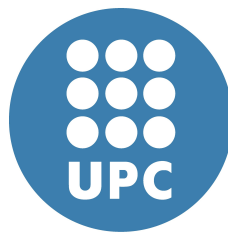


# Deep Learning architectures applied to wind time series multi-step forecasting



**Jaume Manero Font**

**Advisors**

Javier Béjar Alonso

Ulises Cortés García

Computer Science Department  
Universitat Politècnica de Catalunya - BarcelonaTECH

This dissertation is submitted for the degree of

*Ph.D. in Artificial Intelligence  
with International Certification*

Barcelona, June 2020

PhD Thesis Dissertation

*Deep Learning architectures applied to wind time series multi-step forecasting*  
(final version 8 June 2020)

© Jaume Manero Font

jaume.manero@gmail.com / jaume.manero@upc.edu

Universitat Politècnica de Catalunya - BarcelonaTECH

Departament de Ciències de la Computació

[www.cs.upc.edu](http://www.cs.upc.edu)

Creative Commons License BY NC SA EU



*To my parents who educated me  
and to Katrina, Selma and Manu, who made me learn*



## Acknowledgements

In Sweden, when a PhD student is ready for his thesis defense, nails a printed copy of his research to a tree or a wall placed in his Campus, signaling the end of his work and making publicly available the results of his dissertation in a so called 'Spikning' ritual. This tradition replicates the nailing of the 95 theses to the Wittenberg Castle door in 1517 by Martin Luther. Today, I'll do my personal 'Spikning' making public this work which is the culmination of a very long trip, through ups and downs, many jobs, different countries, and even a global pandemic.

All started back in the '80s when I joined a novel Computer Science program at the Technical University of Catalonia. "Life is a Circle", or so they say. When I left University to join the industry, I was supposed to be the first PhD student supervised by Ulises, a young and enthusiastic professor that full of energy sponsored me in a nascent Artificial Intelligence program, but the whims of fate decided to change it, putting this plan on hold. The intermission was very long, but luckily one day our routes crossed again, the stuck clock restarted, and he welcomed me into the Artificial Intelligence program with open arms and a big smile. Since then, his openness, help and joy has made a real difference in hard and good times, and for this I will always be grateful. In our first conversations he told me, with his quintessential optimism, that the journey was going to be much more important than the destiny, a sentence that today still resounds in my head, a sentence I now understand in fully, and makes me smile.

This journey would have been impossible without a guide, and this has been Javier, who has managed me and coped with my clumsiness and inexperience, his talent has been an inspiration, his skill has helped me to weather many storms. I own all the work to him, and I'd like to thank him here for his unconditional and patient support.

Inspiration is elusive as it always appears unexpectedly, here or there. Many people have inspired me, like a young Ramón López de Mántaras who unleashed my interest in Artificial Intelligence or Mateo Valero, who crafted my enthusiasm in the amazingness of computers, there are many more, too many to list them here, however, I want to show my gratitude to

Dr. Srinivas Sampalli who, in the summer of 2019, welcomed me with open arms at his Dalhousie University Laboratory, where I obtained a unique experience with a smart, and insultingly young, group of collaborators, Rob, Amjad, Darshana, Masood, Reetam, Marc, Bader, and Dr. Ohno, they form a truly global team that made me understand better and see further.

This list does not end up here, as there are many more friends who have given me that push, or that small piece of advice when it truly makes a difference, like Emile and E'Rizo with their always wise comments, or my good friend Varou, who patiently listened for hours to my rumbblings when the times were rough. I don't want to forget Alfredo, a big friend, who planted the PhD seed in me, and who I've cursed many nights when the task looked too steep for my ability, or Eladi and David who on-boarded me in quixotic quests, or Victoria and Lola, two friendships that the stochastic gradient descent helped me to recover, or Fabio who volunteered to the torture of reading my manuscript. I must say some thankful words also for the anonymous reviewers that read and helped me improve this dissertation, and many more, too many to count them, who have encouraged me along the way. Thanks to everyone, I am in debt with you all.

Now I have not enough words to thank my parents, who have been always there when I needed support. They taught me, by example, the meaning of perseverance and love. My mother Neus, I know, would be immensely proud if she had the opportunity to see me here today, a proudness she treasured all her life about me. And my father Jaume, possibly the wisest person I have ever known, and the role model in our family, who has always pushed us to learn, always ahead. Their love has always been with me in all the short and long trips of my life.

My daily fuel and energy comes from my family, I am privileged to have Selma and Manu, who have encouraged and inspired me with their ideas, support and unconditional admiration. They are the nicest daughter and son you'd find anywhere in this world.

Finally I can't close this list of acknowledgements without mentioning Katrina, my partner in life who always knows, better than me, what is going on in my head and more important, in my heart. Without her all of this would have been just impossible.

Today, I finish this journey in a world badly hammered by a global pandemic, with rough times ahead. but today is the moment for celebration and I don't want to miss the opportunity to look back, smile and shout out to the wind: *"Wow, What a ride!"*

## Abstract

Forecasting is a critical task for the integration of wind-generated energy into electricity grids. Numerical weather models applied to wind prediction, work with grid sizes too large to reproduce all the local features that influence wind, thus making the use of time series with past observations a necessary tool for wind forecasting. This research work is about the application of deep neural networks to multi-step forecasting using multivariate time series as an input, to forecast wind speed at 12 hours ahead.

Wind time series are sequences of meteorological observations like wind speed, temperature, pressure, humidity, and direction. Wind series have two statistically relevant properties; non-linearity and non-stationarity, which makes the modelling with traditional statistical tools very inaccurate.

In this thesis we design, test and validate novel deep learning models for the wind energy prediction task, applying new deep architectures to the largest open wind data repository available from the National Renewable Laboratory of the US (NREL) with 126,692 wind sites evenly distributed on the US geography. The heterogeneity of the series, obtained from several data origins, allows us to obtain conclusions about the level of fitness of each model to time series that range from highly stationary locations to variable sites from complex areas.

We propose Multi-Layer, Convolutional and Recurrent Networks as basic building blocks, and then combined into heterogeneous architectures with different variants, trained with optimisation strategies like drop and skip connections, early stopping, adaptive learning rates, filters and kernels of different sizes, between others. The architectures are optimised by the use of structured hyper-parameter setting strategies to obtain the best performing model across the whole dataset.

The learning capabilities of the architectures applied to the various sites find relationships between the site characteristics (terrain complexity, wind variability, geographical location) and the model accuracy, establishing novel measures of site predictability relating the fit of the models with indexes from time series spectral or stationary analysis. The designed methods offer new, and superior, alternatives to traditional methods.

**Keywords:** Deep Learning, Wind Prediction, Time Series Forecasting, Multi-step prediction, RNN, CNN, Spectral analysis, Forecastability.





## Resum

### Arquitectures d'aprenentatge profund aplicades a predicció de múltiple esglaó de series temporals de vent

La predicció de vent és clau per a la integració de l'energia eòlica en els sistemes elèctrics. Els models meteorològics es fan servir per predicció, però tenen unes gralles geogràfiques massa grans per a reproduir totes les característiques locals que influencien la formació de vent, fent necessària la predicció d'acord amb les sèries temporals de mesures passades d'una localització concreta. L'objectiu d'aquest treball d'investigació és l'aplicació de xarxes neuronals profundes a la predicció *multi-step* utilitzant com a entrada series temporals de múltiples variables meteorològiques, per a fer prediccions de vent d'ací a 12 hores.

Les sèries temporals de vent són seqüències d'observacions meteorològiques tals com, velocitat del vent, temperatura, humitat, pressió baromètrica o direcció. Les sèries temporals de vent tenen dues propietats estadístiques rellevants, que són la no linearitat i la no estacionalitat, que fan que la modelització amb eines estadístiques sigui poc precisa.

En aquesta tesi es validen i proven models de deep learning per la predicció de vent, aquests models d'arquitectures d'auto aprenentatge s'apliquen al conjunt de dades de vent més gran del món, que ha produït el National Renewable Laboratory dels Estats Units (NREL) i que té 126,692 ubicacions físiques de vent distribuïdes per total la geografia de nord Amèrica. L'heterogeneïtat d'aquestes sèries de dades permet establir conclusions fermes en la precisió de cada mètode aplicat a sèries temporals generades en llocs geogràficament molt diversos.

Proposem xarxes neuronals profundes de tipus multi-cap, convolucionals i recurrents com a blocs bàsics sobre els quals es fan combinacions en arquitectures heterogènies amb variants, que s'entrenen amb estratègies d'optimització com drops, connexions skip, estratègies de parada, filtres i kernels de diferents mides entre altres. Les arquitectures s'optimitzen amb algorismes de selecció de paràmetres que permeten obtenir el model amb el millor rendiment, en totes les dades.

Les capacitats d'aprenentatge de les arquitectures aplicades a ubicacions heterogènies permet establir relacions entre les característiques d'un lloc (complexitat del terreny, variabilitat del vent, ubicació geogràfica) i la precisió dels models, establint mesures de predictibilitat que relacionen la capacitat dels models amb les mesures definides a partir d'anàlisi espectral o d'estacionalitat de les sèries temporals. Els mètodes desenvolupats ofereixen noves i superiors alternatives als algorismes estadístics i mètodes tradicionals.

**Paraules Clau:** Aprenentatge Profund, Predicció Vent, Predicció de series temporals, Predicció de múltiple esglaó, MLP, CNN, RNN, Anàlisi espectral, Predictibilitat.



## Resumen

### Arquitecturas de aprendizaje profundo aplicadas a la predicción en múltiple escalón de series temporales de viento

La predicción de viento es clave para la integración de esta energía eólica en los sistemas eléctricos. Los modelos meteorológicos tienen una resolución geográfica demasiado amplia que no reproduce todas las características locales que influyen en la formación del viento, haciendo necesaria la predicción en base a series temporales de cada ubicación concreta. El objetivo de este trabajo de investigación es la aplicación de redes neuronales profundas a la predicción multi-step usando como entrada series temporales de múltiples variables meteorológicas, para realizar predicciones de viento a 12 horas.

Las series temporales de viento son secuencias de observaciones meteorológicas tales como, velocidad de viento, temperatura, humedad, presión barométrica o dirección. Las series temporales de viento tienen dos propiedades estadísticas relevantes, que son la no linealidad y la no estacionalidad, lo que implica que su modelización con herramientas estadísticas sea poco precisa.

En esta tesis se validan y verifican modelos de aprendizaje profundo para la predicción de viento, estos modelos de arquitecturas de aprendizaje automático se aplican al conjunto de datos de viento más grande del mundo, que ha sido generado por el National Renewable Laboratory de los Estados Unidos (NREL) y que tiene 126,682 ubicaciones físicas de viento distribuidas por toda la geografía de Estados Unidos. La heterogeneidad de estas series de datos permite establecer conclusiones válidas sobre la validez de cada método al ser aplicado en series temporales generadas en ubicaciones físicas muy diversas.

Proponemos redes neuronales profundas de tipo multi capa, convolucionales y recurrentes como tipos básicos, sobre los que se han construido combinaciones en arquitecturas heterogéneas con variantes de entrenamiento como drops, conexiones skip, estrategias de parada, filtros y kernels de distintas medidas, entre otros. Las arquitecturas se optimizan con algoritmos de selección de parámetros que permiten obtener el mejor modelo buscando el mejor rendimiento, incluyendo todos los datos.

Las capacidades de aprendizaje de las arquitecturas aplicadas a localizaciones físicas muy variadas permiten establecer relaciones entre las características de una ubicación (complejidad del terreno, variabilidad de viento, ubicación geográfica) y la precisión de los modelos, estableciendo medidas de predictibilidad que relacionan la capacidad de los algoritmos con índices que se definen a partir del análisis espectral o de estacionalidad de las series temporales. Los métodos desarrollados ofrecen nuevas alternativas a los algoritmos estadísticos tradicionales.

**Palabras Clave:** Aprendizaje profundo, Predicción viento, Predicción de series temporales, Predicción de múltiple escalón, MLP, CNN, RNN, Análisis espectral, Predictibilidad.



# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vii</b>
<b>List of figures</b>	<b>xix</b>
<b>List of tables</b>	<b>xxiii</b>
<b>List of algorithms</b>	<b>xxiv</b>
<b>I Introduction</b>	<b>1</b>
<b>1 Context, motivation, structure, and contributions</b>	<b>3</b>
1.1 Context and Motivation . . . . .	4
1.2 Thesis dissertation structure . . . . .	8
1.3 Thesis Contributions . . . . .	9
1.3.1 Scientific Journal and Conference publications . . . . .	9
1.3.2 Other contributions . . . . .	10
1.3.3 Graduate teaching experience . . . . .	11
1.3.4 Code and Result Notebooks . . . . .	12
<b>II State of the Art</b>	<b>13</b>
<b>2 Time Series</b>	<b>15</b>
2.1 Introduction to Time Series . . . . .	16
2.2 Time Series definition and representation . . . . .	17
2.3 Time Series Properties and Models . . . . .	18
2.3.1 Stationarity and Ergodicity properties . . . . .	18
2.3.2 Decomposing a Time Series in Trend and Seasonality . . . . .	19
2.3.3 Linearity in time series . . . . .	22

## Contents

---

2.3.4	Time series as a stochastic process . . . . .	23
2.4	Forecasting Time Series . . . . .	23
2.4.1	Point and probabilistic forecasting . . . . .	23
2.4.2	Linear and non-linear forecasting methods . . . . .	24
2.4.2.1	Linear forecasting methods . . . . .	25
2.4.2.2	Forecasting series from decomposition . . . . .	27
2.4.2.3	Non-linear time series forecasting methods . . . . .	27
2.4.3	Single-step and Multiple-Step ahead forecasting . . . . .	28
2.5	Signal theory and time series analysis . . . . .	30
2.6	Wind time series . . . . .	30
2.6.1	Non-Stationarity of wind time series . . . . .	33
2.6.2	Non-Linearity of wind time series . . . . .	34
2.6.3	Distribution function in wind time series . . . . .	34
<b>3</b>	<b>Deep learning and time series</b> . . . . .	<b>37</b>
3.1	Machine and Deep learning algorithms for time series forecasting . . . . .	38
3.2	Multi-step forecasting with Neural Networks . . . . .	39
3.2.1	Multiple Input Multiple Output Forecasting . . . . .	42
3.3	DL architectures for time series forecasting . . . . .	43
3.4	MLP for time series . . . . .	43
3.5	CNN architectures for time series . . . . .	45
3.5.1	Padding in the convolutional operation . . . . .	48
3.5.2	Separable convolution . . . . .	49
3.5.3	Skip and Residual connections . . . . .	51
3.5.4	Multi-head models and stacked models . . . . .	53
3.6	Recurrent Neural Networks . . . . .	54
3.6.1	GRU and LSTM units for RNN architectures . . . . .	55
3.6.2	RNN with Attention mechanisms . . . . .	56
3.6.3	RNN MIMO Architectures for time series . . . . .	57
<b>4</b>	<b>State of the Art of wind forecasting with deep learning</b> . . . . .	<b>59</b>
4.1	State of the art of Meteorological based forecasting . . . . .	60
4.1.1	Meteorological models for wind prediction . . . . .	60
4.1.2	NWP forecasting methods . . . . .	63
4.2	State of the art of Machine learning methods for wind time series . . . . .	65
4.2.1	Linear regression methods . . . . .	65
4.2.2	Signal Analysis . . . . .	66
4.2.3	Bayesian methods . . . . .	66
4.2.4	$k$ -NN: $k$ -Nearest Neighbours . . . . .	67

4.2.5	Support Vector Machines (SVM) . . . . .	68
4.2.6	Ensemble methods . . . . .	70
4.2.7	Gradient Boosting based ensembles . . . . .	72
4.2.8	K-means as unsupervised Learning for wind time series . . . . .	73
4.3	State of the art of deep learning applied to wind prediction . . . . .	73
4.3.1	Multi-Layer Perceptrons . . . . .	74
4.3.2	Convolutional Neural Networks . . . . .	76
4.3.3	Recurrent Neural Networks . . . . .	76
 <b>III Experimentation, Methodology and Results</b>		 <b>79</b>
<b>5</b>	<b>Wind Data</b>	<b>81</b>
5.1	The Sotavento wind park dataset . . . . .	82
5.2	The NREL Wind dataset . . . . .	82
5.3	Other datasets . . . . .	88
<b>6</b>	<b>Experimental Framework</b>	<b>89</b>
6.1	Experiment design settings . . . . .	90
6.2	Data preparation . . . . .	91
6.2.1	Time series pre-processing . . . . .	91
6.3	Experiment Setup . . . . .	92
6.4	Hyper-parameter setting strategy . . . . .	95
6.5	Interpreting the results of an experiment . . . . .	98
<b>7</b>	<b>Basic Deep learning architectures for wind time series forecasting</b>	<b>101</b>
7.1	Experiments with Deep learning architectures . . . . .	102
7.2	Experimentation Approach . . . . .	103
7.2.1	Calculation of Baseline methods . . . . .	103
7.3	Determination of best regression strategy . . . . .	108
7.4	Main Experiment: Deep learning Models . . . . .	111
7.5	Discussion . . . . .	118
<b>8</b>	<b>Going Deeper with wind time series forecasting architectures</b>	<b>123</b>
8.1	Introduction . . . . .	124
8.2	MLP architecture variants . . . . .	124
8.2.1	Skip connections in MLP architectures . . . . .	124
8.2.2	Multi-regression architectures with MLP . . . . .	126
8.3	RNN Architectures GRU or LSTM cells and Attention . . . . .	128
8.4	Using series with a 5 minute step . . . . .	129

## Contents

---

8.5	Using a low-resolution weather variable . . . . .	131
8.6	Prediction one hour ahead with five-minute series . . . . .	134
8.7	Discussion . . . . .	135
<b>9</b>	<b>The effectiveness of CNN architectures for wind time series</b>	<b>137</b>
9.1	Experiments with Convolutional Networks for time series forecasting . . . . .	138
9.2	A CNN MIMO approach . . . . .	138
9.3	Baseline Methods . . . . .	140
9.4	Parameter space of the convolutional architectures . . . . .	141
9.5	Convolutional architectures . . . . .	142
9.5.1	Classic Convolutional and separable layers . . . . .	142
9.5.2	Adding Skip and residual connections . . . . .	145
9.5.3	Multi-head architectures . . . . .	147
9.5.4	Adding Gradient Boosting to a CNN Architecture . . . . .	148
9.6	Using CNN in series with a 5 minute step . . . . .	149
9.7	Global and local error evaluation . . . . .	151
9.7.1	Site 37917 in California with average accuracy . . . . .	152
9.7.2	Site 31321 offshore with high accuracy . . . . .	154
9.7.3	Site 118728 in Montana with average accuracy . . . . .	156
9.8	Discussion . . . . .	158
<b>10</b>	<b>Wind time series forecastability</b>	<b>161</b>
10.1	Forecastability and deep learning prediction . . . . .	162
10.2	Experiment settings . . . . .	163
10.3	Initial Testing and Pearson Correlation . . . . .	164
10.4	Statistical measures in wind prediction . . . . .	167
10.5	Terrain complexity as a forecastability descriptor . . . . .	167
10.6	Time series decomposition measures . . . . .	171
10.6.1	Value of decomposition measures for wind time series . . . . .	175
10.7	Entropy analysis applied to wind time series . . . . .	176
10.8	Discussion . . . . .	180
<b>IV</b>	<b>Conclusions &amp; Future Work</b>	<b>181</b>
<b>11</b>	<b>Conclusions</b>	<b>183</b>
11.1	Introduction . . . . .	184
11.2	Conclusions about the deep learning architectures . . . . .	184
11.2.1	Deep learning is more efficient than baselines . . . . .	184
11.2.2	Increase of depth does not correlate with accuracy . . . . .	186



11.2.3 Convolutional separable networks are superior for wind forecasting . . . . .	186
11.3 Conclusions about the wind time-series . . . . .	187
11.3.1 The wind input sequence length used for learning examples is short . . . . .	187
11.3.2 Higher frequency measurements improve accuracy . . . . .	188
11.3.3 Integrating meteorological information with time series . . . . .	190
11.4 Conclusions on site forecastability . . . . .	191
11.4.1 Site accuracy creates clusters of sites with same wind characteristics . . . . .	191
11.4.2 Ruggedness as an isolated measure is not a good predictor . . . . .	191
11.5 Conclusions summary . . . . .	193
<b>12 Future Work</b>	<b>195</b>
12.1 Introduction . . . . .	196
12.2 Use of data from real observation and higher frequency . . . . .	196
12.3 Integration of NWP values in prediction . . . . .	196
12.4 Use of ensembles . . . . .	197
12.5 Prediction of local wind events . . . . .	197
12.6 Prediction using neighbouring site information . . . . .	198
12.7 Wind speed probabilistic forecasting using deep learning . . . . .	198
<b>V Appendices</b>	<b>201</b>
<b>Appendix A Summary of experiments accuracy</b>	<b>203</b>
<b>Appendix B Accuracy and error</b>	<b>207</b>
<b>Appendix C Posters presented at conferences</b>	<b>215</b>
<b>Appendix D Code and Results</b>	<b>223</b>
<b>References</b>	<b>225</b>
<b>Author publications during the research period</b>	<b>241</b>



# List of figures

1.1	Grid Planning time frames . . . . .	4
1.2	Wind prediction example . . . . .	5
2.1	Barcelona average monthly temperature time series . . . . .	17
2.2	Decomposition Time Series Example . . . . .	21
2.3	Stock value time series example . . . . .	21
2.4	Partial auto-correlation Test PACF . . . . .	22
2.5	Bank of England Probabilistic Forecast . . . . .	24
2.6	Wind speed time series site in Techado, New Mexico . . . . .	31
2.7	Wind direction and intensity over a year in Sotavento Wind Park . . . . .	32
2.8	Wind Speed PDF Weibull distribution . . . . .	35
3.1	Input Output sequences in seq2seq model . . . . .	42
3.2	Components and MIMO Architectures for sequence to sequence learning . . . . .	43
3.3	A Perceptron is the basic building block in MLP . . . . .	44
3.4	CNN input is based on examples . . . . .	46
3.5	Convolution 1D and 2D . . . . .	47
3.6	Convolutional operation plus pooling . . . . .	48
3.7	Padding strategies . . . . .	49
3.8	Convolutional 2D separable operation . . . . .	50
3.9	Training stabilisation and Skip and Residual connections . . . . .	52
3.10	Multi-Head architecture . . . . .	53
3.11	Unfolded RNN Figure . . . . .	54
3.12	RNN Units . . . . .	55
3.13	RNN MIMO and unfolding graph . . . . .	56
4.1	ECMWF weather map . . . . .	62
5.1	NREL wind dataset sites in the US map . . . . .	83
5.2	NREL Dataset mean in US Map . . . . .	84
5.3	NREL Dataset variance in US Map . . . . .	84

## List of figures

---

5.4	Wind-roses from 16 random sites in US . . . . .	86
5.5	Wind-roses in Wind Park . . . . .	87
5.6	Map of group of sites in South Dakota . . . . .	88
6.1	NREL wind time series plot . . . . .	91
6.2	Experiment framework . . . . .	93
6.3	Experiment parameters . . . . .	94
6.4	Probability function for Random Forest model . . . . .	98
6.5	MLP map across the US Geography . . . . .	99
7.1	Architectures graphical definition . . . . .	103
7.2	Persistence model result geographical representation . . . . .	105
7.3	Plots for Baseline methods . . . . .	107
7.4	Plots comparing regression approach . . . . .	110
7.5	Boxplot MLP and CNN . . . . .	115
7.6	Boxplot MLP and RNN . . . . .	115
7.7	Best Model Graphical analysis on US Map . . . . .	116
7.8	Plots CNN and RNN architecture . . . . .	117
7.9	Interpretation of strip visualisation . . . . .	119
7.10	Accuracy strips for MLP accuracy above 9.0 . . . . .	120
7.11	Accuracy strips for MLP accuracy below 4.9 . . . . .	121
8.1	Skip connections in MLP architecture . . . . .	125
8.2	Multiple regressions with Sjoin architecture . . . . .	127
8.3	Boxplot MLP 1h vs MLP 5m . . . . .	130
8.4	Distribution plots comparison series at 5m and 1h . . . . .	131
8.5	Density and Distribution series with weather variable . . . . .	132
8.6	Boxplot MLP 5m vs MLP 5m with future . . . . .	133
9.1	CNN structure for MIMO modelling . . . . .	139
9.2	One dimensional convolutional . . . . .	139
9.3	US Map with accuracy of CNN separable network . . . . .	143
9.4	Layers and shapes of best CNN Architecture . . . . .	145
9.5	Site 37917 strips image . . . . .	153
9.6	Site 31321 strips image . . . . .	155
9.7	Site 118728 strips image . . . . .	157
9.8	Comparison of baselines with separable architecture . . . . .	158
10.1	Experimentation process for forecastability indexes . . . . .	164
10.2	Geo-spatial representation and measures . . . . .	166
10.3	Grid of points around a site (site in green) . . . . .	168

10.4 Grid of points around a site (site in green) . . . . .	169
10.5 Terrain ruggedness measures analysis . . . . .	170
10.6 Geo representation and forecastability indexes . . . . .	173
10.7 Trend strength analysis at US State level . . . . .	175
10.8 Sample Entropy (SampEnt) and Spectral Entropy (SpectEnt) Scatter-plots . .	178
10.9 Lump and Stab Scatter-plots . . . . .	179
11.1 Accuracy depth relationship CNN architectures . . . . .	185
11.2 Comparison series 1h with 5 minute . . . . .	189
11.3 Density and Distribution 5m series prediction 12h . . . . .	190
11.4 Spectral Entropy and Trend Strength . . . . .	192
11.5 Comparison of Colorado topographic map with CNN Prediction . . . . .	192
B.1 $R^2$ error comparison in two regressions . . . . .	211
C.1 Poster CCIA Conference . . . . .	217
C.2 Poster Wind2019 Conference . . . . .	219
C.3 Poster Energy3Canada Conference . . . . .	221



# List of tables

2.1	Dickey Fueller test result . . . . .	20
2.2	Comparison of different (p,d,q) ARIMA models . . . . .	27
2.3	ADF test on two NREL dataset sites (using Adfuller test from statsmodels) . . . . .	33
4.1	Comparison of SVM kernels RMS errors for wind speed . . . . .	70
4.2	Deep Learning Literature Review . . . . .	78
5.1	Wind Turbines in Sotavento wind park . . . . .	82
5.2	Dimensions on the NREL Wind dataset . . . . .	83
6.1	Best experiments from example . . . . .	97
7.1	Architectures short description . . . . .	102
7.2	Parameter space for baselines . . . . .	104
7.3	Baseline Methods and pbet architecture parameters used . . . . .	104
7.4	Baselines distributions summary . . . . .	106
7.5	Parameter space for MLP architectures . . . . .	108
7.6	Direct and MIMO strategies description and parameters . . . . .	109
7.7	Regression approaches comparison . . . . .	109
7.8	Parameter space for CNN and RNN architectures . . . . .	112
7.9	Deep learning Basic Architectures details. . . . .	113
7.10	Distribution comparison . . . . .	114
7.11	Mean and standard Distributions Main Architectures . . . . .	116
8.1	MLP additional Architectures . . . . .	124
8.2	Mean and standard Distributions . . . . .	125
8.3	Comparison Cascade with Classic MLP . . . . .	126
8.4	Mean and standard Distributions Multi-regression . . . . .	127
8.5	RNN alternative models . . . . .	128
8.6	Mean and standard Distributions RNN . . . . .	128
8.7	Architectures with series sampled at five min forecast at 12h . . . . .	130

## List of tables

---

8.8	Mean and standard Distributions Series 5m . . . . .	130
8.9	Comparison MLP architectures . . . . .	132
8.10	Architectures with series sampled at 5 min . . . . .	134
8.11	Mean and standard $R^2$ accuracy distributions Series 5m prediction 1h . . . . .	134
9.1	List of CNN Architectures designed for experimentation . . . . .	138
9.2	Baseline Methods . . . . .	140
9.3	Mean & standard deviation for Baselines . . . . .	140
9.4	CNN Skip and Residual Architectures overview . . . . .	142
9.5	Comparison of parameters $\theta$ in CNN and Separable Architectures . . . . .	143
9.6	Mean & standard deviation of classic & separable architectures . . . . .	146
9.7	Layer depth comparison between models (error calculated on Test Dataset) . . . . .	146
9.8	Mean & standard deviation of CNN with skip & residual connections . . . . .	147
9.9	Mean & standard deviation of CNN in multi-head architectures . . . . .	148
9.10	Mean & standard deviation of CNN architectures with Gradient Boosting . . . . .	149
9.11	CNN Architectures with series with period 5 minutes . . . . .	150
9.12	Mean & standard deviation of CNN in series with 5m steps . . . . .	150
9.13	Summary of measures on sites in Strips figures . . . . .	151
10.1	Correlations measures with DL predictions . . . . .	165
10.2	Correlation $f_T$ with $R^2$ CNN sep 2L per state . . . . .	174
10.3	Correlation between decomposition measures and CNN $R^2$ prediction . . . . .	175
10.4	Pearson Correlations between measures, time horizons and DL Prediction . . . . .	177
10.5	Correlation Sample and Spectral Entropy CNN sep 2L per state . . . . .	178
11.1	<i>Lag</i> length training examples . . . . .	187
11.2	Comparison of 1h and 5m series . . . . .	188
A.1	Experiments summary accuracy table . . . . .	204
A.2	Experiments summary Convolutional networks . . . . .	205
A.3	Experiments summary 5m series . . . . .	206



# List of Algorithms

2.1	Time Series decomposition . . . . .	20
2.2	Forecasting from a Decomposition . . . . .	28
4.1	Friedman's Gradient Boosting Algorithm . . . . .	73
6.1	Hyperparameter optimization algorithm . . . . .	96



# Part I

## Introduction

---

*"The wind is an untamed and unharnessed force; and quite possibly one of the greatest discoveries hereafter to be made, will be the taming and harnessing of it."*

(1860) Abraham Lincoln [[124](#)]

*"Prediction is the essence of intelligence"*

(2016) Yann LeCunn [[121](#)]



# Chapter 1

## Context, motivation, structure, and contributions

The new renewable energy paradigm requires precise prediction of future energy generation to maintain stability in the electricity grid. As renewable generation sources increase, forecasting becomes critical to optimise the electricity systems around the world. Renewable forecasting has been declared an essential activity by Artificial Intelligence researchers that have analysed how this discipline can contribute to the worldwide initiative to stop climate change [181].

This thesis is aligned with this global effort and deals with the application of deep learning to forecast future wind intensity from past observations, a hard problem due to the intrinsic physical complexity of wind formation.

Deep learning (DL) has shown outstanding results in many forecasting and classification tasks, and we believe it can be applied to the wind time series multi-step forecasting productively and efficiently. This Thesis proposal shows several novel deep learning approaches applied to wind forecast.

All this research work uses the National Renewable Energy Laboratory (NREL) wind dataset, which is the largest wind resource dataset available for research in this area. The availability of computing power has been made possible by the collaboration with the Barcelona Supercomputing Centre (BSC) [147], which has provided their infrastructure (Minotauro GPU cluster) to support the experiments.

This chapter is structured as follows: in Section 1.1 we develop the motivation for this work and its context then, in Section 1.2 we describe the structure of the dissertation and finally, in Section 1.3 we summarise the main contributions from this research.

## 1.1 Context and Motivation

The actual climate emergency is probably the most critical challenge faced by humankind in its long historical journey, as there will only be future viability of life on earth if this threat is curved down, and if not, the survival of the human race on earth is compromised. Tackling climate change is a global and gigantic task that requires global coordination like the global agreements signed between most nations at the United Nations Climate Change Conferences (COP) of the last years like the summit COP21 held in Paris, the COP23 in Bonn or the recent COP25 in Madrid [228, 159, 39].

The main area of concern is the transformation of our actual carbon-based economy into an emission-free energy generation, by replacing all fossil fuels with renewables, like wind and solar [236] [100].

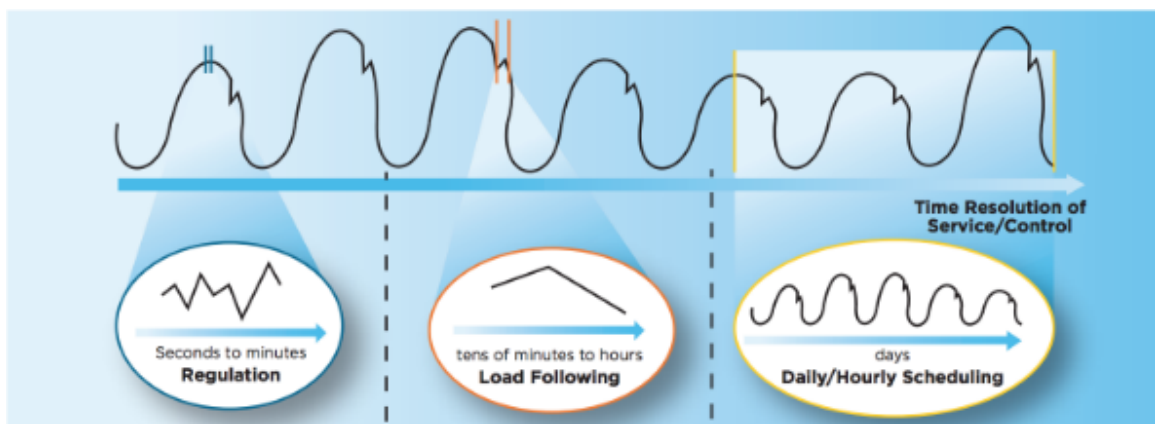


Fig. 1.1 Grid Planning time frames <sup>1</sup>

Wind, like other renewable sources, is intermittent by nature as its intensity depends on a combination of natural phenomena that have complex interactions.

This intermittence would generate instability in the electricity system if it were not continuously orchestrated with advanced forecasting at different time-frames (see Figure 1.1), and for this reason, predicting future renewable energy inputs is critical to assure the security of the electricity system.

The initial developments in wind power forecasting can be found in the '80s when some regression algorithms were applied to the first wind parks installed in California. Since then, this field has grown accordingly with the increase of penetration and sophistication of wind turbines, that has become a critical contributor to the electrical supply systems in many countries around the world. As an example, we can cite some nations with high wind penetration in their electricity generation mix (as a percentage of total production in 2016) like Denmark (36,8%), Ireland (27%), Portugal (24,7%), Spain (19%) or Germany (16%).

<sup>1</sup>Source Scientific American Article by Fares in [50]

Wind forecasting (see Figure 1.2) is an activity performed with two major approaches, depending on the data source, one using time series and the other numerical weather models as an input.

Time series based forecasting is better suited for short-term prediction while Numerical Weather Prediction (NWP) models are more precise for mid and long-term wind forecasting.

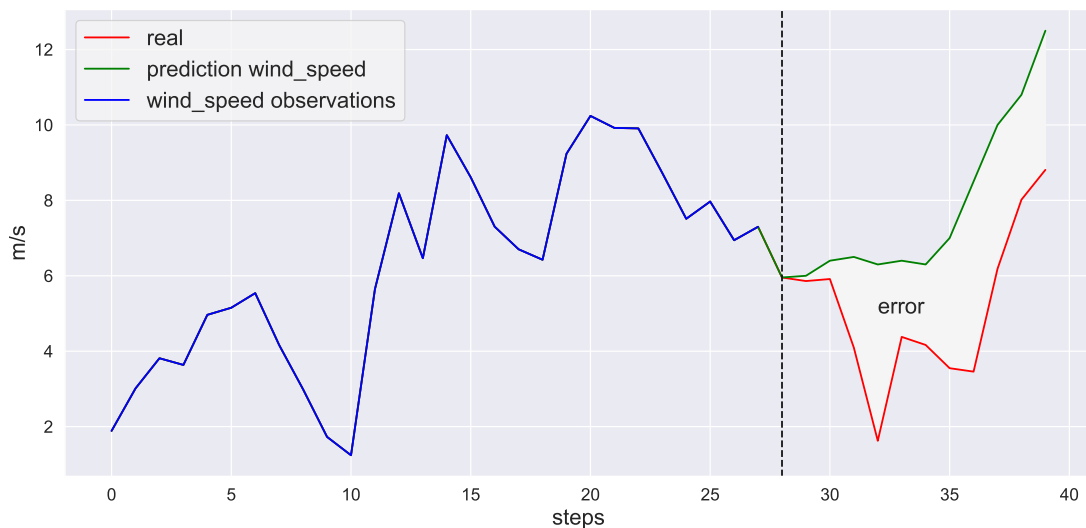


Fig. 1.2 Multi-step prediction with horizon 12 steps

Commercial applications use for forecasting a combination of time series and NWP modelling. Forecasting wind intensity can evolve into several new areas, like adding probabilities to the prediction or analysing small anomalies like wind ramps or wake effects [38, 164, 66, 55, 236].

The wind is the product of complex meteorological interactions, and the multivariate time series obtained from different observations share this complexity, presenting challenges for its modelling, fundamentally because they are non-stationary and non-linear. Due to these two properties, the accuracy of predictions using simple linear methods applied to the time series data appears to be inadequately low. For this reason, the industry includes in their approach numerical weather forecasting when the prediction horizon is over a four to six hours [73].

Algorithms that learn from data have shown high adaptability to all kinds of applications, by using different data sources, like structured data, images, video, sound, natural language and many more. Time series are another relevant data type to be used in deep learning and in particular its application to time series in general and with wind in particular.

The central theme of this dissertation comes from joining two ideas, the importance of forecasting for the renewables and the yet unlocked capabilities of deep learning in this field. The integration of these two areas, artificial intelligence and earth sciences, guides the motivation of this research, that starts from a series of research questions.

The first research question is the fundamental idea and primary motivator.

### **Can deep learning improve the traditional prediction methods accuracy on wind time series and become a tool for Wind forecasting?**

The objective is to contribute to a better understanding of the application of deep learning to the multi-step forecast of multivariate Wind Time Series.

Deep Learning algorithms that learn from large quantities of data are showing strong capabilities to extract hidden patterns in the inputs that are difficult to be perceived even for trained humans. For example in [170], Poplin et al. shows how sex detection from eye images is much more accurate performed by deep learning algorithms than made by humans, or He et al. who in the framework of the Imagenet competition created new approaches improving accuracy [83], or Silver et al. who created the Alphazero algorithm, defeating a Lee Sedol, the GO world champion, showing how reinforcement learning can support sophisticated strategies [195]

Deep learning is based on neural networks used as building blocks in diverse structures and forms [79]. These Networks are based on Neuroscience principles, as they try to replicate how the human brain works. The original idea in this field was to reverse engineer the neuron and its connections, and these first artificial neural networks have evolved into complex and sizeable deep learning models surpassing human skill in some tasks, thanks to the massive availability of software and computing power [27].

Artificial neural networks have shown high effectiveness in many challenging applications, and this discipline is leapfrogging traditional approaches obtaining accomplishments that were unexpected just a few years ago.

Deep learning is currently applied to many areas, and one of them is the application to time series prediction. Different sciences deal with time series, like health sciences, engineering or econometrics, all of them are trying to understand how deep learning will revolution its principles, and this is where this work can be framed, in trying to shed some light in how deep learning can help in the modelling of wind multivariate time series. At this point, the application of deep learning to wind forecasting is not widespread [134, 141], motivating the need for further research. This Thesis proposal tries to contribute to the understanding of how deep models can help to obtain better wind forecasting.

The definition of deep learning is quite broad as it includes many different algorithms, with different behaviours and many variants. This diversity is at the centre of this work, as we designed and tested many different deep learning modelling approaches, to determine which are the best-suited ones when applied to wind considering this our second research question:



### **Which are the best deep learning Algorithms for Wind Time series prediction and Why?**

In this work, we have defined novel architectures using three basic building blocks, Multi-layer perceptrons, Convolutional and Recurrent Neural Networks, and combinations between them. The challenge is to be able to compare methods in a way that is unbiased on the data, and this is not an easy task, being the wind a local phenomenon, that can be easy to predict in one site (with stationarity air patterns) or extremely chaotic on another site [73]. The intimate relationship between terrain features and wind is well proven, and this work has confirmed it in many results, implying that a limited validation on a geographically homogeneous set of observations is not enough to establish the superiority of an approach over others. For this reason, we have chosen to work with the NREL Wind dataset which, with 126,692 sites, is the largest and more heterogeneous open wind dataset with time-series observations available today. With this dataset, which has all kinds of terrain sites distributed in the US, the algorithms work on many different site topologies and characteristics. By testing with this comprehensive example, we can establish correlations between site complexity and modelling, and thus the next research question arises:

### **Which kind of relationship have the methods with site characteristics?**

Wind depends on site roughness, geographical location, altitude, and some more features, but the wind patterns change over time, due to different climate transitions and due to the effect of climate change in our atmosphere. The available data for this work has seven years of length, and one research question that has appeared during the research is related to changing wind patterns. The learning algorithms have identified pattern changes in wind, but can they assess meaningful changes over the seven years? These considerations have motivated our next question:

### **Are the series long enough to allow the deep learning modelling algorithms to identify Climate Change patterns on the wind?**

During this research, our understanding of wind variability and site complexity has increased, which has allowed us to develop a new conceptual idea, the forecastability. We define it as the ease or complexity of forecasting a specific site. There are only naive metrics in the literature, and for this research, more robust approaches have been defined, based on stationarity time series analysis and spectral signal decomposition. These measures have been tested and correlated to the deep learning approaches and used for the description of a wind site complexity. Then, our next question is:

### Can some measures from spectral or time series analysis be a good predictor of wind site complexity?

To forecast the accuracy of a site from the time series structure is known as forecastability. We have explored several indexes for its forecastability and identified promising features in the series that have strong correlations with prediction accuracy. We analysed roughness as a predictor of accuracy, leading to our last question:

### Is terrain topology a good predictor of a deep learning model accuracy when applied to the site?

This work has used three topics; wind, time series and deep learning as prime ingredients trying to answer relevant questions in the intersection of the three disciplines, the results foresee potential future applications of the findings to the wind forecasting science.

## 1.2 Thesis dissertation structure

We have organised the document in four parts, each one divided into chapters and sections:

The first part, **Introduction**, sets up a general outline of the Thesis work with a focus in the context and contributions from this work.

The second part, **Foundations**, analyses the state of the art of the different topic areas that configure this work, starts with an analysis of the time series field, with a focus on the wind time series. We describe deep learning and its application to time series modelling, and the most relevant approaches used in the research. The chapter finalises with a state of the art review of the wind forecasting field, reviewing weather based forecasting, time series forecasting, and then going into the application of deep learning to wind time series.

The third part, **Experimentation**, describes the work performed with a description of the general architecture of the experiments and the novel models designed and implemented, followed with a detailed analysis of the results. This part starts describing the data and the general framework of the experimentation and then we include the four main chapters that identify the most significant areas of experimentation performed, the main algorithms, a review of variants on the main architectures, a detailed study of the convolutional network models and finally forecastability measures for the wind prediction activity.

Fourth and last part, **Conclusions**, summarises the different research areas and conclusions of this thesis work. Further on it outlines a set of ideas for future research directions, which have not been analysed in this thesis, but are reasonable extensions of this work.

Some results of this work have been published in proceedings of peer-reviewed conferences and journals, contributions that are listed in Section 1.3. Some of these articles are self-cited in the content. All the published works and conference material is entirely original and has been created from this thesis research project.

## 1.3 Thesis Contributions

In this section we describe the main contributions generated from this PhD work. Contributions have been in the form of Journal or Conference articles published in different proceedings, posters presented in conferences, presentations at conferences and, finally, software created for the experiment general setup and organisation.

### 1.3.1 Scientific Journal and Conference publications

Several articles in Scientific Journals and Conferences have been produced as by-products of this thesis and are cited in different parts of the dissertation and included in the list of references.

#### Journal Publications generated from this dissertation

- (Jour-2) J. Manero, J. Béjar, and U. Cortés. "Dust in the wind...", deep learning application to wind energy time series forecasting. *Energies*, 12(12):2385, 2019. doi: 10.3390/en12122385
- (Jour-1) J. Manero, J. Béjar, and U. Cortés. Wind energy forecasting with neural networks. a literature review. *Computación y Sistemas*, 22, 2018. ISSN 2007-9737. number 4

#### Journal pending Publications from this dissertation at time of thesis deposit

- (Pend-1) J. Manero. Convolutional networks for wind speed multi-step time-series forecasting. "tbd", 2020
- (Pend-2) J. Manero. Predicting the prediction. site forecastability for 12 hours ahead multi-step wind prediction using deep learning in north-america. "tbd", 2020

#### Conference Publications

- (Con-3) J. Manero, J. Béjar, and U. Cortés. Deep learning is blowing in the wind. deep models applied to wind prediction at turbine level. *Journal of Physics: Conference Series*, 1222:012037, May 2019. doi: 10.1088/1742-6596/1222/1/012037
- (Con-2) J. Manero, J. Béjar, and U. Cortés. Predicting wind energy generation with recurrent neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11314 LNCS:89–98, 2018
- (Con-1) J. Manero, J. Béjar, and U. Cortés. Go with the flow: Recurrent networks for wind time series multi-step forecasting. *Frontiers in Artificial Intelligence and Applications*, 308:79–83, 2018

## **Context, motivation, structure, and contributions**

---

The candidate has presented as a speaker in different congresses.

### **Congress Speaker**

(Spe-2) IRC-2019 : 8<sup>th</sup> International Energy Agency (IEA) International Research Conference 26–28 June 2019 at The Danish School of Education, Aarhus University in Copenhagen

(Spe-1) IDEAL 2018 : 19<sup>th</sup> International Conference on Intelligent Data Engineering and Automated Learning. Nov 21, 2018 - Nov 23, 2018. Madrid Spain.

The candidate has presented Posters in several congresses (for illustration see Appendix C)

### **Congress Posters**

(Pos-3) In energy3canada 2019 conference, (Canada conference on energy with a focus on offshore wind), Halifax, Nova Scotia Canada, October 16-18 2019.

(Pos-2) WindEurope 2019 & Exhibition, (European Wind Industry Conference) Bilbao, Spain, April 2-4 2019.

(Pos-1) 21<sup>st</sup> International Conference of the Catalan Association for Artificial Intelligence, CCIA 2018 October 8-10, Roses, Girona, Spain

## **1.3.2 Other contributions**

During his PhD program, the candidate has collaborated with other research teams in the use of deep learning in time series applications and has generated contributions in several journals and conferences.

### **Deep learning for intrusion with time series anomaly classification and detection**

The candidate is collaborating with Dalhousie University (Canada). The focus of the research is the application of deep learning to cyber-security. The application of deep learning classification techniques to network packets time series is an approach similar to time series forecasting.

This collaboration has allowed testing some of the time series approaches designed for the Wind Prediction field to network intrusion detection and generating several journal publications and opened some new areas for research at the Faculty of Computer Science (Systems, Networks and Security department).

### Journal pending Publications from this dissertation at time of thesis deposit

- (Pend-5) D. Upadhyay, J. Manero, M. Zamanand, and S. Sampalli. An intrusion detection system for power grids based on a gradient boosting algorithm with feature selection. *tbd*, 2020. Article submitted, pending publication
- (Pend-4) A. Alsirhani, J. Manero, S. Sampalli, and P. Bodorik. A DDoS detection framework: Deep learning in a distributed system cluster. *tbd*, 2020. Article submitted, pending publication
- (Pend-3) D. Upadhyay, J. Manero, M. Zamanand, and S. Sampalli. Majority vote ensemble algorithm for intrusion detection in power grids. *tbd*, 2020. Article submitted, pending publication

### Previous Work

Finally, in the framework defined by the PhD coursework, initiated some years ago, the author collaborated in published papers on the application of Fuzzy Logic to bibliographic information databases. These are works performed outside the core interest of this thesis, but as we considered them valuable, we highlight one of those publications.

### Publication

- (Jour-6) R. L. de Mántaras, U. Cortés, J. Manero, and E. Plaza. Knowledge engineering for a document retrieval system. *Fuzzy Sets and Systems*, 38(2):223–240, 1990. ISSN 0165-0114. doi: [https://doi.org/10.1016/0165-0114\(90\)90151-U](https://doi.org/10.1016/0165-0114(90)90151-U). Fuzzy Information and Database Systems

### 1.3.3 Graduate teaching experience

During the Research visit at Dalhousie University, the author has given Seminars to Master, and Doctorate students included in the official MS and PhD Curriculum of the Faculty of Computer Science (credits valid for graduate students).

- "Deep Learning applied to Time Series Forecasting". This seminar was CSCI 6999 eligible, to fulfil curriculum requirements for Master & PhD students in Computer Science, and open to all faculty members
- "Introduction to Deep Learning". This multi-day seminar was open to Graduate and Undergraduate students and faculty from different departments at Dalhousie University and introduced the basic components of Deep Learning and its application to several applications and problems.

## Context, motivation, structure, and contributions

---

During the thesis work period, the author has held an assistant professor position at the IE University teaching a Deep learning course at the official Master program "Master in Management specialisation in Big Data" a pioneering program that helps to understand the new developments in Artificial Intelligence to the new business leaders of tomorrow.

### 1.3.4 Code and Result Notebooks

We share, in an open github repository, a set of jupyter notebooks that contain the results and the code used for the experimentation. These repositories are located in:

**NOTEBOOKS:** <https://github.com/castorgit/Articles-2020>

**CODE:** [https://github.com/castorgit/Wind\\_code](https://github.com/castorgit/Wind_code)

All the research and experimentation has been made using open source tools. Code and specific information about approach and experimentation methodology and processes is available to any researcher under request.

## Part II

# State of the Art

---

*"The most important lesson from 60 years in AI is that the most difficult tasks (disease diagnosis, play chess at master level, etc.) have been quite easy to solve, while tasks that seemed quite easy have shown to be the most difficult ones"*

(2015) - R. López de Mántaras [56]

*"I cannot command winds and weather"*

(1800) Vice-Admiral Horatio Nelson [157]





## Chapter 2

# Time Series

Time is one of the natural dimensions of nature. Data, as an image of our world, must represent it, and one way to do it is by representing the periodic measures over time, in the so-called time series. Time series are sequences of observations obtained at some specific time interval. We find time-stamped data in many areas of our everyday lives, like weather reports, banking records, stock valuations, housing indexes, health measures, grades in school, astronomical data, sea tides, and others.

Time series are long lists of data with recorded observations at predefined time intervals. Humankind, since the inception of time, has tried to analyse data series of events to find inner patterns that allow to predict the future. In this way, we managed to forecast star movements, weather phenomena, health events, all of them just by recording and analysing a long list of data. Forecasting time series is a crucial element in the decision-making process in many industries, a critical skill for many applications.

In this chapter, we analyse the time series formalisation, and then we describe some relevant properties with an emphasis on the main forecasting approaches. Then we describe the use of deep learning techniques used for modelling and prediction of time series. The final section is devoted to wind time series, the central data structure used in this thesis work.

### 2.1 Introduction to Time Series

Time series were defined thousands of years ago by scientists in many ancient civilisations like Babylonia, Egypt, Mesoamerican cultures or in ancient Asian dynasties. They designed sophisticated tables filled with observations that allowed them to forecast astronomical events like eclipses or star movements with astounding precision. All these scientists learned to record periodical observations and then used them to find the internal structure of the data, to predict future events, thus setting up the fundamentals for the first time series theory. They modelled the movements of celestial objects with high precision, and predicted eclipses, comets appearances and planets movements, just from their accurately recorded observations.

In agriculture, they learnt to calculate the harvest seasons and to optimise the annual task cycles. In this way, the crop yields increased by applying the scientific understanding of the calendar, seasons, and moon. Since these early human discoveries, a limitless number of applications have been created by applying prediction and modelling techniques to time series, in almost all areas of knowledge.

Between all the possible applications, one area appears as particularly relevant for the time series field, this being economics. In this area, we can find many time-related data series, like inflation, growth, interest calculations, sales, taxes. In all these sequences of numbers, time is a relevant dimension, forming a time series.

The study of economy series gave birth to the econometrics science, a discipline devoted to model economic-related time series. Econometrics develops methods to analyse time series of data to find relationships and trends and, by applying statistical inference approaches, predict future values from the actual data, performing forecasts that are more or less accurate depending on the defined model strength.

Formally a time series is a temporal sequence of values recorded in fixed intervals (every  $n$  seconds,  $m$  minutes,  $p$  hours). The series then are a discrete representation of an event that is continuous, like an electroencephalograph, a cardiogram, a city temperature, or the temperature in a sensor (see Figure 2.1). The time intervals in the series can vary and are related to the application of the time series, in very short intervals ( $10^{-3}$ ,  $10^{-4}$  seconds), hour intervals ( $10^1$ ,  $10^2$  seconds) or year or even longer ( $10^4$ ,  $10^{10}$  seconds). The period length is a relative characteristic, for an energy load time series, milliseconds can be a reasonable time, but for a planetary orbit prediction, multiyear observations are the reasonable interval.

In a time series, the different steps can have some kind of relationship, and the models we develop have as an objective to identify these relations. Relations that may not be trivial, in this way, they can have an apparent relationship (a linear relationship), a more complex combination (a non-linear function from different step values), or even a no relationship at all if it is a random mechanism [171].

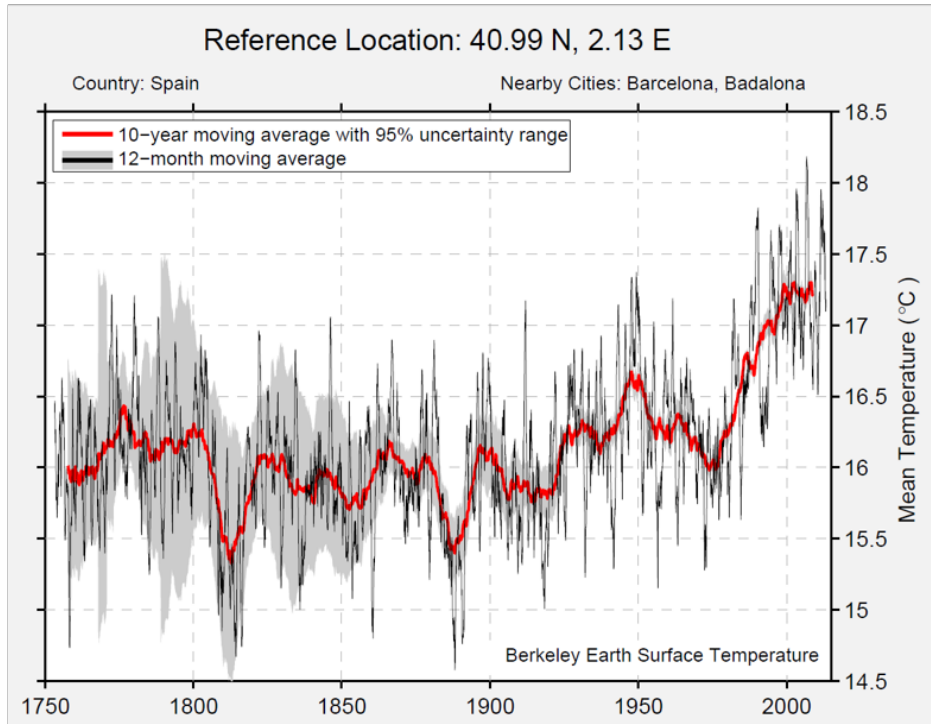


Fig. 2.1 Barcelona average monthly temperature time Series. source: [www.berkeleyearth.org](http://www.berkeleyearth.org)

## 2.2 Time Series definition and representation

Formally we can define a time series  $X$  as a discrete set of tuples or vectors that contain one or several observations belonging to the same time step. Each time series has a fixed period  $t$  between steps, and then we name the steps in the following manner  $t_i \in T$  where  $T \in (t_1, t_2, \dots, t_n)$  or in a more general way  $T \in \mathbb{N}$ .

Time series can be either univariate or multivariate, in the first one, each observation is an individual value, wherein the second one, each step is a list of values, where each value is an observation.

Wind time series are usually multivariate, as each time step contains several observational measures, corresponding to meteorological observations from different sensors.

An example of a multivariate wind time series  $x_i$  is:

$$X : \langle x_1, x_2, \dots, x_n \rangle \quad (2.1)$$

Where each step is a tuple or list of  $m$  values.

$$x_i : \langle a_1, a_2, \dots, a_m \rangle \quad (2.2)$$

and each tuple is composed of several observations:

$a_1$  : wind speed  
 $a_2$  : temperature  
 $a_3$  : wind speed at hub level  
 $a_4$  : wind speed at ground level  
....  
 $a_n$  : others

The observations or values on each time step can either be categorical or numerical. Categorical values usually define a qualitative property and can be completely unrelated or disconnected. An example could be a score in a test (A, B, C, D, F), or a financial rating (A++, A+,...), or many more. A numerical value can be a whole number  $v \in \mathbb{N}$  or a real one  $v \in \mathbb{R}$ .

## 2.3 Time Series Properties and Models

Modelling time series consists of defining a mathematical model that can act as a *function model* that fits the data sequences. For a given time series  $T : (t_1, t_2, \dots, t_n)$  a model will be a function  $f(t)$  where  $f(1) = t_1, f(2) = t_2, \dots, f(n) = t_n$ . A model can be obtained by regression analysis techniques (if the values are real numbers) or using classification algorithms (if the values are categorical). Modelling of time series using regression requires a good understanding of the series properties, and to develop this understanding is at the centre of the forecasting science.

In this chapter, we analyse different time series properties useful for forecasting, and then we review some of the existing modelling approaches, then we focus on the specific properties of wind time series.

### 2.3.1 Stationarity and Ergodicity properties

The first time series property we analyse is ergodicity. To define it we assume that in a series  $t$  of length  $n$  the expectation  $[E_t] = \mu$  and the variance  $V[x_i]_1^n = \sigma^2$  are constant for the complete series  $n \rightarrow \infty$ . Ergodicity is a property that materialises when a part of the time series describes the variance and mean of the whole process.

Two different classes of ergodicity can be defined, depending on the impact in the mean or the variance. Ergodicity in mean:

$$\lim_{T \rightarrow \infty} E \left[ \left( \frac{1}{T} \sum_{t=1}^T x_t - \mu \right)^2 \right] = 0 \quad (2.3)$$

Ergodicity in variance:

$$\lim_{T \rightarrow \infty} E \left[ \left( \frac{1}{T} \sum_{t=1}^T (x_t - \mu)^2 - \sigma^2 \right)^2 \right] = 0 \quad (2.4)$$

Another relevant property is stationarity, present when the joint distributions of random variables from a process are time-invariant.

A time series from a stochastic process is:

- Mean Stationary: If  $\mu$  is constant
- Variance Stationary: If  $\sigma^2$  is constant.
- Covariance Stationarity:  $\text{Cov}[x_t, x_s] = E[(x_t - \mu_s)(x_s - \mu_s)] = \gamma(|s - t|)$

As variance stationarity immediately results from covariance stationarity for  $s = t$ , a stochastic process is stationary when its mean and covariance are stationary.

These properties cannot be proven as we are not able to work with the infinite time series, and thus, they have to be assumed. However, for a time series to have ergodicity, it has to show mean, variance or covariance stationarity.

In summary, if some stationarity is present in mean, variance or covariance, the stochastic process will have ergodicity [109]. An example of such process can be *white noise*<sup>1</sup>.

To measure the stationarity in a time series, we can use the augmented Dickey-Fuller test (ADF) which tests the null hypothesis of a unit root present in the series. The alternate hypothesis explanation is stationarity of the series. By using this test, it is possible to establish if a series is stationary. For instance, in the stock exchange example, the ADF test will show the non-stationarity of the series.

In statistics and econometrics, an augmented Dickey-Fuller test (ADF) tests the null hypothesis that a unit root is present in a time series sample. The alternative hypothesis is different depending on which version of the test is used but is usually stationarity or trend-stationarity. It is an augmented version of the Dickey-Fuller test for a more extensive and more complicated set of time series models.

The augmented Dickey-Fuller (ADF) generates a negative number that indicates the rejection of the hypothesis, where the more negative the result is, the more significant rejection of the assumption or hypothesis will be, in this example, the test statistic rejects the stationarity of the series as it has a -121.684 value on the Test Statistic.

### 2.3.2 Decomposing a Time Series in Trend and Seasonality

When the series show regular fluctuations in fixed periods, it has seasonality. In many econometric series, cycles have an impact in the series. For instance, if we observe that in

---

<sup>1</sup>If a series has no correlation between values, and constant variance is observed it is defined as *white noise*, if the observations follow a normal distribution then is *Gaussian white noise* (source: [153], [193]).

**Table 2.1** Dickey Fueller test results (statsmodels/\* library package)

<b>Results of Dickey-Fuller Test:</b>	
Test Statistic	-121.684243
p-value	0.000000
Lags Used	0.000000
Number of Observations Used	14384.000000
Critical Value (1%)	-3.430805
Critical Value (5%)	-2.861741
Critical Value (10%)	-2.566877

a perfume sales time series, Christmas has a stiff peak in sales, and sales in summer are low, then the series has a strong seasonal pattern. In another example, we know that solar radiation changes between day and night, and we have more radiation in summer versus winter, thus sun generates a time series with two components, daily and summer-winter seasonality.

Inside the time series, some internal components define these seasonal patterns, meanly seasonal and trend. The trend is the long-term increase or decrease of the series values and, in some cases, it may contain fluctuations related to cyclical increases or decreases overlaid on top of others. With some techniques, time series can be decomposed in its internal components.

Usually, series are decomposed in three components, trend, seasonal and residual  $X = Tr_t + S_t + \epsilon_t$  where  $Tr_t$  is the trend,  $S_t$  the seasonality and  $\epsilon$  is the residual element.

As trend and seasonal are elements of the time series, there are several methods defined to decompose a time series in those components, being one the most common the classical decomposition (moving averages) method (see Algorithm 2.1). However, there are other methods like the STL (Seasonal and Trend decomposition using Loess).

---

**Algorithm 2.1:** Time Series decomposition

---

**Data:** initial Series  $X : \langle x_1, x_2, \dots, x_n \rangle$

**begin**

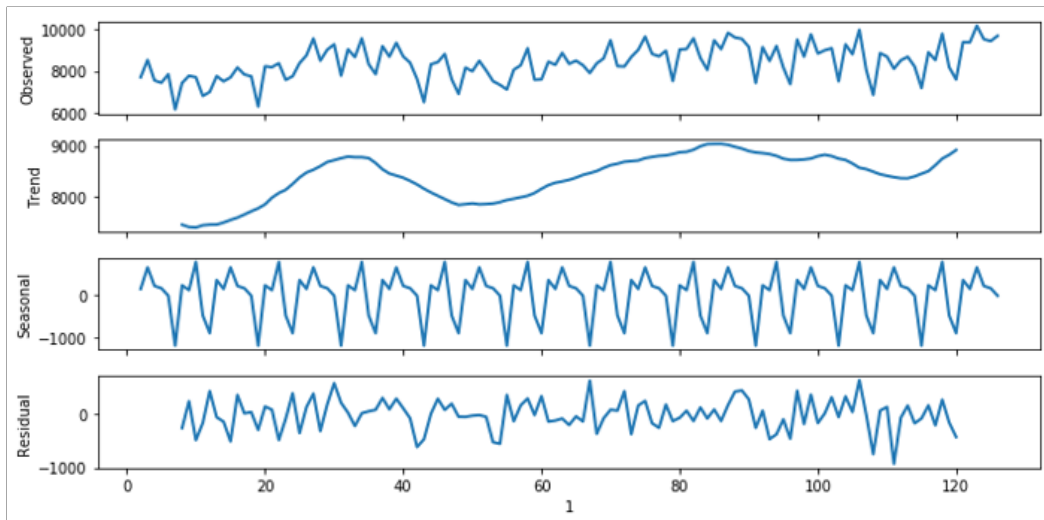
- | Trend is found using moving averages method
- | Subtract trend from the series
- | Calculate Seasonal component by averaging values on each period
- | The remainder trend is obtained by subtracting the trend and seasonality from original series

**end**

**Result:** Trend, Seasonal and Remainder series

---

## 2.3 Time Series Properties and Models



**Fig. 2.2** Decomposition time series example, NN52 series from NN3 competition (source: <http://www.neural-forecasting-competition.com/NN3/>)

Figure 2.2 illustrates how a time series is decomposed in trend, seasonal and residual components.

One way to cope with non-stationarity is to transform the series in a way that we extract the non-stationary components eliminating the trend, by statistically differencing the series.

Statistical differentiation for a given series  $X$  consists in generating a new Series  $X'$  where  $x'_i = (x_{i+1} - x_i)$ .



**Fig. 2.3** Three Time series corresponding to the closing stock value of IBM. The first is the raw value, the second applying logarithm, both are non-stationary. The third one is the differenced example is stationary

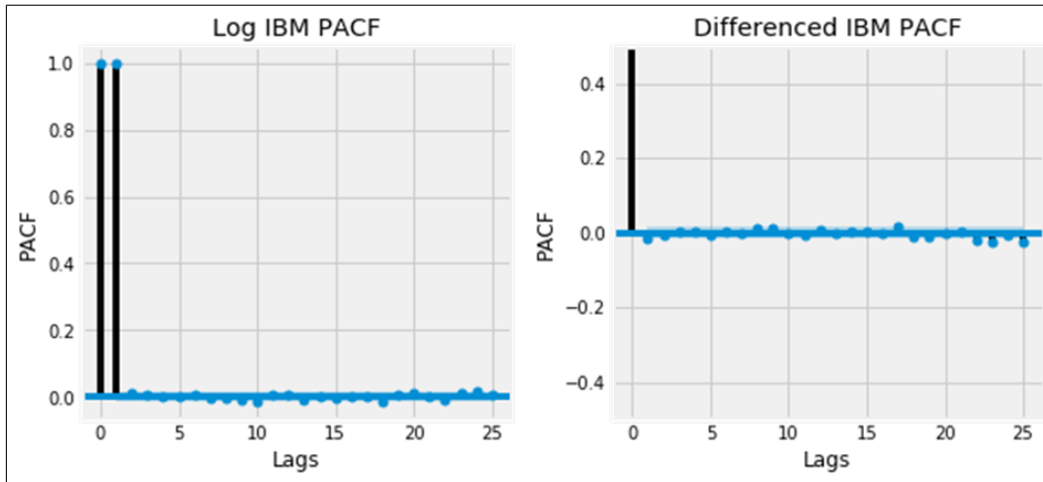


Fig. 2.4 The series has internal correlation however, the differenced series has no correlation

The Partial Autocorrelation Function (PACF) obtains the correlation in a stationary time series. This function finds a correlation in the lagged values inside the series.

Using the Partial Autocorrelation Function (PACF) in the example in Figure 2.4, we can see that the two series correlate. However, if the series are differenced, then the correlation disappears.

### 2.3.3 Linearity in time series

Linearity is a property in a time series where each data step can be obtained from a combination of other steps or differences of steps.

This property tells the ability to use linear models to represent the series if a linear model can model a time series, if the series are linear, however, if we consider linearity as a property, then we require a method for the calculation of the linearity degree of a time series. There is not a single method to rate the linearity, as there are different approaches in the literature, described in Section 2.4.2.

In general, linear methods have two components Auto-Regressive (AR) and Moving Average (MA) (or both).

$$y_t = a_t + \underbrace{\sum_{m=1}^M b_m y_{(t-m)}}_{\text{AR component}} + \underbrace{\sum_{n=0}^N c_n x_{(t-n)}}_{\text{MA component}} \quad (2.5)$$

Models that are a combination of both approaches are Auto-Regressive Moving Average (ARMA). If we perform differentiation in the series, then the model is an Auto-Regressive Integrated Moving Average method (ARIMA), which is the most general linear method.



### 2.3.4 Time series as a stochastic process

From a theoretical point of view, a time series can be formalised by the use of probability theory. A time-series  $\mathbf{X} : \langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T \rangle$  has a corresponding multivariate distribution, this distribution is a series of random variables  $(\mathbf{x}_t)_{t=1}^T$  which define a *stochastic process*. A specific time series then is a subset of the whole series  $\mathbf{X}$  from  $(1 \dots n)$  and, if the series is infinite or very long, it can be considered as the single instance of the whole stochastic process. Otherwise, there can be many instances of the whole process, each instance being a time series.

## 2.4 Forecasting Time Series

Predicting a time series can be considered a regression problem, as the traditional way to perform a forecast is to develop a model that is an empirical representation of the time series, this model will represent a hypothesis on the probability of the values of the steps to be predicted. Forecasting then consists of an optimisation problem to obtain the best parameters for the model. In this sense, the definition of the model is critical and must be verified experimentally to obtain the best fitting approach for each time series characteristics.

Forecasting methods can be classified using different criteria. An initial taxonomy can be created based on the statistical properties of the forecast, in two major classes, point forecasting and probabilistic forecasting. A second possible classification can be made based on the number of steps to predict, with two significant strategies, single step and multiple step forecasting. A third classification is obtained depending on the characteristics of the prediction model, where the main ones are linear and non-linear models.

In the next sections, we analyse different forecast classifications taxonomies, going deeper into the state of the art of the different forecasting techniques.

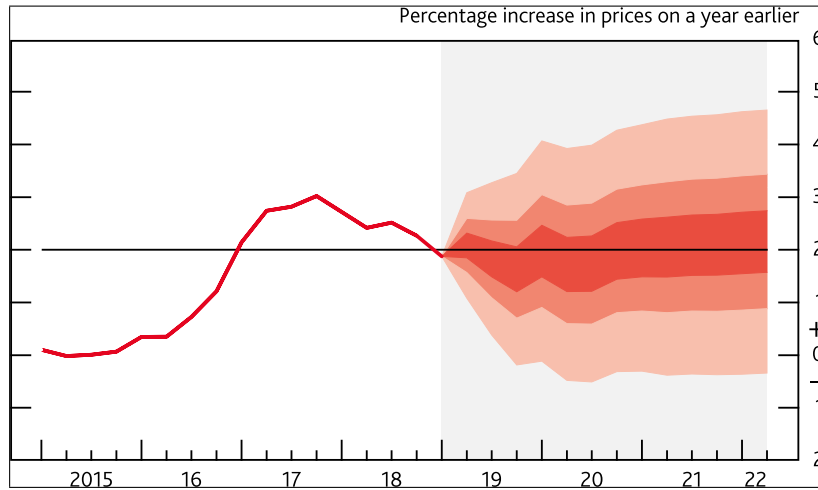
### 2.4.1 Point and probabilistic forecasting

To forecast a single value in the future is a point forecast, and for a multi-step, it means predicting a value for each future step. Nevertheless, there are alternatives to this design, the main one being probabilistic forecasting.

A probabilistic forecast quantifies the uncertainty in a prediction by adding a confidence interval to the predictions. We find examples in several areas of science, and specifically in the wind speed prediction as in [169] or [251].

A probabilistic forecast generates a probability distribution instead of a value for each prediction point. There are two approaches to build the probabilistic distributions. The first one is to transform a point prediction into a probabilistic by using a statistical conversion, like considering the point prediction as the mean of a Gaussian or a Poisson distribution, the

second approach consists in using an ensemble of methods that generate a distribution of values [75].



**Fig. 2.5** Bank of England probabilistic forecast of inflation as a percentage increase in the consumer price index. The shaded bands in the fan chart show prediction intervals in increments of 10%.<sup>2</sup>

This thesis is based on point forecasts as the complexity added by using probabilistic distributions would have increased the computing requirements of the work due to the increase of data to manage, nevertheless, we believe is an interesting line of work that can be researched in future research projects.

The transformation of point forecast into a probabilistic has two major approaches, parametric and non-parametric. The parametric approaches use a prior model to fit the distribution, like a Gaussian or Weibull distribution where non-parametric use algorithms like direct quantile regression or kernel density estimators.

We can find some applications that apply a probabilistic forecast to wind in the literature, like the work by [Gneiting and Katzfuss](#) in [75] where they describe an application of probabilistic forecasting to wind energy. In this case, the point prediction, generated with information from three neighbouring sites, is transformed into a probabilistic prediction by using a truncated Gaussian distribution. In another recent work [Shi et al.](#) in [192] develop a probabilistic prediction based on an RNN neural network that applies Prediction Intervals (PI) as a non-parametric method, testing the model with the Adelaide wind farm dataset from Ontario, Canada [97].

