Spatio-Temporal Wind Speed Forecasting with Models based on Convolutional Neural Networks

IIF-SAS Research Award Report

Bruno Q. Bastos[1], Fernando L. Cyrino Oliveira[1], Ruy L. Milidiú[2]

[1]Industrial Engineering Department, PUC-Rio, Brazil
[2]Informatics Department, PUC-Rio, Brazil

Abstract

The increasing penetration of intermittent renewable energy in power generation brings new challenges to the operation and planning of power systems. One way of supporting operation planning under such circumstance is by enhancing the predictability of renewable resources via accurate and informative forecasting. Convolutional Neural Networks (Convnets) provide a successful Deep Learning technique to process space structured multi-dimensional data. In our work, we investigate the use of Convnets to predict hourly wind speed for a single location and for multiple locations. To develop forecasting models with Convnets, we adapt techniques originally developed to predict frames in video (a spatio-temporal task) to the problem of wind speed forecasting with multiple explanatory variables. We propose the U-Convolutional model, which combines U-Net and Convnet, to solve single-site spatio-temporal wind speed prediction. We also propose a family of architectures, which we name ComPonentNet family, to solve multi-site spatio-temporal wind speed forecasting. The proposed architectures are tested on datasets acquaride from the Climate Forecast System Reanalysis (CFSR) dataset. The U-Convolutional models are compared against other deep learning architectures (Inception, Residual Inception, Residual Convnets, and Convnets), against fully-connected neural networks, and against benchmark statistical methods such as ARIMA and BATS. The results indicate that both solutions are promising for wind speed forecasting.

Chapter 1

Introduction

Worldwide, the awareness concerning climate change has increased since International Panel on Climate Change (IPCC) indicated that human activity was the main cause of recent global warming and of increase in greenhouse gas (GHG) emissions [1]. The use of energy was considered one of the main drivers of the emission of greenhouse gases [1]. As a consequence, the transformation of current energy system aiming long-term sustainability is of great importance to mitigate climate change. Renewable energy sources are in the center of this debate – being a potential tool for mitigating the emission of greenhouse gases [2]. This understanding has been reflected on the development of policies that give a basis for the adoption of long-term sustainable energy resources, such as wind and solar. Policies range from financial incentives (e.g., feed-in tariffs) to penetration targets, and include many other mechanisms (such as renewables quota in energy portfolio, renewable energy auctions, and others). As a consequence of the execution of such policies, penetration of intermittent renewable energy, such as wind and solar energy, has been increasing continuously [3].

Power generation from intermittent renewable resources may vary greatly over the course of the day. Moreover, generation from such sources may present sudden drops throughout the day. Thus, the growing adoption of intermittent energy solutions poses operational and planning challenges to modern power systems. The growth of intermittent energy resources increases the variability of power generation, adding uncertainty into the power generation. With higher variability in power generation, advanced controls for balancing very-short-term generation and demand may be required [4]. In order to backup intermittent power, it may also be required an increase in energy reserves, primarily from hydrothermal units [5]. All of these may lead to an increase in operational costs.

One way of addressing the uncertainty in renewable generation is by producing accurate and informative forecasts, which may serve as a tool for operation planning (short-term), maintenance scheduling (mid-term) and capacity planning (long-term) [4]. Many studies and research addressed the renewable energy forecasting. Methods for wind forecasting include physical models, statistical models and machine learning models. Physical models use nummerical methods, which are built upon the laws that describe atmospheric phenomena, to estimate the state of the atmosphere at a given time [6], [7]. According to [8], this type of model is not adequate for short-term prediction. Traditional statistical models use historic wind data to predict future wind data. This category include classic time series methods, such as the ones proposed by Box and Jenkins [9], which are commonly used in wind speed forecasting [10]. These methods usually present good results for short-, medium- and long-term wind speed forecasting [10]. Spatial correlation models use information of neighboring sites to estimate wind speed at a given coordinate where measurement is not available. Machine learning methods include neural networks [11], support vector machines [12], fuzzy models [13], and many others. This methods are known to outperform other methods due to capability of mapping non-linear functions. For a review of wind forecasting, refer to the following articles [8], [14]–[16] and references therein.

Technological advances in metering and communication have led to an increasing availability of data, which is being collected in higher frequencies and spanning broader areas. With availability of data at different locations over time, spatio-temporal approaches have been gaining ever more attention [17]. The use of spatially-distributed information in modeling has improved accuracy of temporal forecasts overall. For example, in [18], spatio-temporal approach of sparse vector auto-regression (sVAR) outperformed single-site model benchmarks, such as auto-regressive (AR) and vector auto-regressive (VAR) models, on the task of one-step ahead wind power forecast. In [19], spatio-temporal approach of gradient boosted regression trees (GBRTs) provided competitive results of solar power forecasts compared to single-site benchmark models of AR and GBRT, among others.

The notion that the use of spatio-temporal information may improve wind and solar power generation forecasts is quite reasonable, since wind and solar power generation are affected by meteorological conditions which may span through large regions. In this sense, for example, the use of information from nearby meteorological stations could be useful to predict information at a target wind farm. Additionally, the use of spatio-temporal information of meteorological variables, such as humidity, temperature, precipitation, etc., may also be useful in the process of wind and solar modeling. Statistical and machine learning methods have been adopted to forecast renewable power generation with spatio-temporal approaches, some of which consider information of meteorological variables (e.g., [17], [19]). Examples of statistical methods used in spatio-temporal forecasting include compressive spatio-temporal forecasting (CSTF) model [17], spatio-temporal vector auto-regressive model [20], sparse vector auto-regressive model (sVAR) [18], trigonometric direction diurnal (TDD) [21], and others [22]–[24]. Examples of machine learning methods include ensembles of decision trees (DTs) and support vector regression (SVR) [25], gradient boosted regression trees (GBRTs) [19], and fuzzy model [26].

Recently, deep learning framework and techniques have been gaining ever more attention across different areas due to breakthroughs in computer vision, machine translation, speech recognition, and other complex tasks. Convolutional neural networks (Convnets) have been one of the techniques responsible for breakthroughs in computer vision (see [27]). Convnets are neural networks specially designed to process multi-dimensional data, within which the ordering of the elements matters - such as images (2D arrays). Advances in Convnets (e.g., factorized convolutions [28], residual mappings [29]), allowed further improvements to the technique.

Spatio-temporal prediction using models that are based on Convnets is an ongoing research topic in many areas, such as meteorology [30] and computer vision [31]–[33]. Shi et al. [30] proposed a model, known as ConvLSTM, which extends recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) cells [34] to perform convolutional operation. The model was used to solve precipitation nowcasting and video frame prediction. Mathieu et al. [31] proposed a multi-scale framework based on Convnets and adversarial training [35] to predict future frames of video. More recent works on video frame prediction suggest the decomposition of video into content and motion [32], [33]; moreover, these works combine a multitude of techniques such as Convnets, RNNs and adversarial training. U-Net [36], a technique originally developed for image segmentation, has been recently used for video frame prediction [37].

Convnets have already been adopted in the realm of renewable energy prediction. Zhu et al. [38] propose a Convnet that provides univariate wind power predictions by rearranging the univariate data into a 2-D vector. Liu et al. [39] decompose wind speed data into low and high frequencies series. The low frequency series is predicted via Convnet with LSTM layer on top of it, whereas the high frequency series are predicted with standard Convnets. Chen et al. [40] uses a combination of multi-factor correlation, Convnets and LSTM to predict wind speed at a target site. In [40], the authors build a 3D matrix containing meteorological factors for all sites at historical time and use Convnets to extract spatial features of meteorological factors at various sites and time. LSTM is then used to extract temporal features.

1.1 Renewable Energy in Brazil

Brazil is one of the largest and most populated countries in the world. To supply electricity to the population, the Brazilian power system relies on a hydrothermal system. Hydro power generation is the main source of electricity generation, and thermal generation complements hydro power [41]. In 2017, hydro power accounted to approximately 63.8% of total installed capacity in Brazil and to 63.1% (371 TWh) of total electricity generation produced in Brazil. On the other hand, in 2017 thermal power plants were the second main supplier of electricity in Brazil, and accounted for approximately 26.5% of installed capacity.

Hydro generation in Brazil includes large-scale power plants, some of which have large reservoirs that may be used to regulate the dispatch of power plants throughout the day. (Therefore, these hydro power plants with large reservoirs serve as a “battery” to the power system, storing and dispatching energy when necessary.) Hydro is considered a low-cost and highly efficient energy resource [42]. Being costlier than hydro power, thermal generation complements hydro generation and is used as backup to the power system during droughts.

Being highly dependent on hydrology, and covering a vast territory, operation of the Brazilian power system is very challenging. It has to consider future hydrological conditions on different river basins and expected opportunity costs to define whether to dispatch hydro power or thermal power at present time. The decision problem is solved using a multi-stage stochastic optimization set-up [43]. The dispatch of power plants in centralized by an independent system operator (ISO), which defines the dispatch based on costs [44].

Wind energy has been the fastest growing energy resource in Brazil, being incentivized by energy auctions throughout the last decade. Wind installed capacity increased from 1.43 GW in 2011 to 12.28 GW in 2017 – an annual growth rate of 36.00% a.a. Wind energy has also seen a great increase in terms of energy generation – with a growth of 48.15% a.a. Nowadays, 483 wind farms are in operation in Brazil, accouting for 7.82% of total installed capacity. Moreover, 135 wind farms are under construction and 126 wind farms are yet to be constructed.

With the increase of renewables in the electricity mix (translated into higher variability in power generation), and with restrictions on reservoir storage capacity (meaning lower regulating capability), Brazilian power system operation and planning will be even more challenging. As pointed out by Drank and Ferreira [42], to find a solution to this new paradigm, the country needs to undergo deep discussions. Our work targets a small part of the problem, which is related to wind predictability in multiple locations. Specififcally, our case studies target two areas in Brazil - one related to Bahia state and the other to Rio Grande do Norte state (both belonging to Northeast region, the most prominent in wind energy generation in Brazil).

1.2 Contributions

In our work, we address both single-site and multi-site spatio-temporal wind speed forecasting. For single-site forecasting, we propose an architecture which combines U-Net and Convnet, the U-Convolutional model. In this architecture, the U-Net – recently used to predict video frames (see [37]) – is used to synthesize and produce high-level features from spatio-temporal data; the high-level features acquired with U-Net are fed to a Convnet, which produces the wind speed prediction. The proposed architecture is trained jointly; this means that the high-level features obtained via U-Net are produced by optimizing a single value (the wind speed error for a single coordinate). In the context of this work, we also investigate the addition of different elements, such as identity mappings and simplified inception modules (or all of these techniques simultaneously), to the proposed architecture and to standard Convnet. In all architectures, we make the input channels be an explanatory variable [45] that is a spatio-temporal process.

Besides proposing an architecture which differs from the one in Chen et al. [40], our work also differs from Chen et al. [40] in other aspect: we use the meteorological variables (temperature and u- and v-components of wind) to directly map wind speed at a single site. That is, in our approach, the architecture learns the relations between the exploratory variables and output data without being given any specific information of correlation between variables. In this context of single-site, the contributions of this work are the following:

- Proposition of deep learning architecture that combines U-Net architecture and Convnet for forecasting of wind speed at a single location with multiple spatio-temporal explanatory variables as input;
- Investigation of Convnet architectures which include inception modules and identity mappings in the task of spatio-temporal wind speed prediction;
- Comparison of proposed and Convnet architectures against fully-connected neural network modeled in a spatio-temporal setting and against traditional univariate models such as BATS [46], ARIMA [9] and Naïve models;
- We outline a way forward in modeling wind speed with multiple spatio-temporal explanatory data.

For multi-site spatio-temporal wind speed prediction, we design a collection of architectures (which we name ComPonentNet family of architectures), inspired by the U-Net architecture [36], which adopt historic u- and v-components of wind (as input channels) and predict u- and v-components one step ahead of time. With this configuration, the architectures are able to perform multi-step prediction and to provide not only wind speed as output, but also wind direction (both wind speed and wind direction may be calculated from u- and v-components of wind).

The proposed multi-site architectures are composed of three parts, a bottom-section (i.e., the first layers of the architectures, which are closer to the input), a middle-section (i.e., layers that follow the bottom-section layers), and a top-section (i.e., the last layers of the architectures, which follow the middle-section layers, and are closer to the output). As in the U-Net architecture, the bottom-section layers act as a contracting path and top-section layers as a expansion path, and bottom-section layers are connected to the top-section layers.

The proposed architectures process u- and v-wind separately. For example, one of the architectures processes u- and v-components separately at the bottom-section of the architecture (i.e., each component is processed by an specific path); then, it fuses the high-level features at the middle-section of the architecture; finally, it splits the fused features into two paths, which output u- and v-components of wind separately. Differently, other architecture process u- and v-components separately only at top-section. We compare both architectures against the U-Net, which processes u- and v-components alltogether. In the context, we investigate the impacts of processing u- and v-components separately, considering the task of multi-site wind speed forecasting. Considering that both architectures play with components of a unique phenomena, we name these collection of architectures as the *ComPonentNet* family of architectures, which we abbreviate as **CPNet**.

Our contributions to multi-site spatio-temporal forecasting are the following:

- Proposition of spatio-temporal architectures which process u- and v-components of wind separately and are able to perform multi-step prediction for multiple coordinates (sites);
- We outline a way of designing an architecture (by rethinking U-Net architecture) to model phenomena that is composed of multiple factors (in our case, wind phenomena, which is composed of

u- and v-components) for regression tasks;

- We investigate the impact of adopting different processing schemes for u- and v-components of wind on accuracy measures for multi-site wind speed prediction.

1.3 Objective of Research

The aim of this research is to develop spatio-temporal prediction methods for wind speed with deep learning techniques. Specifically, we develop spatio-temporal methods for single-site and multi-site forecasting with architectures which have Convnet and U-Net as a basis. Our contributions include a range of new methods, which include the U-Convolutional model (*U-Conv*) for single-site forecasting and the ComPonentNet architectures (*CPNet*) for multi-site forecasting. Our work provide a basis for new spatio-temporal wind speed forecasting methods based on Convnets. The ideas used for wind speed prediction may also be used to solve other forecasting problems, which involve phenomena that might be decomposed into two or more components (such as wind being decomposed into u- and v-components). The proposed methods may also inspire architectures for other spatio-temporal problems which may take different explanatory variables as input.

Chapter 2

Fundamentals

2.1 Convolutional Neural Networks

Convolutional Neural Networks also known by the acronym *Convnets* are neural networks specially designed to process data that come as multi-dimensional arrays, within which the ordering of the elements in each array matters [47], [48]. As an illustrative example, consider a 2-D array representing an image, which depicts an object. Convnets are composed of stacked non-linear layers, which are sequentially applied to transform the data from raw input to higher-level concepts. Each layer is a mathematical function with trainable weights that maps input values to output values. With each new layer output, a more abstract representation of the data is obtained [48]. The approach of sequentially transforming the data to higher-level concepts allows the learning of complex functions with little feature engineering, since the high-level concepts can be viewed as new features, which are extracted from the raw input. In Figure 2.1 we outline this neural network architecture.

When processing multi-dimensional data, representing spatially structured objects, with standard fully-connected neural networks, most of the spatial information is ignored [49], [50]. Convnets, on the other hand, provide a way of extracting local features by means of the *convolutional layer*, its core component. This Convnets aspect is very relevant for the task we aim to solve in this work. Namely, we want our model to learn and identify spatial patterns of meteorological variables in a given region. These patterns would aid in describing ahead of time the wind phenomena at either one or multiple locations.

2.1.1 Convolutional Layers

A convolutional layer is composed of many planes, known as *feature maps*, each of which contains a given number of units, the so called neurons. Each neuron is bounded to a small location within the planes of the previous layer by means of a set of weights (also called *filter bank* or *kernel*) and receives, as inputs, values from units pertaining that small location [47], [48][1]. Each unit of the convolutional layer may be described as the output of an operation between the units in the small location of the input plane and the weights in the filter bank. This operation is the convolution operation followed by the addition of a bias term, whose result is then passed through a non-linear function [47], [49].

