Felipe Whitaker de Assumpção Mattos Tavares

Short term Wind Speed Scenario Generation for Brazil with Improved Generative Adversarial Networks

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-graduação em Engenharia de Produção of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia de Produção.

Advisor :	Prof. Fernando Luiz Cyrino Oliveira
Co-advisor:	Prof. Marley Maria Bernardes Rebuzzi Vellasco

Rio de Janeiro
September 2024

Felipe Whitaker de Assumpção Mattos Tavares

Short term Wind Speed Scenario Generation for Brazil with Improved Generative Adversarial Networks

Thesis presented to the Programa de Pós–graduação em Engenharia de Produção da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia de Produção. Approved by the Examination Committee:

Rio de Janeiro, September the 18th, 2024

Felipe Whitaker de Assumpção Mattos Tavares

Completed his Bachelor's in Engenharia Industrial with minor in Risk Analysis at Pontifical Catholic University of Rio de Janeiro in 2022. Started his Master's in Operations Research in 2023, which introduced him to the electricity sector's challenges. The University's environment lead him to a research project on the use of climate variables for energy demand prediction, deepening his interest in the sector. Currently works with software development for a renewable energy company. Has experience with Software Engineering, specially regarding data pipelines, from extraction to modeling.

Bibliographic data

Tavares, Felipe Whitaker de Assumpção Mattos

Short term Wind Speed Scenario Generation for Brazil with Improved Generative Adversarial Networks / Felipe Whitaker de Assumpção Mattos Tavares; advisor: Fernando Luiz Cyrino Oliveira; co-advisor: Marley Maria Bernardes Rebuzzi Vellasco. – 2024.

57 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, 2024.

Inclui bibliografia

1. Engenharia Industrial – Teses. 2. Previsão de Velocidade do Vento. 3. Modelagem Generativa. 4. Redes Neurais Convolucionais. I. Cyrino Oliveira, Fernando Luiz. II. Vellasco, Marley M. Bernardes Rebuzzi. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. IV. Título.

CDD:

... Depois sentir o arrepio
Do vento que a noite traz
E o diz-que-diz-que macio
Que brota dos coqueirais ...

Tarde em Itapoã,
Toquinho

To my advisors, for their guidance and faith;
to my friends and family, for their support and encouragement;
to those who, for lack of better a expression, are gone with the wind;
and for the luck I have to share life with them.

Acknowledgments

Thanks for the University's exemption scholarship, and the Brazilian National Council for Scientific and Technological Development (CNPq) under Grants [number 130704/2023-7].

This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brasil - Finance Code 001.

Abstract

Tavares, Felipe Whitaker de Assumpção Mattos; Cyrino Oliveira, Fernando Luiz (Advisor); Vellasco, Marley M. Bernardes Rebuzzi (Co-Advisor). **Short term Wind Speed Scenario Generation for Brazil with Improved Generative Adversarial Networks**. Rio de Janeiro, 2024. 57p. Dissertação de Mestrado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

The variability of renewable energy sources, such as wind power, presents a significant challenge for grid operators in maintaining operational stability. This is specially true to the medium-term (from hours to days ahead), which is both influenced by recent past data and broader trends and heavily influences decision making. This research proposes a Convolutional Generator Network conditioned on the previous step of u- (latitudinal) and v- (longitudinal) wind speed components to generate wind speed scenarios using the Conditional Generative Adversarial Networks training algorithm. The model is compared to the state of the art in weather forecasting, Numerical Weather Prediction Systems. The proposed generator model outperforms the benchmark for a forth of the months in the test dataset when predicting over two weeks (28 12-hourly steps) starting from a single data point with much lower computational cost, less input data and similar long-term stability. Additionally, its forecasts are statistically equal to the state-of-the-art in 71.97% of series.

Keywords

Wind Speed Forecast; Generative Modelling; Convolutional Neural Networks.

Resumo

Tavares, Felipe Whitaker de Assumpção Mattos; Cyrino Oliveira, Fernando Luiz; Vellasco, Marley M. Bernardes Rebuzzi. **Geração de Cenários de Velocidade do Vento no Curto Prazo no Brasil com Redes Adversárias Generativas Melhoradas**. Rio de Janeiro, 2024. 57p. Dissertação de Mestrado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

A variabilidade das fontes de energia renovável, como energia eólica, apresenta um desafio significativo para o operador do sistema elétrico, em especial para o médio prazo (de horas a dias à frente). Isos porque é um período crítico para tomada de decisões do setor, sendo influenciado tanto por dados recentes quanto por padrões mais amplos. O atual estudo propõe a utilização de uma rede convolucional para gerar cenários para as componentes u- (latitudinal) e v- (longitudinal) do vento, utilizando o algoritmo Redes Adversárias Generativas Condicionais para treinamento. O modelo gerador proposto foi comparado com o estado da arte para previsão meteorológica, um sistema de previsão numérica. Os resultados mostram que o modelo - tendo um custo computacional inferior, menos informações de entrada e estabilidade de longo prazo similar - foi capaz de superar o *benchmark* em um quarto dos meses do conjunto de teste na previsão de duas semanas à frente (28 passos de 12 horas). Além disso, as medianas das séries geradas são estatisticamente iguais às previstas pelo estado da arte em 71.97% dos casos.

Palavras-chave

Previsão de Velocidade do Vento; Modelagem Generativa; Redes Neurais Convolucionais.

Table of contents

List of figures

List of tables

List of Abreviations

NN	Neural Networks
CNN	Convolutional NN
GAN	Generative Adversarial Networks
cGAN	Conditional GAN
WGAN	Wasserstein GAN
SN-GANs	Spectrally Normalized GANs
NWP	Numerical Weather Prediction Systems

1 Introduction

The growing prominence of renewable energy sources has created an additional challenge to the grid operator: their variability. Wind power, for example, has this challenge for all time scales: from seconds, for the turbine control system; to minutes and weeks, for the integration of wind power in the electrical grid [1].

One important characteristic in time-series forecasting is its horizon, or how far in the future the model predicts. The medium-term wind speed forecasting period, which ranges from a few hours up to two weeks, presents an unique set of challenges. While short-term forecasts rely heavily on recent past data [2]; long-term forecasts, or weeks to months ahead, are influenced by broader trends, such as the El Niño–Southern Oscillation [3].

Therefore, the medium-term forecasting period is a delicate balance between these two extremes, which is particularly important for the electrical sector as it influences decision making [4, 5], specially if the forecasts accounts for uncertainty [6, 7].

Another aspect of forecasting, connected to the horizon, is its frequency, or step size. It is the time between consequent predictions, and can range from seconds to hours or months, depending on the application. For computationally heavy models, such as Numerical Weather Prediction (NWP) systems, the frequency and spatial resolution must be chosen carefully to balance its forecast value against cost.

Currently, these NWP systems are the state of the art of medium-term weather prediction, but recent developments in machine learning (ML), namely Deep Learning (DL), have rivaled its place by offering similar performance for much lower computational inference cost [8, 9].

Advancements made in image recognition [10], for example, have been applied to wind speed forecasting by treating its u- (longitudinal) and v- (latitudinal) components over a gridded area as channels of an image [11]. This perspective of predicting subsequent images over time is similar to the problem of generating video, with two (u- and v- components) instead of three channels (RGB). Additionally, advancements in generative learning [12, 13, 14, 15] can also be applied to weather forecasting, enabling scenario generation with lower computational cost.

The use of scenarios to deal with uncertainty is very common in the electricity sector. In Brazil's case, as it is common in other systems with significant hydropower, the water's future value is calculated and is at the center of decision making. This process is done using an optimization tool chain that takes into account variables such as precipitation and the different scheduling horizons [16].

However, unlike water that can be stored for months, wind power generation can vary rapidly due to changes in wind speed - which can not be stored -, making the inclusion of its uncertainty in shorter term models progressively more important. This is clear by wind power's generation growth between 2022 and 2023 of 13%, reaching 81.6 TWh: it already corresponds to 11.8% of the country's total energy supply, following hydropower (58.0%) and fossil fuels (15.7%) [17].

Different approaches to include wind's uncertainty have been studied. Maceira et al. [18], for example, adapts the current structure of energy equivalence of hydropower reservoir to wind power for the medium-term model, DECOMP [19]; which could also be expanded to include generation from other sources such as wind and solar, potentially considering their correlations [20].

Nevertheless, there still lacks a method that includes scenarios while considering wind speed's spatial and temporal correlations with similar performance and lower cost than NWP systems.

Therefore, considering recent advancements made in Machine Learning, this work proposes the generation of wind speed and direction scenarios to support the medium-term operation. This is done by training a fully convolutional neural network to generate subsequent scenarios from the current, one step at a time. The model is trained using data from the European Center for Medium-Range Weather Forecast's (ECMWF) most recent reanalysis, ECMWF's ReAnalysis 5 (ERA5). This general approach is then applied in Brazil, considering three characteristics of the energy market: the growing influence of wind power; the trend of considering scenarios for shorter horizons; and an Independent System Operator (ISO) that frequently deals with the consideration of scenarios.

This work has been organized in the following chapters: Chapter 2 presents the theoretical background for wind speed forecast and scenario generation; Chapter 3 gives context on recent developments in Deep Learning with successful applications from other areas; Chapter 4 describes the input variables and data sources; Chapter 4.2 presents the results found; and finally Chapter 5 summarizes this work with final remarks.

2
Literature Review

Forecasting wind speed and direction has been the object of various studies, many focused on its application for forecasting energy generation. Table 2.1 offers an overview of existing works on wind speed forecasting and scenario generation, organized in increasing order of expected model complexity. There are a few takeaways from this table: most studies that focus on short-term forecasts suffer from quick degradation of performance over time; few studies have explored the wind speed's spatial correlation; and it is uncommon to consider scenarios, which are essential for planning and decision-making.

Table 2.1: Related papers

	Data	**Model**	**Summary**
Nielsen et al. [21]	-	Proposes a model called New Reference Persistence as an update to the Persistence (Naive) model, modifying it to approach the long term average; therefore, extends indefinitely	Although simple, it is shown that it can beat the state of the art (NWP) for up to 4 hours [22]

Continues on next page

Table 2.1 – Continued from previous page

	Data	Model	Summary
Erdem and Shi [23]	Wind speed [m/s] and direction [°] are collected from sensors (with anemometers and wind direction) placed in North Dakota (USA) from May 1st to October 21st, 2002; averaging high frequency values to hourly frequency	Three approaches were compared: each wind speed component was modeled by an ARMA process, wind speed and direction were each modeled by another ARMA, and a Vector auto-regressive to forecast the tuple of wind attributes; all evaluated using MAE	Because wind speed and direction are slightly correlated, Vector auto-regressive (VAR) model had a higher forecasting accuracy
Brown et al. [24]	Sources sensor data from Pacific Northwest, Goodnoe Hills and Washington for December 1981, averaging it to hourly	Uses BIC to evaluate an autoregressive process. Chooses an AR(2) to make forecast for up to 3 hours, noting rapid performance degradation on scenarios	Forecasts wind speed, taking into account its autocorrelation, non-Gaussian distribution and diurnal nonstationarity; later interpolates it to the turbine's height to transform into power using the generator power curve

Continues on next page

Table 2.1 – Continued from previous page

	Data	Model	Summary
Pessanha et al. [25]	Sources half-hourly wind speed [m/s] from NWP forecasts from Sintegre for the days 8th through 10th of February 2021	Fits a Weibull distribution using Generalized Additive Model for location, scale and shape (GAMLSS), generating 2,000 scenarios for up to 72 hours	Generates wind speed scenarios (at 100m) for two equivalent wind farms (EWFs), transforming it into power using the turbine power curve
Liu et al. [26]	Sources different climate variables, including wind speed, from the National Renewable Energy Laboratory (NREL) for 304 wind turbines, interpolating to a grid.	Models - Persistence, Lasso Regression, Neural Networks (Feed Forward, LSTM, CNN, STNN-VB), Gaussian Processes (GPR) and Hidden Markov Chains - were tested and evaluated using RMSE and CRPS for up to 3 hours of forecast horizon.	Spatial–Temporal Neural Network and Variational Bayesian inference (STNN-VB) has the best performance (14.1% lower RMSE)

Continues on next page

Table 2.1 – Continued from previous page

	Data	Model	Summary
Bastos et al. [11]	Two-hourly with fixed spatial resolution (0.5° and 0.204°) datasets for different areas containing u- and v-components of wind, and temperature from the Climate Forecast System Reanalysis (CSFR) dataset were sourced from the Research Data Archive (RDA).	Uses a two part network - an UNet [27] to extract spatio-temporal features; and a CNN with a dense network head to map those features to single sites - to model 181 closely related different hourly series, with prediction horizon of up to 6 hours.	The results show that adding calendar variables considerably improves performance of the models. It suggests the use of Recurrent Neural Networks (RNNs) to model such sequential problem; and testing a fully convolutional model.
Jiang et al. [28]	Collects data from NREL Wind Integration Dataset, which provides power data for more than 126.000 sites in the United States of America with 5 minutes temporal resolution. Selects wind farms in Washington State for its experiment.	Uses a convolution model based on image generation articles. It maps noise $z \sim \mathbf{U}[0,1]$ sized 128 to a 24 by 24 grid, correspondent to the series; and uses the Improved GAN [14] algorithm for training.	Generates scenarios for a number of series, considering cross correlation due to the network structure.

Continues on next page

Table 2.1 – Continued from previous page

	Data	Model	Summary
Lam et al. [9]	Sources 39 years (1979-2017) of ERA5 weather data.	Trains a Graph Neural Network forecast hourly weather variables for over 10 days at 0.25° resolution globally to minimize MSE weighted by vertical level.	Machine Learning-based Weather Prediction (MLWP) are now competitive with traditional weather forecasting methods. Additionally, its method also performed well in severed event forecasting.