### 2.4.2 Linear and non-linear forecasting methods

Linearity is a critical property in the time series when it comes to prediction, as this property defines some of the method characteristics and behaviour.

<sup>2</sup>Source: [www.bankofengland.co.uk/inflation-report/2019/may-2019/prospects-for-inflation](http://www.bankofengland.co.uk/inflation-report/2019/may-2019/prospects-for-inflation) .

Linear methods applied to linear time series have been thoroughly analysed in the literature because most series in the science of economics have this property. The Box-Jenkins approach is based on linear series. However, the maturity of basic linear approaches and the irruption of new forecasting requirements on non-linear datasets generates the need for the development of alternative methods.

### 2.4.2.1 Linear forecasting methods

Linear models have been studied extensively. For instance, the AR models were defined as early as 1927 [244], and have been extensively used, improved and sophisticated since then.

The general construction of linear models is described in Equation 2.5, using two components, the AR and the MA component.

Both models can be used independently or combined. We start by analysing the independent MA model where the modelling equation has only the MA component of order  $q$ .

$$y_t = a_t + \sum_{m=1}^q b_m x_{(t-m)} + \epsilon_t \quad (2.6)$$

When the AR model is used independently, the equation shows only the AR component. An AR model of order  $p$  shows this structure:

$$y_t = a_t + \sum_{m=1}^p b_m y_{(t-m)} + \epsilon_t \quad (2.7)$$

In this model  $a_j$  are real constants,  $p$  is the order and  $\epsilon_t$  are the zero-mean uncorrelated random variables that form the so-called *white noise*. Both models show good results when representing linear series and combined evolution to an Auto-regressive moving average model or ARMA model by replacing  $\epsilon_t$  by an average of  $\epsilon_t, \epsilon_{(t-1)}, \dots, \epsilon_{(t-n)}$

$$X_t = a_0 + \sum_{i=1}^p a_i X_{(t-i)} + \sum_{i=0}^n b_i X_{(t-i)} \quad (2.8)$$

ARMA is a well-known method that shows excellent robustness for linear time series modelling but presents limitations on series that show non-stationary (see Section 2.3.1 behaviour).

In the '70s Box et al. defined a new model that reduced the stationarity of data by differencing the series [22].

Series differentiation consists of computing the differences between consecutive observations into a new series where:

$$y'_t = y_t - y_{t-1} \quad (2.9)$$

Differentiation can be recursively applied, and by applying it, the stationarity of the series is removed. Each differentiation order removes a different stationarity type (first-order time differences remove a linear drift, second-order removes a polynomial trend, and so forth),

We define a new model by using differentiation, this is the nonseasonal ARIMA model, which is defined by three values to become the ARIMA(p,d,q) where the coefficients mean:

- p is the number of auto-regressive terms,
- d is the number of nonseasonal differences needed for stationarity, and
- q is the number of lagged forecast errors in the prediction equation.

The forecasting equation which is constructed by differentiation works as follows

$$\begin{aligned} \text{If } d = 0: & \quad y_t = Y_t \\ \text{If } d = 1: & \quad y_t = Y_t - Y_{t-1} \\ \text{If } d = 2: & \quad y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2} \end{aligned}$$

The Second difference is the recursive application of differentiation on the first order one, and the general ARIMA(p,d,q) equation will be:

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} - \theta_1 \epsilon_{t-1} - \dots - \theta_q \epsilon_{t-q} \quad (2.10)$$

The term ARIMA or ARMA can be confusing, as the AR and MA components have the same mathematical form. They are both linear combinations of present and past values of random variables. The general form comes from the Box Jenkins approach. The general approach can be described as:

$$\mathbf{X}_t = \phi_1 \mathbf{X}_{(t-1)} + \phi_2 \mathbf{X}_{(t-2)} \dots \phi_p \mathbf{X}_{(t-p)} + \epsilon_t + \theta_1 \epsilon_{(t-1)} + \dots \theta_q \epsilon_{(t-q)} \quad (2.11)$$

A time lag operator  $L$  is defined as:

$$Lx_t = x_{(t-1)} \quad \text{for all } t \in \mathbb{Z} \quad (2.12)$$

and transforming the equation using the time lag operator, plus Differencing it (in statistics differencing is a transformation applied to time-series data to make it stationary). The stationary time series property does not depend on the time at which the series is observed  $y' = y_t - y_{(t-1)}$  or  $y'' = y_t^* = y'_t - y'_{(t-1)}$ .

$$\left(1 - \sum_{j=1}^p \phi_j L^j\right) X_t = \left(1 - \sum_{j=1}^q \theta_j L^j\right) \epsilon_t \quad (2.13)$$

The Equation 2.13 states that time series  $X_t$  depends on a linear combination of past observations plus a moving average of series  $\epsilon$  forming the general equation origin of the ARMA and ARIMA models. In the ARMA(p,q) model p is the order of the linear regression and q is the order of the moving average errors.

The ARIMA(p,d,q) model is a more general equation

$$\left(1 - \sum_{j=1}^p \phi_j L^j\right) (1 - L)^d X_t = \left(1 - \sum_{j=1}^q \theta_j L^j\right) \epsilon_t \quad (2.14)$$

where d is the order of the differentiation. The letter I in ARIMA name refers to the fact that the dataset has been initially differenced. There are some already defined ARIMA models (see Table 2.2).

Table 2.2 ARIMA(p,d,q) models

Arima Model	p	d	q	Description
ARIMA(0,0,0)	0	0	0	White Noise
ARIMA(0,1,0)	0	1	0	Random walk
ARIMA(0,1,2)	0	1	2	Damped Holt's model
ARIMA(0,1,1)	0	1	1	Basic exponential smoothing

Its accuracy comes from the training strategies utilised in the algorithm, which offers reasonable accuracy in short prediction times. ARIMA methods are found in hybrid combinations with all kinds of other methodologies, like fuzzy, Bayesian transformations, SVM (vector machines) or Gauss process regressions that support the model differenced data.

### 2.4.2.2 Forecasting series from decomposition

Decomposing the time series into its trend, seasonality and residual components allow to handle its complexity. The prediction of the individual components is more straightforward as we eliminated the combined effects. It is possible to decompose the series, then model each component isolated, and then combine them back into a result becoming the single forecast. This approach obtains better results than approaching the combined series monolithically.

We illustrate the general algorithm for forecasting from a decomposition in Algorithm 2.2, and some reference to its application is in [215].

### 2.4.2.3 Non-linear time series forecasting methods

The AR, MA, and ARIMA approaches are all linear methods, suited for linear models. When used with non-linear series, they obtain consistent results for light linearity, but if this property is much stronger, then their results are not feasible, and we need to use non-linear models.

---

**Algorithm 2.2:** Forecasting from a Decomposition

---

**Data:** initial Series  $\mathbf{X} : \langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \rangle$

**begin**

Decompose time series into  $X_t = Tr_t + S_t + \epsilon_t$

Compute deseasonalised series initial Series  $\langle \mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_n \rangle$  by

$\mathbf{A}_t = X_t - S_t = Tr_t + \epsilon_t$

Forecast  $\mathbf{A}_t$  obtaining  $\langle \hat{\mathbf{A}}_{T+1}, \dots, \hat{\mathbf{A}}_{T+H} \rangle$

Forecast  $\mathbf{S}_t$  obtaining  $\langle \hat{\mathbf{S}}_{T+1}, \dots, \hat{\mathbf{S}}_{T+H} \rangle$

Combine results by  $\hat{\mathbf{Y}}_{T+1} = \langle \hat{\mathbf{A}}_{T+1} + \hat{\mathbf{S}}_{T+1} \rangle$

**end**

**Result:**  $\hat{\mathbf{Y}} = \langle \hat{\mathbf{Y}}_{T+1}, \dots, \hat{\mathbf{Y}}_{T+H} \rangle$

---

The extensive experimentation and development of linear models dwarfs the advances in non-linear approaches, which are much less studied in the literature, but with a large number of approaches.

The number of methods in non-linear modelling is vast, and range from non-linear autoregressive methods (NARX) [217], spectral analysis methods [103], state-space and hidden Markov chain models [225], autoregressive conditional heteroskedasticity (ARCH) models to the more recent addition of Machine Learning methods like SVM or neural networks [59].

Neural networks can be combined with autoregressive and moving average components like the non-linear ARMA (NARMA), and the autoregressive neural network (ARNN) to obtain the best of both worlds [212, 211], but usually are used independently.

Neural network approaches are appropriate for non-linear series and can bring its powerful representation capabilities into the non-linear forecasting task.

### 2.4.3 Single-step and Multiple-Step ahead forecasting

Single-step forecasting consists of forecasting a single point in the future. Usually, the forecasting methods are single-step, as they only predict one value in the future; however, forecasting multiple steps in the future is multi-step forecasting.

Multi-step ahead forecasting consists in forecasting a time series of horizon  $H$  (or  $H$  time steps) from a time series that contains  $t(1, 2, \dots, t)$  past observations. Single-step forecasting is the case where the prediction is for a single point (at distance  $H$  into the future and can be applied to any future single step as it is illustrated in Equation 2.15.

$$\langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t \rangle \rightarrow \langle \hat{\mathbf{y}}_{t+H} \rangle \quad (2.15)$$

$$\langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t \rangle \rightarrow \langle \hat{\mathbf{y}}_{t+1}, \hat{\mathbf{y}}_{t+2}, \dots, \hat{\mathbf{y}}_{t+H} \rangle \quad (2.16)$$

Other applications focus on the multi-step approach, where the forecast objective is to obtain a future time series of a predetermined length of  $H$  (see Equation 2.16). The multi-step prediction can be generated using different methods.

- **Recursive:** This method is based on using a prediction step as input for the successive prediction, recursively until the horizon is reached.
- **Direct:** The direct method uses different methodologies depending on how the predictions are obtained. One approach consists of training a model for each step to be predicted, and the result is the concatenation of all individual predictions. This method optimises the error for each time-step separately, although in a computationally-expensive way. An alternative methodology consists of a model that generates multiple outputs by training the model to minimise the error for all the time steps in the horizon at the same time. This method is also known as the Multiple Input Multiple Output (MIMO) approach. A compromise between the two approaches is also possible by obtaining separate models that predict subsets of steps.

Some works in the literature describe the superior results obtained by direct methods over recursive ones [208] [33], specifically in non-linear applications, nevertheless, these findings are based on empirical assumptions, as there is lack of theoretical work in this area, and the superiority of one approach over the others is based on specific examples tied to a problem or a data origin. For this reason, we find opposite conclusions from different problems, due to different characteristics of the data used for the prediction.

Atiya et al. in [5] find better results with the direct strategy over recursive applying neural networks to time series of river flows. Kline in [110] finds better results with the direct approach versus the recursive one with some experimental tests based on the M3 competition<sup>3</sup>, or Chevillon in [33] develops a comprehensive analysis, for linear models, on which one performs better, showing the dependency of each model with the data characteristics.

In linear models, we consider that an implemented model is the true model when the length of the series is infinite  $t \rightarrow \infty$ , in this case, the multi-step forecast using a recursive model will perform with optimal results, becoming the best approach. However, the assumption of a true model is not valid for most problems, and in this cases, the recursive problem may not be the best approach, when the modelling cannot be considered the true model (*misspecified* modelling) the direct regression is superior to recursive.

For non-linear models, the problem gets more complicated, as there are several recursive approaches for non-linear modelling that can improve the results like the naive recursive, the Monte-Carlo or the Bootstrap methods [213].

The question on which approach generates better accuracy remains open, as it depends on several conditions like the characteristics of the original time series, the horizon  $H$  to

<sup>3</sup>M3 competition is an econometric forecasting competition performed periodically and managed by Spyros Makridakis at Georgetown University [136].

predict or the number of steps to predict, usually  $(1, \dots, H)$ , but could be  $(n, \dots, H)$ . The accuracy depends on the modelling technique.

We are interested in machine learning models, and in this category of models, there are some pieces of evidence of better results obtained by using the direct approach [14, 15, 208]. In this thesis work, we have performed comparisons of different approaches applied to wind time series confirming the superior results of direct MIMO approaches (see Chapter 7).

## 2.5 Signal theory and time series analysis

Signal analysis is an area in engineering (usually in electrical engineering) that deals with time-dependent measures from continuous systems like the electric voltage, noise waves, radar sensor and many more. This field overlaps with the time series forecasting field. The signal theory approach instead of trying to develop models that will be used to forecast future steps, it tries to model the signal to extract some information from it, with different objectives, like to develop a noise reduction filter on a voice signal.

The signal analysis thus deals with quite similar data types as time series analysis, but with an emphasis on the internal structure of the series, In this field, the series are usually sampled with very short periods ( $10^{-3}$  seconds for instance). From this area of knowledge, we imported approaches used in this work (see Chapter 10) to find measures and concepts that are helpful to describe the wind time series, applying some entropy and spectral analysis.

Spectral analysis has as an objective to *extract* components from the series that have structure, like its trend or its oscillatory components, and split those parts from others that have no structure or just noise. This technique is successfully applied to time series as we can see in [76].

Entropy analysis comes from Shannon works published in his article from 1948 [189] the basis for entropy analysis related to information communication theory. The main goal for the entropy to measure is the amount of information that can be transmitted in a message, this theory offers tools to evaluate the amount of information present in a signal, and this can help to identify if a time series has some internal structure or if it is just random noise (see Equation 10.9 in Section 10.7).

## 2.6 Wind time series

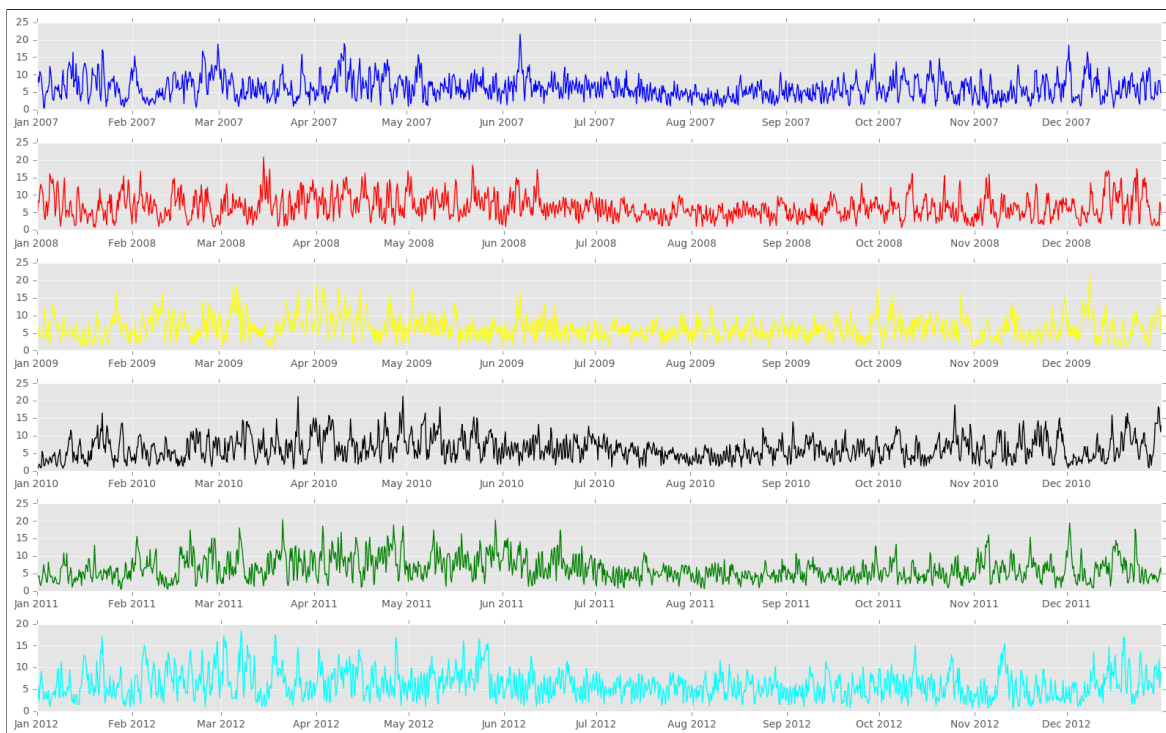
Wind time series come from observations of several weather measures from a specific location. The relevant observations, in a non-exhaustive list, are wind speed, wind direction, temperature, humidity, barometric pressure, solar radiation or power generated by a real



or theoretical turbine). The Wind time series then are composed by tuples of data recorded every  $t$  units of time, that can be milliseconds, seconds minutes or hours.

One characteristic that these series have, which is relevant for modelling, is that they can have lots of noise derived from its high dimensionality. To solve these issues, we can filter the data using signal processing techniques, like filtering or wavelet analysis. Another characteristic is that we do not know how many variables are required in the time series, as the combination of features may be too correlated and bring imperfect information to the predictive task [134].

Graphical representations are useful to illustrate complex sets of data. To show the fluctuations of a wind speed time series in a specific location, we can use a two-dimensional plot like the Figure 2.6 and to see how the dominant wind direction and intensity changes in a place we can use a wind rose representation as in Figure 2.7



**Fig. 2.6** Wind speed time series in a site located in Techado, New Mexico each row corresponds with one year of measures, in the illustration row is one year of data starting 1 January 2007 and ending 31 December 2012<sup>4</sup>

Time Series forecast methods are critical for the GRID operators (see Chapter 1). as they have to assure the integration of renewable energy into the electricity systems. The operators develop complex prediction systems to manage this integration. As an example, Red Eléctrica Española (REE), the Spanish TSO, has engineered a forecasting system named

<sup>4</sup>Wind site located in Techado, New Mexico (USA), latitude N 34°32' 12" longitude W 108°20' 59", from the NREL dataset (see Section 5.2).

SIPREOLICO [186], Germany uses the Wind Power Management System (WPMS) [49], or Denmark uses various prediction tools to determine wind production in an area that comprises the Baltic and continental Denmark [234].

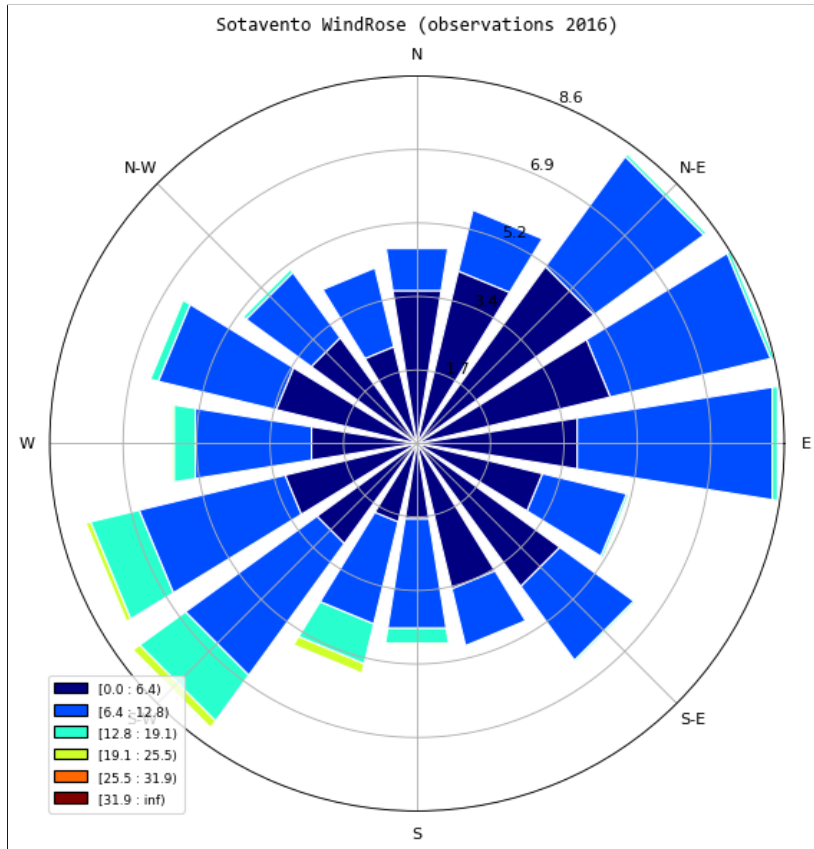


Fig. 2.7 Wind direction dimension in one year of time series measurements in the Sotavento Park located in Galicia, Spain 2016 [43]

A wind time series is a time-stamped sequence of several measures related to wind. The most usual observations measured are; wind power ( $MW$ ), wind direction ( $degrees$ ), air pressure ( $Pa$ ), wind speed ( $m/s$ ), temperature ( $C$  or  $K$ ), air density ( $kg/m^3$ ), relative humidity (%), these last two are volume and pressure air relationships and expressed by the formula  $\rho = p \div R_{air} T$  where  $R_{air}$  is the specific gas constant of air  $287.058 J/(kg \cdot K)$ .

All these observations can be generated at different heights (floor, hub height, half-height) as the wind at 100 meters high (hub turbine height) is the one that moves the blades, becoming the highest relevance measure, while wind direction is essential to understand how the dominant winds might impact wind patterns and intensity. In Figure 2.7 a summary of one-year data from the Sotavento wind park is shown using a wind rose representation, and it can be easily observed the dominance of E/NE and W/SW winds for this specific site.

**Table 2.3** ADF test on two NREL dataset sites (using Adfuller test from statsmodels)

Offshore New Orleans	Edgeley North Dakota
turbine: 3007	turbine: 112500
latitude: 28.580738	latitude 46.292343
longitude -90.734619	longitude -98.736877
ADF Statistic: -31.418378	ADF Statistic: -44.676385
p-value: 0.000000	p-value: 0.000000
Critical Values:	Critical Values:
1%: -3.430	1%: -3.430
5%: -2.862	5%: -2.862
10%: -2.567	10%: -2.567

### 2.6.1 Non-Stationarity of wind time series

Stationarity in a time series is understood as the property where some statistical characteristics such as the mean, the variance or the internal correlation, are constant over time or repeated over time in some repeated frequencies like daily, monthly, weekly, summer/winter.

There are several tests used to analyse the stationarity of a time series. The Dick-Füller (ADF) test and its evolution, the augmented ADF, are the most common [44]. The ADF looks for a *unit root* in a time series sample. A *unit root* is a statistical feature that determines randomness in the series. The ADF Tests sets up a hypothesis that there is a *unit root*. The more negative is the result, the higher the rejection of the hypothesis, and the probability of the time series being non-stationary increases. In Table 2.3 an example of the application of an ADF test to a wind time series is shown. The negative ADF shows clear non-stationarity in two sample turbines in the NREL dataset.

When this test is applied to a time series, if the result is positive, it will show stationarity, oppositely, if the result is negative, then the hypothesis of non-stationarity is confirmed, and then the series is considered as non-stationary. Wind time series is in many physical locations non-stationary, but in some places (steady winds or evident seasonal trends) it can show light stationarity, and the site will be easier to forecast (see Section 10.8).

Stationarity in time series is difficult to assess, as this property can happen in subsets of the time series. For instance, if we only see the winter, it could be highly stationary, but if we see winter and summer we can observe significant differences without stationarity, for this reason, it is crucial to analyse in large series the existence of these properties.

For shorter series, we can find the issue of *co-variance shift* [201] this problem is present in supervised learning when the validation, test and training datasets have strong statistical differences. In this research we avoided this issue by using series with enough data (a 7 years dataset), then we split the dataset into three subsets (training, test and validation) assuring the statistically homogeneity between the three.

### 2.6.2 Non-Linearity of wind time series

Linearity is another relevant property to be found in the wind time series. Linearity is present if the use of linear forecasting methods can model the series, and this property states that each time step  $X_t$  is a linear combination of past or future values.

There are several models that represent these linear combinations like like the auto-regressive (AR) or the moving average (MA) methods.

For instance, an auto-regressive  $p$  linear combination will have this structure:

$$X_t = c + \sum_{i=1}^p \varphi X_{t-i} + \epsilon_t \quad (2.17)$$

or in a moving average order  $p, q$  method

$$X_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \dots + \theta_p \epsilon_{t-p} \quad (2.18)$$

The validation of linearity in a time series is not an easy and straightforward task. The surrogate data method, described by Theiler in [214] is one tool to validate linearity. This test applied to wind time series shows that linearity, in general, is not found in wind time series, and correlations are found in differenced data [68].

If the wind is non-linear, how can linear models be used for forecasting? The answer lies in the fact that the wind series contains inner structures that may be linear. Proper forecasting methods will extract these structures, and by *learning* these patterns will improve the forecasting results.

### 2.6.3 Distribution function in wind time series

Usually, wind distribution functions are approximated using a Gaussian approximation. However, the wind does not fit precisely into a Gaussian curve.

To analyse the distribution function, we represent the wind speed values of a time series, and we try to fit different functions.

The best approximation is obtained by a Weibull function [226]. The Weibull function has the following formulation.

$$f(v) = \frac{k}{c} \left(\frac{v}{c}\right)^{k-1} e^{-\left(\frac{v}{c}\right)^k} \quad (2.19)$$

where  $k$  is the shape (it has no units), and  $c$  is the scale parameter measured in m/s. The cumulative distribution function  $F(v)$  offers the probability that a given wind will exceed

---

<sup>5</sup>Consecutive wind sites from California to Florida crossing the US Geography - sites 5000 to 5499 in the dataset, fitted with the scipy Weibull fit method.

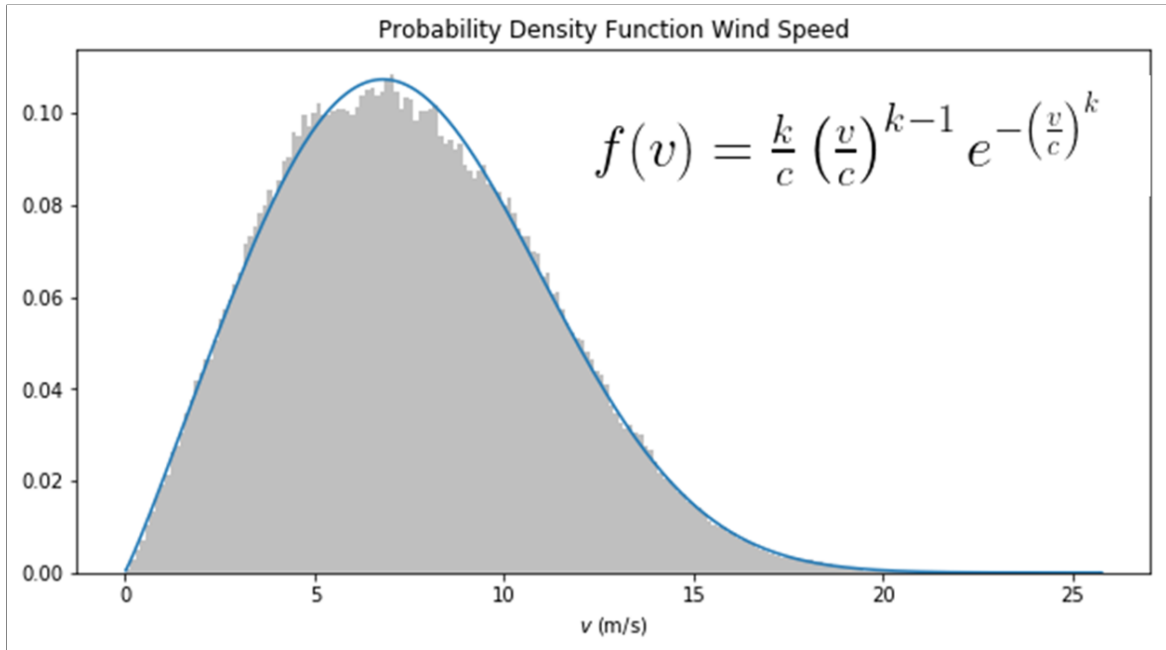


Fig. 2.8 PDF of 500 turbines fitted with a Weibull distribution <sup>5</sup>

the value  $v$  and is:

$$F(v) = e^{-\left(\frac{v}{c}\right)^k} \quad (2.20)$$

A Weibull distribution with  $k=1$  equals an exponential distribution, and with  $k=2$ , we obtain a Rayleigh distribution.

We can see empirically how wind speed follows this distribution by fitting a curve to a set of wind observations, in this case by using 500 wind sites from the National Renewable Laboratory Wind dataset (see Figure 2.8). This distribution is relevant to calculate the energy generation potential of a specific wind site [28], and it has been used in combination of Neural Networks to develop models for wind prediction [101], by combining the Weibull distribution with a simple Neural network, and obtains some improvements over the prediction without fitting the Weibull PDF.

Carrillo et al. in the article [28] fit the wind time series using a Weibull curve, and then, taking into consideration the turbine power generation function, uses it to calculate the potential wind resource in a geographical area. The fitting process, in this case, is not straightforward as it tries to include the wind turbine conversion modelling into the fitting calculations. The solution obtained is the Part Density Energy Method (PDEM) which takes into consideration the cut-in and cut-out points in the energy generation curve in the probability distribution function.



## Chapter 3

# Deep learning and time series

Feature representation consists in extracting the most relevant and characteristic elements hidden a set of raw data. The machine learning methods depend heavily on this feature representation process, a step that by pre-processing the input increases the efficacy of the machine learning model used after. The feature representation step is a time consuming process that requires a good understanding of the subject area and the use of the right algorithms to optimise the data transformation.

Deep learning methods, composed of multiple non-linear transformations, generate simpler representations of data that are combined to form complex hierarchical representations by the use of deeper architectures. Deep learning is a subset of machine learning, but with the capability of automatically represent the world by using a hierarchy of simpler representations, thus making the feature representation step automatically. The major contribution of deep learning lies on its ability to extract features from complex data without an explicit feature representation phase using subject area expertise [18, 79].

Deep learning is breaking performance barriers day after day in extraordinary feats that defeat our traditional beliefs on the limits of the computer algorithms. Understanding images, natural language, generating art, in many areas we see results that challenge traditional approaches reaching or overtaking human skill in some tasks. Deep learning is also applied to time series, an area possibly not as flamboyant as recognising a Van Gogh painting in a museum or translating Chinese to Spanish, but extremely valuable in many critical and widely used applications.

This chapter develops some fundamental Deep learning principles and how these basic principles apply to time-series forecasting.

### 3.1 Machine and Deep learning algorithms for time series forecasting

There is some ongoing controversy about the applicability of Machine Learning methods to time series forecasting, as some econometricians claim supremacy of the traditional statistical methods over machine learning algorithms.

Possibly the most verbal expression of this conflict comes from the MM3 competition held in the year 2000 [136]. This forecasting competition had very few entries based on neural networks, and the entries with Neural Networks and obtained mediocre results. One comment about the use of neural networks was *"Artificial Neural Networks may produce poor results if used under certain conditions"* [11], a sentence that can be explained by the lack of software tools, and the limited forecasting experiences with Neural Networks at that time.

After this competition and in the following years, the machine learning field made notorious advances, generating enough interest in the field to perform a competition only for neural networks, the NN3 (2006/2007) competition.

This new challenge tried to find answers to two questions: *"(1) What is the performance of NN in comparison to established forecasting methods?"* and *"(2) What are the current "best practice" methodologies utilised by researchers to model NN for time series forecasting"* the conclusions showed promising advances made by the NN as the organisers concluded: *"The results highlight the ability of NN to handle complex data, including short and seasonal time series, beyond prior expectations, and thus identify multiple avenues for future research"* [40].

The main organiser of these competitions, Professor Makridakis, announced the M4 competition for 2018 [137]. This competition signals the coming of age of deep learning for time series as the winner was a new Hybrid ES-RNN Model presented by Slawek Smyl from Uber [199], [138], in this competition deep learning had finally the opportunity to beat the statistical approaches competing with them hand in hand.

After the controversies and advancements, we can conclude that Neural Networks offer relevant capabilities for time series forecasting applications, and can be a potent tool for commercial wind forecasting.

Wind prediction from time series is a field where traditional statistical algorithms include additional data from weather forecasting models. This approach has been thoughtfully tested and showed its robustness to the industry.

The introduction of neural networks in forecasting appears in the last decade, but despite the rapid development of the field, they are not widely adopted by all the wind forecasters, more prone to the use of large ensembles of methods with statistical and NWP prediction models.

In this section, we review the most relevant literature that shows the application of machine and deep learning to wind prediction.



There is an early work by [Palit and Popovic](#) that applied of feed-forward networks to forecasting with a fuzzy logic [245] angle, good design but with limited testing as when this work was released, there was limited availability of software that could support the development of experimental exercises.

Later in 2010, we find a preliminary review of Machine Learning applications performed by [Ahmed et al.](#) in [1] revisiting the M3 competition with several Machine Learning approaches (for one-step-ahead forecasting), including neural networks.

Finally, we notice that the adoption of Neural Networks in time series forecasting is increasing. The commercial forecasters are starting to incorporate these networks into large statistical ensembles, which allows them to obtain the best of the traditional statistical approaches with the capabilities of the deep learning, obtaining an increase in the overall precision of the forecasts [112].

There is a long catalogue of Machine Learning algorithms. We can cite  $k$ -means, Bayesian methods,  $k$ -NN neighbours or Support Vector Machines (SVM) which are applied to time series and specifically for wind forecasting, along with the artificial neural networks (see the state of the art review in Chapter 4).

Neural Networks are applied to wind forecasting in many flavours, from the basic Multi Layer Perceptron (MLP) constructions [248], or [211], with modifications like Extremely Learning Machines [93] or using alternative models based on Convolutional or Recurrent Networks (see Chapter 4), their adoption is increasing as the understanding of their capabilities increases along with the availability of powerful programming frameworks that allow to develop complex networks with ease.

In the following sections, we develop how deep learning applies to time series forecasting and which models offer the best fit for the wind prediction activity.

## 3.2 Multi-step forecasting with Neural Networks

Deep learning works on different types of structured and unstructured data, like text, numbers, images, voice, video frames, or written language, and, finally, time series. The deep learning algorithms process these data types with different approaches and techniques.

Depending on the number of variables on each step, the series are univariate or multivariate. This property defines the model dimensionality. With time series, neural networks perform two main operations, classification and regression. Predicting future steps in a time series is a regression, as we we obtain values approximating future points of the series.

Depending on the dimensionality of the output we can distinguish between single-step and multiple-step prediction. Single-step is when the regression algorithm predicts one value at a particular horizon  $H$ , and multiple-step is when the regression obtains a sequence of values. The length of the output sequence is the horizon  $H$  or number of forecast steps (See

this work [15] from [Ben Taieb et al.](#) for a complete classification of multiple-step prediction methodologies).

Multi-step forecasting can be performed with different approaches. One technique consists of integrating several single-step predictions and then concatenate all the results obtaining a sequence or vector of values. An alternative is the Multiple Input, Multiple Output (MIMO) approach, wherewith an input as a sequence, we obtain the full output sequence, in one go.

MIMO or sequence to sequence learning is applied extensively in the Natural Language Processing (NLP) field, where it shows excellent performance results. The seminal work on this area was described in a well-known work from [Sutskever et al.](#) just a few years ago in [204], a contribution that signalled the start of a set of developments that revolutionised the NLP field.

There is some literature about the application of deep learning to time series, [67], and even scarcer for wind, however, there is an undoubted interest in the development of this area. For a review of deep learning application to wind time series see Section 4.3.

The fundamental research questions for the application of deep learning to time series forecasting is How accurate is a specific DL architecture with these kind of series?, Which is the best DL architecture for forecasting?. These two questions hoover around this field, some of them analysed in this section.

- [Motlagh and Khaloozadeh](#) propose in [154] an architecture based on the combination of Recurrent Neural Networks plus a Nonlinear auto-regressive (NARX) construction. This model performs two steps ahead predictions on stock exchange series. The combination of algorithms outperforms the individual performance of each one of them.
- In [172] [Qin et al.](#) propose an RNN with an attention mechanism for non-linear time series. This model outperforms NARX constructions, the predictions are for stock exchange data for ten steps ahead.
- In [238] [Xiong et al.](#) analyse different strategies for one step ahead forecasting vs multiple-step and obtains some meaningful conclusions on the application of the MIMO strategy for multi-step prediction.
- In [29] [Chang et al.](#) use an RNN architecture to forecast long sequences afinding dilation as a useful tool for accuracy improvement when used with recurrent networks.
- In [218], [Torres et al.](#) perform time series forecasting using an MLP network for each prediction time step (with  $h$  models for  $h$  time steps), the horizon chosen is 4 hours for 10 minute sampling (24 time steps). This construction obtains good results, measured in MRE (mean relative error) on an electricity consumption dataset.
- In [220], [Torres and Aguilar](#), use an MLP network to forecast wind generation values from a time series of different observations from a site. The data is obtained from a

numerical weather report model. This work concludes that shallow networks have consistent results compared to deep architectures, considering computing power and training time costs. This same group of researchers further develop these experiments adding convolutional networks to the wind energy generation problem using forecast weather observations as input [221].

All these experiences show a promising outlook for deep learning applied to forecasting, either linear or non-linear, time series. To review different approaches to forecasting we will formalise what a prediction is. Let be the forecasting task a regression which approximates a function  $f(x) \rightarrow y$  to a continuous output variable. A continuous output variable is a real number, such as an integer or floating-point value, these values usually correspond to amounts, sizes or other physical measures. Forecasting then, consists of determining the value of the next step into the future, generated from the time series past.

$$f(x_1, \dots, x_n) \rightarrow \hat{y}_{n+1} \quad (3.1)$$

The predicted time step can be just the immediately next one (as in Equation 3.1), or it could be some other single step in the future, this approach is single-step forecasting. But forecasting requires predictions in more than one step. Many functional domains and particularly wind forecasting, require sequences as predictions, and not just a single value.

Multiple-step ahead forecasting can be defined as a multiple regression problem, where the same data has to be used to obtain multiple answers. There are several methods to approach this problem (see Section 2.4.3) analysed in the literature, like those in [32, 208].

The first one is the recursive approach that estimates one model to estimating one step ahead, and the successive steps are calculated recursively using the previous results as actual values. This method has, as a main disadvantage that the error of the prediction propagates to successive predictions. Usually, this method does not yield the best results.

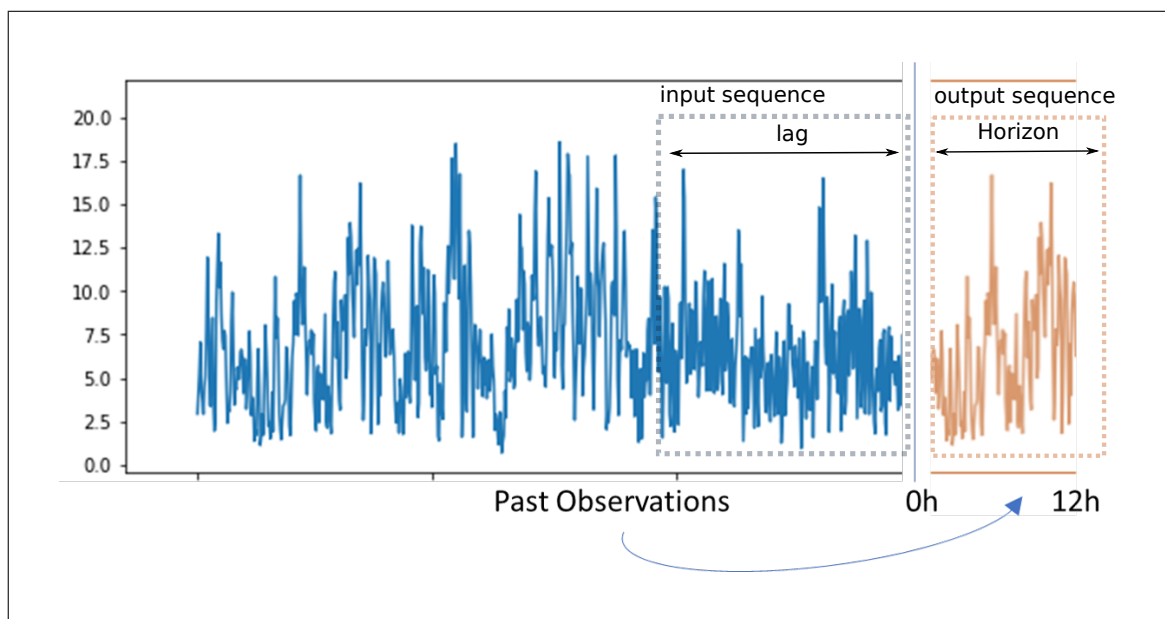
The second approach is the direct strategy. This method has different characteristics, depending on how we obtain the model for the predictions. The first possibility is to develop a model for each step on the horizon to predict. In this case, we train each model to minimise its error for each time step separately, which is a computationally expensive task. The second possibility is to train a model that generates multiple outputs by minimising the error for all the time steps in the horizon at the same time. This method is also known as the MIMO approach. A compromise between these extremes is also possible by obtaining separate models that predict subsets of steps.

To summarise our approach, given the time series  $\mathbf{X}$  of  $n$  elements  $[x_1, x_2, \dots, x_n]$  we want to obtain a prediction for a  $h$  steps ahead (horizon)  $\hat{\mathbf{Y}} = [\hat{y}_{n+1}, \hat{y}_{n+2}, \dots, \hat{y}_{n+h}]$ . With the MIMO approach we perform one regression with one model, obtaining the  $\hat{\mathbf{Y}}$  from the input sequence  $\mathbf{X}$ . In the next section, we develop neural network architectures for MIMO time

series forecasting using the basic models from the main families of networks, Multi-Layer Perceptron, Convolutional Networks and Recurrent Networks.

### 3.2.1 Multiple Input Multiple Output Forecasting

Multiple Input Multiple Output (MIMO) learning happens when the neural network model uses sequences as an input and generates a sequence as an output. In the broad definition of MIMO learning, the sequences can be of unknown length for example as in natural language processing [204], [16], but in the application for wind forecasting, the input and output sequences are of fixed length.



**Fig. 3.1** Input and output sequences in a seq2seq model. *lag* is the length of the input, *horizon* is the length of the output. In multivariate series the time series have multiple measures or observations for each time step

For applications on sequences of unknown length, the encoder-decoder approach is advantageous. It works by transforming (encoding) the input sequence into an internal vector structure, the model transforms this structure and then is decoded into the output. This model obtains very good results in machine translation applications, where the input is a sentence in one language and the output is that sentence in the target language, or in conversational systems, where the input is a question and the answer is the output [204], [79].

With sequences of fixed length, we handle them differently. The network needs to learn a mapping between the past time series (input) and the forecast (output). The inputs are examples of length (*lag*), and the output is the sequence to predict of length (*Horizon*). The network learns from the example sequences and develops a model for modelling the outputs.

The *Lag* is calculated empirically, as this length suggests how much information from the past contains valuable information for the prediction.

### 3.3 Deep learning architectures for time series forecasting

In this section, we review four basic constructions implemented in this research work for MIMO learning, these constructions are MLP, CNN, and Recurrent Networks, these ones with two variants, RNN MIMO and RNN encoder-decoder.

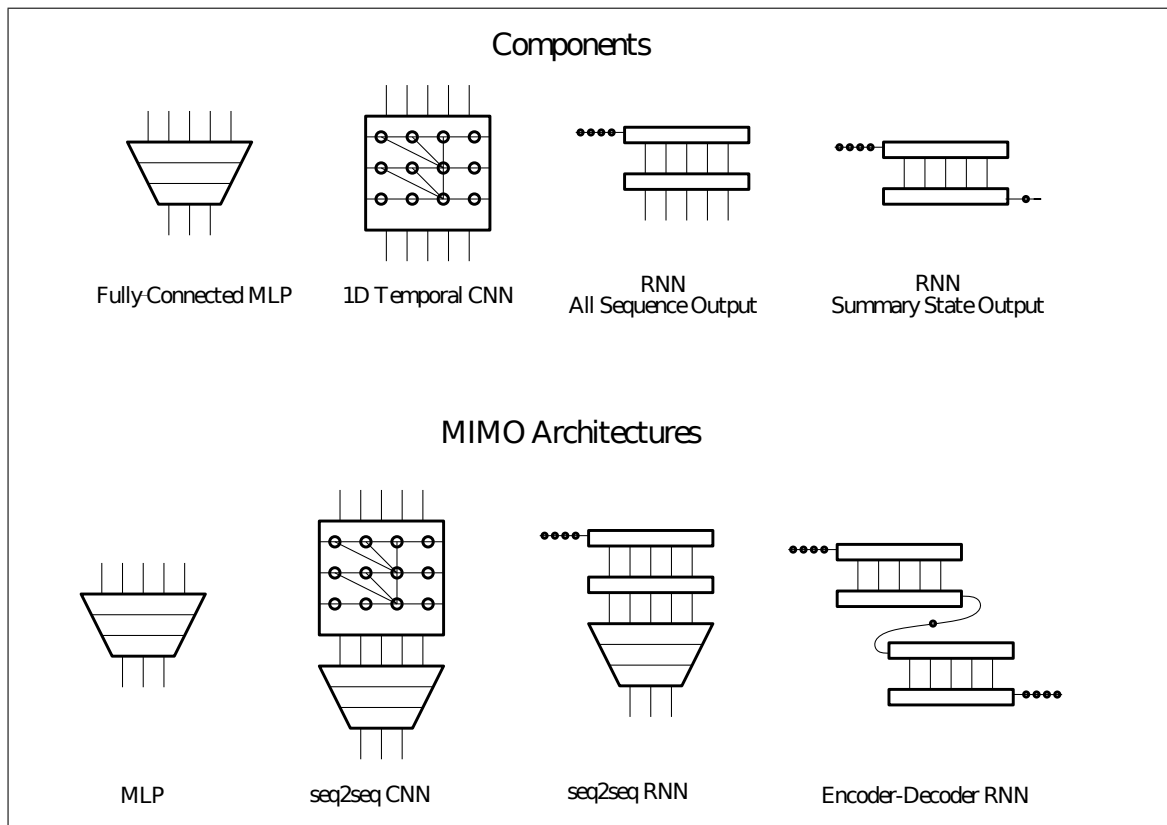


Fig. 3.2 Components and MIMO Architectures for sequence to sequence learning

### 3.4 Multi-layer perceptrons applied to time series

Multi-layer perceptrons can be a good forecaster for multi-step sequence modelling. The multivariate sequences of length *lag* are flattened as input, while the output is a vector with a length *Horizon* equal to the number of steps required by the forecast.

These networks are based on the elemental Rosenblatt perceptron, described as early as 1958 in a pioneering paper [182]. The perceptron has several components that are illustrated in Figure 3.3.

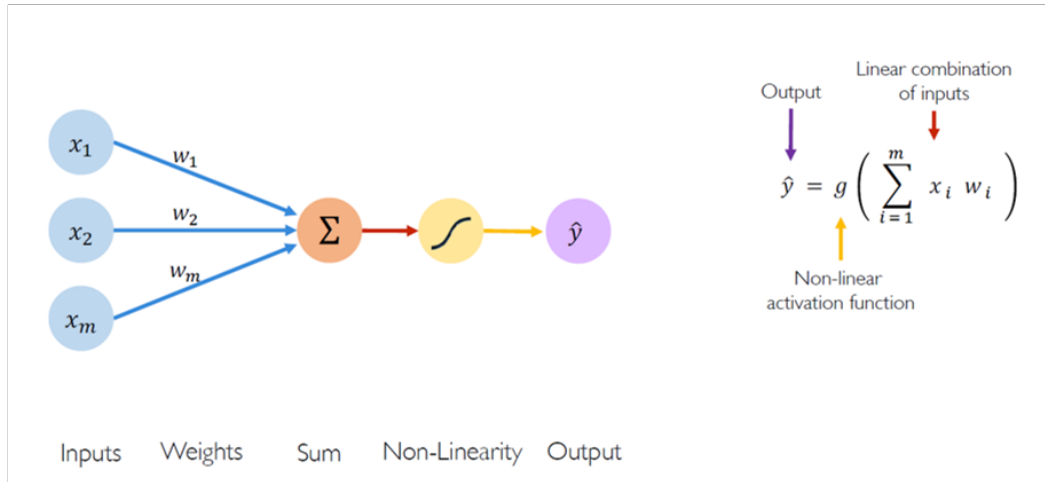


Fig. 3.3 A Perceptron is the basic building block in Multi layer Perceptrons

Multi-layer perceptron networks are fully connected, which implies a large number of parameters to train. Relevant parameters to train are dropout, which improves the generalisation capability of the training, learning rate, batch size, and activation function, plus depth (number of layers) and the number of neurons of the fully connected network. The set of parameters to discover grows as the network is more complex, and as the model variability increases, the number of combinations increases accordingly, and training becomes a harder task [51].

Mathematically the MLP architecture can be formulated as:

$$\mathbf{x}^i = g^i(b^i + \mathbf{W}^i \mathbf{x}^{i-1}) \quad (3.2)$$

where  $i$  is the  $i^{\text{th}}$  layer,  $\mathbf{x}^{i-1}$  is the vector of inputs from the previous layer,  $g^i$  is the activation function of the layer,  $\mathbf{W}^i$  is the weight matrix of the neurons in the layer, and  $b^i$  the vector for the independent terms, or bias for this layer. The MLP networks, and most of the deep learning networks are trained using the backpropagation algorithm, which is the keystone that boosted the neural networks, as it offers a practical approach to set an optimised matrix of weights. The backpropagation algorithm requires an optimisation strategy, being the most common gradient descent, however in this work we have used adamax, a consistent and efficient strategy defined by Kingma and Ba in [108] and widely adopted in many deep learning applications.

The neural network functioning is defined by many parameters that modify its behaviour, like the layer structure, number of neurons, activation, use of connections and many more. The number of combinations exceeds makes impossible to find the optimum set of combinations with trial and error approaches, for this reason we use a hyper-parameter search (see Section 6.4). For instance, in MLP we need to set the parameter depth of the network (number of layers), the number of neurons on each layer, the *lag* of the input examples, or

the activation function used, between others. In Table 7.5 we can see the hyperparameter space defined for the MLP networks.

### 3.5 Convolutional networks for time series

The original idea that allowed the inception of convolutional networks comes from Kunihiko Fukushima who described them in an article in 1980 [63]. In this work, he defined an inspiration of the convolutional networks based on the human perception mechanisms baptized as *Neocognitron*. The name did not hold, but the idea of local feature integration evolved into the Convolutional Neural Network, an approach that burst into the pattern recognition field in 1989 [120], transforming the way computing science understands the artificial vision. Since then, and mainly due to the success of the Alexnet [114], VGG16 and VGG19 [196], ResNet [84] and GoogleNet [205] at the Imagenet challenge [52, 183], convolutional networks have become the method of choice for pattern recognition in images. They are responsible for the revolutionary advances in this area that amazed us some years ago, and now are included into our daily routines in all kinds of applications that make our lives easier.

These networks use a specific kind of layer that applies a *convolution* operation on the input matrix (see Figure 3.6), this layer extracts features from the input matrixes, showing some properties that difference them from the MLP standard networks.

In calculus, convolution is an operation defined by an integral transform of the product of two functions where one is reversed and shifted:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\delta)g(t - \delta)d\delta \quad (3.3)$$

This operation in a discrete space will be:

$$(f * g)(t) = \sum_{-\infty}^{+\infty} f(\delta)g(t - \delta) \quad (3.4)$$

Furthermore, this is the operation that defines a convolutional layer. To perform this operation we need two elements, Input and Kernel, also sometimes called filter. The input is a matrix of data, and the kernel is a matrix of weights. The kernel slides across the matrix performing a summation operation that obtains the convolution result.

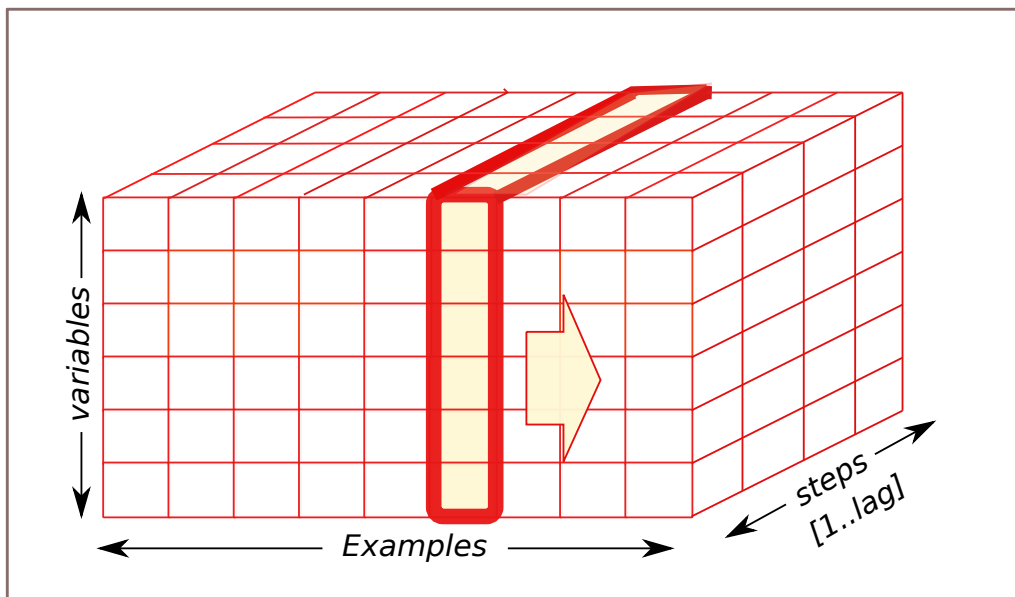
The kernel has a smaller dimensionality than the input. The weights in the kernel come from network training, and they resemble the weights in an MLP layer. We can observe that the number of weights in a kernel is a smaller number than the weights in an MLP layer, making their training more efficient.

The *convolution* operation in a layer consists of two steps. In the first, several convolutions are produced in parallel generating a reduced version of the original matrix. Then in the second step, an activation function is applied to each result, introducing the non-linearity.

After the convolution, it is usual to position an optional pooling layer (with three variants, maximum, minimum, or average). The pooling operation reduces the input and obtains an invariant result. We can consider that the convolutional layer extracts the features and the pooling layer summarises the result making it translation invariant, making easier the feature understanding of the initial image (see Figure 3.6).

The combination of the convolution plus the pooling operation extracts the features from the original matrix, and this output becomes the input of another layer. The result consists of a succession of feature extraction operations (layers plus pooling) that specialise in recognising different properties contained in the input matrix.

To implement the MIMO approach with a CNN network, we combine one or several CNN layers with an MLP network. The CNN extracts the features from the input sequence, and the MLP performs the regression to obtain the output sequence. We illustrate a representation of this architecture in Figure 9.2.



**Fig. 3.4** The wind time series is divided in examples with dimension ( $\text{lag} \times \text{number variables}$ ). The neural network receives sequentially each example as an input



In this thesis, we use CNN architectures with one-dimensional convolutions that process the input sequences. One element of these sequences is a bi-dimensional matrix of depth lag and height equal to the number of variables (see Figure 3.4).

Convolutional networks are composed of a variable number of layers and performs a convolution operation on each one of them. Inputs and outputs on each of those layers can be of multiple dimensions (one, two, three or higher), while the convolution operation can be one, two or three-dimensional as well.

The convolution operation is defined by three elements, kernel, stride and filters which define how is applied. In Figure 3.5 we illustrate a one and a two-dimensional convolutions performed with one and a two-dimensional kernels. The stride property shows how the kernel moves in the matrix.

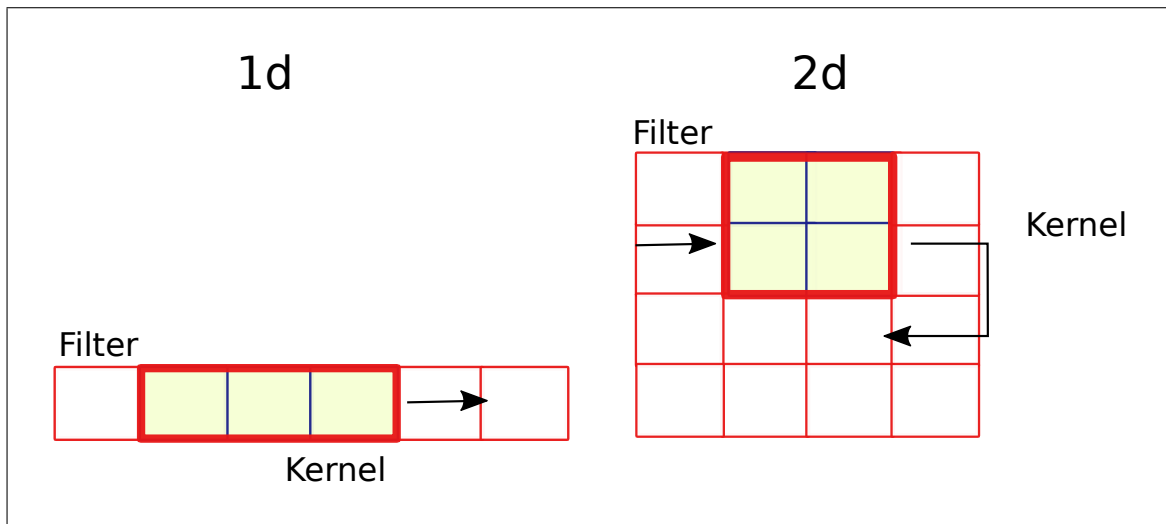


Fig. 3.5 Convolutional operation in 1D and 2D

Filters define the output dimensionality, if filters is greater than 1, then the convolution generates a number of parallel channels. Each one of the output filter matrixes learns a feature of the initial matrix, as the learned weights on each filter are different.

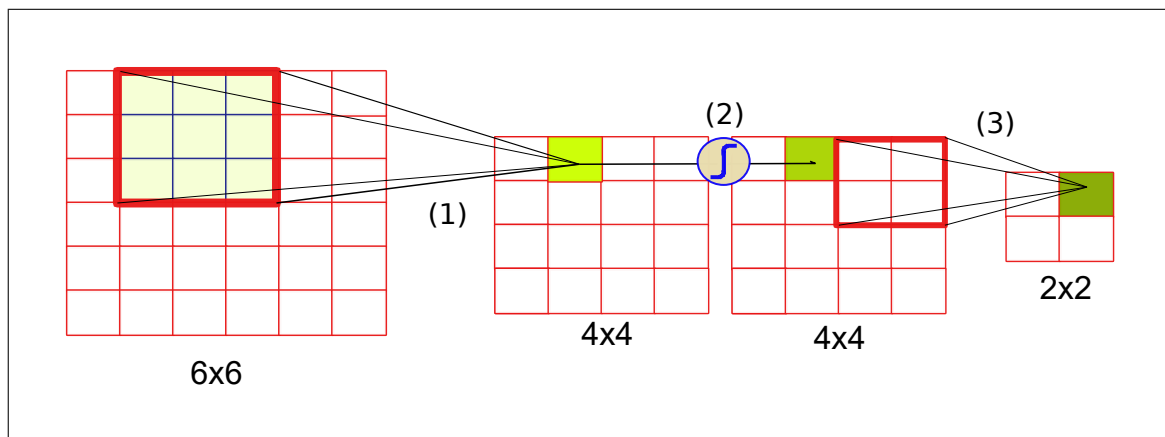
Focused interactions, also called sparse interactions [79], happen when the kernels are smaller than the input, even for a substantial input (a high-resolution image or a long sequence), the kernel looks for a specific pattern or feature in a small part of the input matrix (matrixes are also called tensors). By stacking many kernel layers in sequence, each one extracts different features. Additionally, convolutional networks can obtain results with less processing complexity if compared with MLP networks, as in the standard MLP the number of operations for a layer is  $(m \times n)$  wherein a convolution is  $(k \times n)$ , if the input  $m$  is much higher than the kernel size then the number of operations is reduced by several magnitudes.

Parameter sharing defines the fact that a kernel has a set of common parameters common to all the operation, wherein the MLP version, each connection has a different weight, for

this reason, the number of parameters in a CNN network is smaller, compared to a similar MLP architecture with the same number of layers.

Convolutional layers have an equivariant representation property. Equivariance happens between two functions  $g$  and  $h$ , if they are symmetric like  $g(f(x)) = f(h(x))$ . For a CNN neural network, this means that for two-dimensional inputs like images if this image moves around, the output will move in the same way. This property is handy for image recognition or time-sequenced data because each filter focuses on a feature, and the invariance protects the sequence of actions.

The convolutional layers combine with non-linear activation units (to introduce non-linearity) and max-pooling layers that reduce dimensionality of the output and make the overall process more efficient. A pooling layer reduces the dimensionality of a matrix by



**Fig. 3.6** Convolutional operation plus pooling. (1) application of filter 3x3 on matrix with stride 1 without padding. (2) non-linear activation application. (3) pooling operation

applying a function based on the combination of the interior matrix cells. The functions can be maximum, minimum or averaging. Max pooling maintains the invariant properties and can help to homogenise the output when using inputs of different sizes.

### 3.5.1 Padding in the convolutional operation

A relevant feature used by the convolutional operation is padding (see Figure 3.7). When the convolution operation is applied close to the edges (for a 2D convolution) or at the sequence beginning (for a 1D convolution), the kernel goes out of bounds and exceeds the matrix limits. Padding solves this issue at the edges by adding additional cells (pixels in images, or steps in sequences) on the exterior of the original matrix. Padding has different strategies:

- **Valid padding:** Valid padding consists of no padding: the kernel avoids going off bounds and stops before reaching the borders. This strategy reduces the size of the output matrix.

- **Same padding:** Consists in adding enough cells with zeros at the borders. The output will be of the same size as the input.
- **Causal padding:** Causal padding is used with 1d convolutions and consists in adding zeros at the beginning of the sequence. The input and output sequences are of the same length and cause a 'dilation' that forces the output  $[t + 1]$  depend only in sequences up to input  $[t]$ . Causal padding is practical for sequence forecasting as the convolution operation does not violate the time sequence by using only the past steps [162]. Using padding causal we generate an implicit dilation, and used with sequences the CNN interprets the sequence in order adding a temporal interpretation to the network. For some deeper analysis of dilation applied to neural networks see [243].

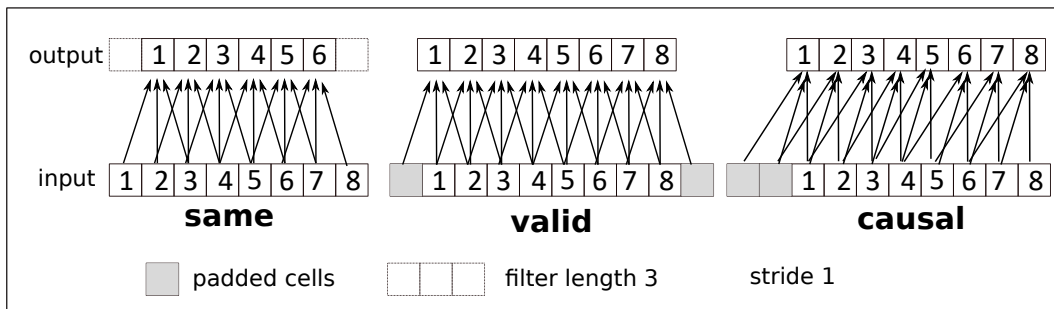


Fig. 3.7 Comparison between different padding strategies, example sequence to sequence with filter size 3

The classic convolutional models include many convolutional layers stacked sequentially or in parallel, and each one of them specialises in different features in the input matrix.

The most relevant parameters in these architectures are *filters* and *strides* which define the convolution, *depth multiplier* which modifies the output channels, *drop* for each channel, and the architecture of the MLP component, number of layers, number of neurons and drop.

As in the MLP, the number of parameters to optimise in one convolutional architecture is very high, and increases along with the number of layers, making necessary the use of a hyperparameter optimisation tool to obtain the best results.

### 3.5.2 Separable convolution

Separable convolutions are a subtle variation of Convolutions created by by Laurent Sifre in [194] and later enhanced by François Chollet in [34] this construction reduces the number of operations to be performed by the network from the standard convolutional approach. The result is a more stable training, improving the accuracy in some cases. Convolutional networks have many variants that make them very versatile for different kinds of data. These variants can be quite efficient for specific data types and applications. Separable convolutions

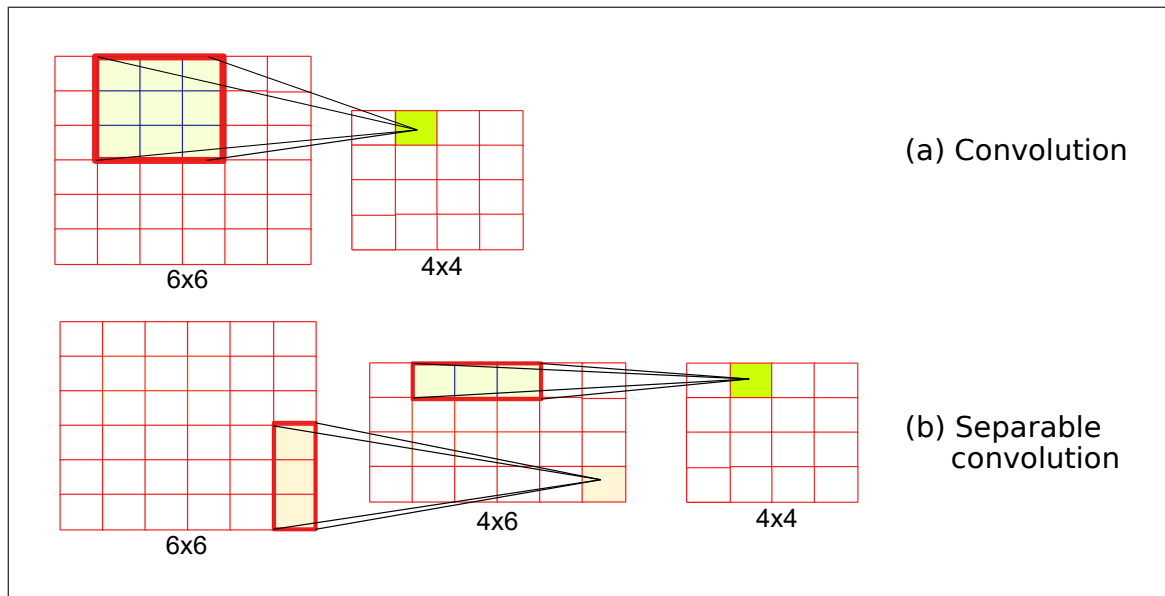


Fig. 3.8 Convolutional 2D separable operation

split the convolutional operation into two steps where two kernels act sequentially. In the standard convolution there is a single kernel in a single step (see Figure 3.8).

Doing the convolution in two steps reduces the number of mathematical operations, thus improving the overall performance of the layer. Filtering an image of  $(m \times n)$  pixels with a filter of size  $p \times q$  requires  $(m \times n \times p \times q)$  multiplications. However with a separable approach it is done in two steps, the first step will have  $(m \times n \times p)$  operations and the second will require  $(m \times n \times q)$  being the total  $(m \times n \times p) + (m \times n \times q)$ , therefore, making the separable approach more efficient.

The approach presented by the Xception (image recognition with separable layer blocks) architecture [34] is to refine the original Inception [206] with separable convolutions. In this case, the first separable convolution is called the *depthwise* convolution and the second the *pointwise*. In the three-dimensional separable operations, the depthwise operation acts on the channels independently, and the point-wise operation mixes the outputs in a new channel space. The obtained separable architecture is not only more efficient but obtains some gains on accuracy compared to the pure Separable approach, results that are consistent with the work presented in in Chapter 9.

Convolutional Networks have been subjected to different refinements, generating variants that show good adaptability to specific conditions, and of these relevant variations is the separable convolution.

Separable convolutions are a subtle variation of convolutions, they were created by Laurent Sifre [194] and later by François Chollet [34]. This variant reduces the number of operations to be performed by the network from the standard convolutional approach.

This reduction impacts in more stable training, and for some applications, there is a slight improvement in the accuracy.

Separable convolutions split the convolutional operations into two steps that combine two kernels instead of one operation with a single kernel (see Figure 3.8). The first step in the separable convolution is called the depthwise convolution and the second the pointwise convolution. In the three-dimensional separable operations, the depthwise operation acts on the channels independently, and the pointwise operation mixes the outputs in a new channel space. For one-dimensional convolutions, used on sequences, the first convolution is depthwise, with multi-variate wind time series each variable is a channel, while the second convolution pointwise only acts in one dimension, as the sequences are uni-dimensional.

Separable convolutions can multiply the output several times, with the depth multiplier. This parameter replicates (multiplies) the output sequence generating deeper sequences. By default, its value is one, and with higher values, increases the number of output channels, which improve the capability of the network to learn more features.

### 3.5.3 Skip and Residual connections

Deep fully connected networks are notoriously difficult to train. The vanishing/exploding gradient is an issue that averts training processes to work properly.

There are different techniques to improve the training stability. One of them is normalisation, which can be applied to the input data, to the batches or to a layer. Data normalisation consists in changing the data elements in the input to a common scale. There are several strategies for this scaling, like rescaling (also called min-max scaling), where data is mapped to a scale of [0,1].

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (3.5)$$

Standardisation or z-standardisation consists of transforming the data to a mean of zero and standard deviation of one.

$$x_{norm} = \frac{x - \mu}{\sigma} \quad (3.6)$$

Other approaches consist of scaling to unit length

$$x_{norm} = \frac{x}{\|x\|} \quad (3.7)$$

The second strategy is batch normalisation, defined by [Ioffe and Szegedy](#) in [99], it is widely used in the main deep learning applications today. this strategy is used for improving the training by stabilising the distributions of layer outputs during the training phase by reducing the internal co-variant shift. A batch normalisation acts on the layer output and

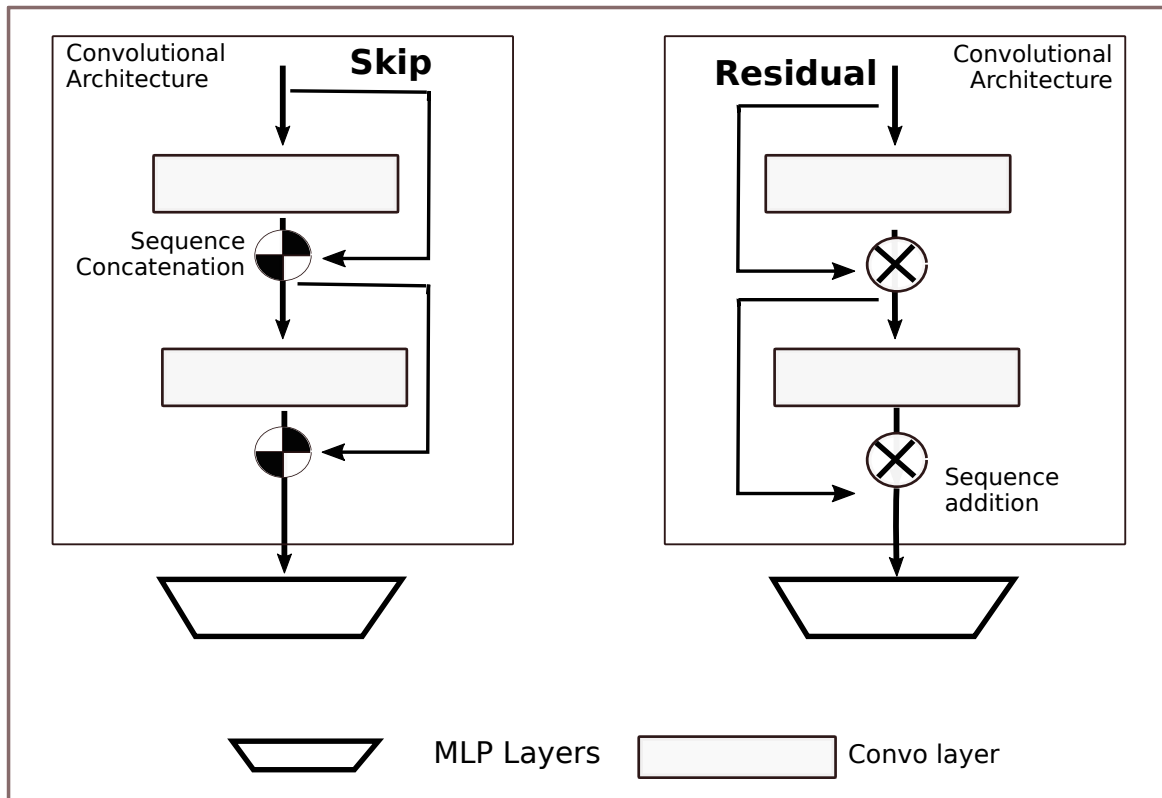


Fig. 3.9 Training stabilisation and Skip and Residual connections

controls the means and variances of those outputs, in a way is like applying the data normalisation strategy to the output of each layer.