Within each feature map, all units share the same set of weights and perform the same operation [49]. By performing the same operation in different localities of the input plane, with the same set

[1]The "planes" in the previous layer (which could be "planes" in the raw input data) are also known as *input feature maps*. The units in a single input feature map are convolved with the weights in the respective filter bank to produce an output feature map (referred to only as "feature map" above).

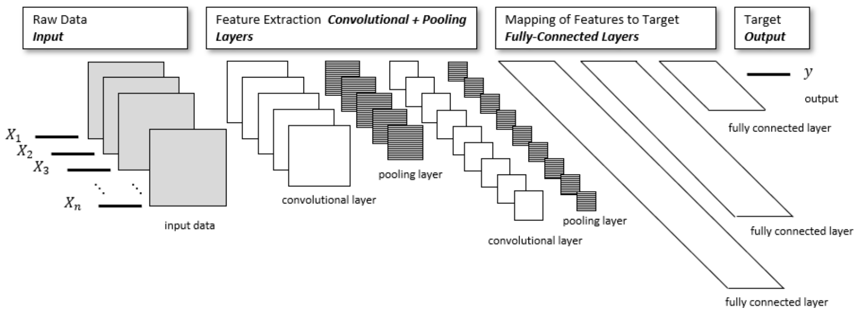


Figure 2.1: Example of Convolutional Neural Networks Architecture

of weights, a single feature map extracts the same features in different localities. This property is very relevant because it makes the Convnets invariant to translations in the input data [51]. Furthermore, by using multiple feature maps, the convolutional layer is able to extract different features from the same neighborhood location in the input plane: each feature map, with a particular set of weights, extracts a different feature for a given neighborhood location.

Feature Extraction in Convolutional Layers

Let X be a 2-D input array with M elements in the i-axis and M elements in the j axis, i.e., $X = (x_{i,j}) \in \mathbb{R}^{M \times M}$. The array X may be the values of an input plane preceding a given convolutional layer, where i and j are axes which retain the spatial disposition of values in the 2-D input array (e.g., height and width axes of an image). Furthermore, let W be a 2-D filter with N elements in the i-axis and N elements in the j-axis, i.e., $W = (w_{i,j}) \in \mathbb{R}^{N \times N}$. Then, the convolution operation of filter W over the 2-D input array X may be described as in Equation 2.1 [48].[2,3]

$$S_{i,j} = (X * W)_{i,j} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X_{i+m,j+n} W_{m,n} \tag{2.1}$$

The output $S_{i,j}$ in Equation (2.1) is related to one single unit in a convolutional layer's feature map. After performing the convolution operation for all units in the given feature map, we obtain $\mathbf{S} = (S_{i,j}) \in \mathbb{R}^{Q \times Q}$, where $Q \times Q$ is the number of units in the resulting feature map, and $Q = M - N + 1$ [52]. [4]

In deep learning, inputs to Convnets are usually 3-D arrays with height, width and depth axes [51]. In images, the height and width axes are related to the spatial disposition of pixels. The depth axis, on the other hand, is related to the number of color *channels* of an image. For this reason, the depth axis is also known as the channels axis. To account for a 3-D input array, the kernel of the convolutional layer is made volumetric (i.e., 3-D) with each channel of the kernel performing a convolution operation on a specific channel of the input array [24]. To form a feature map, the output of each convolution operation

[2]As explained in [48], Equation 2.1 shows the cross-correlation function, which is commonly adopted, in practice, to implement convolution operation in the context of Convnets. For further details, please refer to [48].

[3]Please note that, in Equation 2.1, X and W are square matrices. Nevertheless, the convolution operation does not require neither matrices to be square (please refer to [48] for a broader definition of the convolution operation).

[4]Here we assume a convolution operation with no zero-padding (e.g., no padding of zeros around feature map border) and with stride of size 1 (i.e., we move filters one pixel at time). For more details about calculations of output shapes for convolution operations, please refer to [50] and `http://cs231n.github.io/convolutional-networks/`.

on the channels is summed, and then a bias term is added (element-wise) to each resulting unit in the spatial grid. In Equation (2) we detail this operation, assuming the definition in (2.1) [52], that is,

$$\mathbf{O}_d = b_d + \sum_r \mathbf{S}_{d,r} \tag{2.2}$$

where $S_{d,r}$ is the convolution operation performed for the r-th input channel of the d-th feature map, b_d is the bias term for the d-th feature map, and O_d is the resulting feature map.

After adding the bias term, a non-linearity is applied element-wise to the resulting units in the feature map. This way, we are allowing the convolutional layer to map features which are non-linear transformations of the input. In deep learning, a usual non-linearity is the ReLU activation [53], defined as $g(z) = max(0, z)$. This activation function has lots of properties which are desirable for training deep neural networks and extracting features, such as the production of sparse representations.

2.1.2 Pooling Layers

The pooling layer, also known as subsampling layer [49], reduces the dimension of the feature map by compressing into one single value all the information of several units from a local neighborhood of a feature map. A pooling layer may take, for instance, the maximum value or the average value of units in the local neighborhood, what is respectively called max pooling and average pooling. Applying a pooling layer of size $P \times P$ to the d-th output feature map of the convolution layer O_d of size $Q \times Q$, results in

$$H_{d,i,j} = g_p\{Z_{d,i+p,j+p}\} \tag{2.3}$$

where p takes values in $\{1, \ldots, P\}$, P is the window size and g_p is the function which is applied to compress the information in the local neighborhood of size $P \times P$. As an example, for the max pooling, g_p is max_p. The output of the pooling operation, for a stride of size 1, is a map of size $Q - P + 1$. The procedure described above makes the pooling layer invariant to small translations and distortions in the data [47]–[50], [52].

2.1.3 Fully-Connected Layers

Fully-connected (FC) layers are usually the final layers in a Convnet, being stacked on top of modules of convolutional layers and pooling layers. Since the modules of convolutional layers and pooling layers extract features from data, FC layers act directly upon the extracted features. This way, FC layers map the extracted features of the data to the final target of the Convnet.

Each one of the neurons in a FC layer is connected to all neurons in the preceding layer, and each connection is associated with a trainable parameter (weight). Therefore, the neuron in the FC layer may be computed by the dot product of inputs and its respective weights, followed by the addition of a bias term. Let a neuron in a FC layer have n inputs, let f be an activation function, b be the bias term, and let x_i and w_i be the input and weight values respective to that neuron's i-th input, then the neuron output may be computed as $f((\sum_{i=1}^{n} w_i x_i) + b)$.

The configuration of the Convnet's output layer will depend on the task being solved by the network. For example, if we intend to solve multi-class classification with N output classes, the output layer may be designed to have N neurons with softmax activation function [48] (the softmax function computes a probability for each of the neurons; since each one of the N neurons is related to a class, the probability output of the softmax function denotes the probability that the input is associated to the given class). In the case of wind speed prediction for a single location site, the output layer will have only one neuron.

2.2 Improvements on Convolutional Neural Networks

2.2.1 Residual mappings

One of the most prominent deep learning architectures is the ResNet [29]. Proposed originally in 2015, the ResNet aimed to address the degradation problem of very deep neural networks. It addressed the degradation problem by adopting an architectural increment to neural networks, which would allow layers to perform residual mapping – a mapping constrained by original input features (see Eq. (2.4)). The hypothesis was that the neural network would be easier to optimize with layers that performed residual mappings: without residual mapping, the neural network would have to learn unconstrained mappings, and learning the unconstrained mappings would be harder.

Formally, the residual mapping may be formulated as follows:

$$\mathbf{y} = \mathbf{x} + \mathcal{F}(\mathbf{x}, W_i) \tag{2.4}$$

Where $\mathbf{y}$ and $\mathbf{x}$ are the output and input vectors of the layer(s) performing residual mapping. W_i is the parameter vector related to the mapping performed by the layer(s), and $\mathcal{F}(\mathbf{x}, W_i)$ is the function that will be learned by the residual mapping.

In Figure 2.2, we illustrate the addition of the residual connection (also known as *skip connection*) to a single layer. We show that the layer $\mathcal{H}(\mathbf{x}, W)$ has to learn its weights without any reference (unconstrained setting); by adding the residual connection, the layer $\mathcal{F}(\mathbf{x}, W)$ needs to learn a residual function (constrained setting).

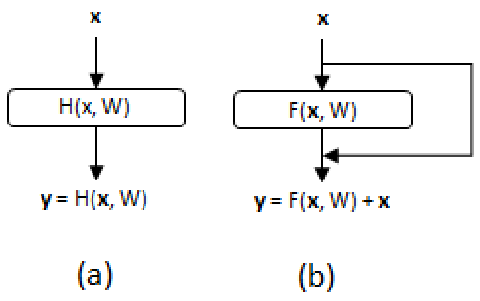


Figure 2.2: Identity mappings with addition of residual connections: (a) unconstrained mapping; (b) residual mapping.

2.2.2 Inception Modules

The Inception architecture [54] follows a great number of core neural network architectures used in computer vision, such as the LeNet [49], VGG [55] and CaffeNet [56] architectures. The Inception architecture is part of a family of models, which began with Inception v1 [54], also known as GoogLeNet, and was later updated and refined in three other versions, Inception v2 [57], Inception v3 [58] and Inception v4 [59].

In classic Convnets, convolutional filters process spatial relations and cross-channel information [28]. Inception architectures redesign the way spatial and cross-channel information are processed, aiming at higher efficiency: instead of having a single convolutional filter processing both spatial and cross-channel, Inception layers would use a series of convolutional filters to process each type of information separately. The layers that implement the Inception hypothesis are usually named *Inception modules*, which compose the Inception architectures. The Inception hypothesis is that cross-channel information is sufficiently different from spatial correlation, that they may be treated separately.

As pointed out in [28], there are several variations of inception modules. However, all share a same idea, which is to decouple cross-channel correlation and spatial correlation mappings of traditional convolution using 1x1 convolutions followed by 3x3 convolutions [28]. Figure 2.3 illustrates two versions of the inception modules – the canonical inception module and the simplified inception module.

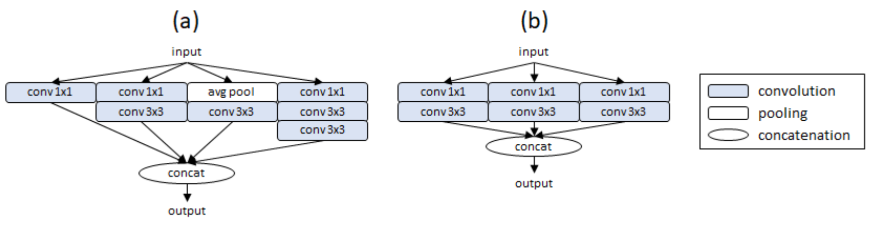


Figure 2.3: Inception modules: (a) canonical version; (b) simplified version. Reproduced from [28]

2.2.3 U-Net

Convnets [60] were developed to solve image classification task where output was a class label, (e.g., CIFAR, ImageNet). However, many other visual tasks require pixel-by-pixel classification with very limited data (e.g., less than thousands of samples). Fully-convolutional neural networks [36], [61] are a type of architecture which was originally designed to solve pixel-by-pixel classification.

One of the most prominent and popular fully-convolutional neural network architecture is the U-Net [36], which performed high-accuracy image segmentation with training on very few examples. Introduced in 2015, the U-Net is a special kind of Convnet, which takes its name from the shape of the neural network architecture.

The U-Net presents a contracting path – which is based on convolutional and subsampling layers – and a expansive path – which is based on transposed convolution and upsampling layers[5]. The contracting path follows the same ideas as traditional convolutional neural networks, and aims to extract features from raw data by means of convolution and pooling operations. The expansive path aims to reproduce high-resolution map from those extracted features by applying tranposed convolutional. The U-Net architecture is trained as any other neural network architecture – with stochastic gradient descent algorithm and backpropagation.

[5]The transposed convolution – also known as deconvolution – performs the inverse operation of the traditional convolution. It expands an input map to a higher-sized output map by applying a filter that associates each of its input units to multiple output units. For more details, please refer to [50]

Chapter 3

Review

3.1 Time Series Forecasting

The prediction of temporal phenomena is a relevant subject for multiple areas. For example, the forecast of growth domestic product (GDP) is important to measure future economic expectations of countries (what may drive future investments in that country). Likewise, demand forecasts are essential for production and operational planning of retail companies. In power systems, the forecast of load demand supports the definition of which generation units should produce power to attend the demand. Many other areas also rely on time series forecasting.

Formally, time series are a collection of observations made sequentially in time [62]. Time series are stochastic processes – both influenced by randomness and by past values.

3.2 Spatio-Temporal Forecasting

Spatio-temporal processes are those that unfold in time and space [63]. Examples of spatio-temporal include the following: meteorological processes (such as air pollution [64], precipitation, wind phenomena and others), transportation processes (such as traffic flow, river flow and others), health-related issues (such as outbreaks of ebola [65], or weight of babies in a region [66]) and video processes (which consists of the variation of pixels on the frames of videos). Formally, spatial-temporal processes may be defined as follows: let $\mathcal{T} \subset \mathbb{R}$ be a time domain, $\mathcal{D} \subset \mathbb{R}^d$ be a spatial domain, and let $d = 2$, a spatio-temporal process $X(\mathrm{s}, t)$ is a random variable that can take a series of outcome values at any location $\mathrm{s} \in \mathcal{D}$ and instant in time $t \in \mathcal{T}$, that is:

$$X(\mathbf{s},\ t) :\ \mathbf{s}\ \in\ \mathcal{D},\ t\ \in\ \mathcal{T} \tag{3.1}$$

There are many contexts in which spatio-temporal forecasting may be needed. Often times, one needs to use spatio-temporal information to infer a single value or multiple values. One example for the former is the task of action recognition in videos, in which frames of video are input to a model, which outputs a value that identifies the action being performed in the video. One example for the latter includes frame-by-frame prediction – a task in which the model predicts the pixel values for a given temporal horizon.

3.3 Methods for Spatio-Temporal Prediction

Many different approaches to predicting spatio-temporal phenomena have been proposed. These range from statistical methods to recent deep learning methods. Below, we list the main techniques used to solve spatio-temporal prediction.

3.3.1 Statistical Methods

In 1999, Cressie and Huang [67] adopted the classical geostatistical method of kriging [68] to model spatio-temporal wind phenomena over a region in the Pacific Ocean. (Kriging is an interpolation method originally developed in the context of geostatistics; it produces a model which gives more weight to points nearest to the prediction point [69].) The article expanded the class of covariance functions which could be adopted in the context of spatio-temporal kriging. The drawback of kriging methods is that many assumptions need to hold true in order for the model to perform well. When assumptions do not hold, the model's performance may deteriorate. In fact, as pointed out in [67], separable models (based on kriging) were chosen due to convenience, and not due to good fit on data. In spite of this, kriging-based methods are still used today. For example, Martinez et al. [70] expanded Median Polish Kriging (MPK) to consider four dimensions (time, latitude, longitude and altitude) in spatio-temporal prediction.

Another statistical approach to spatio-temporal prediction is the one that combines geostatistics and additive modeling – a class of models known as *geoadditive models* [66]. This class of models considers the spatio-temporal process as a sum of space effect and temporal effect [71]. In such models, as pointed out in [72], temporal trend may be modeled via random walk, autoregression, or P-splines. Spatial effect may be modeled via Markov random fields or 2D P-splines. Fahrmeir et al. [72] adopt additive approach to spatio-temporal modeling and infer parameters of the space-time models with bayesian inference. Paez and Gamerman [73] also adopt bayesian inference to estimate additive models for predicting pollutant concentration in Rio de Janeiro. In their work, they compare different specifications for modeling temporal component and spatial component, which are assumed to be separable.

Many other statistical methods, which build upon the classical approaches of kriging, geoadditive models and ANOVA interaction [71], [74], have been developed and are out of the scope of this thesis. For a comprehensive review on statistical methods for spatio-temporal prediction, refer to Diggle and Ribeiro [75], Fahrmeir et al. [72] and González et al. [65].

3.3.2 Machine Learning Methods

When dealing with spatio-temporal prediction problem, we usually have observations of the same variables for multiple location sites; this enables us to build different features and algorithmic designs for predictive purposes. Consequently, in the realm of traditional machine learning (ML) models, works addressing spatio-temporal prediction consider different types of designs. The designs often differ on how features of each site are exposed to the learning algorithm.