The remainder of this chapter delves into various techniques applied to weather forecasting: Section 2.1 briefly overviews Numerical Weather Prediction (NWP) systems, which are the current state-of-the-art of weather forecasting; and Section 2.2 discusses Deep Learning developments.

2.1 Numerical Weather Prediction

Numerical Weather Prediction (NWP) systems rely on complex mathematical models to simulate the behavior of the atmosphere and forecast future weather conditions. The main difference between models is whether the atmosphere is simulated alongiside changes from the ocean (atmosphere-ocean coupling). Its pipeline includes several key steps [29]:

1. Identifying sources of observational data and ensuring high quality;
2. Combining observations with model simulation to produce a best estimate of the current weather situation, called data assimilation;
3. Convert the continuous atmosphere into a grid-based system for computational efficiency, named discretization in space and time;
4. Physical parametrization to represent atmospheric processes, such as clouds, using simplified mathematical schemes, which are tested and validated for accuracy;
5. Provide initial and boundary conditions of the atmosphere;

6. Post-process the model's output to refine and produce usable weather forecasts; and
7. Quantify the forecast uncertainty by running the simulation with slightly different initial conditions, generating an ensemble of forecasts.

Although its performance has improved over the years due to improvements in either: numerical techniques, model resolution or physical process parametrization schemes; it comes at a high computational price [30]. This brings attention to other methods, such as machine learning and recent deep learning methods, for their flexibility and lower inference cost [8].

2.2 Deep Learning

Before considering Deep Learning applications in weather, this section gives a brief description of its development and architectural breakthroughs. For that, Section 2.2.1 describes Neural Networks and its parameter optimization, followed by an explanation on Convolutional Neural Networks. Then, in Section 2.2.2, Generative Adversarial Networks (GANs) [12] are presented and discussed.

2.2.1 Background

Feed Forward Neural Networks are stacked learners - each layer use the output representation from previous ones -, combined with non linear functions, represented in Equation 2-1, which enables the modeling of arbitrary transformations [31].

$$f_\omega = \sigma(W_{N+1} \cdot \sigma(W_N \cdot \sigma(\ldots) + \beta_N) + \beta_{N+1}) \tag{2-1}$$

Where W_i and β_i are the weights and intercept of each layer i, respectively. The non linear function is represented by σ. Each layer is a linear combination of the previous layer's output, X_{i-1}, or $W_i \cdot \sigma(X_{i-1}) + \beta_i$, which is then fed forward to the next one.

This comes at the cost of not having an analytical solution, requiring a flexible algorithm such as back propagation [32, 33], which minimizes the model's loss ($\mathcal{L}$) by updating its weights ($W_i, \beta_i \forall i$) at each iteration step, weighted by the learning rate (α). The update process of weights (W_i) for each layer (i) is represented in Equation 2-2.

$$\hat{W}_i = W_i - \alpha \cdot \frac{\partial \mathcal{L}}{\partial W_i} = W_i - \alpha \cdot \frac{\partial \mathcal{L}}{\partial W_{i+1}} \cdot \frac{\partial W_{i+1}}{\partial W_i} \tag{2-2}$$

Recent developments - namely computational power and data availability - have enabled neural networks to become deeper (higher N, or more stacked layers), baptized Deep Learning. Although this allows for more complex pattern learning, the depth aspect makes training harder [34], as it introduces two main problems: over fitting, as having more parameters augment the network's flexibility; and vanishing or exploding gradients, which make training these networks very challenging.

Regarding over fitting, a possible solution is to use regularization. An example is a technique called Dropout layers: a layer that "turns off" neurons randomly with set probability (p), forcing the network to learn different useful representations of the function, instead of leaning on few neurons [35], as represented in Equation 2-3. In turn, Dropout layers can also be used to model uncertainty: by using dropout during evaluation time, different neurons are activated, resulting in different predictions [36].

$$\mathbb{I}_i[z \geq p] = \begin{cases} 1, & \text{if } z_i \geq p \\ 0, & \text{if } z_i < p \end{cases}, \quad z_i \sim \mathcal{U}(0,1)$$

$$f_\omega = \sigma(W_N \cdot \mathbb{I}_N[z \geq p] \cdot \sigma(X_{N-1}) + \beta_N) \tag{2-3}$$

Where z_i is an uniformly sampled vector with the same size as W_i, X_{N-1} is the output of the previous layer, and $\mathbb{I}[z \geq p]$ is the indicator function, assuming 1 if $z \geq p$ or 0 otherwise.

For the second problem, related to gradient propagation, a solution is to use skip connections, which enables the network to progressively learn more complex representations of the data [37], while also having the benefit of smoothing the loss landscape [38], improving convergence.

Due to its flexibility, Neural Networks have been successfully applied to various previously open problems. One such area that was revolutionized is image recognition, which had a turning point after the development of large-scale Convolutional Neural Networks (CNNs) [10]. These types of networks take into account proximity information by sliding a filter over the input. This filter enables the recognition of simple features, such as edges in an image; and has been shown to be able to activate for more complex patterns, such as eyes or faces, when enough filters are stacked [39]. Equation 2-4 shows the result of sliding the filter, or kernel (matrix of learnable parameters), over the input (original "image"); with no padding or dilatation, and a stride of one.

$$\text{Result}_{(0,1)} = 2 \cdot 0 + 3 \cdot 1 + 6 \cdot 1 + 7 \cdot 0 = 9 \tag{2-4}$$

$$\begin{array}{ccc} \text{Input} & \text{Kernel} & \text{Result} \\ \begin{bmatrix} 1 & \mathbf{2} & \mathbf{3} & 4 \\ 5 & \mathbf{6} & \mathbf{7} & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} & \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} & \begin{bmatrix} 7 & \mathbf{9} & 11 \\ 15 & 17 & 19 \\ 23 & 25 & 27 \end{bmatrix} \end{array}$$

On top of being used for classification, convolutional neural networks have been successfully applied to other problems, such as image segmentation, or the recognition of what and where objects are in an image. A successful approach for this problem is the UNet, which uses two techniques described above: convolutional layers and skip connections [27].

2.2.2 Generative Adversarial Networks

While neural networks were developed as discriminative models for regression or classification tasks, their learning flexibility has also been used to learn data distributions. One of the most successful algorithms for this are Generative Adversarial Networks (GANs).

GAN is an algorithm for training two competing networks: a Generator (G), which learns the mapping between an arbitrary distribution, usually Gaussian, to real data's distribution; and a Discriminator (D), usually a binary classifier, which learns to discriminate between true and generated examples as accurately as possible [12, 15]. It was developed as a min-max optimization problem between two models, represented by Equation 2-5; and each network's loss is shown in Equations 2-6 and 2-7.

$$\min_G \max_D V(D,G) = \mathbf{E}_{x\sim p_x(x)}[\log D(x)] + \mathbf{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (2\text{-}5)$$

$$\mathcal{L}_D = \mathbf{E}[\log D(x)] + \mathbf{E}[\log 1 - D(G(z))] \quad (2\text{-}6)$$

$$\mathcal{L}_G = \mathbf{E}[\log D(G(z))] \quad (2\text{-}7)$$

Where x is the input, $z \sim \mathcal{N}(0,1)$ is the noise vector, and $\mathcal{L}_D$ and $\mathcal{L}_G$ are the network's respective losses.

While initially GANs were primarily used to generate random reasonable instances of the true data, further developments were able to condition the process to generate different examples of a similar instance. An example of this process is the generation of MNIST images (28×28 sized images of hand written digits) given a label (y, e.g. $y = 1$), with very good results [13]. The modified networks' losses are represented in Equations 2-8 and 2-9. There is no current consensus on the best way to condition the network, but it is known that is should not introduce too much noise, nor feed an input that is too close to the output; as the network might either not learn anything or simply reproduce the condition, respectively.

$$\mathcal{L}_D = \mathbf{E}[\log D(x|y)] + \mathbf{E}[\log(1 - D(G(z|y)))] \tag{2-8}$$

$$\mathcal{L}_G = \mathbf{E}[\log D(G(z|y))] \tag{2-9}$$

GAN's training algorithm has an important caveat: the training process is known to be very unstable. Initially, it was suggested to sparsely train the Generator to improve stability, alternating between which model's weights are updated in each batch [12]. This was done because both a weak or a strong Discriminator provide poor signal to the Generator. However, balancing the networks and selecting the appropriate hyperparameters can be difficult.

Another suggestion to stabilize the networks' training was progressively growing the networks: both the Generator and Discriminator start training from a lower resolution image and are progressively grown to the desired resolution [40]. This idea has culminated in a two part Generator network - mapping and synthesis - capable of separating high-level attributes (e.g. pose when trained on human faces), which enables intuitive control for image conditioning [41]. As promising as this idea was, it was later put aside in favor of skip connections [42].

Another approach to improve training is to modify what the Discriminator is predicting: instead of a probability of true or fake, it could now output the distance between true and estimated probability densities, namely the Earth Mover distance. For this problem to be valid, it was shown that the Critic (Discriminator's new name) must be a 1-Lipschitz function (a function with norm below 1), which, initially, was done by clipping its weights, which the authors recognized strongly limited the Critic's learning capability [14]. Although the 1-Lipschitz property is hard constraint, the Generator's loss gains meaning (it becomes the estimated distance between the true and learned distributions) and guarantees convergence of the training process. Moreover, the better the Critic is, the better signal is provided to the Generator, improving the learning process. This creates an incentive to train the Critic a few times (*ncritic*) before providing a signal to the Generator. The updated loss functions for the Critic (Discriminator) and Generator are Equations 2-10 and 2-11

$$\mathcal{L}_D = \mathbf{E}[D(G(z)] - \mathbf{E}[D(x)] \tag{2-10}$$

$$\mathcal{L}_G = -\mathbf{E}[D(G(z)] \tag{2-11}$$

Different methods to approximate 1-Lipschitz property were suggested. Lee and Seok [43] classifies them between regularization or normalization methods. One such method is the inclusion of a gradient penalty (GP) regularization in the critic loss function [44]. This has the benefit of allowing the Critic to model more complex functions, which is an important limitation of weight clipping method. The updated loss is represented in Equation 2-12. This substitution improved the performance of the critic and the overall training process. However, this requires one entire forward and backward propagation to be calculated [45], increasing computational cost.

$$\begin{aligned} \tilde{x} &= \alpha \cdot x + (1-\alpha) \cdot G(z), \quad \alpha \sim \mathcal{U}(0,1) \\ \mathcal{L}_D &= \mathbf{E}[D(G(z))] - \mathbf{E}[D(x)] + \mathbf{E}[(\|\nabla_{\tilde{x}} D_\omega\|_2 - 1)^2] \end{aligned} \tag{2-12}$$

Another alternative is to directly scale each the Critic's weights to have norm $K = 1$. This can be done by dividing each layer's (i) weights by $\sigma(W)_i$, as in Equation 2-13. The scaler $\sigma(W)_i$ is approximated with an iterative heuristic until reaching the desired value of K [45]. Although this could be expensive - it is an iterative process taht may take time to converge -, good results were achieved with as little as one iteration per back propagation update, which reduces the impact on training computational cost [45]. A caveat on scaling the weights is that it also limits back propagation, as weights' norm become limited. A suggestion to circumvent that is to increase the learning rate.

$$\tilde{W}_{SN} = \frac{W_i}{\sigma(W)_i} \tag{2-13}$$

Therefore, given the presented problem, previously applied methods and recent developments; the following chapters, 3 and 4, present the methodology and case study, respectively.

3
Methodology

This work proposes a fully convolutional neural network to generate wind speed scenarios auto-regressively. It was inspired by Bastos et al. [11]'s approach of treating wind speed's components over an area as the channels of an image. In other words, this work interprets wind speed's u- and v-components over a spatial grid (latitude and longitude), as a two channel image, with dimensions (components, latitude, longitude) instead of (channel, height, width).

Figure 3.1 presents an overview of how this Chapter is organized: Section 3.1 introduces how data is preprocessed for the model; Section 3.2 describes the deterministic and stochastic approaches, their architectures, parts and training algorithms; finally Section 3.3 determines the benchmark used and 3.4 contains the evaluation process used.

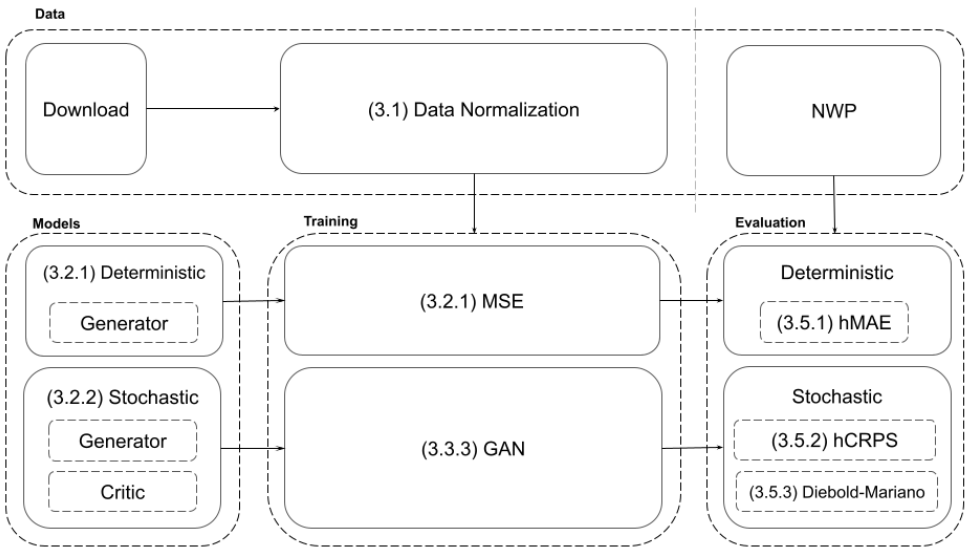

Figure 3.1: Methodology Overview

3.1
Data Normalization

The data is divided in three sets: train, validation and test; keeping years complete (always from January through December). After the split, each combination of latitude (h) and longitude (w) (borrowed from image generation literature as height h and width w) is normalized following the approach below, estimated sequentially from the train set.

