UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

PRONÓSTICO PROBABILÍSTICO BASADO EN REDES NEURONALES PROFUNDAS PARA SERIES DE TIEMPO DE ENERGÍA

Tesis entregada como requerimiento parcial
para optar al grado de

DOCTOR EN INGENIERÍA INFORMÁTICA

por

Cristián Felipe Serpell Carriquiry

Comisión Evaluadora:

Dr. Rer. Nat. Héctor Manuel Allende Olivares (Director de Tesis, UTFSM)

Ph.D. Esteban Manuel Gil Sagas (Co-Director de Tesis, UTFSM)

Ph.D. Raquel Pezoa Rivera (Correferente, UTFSM)

Ph.D. Gonzalo Ruz (Externo Nacional, UAI)

Ph.D. Alejandro César Frery O. (Externo Internacional, VUW)

Ph.D. Mauricio Solar (Presidente Comisión, UTFSM)

MAYO 2023

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática

TÍTULO DE LA TESIS:

PRONÓSTICO PROBABILÍSTICO BASADO EN REDES NEURONALES PROFUNDAS PARA SERIES DE TIEMPO DE ENERGÍA

AUTOR:

CRISTIÁN FELIPE SERPELL CARRIQUIRY

Tesis presentada como requerimiento parcial para optar al grado de **Doctor en Ingeniería Informática** de la Universidad Técnica Federico Santa María.

Director de Tesis:

Dr. Rer. Nat. Héctor Manuel Allende Olivares
Departamento de Informática
Universidad Técnica Federico Santa María

Co-Director de Tesis:

Ph.D. Esteban Manuel Gil Sagas
Departamento de Ingeniería Eléctrica
Universidad Técnica Federico Santa María

Profesor Correferente:

Ph.D. Raquel Pezoa Rivera
Departamento de Informática
Universidad Técnica Federico Santa María

Examinador Externo Nacional:

Ph.D. Gonzalo Ruz
Facultad de Ingeniería y Ciencias
Universidad Adolfo Ibáñez

Examinador Externo Internacional:

Ph.D. Alejandro César Frery O.
School of Mathematics Statistics
Victoria University of Wellington

Presidente Comisión:

Ph.D. Mauricio Solar
Departamento de Informática
Universidad Técnica Federico Santa María

Agradecimientos

Agradezco a mis profesores, Héctor Allende, Esteban Gil, Ricardo Ñanculef y Carlos Valle, quienes son fuente de inspiración y me ayudaron a lograr mis objetivos, a mis compañeros de investigación Ignacio Araya y Tomás Ochoa, quienes mejoraron el producto de este trabajo, a mis compañeros/as del doctorado, con quienes compartimos grandes momentos, al profesor Johan Suykens y a Arun Pandey, quienes me recibieron en mi pasantía en Bélgica, y a Pablo Valdebenito, quien me apoyó para concluir este trabajo.

A todos/as que me animaron a soñar y comenzar este trayecto, además de acompañarme en la ruta de la ciencia y de hacer las cosas cada día mejor, junto con convertirse en grandes amigos y apoyarme en los momentos difíciles, Rafael Labarca, Carlos Hurtado, Alejandro Maass, Mario Ponce, María Paz Cortés, Álex Di Genova, Vanessa Peña Araya, Maximilano Rojo, Sebastián Pérez, Francisco Muñoz, Daniel Valenzuela, Valentina Quiroga, Víctor Ramiro, Manuel Vargas, Felipe Olmos, Raúl Aliaga, Ignacio Fantini, Miguel Toro, Jaime Zúñiga, Maricarmen Vallejos, Hernán Figueroa, Daniel Balparda, Tony Minoru Lopes, Eduardo Cardoso, Silvia Esparrachiari, Jaime Sepúlveda, Nelson Castillo, y muchos más.

Agradezco el apoyo de mi pareja, Carla, quien siempre me ha apoyado con su sabiduría, amor y paciencia, a su familia, Juan, Marisol y Francisco, por la comprensión y ayuda incondicionales, a mis hermanos Ricardo, Daniel, Eduardo y Ximena, y a quienes he conocido gracias a ellos, quienes son parte fundamental de mi vida y me ayudan con su alegría, y, finalmente, agradezco profundamente a mis padres Ricardo y María Eloísa, quienes me inculcaron el pensamiento crítico y curiosidad científica necesarios para desarrollar este trabajo, además de siempre apoyarme con su cariño y amor.

Valparaíso, Chile — Cristián Felipe Serpell Carriquiry

Mayo de 2023

Financiamiento

Este trabajo fue apoyado en parte por la Beca de Doctorado Nacional ANID 2017–21170109, las becas UTFSM 030/2019, 032/2021 y 081/2022, y los proyectos FONDECYT 1170123, BASAL FB0821 y PIA/APOYO AFB220004. Powered@NLHPC: Esta investigación/tesis fue parcialmente apoyada por la infraestructura de supercómputo del NLHPC (ECM-02)

Resumen

Muchas actividades dependen del pronóstico fiable de valores futuros. Para hacerlo, se han concebido modelos de redes neuronales que toman valores previamente observados, y proveen una descripción de la distribución de probabilidad de los valores futuros, en vez de solo un valor esperado. Esta tarea, llamada pronóstico probabilístico, permite prepararse para diferentes situaciones potenciales con distinta probabilidad de ocurencia, en vez de prepararse solo para una situación promedio que podría no ocurrir en la realidad. Estos modelos consideran la incertidumbre de los valores futuros usando diferentes representaciones, que incluyen intervalos de pronóstico, cuantiles, o supuestos distribucionales. En todas ellas, dos fuentes de incertidumbre deben ser consideradas: incertidumbre aleatoria, asociada a la elección del modelo, incluyendo la no consideración de otras variables que podrían otorgar más información sobre la variable pronosticada, e incertidumbre epistémica, relacionada a la falta de suficientes datos de entrenamiento para ajustar los parámetros del modelo. Esta última no es considerada por muchos trabajos, aún siendo especialmente importante para modelos de pronóstico basados en redes neuronales, ya que éstos tienen una gran cantidad de parámetros y los datos de entrenamiento son usualmente escasos para pronóstico de series de tiempo. En este trabajo, se propone un modelo de pronóstico probabilístico de aprendizaje profundo que considera ambas fuentes de incertidumbre. Para la primera, se considera un supuesto distribucional, que permite elegir entre distintas familias de distribuciones, y para la segunda, se usa Monte Carlo Dropout, una técnica que muestrea valores distintos para los parámetros cada vez que se evalua la red neuronal. Este modelo es validado para pronóstico a múltiples pasos de velocidad del viento, potencia eólica y demanda eléctrica, importantes tareas para el sector eléctrico, donde la penetración de fuentes de energía renovables no convencionales ha incrementado la incertidumbre presente en los sistemas eléctricos. Se concluye que el modelo maneja ambas fuentes de incertidumbre para el pronóstico de series de tiempo, y que la elección de la distribución de salida permite mejorar el desempeño para algunas series de tiempo, al usar una distribución diferente a la normal.

Palabras clave: Redes neuronales, Intervalos de pronóstico, Distribución de salida, Monte Carlo Dropout, Pronóstico probabilístico, Series de tiempo, Incertidumbre.

Abstract

Many activities depend on reliable forecasting of future values. To do it, neural network models have been devised that take previously observed values, and provide a description of a probability distribution of future values, instead of only one expected value. This task, called probabilistic forecasting, allows preparing for different potential situations with different probability of occurrence, instead of only preparing for a mean situation that may not happen in reality. These models consider the uncertainty of future values using different representations, including prediction interval bounds, quantile cuts, or distributional assumptions. In any of them, two sources of uncertainty must be considered: aleatoric uncertainty, related to the model choice, including the lack of consideration of other variables that may provide more information about the forecasted variable, and epistemic uncertainty, related to the lack of enough training data to fit the model parameters. This last uncertainty source is not considered by many approaches, being specially important for neural network forecasting models, as they have a big number of parameters and training data is usually scarce for time series prediction. In this work, a deep learning probabilistic forecasting model is proposed that considers both sources of uncertainty. For the former, a distributional assumption is considered, that allows choosing among different distribution families, and for the second, Monte Carlo Dropout is used, a technique that sample different values for the parameters each time the neural network is evaluated. This model is validated on wind speed, wind power and electrical load multistep forecasting, important tasks for the energy sector, where the penetration of variable renewable energy sources has made electrical grids to present more uncertainty. It is concluded that it handles both sources of uncertainty for time series forecasting, and that choosing the output distribution improve performance for some energy related time series, when using a distribution different from the normal one.

Keywords: Neural networks, Prediction intervals, Output distribution, Monte Carlo Dropout, Probability forecasting, Time series, Uncertainty.

Contents

List of Tables

List of Figures

Glossary

Notation	Description	Page List
aleatoric uncertainty	uncertainty that is inherent to a random process and is not related to lack of more data	vii, 2, 3, 7–9, 19, 28, 29, 32, 38, 59
deep learning	family of machine learning methods and models based on neural networks composed by several layers, capable of extracting complex features	vi, 2, 4–6, 9, 10, 12, 13, 15, 20, 21, 28, 29, 31, 38, 83
dropout	deep learning regularization technique that disables some connections or units randomly before each training step	viii, xi, xiii, 21, 22, 30, 31, 34–37, 39, 43, 46, 60, 61, 64–68, 70, 76, 81, 83, 84
ensemble	model composed by the composition of several generally simpler models	vi, viii, xii, 13, 15, 17–20, 23, 27–29, 36, 61, 68–70
epistemic uncertainty	uncertainty related to the model parameters and due to lack of more data	vi, vii, 2, 7–9, 15, 17–23, 28–32, 38, 59, 68, 76, 83
evidence	probability of current data, considering all possible parameter values of a model	8, 20
heteroscedasticity	assumption that the whole distribution of values may change through time, not only their mean	3, 4, 12, 16, 25, 40, 51, 74
homoscedasticity	assumption that the distribution of values remain constant through time, except by its mean	3, 16, 17, 19, 40, 51

Notation	**Description**	**Page List**
likelihood	function of the parameters associated to their fit to observed data	8, 13, 20, 32
log likelihood	the logarithm of the likelihood of the current parameters	16, 20, 22, 27, 35, 71
maximum likelihood	criterion to choose the value of model parameters, based on maximizing the likelihood	4, 9, 16, 20
neural network	learning model composed by many units, which outputs are computed applying simple rules on the outputs of other units	viii, x, xi, 2–4, 9, 10, 12–23, 28–38, 60, 61, 81
posterior	distribution of a random variable that is obtained after observing some data	8, 9, 19–21, 30, 34
prior	distribution that is assumed when no more information is known about a random variable	8, 9, 20, 30
probabilistic forecasting	a task where knowledge of the distribution of future values is required, more than just the expected value	vi–viii, xi, 1, 2, 4–6, 9, 12, 15, 16, 18, 19, 22–24, 27–29, 31, 34, 36–39, 43, 46, 51, 59, 61, 79
scenario	a trajectory of values, one for each time step. Generally referring to future predicted values	1, 4, 19, 22, 23, 27–29, 36
variational inference	technique that intends to approximates a probability distribution with a simpler one	vi, 20, 21, 29, 30, 59

Acronyms

Notation	Description	Page List
ACE	Average Coverage Error	13, 27
CDF	Cumulative Distribution Function	14, 25, 26
CNN	Convolutional Neural Network	12, 17, 19, 23
CRPS	Continuous Ranked Probability Score	xii, 15, 18, 26, 27, 29, 46, 52, 53, 71–73, 79, 81
CWC	Coverage Width Criterion	viii, xii, 12, 13, 25, 27, 29, 43, 46, 60–66, 68–73, 76, 79, 81
ELBO	Evidence Lower Bound	20, 30, 31
ELM	Extreme Learning Machine	13, 15, 18
FNN	Feedforward Neural Network	11, 43
GRU	Gated Recurrent Unit	11, 18, 22
IS	Interval Score	viii, xii, 13, 25, 27, 29, 43, 46, 60–73, 79, 81
LSTM	Long Short-Term Memory	xi, xiii, 11, 15, 17, 18, 22, 23, 28, 29, 38, 39, 60, 68–70, 79, 81–83
LUBE	Lower Upper Bound Estimation	vi, x, 12, 13, 15, 19, 27

Notation	Description	Page List
MAE	Mean Absolute Error	xii, 24–26, 51, 54, 55, 61, 64–66
MCD	Monte Carlo Dropout	vii–x, xiii, 3, 4, 18, 21–23, 27–32, 34, 36, 38, 43, 46, 51, 59–61, 64–68, 70, 76, 79–81
MVE	Mean Variance Estimation	vi, x, 4, 16–19, 22, 27–29, 32
PDF	Probability Density Function	8, 27, 31–34, 36
PI	Prediction Interval	vi, vii, x–xii, 2, 12–15, 17–19, 22–25, 27, 29, 36, 43, 46–51, 60, 61, 68, 71, 74–76
PICP	Prediction Interval Coverage Probability	xii, 13, 24, 25, 27, 43, 46, 60, 62–66, 68–70
PIMSE	Prediction Interval Mean Squared Error	27
PINAW	Prediction Interval Normalized Average Width	13, 25, 27, 43, 46, 64–66, 68–70
PL	Pinball Loss	ix, x, xii, xiii, 14, 15, 17, 25, 27, 29, 46, 52, 53, 71–74, 76–84
QR	Quantile Regression	vi, x, 14, 15, 17, 19, 25, 27, 29, 36, 46, 51
RMSE	Root Mean Squared Error	xii, 24, 51, 54, 55, 61, 64–66, 68–70
RNN	Recurrent Neural Network	x, 11, 13, 15, 17, 18, 21, 38
SVM	Support Vector Machine	13, 15
VAE	Variational Auto Encoder	18, 19, 21

List of Symbols

Notation	Description	Page List
$D_{\mathrm{KL}}(X \mid\mid Y)$	Kullback-Leibler divergence of random variables X and Y	20, 30
L_{t+1}^{c}	Lower bound of a prediction interval of the next future value, with nominal confidence c	x, 12, 13, 24, 25
$P(A)$	probability of an event A	12, 14
U_{t+1}^{c}	Upper bound of a prediction interval of the next future value, with nominal confidence c	x, 12, 13, 24, 25
X_t	random variable associated to the value for the time step t	x, xi, 6–9, 12–14, 16, 17, 19, 21, 22, 34, 36–39
α	quantile level, giving chance of having a value less or equal than the quantile value	x, 14, 15, 25
ϵ	data noise function term, which returns a random variable with mean zero	7, 8
$\mathbb{E}[X]$	expected value of the random variable X. The notation $\mathbb{E}_{z\sim Z}$ is also used when the expected value is computed with values z following the distribution given by Z	7, 8, 20, 26, 30
$\mathbb{I}(\text{condition})$	indicator function, returns 1 if condition is satisfied, else 0	24–26
$\mathbb{V}[X]$	variance of the random variable X	7, 8
$\mathbf{p}$	parameters of a neural network model	xi, 32, 35–39, 82
$\mathbf{x}_t$	vector used as input for a neural network, composed of previously observed values	x, xi, 7–9, 11, 13, 14, 16, 17, 19, 21, 22, 31–37

Notation	Description	Page List
$\mathcal{B}$	a batch which is a subser of training data	31–33
$\mathcal{D}$	set of training data	7–9, 14, 16, 19, 20, 30, 31, 34
$\mathcal{H}(X)$	entropy of the random variable X	30
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2	16, 21, 32, 40, 41
ϕ	parameters of an approximating distribution of the posterior	20, 21, 30, 31, 34
θ	parameters of a neural network model	xi, 7–9, 14, 16, 18–23, 30–37
$\tilde{\mathbf{x}}_t$	vector used as input for a neural network, composed of previously observed values and/or estimated intermediate values	36, 37
$\tilde{x}_t$	sample value for the time step t	xi, 24, 26, 36, 37, 39, 82
f	regression function, that estimates the mean	7, 8, 10, 14, 22, 23
$p(a)$	probability density function of value a, from a distribution deduced by the context	8, 9, 16, 19, 20, 30–32, 34, 35
$q(a)$	probability density function of value a of an approximating distribution of the posterior	20, 30, 31, 34
q_{t+1}^{α}	quantile for next future value, with chance α of having a value less or equal to it	14, 25
x_t	observed value, or stochastic process realization, for the time step t	6–9, 13, 14, 16, 19, 24–26, 31–34, 36, 37, 39, 82

Chapter 1

Introduction

There are many applications in engineering and sciences that require precise estimations of future values, such as finance [CKW20], health [BRGR21], or climatology [DDP15]. For them, knowing the actual probability distribution of future values allows preparing for a variety of future scenarios that may happen with different probability, as mentioned by [KKHN18]. In the energy sector [HGD+19], the task of forecasting wind speed, wind power, or electrical load has become of critical importance, as variable renewable energy sources have been introduced massively in electrical systems [BOSK15], making electrical grids to depend on climatological variables that cannot be directly controlled [BHH+18]. This has made the operation of electrical grids less predictable and thus less reliable. For example, when deploying wind power turbines, wrong wind power predictions may lead to not correctly satisfying overall electrical demand, due to not having prepared other generators in case wind is not available [QSK15], finally making energy prices much higher or lower than expected, and increasing the risks associated with all kind of decisions on the systems, including investment decisions [HF16]. For these applications, many forecasting models have been developed to predict only one value, associated with the most probable, or the expected, future value. Because of this, methods that provide descriptions of the distribution of future values have been developed, calling this task probabilistic forecasting. The descriptions obtained by them are then used to reduce the risk associated with further decisions based on the predicted values [HF16]. Common solutions for probabilistic forecasting are based on numerical simulations, where a precise physical model of the studied phenomenon is developed, and precise measurements of the initial conditions of the system are gathered and passed to the model. These models are hard to develop, requiring deep domain knowledge, and initial conditions are usually hard to determine. Therefore, many machine learning models for probabilistic forecasting have been developed, where the outputs of the model are either of probabilistic nature themselves, or they are used as inputs for further numerical methods, making them more robust [GK14].

As many real time series present non-linear relationships, most traditional machine learning models

used in practice for time series probabilistic forecasting rely on general linear methods, where ad hoc transformations have to be manually defined to extract input features, and mix the outputs via a link function. During recent years, there has been a shift in research towards deep learning approaches, showing that deep models can learn complicated non-linear relationships between past observations and future values. These models do not require the extraction of manually fixed features, do not require complicated output post-processing steps, and have obtained good results [KPB18, VBS$^+$17, LJX$^+$19]. Additionally, the outputs of a deep neural network model can be interpreted as descriptors of the distribution of future values [HF16]. For example, one common probabilistic forecasting task is Prediction Interval (PI) estimation, and one solution is to interpret the outputs as the bounds of the PI. Other models, instead, interpret the same two outputs as the mean and variance of a normal distribution, thus assuming that the predicted value has a normal distribution.

As the majority of statistical models, deep models present aleatoric uncertainty and epistemic uncertainty, as mentioned by [KD09], [KG17], [vASTG20] and [LSS20]. On the one hand, aleatoric uncertainty arises intrinsically from the randomness of the studied phenomenon, and it is related to the lack of consideration of all variables that may be relevant to model the studied phenomenon, and to noise in data measurements, caused by imperfections of sensors. This uncertainty, also called intrinsic uncertainty, data uncertainty or data noise, cannot be reduced by gathering more data. For example, when predicting load demand of a household, it may depend on the activities of those who live in the house, and those activities depend on many factors that cannot be completely measured, such as their health, neighborhood activities, and more. On the other hand, epistemic uncertainty is related to the parameters of the model, due to the finite amount of data used to learn the parameters of the model. After model training, usually model parameters are fixed, thus leaving out many other models that may fit the same training data. This is specially important for time series, where the available data is usually scarce, and neural networks models, where the number of model parameters is big. It is important to make this distinction of uncertainty sources, as different techniques have to be employed to address them. A wrong data noise model would imply that the forecasting model would give wrong estimates of the distribution of future values, even in the case of having a big amount of training data. Similarly, the lack of enough data would eventually show as having big epistemic uncertainty of future values, even in case the data noise model was correct.

To better understand the uncertainty sources, a brief experiment was conducted in a small synthetic dataset corresponding to a one dimensional regression, with data as shown in Figure 1.1. In this case, the aleatoric uncertainty corresponds to the fact that for any input value, there is a range of values that may be observed, given by the standard deviation used to generate these data. Neural networks were trained to estimate the corresponding mean and variance, considering only the training data of this dataset. Three networks were compared: two not considering the epistemic uncertainty, and one considering it

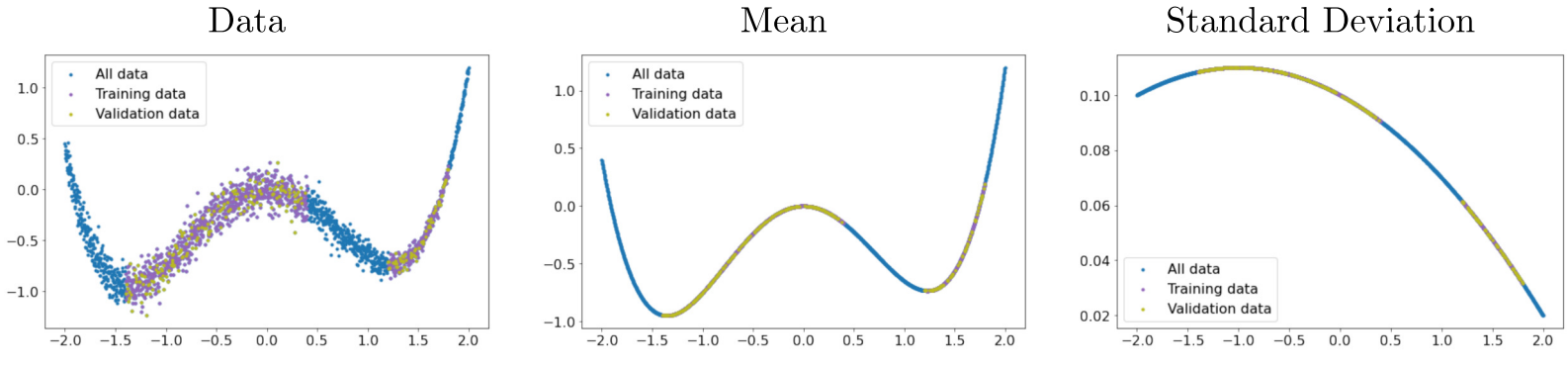


Figure 1.1: Synthetic data used for brief experiment, their mean and their standard deviation.