The third is layer normalisation, firstly defined by Ba et al. in [7], consists of transposing the batch normalisation strategy to the layers but, in this case, before the non-linearity function is applied. This strategy is very effective with Recurrent Neural Networks.

The vanishing gradient issue problem increases as the depth of the architecture grows, and we can reduce its impact by using another technique called dropout. We briefly described dropout in the Multi Layer Perceptron Section 3.4 where we pointed to its effectiveness. Dropout was first described by Hinton et al. in [89], consisting in eliminating neuron connections randomly in the training process, in this way each training pas is performed with a different layer connection layout, thus reducing the noise in the training. Neural networks have a strong adaptability, and then this ability corrects the voided paths by adapting to the new ones. It is a very effective technique widely used in all kinds of deep nets, reducing overfitting and increasing training effectiveness, is effective with networks with a large number of neurons.

Dropout has generated controversy as Google patented the algorithm in 2016 [88], generating strong dissension in the Deep learning community [184], and maybe signalling a change or course in the actual openness of software and algorithms in this field.

Another technique consists in the use of additional connections between layers as short-cuts, that improve the accuracy by reducing the degradation issues. This construction was first proposed by He et al. in [84] and since then adopted in many applications.

These new connections are added to the model. They are useful as they help to reduce the impact of singularities in the neural network; the most notorious are elimination, overlap and linear dependencies between nodes [163]. Elimination happens when one node is suppressed. Overlap appears when nodes collapse into each other, and linear dependencies may appear due to dependent nodes. These connections add or concatenate the input matrix to the output sequence of a layer (see Figure 3.9). If we combine the sequences with a summation, then we call them residual connections, and if we combine them by concatenation, we call them skip. Concatenating increases the size of the original sequence, while adding maintains the original dimension.

The convolution operation has an issue when working at the edges of sequences or matrixes. The output length can be preserved in the convolution operation by the use of the right padding strategy padding (see Figure 3.7 and Section 3.5.1), which makes it possible to generate an output with the right length.

### 3.5.4 Multi-head models and stacked models

Stacked models are those that combine elementary network blocks formed by several individual convolutional layers. This idea is extensively used for image recognition applications, like the VGG or the ResNet architectures which obtained the best results in the Imagenet challenge in 2014 and 2015 [196, 84].

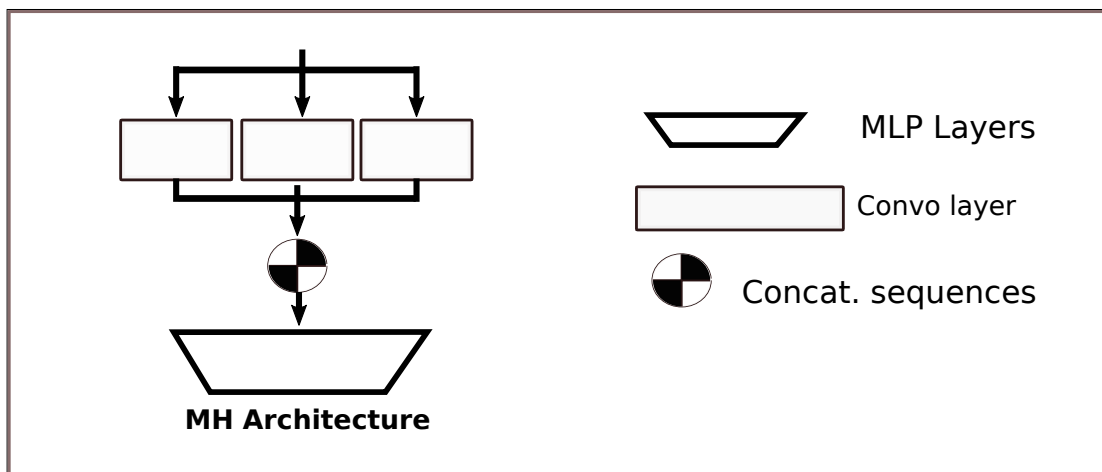


Fig. 3.10 Multi-Head architecture

These models can work with time series too, in architectures like InceptionTime, described in [51], based on the original Inception architecture. The Inception model is the basic building block of GoogLeNet, which started a family of Google architectures that

at the top of the field in image recognition using deep learning [205]. A stacked model is an architecture that combines several blocks, with blocks that are combined sequentially, horizontally or both. The InceptionTime architecture, for instance, combines layers horizontally and sequentially. A horizontal combination of layers is a Multi-Head architecture (see Figure 3.10) and combines the individual blocks in parallel. Each head is a block. As in the sequential combination, each block specialises in a scale of features. Then the outputs from each block are combined.

### 3.6 Recurrent Neural Networks

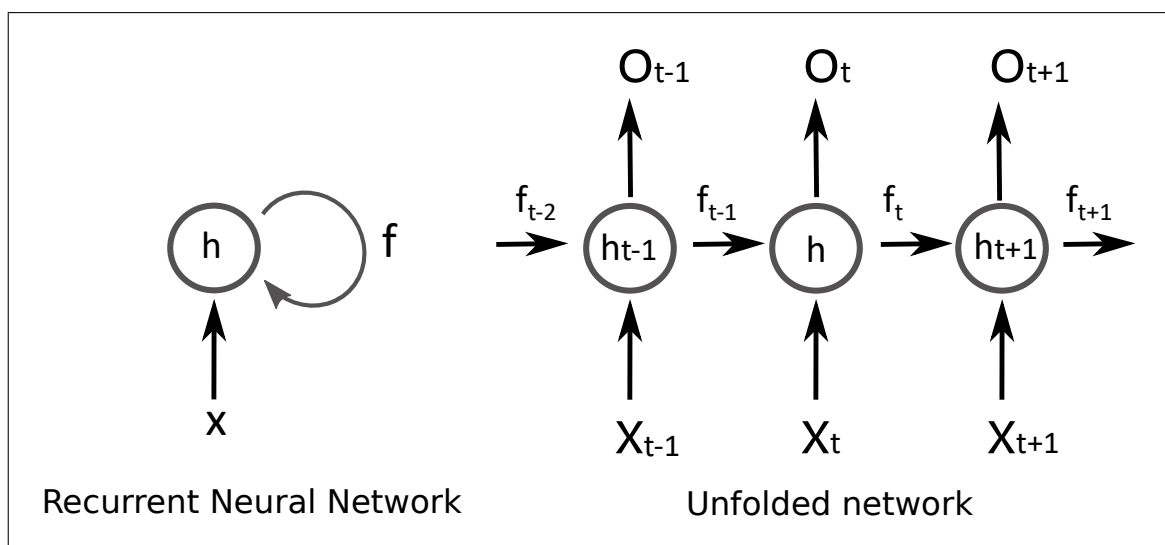


Fig. 3.11 Unfolding, over time, of an RNN architecture

Recurrent Neural Networks are considered as the better adapted architectures for sequence modelling. Their way of working is based on managing the inputs sequentially, as the internal construction of the RNN allows time-related feeding. In Figure 3.11 we illustrate a graph representation of an RNN neuron that shows a cycle around the neuron, and when we unfold the graph we can see the temporal sequence, where the input from previous steps feeds the next step.  $X_i$  are the original sequences and  $f_i$  the neuron output on each step.

The RNN neuron has the ability to use past information on each time step, and has two refinements the Long-short term memory (LSTM) and the Gated Recurrent Unit (GRU) defined below.

A relevant property for RNN is its ability to process sequences of undefined length, this property is vital for processing language sentences because, in a natural language phrase, all the words are relevant to understand the meaning, a sentence can be long, and as the algorithm processes the words one by one, we know that the initial words are important for



the overall meaning, and they need to *remembered* when we are processing the last words in the sequence.

Using an MLP the individual steps are processed independently, and in this way the different steps do not influence the others. An RNN does not have this limitation as the information flows temporally as we process the sequence of words one by one.

RNN are good at processing sentences, but have an issue with long sequences, because due to the vanishing gradient issues, the information from the beginning of the sequence has a small influence (as the value values are very close to zero) in the last sentences (see vanishing gradient issue in networks in 3.5.3)

The LSTM cells solve this issue. They were initially described by Hochreiter and Schmidhuber in [90], and they define an internal gate in the cell that allows the past information to flow or not into the sequence. The GRU cell, which is a simplification of the LSTM was proposed by Chung et al. in Chung et al., is a simpler version of the LSTM that works very well in some problems.

### 3.6.1 GRU and LSTM units for RNN architectures

When working with sequences, especially with natural language sequences, neural networks have the issue of short-term memory. When the sequence is long, the information from the beginning of the sequence is *forgotten* when the final steps are processed.

This problem is called the vanishing gradient issue that appears during the backpropagation phase, after many calculations the gradients decrease in a way that become infinitesimal. In this way when the gradients are close to zero, the network loses its learning capabilities.

To solve this issue Long-short memory units and Gated recurrent units can be used to build up RNN architectures. Both units can, as a main property, maintain in the network information from the parts of the sequence that are far away, allowing the model to 'remember' the first elements processed.

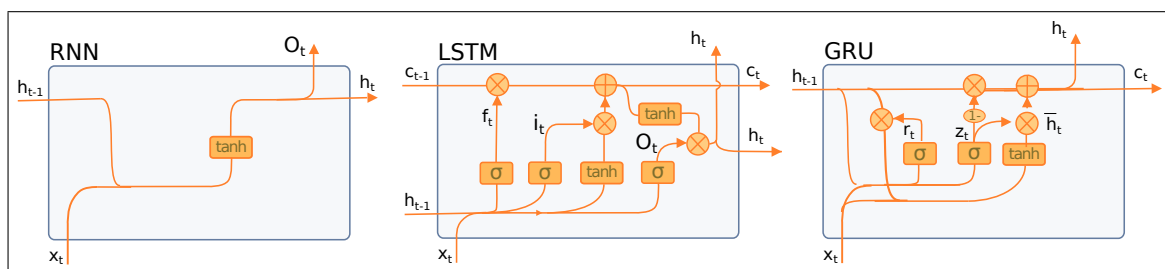


Fig. 3.12 Graphical representation of RNN units, LSTM, GRU

In Figure 3.12 we can see a graphical representation of the different individual cells, the vanilla RNN, the LSTM and the GRU cell.

GRU cells are simpler than LSTM as they lack an output gate. The LSTM has stronger learning capabilities, but for simple problems GRU can be easier to train showing equal or even better results.

The gates regulate the flow of information into and out, allowing the cell to remember information from past time intervals. It is unclear which of them is better adjusted to a problem, requiring experimentation on the specific application and data characteristics.

RNNs show outstanding behaviour with sequences of undetermined length because their dynamic nature adapts very well to the variable length of the sentences, but LSTM and GRU cells helps to control the information flow in long sequences. However, for time series of fixed length, their capabilities do not differ much from an MLP or a CNN architecture, as the experiments show in Chapter 8.

### 3.6.2 RNN with Attention mechanisms

In 2015 Bahdanau et al. in [9] introduced a new mechanism in neural networks, the attention. This new development is quite effective in applications like machine translation and is applied in new fields like computing vision, with success.

In this research, and as a cross-over between different areas, we have designed and implemented an attention layer for an RNN architecture to model wind time series.

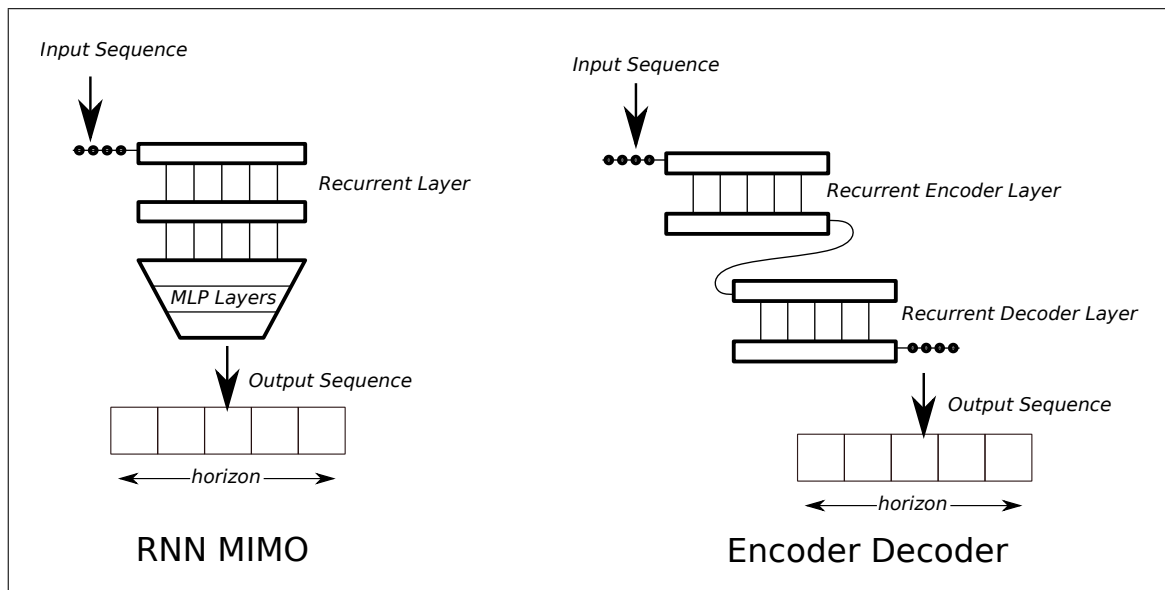


Fig. 3.13 Recurrent Neural Network MIMO strategy, and RNN Encoder-Decoder

An attention layer consists in a vector of trainable weights  $(w_1, w_2, \dots, w_H)$  applied to the input vector, these weights  $w_i$  are values in the interval  $[0, 1]$  and are calculated by gradient descent (or the chosen optimisation algorithm) simultaneously to the rest of the network training. The final values of the attention are obtained by applying a softmax

function, as the intention is to concentrate the values in a part of the sequence. We can see attention as a magnifying glass on one or a few of the time steps that are more relevant for the output.

### 3.6.3 RNN MIMO Architectures for time series

To work with wind time series sequences, we have designed two architectures based on the RNN elementary neuron, which are the RNN ED construction and the RNN MIMO. Both of them can process sequences and produce outputs of predetermined fixed length.

The RNN ED architecture processes sequences directly with the Encoder-Decoder architecture. This model inputs a sequence and produces a sequence as an output. This construction is widely used in Natural Language Processing as it is very effective with sequences of undefined length [204].

The second architecture is the RNN MIMO architecture, where there is a single RNN layer with a stacked MLP network (made of one or several layers). In this model the MLP layer performs the regression after the information has been processed by the RNN (see Figure 3.13).



## Chapter 4

# State of the Art of wind forecasting with deep learning

We analysed the fundamental principles of wind, the structure of time series, and Deep learning fundamentals in previous sections. In this chapter, we analyse the wind forecasting activity using deep and machine learning, and we present a literature analysis of this field.

As we have covered in previous chapters, there are two significant groups of forecasting methods used today in the renewable industry, Meteorological Numerical Weather prediction methods and time series methods. Time series methods, can be classified based on the approach used for the classification, which can be linear statistical methods and non-linear methods, with Machine Learning and Deep learning methods belonging to the second category.

Hybrid methods combine meteorological with time series methods, but these hybrid approaches are not analysed in depth in this chapter, as the focus is mainly in time series based methods.

This chapter has three main sections. First there is a high-level review of meteorological approaches to wind forecasting. Second, we present some relevant works on wind time series forecasting with linear statistical models, and then we analyse in more detail the machine and deep learning models.

The content of this chapter has generated a Journal Article:

- [1] J. Manero, J. Béjar, and U. Cortés. Wind energy forecasting with neural networks. a literature review. *Computación y Sistemas*, 22, 2018. ISSN 2007-9737. number 4

### 4.1 State of the art of Meteorological based forecasting

Meteorology is the primary tool for wind speed prediction. This science tells us that winds are the result of complex atmosphere interactions between global and local features, the local elements (terrain, geography, roughness) are complex to introduce in the actual resolution of the meteorological models used to forecasting wind. The challenge faced by the earth sciences is how to increase the resolution of the models to include these local events while coping with the increasing requirements of computing resources, to model complex weather features like wind.

#### 4.1.1 Meteorological models for wind prediction

Forecasting weather has been an elusive activity pursued by men since the human civilisation exists. Nevertheless, the origins of this discipline as a structured science are quite recent as they are found in Victorian Great Britain and the just-born United States of America, less than 200 years ago. The traffic of goods between the Atlantic was critical for commerce between the empire and the new colonies, but sea navigation, was still a dangerous event. In the North Atlantic there is complex weather system where storms come up unexpectedly. This dangerous sea sank ships by the thousands, that were unable to cope with the fierce strength of winds and waves, just between 1852 and 1855 over 1.000 ships sank due to storms, killing more than 900 men per year. To reduce this toll became a top priority for both countries a new science with new methods and new knowledge was required [130].

A few years before, some events marked the initial steps of this new discipline. In 1743 Benjamin Franklin set up the hypothesis that storms move from one place to another, a revolutionary theory that marks the start of the weather prediction science. From this initial hypothesis it took 100 years for Sir Robert FitzRoy to propose the development of a storm warning network in Britain, while at the other side of the Atlantic a Smithsonian director, Mr Joseph Henry, took the initiative to position a map in the main hall of the museum with information on the weather in different locations of the US, information gathered thanks to a new invention, the telegraph. With this simple and, powerful idea, he created something as extended today as the weather map. It was a sound success and immediately many newspapers copied this pioneering idea and created the popular weather section, making it an essential feature in all papers, from Boston to Philadelphia. Eleven years later, in 1861, The Times published its first structured weather forecast in their pages. The fascination and curiosity of men for the future weather was suddenly fulfilled with the weather prediction information, a feature that persists in the actual newspapers or in sophisticated real-time applications in our smartphones.

We can define weather forecasting as the determination of the atmosphere status in the next future, evolved from the actual situation. The way to solve this problem is by

## 4.1 State of the art of Meteorological based forecasting

---

developing a model that replicates the atmosphere behaviour, and then to synthetically evolution this model to the next state to identify the future conditions.

The actual situation of the atmosphere is a state of the model, and by solving some complex differential equations that simulate the atmosphere interactions, we calculate the next steps of the model situation. The equations depend on the grid resolution which determines the complexity of the system (we understand as the granularity of the prediction the terrain resolution of the model  $50 \times 50$  km,  $25 \times 25$  km,  $10 \times 10$  km) the smaller the grid, the more complex the modelling, and the complexity has a direct impact in the computational requirements to *solve* the model.

The grid size defines the model complexity and the data requirements to manage. The weather grid is three-dimensional (latitude, longitude and altitude) and this motivates the need for large amounts of data, and as a consequence resolving the equations requires vast computing resources. Supercomputing facilities support the large weather forecast models. As the available computing power increases, the resolution can be increased, being the actual state of the art, the use of grid areas of  $4 \times 4$  km for some regional models. The quest for  $2 \times 2$  km grid is underway by many national weather services as the limits of weather predictability have not been reached yet [247]

The NWP models usually have a three-phase processing cycle, namely, pre-processing, processing and post-processing. Pre-processing consists of the definition of a matrix that contains the real weather data for each point. If measurements are unknown, then interpolation is used to fill the gaps. Processing consists in the resolution of the differential equations (weather equations dependent on time) for each point in the matrix. Finally, post-processing is the phase to transform numerical data in the information to be processed, either the well-known meteorological maps or specific media or databases for particular points (wind, clouds, waves).

There are three groups of atmospheric models based on the time-space scale, Macro-scale, Meso-scale and Micro-scale. Some phenomena, like wind, require small grids as they happen at a low-resolution level, and change in short cycles. Other models do not require this granularity and might be able to give predictions with matrices with far fewer points and with more extended periods. Large government agencies are responsible for the maintenance of the global models, which are resolved several times every day, for instance; the National Centre for Environmental Prediction (NCEP) is responsible for the American model, and the European Centre for Weather Forecast (ECMWF) is responsible for the European one.

The global models work with geographical matrixes; for instance, the American Global Forecast System Model (GFS  $1^\circ$ ) has a matrix unit size of 100 km, with calculations every 6 hours. However, this model, at a square size of 100 km, is not granular enough for many kinds of forecasting, for this reason the regional models exist, which must function coordinately with the global model, they have a higher resolution, This regional models are called meso-scale.

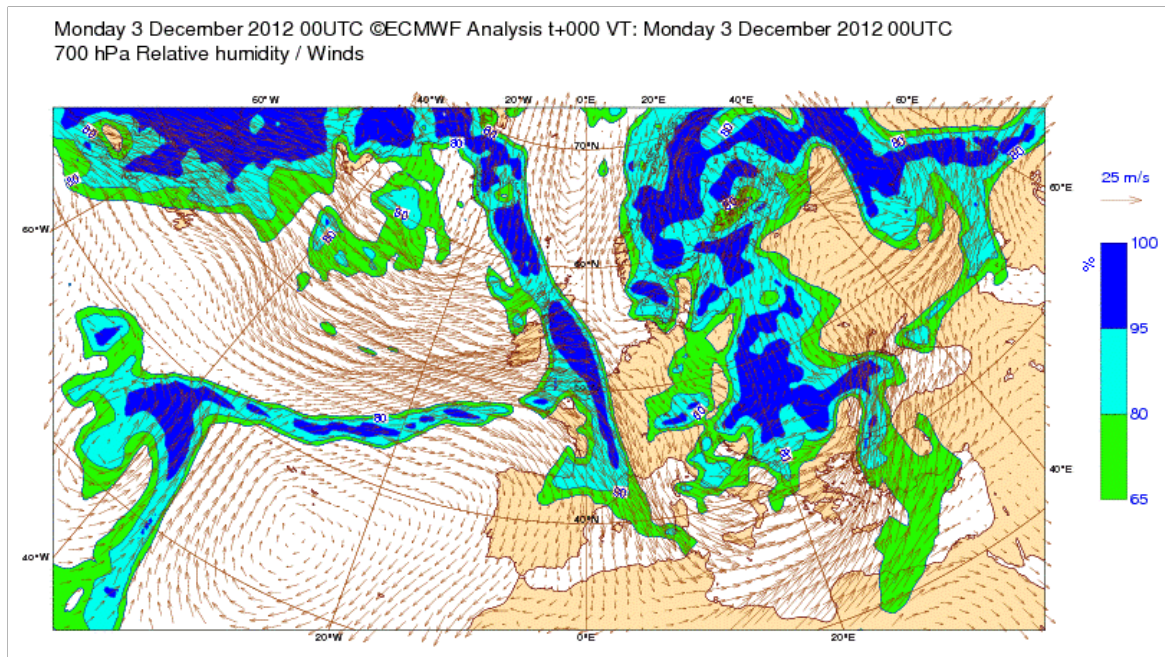


Fig. 4.1 European Weather map obtained from ECMWF modelling

A popular Meso-scale model for the US is the MM5. Created by Penn State University and the National Centre for Atmospheric Research is an open-source project used for testing, and an excellent tool to compare the reliability and error of new models [148].

In Europe, the most used models are the ones generated by the ECMWF. Those models are becoming quite sophisticated, generated coordinately with local government agencies like the *Asociación Española de Meteorología* (AEMET) which refines the models to be more adapted to the local variations of the geography [156].

As computing power has become more accessible, these systems have started to rely on ensemble models (see Section 4.1). An ensemble approach is a combination of different methods that integrated using statistical algorithms; in this way, the resulting combining scenario is more accurate than the individual original modelling.

The ECMWF weather prediction model runs 51 times with different starting points, and each model has some differences in the equations. All the scenarios combine into a forecast (ensemble) or alternative forecasts, and all the results can be used for the probability computation of future weather predictions [25].

Wind prediction has high complexity due to the local characteristics of wind creation. Wind comes from the interaction of pressure gradients that appear over vast distances, and the combination of changes with the local orography produce wind with a specific speed and pattern. The large numerical prediction models are not useful for wind energy prediction as the wind speed has to be determined with a very high resolution, and the global models cannot work at this resolution.



Models are improving due to the increase of available computing power with the widespread use of supercomputing facilities or massive cloud computing.

### 4.1.2 Numerical Weather Prediction methods

Numerical Weather Prediction (NWP) includes all prediction methods where the forecasting is made from a model based on equations.

These models have some issues, first, as they have a level of uncertainty they introduce this error in the prediction, and second, in general, the granularity is not optimal for wind speed prediction (as wind modelling using weather models require grids with very large resolution), this problem is approached by introducing down-sampling algorithms that are applied to the models to increase resolution. But down-sampling generates new errors that combine with previous ones. In an interesting article [185], [Sanchez](#) develops a model showing how the forecasting and the weather model down-sampling errors chain together.

The NWP methods are classified in Global and Regional (limited area) models. Global models have less resolution as their grid size is bigger than the ones used in the regional models. The grid resolution in the regional models (due to advances in resources and models) is increasing from the traditional  $12 \times 12$  km or  $7 \times 7$  km to the much smaller  $2 \times 2$  km or  $1 \times 1$  km grid sizes [73].

The meteorology science tries to find a way to reduce the error in the weather equations while increasing the resolution to a level that is useful for wind forecasting. To increase the resolution of a model is called downscaling. However, it could be that downscaling alone cannot increase resolution in a way that decreases the overall error for renewable prediction; for this reason, we need to develop workarounds or indirect methodologies. These indirect approaches use extensive knowledge of the local conditions (geography, terrain). We can cite, for instance, the approach designed by [Martin et al.](#) in [146] in the early days of wind prediction. They implemented, for a wind park in Algeciras, a method to forecast wind by using the pressure differences between measurements in two points using indirect independent measurements. The measurements came from the Jerez and Málaga airports, obtaining good forecasts when integrating both of them. This approach is an example of how using low-resolution observations to improve the weather model obtains good results with more accuracy. <sup>1</sup>.

The NWP models produce more substantial errors than pure time series approaches for short term predictions, and for this reason, for windows from milliseconds to 2-3 hours, methods based on time series are preferred.

The threshold for a forecast to be considered long term is at the 48 hours line, in this segment NWP work better than time series models. From 2-3 hours to 48 hours, predictions

---

<sup>1</sup>Algeciras is subjected to constant winds from the Mediterranean to the Atlantic and vice versa, Málaga is located in the Mediterranean coast while Jerez de la Frontera is by the Atlantic at roughly the same latitude, the difference of pressure between them predicts the wind direction and intensity.

are considered mid-term, and in this area time series prediction can or cannot be better than NWP, depending on the length (closer to 48 hours is NWP, while closer to two is time series), the terrain, the meteorology or the available data. The prediction science is trying to squash this interval by making the NWP methods to go into shorter time frames (by increasing resolution of the models, for instance) while time series models try to extend their area of comfort by using more sophisticated approaches or combinations of existing models. This thesis contributes to this objective, by designing models that allow the time series models to extend its validity to 12 hours ahead prediction.

This chapter contains a literary review analysis covering forecasting based on time series without covering the NWT approaches with the same detail. However, some of the techniques are common to both fields and are used indistinctly in either approach.

One approach that produces excellent results is the use of Ensembles (see Section 4.2.6). Ensembles are combinations of algorithms that are combined using statistical techniques and obtain better results than any of the individual models. This area of work has generated systems like the windstorms system for the north-west of Europe [175], that generates warning alerts for windstorms. This system is an ensemble that has several methods which combine competitively. Ensembles are relevant because they are used for commercial forecasting systems, as the objective is to get the best possible accuracy, with ensembles the combination improves the accuracy of the individual models.

Another example of work focused on NWP models is the use of Kalman Filtering methodology used for data assimilation. The use of this filter is a way to reduce the limitation of the NWP models to predict sub-grid phenomena accurately.

Downscaling the NWT model is the objective of researchers around the world, and the orthodox approach is by developing more detailed equations and using more computing power to resolve them. However, there are some alternative ways to improve resolution at a lower cost. One of them is Kalman Filtering which consists of a set of mathematical equations that provides an efficient computational solution of the least squares method with an easy adaptation to any alteration of the observations.

A Kalman filter can be used as a recursive filter applied to a linear dynamic system containing a list of noisy measurements over time. It is a widely used technique in several fields and specifically in wind prediction. Kalman filters use the joint probability distribution of the variables on each step and estimate future values of the variables in the future.

In Louka et al. in [132] they develop a practical implementation of a Kalman filter for wind prediction and suggest that instead of using costly computational resources to perform downscaling to smaller grids than 6 km, using adaptive techniques like Kalman filtering, we can obtain accurate predictions at wind farm level.

## 4.2 State of the art of Machine learning methods for wind time series

Time series based models are an alternative to the meteorological weather prediction. There are several relevant analyses of the state of the of wind prediction using time series as an input, like [160], [30] or [58]). In this section we present a selection of the most commonly used methods for wind prediction using statistical or machine learning methods, with some relevant works on each one them.

The list of method families is as follows.

- Linear regression methods
- Signal analysis applied to wind forecasting
- Bayesian models
- k-nearest neighbours ( $k$ -NN)
- Support Vector Machines (SVM)
- Fuzzy methods
- Combined approaches and Ensembles
- K-means for unsupervised analogy detection

### 4.2.1 Linear regression methods

Statistical methods look for data relationships inside the time series. Approaching the wind prediction as a linear regression problem is a common approach, which includes AR (auto-regressive moving average), ARMA (Auto-regressive moving average methods) and ARIMA (Auto-regressive integrated moving average methods) and there are many implementations for wind forecasting using any of these approaches, each one with its particular flavour. The first paper about wind power prediction using statistical regression algorithms appears as early as 1984 by [Brown et al.](#) in [24], that described an auto-regressive Process using a previous Gaussian distribution of the wind speeds. From there, the use of regression has spread all over the industry.

[Lujano-Rojas et al.](#) in [133] used an ARMA model in 5 sites in Navarra, Spain. The ARMA model outperformed the persistence model in one-hour forecasts improving the RMSE and MAE for longer periods up to ten-hour predictions.

[Milligan et al.](#) in [151] applied several ARMA models in forecasts up to 6 hours for wind farms in Minnesota and Iowa and concluded that the training was dependent on using data from a recent period.

[Erdem and Shi](#) in [48] created four approaches based on ARMA and VAR (variable autoregressive) approaches that show a good fit to forecast the wind speed and direction, their conclusions found out that the VAR models perform better than the ARMA in most tests.

Palomares-Salas et al. in [166] develop an ARIMA Algorithm and compare it to a Neural Network, data is sampled every 10 minutes (18,690 steps) from a weather station in Sevilla. Their findings show that the ARIMA and the NN algorithms obtain very similar results. The ARIMA algorithm obtains an RMSE of 1.07 for 1h, 1.31 for 2h and 1.55 for 4 hours.

### 4.2.2 Signal Analysis

Independent Component Analysis (ICA) is based on the assumption that time-series are a linear mixture of some factors such as seasonal components or trend, in this sense the irregularities come from these components hidden in the time series model.

$$x(t) = A \cdot s(t) \quad (4.1)$$

This formula is called the basic ICA model, where  $x(t)$  is the sum of an instantaneous mixture of source signals  $s(t)$ , and  $A$  is the mixing matrix. The goal of ICA is to find a linear transform of  $A$  by which the sources (ICS)  $s(t)$  are statistically independent. There is a variation from ICA which is the Principal Component Analysis PCA, which assumes a Gaussian distribution in the series.

PCA or ICA are usually used to transform the original data to be used in subsequent algorithms. The transformation is later reversed to obtain the final prediction.

Firat et al. in [57] present a statistical method based on independent component analysis (ICA) and autoregressive (AR) linear regression using two ICA algorithms, Fast(ICA) and a second-order blind identification (SOBI). FastICA is based on the maximisation of non-Gaussianity, and SOBI exploits the time structure of the data using second-order statistics. They compared the results with a pure AR model using a dataset from seven wind parks in the Netherlands (low RIX). Their application was inconclusive and left open the use of alternative methods combined with the ICA treatment. For three and six hours obtained a definite improvement with the SOBI algorithm (1.24 MSE for 3 hours and 1.48 MSE 8 hours).

### 4.2.3 Bayesian methods

There are some experiences applying Bayesian methods to wind prediction, based on the application of a probabilistic approach to classification and regression.

For instance, if we want to forecast the temperature in Seville in summer, we can consider a uniform distribution of temperatures between 0°C and 100°C, or we can estimate based on a distribution from historical series. We know that the probability of having a cold temperature in summer is almost zero, and the temperature has a much higher probability of being a value between 35°C and 45°C.

The Bayesian approach allows for the implementation of nested structures, in this sense, it allows for further characterisation of the time series data. For instance, in a wind park,

we could divide the turbines in a wind park in two categories, in one set the turbines in flat terrain, and in another the ones in slope, using this approach we add expert knowledge in the data, thus improving the capabilities to model the overall system.

[Miranda and Dunn](#) in [152] offer a complete methodology to build up a Bayesian model based on two years of data from a weather station in the Shetland Islands. The results showed that the Bayesian model performed slightly better than the persistence (relative RMS 17.3% persistence over 16.4% Bayesian).

[Ibargüengoytia et al.](#) in [95] present a Bayesian approach for wind speed. This approach uses a Dynamic Bayesian Network (DBN). The DBN learns from historical data. This method showed a marginally better behaviour compared to AR and ARIMA models and an RMSE of 9.84% for a 5h prediction.

[Li et al.](#) in [123] develop a Bayesian multi-model method, which is a Bayesian multi-model Averaging (BMA), this model performs a Bayesian integration of 3 Neural Networks. The NN methods include an Adaptive Linear Neuron or later Adaptive Linear Element (ADALINE) with back propagation and using a Radial Basis Function (RBF) plus an ARIMA model. In this paper they show how the best results for integrating the four models are always obtained with the Bayesian Model Averaging (BMA) algorithm, making this approach attractive when there is a need to integrate diverse methods.

Bayesian Model Averaging (BMA) is a technique used to combine different predictions in an ensemble. The BMA offers a framework that includes model uncertainty in the model selection phase. The BMA method weights individual predictions based on their posterior model probabilities, and then the better-performing predictions are assigned higher weights than the worse ones. The BMA method can thus generate an averaged model, especially in cases where more than one model has a non-negligible posterior probability [91]

The different Bayesian approaches, integrated with probabilistic forecasting 2.4.1 , offer a solid framework to build forecasts based on individual predictions that include uncertainty in the result and in the internal mechanisms of the model.

### 4.2.4 $k$ -NN: $k$ -Nearest Neighbours

The  $k$ -NN nearest neighbours approach has been used in wind forecasting, either by estimating temporal proximity of values in a single time series or by finding spatial similarities with geographically close wind sites.

Another method based on similarity calculation is the K-means unsupervised method that we discuss in Section 4.2.8, used extensively in site characterisation.

We can find many applications of  $k$ -NN in wind prediction, and we have selected some representative works.

[Yesilbudak et al.](#) in [242] use a  $k$ -NN algorithm to obtain wind speed using different characteristics like wind direction, atmospheric pressure or relative humidity. This approach,

tested with Manhattan and Minkowski distance calculations, outperforms the persistence model by 25.74% as the model predicts the next time step with series sampled in 10 minutes intervals.

[Heinermann](#) in [85] develops a complete comparison using MSE as error measure, between a Random Forest (RF), Support Vector Regression (SVR) and a  $k$ -NN algorithm. The objective is to predict the power output using wind speed as input for an ensemble method. In this approach, the  $k$ -NN algorithm shows better accuracy than the baseline methods. This research team from the university of Oldenburg (Germany), specialised in renewable energies, use a version of the NREL dataset, and has developed tools to access grouped wind sites from the big set of data.

### 4.2.5 Support Vector Machines (SVM)

SVM or Support Vector Regression methods use alternative dimensional spaces to simplify the modelling in the actual representation. The change of reference space is made by the use of a so-called *Kernel trick* which is a practical mathematical artefact defined in the initial paper by [Cortes and Vapnik](#) in [37] and later enhanced by [Vapnik](#) in his relevant book 'The Nature of Statistical Learning' [232]. The SVM family of methods were popular before Neural Networks became ubiquitous. SVM are a powerful tool for the representation of non-linear classification and function estimation problems. However, its issue is the amount of fine-tuning required, with a complex and lengthy convergence when training in some problems. Another conclusion has been that the size of the datasets is essential. For small training data sets, the error produced by this method can be substantial.

SVM apply linear classification techniques to non-linear classification problems. Being capable to model non-linear relation in an efficient way, it shows excellent performance with time series data with large datasets.

An interesting implementation is found in the work from [Heinermann](#) in [85]. We know that to define a SVM we need a regression algorithm plus a kernel function, in [Heinermann](#) approach the regression algorithm for the SVM is based on the found weights matrix  $w$ , obtained with the Vapnik proposed formulas.

$$\text{minimise } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \quad (4.2)$$

$C$  is a constant ( $C > 0$ ) chosen as a parameter that penalises only those errors greater than  $\xi_i$ . In this work [Heinermann](#) tests other kernel functions, using an RBF function to obtain better definition of the data boundaries.

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \quad (4.3)$$

The performance of an SVM algorithm is affected by the SVM configurations, mainly the kernel function characteristics. In [250] the comparison between 3 different kernel functions is made (see below discussion on this work).

Zeng and Qiao in [246] compare an SVM with an RBF neural network. Applying the algorithms on three years of data for 68 sites in the NREL dataset. He finds the optimal training length is 100 days and concludes that the SVM model is superior to the RBF one, and obtains for the SVM a MAPE of 1.0768% in 1h, 2.8778% in 2h and 5.36% in 3h.

Zhou et al. in [250] develop an LS-SVM (Large Scale-Support Vector Machine). The basic principle of an SVM regression is to estimate the output variable  $y$  from  $\mathbf{x}$  being  $\mathbf{x}$  a vector  $\mathbf{x} = (x_1, x_2, \dots, x_k)^T$  where  $K$  is the order of the SVM. The general model of the SVM regression is:

$$y = \mathbf{w}^T \varphi(\mathbf{x}) + b \quad (4.4)$$

where  $\mathbf{w}$  is the weight vector and  $b$  the bias term. Given a set of training data

$$\{(x_i, y_i)\}_{i=1}^N$$

the LS-SVM determines the optimal weight vector and bias term minimising the following cost function  $R$ .

$$\min_{\mathbf{w}, \mathbf{e}} R(\mathbf{w}, \mathbf{e}) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{2} \gamma \|\mathbf{e}\|^2 \quad (4.5)$$

This formula is a depart from the standard SVM model (that has inequality constraints with slack variables) the SVM-LS instead is subject to the equality constraints

$$y_i = \mathbf{w}^T \varphi(\mathbf{x}_i) + b + e_i, \quad i = 1, 2, \dots, N \quad (4.6)$$

where  $\mathbf{e} = (e_1, e_2, \dots, e_N)^T$  The LS-SVM simplifies the quadratic optimisation problem in the standard SVM which becomes linear in the LS-SVM.

Zhou et al. in [250] develop and analyse three kernel functions; Linear, Polynomial and Gaussian. Linear Kernel

$$k_L(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z} \quad (4.7)$$

Polynomial Kernel

$$k_p(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + c)^d \quad (4.8)$$

where  $c$  and  $d$  are the bias and degree of polynomial kernel respectively Gaussian Kernel

$$k_G(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2 / \sigma^2) \quad (4.9)$$

where  $\|\cdot\|$  is the 2-norm and  $\gamma$  is a constant determining the width of the Gaussian kernel.

This work finds that the optimal parameters of the SVM are related to the size of the dataset. The error measurement is classified by season as the patterns change seasonally, and this fact changes the accuracy of the models (steady wind vs variable winds for instance).

**Table 4.1**

Forecasting error (RMSE) of persistence and 3 kernels (Linear, Gaussian and Polynomial) single step forecasting one step ahead (1h) for wind speed. This table comes from the article [250], and errors are measured with RMSE

RMSE	Persistence	Linear	Gaussian	Polynomial
Spring	1.650	1.421	1.42	1.417
Summer	1.209	1.343	1.33	1.323
Fall	1.584	0.972	0.961	0.966
Winter	1.216	0.956	0.919	0.906

This work finds a strong relationship between the sample sizes and the SVM accuracy. For small sample sizes the SVM under-performs the traditional approaches, obtaining the best results with large sample sizes. where

In conclusion, the LS-SVM approach shows better results with temporal variability. Another conclusion from the exercise is that the training sample size impacts on the accuracy.

### 4.2.6 Ensemble methods

A machine learning ensemble consists of the combination of several learners to obtain a result with better accuracy than any of them [180]. Ensembles are a statistical artefact known for over hundred years based on the principle of "*Wisdom of the Crowds*", the tradition says that Sir Francis Galton, observing a crowd in a cattle fair, made a contest and showed that he was able to determine the weight of an ox by averaging the individual guesses from each person, in a more accurate way than any of the individual guesses [203].

There are four major classes of ensembles, Bagging, Boosting, Voting and Stacking. Bagging and Boosting use the same learner algorithm on the data and each one of them uses different distributions (boosting) or bootstraps samples from a fixed distribution over the data (bagging) [82].

The third class is a simple combination of different ML algorithms by a voting process.

The fourth class, Stacking (called also stacked generalisation) combines several machine learning algorithms using another ML algorithm, the first algorithms generate a set of data that feeds the combiner algorithm, and this one obtains the final result.

We can choose from several approaches in Voting. The simplest one is the Majority Vote method. In this method, we assume that the probability of each classifier being correct is  $(1 - \epsilon)$ , and we formulate the hypothesis that the classification errors are independent. The



method consists in choosing the label prediction by more than 1/2 of the classifiers.

$$\binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k} \quad (4.10)$$

The probability that a majority vote generates an error is obtained with the following equation:

$$\sum_{k=\frac{n}{2}}^n \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k} \quad (4.11)$$

If  $\epsilon < \frac{1}{2}$  and the predictions from the classifier can be considered as independent, the error is, in principle, smaller, and when  $n \rightarrow \infty$  then  $\epsilon \rightarrow 0$

We can find better accuracy over the direct or linear-averaged approach as this one gives the same weight to each one of the votes in a "democratic approach". In this way, if some inputs are stronger than others, the inputs can be quantified and adjusted, for instance in a Bayesian model averaging (BMA) where the weighting is adjusted after training by reviewing the individual contributions to the accuracy one by one. [Slougher et al. in \[198\]](#) develop this approach showing material improvements in accuracy by using a BMA-calibrating approach over the traditional linear-averaged model, this work is made on a 48-hour forecasting of maximum wind speeds in the Pacific. [Traiteur et al. in \[224\]](#) develop an ensemble approach for a wind speed forecast at 1-hour horizon with 21 individual forecasts with different configurations. This work shows how the use for NWP forecasts for short term predictions, usually done with statistical approaches, is valid and how the integration in an ensemble weighted algorithm generates good accuracy.

The use of ensemble algorithms allows the integration of several NWP models with different imperfections, or by developing a prediction algorithm with inputs from several models that work in parallel to compensate different adjustments to the specific conditions of winds and terrains.

NWP models are not used for short-term wind speed forecasting, as the time series approaches obtain better accuracy [73]. However, NWP models have several distinct advantages over statistical models. First, typical statistical time series models predict the future based on the past. This approach has an implicit assumption of stationarity that may not be applicable in a changing environment. For example, empirically identified relationships that govern wind speed are likely to change with changes in climate or land use-land cover. Second, many statistical models avoid the stationarity problem by using adaptive techniques where model parameters are updated frequently. This approach may also pose a problem if the memory of the system is short, for example, in a dynamic environment dominated by small-scale turbulence where winds are changing fast. Hence, numerical models with prognostic differential equations representing the temporal evolution of atmospheric dynamic

and thermodynamic variables are the best option for forecasting wind speeds in changing environments. A direct comparative study, albeit for more extended time scales, has found that appropriately calibrated NWP model ensembles provide better wind power density forecasts than statistical models alone [210]. Finally, another significant advantage of NWP models is that they can simultaneously provide wind speed forecasts and information on atmospheric turbulence with no added computational costs.

For Machine Learning approaches, bagging (see Section 4.3) can be used to develop ensembles. We can find an example in [86] where [Heinermann and Kramer](#) propose an algorithm using neighbouring turbines based on a previous work by [Kramer et al.](#) in [113]. Bagging or bootstrapping aggregation consists of building independent predictors extracting different samples from the training set and averaging the output obtained by the prediction algorithms. To construct an ensemble, the predictors must be independent and loosely correlated, to obtain the best results [180].

We can find other ensemble approaches applied to prediction in the electricity sector, like these two works that analyse deep learning for energy demand forecasting applications [116, 81].

The use of ensembles is a valuable area of research as the ability to run multiple algorithms in parallel is efficient, and the combination of models with different strengths generates improved results [73].

### 4.2.7 Gradient Boosting based ensembles

Gradient boosting is an ensemble algorithm defined by [Friedman](#) in his work [62] where he published the original gradient boosting approach (see Algorithm 4.1)

This algorithm consists of designing a cost or learning function based on the error and minimise its value by applying successive alternative learners. In this case, the learning function becomes a kind of Loss function  $L$  that is optimised.

Gradient boosting is extensively used in combination with decision trees but is applied successfully to any kind of weak learners, including neural networks [17]. In this thesis we apply successfully this approach using convolutional networks as learners.

The gradient boosting approach is generating a lot of interest since several refinements have been recently published, using the original idea of the Adaboost optimisation [60], possibly the most effective being the XGBoost approach [31]. These new methods offer an improvement in the selection of the learner, which results in better accuracy of the result. The XGBoost approach is very effective in classification problems with structured data, being one of the most used algorithm in Kaggle competitions [174].

---

### Algorithm 4.1: Friedman's Gradient Boosting Algorithm

---

**Data:** input data  $(x, y)_{i=1}^N$   
 $M$  = number of iterations  
 $\psi(y, f) \leftarrow$  Loss Function  
 choice of the base learner model  $h(x, \theta)$   
**begin**  
     initialise  $\hat{f}_0$   
     **for**  $t = 1$  to  $M$  **do**  
         compute negative gradient  $g_t(x)$   
         fit a new base learner function  $h(x, \theta_t)$   
         find the best gradient descent step-size  $\rho_t$ :  
          $\arg \min \sum_{i=1}^N \psi \left[ (y_i, \hat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t)) \right]$   
         Update function estimate:  
          $\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho h(x_i, \theta_t)$   
     **end**  
**end**  
**Result:**  $\hat{f}_t$  with minimum error

---

#### 4.2.8 K-means as unsupervised Learning for wind time series

K-means is an unsupervised method that comes from the signal analysis discipline. The objective is to partition a set of  $n$  observations into  $k$  clusters, where each observation is assigned to the cluster with the nearest mean. The mean of a cluster is calculated using a central or centroid calculation.

This widely used algorithm has three major issues, the first one sits in the determination of the number of clusters, that needs to be predetermined and heuristically found, the second has to do with getting sometimes trapped into local minimums, not obtaining the best results, and the third is that the initialisation state can impact the result.

For wind time series, k-means is used to find analogies between different examples of the time series to use them combined with other methods as in [72], or as in [6] where the k-means output is used to feed a neural network then applies a transformation of the wind data using wavelet functions to improve the overall accuracy of the approach.

## 4.3 State of the art of deep learning applied to wind prediction

There are not many practical applications of deep learning to wind energy forecasting applications. The fact that the last decade has seen many developments in this science discipline explains this limited adoption, that is slowly catching up as the first commercial DL implementation are appearing.

For this thesis we performed a state of the art review, where we recollected different applications of deep learning for wind forecasting [141], this work is partially included in this section. We summarise the results in Table 4.2 that illustrates this literature review in a tabular form.

We have grouped the different contributions in three groups based on the deep learning technique used. The first group contains all the experiences using multi-layer perceptrons (MLP), followed by convolutional (CNN) and recurrent networks (RNN). In some cases, different approaches are combined, a situation that we specify in the analysis.

### 4.3.1 Multi-Layer Perceptrons

This section analyses works that develop neural network models using basic and hybrid MLP architectures. Hybrid models in this section mean a combination of several models, that can be neural networks or a set of different architectures with linear and non-linear approaches (where usually the MLP covers the non-linearity side of the model).

Liu and Zhang in [127] explore several ML architectures ( $k$ -NN, REP-tree, SVM, MLP and RBF networks) in 7 datasets, which integrate observations with meteorological data from NWP Models. It uses seven features, temperature, dew point, relative humidity, wind direction, wind speed, station pressure, and wind power, then creates an additional measure for the cube of wind speed. The architectures are tested with several hidden layers (up to 4) with 300 neurons, but increasing the number of layers does not improve the results of the experiment. The conclusions show that the best model is SVM with somehow promising results from the MLP (but with lower RMSE consistently); however, the MLP architectures show better behaviour with longer time scale predictions.

Tao et al. in [209] develop a deep belief network DBN architecture with 3 layers of 100, 200 and 300 nodes. Data from a wind station in Mongolia is used sampled every 10 minutes. They perform several experiments training in three months of data to generate forecasts from 6 to 24h ahead. The results accuracy, measured with MSE and MAE show stable results from 6 to 24h, which demonstrates that the models have potential to capture some of the hidden patterns in the wind series.

Kani and Riahy in [102] develop an MLP architecture integrated with a Markov Chain probabilistic engine to establish forecasts in very short-term (seconds). This short term forecast has the objective to identify turbulences and wind changes for the turbine control. The new model improves the persistence results.

Ranganayaki and Deepa in [173] describe an ensemble architecture of MLPs that obtain accurate results. It integrates several data elements like temperature, wind direction, wind speed and relative humidity. The MLP architectures tested are an MLP, with two variants, one with standard perceptrons and the other with adaline neurons and a Probabilistic Network. The models are tested with a two year dataset with observations from a real wind

farm in India. The research develops a criterion to fix the number of hidden neurons and obtains a sensible improvement from other methods measured in MSE.

[Sapronova et al.](#) in [187] present a DL approach that outperforms linear extrapolation for short-term wind speed predictions (up to 30 min). The DL architecture is not specified in detail, and one of the conclusions of the experiment is that using NWP data does not improve the accuracy for wind speed short term forecasting (at 10 and 30 minutes ahead), in this short article the authors express their belief that shorter time windows in the series can obtain better accuracy, a fact that has been tested in this research work in (see Chapter 8 and Chapter 9).

[Shi et al.](#) in [191] develop a hybrid approach that consist in using a two step model combining an ARIMA model that works on the linear components of the series plus an MLP or an SVM that focuses in the non-linear component. The conclusion is that a hybrid methodology is a viable option, but it is a complex approach with little improvement over the isolated methods.

[Liu et al.](#) in [126] using data sampled every half an hour from a Chinese wind farm in Qinghai develop several hybrid models, an ARIMA linear model, a wavelet (signal decomposition) and an MLP. For the MLP try several training approaches. They conclude that the hybrid algorithms have better performance than the isolated ARIMA or Persistence, and the best training algorithm is the BFGS Quasi-Newton Back Propagation. However, the improvements calculated in terms of MAE, MSE and MAPE are not spectacular. In a similar approach, [Khandelwal et al.](#) in [106] apply a wavelet transformation on the time series to decompose the linear and non-linear components of the data, to apply ARIMA methods to the linear dataset and an MLP to the non-linear. With this approach obtains better results than with the single standard approach.

[Li and Shi](#) in [122] compare several MLP architectures using data observations in North Dakota in the US. He evaluates the results in MAE, RMSE and MAPE. They conclude that there is not a superior architecture as the results depend on the data. They obtain improvements up to 20% with better tuning of the models. The authors propose a post-processing methodology to apply to the forecast results to decrease the model differences. It is interesting to note here that the importance of tuning with the DL models is a conclusion on this thesis which is described in detail in Section 6.4.

Other approaches integrate Solar and Wind data, like [Hossain et al.](#) in [92], where they develop an MLP architecture for Hybrid forecasting (wind and solar). The model includes eleven meteorological observations, like wind speed, wind direction, solar radiation, relative humidity, rainfall, wind speed, wind direction, maximum peak wind gust, evaporation and average barometer measure. The output is a three hour ahead forecasting. The data is from the Australian town of Rockhampton as the observations come from a tower in this town. This work shows the importance of integrating exogenous variables in the prediction that improve the learning quality of the network.

### 4.3.2 Convolutional Neural Networks

Convolutional neural networks outperform other methods in image and pattern recognition, where they obtain outstanding results. The overspecialisation in images has hampered its application to other areas, however, its excellent pattern recognition capabilities are applied to other problems with success.

Díaz et al. in [42] use three years of NWP wind data (8 parameters) from a model sampled every 3 hours and compares the results to real production data from one site (Sotavento Wind Park in Galicia, Spain) and for the whole Country wind energy production (Spain). Three deep learning architectures are tested and compared with a Gaussian SVR model and a Neural Network with just one hidden layer. In the experiments they tested an MLP architecture with two hidden layers of 250 and 300 neurons, a standard CNN with the first layer with 2x6 filters and two fully connected layers of 200 and 400 neurons, the last architecture is a LeNet-5 network with two initial convolutional layers and two fully connected 200 unit layers. Results are measured with MAE, and the values improve around 5% from the SVR algorithm. The forecasts horizon is not specified, the conclusions are promising about the architectures, but some concerns about computational cost and improvement of the parameter setting in future works are made in the document.

Wang et al. in [233] propose a CNN approach that beats a shallow MLP, the persistence method, and a simple regression algorithm. The data comes from a wind park in Sangchuan Island (China), with a length of one year. The time series is decomposed in different frequencies, and each one of them has its own CNN architecture. Results are post-processed into a time-series forecast, beating the other methods by 10%, in the shortest term, and by 100% in a 4-hour time frame. An exciting conclusion is the finding of a remarkable seasonal (winter, summer, spring, autumn) difference between the error results (up to 6x difference).

### 4.3.3 Recurrent Neural Networks

Recurrent Neural Networks are the best candidate for a sequence to sequence learning, as their internal memory gates obtain outstanding results with natural language processing and other applications, they have limited testing with wind time series, but some of the works have encouraging results.

Ghaderi et al. in [71] develop an LSTM and an RNN architecture using spatial information (data from neighbours), they use data from 57 meteorological stations obtained from the Airport Meteorological control in the East coast of the US. With this data, they Develop RNN and LSTM architectures, obtaining excellent results for short-term forecasts. One satisfying conclusion is the excellent performance of the DNN architectures on the site located in Nantucket (this site has stable wind regimes as it is by the sea). The DL methods beat any other method and accomplish to obtain a good forecast based on the observations from the 57 meteorological sites.

### 4.3 State of the art of deep learning applied to wind prediction

---

Cao et al. in [26] use data from a meteorological tower in the Texas university that generates a time series with a 15-minute sampling of wind speed data at five different altitudes. Develops an RNN architecture and compares it with two ARIMA algorithms. The experiments are measured in MAPE, MAE and MSPE. From the experiments two significant findings are obtained, one is that using wind speed measured at different heights improves the ARIMA models sensibly up to 40% (in MAE), second the much better performance of the RNN architecture, over 100% improvement from the ARIMA algorithms, showing that the RNN network acquires the internal patterns of wind, integrating the covariate information of the different heights.

Liu et al. in [128] develop a methodology to forecast the power generated by a wind power plant (wind park composed of several turbines). The procedure is based on a two-step methodology with two NN architectures, first a probabilistic neural network screens the data and identifies which of the turbines are the best representatives of the plant, this representative data feeds an RNN network in a second step obtaining the total power of the plant. The errors from this approach are calculated from 10 minutes ahead to 60 min ahead horizon and range between 7.8% to 9.58% RMSE.

Olaofe in [161] develop an RNN architecture for one hour ahead of wind power prediction, and the test data come from real weather observations in the Slangkop wind site (South Africa). Using sampled data at 1s, mean data at 1h is generated in a dataset composed by five elements (the speed at 50m, gust, pressure, temperature and humidity), this data feed an RNN with two layers. The relevant point is that the training is fitted using the power of the turbine, as it is adjusted to obtain the minimum MSE between the theoretical power the power curve of the turbine and the results from the algorithm, this generates training based on the power output. The results for one-hour prediction ahead (power) are 0.156 RMSE or 0.009 MAE.

Balluff et al. in [12] develop a RNN architecture for long term (24h) prediction. Based on an exercise performed with NWP model data for off-shore points concludes that this architecture has potential but requires a high degree of fine-tuning. It does not develop error comparison but observes good learning potential in the RNN architecture.

Khodayar et al. in [107] test an NN with stacked architecture on a subset of the NREL dataset. The architecture combines an RNN approach with a Stacking of encoding and decoding layers. The results of this construct improve a standard ANN by more than 20% in short term prediction up to 3 hours.

## State of the Art of wind forecasting with deep learning

**Table 4.2** Review summary of methods (MLP:Multilayer Perceptron, CNN:Convolutional Network, RNN: Recurrent Neural Network)

Type	Author	Data	Architectures	Results	Comments
MLP	Liu and Zhang [127]	7 farms with real + meteo data	DNN, SVM, ANN	Best, MAE 6h = 12	Rolling structure of algorithms
MLP	Tao et al. [209]	wind turbine Mongolia 10 minutes	DBF 3 layers 100/200/300 neurons	Stable results 6-24 hours ahead	Better performance for mid-term forecast
MLP	Kani and Riahy [102]	Several sets wind speed 2.5s	2 layers ANN integrated with Markov	15% improvement MAPE with MC	Prob. approach for very short term prediction
MLP	Hossain et al. [92]	Rockhampton Solar and wind data	ANN with 11 variables	non-qualified results	Integration Solar/Wind, use of exogenous vars
MLP	Ranganayaki and Deepa [173]	Two year data observations from 2 wind park sits (India)	ANN ensemble (4 variants)	2-10x improv. over previous exp. in MSE for short term	Methodology for the calculation of hidden nodes
MLP	Sapronova et al. [187]	NA 2.5s	ANN , DL architecture	20/25% improv. over ANN (MAE or RMSE)	Very short term prediction, architecture not specified in detail
MLP	Shi et al. [191]	NREL North Dakota 1 to 7 steps	ANN ARIMA SVM hybrid	Only 3% improvement hybrid over single method	Hybrid does not always generate better performance
MLP	Liu et al. [126]	25 days data Wind Farm Qinghai China	ANN Wavelet ARIMA hybrid	Wavelet + ANN (BFGS) best model	Hybrid is marginally better but more costly
MLP	Li and Shi [122]	North Dakota sites, 1 year hourly sampled	3 ANN architectures	Best model depends on data	There is not a <i>best</i> model
CNN	Dfáz et al. [42]	Meteo Data 1 farm and Areas in Spain	CNN and NN	MAE 5% than SVR algorithm	Exp. algorithms with promising results
CNN	Wang et al. [233]	One year data from 2 wind farms in China	CNN DL Architecture	20% up to 600% improvement in some time frames	Decomposition of time series in signals of different frequency
RNN	Ghaderi et al. [71]	57 locations meteo data	RNN and LSTM architectures	RNN best results	Arch. obtain good results in one site from the others, learning geo-spatial correlation
RNN	Liu et al. [26]	Meteo Texas U. 5 heights 15 min	RNN and arima	RNN better than arima	Covariate usage of wind at 5 heights
RNN	Liu et al. [128]	250 Turbine Wind Farm in Colorado (US)	10min to 60 min 7.8% to 9.58% RMSE	Probabilistic NN feeds RNN	Power results with RNN from selected representatives
RNN	Olaofe [161]	Weather obs. Slangkop and power data	2 RNN architectures (Power)	RMSE 0.156% 1h ahead	Train RNN on Power expected from power curve, with good results
RNN	Balluff et al. [12]	NWP data from offshore sites	RNN	Improvement but not measured	Concludes RNN as the right architecture for wind prediction
RNN	Khodayar et al.[107]	NREL data from points in Idaho, US	RNN and ANN architecture with encoding/ decoding layers	20% RMSE improvement on 3 hours from standard RNN	RNN recommended approach with stacking, using rough set theory on the neurons



# Part III

## Experimentation, Methodology and Results

---

*"You don't understand anything until you learn it more than one way"*

2005 - Marvin Minsky [87]

*"You can't refute a statistic with an anecdote "*

2019 - Pedro Domingos [168]



## Chapter 5

# Wind Data

The wind energy generation industry is a data-rich environment. Wind parks generate and store multiple observations in sequences of data with frequencies that can be as short as seconds or milliseconds. Unfortunately, the wind industry considers this information as commercially and strategically valuable, and for this reason, they lock it up. This issue is a roadblock for the wind research community, as this lack of access to open data blocks the interest for new research [115].

We require large amounts of wind data to apply deep learning algorithms, but the scarcity of available datasets invalidates some works that are based on a small number of wind sites. A generalisation of results from a few sites is not possible as the experimentation in this thesis demonstrates.

For this research, we have obtained access to the most significant public wind dataset available in the world, the NREL (National Renewable Laboratory United States) Wind Toolkit. This complete set of data is the input for all the forecasting exercises in this thesis.

We structure this chapter into three sections. In the first section we describe the Sotavento Park, an open data experimental wind park in Spain that is used in some preliminary experiments. The second section analyses the NREL dataset, which is the data backbone of this Thesis. Finally, the third section looks into some new datasets with potential for future works.

## 5.1 The Sotavento wind park dataset

The Sotavento wind park dataset [131] is a set of observational data generated from sensor measures from the Sotavento wind park. Sotavento is an experimental wind park project funded by the local government of the Galician autonomous region in Spain, formed by 25 turbines from different manufacturers producing 38,500 MWh per year with a total nominal power of 16,56 MW.

Their dataset has data available since its first production date, with time series of over ten years long, which are sampled at 5 minutes intervals containing three measures, wind speed, wind direction and total energy generated. The information comes aggregated from the wind park as a whole (there is not information from an individual turbine).

This dataset is widely present in the literature, with many research works using these data, like [73, 222, 247, 248, 43, 133].

This long list of research articles and conferences shows the value of this park as a scientific resource, however, it lacks weather observations linked to the data (there is a weather station close by, but the data belongs to the meteorological Spanish service and has some access restrictions).

**Table 5.1** Wind Turbines in Sotavento wind park, Galicia Spain

Turbine Type	Power(kW)	Rotor $\phi$ (m)	Tower height(m)	Units
Neg Micon NM-48	750	48	45	4
Gamesa G-47	660	47	45	4
Made AE-46	660	46	45	4
Izar-Bonus MK-IV	600	44	40	4
Ecotecnia 44/640	640	44	46	1
Neg Micon NM-52	900	48	46	1
Made AE-52	800	52	50	1
Made AE-61	1,320	61	60	1
Izar-Bonus 1.3	1,300	62	49	1

In summary, the limited size and the lack of dimensions (variables) of this data do not make it valuable for deep learning experimentation as this kind of algorithms require a more substantial amount of data.

## 5.2 The NREL Wind dataset

The National Renewable Energy Laboratory publishes a wind resource dataset [46] with information from sites in the U.S. This large dataset offers production and meteorological data (wind speed, wind direction, temperature, humidity and pressure) generated synthetically

from meteorological global models, and later cross-verified with real observed data. The dataset contains 126,692 sites evenly distributed across the North America geography. All this information is sampled in 5 minutes intervals, in a seven years long sequence (from 1<sup>st</sup> January 2007 to 31<sup>st</sup> December 2013)

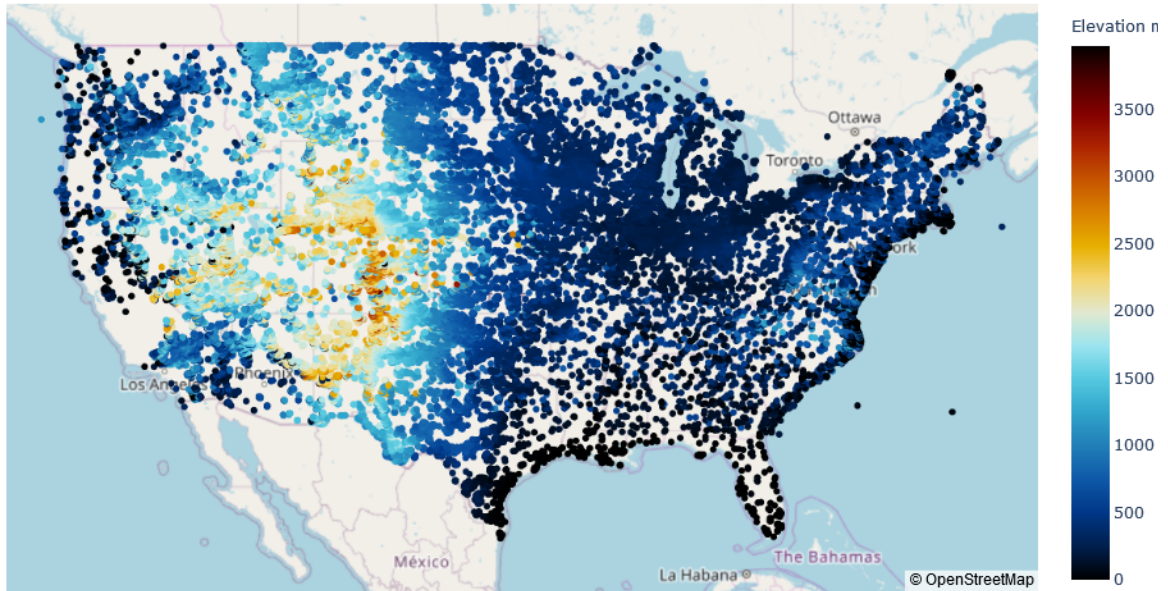


Fig. 5.1 Wind sites in the NREL dataset showing the elevation in meters of each point

This data is probably the largest publicly available wind dataset available today, and it has been the core dataset used in this work. The amount of data requires the availability of large amounts of computing resources which have been provided by the Barcelona Supercomputing Center (BSC) <sup>1</sup>

Table 5.2 Dimensions on the NREL Wind dataset

Measure	Description
Time (UTC)	Time of the recording
Wind speed (m/s)	Wind speed at hub level (100m)
Temperature (K)	Temperature in Kelvin
Wind direction (°)	Wind direction at hub level (100m)
Pressure (Pa)	Barometric pressure at surface level
Air density (kg/m <sup>3</sup> )	Air density at hub level (100m)
Wind Power (MW)	Wind Power every 5 min <sup>2</sup>

<sup>1</sup>The Barcelona Supercomputing Center (BSC-CNS) is the national supercomputing centre in Spain. Specialised in high-performance computing (HPC) manages MareNostrum, one of the most powerful supercomputers in Europe.

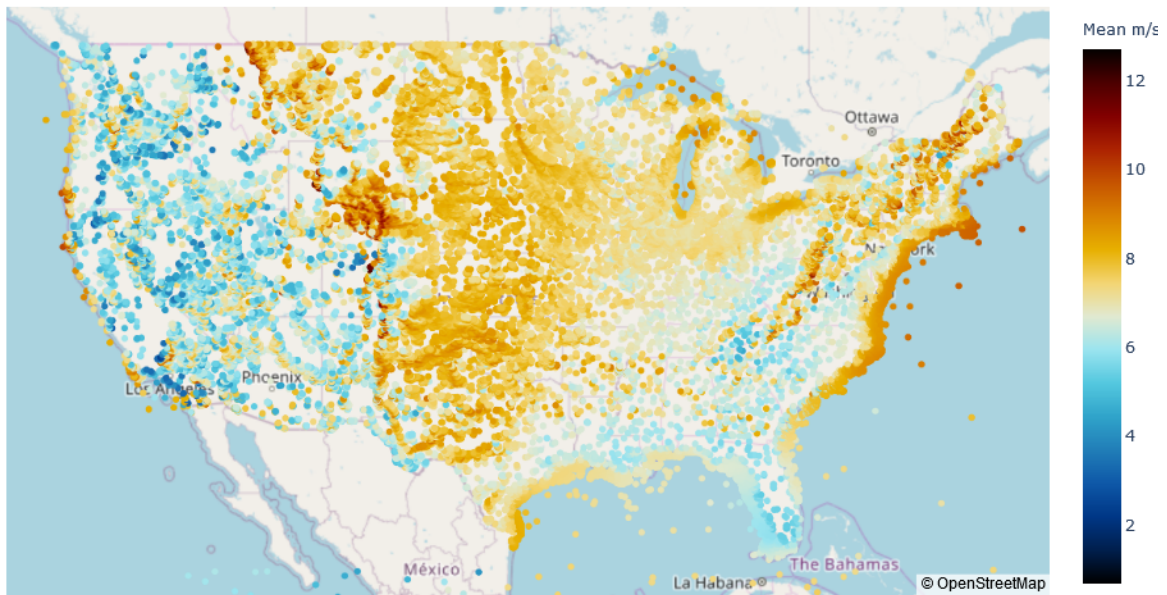


Fig. 5.2 Wind speed mean (over total length of series) in the NREL dataset

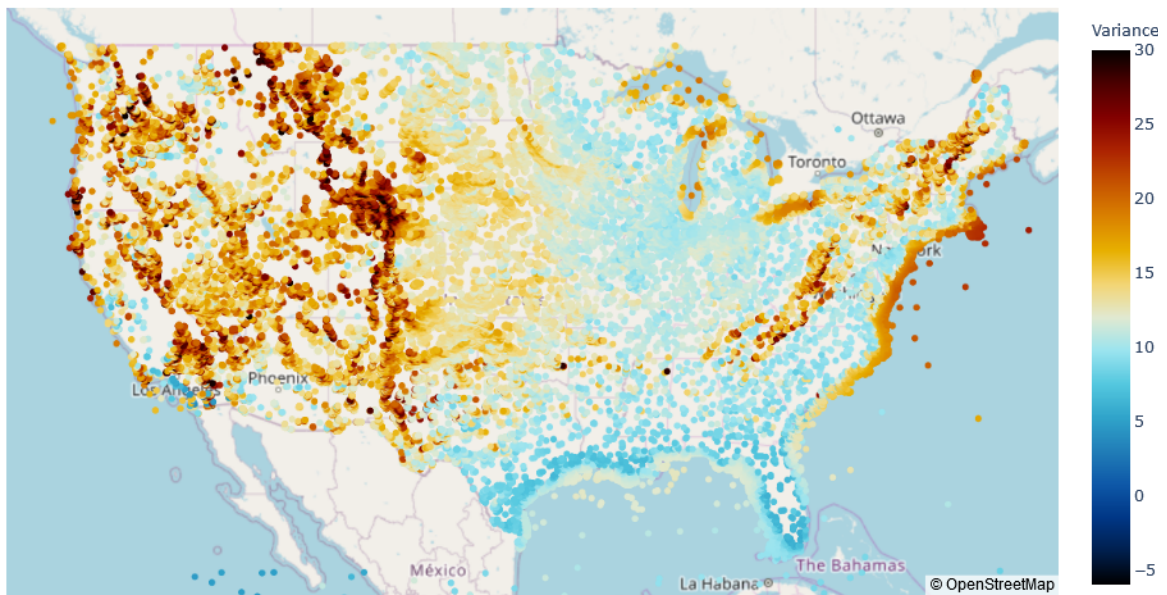


Fig. 5.3 Wind speed variance (over total length of series) in the NREL dataset

This dataset is synthesised from meteorological data, obtained from the Weather Research and Forecasting Model (WRF) version 3.4.1. described in [197]. The NREL dataset has 2km as a minimum distance between sites, complemented with terrain, roughness and soil

---

<sup>2</sup>Wind speeds have been converted to power applying normalised power curves described in [46].

properties obtained from the U.S. Geological Survey GTOPO30 data. The data is verified with real observations to eliminate anomalies and increase the quality of the result. The dataset contents are described in Section 5.2.