1. First, the data is scaled between [0, 1] in a **MinMax** fashion; then
2. The data is **Rescaled** to the interval [-1, 1].

The combination of these steps is summarized in Equation 3-1. They are described above separately to mirror the implementation, which has them separate for testing purposes. This strategy was chosen since it keeps the data's distribution and expected signal: for the u-component, east to west is positive, while west to east is negative.

$$\hat{y}^{h,w} = 2 \cdot \frac{y^{h,w} - m^{h,w}}{M^{h,w} - m^{h,w}} - 1 \tag{3-1}$$

Where $\hat{y}_t^{h,m}$ and $y_t^{h,m}$ correspond to the transformed and original values, respectively; while $M^{h,w}$ is the maximum and $m^{h,w}$ the minimum at location (h, w).

Finally, the inverse transformation is used to transform the model's prediction to the original variable domain.

3.2 Models

Considering how unstable and costly the GAN training algorithm is, this work first trains a deterministic generator in order to set the architecture and hyperparameters, and then develops upon it to generate scenarios. This is reflected in how this section is organized: first, sub Section 3.2.1 contains the deterministic model, its parts and training algorithm; then sub Section 3.2.2 describes the Generator and Critic, noise injection and adapted GAN algorithm.

3.2.1 Deterministic

The deterministic Generator's architecture is represented in Figure 3.2. For input, the Generator receives: date and time information, passed through the encoder described in Section 3.2.1.1; and $(u, v)_t^{(h,w)}$. After processing date and time information and concatenating the inputs on the channel dimension, its result - both components alongside hour and month channels - is fed into an UNet [27] to predict $(u, v)_{t+1}^{(h,w)}$. During evaluation, this prediction is fed back to the model with an update to the date and time feature, forecasting the horizon auto-regressively.

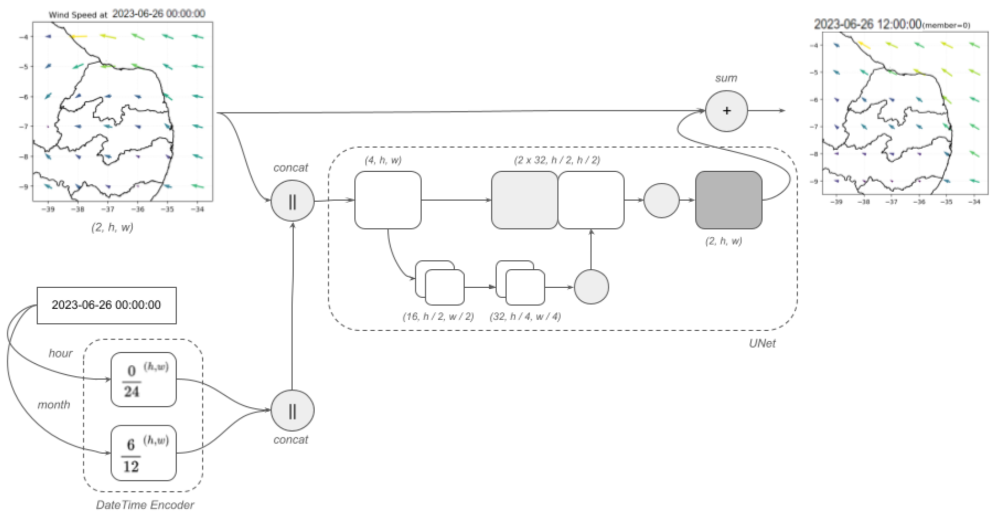

Figure 3.2: Visual Representation of the Deterministic Generator's Architecture

The model contains two parts: an encoder for date and time information, described in Section 3.2.1.1, and an UNet [27]. This combination was chosen because it has shown good performance previously [11]. Moreover, skip connections was shown to provide potentially higher performance [42], on top of requiring less logic to implement, than the alternative of progressively expanding the network [40].

Moreover, the activation function chosen was the hyperbolic tangent since it has been previously applied in weather prediction [46]. An important note is that the input is directly added to the output, making the model predict the first difference of the series [9].

3.2.1.1 Date time Encoder

In order to enable the networks to learn date and time related patterns, the network receives epoch seconds (dt), which is transformed linearly to $\{H, M\} \in [0,1]^2$ interval representing hour (H) and month (M). The implementation uses integer arithmetic (*mod* and *floor* operators) to extract the hour (H) and month (M), dividing it by the maximum value (24 and 12, respectively), resulting in values in the chosen range. For example, date 2012-03-04 04:00:00 is transformed into $H = \frac{4}{24} = \frac{1}{6}$ and $M = \frac{3}{12} = \frac{1}{4}$.

Having the values for H and M (for the given time step τ), two matrices are built to have the same dimensions as $(u,v)_t^{(h,w)}$, and are concatenated as a channel, becoming a tensor like $(u,v,H,M)_t^{(h,w)}$.

3.2.1.2
Training

The Algorithm 1 is used to train the deterministic model. Overall, it iterates over batches of examples and learns to minimize MSE using back propagation [33].

Algorithm 1: Let Deterministic Generator (G) parameters be θ. Let wind components $(u, v)_t^{(h,w)}$ be shortened to y_t, where height h and width w are latitude and longitude, respectively; t be the current time step, T be the prediction horizon, $\alpha_\tau = \frac{1}{T}$ the loss weight for each step.

```
Hyperparameters:
batch size $m = 64$, $N_{iter} = 100$, $T = 4$; learning rate: $\nabla_\theta = 10^{-4}$ [44],
Adam's $\beta_1 = 0.0, \beta_2 = 0.9$ [44],
Data:
wind components $y_t$ are normalized, $y_t \in [-1, 1]^{(h,w)}$
datetime $dt \in \mathcal{N}$ in seconds
horizon $T$ as the amount of steps to be optimized for
1 for $N_{iter}$ training iterations do
      /* optimize through horizon $T$ */
2     for $\tau = 0, \ldots, T$ do
3         $\hat{y}_{t+\tau+1} \leftarrow G_\theta(dt, y_{t+\tau})$
4         $\mathcal{L}_{MSE} \leftarrow \mathbf{E}\Big[(y_{t+\tau+1} - \hat{y}_{t+\tau+1})^2\Big]$
5         $\mathcal{L}^{\tau}_{G_\theta} \leftarrow \mathcal{L}_{MSE}$
6     end
7     $L_{G_\theta} \leftarrow \Sigma^{T}_{\tau=0} \alpha_\tau \mathcal{L}^{\tau}_{G_\theta}$
      /* Update the model's weights */
8     $\theta \leftarrow \theta - \nabla_\theta Adam(L_{C_\theta}, \theta, \beta_1, \beta_2)$
9 end
```

3.2.2
Stochastic

Having found a good combination of hyper parameters, the deterministic model's UNet is modified to include a Noise layer, described in Section 3.2.2.1, after every layer of the expanding path, as shown in Figure 3.3. The motivation behind this is to first extract features (contracting path) and then generate a scenario (expanding path), similar to Karras et al. [41]. Importantly, the stochastic model is trained from scratch: the weights learned in the deterministic flow are not used.

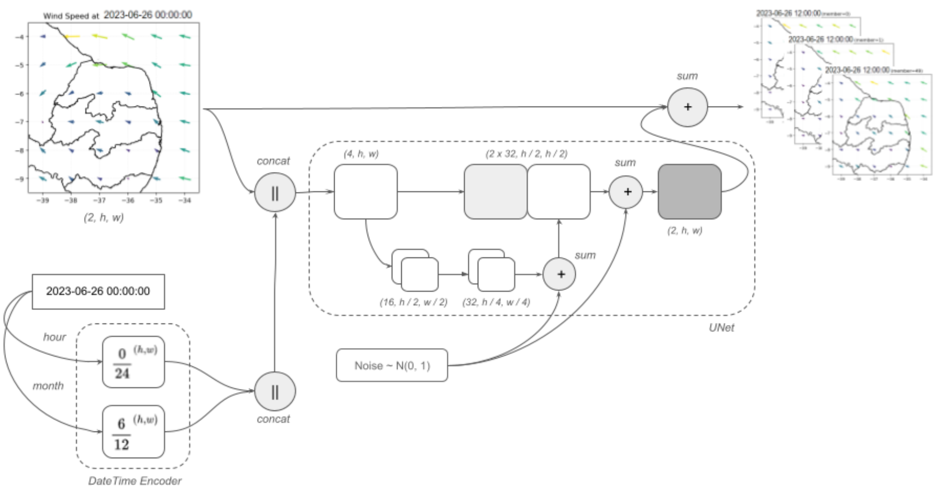

Figure 3.3: Visual Representation of Stochastic Generator's Architecture

3.2.2.1 Noise

To generate scenarios, the proposed model depends on receiving noise. This is done using a layer with a per channel parameter with the same shape as its channel input [41], as shown in Equation 3-2. The weights are initialized as zeros and are learned during training. This eliminates the need for specifying the size of the noise vector, as there is no conclusive evidence on its impact on the model [47].

$$c_i^{(h,w)} = (c_i + w_i \cdot z)^{(h,w)} \; \forall i \in \text{channels}; \quad w_i \in \mathcal{R},\; z^{(h,w)} \sim \mathcal{N}(0,1) \qquad (3\text{-}2)$$

Where: $(u, v)^{(h,w)}$ and $(\hat{u}, \hat{v})^{(h,w)}$ are the input and output of this layer, respectively; w_c is the layer's weights for each component ($c \in \{u, v\}$); and $z^{(h,w)}$ is "per pixel" white noise.

3.2.2.2 Critic

The Critic's role is to discern between true and fake data instances, providing signal for the generator to learn the data distribution. To ease implementation, Radford et al. [48]'s work was used as inspiration. It features a fully convolutional network that receives either real or fake input, alongside with the related condition, and outputs a classification for real or fake. Its visual representation can be observed in Figure 3.4. Considering the differences to this work's context, a few adaptations were made:

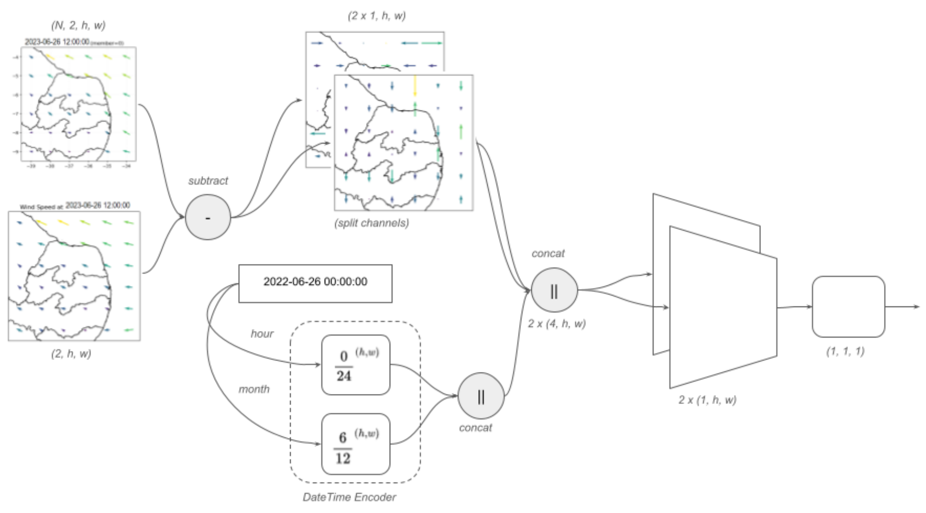

Figure 3.4: Visual representation of the Critic

- Each component is processed separately since it has empirically helped stabilize training; and
- Taking into consideration the training algorithm described in Section 3.2.2.3:
    - The Critic is updated *ncritic* times per batch, restricting the use of Batch Normalization. To substitute it, Instance Normalization is used;
    - DCGAN's Critic last activation function (*sigmoid*) [48] was removed and not replaced, since the algorithm transforms the classification problem into a regression one.

3.2.2.3 Training

Considering the developments explored in chapter 2, the Improved GAN algorithm [14] was chosen for its more stable training and interpretable loss function. A Mean Squared Error (MSE) was included in the Generator's loss function to improve learning. In order to adhere to the 1-Lipschitz restriction, Critic's weights are spectrally normalized [45]. Finally, the models are trained over an horizon T, equally averaging the losses [49].

The inputs for both networks are: the current step $(u, v)_t^{(h,w)} \in [-1, 1]^{h,w}$ wind components at time t with coordinates (h, w), and date and time variable dt (in seconds) as inputs.

The training flow is shown in Algorithm 2.

