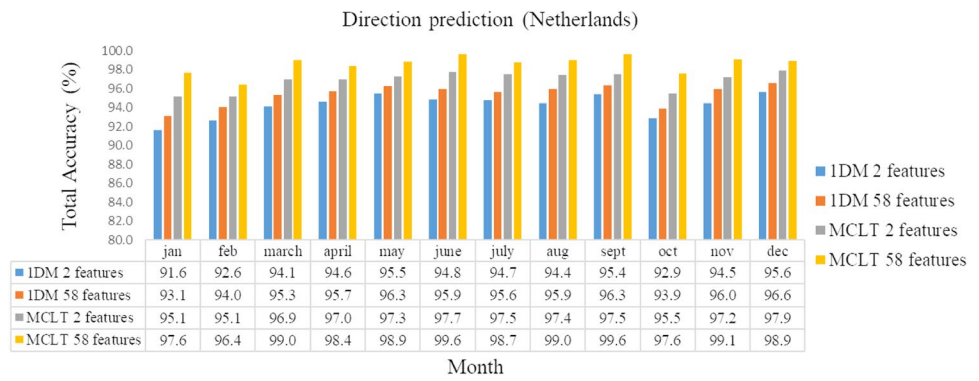
**Fig. 7** Total accuracies in percentage for different months of Netherlands for dominant speed prediction



of feature maps in  $C_1, C_2, C_3$  and  $C_4$  of each of  $CNN_1, CNN_2, CNN_3, CNN_4,$  and  $CNN_5,$  of the MCLT architecture are 16, 28, 32 and 32, respectively, whereas the number of neurons in  $LSTM_1$  and  $LSTM_2$  are 200 and 200 respectively. Values of  $k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$  and  $k_{10}$  (Table 1) were empirically determined as 0.05, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80 and 1.0, respectively (same for both speed and direction), so that sufficient number of samples occurs in each class (He et al. 2008), by observing the histograms comprising of 21 bins corresponding to 21 classes. Moreover,  $K_B$  and  $K_F$  were taken as 60.  $K_F$  multiplied by the temporal resolution gives a time frame of future prediction as per

user desire. Figure 4 shows the variations in total accuracy of the MCLT with 58 features by varying  $K_B$  (here  $K_F = K_B$ ). In this work,  $K_B$  is taken as 60 as accuracy increases till 60 and after that remains similar as shown in Fig. 4. Exponential Linear Units (ELUs) ((Clevert et al. 2017; Pedamonti 2018)) with  $\alpha$  of 3.0 have been used as activation function in the MCLT. The higher value of  $\alpha$  3.0 was chosen to avoid dead neurons problem during training, with highly variable wind datasets (Clevert et al. 2017; Nair and Hinton 2010). Kernel size of three along with stride of one has been applied for all the convolutional layers. Batch normalisation (Jung et al. 2019) and dropout (Srivastava et al. 2014) of 0.45 have been

**Fig. 8** Total accuracies in percentage for different months of Netherlands for dominant direction prediction



**Table 2** Maximum, minimum, mean and standard deviation total accuracies for dominant speed prediction

TA (%)	Stuttgart				Netherlands			
	IDM 2 features	IDM 58 features	MCLT 2 features	MCLT 58 features	IDM 2 features	IDM 58 features	MCLT 2 features	MCLT 58 features
Maximum (%)	94.1	95.5	96.8	99.1	94.8	96.0	97.4	99.9
Minimum (%)	89.2	90.6	92.4	94.9	91.4	92.8	94.8	97.5
Mean (%)	92.2	93.5	95.1	97.2	93.6	94.9	96.5	98.9
Standard deviation (%)	0.9	0.8	1.2	0.9	1.2	1.6	1.3	0.7

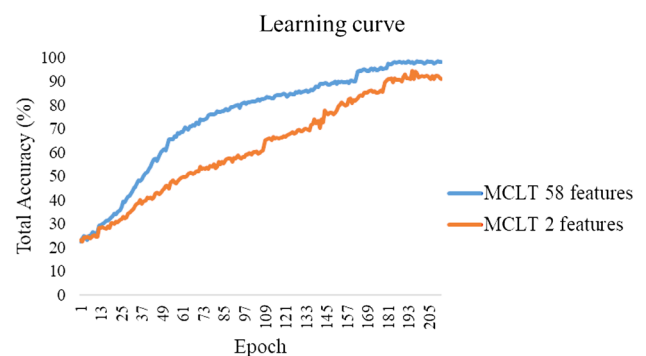
**Table 3** Maximum, minimum, mean, and standard deviation total accuracies for dominant direction prediction