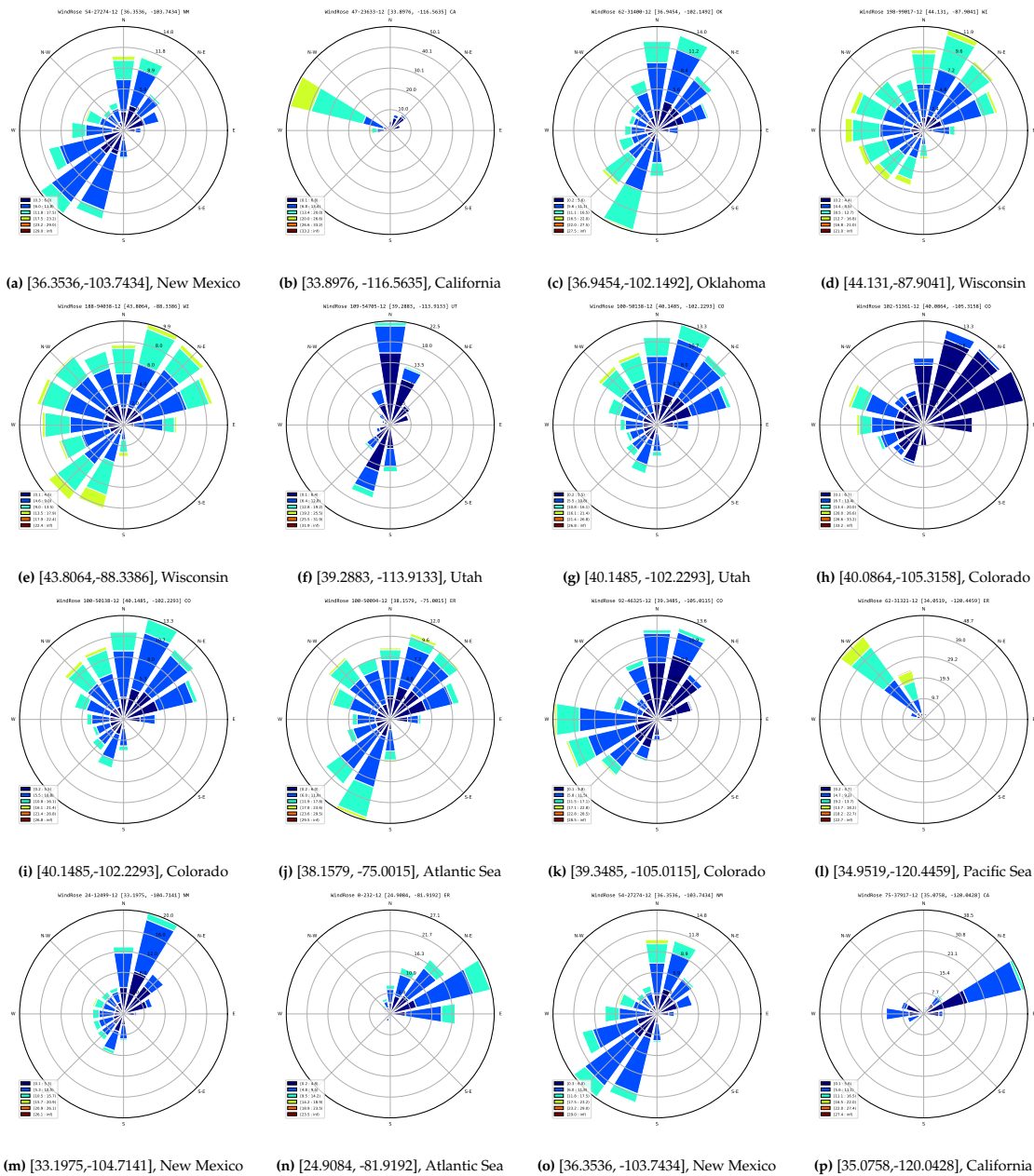
A preliminary statistical analysis of the dataset allows us to observe some characteristics of the wind regimes in North America (see Figures 5.2 and 5.3). In this map, we can see that the most variable sites (high  $\sigma$ ) are in the planes just at the east of the Rocky mountains range, in some westernmost states (like Oregon, Idaho, Nevada) and in the North Atlantic Coast. It is relevant to note the high heterogeneity of the sites. We can see in Figure 5.4 how different are the dominant wind directions in sixteen sites chosen randomly, and in Figure 5.5 we can see how similar the directions are for sites located in the same geographical area (an area of around 20 km in the illustration).

This dataset is virtually the data source of choice for large deep learning research on wind prediction, and there are some works recently published that use it. We can cite, between others, some works developed using the NREL dataset: [54, 53, 149, 86, 85].

We illustrate some basic statistical descriptive maps in Figures 5.2 and 5.3, Then in Figures 5.4 and 5.5 we can see how 16 sites randomly selected offer a variable picture of wind direction and intensity, while 16 sites chosen in the same area (see Figure 5.5) show very similar patterns, this two illustrations help to understand why it is important, when working with wind data, to obtain many very different sites, otherwise the results have a high risk to be biased.

From several experiments we build some geo-spatial representations relating those representations with method accuracy and other properties. We specifically devote Chapter 10 to the analysis of the relationship of a physical location with its forecastability.

# Wind Data



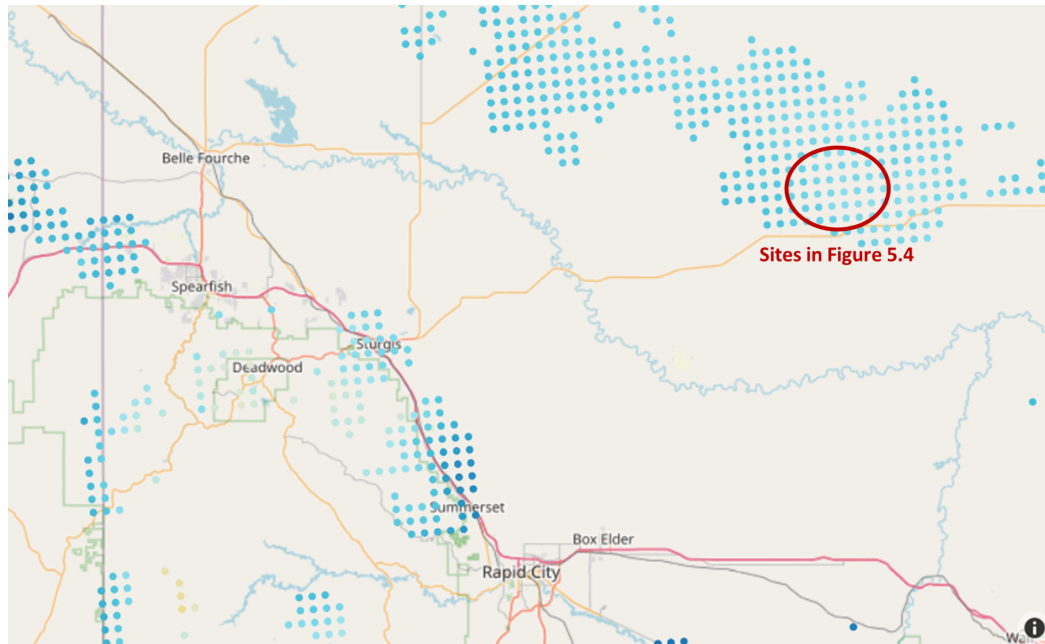
**Fig. 5.4** Wind-roses from several sites chosen randomly, we can see how the dominant winds direction differ in 16 random sites



## 5.2 The NREL Wind dataset



**Fig. 5.5** Wind-roses from several sites located in a square of  $10 \times 10$  km near Rapid City in South Dakota. All sites have the same dominant winds. See Figure 5.6 for an illustration of the sites location in a Map



**Fig. 5.6** Location of a group of turbines in South Dakota illustrated in Figure 5.5

### 5.3 Other datasets

There are other datasets that offer wind and power information from a wind park or from some turbines. We can cite the Dataset from the independent TSO in Canada, that offers power information from a wind farm in Ontario, a dataset that has been used in works like [192]. This dataset, even if limited, is a useful tool as it comes from real observations [97].

The generation of a massive new dataset is a costly project, but there is interest in developing new resources, like the INDECIS project [96], that is developing an open data wind dataset with information from tall towers around the world.

This project is grouping and cleansing data from many sources, that will be available for studies in the next future.

In summary, we can conclude that a common platform for results comparison between different research projects will push innovation and new ideas in this field. This dataset that does not exist yet and must include wind data sites from as many places as possible from real observations, and with very long series. A resource of this kind will accelerate the application of deep learning to wind forecasting.

## Chapter 6

# Experimental Framework

To verify the applicability of the wind prediction deep learning models, we must develop an unbiased experimentation framework to run the experimentations. In Chapter 5 we analysed the data, and in this chapter, we describe the experimentation details of the process, regarding programs, executions, architectures and how the experiments have been designed and executed.

The research objective is to develop new deep learning architectures to forecast wind speed. The experimentation consists of designing architectures and then testing and validating them with the NREL dataset. We compare the results using standardised accuracy measures.

This chapter has four sections. The first is about the general design principles, followed by a section describing how the data is prepared and pre-processed for the algorithms ingestion. The third section describes the general experiments set up with a description of how to interpret the outputs (results distributions). The last section describes in detail the hyperparameter search used to optimise the parameters in the different architectures.

All the experimental work (as seen in Chapters 7, 8, 9 and 10) has been performed using this common experimental setup.

### 6.1 Experiment design settings

The architectures have been developed using Python 3.6, with add-on packages, being the most relevant:

- Machine Learning platform: Tensorflow 1.14.0
- Deep learning library: Keras 2.2.4
- Scikit-learn machine learning library: 0.21.4
- Statistical support: statsmodels 0.11

The experiments have been supported by a NVIDIA GPU based computer at the Super Computing Center [147]. The resource is the Minotauro cluster which is build with BULL and it has 39 bullx R421-E4 servers, each server with:

- 2 Intel Xeon E5-2630 v3 (Haswell) 8-core processors, (each core at 2.4 GHz, and with 20 MB L3 cache)
- 2 K80 NVIDIA GPU Cards
- 128 GB of Main memory, distributed in 8 DIMMs of 16 GB – DDR4 @ 2133 MHz - ECC SDRAM –
- 1 PCIe 3.0 x8 8GT/s, Mellanox ConnectX®-3FDR 56 Gbit
- 4 Gigabit Ethernet ports.

The full machine provides a Peak Performance of 250.94 TERAFLUPS distributed as 226.98 TERAFLUPS (K80) + 23.96 TERAFLUPS (Haswell)

The available computing power has allowed to perform the experimentation to understand the modelling dynamics of each deep learning architecture, to identify the best architecture and its best features for the problem. From this statement, we have defined a list of design principles that guide all the work:

1. The experimental data is the NREL Dataset
2. The experiments have to show comparable results, thus the sampling approach to the sites must be consistent.
3. All the experiments must be reproducible
4. The deep learning architectures must be optimised through an iterative process until we obtain the best possible value
5. The comparison of results will be made using data from all sites

The experimentation consists of three significant steps or phases, Experiment preparation, Experiment execution and Experiment results analysis. In the following sections, we develop each one of them.

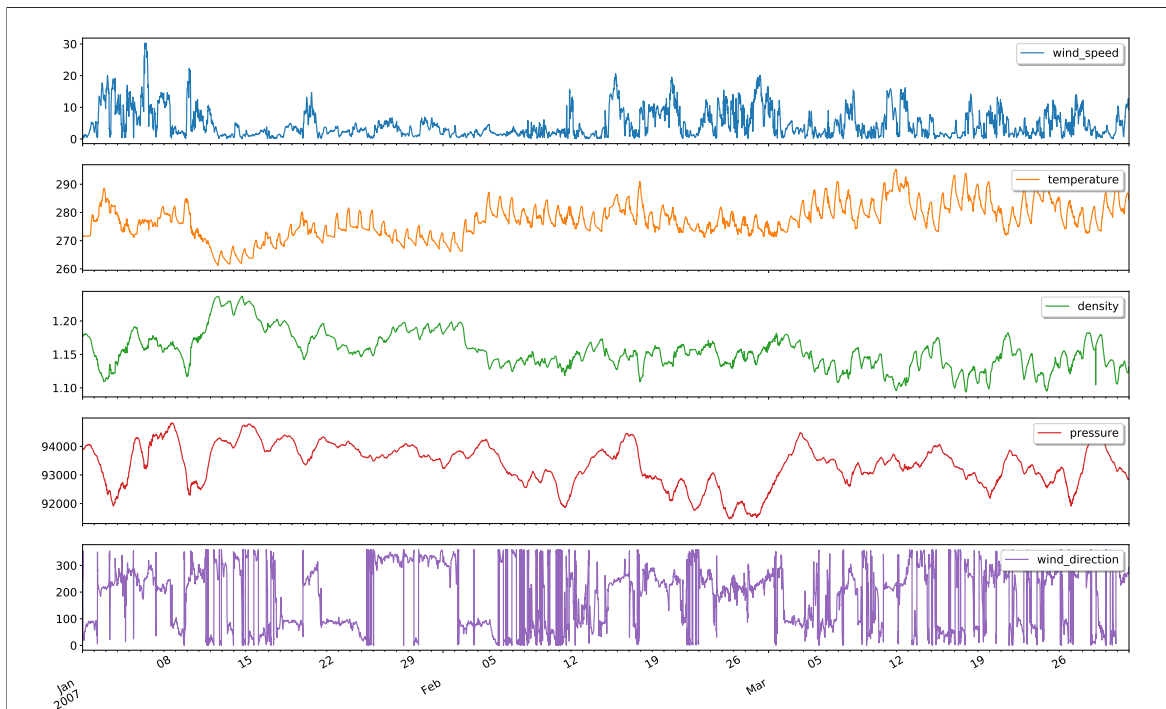
## 6.2 Data preparation

The NREL Dataset is a set of 126,692 files that contain the time series data for each one of the sites. These files have the extension `.nc`, an extension used for files encrypted with the Linux `mcrpyt` format. They use the `md5` algorithm, which creates a hash signature to assure the consistency of contents on each file through transmission or storage operations. The data format in the files is NetCDF, a format defined by the University Corporation for Atmospheric Research (UCAR) which is an international standard of the Open Geo-spatial Consortium [155].

This format is a secure format for sharing and storing the data, but computationally expensive to read and manage, for this reason, we decided to transform the original data format into a common `.npy` file, transforming the data to time series represented by `numpy` arrays (with dimensions = steps, variables).

### 6.2.1 Time series pre-processing

The original time series has the structure described in Table 5.2, a multivariate series with five dimensions for each time step. (Wind speed, temperature, wind direction, pressure and air density), we can see an illustration with the series from a single site (see Figure 6.1).



**Fig. 6.1** NREL wind time series plot, decomposed by its variables (see Table 5.2) for site located in Horseheaven, Oregon [44.733, -120.462], the bottom measure is direction (measured in  $360^\circ$ ) which has an unusual shape due to continuity issues when wind changes from  $360^\circ$  to  $0^\circ$

## Experimental Framework

---

The first operation is the transformation of the wind direction. We transform this variable from degrees to the sines and cosines of the angle. This formula is called Beers transformation, firstly mentioned in [13], and generates two sub-series one for the sines and another for the cosines of the angle. With this new structure, the information of wind direction is preserved, and the data is ready to train the architectures. With this transformation, the multivariate series becomes a six variable, time series, with wind speed, temperature, density, pressure, cosine wind direction and sine wind direction.

The next step is to normalise the data, a process that increases the stability of the neural network training. We have tested several methods of data normalisation, like feature scaling (re-scaling data between 0 and 1) or standardisation (using z-score or t-scores). Based on preliminary testing and literature recommendations, we chose the z-standardisation approach. Data z-standardisation consists of adjusting the values to a mean zero and variance  $\sigma^2$  of one. This data transformation generates better results and optimises resources by making the training more stable [188, 79].

We decided to average the series in one-hour intervals. Originally the series have been sampled at 300 seconds (5 minutes) intervals, but as the prediction is for 12 hours ahead, we can optimise the training time by re-sampling the series to one-hour intervals. We believe that reducing the amount of data by a factor of 12 will help to optimise the training times of the experiments. However, we will perform some experiments with the original series, using data with five-minute steps, to verify the accuracy differences with experiments with different step length (see Sections 8.4 and 9.6 for experimentation using 5 minutes series).

From the original series we created two different sets of data, one with the series with their initial step of 5 minutes, and the other with data aggregated (averaged) hourly, where the time steps frequency is 1 hour.

### 6.3 Experiment Setup

The platform for the experimental setup requires to cover several objectives. Firstly, it needs to define a common playground for all the different models. Secondly, it needs to assure replicability of the process with traceability of the different steps, and thirdly it must help the efficiency of the process by sharing the common parts of the code. We illustrate the overall structure in Figure 6.2, where we can see a graphical representation of the different blocks of the experimental process.

The definition for each experiment is based on a configuration `.json` file (see Figure 6.3) that defines the architecture and the set of hyperparameters. The definition of architecture contains three major areas, **Data** which specifies the data characteristics with a definition of the input length of the training examples and the steps to predict, **Arch**, that defines the parameters of the architecture, and **Training**, which specifies some training parameters. In this last category, we include the way on how each model processes the input data (shape of

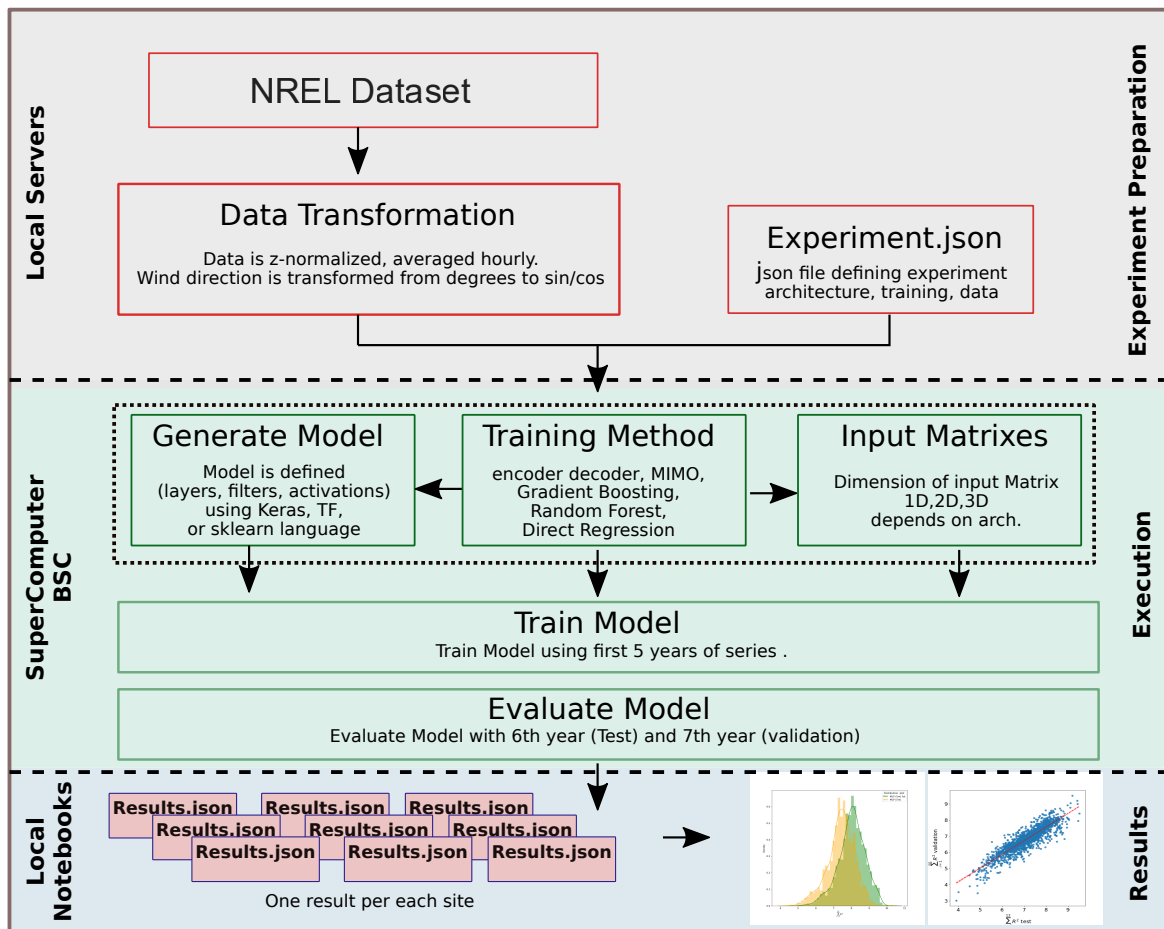


Fig. 6.2 Experiment framework

the input sequences) and the specific characteristics of its training process ("mode"). In this way, each experiment has a corresponding .json file that defines all the characteristics of an experiment.

The data characteristics define, variables used as multivariate input, the *lag* length of the examples for training and the number of steps to predict  $H$  horizon.

We define the architecture in the segment "arch", which contains the different parameters of the specific model. The architectures are defined to be interpreted either by Keras, Tensorflow or Scikit frameworks.

The Keras framework is a library of functions developed on top of TensorFlow, based on python classes that allow defining the network architectures using some predefined building blocks. Keras acts as a simplified interface to tensorflow speeding the development.

The architecture defines the model behaviour with parameters like the number of layers, the number of neurons, activation functions, drops or skip connections. Each group of architectures requires a different set of parameters, depending on its inner structure and requirements.

## Experimental Framework

---

Finally, the training segment defines some training characteristics like the optimisation algorithm, learning rate, batch size or epochs number. This training segment is common in most of the experiments. The ("mode") parameter defines the training mode for this architecture, as there can be several training strategies like:

- Sequence to Sequence training: used for MIMO architectures
- Direct Regression: trains direct regression architectures
- Joint Multiple Regression: Generates several regressions slicing the output sequence in several objectives to regress.
- Recursive: Obtains output recursively
- Gradient Boosting: Obtains regression result by applying a Gradient Boosting algorithm
- Scikit training, MIMO, Direct to be used with Scikit routines (Random forest,  $k$ -NN)
- persistence: persistence and other simple methods

```
"data": {
  "datanames": ["11-5794-12"],
  "scaler": "standard",
  "vars": "all",
  "datasize": 43834,
  "testsize": 17534,
  "dataset": "onesitemanyvar",
  "lag": 18,
  "ahead": [1,12]
},
"arch": {
  "filters": [1024],
  "strides": [2],
  "kernel_size": [9],
  "depth_multiplier": 8,
  "activation": ["elu",0.3],
  "drop": 0.6,
  "filters2": [1024],
  "strides2": [4],
  "kernel_size2": [1],
  "depth_multiplier2": 7,
  "activation2": ["elu",0.3],
  "drop2": 0.6,
  "dilation": false,
  "k_reg": "None",
  "k_regw": 0.1,
  "rec_reg": "None",
  "rec_regw": 0.1,
  "activation_full": ["elu",0.4],
  "fulldrop": 0.10,
  "full": [1024],
  "mode": "CNN_sep_2l_s2s"
},
"training": {
  "iter": 1,
  "batch": 1024,
  "epochs": 200,
  "patience": 10,
  "optimizer": "adamax",
  "lr": 0.001
}
```

**Fig. 6.3** Experiment parameters for a convolutional separable model with 2 layers using as an input series averaged at 1h and training *lag* of 18 steps

The configuration file determines the behaviour of the model. The set of values defines a multi-dimensional parameter space that is analysed with the hyperparameter optimisation technique to obtain the best set of results.



A prediction can be formalised in the following way, let  $\mathbf{x}_i$  be the multivariate series for step  $i$  and  $lag$  the total length of the input samples, and  $H$  the horizon of the multi-step prediction.

$$\underbrace{\langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{lag} \rangle}_{\text{multivariate input series}} \longrightarrow \underbrace{(\hat{y}_1, \hat{y}_1, \dots, \hat{y}_H)}_{\text{single variable multi-step prediction}} \quad (6.1)$$

The accuracy is calculated by comparing the values predicted with the real values for the Horizon steps. The algorithm generates a `.json` file with resulting error measures, which are:  $R^2$ , MSE, MAE, RMSE, nRMSE and nMAEval. For instance, for the coefficient of determination  $R^2$  is calculated as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^H (y_i - \hat{y}_i)^2}{\sum_{i=1}^H (y_i - \bar{y})^2} \quad (6.2)$$

Where  $y_i$  are the real values and  $\hat{y}_i$  the predicted ones.

## 6.4 Hyper-parameter setting strategy

As the deep learning architectures have many parameters, their combination generates many possibilities that define the behaviour of the algorithm and its performance. From the beginning of the experimentation in wind time series, we observed that the differences for a single experiment, just by small changes in parameters, was noticeable. In some architectures, the performance between an optimised set of parameters and a random selection can be as high as 20%, a fact that points to the need for a structured parameter optimisation approach.

The simplest method for parameter optimisation considered is the naive method. This method consists of combining the parameters by intuition or trial and error, this is an obvious method but can only work for simple architectures. However, as the complexity increases, this modelling can be frustrating as finding the right measures is time-consuming, and in some cases, almost impossible as the number of combinations is too high. For this reason, we decided to use a more sophisticated automated and structured approach.

To define an approach, we need first to analyse our problem. For the experimental work at hand, the objective is to obtain a single parameter optimisation of an algorithm considering all the wind sites in the dataset. We search for a unique optimisation of parameters valid for all data points. The quality or cost function, in the hyperparameter optimisation, must consider that the requirement is a global optimisation and not an optimisation for a single site.

The algorithm parameter optimisation, for this problem, can be formalised as follows: Given an algorithm  $A$  with a set of parameters  $P$ , and a set (or distribution) of sites  $s$  and a

## Experimental Framework

---

quality metric  $Q$ . In this environment, the optimisation must find the  $P$  settings on  $A$  that minimise the  $Q$  on  $s$ .

We can approach this problem with model-free algorithms that search using a pre-defined strategy on the set  $P$ , this family of optimisation algorithms has a fixed model before the execution starts and does not change. An example of this approach is the widely used grid-search that combines all the parameters and performs a systematic search. Another approach is a random search that seeks random combinations and executes the algorithm  $A$  for a number of iterations, selecting the best result [36].

---

### Algorithm 6.1: Hyperparameter optimization algorithm

---

**Data:**

Set of Parameters  $P(\phi, o)$  Algorithm  $A$

Selected Sites:  $s$

**begin**

**while** *accuracy keeps improving* **do**

        Phase 1: Generate Random experiments in Selected Sites 1.1: Intensify,  
                creating more experiments from best configurations

        Phase 2: Generate Score prediction using Random Forest strategy and use it to  
                create more combinations with *approach*

**case** *approach* **do**

            (random): Randomly from parameter surface

            (best score): From best score prediction randomly

            (cross-over): Doing cross-over of best score predictions

**end**

        Phase 3: Create (equalise) most promising parameters with more sites

**end**

**end**

**Result:** Best Set of Parameters  $P(\phi, o)$

---

Model-based methods have the property to use the results from a previous execution of the algorithm  $A$  to select next parameter  $P$  combinations, in this sense the algorithm changes its parameter selection based on the results of the executions, following a modelled strategy. These methods have the ability, when the set  $P$  is vast, to find right combinations of parameters in a more efficient way than using model-free strategies.

In the model-based methods, we can find a subfamily, the sequential model-based optimisation (SMBO) that is quite effective. It consists of a formalisation of Bayesian optimisation. Bayesian optimisation tries to generate the next set of parameters based on a surrogate probability method that is easier to optimise than the quality function. A Bayesian logic approach can be quite similar to the kind of reasoning that we do intuitively trying to solve a problem, as our brain is quite good at simplifying problems to be able to solve them [19].

The SMBO performs tests sequentially, and each iteration tries to find a better set  $P$  using Bayesian reasoning using a surrogate or probability function. This surrogate function can be challenging to define.

There is another approach to the sequential model-based algorithm configuration (SMAC) that uses a Random-Forest algorithm to determine the next configuration. This approach is practical and works with categorical values, a relevant feature for deep learning parameter optimisation [94],

For this specific problem, we have used an adaptation of a SMAC approach, which is a novel combination of several steps that have proven to be effective in revealing good parameter combinations for wind prediction algorithms optimisation.

The approach consists of three phases. The first phase creates several random combinations of parameters (subjected to some constraints), then in a second phase the most promising combinations are exploited to generate new sets of combinations, are created from the best initial combinations. The second phase uses one of three different techniques (random forest, further combinations or cross over) that is chosen by the operator. The final phase consists in increasing the number of sites for the better-ranked, in order to have a comparable number of sites with similar parameter combinations (using this strategy we can obtain good parameter combinations with a small number of sites that under-perform when tested with a broader set of sites.

This approach is not fully automated but it is efficient to obtain a good set of parameters in the algorithm. From the experience on the wind series we know that the surface of solutions has many local maximum, and we can obtain many combinations of parameters that obtain results that differ only by one or two hundredths.

**Table 6.1**

Best Experiments from optimisation example. These are the six best parameter combinations, the differences between the best ones are small

n	Activation	full	drop	kernel size	lag	Test mean	count	Val mean
1	['elu', 0.3]	[128]	0.0	[1]	6	7.19171	25	6.81979
2	['elu', 0.3]	[128]	0.2	[1]	6	7.19895	50	6.87788
3	['leaky', 0.1]	[64]	0.2	[9]	12	7.18365	50	6.81057
4	['elu', 0.4]	[256]	0.3	[1]	6	7.19677	25	6.81856
5	['elu', 0.3]	[128]	0.2	[9]	12	7.15832	50	6.87765
6	['leaky', 0.2]	[256]	0.3	[9]	12	7.19851	25	6.85464

In Table 6.1 we present an example from an optimisation exercise, the six best combinations are promising, but they need to be tested on a larger number of sites, as 25-50 are not significant yet. Equalising by increasing the number of sites with the parameter combinations will obtain a new batch with a better assessment of the most promising combination.

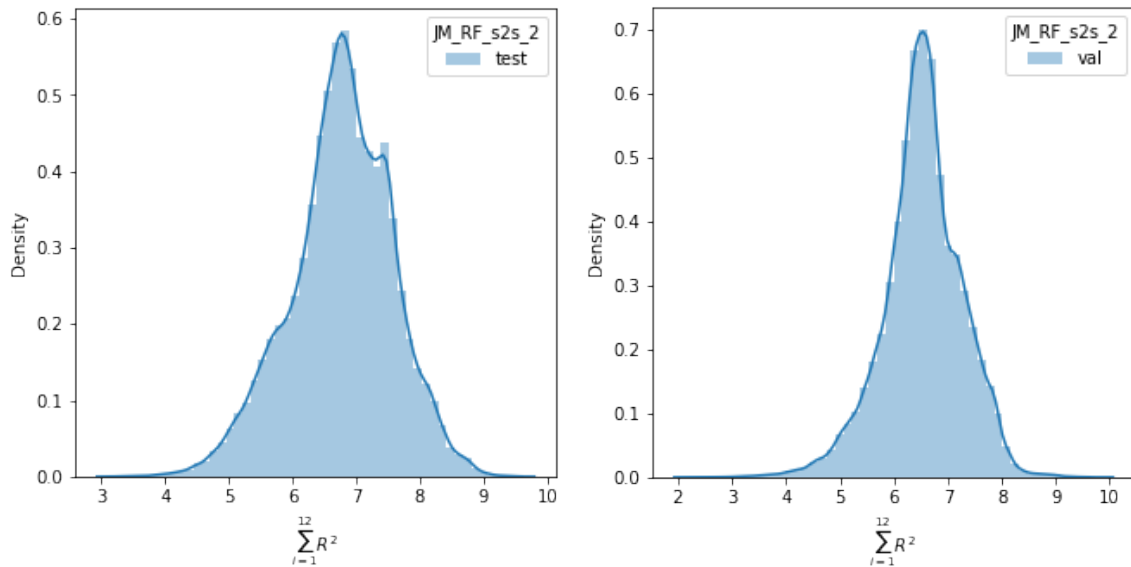
## Experimental Framework

Not all the parameters are optimised, as some parameters are manually set, like the optimiser, where the choice is an adaptive gradient descent search method (Adamax) [108]. All executions have limited the number of training epochs to 200 with an early stopping strategy that ends the optimisation when the accuracy does not improve for ten epochs. For a detailed description of each architecture component, see Tables (7.6, 7.9, 8.1).

### 6.5 Interpreting the results of an experiment

For each experiment, the model trains with five years of data (Training set). This model is optimised using hyperparameter optimisation techniques (see Section 6.4). We calculate the model accuracy with the validation and test datasets, each one of one year long. Each model obtains an accuracy value for each site adding the  $H$  time steps (for instance for the coefficient of determination  $R^2$  is  $\sum_1^H R^2_i$ ), and for the whole experiment is the average value of all sites used in the experiment.

When the experiment has been performed over a set of sites, each site result can be considered an error probability distribution (see Figure 6.4).



**Fig. 6.4** Probability density function for test and validation data in a Random Forest model experiment on the 126,692 NREL sites

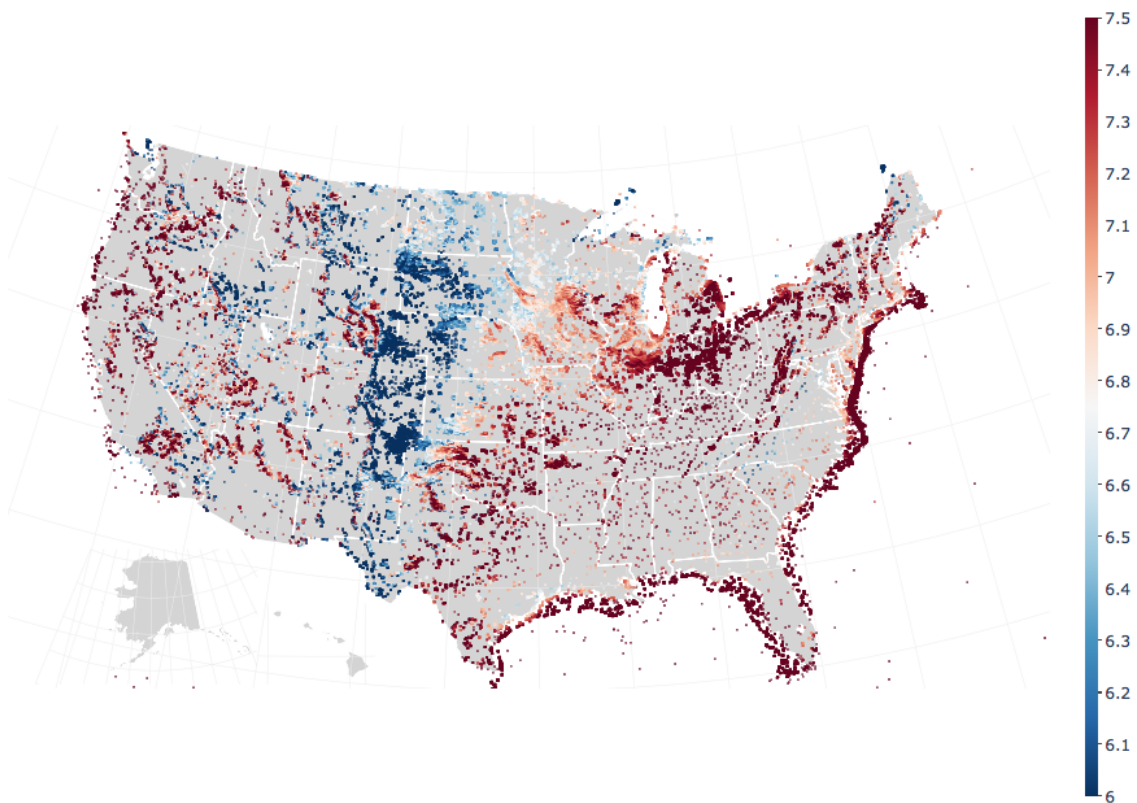
We have considered  $R^2$  better suited for comparison between comparisons, and is widely used in most experiments and representations.

As a summary, we can conclude that a site is characterised by a value or a score, while an experiment generates a distribution of values (as many as sites used for the experiment). To compare the results of two experiments, we need to compare their distributions, and for this comparison, we need to use some statistical tools.

## 6.5 Interpreting the results of an experiment

- ANOVA test: To determine if the differences of means in a group of experiments are statistically significant
- Tukey's honestly significant difference test: Anova tells if there is a significant difference, and the Tukey's test determines the difference
- Kolmogorov Smirnov test: to determine if two experiments generate statistically different distributions

We developed some graphical representations to see the results of the experiments on a map, a representation illustrating how the accuracy of a method corresponds with the geography. The maps show the correlation between accuracy and wind resource intensity areas (see Chapter 10).



**Fig. 6.5** This map shows the accuracy of MLP architecture over 126,692 wind sites on a map of the U.S. The colours indicate the error measures as the sum of  $R^2$ , the best sites are in the Atlantic and Pacific Coast where the lowest accuracy sites are in the plains



## Chapter 7

# Basic Deep learning architectures for wind time series forecasting

This chapter describes the first wave of experimentation performed with DL architectures to predict wind speed. For the experimentation, we use the data described in chapter 5, and pre-processed to optimise the use of neural networks (see chapter 6 for details). The architectures used are deep learning models that fall into three major families, Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

This chapter is structured as follows. First, we describe the baseline methods developed for comparison with the deep learning algorithms. We discuss how the best regression strategy has been identified, and finally, a detailed analysis of the main methods developed with a focus on its characteristics and performance.

This chapter has led to publications in several congresses and journals:

- [1] J. Manero, J. Béjar, and U. Cortés. "Dust in the wind...", deep learning application to wind energy time series forecasting. *Energies*, 12(12):2385, 2019. doi: 10.3390/en12122385
- [2] J. Manero, J. Béjar, and U. Cortés. Deep learning is blowing in the wind. deep models applied to wind prediction at turbine level. *Journal of Physics: Conference Series*, 1222:012037, May 2019. doi: 10.1088/1742-6596/1222/1/012037
- [3] J. Manero, J. Béjar, and U. Cortés. Go with the flow: Recurrent networks for wind time series multi-step forecasting. *Frontiers in Artificial Intelligence and Applications*, 308:79–83, 2018
- [4] J. Manero, J. Béjar, and U. Cortés. Predicting wind energy generation with recurrent neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11314 LNCS:89–98, 2018

## 7.1 Experiments with Deep learning architectures

This chapter applies several deep learning models (see Table 7.1) to the wind forecasting problem, to predict wind speed at 12 hours ahead horizon. In the wind prediction field, a horizon of 12 hours is considered mid-term prediction, and usually, approached with NWP approaches [73]., however in order to test the capability of the deep learning models, we require a challenging horizon to obtain meaningful results.

To answer the fundamental research question of this work, **Can DL be used for wind forecasting?** we need to prepare the experimentation accordingly. The first step is to establish the horizon  $H$  for the algorithms. In this case, we choose 12 hours as the horizon. A longer horizon will generate too large errors when using time series, and with a shorter horizon, the problem will be less complex. With this horizon, rarely used for time series prediction, the experiments can show its contribution to the prediction problem.

All predictions in this work are multi-step (versus a single prediction) obtaining as an output of the algorithms 12 steps of data that correspond to the future 12 predictions of wind speed ( $\hat{y}_{t+1}, \dots, \hat{y}_{t+H}$ ). All the experiments are executed on the NREL dataset (with 126,692 wind sites).

**Table 7.1**

Models developed in the experiments with short description and abbreviation used in this chapter

Family	Description	Abbr Used
Baseline	Prediction persistence	Persistence
Baseline	Random Forest sequence to sequence	RF
Baseline	$k$ -NN neighbors	$k$ -NN
MLP	MLP MIMO	MLP MIMO
MLP	MLP recursive prediction	MLP rec
MLP	MLP with Direct regression	MLP Dir
MLP	MLP sequence to sequence	MLP MIMO
CNN	CNN sequence to sequence 2 layers	CNN 2l
RNN	RNN Encoder Decoder	RNN ED
RNN	RNN MIMO	RNN

The architectures belong to four major families (see Figure 7.1), which are:

- Baseline - Baseline architectures (persistence or statistical)
- MLP - Multi layer Perceptron
- CNN - Convolutional Network
- RNN - Recurrent Neural Network

All the architectures are MIMO (Multiple-Input Multiple-Output), as they have a multi-variable sequence as an input and generate a sequence as an output. The MLP are well



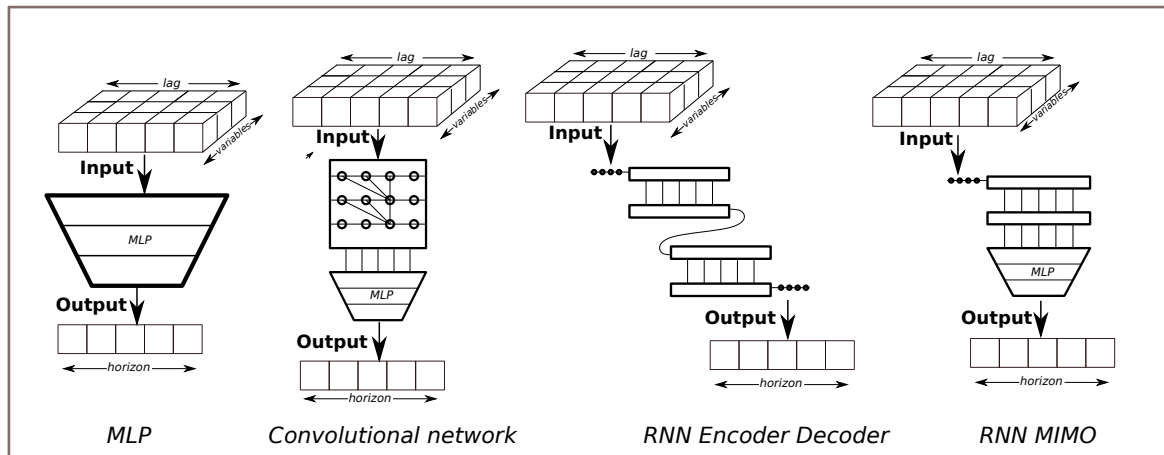


Fig. 7.1 Multiple Input Multiple Output (MIMO) deep learning architectures. MLP: MIMO ; CNN: Convolutional Network MIMO ; RNN MIMO and RNN encoder-decoder (ED) (see Section 7.1)

suitable for this construction as the first layer can be interpreted as a sequence and the output as another sequence. Convolutional networks have a MIMO structure by being stacked on top of an MLP, as the MLP performs the regression to the prediction sequence, and finally, the RNN performs the MIMO with two architectures. First, with a sequence-to-sequence architecture stacking an RNN with an MLP (similar to the CNN) or second, with just an encoder-decoder construction that obtains a sequence of fixed length directly, without the need of an MLP. In Figure 7.1 we show a graphical representation of both designs.

## 7.2 Experimentation Approach

The experimentation has 3 phases. The first one consists of the development of baseline methods, which are persistence, Random Forest and  $k$ -NN methods. The second one is to choose of the best multi-step forecast regression method, a selection to be made between direct, recursive and the MIMO approaches, the final phase is the development and experimentation of the different DL architectures with the complete set of data. To the obtained results, we apply a set of statistical analysis to infer a set of conclusions presented in the next sections and Chapter 11.

Table 7.1 shows the list of developed architectures in this phase, while the complete list of results from these experiments is in the Appendix A in Tables A.1, A.2 and A.3.

### 7.2.1 Calculation of Baseline methods

Auto-regressive or moving average methods have been discarded as baselines, as they cannot cope with the wind time series non-linearity and non-stationarity.

## Basic Deep learning architectures for wind time series forecasting

Persistence is the standard baseline method for wind prediction experiments. This method uses the last time step  $x_t$  as the prediction for each one of the  $H$  future time steps  $(x_{t+1}, \dots, x_{t+H})$ , for short term predictions, this naive method can be accurate, but for more than a few hours ahead predictions, its accuracy and deviation is high, voiding it as a feasible methodology for prediction. Nevertheless, we have chosen persistence as the first and primary baseline method.

The possible parameters for the optimisation are illustrated in Table 7.2. The architectures for the baseline methods are simpler, nevertheless requiring exploration with the methods described in Section 6.4.

**Table 7.2** Parameter space for baseline architectures

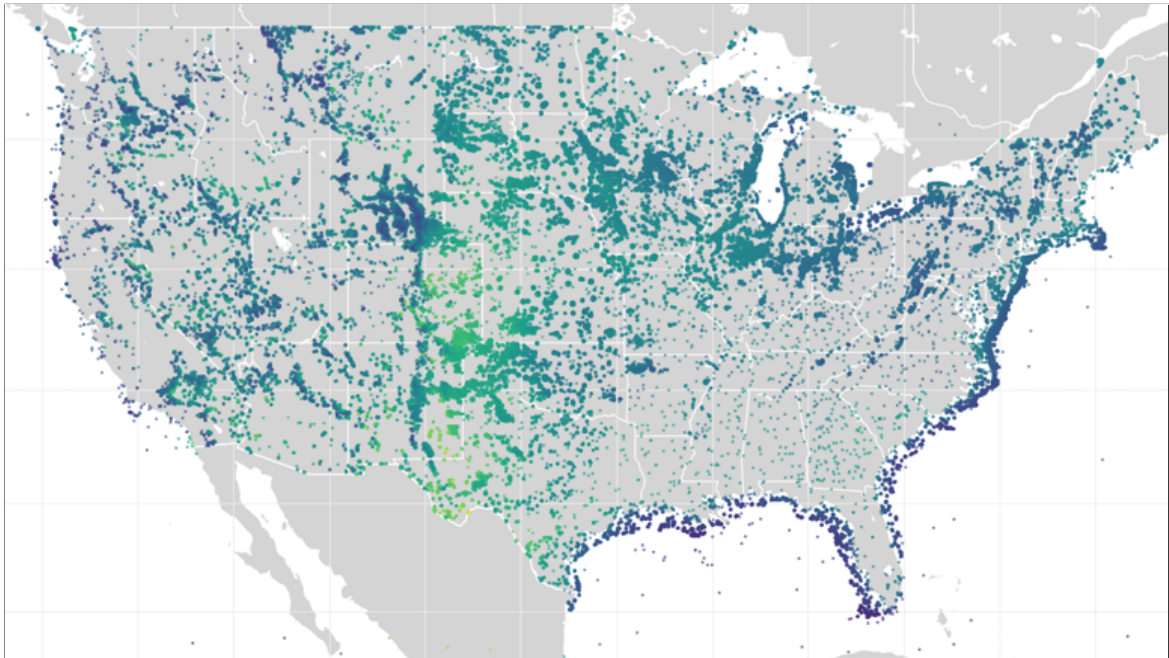
Method	Parameter	Space - Values - Comments
Persistence	None	No optimisation is required method has fixed parameters
RF	n_estimators	The number of trees in the Forest, the search values are 100, to 800
	max_features	How to consider features. It has been set to AUTO which implies that $\text{max\_features} = \sqrt{\text{n\_features}}$
	lag	Lag is set from 6 to 18 steps or hours
k-NN	n_neighbours	Number of neighbours space used 60 to 200 neighbours
	weights	Method for weighting distance. uniform (weighted equally) or distance (weighted inverse of distance)
	lag	Lag is set from 4 to 18 steps or hours

**Table 7.3** Baseline Methods and pbet architecture parameters used

Architecture	Description
Persistence	Persistence consists in using the real wind speed for time step $t$ as prediction for all the time steps from 1 to 12.
RF	Random Forest (RF). Ensemble of Trees, using bootstrapping new trees are generated recursively. All the features are used on each split and the number of trees generated is limited to 400, best lag parameter has been chosen to 12
K-NN	K-Nearest Neighbour using pre-processed examples data, lag 4 steps in length, and using 101 nearest neighbours (distance calculation Euclidean) and uniform weights- the prediction is obtained by averaging the next 12 hours of each neighbour.

Persistence is the naive method for a time series  $\langle x_1, x_2, \dots, x_n \rangle$  and uses the value  $x_n$  as a prediction for a  $\hat{Y}$  series with horizon  $H$  like  $x_{n+1} = x_{n+2} = \dots = x_{n+h} = x_n$ .

The Persistence calculation confirmed the literature findings [73], generated results that showed a steep accuracy ( $R^2$ ) deterioration as time steps increase (each time step worse accuracy), becoming negative for some wind sites, and, for most sites, the error in step 12 is high. When the results ( $\sum R^2$  of each wind site for the 12-hour steps) are positioned in a map, the areas with more variability of winds can be observed graphically (see Figure 7.2); comparing these areas with wind resource studies of the U.S. geography, (like the global wind atlas [8]). We observe that where the persistence ratings are weak, the variability of winds is very high. The most significant errors located in areas with complex terrains (Rocky Mountains) and with high wind variability (West Coast and Central Plains). As a result of this comparison, a research question arises. Is the error in persistence an indication of the terrain complexity of the site? This question is developed in detail in Chapter 10.



**Fig. 7.2** This map shows the 126,692 wind sites on a map of the U.S. The colours indicate the error measures as the sum of  $R^2$  for applying a persistence method 12 h ahead for all the sites. Dark is low error, while lighter colour shows a higher error, a measure that is an indication of the forecast complexity of the site.

Random Forest is the second baseline method. This choice is based on the known ability of decision trees to perform valid predictions on wind time series [222], and thus becoming a useful benchmark measure. The Random Forest regression model is an ensemble method based on decision trees with bootstrapping. The time-series is transformed into a set of examples  $E^*$  for training, and from this set, a random new set  $Z^*$  is constructed using the bootstrap technique with size  $N$ . (Bootstrapping consists in generating sets of samples from an initial set randomly with replacement). Then Trees are built recursively with the

bootstrapped set of data  $Z^*$ , generating an ensemble of  $N$  trees  $\{T_b\}_1^N$  the regression function will be then:

$$\hat{y} = (\hat{f})_N(x) = \frac{1}{N} \sum_{b=1}^N T_b(x) \quad (7.1)$$

The bias of the forest usually increases compared to a simple tree, but due to the averaging of the bootstrapped data, its variance decreases, hence generating a better learner than using individual trees. The strategy followed in the experimentation is to use all the features on each tree split action, while the number of trees chosen is 100.

Random Forest obtains excellent results on the training dataset, but with costly use of computer resources as the recursive approach is costly. The results show a 6.770  $R^2$  average over all the NREL sites for the test data (sixth year) and 6.532 for the validation (seventh year).

The last baseline consists of the application of a  $k$ -Nearest Neighbours ( $k$ -NN) approach. We have optimised the number of neighbours and how to combine its predictions, and the best  $k$ -NN model found uses pre-processed examples data with time windows of 6 h in length, using 15 nearest neighbours (applying Euclidean distance) and with prediction obtained by unweighted averaging of the next 12 h of each neighbour. The results of the  $k$ -NN accumulated accuracy (adding the accuracy of the 12 steps  $\sum R^2$ ) is 4.91 on average, better than persistence, but not close to the Random Forest, which obtained the best performance. The results for the three baselines are presented in Table 7.4.

**Table 7.4**  
Baselines distributions summary

Architecture	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
Persistence	2.699	1.951	2.486	1.919
Random Forest	6.770	0.804	6.532	0.737
$k$ -NN	4.675	1.02	4.430	0.941

Figure 7.3 compares the distributions in test and validation dataset for each experiment and shows the density plots of both sets of distributions.

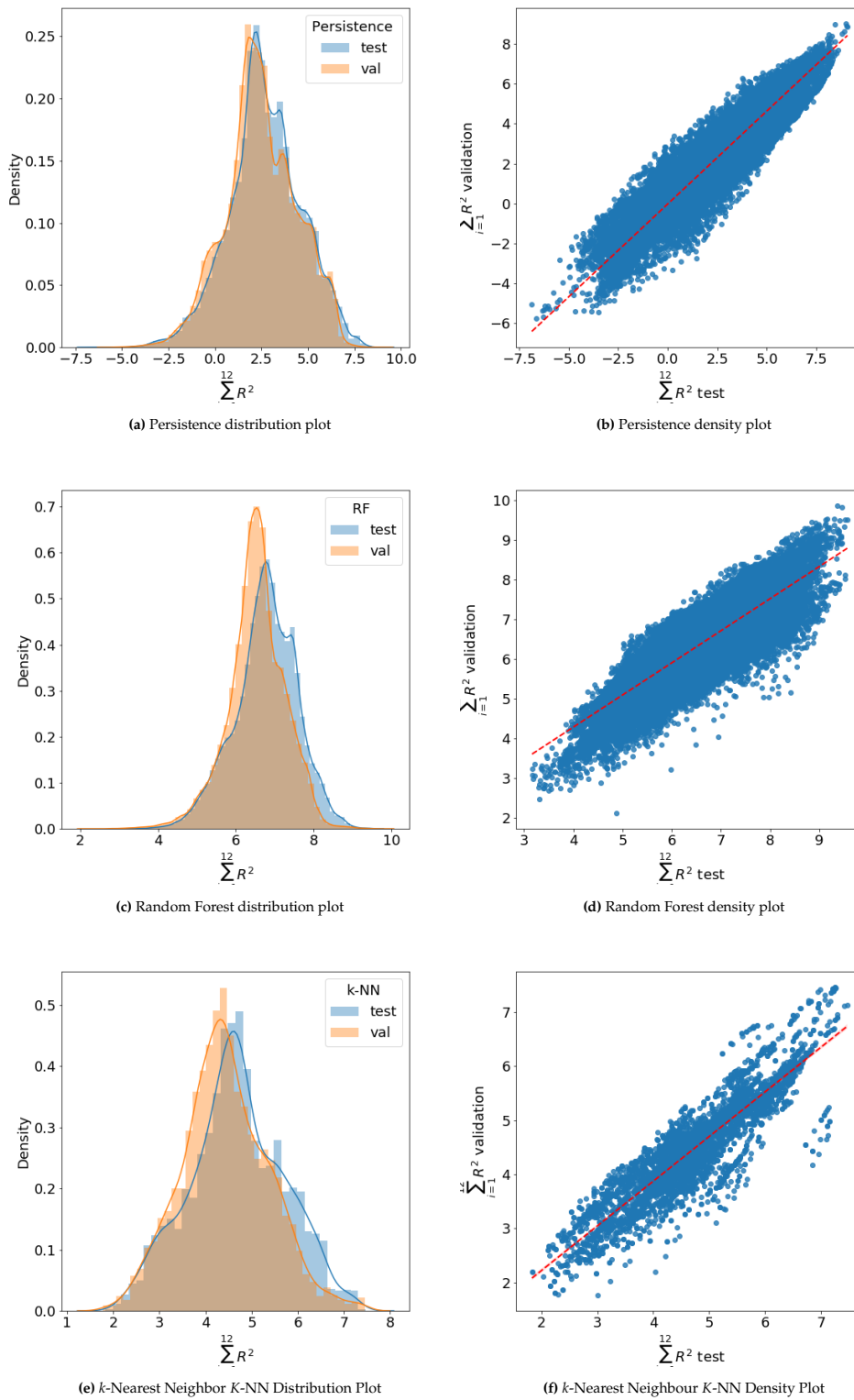


Fig. 7.3 Baseline plots, Persistence, Random Forest on whole dataset and  $k$ -NN on 4,200 sites

### 7.3 Determination of best regression strategy

In Section 2.4.3, we described different strategies for a multi-step-ahead forecast, which we summarise in three groups, regressive, direct or MIMO. The objective of this initial experimentation is to analyse, which is the best regression approach for the wind forecasting problem. This result is relevant as all the models and architectures will apply the best approach in the experiments. To determine the best regression strategy, we need to analyse

Table 7.5 Parameter space for MLP architectures

Method	Parameter	Space - Values - Comments
MLP	activation	The activation function to be used in the architecture. ReLU, Elu (with alpha), Leaky relu (with alpha), Prelu (with alpha), Linear, Hard_sigmoid, Tanh
	layers	We considered several combinations of layers and neurons per layer. Basically 1 to 6 layers with combinations of neuron numbers like [32, 64, 128, 256, 512, 1024, 2048]
	drop	0.0, 0.1, 0.2, 0.3, 0.4, 0.5. MLP architectures share same drop across layers
	lag	6, 12, 18, 24

the three different approaches empirically, as there is not a consensus on which is the best approach, as this depends on the problem and the data [14].

The main issue with wind time series is their non-linearity property. Most of the work about multi-step forecasting uses linear models because these methods are effective with linear time series. In some cases, with series that are weakly non-linear, the linear methods work quite well. Wind time series can be strongly non-linear, and for this reason, we can presume the superiority of the direct strategy over the recursive, as it is pointed, in previous works like [200, 110].

MIMO is a variant or improvement from the Direct approach. While the direct approach consists of applying one model per each time step to forecast, the MIMO approach generates them in "one go" (all steps at the same time). This model is inspired in the works of [110], [21] and [14], researchers that have gone more in-depth in the development of MIMO approach for non-linear time series forecasting. There are works in the literature that describe how the recursive approach introduces some uncertainty in the forecast, thus becoming less accurate than the other methods [208], despite knowing those precedents we tested this approach in the experimentation to verify its accuracy.

We tested the three approaches with the MLP (direct, MIMO and recursive). The experiment consisted of the execution on 2,000 sites (architectures with two layers of depth and input sequences with *lag* of 12).

### 7.3 Determination of best regression strategy

**Table 7.6**  
Direct and MIMO strategies description and parameters

Architecture	Description
MLP Direct	MLP with Direct regression, it has with two hidden layers with 1024 and 512 neurons with the ReLU activation function and dropout layers of 0.2 and one output layer with one neuron with linear output. This architecture requires being used as many times as time steps to forecast, 12 times (for horizon 12 h) in this work.
MLP MIMO	MLP sequence to sequence or Multi-Input Multi-Output (MIMO) series consists in predicting the output sequence in one regression.
MLP rec	The MLP recursive approach consists in a multi-regression but using (recursively) previous predicted values as input for the next step.

**Table 7.7**  
Direct Regression compared to Recursive and seq2seq (MIMO) approach

Architecture	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
MLP MIMO	7.254	$\pm 0.792$	6.954	$\pm 0.732$
MLP Direct Regression	7.241	$\pm 0.791$	6.938	$\pm 0.733$
MLP recursive	7.109	$\pm 0.788$	6.826	$\pm 0.728$

The experiment produced three distributions of accuracy results measured in  $R^2$ . Figure 7.4 illustrates the comparison between the distributions of the three methods and the means and variances of the distributions are in Table 7.7 for the Test and Validation datasets. To compare the mean values, we need to verify that these means are significant enough for comparison.

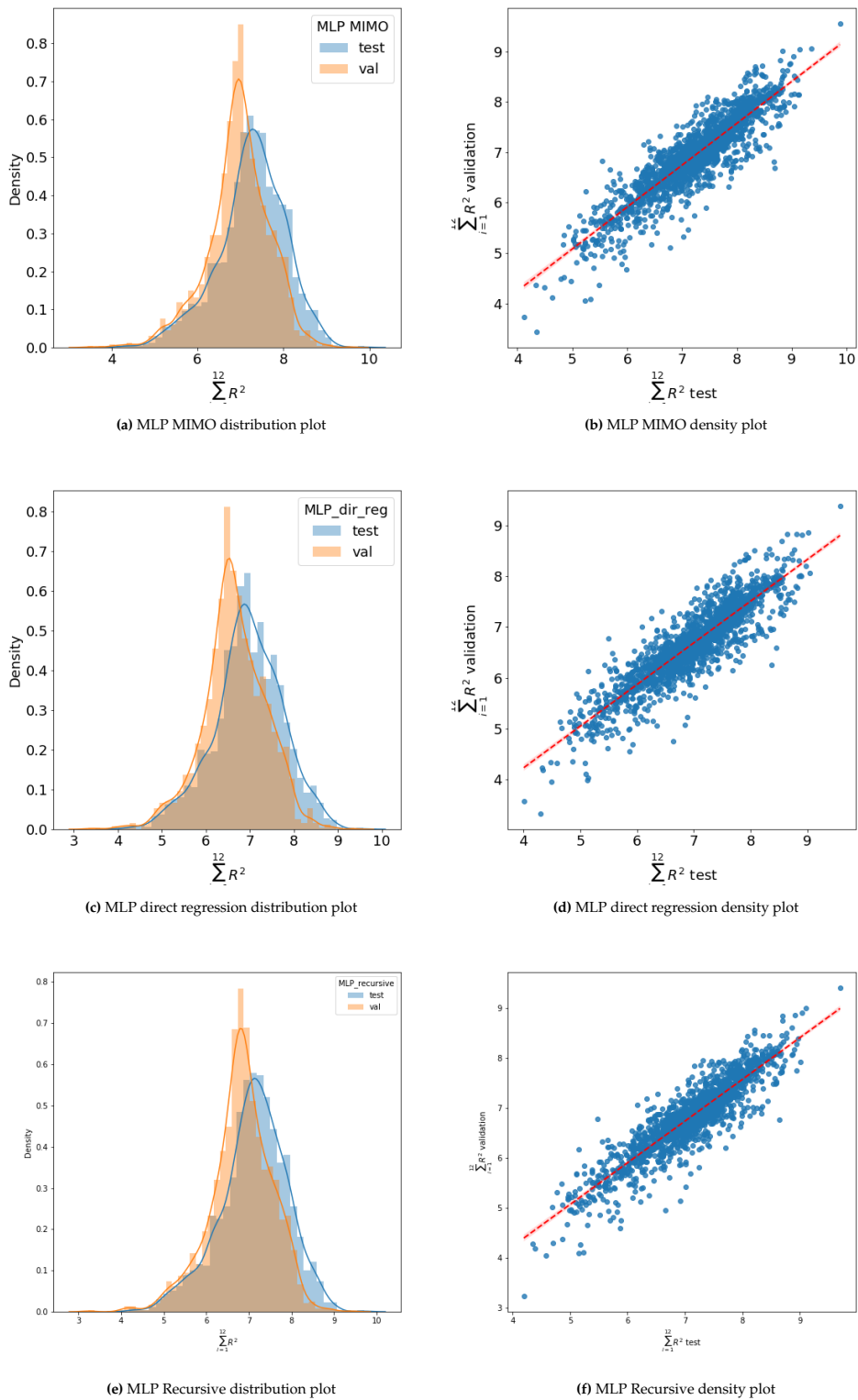


Fig. 7.4 Plots comparison for regression approach selection: MIMO, Direct Regression and recursive architectures



## 7.4 Main Experiment: Deep learning Models

To verify if the three distributions are different and comparable, we perform a one-way ANOVA test on the three distributions, which rejects the hypothesis that are the same. We apply a Tukey’s Honestly Significant Difference (HSD) test to verify the significance of the means from the distribution, which also rejects the hypothesis that are the same. In this way, we can assure that the MIMO approach represents a statistically significant improvement over the other two methods.

```
One-way ANOVA Test set
```

```
=====
```

```
F value: 69.89231156533315
```

```
P value: 1.0051486082850431e-30
```

```
Multiple Comparison of Means - Tukey HSD,FWER=0.05
```

```
=====
```

group1	group2	meandiff	lower	upper	reject
JM_MLP_DirREG	JM_MLP_MIMO	0.1371	0.0777	0.1964	True
JM_MLP_DirREG	JM_MLP_recursive	-0.162	-0.2214	-0.1026	True
JM_MLP_MIMO	JM_MLP_recursive	-0.2991	-0.3585	-0.2397	True

```
-----
```

The statistical analysis helps us to determine that the MIMO approach is slightly more accurate, and taking into account the higher computational requirements (as every step in the horizon prediction requires an individual model) of the direct regression, we can see that the MIMO approach has better accuracy in a wind forecasting application. The conclusion from this experiment is to use the MIMO approach in the architectures.

## 7.4 Main Experiment: Deep learning Models

We choose four architecture families for the experimentation, Multi-Layer Perceptron MIMO (MLP), Convolutional Network seq2seq (CNN), Recurrent Neural Network seq2seq (RNN), and Recurrent Network with the Encoder-Decoder architecture (RNN ED).

The universe of parameters for CNN and RNN is illustrated in Table 7.8, for the MLP is in Table 7.5.

## Basic Deep learning architectures for wind time series forecasting

**Table 7.8** Parameter space for CNN and RNN architectures. Some combinations are missing as some architectures allow for different parameters per layer (for example different kernels or filters per layer)

Method	Parameter	Space - Values - Comments
CNN	activation	The activation function to be used in the architecture. ReLU, Elu (with alpha), Leaky relu (with alpha), Prelu (with alpha), Linear, Hard_sigmoid, Tanh
	filters	The number of filters to be used in the convolution [16,32, 64, 128, 256, 512, 1024, 2048]
	strides	Strides in the convolution [1,2,3,4,5]
	kernel_size	kernel size for the convolution [1,2,3,4,5,6,7,8,9]
	drop	drop to apply to the layers[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
	depth_multiplier	Depth multiplier allows to multiply the number of channels in the convolution output [1,2,3,4,5,6,7,8]
	fulldrop	drop to use in the MLP layers [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
	full	layers and neurons of the MLP network [64],[128],[256], [512], [1024], [2048], [4096], [64,32], [128,64], [256,128], [512,256], [1024,512], [2048,1024], [128,64,32],[256,128,64], [512,256,128], [1024,512,256]
	activation full	Activation for MLP (with alpha), Prelu (with alpha), Linear, Hard_sigmoid, Tanh
lag	Input sequence [6, 12, 18, 24]	
RNN	activation	The activation function to be used in the architecture. ReLU, Elu (with alpha), Leaky relu (with alpha), Prelu (with alpha), Linear, Hard_sigmoid, Tanh
	neurons	Number of neurons in RNN layer [32, 64, 128, 256, 512, 1024, 2048]
	neuronsE	Number of neurons in Encoder layer for RNN ED [32, 64, 128, 256, 512, 1024, 2048]
	neuronsD	Number of neurons in Decoder layer for RNN ED layer [32, 64, 128, 256, 512, 1024, 2048]
	drop	drop in RNN layer [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
	nlayers	number of layers in RNN [1,2,3] there is a related parameter that is nlayersE and nlayersD which are the number of encoder-decoder layers
	activation full	Activation for MLP (with alpha), Prelu (with alpha), Linear, Hard_sigmoid, Tanh
	fulldrop	drop to use in the MLP layers [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
	full	layers and neurons of the MLP network [64],[128],[256], [512], [1024], [2048], [4096], [64,32], [128,64], [256,128], [512,256], [1024,512], [2048,1024], [128,64,32],[256,128,64], [512,256,128], [1024,512,256]
lag	Input sequence [6, 12, 18, 24]	

## 7.4 Main Experiment: Deep learning Models

For all these models, the selection of parameters is obtained by hyper-parameter searching using a selection of controlled sites and then tested with the whole dataset (see Section 6.4).

The question to answer is: Which method is better for wind forecasting? After performing all the experiments, we compare the  $R^2$  distributions as we illustrate in Figure 7.8.

The first issue would be how to compare the different distributions, and this was accomplished by performing an Analysis Of Variance (ANOVA) test, which determines if there are significant statistical differences between the means of all the obtained distributions.

**Table 7.9** Deep learning Basic Architectures details.

Architecture	Description
MLP MIMO	MLP with two hidden layers with 258 and 128 neurons with the ReLU activation function and dropout layers of 0.4 and one output layer with 12 neurons (equal to the prediction horizon of 12 h) with linear output.
CNN	One 1D convolutional layer with the ReLU activation function, 256 filters with a stride of 1 and a kernel size of 5, followed by an MLP with an input layer of 32 neurons with the linear activation function and one output layer with as many neurons as the prediction horizon with the linear output, in this case 12, as the horizon is 12 h.
RNN MIMO	Two recurrent layers with 32 neurons each, using GRU units with a hard sigmoid recurrent activation function, a ReLU activation, and with recurrent dropout of 0.1 followed by an MLP with an input layer of 512 neurons with a sigmoid activation, a dropout of 0.1, and a final linear output layer with as many neurons as the prediction horizon, in this case 12, as the horizon is 12 h.
RNN ED	An RNN encoder with two layers of GRU units of 96 neurons and an RNN decoder with one layer of GRU units with 64 neurons both with the hard sigmoid recurrent activation function, ReLU activation function, and a dropout of 0.3, each time step of the prediction horizon is with a linear activation function.

The question to answer is: Which method is better for wind forecasting? After performing all the experiments, we compare the  $R^2$  distributions as we illustrate in Figure 7.8.

The first issue would be how to compare the different distributions, and this was accomplished by performing an Analysis Of Variance (ANOVA) test, which determines if there are significant statistical differences between the means of all the obtained distributions.

The test result obtains an F-value of 2.655 and a  $p$ -value of 0.0317. These results tell significant differences among the group of means. To assess the individual differences, we use Tukey's honestly significant difference test (see Table 7.10). This test compares the distributions assessing the pair differences.

## Basic Deep learning architectures for wind time series forecasting

**Table 7.10** Distribution comparison between the experiment using Tukey’s honest significant difference test, Family wise error rate  $FWER = 0.05$ , after the ANOVA test of a  $F$ -value of 2.655 and a  $p$ -value of 0.0317.

Group 1	Group 2	Meandiff	Lower	Upper	reject
CNN	MLP	0.1485	0.1402	0.1568	True
CNN	RNN	-0.0566	-0.0649	-0.0483	True
CNN	RNN ED	-0.0409	-0.0491	-0.0326	True
MLP	RNN	-0.2051	-0.2134	-0.1968	True
MLP	RNN ED	-0.1894	-0.1976	-0.1811	True
RNN	RNN ED	0.0158	0.0075	0.0241	True

The results point to the MLP method as the better overall, with a more significant positive difference in the mean compared to the other methods. CNN was better than both RNN variations while RNN encoder-decoder was marginally better than RNN sequence to sequence. From the closeness of the results, we observe that the worse and best models are not very far from one another.

It could seem to counter-intuitive that recurrent neural networks models do not have the best performance in this domain, given that they are specifically tailored to process sequential data. There has been recent literature questioning the necessity of recurrent models in certain domains including time series prediction where high dependencies are not needed [190, 10] or because the recurrent models used are stable (no gradient problems during the optimisation) and can be approximated by feed-forward models [150]. Some feed-forward and convolutional architectures outperform recurrent ones in some examples of automatic translation [69] or speech synthesis [231].

A boxplot of the distribution of  $R^2$  for each time horizon (see Figures 7.5 and 7.6) also shows some relevant characteristics. The RNN and CNN models performed better in the short term (1–2 h) than the MLP ones, which showed better results in the 3–12-h interval. In this sense, it is observed that RNN and CNN are better at learning the short-term characteristics of the series, while MLP is more consistent in the full 12-h prediction. Again, the differences were small, but relevant, as calculated on the 126,692 sites, and showed a strong trend.

We developed a map of the persistence accuracy across the North America geography (see Figure 7.2). With all the experimental results obtained, we include in the map the best architecture for each site (see Figure 7.7), allowing a graphical analysis of the results to identify geographical patterns related to architectures.

- MLP showed better accuracy in most places
- CNN showed the best accuracy in the Rocky Mountains and the Florida and North Carolina Atlantic Coastline.

## 7.4 Main Experiment: Deep learning Models

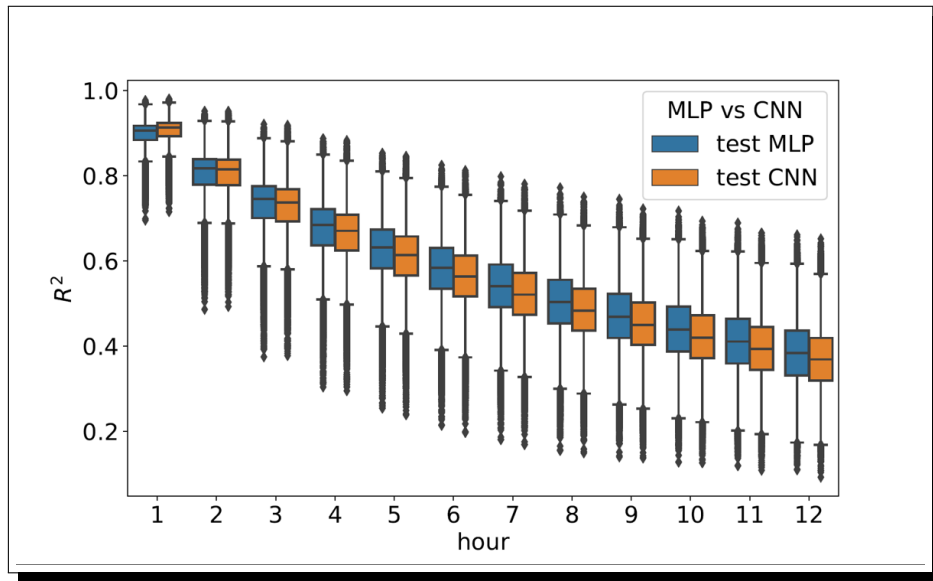


Fig. 7.5 Boxplot comparing the hourly distribution of  $R^2$  error results between the MLP and CNN methods.