Algorithm 2: Proposed algorithm, following WGAN [14] and SN-GAN [45]. Let Critic (D) and Generator (G) parameters be ω and θ. Let wind components $(u, v)_t^{(h,w)}$ be shortened to y_t, where height h and width w are latitude and longitude, respectively; $T = \min(1, \lfloor \frac{\text{epoch}}{10} \rfloor)$ be the prediction horizon, $\alpha_\tau = \frac{1}{T}$ the loss weight for each step; γ_i the Generator's loss interpolation scalar use for the to the ith epoch.

```
Hyperparameters:
batch size $m = 64$, $N_{iter} = 200$, $N_{critic} = 5$, $T = 4$; learning rate:
$\nabla_\omega = 10^{-3}$ and $\nabla_\theta = 10^{-4}$, Adam's $\beta_1 = 0.0, \beta_2 = 0.9$, $\gamma_i = \frac{1}{3}$
Data:
wind components $y_t$ are normalized, $y_t \in [-1, 1]^{(h,w)}$
datetime $dt \in \mathcal{N}$ in seconds
horizon $T$ as the amount of steps to be optimized for
1  for $N_{iter}$ training iterations do
       /* train Critic */
2      for $i = 1, \ldots, N_{critic}$ do
           /* optimize through horizon $T$ */
3          for $\tau = 0, \ldots, T$ do
               /* generate next step with true previous step */
4              $\hat{y}_{t+\tau+1} \leftarrow G_\theta(dt, y_{t+\tau})$
               /* calculate fake loss */
5              $\mathcal{L}_{\hat{w}}^\tau \leftarrow \mathbf{E}\Big[D_\omega(dt_{t+\tau}, y_{t+\tau}, \hat{y}_{t+\tau+1})\Big]$
               /* calculate real loss */
6              $\mathcal{L}_{w}^\tau \leftarrow \mathbf{E}\Big[D_\omega(dt_{t+\tau}, y_{t+\tau}, y_{t+\tau+1})\Big]$
               /* calculate total critic's loss */
7              $\mathcal{L}_{D_\omega}^\tau \leftarrow \mathcal{L}_{\hat{w}}^\tau - \mathcal{L}_{w}^\tau$
8          end
9          $L_{D_\omega} \leftarrow \Sigma_{\tau=0}^{T} \alpha_\tau \mathcal{L}_{D_\omega}^\tau$
10         $\omega \leftarrow \omega - \nabla_\omega Adam(L_{D_\omega}, \omega, \beta_1, \beta_2)$
           /* normalize Critic's weights */
11         $\omega \leftarrow \frac{\omega}{\sigma(\omega)_i}$
12     end
       /* train Generator */
13     for $\tau = 0, \ldots, T$ do
           /* generate next step with true previous step */
14         $\hat{y}_{t+\tau+1} \leftarrow G_\theta(dt, y_{t+\tau})$
           /* approximate the distance between learned and
              true distributions */
15         $\mathcal{L}_{GAN} \leftarrow \mathbf{E}\Big[D_\omega(dt_{t+\tau}, y_{t+\tau}, \hat{y}_{t+\tau+1})\Big]$
16         $\mathcal{L}_{MSE} \leftarrow \mathbf{E}\Big[(y_{t+\tau+1} - \hat{y}_{t+\tau+1})^2\Big]$
17         $\mathcal{L}_{G_\theta}^\tau \leftarrow -\gamma_i \cdot \mathcal{L}_{GAN} + (1 - \gamma_i) \cdot \mathcal{L}_{MSE}$
18     end
19     $L_{G_\theta} \leftarrow \Sigma_{\tau=0}^{T} \alpha_\tau \mathcal{L}_{G_\theta}^\tau$
20     $\theta \leftarrow \theta - \nabla_\theta Adam(L_{C_\theta}, \theta, \beta_1, \beta_2)$
21 end
```

3.3 Benchmark

For the medium-term range, NWP systems are the state of the art. Therefore, it is used to benchmark our proposal.

3.4 Evaluation

Evaluation metrics allow direct comparison between models, enabling forecasters to choose the best model given a metric. Sections 3.4.1 and 3.4.2 present metrics related to deterministic and stochastic forecasts, respectively; while Section 3.4.3 describes the statistical test used to confront whether the forecast generated by our proposal is significantly different or not from the benchmark.

3.4.1 Mean Absolute Error

For point forecasting, models are usually evaluated by their Mean Absolute Error (MAE) or Mean Squared Error (MSE) [1]. Neither take into account multi-step prediction. In order to consider the prediction horizon, the arithmetic average of MAE is taken, named or horizon Mean Absolute Error (hMAE), described in Equation 3-3. Moreover, it is tracked of model learning during training, since it is simpler to calculate over batches.

$$hMAE = \frac{1}{(T \cdot \#h \cdot \#w)} \sum_{i=0}^{T} \sum_{\forall h,w} |y_i^{h,w} - \hat{y}_i^{h,w}| \tag{3-3}$$

Where T is the forecast horizon, h and w are the latitudes and longitudes, and $y_i^{h,w}$ and $\hat{y}_i^{h,w}$ are the true and predicted value, respectively.

3.4.2 Continuous Ranked Probability Score

In contrast to point forecasting, the evaluation of probabilistic forecasts should take into account the uncertainty information provided in the prediction.

This can be done by measuring the average distance between the cumulative distribution function (CDF) of the forecast and the observation, which can be interpreted as the expected value of the absolute error (MAE) between a random drawn sample from the forecast distribution and the observation. This metric is called Continuous Ranked Probability Score (CRPS).

$$\text{CRPS}(F) = \int_{-\infty}^{\infty} (F_f(y) - F_o(y))^2 dy \tag{3-4}$$

F is the cumulative distribution associated with:

- the forecast, f; and
- the empirical observation, o.

As an example, consider a probabilistic forecast of a normal distribution with mean 0 and standard deviation 1, and an observation of 0.5.

On another hand, considering its probabilistic nature, the CRPS of a normal distribution returns a value of 0.234. This means that the average absolute error of the forecast is 0.234 units. On one hand, if the forecast's probabilistic nature is not considered, one can take its average ($\mathbf{E}[\mathcal{N}(0,1)] = 0$) and calculate its MAE ($=|y - \hat{y}|=|0 - 0.5|= 0.5$), which is equivalent to the CRPS of a point prediction. Both are represented in Figure 3.5, respectively.

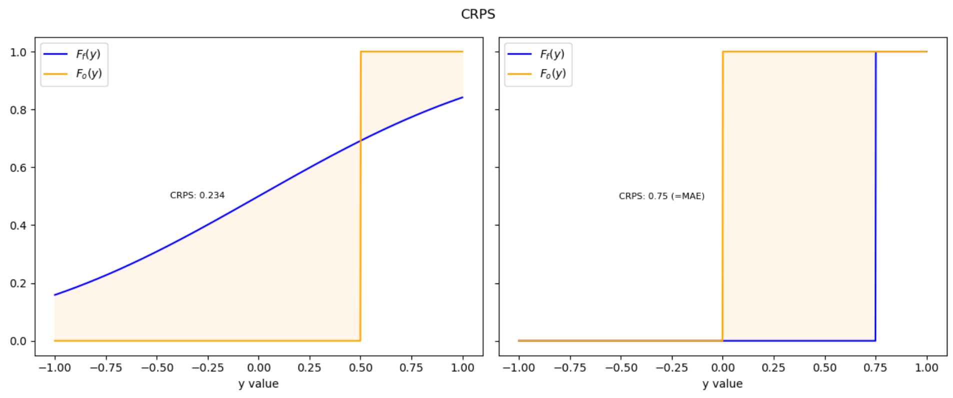

Figure 3.5: CRPS visualization example

Finally, since this work proposes multi-step forecasts, the same adaptation done for $hMAE$ is done for $CRPS$: after calculating $CRPS$ for each step τ, they are averaged, $hCRPS = \frac{1}{T}\Sigma_{\tau}^{T}CRPS_{\tau}$.

3.4.3 Diebold-Mariano test

In addition to directly comparing forecasts, it is important to understand whether these differences are statistically significant. A solution for this problem was proposed by Diebold and Mariano [50] (apud [51]) and later improved [52] (apud [51]). The Diebold-Mariano test steps are described in the following equations, as implemented by Hyndman and Khandakar [51]. Its null hypothesis is that both methods have the same forecasting accuracy.

1. calculate both methods', f and g, residuals

$$e_i = y_i - f_i \quad r_i = y_i - g_i \quad \forall i \in [1, n]$$

2. define d_i given a measure, such as MSE, which is commonly used. It is important to note that the test assumes d_i is stationary

$$d_i = e_i^2 - r_i^2$$

3. consider the autocorrelation γ_k at lag k, for $n > k \geq 1$

$$\gamma_k = \frac{1}{n} \sum_{i=k+1}^{n} (d_i - \bar{d}) \cdot (d_{i-k} - \bar{d}) \qquad \bar{d} = \frac{1}{n} \sum_{i=1}^{n} d_i$$

4. for $h \geq 1$, the test's horizon parameter, define Diebold-Mariano statistic. $w_k = 1 \forall k$. Under the assumption that $\mu = 0$, DM follows a standard normal distribution

$$DM = \frac{\bar{d}}{\sqrt{(\gamma_0 + 2 \cdot \sum_{k=1}^{\tau-1} w_k \cdot \gamma_k)/n}} \sim \mathcal{N}(0,1)$$

5. since the Diebold-Mariano test tends to reject the null hypothesis for small samples, Harvey et al. [52] proposed the following modification

$$HLN = DM \cdot \sqrt{(n + 1 - 2 \cdot \tau + \tau \cdot (T-1)/n} \sim t_{n-1}$$

Where y is the ground truth. The forecasts from model A and B are f and g, respectively. The forecast size is n and τ is the forecast prediction horizon (as in, how many steps ahead does the model predicts, $y_{t+\tau}$).

A caveat for the presented test is that, since the autocorrelation γ_k can be negative, the square root in DM's denominator can fail to be calculated. In such cases, Bartlett's weights ($w_k = 1 - \frac{k}{n}$) are used [50] (apud [51]).

Finally, since the proposed model forecasts one step at a time, parameter horizon τ is set to 1.

4
Case Study

This chapter presents a case study done for a delimited square in Brazil's northeast, chosen as it contains high wind potential and many of Brazil's wind farms - both built (yellow) and planned (blue) -, as depicted in Figure 4.1.

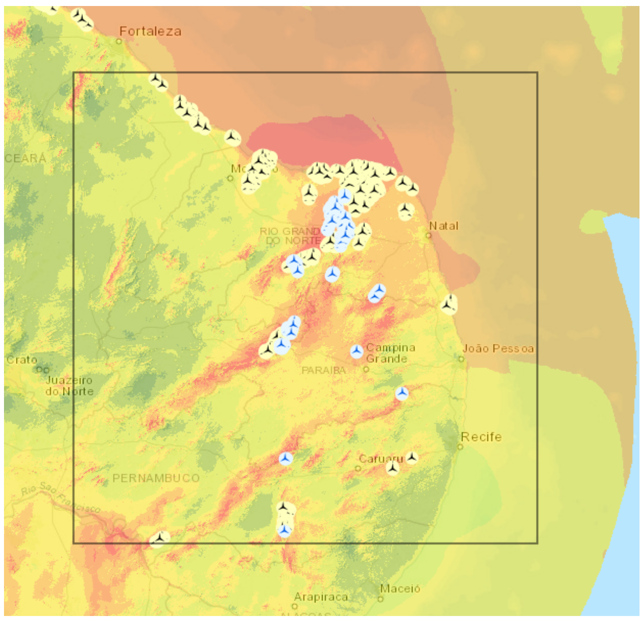

Figure 4.1: Northeast of Brazil Wind Farms with Wind Potential

Source: Empresa de Pesquisa Energética's Webmap

The rest of this chapter contains an overview about data used, Section 4.1; and results from applying the proposed methodology, Section 4.2.

4.1 Gathering Data

This study utilizes the u- (longitudinal) and v-components (latitudinal) of wind speed generated by Numerical Weather Prediction Systems (NWP, see Section 2.1) obtained from the Climate Data Store (CDS), an online repository for weather data. Specifically, two datasets were sourced, and are described below. Both are located in the northeast region of Brazil. Their coordinates are (-4.0, -39.0, -9.0, -34.0) for north, west, east and south, respectively; at the same pressure level, 1000 hPa, or approximately 100 meters above sea level, corresponding to the typical height for a wind turbine.

The datasets characteristics are described below:

– **Reanalysis** Data: contains hourly wind speed data at 0.25 degrees of spatial resolution ($\approx$ 25km original resolution) from the fifth European Center for Medium-Range Weather Forecasts (ECMWF) Re-Analysis (ERA5) [53]. In total 13 years of data were used: from 2011 to 2023 (10 years for training, 2 for validation and 1 for test); and

– **Forecast** Data: has 12-hourly temporal resolution and 1 degree spatial resolution and was sourced from ECMWF's Seasonal Forecast [54]. To account for uncertainties - for example, related to the initial conditions -, a number of members (scenarios) are available (in this case, 50 members).

Although they have different spatial and temporal resolution - *Forecast* has a much lower temporal and spatial resolution, mostly due to the computational cost of NWP systems [30] -, the comparison between them is valid since both come from the same institute and use similar, if not the same, data pipeline.

ECMWF's data was chosen because of its easy access combined to being the best suited data for wind modeling [55]. An important note is that, although reanalysis data are not observational measurements, it represents the true phenomenon, even in Brazil [56].

To further illustrate the data being used, Table 4.1 contains a data sample, and each row is referenced as $(u, v)_t^{(h,w)}$. For example, $(u, v)_\tau^{(-4,-39)} = (-2.67, -0.46)$ at τ = 2020-01-01 00:00:00.

Table 4.1: Example of the first five rows of data before processing

latitude	longitude	time	u	v
		2011-01-01 00:00:00	**-2.6660004**	**-0.45802307**
		2011-01-01 01:00:00	-2.4021454	-0.21806335
-4.0	-39.0	2011-01-01 02:00:00	-2.1911316	-0.009429932
		2011-01-01 03:00:00	-2.0295563	0.098861694
		2011-01-01 04:00:00	-1.817688	0.097579956

The rest of this chapter is divided as: the adjustments needed to match *Forecast* with *Reanalysis*, in Section 4.1.1; and an exploratory analysis over *Reanalysis* data, under Section 4.1.2.