For example, Heinermann and Kramer [25] consider past observations of neighboring sites as features to their ML models – thus, adding dimensionality to the input space. Differently, Persson et al. [19] develop a model that adds information for a given time of a given location site (e.g., past observations, latitude, longitude, zenith angle, and other features) as examples on the input data – thus, adding rows to the input data.

The works that adopt ML methods deal with the paradigm of building a algorithm that learns the relation between input pattern and target data. Methods for developing such an algorithm are various, and may include a single Support Vector Regression (SVR) (as in [76]), single neural network (as in [77]), ensemble of predictors (such as the ones developed in [25] and [19]), and many others.

3.3.3 Deep Learning Methods

In this section, we review deep learning techniques used specifically for video frame prediction, which is a spatio-temporal problem. Consequently, the techniques used for video processing may be used as a basis for spatio-temporal wind speed prediction (in a setting similar to the one of frame-by-frame prediction). Reviewing such techniques is thus useful to understand state-of-the-art of deep learning for spatio-temporal prediction.

Video frame prediction is a challenging computer vision task, which requires one to develop a model able to learn not only the structure of images but also the motion related to them [78]. To solve video frame prediction, a wide range of deep learning techniques may be adopted. These may include, for example: recurrent neural networks (RNNs) [34], fully convolutional neural networks (FCNs) [61], residual connections [79], adversarial training [80], attention mechanisms, feature fusion, and many others.

In early years, solutions with recurrent neural networks (e.g., LSTM) were the ones proposed to solve video frame prediction. The approaches were inspired by sequence-to-sequence modeling – originally used for natural languange modeling. In 2014, Ranzato et al. [81] proposed the use of RNNs originally developed for language modeling to solve video frame prediction. The work transformed the pixel space into a quantized space encoded via k-means to be able to achieve good results with the proposed models. In 2015, Srivastava et al. [82] used LSTM encoder-decoder framework to predict video frames as a sequential problem. The work encoded input frames either as image patches or as features extracted from last layers of state-of-the-art image recognition models. One of the proposed models in [82] was the LSTM Future Predictor Model, an Encoder-Decoder model where the Encoder runs through a sequence of frames to learn a representation of it and the Decoder produces a target sequence (future video frames) from the representation.

Following LSTM-based models, new architectures began to leverage convolutional operations to process the spatial information. Shi et al. [30] proposed a model, known as ConvLSTM, which extended recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) cells [34] to perform convolutional operation. The ConvLSTM model was originally developed to solve precipitation nowcasting. However, the method started a new line of approach to video frame prediction, and, until today, the ConvLSTM is used as benchmark for video frame prediction (e.g., see [78]). ConvLSTM is also used as backbone layer for new architectures used for video frame prediction (e.g., see [32]).

Finally, recent works on video frame prediction proposed to disentangle content and motion in videos (e.g., see [32], [33], [83], [84]), and to model the problem using Encoder-Decoder architectures that combine FCNs, ConvLSTM or LSTMs (separately or altogether). Additionally, the learning framework proposed in most recent works is that of adversarial training [80] (e.g., see [32], [33], [84]). Some known models, which have become a reference in present time [83], are MCNet [32] and DrNet [33].

Villegas et al. [32] is one of the works that adopts the disentanglement of content and motion in videos to address video frame prediction. The work uses an Encoder-Decoder framework, and adopts one encoder for motion in video and another encoder for video content. The motion encoder is implemented with a FCN followed by a ConvLSTM, whereas the content encoder is implemented with a standard Convnet, specialized on extracting features from a single frame [32]. Content and motion features obtained by the encoders are then fused and fed to a decoder (a deconvolution network [85]), which finally predicts the future frames.

Chapter 4

Single-Site Spatio-Temporal Wind Speed Forecasting

4.1 Methodology

The use of spatial information has been shown to enhance the predictability of wind [17]–[19]. Thus, adopting more advanced techniques, which process spatio-temporal information and predict wind speed, may lead to better tools for power systems operations. Convnets is a powerful technique for processing spatial data. Here, we develop architectures based on convolutional and fully-convolutional neural networks to leverage spatio-temporal information of multiple explanatory variables with the goal of predicting wind speed in a single location.

Let there be m explanatory variables, which are spatio-temporal processes. Then, each explanatory variable is a random variable which may take values in location $s \in \mathcal{S}$ and time $t \in \mathcal{T}$, where $\mathcal{S}$ is the spatial domain (in our case, $\mathcal{S} \subset \mathbb{R}^2$) and $\mathcal{T} \subset \mathbb{R}$ is the time domain. In the context of wind prediction, potential explanatory variables are temperature, pressure, presence of rain, the components u and v of wind, and others.

To process spatio-temporal variables with Convnets, we propose assigning each explanatory variable to one of the input channels of the Convnet. Considering this set-up, for a given input channel, the height and width axis of the Convnets are going to represent latitude and longitude, respectively, and each value at a given position in the input channel will be the value of the associated explanatory variable at the respective latitude-longitude pair. Moreover, all values in a given input channel will be related to the same time step. For clarity, we write input variable X_1 as $X_{1,t}$. One could add information of a given variable at different time steps by including lags of the spatio-temporal variables in the input channels. Additionally, one could consider multiple explanatory variables in the input channels. Let $l \in \{0, \ldots, L\}$ denote lagged steps and let $m \in \{1, \ldots, M\}$ denote explanatory variables, the multivariate input to the Convnets, $\mathbf{X}_t$, may be written as $\mathbf{X}_t = \{X_{m,t-l}\}$.

Considering this setting, the objective is to use information of the multivariate 3D input tensor, $\mathbf{X}_t$, to predict a single value y_{t+1}, which denotes the wind speed one-step ahead of time for a single site (i.e., for a single latitude-longitude coordinate). We propose the use of neural networks to perform such a mapping. Specifically, we adopt deep learning techniques to design neural network architectures which extract high-level features from the multi-dimensional tensors and map the extracted features into the output variable in $\mathbb{R}$. Below, we give details of the proposed architectures. The backbones of the proposed architecture are the U-Net and Convnet architectures.

4.1.1 U-Convolutional Models

The U-Net maps an image to a segmentation mask, solving pixel-by-pixel classification. It is a powerful technique for image segmentation. More recently, the U-Net was adopted to predict future frames of videos. Specifically, in [37], an adversarial framework is adopted, and the generator, which produces the prediction for the next frame, is an U-Net. As pointed out in [37], the U-Net is selected as the predictor due to its good performance on image-to-image translation. The inputs to the U-Net, in [37], are frames in present and past times; the output is the frame at one step ahead of present time.

In the proposed architecture, which will be referred to as *U-Conv model*, the U-Net is used differently, since its final layer is not associated with a target tensor (e.g., segmentation mask, image or video frame). Instead, the final layer of the U-Net is connected to a Convnet, which outputs a single value (i.e., wind speed at one-step ahead of time). Consequently, the U-Net's final layer is a feature map that is obtained as result of the optimization process that minimizes weights considering only a single output variable. With this setting, we design the U-Net to map the 3D input tensor, $\mathbf{X}_t$, to a 3D output tensor that has the same dimensions of the input tensor. This output tensor is connected to a Convnet-based architecture, which produces the final single-value output for our problem (the wind speed prediction).

We hypothesize that U-Net would act as a more enhanced feature synthesizor than traditional convolutional operation applied sequentially (deep Convnets), which would facilitate Convnet processing and final prediction. This hypothesis considers the fact that, different from mapping with sequential convolutional layers, U-Net combines features from layers at different locations in the architecture, providing high-level features that would be more refined that those in traditional deep Convnets.

Input design

In our U-Conv model design for wind speed prediction, we consider u-component of wind, v-component of wind and temperature at times t, $t-1$ and $t-2$ as input variables. Let these three variables be, respectively, U, V and K. We may write the input tensor as $\mathbf{X}_t = \{U_{t-l}, V_{t-l}, K_{t-l}\},\ l \in \{0, 1, 2\}$. Hence, $\mathbf{X}_t$ totals nine input variables. Each variable is assigned to one input channel. The output variable is the wind speed at time $t+1$, i.e., $y^{s_1}_{t+1}$. We have, thus, an input array of shape $n_{latitude} \times n_{longitude} \times 9$, which accounts, respectively, for the number of latitude points (height axis), number of longitude points (width axis) and the three explanatory variables at times t, $t-1$ and $t-2$ (channels axis).

Figure 4.1 illustrates the solution we give to wind speed spatio-temporal prediction with the U-Conv model. In (A), we illustrate the variables that compose the input tensor: variables U, V and K at times t, $t-1$ and $t-2$. In (B) we detail how the input tensor to the Convnet is structured: each 2D tensor in (A) is concatenated in the channel axis to form the input tensor of dimensions $n_{latitude} \times n_{longitude} \times 9$. Illustration (C) shows the U-Conv model overall structure: a model that is composed of two parts – a U-Net model and a Convnet-based model. The U-Conv model outputs a single value (one-step-ahead wind speed). In (D), we illustrate the target variable, y_{t+1}.

U-Net design

The design of the U-Net part of our U-Conv model follows the structure proposed in [36]. In our case, the spatial resolution of input and output is smaller than the one in [36]: the datasets we use have spatial dimension of 10×10 and 9×9 (accounting for specific regions in Brazil), whereas in [36], spatial dimension is 512×512 (accounting for medical images with 512×512 pixel resolution). In [36], the contracting path is composed of 3x3 convolutions and 2x2 maxpooling with stride of 2. Considering the reduced spatial information in our input setting, we adopt 2x2 convolutions and 2x2 maxpooling with stride of 1 in our contracting path. Moreover, despite following the same idea of doubling the number of filters at each 2x2 subsampling, our design also differs in terms of the number of filters at convolutional

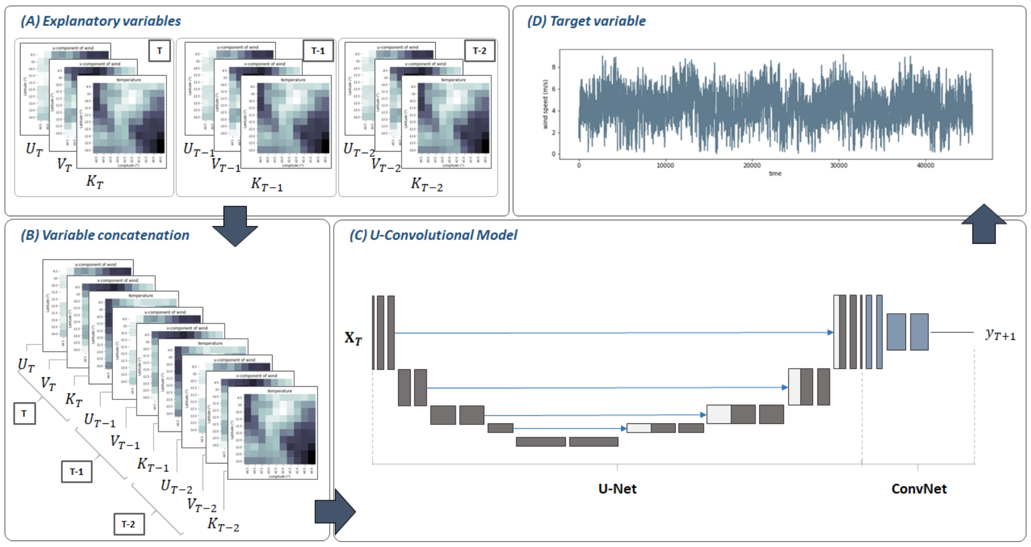


Figure 4.1: Illustration of Input-Output Mapping with U-Conv model

layers: considering that the problem is an hourly forecasting problem, we chose 24 filters at U-Net's first convolutional model, attempting to have filters for different hourly patterns in one day. (The layers that follow the first layer are multiples of 24.) The expansive path follows the idea of the original work: upsampling of feature maps followed by convolutional layers that halves the number of features, and concatenation of convolutional layers followed by another convolutional layer. The expansion path is symmetric to the contracting path, in terms of number of filters in the convolutional layers.

Convnet design

To build the Convnet part of the model, we took as a basis the idea of combining convolution and subsampling layers, increasing the number of feature maps (representational power) as the size of the feature maps (resolution) decreases [49]. Considering this setting, we created a module comprising two convolutional layers followed by one pooling layer. We combined two modules of two convolutional layers followed by one of pooling – a design which might be found, for example, in the first layers of the VGG-19 model [29], [55].

The Convnet design ended up with 9 layers. The first two layers were convolutional layers with kernels of size 2×2; the third layer was as max pooling layer of size 2×2. The fourth and fifth layers were convolutional layers with kernels of size 2×2. The sixth layer was a max pooling layer of size 2×2. The first six layers are the ones that perform feature extraction. After the sixth layer, we reshape the data to be a one-dimensional vector. The one-dimensional vector is then fed to a fully-connected layer. The seventh layer is followed by a dropout layer, which we adopt to prevent overfitting our data. The last layer is the output layer comprised of a single neuron with linear activation.[1]

We propose a second version of the U-Conv model, which we name *residual U-Conv model*, where we add identity mappings to the Convnet part of the model. We adopt the identity mapping in every convolutional module of the Convnet (see Figure 4.2).

[1]The number of filters in the convolutional layers and the number of neurons in the fully-connected layer may vary in the experiments. Nevertheless, all U-Conv models follow the idea that the number of filters in the first convolutional module is lower than that in the second convolutional module.

The full architectures of the U-Conv and residual U-Conv models are illustrated in Figures 4.2. In the Figures 4.2(a) and 4.2(b) we refer the filters in the first and second convolutional modules as *f1* and *f2*, respectively. Fully-connected layers are referred to as *fc*, and output layer is named as *out*. We specify the number of filters and regularization in Section 4.3. We apply zero padding when needed in both architectures.

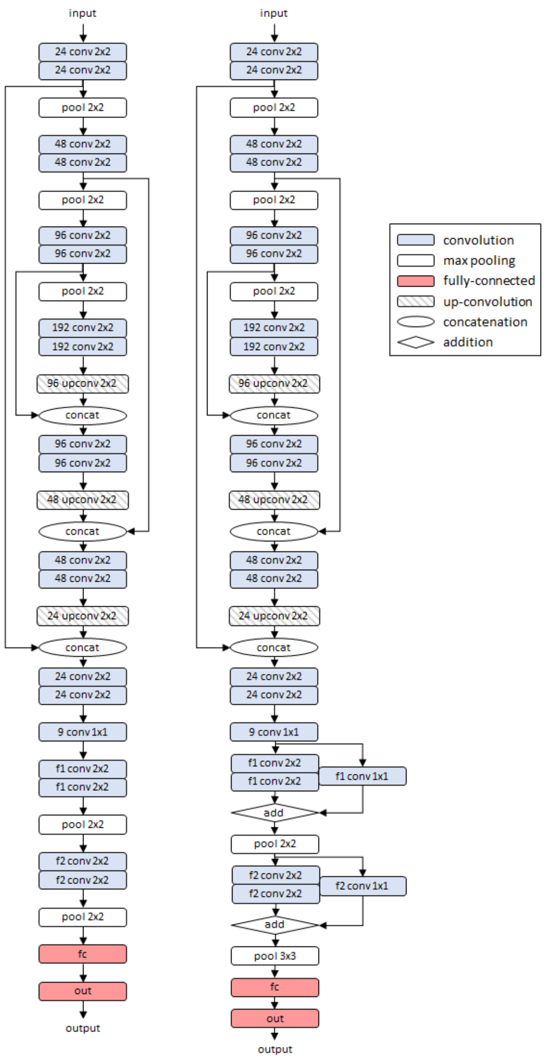


Figure 4.2: (left) U-Convolutional model; (right) Residual U-Convolutional model

4.1.2 Convnet and Inception Models

Following the idea of using explanatory variables in the channel axis of the 3D input tensor, it was also possible to develop other Convnet models to solve spatio-temporal wind speed forecasting. Specifically, we develop standard Convnet, Convnet with identity mappings, Convnet with inception modules and Convnet with inception modules and identity mappings. These architectures (which we refer to as *Convnet*, *Residual Convnet*, *Inception*, and *Residual Inception*, respectively) are illustrated in Figures 4.3(a), 4.3(b), 4.3(c), and 4.3(d), respectively. By investigating other Convnet architectures, we may be able to identify which other types of architectures would be fit to solve spatio-temporal wind speed forecasting.