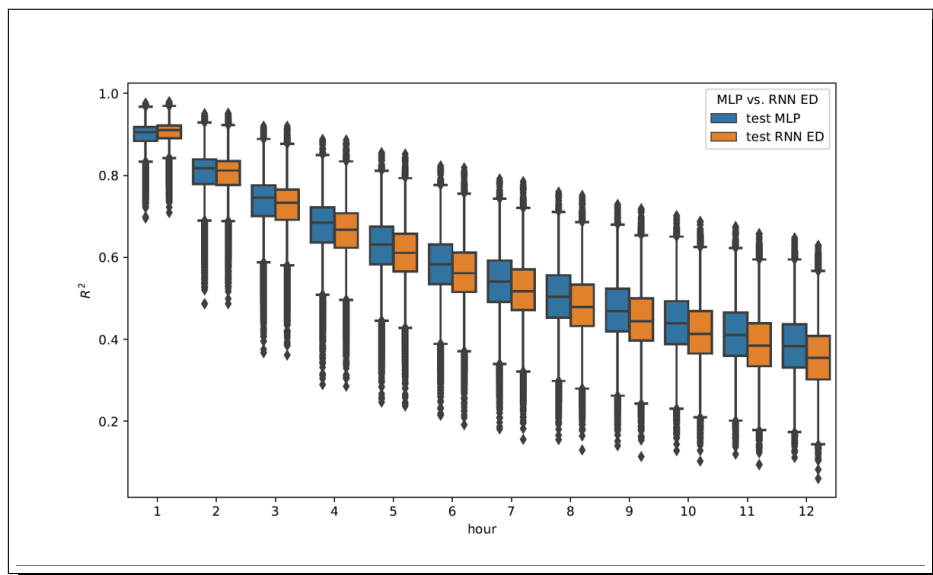


Fig. 7.6 Boxplot comparing the hourly distribution of  $R^2$  error results between the MLP and RNN ED methods.

- RNN (encoder-decoder) and RNN seq2seq became the best methods in the western area of the U.S. between the Rockies and the Pacific West Coast, especially in the Nevada and Arizona deserts.

From the information on the comparison map (see Figure 7.7), a relationship between site location (terrain and local characteristics) and the best model can be drawn. This relationship opens possible new areas of research, some of them discussed in Section 7.5. Is the accuracy

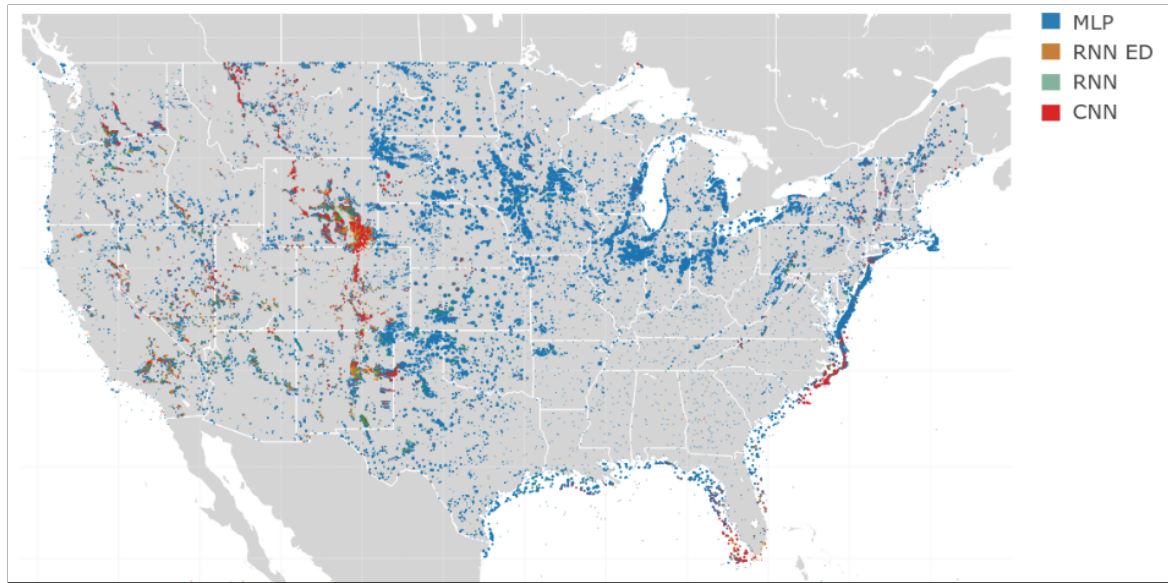


Fig. 7.7 Graphical analysis of the best model for each site. Each site is coloured with the best method in that site and shows some geographical wind patterns in the U.S. geography, opening the discussion of the relationship between wind site topology and deep learning method.

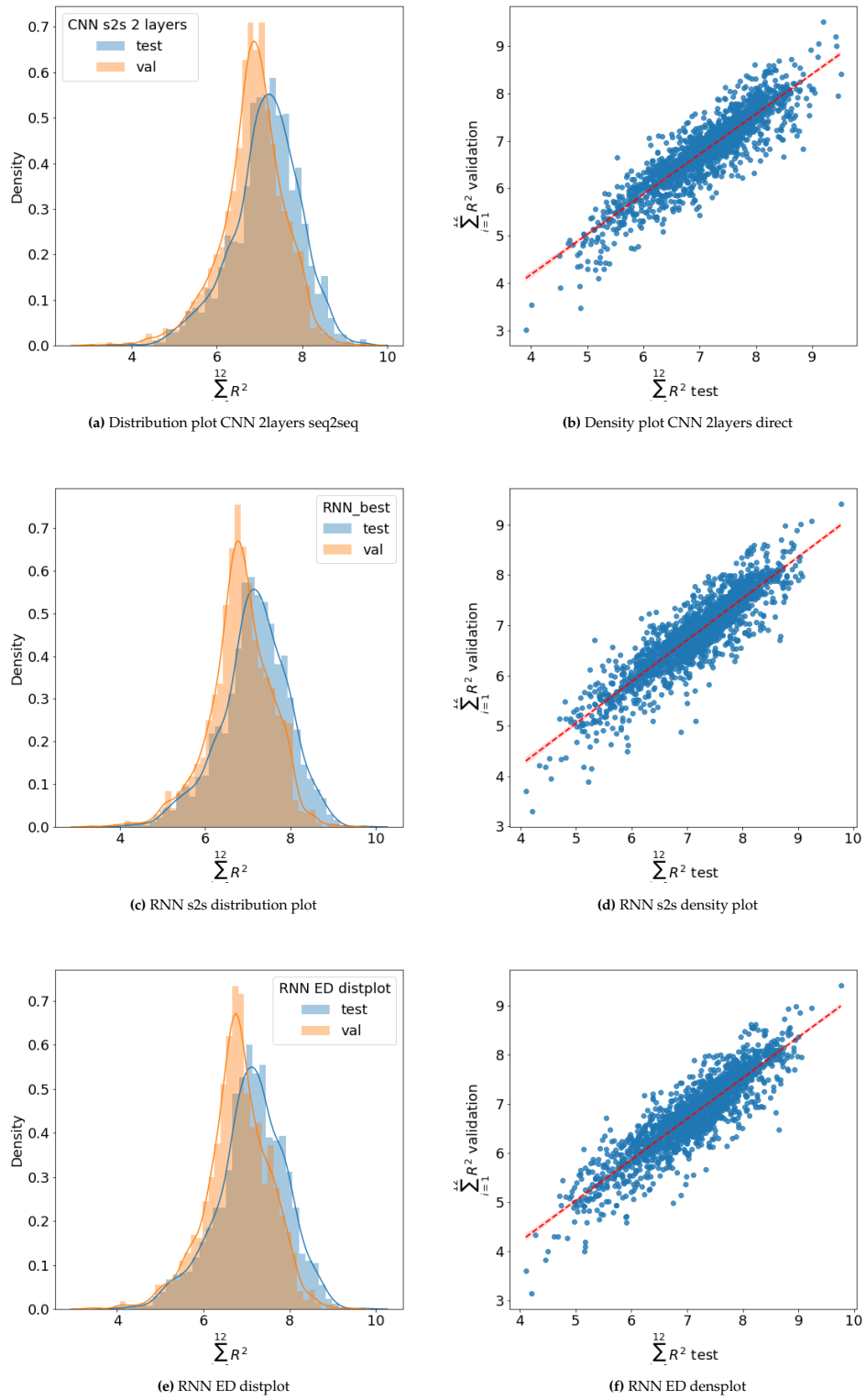
Table 7.11 Mean and standard deviation of probability distributions for main architectures

Architecture	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
MLP MIMO	7.254	$\pm 0.792$	6.954	$\pm 0.732$
CNN 2 layers	7.146	$\pm 0.795$	6.840	$\pm 0.751$
RNN MIMO	7.147	$\pm 0.797$	6.823	$\pm 0.736$
RNN ED	7.101	$\pm 0.804$	6.790	$\pm 0.743$

constant over the years? If there are changes caused by drifting winds, seasonal weather model modifications, and climate change, then the accuracy will vary over time. With seven years of data available, five for the initial training, one year for a test, and another for validation, an analysis of the accuracy evolution was made. The comparison between the test and validation datasets that used the sixth and seventh year respectively showed differences in their distributions. In this case, applying a  $t$ -test for two related samples, given that the scores were computed for the same population, resulted in a significant difference in means, as can be seen in the Figure 7.8 that summarises the results for the test and mean datasets of the experiments.

Computing a linear regression between the validation and the test dataset results, we obtain a value of 0.97. This high value is showing a great correlation between the predictions, which tells that the model is consistent for the 6<sup>th</sup> and the 7<sup>th</sup> years.

## 7.4 Main Experiment: Deep learning Models



**Fig. 7.8** Plots of CNN and RNN architectures

### 7.5 Discussion

The experimentation shows that the series *lag* adequate for training is quite short (12-18 steps or hours), a finding aligned with the physics of wind formation.

Another finding is related to the importance of hyper-parameter optimisation. A sample architecture shows over 20% variability between its vanilla version (manual selection of parameters) and the tuned one (after an extensive parameter space search). This result is another expected outcome as the deep learning models have many parameters that require a structured parameter optimisation to release all their potential.

Regarding the specific architectures, we observe that the RNN have not rated as the best, a finding that conflicts with the naive understanding about how the recurrent neural networks are better for interpreting sequences. In this case, we can infer that the recurrent networks do not perform so well due to the nature of the problem, with short input and output sequences, both of fixed length.

MLP and CNN models have shown the best results. with a slight advantage on the MLP. However, from the experimentation we have observed higher variability in the CNN models, for this reason we have chosen the convolutional operation as a target for further specific experimentation in order to understand better the capacity of the convolutional operations on multi-step wind time series forecasting (see Chapter 9).

On the question regarding the validity of training for the sixth and seventh year, we have shown that the training is still valid for both years.

Finally, we have noticed that the methodology is well defined for overall accuracy assessment, but with a single value for an architecture it is impossible to understand the dynamics of the prediction in a single site. In the thesis the value that qualifies the result of an architecture (see result tables in Appendix A) comes from calculating the mean of the accuracy distribution results ( $R^2$ ). However, as the distributions come from thousands of sites, it is difficult to understand the behaviour of the prediction for a specific geographical location. To interpret the values in a single site, we have designed a representation that allows to visually observe how the algorithm performs in a site over a year (see Figure 7.9)

The strip visualisation allows to understand the accuracy of the prediction for a specific site. In the next pages we use this representation to illustrate the accuracy over a whole year of a method applied to a single site. The horizontal axis correspond to time steps and covers a full year, while the vertical axis to the 12 hours prediction.

The accuracy calculation in the figure is made based on the difference between predicted and real values, measured in  $m/s$ . A point in the figure, for wind speed, is the difference between prediction and real value for a time step  $i$  and for a prediction step  $j$ , the value for one step  $i = (1, \dots, 8760)$  and  $j = (1, \dots, 12)$  is  $v_{ij} = \hat{y}_{ij} - y_{(i+j)}$ . The colours represent the absolute value of  $v_i$ . White colour is zero, which is a perfect prediction, while dark red signals a positive deviation of 15  $m/s$  or more. Dark blue is a negative deviation of -15  $m/s$ .



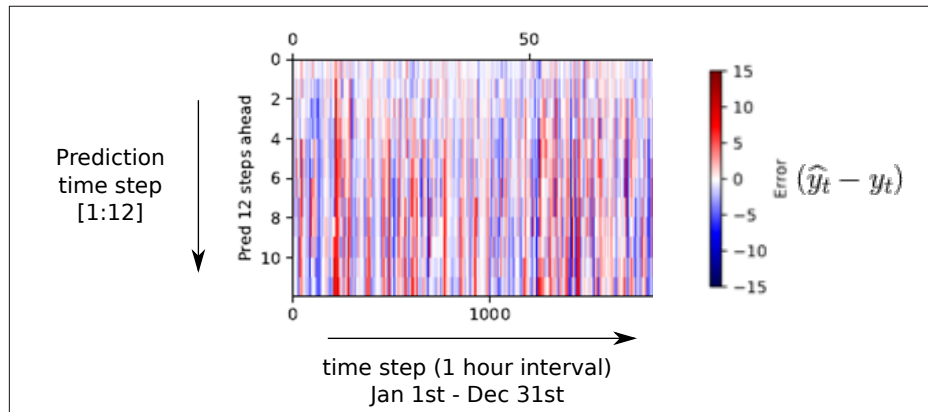


Fig. 7.9 Interpretation of strip visualisation

The visualisation is used to understand the accuracy patterns of a given prediction for a site. By observing how the colours change between seasons, or by comparing them with the results from other sites we can understand which model is more accurate and how it represents seasonality, under-prediction or over-prediction.

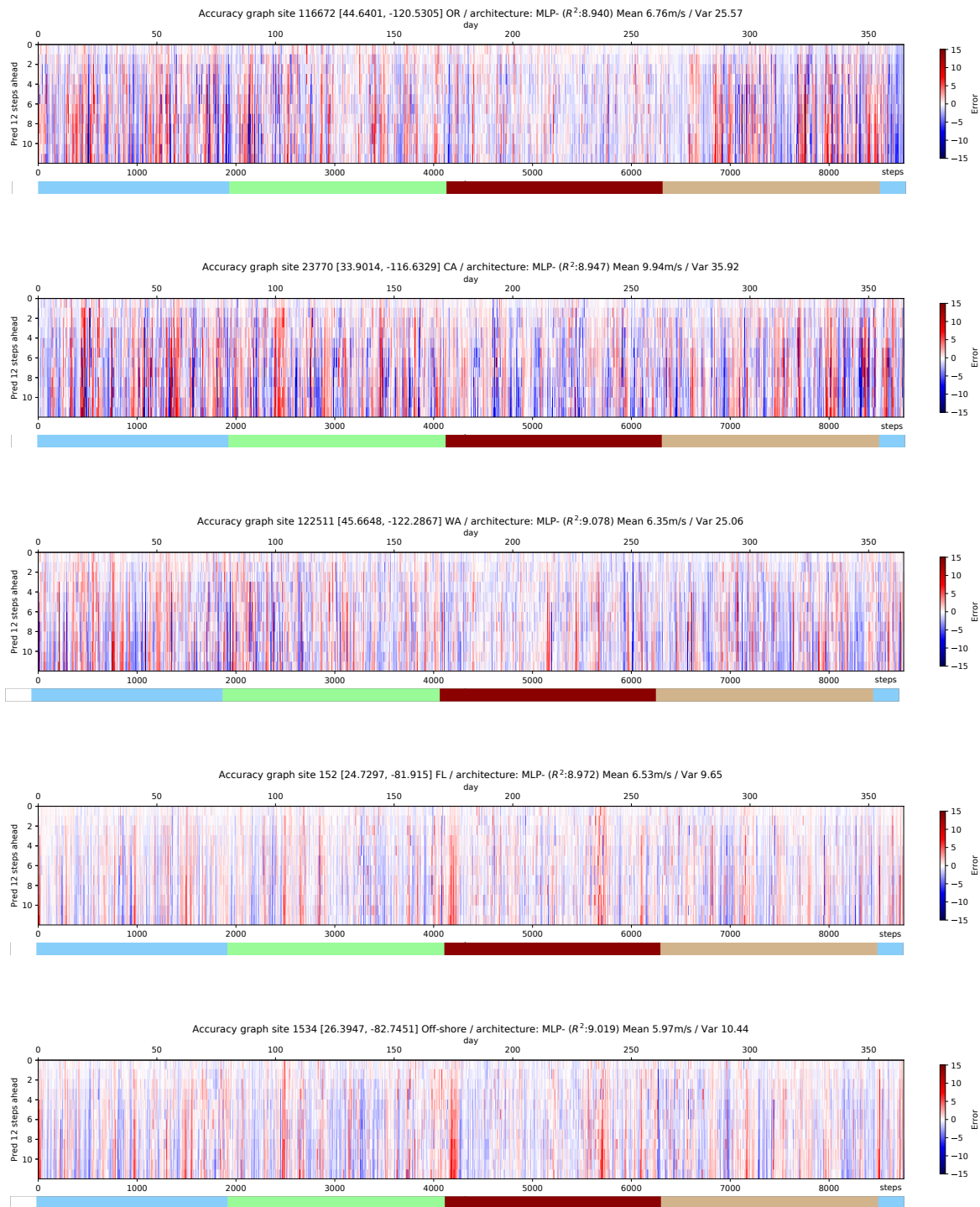
In the next pages we illustrate these interpretations with ten strip representations, in two groups of figures, in the first we selected randomly between the best accuracy sites, and in the second from the worst, as follows,

1. Figure 7.10 : Prediction accuracy. Sites selected from best accuracy measured in  $R^2$  (above 9.0 average)
2. Figure 7.11 : Prediction accuracy. Sites selected from low average accuracy measured in  $R^2$  (below 4.9 average)

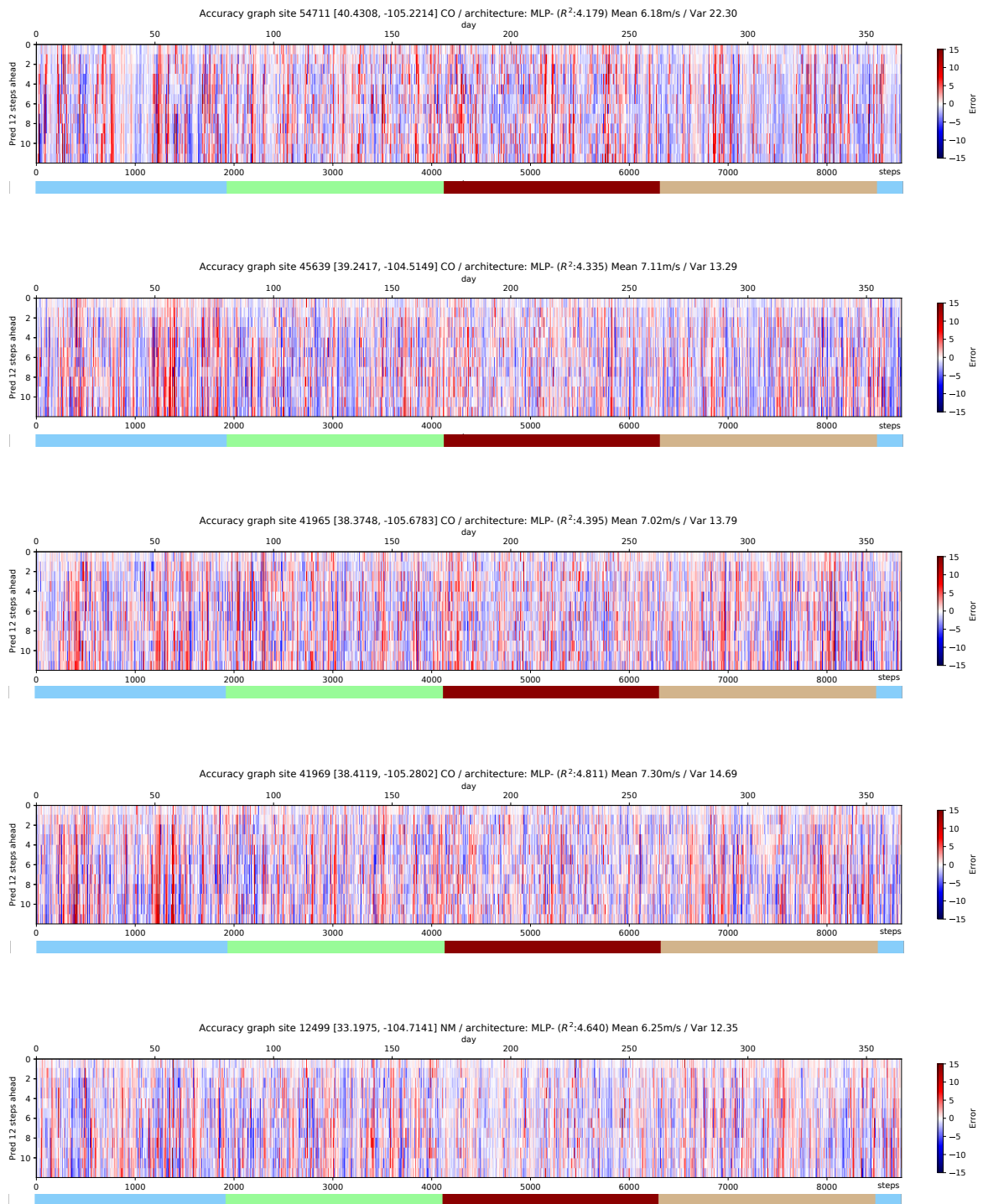
In Figure 7.10 we can observe how the top two sites (116672 and 23770) show a clear seasonality between summer and winter. The other three sites have less seasonality and specially the bottom two (152, 1534) show good prediction across all seasons. Both sites are located in the same area (Florida-Gulf of Mexico) and we see corresponding errors in the accuracy (Red vertical stripes around step 4200 and blue around step 7000).

In Figure 7.11 there predictions have less accuracy in general (darker colours) and we can observe better accuracy in summer in the bottom site (12499 - New Mexico) and in the top one (site 54711 - Colorado) we see a period of good accuracy at the end of winter.

## Basic Deep learning architectures for wind time series forecasting



**Fig. 7.10** Accuracy strips for 5 sites with MLP  $R^2$  accuracy above 9.0. Vertical axis represents the 12 steps in the horizon (top is first step, bottom 12<sup>th</sup> step), horizontal axis are the hours in a year. The horizontal coloured bar illustrates seasons (winter: blue, spring: green, summer: red, fall: brown)



**Fig. 7.11** Accuracy strips for 5 sites with MLP  $R^2$  accuracy below 4.9. Vertical axis represents the 12 steps in the horizon (top is first step, bottom 12<sup>th</sup> step), horizontal axis are the hours in a year. The horizontal coloured bar illustrates seasons (winter: blue, spring: green, summer: red, fall: brown)



## Chapter 8

# Going Deeper with wind time series forecasting architectures

After the first wave of experiments, we designed new architectures with new features and models. We use RNN and MLP basic models, extending them with new characteristics. Another addition has been the use of series sampled every five minutes, to predict at twelve hours and one hour ahead. A final set of experiments consists of adding some weather information into the series. This variable is a low-resolution weather forecast measure (in these experiments is temperature), and we analyse the impact in the model accuracy. This chapter has four main sections. The first one analyses variants on the MLP architecture. The second section analyses new approaches with RNN architectures using GRU and LSTM cells. Then we describe how the results improve by the use of higher frequency series, and the last section introduces some weather forecast information in the series.

## 8.1 Introduction

In Chapter 7, we described a set of experiments with the basic MLP, CNN and RNN architectures. In this chapter, we perform experimentation on MLP and RNN architectures using different variations and features that add some new capabilities to them. The Convolutional networks are analysed in depth in Chapter 9. Then we perform two main additional changes in the experimentation. Firstly, we experiment with series with a shorter period, and we use them for prediction at one hour ahead, and twelve hours ahead. The final experiments deal with the introduction of some weather information into the network. The purpose is to verify if there is a change in the accuracy of the networks.

## 8.2 MLP architecture variants

In Chapter 7, the experimentation showed that the MLP architectures are a good fit for multi-step wind speed forecasting. In this section, we explore two variations on the classic architecture, one with cascade connections and then with the SJOINT model. We analyse several regression approaches alternative to the direct regression and the MIMO approach.

Table 8.1 MLP additional Architectures

Architecture	Description
MLP skip	Based on the baseline MLP Architecture, connections between input and input of new layers are connected. This allows for deeper architectures. The example shown in the results has one layer of 1024 neurons, followed by two layers of 512 neurons.
MLP SJOINT	Multihorizon SJOINT strategy Training is done separately for blocks of horizons (if the block size is one this is a direct regression) as generating a model with multiple regressions

Both architectures are refinements of the classic MLP. MLP Cascade adds connections between layers, while MLP SJOINT experiments with training strategies that try to extract information to regress not a single output but multiple outputs.

### 8.2.1 Skip connections in MLP architectures

Skip connections are defined to avoid the vanishing gradient issue with neural networks by adding additional connections from the input to the network to the intermediate layers and concatenating this input with the intermediate layer output. Skip connections are used in large Convolutional networks but can be added to MLP networks as described in [241] where they use an MLP with a single skip connection.

In this section, we perform several experiments with skip connections in an MLP network. (see Figure 8.1). This architecture shares the inter-layer connections with the skip models. The connections allow maintaining the input values across the layers avoiding vanishing values when close to zero.

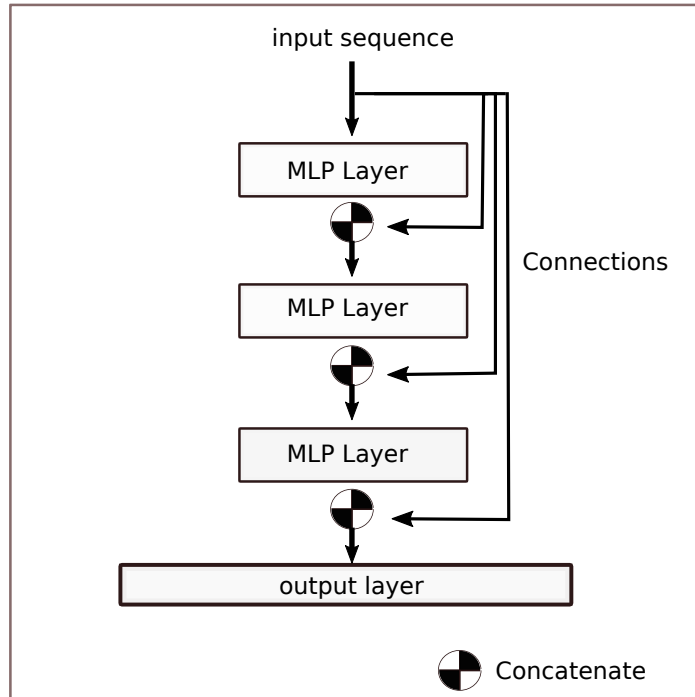


Fig. 8.1 Skip connections in MLP architecture

Table 8.2 Mean and standard deviation of probability distributions for MLP additional architectures

Architecture	Layers	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
MLP SKIP	1L	7.170	$\pm 0.801$	6.847	$\pm 0.737$
MLP SKIP	2L	7.253	$\pm 0.796$	6.948	$\pm 0.734$
MLP SKIP	3L	7.250	$\pm 0.801$	6.940	$\pm 0.739$
MLP SKIP	4L	7.233	$\pm 0.800$	6.922	$\pm 0.741$
MLP SKIP	5L	7.216	$\pm 0.800$	6.903	$\pm 0.744$

The different experiments use the same parameters as the classic MLP architecture for comparison results, and we have experimented with skip connections with one to five layers. The results obtained show how the skip connection maintains the accuracy in deeper models, a reasonable outcome, as the cascade connection avoids the vanishing gradient to happen in the deepest layers by refreshing the input sequence on each layer.

The comparison experiment uses similar parameters for the cascade and for the classic architectures, to obtain a valid comparison. It is relevant to mention that we did not perform

**Table 8.3** Comparison Cascade with Classic MLP

Layers	Classic MLP	MLP skip2
1 Layer	7.169	7.170
2 Layers	7.253	7.253
3 Layers	7.211	7.250
4 Layers	7.181	7.233
5 Layers	7.111	7.216

a complete parameter optimisation for each one of the architectures, to level the field for the comparison. For illustration purposes, we compare the cascade connections in Table 8.3.

### 8.2.2 Multi-regression architectures MLP SJOINT

In Chapter 7, we performed the experimentation with three regression strategies, which are; MIMO, that generates the whole output sequence in one simultaneous step, recursive regression, where a regressed step is used by the next regression recursively, and the multi-regression approach, where there is an independent regression for each future prediction time step.

However, there are intermediate strategies like the SJOINT, which is an intermediate approach [208], sitting between the direct regression and the multi-step regression. It divides the output sequence into blocks of consecutive steps, and then the architecture performs a regression for each block. There will be a different set of parameters  $\theta$  for each set of parameters.

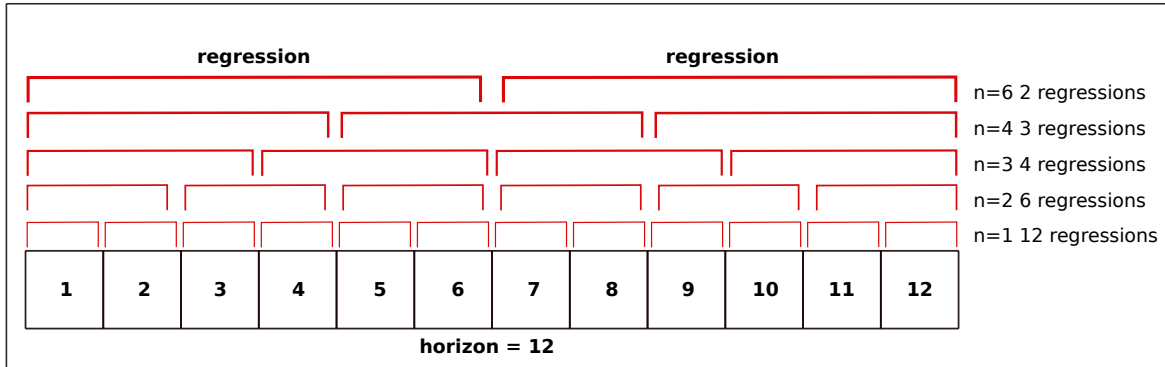
Being  $n$  the length of the block and  $H$  the number of steps in the prediction, the number of blocks is  $L = H/n$ .  $L$  is also the number of regressions performed in the model (see Figure 8.2).

Our SJOINT architecture only performs regressions on sequences of equal length, thus reducing the total number of possibilities, and making the experiments more straightforward. For a horizon  $H$  of 12, we can perform this the SJOINT regression in six different ways (see Figure 8.2).

Each regression trains on a slice of output sequence. As the horizon is twelve, the number of possible slices is the divisors decomposition of  $H$ , in this case (1,2,3,4,6,12), the slice determines the block size for the regression. As an example, for a slice block of one, there are twelve possible regressions, where for a slice block of six, there are two. When the model performs a single regression, then is the equivalent to the MIMO approach, and when performs twelve regressions is equivalent to the already analysed direct regression (see Section 7.3).

In the analysis of Direct Regression compared to the MIMO approach, the MIMO architecture obtained a slightly better result, but with a short margin (just over 1%). When





**Fig. 8.2** Multiple regressions with Sjoin architecture,  $n$  is the size of the contiguous steps to be considered for training

performing the SJOINT comparison, we obtain the SJOINT with block size six (two regressions) as the best performing architecture. The architecture with three regressions and block size four obtains a better result than the MIMO approach.

These results are telling us that the algorithms perform better if we divide the sequence to predict in chunks of three or four hours. The time length aligns with the physical properties of wind, which changes in periods of approximately 3 hours [117] and with the conclusions from the experimentation in this thesis regarding the wind-speed series length (see Section 11.3.1).

We leave an open question regarding *how* the SJOINT multiple regression strategy behaves with other deep learning architectures (like CNN or RNN), nevertheless is relevant to point out that the small improvement of accuracy by using SJOINT require to double (or triple) the computer resources for the training process.

**Table 8.4** Mean and standard deviation of probability distributions for MLP Multi-regressions. Slice is the size of the block to predict (slice 2 =  $12/2 = 6$  regressions)

Architecture	Block	Regressions	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
MLP SJOINT	1	12	7.241	$\pm 0.791$	6.938	$\pm 0.733$
MLP SJOINT	2	6	7.247	$\pm 0.790$	6.943	$\pm 0.731$
MLP SJOINT	3	4	7.252	$\pm 0.788$	6.947	$\pm 0.731$
MLP SJOINT	4	3	7.255	$\pm 0.788$	6.949	$\pm 0.730$
MLP SJOINT	6	2	7.257	$\pm 0.790$	6.953	$\pm 0.730$
MLP SJOINT	12	1	7.253	$\pm 0.792$	6.951	$\pm 0.731$

Performing an Tukey's honestly test with the distributions to check if they are significantly different, we obtain false results, concluding that the differences are not significant.

### 8.3 RNN Architectures GRU or LSTM cells and Attention

We analysed two basic RNN architectures in Section designed explicitly for wind prediction. The RNN Encoder-Decoder and the RNN MIMO. Both architectures use the basic RNN cell. In this section, the objective is to understand the impact on the architecture accuracy by using alternative cells, GRU or LSTM.

**Table 8.5** RNN alternative models

Architecture	Description
RNN MIMO GRU	RNN MIMO architecture with GRU Units
RNN MIMO LSTM	RNN MIMO architecture with LSTM units
RNN ED s2s GRU	RNN ED with GRU units
RNN ED s2s LSTM	RNN ED with LSTM units
RNN MIMO GRU Attention	RNN Attention GRU
RNN MIMO LSTM Attention	RNN Attention LSTM

In Section 3.6, we analysed the RNN architectures, describing the different cells used by the recurrent networks. The primary three cells are; LSTM designed to avoid the vanishing gradient with long sequences, GRU, which is a simplified version of LSTM that works better for shorter sequences, and finally the vanilla RNN cell.

In the initial parameter exploration for RNN in Chapter 7, we used GRU cells for the RNN architectures. In this section, we want to see how the LSTM cells impact the accuracy of the RNN networks.

The expectation for the LSTM cells is to work better for longer and more complex sequences, and the question is if the horizon and *lag* of the learning sequences can be considered short or long. In principle, our understanding is that series of 6 to 18 steps long (*lag*), or 12 steps (Horizon) are not long enough for the LSTM cell to deploy all its capabilities. This assumption is confirmed by our experimentation (see summary Table 8.6).

**Table 8.6** Mean and standard deviation of probability distributions for alternative RNN architectures

Architecture	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
RNN s2s GRU	7.147	$\pm 0.798$	6.822	$\pm 0.735$
RNN s2s LSTM	6.952	$\pm 0.805$	6.635	$\pm 0.738$
RNN ED s2s GRU	7.109	$\pm 0.805$	6.786	$\pm 0.749$
RNN ED s2s LSTM	6.965	$\pm 0.807$	6.652	$\pm 0.749$
RNN s2s GRU att	6.598	$\pm 0.826$	6.290	$\pm 0.743$
RNN s2s LSTM att	7.003	$\pm 0.804$	6.700	$\pm 0.744$

The results with LSTM cells are consistently less accurate than with GRU cells, and this confirms our hypothesis. The fact that the series are of fixed length is another element that makes its sequence management by the recurrent network easier, explaining an additional advantage that the GRU cells have with this problem.

Finally, we added attention capabilities to the RNN cell. Attention consists of creating a trainable layer of weights that act as context, used by the network to improve the sequence learning. It is used in applications where the sequences of data are long and complex to understand (like Natural Language Processing). The mechanism implemented is based on the attention model by Bahdanau et al. described in 2015 in [9]. This attention is additive and considers the whole context.

During the experimentation phase with all the architectures, we observed that the optimal input length of the sequences for the prediction is short (the *lag* parameter) sometimes as short as six steps, a fact that limits the impact of mechanisms defined for longer and more complex sequences. When applied to the wind speed prediction problem, the attention mechanism has a little impact, not generating improvements in the algorithm accuracy. However, we observe a better behaviour with GRU cells over LSTM.

### 8.4 Using series with a 5 minute step

The series used for the experimentation in this thesis have steps with one-hour period (see Section 6.2.1). However, the original data was created with five minutes periods. Based on previous works [80] where they observe better results from series with higher frequency, in this section we outline several experiments intending to confirm if shorter period series have more useful information, allowing the models to obtain better results with higher accuracy. With this objective, we plan an experiment with series that have the original period of five minutes.

The series sampled at five minutes have more data than when sampled at one hour ( $12 \times$  times) and, if the models use this additional data, they should increase the result accuracy. In this section, to test this hypothesis, we design an experiment with an MLP that predicts 144 steps, which is the equivalent to twelve hours ( $144 \times 5m = 12h$ ).

The forecasting is made by an MLP with the MIMO structure (see Section 7.4), the parameters used in this architecture are described in Table 8.7)

With these parameters, the results obtain a jump in the accuracy, as the best accuracy obtained with a similar architecture with series at 1h is 7.254 (Test) and 6.953 (Validation).

For illustration purposes, we average the output hour per hour over the prediction horizon, and we generated a boxplot to compare the distributions between two methods, one applied to series sampled at one hour and the other a series with a period of five minutes (see Figure 8.3). In this boxplot We observe that the effect of the higher frequency series is relevant for the first steps reducing its impact as the sequence increase, this fact tells us that

## Going Deeper with wind time series forecasting architectures

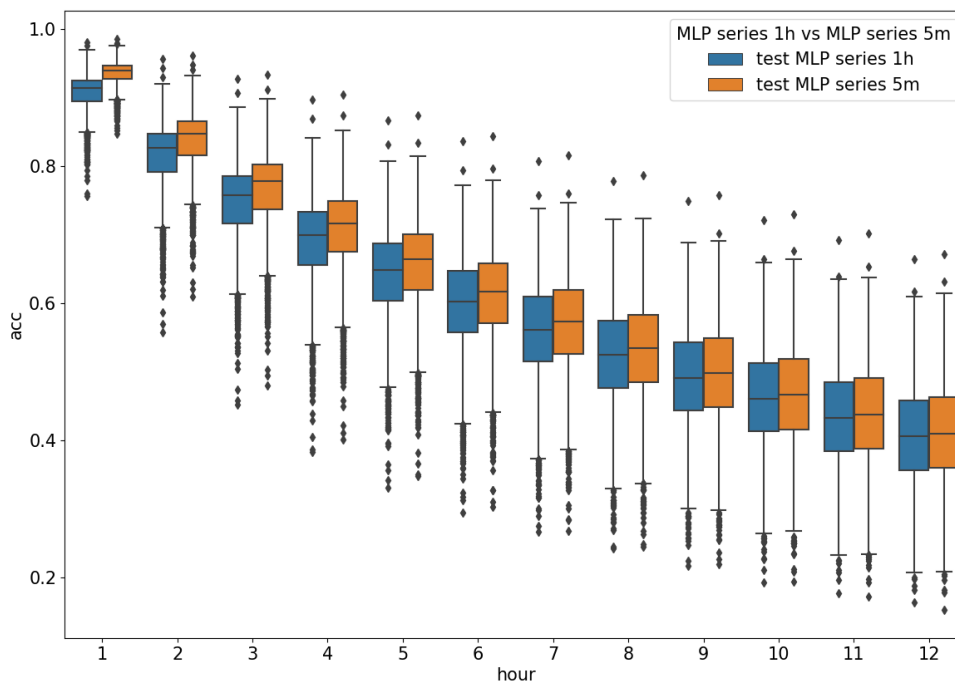
**Table 8.7** Architectures with series sampled at five min forecast at 12h

Architecture	Description
MLP 5m 12h	Architecture based on the basic MLP and predicts 144 steps aggregated at 1h (12 hours ahead). It is a two-layer network with 512 and 256 neurons, with drop connections (0.3) and uses a Leaky activation with $\alpha = 0.3$ . The <i>lag</i> is 144 steps (12) hours for the inputs, and the prediction is for 144 steps ahead (12 hours)

**Table 8.8** Mean and standard deviation of probability distributions for Series at 5m

Architecture	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
MLP 5m 12h	7.409	$\pm 0.775$	7.095	$\pm 0.724$

the increase of information, when we use a shorter period series, is more relevant for the first hours to predict than for the last ones.



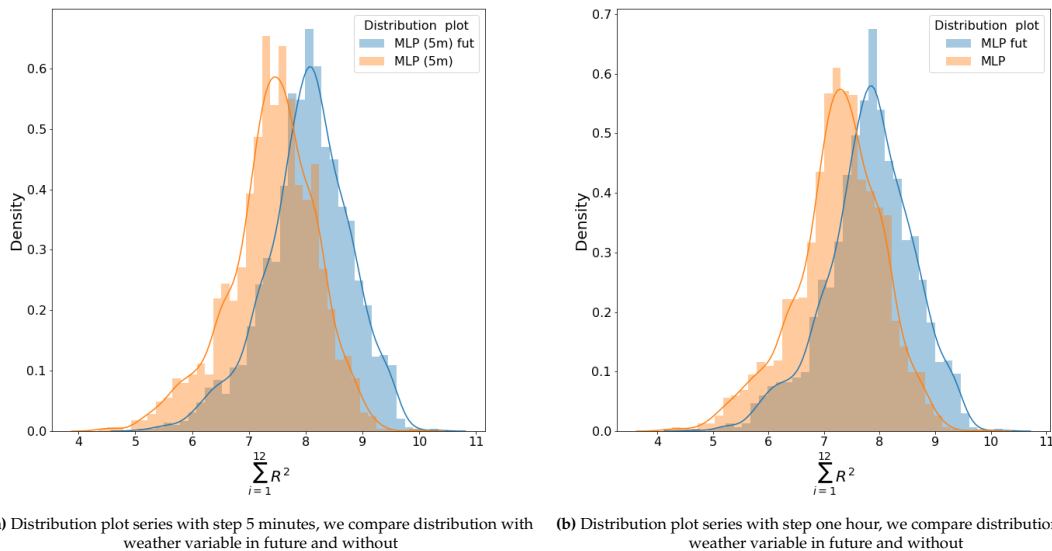
**Fig. 8.3** Boxplot comparing the hourly distribution of  $R^2$  error results between the MLP of series at 5 minutes and MLP of series averaged at 1h

## 8.5 How using a low-resolution weather variable increases accuracy over time series of different frequencies

In commercial applications of wind energy prediction, the models use weather forecast measures from weather services that provide this information. In these models, the accuracy depends on the reliability of the Numerical Weather Prediction (NWT) model.

In Weather forecasting, the uncertainty of variable type differs, for instance, it is easier to predict temperature than to predict wind speed, as the temperature is more stable over a large area, wind, is much more difficult to forecast as it depends on many local features. Accurate weather models require tiny grids (also defined as high-resolution grids) to model weather with enough granularity to describe the local features. Variables that can be forecast with larger matrixes are *low resolution variables*, as they are not impacted by local features as much as wind, for example, which depends on tiny terrain features.

In this approach, we use temperature as a low-resolution variable. Temperature is more accessible to forecast than wind, and, by adding it to the models, we can increase the accuracy of the wind production. There are experiences in the literature that use low-quality weather measurements to increase the accuracy of a wind speed prediction model [167]. With this



**Fig. 8.4** Comparison of distribution plots MLP with and without low resolution variable in 5m step series (left) and 1h step series (right)

architecture, the improvement is significant, showing an almost 10% increase in the overall accuracy (see Table 8.9). With this information, we can conclude that the use of exogenous information from the time series, even if this information is low resolution, increases the accuracy of the result.

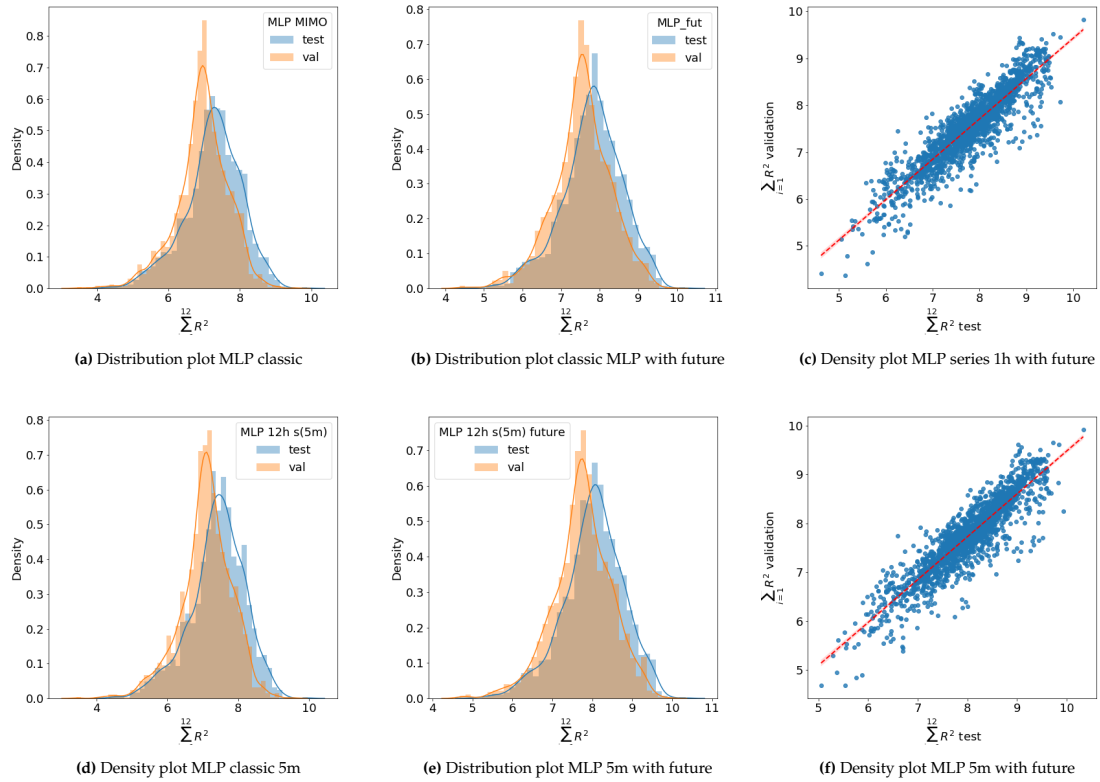


Fig. 8.5 Density and Distribution Series with future weather variable and without

Table 8.9 Comparison MLP architectures using future weather variable

Architecture	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
MLP MIMO	7.254	$\pm 0.792$	6.954	$\pm 0.732$
MLP MIMO future	7.806	$\pm 0.785$	7.541	$\pm 0.749$
MLP 5m MIMO	7.409	$\pm 0.775$	7.095	$\pm 0.724$
MLP 5m MIMO future	8.026	$\pm 0.754$	7.748	$\pm 0.733$

Performing a boxplot comparison in series at five minutes between with or without future we obtain a comparison where it can be observed an improvement of accuracy on all the twelve steps for both, series at one hour and series at five minutes (see Figure 8.6).

## 8.5 Using a low-resolution weather variable

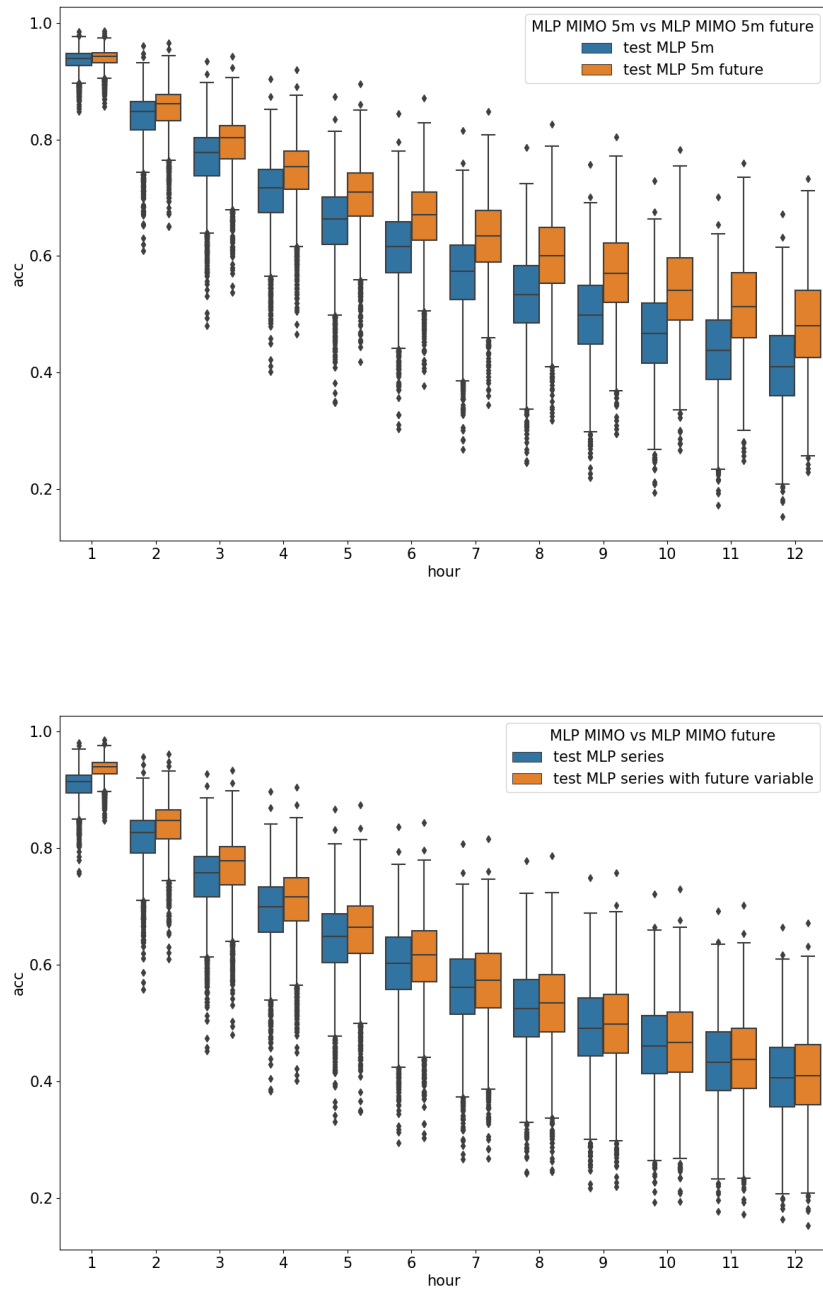


Fig. 8.6 Top Boxplot shows the comparison between series at five minutes with and without future weather variable. Bottom Boxplot compares results for series at 1h.

## 8.6 Can the MLP architecture obtain better accuracy than persistence for prediction one hour ahead?

This thesis designs and performs all the experimentation on a Horizon of twelve steps for the one-hour series. However, we want to validate the MLP method for predictions made with series with a period of 5m and one hour ahead prediction.

In the wind prediction field using time series, one hour ahead can be considered as a short term prediction, and, in principle, the accuracy of the MLP should improve the persistence. We designed this experiment to test if these deep learning algorithms can outperform persistence on this prediction horizon.

First, we need to calculate the persistence method for this series as well. We can see the architectures tested in Table 8.10.

**Table 8.10** Architectures with series sampled at 5 min

Architecture	Description
Persistence 5m 1h	This persistence shows the results when applied to a 1 hour ahead prediction when done with series at 5m.
MLP 5m 1h	This architecture is a MLP with same conditions as the MLP basic structure but with series sampled 5 minutes and predicts one hour ahead (12 steps)

We perform a comparison with the persistence  $R^2$  accuracy, that for this model is 10.946, obtaining a result with the MLP slightly better with a 2% improvement over persistence. The improvement is nonetheless relevant aligned with the expected wind short term prediction accuracy, as we validated with previous works like [73, 219].

It is important to note that the architectures used are like the ones used for twelve hours ahead forecasting. It is possible that by a complete exploration of the architectures-parameters for the higher frequency series, the accuracy value improves even more. [78]

**Table 8.11**

Mean and standard deviation of distributions from the prediction one hour ahead using series with five minutes period

Architecture	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
Persistence 5m 1h	10.946	$\pm 0.344$	10.749	$\pm 0.436$
MLP 5m 1h	11.123	$\pm 0.267$	10.951	$\pm 0.347$



## 8.7 Discussion

In this chapter, we verified some hypothesis and complemented some architectures initially described in Chapter 7. The initial experiments, based on vanilla architectures, give an idea of the accuracy levels of the different models. We completed those basic structures in this second phase of research, and in this Section, we want to discuss the obtained findings. We structure this discussion in sections by families of architectures.

### MLP architectures

We showed how the skip connections add stability to the network, with significant improvements when the network is deeper. Skip connections have a positive impact on the vanishing gradient issues, and it shows in the increase of stability in four or five-layer networks (see Table 8.3).

The best performing networks still is the two-layer model, but architectures with three layers are a very close second. We believe that the use of skip connections is convenient for deeper networks; nevertheless, for this wind time series depth at two-layers is enough.

We wanted to go deeper into the comparison between regression models. In Section 7.3, we identified the MIMO approach as the best regression model, compared with a multiple-regression (12) regressions approaches, but we want to know if partial regression improves the results. For this reason, we designed the SJOINT architecture that allows variable lengths of regression in the model.

The results obtain an exciting conclusion, as the SJOINT approach with regressions for every 3-4 hours is better than the MIMO approach, based on a single regression. The 3-4 hours have a physical interpretation as from the wind formation theory [117], we know that wind changes patterns in periods of three to four hours, a conclusion can be applied in wind forecasting to refine the best predictions, but at the cost of increasing the computing resource consumption significantly.

### RNN networks

The RNN experimentation tried to explain the differences between alternative GRU and LSTM cells. The differences are motivated by the nature of the wind data and the forecasting model. Firstly, the wind data has a short *lag* which do not require the long term memory approach offered by the LSTM cells, this fact added to the prediction structure (fixed input length and fixed prediction) reduces the complexity, prioritise the less complex GRU cells over the LSTM.

### Changing the time series frequency

We analyse architectures working with the same data but, sampled with higher frequency, five minutes in this case (five minutes is the initial step on the data). Just by using the series at five minutes, we obtain an increase of the accuracy, as the series now contain more information which can be used by the learning algorithms. We obtained this conclusion with MLP models, but in Section 9.6 we perform a similar experiment with CNN network. We can affirm that using series with more information increases the accuracy of the result.

### Short term prediction

With the series sampled at 5-minute steps, we performed a forecasting experiment to model shorter periods, in this case, one hour ahead. In the experiments with a 12-hour prediction window, persistence obtains poor results, but at one hour ahead, the accuracy of persistence is remarkable. However, with an MLP model, we improve the accuracy of the persistence method by 1.5%.

The MLP architecture used is like the ones applied to 12 hours prediction. The creation of optimised architectures for short term prediction using deep learning is a future area of research that we address in 12.2.

### Introducing weather information

Some weather variables are easier to predict (and with less uncertainty) than others, using just a temperature prediction in the learning algorithm, the results improve accuracy significantly, opening the door for experimentation with hybrid models with Weather prediction and time series.

## Chapter 9

# The effectiveness of CNN architectures for wind time series

Deep learning Convolutional Neural Networks (CNN) are present in many applications. Its versatility lies in the reduction of the size of the parameters to train (compared to fully connected networks) while maintaining or improving the feature representation capabilities by the application of convolutional and pooling operations. This chapter goes in-depth on the use of convolutional networks applied to multi-step forecasting.

The separable convolutional layer has shown the best performance on the wind time series, while other models with several variants have shown excellent performance but with some added computing costs.

Convolutional networks used for MIMO transformations (Multiple Input Multiple Output) are the right candidate for wind time series.

This chapter starts with an analysis of the different architectures designed and then, later, analyses the complete experimentation performed with each one of them. The chapter finalises with a summary of the results and some preliminary conclusions.

This chapter is generating a submission to a journal.

## 9.1 Experiments with Convolutional Networks for time series forecasting

In this chapter, we analyse specifically convolutional architectures. In previous experiments in Section 7.4, we identified the potential of the convolution operation for wind time series representation. In the experiments, the classic CNN architecture was not the best performing, but in some preliminary tests, by deepening into the hyperparameter search, we found out that the CNN family of architectures has more potential to improve the accuracy. With this hypothesis, we decided to analyse the convolution operation from diverse angles to release all the potential of these models, in this hunt of the best forecasting approach for wind time series.

In this chapter, we design several convolutional architectures enumerated in Table 9.1, with a different design, features intending to find the best performing on the full wind dataset. For the readers' convenience, we place a summary table of this chapter results in Table A.2 and, for comparison purpose, there is an Appendix A which gathers all the experiments results from the different experimentation in this thesis.

**Table 9.1** CNN Architectures designed for the experimentation

Model	Description
Random Forest	MIMO Baseline
CNN Classic	CNN 1,2,3,4,5,6,7,8 layers
CNN Separable	CNN sep 1,2,3,4,5,6,7,8 layers
CNN Skip	CNN skip 1,2,3,4,5,6,7,8 layers
CNN Residual	CNN res 1,2,3,4,5,6,7,8 layers
CNN Multi Head	2,3,4 Parallel heads
CNN GB	Gradient boosting for sep and Classic
CNN 5m	Classic CNN with series at 5 min
CNN sep 5m	CNN sep with series at 5 min

## 9.2 A CNN MIMO approach

In Sections 7.3 and 2.4.3 we analysed the main MIMO approaches. In those sections, we concluded that the MIMO approach offers better accuracy using fewer resources, and we consider it as the right structure for a sequence to sequence learning with wind time series. For this reason, we decided to implement a MIMO architecture for CNN networks.

The CNN MIMO implementation consists in using two blocks. The first block, based on convolutions, performs the feature extraction, and the second block based on an MLP, performs the regression that, using the tensor from the output of the first block with the features extracted, uses it to regress the final forecast sequence (see Figure 9.1).

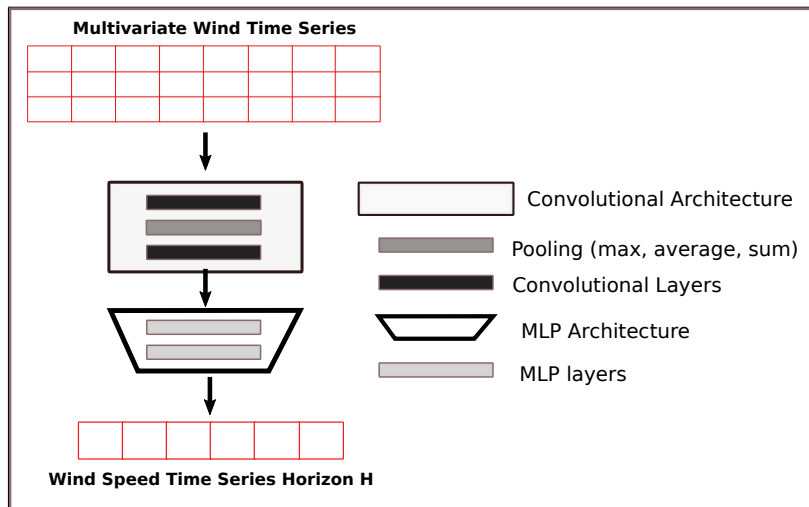


Fig. 9.1 MIMO approach to prediction using a convolutional architecture plus an MLP Network

The multivariate wind time series is the input for the CNN block, build up from a combination of primary elements, like one or many convolutional layers, with or without pooling, dropping connections or adding batch normalisation.

The combination of one or several CNN blocks with an MLP at the exit is not a novel approach and already present in most architectures from the Imagenet Challenges [114, 196, 84, 205, 52, 183]. However, the applications for time series are more limited. A relevant characteristic of the CNN architectures designed in this work is the use of a 1-dimensional convolution. This convolution operates the kernels only in one axis, and not bi-dimensionally like the standard 2D convolution used in most applications. 1-dimensional convolutions can work with one-dimensional or two-dimensional matrixes.

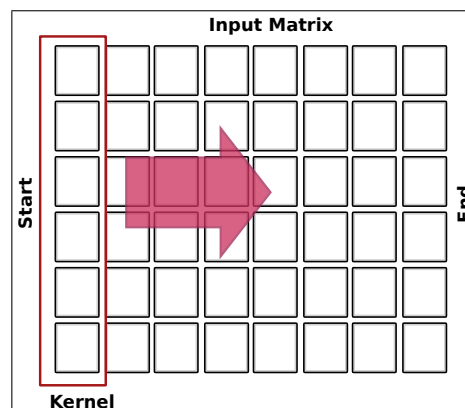


Fig. 9.2 One dimensional convolution

### 9.3 Baseline Methods

Usually, in wind prediction experiments, the persistence method is used as a baseline. Persistence consists in using the last time step  $x_t$  as the prediction for each one of the  $H$  future time steps  $(x_{t+1}, \dots, x_{t+H})$ , for short term predictions, is accurate only for short time predictions, and for 12 hours prediction its error is too high to be used as a comparison.

**Table 9.2** Baseline Methods

Architecture	Description
Persistence	Persistence consists in using the real wind speed for time step $t$ as prediction for all the time steps from 1 to 12. It has been discarded as a baseline method due to its low accuracy.
RF	Random Forest (RF). Ensemble of Trees, using bootstrapping with trees generated recursively. All the features are used on each split and the number of trees generated is limited to 400.

For this reason, we chose Random Forest MIMO as a baseline method. Random forest is used in wind prediction successfully [222], and we considered it the right benchmark candidate.

Random Forest is an ensemble method based on decision trees with bootstrapping. The training time-series sequences are transformed into a set of examples  $E^*$ , and from this set, a random new subset  $Z^*$  is constructed using the bootstrap technique with size  $N$ . Bootstrapping consists of generating sets of samples from an initial set randomly with replacement. Then decision trees are built recursively with the bootstrapped set of data  $Z^*$ , generating an ensemble of trees  $\{T_b\}_1^N$ , with this approach the regression then is:

$$\hat{y} = (\hat{f})_{rf}^N(x) = \frac{1}{N} \sum_{b=1}^N T_b(x) \tag{9.1}$$

The bias of the forest usually is higher than the one of a simple tree, but due to the averaging of the bootstrapped data, its variance decreases, hence generating a better learner than one using individual trees. We obtain the best results with 400 trees, and no limit for tree depth with an unlimited number of examples per node.

We apply the RF to the 126,692 sites. Table 9.3 compares the baseline results.

**Table 9.3** Mean and standard deviation of probability distributions for baseline architectures (T): Test data, (V): Validation data, (Dev): Deviation, (Mean): Mean value

Model	Mean-T	Dev-T	Mean-V	Dev-V
Persistence	2.699	±1.951	2.486	±1.919
RF	6.770	±0.805	6.532	±0.737

## 9.4 Parameter space of the convolutional architectures

In this section, we describe the parameters used in the convolutional layers, which control the behaviour of the models and require thoughtful experimentation to obtain the best fitting strategy.

Firstly, in all the models, we need to select the length *lag* that defines the length of the input matrixes for training. We confirmed experimentally that this length influences the results, and for this reason, we include it in the parameter list to optimise. We have obtained as a result, not very high values (six to eighteen hours), a finding that is aligned with the nature of wind formation as it is described in the literature [73, 117]. From this value, we can infer that the wind now does not influence the future wind 18 hours ahead into the future.

We use two different kinds of convolutional layers, the 1-dimensional classic or vanilla layer and the 1-dimensional separable layer. Separable convolutions perform the convolution in two steps (see Section 3.5.2. We have not defined hybrid models with both convolutional layers; our architectures are either separable (for all layers) or classic (for all layers).

The convolutional layers have multiple parameters that influence the inner operation mechanism. The first parameter is the *kernel\_Size*. A kernel is a filter that is applied by the convolution to the input matrix, and this parameter fixes its length; the next parameter is *filters*, which is an integer that defines the dimensionality of the output (number of filters). The kernel moves in steps that jump in *strides*, and the parameter *strides* defines how the kernel moves in the convolution.

*Padding* is another critical parameter (see Section 3.5.1) that controls how the convolution behaves near the edges. The preference has been to use causal padding as this strategy introduces time-sensitive knowledge into the convolution being more efficient when used to predict with sequences.

Depth Multiplier is another parameter which multiplies the channels after the convolution, impacting the behaviour of the network, we have observed that has a material impact in the accuracy obtained by the architectures.

The activation function introduces non-linearity in the operation, and we have identified the Exponential Linear (ELU) and the Leaky Rectified linear functions (leaky ReLU) as the better performing. Both activations use an  $\alpha$  parameter that shapes the function. The best  $\alpha$ , obtained from the different hyper-parameter optimisations, is usually a value between 0.2 and 0.4.

ELU and leaky ReLU avoid the zero-dying issue [239] when there are numbers close to zero or negatives. As the data has been z-normalised, there are negative values, which are better captured by these two activation functions.

*Dropout* is another parameter associated with the layer connections. This parameter has a significant impact on the model accuracy, and we apply it to the CNN layers, to the MLP layers, or both.

We find differences between a non-optimised architecture with the best architecture of over 10%-15% of the resulting error. This percentage is a significant value that shows how important it is to find the best parameters for a model.

However, for each experiment, there is not only a single combination with the best result, but many parameters that generate results very close to the best, sometimes with minimal differences. In the results table (see Table A.2), we present the best result for some architectures, but we need to point out that there is not a single *perfect* parameter combination as we obtain many quasi-optimal models, where the accuracy is just some hundredths from the best result

## 9.5 Convolutional architectures

This section is devoted to experimentation. We describe the different architecture designs, parameters and their results. We start with the basic models, and we add different modifications to the original models. We have structured this section in subsections, one, for each group of architectures.

### 9.5.1 Classic Convolutional and separable layers

All the convolutions in the different models are 1D-convolutions. Each defined architecture has a parameter space that defines the behaviour of the architecture (described in Section 9.4). The number of possible parameter combinations has required the use of a hyperparameter searching strategy described in Section 6.4. This strategy is relevant due to the high sensibility of the architecture results in minor changes in parameters. We estimated a 10% to 15% variation between a vanilla model and its optimised version, which tells the need for a structured approach to parameter setting

The parameter space already described (see Section 9.4) has been used to obtain the most accurate network. Possibly the most relevant exploration parameter is the number of layers or network depth. In the experiments, we defined architectures from one to eight layers.

Table 9.4 CNN Skip and Residual Architectures overview

Architecture	Description
Classic Convolutional	Persistence consists in using the real wind speed for time step $t$ as prediction for all the time steps from 1 to 12. It has been discarded as a baseline method due to low accuracy.
Separable Convolutional	These architectures use separable 1 dimensional architectures

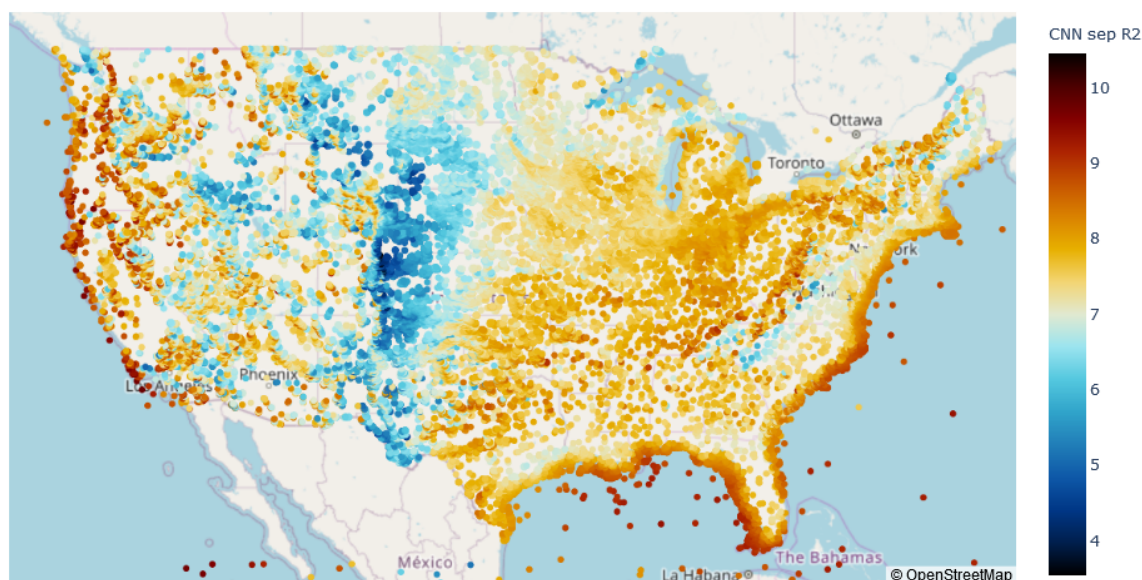


**Table 9.5** Parameters  $\theta$  for each best model (Classic CNN and Separable models from one to three layers)

Model	$\theta$	CNN-1L	CNN-2L	CNN-3L	CNN-sep-1L	CNN-sep-2L	CNN-sep-3L
CNN	lag	6	<b>12</b>	6	12	<b>12</b>	18
	kernel size	[3]	<b>[3,1]</b>	[3,1,1]	[9]	<b>[9,1]</b>	[5,3,3]
	filters	[128]	<b>[128,128]</b>	[512]	[512]	<b>[1024,1024]</b>	[512,1024,1024]
	strides	[1]	<b>[1,1]</b>	[1,1,1]	[3]	<b>[2,4]</b>	[3,4,4]
	activation	[elu,0.4]	<b>[elu,0.4]</b>	[elu,0.4]	[elu,0.3]	<b>[(elu,0.3),(elu,0.4)]</b>	[elu,0.4]
	dropouts	[0.3]	<b>[0.3,0.2]</b>	[0.2,0.6,0.5]	[0.4]	<b>[0.6,0.6]</b>	[0.6,0.6,0.7]
	depth mul.	[1]	<b>[1,1]</b>	[1,1,1]	[5]	<b>[8,7]</b>	[4,7,4]
MLP	layers	[512]	<b>[512]</b>	[512]	[1024]	<b>[1024]</b>	[4096]
	dropout	[0.2]	<b>[0.3]</b>	[0.1]	[0.4]	<b>[0.1]</b>	[0.0]
	activation	[leaky,0.2]	<b>[leaky,0.2]</b>	[elu,0.3]	[leaky,0.2]	<b>[elu,0.4]</b>	[elu,0.3]
	$\Sigma(R^2)$	[7.219]	<b>[7.226]</b>	[7.222]	[7.301]	<b>[7.321]</b>	[7.302]

We build the classic convolutional models with one-dimensional convolutions. We have implemented six architectures with a depth of one to eight sequential layers. The results comparing the accuracy of the models with different depths is in Figure 9.7.

The most relevant parameters in these architectures are *filters* and *strides* which define the convolution, *depth multiplier* which modifies the output channels, *drop* for each channel, and the architecture of the MLP component, number of layers, number of neurons and drop. The number of parameters on each architecture is very high and increases along with the number of layers used. We show in Table 9.5 a comparison between relevant parameters  $\theta$  for the three main classic architectures and the three main separable ones, in bold the two best performing ones.

**Fig. 9.3** Accuracy of convolutional separable networks applied to the NREL dataset - blue less accurate, dark red more accurate

Separable models (see Section 3.5.2) perform the convolution in a two-step process. They reduce some computational cost and have shown the best results for the experiments.