TA (%)	Stuttgart				Netherlands			
	IDM 2 features	IDM 58 features	MCLT 2 features	MCLT 58 features	IDM 2 features	IDM 58 features	MCLT 2 features	MCLT 58 features
Maximum (%)	96.4	97.6	98.8	99.9	95.6	96.6	97.9	99.6
Minimum (%)	90.1	91.2	92.5	94.4	91.6	93.1	95.1	96.4
Mean (%)	94.4	95.6	97.0	98.7	94.2	95.4	96.8	98.6
Standard deviation (%)	1.4	1.7	1.3	0.9	1.1	0.8	1.1	0.9

employed after every convolution layer. This helps to prevent over-fitting, and the MCLT architecture learns better. The parameters comprise weights and biases of neurons of convolutional and LSTM layers that are learned during training. The neurons in a feature map in a convolutional layer share weights and biases. Adam optimisation (Chollet 2017) has been used that takes care of learning rate during training. Initially, weights and biases were initialised using (He et al. 2015) method. Cross entropy loss function has been used during training of the MCLT (Chollet 2017; Nielsen 2015)

### 4.2 Model Accuracies

The total (overall) accuracy for different months of Stuttgart for the test samples, obtained using the MCLT is shown in Figs. 5 and 6. The total accuracy is the number of correct



**Fig. 9** Learning curves for MCLT with 2 and 58 features

predictions divided by the total number of predictions (Congalton and Green 2010). In these figures, MCLT with 58

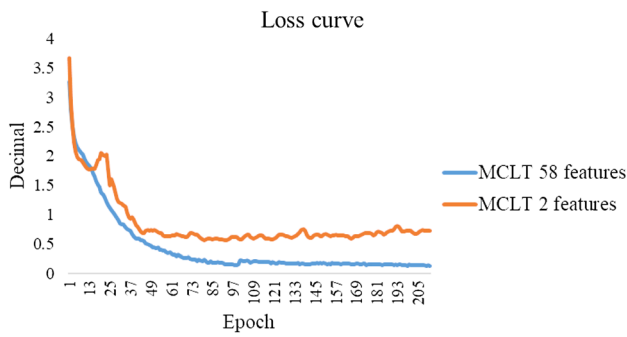


Fig. 10 Loss curves for MCLT with 2 and 58 features

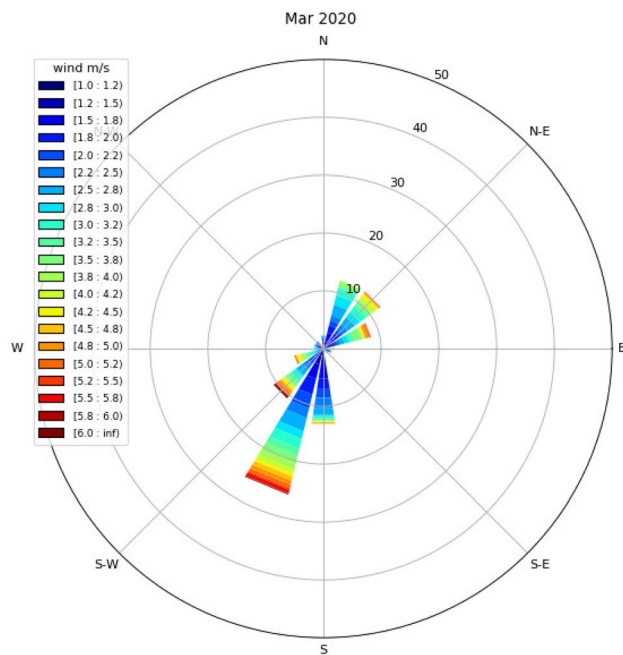


Fig. 11 Wind rose plot for Mar 2020 (sensor's measurements)

features means that all the columns (or features) of  $M_{i,j}$  have been used in the input layers of the MCLT, whereas MCLT with 2 features means only first two columns (of speed and direction) of  $M_{i,j}$  have been used in the input layers. Similar are the interpretations of IDM with 58 features and 2 features. Figures 5 and 6 represent total accuracies for dominant speed and direction prediction for different months of Stuttgart, respectively. Figures 7 and 8 represent total accuracies for dominant speed and direction prediction for different months of Netherlands, respectively.