In [29], the residual mapping is adopted to every few layers; for example, the 34-layer residual network has a residual mapping that encompasses two consecutive layers. In our work, the architectures have identity mapping for every convolutional (or inception) module. This is illustrated in Figure 4.3(b) (and Figure 4.3(d)).

Concerning the inception-based models, we create an architecture where we substitute convolutional layers in Convnets and Residual Convnets for inception modules (see Figures 4.3(a) and (b) versus Figures 4.3(c) and (d)). The idea is to understand if switching standard convolution for convolutions that split cross-correlation from spatial correlation improves spatio-temporal wind forecasting.

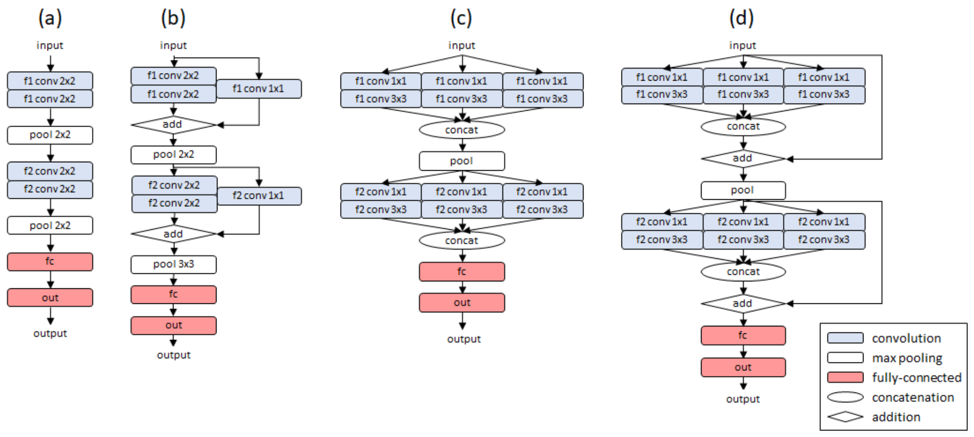


Figure 4.3: (a) Convnet, (b) Residual Convnet, (c) Inception, (d) Residual Inception

4.1.3 Training

To build the proposed models, we need to define a loss function, which provides a measure of how bad the model is. The loss function we select for our training is the mean squared error, a popular loss function for regression tasks (see Equation 4.1). To estimate the models, we also need to define the learning algorithm that minimizes the loss function we selected. To search weights in the Convnets' parametric space, such that the loss function of our task is minimized, we select the Adam algorithm [86]. We select Adam because it combines properties from Adagrad [87] (dealing with sparse gradients) and RMSprop (dealing with non-stationary objectives), and because Adam is usually considered robust to hyperparameters' choices [48].

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.1}$$

Where y_i is the true value for the i-th example and $\hat{y}_i$ is the predicted value for the i-th example and N is the number of examples.

4.2 Empirical Evaluation

4.2.1 Datasets

In this work, we use two samples from the Climate Forecast System Reanalysis (CFSR) dataset [88]. The dataset has been obtained at the Research Data Archive (RDA) website. Both samples from the CFSR dataset contain data of u-component of wind, v-component of wind and temperature for latitude-longitude pairs. The first sample of data considers latitude-longitude pairs where latitudes range from -9.5° to -14.0° at 0.5° spatial resolution and longitudes range from -44.5° to -40.0° at 0.5° spatial resolution. These coordinates are related to an area in Brazil comprising, approximately, the entire state of Bahia. For this purpose, we refer to this first sample of the CFSR dataset as *Bahia Wind*.

For the other sample, the data is acquired for latitude-longitude pairs where latitudes range from -4.191° to -5.826° at, approximately, 0.204° spatial resolution, and longitudes range from -36.000° to -37.841° at, approximately, 0.204° spatial resolution. These coordinates are related to an area in Brazil comprising a region of the State of Rio Grande do Norte, one of the regions with highest incidence of constant wind in Brazil. We refer to this second sample of the CFSR dataset as *Rio Grande do Norte Wind*.

The data we acquire for each variable at each coordinate consists of an hourly time series which dates from 2011-04-01 00:00 until 2017-01-01 0:00 for Bahia Wind and from 2011-04-01 00:00 until 2018-03-01 00:00 for Rio Grande do Norte Wind.

At the end, we have 50448 hourly data for each one of the 10 x 10 locations (latitude-longitude pair) of each of the 3 variables we consider for Bahia Wind, and we have 60648 hourly data for each one of the 9 x 9 locations of each one of the 3 variables we consider for Rio Grande do Norte Wind.

In Figures 4.4 and 4.5, we show the explanatory variables heatmap of the time step of $t = 0$ hours for Bahia Wind and Rio Grande do Norte Wind, respectively.

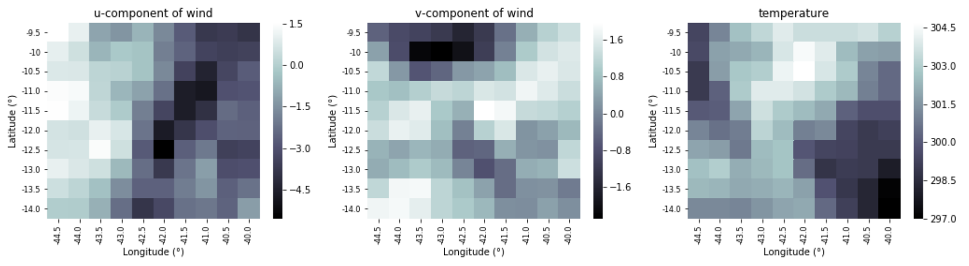


Figure 4.4: Heatmaps of Meteorological Variables for Time Step $t = 0$ hours of Bahia Wind

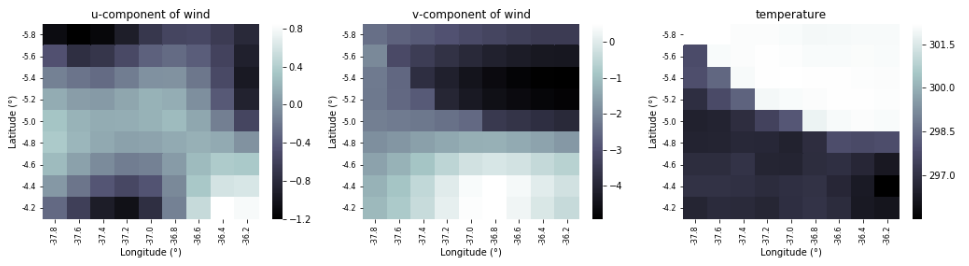


Figure 4.5: Heatmaps of Meteorological Variables for Time Step $t = 0$ hours of Rio Grande do Norte Wind

4.2.2 Experimental Setup

We used the proposed models to predict wind speed one hour ahead of time for a single coordinate within the regions of study. For Bahia Wind, target wind speed was measured at latitude of -12.0° and longitude of -42.0°; for Rio Grande do Norte Wind, target wind speed was measured at latitude -5.0° and longitude -37.0°. In our design – following the ideas of video frame prediction framework –, we consider lagged variables in the input channels of the architectures. To account for the lagged variables (one and two-steps back in time) as features in our model, we had to reduce both datasets: Bahia Wind from 50448 to 50445 and Rio Grande do Norte Wind from 60648 to 60645. This way, for Bahia Wind, input data had a shape of $50445 \times 10 \times 10 \times 9$ and for Rio Grande do Norte Wind, input data had a shape of $60645 \times 9 \times 9 \times 9$.

We split both datasets into three sets of data: training, validation and test. The training set contained samples we used to estimate the weights of our Convnets, and included 90% of the dataset's samples (accounting for 45401 samples in Bahia Wind and 54581 in Rio Grande do Norte Wind). The validation set was used to check how well the models we trained predicted data that were not used for training. The validation set contained 5% of the dataset's samples (2522 samples in Bahia Wind and 3032 samples in Rio Grande do Norte Wind). The training and validation sets were used for model selection: we tuned the hyperparameters of our models and checked the accuracies in training and validation sets. The test set totalized the remaining 5% samples (2522 samples for Bahia Wind and 3032 samples for Rio Grande do Norte Wind). After selecting the best models based on training and validation set accuracies, we use the test set to compute the accuracies which would be obtained with unseen data.

4.2.3 Accuracy Measures

The accuracy measure we adopted to evaluate our models was the mean absolute error (MAE), a scale-dependent metric that is very common in the forecasting literature [89], in general, and in wind speed prediction (e.g., [7], [90], [91]). Additionally, MAE has been used as the main accuracy measure in a competition of wind speed forecasting [2], what motivated us to select it as our main metric. We did not select percentage-related measures, such as the mean absolute percentage error (MAPE) to evaluate our results, because the wind speed data presents values smaller than 1. In such cases, the percentage-related measures, which divide the error by the observed data, may be distorted.

Let y_i and $\hat{y}_i$ be, respectively, the true and predicted values of the i-th example, and let N be the total number of examples, then MAE is described as follows:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{4.2}$$

4.3 Results and Discussion

We develop our work in Python and use *Keras* [92] to create and train neural network architectures. For each architecture, we tested a different set of selected hyperparameter configurations (see Appendix). We executed training five times for each hyperparameter configuration, and we selected the best one as the configuration that produced lowest average MAE on validation set. Following this, we predicted the test set with the (five) models from the best configuration. This evaluation allows us to measure the mean and variance of test set error for each architecture, what may indicate which architectures are more stable in

[2]http://eem2017.com/program/forecast-competition

the context of the proposed spatio-temporal forecasting setting and which are more prone to be affected by weight initialization.

After discussing about the Convnet architectures, we compare the best models of each architecture against more traditional benchmark models: fully-connected neural networks (with spatio-temporal inputs) (ANN), ARIMA model [9], BATS model [46] and Naïve model. The last three methods were developed in a univariate forecasting setting, and were implemented with the *forecast* package in R [93]. In the same section, we provide Diebold-Mariano test statistics for the null hypothesis that a given model is less accurate than the other model.

4.3.1 Comparing Architectures

In Table 4.1, we evaluate the accuracy measures of the proposed architectures (U-Conv model and Residual U-Conv model), improved Convnets (Residual Inception, Inception, and Residual Convnet), and standard Convnet. We provide the mean and standard deviation of test set error for each architecture.

The models trained to predict Rio Grande do Norte (RN) Wind performed better than those trained to predict Bahia Wind: errors for the former are lower than for the latter, and standard deviations are also overall lower for RN Wind. This might have been expected, since it is known that the wind in Rio Grande do Norte region is more regular than the wind process of Bahia Wind. (This observation may also be noted from Figures 4.4 and 4.5 – it appears that v-component of wind and temperature are strongly related in Dataset RN Wind, whereas such a relation appears to be lacking for Bahia Wind.)

The results indicate that the U-Convolutional models outperform Residual Convnet and Residual Inception on predicting Bahia dataset. On the other hand, the U-Convolutional models are outperformed by the Residual Convnet and the Residual Inception on predicting RN dataset. The hypothesis we make is that the U-Net part of the U-Convolutional models is able to synthesize wind turbulence (reflected mainly in u- and v-wind components) into features better than the other architectures. Since RN dataset presents more steady wind, with less turbulence, this kind of property would not be much needed in its modeling – and thus Residual Convnet and Residual Inception perform better than U-Convolutional models for this dataset. Such hypothesis should be further investigated in other experimental settings, however.

Furthermore, we note, for Bahia Wind dataset, that Inception models, which creates separated information flows within Convnets, peform better than standard Convnet models. It appears that architectures that are able to process channel information in a more specialized way present better results in this dataset (U-Convolutional models synthesize input data into a 3D tensor of same size and perform better than all architectures in Bahia Wind dataset). This might indicate that, in order to model turbulent wind, one should prioritize using models with separable convolutions or other type of operation (such as U-Net processing).

For RN Wind dataset, we note that architectures with identity mappings are the dominant ones: Residual Convnet is the one with lowest average MAE, followed by Residual Inception and Residual U-Convolutional model. It appears that identity mappings and simple operations in architectural design are the properties that stand out in modeling data from RN Wind with Convnet framework. In all cases, standard Convnet is outperformed by the other architectures.

4.3.2 Comparing Best Models

In Table 4.2, we show the accuracy measures of the best models from the Convnet architectures, ANN, ARIMA model, BATS model and Naïve model. We included the spatio-temporal explanatory variables as input to the ANN models, i.e., each input channel data (as illustrated in Figure 4.1(B)) was transformed into a one dimensional array (i.e., the matrices were flattened); all were concatenated into a long one

Table 4.1: Results on Test Set: Average MAE (MAE Standard Deviation)

Model	Bahia Wind	Model	RN Wind
U-Conv	0.1906 (0.0023)	ConvRes	0.1607 (0.0008)
U-ConvRes	0.1942 (0.0021)	InceptionRes	0.1625 (0.0003)
InceptionRes	0.1956 (0.0036)	U-ConvRes	0.1643 (0.0014)
Inception	0.1966 (0.0015)	U-Conv	0.1668 (0.0021)
ConvRes	0.2018 (0.0037)	Inception	0.1718 (0.0022)
Convnet	0.2086 (0.0026)	Convnet	0.1802 (0.0018)

dimensional array, which served as input to the ANN models. We normalized the data for ANN modeling the same way we did for Convnet architectures (by scaling input data to zero mean and one variance considering data from training). Furthermore, we applied Principal Component Analysis (PCA) [94] to decorrelate spatio-temporal input data, what would facilitate ANN weight optimization [95][3]. The BATS model was developed considering seasonality of 24 hours (daily seasonality). The ARIMA model was implemented with auto.arima, resulting in models ARIMA(1, 1, 5) and ARIMA(2, 1, 2) for Bahia and RN Wind datasets, respectively.

The results indicate that the proposed U-Convolutional model presented the best one-step-ahead forecasts for the Bahia Wind dataset. However, the Diebold-Mariano test (in Figure 4.6) indicates that one cannot say that the best Residual Inception and best Residual U-Convolutional models are less accurate than the best U-Convolutional model at $\alpha = 5\%$. Nevertheless, the test indicates that all other models are less accurate than the three aforementioned models (i.e., U-Convolutional, Residual Inception and Residual U-Convolutional) at the same confidence level. One interesting observation is that the p-value for Diebold-Mariano test indicates that statistically, it can not be said that BATS and ARIMA models are less accurate than best ANN model for the Bahia Wind. It remains to be investigated the reason why ANN outperforms BATS and ARIMA model for RN Wind and does not outperform both models for Bahia dataset (one hypothesis could be that ANN learns noise in Bahia dataset – considering that Bahia wind is more turbulent than RN wind).

The Residual Convnet was the model that provided best forecast for the RN Wind dataset. The p-value of Diebold-Mariano test (in Figure 4.7) indicates that Residual Convnet provides better forecasts than every other model for RN Wind at $\alpha = 5\%$. The results indicate that it is not possible to say that Residual U-Convolutional and U-Convolutional models have lower accuracies than Residual Inception model, which appears as second best model. However, all three of these models (i.e., Residual Inception, U-Convolutional and Residual U-Convolutional) have higher accuracies than the other models we tested (with exception of the Residual Convnet). Moreover, we note that, for both datasets, the proposed and the improved Convnets outperformed standard Convnets, ANN, BATS, ARIMA and Naïve models in predicting wind speed for a single-site.