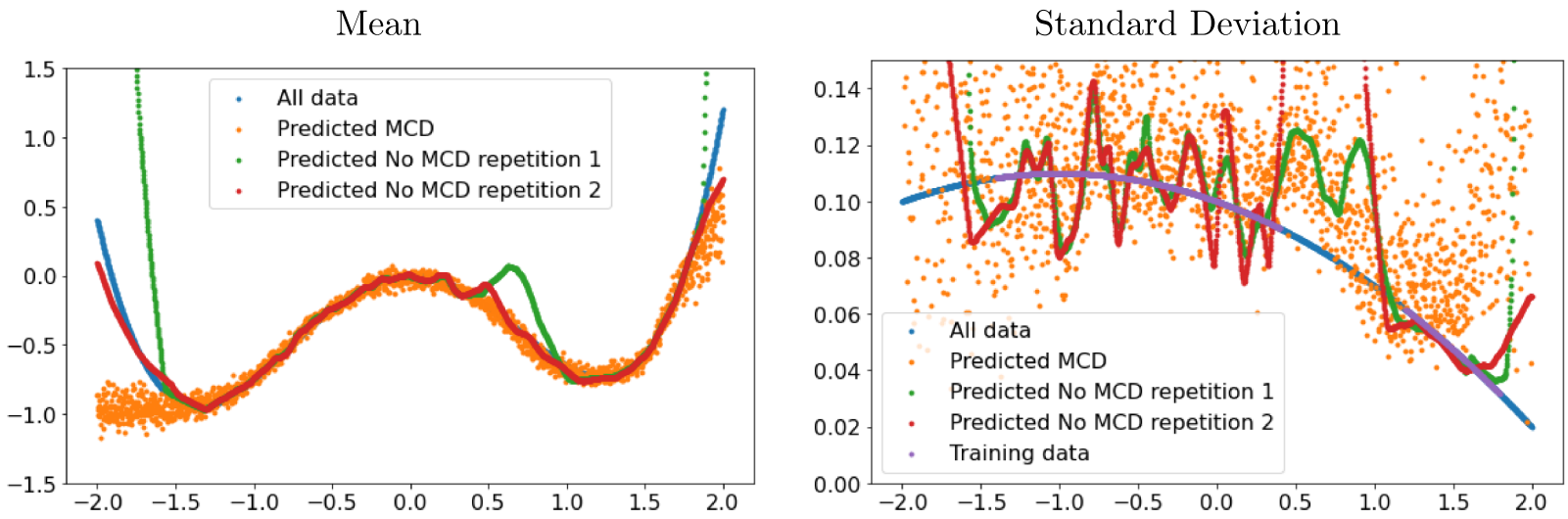


Figure 1.2: Mean and standard deviation estimated by neural network using MCD and not using it.

using Monte Carlo Dropout (MCD). The estimation of the mean and variance produced by them is shown in Figure 1.2, where it can be seen that while the mean prediction of one of the no MCD repetitions is good in general, this is not true for the other repetition, where predictions are wrong in regions far from training data. Regarding the standard deviation, it can be seen that for the repetition that correctly predicted the mean, the variance estimation was too high. In the case of MCD, instead, the estimation of the mean considers the lack of knowledge of data distribution out of training regions. Sampling these distributions produces final predictions depicted in Figure 1.3, and it can be seen that the network using MCD produces better predictions in the middle region that lacked training data, in comparison to the two other models. While one of these repetitions produced better predictions in the left region that also lacked training data, it can be seen that this is due to the random initialization of network parameters, as the other repetition did not work in that region.

When considering the aleatoric uncertainty, two cases have to be distinguished. The first one is the case of homoscedasticity, where it is assumed that the distribution of future values may change only regarding its mean, while the variance or other properties may remain constant. Nevertheless, the distribution of values of many real time series may change through time, a feature called heteroscedasticity.

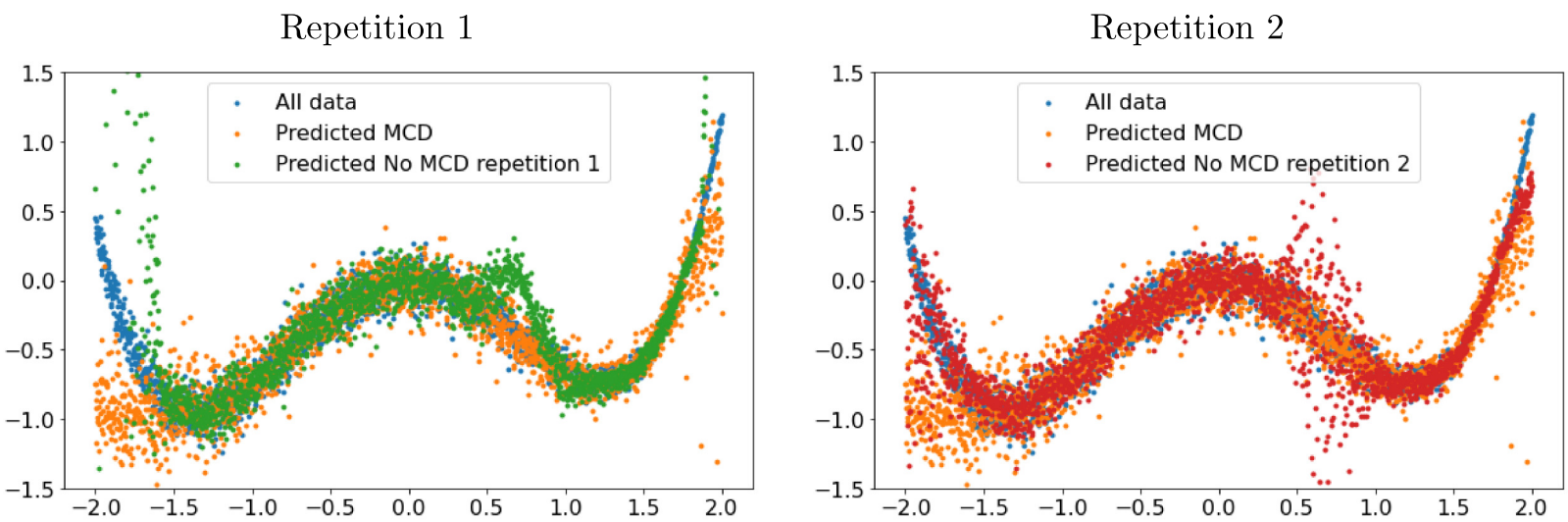


Figure 1.3: Final predictions produced by neural networks using MCD and not using it.

This is clear for wind speed or wind power time series, where during some hours of the day, there is much less uncertainty about the future wind resource. For heteroscedasticity, some works assume a fixed family of distributions, where the parameters of the distribution are obtained by a neural network model. For example, in the case of a normal distribution, this method is called Mean Variance Estimation (MVE) method, proposed by [NW94], and used by many works, such as [KN14] and [SFGJ20]. These models are trained using the maximum likelihood criterion, and can be used to describe the whole distribution of future values, with just two outputs per time step. Knowing the whole distribution of future values allows sampling of future scenarios for multistep forecasting tasks.

Unfortunately, the normality assumption does not hold for most real time series, specially for those that have non-symmetrical distributions, and, because of this, most works need ad hoc pre- and post-processing steps, trying to standardize the data. For example, in the case of wind speed, its value is always positive, which immediately presents a problem for distributions whose support includes negative values, like the normal distribution. In case the next value is zero, a model based on a normal distribution can predict a zero mean, when no wind is expected, but also gives the same probability to values that are below to those that are above zero. To alleviate this issue, this work proposes using other distribution families, like Log-normal, Gamma or Weibull, that may be better fit for the natural distribution of data, and do not require many parameters, thus not making the training procedure more complicated, than when assuming a normal distribution. While some approaches use a mixture of distributions to approximate any possible distribution of data, like [FKG10] and [ZLY$^+$20], based on the proposal of [Bis94], they require training a much bigger number of neural network outputs, as the parameters of each member of the mixture have to be obtained, plus the mixing coefficient parameters.

This work proposes a deep learning model framework to perform multistep time series probabilistic forecasting, handling both aforementioned sources of uncertainty, based on simple distributional assumptions, and MCD. To assess the proposal, it is applied to wind power, wind speed and electrical load

probabilistic forecasting tasks, which are representative of the problem [GK14].

This document is organized in the following chapters to better define the proposal and its assessment: In Chapter 2, a detailed theoretical foundation of this work is presented, specially focused on deep learning models for probabilistic forecasting, together with bibliographic notes regarding related work of each topic, a summary of this related work, and the metrics that are currently being used by the community to evaluate probabilistic forecasting methods. Then, in Chapter 3, a problem definition statement is made, the objectives and hypothesis of this work are presented, and the proposal of this work is specified in detail, in terms of the previous theoretical foundations and the defined problem. It explains how the proposed model is trained, how the output distribution is changed, and how it is used for multistep forecasting. In Chapter 4, experiments to assess the proposal are presented with their results, including synthetic dataset examples, and real data. Finally, in Chapter 5, the conclusions of this work are summarized, and future work directions are presented.

Chapter 2

Theoretical Framework

In this chapter, a detailed theoretical foundation of this work is presented. First, time series concepts are specified, and the sources of uncertainty that appear when forecasting are defined. Then, deep learning probabilistic forecasting models are presented, together with bibliographic notes regarding related work of each topic. Subsequently, a summary of the related work is presented, and finally the metrics that are currently being used by the community to evaluate probabilistic forecasting models are presented.

2.1 Uncertainty and time series forecasting

One of the most common ways to handle a forecasting application is to assume that the studied phenomenon can be described as a time series generated by a stochastic process. Formally, a stochastic process is defined as a function that associates a set of time indices T with a set of random variables $\{X_t\}_{t \in T}$. In this work, the indices are assumed to be countable, and equally spaced over time, thus implicitly defining a sequence of random variables, each one with a particular distribution. A sequence of observations of the phenomenon can be seen as one realization of the stochastic process, having the observed values $\{x_t\}_{t \in T}$. In this work, this sequence is considered to be a sequence of univariate real numbers, meaning that $x_t \in \mathbb{R}$. The base assumption when building a forecasting model, and in this work in particular, is that the stochastic process has an underlying sequential relationship for these values, according to an unknown function g, thus having

$$X_{t+1} = g\left(\ldots, X_{t-1}, X_t, \eta\right), \tag{2.1}$$

where η is a random variable that cannot be observed and groups all uncertainty of X_{t+1} given previous values. In other words, given known values $\{\ldots, x_{t-1}, x_t\}$ for all previous time steps, the uncertainty of X_{t+1} is given only by the uncertainty of η. Note that η includes the uncertainty generated by not including

additional variables that may be related to X_{t+1} in some way. As the function g is unknown, statistical or machine learning models are used to find an approximation function from a family of functions. Given training data of previous values, denoted by $\mathcal{D}$, and an optimality criteria, they use a learning algorithm to find the best function f from the family. f is called regression function or regression model, and may be different to the actual function, thus having an associated uncertainty when predicting X_{t+1}. To clearly see this, X_{t+1} can be written as

$$X_{t+1} = f\left(\ldots, X_{t-1}, X_t\right) + \epsilon\left(\ldots, X_{t-1}, X_t\right), \tag{2.2}$$

where $\epsilon\left(\ldots, X_{t-1}, X_t\right)$ is a random variable. Hence, $\epsilon\left(\ldots, X_{t-1}, X_t\right)$ encapsulates the uncertainty of the prediction because of using the model function f, which may be different to g, and the fact that η cannot be observed. Thus, the uncertainty derived from choosing one family of functions and choosing the variables included in the model is encapsulated in $\epsilon\left(\ldots, X_{t-1}, X_t\right)$, and it is called data noise, inherent uncertainty, or aleatoric uncertainty [KG17].

Most traditional forecasting applications center their attention on finding the expected (mean) future value, assuming that $\mathbb{E}\left[\epsilon\left(\ldots, X_{t-1}, X_t\right)\right] = 0$, and focusing only on $f\left(\ldots, X_{t-1}, X_t\right)$. In probabilistic forecasting, the goal is on learning regarding the distribution of X_{t+1}, or proxy values that will be described below, thus focusing the attention in both functions f and ϵ, instead of only f.

In practice, there is a finite number of previously observed values, and considering all of them as input of f and ϵ may be impractical, so models assume they take a fixed finite number M of previously observed values as inputs, receiving the input vector $\mathbf{x}_t = (x_{t-M+1}, x_{t-M+2}, \ldots, x_{t-1}, x_t)$. This assumption is called the Markov assumption. In this work, the attention is centered in uni-variate time series, meaning that the time series corresponds to a single value per time step. In the case of parametric models, the exact functions can be described as a member of a family parameterized by a set of values θ, so, Eq. (2.2) can now be written as

$$X_{t+1} = f_\theta\left(\mathbf{x}_t\right) + \epsilon_\theta\left(\mathbf{x}_t\right). \tag{2.3}$$

This introduces another source of uncertainty to the problem, which is known as model uncertainty or epistemic uncertainty [HKKH+02], and refers to the possibility that different functions f_θ and ϵ_θ could be chosen to model the relationship of X_{t+1} and $\mathbf{x}_t$ based on the same finite available training data $\mathcal{D}$. To better understand the relationship of the aforementioned random variables, a dependencies network diagram is presented in Figure 2.1. For example, let us analyze the variance of the prediction of the future value X_{t+1}. If the learning algorithm selects one fixed value for the parameters θ, say θ^*, it implicitly assumes no model uncertainty, and the variance of X_{t+1} would be given by the variance of $\epsilon_{\theta^*}\left(\mathbf{x}_t\right)$:

$$\mathbb{V}\left[X_{t+1} \mid \mathbf{x}_t\right] = \mathbb{V}\left[f_{\theta^*}\left(\mathbf{x}_t\right) + \epsilon_{\theta^*}\left(\mathbf{x}_t\right)\right] = \mathbb{V}\left[\epsilon_{\theta^*}\left(\mathbf{x}_t\right)\right].$$

If predictions are built using this variance around the value of a wrong value $f_{\theta^*}\left(\mathbf{x}_t\right)$, they could be misplaced because the assumption of correct parameters θ^*. On the other hand, if θ are considered as

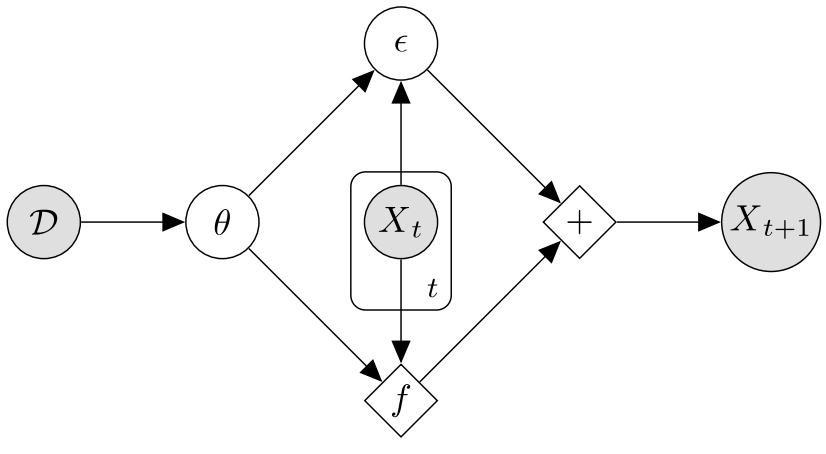


Figure 2.1: Diagram of dependencies of random variables associated with uncertainties in forecasting.

random variables, and assuming ϵ and f are conditionally independent on θ, now the variance is

$$\mathbb{V}\left[X_{t+1} \mid \mathbf{x}_t\right] = \mathbb{V}_\theta\left[\mathbb{E}_\theta\left[X_{t+1} \mid \mathbf{x}_t, \theta\right]\right] + \mathbb{E}_\theta\left[\mathbb{V}_\epsilon\left[X_{t+1} \mid \mathbf{x}_t, \theta\right]\right] = \mathbb{V}_\theta\left[f_\theta\left(\mathbf{x}_t\right)\right] + \mathbb{E}_\theta\left[\mathbb{V}_\epsilon\left[\epsilon_\theta\left(\mathbf{x}_t\right)\right]\right],$$

a result that comes from applying the law of total variance [GA14]. Then, the variance now depends also on the variance of the regression function, with respect to θ. Furthermore, instead of having a single data noise variance, its expected value is required. While this exercise has been done for one time step forecasting, the variance of following time steps gets influenced by the model uncertainty on each time step, thus making the variance bigger. Hence, if the model uncertainty is not considered, multistep forecasting is more uncertain and less reliable, so any probabilistic forecasting application requires considering both the aleatoric uncertainty and the epistemic uncertainty sources, to correctly estimate the distribution of future values, or proxy values such as the variance, specially if making multistep predictions.

To clarify this, and to be more precise, the conditional distribution of X_{t+1}, given previous observations $\mathbf{x}_t$, can be written from a Bayesian point of view, as a marginal posterior distribution in terms of the distribution of the model parameters θ, which is in turn a posterior distribution obtained after observing training data starting from a prior distribution. In other words, the conditional distribution of X_{t+1}, after observing training data $\mathcal{D}$, is given by:

$$p\left(x_{t+1} \mid \mathbf{x}_t, \mathcal{D}\right) = \int_{\theta \in \Theta} p\left(x_{t+1} \mid \mathbf{x}_t, \theta\right) p\left(\theta \mid \mathcal{D}\right) d\theta = \mathbb{E}_{\theta \sim p(\theta \mid \mathcal{D})}\left[p\left(x_{t+1} \mid \mathbf{x}_t, \theta\right)\right], \tag{2.4}$$

where $p\left(x_{t+1} \mid \dots\right)$ corresponds to the associated Probability Density Function (PDF) of the random variable X_{t+1} for the value x_{t+1}. The integral in this equation expresses the contribution of every possible model, by assuming different values for the parameters θ, from the set of all possible values Θ, weighted by the density of the specific values of θ, the posterior $p\left(\theta \mid \mathcal{D}\right)$. This posterior distribution is given by

$$p\left(\theta \mid \mathcal{D}\right) = \frac{p\left(\mathcal{D} \mid \theta\right) p\left(\theta\right)}{p\left(\mathcal{D}\right)} = \frac{p\left(\mathcal{D} \mid \theta\right) p\left(\theta\right)}{\int_{\theta' \in \Theta} p\left(\mathcal{D} \mid \theta'\right) p\left(\theta'\right)} = \frac{p\left(\mathcal{D} \mid \theta\right) p\left(\theta\right)}{\mathbb{E}_{\theta' \sim p(\theta')}\left[p\left(\mathcal{D} \mid \theta'\right)\right]}, \tag{2.5}$$

where $p\left(\mathcal{D} \mid \theta\right)$ is called the likelihood of the parameters θ, $p\left(\theta\right)$ is the prior distribution, and $p\left(\mathcal{D}\right)$ is called the evidence of the model. This Bayesian approach gives a principled way to incorporate the

subjective prior knowledge of the parameters, defining the prior distribution. Note that the conditional distribution $p\left(x_{t+1} \mid \mathbf{x}_t, \mathcal{D}\right)$ does not depend on any parameters, because all possible values are considered through the integral. When a single set of values for the parameters θ is assumed, thus not considering model uncertainty, it is equivalent to select a fixed θ^* and approximate the integral of Eq. (2.4) simply by $p\left(x_{t+1} \mid \mathbf{x}_t, \mathcal{D}\right) = p\left(x_{t+1} \mid \mathbf{x}_t, \theta^*\right)$. Common techniques to find θ^* are the maximum likelihood criterion, where

$$\theta^* = \arg\sup_{\theta \in \Theta} p\left(\mathcal{D} \mid \theta\right),$$

and the maximum a posteriori criterion, where a prior is chosen for the values of θ, $p\left(\theta\right)$, and then the value of θ^* that maximizes the posterior $p\left(\theta \mid \mathcal{D}\right)$ is selected, according to

$$\theta^* = \arg\sup_{\theta \in \Theta} p\left(\theta \mid \mathcal{D}\right) = \arg\sup_{\theta \in \Theta} \frac{p\left(\mathcal{D} \mid \theta\right) p\left(\theta\right)}{p\left(\mathcal{D}\right)} = \arg\sup_{\theta \in \Theta} p\left(\mathcal{D} \mid \theta\right) p\left(\theta\right).$$

As mentioned, training data is not infinite, so even when the model is flexible to model complex data distributions, these two options may be strong assumptions. Furthermore, the posterior distribution of Eq. (2.4) encapsulates all available information from the prior and the data, and knowing it would lead to a model that is more robust to disturbances. Considering only the parameters obtained by maximum likelihood or maximum a posteriori, may lead to loss of information and consistent failures.