The maximum, minimum, and mean total accuracies for dominant speed prediction (for Stuttgart) using the MCLT with 58 features are 99.1%, 94.9%, and 97.2%, respectively, as shown in Table 2. The maximum, minimum, and mean total accuracies for dominant speed prediction (for

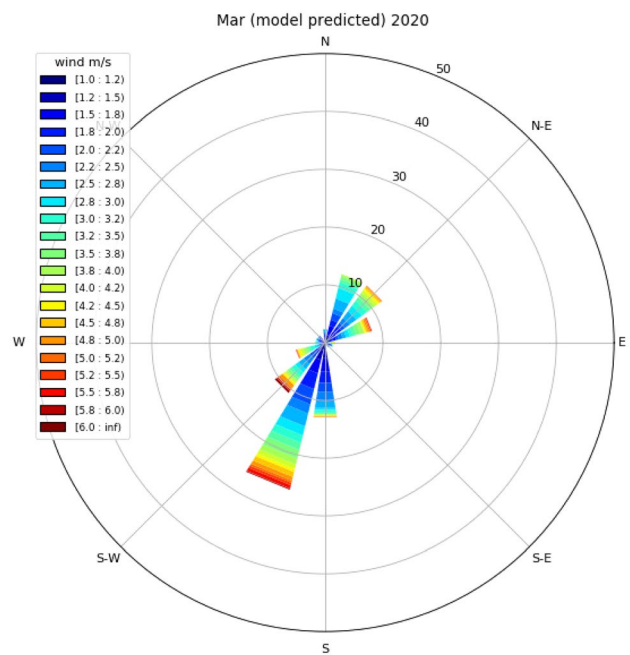


Fig. 12 Wind rose plot for Mar 2020 (model predictions)

Stuttgart) using the MCLT with 2 features are 96.8%, 92.4%, and 95.1%, respectively (Table 2). Similarly, the maximum, minimum, and mean total accuracies for dominant direction prediction (for Stuttgart) using MCLT with 58 features are 99.9%, 94.4%, and 98.7%, respectively (Table 3). The maximum, minimum, and mean total accuracies for dominant direction prediction (for Stuttgart) using MCLT with 2 features are 98.8%, 92.5%, and 97.0%, respectively (Table 3). Figures 5, 6, 7 and 8, Tables 2 and 3 also represent results when the IDM architecture with 2 and 58 features is used for prediction. Learning curves and loss curves (for speed prediction) of January month's test samples of Stuttgart using the MCLT with 2 and 58 features are shown in Figs. 9 and 10, respectively.

### 4.3 Discussion

The proposed MCLT architecture shows promising results for dominant wind speed and direction prediction of temporal wind datasets from Stuttgart and Netherlands. Below subsections 4.3.1, 4.3.2 and 4.3.3 discuss the results with the help of rose plot, comparison among 2 and 58 features, and comparison with other suitable approaches, respectively.

#### 4.3.1 Rose Plots

Wind rose plot helps in the visualisation of wind speed and direction in the same graph, in a circular format. The length of each spoke around the circle indicates the number of times (count) that the wind blows from the indicated direction.

**Table 4** Difference in accuracies of MCLT with 58 features and IDM with 2 features. Positive value denotes MCLT has higher accuracy than IDM

TA (%)	Stuttgart		Netherlands	
	Dominant wind speed prediction	Dominant wind direction prediction	Dominant wind speed prediction	Dominant wind direction prediction
Maximum (%)	5.7	4.3	6.1	4.8
Minimum (%)	5.0	3.5	5.1	4.0
Mean (%)	5.0	4.3	5.3	4.4

Colors along the spokes indicate classes of wind speed. The data of March (Mar) 2020 of Stuttgart are used to represent the real-world sensor's measurements (ground-truth values) and prediction outcomes of the MCLT in Figs. 11 and 12, respectively. The high resemblance among Figs. 11 and 12, indicates that the prediction results are similar to the ground-truth values. This augments visually the accuracies obtained previously in the results Sect. 4.2. In these figures, there are 21 different color ranges denoting the wind speed divided into 21 classes with the wind rose circular format shows the direction the winds blew from. The varying spoke length around the circle shows how often the wind blew from that direction, highlighting the wind nature insight from the indicated directions in this study.