For both architectures, separable and classic, we can observe that including dropout the accuracy increases, the dropout values vary, and the best usually are between 0.3 and 0.5. We learnt through these experiments the importance of dropout in training these architectures, adding dropout to the hyper-parameter search, allowed to improve the overall accuracy of the networks significantly. Dropout improves the generalisation of the network by randomly dropping connections during the training. This feature is considered a normalisation feature. Another normalisation that has revealed not to be as essential and not used for parameter setting is the kernel regularisation.

For filters and kernels length, the best values usually decrease when the layer number increases, for the initial layers, the filters are bigger than for the deeper ones.

We use strides one in the classic convolutional network, while for the separable, strides used are three or four which offer better results.

Regarding the activation functions, the Elu and Leaky units are superior to other functions. Altogether the best models have two layers, in both; classic and separable architectures. In Table 9.5, we present a complete analysis of the best parameters.

Finally, some parameters may seem complementary, but when we push for the best result, become essential. *Depth Multiplier* is vital for the separable convolutions, as this parameter multiplies the output channels in the convolution, increasing its dimension, its effectiveness is another contribution to the final optimal result.

The architecture with two separable convolutional layers is the best performing from all the different experiments, and comparing it with a baseline random forest model, shows better accuracy for each one of the 12 predicted steps as we illustrate in Figure 9.8. This hourly (step) comparison shows the superiority of the separable model over the baseline, with an accuracy that increases at every step, obtaining the most significant difference at the 12<sup>th</sup> step.

The *best* architectures have two and three layers, but its accuracy deteriorates fast with the addition of more layers as we illustrated in Table 9.6 that shows the results of classic and separable architectures for comparison purposes.

The best architectures are not very deep, and this finding is common to all the deep learning architectures designed in this work. In principle, deeper should obtain better results than shallow, but the consistency of this behaviour shows that the wind data series available do not have the pattern complexity that will require deeper series. State of the art (see Section 4.3) confirms this point, as the published experiences in the literature conclude that deep networks for wind prediction are shallow.

We can observe in Figure 9.3 how the accuracy results follow geographical areas. The map is divided clearly between areas of high wind stability, the Pacific coastline and the Gulf of Mexico, an area of very low accuracy that spreads over the plains from Canada to

Mexico and areas of with mixed high and low accuracy results showing complexity located in Idaho, Oregon and Colorado. This map is telling us that the wind characteristics in a site define how difficult is the forecasting of the wind speed for that location.

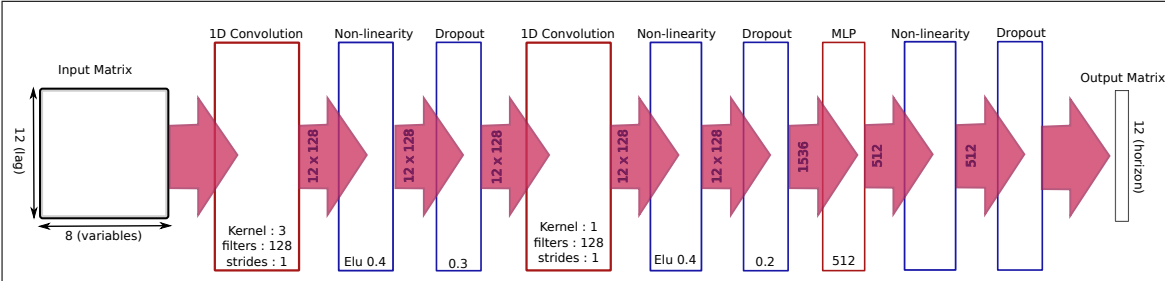


Fig. 9.4 Layers and shapes of best CNN Architecture

In Figure 9.4, we can see all the different operations sequentially applied to the input matrix to obtain the prediction. The different chained operations give us an idea of the shape transformations and different operations that are applied to each step. Even considering them as not very deep they have high complexity with a sizeable number of weight connections to learn, for example on this architecture the number of weights (parameters) to train is 812,812, and for some of the models developed in this chapter, the number of trainable elements is well over a million.

The better behaviour of the separable layers is an interesting finding. Chollet in the article where he defined this convolution operation [34] already pointed this better capability of separable models for image recognition applications. Separable layers combine efficiency in their training, due to a less computationally complex operation (see Section 3.5.2), with a higher representation capability for some applications. Our contribution lies in this finding. We can affirm that for the wind forecasting problem, separable one-dimensional convolutions show better accuracy over classic convolutions tested with diverse wind sites and with several architecture structures (number of layers, input matrixes size, parameters)

9.5.2 Adding Skip and residual connections

Skip, and residual connections are used in convolutional and MLP networks. These connections have as an objective to avoid the vanishing gradient issue, a problem that happens training the network, when the weights in a connection get very close to zero losing the capability to transmit information to the connected layer or cell, the skip or residual connections bring the input information to the intermediate layer. With skip connections, the layer output is concatenated with the network input, and the residual connections instead of concatenation they us the addition operation with the sequences.

## The effectiveness of CNN architectures for wind time series

**Table 9.6** Mean and standard deviation of probability distributions for classic and separable architectures (T): Test data, (V): Validation data, (Dev): Deviation, (Mean): Mean  
The best architectures are in bold typeface

Model	Mean-T	Dev-T	Mean-V	Dev-V
RF	6.770	$\pm 0.805$	6.532	$\pm 0.737$
CNN - 1l	7.185	$\pm 0.802$	6.867	$\pm 0.754$
<b>CNN - 2l</b>	<b>7.226</b>	<b><math>\pm 0.800</math></b>	<b>6.908</b>	<b><math>\pm 0.736</math></b>
CNN - 3l	7.222	$\pm 0.801$	6.895	$\pm 0.734$
CNN - 4l	7.190	$\pm 0.804$	6.858	$\pm 0.733$
CNN - 5l	7.152	$\pm 0.809$	6.816	$\pm 0.735$
CNN - 6l	7.106	$\pm 0.809$	6.771	$\pm 0.733$
CNN-sep 1l	7.301	$\pm 0.798$	6.991	$\pm 0.752$
<b>CNN-sep 2l</b>	<b>7.321</b>	<b><math>\pm 0.797</math></b>	<b>6.988</b>	<b><math>\pm 0.745</math></b>
CNN-sep 3l	7.302	$\pm 0.796$	6.988	$\pm 0.746$
CNN-sep 4l	7.270	$\pm 0.798$	6.954	$\pm 0.754$
CNN-sep 5l	7.156	$\pm 0.811$	6.848	$\pm 0.736$
CNN-sep 6l	7.116	$\pm 0.809$	6.806	$\pm 0.736$

We introduced skip and residual connections to classic convolutional models and tested them with one to eight layers. The skip and residual connections link the layers sequentially (see Figure 3.9),

The results are slightly worse than the ones obtained with classic and separable convolutional models, but looking at the results we can observe (see Table 9.7) that these additional connections bring stability to the networks as they show better results than the rest of architectures from five layers onward.

The results from the depth data and the experiment results are in Tables 9.8 and 9.7, where we can see the better stability of the skip and residuals connections. In this table, the models with more layers perform better with the skip and residual architectures, showing

**Table 9.7** Layer depth comparison between models (error calculated on Test Dataset)

Layers	CNN	CNN-sep	CNN-skip	CNN-res
1L	7.185	7.301	6.935	6.862
2L	7.226	7.321	7.223	7.136
3L	7.222	7.302	7.221	7.134
4L	7.190	7.270	7.201	7.118
5L	7.152	7.156	7.183	7.095
6L	7.106	7.116	7.157	7.076
7L	7.052	7.054	7.130	7.055
8L	5.450	5.448	7.090	7.036

the effect of the added connection. Another finding is that skip connections behave better than Residuals.

We believe that for shorter frequency time series or series with more variables, with more data, the best architectures would be deeper, and in this case, the best architectures will require this kind of connections.

**Table 9.8** Model Experimentation  $R^2$  results for skip and residual architectures, (T): Test data, (V): Validation data, (Dev): Deviation, (Mean): Mean value  
The best architectures are in bold typeface

Model	Mean-T	Dev-T	Mean-V	Dev-V
CNN-skip 1l	6.935	$\pm 0.799$	6.671	$\pm 0.730$
<b>CNN-skip 2l</b>	<b>7.223</b>	$\pm 0.798$	<b>6.904</b>	$\pm 0.738$
CNN-skip 3l	7.221	$\pm 0.798$	6.903	$\pm 0.738$
CNN-skip 4l	7.201	$\pm 0.806$	6.871	$\pm 0.739$
CNN-skip 5l	7.183	$\pm 0.810$	6.850	$\pm 0.738$
CNN-skip 6l	7.157	$\pm 0.813$	6.820	$\pm 0.739$
CNN-skip 7l	7.130	$\pm 0.814$	6.790	$\pm 0.739$
CNN-skip 8l	7.090	$\pm 0.818$	6.751	$\pm 0.739$
CNN-res 1l	6.862	$\pm 0.796$	6.602	$\pm 0.735$
<b>CNN-res 2l</b>	<b>7.136</b>	$\pm 0.806$	<b>6.834</b>	$\pm 0.739$
CNN-res 3l	7.134	$\pm 0.805$	6.832	$\pm 0.739$
CNN-res 4l	7.118	$\pm 0.804$	6.807	$\pm 0.733$
CNN-res 5l	7.095	$\pm 0.807$	6.785	$\pm 0.736$
CNN-res 6l	7.076	$\pm 0.806$	6.767	$\pm 0.734$
CNN-res 7l	7.055	$\pm 0.810$	6.746	$\pm 0.737$
CNN-res 8l	7.036	$\pm 0.809$	6.728	$\pm 0.739$

### 9.5.3 Multi-head architectures

Multi-head architectures (see Figure 3.10) consist of combining several convolutional layers in parallel and, after the convolutions, combining the output. We have tested three models with parallel heads, with two, three and four heads.

Adding parallel convolutional layers, we obtain a robust architecture. These multi-head constructions are widely used in the sophisticated Imagenet recognition approaches, and we wanted to experiment with this model applied to wind time series.

The results (see Figure 9.9) show as the best results the multi-head architecture with three heads. Again, the results are a bit short from those of the sequential classic or separable architectures, but not very far away.

These results seem to confirm that with wind time series, the most complex architectures do not outperform simpler ones; however, the three-headed model shows superior ability to predict wind over four and two heads.

## The effectiveness of CNN architectures for wind time series

**Table 9.9** Model Experimentation  $R^2$  results for multi-head architectures, (T): Test data, (V): Validation data, (Dev): Deviation, (Mean): Mean value  
The best architectures are in bold typeface

Model	Mean-T	Dev-T	Mean-V	Dev-V
CNN-MH 2Heads	7.145	$\pm 0.792$	6.829	$\pm 0.731$
<b>CNN-MH 3Heads</b>	<b>7.170</b>	$\pm 0.796$	<b>6.854</b>	$\pm 0.735$
CNN-MH 4Heads	7.125	$\pm 0.788$	6.816	$\pm 0.728$

### 9.5.4 Adding Gradient Boosting to a CNN Architecture

Gradient Boosting is a machine learning technique, initially defined by [Friedman](#) as early as 1999 in [61]. We introduced this ensemble method in Section 4.2.7. Gradient Boosting has come back thanks to the actual popularity of XGBoosting (defined recently by [Chen and Guestrin](#) in [31]) an algorithm based on the refinement of the original that has offers excellent performance with structured data.

The main idea in this algorithm consists of training sequentially different models on the residual value that is left by the previous iterations (see 4.1).

Our model is a pseudo implementation of gradient boosting. It is an ensemble model that constructs the result by using a linear combination of predictions on re-weighted data. In the canonical implementation, the weights for each step are calculated by an optimisation algorithm (gradient descent or similar).

$$\hat{Y}(x) = \sum_{i=1}^n \gamma_i h_i(x) + C \quad (9.2)$$

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, Y_{m-1}(x_i)) - \gamma \nabla_{Y_{m-1}} L(y_i, Y_{m-1}(x_i)) \quad (9.3)$$

The weights  $\gamma$  are calculated with the equation 9.3. For our problem, we simplified the calculation of the weight by simulating a list of successive values by using two parameters,  $\alpha$  and *decay*. The weights sequence starts at 1 one, and the successive values are generated as follows.

$$w_1 = 1 \quad (9.4)$$

$$w_2 = \alpha \cdot \text{decay} \quad (9.5)$$

$$w_3 = w_2 * \cdot \text{decay} \quad (9.6)$$

$$w_4 = \dots \quad (9.7)$$

With  $\alpha = 0.3$  and  $decay = 0.9$  the weights sequence, for  $n = 5$  is  $w = (1, 0.27, 0.243, 0.219, 0.197)$ .

We use this architecture with the classic convolution and with the separable convolution, both with two layers. In the hyper-parameter search process, we iterated over the values of  $\alpha$  and  $decay$  to find out the best combination of values, and after the search, we obtained the best behaviour of the architecture with  $\alpha = 0.3$  and  $decay = 0.9$ .

The results of this architecture were outstanding, as obtains the best method performance overall with the time series at one hour (see a summary of results in Appendix A.2). This model generalises adequately and beats all the models on the test and the validation dataset. (See Table 9.10).

**Table 9.10** Model Experimentation  $R^2$  results for CNN architectures with gradient boosting, (T): Test data, (V): Validation data, (Dev): Deviation, (Mean): Mean value

Model	Mean-T	Dev-T	Mean-V	Dev-V
CNN - 2l GB	7.232	$\pm 0.794$	6.959	$\pm 0.731$
CNN sep 2L GB	7.341	$\pm 0.796$	7.043	$\pm 0.753$

## 9.6 Using CNN in series with a 5 minute step

In Section 8.4, we describe several experiments performed with series sampled at a higher frequency. Initially we decided to average the data hourly in the experimentation (see Section 6.2.1, but after performing the main experimentation, we want to understand which effect can have in the accuracy to work with the original time series with 5 minute steps. We lay the hypothesis that averaging the series imply to lose some valuable information, thus making the algorithm to obtain inferior accuracy, in this way, if we use higher frequency series, the accuracy will improve. We designed a new set of experiments, this time using the series with their original period-step of 5 minutes. The immediate implication by using these series is a multiplication of the amount of input data by twelve.

To compare in a level field, we try similar architectures with the two sets of series. We know that with the series at 5m, deeper architectures can work better, at the expense of a more expensive training, but this is not the point to explore. We just want to understand if the similar architectures improve accuracy by working with more data, this being a classical trade-off in the Machine learning field [45].

We test four architectures, a Classic CNN and a CNN-separable, both of them with just one or two layers. We did not want to increase the depth of the architectures, but we increased some of the parameters, like *Kernel size* or *filters*, as these increases make the network to train smoothly with more data.

## The effectiveness of CNN architectures for wind time series

**Table 9.11** CNN Architectures with series with period 5 minutes

Architecture	Description
CNN 1 Layer	CNN with 1 classic convolutional layer. The kernel size is larger than the ones used with hourly averaged series. kernel size and stride have similar length of 1 and 64 filters
CNN 2 Layer	CNN with 2 classic convolutional layers. The main parameters $\theta$ are Kernel size = 17 in the first layer and kernel size = 1 in the second layer, strides are in the first layer = 9 and in the second layer = 1. Filters are 256 in first layer and 128 in second layer
CNN separable 1 Layer	This separable architecture uses Kernel size=17 and depth multiplier=9. Filters are 256 and strides are 9
CNN separable 2 Layers	In this case the 2 layer architecture has Kernel size [17,1], activations [[elu,0.4], [leaky,0.2]], depth multiplier [9,9], filters [256,128] and strides [9,1]

The experiments confirmed the hypothesis, as all the architectures obtained positive increases in accuracy (see Table 9.12). These results reveal that there is useful additional information in the series with shorter periods.

As usual, understanding better a problem means discovering new questions. By increasing the resolution of the data, we obtained better accuracy, therefore using data with shorter sampling steps will be even better. This opens a new exploration to understand what the optimal sample period is. Would series sampled at seconds be more valuable?

**Table 9.12** Model Experimentation  $R^2$  results for high frequency series (5minutes) architectures, (T): Test data, (V): Validation data, (Dev): Deviation, (Mean): Mean value  
The best architectures are marked in bold typeface

Model	Mean-T	Dev-T	Mean-V	Dev-V
CNN 1l series 5m	7.409	$\pm 0.775$	7.095	$\pm 0.724$
<b>CNN 2l series 5m</b>	<b>7.524</b>	<b><math>\pm 0.761</math></b>	<b>7.212</b>	<b><math>\pm 0.716</math></b>
CNN-sep 1l series 5m	7.524	$\pm 0.763$	7.212	$\pm 0.718$
CNN-sep 2l series 5m	7.507	$\pm 0.766$	7.211	$\pm 0.716$



## 9.7 Global and local error evaluation

We want to understand how good or bad is an architecture in a specific prediction, to obtain a single value we forecast, using a sliding window technique, the algorithm for two years, one year is the test data, and the other year the validation data. All the forecast errors are summarised as per time step, for a single experiment in a site we obtain twelve values, each one being the accuracy in  $R^2$  for the step. The result is obtained by adding the twelve values, rating the accuracy of the algorithm for a site.

The results of a whole experiment consist of a summarisation of all the individual results from all the sites. All the summarisations form a set of values that can be considered as the probability distribution of the errors. The final value (appearing in tables or results, in the different chapters and the Appendix A) is the mean value of the distribution.

With this methodology, we distil an experiment into a number, valid for comparison but unable to describe the intricacy and complexity of an experiment. To understand better this prediction error, we have designed a tool that allows understanding the accuracy of a deep learning architecture over a year. This representation, described in Section 7.5, is a useful tool for visually identifying patterns and strengths or weaknesses of a specific prediction method.

In Chapter 7, we used this representation to understand how sites differentiate between them, comparing them with other locations with similar accuracy.

The values for the representation are obtained by subtracting the real prediction from the real value. The Colour white represents a perfect prediction, while colour dark red is positive deviation over 15 m/s and colour dark blue negative deviation of -15 m/s. there is a colour bar below each plot that identifies the seasons and allows to compare seasonality if it appears.

In this section we use the representation to compare different methods on the same site. We can see the representation for three sites (Figures 9.5, 9.6 and 9.7).

We have chosen three sites that offer three different site typologies, the first one has an extensive range of values between the different methods, the second one has a minimal

**Table 9.13** Accuracy, mean and variance in strips representation for three sites

Measure/site	37917	31321	118728
CNN-sep-2L $R^2$ accuracy	7.268	9.615	7.130
CNN-1L $R^2$ accuracy	6.844	9.512	7.055
RF $R^2$ accuracy	6.364	9.380	6.282
$k$ -NN $R^2$ accuracy	5.208	8.648	5.208
Persistence $R^2$ accuracy	-2.635	7.987	2.520
Mean (m/s)	7.36	8.65	8.62
Variance	15.69	7.987	15.42

variation, and the third one has the results with accuracies evenly distributed between the methods.

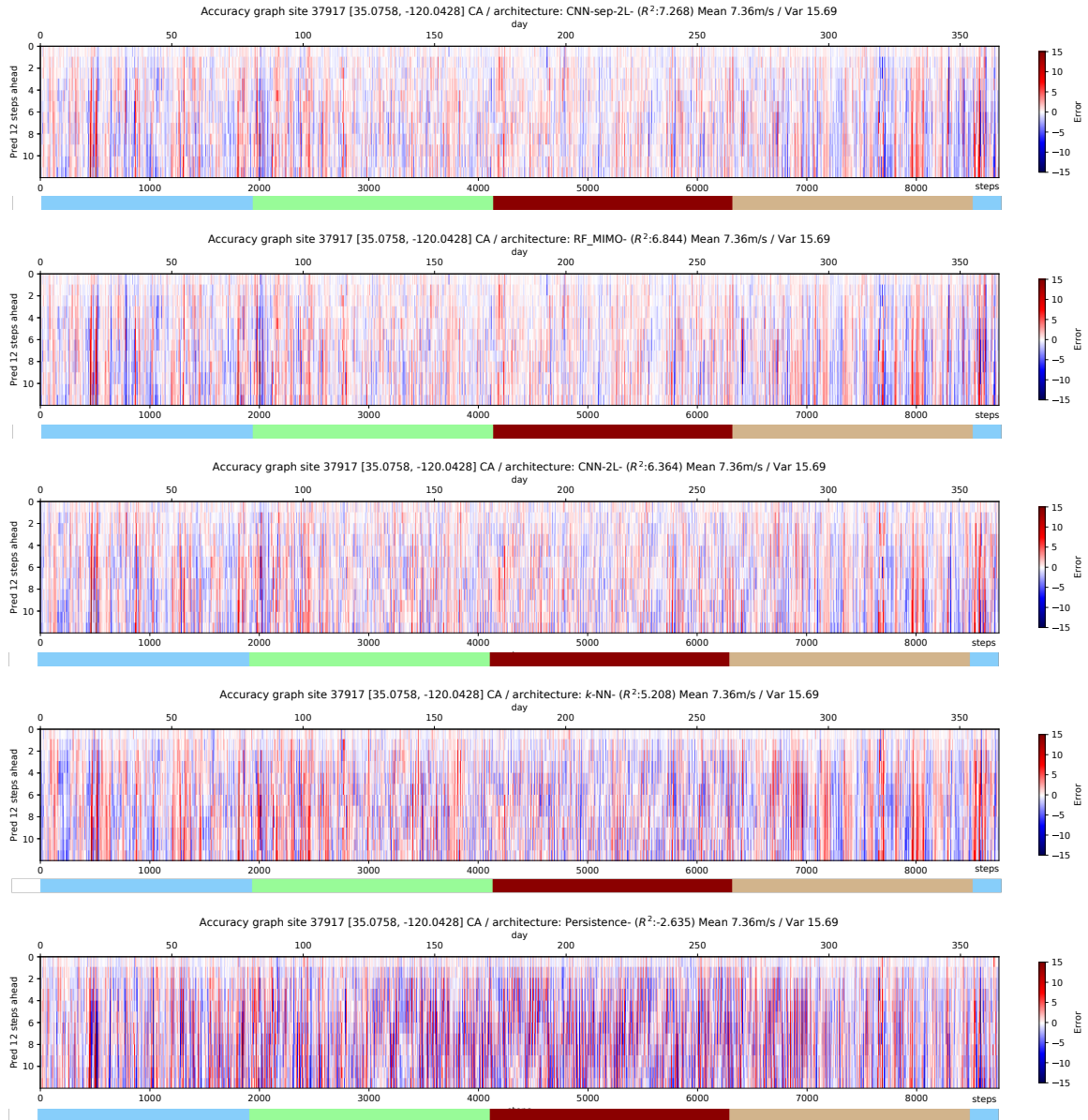
### 9.7.1 Site 37917 in California with average accuracy

The Site analysed in Figure 9.5 is number 37917 which is located at [35.0758, -120.0428] in California. We compare five models, with different accuracy. The models are sorted top-down based on the accuracy, with the best accuracy on top. The more accurate model is a Convolutional network with two layers, followed by random forest, then a classic convolutional with two layers, a  $k$ -NN and the bottom one is the Persistence. In this case, Random Forest outperforms CNN 2L, which does not often happen (as Random Forest usually under-performs the network models)

The  $R^2$  values are, from top to bottom 7.268, 6.844, 6.364, 3.8137 and -2.635. The Y-axis represents the predictions of 12 steps (0-12) in a vertical line and the X-axis the steps (hours). The values are calculated by subtracting the real value from the predicted.

We can observe how the patterns repeat across the models (lines with intense red or blue colour), and higher complexity of prediction in winter compared to the summer season. The better prediction capabilities of the top models can be visually perceived as the colours are lighter at the top methods.

## 9.7 Global and local error evaluation



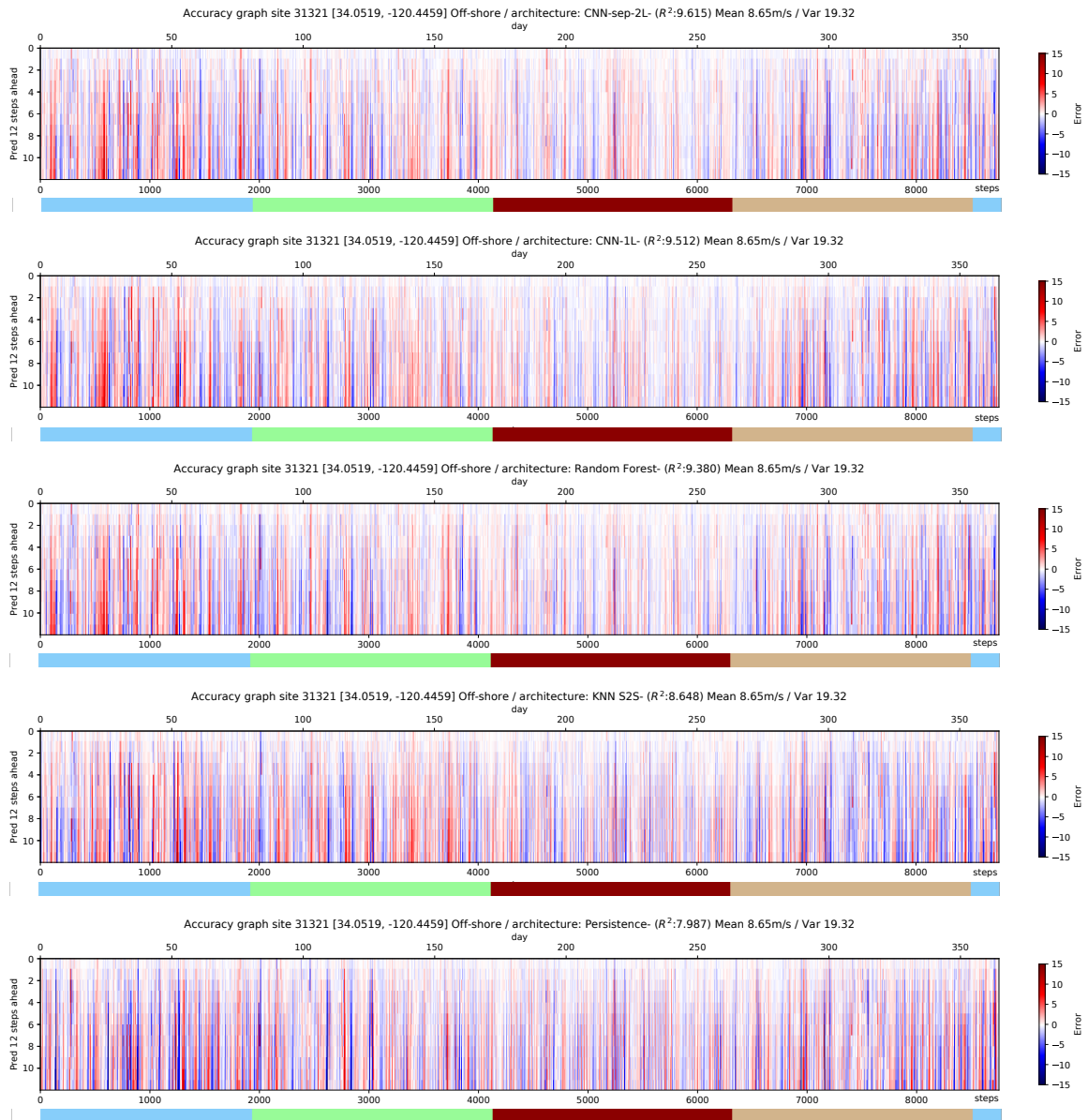
**Fig. 9.5** Graphical visualisation of prediction error for one year. Site number is 37917 located at [35.0758, -120.0428] in California. Vertical axis represents the 12 steps in the horizon (top is first step, bottom 12<sup>th</sup> step), horizontal axis are the hours in a year. The horizontal coloured bar illustrates seasons (winter: blue, spring: green, summer: red, fall: brown)

### 9.7.2 Site 31321 offshore with high accuracy

The Site analysed in Figure 9.6 is number 31321. It is a visualisation of the prediction error for one year. This site is located at [34.0519, -120.4459] which is an offshore site, in the Pacific in the California coast. We compare five models, with different accuracy. The top model is Convolutional separable network with two layers, above there is a classic convolutional with one layer and a Random Forest, then  $k$ -NN nearest neighbours and the bottom one is the Persistence. The  $R^2$  values are, from top to bottom 9.615, 9.512, 9.380, 8.648 and 7.987 The Y-axis represents the predictions of 12 steps (0-12) in a vertical line and the X-axis the steps (hours). Red is an excess prediction, blue a short and white a perfect one with zero error, the values are calculated by subtracting the real value from the predicted. We can observe how the patterns repeat across the models (lines with intense red or blue colour), and higher complexity of prediction in winter compared to the summer season season.

This site has better results in summer over winter, particularly at the end of summer where the stationarity of the series must be quite acute. The darkness increases top-down and can be observed plot to plot.

## 9.7 Global and local error evaluation



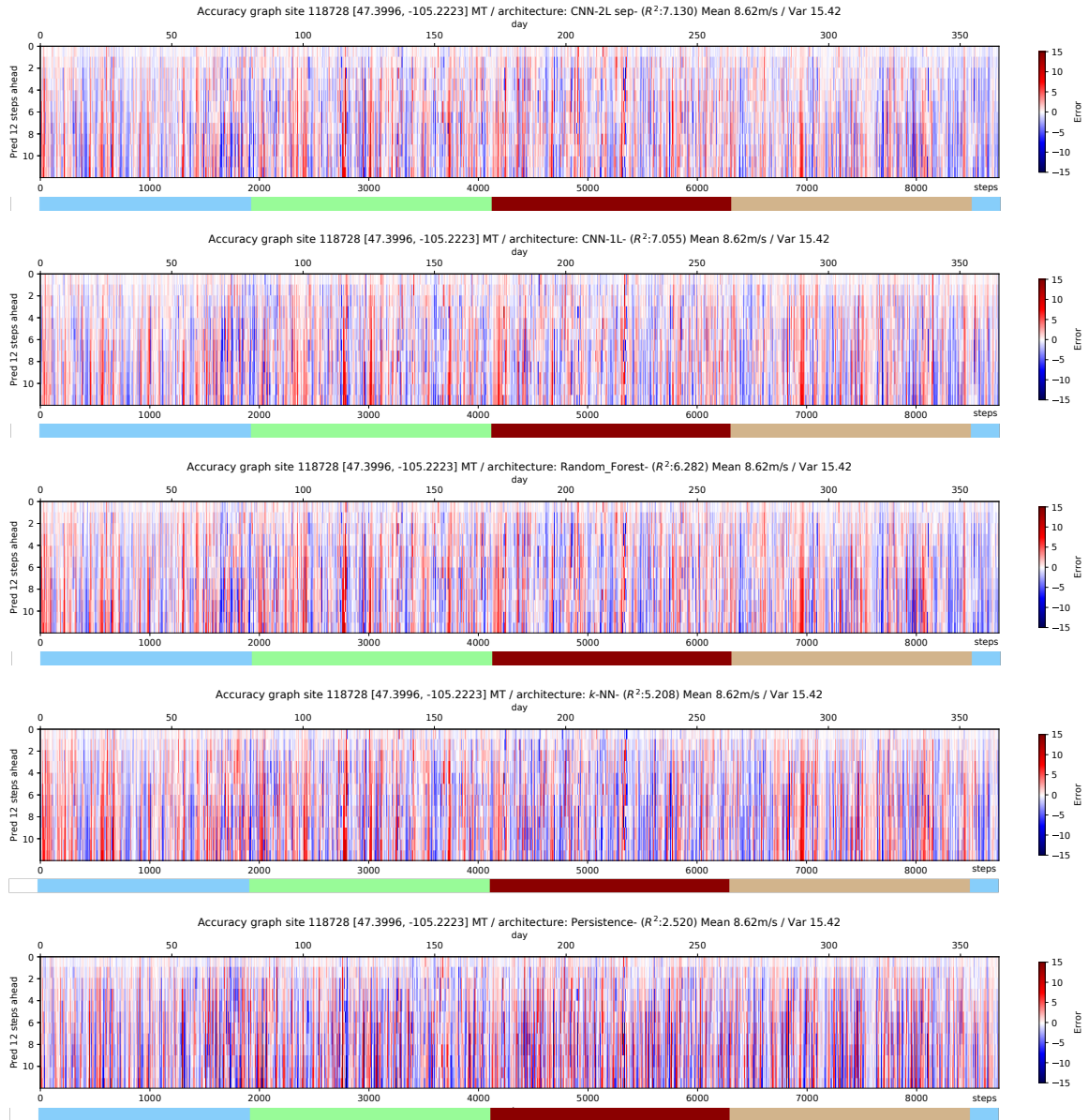
**Fig. 9.6** Graphical visualisation of prediction error for one year. The site number is 31321 is located at [35.0758, -120.0428] offshore in the Pacific. Vertical axis represents the 12 steps in the horizon (top is first step, bottom 12<sup>th</sup> step), horizontal axis are the hours in a year. The horizontal coloured bar illustrates seasons (winter: blue, spring: green, summer: red, fall: brown)

### 9.7.3 Site 118728 in Montana with average accuracy

In Figure 9.7, we see a graphical visualisation of prediction error for one year. The site number is 118728, located at [47.3996, -105223] in Montana. We compare five models, with different accuracy. The top model is Convolutional separable network with two layers, above there is a classic convolutional with one layer and a Random Forest, then  $k$ -NN nearest neighbours and the bottom one is the Persistence. The  $R^2$  values are, from top to bottom 7.130, 7.055, 6.282, 5.208, 2.520. The Y-axis represents the predictions of 12 steps (0-12) in a vertical line and the X-axis the steps (hours). Red is an excess prediction, blue a short and white a perfect one with zero error, the values are calculated by subtracting the real value from the predicted. We can observe how the patterns repeat across the models (lines with intense red or blue colour), and higher complexity of prediction in winter compared to the summer season.

This figure shows no seasonal components. What we can observe is a sharp variance, which tells there is a regime of changing winds, which as it is shown in the Figures.

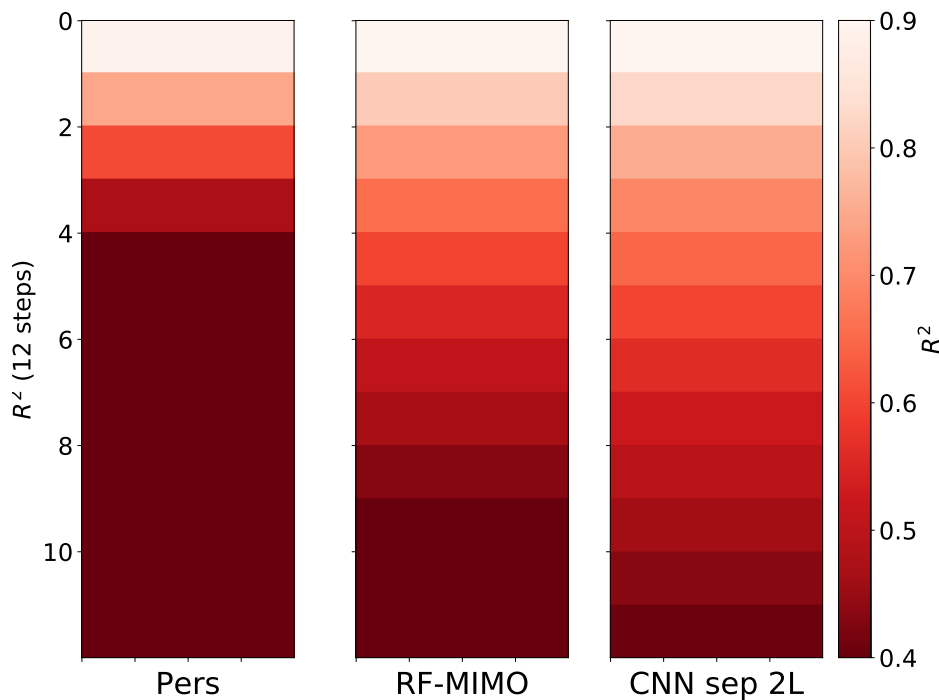
## 9.7 Global and local error evaluation



**Fig. 9.7** Graphical visualisation of prediction error for one year. The site number is 118728, located at [47.3996,-105.2223] in Montana. Vertical axis represents the 12 steps in the horizon (top is first step, bottom 12<sup>th</sup> step), horizontal axis are the hours in a year. The horizontal coloured bar illustrates seasons (winter: blue, spring: green, summer: red, fall: brown)

## 9.8 Discussion

Wind prediction from time series is a hard problem due to the complexity of the wind time series, and the variability of results, depending on the site characteristics, make necessary to test the algorithms on a broad spectrum of sites. Our approach has been a single algorithm but trained on each site. Each architecture has been tested on the total dataset or random samples of the dataset to assure statistical robustness to the analysis. All the convolutional methods have improved the baseline statistical method, showing the feasibility of the application of deep learning architectures to the wind forecasting problem. We learned from



**Fig. 9.8** Comparison of  $R^2$  between different methods step by step. Steps are represented top-down. Dark red is high error. CNN models obtain better results than persistence or Random Forest methods across all steps, all the results are gathered in [A](#)

past work [144] that deep learning has the potential to perform accurate wind time series forecasting and between the different approaches (MLP, CNN, RNN), the convolutional architectures obtain the best results, motivating the execution of a set of experiments with convolutional architectures and variants that are described in this chapter.

Regarding the convolutional networks, we observe better results with the separable convolution over the classic (see Table 9.6). The circumstance that the use of separable convolutions improves the classic convolutional operation is a fact already observed in image recognition tasks [34], but there is no reference for this result for time series.

In the experimentation, we have reviewed the impact of several parameters, from depth multiplier to the best activation functions to the length of strides or best kernel sizes, a



---

parameter combination illustrated in Table 9.5 for one-hour series, and in Table 9.11 for series at 5 minutes. We conclude that the number of parameters requires a structured approach for hyperparameter setting as there are many combinations of parameters  $\theta$  that generate results with accuracy differences of some hundredths. In this environment, ensembles, approaches can be useful as it shows the improvements of accuracy using a naive boosting approach, a method that is the best overall (see Section 9.5.4).

As a collateral conclusion, we have not been able to obtain any improved accuracy using stacked block architectures like the multi-headed models.

An initial question was how deep in the past had to be the examples for training. As we are using a MIMO approach, the input sequence has a length *lag* defining how many past values to be used for training. The conclusion is that past observations go from six to eighteen hours. More extended periods do not add accuracy to the result. This finding is aligned with earth sciences which describe the short time-span for wind formation [117].

The original series have a period of five minutes, but we decided to average them to one hour for practical reasons (for a discussion on data pre-processing see Section 6.2.1). In this way, the total length of a time series is 61,368, (divided in 43,834 for training - five years- and 17,534 for testing and validation -two years-). We have observed that an increase in the amount of data leads to an increase in accuracy with similar algorithms. an outcome that appears comparing the results from the same algorithms working with the different period series. However, we need to consider that, to obtain the accuracy gains, the data increases  $\times$  twelve times, and the use of computing resources increases accordingly.



## Chapter 10

# Wind time series forecastability

Characterising a wind site for its predictability is a useful tool for wind resource estimation. Multi-variate time series of meteorological observations obtained from a site contain relevant information to produce a short-mid term multi-step ahead prediction.

In this chapter, we define measures based on time series properties, like spectral, entropy and stationarity analysis that show high correlation with prediction accuracy.

To test the correlation, we apply deep learning methods to the NREL dataset, and we test and compare the accuracy results with the measures. The results show the correlations of the measures with the deep learning method accuracy.

With the results, we define forecastability measures that are useful indexes of the future applicability of the methods, introducing a new way to classify wind sites that can be valuable for the industry.

This chapter will generate an article submission.

### 10.1 Forecastability and deep learning prediction

When we evaluate the potential of a wind site, there are several features to be considered, like the wind strength, the site accessibility, or its connectivity to the existing grid [23]. The first, and possibly, the essential feature is the wind intensity, as this defines the amount of energy to be generated. However, for this energy to become economically viable, it needs to be predictable. For this reason, site predictability becomes a feature that contributes to the value of a generation location.

Forecasting is critical for all the actors in the electricity system management which are; the power producer, the transmission system operators (TSO) Transmission system operators, or the commercial companies responsible for the access to electricity to consumers. The introduction of smart grids with smart (real-time) metering capabilities, has generated demand-response markets [223], a mechanism that allows changes of demand based on the electricity spot price, and electricity trades by continuous bidding between generators and consumers. In this scenario, forecasting demand and generation are critical, with relevant economic impact. The growth of renewable energy is unstoppable. With self-generation widespread and batteries economically viable, the electricity systems become bi-directional, and its control based on sophisticated algorithms that assure continuous balancing of the supply and demand in the electricity system [100].

Forecasting is an essential activity for the wind industry. We can forecast wind energy either at a turbine level or at a wind park level [129], and for some applications, we require prediction in a wide area (like at country level) [77]. This thesis focuses in prediction for a single site, or turbine, and in this way, we align the prediction to a wind time series, as predicting for a broader scope cannot be done on a single series. For this individual prediction, the accuracy of a method is based on one time series exclusively, and then the result of a prediction is the combination of two effects, the capability of the algorithm to model the wind series, and the complexity of these series. In previous chapters, we work in the accuracy of the algorithms, in this one we look into the inner structure of the series to define its forecastability or predictability, a property determined by the inner characteristics of the wind series which defines its prediction complexity [104, 178].

Forecastability goes far beyond the traditional statistical definitions like mean or variance, as those simple measures do not show strong correlations with a prediction. We illustrate this point in Figure 10.2, where we can see in three scatter-plots with measures distributed in the map with a corresponding scatter-plot of those measures with a prediction made with a deep learning algorithm.

Forecastability is an essential measure for two reasons, firstly it has economic value, as a site easy to predict has more value than a difficult one, and secondly, it is a relevant measure to evaluate the prediction outcome from forecasting algorithms. A prediction needs to be

rated evaluating the site forecastability. The algorithm result needs to be related to the site complexity; in this way, we can rate better the algorithm efficiency.

As new tools and algorithms appear for wind prediction, the interest in predictability measures is growing as some recent works show [53, 54].

This thesis defines some forecastability measures based on spectral and stationarity analysis for wind time series. The described measures have strong correlations with deep learning multi-variate and multi-step models for 12h ahead predictions [145]. Additionally, this article verifies the impact of terrain ruggedness in forecastability, finding a low general correlation between ruggedness and predictability. All the experimental work has been performed on the NREL Wind dataset, that has 126,692 wind sites across North America [46].

This chapter is organised as follows. First we discuss the role of forecastability in wind time series and its relationship with deep learning, then in Sections 10.4 and 10.4, we analyse terrain complexity as a possible descriptor in Section 10.5 followed by the analysis of decomposition and entropy-based indexes in Sections 10.6 and 10.7.

## 10.2 Experiment settings

To build and analyse forecastability indexes, we followed a process that is synthesised in Figure 10.1. This process has as an objective to validate the properties of several forecastability indexes, based on three significant methodologies with a set of predictions generated by deep learning methods using wind speed time series from a broad sample of sites (NREL wind dataset).

The deep learning forecasting results come from over thirty different methods that perform a forecast with a 12h ahead horizon. The deep learning models belong to three major categories, which are CNN, MLP and RNN, and have been analysed in previous chapters of this thesis. The experiment results expressed as mean and variance on the test and validation sets are in Appendix A.

We distinguish between simple statistical measures and forecastability indexes. Simple statistical measures like the mean or the variance of the results distributions (see Figure 10.2). These measures are descriptive and can help to understand how the wind resource is distributed geographically, thanks to the geographical distribution of the source data. Geography is essential for wind as its formation depends on the integration of local features with global phenomena, and by representing them in maps allows us to understand relationships between local and global terrain features.

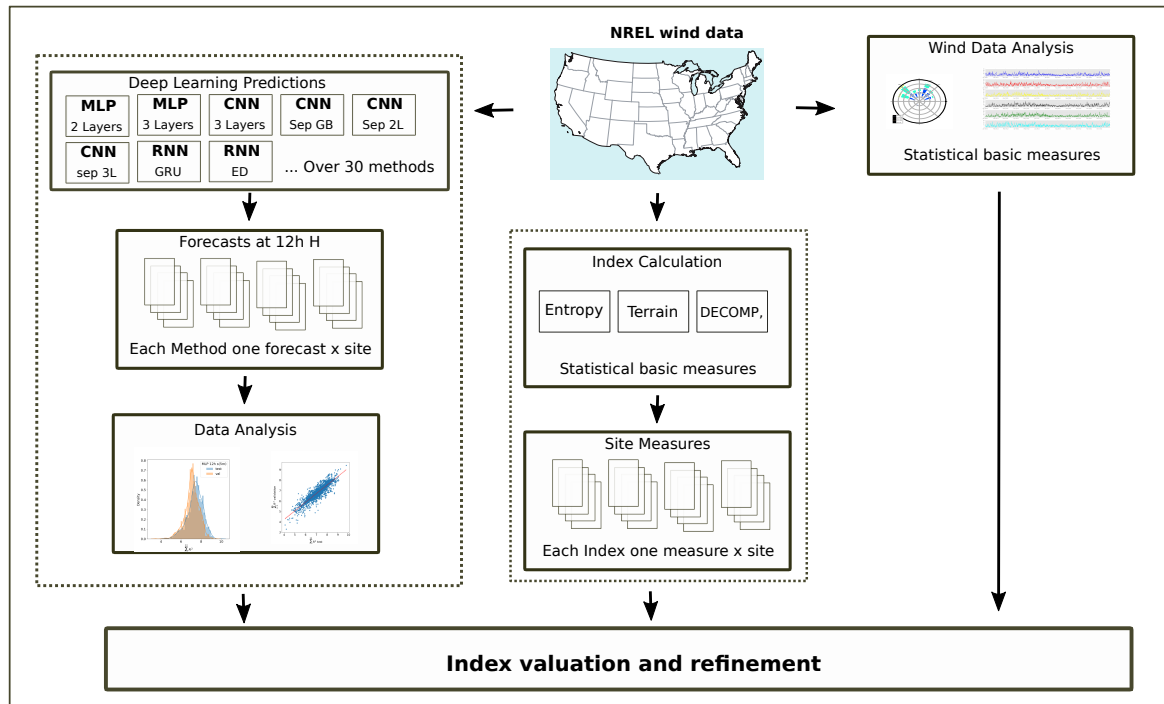


Fig. 10.1 Experimentation process for forecastability indexes

### 10.3 Initial Testing and Pearson Correlation

We calculated the mean and variance for all the wind series in the NREL dataset. Additionally, we added the elevation in  $m/s$  to each site. The elevation height can represent indirectly (by relating neighbouring points) how terrain changes its slope. The changing colour in the elevation map shows mountains. A stable colour shows flat terrains. The elevation is required to analyse the relationship between terrain complexity and accuracy. In Figure 10.2, we compare the three representations, mean, variance and elevation, which follow the major orography features of the US geography. In the referred figure we accompany each measure with a scatter-plot made with the accuracy distribution (measured in  $R^2$ ) of the results from a deep learning prediction method, in this case, a two-layer CNN with separable convolutions. The regression lines in the scatter-plots show a, albeit small, relationship between the distributions, this relationship is more consistent in mean and elevation and very disperse with variance. To see the correlation between the distributions, we choose the Pearson coefficient, which is represented by:

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \quad (10.1)$$

where  $\sigma$  is the standard deviation of the distribution  $X$ , and  $E$  is the expectation or weighted mean of the distribution. We calculate the correlation between four prediction methods and the three measures, mean, variance and elevation (see Table 10.1). This table illustrates the

relationships and confirms some of our intuitive observations previously made from the visual interpretation of the scatter-plots. The results confirm a negative correlation of the  $R^2$  accuracy of the deep learning prediction methods with the elevation of a site.

The four methods show differences. Persistence has different dynamics than the rest as it has the highest correlations with mean and variance and the lowest with elevation. It is not surprising as this method must follow the series behaviour, and its results show this closeness. Random Forest has the lowest correlation with mean and more variance than the deep learning methods. Finally, the two deep learning methods are very similar. The best method, the CNN separable 2 Layers, has the highest correlation with elevation and with the mean, with an MLP 2L very close.

**Table 10.1**

Correlation between Simple statistical measures and five DL forecasts over the NREL dataset.  
The models are: CNN separable convolution 2 layers, MLP two layers, Random Forest and Persistence

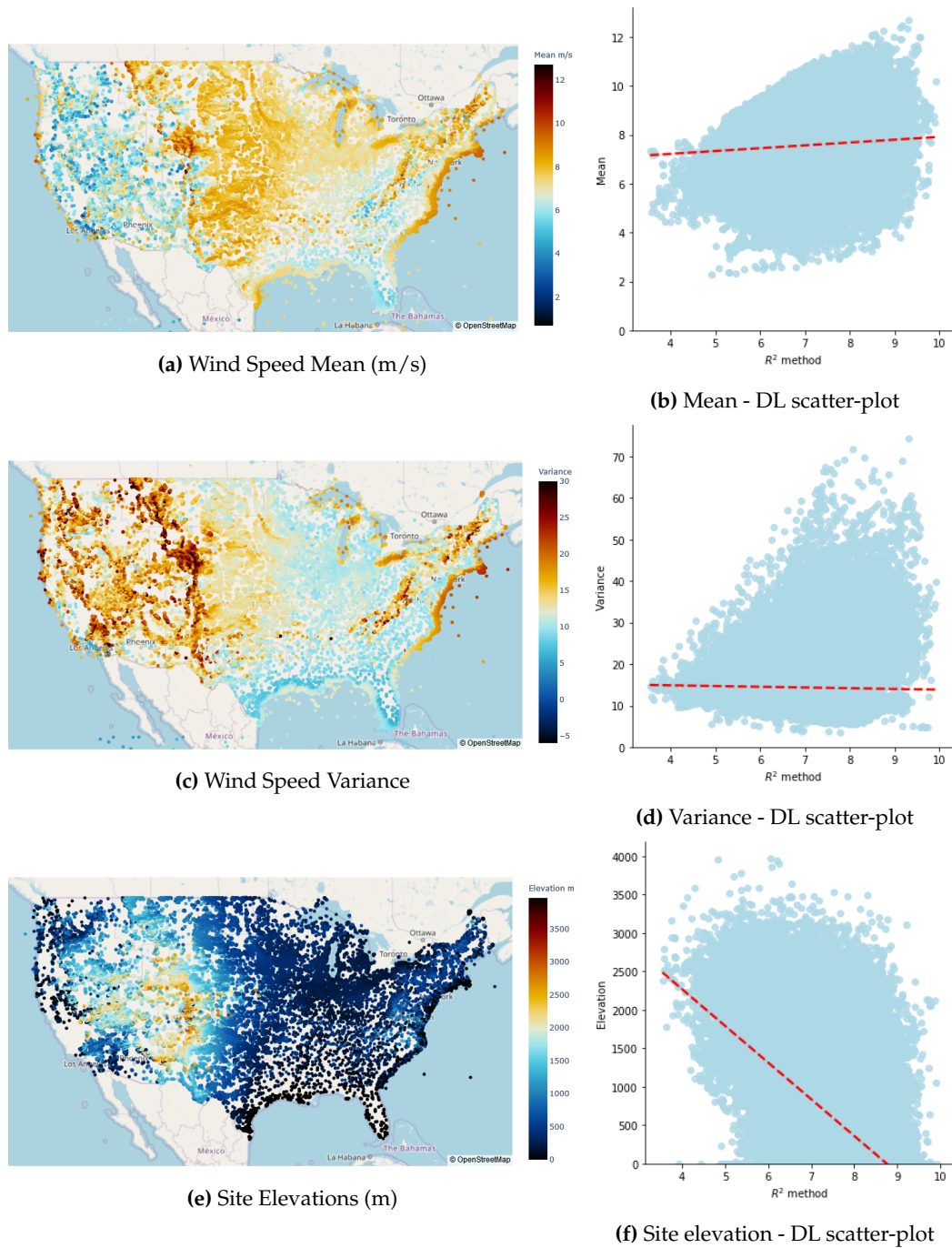
index	CNN sep 2L	MLP 2L	RF	Persistence
Mean	0.136	0.129	0.113	0.161
Var	0.036	0.066	0.114	0.272
Elevation	-0.507	-0.474	-0.423	-0.267

Our objective in this chapter is not to design another forecasting algorithm, but to understand how the wind time series internal characteristics or some physical features, like elevation or ruggedness, impact in the deep learning results accuracy of several algorithms. We want to understand better the forecastability capability of a site to improve the forecasting methods, in particular, the deep learning ones.

We classify the origin of the measures in three major group categories:

- Statistical: these are mean, variance and wind-speed variability.
- Physical: ruggedness and elevation.
- Signal analysis: Spectral Entropy, Lumpiness and Stability
- Series decomposition: From the decomposition of a series into trend, seasonality and residual we obtain several elements that combined offer new measures that give us an insight on the inner series structure.

## Wind time series forecastability



**Fig. 10.2** Geo-spatial representation of mean, variance and site elevation for the NREL dataset, plus its plot against the accuracy of wind speed site prediction based on a neural network with two layers of separable 1d convolutions



## 10.4 Statistical measures in wind prediction

We understand as basic characterisation measures those that define the structure of the wind series using some simple statistical tests. In this work, for the NREL dataset, we include in this category mean, and variance.

Wind is a very complex phenomenon because it is generated from global interactions of the atmosphere at planetary level, like the cold weather currents from the Arctic, or the trade winds in the Atlantic. However, these global events impact local features, in climate-related anomalies like deserts, forests or rivers, and then act on even more local attributes, like small hills or a canyon. For the most local qualities, the aerodynamics analysis is more important than the climate, as minor turbulence can have a significant effect on the wind speed. For instance, the wake effect between turbines is an essential event in wind parks that appears a consequence of the turbulence generated by a turbine into the neighbour ones [78].

This variability needs to be present in the data. Performing an analysis on series that are in a local wind cluster can be extremely biased, as the data needs to be diverse and from a wide area. For this reason, we use the NREL dataset that contains very different geographies and wind topologies, allowing us to perform analysis using very different time series.

We have performed several tests on the NREL dataset to characterise its variability and main characteristics (see Figure 10.2) focusing on wind speed, which is the most relevant variable for prediction. We have drawn three maps, for mean, variance and elevation.

These time series descriptive measures have different correlations with deep learning predictions (see Figure 9.3 that shows a distribution in a map of a deep learning prediction accuracy  $R^2$ ). The application of the statistical measures on the map offers some descriptive information. We illustrate in Table 10.1 the correlations between variance and mean, plus an analysis of elevation with different forecasting methods.

The first conclusion is that simple statistical measures do not grasp the wind structure, or using the new definition, they do not show good forecastability and, as a consequence, they cannot appear as forecastability indexes. We need more robust measures to obtain forecastability ability on the indexes.

## 10.5 Terrain complexity as a forecastability descriptor

Terrain complexity is considered a measure of the prediction difficulty of a specific wind site. This idea is cited in different works in the literature in several works [104, 73]. However, a conceptualisation of terrain complexity is not found in in a structured way.

Tabas et al. in [207] describe this complexity based on micro terrain features, topography, forest structure and wake effects from other close turbines.

Kariniotakis et al. in [105] displays the idea of high error related to terrain complexity. However, in this same work, they found that off-shore sites, with no terrain complexity, had a high prediction error, possibly due to lack of skill on the prediction task for that kind of sites, with this finding, they question the impact of terrain complexity isolated, as it needs to be considered associated to other elements.

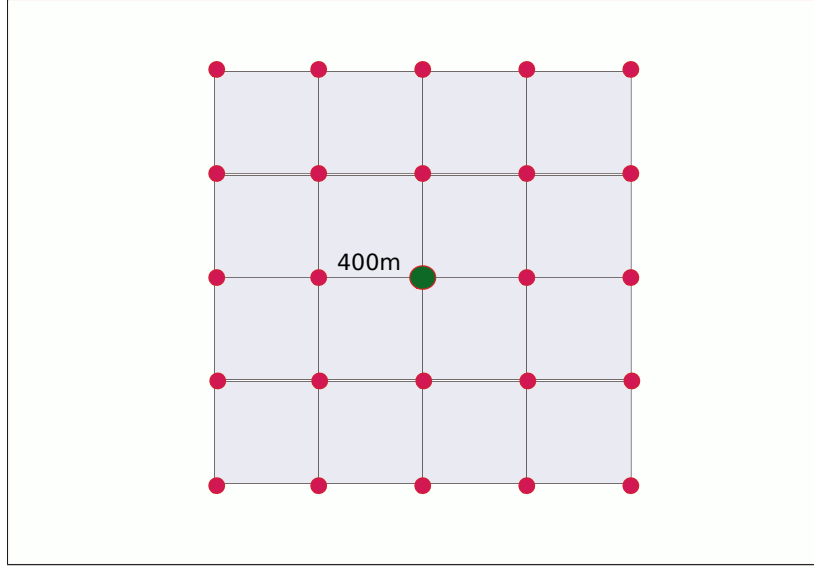


Fig. 10.3 Grid of points around a site (site in green)

We want to explore the ability of a ruggedness measure to correlate with prediction accuracy, isolated from any other circumstance. This index needs to tell if a site in a mountainous area is more difficult to predict than a one in a non-complex or flat area.

First, we need predictions. We have performed deep learning predictions for all sites in the NREL dataset, and the results show a prediction distribution, measured in  $R^2$ . The means and variances of all the DL methods are compiled in Appendix A.

To calculate the ruggedness first, we define a grid of points around each site as it is displayed in Figure 10.3, then we assign to each point its elevation calculated as its height from sea level calculated in  $m/s$ . The elevations are obtained from the US Geological Survey's (USGS) National Map [70].

This grid will allow implementing two two Ruggedness indexes, that we named  $IR_1$  and  $IR_2$ .

Index 1 ( $IR_1$ ) is designed based on the method described by Riley and Elliot in [177], using the equation:

$$I_1 = \sqrt{\sum_{i=1}^{16} (c_i - c_{site})^2} \quad (10.2)$$

## 10.5 Terrain complexity as a forecastability descriptor

Where  $c$  is the average elevation on each grid square (average of 4 vertex elevations), and  $c_{site}$  is the site elevation.

Index 2 ( $IR_2$ ) is inspired in a work from Liu et al. in [125], They present some ideas for the calculation of ruggedness indexes. The core idea is to calculate an angle based on the individual normal vectors of the sub-grid planes around the site. To calculate this index, we divide each one of the 16 grid squares in two triangles, consequently a plane. Each plane has a normal vector that we calculate. The last step consists of obtaining the angle of each normal vector with the vertical (90%), summarising the values of the angles we obtain the  $I_2$  index (see Equation 10.4).

$$cf_i = e^{1-\cos(\alpha_i)} \quad (10.3)$$

$$I_2 = \sum_{i=1}^{32} cf_i \quad (10.4)$$

These two indexes represent the terrain variations around every site accurately, for a very complex site, the values are very low, close to 0 and, for flat terrain is 1. In Figure 10.5, we can see a representation of 3 grids from different sites with flat, mountainous and inclined terrain.

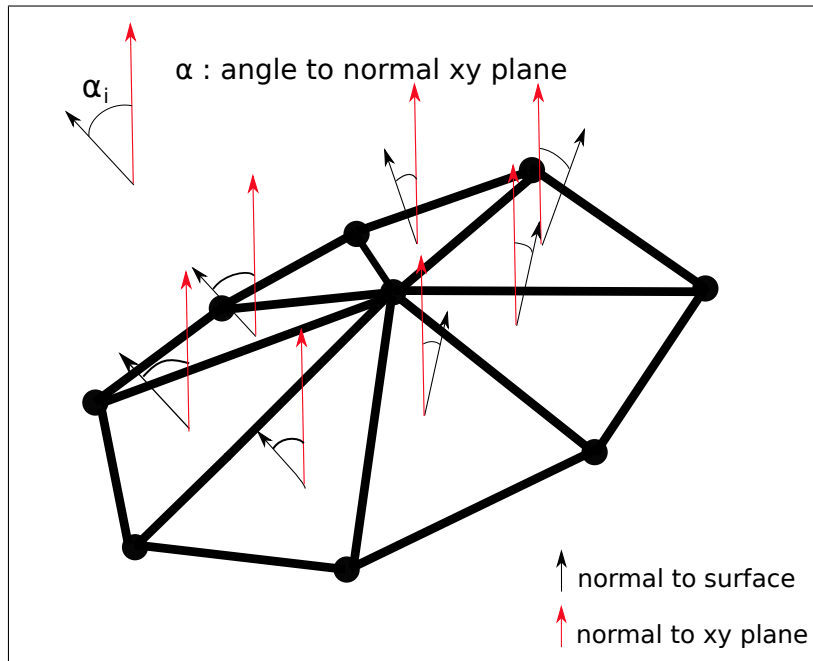
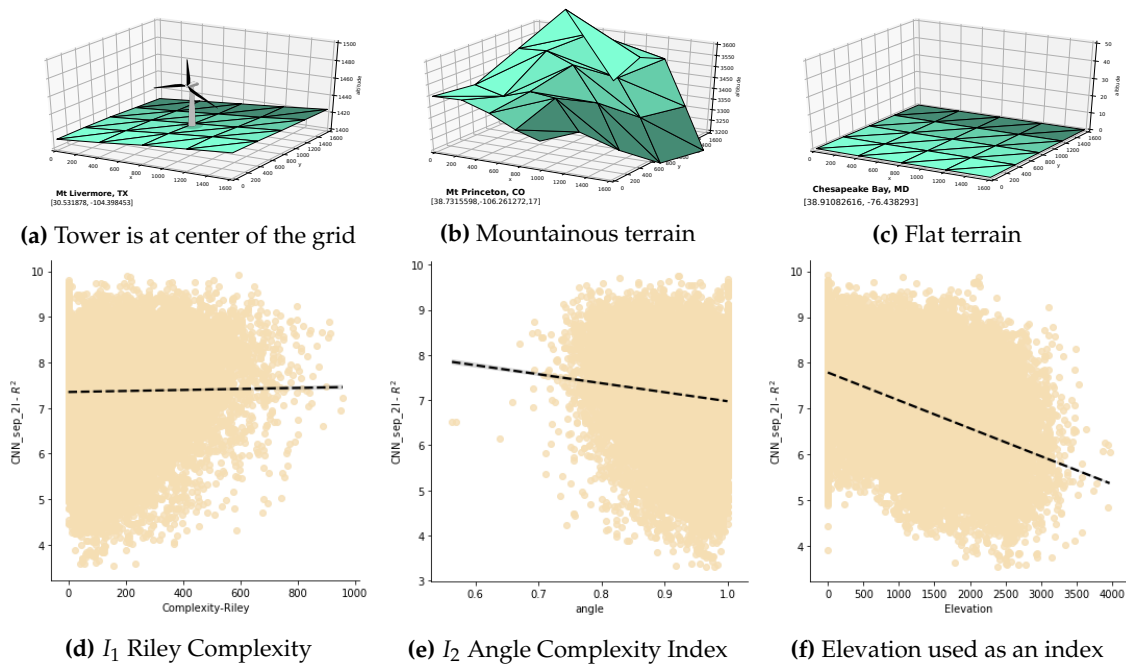


Fig. 10.4 Grid of points around a site (site in green)

However, the correlations of these indexes with the Prediction error of the deep learning methods is very low.

## Wind time series forecastability



**Fig. 10.5** Representation of flat and mountainous terrain. Visualisation of Wind Tower in the terrain (a), Visualisation of different terrain complexity indexes

Figure 10.5 shows, in subplots (d), (e) and (f) the scatter-plots that between three measures  $I_1$ ,  $I_2$  and Elevation. In those figures, we can see how the clouds of points are not concentrated around on the regression line, showing a low correlation.

We can conclude that ruggedness only is not a good predictor of site forecastability, and we believe the reasons sit on the data. In the dataset there are many low-complexity (on flat terrain) sites with high errors. This finding aligns with the already mentioned issues with off-shore sites in [105], flat terrain is not correlated with accuracy, and this disturbs the exploration of errors in the mountainous sites, where we find accurate and inaccurate results. The conclusions on this analysis are that indexes only based on terrain are not good predictors and cannot be used as forecastability indexes.

Intuitively a site can be located in a simple terrain like a plain or a sea-shore and be subjected to easy wind patterns or, in the opposite side, it can be located in rugged terrain with fair and stationary winds, becoming a site easy to predict. Terrain has a strong influence in the wind, as it is described in the literature [117], but indexes only based in elevation or ruggedness, are not suitable to establish the predictability of a specific site.

The indexes  $I_1$  and  $I_2$  of ruggedness interpret terrain very close to the site, as we consider 800-1000 meters around a geographical location. The ruggedness of a larger area may be a better measure to understand the forecastability of a point, but we do not expect these indexes to capture forecastability. We observe, in the US geography, that wind complex patterns come in flat and mountainous areas and vice-versa. Level terrain can be complex

like some coastline environments, or in the central plains, which have some of the most variable winds (see a depiction of error complexity from a forecast in Figure 9.3). The more difficult area to predict in the US are those flat terrain sites that form the plains from Canada to Mexico.

As in the NREL dataset, there are over 15,000 sites in the plains. This number has an impact when calculating the correlations, making flatness a lousy predictor for accuracy. We observe in the dataset sites with low ruggedness and high complexity in the plains and, as there is such a high number, it impacts the overall correlation.

In conclusion, we see that ruggedness is not a good predictor, as there are flat complex areas and easy mountainous ones. The complexity of a site requires additional information to ruggedness.

## 10.6 Time series decomposition measures

Time series can be decomposed into components using different methods. Initially, we have used a moving averages method for decomposition.

If a wind time series repeats in cycles, it implies seasonal pattern, like a day-night flow, or winter-summer, or seasonal modifications of wind over the years. A good predictor will identify these patterns, and the highly seasonal sites will have good forecastability. If we can find seasonality in a time series, most probably we will be able to correlate this seasonality with accuracy.

An addition decomposition of a series  $W_t$  has the following structure:

$$W_t = S_t + T_t + E_t \quad (10.5)$$

Where  $S_t$  is the seasonal component,  $T_t$  the trend and  $E_t$  the residual or error component. The decomposition creates three components, where the last component,  $E_t$  is a random error with zero-mean and correlated over time.

Seasonality in a time series is defined as a pattern that repeats over time. A significant auto-correlation coefficient identifies seasonality, that can be partial and related to the seasonal portion.

For data with a strong trend, the seasonality adjusted data (deseasonalised data)  $W_t - S_t$  has more variation than the residual component, for data without trend the two variances would be the same. In this sense, we define the strength of trend as:

From the decomposition components, we define two indexes inspired in previous work from [235], which are  $f_T$  and  $f_T$

$$f_T = 1 - \frac{\text{var}(E_t)}{\text{var}(W_t - S_t)} \quad (10.6)$$

## Wind time series forecastability

---

Strength of seasonality is defined in the same way, in this case the variance used is the detrended data,  $W_t - T$

$$f_S = 1 - \frac{\text{var}(E_t)}{\text{var}(W_t - T_t)} \quad (10.7)$$

Where  $(W_t - T_t)$  is the detrended series. These indexes contain valuable information about the inherent seasonality and trend structure in the time series (see Table 10.3). A time series with seasonality strength  $f_S$  equal to 0 has no seasonality, and when  $f_S$  is close to 1, it points to a strong seasonality [235].

Our goal is to characterise wind time series, and we apply the indexes to all the NREL dataset series. With the results we represent the indexes on the US geography to compare them and interpret their value Figure 10.6 shows the index values representation on the US map and three scatter-plots comparing the index value with a deep learning prediction with a CNN separable with two layers.

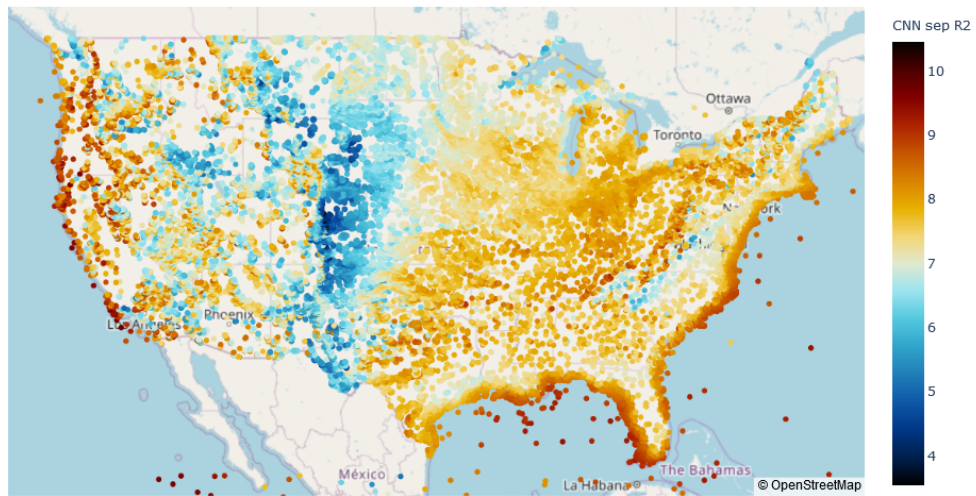
We obtain several insights from the representations.

- The strength trend  $f_T$  has good correlation with the  $R^2$  results from the prediction (CNN sep 2L). There is a set of outliers in Figure 10.6c with small strength trend. We analysed the outliers, and they are not geographically related as they are spread on the geography.
- Strength seasonality  $f_S$  (Figure 10.6e) does not show a useful correlation, and the cloud of points is very dispersed.

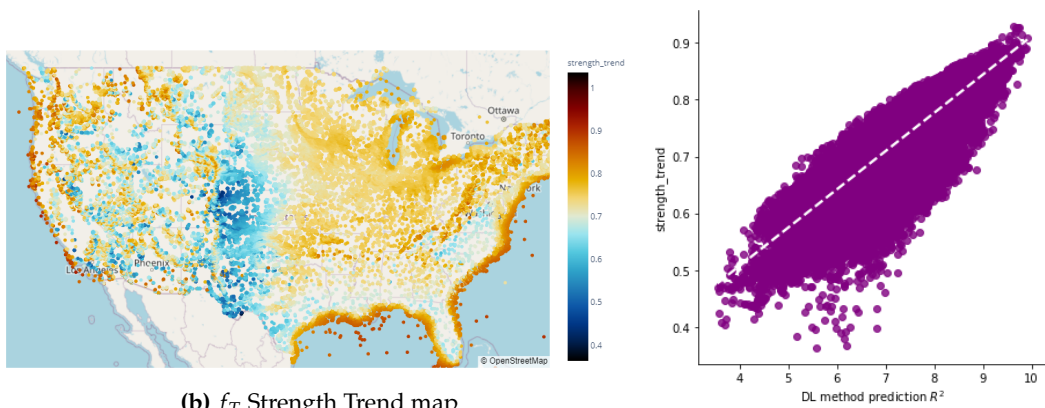
Something that we observe in the geographical representation from Figure 10.6 is that the index values generate geographical areas, for instance, the plains and the Rockies are highlighted while the coastlines show different values profiling different time series characteristics.

With this finding, we want to explore the relationship between the  $f_T$  index (which has shown to be more informative) and all the US states to verify a relationship between geographical areas and index values. We know that the states are political divisions and not necessarily common geographical areas, but having a smaller size, they may raise the homogeneity of the results. For this exploration, we design an experiment where we calculate the correlation between the index  $f_T$  and a forecast  $R^2$  error values state by state.