In the case of multistep forecasting, the distribution of X_{t+K}, the value on a future time step K, is given by

$$p\left(x_{t+K} \mid \mathbf{x}_t\right) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} p\left(x_{t+K} \mid \mathbf{x}_{t+K-1}\right) p\left(x_{t+K-1} \mid \mathbf{x}_{t+K-2}\right) \cdots p\left(x_{t+1} \mid \mathbf{x}_t\right) dx_{t+1} \cdots dx_{t+K-1}.$$

Note that $\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+K-1}$ are not known in advance when forecasting, as they depend on intermediate values $x_{t+1}, \ldots, x_{t+K-1}$ that have not been observed. Then, uncertainty from each time step gets accumulated, making the uncertainty of X_{t+K} to be much higher as K grows. If the aforementioned aleatoric uncertainty and epistemic uncertainty are not both considered correctly, the uncertainty of X_{t+K} would be incorrectly estimated, thus producing poor probabilistic forecasting tools.

While there are several different kinds of parametric models which may be used, this work is centered in deep learning models. The reason is that they have proven to be successful for probabilistic forecasting in different fields, and different models have been proposed, with different advantages and disadvantages. In the following section a summary of these methods is presented.

2.2 Deep learning probabilistic forecasting

Deep learning is a family of learning models, based on artificial neural networks, which are composed of multiple hidden layers of simple processing units organized in a hierarchical manner, having the outputs of the units in one layer being the inputs of the next one, as shown in Figure 2.2. When trained, each unit

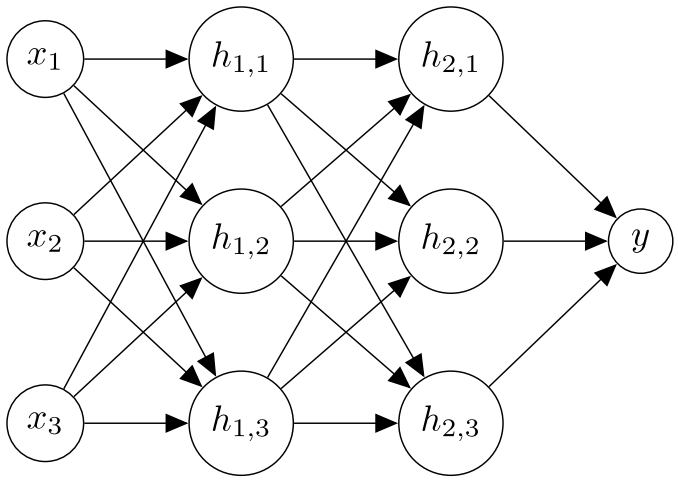


Figure 2.2: Connectivity of a simple artificial neural network with an input vector $\mathbf{x} = (x_1, x_2, x_3)$ of dimension 3, two hidden layers with dimension 3, and a dimension 1 output y. Image based on [GBC16].

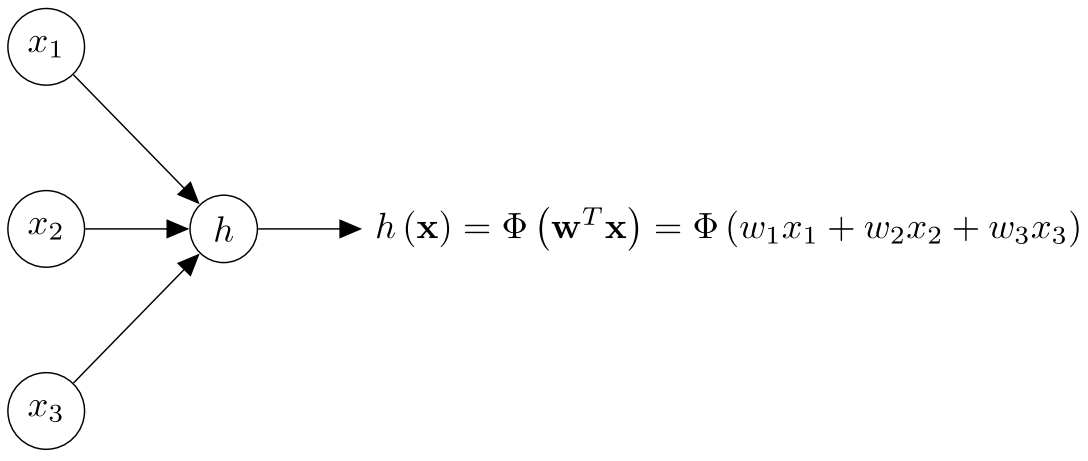


Figure 2.3: Description of the function applied by a single neuron unit. Φ represents the activation function, and $\mathbf{w}$ is the vector of weights associated with the connections from the previous layer

learns to represent a feature of the data, and the higher a unit is in the network, the more complex the feature it can represent, as more computations have been made to get its value. Deep learning has gained popularity due to the availability of increased amounts of data [DDS+09] and computational processing capacity to train the models, reaching state-of-the-art results in many domains [KPB18]. In general, it is assumed that each unit computes a linear transformation using the connection weights, and then applies a non-linear activation function, as presented in Figure 2.3. The use of a non-linear activation is fundamental, as otherwise all different layers may be collapsed into just one linear function. Indeed, it has been shown that neural networks built with non-linear activation functions with at least one hidden layer are universal approximators [Cyb89] [HSW89], meaning that they can be used to approximate any real function with any degree of accuracy, given enough training data and a big enough network. In the context of the time series forecasting, this means they would be able to learn a function f that is close to g.

The arrangement of neuron units and their connectivity define an architecture. The simple architecture

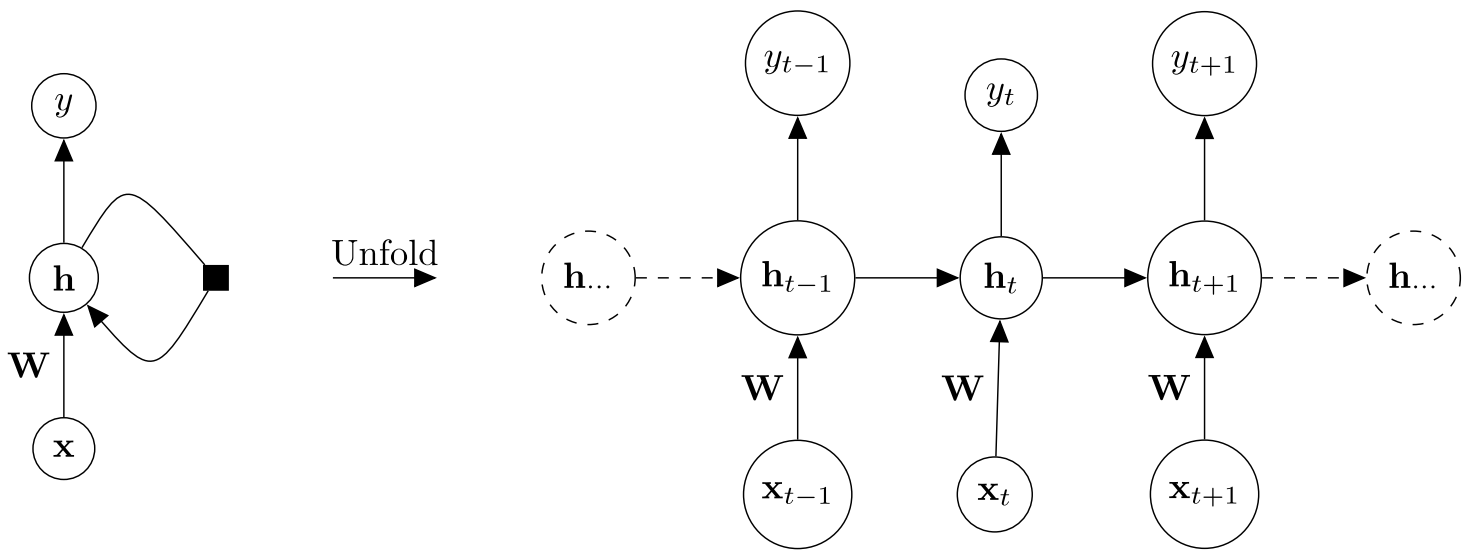


Figure 2.4: An unfolded RNN, which at each time step receives a new input vector $\mathbf{x}_t$, updates its hidden state to $\mathbf{h}_t$ using its previous hidden state $\mathbf{h}_{t-1}$, and produces an output y_t. Image adapted from [GBC16].

shown in Figure 2.2 is called Feedforward Neural Network (FNN), which connects all units from one layer to all units in the next one, having a different weight associated with each connection, represented by a matrix $\mathbf{W}_\ell$ of weights, where ℓ is the number of the layer. Thus, the output of each hidden layer may be written as a vector $\mathbf{h}_\ell$, and computed from the previous layer output as $\mathbf{h}_\ell = \Phi(\mathbf{W}_\ell \mathbf{h}_{\ell-1})$. While this architecture can be used for any learning task, different architectures have been proposed to tackle different kind of problems. A class of architectures specially devised to work with sequences, such as those of time series, are Recurrent Neural Networks (RNNs), which use recurrent connections in their processing units to reuse their processing layer across multiple time steps. Each time a new input from a sequence is given to the network, it updates the internal states of its processing units using both this new input and its previous internal state. To understand how these networks implement deep computations, it is useful to draw the RNN model as a graph of an unfolded network, with shared parameters, as shown in Figure 2.4, displaying the temporal dependencies between the internal state and its past values, which span multiple layers of computations. It can be said that each internal state contains a summary of the information from all previous time steps. Therefore, the final internal state, which presumably contains information from the whole sequence, can be used to compute the output of the network. General RNNs present vanishing gradient difficulties when learning using gradient descent based algorithms, that hinders the model from learning long term dependencies [PMB13], so memory block architectures such as Long Short-Term Memory (LSTM) [HU97] and Gated Recurrent Unit (GRU) [CVG+14] have been developed to improve them, and have been widely used by the time series community [Gam17]. Also, it is possible to stack different layers of memory blocks, making RNNs also deep in the sense of extracting complex features that are used by recurrent units high in the hierarchy of the network. In the context of time series, this allows the network to learn multiscale features, and has been applied to aforementioned architectures, making Stacked LSTM and Stacked GRU networks, such as in [SSB+18].

Other deep learning models that have been explored for time series probabilistic forecasting include Convolutional Neural Networks (CNNs), which take advantage of a hierarchical architecture, sharing parameters of many units in the same level of the hierarchy. CNNs are usually devised to work with image (2D) data, but they have been adapted to work with sequential data, applying 1D convolutions instead. Recently, some other deep learning architectures to process sequential data based on attention mechanisms have been proposed to address the vanishing gradient problem, such as the Transformer networks [VBS+17]. They apply transformations on input data, giving more importance to some values or regions of the input, thus making the network to "focus" on the most relevant input for the desired task. Its application for forecasting is a matter of heavy current research, specially how to reduce their data and memory usage requirements, such as in [LJX+19], where performance improvements are made to apply it for point forecasting of power load, wind power and other time series. As it is not the objective of this document to present a full review over deep learning models in general, it is recommended to go to [GBC16] for a detailed review on deep learning models and their history.

When used for probabilistic forecasting, the outputs of these neural network models are trained to provide insight on the distribution of future values using different techniques described below. Bibliographic notes are summarized after presenting each technique, where some metrics to measure performance are mentioned. These metrics are described in detail in Section 2.4.

2.2.1 Prediction Intervals and the Lower Upper Bound Estimation method

A product of probabilistic forecasting of general interest is the estimation of PIs. In this task, given a nominal confidence level c, the objective is to estimate an interval $I_{t+1}^c = \left[L_{t+1}^c, U_{t+1}^c\right]$, where L_{t+1}^c and U_{t+1}^c correspond to the lower and upper bounds of the interval. The nominal confidence defines the expected probability of occurrence of a value within the interval. In other words, the interval is built to satisfy that

$$P\left(L_{t+1}^c < X_{t+1} \leq U_{t+1}^c\right) \geq c.$$

In general, it is assumed that the interval is centered, meaning that

$$P\left(X_{t+1} \leq L_{t+1}^c\right) \leq \frac{1-c}{2} \text{ and } P\left(X_{t+1} > U_{t+1}^c\right) \leq \frac{1-c}{2}.$$

The Lower Upper Bound Estimation (LUBE) method directly estimates the bounds of the PI as outputs of a neural network, as shown in Figure 2.5, generally using ad-hoc loss functions that ensure a minimum coverage of the interval, such as the Coverage Width Criterion (CWC) [KNCA11]. These models can be regarded as assuming heteroscedasticity, as they can produce intervals of varying width depending on the input of the model, although they cannot be used to describe the full distribution of future values more precisely. This makes them hard to extend for the multistep forecasting task, where the distribution of a value further in the future may depend on the distribution of intermediate future values. Additionally,

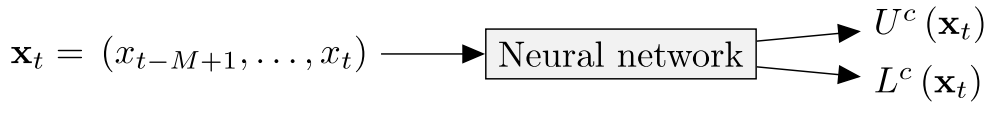


Figure 2.5: LUBE method: a neural network is trained to compute $L^c(\mathbf{x}_t)$ and $U^c(\mathbf{x}_t)$, which estimate the lower (L^c_{t+1}) and upper (U^c_{t+1}) bounds of a PI for X_{t+1}. The neural network may have any architecture.

they use different optimization algorithms and objectives functions that require ad-hoc algorithms, like Particle Swarm Optimization (PSO), instead of stochastic gradient descent, commonly used in deep learning.

Bibliographic notes

Different algorithms and objective functions have been proposed for PI estimation using the LUBE method. [KNCA11] proposed a method minimizing the CWC to train a neural network for one-step forecasting, and extended it in [KN13] to consider the model uncertainty, although their work was criticized by [WXO+14] because CWC prefers PIs that are too wide, so useless in practice. Similarly, [WLCC19] produces PI minimizing another cost function based on the CWC and evaluated using the PIRD metric. In [QSK14], the Prediction Interval Normalized Average Width (PINAW) is used as loss function, introduced in [QSK12], while constraining the Prediction Interval Coverage Probability (PICP), to ensure the coverage is met. In [SLP16], the LUBE method is posed as a multi-objective optimization problem to train two Support Vector Machines (SVMs) for the lower and upper bound respectively, minimizing the Average Coverage Error (ACE) and reducing the absolute value of the Interval Score (IS). [SLD18a] proposed the use of an RNN, with a variation of CWC as loss function, and using a complex optimization algorithm, instead of gradient descent optimization as usually done in deep learning. Other work applying the LUBE method is [KKKF+21], which estimates PI for multistep forecasting of power load and wind power. They enhanced their study in [KKNN20], where they studied the sensitivity of the method when changing the input to the network. Similarly, [KFKN16] developed a LUBE method based on the maximization of PICP and minimization of PINAW, for one-step PI estimation, considering the model uncertainty problem through the use of a fuzzy ensemble. In the same line, [PZBN18] proposes to use a linear penalization of the missed PICP, instead of the exponential one given by the CWC loss, based on a Bernoulli likelihood, and then using an ensemble for the model uncertainty. In [MDM18], a Self-Adaptive Evolutionary Extreme Learning Machine (ELM) is trained to compute multiple PIs at once by maximizing a set of PICP functions and minimizing a set of PINAW functions, using multiple constraints to keep lower and upper bounds coherent with one another. This work considers model uncertainty using a bootstrap ensemble. Similarly, an ELM is used with another cost function based on the CWC and evaluated using the PIRD in [WLCC19]. Another work based on ELM is [WXP13], which estimates PI of wind power for one-step forecasting, and that was extended in [WXP+14a] adding the evaluation using the CWC, although not considering the model uncertainty. The model proposed in [ZWW+15] is also based on ELM for wind power PI estimation, although the input of the network is an Empirical Mode Decomposition (EMD) of the time series, and this decomposition is done using an ensemble, that in a way would be considering model uncertainty. These last works are based on ELM, which are networks models not specially devised for sequential data such as time series. A hybrid approach using Variational Mode Decomposition to first decompose the original time series and Multi-Kernel Ridge Regression as the base model is proposed in [NBD18], where PINAW is the function to be minimized. A Wavelet neural network is employed in [SLD18b] for a multi-objective LUBE problem, minimizing PINAW while maximizing PICP of the PIs.

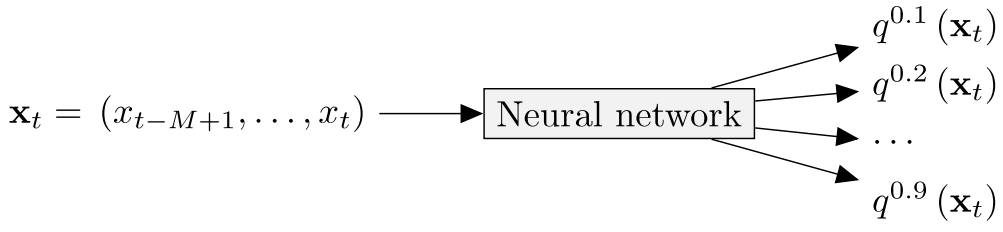


Figure 2.6: QR using a neural network. Each output is trained to estimate a desired quantile of X_{t+1}. The neural network may have any architecture.

2.2.2 Quantile Regression

Quantile Regression (QR) seeks to model the distribution of X_{t+1}, estimating the points q^{α}_{t+1} where its associated Cumulative Distribution Function (CDF) matches a desired value $\alpha \in (0, 1)$. In other words,

$$q^{\alpha}_{t+1} = \inf \{q \in \mathbb{R} : P(X_{t+1} \leq q) \geq \alpha\},$$

where α is the cut value of the CDF. In practice, q^{α}_{t+1} is modeled as a function from past values, $q^{\alpha}_{\theta}(\mathbf{x}_t)$, parameterized by θ, similar to f_θ of Eq. (2.3), using a neural network as in Figure 2.6. To find such function, the following minimization problem is solved, as proposed by [Tay00] and [KH01]:

$$\min_{\theta \in \Theta} \sum_{(\mathbf{x}_t, x_{t+1}) \in \mathcal{D}} \mathrm{PL}_{\alpha}(x_{t+1} - q^{\alpha}_{\theta}(\mathbf{x}_t)), \tag{2.6}$$

where PL_{α} is the Pinball Loss (PL), defined as

$$\mathrm{PL}_{\alpha}(z) = \begin{cases} \alpha z & \text{if } z \geq 0, \\ (1-\alpha)(-z) & \text{if } z < 0. \end{cases} \tag{2.7}$$

Note that, instead of X_{t+1}, Eq. (2.6) mentions x_{t+1} and $\mathbf{x}_t$, which represent observed values from training data $\mathcal{D}$, instead of the unobserved random variable. The PL can be seen as a generalization of the absolute error, as seen in Figure 2.7, giving a linear penalization to observations further from the center, with a higher slope for those in one side, pushing observations to lay in the side with less slope. The loss is minimized when the proportion of observations that lay on the left side of z is exactly α. In the case of $\alpha = 0.5$, this is equivalent to half the absolute error, and the minimum is obtained when z is the median. Constraints are added to this optimization problem so that quantile cuts do not overlap. In general, direct QR models require solving complex optimization problems, due to the feasibility constraints and because they have many outputs, one for each quantile, making them hard to optimize. Also, if a point prediction of the mean is needed, direct PI or QR models cannot estimate it directly and an additional output is needed, trained specifically for that task.

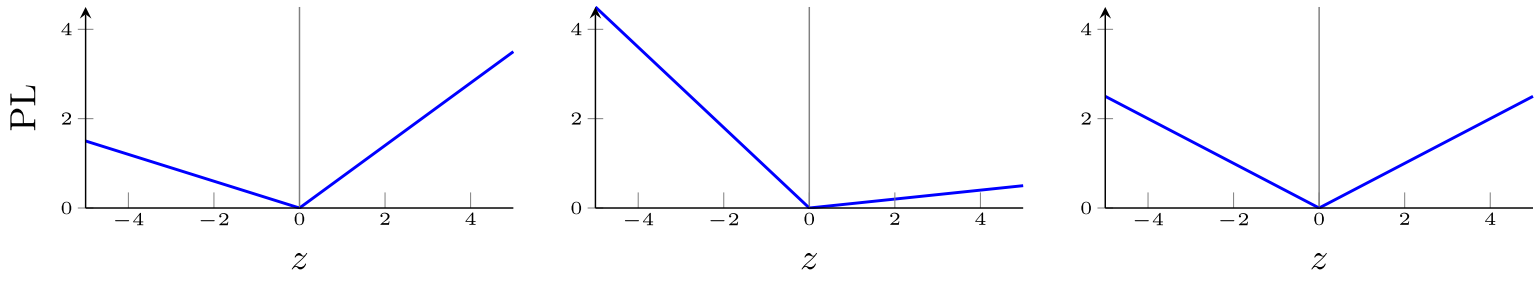


Figure 2.7: PL, for three values of α. Left: $\alpha = 0.7$. Center: $\alpha = 0.1$. Right: $\alpha = 0.5$.