4.3.3 Discussion

Our empirical results indicate that the proposed approach is promising, providing accuracy measures that were considerably lower than that of standard Convnet model and fully-connected neural networks adapted to single-site forecasting with multiple spatio-temporal explanatory variables as input, and of traditional univariate models such as BATS, ARIMA and Naïve models. By outperforming univariate

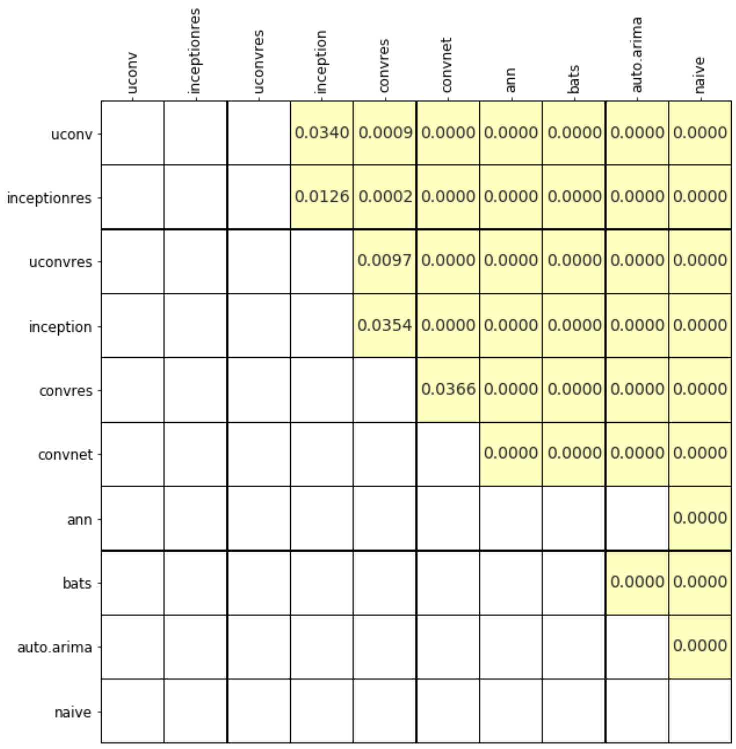
[3]We chose to keep all principal components as input to the ANN, so that the ANN model would process the same amount of information as the Convnet models

Table 4.2: MAE Results on Test Set

Model	Bahia Wind	Model	RN Wind
U-Conv	0.1873	ConvRes	0.1595
InceptionRes	0.1908	InceptionRes	0.1620
U-ConvRes	0.1922	U-ConvRes	0.1627
Inception	0.1946	U-Conv	0.1633
ConvRes	0.1984	Inception	0.1687
Convnet	0.2055	ANN+PCA	0.1745
ANN+PCA	0.2300	Convnet	0.1787
BATS	0.2366	BATS	0.2001
ARIMA	0.2530	ARIMA	0.2774
Naïve	0.3146	Naïve	0.3308

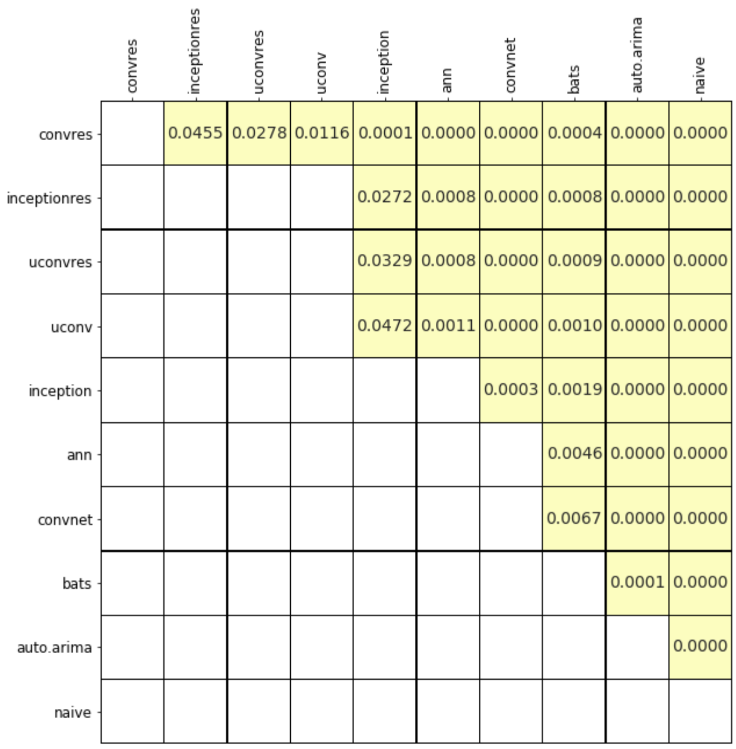
models, we corroborate the idea that the inclusion of spatio-temporal information enhances wind speed prediction (in our case, for single-site prediction; in [96], for multi-site prediction).

The proposed approach could be applied in a real-world setting where the information (of the meteorological variables) is updated hourly and the models are executed hourly to predict one-hour ahead of time. Real power system operation may require, however, that the models provide hourly predictions 24 hours ahead of time. This is the case of the Brazilian Power System, where the centralized operation considers hourly forecasts 24 hours ahead of time to plan daily operation. To apply the proposed approach under such circumstance, one could create 24 models, where each model would use information available at time t to predict wind speed for a given step-ahead (i.e., one model would be developed to predict wind speed at time $t+1$, a second model would be developed predict wind speed at time $t+2$, and so on). Despite the implementation drawback for 24 hours ahead of time, the proposition and study of models for one-hour ahead of time prediction are frequent in the literature, what indicates the relevance of the study of such models [18], [97], [98].



	uconv	inceptionres	uconvres	inception	convres	convnet	ann	bats	auto.arima	naive
uconv				0.0340	0.0009	0.0000	0.0000	0.0000	0.0000	0.0000
inceptionres				0.0126	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000
uconvres					0.0097	0.0000	0.0000	0.0000	0.0000	0.0000
inception					0.0354	0.0000	0.0000	0.0000	0.0000	0.0000
convres						0.0366	0.0000	0.0000	0.0000	0.0000
convnet							0.0000	0.0000	0.0000	0.0000
ann										0.0000
bats									0.0000	0.0000
auto.arima										0.0000
naive										

Figure 4.6: Diebold-Mariano Test for Bahia Wind (p-values) (The figure should be read the following way: the model on a given row is more accurate than the model on the given column if a number (p-value less than 5%) appears in the row-column cell; otherwise, one cannot say that the model on the given row is more accurate than the model on the given column.)



	convres	inceptionres	uconvres	uconv	inception	ann	convnet	bats	auto.arima	naive
convres		0.0455	0.0278	0.0116	0.0001	0.0000	0.0000	0.0004	0.0000	0.0000
inceptionres					0.0272	0.0008	0.0000	0.0008	0.0000	0.0000
uconvres					0.0329	0.0008	0.0000	0.0009	0.0000	0.0000
uconv					0.0472	0.0011	0.0000	0.0010	0.0000	0.0000
inception							0.0003	0.0019	0.0000	0.0000
ann								0.0046	0.0000	0.0000
convnet								0.0067	0.0000	0.0000
bats									0.0001	0.0000
auto.arima										0.0000
naive										

Figure 4.7: Diebold-Mariano Test for RN Wind (p-values) (The figure should be read the following way: the model on a given row is more accurate than the model on the given column if a number (p-value less than 5%) appears in the row-column cell; otherwise, one cannot say that the model on the given row is more accurate than the model on the given column.)

Chapter 5

Multi-Site Spatio-Temporal Wind Forecasting

5.1 Methodology

Oftentimes, it is needed that the wind speed predictions be made for multiple locations within a region. In this case, one could adopt multiple single-site forecasting models (where each model, specifically constructed for each site, would provide forecast for a single location), or one could adopt a single multi-site forecasting model (where the model, designed to produced forecast for all sites, would provide the forecast for all locations within the region). In the context of our convolutional-based models and grid-like data, the development of multi-site spatio-temporal model would be analogous of that of video frame prediction, where we want to predict the value of each pixel (of each coordinate, in our case) for a given variable (or for multiple variables).

Let there be two explanatory, u- and v-components of wind, which are spatio-temporal processes. Both variables are random variables which may take values in location $s \in \mathcal{S}$ and time $t \in \mathcal{T}$, where $\mathcal{S}$ is the spatial domain and $\mathcal{T} \subset \mathbb{R}$ is the time domain. As described for the single-site case, to process the spatio-temporal variables, we assign each explanatory variable to one of input channels of the architecture. Let $l \in \{0, \ldots, L\}$ denote lagged steps for the variables and let $m \in \{1, 2\}$ denote explanatory variables u- and v-components, the multivariate input to the Convnets, $\mathbf{X}_t$, may be written as $\mathbf{X}_t = \{X_{m,t-l}\}$.

The objective is to use information of the multivariate 3D input tensor, $\mathbf{X}_t$, to predict a matrix Y_{t+1}, which denotes the wind speed one-step ahead of time for multiple sites (i.e., for multiple latitude-longitude coordinates – the same coordinates of variables in input channels). We propose the use of fully-convolutional neural networks to perform such a mapping. Below, we detail the proposed architectures, which redesign the U-Net architecture to process u- and v-components of wind in different ways.

5.1.1 ComPonentNet Architectures

We propose a collection of architectures, which we name *ComPonentNet* architectures, and abbreviate *CPNet*. The CPNet architectures were designed for multi-site spatio-temporal forecasting of a variable which may be decomposed in multiple components (factors), such as wind (which may be decomposed into u- and v-components of wind). The idea is that CPNet architectures process historic data of the components, and predict the components for a given time horizon. The forecast of the original variable may then be calculated from the component's forecasts.

We apply this concept to the prediction of wind speed: all CPNet architectures developed in our work use u- and v-components of wind as input and predict the one-step ahead u- and v-components of wind, which are combined to produce the wind speed. By using u- and v-components as inputs and predicting one-step ahead u- and v-components, we provide a way of performing multi-step prediction without recurring to external methods to predict the explanatory variables. In such sense, we overcome a limitation of the models proposed in the previous chapter.

The CPNet architectures are inspired by the U-Net architecture [36], and, as described in the introduction, are composed of three parts: a bottom-section (which contains the layers that are located closer to the input), a middle-section (which includes the layers that follow the bottom-section layers), and a top-section (which includes the last layers of the architecture, which follow the middle-section layers, and are closer to the output). As in the U-Net architecture, the bottom-section layers act as a contracting path and top-section layers as a expansion path, and bottom-section layers are connected to the top-section layers.

CPNet architecture

In Figure 5.1, we illustrate the core architecture of the CPNet family, which we thus name CPNet. In the CPNet architecture, u- and v-components are designed to have separate branches in an architecture which resembles a "double U-Net architecture".

In the bottom-section of the architecture (i.e., in the initial layers of the architecture), the wind components are processed by separate branches. Each branch applies three modules of two convolutional layers followed by a pooling layer. The first convolutional layer applies 24 filters, and the subsequent layers doubles the filters – following the same idea of U-Conv models in regard to number of filters in first layer (attempt of learning one hourly pattern per filter) and of the U-Net architecture in regard to strategy of number of filters for subsequent layers (doubling filters while reducing resolution in contracting path and halving filters while expanding resolution in the expanding path). This way, in each branch, the convolutional layers of the first module use 24 filters, the convolutional layers of the second module use 48 filters, and the convolutional layers of the third module use 96 filters. All pooling layers perform 2 x 2 pooling, and we define the pooling strategy (max or average pooling) in hyperparameter search.

The output feature maps of the branches are concatenated when the features reach low level resolution (in our architecture, when filters reach 1 x 1 resolution). Each branch outputs feature maps of 96 channels of 1 x 1 resolution. After concatenation, the feature maps are composed of 192 channels of 1 x 1 resolution. Following this, in the middle-section of the architecture, the concatenated features are processed alltogether by two convolutional layers in sequence (of 192 filters each).

The top-section of the CPNet architecture performs upconvolution in order to expand the low-resolution maps to original size. Right after middle-section layers processing, we create two separate branches to process u- and v-wind separately. Each branch applies (1) three modules of upconvolutional layer followed by two convolutional layers, and (2) one convolutional layer, which outputs feature map on original spatial size for the respective wind component. In the first module, the first upconvolutional layer has 96 filters; its output is concatenated (channel axis) with the output of the layer (from the bottom-section) that has 96 filters; the concatenated output has 192 channels, which is then processed by two convolutional layers that have 96 filters (which outputs a feature map of 96 channels – half the number of channels of the output from the previous module output, i.e., from the last convolutional layer of the bridge module). The other modules follow the same idea, differing only in the number of filters of the layers: in the second module, the upconvolutional and convolutional layers have 48 filters; in the third module, the upconvolutional and convolutional layers have 24 filters.

Note that, despite having feature maps concatenated in the middle-section, the top-section specializes in the respective component not only due to target exposition (each branch outputs a specific component) but also due to feature concatenation from bottom-section layers of the same branch. Following the output of each branch, let $w_{i,j,T+h}$, $u_{i,j,T+h}$ and $v_{i,j,T+h}$ denote, respectively, wind speed, u-component of wind and v-component of wind for latitude i and longitude j at time step $t+h$, where t is the present time step at forecasting time and h is the horizon of forecasting, then we calculate wind speed for each coordinate as follows:

$$w_{i,j,t+h} = \sqrt{u^2_{i,j,t+h} + v^2_{i,j,t+h}} \tag{5.1}$$

Then, the output wind speed of CPNet model is $W_{t+h} = \{w_{i,j,t-l}\}$ with $(i,j) \in \mathcal{S}$, the spatial domain of our problem (i.e., latitude-longitude coordinates).

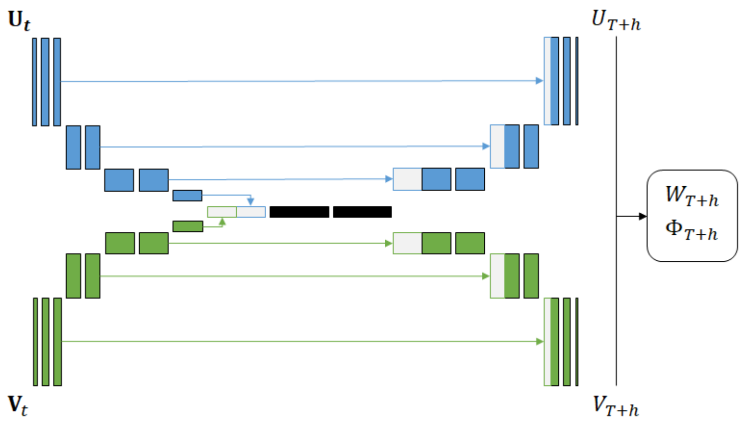


Figure 5.1: CPNet architecture

Bottom-Fused CPNet architecture

In Figure 5.2, we illustrate the bottom-fused CPNet architecture. In this architecture, u- and v-components are concatenated in the channels of the input data, and are processed alltogether in the bottom- and middle-section of the architecture. The u- and v-components are designed to have separate branches only in the top-section of the architecture. The macro architectural design of the bottom-fused CPNet follows the ideas of the CPNet architecture: bottom-section (which now is composed of a single branch that processes u- and v-wind alltogether) applies three modules of two convolutional layers followed by pooling layer, where the convolutional layers of the first module have 24 filters, the convolutional layers of the second module have 48 filters, and the layers of the last module have 96 filters (following the idea of doubling filters in contracting path, while reducing filter resolution); in the middle-section of the architecture, the low-resolution output maps of the bottom-section layers are processed by two convolutional layers (both with 192 filters), which are applied sequentially; the top-section of the bottom-fused CPNet has the same design of the core CPNet; the first convolutional. Equal to the core CPNet, the output of the bottom-fused CPNet are u- and v-components of wind. We obtain wind speed with Eq. 5.1.

In the bottom-fused CPNet, bottom-section layers do not pass specialized information to top-section layers, once u- and v-components are processed together at bottom-section layers. This way, the output branches of bottom-fused CPNet only specializes in each component by being exposed to the specific component's target (i.e., only via backpropagation of errors related to the specific component target). In our experiments, we investigate whether this setting is better than the core CPNet in terms of forecasting metrics.