## 10.6 Time series decomposition measures

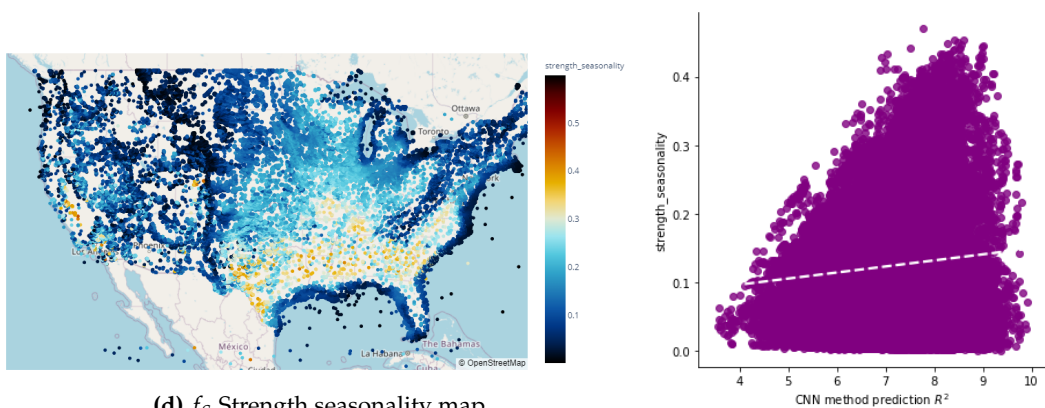


(a) CNN separable 2L  $R^2$  accuracy results



(b)  $f_T$  Strength Trend map

(c) Strength trend scatter-plot



(d)  $f_S$  Strength seasonality map

(e) Strength seasonality scatter-plot

Fig. 10.6 Geographical representation of strength trend  $f_T$  and strength seasonal  $f_S$

## Wind time series forecastability

This analysis needs to be balanced with the number of sites on each state, as there are states with a higher density of sites, and other states are larger (example Texas, Wyoming or South Dakota). Another point to consider is the low density of states like New Mexico (the sites are in feasible wind generation areas outside populated urban areas).

Table 10.2 shows the results. We want to highlight a couple of findings.

- The high correlation in Wyoming and Oklahoma, both with a large number of sites.
- The low correlation with Iowa and Illinois, both with a large number of sites.
- The differences between Illinois and Indiana, two adjacent states that show significant differences, point two different wind pattern areas (in this case, as a hypothesis, possibly for the influence of the Great Lakes in Illinois winds).

**Table 10.2**

Correlation between wind speed prediction accuracy measured in  $R^2$  using a CNN separable with 2 layers with  $f_T$  on all US states.

State	Corr	Sites	State	Corr	Sites	State	Corr	Sites
Alabama - AL	0.491	89	Louisiana - LA	0.791	558	Ohio - OH	0.652	2936
Alaska - AK	NA	0	Maine - ME	0.947	1201	Oklahoma - OK	0.845	4303
Arizona - AZ	0.785	2376	Maryland - MD	0.921	334	Oregon - OR	0.776	2636
Arkansas - AR	0.761	823	Massachusetts - MA	0.904	676	Pennsylvania - PA	0.815	1502
California - CA	0.643	4053	Michigan - MI	0.539	3609	Rhode Island - RI	0.935	189
Colorado - CO	0.698	3652	Minnesota - MN	0.771	6372	South Carolina - SC	0.916	271
Connecticut - CT	0.813	156	Mississippi - MS	0.527	187	South Dakota - SD	0.890	6012
Delaware - DE	0.927	151	Missouri - MO	0.814	1836	Tennessee - TN	0.841	287
Florida - FL	0.675	1040	Montana - MT	0.931	4091	Texas - TX	0.793	8061
Georgia - GA	0.524	273	Nebraska - NE	0.928	3708	Utah - UT	0.789	1715
Hawaii - HI	NA	0	Nevada - NV	0.622	3865	Vermont - VT	0.916	443
Idaho - ID	0.821	1451	New Hamps. - NH	0.927	409	Virginia - VA	0.814	869
Illinois - IL	0.394	5253	New Jersey - NJ	0.868	483	Washington - WA	0.853	1632
Indiana - IN	0.886	3195	New Mexico - NM	0.816	6584	West Virginia - WV	0.738	545
Iowa - IA	0.567	5253	New York - NY	0.743	3156	Wisconsin - WI	0.416	2883
Kansas - KS	0.807	4150	North Carol. - NC	0.896	849	Wyoming - WY	0.908	8201
Kentucky - KY	0.819	347	North Dakota - ND	0.789	3380			

To go deeper into this analysis, we represent three states, one with a lower correlation (Florida) another one with a very high correlation and a large number of sites (Wyoming) and finally Maine which has a low number of sites and high correlation.

In Figure 10.7 we can see the different scatter-plots. For Florida we see a high dispersion of the values (see Figure 10.7d), which can be explained by a site list very distributed on the whole state, Wyoming, has a great number of sites, but concentrated in wind corridors in the south-east part of the state (see Figure 10.7e), this explains the homogeneity of the results in such a large state. Maine has all the sites concentrated in the north-east part of the state, in a very homogeneous wind area 10.7f).



10.6.1 Value of decomposition measures for wind time series

With this exploration on the application of  $f_T$  and  $f_S$ , we see, first that the  $f_S$  index in its actual formulation is not useful, on the contrary, the  $f_T$  index shows high competence to qualify a future prediction.

In a forecastability measures tool-set, the  $f_T$  index has to be considered for its ability to represent patterns that later influence the results of a forecast.

Table 10.3

Correlation between decomposition measures and wind speed prediction accuracy measured in  $R^2$  using a CNN separable with 2 layers

Measure	Description	NREL Dataset	Florida	Wyoming	Maine
$f_T$	Trend Strength	0.841	0.726	0.936	0.906
$f_S$	Seasonality Strength	0.110	-0.619	-0.257	-0.266

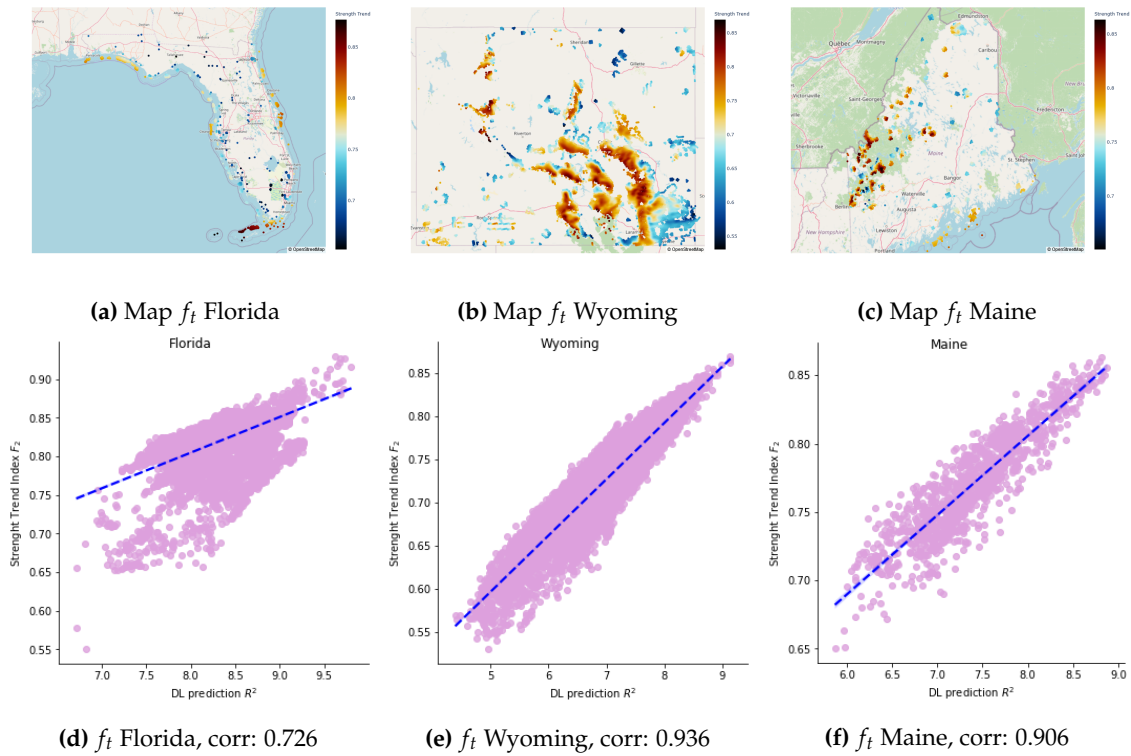


Fig. 10.7 Trend strength  $f_t$  for three states, Florida, Wyoming and Maine. The three states show high correlation between the correlation index  $f_T$  and the prediction.

The wind time series do not show a trend, and in this case, there are many decomposition strategies. We applied a moving averages decomposition, and created the  $f_T$  and  $f_S$  indexes. However, we are currently exploring the application of Loess methods (the seasonal and Trend Decomposition method STL) that offer alternative decomposition tools to determine a more effective approach for determining the seasonality.

### 10.7 Entropy analysis applied to wind time series

Entropy is a measure of the uncertainty of a random variable. In the time series field, it is often used as a property to qualify the inner structure of a temporal series.

For wind, we have identified some works that use entropy as support for forecast improvement like [Sun and Wang](#) in [202] that use entropy to decompose the original series to apply a novel method on the decomposed sub-series.

In this section, we want to identify the capabilities of indexes inspired by the entropy analysis to act as forecastability predictors. We define four indexes, sample entropy (SampEnt), spectral entropy (SpecEnt), Lumpiness (Lump) and stability (Stab).

- **Sample entropy** assesses the complexity of a time series, a large value indicates high complexity, while a small one indicates low complexity or a more regular series. It was initially created for physiological time-series signals [176].

$$H(x, m, r) = -\log \left( \frac{C(m+1, r)}{C(m, r)} \right) \quad (10.8)$$

where  $m$  is the embedding dimension (and equals to the order),  $r$  is the radius of the neighbourhood (default =  $0.2 \text{ std}(x)$ ),  $C(m+1, r)$  is the number of embedded vectors of length  $m+1$  having a Chebyshev distance inferior to  $r$  and  $C(m, r)$  is the number of embedded vectors of length  $m$  having a Chebyshev distance inferior to  $r$ .

- **Spectral Entropy**

Spectral entropy is defined to be the Shannon Entropy of the Power Spectral Density (PSD) [98] of the data:

$$H(x, sf) = - \sum_{f=0}^{f_s/2} PSD(f) \log_2[PSD(f)] \quad (10.9)$$

Where  $PS$  is the normalised PSD, and  $f_s$  is the sampling frequency.

- **Lumpiness** looks for the series aggregation over some predefined time frame. First, we define a period that is meaningful for recurring seasonal patterns (like day, month or season), for daily, for instance, the period  $P$  is 24 h, then we build the series  $S_n$  with the mean of values on each interval  $P$ . *Lump* then is the standard deviation of the series.

## 10.7 Entropy analysis applied to wind time series

$$S_n = \frac{1}{P} \sum_{i=n}^{P+n} x_i \quad (10.10)$$

$$Lump = \sqrt{\frac{\sum |S_i - \bar{S}|^2}{n}} \quad (10.11)$$

- **Stability** tries to define the stability of the series, is similar to Lumpiness but analysing the variance instead of the mean. The formulation of *Stab* is described in Formula 10.12

$$Stab = \frac{\sum (S_i - \bar{S})^2}{n} \quad (10.12)$$

In this experiment we explore the application of the different indexes to the whole dataset, obtaining a set of values that are illustrated in 10.4

**Table 10.4**

Pearson Correlations between indexes, time horizons and wind speed prediction accuracy measured in  $R^2$  using a CNN separable with 2 layers

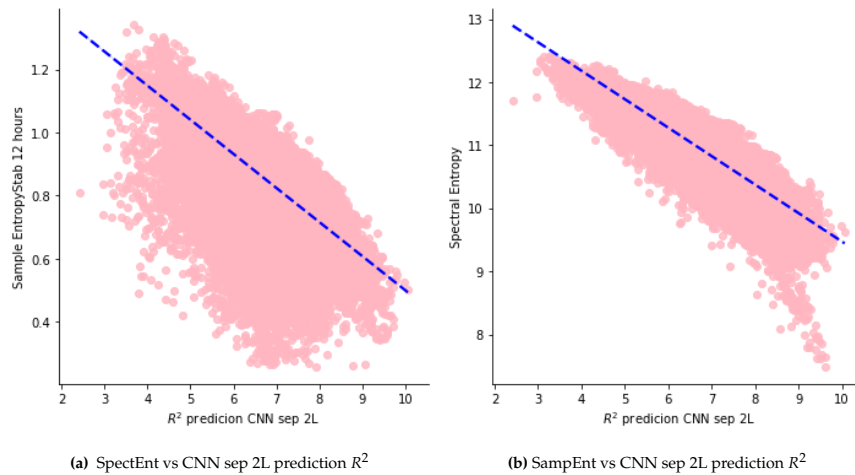
index	global value	12h	24h	1w	1m	3m	6m
Spectral Entropy	-0.835						
Sample Entropy	-0.641						
Lumpiness		0.247	0.282	0.324	0.352	0.346	0.117
Stability		-0.282	-0.058	0.110	0.075	0.038	-0.086

Applying these frequencies to the whole dataset, we can find different correlations between the measures and the accuracy, measured in  $R^2$  of a deep learning prediction with a CNN separable with two layers. The correlation between the spectral entropy and the prediction accuracy is -0.835 and between sample entropy and prediction accuracy is -0.641, both showing a strong relationship between the index and the prediction error. For Lump and Stab analysis, as they are dependent on a determined cycle length (12 hours, 24 hours, one week, one month, three months and six months) we obtain six results for each one. The summary of all the outcomes is represented in Table 10.4.

An illustration of these results can be obtained by generating a scatter-plot with each index and the prediction. In Figure 10.8 we illustrate the Spectral Entropy and Sample Entropy compared with the prediction accuracy measured in  $R^2$  of a CNN separable with 2 Layers, In Figure 10.9 we compare the two indexes, Lumpiness and Stability, for six periods each (12 hours, 24 hours, one week, one month, three months and six months)

If we analyse the Spectral Entropy and the Sample Entropy correlations, we observe that both have correlation with the prediction accuracy. In Figure 10.8a in the Sample Entropy subplot, we observe a set of points with a low error in the original algorithm that seem off

## Wind time series forecastability



**Fig. 10.8** Scatter-plots between wind speed prediction accuracy measured in  $R^2$  using a CNN separable with 2 layers with indexes SampEnt and SpecEnt.

with the regression, analysing these points we see that they not correspond to an specific area but they are distributed evenly in the geography.

**Table 10.5**

Correlation between wind speed prediction accuracy measured in  $R^2$  using a CNN separable with 2 layers with indexes SampEnt and SpecEnt for all the US states.

State	SampEnt	SpecEnt	Sites	State	SampEnt	SpecEnt	Sites	State	SampEnt	SpecEnt	Sites
Alabama	-0.805	-0.405	89	Louisiana	-0.93	-0.84	558	Ohio	-0.505	-0.65	2936
Alaska			0	Maine	-0.842	-0.872	1201	Oklahoma	-0.951	-0.901	4303
Arizona	-0.383	-0.799	2376	Maryland	-0.774	-0.888	334	Oregon	-0.412	-0.808	2636
Arkansas	-0.761	-0.576	823	Massachus.	-0.806	-0.828	676	Pennsylvania	-0.692	-0.836	1502
California	-0.499	-0.716	4053	Michigan	-0.297	-0.818	3609	Rhode Island	-0.601	-0.909	189
Colorado	-0.488	-0.863	3652	Minnesota	-0.661	-0.795	6372	S.Carolina	-0.777	-0.351	271
Connect	-0.779	-0.955	156	Mississippi	-0.886	-0.27	187	S. Dakota	-0.847	-0.879	6012
Delaware	-0.889	-0.619	151	Missouri	-0.7	-0.842	1836	Tennessee	-0.674	-0.856	287
Florida	-0.651	-0.776	1040	Montana	-0.664	-0.867	4091	Texas	-0.635	-0.857	8061
Georgia	-0.513	-0.454	273	Nebraska	-0.902	-0.879	3708	Utah	-0.395	-0.817	1715
Hawaii			0	Nevada	-0.327	-0.721	3865	Vermont	-0.833	-0.781	443
Idaho	-0.518	-0.884	1451	N Hampsire	-0.606	-0.709	409	Virginia	-0.709	-0.761	869
Illinois	-0.417	-0.538	5253	New Jersey	-0.828	-0.681	483	Washington	-0.042	-0.796	1632
Indiana	-0.755	-0.89	3195	N Mexico	-0.703	-0.872	6584	W Virginia	-0.819	-0.89	545
Iowa	-0.604	-0.792	5253	New York	-0.708	-0.829	3156	Wisconsin	-0.427	-0.749	2883
Kansas	-0.92	-0.882	4150	N. Carolina	-0.9	-0.55	849	Wyoming -	-0.632	-0.851	8201
Kentucky	-0.79	-0.617	347	N. Dakota	-0.671	-0.735	3380	Off-shore	-0.317	-0.463	9438

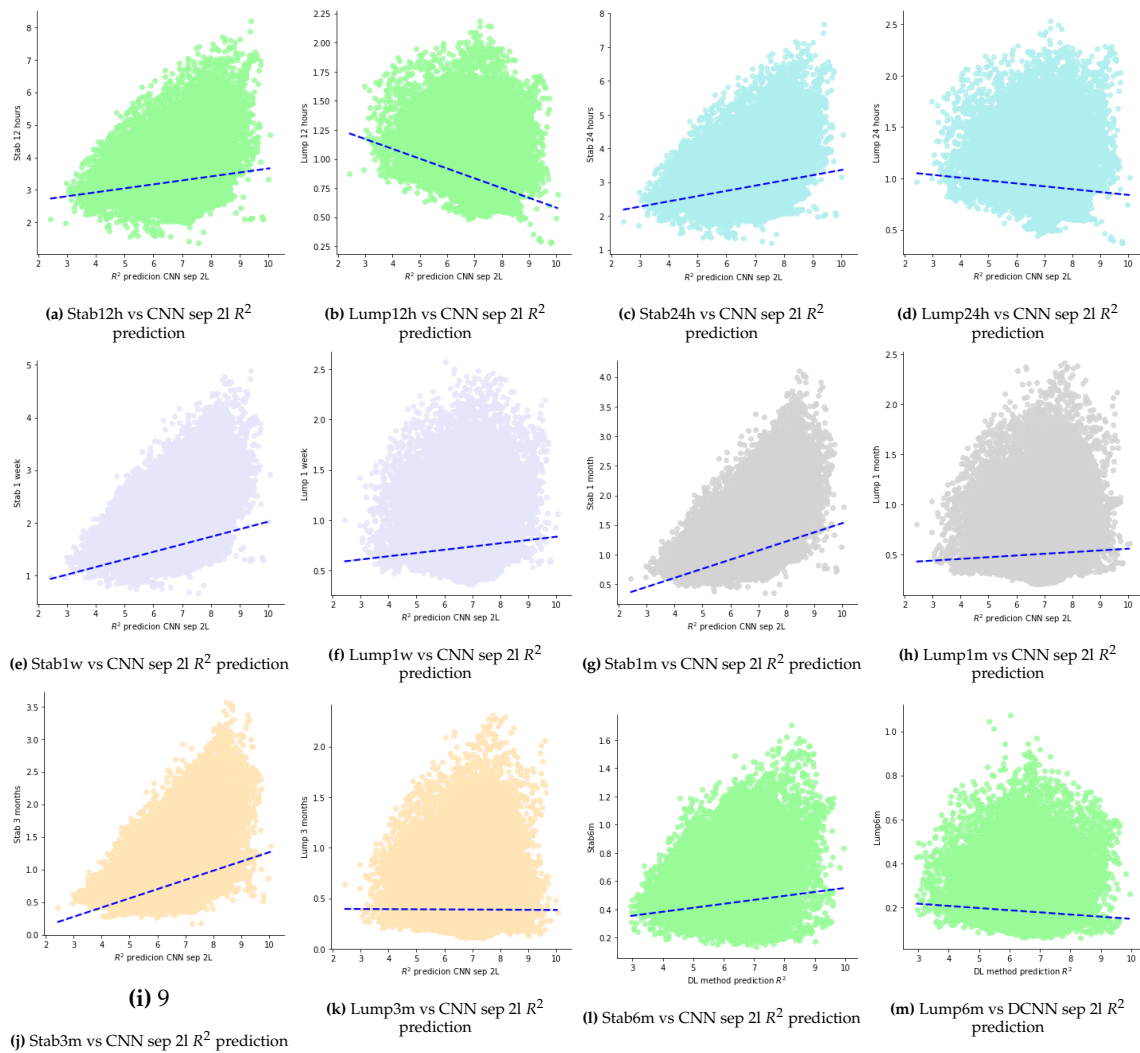
To relate the results with the geography, we experimented with analysing the values per site. Table 10.5.

The results show that there is no correlation between the two indexes and can be complementary. For instance, in Arkansas, or Alabama Sample Entropy has a better correlation than Spectral Entropy, while the inverse happens in Utah or Wyoming, where Spectral Entropy is much better correlated with prediction. This table can be useful to understand

## 10.7 Entropy analysis applied to wind time series

the relationships between wind typologies and the indexes, to develop a combination of them that achieves the maximum forecastability.

Lumpiness and stability are two indexes that try to explain two properties in the wind time series, the aggregation capability over some time and the stability of the series over a period as well (See equations 10.11 and 10.12). In Table 10.4, we can see that both have a much lower correlation than other indexes, with lumpiness a with better correlation. In Figure 10.9, we illustrate how both indexes compare side by side with each other.



**Fig. 10.9**

Scatter-plots between wind speed prediction accuracy measured in  $R^2$  using a CNN separable with 2 layers with indexes stability (Stab) and lumpiness (Lump) for different periods (12h, 24h, 1w, 1m, 3m, 6m)

From the representations, we see low correlations in the scatter-plots as the points are concentrated around a regression line. Nevertheless, lumpiness shows better ability to characterise forecastability.

### 10.8 Discussion

Forecastability is an exciting nascent area of research that can be valuable for the wind prediction field. In this experimentation, we have analysed existing and novel measures that can become forecastability indexes.

The  $f_T$  and  $f_S$  indexes come from the time series decomposition. The  $f_T$  shows the strength of the trend and  $f_S$  the strength of the seasonality.

We have discarded ruggedness as an isolated measure as it is not a good predictor of the complexity of the prediction, elevation, in the other hand, which is a physical descriptor has some value for predictability, as its correlation with an accuracy of deep learning methods of around 0.5.

The entropy which can be sample and spectral entropy, shows potential, especially in the spectral entropy, that obtains a Pearson correlation of -0.835 for the whole dataset while sample entropy has a -0.641 as overall correlation. Focusing on particular states, we can see how the correlation increases, (for instance sample entropy in Oklahoma, with over 4,000 sites is -0.951 or spectral entropy in Indiana is -0.89). Spectral entropy is consistently higher with fewer oscillations across the geography while sample entropy has ups and downs.

We observe that spectral entropy captures a general trend in the time series while sample entropy does it with some specific traits, hence shows a very high correlation in specific places, while in others it has low values.

Lumpiness and stability are two measures that have low correlation but can capture some additional features from the time series, possibly by combination with more general indexes.

## Part IV

# Conclusions & Future Work

---

*"The sun comes up just about as often as it goes down, in the long run, but this doesn't make its motion random"*

1969 - Donald Knuth [111]

*If one way be better than another, that you may be sure is Nature's way*

(4th century B.C.) — Aristotle [4]





# Chapter 11

## Conclusions

This research work, guided by the questions stated in chapter 1, has tried to shed some light to the applicability of deep learning to the wind forecasting problem. After a long and intense experimentation process, we can answer to some of the initial inquiries, but by going in-depth into the problem, we acquired new perspectives that have generated further questions and the definition of alternative approaches.

Research is a continuous process, not always linear, in a never-ending cycle of checking and re-checking hypothesis with findings. We started this work with some initial questions in mind, that have grown along the way. Now is time to wrap-up the results, and to assimilate the lessons learnt while drafting future works for new questions.

This Chapter summarises our findings, relating them to the original questions laid out in the introduction (see Section 1.1).

### 11.1 Introduction

The primary research question for this thesis is:

**Can deep learning improve the accuracy of traditional prediction methods on wind time series for Wind forecasting?** This is the central motivation for this work.

We focused the work in a specific area of research that is the multi-step prediction on wind time series, traditionally, a prediction is based on single-step strategies, but we believe that multi-step is an area well suited for the deep learning algorithms characteristics, motivating us to go deeper into this area.

This thesis research tries to determine the applicability of deep learning algorithms to forecast wind speed time series. In the current literature, we find a lack of general studies on wind time series, as the access to real wind time series, generated from real observations, is very difficult. The studies on this area rely only on findings from a limited set of locations and small sets of data, generating a lack of generalisation of the results.

This work tries to avoid the *anecdotal evidence* bias (that comes from testing approaches in only a few time series) and uses a collection of data that represents all the possible wind topologies in a vast geographical area (the U.S. in this case). The sheer size of this data increases the complexity of the experimentation and requires the use of considerable resources, but the obtained findings from such a significant time series dataset are reliable and consistent.

We confirmed conclusions from hypothesis laid out by other researchers in previous works, and others from our contribution to this area, like the convolutional approach or the comprehensive exploration of the effect of the different architectural features in the deep learning algorithms. However, we leave some open questions that we describe in the future work chapter (see Chapter 12)

In the next sections, we analyse all the conclusions, providing the facts and findings that support them.

### 11.2 Conclusions about the deep learning architectures

In this section, we review the conclusions related to the deep learning architectures. The findings here are related to the application of deep learning to forecasting.

#### 11.2.1 Deep learning is more efficient than baselines

We use baselines to use them as a starting point for comparison. In wind prediction, the first baseline to be used is persistence, which is accurate for short term prediction (up to three hours) but highly inaccurate for more extended periods [73], a statement that we have

## 11.2 Conclusions about the deep learning architectures

verified, as the results obtained in the whole dataset have been low ( $R^2$  accumulated for the twelve steps is 2.699) (see Section 7.2.1).

The second family of algorithms applied are combinations of linear AR, MA and ARIMA methods. The results obtained have been irregular, with irrelevant results across the whole dataset, the conclusion, like persistence, is that they are not adequate to act as baseline methods.

Finally, the random forest and  $k$ -NN methods have shown good results that allow them to act as baseline methods. Random Forest, with a cumulative value of  $R^2$  of 6.770, and  $k$ -NN with a value of 4.675. These are the values that we have considered as reference values for baseline comparison.

Then, we can quickly observe in Appendix A that all the deep learning methods experimented offer results above the baselines, which allows us to conclude the supremacy of the deep learning models over the baseline ones.

The consistency of the results show a better prediction made by the deep learning methods than the baseline methods, showing, particularly for a prediction 12 hours ahead, better learning capabilities than more straightforward machine learning methods like Random Forest or  $k$ -Nearest Neighbours.

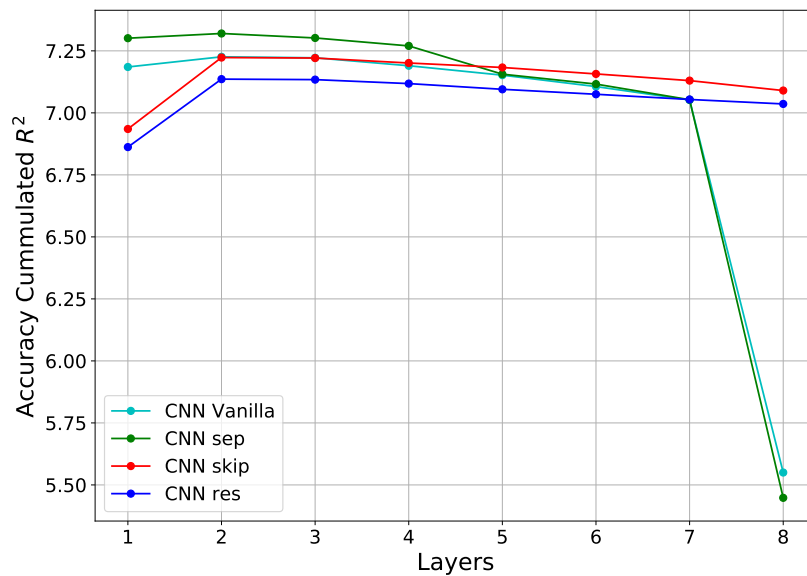


Fig. 11.1 Accuracy depth relationship in four CNN architectures

### 11.2.2 Increase of depth does not correlate with accuracy

Deep networks have better representation capabilities than shallow ones, a general rule based on the principle that deep networks, with higher internal complexity, can represent richer mappings of input to outputs.

However, the experimentation shows that the *best* architectures for wind prediction have two or three layers, as deeper networks do not show accuracy improvements over the shallower approaches.

In Figure 11.1, we compare the accuracy of several convolutional architectures with depth from one layer to eight layers. The residual and skip architectures maintain the accuracy in the deeper models, as skip and residual avoid the vanishing gradient issue, however, with separable and standard convolutional the gradient deteriorates significantly, and the accuracy plummets from the seventh layer.

### 11.2.3 Convolutional separable networks are superior for wind forecasting

After the experiments with all the different architectures, the results have shown that the separable convolutional layers obtain better accuracy for wind prediction (see Appendix A). These models obtain better results than any other, showing how the two-step separable operation is the most appropriate architectures for this task.

Hyper-parameter setting optimisation obtains remarkable improvements from random parameter setting strategies.

The experiments show the strength of the separable convolution function with significant advantages; it offers superior representation capabilities compared to the classic convolutional operation and is more efficient. In this way, the best results from the experiments come with the application of separable one-dimensional convolutions.

These findings replicate what the creators of the separable convolution operation described in their article [34], in this study, they defined the separable convolution operation and applied it to images. In that research article, it was with a two-dimensional convolution. The author, François Chollet, reports improvements in accuracy with better computing resources efficiency.

When we started this research, we expected the RNN architectures, either with LSTM or GRU cells to outperform other architectures in wind prediction, as they are widely used for modelling temporal sequences. However, the results have not confirmed this hypothesis as their results have been inferior to the convolutional approaches. One of the reasons for this conclusion lies in the wind sequence input length, which, after the experimentation, we have concluded that is always short.

After performing extensive hyper-parameter searching, we have concluded that the deep learning algorithms learn wind patterns from short wind series sequences, defining short as intervals of six to eighteen hours, using longer sequences does not increase the

accuracy of the results. This fact makes the sequence memory mechanism, inherent to the RNN, ineffective, as they work better with longer sequences. As the series used for experimentation have a step length of one hour (averaging the original five minutes series), there is still an open question regarding the effectiveness of RNN with series with higher frequencies. Will the RNN still perform worse than the CNN? Alternatively, with series with more data steps, the LSTM or GRU cells will bring better representation capabilities for the wind sequences. We offer a full discussion about wind time-series sequence length in Section 11.3.1.

## 11.3 Conclusions about the wind time-series

### 11.3.1 The wind input sequence length used for learning examples is short

The wind formation theory tells that wind is the result of global and local atmospheric interactions, in events that show changes in the atmosphere in short cycles of three to six hours [117].

In the experiments from Chapters 7, 8 and 9 (for a summary of the experiments results see Appendix A) we consistently obtain as an outcome an optimal input sequence length that was short. It varies from architecture to architecture (see Table 11.1) but is always a length between 6 and 18 time steps (hours).

**Table 11.1**  
*Lag* length training examples

Architecture	length	Architecture	length	Architecture	length
CNN 1L	12	CNN-sep 1L	12	CNN-skip 1L	12
CNN 2L	12	CNN-sep 2L	12	CNN-skip 2L	12
CNN 3L	6	CNN-sep 3L	12	CNN-skip 3L	12
CNN 4L	6	CNN-sep 4L	12	CNN-skip 4L	6
CNN 5L	12	CNN-sep 5L	12	CNN-skip 5L	6
CNN 6L	12	CNN-sep 6L	12	CNN-skip 6L	6
MLP	12	MLP-rec	12	MLP-fut	12
MLP-cas	12	MLP-sjoint	12	MLP-dir-reg	12
RNN MIMO	18	RNN-ED	12	RNN-Att	12

This input length, named *lag* in this thesis, is a parameter adjusted in the hyperparameter setting phase. Moreover, we can obtain the following conclusions:

- The *lag* used in the experiments is always between 6 and 18 steps, as longer sequences did not improve the accuracy.
- Accuracy differences between architectures trained with sequences of different lengths (if the lengths are between the 6 and 18 margins) are small.

## Conclusions

---

- From the experiments, we can conclude that an optimal *lag* length of sequences is between six and eighteen steps. Longer sequences do not show any accuracy improvement.

With all these facts, we can conclude that the learning algorithms, when used with wind, learn from sequences with short *lag*.

### 11.3.2 Higher frequency measurements improve accuracy

The data used in this work came from the NREL dataset (see Chapter 5) and sampled initially with a frequency of five minutes. For easier data handling, the data is averaged hourly and z-standardised before being used for neural network training.

This decision was taken for practical purposes as we reduce the amount of data and the training time for each model, however, we are aware that working with this approach we are reducing the potential best accuracy of the models.

This improvement happens across different architectures, we have shown better results with some MLP and CNN models (see Tables A.3 and 11.2).

**Table 11.2**

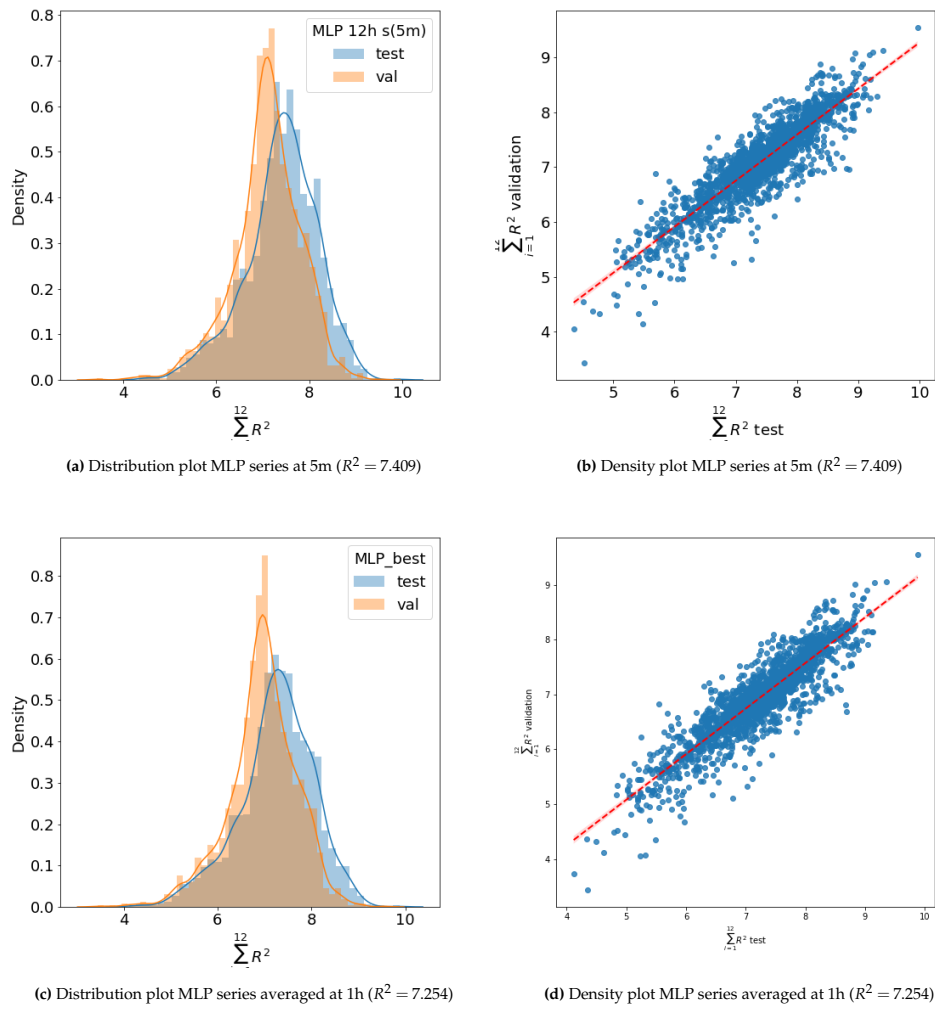
Comparison of Means of probability distributions of 5m series and 1h series on the test data. Improvement calculated from the difference of means

Architecture	Slice/layers	5m	Avg. 1h	Improvement
MLP		7.409	7.254	↑ 2%
MLP future		8.026	7.806	↑ 3%
CNN classic	1L	7.373	7.185	↑ 3%
CNN classic	2L	7.524	7.226	↑ 4%
CNN-sep	1L	7.524	7.301	↑ 3%
CNN-sep	2L	7.507	7.320	↑ 2%

We measured the increase of accuracy, that is, for the six experiments, around a 3% increase, which is a relevant improvement.

With this limited experimentation, we cannot answer the general question on how much the higher frequency series improve the prediction, but in the case of one hour to five minutes, there is a significant improvement.

As future work, we leave the question open on which is the best-sampled period for the series to maximise the prediction result. The limitation issue, though, is the availability of series sampled at the different lengths and for enough different sites.

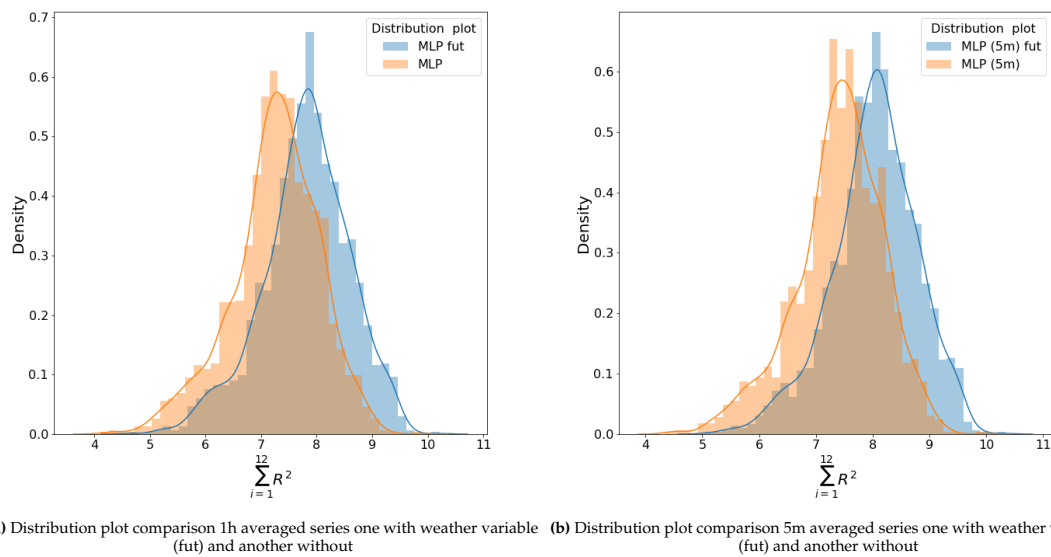


**Fig. 11.2** Comparison of  $R^2$  results with series averaged at 1h (bottom) and with series sampled at 5 minute (top)

## Conclusions

### 11.3.3 Integrating time-series with meteorological information increases accuracy

The research strategy in this work is a multi-step forecasting based exclusively in time series information, with a prediction horizon of twelve hours. Twelve hours marks a border in the prediction window where the models can benefit using weather forecasting information (see chapter 4.1).



**Fig. 11.3** Comparison between architectures with and without a low resolution variable, improvement of accuracy happens in series with step at five minutes and averaged at one hour

Weather forecasting offers predictions with different level of uncertainty depending on the predicted index. Some predictions like temperature or solar radiation can be predicted with very high accuracy as they are similar in very wide areas. In this way, the weather forecasting methods can use a wide grid for their forecasting and obtain a precise measure, notwithstanding other indexes depend on local geographical features, like rainfall or particularly wind, which depends on small terrain attributes. For this reason, a good wind prediction based on weather forecast requires small grids, grids that are not yet available for the largest models requiring to use down-sampling techniques that increment the uncertainty of the result [3].

In this way, we divide the measures to forecast between *low resolution*, which are the indexes that can be predicted with large grids, like temperature or *high resolution* the ones that require small grids, like wind-speed. *Low resolution* are easier to predict and have low uncertainty while *high resolution* are more costly to predict and have higher uncertainty.

Our approach is to add to the time series a prediction like temperature, which is a *low resolution* variable. In this way the networks are trained with the time series information



plus, some additional input on the temperature prediction for the *lag* length, this experiment is carried out with series averaged hourly and with series averaged every five minutes. (see Sections 8.5).

The results (see Figure 11.3) show how the accuracy noticeably increases when adding temperature prediction as an input variable. The accuracy improvements appear in both, the five minutes and the hourly series with improvements of 7%-8% (see Table 8.9).

### 11.4 Conclusions on site forecastability

In Chapter 10, we have developed some theory and application of the notion of forecastability measures applied to wind.

Predictors of complexity for a time series can add value to the field as they can help to understand the expected accuracy for a method and can even be introduced into new forecasting methods that adapt to the identified complexity features.

In our work, we have defined two different approaches to describe the series, by analysing its decomposition elements or by applying signal processing approaches like entropy analysis (see Sections 10.6 and 2.5).

#### 11.4.1 Site accuracy creates clusters of sites with same wind characteristics

Plotting the accuracy  $R^2$  of the results of a deep learning method generates illustrations that show groups of sites signalling the wind formation characteristics on the maps, we see these distributions in these Figures 9.3, 7.7 or 6.5 of different deep learning methods applied to the whole dataset.

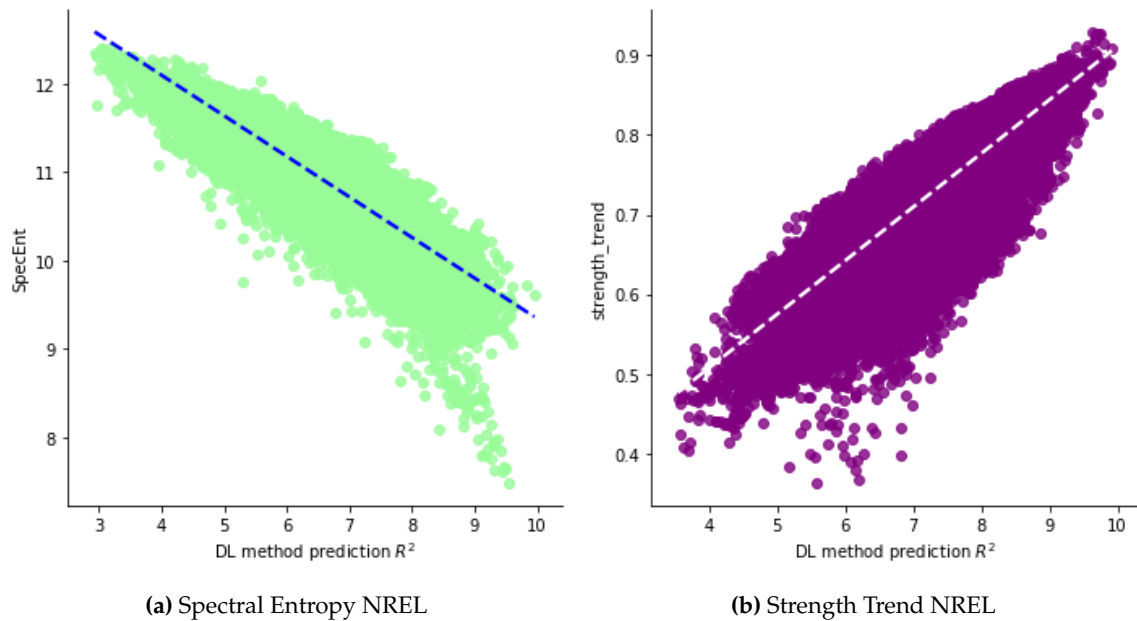
The representation shows how the accuracy is related to inner time series characteristics. In Section 10.7 and 10.6 we analysed, using tools from signal analysis and statistical decomposition, several features that allow defining indexes that have a high correlation with the accuracy of the deep learning results when applied to the wind series. Using these techniques, we can describe the internal structure of the time series advancing in the understanding of how to best adapt the forecasting methods to these structures.

In the maps, we see that using indexes obtained from the internal structure of the time series. We obtain insights that describe how are the wind patterns in a wide area. These predictor indexes can be used to define more powerful learning algorithms as they offer information on how to approach the prediction problem before training the algorithm.

#### 11.4.2 Ruggedness as an isolated measure is not a good predictor

Terrain complexity is commonly associated with difficult a prediction but lacking a definition for this complexity; it is usually is associated with mountainous or steep terrain. To verify

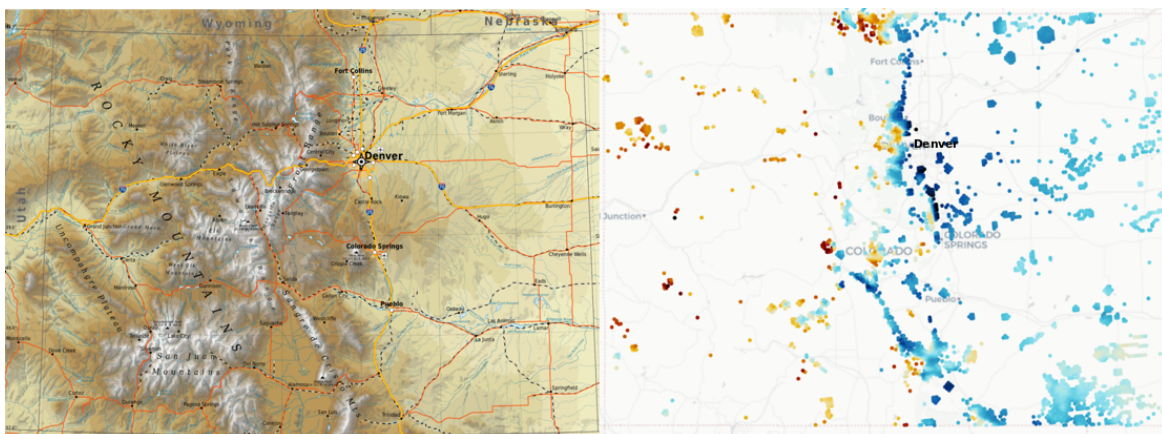
## Conclusions



**Fig. 11.4** Spectral Entropy and Strength Trend for the NREL dataset

this common understanding, we explored correlations between terrain ruggedness features and predictability in (See Figure 10.5).

The result of the experiments did not find a correlation between isolated terrain measures and forecastability. Site predictability cannot be defined by its ruggedness as an isolated measure, as wind seasonality is much more relevant than the mountainous location of a site, as sites in flat terrain can have inferior forecastability than sites in steep areas (see Figure 11.5). Wind seasonality or local gusts are more relevant to obtain suitable or weak predictions than isolated ruggedness measures.



**Fig. 11.5** Comparison of a topographic map of Colorado with a site prediction result with a CNN separable two-layers. Blue is less accurate and Red is more accurate. The lowest accuracy is in the East, at the plains, and in sites in the Rockies (West) the accuracy is higher. (Topographic map source: Topography: ETOPO1; Public domain data provided by the National Atlas of the United States of America)

### 11.5 Conclusions summary

This research that integrates the wind forecasting and the deep learning disciplines offers several contributions.

We confirmed the relevance of deep learning methods to predict wind from time series speed by applying the algorithms to a wide range of wind sites (see Chapters 7, 8, 9). Through this process, we understood the impact and effect of the deep learning architectures on wind that allowed us to design better-performing approaches (see Section 9.8) with outcomes in areas like which are the best fitting parameters, and their impact on the accuracy (see Section 9.4) or a better understanding of the dynamics of the architecture depth on the wind prediction task (see Section 11.2.2).

We offer insights in the internal structure of the wind series specifically in the length that is useful for the learning algorithms 11.3.1, and on the optimal period for the time series data sampling (see Section 11.3.2), and we set the ground for the inclusion of inputs from weather models in the process (see Section 11.3.3).

Altogether, this is a first step in understanding better wind and deep learning, and this step has allowed us to define several future development areas for future work described in the final chapter 12.



## Chapter 12

# Future Work

The lessons learned from this research work allow us to foresee future lines of work. These future developments envisioned, are based on the essential principles that this research contributed to generate, like the application of deep learning and the search of generalisation in the forecasting by applying it to a diverse group of wind time series.

Some of these new approaches are already catching up the attention of the scientific community and will undoubtedly impact the wind forecasting industry in the future. In this chapter, we review the future areas of research that we have identified during this work, that cover different angles of the wind forecasting.

### 12.1 Introduction

The problem of wind prediction using deep learning algorithms is far from solved. The combination of the inherent complexity of wind formation, plus the demanding requirements from the electricity generation industry increase the challenge of this activity.

For deep learning to advance in this application, we foresee future advancements in different lines of work, either by the development of new algorithms or approaches or using more and better data. It is common knowledge in the field that more data improves the results of learning algorithms, and the wind forecasting problem follows this rule. We require more extended sets of data generated at a higher frequency.

As new future lines of work, we propose the use of time series at a higher frequency, the integration of Meteorological info as input to the models, the use of ensembles of methods, the forecasting of local wind events (like ramps, tornadoes) and the use of neighbouring sites to forecast in wider areas.

### 12.2 Use of data from real observation and higher frequency

The original time series used for the experimentation are series sampled at five-minute steps, and to make them better suited for the experiments we have averaged them hourly creating series with one-hour steps. We observed how the series averaged at five-minute steps help the algorithms to increase their accuracy, a clear signal of the principle "*more data generates better accuracy*" [45].

To go deeper into this point we need series with higher frequency, and with very short periods, the series will contain sudden changes of wind, like gusts, that can be very informative to the algorithms. The work on time series with periods of two or three seconds, and coming from unfiltered observations (without outlier smoothing) contain useful information that can be processed by the deep learning algorithms and improve the overall accuracy at mid and short term forecasting.

### 12.3 Integration of NWP values in prediction

The title of this thesis clearly defines the process followed as "multi-step time series forecasting", a method that uses the past predictions to forecast future sequences in the future.

As future work, we propose to add as an input for the deep learning architectures meteorological information from weather model forecasts. This hybrid approach will obtain better accuracy results while increasing the complexity of the inputs in several dimensions:

- The NWP models have stochastic errors induced into the data by the model itself or by the original measurements.

- Meteorological models require down-sampling strategies, which consist in run the models in large areas and then use algorithms that transform this wide Prediction into a local one [249].

In the same way that there are applications of deep learning that can self-generate information from a blurred origin (sharpening an image or painting an old black and white photo) we believe that applying deep learning to down-sampled forecasts with uncertainty can improve the accuracy of the results and apply these models to long term predictions (over 12 hours). This is an entirely new area of research, with limited contributions [179] and potentially with high impact in the wind prediction field.

## 12.4 Use of ensembles

Working with a vast dataset like the NREL wind toolkit allows the observation of the diversity of wind patterns across sites distributed in a wide geographical area.

In this work, we have followed a *one model for all* strategy, which consists in developing deep learning architectures and applying them to the whole dataset with the same set of parameters. The best performing model is the one with the highest average accuracy over the whole spectrum of sites. To improve this accuracy, we can use alternative strategies, to the *one model for all* approach.

The first approach is to develop a specific architecture for each site, by performing a hyper-parameter search using one site, or for a set of sites with similarities. To create the site sets, we can use forecastability measures like the ones analysed in Chapter 10. In this way, the individual performance and overall average will increase, by the development of deep learning architectures that can adapt to the local features of a site to optimise its training.

## 12.5 Prediction of local wind events

A relevant area for future development is the application of anomaly detection to wind time series. Several meteorological phenomena impact the industry and are characterised by the difficulty of Prediction and for causing massive distress in generation assets.

These events are:

- **Wind ramps:** Strong weather events, though rare, cause sudden changes in wind intensity, thus modifying the energy generation profile increasing the complexity of system management by the operators. Wind ramps depend on local features and have different ramping characteristics. Wind prediction is poor in predicting wind ramp events, as wind ramps are a special kind of outlier when wind ramps happen, the TSO requires using contingency generation plants to balance the grid, at a very high cost [55, 47, 65, 66].

- **Local and sudden short meteorological phenomenon:** In this category, we include extremely local events. Hurricanes, Tropical cyclones have a much larger scope and must be predicted at a global level. Local events can be considered a specific kind of ramping event that is generated locally in severe weather situations. The common characteristic is its violence that can break or distort the generation engines completely. One feature for this Prediction is the requirement for using short frequency time series, as long frequency series do not have the resolution to anticipate these changes that happen in minutes [237].
- **Wake effects:** Wake effects is the impact on energy production in a wind park from the wind complex interactions between turbines. The impact of wind with a turbine generates turbulences that impacts the other turbines located downward wind. Wake effects have more to do with aerodynamics than with meteorology but impact the generation and must be considered when predicting the integrated energy output in a wind park. Wake effects are complex and not fully understood, making an interesting field for deep learning modelling [240, 216].

### 12.6 Prediction using neighbouring site information

This research work is based on Prediction at the site level. We use a multivariate time series, and the algorithm is trained for each site. Practically, the wind turbines are not isolated but in sets of towers that share a common area of wind. As we showed in Sections 9.6 and 8.4, using data with higher frequency obtains better results, more data improves the learning process of the algorithms. Predicting at wind park level implies using simultaneously contiguous data points, obtaining much richer data that can help to model the output better. This prediction approach is fully aligned with commercial requirements, but the experimental analysis is scarce, possibly due to the lack of available data. This experimentation needs to explore the impact on the resulting accuracy by using neighbour sites.

There is a need for research on the application of deep learning approaches to large groups of sites to analyse its impact on accuracy, and possibly including the transformation of wind speed into wind energy.

### 12.7 Wind speed probabilistic forecasting using deep learning

The development of probability forecasts for wind speed is a subject area with limited scientific work but with interest from the industry. The International Energy Agency, in their task-force 36, is developing a comprehensive approach for the implementation of probabilistic Prediction in the wind generation industry [74].



## 12.7 Wind speed probabilistic forecasting using deep learning

---

The research group assembled for task-force 36 will develop, in its sub-task 2.3, a complete analysis of the uncertainty origins in wind forecasting through the whole forecasting chain. Probabilistic studies on how the uncertainty propagates are underway to set up a best practice recommendation for the industry forecasters.

The intersection of deep learning, uncertainty and probability forecasting is an exciting area of research that can bring applications of immediate use for the industry. The required tool-sets for this area of research are somehow parallel to this thesis work.

The limited existing literature about the application of deep learning to probability forecasting is surprising due to the intense interest in the industry. However, we observe that the number of possible sources of uncertainty in the forecasting chain increases the complexity of this problem. Adding the inherent uncertainty of the deep learning approach generates a whole new area for research [64].



It is my hope that the research work presented in this thesis will encourage new researchers to push ahead this exciting area of study and make it a central piece in the wind forecasting framework of the industry.





# Part V

## Appendices

---

*"Machines take me by surprise with great frequency"*

(1950) Alan Turing [\[227\]](#)

*"All models are wrong, but some are useful"*

(1919-2013) George E. P. Box



## Appendix A

# Summary of experiments accuracy

This appendix compiles the results of the experiments with all the different architectures. All the values in this page are already included in the different chapter, but for an easier access and comparison we integrated them in three tables.

## Summary of experiments accuracy

**Table A.1**

Mean and standard deviation of probability distributions of  $R^2$  (added for 12 time steps) for all the referenced experiments

Architecture	Slice/layers	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
Persistence		2.699	$\pm 1.951$	2.486	$\pm 1.919$
Random Forest		6.770	$\pm 0.804$	6.532	$\pm 0.737$
$k$ -NN		4.675	$\pm 1.02$	4.430	$\pm 0.941$
MLP Direct Regression		6.955	$\pm 0.787$	6.650	$\pm 0.722$
MLP MIMO 2 Layers		7.254	$\pm 0.792$	6.954	$\pm 0.732$
MLP recursive		7.109	$\pm 0.788$	6.826	$\pm 0.728$
MLP future		7.806	$\pm 0.785$	7.541	$\pm 0.749$
MLP MIMO	1L	7.169	$\pm 0.803$	6.845	$\pm 0.742$
MLP MIMO	2L	7.253	$\pm 0.793$	6.953	$\pm 0.731$
MLP MIMO	3L	7.211	$\pm 0.793$	6.926	$\pm 0.734$
MLP MIMO	4L	7.181	$\pm 0.792$	6.899	$\pm 0.732$
MLP MIMO	5L	7.111	$\pm 0.785$	6.848	$\pm 0.727$
MLP skip	1L	7.170	$\pm 0.801$	6.847	$\pm 0.737$
MLP skip	2L	7.253	$\pm 0.796$	6.948	$\pm 0.734$
MLP skip	3L	7.250	$\pm 0.801$	6.940	$\pm 0.739$
MLP skip	4L	7.233	$\pm 0.800$	6.922	$\pm 0.741$
MLP skip	5L	7.216	$\pm 0.800$	6.903	$\pm 0.744$
MLP SJOINT	1	7.241	$\pm 0.791$	6.938	$\pm 0.733$
MLP SJOINT	2	7.247	$\pm 0.790$	6.943	$\pm 0.731$
MLP SJOINT	3	7.252	$\pm 0.788$	6.947	$\pm 0.731$
MLP SJOINT	4	7.255	$\pm 0.788$	6.949	$\pm 0.730$
MLP SJOINT	6	7.257	$\pm 0.790$	6.953	$\pm 0.730$
MLP SJOINT	12	7.253	$\pm 0.792$	6.951	$\pm 0.731$
CNN 2 layers		7.146	$\pm 0.795$	6.840	$\pm 0.751$
RNN MIMO GRU		7.147	$\pm 0.798$	6.822	$\pm 0.735$
RNN MIMO LSTM		6.952	$\pm 0.805$	6.635	$\pm 0.738$
RNN ED GRU		7.109	$\pm 0.805$	6.786	$\pm 0.749$
RNN ED LSTM		6.965	$\pm 0.807$	6.652	$\pm 0.749$
RNN MIMO GRU att		6.598	$\pm 0.826$	6.290	$\pm 0.743$
RNN MIMO LSTM att		7.003	$\pm 0.804$	6.700	$\pm 0.744$

**Table A.2**  
Mean and standard deviation of probability distributions of  $R^2$  (added for 12 time steps) for all the Convolutional Networks experiments

Architecture	Slice/layers	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
CNN	1L	7.185	$\pm 0.802$	6.867	$\pm 0.754$
CNN	2L	7.226	$\pm 0.800$	6.908	$\pm 0.736$
CNN	3L	7.222	$\pm 0.801$	6.895	$\pm 0.734$
CNN	4L	7.190	$\pm 0.804$	6.858	$\pm 0.733$
CNN	5L	7.152	$\pm 0.810$	6.816	$\pm 0.733$
CNN	6L	7.106	$\pm 0.809$	6.817	$\pm 0.735$
CNN	7L	7.052	$\pm 0.808$	6.750	$\pm 0.734$
CNN	8L	5.450	$\pm 3.043$	5.203	$\pm 0.905$
CNN-sep	1L	7.301	$\pm 0.798$	6.991	$\pm 0.752$
CNN-sep	2L	7.320	$\pm 0.797$	6.988	$\pm 0.745$
CNN-sep	3L	7.302	$\pm 0.796$	6.988	$\pm 0.746$
CNN-sep	4L	7.270	$\pm 0.798$	6.954	$\pm 0.754$
CNN-sep	5L	7.152	$\pm 0.809$	6.816	$\pm 0.735$
CNN-sep	6L	7.106	$\pm 0.809$	6.771	$\pm 0.732$
CNN-sep	7L	7.054	$\pm 0.809$	6.749	$\pm 0.736$
CNN-sep	8L	5.448	$\pm 3.050$	5.207	$\pm 2.916$
CNN-skip	1L	6.935	$\pm 0.799$	6.671	$\pm 0.730$
CNN-skip	2L	7.223	$\pm 0.798$	6.904	$\pm 0.738$
CNN-skip	3L	7.221	$\pm 0.798$	6.903	$\pm 0.738$
CNN-skip	4L	7.201	$\pm 0.806$	6.871	$\pm 0.739$
CNN-skip	5L	7.183	$\pm 0.810$	6.850	$\pm 0.738$
CNN-skip	6L	7.157	$\pm 0.813$	6.820	$\pm 0.739$
CNN-skip	7L	7.130	$\pm 0.814$	6.790	$\pm 0.739$
CNN-skip	8L	7.090	$\pm 0.818$	6.751	$\pm 0.739$
CNN-res	1L	6.862	$\pm 0.796$	6.602	$\pm 0.735$
CNN-res	2L	7.136	$\pm 0.806$	6.834	$\pm 0.739$
CNN-res	3L	7.134	$\pm 0.805$	6.832	$\pm 0.739$
CNN-res	4L	7.118	$\pm 0.804$	6.807	$\pm 0.733$
CNN-res	5L	7.095	$\pm 0.807$	6.785	$\pm 0.736$
CNN-res	6L	7.076	$\pm 0.806$	6.767	$\pm 0.734$
CNN-res	7L	7.055	$\pm 0.810$	6.746	$\pm 0.737$
CNN-res	8L	7.036	$\pm 0.809$	6.728	$\pm 0.738$
CNN-MH 2Heads		7.145	$\pm 0.792$	6.829	$\pm 0.731$
CNN-MH 3Heads		7.170	$\pm 0.796$	6.854	$\pm 0.735$
CNN-MH 4Heads		7.125	$\pm 0.788$	6.816	$\pm 0.728$
CNN-InceptionTime		6.841	$\pm 0.793$	6.549	$\pm 0.743$
<b>CNN - GB</b>	2L	<b>7.228</b>	<b><math>\pm 0.796</math></b>	<b>6.951</b>	<b><math>\pm 0.734</math></b>
<b>CNN sep GB</b>	2L	<b>7.341</b>	<b><math>\pm 0.796</math></b>	<b>7.043</b>	<b><math>\pm 0.753</math></b>

**Table A.3**

Mean and standard deviation of probability distributions of  $R^2$  forecasting Horizon is at 1h (12 steps) or at 12h (144 steps ahead) using 5m series experiments

Architecture	Slice/layers	Test-mean	Test- $\sigma$	Val-mean	Val- $\sigma$
MLP 5m 12h		7.409	$\pm 0.775$	7.095	$\pm 0.724$
MLP 5m 12h future		8.026	$\pm 0.754$	7.748	$\pm 0.733$
Persistence 5m 1h		10.946	$\pm 0.344$	10.749	$\pm 0.436$
MLP 5m 1h		11.123	$\pm 0.267$	10.951	$\pm 0.347$
CNN classic 5m	1L	7.373	$\pm 0.763$	7.077	$\pm 0.710$
<b>CNN classic 5m</b>	<b>2L</b>	<b>7.524</b>	<b><math>\pm 0.761</math></b>	<b>7.212</b>	<b><math>\pm 0.716</math></b>
CNN-sep 5min	1L	7.524	$\pm 0.763$	7.212	$\pm 0.718$
CNN-sep 5min	2L	7.507	$\pm 0.766$	7.211	$\pm 0.716$



## **Appendix B**

# **Accuracy and error**

The accuracy estimation is critical to understand the effectiveness of a forecast. In this chapter, we analyse different error measures used in wind prediction and review some of their main properties.

## Introduction

There is no consensus on which measure is the best to rate a wind forecast. There is a diversity of measures found in the literature that describe the error generated by the prediction algorithms. This diversity the difficulty of comparison between different experiments performed in different turbines, wind parks and geographies. Based on the work of Madsen developed in the framework of the Anemos project [135] and [119], in this section, we describe the most widely used measures of error in the field.

the nomenclature used in the formulas is:

$$\begin{aligned}
 \text{observations} &= \mathbf{X} : \langle \mathbf{x}_1, \dots, \mathbf{x}_n \rangle \\
 \text{real labels} &= \mathbf{Y} : \langle \mathbf{y}_1, \dots, \mathbf{y}_n \rangle \\
 \text{predictions} &= \hat{\mathbf{Y}} : \langle \hat{\mathbf{y}}_{n+1}, \dots, \hat{\mathbf{y}}_{n+H} \rangle \\
 \text{Prediction Horizon} &= H \\
 \text{Error } e_i &= (\hat{y}_i - y_i) \\
 \text{mean } \bar{y} &= \frac{1}{N} \sum_{i=1}^N y_i
 \end{aligned}$$

## Standard deviation and variance

The dispersion of values in a time series calculated to its mean is the standard deviation ( $\sigma$ ).

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2 \tag{B.1}$$

Standard deviation determines the dispersion of a set of values. If it is required to compare two sets of values, for instance, predictions and observations, the variance ( $\sigma^2$ ) of the difference of means is the sum of variances of each respective mean.

Standard deviation is not a measure of error but an indicator of the dispersion of values in a list of values.

## Bias or systematic error

Model Bias is the average error over the whole period, computed for each time step [135].

$$\text{BIAS} = \bar{e}_i = \frac{1}{H} \sum_{i=1}^H e_i \tag{B.2}$$

---

It is considered as the systematic error of the model. In statistical literature, there is a distinction between observational or systematic error and random error; however, for this work, both of them are considered systematic errors in line with [135].

For this work, we have designed a variation of this approach, as the horizon steps are fixed across all models ( $H = 12$ ), the measure used to compare the errors is the cumulative value of errors for the 12 times steps, without averaging it,  $\sum_{i=1}^{H=12} e_i$  [145].

### Mean Error (ME)

Mean error is the average of errors in the prediction. It is seldom used as the negative errors are cancelled by the positive ones and the total does not give valuable information of the accuracy of the result [118].

$$ME = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i) = \frac{1}{N} \sum_{i=1}^N e_i \quad (\text{B.3})$$

For this reason, this measure is not usually used, and it is not present in this work.

### Mean Absolute Error (MAE)

Mean Absolute Error is a fundamental error measure and is the result of the average of differences between prediction and real observations for a specific prediction outcome.

$$MAE = \frac{1}{N} \sum_{i=1}^N |(y_i - \hat{y}_i)| = \frac{1}{N} \sum_{i=1}^N |e_i| \quad (\text{B.4})$$

MAE is a good measure of the prediction quality and is widely present in many works in the literature, however as it is not normalised, sites with larger values of wind speed have larger MAE than sites with smaller wind intensity, as the larger values impact in the calculation of the error more than small values.

### Mean Squared Error (MSE)

Mean squared error is calculated by squaring the errors. The square operation avoids the negatives in the results. By squaring the values penalises large values by magnifying them and favours small values, in this way small values have more significance than with the MAE error measurement

$$MAE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (\text{B.5})$$

## Root mean squared error (RMSE)

The RMSE is the square root of the Mean Square Error (MSE) and shares the advantage of obtaining higher error values, penalizing more significant errors compared to smaller ones.

The formulation of the RMSE (also known as RMS) is:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2} \quad (B.6)$$

There are some recommendations [73] to use different evaluations for the same exercise as some models may generate lower MAE than RMSE or the other way around. In some situations, a positive RMSE can have a corresponding negative MSE.

Squared errors (RMSE, MSE) favour small differences over absolute errors (MSE, ME). However, both measures have the issue that the errors depend on the scale of the numbers, as the sites may have wind speed in different ranges, this affects the error measure, and makes the error comparison between different sites difficult. For this reason, the errors are normalised. (see Section B)

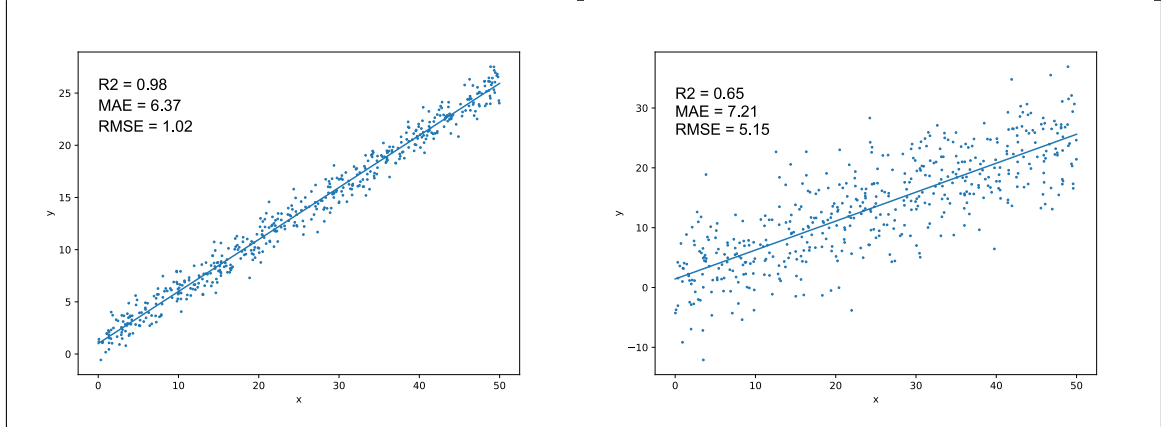
## Coefficient of determination $R^2$

An relevant measure is the  $R^2$  used to describe the degree of fitness of a regression. This measure is an accuracy measure (higher more accurate) while MSE is an error measure (higher more error).

This measure is in a range of  $[0,1]$  but for a regression that is outside the target can generate negative values as well. The  $R^2$  has high semantic content as it shows how a regression fits a function (see Figure B.1). The asymptotic variance of the prediction errors is more significant than the mean-variance, and this can cause negative error values for wrong predictions. For instance, in the persistence method, the variance of the prediction errors is twice the variance of the mean prediction [158].

$$R^2 = 1 - \frac{\sum_{i=1}^H (y_i - \hat{y}_i)^2}{\sum_{i=1}^H (y_i - \bar{y})^2} = 1 - \frac{SS_{RES}}{SS_{TOT}} \quad (B.7)$$

The coefficient of determination is a useful for method comparison, but rarely used in the wind prediction literature.



**Fig. B.1** Two Regression exercises with their  $R^2$  errors. It can be seen how the point cloud shape influences the resultant  $R^2$  value

## Mean Absolute Percentage Error (MAPE)

Very similar approach to MAE, but obtains the result as a percentage, and is defined by the formula:

$$MAPE = \frac{1}{N} \sum_{i=1}^N \frac{|y_i - \hat{y}_i|}{y_i} = \frac{1}{N} \sum_{i=1}^N \frac{|e_i|}{y_i} \quad (B.8)$$

It weights the absolute value of the error by the size of the measure, which helps for comparison of measures. It can be found in some works in the literature, but is as not widely used as the standard MAE and RMSE.

## sMAPE: Symmetrical MAPE

MAPE is a percentage measure as it relates the error with a percentage. It can be multiplied by 100 in order to obtain a number between  $-200$  and  $+200$  or just a number between 0 and 1. This measure tries to avoid the problem of significant errors when the actual wind power values are close to zero, by dividing the error by the measures added to the average.

$$sMAPE = \frac{2}{N} \sum_{i=1}^N \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|} \cdot 100\% \quad (B.9)$$

This measure, corresponds to the one proposed by [Makridakis and Hibon](#) in [136], it avoids large errors when  $y_i$  is greater than  $\hat{y}_i$  or vice versa. The symmetric measure has bounds when the non-symmetric does not have an upper or lower limit.

This measure has been used in time series forecasting but is not widely present in wind literature.

## Normalised errors (nMAPE, nMAE, nRMSE)

Normalisation is required if we want to compare error measures that are obtained from different sites, as each site has a range of power and wind. Normalisation requires to divide the values by the rated wind-speed or power (depending on the nature of the measurement). The normalisation will be then obtained by dividing the error by the rated power value or rated wind speed value.

$$nMAE = \frac{MAE}{\text{Rated value}} \quad (\text{B.10})$$

$$nRMSE = \frac{RMSE}{\text{Rated value}} \quad (\text{B.11})$$

$$nMSE = \frac{MSE}{\text{Rated value}} \quad (\text{B.12})$$

As the maximum speed may be affected by some events that occur only once every two or three years (intense hurricanes, for example), it is advisable to remove outliers in the calculation of the rated maximum wind in a site.

## Analysis of errors and its usability for wind prediction

The error/accuracy results evaluate model performance, but this analysis is not a straightforward task. A model can show good results with one error metric and worse results with others.

Another problem is to compare different sites and obtain conclusions from the comparison. A commonly used method for this is to use a naive method as a comparison. In the wind prediction world, this method is persistence, which is one of the most natural methods (the actual value is the prediction for all steps). This method is accurate for short term predictions, but after 3-4 hours its error is too high, being easy to beat [73]. For this reason, persistence is not very useful for longer prediction horizons.