4.1.1
Adjustments

Before exploring patterns in the data, the difference in spatial and time resolutions between *Reanalysis* and *Forecast* must be handled. Figure 4.2 has a colored dot at all the locations the have a series: it is clear that the *Forecast* set has a much lower spatial resolution than *Reanalysis*. Moreover, Figure 4.3 compares *Forecast*'s members (scenarios) as orange dots with the higher frequency *Reanalysis* data in blue.

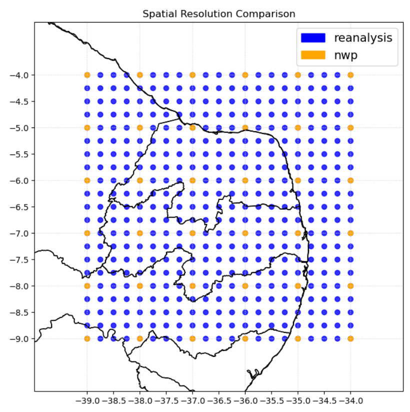

Figure 4.2: Spatial resolution comparison

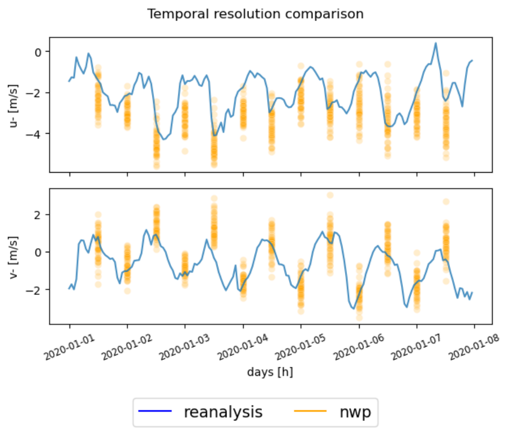

Figure 4.3: Temporal resolution comparison

This work chose to filter *Reanalysis* to match *Forecast*'s resolution, in order to enable comparison and evaluation this work's methodlogy - on top of reducing the computational cost. This choice must be kept in mind, since: it is not a limitation from the proposed methodology; and benefits the benchmark model, since *Forecast* was not restricted to the chosen area when it was produced.

4.1.2
General Characteristics

The data used in this work possesses a multi-dimensional structure with four indexes: component (either u or v), latitude, longitude and time. This complexity makes it challenging to visualize the data in a single image. To overcome this limitation, separate images are presented to show either temporal or spatial patterns, while maintaining the other dimension fixed.

One strategy for keeping the spatial dimension fixed is to calculate its spatial average, represented in Equation 4-1, and visualizing it over time.

$$(\bar{u}, \bar{v}) = \frac{1}{\#h \cdot \#w} \Sigma_{i,j}^{h,w} (u, v)_t^{i,j} \tag{4-1}$$

4.1.2.1
Time Patterns

As the representation of a physical phenomenon, wind speed is expected to exhibit a consistent average value. This is made evident in Figure 4.4, which displays the spatial average of hourly wind speed with a 4-week moving average. The figure also illustrates a clear seasonality pattern, with higher wind speed during late autumn and winter and lower during summer.

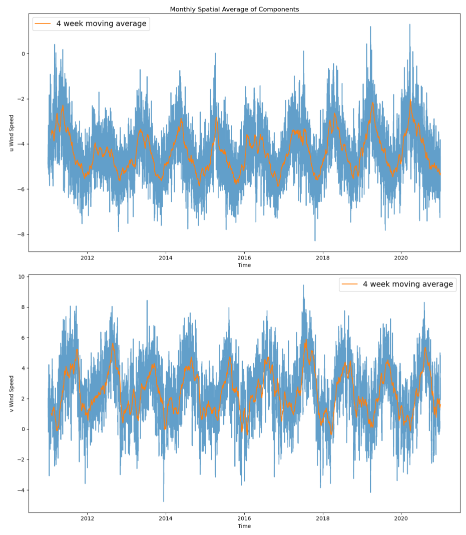

Figure 4.4: Hourly spatial average of the components over the years

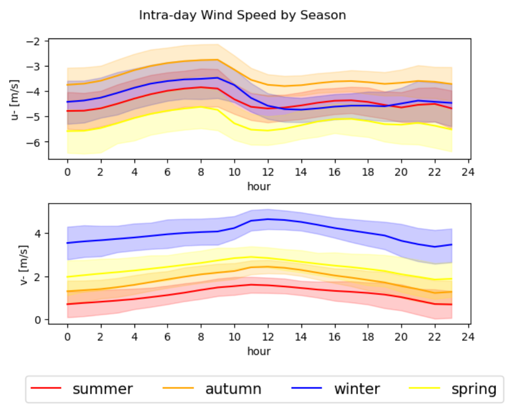

Figure 4.5: Spatial average of hourly components per season

The seasonal variation in wind speed becomes more apparent in Figure 4.5, which displays the spatial average of hourly wind speed for each season. The (southern hemisphere) seasons are named after the initial letter of their respective months: DJF (December, January and February) corresponds to summer, MAM (March, April and May) to autumn, JJA (June, July and August) to winter, and SON (September, October and November) to spring. The figure reveals that each season has a distinct bias, with spring (SON) having higher (more negative) u-component values and winter (JJA) with higher v-component values.

Figure 4.6 displays the Pearson autocorrelation, Equation 4-2, of each component on the map given a lag. This representation illustrates the temporal dependence, which slowly decays over time, becoming zero or even negative after 12 hours, and then becoming strongly correlated again on the next day (after 24h).

$$\rho(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sigma_x \sigma_y} \tag{4-2}$$

$$\phi^{h,w} = \rho(y_t^{h,w}, y_{t-\tau}^{h,w}) \tag{4-3}$$

$$\forall\, y \in \{u, v\}$$
$$\forall\, \tau \in \{1, 4, 12, 24\},$$
$$\forall\, h \in [-9, -4],$$
$$\forall\, w \in [-39, -34]$$

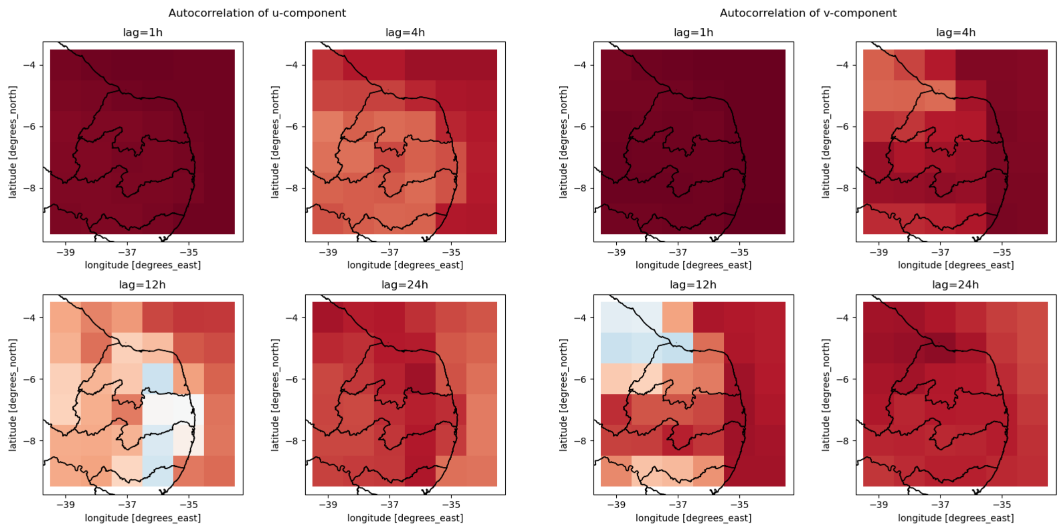

Figure 4.6: Wind components' auto-correlation on varying lags

The time dependence and correlation with the previous day is even more pronounced in Figure 4.7, which presents the speed autocorrelation all series over hourly lags. This type of plot usually considers a single series and is presented as an histogram; however, to allow patterns from all series to be visible, this Figure showcases the autocorrelation distribution over the lags. The magnitude of the wind speed (ω) was calculated with the Pythagorean theorem, Equation 4-4. Notice that it decays differently for each series, but clusters back together after 24 hours.

$$\omega_t^{h,w} = (u_{h,w}^2 + v_{h,w}^2)^{\frac{1}{2}} \tag{4-4}$$

4.1.2.2
Spatial Patterns

In addition to seasonality, wind speed patterns also vary based on the location of the series. To illustrate this, the calculated speed was separated in two groups: ocean and land. Then, the Pearson correlation, Equation 4-2, was

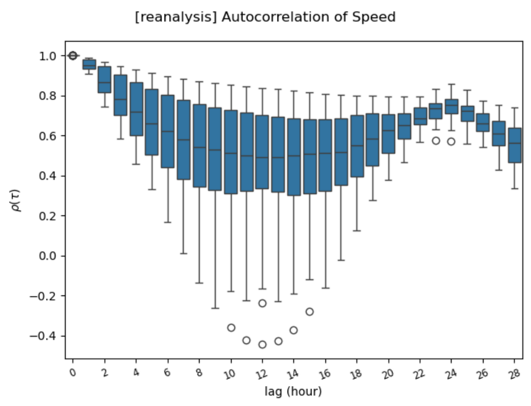

Figure 4.7: Partial Autocorrelation of series

calculated for all series in comparison to a fixed lagged series represented by the blue dot, one on the ocean and another on land, represented in Equations 4-5. Its result is shown in Figure 4.8, which highlights that the autocorrelation statistic is higher for series that are spatially closer to the reference series, with a faster decay observed for land compared to ocean, observable for 4-hour lag. Furthermore, the 24-hour lag (day before) autocorrelation is higher for land than it is for the ocean.

$$\begin{aligned} \phi^{h,w} &= \rho(\omega_t^{h,w}, \omega_{t-\tau}^{P}) \\ &\forall\ P \in \{(-35,-4),(-37,-7)\} \\ &\forall\ \tau \in \{1,4,12,24\}, \\ &\forall\ h \in [-9,-4], \\ &\forall\ w \in [-39,-34] \end{aligned} \tag{4-5}$$

Another perspective on the spatial dependence is displayed in Figures 4.9 and 4.10. It contains the same data as Figure 4.8, but shown over time instead of on the map. The proximity - measured using Chebyshev distance, Equation 4-6, a natural distance metric considering that convolutions are used in the model - strongly influences the series' decay, and its overall pattern. These figures expose how much more pronounced the pattern against the land series is in comparison to the ocean's.

$$D_{\text{Chebyshev}}(y^{h,w}, y^{P}) = \max_{c}\{|h - P_h|, |w - P_w|\} \tag{4-6}$$

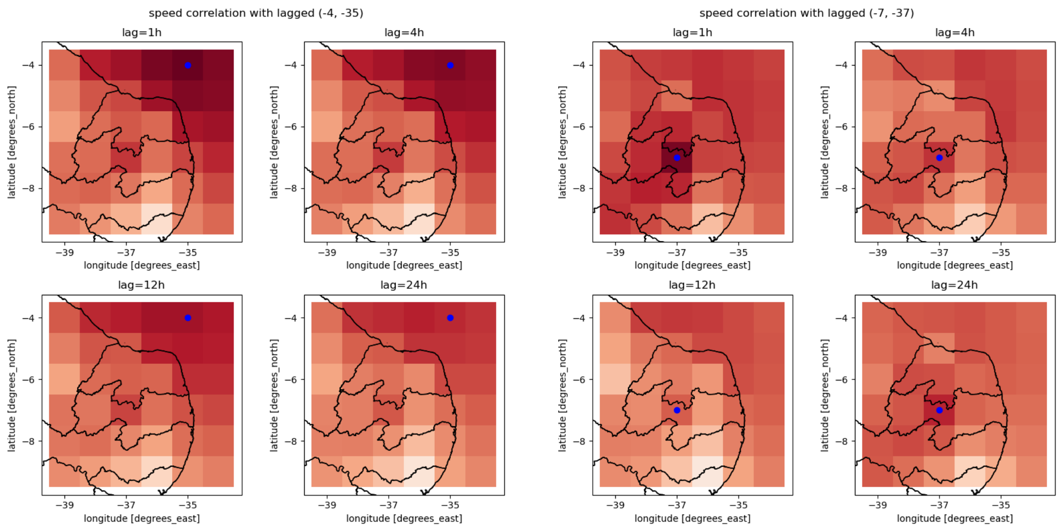
speed correlation with lagged (-4, -35)
speed correlation with lagged (-7, -37)
lag=1h
lag=4h
lag=12h
lag=24h
latitude [degrees_north]
longitude [degrees_east]

Figure 4.8: Correlation of wind speed with fixed series

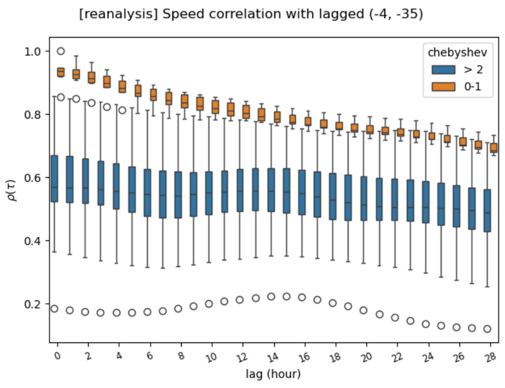

Figure 4.9: Ocean Speed ACF

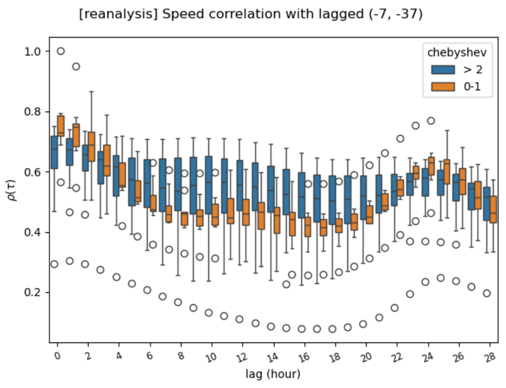

Figure 4.10: Land Speed ACF

4.2 Results

This chapter presents the results obtained with the proposed methodology. It is organized as follows: Section 4.2.1 presents the deterministic model's results; and Section 4.2.2 presents the results considering scenario generation.