#### 4.3.2 Comparison Between 2 and 58 Features

The 58 multiple features in the input layers help the MCLT to learn the temporal variations in the samples. These features are based on percentage difference, standard deviation, correlation coefficient, eigenvalues, and entropy, that are calculated by taking into account some of the nearby temporal values. As the temporal values adjacent to a time instance change, the values of these features also adapt to these changes. Thus, these features help in comprehensive description of wind speed and direction, describing the trends like increase, decrease, stationary, sudden turbulence, rate of increase and decrease, deviation from the mean, behaviour of speed with respect to direction (*i.e.* correlation), energy (*i.e.* entropy) of the adjacent temporal values and its variation. Therefore, they provide additional information about samples. Moreover, the movements of the 1D kernels in the convolutional layers further help the convolutional layers to learn their own features in the form of weights and biases during the training phase of the MCLT. When only two features were used in the input layers of the MCLT, maximum total accuracy was 96.8% and 97.4% for Stuttgart and Netherlands, respectively, for speed (Table 2) and 98.8% and 97.9% for Stuttgart and Netherlands, respectively, for direction (Table 3). The maximum total accuracy for MCLT with 58 features is increased by 2.3% and 2.5% for Stuttgart and Netherlands, respectively, for speed (Table 2) and by 1.1% and 1.6% for Stuttgart and Netherlands, respectively,

for direction (Table 3) in comparison to MCLT with 2 features. Similarly, the effect of these 58 features over 2 features can also be seen in the case of IDM (Table 2, Table 3) where maximum total accuracy for speed improved by 1.4% and 1.2% for Stuttgart and Netherlands, respectively, and by 1.2% and 1.0% for Stuttgart and Netherlands, respectively, for direction. Learning of the MCLT with 58 features is better than 2 features as shown by respective learning curves in Fig. 9 and by the loss curves in Fig. 10.

Convolutional layers ( $C_1, C_2$ ) near the input layers learn the features in smaller neighbourhood, while the convolutional layers ( $C_3, C_4$ ) near the output layer learn features in larger neighbourhood (He et al. 2016; Huang et al. 2017; Krizhevsky et al. 2012; Xie et al. 2017).  $C_3$  takes as input the learnt features from both  $C_1$ , and  $C_2$ , while  $C_4$ , takes as input the features from  $C_1, C_2$ , and  $C_3$ , therefore, the MCLT gets trained by learning features at different scales. Further, as the convolutional layers ( $C_3, C_4$ ) are connected to all the previous convolutional layers, providing that gradient vanishing problem would not occur, *i.e.* MCLT learning does not slow down during training via back-propagation (He et al. 2016; Huang et al. 2017; Xie et al. 2017). Moreover, LSTM layers after the last convolutional layers ( $C_4$ ), have memory units that retain the learnt features from previous output of the neurons and operate upon them with features learnt from the current output of the neurons. This gives better learning over the fully connected layers (present in traditional CNNs) that lack these memory units. Additionally, the memory units in the LSTM help in finding correlations between patterns learnt across different time, as a recent pattern is a function of pattern learnt at previous time.

#### 4.3.3 Comparison with Existing Related Work

The proposed MCLT architecture is compared with the IDM. The MCLT with 2 features as well as 58 features performs better than the IDM with 58 features, as shown in Figs. 5, 6, 7 and Fig. 8 for both Stuttgart and Netherlands. Minimum, maximum and mean total accuracies of the MCLT with 58 features are compared with IDM with 2 features in Table 4. Thus, the MCLT performs better than the IDM. Moreover, the MCLT with 58 features efficiently predicts for the larger time frame in future ( $K_F$  as 60, multiply