Bibliographic notes

Several deep learning works have applied QR to represent the future distribution of values. In [HNM14], various regression and preprocessing techniques, such as Wavelet Transform (WT), Fuzzy ARTMAP Network, Firefly optimization and SVM, are used to provide quantiles. This mix of complex models makes the solution hard to interpret and to optimize, and it does not consider the model uncertainty. In [WLW+17a], an ELM is used for QR and PI and, while they make multistep probabilistic forecasting, it is not clear what methodology is employed to do so. In [HL18] a single layer FNN is used to produce quantiles and PI, and the complete probability distribution is estimated using Kernel Density Estimation (KDE), although not considering the model uncertainty. An RNN is used in [GWYK18], together with QR for power load forecasting using the PL, while not considering the model uncertainty. A general model is proposed in [WTNM17], where a sequence-to-sequence model based on a stacked LSTM is used for multistep forecasting, directly estimating many steps, instead of re-injecting the output of a one-step model, as it is usually done, mentioning that their approach would work best. Similarly, [HLSK17] performs QR using a modified loss called smooth PL for wind speed data, and then it uses the model to produce PI and quantiles. The QR network proposed by [ZQS19] is also evaluated for the production of PI, while not considering the epistemic uncertainty. This work was extended in [ZQG+20] to study the effect of skip connections on the network. In [GBW+19], an LSTM is optimized to produce quantiles based on an adjustable distribution shaped by splines, minimizing the Continuous Ranked Probability Score (CRPS). This work is interesting because it focuses on many time series prediction having a flexible distribution, and it is based on a recurrent network similar to the one of this work, although it requires more parameters than other works based on simpler distributions, and it does not consider the model uncertainty. Similarly, [GRK+21] proposed an RNN that performs QR and sampling on each step, minimizing CRPS as loss, and not considering epistemic uncertainty. In [WZT+18], quantiles are produced based on an ensemble of models, each one performing QR, some of them being neural networks, having a complex algorithm to mix quantiles produced by each model, which makes this approach hard to implement and interpret, although considering model uncertainty. In [LNHW17], QR is performed using an ensemble of point forecasting models, considering the model uncertainty. Interestingly, in [LZM+20] an ensemble is proposed, where one of the models uses a LUBE method, and another uses a QR neural network, among others, and with this ensemble they would be considering the model uncertainty, although they only use it for one-step forecasting and evaluate it only for the task of PI estimation. The work of [SH12] performs wind speed QR using Radial Basis Functions and uses an ensemble of point models to consider the model uncertainty. Another approach is [WWL+16], that uses a Deep Belief Network model, much harder to optimize and use than the RNN used in this and other works. While it mentions the importance of Weibull distribution for modeling the wind speed, it does not seem to use this fact in their model. A recent and interesting work is [LALP19], that proposed enhancing QR with the use a transformer network, being able to consider long and short term dependencies on the output distribution, for multi variate time series. While they do not consider the model uncertainty directly, they perform a kind of feature selection to reduce variance of the results.

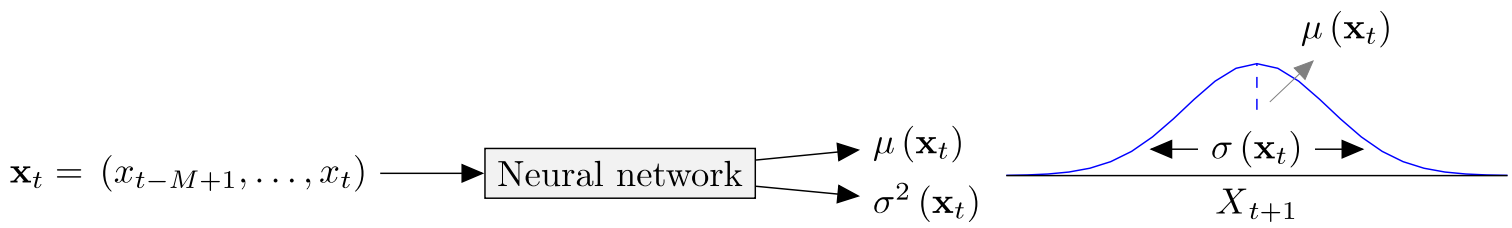


Figure 2.8: MVE method: the outputs of a neural network are trained to get the mean and variance of a normal distribution describing the distribution of X_{t+1}. The neural network may have any architecture.

2.2.3 Distributional assumption and the Mean Variance Estimation method

Another way to tackle probabilistic forecasting tasks is by assuming a specific distribution for the data noise term of Eq. (2.3), which in turn conditions a distribution for X_{t+1}. If a family of parameterized distributions is considered, then several methods can be used to estimate the parameters of such distribution. The simplest models assume constant variance (homoscedasticity), with data noise modeled as a normal random variable whose variance remains constant through time; hence, it can be easily estimated from the residuals during training. The homoscedasticity assumption may be unrealistic, as most real time series present changing variance through time. For time series with heteroscedasticity, the MVE method [NW94] may be used, where a neural network is used to estimate both the location and scale parameters of a normal distribution, as shown in Figure 2.8. As the mean and variance are obtained evaluating a network with parameters θ, over input values $\mathbf{x}_t$, then they are written as $\mu_\theta(\mathbf{x}_t)$ and $\sigma^2_\theta(\mathbf{x}_t)$, and the assumption is that $X_{t+1} \sim \mathcal{N}\left(\mu_\theta(\mathbf{x}_t), \sigma^2_\theta(\mathbf{x}_t)\right)$. Then, the relationship of Eq. (2.3) is given by

$$X_{t+1} = \mu_\theta(\mathbf{x}_t) + \sigma_\theta(\mathbf{x}_t)\epsilon, \tag{2.8}$$

where ϵ is a standard normal random variable.

To estimate the weights θ, generally the maximum likelihood criterion is applied, maximizing the log likelihood of a normal distribution:

$$\max_{\theta\in\Theta} \log p(\mathcal{D} \mid \theta) = \max_{\theta\in\Theta} \sum_{(\mathbf{x}_t, x_{t+1})\in\mathcal{D}} \log p(x_{t+1} \mid \mathbf{x}_t, \theta) =$$

$$\max_{\theta\in\Theta} \sum_{(\mathbf{x}_t, x_{t+1})\in\mathcal{D}} \log \frac{1}{\sqrt{2\pi\sigma^2_\theta(\mathbf{x}_t)}} \exp\left(-\frac{(\mu_\theta(\mathbf{x}_t) - x_{t+1})^2}{2\sigma^2_\theta(\mathbf{x}_t)}\right). \tag{2.9}$$

Removing constants, this is equivalent to minimizing

$$\min_{\theta\in\Theta} \sum_{(\mathbf{x}_t, x_{t+1})\in\mathcal{D}} \left(\log \sigma^2_\theta(\mathbf{x}_t) + \frac{(\mu_\theta(\mathbf{x}_t) - x_{t+1})^2}{\sigma^2_\theta(\mathbf{x}_t)}\right). \tag{2.10}$$

This optimization is generally done using one of many available stochastic gradient descent algorithms, that randomly chooses mini-batches of samples (indexes), to estimate the gradient of Eq. (2.10). Note

that if the variance output is removed from the model, it is possible to assume a variance that does not depend on the input $\mathbf{x}_t$, being constant through time. This in turn reduces to the traditional regression case, where Eq. (2.10) translates into least squares minimization, and the network learns to predict the mean of X_{t+1}.

In general, the normal distribution is a strong assumption, as many real time series do not naturally present a normal distribution. For example, usually wind speed is modeled using a Weibull distribution. This fact is typically tackled applying transformations to the original data, like a standardization step, that is fixed during the training of the model. It is applied to the data before the network training, and, when evaluating the model, the inverse is applied to recover the original series distribution. Standardizing the data does not solve the problem for data that is non-symmetrical, it is multi-modal, or that has a lower or upper bound, a fact given by physical constraints. For example, solar power generation cannot be negative. Although the bounds problem can be alleviated by applying logarithm to the series, this can make the normal distributional assumption even more incorrect in comparison to the real distribution. A solution that has been used by many authors to model complex distributions is the use of a Mixture of Gaussians, as proposed originally by [Bis94], and that has been posed for multistep forecasting using RNNs by [Gra13]. This approach requires a bigger number of neural network outputs to train, and also needs to determine the number of components of the mixture, making this approach more complicated than the MVE method.

Bibliographic notes

Regarding distribution based methods, [WZC+18] uses an ensemble of point models to estimate quantiles, assuming a constant variance around the point estimate. Similarly, [MYLL07] assume homoscedasticity, with data noise modeled as a normal random variable whose variance remains constant, and it is estimated from the residuals during training. The approach of [KNC10] is similar, with the addition of considering the epistemic uncertainty using the delta method for power load PI forecasting. In [CZD+18], a fuzzy ensemble of RNNs is proposed to estimate PI, each network assuming homoscedasticity. In [WNL+19], another approach assuming homoscedasticity is proposed, where a distribution known as the T Location Scale is assumed from the output of an Echo State Network, instead of a normal distribution. A related approach is the one of [MB07] and [ZL16], who use a different normal distribution for each period of a day. In [WLW+17b], a hybrid approach is proposed for wind power forecasting, where WT is used to decompose previous values, then the obtained time series are arranged in 2D arrays, and a CNN is applied as if they were images, and the outputs estimate a normal distribution like MVE. An ensemble of these models is trained to account for model uncertainty. This is a complex model, in comparison with other RNNs presented in this work. A work that compares the use of MVE and QR based on the use of a bidirectional LSTM is [TBVD19], which also compares them to using empirical copulas for multivariate forecasting. They conclude that QR is superior to MVE in terms of the PL, which would be given by the ability to handle non-normal distributions by QR, although having more parameters and being harder to train. Their copula method works for multivariate forecasting but for low dimensions. In contrast, [SBSC+19] proposes an LSTM network with the use of Gaussian copulas, parameterized by a low rank matrix, being able to handle high dimensionality time series, either in number of time steps or in number of time series. These two works do not mention the epistemic uncertainty issue. The MVE method with a normal distribution assumption is used by [KN14] to produce PI of wind power for one-step forecasting only,

and not considering the epistemic uncertainty. The same authors proposed a similar normal distribution MVE network and addressed the model uncertainty problem with an ensemble for wind power forecasting in [KNC13], but it is trained using an ad-hoc metric for PI, which cannot be easily extended to other probabilistic forecasting tasks. Similarly, [WXP+14b] uses ELM networks with the MVE method for wind power forecasting, assuming a normal or a Beta output distribution. They show that assuming Beta distributions works for data with known bounds, such as wind power in this case, which has a minimum and a maximum value. They use a bootstrap ensemble for the epistemic uncertainty, and they assess their method only considering the PI estimation task. The work of [GG16a] proposed using MCD for time series forecasting, although using a constant variance. Then, in [ZL17], this was extended to the MVE method with non-constant variance, applied to forecast the number of Uber trips in a day and a corresponding PI, using a normal distribution assumption. It focuses mostly on the coverage of the PIs produced with this technique, instead of focusing in general probabilistic forecasting tasks, and studies only the one-step forecasting case. The model proposed in [Sha19] uses an MVE model with normal distribution to forecast wind power, and applies MCD to handle the model uncertainty, differing from this work because they use a GRU recurrent network instead of an LSTM, they apply it only to wind power forecasting, and they do not explore other output distributions. The DeepAR model [SFGJ20] uses an RNN similar to the one presented in this work, although it does not consider the epistemic uncertainty, and only uses normal distributions for their assessment. An extension to this work, based on a deep factor model of latent features, is [WSM+19], where it is shown that considering common latent features, and not necessarily assuming a normal distribution, improves the results. An interesting work that challenges the normal assumption is [Egi16], which considers a Weibull distribution with an MVE-like network, although performing only one-step forecasting and not considering the epistemic uncertainty. While not a distribution based method, [PZBN18] estimate PIs and compare their method with an MVE based on a normal and an exponential distribution, noting that the normal distribution assumption is not always correct.

Regarding works based on mixtures of distributions, generally a mixture of Gaussians is used, such as in [VFM18], where the future distribution for power load is estimated and compared using CRPS and distribution plots. While [SAR+19] proposes a mixture of Gaussians as well, it shows the results of simulations using the model, but it does not compare any quantitative metric. In [FKG10] and [ZYI+19], a mixture of Gaussians is used as the output of an LSTM to forecast wind power, using also an external numerical weather prediction model as input. All these works do not consider the model uncertainty problem. Differently from those, [MYL+16] proposes an ensemble of mixture of Gaussians, where each network of the ensemble may be able to represent any distribution with a sufficiently big number of outputs, and considering the epistemic uncertainty through the ensembles. This solution, while flexible, requires a big number of neural network outputs to train for each member of the ensemble, thus being hard and slow to train. A work using a different distribution than normal is [ZLY+20], that compares using a mixture of Gaussians with using a mixture of Beta distributions, which are bounded in the lower and the upper side. To build PIs, they choose the one that leaves the same amount of probability below the lower bound and over the upper bound. They do not consider the model uncertainty problem. An interesting work is [NTS13], which uses a mixture of GARCH models, and a bootstrap ensemble of mixtures, thus handling both sources of uncertainty, but with a complex model, and also not focusing on electrical time series. Finally, another work is [FRHC21], which estimates PI and studies the output distribution of a model based on a conditional Variational Auto Encoder (VAE), which transforms the input into a reduced dimensionality latent space, hoping that the model gathers a disentangled representation of important features that can be sampled and used to simulate future values, in this case of wind speed, and it also applies bootstrap to consider the model uncertainty.

2.2.4 Ensembles for model uncertainty representation

To tackle the epistemic uncertainty of probabilistic forecasting, a popular method to approximate Eq. (2.4) is using an ensemble of models, each one with a particular value for θ, changing the integral by a weighted

sum. In an ensemble of R models, the weight associated with each model is an estimation of the posterior distribution $p(\theta \mid \mathcal{D})$, approximating Eq. (2.4) as

$$p(x_{t+1} \mid \mathbf{x}_t, \mathcal{D}) = \sum_{r=1}^{R} p(x_{t+1} \mid \mathbf{x}_t, \theta_r) \text{Weight}(\text{Model}_r), \text{ with } \sum_{r=1}^{R} \text{Weight}(\text{Model}_r) = 1. \tag{2.11}$$

The distribution, then, considers R different possible values of θ, thus including model uncertainty for probabilistic forecasting. To estimate the distribution of X_{t+1}, they sample predictions from all models, obtaining a big number of samples from the target distribution. With these samples, they produce the output of the probabilistic forecasting, such as empirical quantiles or lower and upper interval bounds. This method, nevertheless, has downsides: during training, it has to train R models, which may be slow if R is large, and, when using it, all R models have to be evaluated, either in parallel, storing them in memory, having big memory requirements, or sequentially, which is slow.

Bibliographic notes

Ensemble models are the most common approach to handle the epistemic uncertainty. Initially, the LUBE method was extended with ensembles by [KN13]. They produced a more complex model for PI estimation based on distribution methods using a bootstrap ensemble in [KNC13]. Then, other works have followed them with used more specialized techniques, such as [ZWW+15], where an Empirical Mode Decomposition (EMD) is done with an ensemble, [KFKN16] with a fuzzy ensemble, [MDM18] with a bootstrap ensemble, and [PZBN18]. For QR, different ensemble approaches have been studied. [WZT+18] propose an ensemble of QR models, which is complex due to difficulty of mixing quantiles from different models. [LZM+20] show an ensemble of LUBE models to estimate quantiles. [LNHW17] and [SH12] are ensembles of point forecasting models, where quantiles are returned from the outputs of all models, thus considering the model uncertainty, but not the aleatoric uncertainty. Input scenario methods with ensembles include [MYL+14], where an ensemble of point prediction models is trained for the estimation of PI of wind speed and wind power, evaluated with example plots, but without using any quantitative metric. Also, in [XH16], a simulation using different temperature scenarios is performed, and an ensemble of models is used, where one of them is a neural network that assumes a fixed normal variance over 24 hour groups. Regarding distribution based methods, [WZC+18] use an ensemble of point models with an estimated constant variance, thus considering homoscedasticity. Similarly, [CZD+18] proposes a fuzzy ensemble of models assuming homoscedasticity. In [WLW+17b], time series values are arranged in 2D arrays and then an ensemble of CNNs is used to estimate normal distributions. In [WXP+14b], an MVE method was proposed that assumes normal or Beta distributions, considering a bootstrap ensemble for the epistemic uncertainty, and they assess their method only considering the PI estimation task. A more complex model based on ensembles has been proposed by [MYL+16], with a mixture of Gaussians. This solution, while flexible, requires a big number of neural network outputs to train for each member of the ensemble, thus being harder to train, and slower due to the fact that it is an ensemble of models. Also, [NTS13] proposed a bootstrap ensemble of GARCH models, which is quite complex, handling both sources of uncertainty, although not focusing on electrical time series. Finally, [FRHC21] propose a PI estimation method based on a conditional VAE, which transforms the input into a reduced dimensionality latent space, hoping that the model gathers a disentangled representation of important features that can be sampled and used to simulate future values, in this case of wind speed, and it also applies bootstrap to consider the model uncertainty.

2.2.5 Variational inference for epistemic uncertainty

Ensemble methods, as shown, are computationally expensive, a concern that becomes exacerbated when working with deep learning models. Recently, many techniques based on variational inference have been developed that allows modeling the uncertainty of the parameters of a neural network [ZBKM19]. Following this approach, a neural network is seen as a parametric model, where its parameters θ have a probability distribution, as specified in Eq. (2.4). The posterior distribution $p(\theta \mid \mathcal{D})$ can be extremely complex for even the simplest neural networks and cannot be described in a closed form [KSW15], so variational approaches model it using another known distribution $q(\theta \mid \phi)$, which in turn is parameterized by a new set of parameters ϕ. This is known as the reparameterization trick, as explained in [Gra11]. These models are trained using the maximum likelihood criterion to learn ϕ, but the actual likelihood cannot be obtained explicitly; thus they maximize a lower bound of the evidence, as defined in Eq. (2.5), called variational bound, or Evidence Lower Bound (ELBO) [Gra11]. To more clearly see how this is done, the log-evidence of the training data $\mathcal{D}$ is written as:

$$\begin{aligned}
\log p(\mathcal{D}) &= \left(\int_{\theta\in\Theta} q(\theta \mid \phi)d\theta\right)\log p(\mathcal{D}) = \int_{\theta\in\Theta} q(\theta \mid \phi)\log p(\mathcal{D})d\theta \\
&= \int_{\theta\in\Theta} q(\theta \mid \phi)\left(\log p(\mathcal{D}) + \log\frac{p(\theta \mid \mathcal{D})}{q(\theta \mid \phi)} - \log\frac{p(\theta \mid \mathcal{D})}{q(\theta \mid \phi)}\right)d\theta \\
&= \int_{\theta\in\Theta} q(\theta \mid \phi)\left(\log\left(p(\mathcal{D})p(\theta \mid \mathcal{D})\right) - \log q(\theta \mid \phi)\right)d\theta - \int_{\theta\in\Theta} q(\theta \mid \phi)\log\frac{p(\theta \mid \mathcal{D})}{q(\theta \mid \phi)}d\theta \\
&= \int_{\theta\in\Theta} q(\theta \mid \phi)\left(\log p(\mathcal{D} \mid \theta) + \log\frac{p(\theta)}{q(\theta \mid \phi)}\right)d\theta + D_{\mathrm{KL}}\left(q(\theta \mid \phi) \parallel p(\theta \mid \mathcal{D})\right).
\end{aligned}$$

Using the fact that the KL divergence is always positive, the ELBO is defined as:

$$\begin{aligned}
\log p(\mathcal{D}) &\geq \int_{\theta\in\Theta} q(\theta \mid \phi)\log p(\mathcal{D} \mid \theta)d\theta - D_{\mathrm{KL}}\left(q(\theta \mid \phi) \parallel p(\theta)\right) \\
&= \mathbb{E}_{\theta\sim q(\theta\mid\phi)}\left[\log p(\mathcal{D} \mid \theta)\right] - D_{\mathrm{KL}}\left(q(\theta \mid \phi) \parallel p(\theta)\right) = \mathrm{ELBO}(\mathcal{D}, \phi). \qquad (2.12)
\end{aligned}$$

Then, the ELBO is composed by two terms: an expected log likelihood of the parameters to be maximized, using the estimated distribution $q(\theta \mid \phi)$ instead of the posterior distribution $p(\theta \mid \mathcal{D})$, and the KL divergence between the estimated distribution and a prior distribution $p(\theta)$ that has to be chosen in advance, to be minimized. The associated optimization problem used to find ϕ is

$$\max_{\phi\in\Phi} \mathrm{ELBO}(\mathcal{D}, \phi) \leq \log p(\mathcal{D}).$$

Hence, maximizing this bound through searching values for ϕ implicitly maximizes the log likelihood. This is usually done with a stochastic gradient algorithm, that takes samples of neural network parameters θ based on the current estimated distribution $q(\theta \mid \phi)$, and then evaluates the ELBO considering it. For many prior distributions, the second term can be written explicitly, thus facilitating the evaluation of the

bound and acting as a regularizer. Different distributions are found in the literature, for example using a normal distribution $\theta \sim \mathcal{N}(\phi_\mu, \phi_{\sigma^2})$ as in [BCKW15], mixture of Gaussians in [LW17], or complex distributions modeled with neural networks in [PBL+17]. Recently, a work proposed the use of a neural network that estimates confidence intervals for the parameters individually, without having to assume normality [OHM+20]. As explained by [GG15], many regularization techniques usual in the deep learning community can be interpreted as assumptions on the parameters distributions. One of them is MCD, introduced in [SHK+14], a technique that disables some connections of the network, adding stochasticity to its output. According to [GG16b], MCD approximates the posterior distribution of the parameters when they are given by a Gaussian Process. This was extended for RNNs in [GG16a], and recently used with self-attention networks for a chatbot text data by [YCZ21]. It is important to mention the difference between MCD and traditional dropout: in traditional dropout, the deactivation of connections is performed only during network training, but not during inference, so it is used as a regularization technique, while in MCD, the deactivation is also performed during inference, performing an implicit sampling of the posterior distribution. Some authors have explored using a Monte Carlo variation of batch normalization following the same idea, and with competitive results [TAS18]. Also, some authors have shown recently that other deep models may be used to directly model the epistemic uncertainty based on distance to prototype points, like [LLP+20], [vASJ+21], and [MKvA+21], that work similar to RBF networks, although they require major architectural changes, instead of using similar models, as when using MCD. Indeed, MCD can be adapted to be used even for already trained deep models, with a minimal re-training procedure, as mentioned by [LSS20].