The primary metrics used are MAE, RMSE,  $R^2$ , the comparison with baseline models like persistence or others [158], or showing the error distribution as a histogram bias, MAE, RMSE, the coefficient of determination  $R^2$ , the skill score for comparison with other models, and finally showing the error distribution graphically (as histogram or function shape) [135].

A recommendation from a paper [135] by Madsen et al. is to use normalised error metrics (like NMAPE or NRMS) and to use the wind - power prediction is recommended to use the installed capacity as normalisation factor and not the production average, basically because the production average might not be known for new wind-sites. The Spanish wind energy association punishes MAPE as error measure on the forecasts, the model miscalculations in the power schedule are penalised based on MAPE.

---

All-together, there is not a simple answer to the question of which is the best error measure to be used in site evaluation. The behaviour of each measure differs on each site topology, and the model applied. MAE and RMSE are the most used methods, while  $R^2$  is a practical tool for regressions.





## Appendix C

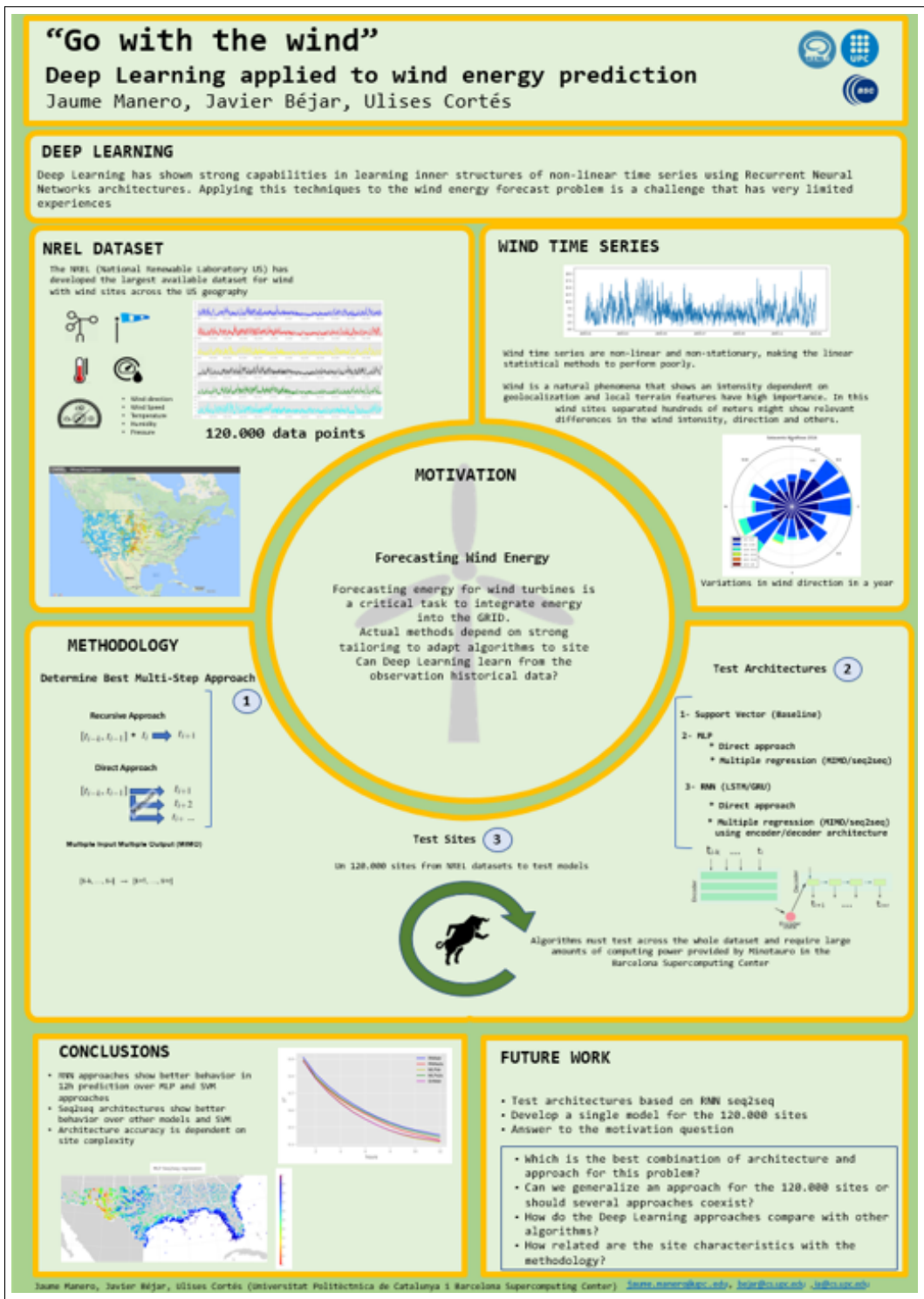
### Posters presented at conferences

The candidate has participated in several conferences where he presented the research in the form of posters. In this appendix we show the contents of the posters displayed on the three conferences.

- (Pos-3) In energy3canada 2019 conference, (Canada conference on energy with a focus on offshore wind), Halifax, Nova Scotia Canada, October 16-18 2019.
- (Pos-2) WindEurope 2019 & Exhibition, (European Wind Industry Conference) Bilbao, Spain, April 2-4 2019.
- (Pos-1) 21<sup>st</sup> International Conference of the Catalan Association for Artificial Intelligence, CCIA 2018 October 8-10, Roses, Girona, Spain

## **CCIA October 2018**

21st International Conference of the Catalan Association for Artificial Intelligence, October 8th-10th 2018 in Roses, Girona Spain



#### CONCLUSIONS

- MIMO approaches show better behavior in 12h prediction over MLP and SVM approaches
- Seq2seq architectures show better behavior over other models and SVM
- Architecture accuracy is dependent on site complexity




#### FUTURE WORK

- Test architectures based on RNN seq2seq
- Develop a single model for the 120.000 sites
- Answer to the motivation question

- Which is the best combination of architecture and approach for this problem?
- Can we generalize an approach for the 120.000 sites or should several approaches coexist?
- How do the Deep Learning approaches compare with other algorithms?
- How related are the site characteristics with the methodology?

Fig. C.1 Poster CCIA 2018 Conference

## **WindEurope Conference April 2019**

WindEurope 2019 & Exhibition, (European Wind Industry Conference) Conference in Bilbao, Spain April 2nd-4th 2019

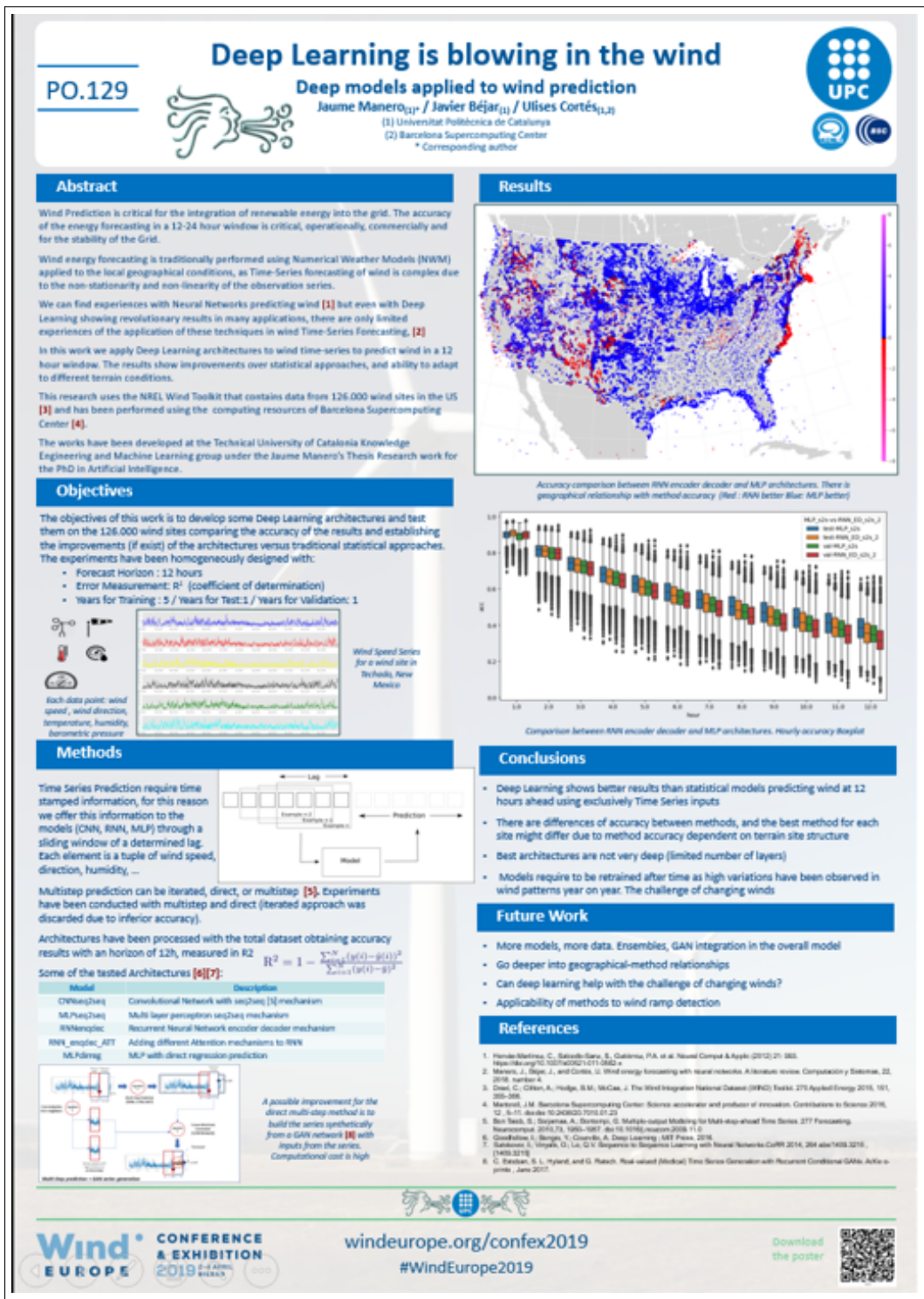



Fig. C.2 Poster Wind2019 Conference

## **Energy3Canada Conference October 2019**

Energy3canada 2019 conference, (Canada conference on energy with a focus on offshore wind), Halifax, Nova Scotia Canada, October 16-18 2019.




**DALHOUSIE UNIVERSITY**

# Wind prediction with Deep Learning

## Predicting Wind Time Series with Neural Networks

Jaume Manero<sup>1,2\*</sup> / Javier Béjar<sup>1,3</sup> / Ulises Cortés<sup>2,3,4</sup>  
<sup>(1)</sup> Universitat Politècnica de Catalunya  
<sup>(2)</sup> Barcelona Supercomputing Center  
<sup>\*</sup> Corresponding author



**UPC**

---

### Abstract

Wind Prediction is critical for the integration of renewable energy into the grid. The accuracy of the energy forecasting in a 12-24 hour window is critical, operationally, commercially and for the stability of the Grid. Wind energy forecasting is traditionally performed using Numerical Weather Models (NWMs) applied to the local geographical conditions, so Time-series forecasting of wind is complex due to the non-stationarity and non-linearity of the observation series.

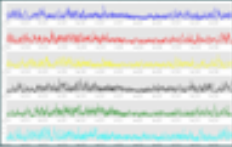
There are limited experiences of Neural Networks predicting wind [1] [2], but Deep Learning has shown interesting results when faced with large sets of data, and this work opens the door for its introduction into the field. This work applies Deep Learning architectures to wind time-series to predict wind in a 12-hour window. The results show improvements over statistical approaches, in sites with very different terrain and wind conditions (Across all U.S.)

This research uses the NREL Wind Toolkit that contains data from 126.692 wind sites in the US [3] and has been performed using the computing resources of the Barcelona Supercomputing Center [4] and is the content of Jaume Manero's Thesis to obtain the title of PhD of Artificial Intelligence with International Certification at the Technical University of Catalonia. He has done a Research visit to Dalhousie University during Summer Term 2019.

### Objectives

The objectives of this work is to develop Deep Learning architectures and test them on the 126.692 wind sites present in the NREL North America dataset in order to analyze, in real life conditions, the feasibility of deep learning as a tool to learn from multivariate series data.

To include Deep Learning as a forecasting tool will allow to increase accuracy and management of Wind as vital renewable energy resource.

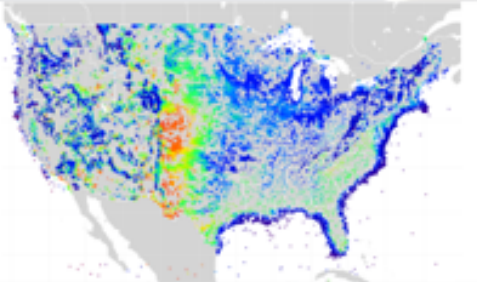



Wind Speed Series for a wind site in Tehuacan, New Mexico

Each data point: wind speed, wind direction, temperature, humidity, barometric pressure

---

### Results

The site complexity for prediction can be established by the seasonality structure of the time series. This is a predictor of the effectiveness of the methods (Red: low stations, blue: more stations)

Methods are better in some areas depending on the site characteristics, so their learning adapts better to some site properties (CNN and MLP in this figure)

### Conclusions


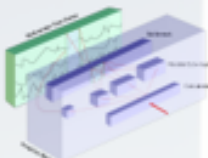
- Deep Learning is suitable for wind prediction
- Lag length is very short – wind time series have short “memory”
- Results are valid with 2,000-3,000 sites don't need 126,000
- Open the possibility of multi-model approaches (hourly-site / dependent)
- High consumption of computing resources, but don't need human supervision or feature engineering

---

### Methods

Fully Connected Networks, Convolutional Networks and Recurrent Neural Networks and combinations of several approaches have been created and tested on Wind Data.

More than 30 different approaches have been defined many of them novel to the wind prediction task

**10 MLP models**

- MLP\_10\_100\_100 - 10 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100 - 100 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100 - 100 - 100 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100
- MLP\_10\_100\_100 - 10 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100

**10 CNN models**

- CNN\_10\_100 - CNN
- CNN\_10\_100 - CNN - CNN
- CNN\_10\_100 - CNN - CNN - CNN
- CNN\_10\_100 - CNN - CNN - CNN - CNN
- CNN\_10\_100 - CNN - CNN - CNN - CNN - CNN
- CNN\_10\_100 - CNN - CNN - CNN - CNN - CNN - CNN
- CNN\_10\_100 - CNN - CNN - CNN - CNN - CNN - CNN - CNN
- CNN\_10\_100 - CNN - CNN - CNN - CNN - CNN - CNN - CNN - CNN
- CNN\_10\_100 - CNN - CNN - CNN - CNN - CNN - CNN - CNN - CNN - CNN
- CNN\_10\_100 - CNN - CNN - CNN - CNN - CNN - CNN - CNN - CNN - CNN - CNN

**10 LSTM models**

- LSTM\_10\_100 - LSTM
- LSTM\_10\_100 - LSTM - LSTM
- LSTM\_10\_100 - LSTM - LSTM - LSTM
- LSTM\_10\_100 - LSTM - LSTM - LSTM - LSTM
- LSTM\_10\_100 - LSTM - LSTM - LSTM - LSTM - LSTM
- LSTM\_10\_100 - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM
- LSTM\_10\_100 - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM
- LSTM\_10\_100 - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM
- LSTM\_10\_100 - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM
- LSTM\_10\_100 - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM - LSTM

**Legend:**

- Good results for the problem
- Medium for the problem
- Bad for the problem
- No convergence
- Hyperparameter optimization
- Not implemented
- Not used

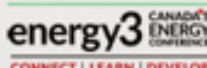
### Future Work

- Go deeper into geographical-method relationships
- Can deep learning help with the challenge of changing winds due to climate change?
- Applicability of methods to wind ramp detection
- Introduce Meteorological prediction data into the methods


---

### References


1. Hernandez-Martinez, C., Garcia-Solis, S., Gutierrez, P.A. et al. Neural Comput & Applic (2017) 29: 865. <https://doi.org/10.1007/s00521-017-2820-2>
2. Hernandez, J., Muga, J., and Cortes, U. Wind energy forecasting with neural networks. A Review article. *Computación y Sistemas*, 20, 2016, number 4.
3. Cortes, U., Gilen, A., Prada, S.M., McCaa, J. The Wind Integration National Dataset (WIND) Toolkit. *21st Applied Energy* 2016, 169, 686-696.
4. Manero, J.M. Barcelona Supercomputing Center: Science accelerator and producer of innovation. *Contributions to Science* 2019, 12, 1-14. [https://doi.org/10.1007/978-3-030-23323-2\\_1](https://doi.org/10.1007/978-3-030-23323-2_1)
5. Béjar, J., Cortés, U., Manero, J.M., Manero, J.M. Multiple-output Modeling for Multi-dimensional Time Series. *21st Forecasting* (NeuroForecasting) 2019, 173, 1660-1667. [https://doi.org/10.1007/978-3-030-23323-2\\_173](https://doi.org/10.1007/978-3-030-23323-2_173)
6. Hernandez, J., Gilen, A., Cortés, U. Deep Learning. *WPI Press*, 2016.
7. Subramani, S., Wang, D., Lu, L.Y. Sequence to Sequence Learning with Neural Networks. *CoRR* 2014, abs/1409.3215, 2014.09.2015.
8. Jaume Manero, Javier Béjar, and Ulises Cortés. "Wind in the wind... Deep learning application to wind energy time series forecasting." *Energy*. 167:1036-1044.



**energy3 CANADA'S ENERGY CONFERENCE**  
CONNECT | LEARN | DEVELOP



**NOVA SCOTIA**  
PRESENTING SPONSOR



**energy3canada.com**  
Halifax 16-18 October

Fig. C.3 Poster Wind2019 Conference





# Appendix D

## Code and Results

There are three resources created around this thesis research work.

- Notebooks with the analysis of the experiment results
- MongoDB database with all the individual experiment results
- Code used in the experimentation

We provide a link to the code used in the thesis for replicability purposes, plus a link to another github repository where all the experiment results cited in the theses are available, plus most of the generated illustrations from data.

The MongoDB database contains all the individual experiment results, and is available under request. Nevertheless, is important to point the complexity to reproduce all the experiments due to the computing requirements required. All the experiments have been performed in 2018, 2019 and 2020, and have been supported by the Barcelona Supercomputing Center.

**NOTEBOOKS:** <https://github.com/castorgit/Articles-2020>

**CODE:** [https://github.com/castorgit/Wind\\_code](https://github.com/castorgit/Wind_code)

**DATABASE:** Mongodb database available under request



# References

- [1] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621, 2010. doi: 10.1080/07474938.2010.481556.
- [2] A. Alsirhani, J. Manero, S. Sampalli, and P. Bodorik. A DDoS detection framework: Deep learning in a distributed system cluster. *tbd*, 2020. Article submitted, pending publication.
- [3] J. R. Andrade and R. J. Bessa. Improving renewable energy forecasting with a grid of numerical weather predictions. *IEEE Transactions on Sustainable Energy*, 8(4):1571–1580, Oct 2017. ISSN 1949-3037. doi: 10.1109/TSTE.2017.2694340.
- [4] Aristotle. *Aristotle: Nicomachean Ethics*. Cambridge Texts in the History of Philosophy. Cambridge University Press, 2000. doi: 10.1017/CBO9780511802058. Crisp, Roger Editor.
- [5] A. F. Atiya, S. M. El-Shoura, S. I. Shaheen, and M. S. El-Sherif. A comparison between neural-network forecasting techniques-case study: river flow forecasting. *IEEE Transactions on Neural Networks*, 10(2):402–409, March 1999. ISSN 1045-9227. doi: 10.1109/72.750569.
- [6] R. Azimi, M. Ghofrani, and M. Ghayekhloo. A hybrid wind power forecasting model based on data mining and wavelets analysis. *Energy Conversion and Management*, 127:208 – 225, 2016. ISSN 0196-8904. doi: <https://doi.org/10.1016/j.enconman.2016.09.002>.
- [7] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [8] J. Badger, N. Davis, A. Hahmann, X. G. Larsen, M. Badger, M. Kelly, B. T. Olsen, N. G. Mortensen, H. E. Joergensen, I. Troen, E. L. Petersen, P. Volker, and J. Lange. The global wind atlas, October 2015. Final Report Technical University of Copenhagen.
- [9] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*, 1 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.
- [10] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.
- [11] S. D. Balkin and J. Ord. Automatic neural network modeling for univariate time series. *International Journal of Forecasting*, 16(4):509 – 515, 2000. ISSN 0169-2070. doi: [https://doi.org/10.1016/S0169-2070\(00\)00072-8](https://doi.org/10.1016/S0169-2070(00)00072-8). The M3- Competition.
- [12] S. Balluff, J. Bendfeld, and S. Krauter. Short term wind and energy prediction for offshore wind farms using neural networks. In *2015 Int. Conf. on Renewable Energy Research and Applications (ICRERA)*, pages 379–382, Nov 2015. doi: 10.1109/ICRERA.2015.7418440.
- [13] T. W. Beers, P. E. Dress, and L. C. Wensel. Notes and Observations: Aspect Transformation in Site Productivity Research. *Journal of Forestry*, 64(10):691–692, 10 1966. ISSN 0022-1201. doi: 10.1093/jof/64.10.691.
- [14] S. Ben Taieb, A. Sorjamaa, and G. Bontempi. Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomput.*, 73(10-12):1950–1957, 06 2010. ISSN 0925-2312. doi: 10.1016/j.neucom.2009.11.030.

## References

---

- [15] S. Ben Taieb, G. Bontempi, A. Atiya, and A. Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *arXiv e-prints*, art. arXiv:1108.3259, Aug 2011.
- [16] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003. ISSN 1532-4435.
- [17] Y. Bengio, N. L. Roux, P. Vincent, O. Delalleau, and P. Marcotte. Convex neural networks. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 123–130. MIT Press, 2006.
- [18] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [19] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, Feb. 2012. ISSN 1532-4435.
- [20] Bob Dylan. *Blowin’ in the wind*, 1963. Sony/ATV Music Publishing.
- [21] G. Bontempi, S. Ben Taieb, and Y.-A. Le Borgne. *Machine Learning Strategies for Time Series Forecasting*, pages 62–77. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-36318-4. doi: 10.1007/978-3-642-36318-4\_3. editor: Aufaure, Marie-Aude and Zimányi, Esteban.
- [22] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis, Fourth Edition*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 2008. ISBN 978111861919. doi: 10.1002/9781118619193. first published 12 June 2008.
- [23] M. Brower. *Wind Resource Assessment: A Practical Guide to Developing a Wind Project*. Wiley, 2012. ISBN 9781118022320.
- [24] B. G. Brown, R. W. Katz, and A. H. Murphy. Time series models to simulate and forecast wind speed and wind power. *Journal of Climate and Applied Meteorology*, 23(8):1184–1195, 1984. doi: 10.1175/1520-0450(1984)023<1184:TSM TSA>2.0.CO;2.
- [25] R. Buizza. Weather risk management with the ECMWF ensemble prediction system. In *EGS Conference on Mediterranean Storms*, 2001.
- [26] Q. Cao, B. T. Ewing, and M. A. Thompson. Forecasting wind speed with recurrent neural networks. *European Journal of Operational Research*, 221(1):148 – 154, 2012. ISSN 0377-2217. doi: <https://doi.org/10.1016/j.ejor.2012.02.042>.
- [27] M. Carandini, J. B. Demb, V. Mante, D. J. Tolhurst, Y. Dan, B. A. Olshausen, J. L. Gallant, and N. C. Rust. Do we know what the early visual system does? *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 25 46:10577–97, 2005.
- [28] C. Carrillo, J. Cidrás, E. Díaz-Dorado, and A. F. Obando-Montaño. An approach to determine the weibull parameters for wind energy analysis: The case of Galicia (Spain). *Energies*, 7(4): 2676–2700, 2014. ISSN 1996-1073. doi: 10.3390/en7042676.
- [29] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. Hasegawa-Johnson, and T. S. Huang. Dilated Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:1710.02224, Oct 2017.
- [30] W.-Y. Chang. A literature review of wind forecasting methods. *Journal of Power and Energy Engineering*, 2:161–168, 2011. doi: <http://dx.doi.org/10.4236/jpee.2014.24023>.
- [31] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16*, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785.
- [32] H. Cheng, P.-N. Tan, J. Gao, and J. Scripps. Multistep-ahead time series prediction. In W.-K. Ng, M. Kitsuregawa, J. Li, and K. Chang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 765–774, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33207-7.
- [33] G. Chevillon. Direct multi-step estimation and forecasting. *Journal of Economic Surveys*, 21(4): 746–785, 2007. doi: 10.1111/j.1467-6419.2007.00518.x.

- [34] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.
- [35] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv e-prints*, art. arXiv:1412.3555, Dec 2014.
- [36] M. Claesen and B. De Moor. Hyperparameter Search in Machine Learning. *arXiv e-prints*, art. arXiv:1502.02127, Feb 2015.
- [37] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ISSN 1573-0565. doi: 10.1007/BF00994018.
- [38] A. Costa, A. Crespo, J. Navarro, G. Lizcano, H. Madsen, and E. Feitosa. A review on the young history of the wind power short-term prediction. *Renewable and Sustainable Energy Reviews*, 12(6):1725 – 1744, 2008. ISSN 1364-0321. doi: <https://doi.org/10.1016/j.rser.2007.01.015>.
- [39] Council of European Union. Council resolution (EU) no 2019/2712/rsp, 2019. [https://oeil.secure.europarl.europa.eu/oeil/popups/ficheprocedure.do?lang=en&reference=2019/2712\(RSP\)](https://oeil.secure.europarl.europa.eu/oeil/popups/ficheprocedure.do?lang=en&reference=2019/2712(RSP)).
- [40] S. F. Crone, M. Hibon, and K. Nikolopoulos. Advances in forecasting with neural networks? empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, 27(3):635 – 660, 2011. ISSN 0169-2070. doi: <https://doi.org/10.1016/j.ijforecast.2011.04.001>. Special Section 1: Forecasting with Artificial Neural Networks and Computational Intelligence Special Section 2: Tourism Forecasting.
- [41] R. L. de Mántaras, U. Cortés, J. Manero, and E. Plaza. Knowledge engineering for a document retrieval system. *Fuzzy Sets and Systems*, 38(2):223–240, 1990. ISSN 0165-0114. doi: [https://doi.org/10.1016/0165-0114\(90\)90151-U](https://doi.org/10.1016/0165-0114(90)90151-U). Fuzzy Information and Database Systems.
- [42] D. Díaz, A. Torres, and J. R. Dorronsoro. Deep neural networks for wind energy prediction. In I. Rojas, G. Joya, and A. Català, editors, *Advances in Computational Intelligence*, pages 430–443, Cham, 2015. Springer International Publishing. ISBN 978-3-319-19258-1.
- [43] E. Díaz-Dorado, C. Carrillo, J. Cidras, and E. Albo. Estimation of energy losses in a wind park. In *2007 9th Int. Conf. on Electrical Power Quality and Utilisation*, pages 1–6, Oct 2007. doi: 10.1109/EPQU.2007.4424125.
- [44] D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427–431, 1979. ISSN 01621459.
- [45] P. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10): 78–87, Oct. 2012. ISSN 0001-0782. doi: 10.1145/2347736.2347755.
- [46] C. Draxl, A. Clifton, B.-M. Hodge, and J. McCaa. The wind integration national dataset (wind) toolkit. *Applied Energy*, 151:355–366, 2015.
- [47] E. Ela and J. Kemper. Wind plant ramping behavior. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2009.
- [48] E. Erdem and J. Shi. ARMA based approaches for forecasting the tuple of wind speed and direction. *Applied Energy*, 88(4):1405 – 1414, 2011. ISSN 0306-2619. doi: <https://doi.org/10.1016/j.apenergy.2010.10.031>.
- [49] B. Ernst, B. Oakleaf, M. L. Ahlstrom, M. Lange, C. Moehrlen, B. Lange, U. Focken, and K. Rohrig. Predicting the wind. *IEEE Power and Energy Magazine*, 5(6):78–89, Nov 2007. ISSN 1540-7977. doi: 10.1109/MPE.2007.906306.
- [50] R. Fares. Energy intermittency explained, challenges, solutions and opportunities. *Scientific American*, pages –, 2015. Nature Publishing Group.
- [51] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller. Deep learning for time series classification: a review. *arXiv e-prints*, art. arXiv:1809.04356, Sep 2018.
- [52] L. Fei-Fei, J. Deng, and K. Li. Constructing a large-scale image database. *Journal of Vision*, 9(8): 1037, 2009. doi: 10.1167/9.8.1037.

## References

---

- [53] 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*, 190:1245 – 1257, 2017. ISSN 0306-2619. doi: <https://doi.org/10.1016/j.apenergy.2017.01.043>.
- [54] C. Feng, M. Sun, M. Cui, E. K. Chartan, B.-M. Hodge, and J. Zhang. Characterizing forecastability of wind sites in the united states. *Renewable Energy*, 133:1352 – 1365, 2019. ISSN 0960-1481. doi: <https://doi.org/10.1016/j.renene.2018.08.085>.
- [55] C. Ferreira, J. Gama, and L. Matias. A survey on wind ramp forecasting, 2011. Argonne National Laboratory report.
- [56] R. M. Ferrer and A. M. Astobiza. Entrevista a Ramón Pérez de Mántaras. *Dilemata*, pages 301–309, 2017. ISSN 1989-7022.
- [57] U. Firat, S. N. Engin, M. Saraclar, and A. B. Ertuzun. Wind speed forecasting based on second order blind identification and autoregressive model. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 686–691, Dec 2010. doi: 10.1109/ICMLA.2010.106.
- [58] A. M. Foley, P. G. Leahy, A. Marvuglia, and E. J. McKeogh. Current methods and advances in forecasting of wind power generation. *Renewable Energy*, 37:1–8, 2011. doi: <http://dx.doi.org/10.1016/j.renene.2011.05.03>.
- [59] L. Fortuna, G. Nunnari, and S. Nunnari. *Nonlinear Modeling of Solar Radiation and Wind Speed Time Series*. Springer Briefs in Energy. Springer, 2016. ISBN 9783319387635. doi: 10.1007/978-3-319-38764-2.
- [60] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.
- [61] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.
- [62] J. H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, Feb. 2002. ISSN 0167-9473. doi: 10.1016/S0167-9473(01)00065-2.
- [63] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.
- [64] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.
- [65] C. Gallego, Á. Cuerva, and A. Costa. Detecting and characterising ramp events in wind power time series. *Journal of Physics: Conference Series*, 555:012040, dec 2014. doi: 10.1088/1742-6596/555/1/012040.
- [66] C. Gallego-Castillo, A. Cuerva-Tejero, and O. Lopez-Garcia. A review on the recent history of wind power ramp forecasting. *Renewable and Sustainable Energy Reviews*, 52:1148–1157, 2015.
- [67] J. C. B. Gamboa. Deep learning for time-series analysis. *CoRR*, abs/1701.01887, 2017.
- [68] M. Gan, H. X. Li, C. L. P. Chen, and L. Chen. A potential method for determining nonlinearity in wind data. *IEEE Power and Energy Technology Systems Journal*, 2(2):74–81, June 2015. doi: 10.1109/JPETS.2015.2424700.
- [69] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning, 2017.
- [70] U. Geological Survey (USGS). 3d elevation program, 2019. URL <https://www.usgs.gov/core-science-systems/ngp/3dep>. online elevations resource accessed November 2019.
- [71] A. Ghaderi, B. M. Sanandaji, and F. Ghaderi. Deep forecast: Deep learning-based spatio-temporal forecasting. *CoRR*, abs/1707.08110, 2017.
- [72] M. Ghofrani, M. de Rezende, R. Azimi, and M. Ghayekhloo. K-means clustering with a new initialization approach for wind power forecasting. In *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T D)*, pages 1–5, 2016.

- [73] G. Giebel, R. Brownsword, G. Kariniotakis, M. Denhard, and C. Draxl. *The State-Of-The-Art in Short-Term Prediction of Wind Power: A Literature Overview, 2nd edition*. ANEMOS.plus, 2011. doi: 10.11581/DTU:00000017. Project funded by the European Commission under the 6th Framework Program, Priority 6.1: Sustainable Energy Systems.
- [74] G. Giebel, J. Cline, H. Frank, W. Shaw, P. Pinson, B.-M. Hodge, G. Kariniotakis, J. Madsen, and C. Möhrlen. Wind power forecasting: Iea wind task 36 & future research issues. In *The Science of Making Torque from Wind 2016, TORQUE 2016*, volume 753. IOP Publishing, 2016. doi: 10.1088/1742-6596/753/3/032042.
- [75] T. Gneiting and M. Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1(1):125–151, 2014. doi: 10.1146/annurev-statistics-062713-085831.
- [76] N. Golyandina and A. Zhigljavsky. *Computational Intelligence in Time Series Forecasting*. Springer-Verlag Berlin Heidelberg, 1 edition, 2013. ISBN 978-3-642-34913-3. doi: 10.1007/978-3-642-34913-3.
- [77] G. Gonzalez, B. Diaz-Guerra, F. Soto, S. Lopez, I. Sanchez, J. Usaola, M. Alonso, and M. G. Lobo. Sipleólico-wind power prediction tool for the spanish peninsular power system. *Proceedings of the CIGRÉ 40th general session and exhibition, Paris, France, 2004*.
- [78] F. González-Longatt, P. Wall, and V. Terzija. Wake effect in wind farm performance: Steady-state and dynamic behavior. *Renewable Energy*, 39(1):329 – 338, 2012. ISSN 0960-1481. doi: <https://doi.org/10.1016/j.renene.2011.08.053>.
- [79] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. ISBN 978-0262035613.
- [80] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24:8–12, 2009.
- [81] S. Hassan, A. Khosravi, and J. Jaafar. Examining performance of aggregation algorithms for neural network-based electricity demand forecasting. *International Journal of Electrical Power & Energy Systems*, 64:1098 – 1105, 2015. ISSN 0142-0615. doi: <https://doi.org/10.1016/j.ijepes.2014.08.025>.
- [82] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009. ISBN 9780387848570.
- [83] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.
- [84] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90.
- [85] J. Heinermann. *Wind Power Prediction with Machine Learning Ensembles*. PhD thesis, University of Oldenburg, 2016.
- [86] J. Heinermann and O. Kramer. Machine learning ensembles for wind power prediction. *Renewable Energy*, 89:671 – 679, 2016. ISSN 0960-1481. doi: <https://doi.org/10.1016/j.renene.2015.11.073>.
- [87] R. Herold. *Managing an Information Security and Privacy Awareness and Training Program*, chapter 9 Identify Training and Awareness Methods page 101. Auerbach Publications, 1 edition, 2005.
- [88] G. E. Hinton and A. K. I. S. N. Srivastva. System and method for addressing overfitting in a neural network, U.S. Patent US9406017B2, Aug. 2016.
- [89] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- [90] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [91] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: A tutorial. *STATISTICAL SCIENCE*, 14(4):382–417, 1999.

## References

---

- [92] R. Hossain, A. M. T. Ooa, and A. B. M. S. Alia. Historical weather data supported hybrid renewable energy forecasting using artificial neural network (ANN). *Energy Procedia*, 14:1035 – 1040, 2012. ISSN 1876-6102. doi: 10.1016/j.egypro.2011.12.1051. 2011 2<sup>nd</sup> Int. Conf. on Advances in Energy Engineering (ICAEE).
- [93] G.-B. Huang, D. Wang, and Y. Lan. Extreme learning machines: a survey. *Int. J. Machine Learning & Cybernetics*, 2:107–122, 2011.
- [94] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. C. Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25566-3.
- [95] P. H. Ibarguengoytia, A. Reyes, I. Romero-Leon, D. Pech, U. A. García, L. E. Sucar, and E. F. Morales. Wind power forecasting using dynamic bayesian models. In A. Gelbukh, F. C. Espinoza, and S. N. Galicia-Haro, editors, *Nature-Inspired Computation and Machine Learning: 13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutiérrez, Mexico, November 16–22, 2014. Proceedings, Part II*, pages 184–197. Springer International Publishing, Cham, 2014. ISBN 978-3-319-13650-9. doi: 10.1007/978-3-319-13650-9\_17.
- [96] INDECIS project. Web site describing structure and objectives project indecis. Technical report, European Union, 2018. URL [www.indecis.eu](http://www.indecis.eu).
- [97] Independent Electricity System Operator (IESO). Power data, 2019. Available: <http://www.ieso.ca/en/power-data/data-directory>. Accessed on Apr. 2, 2019.
- [98] T. Inouye, K. Shinosaki, H. Sakamoto, S. Toi, S. Ukai, A. Iyama, Y. Katsuda, and M. Hirano. Quantification of eeg irregularity by use of the entropy of the power spectrum. *Electroencephalography and Clinical Neurophysiology*, 79(3):204 – 210, 1991. ISSN 0013-4694. doi: [https://doi.org/10.1016/0013-4694\(91\)90138-T](https://doi.org/10.1016/0013-4694(91)90138-T).
- [99] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.
- [100] M. Z. Jacobson, M. A. Cameron, E. M. Hennessy, I. Petkov, C. B. Meyer, T. K. Gambhir, A. T. Maki, K. Pfleeger, H. Clonts, A. L. McEvoy, M. L. Miccioli, A.-K. von Krauland, R. W. Fang, and M. A. Delucchi. 100% clean and renewable wind, water, and sunlight (wws) all-sector energy roadmaps for 53 towns and cities in north america. *Sustainable Cities and Society*, 42:22 – 37, 2018. ISSN 2210-6707. doi: <https://doi.org/10.1016/j.scs.2018.06.031>.
- [101] A. A. Kadhem, N. I. A. Wahab, I. Aris, J. Jasni, and A. N. Abdalla. Advanced wind speed prediction model based on a combination of weibull distribution and an artificial neural network. *Energies*, 10(11), 2017. ISSN 1996-1073. doi: 10.3390/en10111744.
- [102] S. A. P. Kani and G. H. Riahy. A new ANN-based methodology for very short-term wind speed prediction using markov chain approach. In *2008 IEEE Canada Electric Power Conference*, pages 1–6, Oct 2008. doi: 10.1109/EPC.2008.4763386.
- [103] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, 2 edition, 2003. doi: 10.1017/CBO9780511755798.
- [104] G. Kariniotakis. *Renewable Energy Forecasting: From Models to Applications*. Woodhead Publishing Series in Energy. Elsevier - Woodhead Publishing, 2017.
- [105] G. Kariniotakis, I. Marti, D. Casas, P. Pinson, T. Nielsen, H. Madsen, G. Giebel, J. Usaola, I. Sanchez, A. Palomares, R. Brownsword, J. Tambke, U. Focken, M. Lange, P. Pouka, G. Kallos, C. Lac, G. Sideratos, and G. Descombes. What performance can be expected by short-term wind power prediction models depending on site characteristics? In *Proceedings CD-ROM*, pages 1–9. European Wind Energy Association (EWEA), 2005.
- [106] I. Khandelwal, R. Adhikari, and G. Verma. Time series forecasting using hybrid ARIMA and ANN models based on DWT decomposition. *Procedia Computer Science*, 48:173 – 179, 2015. ISSN 1877-0509. doi: 10.1016/j.procs.2015.04.167. Int. Conf. on Computer, Communication and Convergence (ICCC 2015).



- [107] M. Khodayar, O. Kaynak, and M. E. Khodayar. Rough deep neural architecture for short-term wind speed forecasting. *IEEE Transactions on Industrial Informatics*, 13(6):2770–2779, Dec 2017. ISSN 1551-3203. doi: 10.1109/TII.2017.2730846.
- [108] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [109] G. Kirchgässner and J. Wolters. *Introduction to Modern Time Series Analysis*. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-73291-4.
- [110] D. M. Kline. Methods for multi-step time series forecasting with neural networks. In G. P. Zhang, editor, *Neural Networks in Business Forecasting*, chapter 12, pages 227–250. Information Resources Press, Arlington, VA, USA, 2003. ISBN 1591402158.
- [111] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*, chapter Section 3.3.2 part B. Addison-Wesley, Reading, Massachusetts, second edition, 10 Jan. 1981.
- [112] N. Kourentzes, D. K. Barrow, and S. F. Crone. Neural network ensemble operators for time series forecasting. *Expert Systems with Applications*, 41(9):4235 – 4244, 2014. ISSN 0957-4174. doi: <https://doi.org/10.1016/j.eswa.2013.12.011>.
- [113] O. Kramer, F. Gieseke, and B. Satzger. Wind energy prediction and monitoring with neural computation. *Neurocomput.*, 109:84–93, June 2013. ISSN 0925-2312. doi: 10.1016/j.neucom.2012.07.029.
- [114] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [115] A. Kusiak. Renewables: Share data on wind energy. *Nature*, 529:19–21, 2016. doi: 10.1038/529019a.
- [116] A. Kusiak, H. Zheng, and Z. Song. Short-term prediction of wind farm power: A data mining approach. *IEEE Transactions on Energy Conversion*, 24(1):125–136, March 2009. ISSN 0885-8969. doi: 10.1109/TEC.2008.2006552.
- [117] L. Landberg. *Meteorology for Wind Energy*. John Wiley & Sons Ltd, 2015. ISBN 9781118913444. doi: 10.1002/9781118913451.
- [118] L. Landberg and S. J. Watson. Short-term prediction of local wind conditions. *Boundary-Layer Meteorology*, 70(1):171–195, 1994. ISSN 1573-1472. doi: 10.1007/BF00712528.
- [119] M. Lange and D. Heinemann. Accuracy of short term wind power predictions depending on meteorological conditions. In *Proceedings of the 2002 Global Windpower Conference*, 2002. Proceedings of the 2002 Global Windpower Conference.
- [120] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, Nov 1989. ISSN 0163-6804. doi: 10.1109/35.41400.
- [121] Y. LeCun, 2016.
- [122] G. Li and J. Shi. On comparing three artificial neural networks for wind speed forecasting. *Applied Energy*, 87(7):2313 – 2320, 2010. ISSN 0306-2619. doi: 10.1016/j.apenergy.2009.12.013.
- [123] G. Li, J. Shi, and J. Zhou. Bayesian adaptive combination of short-term wind speed forecasts from neural network models. *Renewable Energy*, 36(1):352 – 359, 2011. ISSN 0960-1481. doi: <https://doi.org/10.1016/j.renene.2010.06.049>.
- [124] A. Lincoln. *The Collected Works of Abraham Lincoln Volume II - 1860*. Wildside Press LLC, 2008. ISBN 978-1434477033.
- [125] C. Liu, W. Sun, and H. Wu. Determination of complexity factor and its relationship with accuracy of representation for dem terrain. *Geo-spatial Information Science*, 13(4):249–256, Dec 2010. ISSN 1993-5153. doi: 10.1007/s11806-010-0390-y.

## References

---

- [126] H. Liu, H. qi Tian, D. fu Pan, and Y. fei Li. Forecasting models for wind speed using wavelet, wavelet packet, time series and artificial neural networks. *Applied Energy*, 107:191 – 208, 2013. ISSN 0306-2619. doi: 10.1016/j.apenergy.2013.02.002.
- [127] Y. Liu and H. Zhang. An empirical study on machine learning models for wind power predictions. In *2016 15th IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, pages 758–763, Dec 2016. doi: 10.1109/ICMLA.2016.0135.
- [128] Z. Liu, W. Gao, Y. H. Wan, and E. Muljadi. Wind power plant prediction by using neural networks. In *IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 3154–3160, Sept 2012. doi: 10.1109/ECCE.2012.6342351.
- [129] M. G. Lobo and I. Sanchez. Regional wind power forecasting based on smoothing techniques, with application to the spanish peninsular system. *IEEE Transactions on Power Systems*, 27(4): 1990–1997, Nov 2012. ISSN 1558-0679. doi: 10.1109/TPWRS.2012.2189418.
- [130] J. Lockett. *The Discovery of Weather: Stephen Saxby, the Tumultuous Birth of Weather Forecasting, and Saxby's Gale of 1869*. Formac Publishing Company Limited, 2012. ISBN 9781459501294.
- [131] J. López, A. Dorado, J. Feijo, C. Carrillo, J. Codrás, and E. Menéndez. The sotavento experimental wind park. In *Global windpower conference, Paris*, April 2002.
- [132] P. Louka, G. Galanis, N. Siebert, G. Kariniotakis, P. Katsafados, I. Pytharoulis, and G. Kallos. Improvements in wind speed forecasts for wind power prediction purposes using kalman filtering. *Journal of Wind Engineering and Industrial Aerodynamics*, 96(12):2348 – 2362, 2008. ISSN 0167-6105. doi: <https://doi.org/10.1016/j.jweia.2008.03.013>.
- [133] J. M. Lujano-Rojas, J. L. Bernal-Agustín, R. Dufo-López, and J. A. Domínguez-Navarro. Forecast of hourly average wind speed using arma model with discrete probability transformation. In M. Zhu, editor, *Electrical Engineering and Control: Selected Papers from the 2011 International Conference on Electric and Electronics (EEIC 2011) in Nanchang, China on June 20-22, 2011, Volume 2*, pages 1003–1010. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21765-4. doi: 10.1007/978-3-642-21765-4\_125.
- [134] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11 – 24, 2014. ISSN 0167-8655. doi: <https://doi.org/10.1016/j.patrec.2014.01.008>.
- [135] H. Madsen, P. Pinson, G. Kariniotakis, H. A. Nielsen, and T. S. Nielsen. Standardizing the performance evaluation of short-term wind power prediction models. *Wind Engineering*, 29(6): 475–489, 2005. doi: 10.1260/030952405776234599.
- [136] S. Makridakis and M. Hibon. The m3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451 – 476, 2000. ISSN 0169-2070. doi: [https://doi.org/10.1016/S0169-2070\(00\)00057-1](https://doi.org/10.1016/S0169-2070(00)00057-1). The M3- Competition.
- [137] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802 – 808, 2018. ISSN 0169-2070. doi: <https://doi.org/10.1016/j.ijforecast.2018.06.001>.
- [138] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 2019. ISSN 0169-2070. doi: <https://doi.org/10.1016/j.ijforecast.2019.04.014>.
- [139] J. Manero. Predicting the prediction. site forecastability for 12 hours ahead multi-step wind prediction using deep learning in north-america. "tbd", 2020.
- [140] J. Manero. Convolutional networks for wind speed multi-step time-series forecasting. "tbd", 2020.
- [141] J. Manero, J. Béjar, and U. Cortés. Predicting wind energy generation with recurrent neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11314 LNCS:89–98, 2018.
- [142] J. Manero, J. Béjar, and U. Cortés. Wind energy forecasting with neural networks. a literature review. *Computación y Sistemas*, 22, 2018. ISSN 2007-9737. number 4.

- [143] J. Manero, J. Béjar, and U. Cortés. Go with the flow: Recurrent networks for wind time series multi-step forecasting. *Frontiers in Artificial Intelligence and Applications*, 308:79–83, 2018.
- [144] J. Manero, J. Béjar, and U. Cortés. Deep learning is blowing in the wind. deep models applied to wind prediction at turbine level. *Journal of Physics: Conference Series*, 1222:012037, May 2019. doi: 10.1088/1742-6596/1222/1/012037.
- [145] J. Manero, J. Béjar, and U. Cortés. "Dust in the wind...", deep learning application to wind energy time series forecasting. *Energies*, 12(12):2385, 2019. doi: 10.3390/en12122385.
- [146] F. Martin, R. Zubiaur, P. Moreno, S. Rodriguez, M. Cabre, M. Casanova, A. Hormigo, and M. Alonso. Operational tool for short term prediction model of energy production in wind power plants at Tarifa (Spain). In *ECWEC at Travemünde 1993*, pages 802–803, 1993. ISBN 0-9521452-0-0.
- [147] J. M. Martorell. Barcelona supercomputing center: Science accelerator and producer of innovation. *Contributions to Science*, 12(1):5–11, 2016. ISSN 0960-1481. doi: doi:10.2436/20.7010.01.238.
- [148] C. F. Mass and Y.-H. Kuo. Regional real-time numerical weather prediction: Current status and future potential. *Bulletin of the American Meteorological Society*, 79(2):253–263, 1998. doi: 10.1175/1520-0477(1998)079<0253:RRTNWP>2.0.CO;2.
- [149] J. Miettinen, H. Holttinen, and B.-M. Hodge. Simulating wind power forecast error distributions for spatially aggregated wind power plants. *Wind Energy*, 0(0), 2019. doi: 10.1002/we.2410.
- [150] J. Miller and M. Hardt. Stable recurrent models, 2018.
- [151] M. Milligan, M. Schwartz, Y. Wan, and N. R. E. L. (U.S.). *Statistical wind power forecasting for U.S. wind farms [electronic resource]: preprint / M. Milligan, M.N. Schwartz, Y. Wan*. National Renewable Energy Laboratory Golden, Colo, 2003.
- [152] M. S. Miranda and R. W. Dunn. One-hour-ahead wind speed prediction using a bayesian methodology. In *2006 IEEE Power Engineering Society General Meeting*, pages 6 pp.–, 2006. doi: 10.1109/PES.2006.1709479.
- [153] D. C. Montgomery, C. L. Jennings, and M. Kulahci. *Introduction to time series analysis and forecasting; 2nd ed.* Wiley series in probability and statistics. Wiley, Hoboken, NJ, 2015.
- [154] M. T. Motlagh and H. Khaloozadeh. A new architecture for modeling and prediction of dynamic systems using neural networks: Application in tehran stock exchange. In *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*, pages 196–201, Jan 2016.
- [155] S. Nativi and B. Domenico. The OGC CF-netCDF specification: towards a common data model for feature, coverage and specimen data. In *EGU General Assembly Conference Abstracts*, EGU General Assembly Conference Abstracts, pages EGU2013–7334, Apr 2013.
- [156] B. Navascués, J. Calvo, G. Morales, C. Santos, A. Callado, A. Cansado, J. Cuxart, M. Díez, P. del Río, P. Escribà, O. García-Colombo, J. García-Moya, C. Geijo, E. Gutiérrez, M. Hortal, I. Martínez, B. Orfila, J. Parodi, E. Rodríguez, J. Sánchez-Arriola, I. Santos-Atienza, and J. Simarro. Long-term verification of HIRLAM and ECMWF forecasts over southern europe: History and perspectives of numerical weather prediction at {AEMET}. *Atmospheric Research*, 125–126:20 – 33, 2013. ISSN 0169-8095. doi: <https://doi.org/10.1016/j.atmosres.2013.01.010>.
- [157] V. -. Nelson, Horatio Nelson. *Letters and despatches of Horatio, Viscount Nelson*. John Knox Laughton, London, Longmans, Green, 1886.
- [158] T. S. Nielsen, A. Joensen, H. Madsen, L. Landberg, and G. Giebel. A new reference for wind power forecasting. *Wind Energy*, 1(1):29–34, 1998. ISSN 1099-1824. doi: 10.1002/(SICI)1099-1824(199809)1:1<29::AID-WE10>3.0.CO;2-B.
- [159] W. Obergassel, C. Arens, L. Hermwille, N. Kreibich, F. Mersmann, H. E. Ott, and H. Wang-Helmreich. The calm before the storm: An assessment of the 23rd climate change conference COP23 in bonn, 2018. Wuppertal Institut für Klima, Umwelt, Energie gmbH.

## References

---

- [160] I. Okumus and A. Dinler. Current status of wind energy forecasting and a hybrid method for hourly predictions. *Energy Conversion and Management*, 123:362–371, 2016. doi: <http://dx.doi.org/10.1016/j.enconman.2016.06.053>.
- [161] Z. O. Olaofe. A 5-day wind speed and power forecasts using a layer recurrent neural network (LRNN). *Sustainable Energy Technologies and Assessments*, 6:1 – 24, 2014. ISSN 2213-1388. doi: 10.1016/j.seta.2013.12.001.
- [162] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [163] A. E. Orhan. Skip connections as effective symmetry-breaking. *CoRR*, abs/1701.09175, 2017.
- [164] T. Ouyang, X. Zha, and L. Qin. A survey of wind power ramp forecasting. *Energy and Power Engineering*, 5(4B):368–372, 2013. doi: 10.4236/epe.2013.54B071.
- [165] A. K. Palit and D. Popovic. *Computational Intelligence in Time Series Forecasting*. Springer-Verlag London, 1 edition, 2005. ISBN 978-1-84628-184-6. doi: 10.1007/1-84628-184-9.
- [166] J. C. Palomares-Salas, J. J. G. de la Rosa, J. G. Ramiro, J. Melgar, A. Aguera, and A. Moreno. Arima vs. neural networks for wind speed forecasting. In *2009 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pages 129–133, May 2009. doi: 10.1109/CIMSA.2009.5069932.
- [167] J. C. Palomares-Salas, A. Agüera-Pérez, J. J. G. de la Rosa, and A. Moreno-Muñoz. A novel neural network method for wind speed forecasting using exogenous measurements from agriculture stations. *Measurement*, 55:295 – 304, 2014. ISSN 0263-2241. doi: <https://doi.org/10.1016/j.measurement.2014.05.020>.
- [168] Pedro Domingos @pmdomingos, 2018. You can't refute a statistic with an anecdote. Jun 21, 2018 5:09 PM · Tweet.
- [169] P. Pinson, H. Madsen, H. A. Nielsen, G. Papaefthymiou, and B. Klöckl. From probabilistic forecasts to statistical scenarios of short-term wind power production. *Wind Energy*, 12(1): 51–62, 2009. doi: 10.1002/we.284.
- [170] R. Poplin, A. V. Varadarajan, K. Blumer, Y. Liu, M. V. McConnell, G. S. Corrado, L. Peng, and D. R. Webster. Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, 2(3):158–164, 2018. ISSN 2157-846X. doi: 10.1038/s41551-018-0195-0.
- [171] M. Priestley. *Non-linear and non-stationary time series analysis*. Academic Press, London, 1988. ISBN 0-12-564910-X.
- [172] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *arXiv e-prints*, art. arXiv:1704.02971, Apr 2017.
- [173] V. Ranganayaki and S. N. Deepa. An intelligent ensemble neural network model for wind speed prediction in renewable energy systems. *ScientificWorldJournal*, 2016:9293529, Mar 2016.
- [174] I. Reinstein. XGBoost, a top machine learning method on kaggle, explained, October 2017. <https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html>.
- [175] D. Renggli, G. C. Leckebusch, U. Ulbrich, S. N. Gleixner, and E. Faust. The skill of seasonal ensemble prediction systems to forecast wintertime windstorm frequency over the north atlantic and europe. *Monthly Weather Review*, 139(9):3052–3068, 2011. doi: 10.1175/2011MWR3518.1.
- [176] J. S. Richman and J. R. Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, 278(6): H2039–H2049, 2000.
- [177] S. D. D. Riley, S. J. and R. Elliot. A terrain ruggedness index that quantifies topographic heterogeneity. *Intermountain Journal of Sciences*, 5(7):23–27, 1999. doi: 10.3390/en12071311.

- [178] J. Rodrigo, L. Paredes, N. Stoffels, and L. Bremen. Wind power predictability assessment from large to local scale. *European Wind Energy Conference and Exhibition, EWEC 2013*, 1:625–634, 01 2013.
- [179] E. R. Rodrigues, I. Oliveira, R. Cunha, and M. Netto. Deepdownscale: a deep learning strategy for high-resolution weather forecast. In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 415–422. IEEE, 2018.
- [180] L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39, 2010. ISSN 1573-7462. doi: 10.1007/s10462-009-9124-7.
- [181] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. Slavov, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, A. Luccioni, T. Maharaj, E. D. Sherwin, S. Karthik Mukkavilli, K. P. Kording, C. Gomes, A. Y. Ng, D. Hassabis, J. C. Platt, F. Creutzig, J. Chayes, and Y. Bengio. Tackling Climate Change with Machine Learning. *arXiv e-prints*, art. arXiv:1906.05433, Jun 2019.
- [182] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, Nov 1958.
- [183] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, Dec. 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y.
- [184] R. Sagar. Is Google’s claim to patent batch normalization a step towards monopolizing algorithms?, 2019. URL <https://analyticsindiamag.com/is-googles-claim-to-patent-batch-normalization-a-step-towards-monopolizing-algorithms/>. Analytics India Magazine Online, 2019-08-07. Accessed: 2020-05-05.
- [185] I. Sanchez. Short-term prediction of wind energy production. *International Journal of Forecasting*, 22(1):43 – 56, 2006. ISSN 0169-2070. doi: <https://doi.org/10.1016/j.ijforecast.2005.05.003>.
- [186] I. Sanchez, J. Usaola, O. Ravelo, C. Velasco, J. Dominguez, M. G. Lobo, G. Gonzalez, F. Soto, B. D. Guerra, and M. Alonso. Sipleólico - a wind power prediction system based on flexible combination of dynamic models. application to the spanish power system. *Proceedings of the first IEA Joint Action Symposium on Wind Forecasting Techniques*, pages 197–214, 2002.
- [187] A. Saprónova, C. Meissner, and M. Mana. Short time ahead wind power production forecast. *Journal of Physics: Conference Series*, 749(1):012006, 2016.
- [188] M. Shanker, M. Hu, and M. Hung. Effect of data standardization on neural network training. *Omega*, 24(4):385 – 397, 1996. ISSN 0305-0483. doi: [https://doi.org/10.1016/0305-0483\(96\)00010-2](https://doi.org/10.1016/0305-0483(96)00010-2).
- [189] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3): 379–423, 1948.
- [190] V. Sharan, S. Kakade, P. Liang, and G. Valiant. Prediction with a short memory. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing - STOC 2018*, 2018. doi: 10.1145/3188745.3188954.
- [191] J. Shi, J. Guo, and S. Zheng. Evaluation of hybrid forecasting approaches for wind speed and power generation time series. *Renewable and Sustainable Energy Reviews*, 16(5):3471 – 3480, 2012. ISSN 1364-0321. doi: 10.1016/j.rser.2012.02.044.
- [192] Z. Shi, H. Liang, and V. Dinavahi. Direct interval forecast of uncertain wind power based on recurrent neural networks. *IEEE Transactions on Sustainable Energy*, 9(3):1177–1187, 2018.
- [193] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387989501.
- [194] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1233–1240, June 2013. doi: 10.1109/CVPR.2013.163.

## References

---

- [195] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354 EP –, Oct 2017. Article.
- [196] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, pages 1–14, 2015.
- [197] W. C. Skamarock, J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang, and J. G. Powers. A description of the advanced research wrf version 3, 2008. National Center for Atmospheric Research, Boulder (CO) (2008).
- [198] J. M. Sloughter, T. Gneiting, and A. E. Raftery. Probabilistic wind speed forecasting using ensembles and bayesian model averaging. *Journal of the American Statistical Association*, 105(489):25–35, 2010. doi: 10.1198/jasa.2009.ap08615.
- [199] S. Smyl, J. Ranganathan, and A. Pasqua. M4 forecasting competition: Introducing a new hybrid es-rnn model, June 2018. URL <https://eng.uber.com/m4-forecasting-competition>. Accessed 20 July 2019.
- [200] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, and A. Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16):2861 – 2869, 2007. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2006.06.015>. Neural Network Applications in Electrical Engineering Selected papers from the 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005).
- [201] M. Sugiyama and M. Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. The MIT Press, 2012. ISBN 0262017091.
- [202] W. Sun and Y. Wang. Short-term wind speed forecasting based on fast ensemble empirical mode decomposition, phase space reconstruction, sample entropy and improved back-propagation neural network. *Energy Conversion and Management*, 157:1 – 12, 2018. ISSN 0196-8904. doi: <https://doi.org/10.1016/j.enconman.2017.11.067>.
- [203] J. Surowiecki. *The Wisdom of Crowds*. Anchor, 2005. ISBN 0385721706.
- [204] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.
- [205] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- [206] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.
- [207] D. Tabas, J. Fang, and F. Porté-Agel. Wind energy prediction in highly complex terrain by computational fluid dynamics. *Energies*, 12(7):1311, Apr 2019. ISSN 1996-1073. doi: 10.3390/en12071311.
- [208] S. B. Taieb and A. F. Atiya. A bias and variance analysis for multistep-ahead time series forecasting. *IEEE transactions on neural networks and learning systems*, 27(1):62–76, 2016.
- [209] Y. Tao, H. Chen, and C. Qiu. Wind power prediction and pattern feature based on deep learning method. In *IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, pages 1–4, Dec 2014. doi: 10.1109/APPEEC.2014.7066166.
- [210] J. W. Taylor, P. E. McSharry, and R. Buizza. Wind power density forecasting using ensemble predictions and time series models. *IEEE Transactions on Energy Conversion*, 24(3):775–782, Sept 2009. ISSN 0885-8969. doi: 10.1109/TEC.2009.2025431.
- [211] A. Tealab. Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, 3(2):334 – 340, 2018. ISSN 2314-7288. doi: <https://doi.org/10.1016/j.fcij.2018.10.003>.

- [212] A. Tealab, H. Hefny, and A. Badr. Forecasting of nonlinear time series using ann. *Future Computing and Informatics Journal*, 2(1):39 – 47, 2017. ISSN 2314-7288. doi: <https://doi.org/10.1016/j.fcij.2017.05.001>.
- [213] T. Teräsvirta, D. Tjøstheim, and C. Granger. *Modelling Nonlinear Economic Time Series*. Advanced Texts in Econometrics. OUP Oxford, 2010. ISBN 9780199587148.
- [214] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. D. Farmer. Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1):77 – 94, 1992. ISSN 0167-2789. doi: 10.1016/0167-2789(92)90102-S.
- [215] M. Theodosiou. Forecasting monthly and quarterly time series using stl decomposition. *International Journal of Forecasting*, 27(4):1178 – 1195, 2011. ISSN 0169-2070.
- [216] Z. Ti, X. W. Deng, and H. Yang. Wake modeling of wind turbines using machine learning. *Applied Energy*, 257:114025, 2020.
- [217] H. Tong. *Non-linear Time Series: A Dynamical System Approach*. Dynamical System Approach. Clarendon Press - Oxford University Press, 2003. ISBN 9780198523009. Oxford Statistical Science Series - 6.
- [218] A. M. Torres, J. F. Fernández, A. Troncoso, and F. Martínez-Álvarez. Deep learning-based approach for time series forecasting with application to electricity load. In J. M. Ferrández Vicente, J. R. Álvarez-Sánchez, F. de la Paz López, J. Toledo Moreo, and H. Adeli, editors, *Biomedical Applications Based on Natural and Artificial Computing*, pages 203–212, Cham, 2017. Springer International Publishing. ISBN 978-3-319-59773-7.
- [219] J. Torres, A. García, M. De Blas, and A. De Francisco. Forecast of hourly average wind speed with ARMA models in navarre (spain). *Solar Energy*, 79(1):65 – 77, 2005. ISSN 0038-092X. doi: <https://doi.org/10.1016/j.solener.2004.09.013>.
- [220] J. M. Torres and R. M. Aguilar. Using deep learning to predict complex systems: A case study in wind farm generation. *Complexity*, 2018:10, 2018. doi: 10.1155/2018/9327536.
- [221] J. M. Torres and R. M. Aguilar. Using Deep Learning to Predict Complex Systems: A Case Study in Wind Farm Generation. *Complexity*, 2018:10, 2018. doi: 10.1155/2018/9327536.
- [222] A. Torres-Barrán, Á. Alonso, and J. R. Dorronsoro. Regression tree ensembles for wind energy and solar radiation prediction. *Neurocomputing*, 326-327:151 – 160, 2019. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2017.05.104>.
- [223] J. Torriti, M. Hassan, and M. Leach. Demand response experience in europe: Policies, programmes and implementation. *ENERGY*, 35(4):1575 – 1583, April 2010. doi: 10.1016/j.energy.2009.05.021.
- [224] J. J. Traiteur, D. J. Callicutt, M. Smith, and S. B. Roy. A short-term ensemble wind speed forecasting system for wind power applications. *Journal of Applied Meteorology and Climatology*, 51(10):1763–1774, 2012. doi: 10.1175/JAMC-D-11-0122.1.
- [225] R. Tsay and R. Chen. *Nonlinear Time Series Analysis*. Wiley Series in Probability and Statistics. Wiley, 2018. ISBN 9781119264057.
- [226] S. E. Tuller and A. C. Brett. The characteristics of wind velocity that favor the fitting of a weibull distribution in wind speed analysis. *Journal of Climate and Applied Meteorology*, 23(1):124–134, 1984. doi: 10.1175/1520-0450(1984)023<0124:TCOWVT>2.0.CO;2.
- [227] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. ISSN 00264423.
- [228] United Nations Framework Convention on Climate Change (UNFCCC). Secretariat. Report of the conference of the parties on its twenty-first session, held in Paris from 30 November to 13 December 2015. addendum. part two: Action taken by the conference of the parties at its twenty-first session, 2015. United Nations Office at Geneva.
- [229] D. Upadhyay, J. Manero, M. Zamanand, and S. Sampalli. Majority vote ensemble algorithm for intrusion detection in power grids. *ibid*, 2020. Article submitted, pending publication.

## References

---

- [230] D. Upadhyay, J. Manero, M. Zamanand, and S. Sampalli. An intrusion detection system for power grids based on a gradient boosting algorithm with feature selection. *tbd*, 2020. Article submitted, pending publication.
- [231] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *SSW*, 125, 2016.
- [232] V. Vapnik. *The Nature of Statistical Learning Theory II edition*. Springer International Publishing, 1999. ISBN 978-1-4757-3264-1.
- [233] H. Wang, G. Li, G. Wang, J. Peng, Hui Jiang, and Y. Liu. Deep learning based ensemble approach for probabilistic wind power forecasting. *Applied Energy*, 188:56 – 70, 2017. ISSN 0306-2619. doi: <https://doi.org/10.1016/j.apenergy.2016.11.111>.
- [234] J. Wang, Y. Zong, S. You, and C. Træholt. A review of danish integrated multi-energy system flexibility options for high wind power penetration. *Clean Energy*, 1(1):23–35, 2017. doi: 10.1093/ce/zkx002.
- [235] X. Wang, K. Smith, and R. Hyndman. Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, 13(3):335–364, Nov 2006. ISSN 1573-756X. doi: 10.1007/s10618-005-0039-x.
- [236] Wind Europe organization. Wind in power 2016 European Statistics, 2017. Windeurope organization, Belgium.
- [237] I. Würth, L. Valldecabres, E. Simon, C. Möhrten, B. Uzunoğlu, C. Gilbert, G. Giebel, D. Schlipf, and A. Kaifel. Minute-scale forecasting of wind power—results from the collaborative workshop of iea wind task 32 and 36. *Energies*, 12(4):712, Feb 2019. doi: 10.3390/en12040712.
- [238] T. Xiong, Y. Bao, and Z. Hu. Beyond one-step-ahead forecasting: Evaluation of alternative multi-step-ahead forecasting models for crude oil prices. *Energy Economics*, 40:405 – 415, 2013. ISSN 0140-9883. doi: <https://doi.org/10.1016/j.eneco.2013.07.028>.
- [239] B. Xu, N. Wang, T. Chen, and M. Li. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv e-prints*, art. arXiv:1505.00853, May 2015.
- [240] C. Yan. *Wind turbine wakes: from numerical modeling to machine learning*. PhD thesis, University of Delaware, 2018.
- [241] I. Yassin, R. Jailani, M. Ali, R. Baharom, A. Hassan, and Z. Rizman. Comparison between cascade forward and multi-layer perceptron neural networks for narx functional electrical stimulation (fes)-based muscle model. *International Journal on Advanced Science, Engineering and Information Technology*, 7:215, 02 2017. doi: 10.18517/ijaseit.7.1.1388.
- [242] M. Yesilbudak, S. Sagiroglu, and I. Colak. A new approach to very short term wind speed prediction using k-nearest neighbor classification. *Energy Conversion and Management*, 69:77 – 86, 2013. ISSN 0196-8904. doi: <https://doi.org/10.1016/j.enconman.2013.01.033>.
- [243] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017.
- [244] G. U. Yule. Vii. on a method of investigating periodicities disturbed series, with special reference to wolfer’s sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636–646):267–298, 1927. doi: 10.1098/rsta.1927.0007.
- [245] L. Zadeh. The concept of a linguistic variable and its application to approximate reasoning—i. *Information Sciences*, 8(3):199 – 249, 1975. ISSN 0020-0255. doi: [https://doi.org/10.1016/0020-0255\(75\)90036-5](https://doi.org/10.1016/0020-0255(75)90036-5).
- [246] J. Zeng and W. Qiao. Support vector machine-based short-term wind power forecasting. In *2011 IEEE/PES Power Systems Conference and Exposition*, pages 1–8, March 2011. doi: 10.1109/PSCE.2011.5772573.



- [247] F. Zhang, Y. Qiang Sun, L. Magnusson, R. Buizza, S.-J. Lin, J.-H. Chen, and K. Emanuel. What is the predictability limit of midlatitude weather? *Journal of the Atmospheric Sciences*, 0(0):null, 0. doi: 10.1175/JAS-D-18-0269.1.
- [248] G. P. Zhang. *Neural Networks for Time-Series Forecasting*, pages 461–477. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-540-92910-9. doi: 10.1007/978-3-540-92910-9\_14. edited by: Rozenberg, Grzegorz and Bäck, Thomas and Kok, Joost N.
- [249] W. Zhao, Y.-M. Wei, and Z. Su. One day ahead wind speed forecasting: A resampling-based approach. *Applied Energy*, 178(C):886–901, 2016. doi: 10.1016/j.apenergy.2016.0.
- [250] J. Zhou, J. Shi, and G. Li. Fine tuning support vector machines for short-term wind speed forecasting. *Energy Conversion and Management*, 52(4):1990 – 1998, 2011. ISSN 0196-8904. doi: <https://doi.org/10.1016/j.enconman.2010.11.007>.
- [251] X. Zhu and M. G. Genton. Short-term wind speed forecasting for power system operations. *International Statistical Review / Revue Internationale de Statistique*, 80(1):2–23, 2012. ISSN 03067734, 17515823.



# Author publications during the research period

## Published Publications

J. Manero, J. Béjar, and U. Cortés. "Dust in the wind...", deep learning application to wind energy time series forecasting. *Energies*, 12(12):2385, 2019. doi: 10.3390/en12122385

J. Manero, J. Béjar, and U. Cortés. Deep learning is blowing in the wind. deep models applied to wind prediction at turbine level. *Journal of Physics: Conference Series*, 1222:012037, May 2019. doi: 10.1088/1742-6596/1222/1/012037

J. Manero, J. Béjar, and U. Cortés. Predicting wind energy generation with recurrent neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11314 LNCS:89–98, 2018

J. Manero, J. Béjar, and U. Cortés. Wind energy forecasting with neural networks. a literature review. *Computación y Sistemas*, 22, 2018. ISSN 2007-9737. number 4

J. Manero, J. Béjar, and U. Cortés. Go with the flow: Recurrent networks for wind time series multi-step forecasting. *Frontiers in Artificial Intelligence and Applications*, 308:79–83, 2018

## Pending journal publications at time of thesis deposit

J. Manero. Convolutional networks for wind speed multi-step time-series forecasting. "*tbd*", 2020

J. Manero. Predicting the prediction. site forecastability for 12 hours ahead multi-step wind prediction using deep learning in north-america. "*tbd*", 2020

D. Upadhyay, J. Manero, M. Zamanand, and S. Sampalli. An intrusion detection system for power grids based on a gradient boosting algorithm with feature selection. *tbd*, 2020. Article submitted, pending publication

A. Alsirhani, J. Manero, S. Sampalli, and P. Bodorik. A DDoS detection framework: Deep learning in a distributed system cluster. *tbd*, 2020. Article submitted, pending publication

D. Upadhyay, J. Manero, M. Zamanand, and S. Sampalli. Majority vote ensemble algorithm for intrusion detection in power grids. *tbd*, 2020. Article submitted, pending publication

*Yes, 'n' how many times must a man look up  
Before he can see the sky?  
The answer, my friend, is blowin' in the wind  
The answer is blowin' in the wind*

---

(1963) Bob Dylan [20]