by the temporal wind dataset resolution) whereas the IDM with 2 features could only predict for 50 values in future (Harbola and Coors 2019b). Furthermore, the MCLT is also compared with the methods in the existing literature that are near to the proposed architecture. 1D CNN algorithm proposed by (Liu et al. 2018) has used regression technique working on the smoothed and filtered data, thereby losing the originality of the wind dataset. The same samples comprising of  $K_B = 60$ , input values without applying smoothening and filtering, that have been employed for the proposed MCLT, are also used to train and test the regression CNN architecture (Liu et al. 2018). In this case, Symmetric Mean Absolute Percentage Error (SMAPE) (Flores 1986) for wind speed in Stuttgart is 20.5% for  $K_B = 8$  and reaches up to 25.5% for  $K_B = 60$ , while 14.9% for  $K_B = 15$  and reaches up to 21.2% for  $K_B = 60$  for wind speed in Netherlands. SMAPE of wind direction were moreover similar to these patterns. It may be noted that, here, the labels of the samples are designed using the real values (*i.e.* regression); whereas, MCLT predictions are based on the class labels (*i.e.* classification). SMAPE was also calculated for MCLT prediction results. The center of the interval of each class (Table 1) was calculated by taking the average of lower range and upper range. The class predicted by the MCLT for a test sample along with the corresponding center of the interval of the predicted class was noted. This was done for all the test samples. SMAPE was calculated using the center of the interval of the predicted class and the center of the interval of the ground-truth class for all the test samples. SMAPE for wind speed in Stuttgart was 3.5% for  $K_B = 8$ , 1.4% for  $K_B = 35$  and 0.4% for  $K_B = 60$ . Similar were the SMAPE values for wind direction. As the future time frame of prediction increases, error also increases using the state-of-the-art CNN-based regression method (Liu et al. 2018). However, the proposed MCLT based on classification shows high accuracy and mean total accuracy reaches up to 99.9% for  $K_B = 60$  (and SMAPE = 0.4%), without smoothening and filtering the original wind data. Thus, the proposed MCLT method gives satisfactory results for predicting dominant speed and direction for a greater time duration in the future unlike (Liu et al. 2018). The limitation of 58 input features is only due to hardware constraints and more features can be designed with more GPUs. The accuracies achieved using the designed MCLT can be further improved with better hardware resources using a greater number of feature maps, neurons, convolutional and LSTM layers. Thus, the use of multiple features at various levels in the MCLT, *viz.* (a) 58 features in the input layers, (b) inputting a convolutional layer with features from all the previous convolutional layers, and (c) retaining the memory of learnt features by LSTM from previous outputs (of neurons) during training, helps the proposed architecture to predict the dominating speed and direction classes with good accuracy. Further, as the number

of classes of the samples increases, detailed patterns of the nonlinear nature of the wind can be analyzed but at the same time ambiguity in classification also increases. However, the proposed MCLT architecture is able to overcome this ambiguity by learning multiple features and performs well even with 21 classes.

## 5 Conclusion

Wind speed and direction predictions are critical to new wind farm installations and for smart city planning in proper utilisation of green and freely available energy resources. In this paper, a deep learning architecture is successfully designed and demonstrated to predict the dominant speed and direction classes in the future for the temporal wind datasets. The proposed MCLT architecture uses 58 features in the input layers that are designed using wind speed and direction values. These features are based on percentage difference, standard deviation, correlation coefficient, eigenvalues, and entropy, for comprehensively and efficiently describing the wind trend and its variations. LSTM layers at the end of the last convolutional layers have memory units that employ features learnt during current as well as the previous output of the neurons. Further, densely connected convolutional layers in the MCLT help the convolutional layers to learn features of other convolutional layers as well. Two large wind datasets from Stuttgart and Netherlands are used for training and testing the MCLT. The maximum total accuracies for speed and direction prediction are 99.9% and 99.9%, respectively. The average total accuracies reach up to 98.9% and 98.7%, for speed and direction prediction, respectively. The model's real-world prediction demonstration analysis support the novelty of the work while explaining visually with the help of wind rose plots. Thus, the MCLT shows promising results for different wind datasets. The limited hardware resources restricted this study to using 58 features in the input layers.

Accuracies achieved in this work could be further improved with better hardware resources using a greater number of feature maps, neurons, convolutional and LSTM layers. Most importantly, this analysis would help to devise a new set of inflow boundary conditions that are prerequisites for obtaining reasonable wind flow fields. Computational Fluid Dynamics (CFD) simulations use wind speed and direction measured at a nearby meteorological station as the inflow boundary conditions, which could be decided using the proposed work. The performed wind nature analysis has the potential for helping city development authorities and planner in identifying high wind areas with detailed temporal wind information about its magnitude and dominating direction and for selecting the optimum wind energy conversion systems. In future, the authors will improve the

proposed algorithm and work for the visual analysis of the temporal wind dataset. Moreover, the proposed deep learning concept for temporal data could be implemented to other time-series datasets like finance, trends analysis, and sensor health monitoring applications.