Finally, another general approach based on variational inference is the VAE [KW13], that assumes the uncertainty may be modeled using a set of latent variables $\mathbf{z}_t$, inferred from $\mathbf{x}_t$, with Gaussian posterior distribution, which are then used as input to forecast the value of X_{t+1}. An example of using it combined with an RNN can be found in [FvA14]. The VAE idea has been extended to approximate posteriors with complex distributions, and not only normal distributions, through models called normalizing flows [RM15]. In these models, the posterior of the latent variables is modeled with invertible transformations starting from a simple distribution, for example normal, where the parameters of such transformations are estimated using a neural network. Recently, these techniques have proven to be successful to model complex distributions, such as speech and images, using autoregressive transformations [PPM17]. Furthermore, as these models capture the complex relationship between past and future data, they are able to generate new synthetic data that can be different to the train data, and that is coherent with the expected kind of output. For example, the model of [OLB+18] generates speech that is human like in many languages. These recent variational techniques, including normalizing flows, have been mostly used as generative models, and have been little explored for the task of time series probabilistic forecasting. In comparison with MCD, all these solutions require reparameterization of a neural

network parameters, thus requiring extra memory and training time to fit all of them.

Bibliographic notes

The work of [GG16a] was the first to propose using MCD for time series forecasting based on log likelihood maximization, although using a constant variance. Similarly, [TAS18] proposed using a similar Monte Carlo approach for batch normalization instead of dropout, applied to power generation, while not exploring the inherent uncertainty of time series forecasting. In [ZL17], MCD was extended to the MVE method with non-constant variance, applied to forecast the number of Uber trips in a day and a corresponding PI, using a normal distribution assumption. It focuses mostly on the coverage of the PIs produced with this technique, instead of focusing in general probabilistic forecasting tasks, and studies only the one-step forecasting case. The model proposed in [Sha19] is close to this thesis work, using an MVE model with normal distribution to forecast wind power, and applying MCD to handle the model uncertainty, differing from this work because they use a GRU recurrent network instead of an LSTM, they apply it only to wind power forecasting, and they do not explore other output distributions. The work of [RSS+20] proposes the use of normalizing flows to tackle complex distributions for forecasting, instead of mixture models or fixed distributions families, using a Masked Autoregressive Flow as the output of LSTM, GRU, or a transformer self attention network, applying it to power load data, among other datasets. This is one of the current directions of using variational techniques for forecasting, being able to handle big dimensionality and flexible distributions. They present opportunities for future work based on this thesis work, as most do not take into account the epistemic uncertainty.

2.2.6 Other methods based on neural networks

Input scenarios

Some authors model the distribution of X_{t+1} assuming that it should be empirically similar to previous observed data, while assuming there is uncertainty on other variables that the stochastic process can depend on. For example, wind power may depend on the temperature and wind speed. They use a neural network trained for point forecasting of X_{t+1} that receives other exogenous variables as input, and then inject many possible realizations of these exogenous input variables, called scenarios, generating an empirical output distribution. In general, these methods do not use neural networks specially devised for probabilistic forecasting, but change their input to produce different outputs on each simulation.

Delta method

A method that was used in the past to considerate the model uncertainty is the delta method, as explained by [CLR96], which builds confidence intervals for the model parameters. Confidence intervals refer to the uncertainty of the parameter values, in contrast with the PIs, which are related to the uncertainty of the prediction. When using this method, a normal noise term with constant variance σ^2 is assumed, and a linear approximation of the output of the neural network is considered, assuming there exists the best set of model parameters, θ^*, that are close to the estimated model parameters θ, and then

$$f_{\theta^*}(\mathbf{x}_t) \approx f_\theta(\mathbf{x}_t) + \nabla f^T(\theta - \theta^*),$$

where ∇f is the gradient of f with respect to θ. Then, the variance of the predictions can be approximated as $\sigma^2 + \sigma^2 \nabla f^T \left(\mathbf{F}^T \mathbf{F}\right)^{-1} \nabla f$, where $\mathbf{F}$ is the Jacobian matrix of $\left(f\left(\mathbf{x}_1\right), \ldots, f\left(\mathbf{x}_n\right)\right)$. While this method was used in the past for probabilistic forecasting, such as in [HD97], it has not been used recently in the literature, probably because neural networks are non-linear in nature, thus behaving badly when linear approximations are used, and also because the Jacobian matrix that has to be inverted may be huge, given that neural networks usually have too many parameters.

Bibliographic notes

Works with the input scenario approach include [MYL+14], where the model uncertainty is tackled with the use of an ensemble of point prediction models, thus not considering the inherent uncertainty, for the estimation of PI of wind speed and wind power, evaluated with example plots, but without using any quantitative metric. In [XH16], a simulation using different temperature scenarios is performed, and an ensemble is used, where one of them is a neural network that assumes a fixed normal variance over 24 hour groups, and then quantiles of the distribution of future values are estimated. The approach of [XHLK17] is similar, while using one model, thus not considering the model uncertainty. In [WCQL16], a combined approach using a CNN and a stacked LSTM, is used to make point forecasts using previous wind power and forecasted weather values. The data is segmented in groups of similar features, in which the data noise is modeled similar to a KDE approach. The point forecast is converted to a probabilistic forecasting using the modeled error of the group it falls into. The method shows superior performance in point forecasting when compared against other deep models, but does not perform comparisons in the probabilistic case, performing just a standard analysis of the residuals. Also, it does not consider the epistemic uncertainty.

A notable work using the delta method is [KNC10], which estimates the variance of predictions with the training residuals and then assumes a normal distribution, for power load PI forecasting, and was influenced by the work of [HD97] that preceded it a long time with the use of the delta method similarly.

2.3 Related work summary

There are plenty of approaches in the literature for probabilistic forecasting using neural networks, applied to wind power, wind speed or power load data, as listed in Table 2.1. They have different scopes, as some focus on the application of the models, while others tackle the difficulties of using neural network models for probabilistic forecasting tasks. Most of the reviewed works do not consider the epistemic uncertainty problem, and those that do so, are mostly based on ensemble methods. Some recent research has shown there are different alternatives to ensembles, but they have barely been explored for time series probabilistic forecasting. A few exceptions are using MCD, although assuming a normal distribution as the inherent noise distribution for the forecasting. In this work, it is shown that MCD is a valid alternative to ensemble models, and that other distributions may be assumed with minor changes to the base models.

2.4 Metrics

All the presented probabilistic forecasting methods may be compared in terms of different evaluation criteria, depending on the task that will be tackled by them. So far, the research community has developed many metrics to quantitatively evaluate the quality of models, which can also be useful to evaluate the proposal of this work. Thus, the most commonly used are described here, explaining what feature the metric focuses on when comparing models.

To describe them, it is assumed that metrics are applied over a test set with indices $\mathcal{G}$. As only one real observation is available for each time step in time series, some metrics focus on global averages instead of characteristics of each time step distribution.

2.4.1 Point predictions

Many probabilistic forecasting models, specially distributional assumption models may also be used to forecast not probabilistically, producing the median or the most expected future value. In this situation, the most common metrics used are the Mean Absolute Error (MAE), which allows to compare the models in terms of their median point forecasting performance:

$$\text{MAE} = \frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} |x_{t+1} - \tilde{x}_{t+1}|,$$

and the Root Mean Squared Error (RMSE), which allows to compare the models in terms of their mean point forecasting performance:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} (x_{t+1} - \tilde{x}_{t+1})^2}.$$

2.4.2 Prediction Intervals

Generally, PIs are assessed considering the following three criteria, based on the lower and upper bounds of the PI. Note that the metrics depend on the predicted bounds $\left[\tilde{L}^c_{t+1}, \tilde{U}^c_{t+1}\right]$ associated with a nominal confidence c:

- *Coverage*: The coverage is related to the proportion of observations that lay between the bound of the PI. Ideally, all the observations should be inside the PI, but in reality this does not always happen, thus it is possible to measure the global proportion, usually called PICP:

$$\text{PICP} = \frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} \mathbb{I}\left(x_{t+1} \in \left[\tilde{L}^c_{t+1}, \tilde{U}^c_{t+1}\right]\right),$$

  where $\mathbb{I}$ is the indicator function. It is important for the PI that PICP is at least the desired confidence c used to tune the model.

- *Sharpness*: It would be easy to satisfy the coverage requirement by just making the PIs as wide as possible, but then they would be useless. Because of this, it is desired that they be as narrow as possible, while still covering the required confidence c. A global metric of the width of the PI is the PINAW, that measures the average width of the PIs, compared to the range of data values:

$$\text{PINAW} = \frac{1}{R\,|\mathcal{G}|} \sum_{t \in \mathcal{G}} \left(\tilde{U}^c_{t+1} - \tilde{L}^c_{t+1} \right), \text{ where } R = \max_{t \in \mathcal{G}} [x_{t+1}] - \min_{t \in \mathcal{G}} [x_{t+1}].$$

- *Resolution*: When the task is associated with a time series having heteroscedasticity, the width of the PI should change in time, adapting to the process uncertainty of each time period. For this, two metrics are usually computed in the community. First, the CWC, which considers both PI width and coverage, adding an exponential penalty when the coverage is not met:

$$\text{CWC} = \text{PINAW} + e^{-\eta(\text{PICP}-c)} \mathbb{I}\,(\text{PICP} < c),$$

with η a constant, usually 50. In practice, it favors models that correctly cover the given confidence, and narrow PIs among them. Secondly, the Winkler Loss, or IS, which gives a linear penalty to points laying out of the PI, instead of an exponential global penalty, thus being much less affected when the coverage is not completely met:

$$\text{IS} = \frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} \left(2\,(1-c) \left(\tilde{U}^c_{t+1} - \tilde{L}^c_{t+1} \right) + 4 \begin{cases} \tilde{L}^c_{t+1} - x_{t+1} & \text{if } x_{t+1} < \tilde{L}^c_{t+1}, \\ x_{t+1} - \tilde{U}^c_{t+1} & \text{if } \tilde{U}^c_{t+1} < x_{t+1}, \\ 0 & \text{otherwise.} \end{cases} \right).$$

As it is computed in a per-point basis, it is possible to identify which are the points greatly increasing it. Some authors use this same metric, but divided by the constant $2\,(1-c)$.

There are other less commonly used metrics for PIs, which are not included here, but are referenced in Table 2.1 for a curious reader.

2.4.3 Quantiles

For the case of QR, a model returns a list of predicted quantile cuts $\tilde{q}^{\alpha_1}_{t+1}, \ldots, \tilde{q}^{\alpha_Q}_{t+1}$ associated with different levels $\alpha_1, \ldots, \alpha_Q$ of the CDF, and it is usually assessed with the PL, already mentioned in Eq. (2.7), also called Quantile Score, which gives a linear penalty that is equal to the difference with the real observation, but with a different slope depending on the observation being up or down the cut value:

$$\text{PL}_\alpha = \frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} \begin{cases} \alpha \left(x_{t+1} - \tilde{q}^\alpha_{t+1} \right), & \text{if } x_{t+1} \geq \tilde{q}^\alpha_{t+1}, \\ (1-\alpha) \left(\tilde{q}^\alpha_{t+1} - x_{t+1} \right), & \text{if } x_{t+1} < \tilde{q}^\alpha_{t+1}. \end{cases}$$

It is important to note that, in the case of $\alpha = 0.5$, associated with the median of the distribution, this is equivalent to the MAE. In general, the average of all levels is reported, thus $\text{PL} = \frac{1}{Q} \sum_{i=1}^{Q} \text{PL}_{\alpha_i}$.

2.4.4 Distribution

When considering distributional methods, specially those based on sampling, it is possible to assess the obtained empirical distribution, and the CRPS is used. This metric computes the L_2 distance between the obtained empirical CDF and a Dirac delta distribution where the real observation is. Specifically, if F is the estimated distribution CDF,

$$\text{CRPS} = \frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} \int_{-\infty}^{+\infty} \left(F\left(y\right) - \mathbb{I}\left(x_{t+1} \leq y\right)\right)^2 dy.$$

It has been shown [GK14] that this is equivalent to

$$\text{CRPS} = \frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} \left(\mathbb{E}_{y \sim F}\left[\,|y - x_{t+1}|\right] - \frac{1}{2}\mathbb{E}_{y \sim F, z \sim F}\left[\,|y - z|\right] \right),$$

which is approximated using the I samples of the empirical distribution, as the following sum:

$$\text{CRPS} \approx \frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} \left(\frac{1}{I} \sum_{i=1}^{I} \left|\tilde{x}_{t+1}^{i} - x_{t+1}\right| - \frac{1}{2I^2} \sum_{i=1}^{I} \sum_{j=1}^{I} \left|\tilde{x}_{t+1}^{i} - \tilde{x}_{t+1}^{j}\right| \right).$$

In this expression, the first term is the MAE of all predictions, while the second term is a measure of dispersion between samples. Hence, a low CRPS means, by it first term, that the prediction error of the samples is low, thus close to the target value, and by its second term, that the difference among the samples is high, thus being far between them. This ensures that obtained samples are accurate and diverse enough to represent the potential distribution around the target value.

Table 2.1: Probabilistic forecasting works reviewed in this work. MS stands for multistep forecasting, L for power load, WS for wind speed, WP for wind power, and PV for photovoltaic power.

Data	Task	Inherent Uncertainty	Model Uncert.	MS	Metrics	Extra Inputs	Ref.
Others	PI	LUBE	—	N	CWC, PICP, PINAW		[KNCA11]
WP	PI	LUBE	—	N	ACE, IS, PICP		[WXP13]
WP	PI	LUBE	—	N	ACE, CWC, IS, PICP		[WXP+14a]
WP	PI	LUBE	—	N	CWC, PICP, PINAW, PIMSE		[SLD18a]
WS	PI	LUBE	—	N	ACE, AWD, CWC, IS, PICP, PINAW		[SLD18b]
WP	PI	LUBE	—	N	CWC, PICP, PINAW, PIRD		[WLCC19]
L, WP	PI	LUBE	—	Y	CWC, PICP, PINAW, PINRW		[QSK14]
L, WP	PI	LUBE	—	Y	ACE, CWC, IS, PICP, PINAW		[KKKF+21]
L, WP	PI	LUBE + input sensitivity	—	Y	CWC, PICP, PINAW		[KKNN20]
L, others	PI	LUBE	Ensemble	N	PICP, PINAW		[PZBN18]
WP	PI	LUBE	Ensemble	N	CWC, PICP, PINAW		[KN13]
WP	PI	LUBE	EMD ensemble	Y	CWC, PICP, PINAW		[ZWW+15]
WP	PI	LUBE	Fuzzy ensemble	N	PICP, PINAW		[KFKN16]
WP	PI	LUBE	Bootstrap	N	ACE, IS, PICP, PINAW		[MDM18]
WP	PI — Normal	Fixed normal	—	N		WS	[MYLL07]
WP	PI	Fixed T-LS	—	N	AWD, ACE, PICP, PINAW	WS & Dir.	[WNL+19]
Others	PI	Fixed normal	Delta method	N	PICP		[CLR96]
L	PI	Fixed normal	Delta method	N	PICP, PINAW, Plots	Temp.	[HD97]
L	PI	Fixed normal	Delta method	Y	CWC, PICP, PINAW		[KNC10]
Others	Distrib.	Fixed normal	MCD	Y	log likelihood, perplexity		[GG16a]
WS	PI	Fixed normal	Fuzzy ensemble	Y	ACE, IS		[CZD+18]
L	QR	Fixed discrete	Grad. boosting	Y	PL	Temp., Irrad.	[WZC+18]
WP	PI	Fixed empirical PDFs by groups	—	N	ACE, IS	WS	[WCQL16]
WP, WS	PI	Point predictions	Ensemble	Y	Plots		[MYL+14]
WP	QR	Input scenarios	Ensemble	Y	APD, CRPS, PINAW	WS & Dir.	[SH12]
L	QR	Input scenarios, Fixed normal by groups	—	Y	PL	Temp.	[XH16]
L	QR	Input scenarios, Fixed normal by groups	—	Y	PL	Temp., GDP	[XHLK17]
WP	QR	QR	—	Y	PICP, PINAW, PL, Plots	WS	[HNM14]
WS	PI, QR	QR	—	Y	ACE, CRPS, IS, PICP		[WWL+16]
WP	PI, QR	QR	—	Y	ACE, IS, PICP, PL	WS	[HLSK17]
L, others	QR	QR	—	Y	PL	Temp.	[WTNM17]
WP	PI, QR	QR	—	Y	APD, PL		[WLW+17a]
L	QR	QR	—	N	PL	Temp.	[GWYK18]
WP	PI, QR	QR	—	Y	PICP, PINAW, PL		[HL18]
L	PI, QR	QR	—	Y	IS, PL	Temp.	[ZQS19]
L	PI, QR	QR	—	Y	IS, PL	Temp.	[ZQG+20]
L, others	QR	QR	—	Y	PL		[LALP19]
L, others	QR, sampling	QR	—	N	CRPS		[GRK+21]
L, others	QR, sampling	QR with splines	—	Y	IS, PL		[GBW+19]
L	QR	QR	Ensemble	Y	PL		[WZT+18]
L	QR	QR with point forecasts as input	Ensemble	Y	IS, PL	Temp.	[LNHW17]
WP	PI	LUBE & QR	Ensemble	N	CWC, PICP, PINAW		[LZM+20]
Others	Distrib.	MVE — Normal	—	N	Plots		[NW94]
WP	PI	MVE — Normal	—	N	CWC, PICP, PINAW		[KN14]
Others	Distrib.	MVE — Weibull	—	N	log likelihood, Plots		[Egi16]
L, others	Distrib.	MVE — Normal	—	Y	PL, Coverage (quantile PICP), Plots		[SFGJ20]
L, others	Distrib.	MVE — Normal, GP, others	—	Y	PL, Coverage (quantile PICP), Plots		[WSM+19]
WP	PI	MVE — Normal	Bootstrap	Y	CWC, PICP, PINAW		[KNC13]
WP	PI	MVE — Censored Gaussian or Beta	Bootstrap	Y	ACE, IS, PICP		[WXP+14b]
WP	PI, Distrib.	MVE — Normal	Ensemble	Y	ACE, CRPS, IS		[WLW+17b]
Uber trips	PI	MVE — Normal	MCD	N	PICP		[ZL17]
WP	Distrib.	MVE — Normal	MCD	Y	CRPS	WS & Dir.	[Sha19]
L, PV, others	Distrib.	MVE & Gaussian Copula	—	Y	CRPS		[SBSC+19]
L, PV, WP	PI, Distrib.	MVE, QR & Empirical Copula	—	Y	PL		[TBVD19]
Others	Distrib.	Mixture of Gaussians	—	N	Qualitative		[Bis94]
Others	Distrib.	Mixture of Gaussians	—	Y	log likelihood, bits p/ char, perplexity		[Gra13]
WP	Distrib.	Mixture of Gaussians	—	?	PL	NWP	[FKG10]
L	Distrib.	Mixture of Gaussians	—	Y	CRPS, Plots		[VFM18]
L	Distrib.	Mixture of Gaussians	—	?			[SAR+19]
WP	Distrib.	Mixture of Gaussians	—	Y	PL	NWP, WS	[ZYI+19]
WP	Distrib.	Mixture of Gaussians or Betas	—	N	ACE, CRPS, IS, PINAW	NWP, WS	[ZLY+20]
WP, WS	PI, Distrib.	Mixture of Gaussians	Ensemble	Y	CRPS		[MYL+16]
Others	Distrib.	Mixture of GARCH models	Bootstrap	N	Hit Rate (HR) (PICP-like)		[NTS13]
L, others	Distrib.	Normalizing flows	—	Y	CRPS		[RSS+20]
WS	PI, Distrib.	CVAE	Bootstrap	?	CRPS, Dispersion, RH	NWP	[FRHC21]

Chapter 3

Problem Definition and Proposal

3.1 Problem Definition

Previous chapters have introduced the concepts and general challenges of probabilistic forecasting, showing that models have to handle both the aleatoric uncertainty and epistemic uncertainty. For deep learning models, the epistemic uncertainty problem gets exacerbated, because they have a big number of processing units and corresponding parameters. While ensemble methods may be used to alleviate this problem, this can be a costly process in terms of computational time or memory, limiting the applicability on real probabilistic forecasting tasks. At the same time, most current solutions provide forecasting models that consider a normal distribution assumption for the aleatoric uncertainty. While this is enough as an initial approximation, real time series are rarely normally distributed, so it is needed to explore other output distribution assumptions that are more flexible among parameterized distributions. The importance of handling the epistemic uncertainty relies on the difficulty of obtaining enough training data to train deep learning models for time series probabilistic forecasting, while the importance of handling the aleatoric uncertainty relies on that, even in the case of being able to gather more data, there is an intrinsic noise that the model has to correctly cope, because important related variables may not be available. This work focuses on the problem of developing a deep learning probabilistic forecasting model that can handle aleatoric uncertainty and epistemic uncertainty, allowing to properly forecast future scenarios and finally take more informed decisions, avoiding the problems identified in current deep learning solutions, based on ensembles or assuming a normal output distribution. A model is proposed to handle the epistemic uncertainty applying MCD to a stacked LSTM, while handling of the aleatoric uncertainty by interpreting the outputs of the neural network as the parameters of a distribution. LSTM is a deep learning architecture known to work well for sequential data. The output distribution may be chosen among normal, as the MVE method does, or other families, such as Weibull, Log-normal, or Gamma.