For each section, the training metrics are presented. Then, a specific location was chosen to visualize and validate whether the model has learned patterns present in the series. Finally, the evaluation described in Section 3.4 is used to compare the proposal and benchmark.

Before presenting the results, it is important to highlight the differences between the proposal and the current state of the art in weather forecasting. The first difference relates to the computational cost: while Numerical Weather Prediction (NWP) systems solve differential equations, requiring supercomputers and expertise; the proposed method is restricted to learning all patterns from data and was run using consumer graded hardware. Therefore, while the spatial and temporal resolutions are restrictions from the benchmark, the methodology proposed can be applied on coarser resolution data - higher spatial and time resolutions. Besides, the results shown are not completely fair with our model: it was trained with data restricted to the chosen area, while the benchmark outputs were generated alongside and with access to other weather variables from entire world.

4.2.1 Deterministic Model

Figure 4.11 presents the training and validation loss curves for the model, alongside the hMAE, presented in Section 3.4, for both sets. Moreover, Figure 4.12 validates that the model has learned the data's distribution by comparing its predictions for an arbitrary month from the validation set to the same months from the training years and their average. It is interesting to note that, not only the model learned the 12-hourly pattern, but it seems to predict the long term average after a couple of steps, an improvement that has been previously suggested [21].

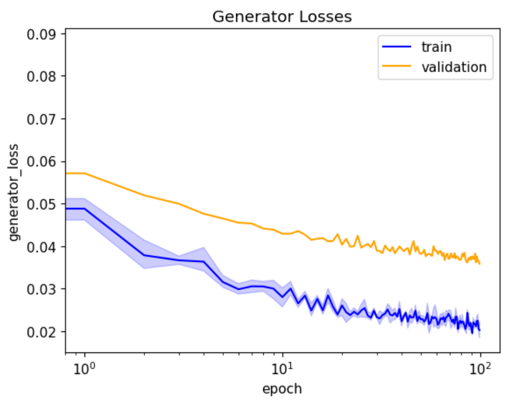

Figure 4.11: Training metrics of the Deterministic Model

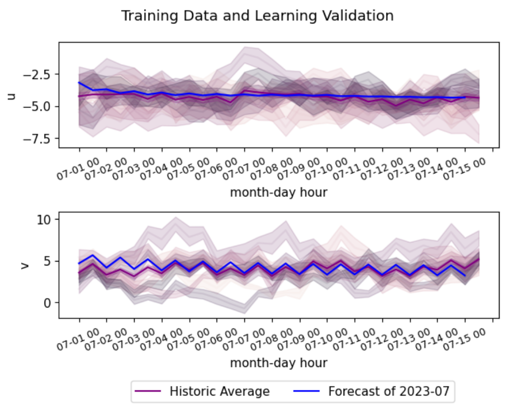

Figure 4.12: Validation of the model's learning

An example of the model's forecast is depicted in Figure 4.13. It is clear that, while both model forecasts and NWP's median correctly capture the true data's patterns, the model's forecasts could be improved. Still, neither come close to following the original hourly dataset variability, represented by the light black line.

Finally, Table 4.2 compares the deterministic model with the benchmark. Since the former does not learn the data's distribution, it is compared to the median of the latter's scenarios. While the benchmark's median performs better than the deterministic model, it is important to reiterate that it has

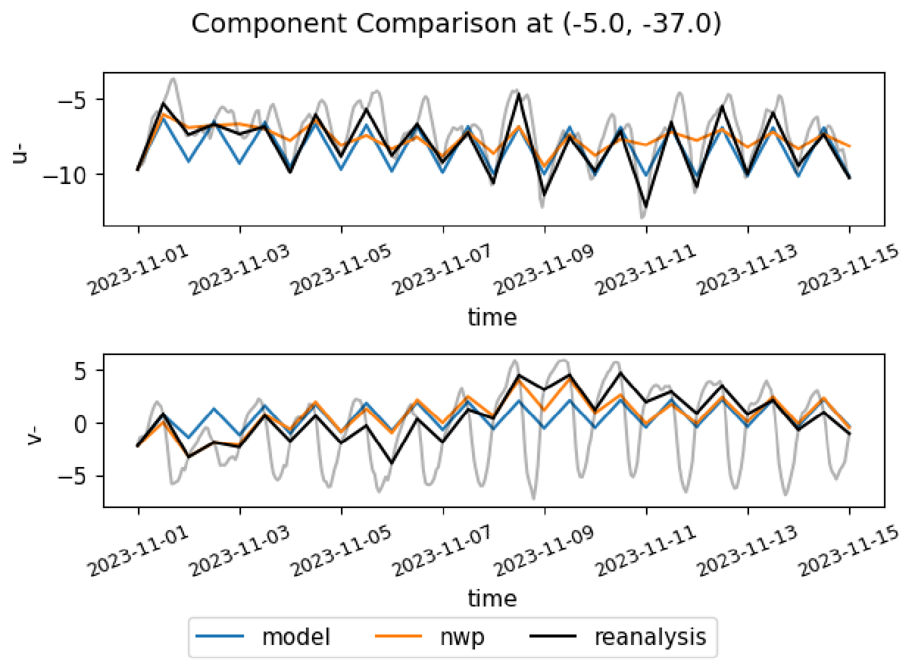

Figure 4.13: Forecast comparison

a much higher computational cost and access to more information than what was used in our model.

Table 4.2: Average of $hMAE$ for 28 steps (two weeks) for all test set months, per component. Values in bold are the lowest per component.

	Generator		median(NWP)	
month	u	v	u	v
JAN	0.926	**0.895**	**0.783**	1.075
FEV	**0.900**	**0.963**	0.969	1.02
MAR	**0.951**	**0.971**	0.978	1.167
APR	1.292	1.006	**1.000**	**0.922**
MAY	1.015	1.537	**0.914**	**1.300**
JUN	**0.865**	1.1	0.872	**0.942**
JUL	0.944	1.768	**0.882**	**1.316**
AUG	**0.891**	1.136	0.916	**1.131**
SET	0.98	1.929	**0.959**	**0.871**
OCT	0.812	1.109	**0.804**	**0.993**
NOV	**0.760**	1.491	0.826	**1.058**
DEZ	**0.726**	1.149	0.797	**1.091**

4.2.2 Stochastic Model

Having validated the architecture, the the Generator and Critic were trained. The following results were calculated using 100 members (scenarios), double the scenarios available from the benchmark (50). This allows for a better estimation of the evaluation metric without requiring more memory than available.

Figure 4.14 presents four plots: top left contains the Generator's losses: the train loss is blue, while orange is the weighted average between MSE (green, *val mse*, $1-\gamma = \frac{2}{3}$) and Critic's signal (red, *val adversarial*, $\gamma = \frac{1}{3}$) - scaled along y axes; the top right presents the Critic's loss (with inverted y scale, since the Critic's game is to maximize the Generator's loss), for both training (blue) and validation (orange); bottom left shows the hMAE; and bottom right keeps track of the Critic's norm, which reaches and stays close to 1 after the 100th epoch, as needed for convergence.

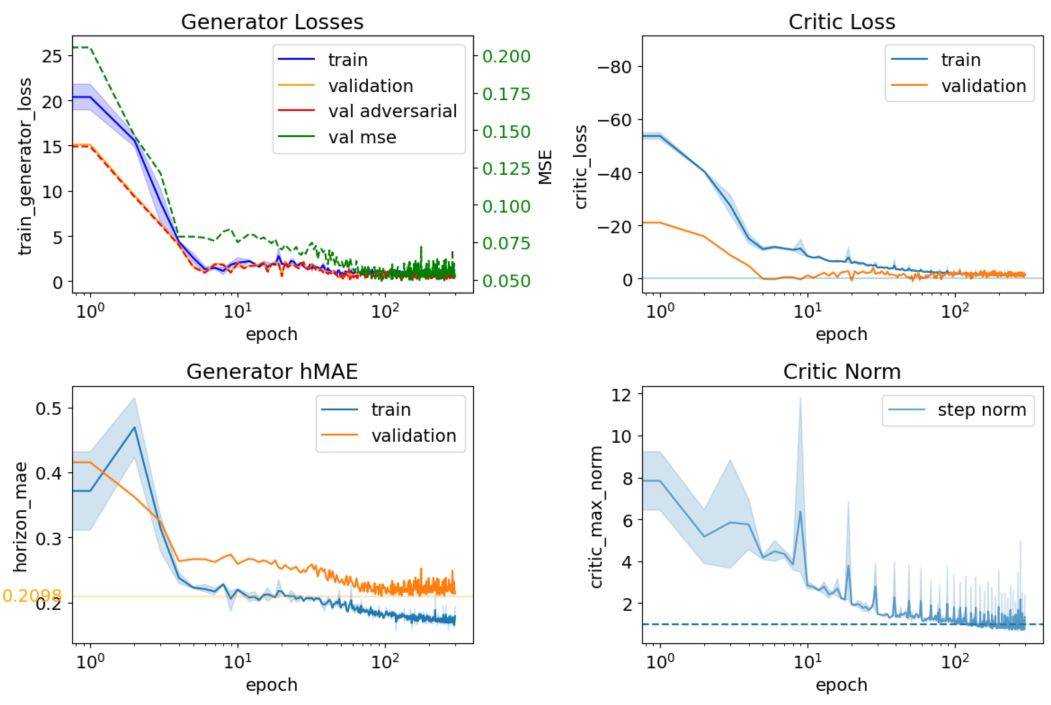

Figure 4.14: WGAN training metrics

In addition to validating the Critic's norm, it is expected that, under the WGAN algorithm, the model distributes its weight to better utilize them. This is shown in Figure 4.15 in comparison to Figure 4.16: although it did not fully utilize its capacity (some weights are very close to zero), the weights are distributed in a mostly smooth manner.

Another interesting visualization is the noise layer weight distribution, displayed in Figure 4.17. The legend identifies the weights by their processing order: the lowest the number, the closest it is to the input: the layer closest to the input has a flatter distribution than the one run closer to the prediction.

In order to compare the deterministic and the probabilistic model, Figure 4.18 shows the same location as Figure 4.13. From the image, it is clear that, not only the probabilistic model better forecasts the true data, it also captures

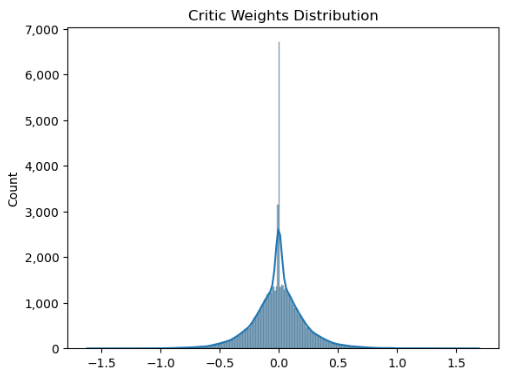

Figure 4.15: Untrained Critic's Weights Distribution

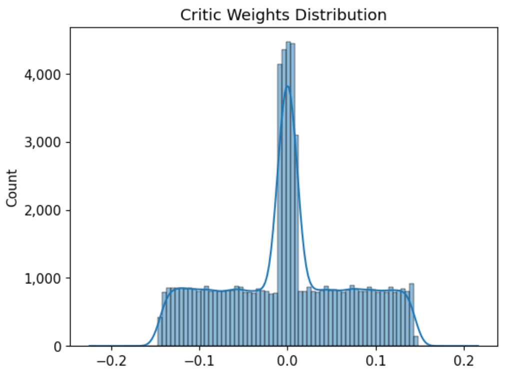

Figure 4.16: Critic's Weights Distribution

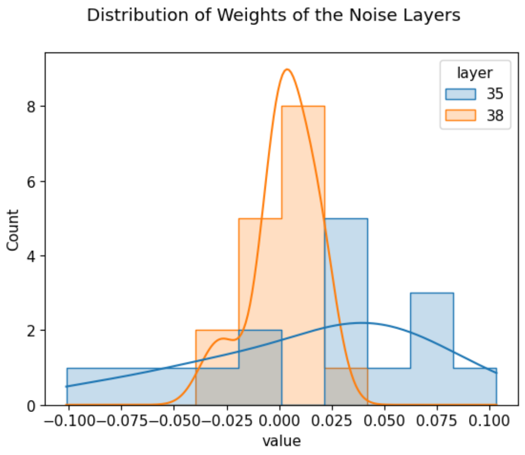

Figure 4.17: Distribution of Weights in the Noise Layers

the uncertainty of the horizon: the farther it is, the less certain they are. On top of that, it seems better at capturing the hourly variation while having only trained on 12-hourly increments.

Figure 4.19 showcases the speed scenarios for a specific location. While the model has captured the intra-day pattern well, it does not follow the original series as well as the benchmark.

Before presenting a consolidated view on results evaluation, Figure 4.20 displays the speed correlation of the model and benchmark, alongside the true data. The rows contain the Partial Autocorrelation Function (ACF), the correlation between all series and Land (Land-ACF) and the Ocean (Ocean-ACF); while the columns showcase the true, proposed and benchmark values. Overall, the model seems to have captured the series structure well, with a stronger amplitude. On the other hand, the benchmark still captures the dependency better - even if not well for the *Land* correlation.