**Acknowledgements** The wind dataset of Stuttgart has been downloaded from the website of City of Stuttgart (<https://www.stadtlima-stuttgart.de>), Germany, Department of Environmental Protection, Department of Urban Climatology (DEPUC). The wind dataset of Netherlands has been provided by The Royal Netherlands Meteorological Institute (KNMI) website ([https://www.projects.knmi.nl/klimatologie/onderzoeksgegevens/potentiele\\_wind/](https://www.projects.knmi.nl/klimatologie/onderzoeksgegevens/potentiele_wind/)). This work has been developed within the Joint Graduate Research Training Group Windy Cities. This research program (Funding number: 31-7635.65-16) is supported by the Ministry of Science, Research and Art of Baden-Württemberg, Germany.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Aissou S, Rekioua D, Mezzai N, Rekioua T, Bacha S (2015) Modeling and control of hybrid photovoltaic wind power system with battery storage. *Energy Convers Manag* 89(1):615–625. <https://doi.org/10.1016/j.enconman.2014.10.034>
- Aslipour Z, Yazdizadeh A (2019) Identification of nonlinear systems using adaptive variable-order fractional neural networks (case study: a wind turbine with practical results). *Eng Appl Artif Intell* 85(1):462–473. <https://doi.org/10.1016/j.engappai.2019.06.025>
- Birenbaum A, Greenspan H (2017) Multi-view longitudinal cnn for multiple sclerosis lesion segmentation. *Eng Appl Artif Intell* 65(7):111–118. <https://doi.org/10.1016/j.engappai.2017.06.006>
- Chollet F (2017) *Deep learning with python*, Manning Publications Co., Greenwich, CT, USA ©2017, p 384
- Clevert D, Unterthiner T, Hochreiter S (2017) Fast and accurate deep network learning by exponential linear units (elus). *Proc Int Conf Learn Rep* 47:21–28
- Colak I, Sagiroglu S, Yesilbudak M (2012) Data mining and wind power prediction: a literature review. *Renew Energy* 46(6):241–247. <https://doi.org/10.1016/j.renene.2012.02.015>
- Congalton G R, Green K (2010) *Assessing the accuracy of remotely sensed data: principles and practices.*, 8th edn. CRC press, Boca Raton
- Daraeepour A, Echeverri DP (2014) Day-ahead wind speed prediction by a neural network-based model. In: *Proceedings of the IEEE international conference on innovative smart grid technologies* ISGT - 2014, vol 17, pp 220 – 226. <https://doi.org/10.1109/ISGT.2014.6816441>
- de Andrade A (2019) Best practices for convolutional neural networks applied to object recognition in images. [arXiv:1910.13029](https://arxiv.org/abs/1910.13029)
- De Giorgi M, Campilongo S, Ficarella A, Congedo P (2014) Comparison between wind power prediction models based on wavelet decomposition with least-squares support vector machine (ls-svm) and artificial neural network (ann). *Energies* 7(8):5251–5272. <https://doi.org/10.3390/en7085251>
- De Giorgi MG, Ficarella A, Russo MG (2009) Short-term wind forecasting using artificial neural networks (anns). *WIT Trans Ecol Environ* 121(1):197–208. <https://doi.org/10.2495/ESU090181>
- Duta CI, Liu L, Zhu F, Shao L (2020) Improved residual networks for image and video recognition. [arXiv:2004.04989](https://arxiv.org/abs/2004.04989)
- El-Fouly T, El-Saadany F, Salama M (2008) One day ahead prediction of wind speed and direction. *IEEE Trans Energy Convers* 23(1):191–201. <https://doi.org/10.1109/TEC.2007.905069>
- Flores BE (1986) *A pragmatic view of accuracy measurement in forecasting*, 2nd edn. Omega, Oxford
- Ghilani CD (2010) *Adjustment computations spatial data analysis*, 5th edn. Wiley, Amsterdam
- Harbola S, Coors V (2019a) Comparative analysis of lstm, rf and svm architectures for predicting wind nature for smart city planning. *ISPRS Annals of the Photogrammetry. Remote Sens Spatial Inf Sci IV-4/W9*:65–70. <https://doi.org/10.5194/isprs-annals-IV-4-W9-65-2019>. <https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-4-W9/65/2019/>
- Harbola S, Coors V (2019) One dimensional convolutional neural network architectures for wind prediction. *Energy Convers Manag* 195(1):70–75. <https://doi.org/10.1016/j.enconman.2019.05.007>
- He H, Bai Y, Garcia EA, Li S (2008) Adasyn: adaptive synthetic sampling approach for imbalanced learning. In: *International joint conference on neural networks*, vol 11. <https://doi.org/10.1007/BF03251578>
- He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers surpassing human-level performance on imagenet classification. *ArXivorg* 23(9). [arXiv:1502.01852v1](https://arxiv.org/abs/1502.01852v1)
- He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit* 112:770–778. <https://doi.org/10.1109/CVPR.2016.90>
- Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit* 101:2261–2269. <https://doi.org/10.1109/CVPR.2017.243>
- Janssens O, Noppe N, Devriendt C, Walle R, Hoecke SV (2016) Data-driven multivariate power curve modeling of offshore wind turbines. *Eng Appl Artif Intell* 55(7):331–338. <https://doi.org/10.1016/j.engappai.2016.08.003>
- Jung D, Jung W, Lee S, Rhee W, Ahn JH (2019) Restructuring batch normalization to accelerate cnn training. *ArXiv* 1(3). [arXiv:1807.01702](https://arxiv.org/abs/1807.01702)
- Jursa R, Rohrig K (2008) Short-term wind power forecasting using evolutionary algorithms for the automated specification of artificial intelligence models. *Int J Forecast* 24(4):694–709. <https://doi.org/10.1016/j.ijforecast.2008.08.007>
- Kang A, Tan Q, Yuan X, Lei X, Yuan Y (2017) Short-term wind speed prediction using eemd-lssvm model. *Adv Meteorol* 1(1). <https://doi.org/10.1155/2017/6856139>
- Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Stanford Vis Lab* 60(6):84–90. <https://doi.org/10.1145/3065386>