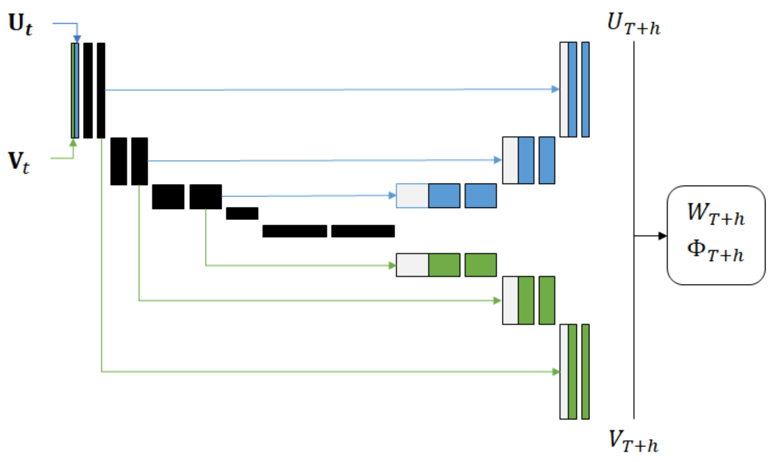


Figure 5.2: Bottom-Fused CPNet architecture

Fully-Fused CPNet architecture

In Figure 5.3, we illustrate the fully-fused CPNet architecture. In this architecture, u- and v-components are concatenated in the channels of the input data, and are processed alltogether in all layers of the architecture, except the last layer, which splits u- and v-components and outputs them separately.

The macro architectural design of the fully-fused CPNet is the following: bottom-section (composed of a single branch) applies three modules of two convolutional layers followed by one pooling layer; in the middle-section of the architecture, the low-resolution output maps of the bottom-section layers are processed by two convolutional layers, as in the bottom-fused CPNet; the top-section of the architecture is composed of a single branch, which applies three modules of upconvolutional layer followed by two convolutional layers. After the three modules, two branches are created, and each applies one convolutional layer to the output of the last top-section module. Equal to the core CPNet, the output of the fully-fused CPNet are u- and v-components of wind. We obtain wind speed with Eq. 5.1.

The fully-fused CPNet basically consists of a U-Net architecture with components (at present and past times) concatenated in the channels of input data and with two output maps – one for each component (at future time step). Following core CPNet and bottom-fused CPNet, the number of filters of modules at bottom-section double at each module: 24 filters in the first module, 48 filters in the second module and 96 in the third module; the filters of middle-section modules are 192; and the filters of modules at top-section halve at each module: 96 filters in the first module, 48 in the second module, and 24 filters in the third module.

In the fully-fused CPNet, u- and v-components are processed together at all layers. This way, the output branches of bottom-fused CPNet only specializes in each component by being exposed to the specific component's target.

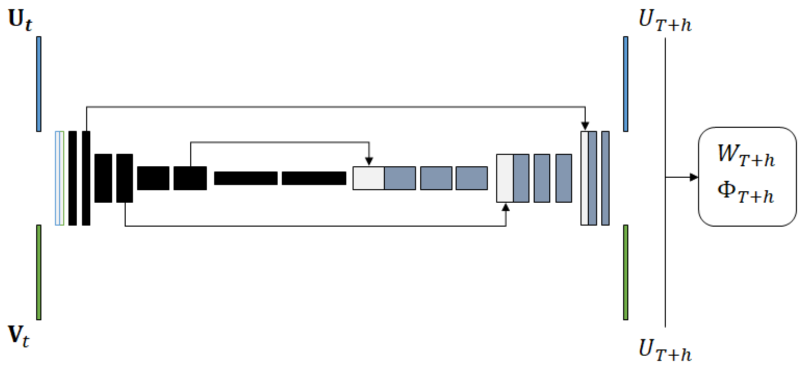


Figure 5.3: Fully-Fused CPNet architecture

5.2 Empirical Evaluation

5.2.1 Dataset and Experimental Setup

For the multi-site spatio-temporal forecasting experiments, we use data of u- and v-components of wind from the Bahia Wind Dataset, which we described in Section 4.2. For each wind component, we have 50448 hourly data for each one of the 10 x 10 locations (latitude-longitude pair). . Bahia Wind latitude-longitude pairs contemplate latitudes that range from -9.5° to -14.0° at 0.5° spatial resolution and longitudes that range from -44.5° to -40.0° at 0.5° spatial resolution.

In our design, we consider lagged variables in the input channels of the architectures. To account for the lagged variables (one to five-steps back in time) as features in our model, we had to reduce the number of samples of the dataset from 50448 to 50442 samples. Furthermore, instead of using data from 10 x 10 locations, we selected only 9 x 9 locations of the grid data (we do so aiming to dismiss the use of zero padding for feature concatenation in top-section layers).

Considering that the forecast task we aim to solve is the prediction of the wind speed for all coordinates of the Bahia Wind region, and considering that the proposed architectures now predict u- and v-components separately, the target data now is composed of two tensors with spatial dimension of 9 x 9. This way, the dataset for our multi-site setting (in CPNet context) had a shape of $50442 \times 9 \times 9 \times 6 \times 2$, where 50442 is the number of samples, 9 is the number of latitude points, 9 is the number of longitude points, 6 is the number of time steps (present and past) considered as historic in the input channels of each component, and 2 is the number of components.

We split the dataset into three sets of data: training, validation and test, following the same idea for single-site forecasting. The training set included 90% of the dataset's samples (accounting for 45398 samples). The validation set (used to check how well the models we trained predicted data that were not used for training) contained 5% of the dataset's samples (2522 samples). The test set totalized the remaining 5% samples (2522 samples). After selecting the best models based on training and validation set accuracies, we use the test set to compute the accuracies which would be obtained with unseen data.

5.2.2 Accuracy Measures

The accuracy measure we adopt to evaluate the proposed architectures is the average value of the mean absolute error (MAE) over all sites contained in the grid coordinates. We discussed the main reasons why we adopted the MAE as the accuracy measure in the last chapter.

The MAE for a single site was described in Equation4.2. Following that definition, the accuracy measure we use for the multi-site prediction is the following:

$$avgMAE = \frac{1}{C}\sum_{i=1}^{C} MAE_i \tag{5.2}$$

Where C is the number of coordinates for which the predictions are made.

5.3 First Results and Discussion

We apply the core CPNet (which we will refer to as CPNet), bottom-fused CPNet (which we will refer to as BF-CPNet) and the fully-fused CPNet (FF-CPNet) architectures in the prediction of one-step ahead multi-site wind speed for the Bahia Wind dataset, as detailed in the previous section. We also apply a U-Net architecture which takes u- and v-components of wind as input (which are concatenated, as in BF- and FF-CPNet architectures) and outputs directly wind speed one-step ahead of time. The latter architecture is used as benchmark for the problem.

We adopted the exponential linear unit (elu) as the activation function of the neurons in the U-Net. This activation function was selected over rectified linear unit (relu) after we observed that it was more suitable to the U-Net in the context of single-site prediction. In spite of this, more experiments must be made to understand other designs for our architecture. Batch size for all architectures (and respective configurations) was of 32 (a standard value in deep learning research). In all architectures, we tested max pooling and average pooling in contracting path. We also tested learning rates of 0.001 and 0.0001 (both with decay of 0.0001) for all architectures. We adopted early stopping (of 10 epochs) in all experiments.

For the BF-CPNet, we also tested weight regularization of 0.00005 against no weight regularization in convolutional layers, and tested dropout rate of 0.1 against no dropout rate (between convolutional layers – in all sections of the architecture). For the CPNet, we tested the same weight regularization hyperparameters; the dropout between convolutional layers was set to be zero. For the FF-CPNet and the U-Net, we test dropout rate of 0.1 and no dropout rate between convolutional layers, and do not adopt weight regularization.

Best CPNet model had average pooling, no weight regularization, no dropout rate, and learning rate of 0.001. Best FF-CPNet model also had no weight regularization, no dropout rate, and average pooling; however, learning rate was of 0.0001. Best BF-CPNet had same hyperparameter configuration as best CPNet model. Best U-Net configuration had same hyperparameter configuration as best FF-CPNet. Table 5.1 illustrates the $avgMAE$ results on training, validation and test sets. The best architecture, in our experiments, was the BF-CPNet, considering $avgMAE$ on test set.

Table 5.1: Results of average MAE (avgMAE) on training, validation and test sets

Model	Training	Validation	Test
BF-CPNet	0.1614	0.1555	0.1988
CPNet	0.1640	0.1558	0.2004
FF-CPNet	0.1650	0.1594	0.2025
U-Net	0.1644	0.1608	0.2036

Figure 5.4 illustrates a heatmaps of the errors of all architectures for the test set (note that each heatmap has a different scale bar, and thus colors in one heatmap do not relate to colors in other

heatmaps). We note that the error heatmap indicates that all models have a hard time predicting coordinates of the top right corner of the map. On the other hand, models provide lower MAEs for coordinates in bottom left corner of the map. The two models with lowest $avgMAE$ – FF-CPNet and U-Net – present highest disparities between top right corner errors and bottom left corner errors. It remains to be investigated why this is the case.

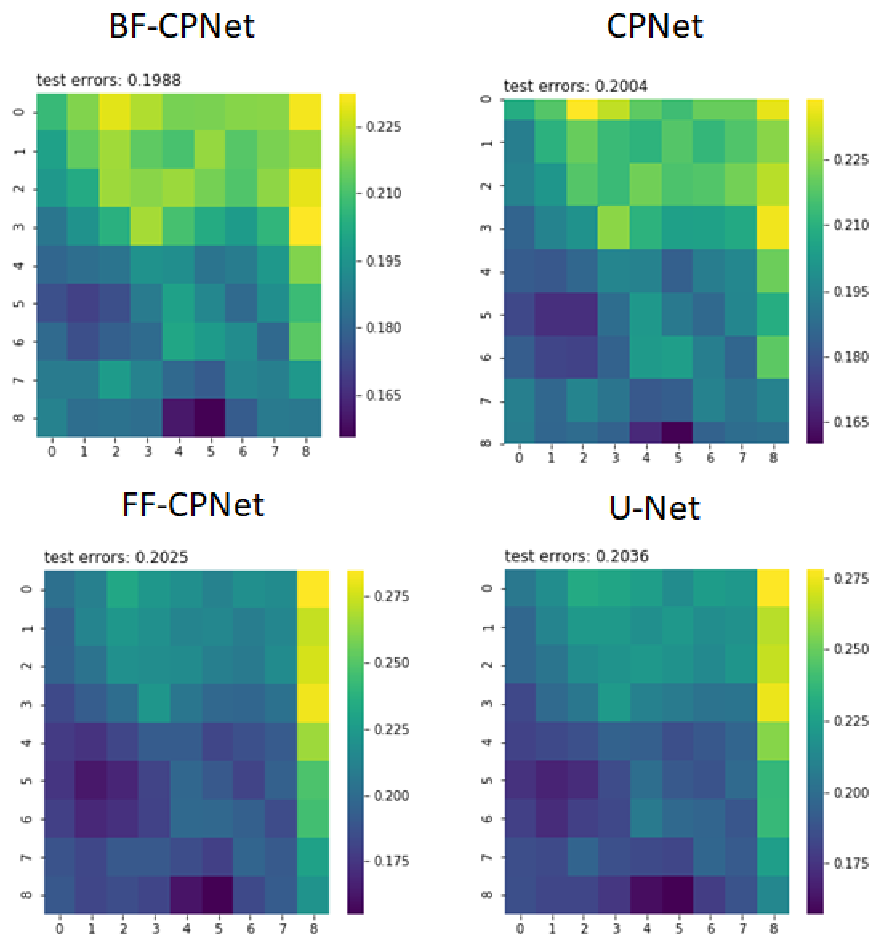


Figure 5.4: Error maps for the test set: BF-CPNet (upper left), CPNet (upper right), FF-CPNet (lower left) and U-Net (lower right)

Evaluating the diebold-mariano statistics for the $avgMAE$ on test set examples, we can say that the BF-CPNet presents more accurate predictions than FF-CPNet ($p = 0.0331$) and than U-Net ($p = 0.0011$) at $\alpha = 5\%$. However, we can not say that BF-CPNet presents more accurate predictions than CPNet ($p = 0.2461$). One may also say that CPNet's predictions are more accurate than U-Net predictions ($p = 0.0107$) at $\alpha = 5\%$; however, the same is not true if we consider CPNet's predictions against FF-CPNet predictions ($p = 0.1121$). We can not say that the predictions of the FF-Net model are more accurate than the predictions of the U-Net model.

Chapter 6

Conclusions

The increasing penetration of intermittent renewable energy in modern power systems is a challenge to power systems operation. A way of providing support to the system's operation is by developing advanced models, which provide accurate forecasts for the intermittent resources. In this work, we propose spatio-temporal methods based on deep learning for forecasting wind speed. We propose the U-Convolutional model, which combines U-Net architecture and Convnet, to predict wind speed for a single location using multiple spatio-temporal explanatory variables as input to the model. Additionally, we propose the ComPonentNet family of architectures to predict wind speed at multiple sites. For single-site prediction, we also investigate other Convnet architectures, which include Convnets with inception modules and/or identity mappings, and standard Convnets.

The results of our work indicate that the proposed U-Convolutional approach is a promising approach for single-site spatio-temporal forecasting. The results indicate that the proposed U-Convolutional models are competitive in the task of wind speed forecasting with higher accuracies than traditional univariate forecasting models and than fully-connected neural network modeled in a spatio-temporal. Our findings also indicate that the proposed U-Convolutional models, together with Residual Inception architecture, appear to do a better job in modeling the turbulent wind of Bahia than the other architectures. Conversely, Residual Convnet are considerably better than all other architectures when it comes to modeling the steady wind of Rio Grande do Norte.

In this work, to account for the temporality in our task, we considered a modeling setup where features at present and past times are included in the channel axis. This type of setup is present in video frame prediction task. For future work, we propose to account the temporality in our task by hybridizing ConvNet with Recurrent Neural Networks (RNNs). RNNs are a type of neural network that present feedback loops in their architecture, allowing them to keep, in the network, information about past inputs. Due to this property, RNNs are well suited to model data that are sequentially generated, such as time series, speech signal, and many others.

One topic of interest in future works is related with weight initialization in the different Convnet architectures. We noted, in our results, that the Residual Inception model had a high variance when modeling Bahia Wind dataset. As a consequence, the best model of Residual Inception appeared as second best when predicting Bahia dataset, despite being the third best algorithm in terms of average MAE. This means that there are certain initial weight values that lead to better models for a given dataset. In this context, it would be highly valuable to study the initialization strategies that lead to the best models for a given dataset – for the different architectures we tested in this work. With these observations, we outline a way forward in modeling wind speed with multiple explanatory data and Convnet modeling in single-site prediction set-up.

For multi-site prediction, we note that new experiments must be made to understand the performance of the proposed ComPonentNet architectures. In the first experiments, the Bottom-Fused ComPonentNet model produced best results. Next steps include the execution of experiments with addition of other features to the model (e.g., attention mechanism). Furthermore, one should evaluate statistical methods in the same task in order to compare performance of other classes of methods.

Appendices

Appendix A

Single-Site Hyperparameter Configurations

In this appendix, we provide the configuration designs we tested in our experiments. The configuration includes (1) architectural information, such as the number of filters in first and second convolutional modules in Convnet designs (f1 and f2, respectively, as depicted in Figure 4.2 and 4.3), (2) learning protocal information, which includes two cases – one with fixed learning rate and learning rate decay, and other with scheduled learning rate (in this case, we use a learning rate for 50 epochs and the other learning rate for the rest of the epochs; when we write 0.0005-0.0001, it means that for the first 50 epochs, we adopted learning rate of 0.0005, and for the remaining epochs, we adopted learning rate of 0.0001), (3) regularization information (dropout and weight regularization), and (4) average MAE (validation set) of five models adjusted with the given hyperparameter configuration.