To better provide an overall view of results, Figure 4.21 represents the spatial average of CRPS over the prediction horizon (T), Equation 4-7. It starts

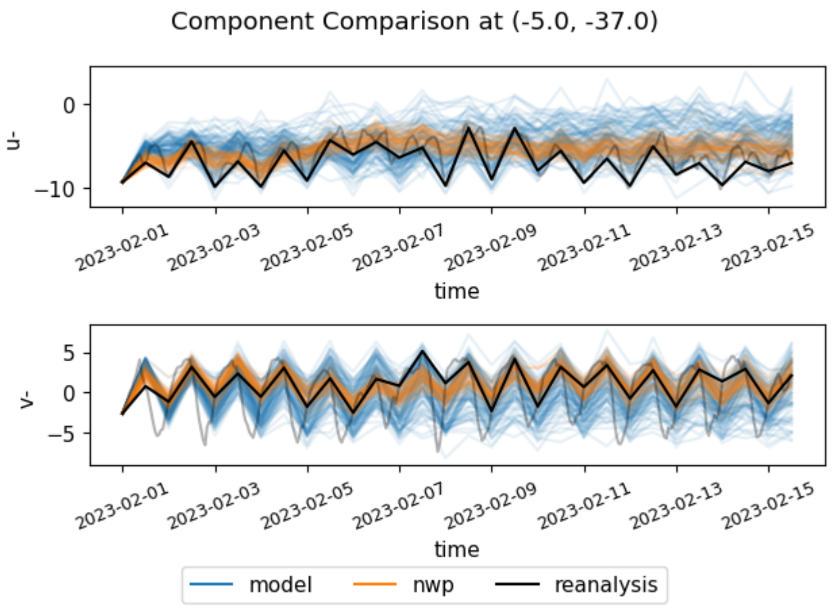

Figure 4.18: Scenarios Comparison

at zero (perfect prediction, since the zeroth step is the true data), and then is the average of each step (i) over all months. The dashed lines represent the overall average. It is interesting to note that, while the model's predictions worsen faster than NWP's scenarios initially, its scenarios seem to stay close by to the benchmark's for many steps ahead.

$$CRPS_i = \frac{1}{\#h \times \#w} \sum_{h,w} CRPS(y_i^{h,w}, \hat{y}_{i_S}^{h,w}) \quad \forall i \in [1, T]\,, S \in \mathcal{N} \tag{4-7}$$

Finally, both models are evaluated. Table 4.3 provides the CRPS' averages for each month of the test set. Since CRPS is a generalization of MAE, this table can be compared to Table 4.2, showing that including uncertainty improves performance.

Another perspective over these results comes from applying the Diebold-Mariano's test on each model's forecasts medians. For each combination of latitude and longitude, the amount of months when the null hypothesis is accepted (for p-value greater or equal to 0.01) is counted. This is shown in Table 4.4. Since the maximum value for each table entry is 12 (amount of months in a year), it can be concluded that, for 71.97% of test series (593 out of 824 $= h \times w \times$ months $\times$ components $= 6 \cdot 6 \cdot 12 \cdot 2$), our proposal is comparable to the current state-of-the-art.

In conclusion, the proposed methodology delivered similar results to the state of the art, while depending on much less data and requiring much less computational power. Although NWP's forecasts showed better performance

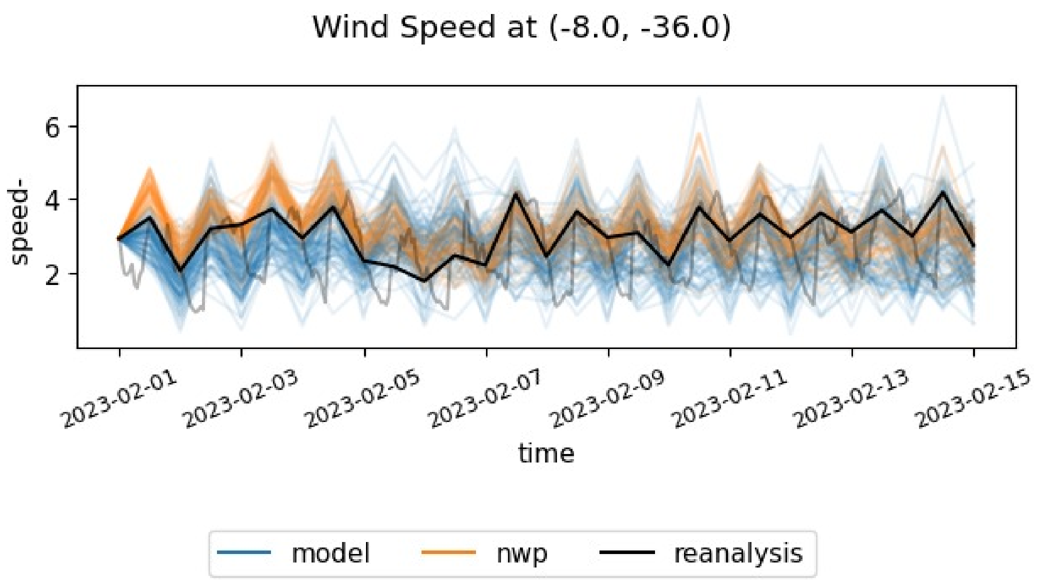

Figure 4.19: Example of Generated Wind Scenarios

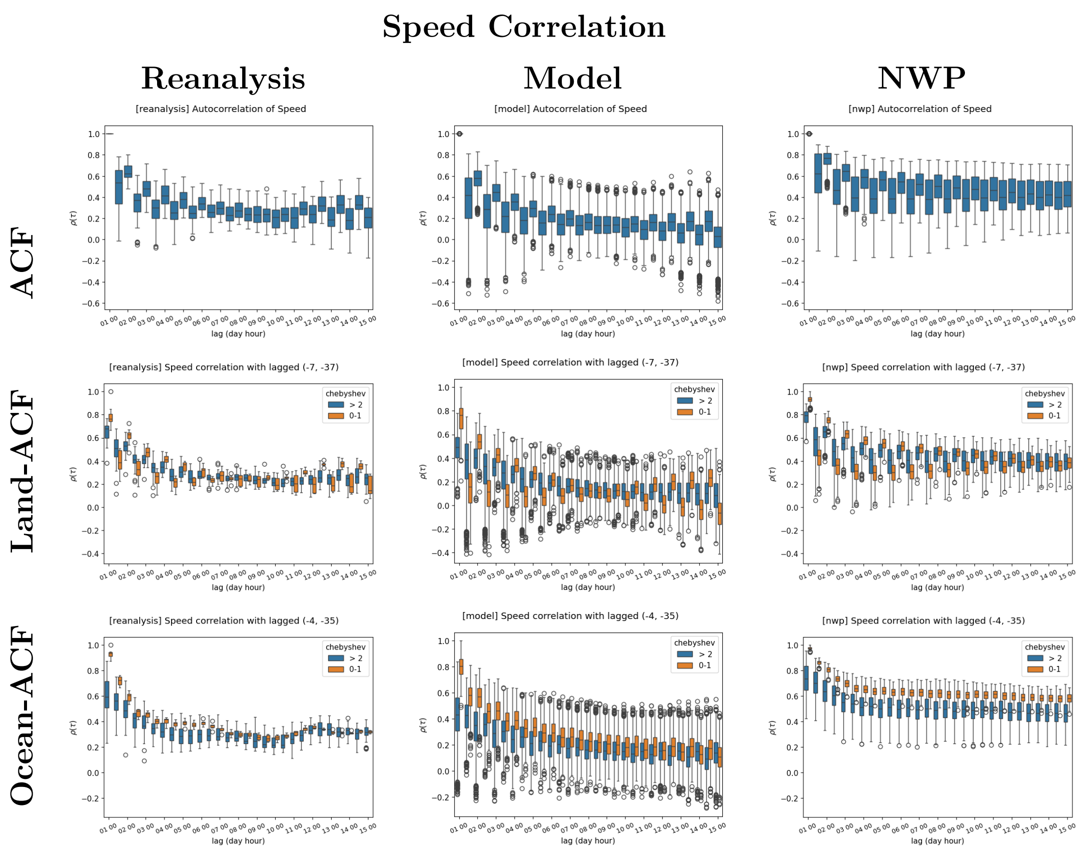

Figure 4.20: Speed Correlation boxplot comparison

when comparing CRPS, it is statistically equal than our proposal for 71.97% of the test series.

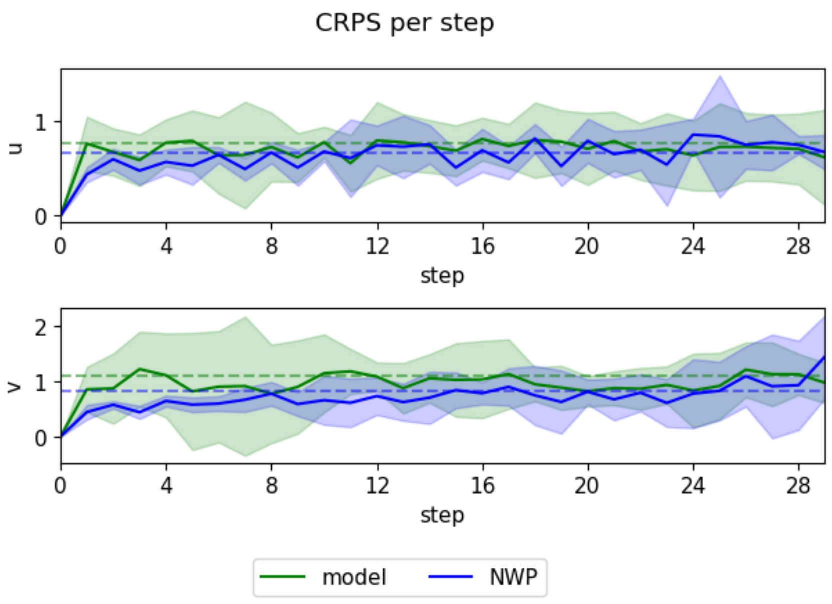

Figure 4.21: CRPS over step

Table 4.3: Average of CRPS for 28 steps (two weeks) for all test set months, per component. Values in bold are the lowest per component.

	Generator		NWP	
month	u	v	u	v
FEV	0.892	0.95	**0.717**	**0.743**
MAR	0.799	**0.809**	**0.725**	0.888
APR	0.901	0.868	**0.752**	**0.695**
MAY	0.797	1.039	**0.679**	**0.982**
JUN	**0.648**	0.836	0.65	**0.745**
JUL	0.768	1.432	**0.646**	**1.023**
AUG	**0.642**	**0.828**	0.69	0.874
SET	**0.705**	1.569	0.725	**0.649**
OCT	0.745	1.876	**0.604**	**0.754**
NOV	**0.602**	1.062	0.611	**0.788**
DEZ	0.613	1.004	**0.607**	**0.867**

Table 4.4: Amount of months when the Diebold-Mariano's test null hypothesis is accepted (p-value ≥ 0.01) for components u- and v-

	longitude						
latitude	-39.0	-38.0	-37.0	-36.0	-35.0	-34.0	total
-4.0	(3, 8)	(9, 11)	(6, 8)	(9, 5)	(9, 9)	(9, 9)	
-5.0	(10, 11)	(6, 11)	(11, 8)	(9, 7)	(10, 8)	(9, 9)	
-6.0	(10, 9)	(9, 10)	(8, 8)	(8, 5)	(9, 6)	(8, 6)	
-7.0	(12, 6)	(8, 9)	(9, 5)	(7, 9)	(8, 5)	(9, 7)	
-8.0	(9, 9)	(10, 10)	(8, 9)	(10, 10)	(10, 5)	(9, 5)	
-9.0	(8, 12)	(6, 9)	(9, 10)	(0, 8)	(9, 7)	(10, 7)	
total							593

5 Conclusions

The variability of renewable energy sources intensify the grid operator's challenge of maintaining stability. Wind power, for example, can vary rapidly due to changes in wind speed, making skillful forecasts progressively more important - specially for the medium-term (few hours to a few days) [4, 5] and if they include uncertainty [6, 7].

This need, aligned with: the high computational cost of Numerical Weather Prediction (NWP) systems and recent developments in Deep Learning (DL), opens space for new developments in the area [57, 58].

This work proposed a fully Convolutional Neural Network to generate 12-hourly wind speed scenarios. The model, estimated with consumer-grade hardware, is compared with a state-of-the-art Numerical Weather Prediction (NWP) system.

Having adjusted the data to the NWP time and spatial resolutions, the proposed model showed better performance in some months of the test data (6 out of 24), demonstrating long term stability and being statistically indistinguishable to the state of the art in 71.97% months (593 out of 824). This achievement is quite impressive, considering that the benchmark was produced with data all around the globe while the model was limited by the chosen area.

In addition, the proposed methodology can be easily adapted to coarser time steps and spatial grid, with little to no modification. Furthermore, the trained scheme seemed to reduce the double loss problem, a common problem when forecasting over a grid [9]. This problem is the result of a correct prediction on the wrong location, which incentivises the model to blur its predictions.

However, the GAN algorithm did not come without downsides. It was very hard to stabilize the training process. Here are some relevant choices that were made during the development that helped stabilize it:

- interpolating between MSE's (similar to reconstruction loss) and GAN's loss was of utmost importance;
- slowly increasing the training horizon T (every 10 epochs), since both model's were able to slowly adapt, also improved forecasts;

- separating both components for the Critic reduced the amount of weights and cross components interactions, allowing a bigger Critic to be trained;
- ensuring the 1-Lipschitz Critic's property was extremely challenging, specially since a diverging Critic would also make the Generator diverge, entering a vicious cycle. This restriction was only achieved and kept with spectral normalization [45]: using Gradient Penalty [44], on top of being more expensive, always diverged after more training (around the 100th epoch).