- Kuo CC (2016) Understanding convolutional neural networks with a mathematical model. *J Vis Commun Image Rep* 41(1):406–413. <https://doi.org/10.1016/j.jvcir.2016.11.003>
- Kusiak A, Zhang Z (2010) Short-horizon prediction of wind power: a data-driven approach. *IEEE Trans Energy Convers* 25(4):1112–1122. <https://doi.org/10.1109/TEC.2010.2043436>
- Kusiak A, Zheng H, Song Z (2009) Models for monitoring wind farm power. *Renew Energy* 34(3):583–590. <https://doi.org/10.1016/j.renene.2008.05.032>
- Kusiak A, Zheng H, Song Z (2009) Short-term prediction of wind farm power: a data mining approach. *IEEE Trans Energy Convers* 24(1):125–136. <https://doi.org/10.1109/TEC.2008.2006552>
- Lawan SM, Abidin WAWZ, Chai WY, Baharun A, Masri T (2014) Different models of wind speed prediction; a comprehensive review. *Int J Sci Eng Res* 5(8):1760–1768. <https://www.ijser.org/researchpaper/Different-Models-of-Wind-Speed-Prediction-A-Comprehensive-Review.pdf>
- Liu H, Mi X, Li Y (2018) Smart deep learning based wind speed prediction model using wavelet packet decomposition, convolutional neural network and convolutional long short term memory network. *Energy Convers Manag* 166(12):120–131. <https://doi.org/10.1016/j.enconman.2018.04.021>
- Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. *ArXivorg* 24(8):3431–3440. <https://doi.org/10.1109/CVPR.2015.7298965>
- Louka P, Galanis G, Siebert N, Kariniotakis G, Katsafados P, Pytharoulis I, Kallos G (2008) Improvements in wind speed forecasts for wind power prediction purposes using kalman filtering. *J Wind Eng Ind Aerodyn* 96(1):2348–2362. <https://doi.org/10.1016/j.jweia.2008.03.013>
- Marović I, Sušanji I, Ožanić N (2017) Development of ann model for wind speed prediction as a support for early warning system. *Complexity* 10(1). <https://doi.org/10.1155/2017/3418145>
- Martínez-Arellano G, Nolle L, Cant R, Lotfi A, Windmill C (2014) Characterisation of large changes in wind power for the day-ahead market using a fuzzy logic approach. *KI - Künstliche Intelligenz* 28(13):239–253. <https://doi.org/10.1007/s13218-014-0322-3>
- Miranda M, Dunn R (2006) One-hour-ahead wind speed prediction using a bayesian methodology. *IEEE Power Eng Soc General Meet* 6(21):1–6. <https://doi.org/10.1109/PES.2006.1709479>
- Monfared M, Rastegar H, Kojabadi HM (2009) A new strategy for wind speed forecasting using artificial intelligent methods. *Renewable Energy* 34(1):845–848. <https://doi.org/10.1016/j.renene.2008.04.017>
- Nair V, Hinton G (2010) Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the International Conference on Machine Learning (ICML-10)*, vol 27, pp 1–7
- Nielsen M (2015) *Neural networks and deep learning*. <http://neuralnetworksanddeeplearning.com/>. Accessed 18 July 2021
- Pedamonti D (2018) Comparison of non-linear activation functions for deep neural networks on mnist classification task. *arXiv:1804.02763*
- Qi CR, Su H, Niessner M, Dai A, Yan M, Guibas LJ (2016) Volumetric and multi-view cnns for object classification on 3d data. In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, vol 1. <https://doi.org/10.1109/CVPR.2016.609>
- Reed TC, Fiffick SA, Sawyers DR (2011) Wind data analysis and performance predictions for a 400-kw turbine in northwestern ohio. In: *American Society for Engineering Education, ASEE - 2011. Proceedings*, vol 11, pp 1–9