A.1 Bahia Wind Dataset

Table A.1: Convnet - Bahia Wind Dataset

	Hyperparameters	(I)
architecture	filters - 1st module	32
	filters - 2nd module	168
	neurons - dense layer	280
	activation - output layer	Linear
	activation - other layers	ReLU
	pooling function	max
learning	learning scheduler	No
	learning rate	0.001
	learning rate decay	0.0001
	early stopping	No
	epochs	100
	batch size	32
regularization	dropout - last layer	0.25
	weight - conv layers	0.00005
	weight - dense layers	0.00005
Result	MAE on validation set	0.1755

Table A.2: Inception Model - Bahia Wind Dataset

	Hyperparameters	(I)	(II)	(III)
architecture	filters - 1st module (per path)	8	8	24
	filters - 2nd module (per path)	56	56	168
	neurons - dense layer	168	168	504
	activation - output layer	Linear	Linear	Linear
	activation - other layers	ReLU	ReLU	ReLU
	pooling function	max	max	max
learning	learning scheduler	No	No	No
	learning rate	0.001	0.001	0.001
	learning rate decay	0.0001	0.0001	0.0001
	early stopping	No	Yes	Yes
	epochs	100	100	100
	batch size	32	32	32
regularization	dropout - last layer	0.25	0.25	0.25
	weight - conv layers	0.00005	0.00005	0.00005
	weight - dense layers	0.00005	0.00005	0.00005
Result	MAE on validation set	0.1866	0.1860	0.1779

Table A.3: Residual Inception Model - Bahia Wind Dataset

	Hyperparameters	(I)
architecture	filters - 1st module (per path)	24
	filters - 2nd module (per path)	168
	neurons - dense layer	504
	activation - output layer	Linear
	activation - other layers	ReLU
	pooling function	max
learning	learning scheduler	No
	learning rate	0.001
	learning rate decay	0.0001
	early stopping	Yes
	epochs	100
	batch size	32
regularization	dropout - last layer	0.25
	weight - conv layers	0.00005
	weight - dense layers	0.00005
Result	MAE on validation set	0.1773

Table A.4: U-Convolutional Model - Bahia Wind Dataset

	Hyperparameters	(I)	(II)
architecture	filters - 1st module	32	32
	filters - 2nd module	168	168
	neurons - dense layer	280	168
	activation - U-Net	ELU	ELU
	activation - output layer	Linear	Linear
	activation - other layers	ReLU	ReLU
	pooling function	max	max
learning	learning scheduler	Yes	Yes
	learning rate	0.0005-0.0001	0.0005-0.0001
	learning rate decay	0.0001	0.0001
	early stopping	No	No
	epochs	100	100
	batch size	32	32
regularization	dropout - U-Net	0.1	0.1
	dropout - last layer	0.1	0.1
	weight - conv layers	0.00005	0.00005
	weight - dense layers	0	0
Result	MAE on validation set	0.1697	0.1688

Table A.5: Residual Convnet - Bahia Wind Dataset

	Hyperparameters	(I)	(II)	(III)	(IV)	(V)
architecture	filters - 1st module	32	32	32	32	32
	filters - 2nd module	168	168	168	168	168
	neurons - dense layer	280	168	168	168	168
	activation - output layer	Linear	Linear	Linear	Linear	Linear
	activation - other layers	ReLU	ReLU	ReLU	ReLU	ReLU
	pooling function	max	max	max	max	avg
learning	learning scheduler	No	No	Yes	Yes	No
	learning rate	0.001	0.001	0.0005-0.0001	0.0005-0.0001	0.001
	learning rate decay	0.0001	0.0001	0.0001	0.0001	0.0001
	early stopping	No	No	No	No	No
	epochs	100	100	100	100	100
	batch size	32	32	32	32	32
regularization	dropout - last layer	0.25	0.25	0.25	0.25	0.25
	weight - conv layers	0.00005	0.00005	0	0.00005	0.00005
	weight - dense layers	0.00005	0.00005	0.0001	0	0.00005
Result	MAE on validation set	0.1761	0.1794	0.1838	0.1842	0.1817

Table A.6: Residual U-Convolutional Model - Bahia Wind Dataset

	Hyperparameters	(I)	(II)	(III)	(IV)	(V)
architecture	filters - 1st module	32	32	32	32	32
	filters - 2nd module	168	168	168	168	168
	neurons - dense layer	280	168	168	168	168
	activation - U-Net	ELU	ELU	ELU	ELU	ELU
	activation - output layer	Linear	Linear	Linear	Linear	Linear
	activation - other layers	ReLU	ReLU	ReLU	ReLU	ReLU
	pooling function	max	max	max	max	max
learning	learning scheduler	Yes	Yes	Yes	Yes	Yes
	learning rate	0.0005-0.0001	0.0005-0.0001	0.0005-0.0001	0.0005-0.0001	0.0005-0.0001
	learning rate decay	0.0001	0.0001	0.0001	0.0001	0.0001
	early stopping	No	No	No	No	No
	epochs	100	100	100	100	100
	batch size	32	32	32	32	32
regularization	dropout - U-Net	0.1	0.1	0.1	0.1	0.1
	dropout - last layer	0.1	0.1	0.1	0.1	0.25
	weight - conv layers	0.00005	0.00005	0.00005	0	0.00005
	weight - dense layers	0.00005	0	0.00005	0	0
Result	average MAE on validation set	0.1813	0.1652	0.1683	0.1715	0.1678

A.2 Rio Grande do Norte Wind Dataset

Table A.7: Convnet - RN Wind Dataset

	Hyperparameters	(I)
architecture	filters - 1st module	24
	filters - 2nd module	168
	neurons - dense layer	168
	activation - output layer	Linear
	activation - other layers	ReLU
	pooling function	max
learning	learning scheduler	Yes
	learning rate	0.0005-0.0001
	learning rate decay	0.0001
	early stopping	No
	epochs	100
	batch size	32
regularization	dropout - last layer	0.25
	weight - conv layers	0.00005
	weight - dense layers	0.00005
Result	MAE on validation set	0.2036

Table A.8: Residual Convnet - RN Wind Dataset

	Hyperparameters	(I)	(II)
architecture	filters - 1st module	24	32
	filters - 2nd module	168	168
	neurons - dense layer	168	168
	activation - output layer	Linear	Linear
	activation - other layers	ReLU	ReLU
	pooling function	max	max
learning	learning scheduler	Yes	Yes
	learning rate	0.0005-0.0001	0.0005-0.0001
	learning rate decay	0.0001	0.0001
	early stopping	No	No
	epochs	100	100
	batch size	32	32
regularization	dropout - last layer	0.25	0.25
	weight - conv layers	0	0
	weight - dense layers	0.0001	0.0001
Result	MAE on validation set	0.1891	0.1845

Table A.9: Inception Model - RN Wind Dataset

	Hyperparameters	(I)	(II)
architecture	filters - 1st module (per path)	8	24
	filters - 2nd module (per path)	56	168
	neurons - dense layer	168	504
	activation - output layer	Linear	Linear
	activation - other layers	ReLU	ReLU
	pooling function	max	max
learning	learning scheduler	No	No
	learning rate	0.001	0.001
	learning rate decay	0.0001	0.0001
	early stopping	No	Yes
	epochs	100	100
	batch size	32	32
regularization	dropout - last layer	0.25	0.25
	weight - conv layers	0	0
	weight - dense layers	0.0001	0.0001
Result	MAE on validation set	0.2005	0.1950

Table A.10: Residual Inception Model - RN Wind Dataset

	Hyperparameters	(I)	(II)
architecture	filters - 1st module (per path)	8	24
	filters - 2nd module (per path)	56	168
	neurons - dense layer	168	504
	activation - output layer	Linear	Linear
	activation - other layers	ReLU	ReLU
	pooling function	max	max
learning	learning scheduler	Yes	Yes
	learning rate	0.0005-0.0001	0.0005-0.0001
	learning rate decay	0.0001	0.0001
	early stopping	No	No
	epochs	100	100
	batch size	32	32
regularization	dropout - last layer	0.25	0.25
	weight - conv layers	0	0
	weight - dense layers	0.00005	0.0001
Result	MAE on validation set	0.1891	0.1800

Table A.11: U-Convolutional Model - RN Wind Dataset

	Hyperparameters	(I)	(II)
architecture	filters - 1st module	32	32
	filters - 2nd module	168	168
	neurons - dense layer	280	168
	activation - U-Net	ELU	ELU
	activation - output layer	Linear	Linear
	activation - other layers	ReLU	ReLU
	pooling function	max	max
learning	learning scheduler	Yes	Yes
	learning rate	0.0005-0.0001	0.0005-0.0001
	learning rate decay	0.0001	0.0001
	early stopping	No	No
	epochs	200	100
	batch size	32	32
regularization	dropout - U-Net	0.1	0
	dropout - last layer	0.25	0.25
	weight - conv layers	0.00005	0
	weight - dense layers	0.00005	0.00005
Result	MAE on validation set	0.1902	0.1857

Table A.12: Residual U-Convolutional Model - RN Wind Dataset

	Hyperparameters	(I)	(II)	(III)	(IV)	(V)	(VI)	(VII)	(VIII)
architecture	filters - 1st module	32	32	32	32	32	32	32	32
	filters - 2nd module	168	168	240	168	168	168	168	168
	neurons - dense layer	280	280	280	168	168	168	168	168
	activation - U-Net	ELU	ELU	ELU	ELU	ELU	ELU	ELU	ELU
	activation - output layer	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
	activation - other layers	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
	pooling function	max	max	max	max	max	max	max	max
learning	learning scheduler	No	No	No	No	No	Yes	Yes	Yes
	learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0005-0.0001	0.0005-0.0001	0.0005-0.0001
	learning rate decay	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	early stopping	No	No	No	No	No	No	No	No
	epochs	200	200	200	200	200	200	200	200
	batch size	32	32	32	32	32	32	32	32
regularization	dropout - U-Net	0	0	0	0	0	0	0	0
	dropout - last layer	0.1	0.1	0.1	0.1	0.1	0.1	0.25	0
	weight - conv layers	None	None	None	None	None	None	None	None
	weight - dense layers	0.00005	0.0001	0.0001	0.00005	None	0.00005	0.0001	0.0001
Result	average MAE on validation set	0.1934	0.1963	0.1919	0.1941	0.1927	0.1826	0.1911	0.1842

Bibliography

[1] R. Quadrelli and S. Peterson, "The energy–climate challenge: Recent trends in co2 emissions from fuel combustion", *Energy Policy*, vol. 35, no. 11, pp. 5938–5952, 2007, ISSN: 0301-4215. DOI: `https://doi.org/10.1016/j.enpol.2007.07.001`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0301421507003126`.

[2] IPCC, "Renewable Energy Sources and Climate Change Mitigation", Tech. Rep., Nov. 2012. [Online]. Available: `https://www.ipcc.ch/report/renewable-energy-sources-and-climate-change-mitigation/`.

[3] IEA, *World Energy Outlook 2017*. IEA, 2017, pp. 1–15, ISBN: 9789264243668. DOI: `10.1016/0301-4215(73)90024-4`. [Online]. Available: `https://www.iea.org/weo2017/`.

[4] Le Xie, P. M. S. Carvalho, L. A. F. M. Ferreira, Juhua Liu, B. H. Krogh, N. Popli, and M. D. Ilic, "Wind Integration in Power Systems: Operational Challenges and Possible Solutions", *Proceedings of the IEEE*, vol. 99, no. 1, pp. 214–232, Jan. 2011, ISSN: 0018-9219. DOI: `10.1109/JPROC.2010.2070051`.

[5] B. François and F. Galiana, "Stochastic security for operations planning with significant wind power generation", *IEEE Transactions on Power Systems*, vol. 23, no. 2, pp. 306–316, May 2008, ISSN: 08858950. DOI: `10.1109/TPWRS.2008.919318`. [Online]. Available: `https://ieeexplore.ieee.org/document/4470561`.

[6] O. Ait Maatallah, A. Achuthan, K. Janoyan, and P. Marzocca, "Recursive wind speed forecasting based on Hammerstein Auto-Regressive model", *Applied Energy*, vol. 145, pp. 191–197, May 2015, ISSN: 0306-2619. DOI: `10.1016/J.APENERGY.2015.02.032`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0306261915002093`.

[7] Q. Hu, S. Zhang, M. Yu, and Z. Xie, "Short-Term Wind Speed or Power Forecasting With Heteroscedastic Support Vector Regression", *IEEE Transactions on Sustainable Energy*, vol. 7, no. 1, pp. 241–249, Jan. 2016, ISSN: 1949-3029. DOI: `10.1109/TSTE.2015.2480245`. [Online]. Available: `http://ieeexplore.ieee.org/document/7335638/`.

[8] X. Zhu and M. G. Genton, "Short-Term Wind Speed Forecasting for Power System Operations", *International Statistical Review*, vol. 80, no. 1, pp. 2–23, Apr. 2012, ISSN: 03067734. DOI: `10.1111/j.1751-5823.2011.00168.x`. [Online]. Available: `http://doi.wiley.com/10.1111/j.1751-5823.2011.00168.x`.

[9] G. Box and G. Jenkins, *Time series Analysis: Forecasting and Control*. San Francisco, California: Holden-Day, 1976.

[10] J. Z. Wang, Y. Wang, and P. Jiang, “The study and application of a novel hybrid forecasting model - A case study of wind speed forecasting in China”, *Applied Energy*, vol. 143, pp. 472–488, Apr. 2015, ISSN: 03062619. DOI: 10.1016/j.apenergy.2015.01.038. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261915000446.

[11] S. Haykin, *Neural Networks and Learning Machines*, 3rd. New Jersey: Pearson, 2009.

[12] V. N. Vapnik, *An overview of statistical learning theory*, 1999. DOI: 10.1109/72.788640. [Online]. Available: http://ieeexplore.ieee.org/document/788640/.

[13] L.-X. Wang and J. Mendel, “Generating fuzzy rules by learning from examples”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992. DOI: 10.1109/21.199466.

[14] J. Jung and R. P. Broadwater, “Current status and future advances for wind speed and power forecasting”, *Renewable and Sustainable Energy Reviews*, vol. 31, pp. 762–777, Mar. 2014, ISSN: 1364-0321. DOI: 10.1016/J.RSER.2013.12.054. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032114000094.

[15] M. Lei, L. Shiyan, J. Chuanwen, L. Hongling, and Z. Yan, “A review on the forecasting of wind speed and generated power”, *Renewable and Sustainable Energy Reviews*, vol. 13, no. 4, pp. 915–920, May 2009, ISSN: 1364-0321. DOI: 10.1016/J.RSER.2008.02.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032108000282.

[16] S. A. Vargas, G. R. T. Esteves, P. M. Maçaira, B. Q. Bastos, F. L. C. Oliveira, and R. C. Souza, “Wind power generation: A review and a research agenda”, *Journal of Cleaner Production*, vol. 218, pp. 850–870, 2019, ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.02.015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959652619303944.

[17] A. Tascikaraoglu, B. M. Sanandaji, G. Chicco, V. Cocina, F. Spertino, O. Erdinc, N. G. Paterakis, and J. P. Catalao, “Compressive Spatio-Temporal Forecasting of Meteorological Quantities and Photovoltaic Power”, *IEEE Transactions on Sustainable Energy*, vol. 7, no. 3, pp. 1295–1305, Jul. 2016, ISSN: 19493029. DOI: 10.1109/TSTE.2016.2544929.

[18] J. Dowell and P. Pinson, “Very-Short-Term Probabilistic Wind Power Forecasts by Sparse Vector Autoregression”, *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 763–770, 2016, ISSN: 19493053. DOI: 10.1109/TSG.2015.2424078.

[19] C. Persson, P. Bacher, T. Shiga, and H. Madsen, “Multi-site solar power forecasting using gradient boosted regression trees”, *Solar Energy*, vol. 150, pp. 423–436, Jul. 2017, ISSN: 0038092X. DOI: 10.1016/j.solener.2017.04.066. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X17303717.

[20] M. André, S. Dabo-Niang, T. Soubdhan, and H. Ould-Baba, *Predictive spatio-temporal model for spatially sparse global solar radiation data*, Sep. 2016. DOI: 10.1016/j.energy.2016.06.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544216307769.

[21] L. Xie, Y. Gu, X. Zhu, and M. G. Genton, “Short-term spatio-temporal wind power forecast in robust look-ahead power system dispatch”, *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 511–520, Jan. 2014, ISSN: 19493053. DOI: 10.1109/TSG.2013.2282300.

[22] M. Koivisto, J. Seppänen, I. Mellin, J. Ekström, J. Millar, I. Mammarella, M. Komppula, and M. Lehtonen, “Wind speed modeling using a vector autoregressive process with a time-dependent intercept term”, *International Journal of Electrical Power and Energy Systems*, vol. 77, pp. 91–99, May 2016, ISSN: 01420615. DOI: 10.1016/j.ijepes.2015.11.027. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142061515004470.