Beyond this work, a few suggestions can be made. From a modeling perspective, future work could further explore wind's time dependence - possibly including steps further in the past. Moreover, considering that this problem is translation and rotational variant, other backbone algorithms could have been used, such as Graph Neural Networks, which are also more similar to NWP's current modeling. Finally, a similar venue is to directly map previous wind speed to wind power output, since that's what the grid operator's optimization model uses [22].

Bibliography

[1] G. Giebel, G. Kariniotakis, R. Brownsword, The state-of-the-art in short term prediction of wind power from a danish perspective, in: 4th International Workshop on large scale integration of wind power and transmission networks for offshore wind farms, Billund, Denmark, 2003. URL: `https://minesparis-psl.hal.science/hal-00529986`.

[2] D. Bouche, R. Flamary, F. d'Alché Buc, R. Plougonven, M. Clausel, J. Badosa, P. Drobinski, Wind power predictions from nowcasts to 4-hour forecasts: A learning approach with variable selection, Renewable Energy 211 (2023) 938–947.

[3] P. M. Maçaira, F. L. C. Oliveira, P. G. C. Ferreira, F. V. N. de Almeida, R. C. Souza, INTRODUCING a CAUSAL PAR(p) MODEL TO EVALUATE THE INFLUENCE OF CLIMATE VARIABLES IN RESERVOIR INFLOWS: A BRAZILIAN CASE, Pesquisa Operacional 37 (2017) 107–128.

[4] C. Möhrlen, R. J. Bessa, N. Fleischhut, A decision-making experiment under wind power forecast uncertainty, Meteorological Applications 29 (2022) e2077.

[5] B. P. Cotia, C. L. Borges, A. L. Diniz, Optimization of wind power generation to minimize operation costs in the daily scheduling of hydrothermal systems, International Journal of Electrical Power & Energy Systems 113 (2019) 539–548.

[6] A. Tuohy, P. Meibom, E. Denny, M. O'Malley, Unit commitment for systems with significant wind penetration, IEEE Transactions on Power Systems 24 (2009) 592–601.

[7] F. Bouffard, F. D. Galiana, Stochastic security for operations planning with significant wind power generation, IEEE Transactions on Power Systems 23 (2008) 306–316.

[8] M. G. Schultz, C. Betancourt, B. Gong, F. Kleinert, M. Langguth, L. H. Leufen, A. Mozaffari, S. Stadtler, Can deep learning beat numerical weather prediction?, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 379 (2021) 20200097.

[9] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, P. Battaglia, Learning skillful medium-range global weather forecasting, Science 382 (2023) 1416–1421.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision 115 (2015) 211–252.

[11] B. Q. Bastos, F. L. Cyrino Oliveira, R. L. Milidiú, U-convolutional model for spatio-temporal wind speed forecasting, International Journal of Forecasting 37 (2021) 949–970.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (2020) 139–144.

[13] M. Mirza, S. Osindero, Conditional generative adversarial nets, CoRR abs/1411.1784 (2014).

[14] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein gan, 2017. URL: `https://arxiv.org/abs/1701.07875`. doi:`10.48550/ARXIV.1701.07875`.

[15] J. Gui, Z. Sun, Y. Wen, D. Tao, J. Ye, A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications, IEEE Transactions on Knowledge and Data Engineering 35 (2023) 3313–3332.

[16] A. Helseth, A. C. G. Melo, Q. M. Ploussard, B. Mo, M. E. P. Maceira, A. Botterud, N. Voisin, Hydropower scheduling toolchains: Comparing experiences in brazil, norway, and USA and implications for synergistic research, Journal of Water Resources Planning and Management 149 (2023).

[17] E. R. Office, Brazilian Energy Balance: summary report 2023 - reference year 2022, Report, Energy Research Office – Brazil, Esplanada dos Ministérios - Bloco U. Ministério de Minas e Energia - Sala 744 - 7º andar, 70065-900 Brasília – DF, 2023. URL: `https://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-748/topico-687/BEN2023.pdf`.

[18] M. E. P. Maceira, A. C. G. Melo, J. F. M. Pessanha, C. B. Cruz, V. A. Almeida, T. C. Justino, Combining monthly wind and inflow uncertainties in the stochastic dual dynamic programming, Energy Systems (2023).

[19] CEPEL, Cepel, 2021. URL: https://www.cepel.br/wp-content/uploads/2022/05/DECOMP%5FManualReferencia%5FOut2021.pdf, accessed at 2024/02/05 from https://www.cepel.br/produtos/documentacao-tecnica/.

[20] A. Luiz Diniz, F. Da Serra Costa, M. Elvira Maceira, T. Norbiato dos Santos, L. C. B. Dos Santos, R. Neves Cabral, Short/mid-term hydrothermal dispatch and spot pricing for large-scale systems-the case of brazil, in: 2018 Power Systems Computation Conference (PSCC), 2018, pp. 1–7. doi:10.23919/PSCC.2018.8442897.

[21] T. S. Nielsen, A. Joensen, H. Madsen, L. Landberg, G. Giebel, A new reference for wind power forecasting, Wind Energy 1 (1998) 29–34.

[22] G. G. et al., The state-of-the-art in short-term prediction of wind power a literature overview, 2nd edition document type deliverable, 2011.

[23] E. Erdem, J. Shi, Arma based approaches for forecasting the tuple of wind speed and direction, Applied Energy 88 (2011) 1405–1414.

[24] B. G. Brown, R. W. Katz, A. H. Murphy, Time series models to simulate and forecast wind speed and wind power, Journal of Climate and Applied Meteorology 23 (1984) 1184–1195.

[25] J. F. M. Pessanha, A. C. G. Melo, M. E. P. Maceira, V. Almeida, Generation of short-term wind power scenarios from an ensemble of hourly wind speed forecasts, in: 2022 17th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS), 2022, pp. 1–6. doi:10.1109/PMAPS53380.2022.9810569.

[26] Y. Liu, H. Qin, Z. Zhang, S. Pei, Z. Jiang, Z. Feng, J. Zhou, Probabilistic spatiotemporal wind speed forecasting based on a variational bayesian deep learning model, Applied Energy 260 (2020) 114259.

[27] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, 2015. arXiv:1505.04597.

[28] C. Jiang, Y. Mao, Y. Chai, M. Yu, S. Tao, Scenario Generation for Wind Power Using Improved Generative Adversarial Networks, IEEE Access 6 (2018) 62193–62203.

[29] S. E. Haupt, P. A. Jiménez, J. A. Lee, B. Kosović, 1 - principles of meteorology and numerical weather prediction, in: G. Kariniotakis (Ed.), Renewable Energy Forecasting, Woodhead Publishing Series in Energy, Woodhead Publishing, 2017, pp. 3–28. URL: https://www.sciencedirect.com/science/

article/pii/B9780081005040000019. doi:https://doi.org/10.1016/B978-0-08-100504-0.00001-9.

[30] D. J. Stensrud, Parameterization schemes: Keys to understanding numerical weather prediction models, Cambridge University Press, 2009.

[31] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[32] S. Linnainmaa, The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors, 1970.

[33] D. P. Kingma, J. L. Ba, Adam: A method for stochastic optimization, 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015) 1–15.

[34] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Journal of Machine Learning Research - Proceedings Track 9 (2010) 249–256.

[35] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012. arXiv:1207.0580.

[36] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, 33rd International Conference on Machine Learning, ICML 2016 3 (2016) 1651–1660.

[37] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-December (2016) 770–778.

[38] H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein, Visualizing the loss landscape of neural nets, Advances in Neural Information Processing Systems 2018-December (2018) 6389–6399.

[39] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

[40] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, CoRR abs/1710.10196 (2017).

[41] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, CoRR abs/1812.04948 (2018).

[42] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, T. Aila, Analyzing and improving the image quality of stylegan, CoRR abs/1912.04958 (2019).

[43] M. Lee, J. Seok, Regularization methods for generative adversarial networks: An overview of recent studies, 2020. URL: `https://arxiv.org/abs/2005.09165`. `arXiv:2005.09165`.

[44] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, Improved training of wasserstein GANs, Advances in Neural Information Processing Systems 2017-Decem (2017) 5768–5778.

[45] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, CoRR abs/1802.05957 (2018).

[46] J. A. Weyn, D. R. Durran, R. Caruana, Can machines learn to predict weather? using deep learning to predict gridded 500-hpa geopotential height from historical weather data, Journal of Advances in Modeling Earth Systems 11 (2019) 2680–2693.

[47] P. Manisha, D. Das, S. Gujar, Effect of input noise dimension in gans, CoRR abs/2004.06882 (2020).

[48] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. `arXiv:1511.06434`.

[49] J. A. Weyn, D. R. Durran, R. Caruana, Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere, Journal of Advances in Modeling Earth Systems 12 (2020).

[50] F. Diebold, R. Mariano, Comparing predictive accuracy, Journal of Business & Economic Statistics 13 (1995) 253–63.

[51] R. J. Hyndman, Y. Khandakar, Automatic time series forecasting: the forecast package for R, Journal of Statistical Software 27 (2008) 1–22.

[52] D. Harvey, S. Leybourne, P. Newbold, Testing the equality of prediction mean squared errors, International Journal of Forecasting 13 (1997) 281–291.

[53] J. Muñoz Sabater, Era5-land hourly data from 1950 to present., 2019. URL: `https://cds.climate.copernicus.eu/cdsapp#!/dataset/10.24381/cds.e2161bac?tab=overview`. doi:`10.24381/cds.e2161bac`.

[54] C. D. S. Copernicus Climate Change Service, Seasonal forecast subdaily data on pressure levels, 2018. URL: `https://cds.climate.copernicus.`

eu/cdsapp#!/dataset/seasonal-original-pressure-levels?tab=overview. doi:10.24381/cds.50ed0a73.

[55] J. Olauson, ERA5: The new champion of wind power modelling?, Renewable Energy 126 (2018) 322–331.

[56] S. C. de Aquino Ferreira, F. L. C. Oliveira, P. M. Maçaira, Validation of the representativeness of wind speed time series obtained from reanalysis data for brazilian territory, Energy 258 (2022) 124746.

[57] S. Lang, M. Alexe, M. Chantry, J. Dramsch, F. Pinault, B. Raoult, M. C. A. Clare, C. Lessig, M. Maier-Gerber, L. Magnusson, Z. B. Bouallègue, A. P. Nemesio, P. D. Dueben, A. Brown, F. Pappenberger, F. Rabier, Aifs - ecmwf's data-driven forecasting system, 2024. URL: https://arxiv.org/abs/2406.01465. arXiv:2406.01465.

[58] A. McNally, C. Lessig, P. Lean, E. Boucher, M. Alexe, E. Pinnington, M. Chantry, S. Lang, C. Burrows, M. Chrust, F. Pinault, E. Villeneuve, N. Bormann, S. Healy, Data driven weather forecasts trained and initialised directly from observations, 2024. URL: https://arxiv.org/abs/2407.15586. arXiv:2407.15586.

[59] G. Van Rossum, F. L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.

[60] C. C. C. Service, Era5 hourly data on pressure levels from 1940 to present, 2018. URL: https://cds.climate.copernicus.eu/doi/10.24381/cds.bd0915c6. doi:10.24381/CDS.BD0915C6.

[61] C. C. C. Service, Seasonal forecast subdaily data on pressure levels, 2018. URL: https://cds.climate.copernicus.eu/doi/10.24381/cds.50ed0a73. doi:10.24381/CDS.50ED0A73.

[62] S. Hoyer, H. Joseph, xarray: N-D labeled Arrays and Datasets in Python, Journal of Open Research Software 5 (2017).

[63] Dask Development Team, Dask: Library for dynamic task scheduling, 2016. URL: https://dask.org.

[64] W. McKinney, Data structures for statistical computing in python, in: S. van der Walt, J. Millman (Eds.), Proceedings of the 9th Python in Science Conference, Proceedings of the Python in Science Conference, SciPy, 2010, pp. 56–61. doi:10.25080/Majora-92bf1922-00a.

[65] M. L. Waskom, seaborn: statistical data visualization, Journal of Open Source Software 6 (2021) 3021.

[66] J. D. Hunter, Matplotlib: A 2d graphics environment, Computing in Science & Engineering 9 (2007) 90–95.

[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[68] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, S. Chintala, PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation, in: 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24), ACM, 2024. URL: https://pytorch.org/assets/pytorch2-2.pdf. doi:10.1145/3620665.3640366.

[69] W. Falcon, The PyTorch Lightning team, PyTorch Lightning, 2019. URL: https://github.com/Lightning-AI/lightning. doi:10.5281/zenodo.3828935.

[70] B. Leon, H. Stephan, K. Alex, O. Drew, properscoring, https://github.com/properscoring/properscoring, 2015.

6
Appendix A. Supplementary Material

Supplementary material (code, tables and visualizations) to this work is available online at github.com/felipewhitaker/windscenarios.

The authors thank the `Python` programming language [59] and other important projects, listed below alongside what it was used for:

- **Data Download** was done through the `CDSAPI`, with specific citations for each dataset: *Reanalysis* [60] and *Forecast* [61];
- **Data Interface** was done using `xarray` [62], using `dask` [63] for heavy workflows and `pandas` [64] for easy manipulations;
- **Visualizations** were created using `seaborn` [65], with some manipulations done with `matplotlib` [66];
- Although all **preprocessing** was done with `xarray` [62], `sklearn`'s [67] structure was used;
- The **model was developed** using `Pytorch` [68], and training engineering abstractions done by `Pytorch Lightning` [69]; and
- `properscoring`'s CRPS implementation [70] was used for **evaluation**

Finally, the computer configuration - all consumer-grade components - is stated below:

Processor (CPU)	AMD Ryzen 5 5600G with Radeon Graphics, 3.90 GHz
Graphics Processing Unit (GPU)	NVIDIA GeForce RTX 3060 12GB
Installed RAM	64.0 GB (63.9 GB usable)
System type	64-bit operating system, x64-based processor