3.2 Main hypothesis

The use of deep learning models based on variational inference with different output distribution assumptions allows producing probabilistic forecasting (Prediction Intervals and/or quantiles) for time series that are better, in terms of metrics currently used by the community, to those produced by deep learning models with the same architecture with fixed parameters and with fixed normal output distribution.

3.3 General objective

The objective of this proposal is to design and implement a deep learning probabilistic forecasting algorithm based on variational inference techniques that handles both epistemic uncertainty and aleatoric uncertainty, and apply it to time series of the energy domain.

3.4 Specific objectives

1. Review state-of-the-art deep learning models for probabilistic forecasting in the electricity domain.
2. Design and implement a deep learning probabilistic forecasting model based on current variational inference tools.
3. Adapt the proposed model for the tasks of PI estimation, QR and scenario generation.
4. Compare the model with similar state-of-the-art models that assume only normally distributed outputs, based on probabilistic forecasting metrics currently used by the community, such as PL, CWC, IS and CRPS.
5. Compare the model with an ensemble of models in terms of consideration of the epistemic uncertainty.
6. Compare different approaches to perform multistep probabilistic forecasting with the model.

3.5 Proposal

As mentioned, the general proposal of this work is a deep learning model to handle both the epistemic uncertainty and aleatoric uncertainty. For the former, MCD is applied to a stacked LSTM, while for the second, the outputs of the neural network are interpreted as the parameters of a distribution, as the MVE method does. This is explained in detail in the following.

3.5.1 Monte Carlo Dropout and epistemic uncertainty

When applying variational inference, as explained in Section 2.2.5, the posterior distribution of the parameters, $p(\theta \mid \mathcal{D})$, is approximated using another distribution $q(\theta \mid \phi)$, parameterized by a set of new parameters ϕ. These parameters ϕ are trained maximizing the ELBO defined in Eq. (2.12):

$$\text{ELBO}(\mathcal{D}, \phi) = \mathbb{E}_{\theta \sim q(\theta|\phi)} [\log p(\mathcal{D} \mid \theta)] - D_{\text{KL}}(q(\theta \mid \phi) \parallel p(\theta)) \leq \log p(\mathcal{D}), \tag{3.1}$$

where $\mathcal{D}$ represents the set of training data. While this is the general variational inference approach, the particular choice of $q(\theta \mid \phi)$ has to be done. A straightforward distribution to use is the derived when using dropout: instead of representing each neural network weight with one fixed value, model it as a Bernoulli distribution, with a fixed probability of having that value, and otherwise having value zero. This new representation does not require more computer memory, as the number of stored values is the same number as the number of weights. With this assumption for one particular weight of the neural network, and a standard Gaussian prior, the second term of Eq. (3.1) translates into

$$\begin{aligned} -D_{\text{KL}}(q(\theta \mid \phi) \parallel p(\theta)) &= -d \log \frac{d}{p(0)} - (1-d) \log \frac{1-d}{p(\phi)} \\ &= -(d \log d + (1-d) \log(1-d)) - \frac{d}{2} \log(2\pi) - \frac{1-d}{2} \log(2\pi) - \frac{1-d}{2} \phi^2 \\ &= \mathcal{H}(d) - \frac{1}{2} \log(2\pi) - \frac{1-d}{2} \phi^2, \end{aligned}$$

where d is the dropout probability, that is, the probability of setting the parameter to 0, and $\mathcal{H}(d)$ is the entropy of a Bernoulli random variable with parameter d. Additionally, the first term can be written as

$$\mathbb{E}_{\theta \sim q(\theta|\phi)} [\log p(\mathcal{D} \mid \theta)] = d \log p(\mathcal{D} \mid \theta = 0) + (1-d) \log p(\mathcal{D} \mid \theta = \phi).$$

In reality, there are many weights in the neural network, hence this first term is given by a sum of likelihoods for all possible weights, each one weighted by the probability of those particular weights given the Bernoulli distributions for each one. Similarly, the second term requires adding the squared value of all weights, according to the value given after applying dropout, and again each weighted by the probability of the weights happening. These sums would require considering an exponential number of possible values for the weights, thus being impractical. Instead, a simple approximation may be computed applying a Monte Carlo procedure to these sums. The idea is to sample weight values $\tilde{\theta}$ from $q(\theta \mid \phi)$ and then approximate Eq. (3.1) as the sum

$$\text{ELBO}(\mathcal{D}, \phi) \approx \mathcal{H}(d) + \frac{1}{B} \sum_{\tilde{\theta} \sim q(\theta|\phi)} \left(\log p\left(\mathcal{D} \mid \tilde{\theta}\right) - \frac{1}{2} \phi^2 \right),$$

with B the number of samples. Here it has been assumed that the dropout probability is the same for all neural network weights. In general, the dropout rate is fixed before starting the training of a

neural network, thus being considered a hyperparameter. Assuming this, the ELBO maximization can be written as a minimization of its negative value, having

$$\max_{\phi} \text{ELBO}\left(\mathcal{D}, \phi\right) = -\min_{\phi} \text{ELBO}\left(\mathcal{D}, \phi\right) \approx \min_{\phi} \sum_{\tilde{\theta} \sim q(\theta|\phi)} \left(-\log p\left(\mathcal{D} \mid \tilde{\theta}\right) + \frac{1}{2}\phi^2 \right). \tag{3.2}$$

Note that the term ϕ^2 is in fact representing the squared Euclidean norm of the vector of weights, thus resembling the ℓ^2 regularization term usually added to neural network training, also called weight decay. Additionally, the first term is the negative log-likelihood of the sampled weights, to be minimized. Then, maximizing the ELBO can be seen as training the neural network maximizing the likelihood, and using weight decay. This insight has been the base of how MCD is used to handle the epistemic uncertainty for deep learning models: rethinking the model training using dropout, allows exploring different weight values according to the implicit Bernoulli distributions for each one. As the number of weights of the neural network is generally big, there is a huge number of possible combinations of disabled weights, thus having a potentially big set of different outputs of the neural network, for the same input. Therefore, the epistemic uncertainty handling with MCD may produce quite complex output distributions for a particular input value, given only the uncertainty of the weights.

This general approach needs to be adapted for time series probabilistic forecasting. As explained in Section 2.1, the training set $\mathcal{D}$ can be considered as a set of tuples $\{(\mathbf{x}_t, x_{t+1})\}$, where $\mathbf{x}_t$ is a sequence of M previous values, and x_{t+1} the first value after them. Considering this, the first minimization term in Eq. (3.2) can be rewritten in terms of the PDF of each sample in the training set, assuming each is independent, and then Eq. (3.2) is written as:

$$\sum_{\tilde{\theta} \sim q(\theta|\phi)} \left(\sum_{(\mathbf{x}_t, x_{t+1}) \in \mathcal{D}} -\log p\left(x_{t+1} \mid \mathbf{x}_t, \tilde{\theta}\right) + \lambda\phi^2 \right), \tag{3.3}$$

where a hyperparameter λ was added to the weight decay term, as its importance as a regularization term will have to be tuned this way. Moreover, for the training of the neural network, stochastic gradient descent-based algorithms are used, that instead of evaluating this expression for all training samples in $\mathcal{D}$, they select a random batch $\mathcal{B}$ of fixed size, defined by a subset $\mathcal{B} \subset \mathcal{D}$. This process adds more stochasticity to the training process, and the common practice is that, for each batch $\mathcal{B}$, a single sample $\tilde{\theta}_{\mathcal{B}}$ is obtained, producing a dropout mask that is similar for all elements in the batch. Then, Eq. (3.3) is evaluated using the sampled weights $\tilde{\theta}_{\mathcal{B}}$ only, assuming that the stochasticity of the algorithm will maximize the ELBO globally, even using only one sample $\tilde{\theta}_{\mathcal{B}}$ at each batch step of the algorithm. Therefore, for each batch $\mathcal{B}$, Eq. (3.3) is approximated by the following expression, which corresponds to the batch loss:

$$\sum_{(\mathbf{x}_t, x_{t+1}) \in \mathcal{B}} -\log p\left(x_{t+1} \mid \mathbf{x}_t, \tilde{\theta}_{\mathcal{B}}\right) + \lambda\phi^2. \tag{3.4}$$

3.5.2 Output distribution and aleatoric uncertainty

After defining how the epistemic uncertainty is considered, and how an optimization algorithm can train the neural network weights, it is needed to discuss how to handle the aleatoric uncertainty. The conditional PDF $p\left(x_{t+1} \mid \mathbf{x}_t, \tilde{\theta}_{\mathcal{B}}\right)$ in Eq. (3.4) equals to the PDF of the distribution assumed as output of the model. For example, when a normal distribution is assumed, as in the MVE model, it equals to

$$\log p\left(x_{t+1} \mid \mathbf{x}_t, \tilde{\theta}_{\mathcal{B}}\right) = \log p_{\mathcal{N}\left(\mu_{\tilde{\theta}_{\mathcal{B}}}(\mathbf{x}_t), \sigma^2_{\tilde{\theta}_{\mathcal{B}}}(\mathbf{x}_t)\right)}\left(x_{t+1}\right) = -\frac{1}{2}\left[\log 2\pi + \log \sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right) + \frac{\left(x_{t+1} - \mu_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)\right)^2}{\sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)}\right],$$

where $\mu_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)$ and $\sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)$ are the network outputs. Based on this likelihood, and removing constants, the loss from Eq. (3.4) is then

$$\sum_{(\mathbf{x}_t, x_{t+1}) \in \mathcal{B}} \left(\log \sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right) + \frac{\left(x_{t+1} - \mu_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)\right)^2}{\sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)} \right) + \lambda \tilde{\theta}^2_{\mathcal{B}}. \tag{3.5}$$

The real distribution of data is rarely normal. Part of this proposal is to change the normal assumption done by most models, allowing the modeler to choose any parametric distribution. To do this, the outputs of the neural network are interpreted as a vector $\tilde{\mathbf{p}}_{t+1}$ of parameters of the chosen distribution, and the corresponding PDF is used during training as the likelihood in Eq. (3.4). For example, if it is known that the values are always positive, a log-normal distribution may be assumed, and the minimization of Eq. (3.4) changes into the following loss, after removing normalization constants:

$$\sum_{(\mathbf{x}_t, x_{t+1}) \in \mathcal{B}} \left(2 \log x_{t+1} + \log \sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right) + \frac{\left(\log x_{t+1} - \mu_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)\right)^2}{\sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)} \right) + \lambda \phi^2.$$

As the minimization is over the parameters of the network, the first term is constant, so it can be discarded from the objective function, leading to

$$\sum_{(\mathbf{x}_t, x_{t+1}) \in \mathcal{B}} \left(\log \sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right) + \frac{\left(\log x_{t+1} - \mu_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)\right)^2}{\sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)} \right) + \lambda \phi^2.$$

As the square value is positive, the optimization will try to make $\sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)$ values as big as possible to reduce the value of the second term, and, at the same time, will try to make $\log \sigma^2_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right)$ as little as possible, to reduce the value of the first term, thus reaching an equilibrium. An important change when using a log-normal distribution, in comparison with Eq. (3.5), is the fact that the objective function is non-symmetrical with respect to x_{t+1}, because it is minimized when $\mu_{\tilde{\theta}_{\mathcal{B}}}\left(\mathbf{x}_t\right) = \log x_{t+1}$, but its value changes differently when x_{t+1} grows than when it is decreased.

As seen, it is possible to change the distribution assumed for a predicted value, by changing the loss function that is minimized during the network training, without changing the MCD training procedure. This, in turn, allows handling time series with different kinds of inherent noise distributions, and considering the aleatoric uncertainty. This study limited to distributions that have only two parameters, shown

Table 3.1: Distributions considered in this work, parametrized with the outputs of a neural network.

Distribution	Parameters	PDF on a value y
Normal	μ, σ^2	$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$
Log-normal	μ, σ^2	$\frac{1}{y\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\log y-\mu)^2}{2\sigma^2}\right)$
Weibull	k, λ	$\frac{k}{\lambda}\left(\frac{y}{\lambda}\right)^{k-1} \exp\left(-\left(\frac{y}{\lambda}\right)^k\right)$
Gamma	α, β	$\frac{\beta^\alpha}{\Gamma(\alpha)} y^{\alpha-1} \exp\left(-\beta y\right)$

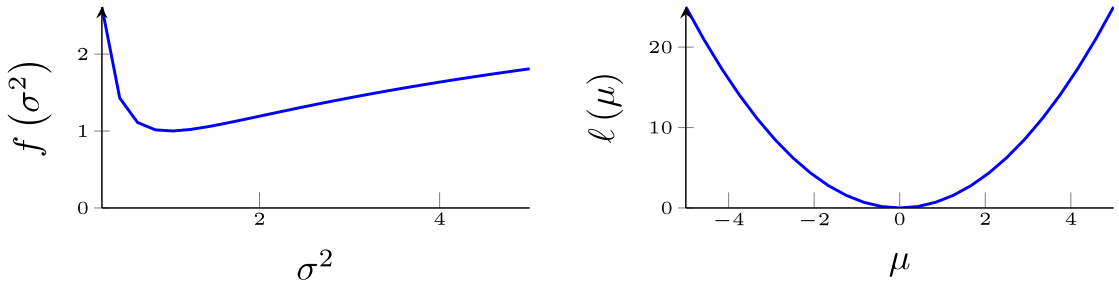


Figure 3.1: How the loss changes when the parameters of a normal or a log-normal distribution change. Left: $f\left(\sigma^2\right) = \log\sigma^2 + \frac{c}{\sigma^2}$, for $c = 1$, which is non convex, but has one minimum. Right: $\ell\left(\mu\right) = \mu^2$.

in Table 3.1. Aside from the normal one, these distributions have non-negative domain and unbound by high values, allowing to properly model data that is non-negative, such as power load, wind speed, and others. Additionally, all the same number of parameters: two, therefore they keep the neural network with the same complexity, than when assuming a normal distribution. In principle, it is possible to add more outputs to the neural network if a distribution with more parameters is required.

To better understand how the loss changes when changing the output distribution, Figure 3.1 shows loss function values in terms of the outputs μ and σ^2, either for a normal distribution, or for a log-normal distribution. In practice, instead of interpreting the output of the network directly as σ^2, it is interpreted as $\log\sigma^2$, or as $\log\left(e^\sigma - 1\right)$, to avoid adding a constraint to have only non-negative values. Using this trick, the loss function behaves as shown in Figure 3.2.

A similar analysis can be made for other distributions, even for some that are not part of the exponential family. For example, when a Weibull distribution is assumed, Eq. (3.4) leads to

$$\sum_{(\mathbf{x}_t, x_{t+1})\in\mathcal{B}} \left(\left(\frac{x_{t+1}}{\lambda_{\tilde{\theta}_\mathcal{B}}\left(\mathbf{x}_t\right)}\right)^{k_{\tilde{\theta}_\mathcal{B}}\left(\mathbf{x}_t\right)} - k_{\tilde{\theta}_\mathcal{B}}\left(\mathbf{x}_t\right) \log \frac{x_{t+1}}{\lambda_{\tilde{\theta}_\mathcal{B}}\left(\mathbf{x}_t\right)} - \log k_{\tilde{\theta}_\mathcal{B}}\left(\mathbf{x}_t\right) \right) + \lambda\phi^2. \tag{3.6}$$

In this case, the optimization will try to make $\lambda_{\tilde{\theta}_\mathcal{B}}\left(\mathbf{x}_t\right)$ big to reduce the first term, which is always positive, and, at the same time, will make it little to increase the second term, which is subtracted, reaching an equilibrium that is also non-symmetrical with respect to x_{t+1}. This can clearly be seen in Figure 3.3, where it is shown how the loss changes when each of the distribution parameters change, depending on the interpretation of the output of the network. Therefore, using the proposed approach

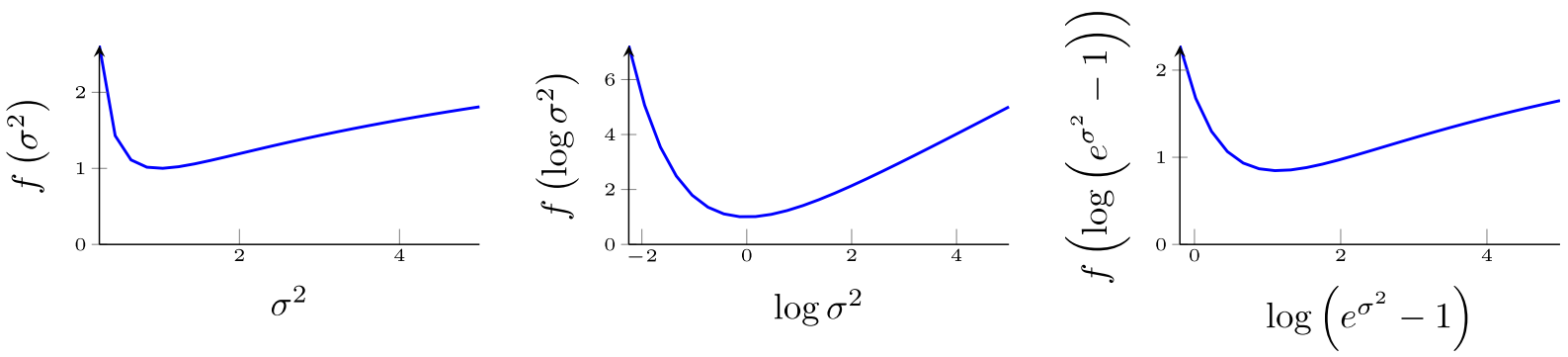


Figure 3.2: How the loss changes depending on the variance parameter representation of a normal or a log-normal distribution. Left: Same as left in Figure 3.1. Center: $f(\eta) = \eta + \frac{c}{e^{\eta}}$, with $c = 1$, parameterized by $\eta = \log \sigma^2$, to avoid the non-negativity constraint. Right: $f(\eta) = \log\log(1 + e^{\eta}) + \frac{c}{(\log(1+e^{\eta}))^2}$, another common parameterization used for non-negative variances, assuming $\eta = \log\left(e^{\sigma^2} - 1\right)$.

to change the output distribution, and applying the optimization procedure previously explained, it is possible to tackle prediction tasks even when the time series distribution is not part of the exponential family.

3.5.3 Probabilistic forecasting with the proposed model

In the previous sections, it was explained how to build a custom loss for a neural network model according to an output distribution, and how an optimization algorithm can be used to minimize it, using dropout. To summarize these ideas, and to clearly state a concrete training algorithm based on them for time series, the steps to be performed are presented in Algorithm 1. The whole point of training a neural network with this algorithm is to produce a probabilistic forecasting model to get estimations of the distribution of future values. To do so, it is important to note that dropout is commonly applied during training as regularizer, with a similar procedure as the one explained here, but then the weights are fixed during inference. With MCD, instead, the weights are sampled during inference on new test data as well, thus exploring the approximate posterior distribution $q(\theta \mid \phi)$ given by dropout. In other words, when making inference using the model, a sampling procedure is performed for the parameters θ, similarly as done during the model training. Recall from Eq. (2.4) that the PDF of the forecasted variable X_{t+1} that is being estimated is

$$p(x_{t+1} \mid \mathbf{x}_t, \mathcal{D}) = \int_{\theta \in \Theta} p(x_{t+1} \mid \mathbf{x}_t, \theta) p(\theta \mid \mathcal{D}) d\theta.$$

Instead of integrating over the unknown posterior $p(\theta \mid \mathcal{D})$, I dropout samples $\tilde{\theta}^1, \ldots, \tilde{\theta}^I$ of network weights are obtained, and the target PDF is approximated with a Monte Carlo sum, as

$$p(x_{t+1} \mid \mathbf{x}_t, \mathcal{D}) \approx \frac{1}{I} \sum_{i=1}^{I} p\left(x_{t+1} \mid \mathbf{x}_t, \tilde{\theta}^i\right).$$

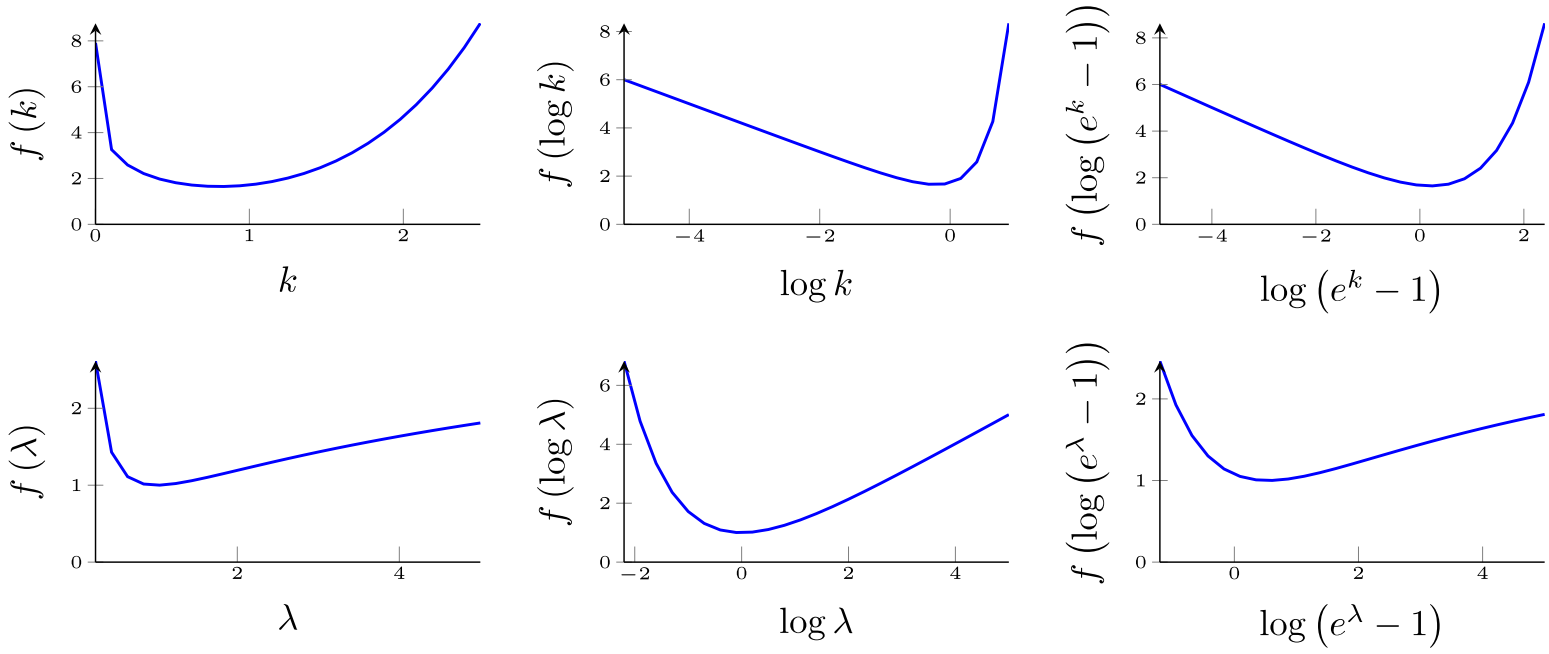


Figure 3.3: How the loss changes with the parameters of a Weibull distribution, changing the parameter representation in the last neural network layer.