- Sapronova A, Johannsen K, Thorsnes E, Meissner C, Mana M (2016) Deep learning for wind power production forecast. In: *CEUR Workshop - 2016, Proceedings*, vol 18, pp 28–33
- Srivastava N, Hinton G, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(56):1929–1958. <http://jmlr.org/papers/v15/srivastava14a.html>
- Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit* 7:1–9. <https://doi.org/10.1109/CVPR.2015.7298594>
- Tarade RS, Katti PK (2011) A comparative analysis for wind speed prediction. In: *Proceedings of international conference on energy, automation and signal, ICEAS - 2011*, vol 21, pp 556–561. <https://doi.org/10.1109/ICEAS.2011.6147167>
- Treiber N, Heineremann J, Kramer O (2016) Wind power prediction with machine learning. *Stud Comput Intell* 645(1). [https://doi.org/10.1007/978-3-319-31858-5\\_2](https://doi.org/10.1007/978-3-319-31858-5_2)
- Vargas L, Paredes G, Bustos G (2010) Data mining techniques for very short term prediction of wind power. *IREP Sympos Bulk Power Syst Dyn Control - VIII (IREP)* 8(6):1–7. <https://doi.org/10.1109/IREP.2010.5563273>
- Vladislavleva E, Friedrich T, Neumann F, Wagner M (2013) Predicting the energy output of wind farms based on weather data: important variables and their correlation. *Renewable Energy* 50(6):236–243. <https://doi.org/10.1016/j.renene.2012.06.036>
- Vogado H, Veras MSR, Araujo HDF, Silva R, Aires R (2018) Leukemia diagnosis in blood slides using transfer learning in cnns and svm for classification. *Eng Appl Artif Intell* 72(1):415–422. <https://doi.org/10.1016/j.engappai.2018.04.024>
- Wang H, Li G, Wang G, Peng J, Jiang H, Liu Y (2017) Deep learning based ensemble approach for probabilistic wind power forecasting. *Appl Energy* 188(33):56–70. <https://doi.org/10.1016/j.apenergy.2016.11.111>
- Weinmann M, Schmidt A, Mallet C, Hinz S, Rottensteiner F, Jutzi B (2015) Contextual classification of point cloud data by exploiting individual 3d neighbourhoods. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol 55. <https://doi.org/10.5194/isprsannals-II-3-W4-271-2015>
- Xie S, Girshick R, Doll P (2017) Aggregated residual transformations for deep neural networks. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit* 107:2261–2269. <https://doi.org/10.1109/CVPR.2017.634>
- Yang H, Chen Y (2019) Representation learning with extreme learning machines and empirical mode decomposition for wind speed forecasting methods. *Artif Intell* 277(1). <https://doi.org/10.1016/j.artint.2019.103176>
- Yesilbodak M, Sagiroglu S, Colak I (2017) A novel implementation of knn classifier based on multi-tupled meteorological input data for wind power prediction. *Energy Convers Manag* 135(7):434–444. <https://doi.org/10.1016/j.enconman.2016.12.094>
- Yesilbudak M, Sagiroglu S, Colak I (2013) A new approach to very short term wind speed prediction using k-nearest neighbor classification. *Energy Convers Manag* 69(1):77–86. <https://doi.org/10.1016/j.enconman.2013.01.033>
- Yuan X, Chen C, Yuan Y, Huang Y, Tan Q (2015) Short-term wind power prediction based on lssvm-gsa model. *Energy Convers Manag* 101(1):393–401. <https://doi.org/10.1016/j.enconman.2015.05.065>
- Zhao Z, Zheng P, Xu S, Wu X (2019) Object detection with deep learning: a review. *arXiv* 2(11). [arXiv:1807.05511](https://arxiv.org/abs/1807.05511)