[23] A. W. Aryaputera, D. Yang, L. Zhao, and W. M. Walsh, "Very short-term irradiance forecasting at unobserved locations using spatio-temporal kriging", *Solar Energy*, vol. 122, pp. 1266–1278, Dec. 2015, ISSN: 0038092X. DOI: 10.1016/j.solener.2015.10.023. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X15005745.

[24] M. He, L. Yang, J. Zhang, and V. Vittal, "A spatio-temporal analysis approach for short-term forecast of wind farm generation", *IEEE Transactions on Power Systems*, vol. 29, no. 4, pp. 1611–1622, Jul. 2014, ISSN: 08858950. DOI: 10.1109/TPWRS.2014.2299767. [Online]. Available: http://ieeexplore.ieee.org/document/6727513/.

[25] J. Heinermann and O. Kramer, "Machine learning ensembles for wind power prediction", *Renewable Energy*, vol. 89, pp. 671–679, Apr. 2016, ISSN: 18790682. DOI: 10.1016/j.renene.2015.11.073. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148115304894.

[26] I. G. Damousis, M. C. Alexiadis, J. B. Theocharis, and P. S. Dokopoulos, "A fuzzy model for wind speed prediction and power generation in wind parks using spatial correlation", *IEEE Transactions on Energy Conversion*, vol. 19, no. 2, pp. 352–361, Jun. 2004, ISSN: 08858969. DOI: 10.1109/TEC.2003.821865. [Online]. Available: http://ieeexplore.ieee.org/document/1300701/.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", *Advances In Neural Information Processing Systems*, pp. 1–9, 2012, ISSN: 10495258. DOI: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[28] F. Chollet, "Xception: Deep learning with separable convolutions", *arXiv preprint arXiv:1610.02357*, pp. 1–14, 2016. DOI: 10.1109/CVPR.2017.195. [Online]. Available: https://arxiv.org/abs/1610.02357.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", Dec. 2015. [Online]. Available: http://arxiv.org/abs/1512.03385.

[30] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO, "Convolutional lstm network: A machine learning approach for precipitation nowcasting", in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 802–810. [Online]. Available: http://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf.

[31] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error", *CoRR*, vol. abs/1511.05440, 2015. arXiv: 1511.05440. [Online]. Available: http://arxiv.org/abs/1511.05440.

[32] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, "Decomposing motion and content for natural video sequence prediction", *CoRR*, vol. abs/1706.08033, 2017. arXiv: 1706.08033. [Online]. Available: http://arxiv.org/abs/1706.08033.

[33] E. Denton and V. Birodkar, "Unsupervised learning of disentangled representations from video", *CoRR*, vol. abs/1705.10915, 2017. arXiv: 1705.10915. [Online]. Available: http://arxiv.org/abs/1705.10915.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets", in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[36] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, 2015, ISSN: 1611-3349. DOI: 10.1007/978-3-319-24574-4_28. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-24574-4_28.

[37] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection – a new baseline", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[38] A. Zhu, X. Li, Z. Mo, and R. Wu, "Wind power prediction based on a convolutional neural network", in *2017 International Conference on Circuits, Devices and Systems (ICCDS)*, IEEE, Sep. 2017, pp. 131–135, ISBN: 978-1-5386-1870-7. DOI: 10.1109/ICCDS.2017.8120465. [Online]. Available: http://ieeexplore.ieee.org/document/8120465/.

[39] H. Liu, X. Mi, and Y. Li, "Smart deep learning based wind speed prediction model using wavelet packet decomposition, convolutional neural network and convolutional long short term memory network", *Energy Conversion and Management*, vol. 166, pp. 120–131, 2018, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2018.04.021. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S019689041830356X.

[40] Y. Chen, S. Zhang, W. Zhang, J. Peng, and Y. Cai, "Multifactor spatio-temporal correlation model based on a combination of convolutional neural network and long short-term memory neural network for wind speed forecasting", *Energy Conversion and Management*, vol. 185, pp. 783–799, 2019, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2019.02.018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196890419302006.

[41] A. R. Silva, F. M. Pimenta, A. T. Assireu, and M. H. C. Spyrides, "Complementarity of brazil's hydro and offshore wind power", *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 413–427, 2016, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.11.045. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364032115013106.

[42] G. G. Dranka and P. Ferreira, "Planning for a renewable future in the brazilian power system", *Energy*, vol. 164, pp. 496–511, 2018, ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2018.08.164. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544218317006.

[43] B. Bezerra, Á. Veiga, L. A. Barroso, and M. Veiga, "Stochastic long-term hydrothermal scheduling with parameter uncertainty in autoregressive streamflow models", *IEEE Transactions on Power Systems*, vol. 32, pp. 999–1006, 2 2017, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2016.2572722. [Online]. Available: https://ieeexplore.ieee.org/document/7480404/.

[44] A. Street, D. A. Lima, A. Veiga, B. Fanzeres, L. Freire, and B. S. Amaral, "Fostering wind power penetration into the brazilian forward-contract market", *2012 IEEE Power and Energy Society General Meeting*, pp. 1–8, 2012.

[45] P. M. Maçaira, A. M. T. Thomé, F. L. C. Oliveira, and A. L. C. Ferrer, "Time series analysis with explanatory variables: A systematic literature review", *Environmental Modelling & Software*, vol. 107, pp. 199–209, 2018, ISSN: 1364-8152. DOI: https://doi.org/10.1016/j.envsoft.2018.06.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S136481521730542X.

[46] A. M. D. Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing", *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1513–1527, 2011. DOI: 10.1198/jasa.2011.tm09771. eprint: https://doi.org/10.1198/jasa.2011.tm09771. [Online]. Available: https://doi.org/10.1198/jasa.2011.tm09771.

[47] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539. [Online]. Available: http://www.nature.com/doifinder/10.1038/nature14539.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st. Cambridge, MA: MIT Press, 2017, ISBN: 9780262035613.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, ISSN: 00189219. DOI: 10.1109/5.726791.

[50] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning", Tech. Rep., Mar. 2016. [Online]. Available: https://arxiv.org/abs/1603.07285.

[51] F. Chollet, *Deep learning with Python*. Manning Publications Co., 2018, p. 361, ISBN: 9781617294433. [Online]. Available: https://www.manning.com/books/deep-learning-with-python.

[52] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle, "Brain tumor segmentation with Deep Neural Networks", *Medical Image Analysis*, vol. 35, pp. 18–31, Jan. 2017, ISSN: 1361-8415. DOI: 10.1016/J.MEDIA.2016.05.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841516300330.

[53] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks", *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, vol. 15, pp. 315–323, 2011, ISSN: 15324435. DOI: 10.1.1.208.6449.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[55] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", Sep. 2014. [Online]. Available: https://arxiv.org/abs/1409.1556.

[56] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14, Orlando, Florida, USA: ACM, 2014, pp. 675–678, ISBN: 978-1-4503-3063-3. DOI: 10.1145/2647868.2654889. [Online]. Available: http://doi.acm.org/10.1145/2647868.2654889.

[57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html.

[58] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[59] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning", in *AAAI*, 2016.

[60] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network", in *Advances in Neural Information Processing Systems 2*, Morgan-Kaufmann, 1990, pp. 396–404.

[61] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", *CoRR*, vol. abs/1411.4038, 2014. arXiv: `1411.4038`. [Online]. Available: `http://arxiv.org/abs/1411.4038`.

[62] C. Chatfield, *The Analysis of Time Series: An Introduction*, 4th. London, UK: Chapman and Hall/CRC, 1989, ISBN: 9781584883173.

[63] J. L. Harvill, "Spatio-temporal processes", *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 3, pp. 375–382, 2010. DOI: `10.1002/wics.88`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.88`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.88`.

[64] M. S. Paez, D. Gamerman, and V. D. Oliveira, "Interpolation performance of a spatio-temporal model with spatially varying coefficients: Application to pm10 concentrations in rio de janeiro", *Environmental and Ecological Statistics*, vol. 12, no. 2, pp. 169–193, Jun. 2005, ISSN: 1573-3009. DOI: `10.1007/s10651-005-1040-7`. [Online]. Available: `https://doi.org/10.1007/s10651-005-1040-7`.

[65] J. A. González, F. J. Rodríguez-Cortés, O. Cronie, and J. Mateu, "Spatio-temporal point process statistics: A review", *Spatial Statistics*, vol. 18, pp. 505–544, 2016, ISSN: 2211-6753. DOI: `https://doi.org/10.1016/j.spasta.2016.10.002`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S2211675316301130`.

[66] E. E. Kammann and M. P. Wand, "Geoadditive models", *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 52, no. 1, pp. 1–18, 2003. DOI: `10.1111/1467-9876.00385`. eprint: `https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9876.00385`. [Online]. Available: `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9876.00385`.

[67] N. Cressie and H.-C. Huang, "Classes of nonseparable, spatio-temporal stationary covariance functions", *Journal of the American Statistical Association*, vol. 94, no. 448, pp. 1330–1340, 1999, ISSN: 01621459. [Online]. Available: `http://www.jstor.org/stable/2669946`.

[68] D. G. Krige, "A statistical approach to some mine valuation and allied problems on the witwatersrand", Master's thesis, University of the Witwatersrand, 1951.

[69] W. C. M. van Beers and J. P. C. Kleijnen, "Kriging for interpolation in random simulation", *The Journal of the Operational Research Society*, vol. 54, no. 3, pp. 255–262, 2003, ISSN: 01605682, 14769360. [Online]. Available: `http://www.jstor.org/stable/4101619`.

[70] W. A. Martínez, C. E. Melo, and O. O. Melo, "Median polish kriging for space–time analysis of precipitation", *Spatial Statistics*, vol. 19, pp. 1–20, 2017, ISSN: 2211-6753. DOI: `https://doi.org/10.1016/j.spasta.2016.10.003`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S2211675316301336`.

[71] D.-J. Lee and M. Durbán, "P-spline anova-type interaction models for spatio-temporal smoothing", *Statistical Modelling*, vol. 11, no. 1, pp. 49–69, 2011. DOI: 10.1177/1471082X1001100104. eprint: https://doi.org/10.1177/1471082X1001100104. [Online]. Available: https://doi.org/10.1177/1471082X1001100104.

[72] L. Fahrmeir, T. Kneib, and S. Lang, "Penalized structured additive regression for space-time data: A bayesian perspective", *Statistica Sinica*, vol. 14, no. 3, pp. 731–761, 2004, ISSN: 10170405, 19968507. [Online]. Available: http://www.jstor.org/stable/24307414.

[73] M. S. Paez and D. Gamerman, "Study of the space–time effects in the concentration of airborne pollutants in the metropolitan region of rio de janeiro", *Environmetrics*, vol. 14, no. 4, pp. 387–408, 2003. DOI: 10.1002/env.594. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/env.594.

[74] Z. Luo, G. Wahba, and D. R. Johnson, "Spatial–temporal analysis of temperature using smoothing spline anova", *Journal of Climate*, vol. 11, no. 1, pp. 18–28, 1998. DOI: 10.1175/1520-0442(1998)011<0018:STAOTU>2.0.CO;2. eprint: https://doi.org/10.1175/1520-0442(1998)011<0018:STAOTU>2.0.CO;2. [Online]. Available: https://doi.org/10.1175/1520-0442(1998)011%3C0018:STAOTU%3E2.0.CO;2.

[75] P. Diggle and P. J. Ribeiro, *Model-based Geostatistics*, 1st. Springer-Verlag New York, 2007, ISBN: 9780387485362.

[76] O. Kramer, F. Gieseke, and B. Satzger, "Wind energy prediction and monitoring with neural computation", *Neurocomputing*, vol. 109, pp. 84–93, 2013, New trends on Soft Computing Models in Industrial and Environmental Applications, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2012.07.029. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231212006674.

[77] M. Bilgili, B. Sahin, and A. Yasar, "Application of artificial neural networks for the wind speed prediction of target station using reference stations data", *Renewable Energy*, vol. 32, no. 14, pp. 2350–2360, 2007, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2006.12.001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148106003429.

[78] J. Xu, B. Ni, Z. Li, S. Cheng, and X. Yang, "Structure preserving video prediction", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[79] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[80] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale", *CoRR*, vol. abs/1611.01236, 2016. arXiv: 1611.01236. [Online]. Available: http://arxiv.org/abs/1611.01236.

[81] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra, "Video (language) modeling: A baseline for generative models of natural videos", *CoRR*, vol. abs/1412.6604, 2014. arXiv: 1412.6604. [Online]. Available: http://arxiv.org/abs/1412.6604.

[82] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms", in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 843–852. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045209.

[83] J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles, "Learning to decompose and disentangle representations for video prediction", in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 515–524. [Online]. Available: http://papers.nips.cc/paper/7333-learning-to-decompose-and-disentangle-representations-for-video-prediction.pdf.

[84] S. Tulyakov, M. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation", *CoRR*, vol. abs/1707.04993, 2017. arXiv: 1707.04993. [Online]. Available: http://arxiv.org/abs/1707.04993.

[85] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning", in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11, Washington, DC, USA: IEEE Computer Society, 2011, pp. 2018–2025, ISBN: 978-1-4577-1101-5. DOI: 10.1109/ICCV.2011.6126474. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126474.

[86] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", pp. 1–15, 2014, ISSN: 09252312. DOI: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503. [Online]. Available: http://arxiv.org/abs/1412.6980.

[87] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *The Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011, ISSN: 1532-4435. DOI: 10.1109/CDC.2012.6426698. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html%20http://dl.acm.org/citation.cfm?id=2021068%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.232.1303&rep=rep1&type=pdf#page=265.

[88] S. Saha, S. Moorthi, X. Wu, J. Wang, S. Nadiga, P. Tripp, D. Behringer, Y. T. Hou, H. Y. Chuang, M. Iredell, M. Ek, J. Meng, R. Yang, M. P. Mendez, H. Van Den Dool, Q. Zhang, W. Wang, M. Chen, and E. Becker, "The NCEP climate forecast system version 2", *Journal of Climate*, vol. 27, no. 6, pp. 2185–2208, Mar. 2014, ISSN: 08948755. DOI: 10.1175/JCLI-D-12-00823.1. [Online]. Available: http://journals.ametsoc.org/doi/abs/10.1175/JCLI-D-12-00823.1.

[89] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy", *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, Oct. 2006, ISSN: 01692070. DOI: 10.1016/j.ijforecast.2006.03.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207006000239?via%3Dihub.

[90] G. Li, J. Shi, and J. Zhou, "Bayesian adaptive combination of short-term wind speed forecasts from neural network models", *Renewable Energy*, vol. 36, no. 1, pp. 352–359, Jan. 2011, ISSN: 09601481. DOI: 10.1016/j.renene.2010.06.049. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148110003228.

[91] A. Tascikaraoglu, B. M. Sanandaji, K. Poolla, and P. Varaiya, "Exploiting sparsity of interconnections in spatio-temporal wind speed forecasting using Wavelet Transform", *Applied Energy*, vol. 165, pp. 735–747, Mar. 2016, ISSN: 03062619. DOI: 10.1016/j.apenergy.2015.12.082.

[92] F. Chollet *et al.*, *Keras*, \url{https://keras.io}, 2015.

[93] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for R", *Journal of Statistical Software*, vol. 26, no. 3, pp. 1–22, 2008. [Online]. Available: http://www.jstatsoft.org/article/view/v027i03.

[94] H. Abdi and L. J. Williams, "Principal component analysis", *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010. DOI: 10.1002/wics.101. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.101. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101.

[95] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop", in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48, ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_3.

[96] J. Dowell, S. Weiss, D. Hill, and D. Infield, "Short-term spatio-temporal prediction of wind speed and direction", *Wind Energy*, vol. 17, no. 12, pp. 1945–1955, Dec. 2014, ISSN: 10991824. DOI: 10.1002/we.1682. [Online]. Available: http://doi.wiley.com/10.1002/we.1682.

[97] C. Feng, M. Cui, B.-M. Hodge, and J. Zhang, "A data-driven multi-model methodology with deep feature selection for short-term wind forecasting", *Applied Energy*, vol. 190, pp. 1245–1257, 2017, ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2017.01.043. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261917300508.

[98] J. Hu and J. Wang, "Short-term wind speed prediction using empirical wavelet transform and gaussian process regression", *Energy*, vol. 93, pp. 1456–1466, 2015, ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2015.10.041. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544215014097.