Algorithm 1 Neural network model training

Require: Training values $x_1, \ldots, x_T$, output distribution family $\mathcal{F}$, ℓ_2 regularization weight λ, batch size B, number of epochs E.

Initialize network weights θ with random values.
for epoch $e = 1$ **to** E **do**
  for batch $j = 1$ **to** $\lceil T/B \rceil$ **do**
    $\tilde{\theta} \leftarrow$ Sample weights applying dropout to current parameters θ.
    $L \leftarrow 0$. {Total batch log likelihood}
    for batch sample $b = 1$ **to** B **do**
      $t \leftarrow$ One random training time step.
      $\mathbf{x} \leftarrow (x_{t-M+1}, \ldots, x_t)$. {Neural network input}
      $\tilde{\mathbf{p}} \leftarrow$ Evaluate network with input $\mathbf{x}$ and weights $\tilde{\theta}$ to get vector of distribution parameters.
      $L \leftarrow L + \log p(x_{t+1} \mid \tilde{\mathbf{p}}) = L + \log p_{\mathcal{F}(\tilde{\mathbf{p}})}(x_{t+1})$. {Log likelihood according to distribution}
    end for
    Loss $\leftarrow -L + \lambda\theta^2$.
    Update θ applying gradient descent algorithm, based on Eq. (3.4).
  end for
end for
return Trained network weights θ.

Note that this approximation is similar to the one used by ensemble models in Eq. (2.11), although in this case, all sampled parameters $\tilde{\theta}^i$ have the same weight $\frac{1}{I}$, assuming that the most probable parameter values will appear more times in the sampling process, having more weight in the sum. In summary, MCD samples the network weights during training *and* during inference, producing Monte Carlo approximations of integrals, and hence its name. As the final PDF depends on the PDF of several distributions that may have different parameters, it may be quite complex. Thus, instead of computing the value of the PDF at any given value x_{t+1} of X_{t+1}, what it is done is to get samples of the distributions given by each parameters vector $\tilde{\mathbf{p}}^i_{t+1}$ obtained using the $\tilde{\theta}^i$ weights, getting a set of samples of the distribution of X_{t+1}. Using these samples, the mean can be computed for point forecasting, PIs may be estimated, QR may be performed, or the empirical distribution may be used directly. Therefore, sampling the neural network parameters and the output distributions can be used to perform general probabilistic forecasting tasks.

3.5.4 Multistep probabilistic forecasting

So far the model for one-step forecasting has been discussed. It needs to be extended for multistep probabilistic forecasting, considering that the distribution of further future values, like X_{t+2}, may depend on the distribution of previous values, in this case X_{t+1}. Although the model presented so far is trained to perform one-step ahead forecasts, it can be used to produce probabilistic forecasting for horizons larger than a single time step considering this sequential dependence, adapting it like follows: a sample $\tilde{x}_{t+1}$ from the distribution defined by $\tilde{\mathbf{p}}_{t+1}$ is obtained, and then it is used as input for the following time step, building a new input $\tilde{\mathbf{x}}_{t+1}$, which is made discarding the oldest member of $\mathbf{x}_t$ and concatenating the new sample. Therefore, if $\mathbf{x}_t = (x_{t-M+1}, \ldots, x_t)$, then $\tilde{\mathbf{x}}_{t+1} = (x_{t-M+2}, \ldots, x_t, \tilde{x}_{t+1})$. Finally, the network is evaluated on this new input, producing a new set of distribution parameters $\tilde{\mathbf{p}}_{t+2}$ and a new sample $\tilde{x}_{t+2}$, and so on, until the desired number of steps K is obtained, as described in Figure 3.4, producing a sample trajectory of future values, also called scenario. To apply MCD, one particular dropout mask is obtained before producing each sample trajectory, similarly to the one-step case, thus producing I sample output trajectories, each one of length K time steps, as specified in Algorithm 2, allowing to explore the output distribution for each time step empirically. Regarding the computational time complexity of Algorithm 2, it is similar to any sampling mechanism given by evaluating a RNN I times. Then, it is given by $O\left(I\left(M+K\right)D\right)$ if evaluated sequentially, where D is the time complexity of evaluating the recurrence of the neural network. D is given by the number of units and layers of the LSTM and dense layers. Fortunately, it is possible to parallelize the evaluation of each weight sample $\tilde{\theta}$, reducing the time complexity to $O\left(\left(M+K\right)D\right)$.

The obtained samples for each time step can then be used to estimate either PI or perform QR. For the former, the values are sorted and the ones that correspond to the proportions $\frac{1-c}{2}$ and $\frac{1+c}{2}$ are returned,

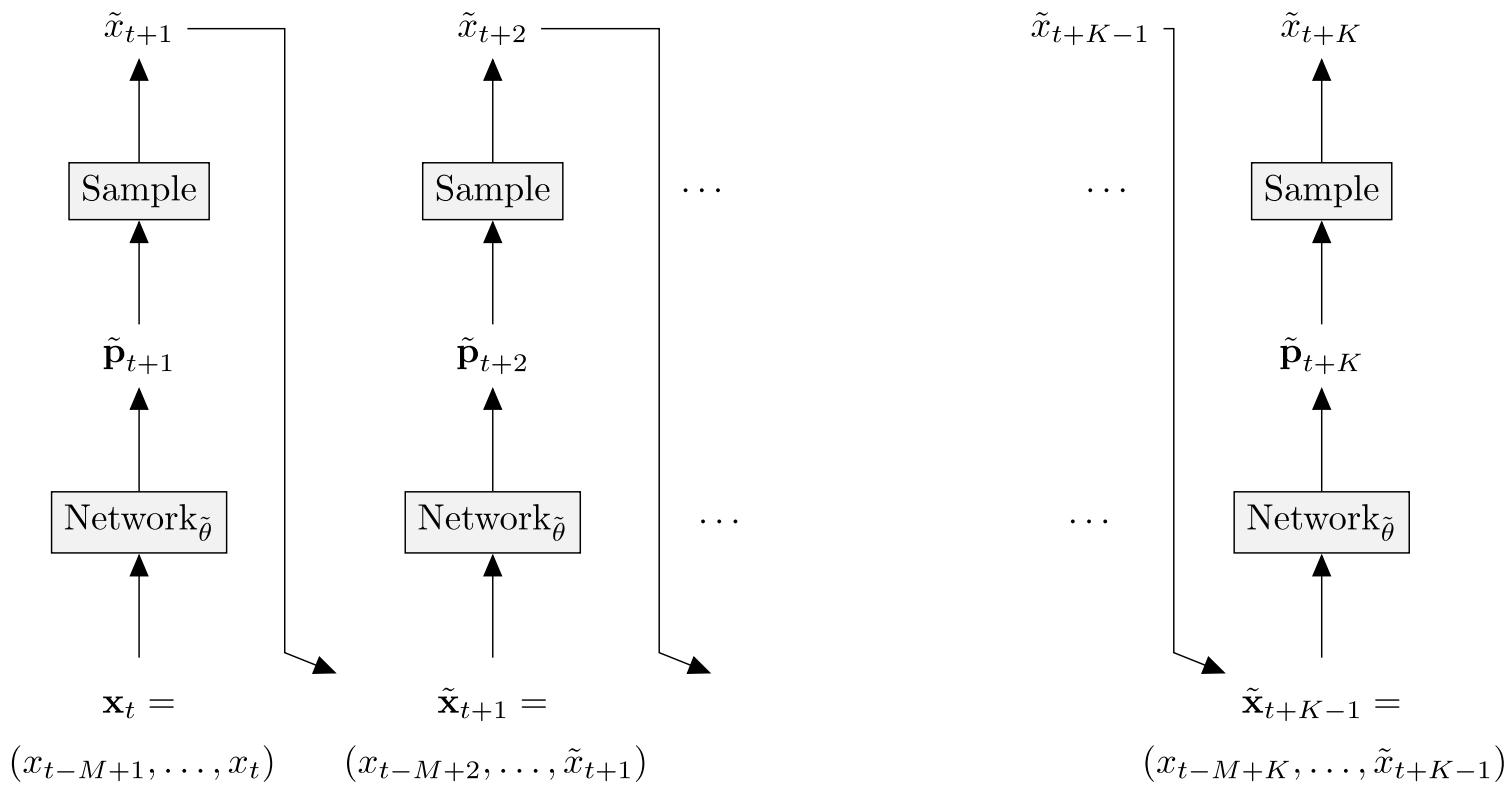


Figure 3.4: To build a trajectory of K steps, a sample $\tilde{\theta}$ of θ is obtained using dropout, defining the weights of the network. Then, the network is evaluated on the input $\mathbf{x}_t$ using the sampled weights $\tilde{\theta}$, obtaining a vector of parameters $\tilde{\mathbf{p}}_{t+1}$ of the target distribution X_{t+1}, which is sampled to get an output sample $\tilde{x}_{t+1}$. Then, the output is re-injected in a new evaluation of the network, producing an output for the next time step, and so on, completing the output of a trajectory of K values $(\tilde{x}_{t+1}, \ldots, \tilde{x}_{t+K})$.

Algorithm 2 Multistep probabilistic forecasting

Require: Neural network with parameters θ, output distribution family $\mathcal{F}$, input values $x_{t-M+1}, \ldots, x_t$, number of time steps K, number of trajectories I.

1: **for** $i = 1$ **to** I **do**
2: $\quad\tilde{\theta} \leftarrow$ Sample weights applying dropout to θ.
3: $\quad$**for** $k = 1$ **to** K **do**
4: $\quad\quad\mathbf{x} \leftarrow \left(\tilde{x}^i_{t-M+k}, \ldots, \tilde{x}^i_{t-1+k}\right)$, with $\tilde{x}^i_j = x_j$ if $j \leq t$.
5: $\quad\quad\tilde{\mathbf{p}} \leftarrow$ Evaluate network with input $\mathbf{x}$ and weights $\tilde{\theta}$ to get distribution parameters.
6: $\quad\quad\tilde{x}^i_{t+k} \leftarrow$ Sample from distribution $\mathcal{F}(\tilde{\mathbf{p}})$.
7: $\quad$**end for**
8: **end for**
9: **return** $\left[\left(\tilde{x}^1_{t+1}, \ldots, \tilde{x}^1_{t+K}\right), \ldots, \left(\tilde{x}^I_{t+1}, \ldots, \tilde{x}^I_{t+K}\right)\right]$.

where c is the desired confidence value (for example 80%, 90% or 95%), thus a centered PI that leaves the same proportion of probability below the lower and above the upper bound is returned. The proportions may change in practice according to application requirements, for example, in systems that have a fixed minimum value, and require exploring only an upper bound, the centering may be changed. For QR, the samples are sorted as well, and the ones corresponding to the desired quantile cuts are returned. For instance, if 100 samples are obtained, and quantiles for 25%, 50% and 75% are required, the samples in position 25, 50 and 75 are returned. It is important to note that for traditional point forecasting, the samples can also be used, either returning the median (quantile 50%), or their mean.

So far, the method has been presented for any given neural network. To use a neural network specially devised for sequential data, a stacked LSTM architecture was adapted, because it is known for its ability to handle sequential data [Gra13], and there are many implementations available. In this architecture, the output of each LSTM block layer is connected as input to a higher level LSTM, and finally, to a dense layer which computes the parameters $\tilde{\mathbf{p}}_{t+1}$ of the distribution of X_{t+1}, as detailed in Figure 3.5. When using an RNN such as this, the network can be evaluated one-step at a time, storing its internal recurrent state, instead of evaluating the whole sequence of previous values for each output time step. Additionally, using this architecture allows the training procedure to consider more than just one time step for the training loss, as depicted in Figure 3.5 as well. This modification got improved results, that will be explored in Chapter 4. This complete proposed model, then, is available to perform general probabilistic forecasting tasks, considering the epistemic uncertainty by using MCD, with the flexibility to handle different output distributions for the aleatoric uncertainty, and built using a deep learning architecture that is suitable for sequential time series data, and that has several available implementations available.

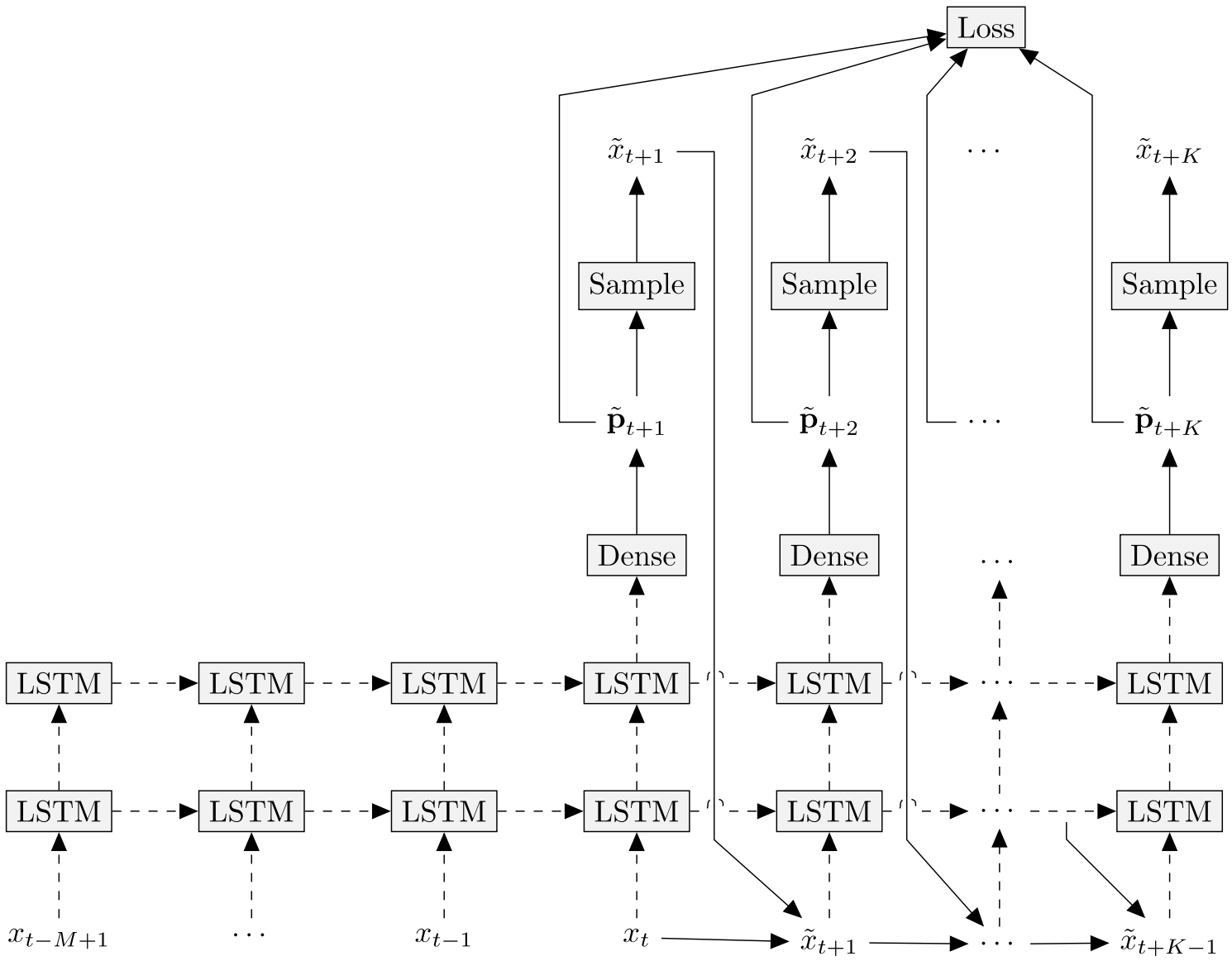


Figure 3.5: Unfolded version of a probabilistic forecasting stacked LSTM architecture. The first $M-1$ left steps of the recurrence are applied over the network input data, producing parameters $\tilde{\mathbf{p}}_{t+1}$ of the distribution of X_{t+1} with a dense layer. From these parameters, an output $\tilde{x}_{t+1}$ is sampled, which is re-injected for the next recurrence step, iterating until K steps have been obtained. In this figure, the training loss considers all K outputs, and dropout is applied over dashed connection in the figure, where the same dropout mask, thus the same network parameters, is applied on each step of the recurrence, making all layers displayed on a figure row to work with the same weights.

Chapter 4

Experiments

To assess the proposed model, several experiments were conducted. First, a group of synthetic datasets were created and evaluated to explore the behavior of the model in data that is under the experimenter control, and then a group of real datasets were considered.

4.1 Synthetic datasets

4.1.1 Description

Seven time series instances were obtained, using the following generation processes:

1. A sinusoidal signal sampled with a fixed frequency and a random noise with normal distribution with constant variance. The samples were generated using the function $S_t^1 = \sin(\omega t) + \eta$, where $\eta \sim \mathcal{N}\left(0, 0.3^2\right)$ and $\omega = \frac{2\pi}{15}$. Note that the noise in this case does not depend on the time variable t, thus it assumes homoscedasticity. The first 300 values of the realization used in this work are shown in Figure 4.1. The goal of this series is to check that the model can handle homoscedastic periodic signals that have no trend.

2. The same sinusoidal signal, but with a random noise that has a time dependent variance. The samples are generated using the same function $S_t^2 = \sin(\omega t) + \eta_t$, with $\omega = \frac{2\pi}{15}$, but the noise follows $\eta_t \sim \mathcal{N}\left(0, \left(0.22 + 0.2\sin\left(\frac{\omega t}{100}\right)\right)^2\right)$. In this case, the noise variance depends on t, thus having heteroscedasticity, having a frequency that is 100 times less than the signal frequency. The first 300 values of the realization used in this work are shown in Figure 4.1, where it can be noted that, in comparison with S^1, the noise size changes in time. The goal of this series is to check that the model can also handle heteroscedastic periodic signals that have no trend.

3. An autoregressive process (AR), with lag 2, of the form $S_t^3 = 0.75S_{t-1}^3 - 0.75S_{t-2}^3 + \epsilon_t$, where ϵ_t is a random noise with normal distribution $\mathcal{N}\left(0, 0.1^2\right)$. The first 300 values of the realization used in this work are shown in Figure 4.2. The goal of this series is to check that the model can handle signals that can be easily handled by traditional AR models.

4. An autoregressive moving average process (ARMA), with parameters $(1, 3)$, of the form $S_t^4 = -0.0001S_{t-1}^4 + 0.1\epsilon_{t-1} + 0.9\epsilon_{t-2} + 0.9\epsilon_{t-3} + \epsilon_t$, where ϵ_t is a random noise with normal distribution $\mathcal{N}\left(0, 0.1^2\right)$. The first 300 values of the realization used in this work are shown in Figure 4.2. The goal of this series is to check that the model can handle signals that can be easily handled by traditional ARMA models.

5. A mixture of the previous two processes, according to $S_t^5 = \pi_t^5 S_t^3 + \left(1 - \pi_t^5\right) S_t^4$, where π_t^5 is a mixture weight, defined as $\pi_t^5 = \frac{1}{2}\left(1 + \sin\left(\omega t\right)\right)$, with $\omega = \frac{2\pi}{150}$. This process changes between the third and fourth processes, with a constant frequency. The first 300 values of the realization used in this work are shown in Figure 4.3, where it is clear that it fluctuates between S^3 and S^4. The goal of this series is to check that the model can handle signals that are more complex, as it is a time changing mixture of an AR and an ARMA model.

6. The fourth process with an additional random measurement noise that is similar to the used by the second process, although with a reduced variance. In summary, it is $S_t^6 = S_t^4 + \eta_t$, with $\eta_t \sim \mathcal{N}\left(0,\ \left(0.0154 + 0.014\sin\left(\frac{\omega t}{100}\right)\right)^2\right)$. The first 300 values of the realization used in this work is shown in Figure 4.3. The goal of this series is to check that the model can handle signals from an ARMA model, but having an heteroscedastic noise signal.

7. A mixture of the third and the fifth processes, according to $S_t^7 = \pi_t^7 S_t^3 + \left(1 - \pi_t^7\right) S_t^5$, where π_t^7 is a mixture weight, defined as $\pi_t^7 = \frac{1}{2}\left(1 + \sin\left(\omega t\right)\right)$, with $\omega = \frac{3\pi}{250}$. This process changes between the third and fifth processes, with a constant frequency. As the fifth process is already a mixture, this is a more complex mixture of the third and the fourth processes. Indeed, it can be shown that it is equivalent to $S_t^7 = \pi'_t S_t^3 + (1 - \pi'_t) S_t^4$, with $\pi'_t = \pi_t^5 + \pi_t^7 - \pi_t^5 \pi_t^7 = \frac{1}{4}\left(3 + \sin\left(\frac{2\pi t}{150}\right) + \sin\left(\frac{3\pi t}{250}\right) - \sin\left(\frac{2\pi t}{150}\right)\sin\left(\frac{3\pi t}{250}\right)\right)$. The first 300 values of the realization used in this work are shown in Figure 4.3, where the changes between S^3 and S^4 are less clearly seen by the analyst. The goal of this series is to check that the model can handle signals that are more complex, as it is a complex time changing mixture of an AR and an ARMA model.

4.1.2 Results

A comparison was made between using MCD and not using it, for all 7 synthetic datasets, for the task of one time step prediction, training a similar FNN network in both cases. It is important to note

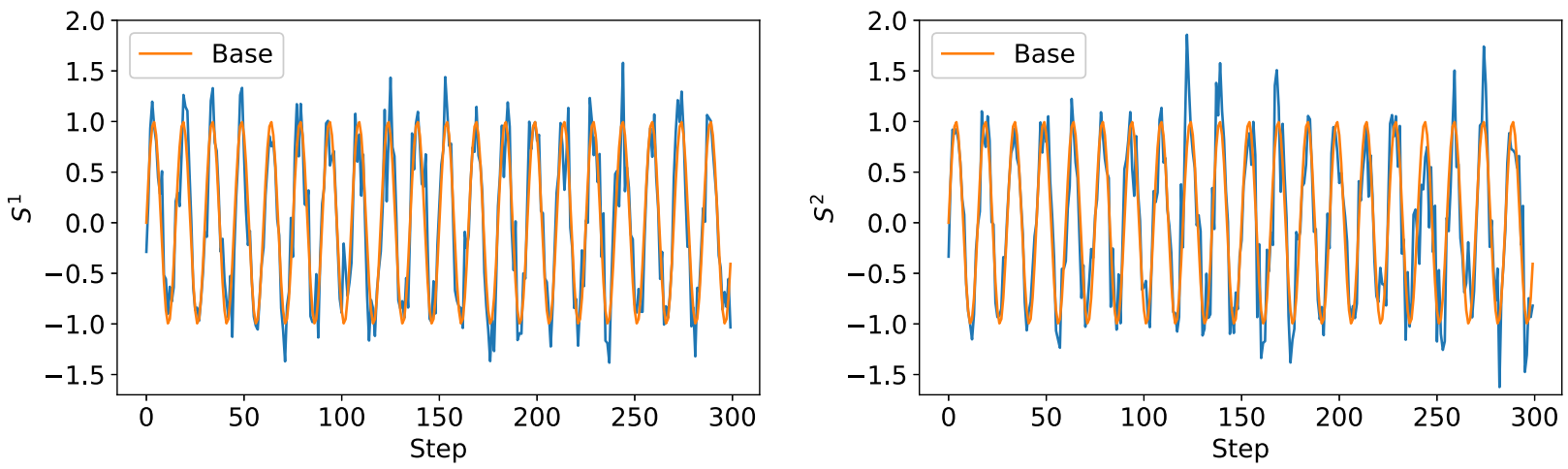


Figure 4.1: First 300 values of S^1 and S^2, and the base sinusoidal signal that was used to generate them.

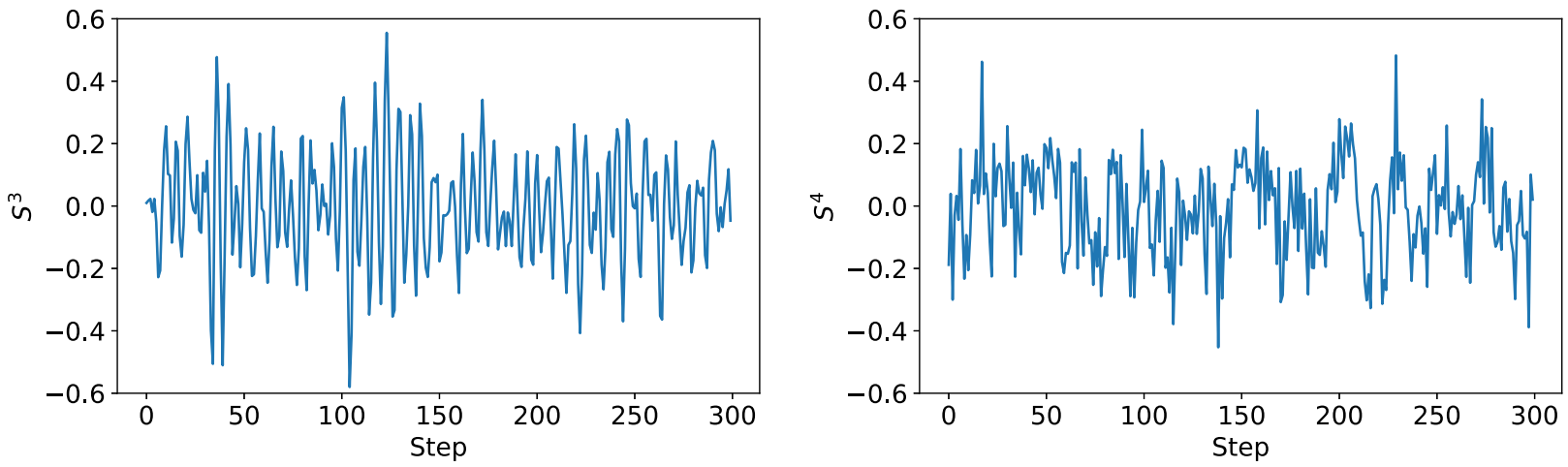


Figure 4.2: First 300 values of S^3 and S^4.

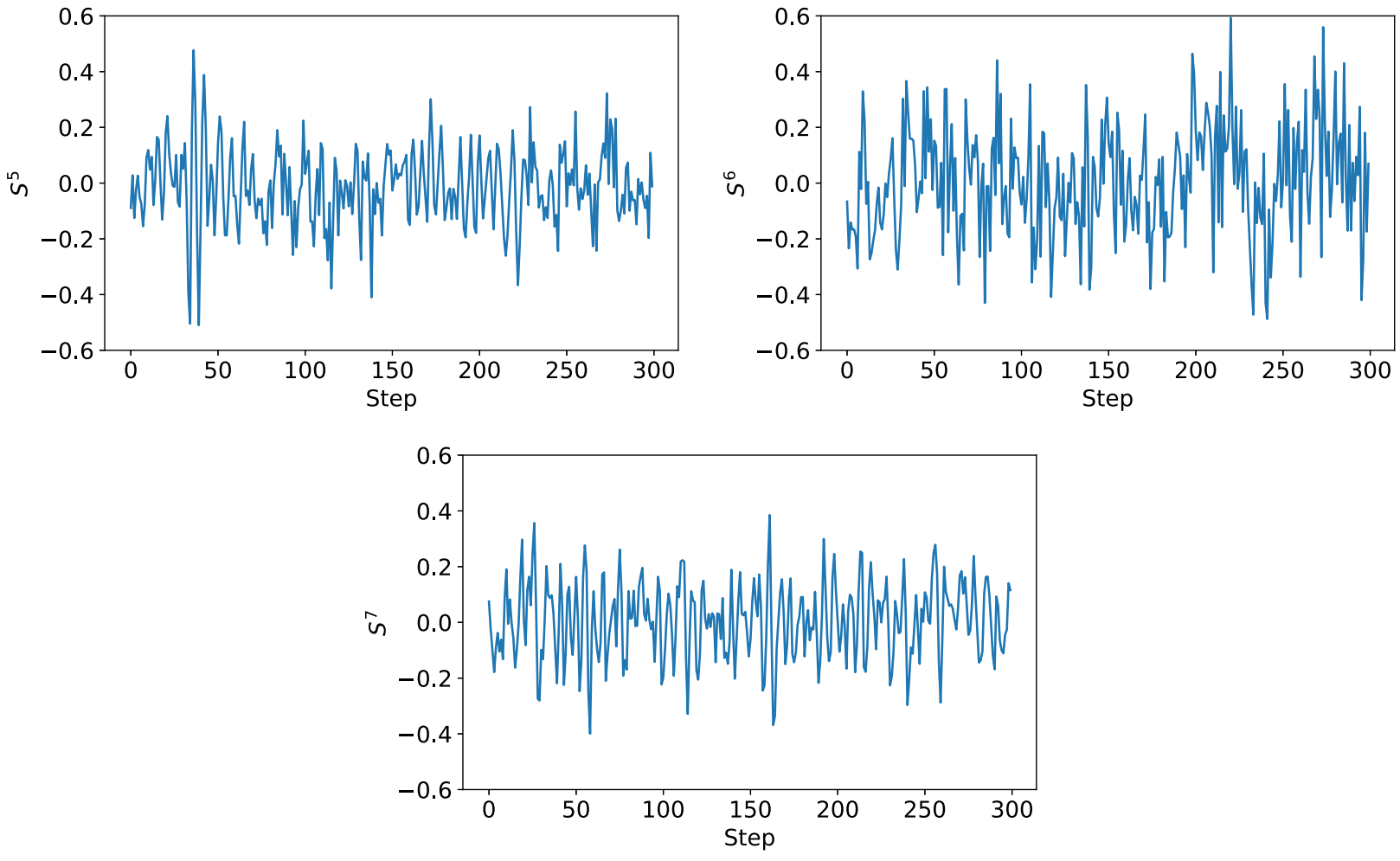


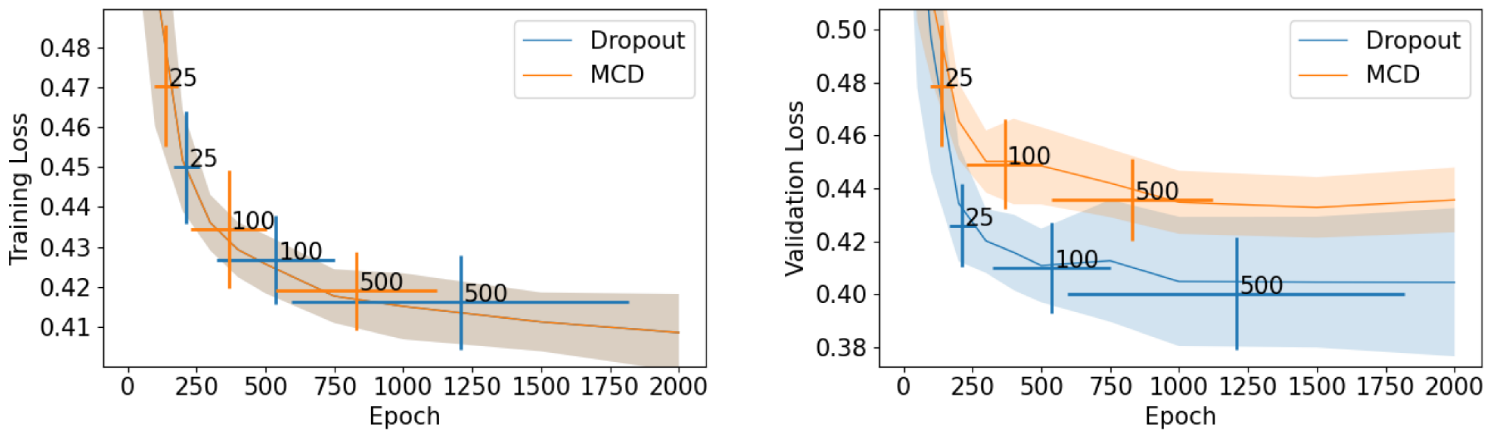
Figure 4.3: First 300 values of S^5, S^6 and S^7.

that the training procedure is similar both using dropout only during training, and when using MCD. The validation and test procedure changes according to the use of MCD. The network had an input of 28 units corresponding to the previous 28 time steps, 3 hidden layers of 10 hidden units each, with ReLU activation, and an output layer consisting of 2 units, interpreted as the parameters of a normal distribution. Each experiment was performed 10 times with different random seeds each, thus having 10 repetitions of each experiment. For the sake of simplicity, most hyperparameters were left fixed, except the number of epochs. For this hyperparameter, two ways of setting it were explored: first, using a fixed value that has to be set at the start of the training procedure, and, otherwise, having a patience parameter that ends the training when the validation loss does not improve after the given number of patience epochs.

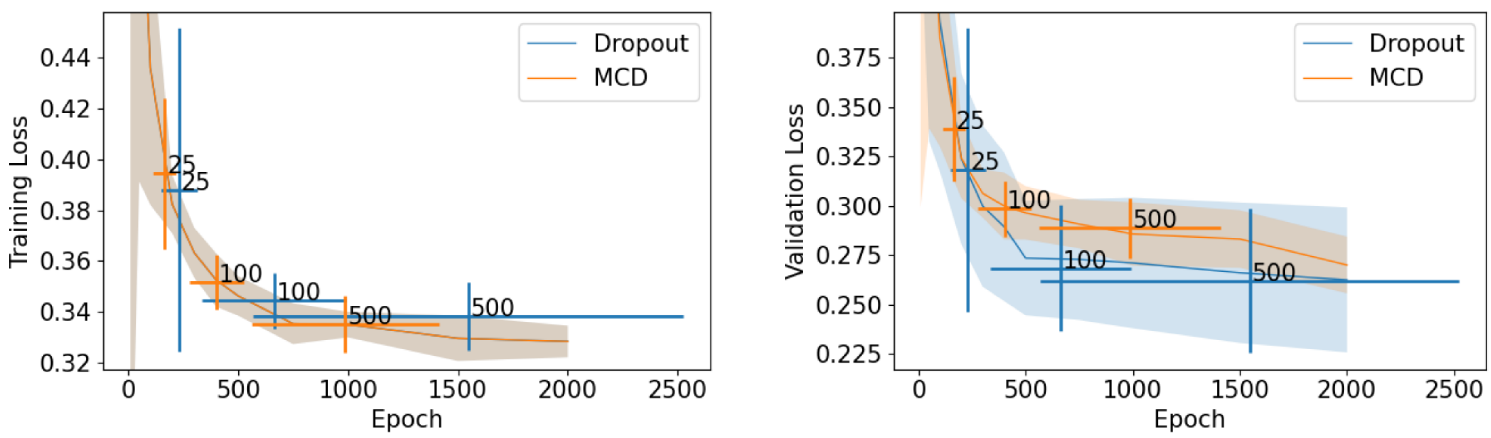
The first thing to check, is that the training procedure works, as expected, similarly in both cases, but the validation loss is computed differently, as shown in Figures 4.4 and 4.5. In it, the losses after different number of training epochs are displayed, together with a shaded area showing the standard deviation of them, when considering the 10 repetitions of each experiment. Also, crosses are displayed, showing the value when the training algorithm stops, when using different values of patience. It can be seen that, for all datasets, the training loss has the same value in both cases, and with the same standard deviation. Nevertheless, the stop epoch selected when using patience differs, because, as it can be seen on the right side for each set, the validation loss is different when considering MCD. The value of the validation loss is higher when using MCD.

The first probabilistic forecasting task reviewed with these datasets is the PI generation, with metrics shown in Figures 4.6 and 4.7. Here, the behavior of the model for each dataset differs. In the case of S^1, a sinusoidal signal with constant variance noise, the IS and CWC show that the MCD model performs better, having lower values. Most importantly, the CWC metric shows an important feature of the proposed model: it handles better the uncertainty given by the random seed in the network initialization values, because when using traditional dropout, it can be seen that the standard deviation of the metric is big. This is due to the exponential penalty of this metric when the coverage is not met. Nevertheless, when looking at PICP and PINAW at Figures 4.8 and 4.9, it is clear that this was done at the expense of having wider PIs, and not risking having a coverage (PICP) less than the required value, in this case 90%. In this case, then, while the traditional dropout model was able to produce PIs within the expected coverage in some cases, it failed in some repetitions, and this translated into having a substantial average CWC when training the model for more epochs, and displaying a bigger standard deviation. For S^2, the improvement of the MCD model was mostly regarding the variance of the metrics. In the case of IS, while being only slightly better in average, its variance is lower, meaning that all repetitions performed more similarly. For CWC, the same is observed, even having a bigger value in average, it is clear that the outcome of the model when using MCD is more similar for different repetitions, having much smoother

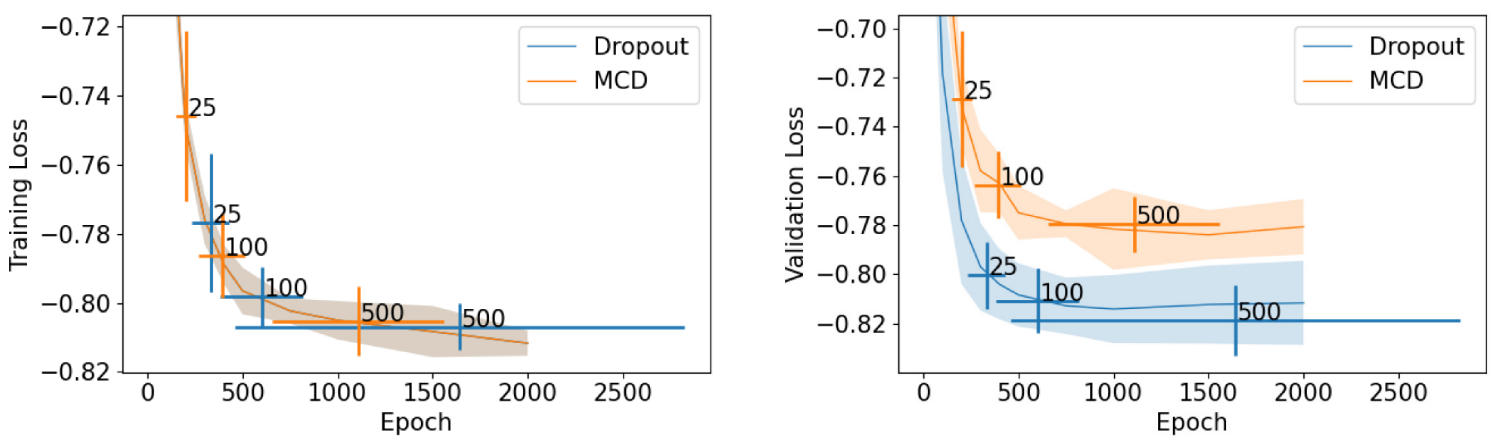
S^1



S^2



S^3



S^4

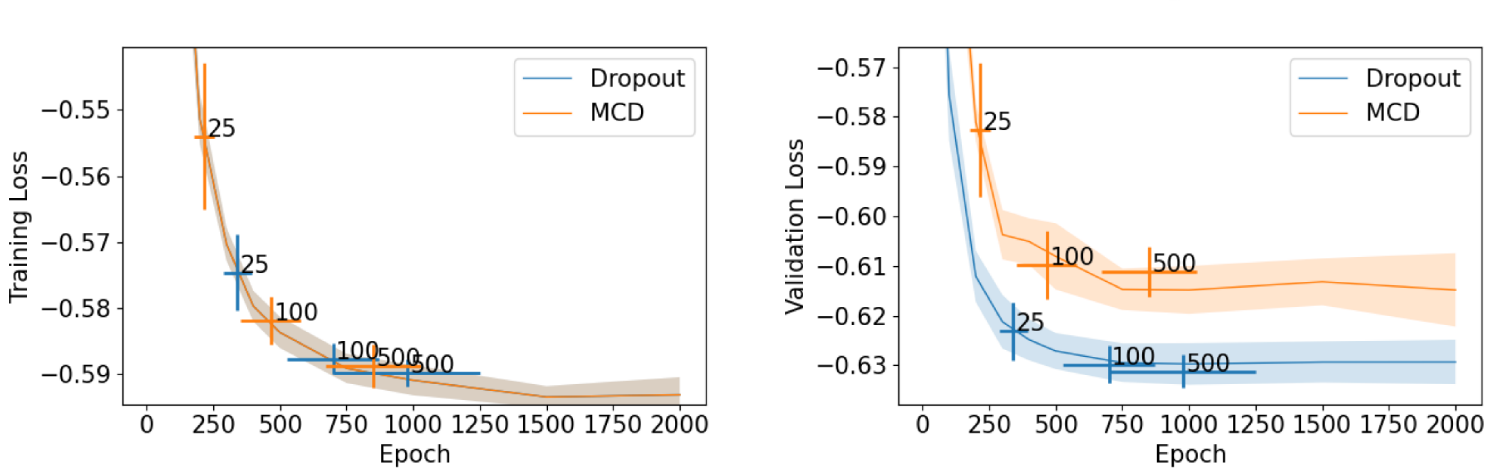


Figure 4.4: Training and validation losses of synthetic datasets, part 1. Shaded areas show standard deviation of values. Crosses show standard deviations when using given early stopping patience steps.

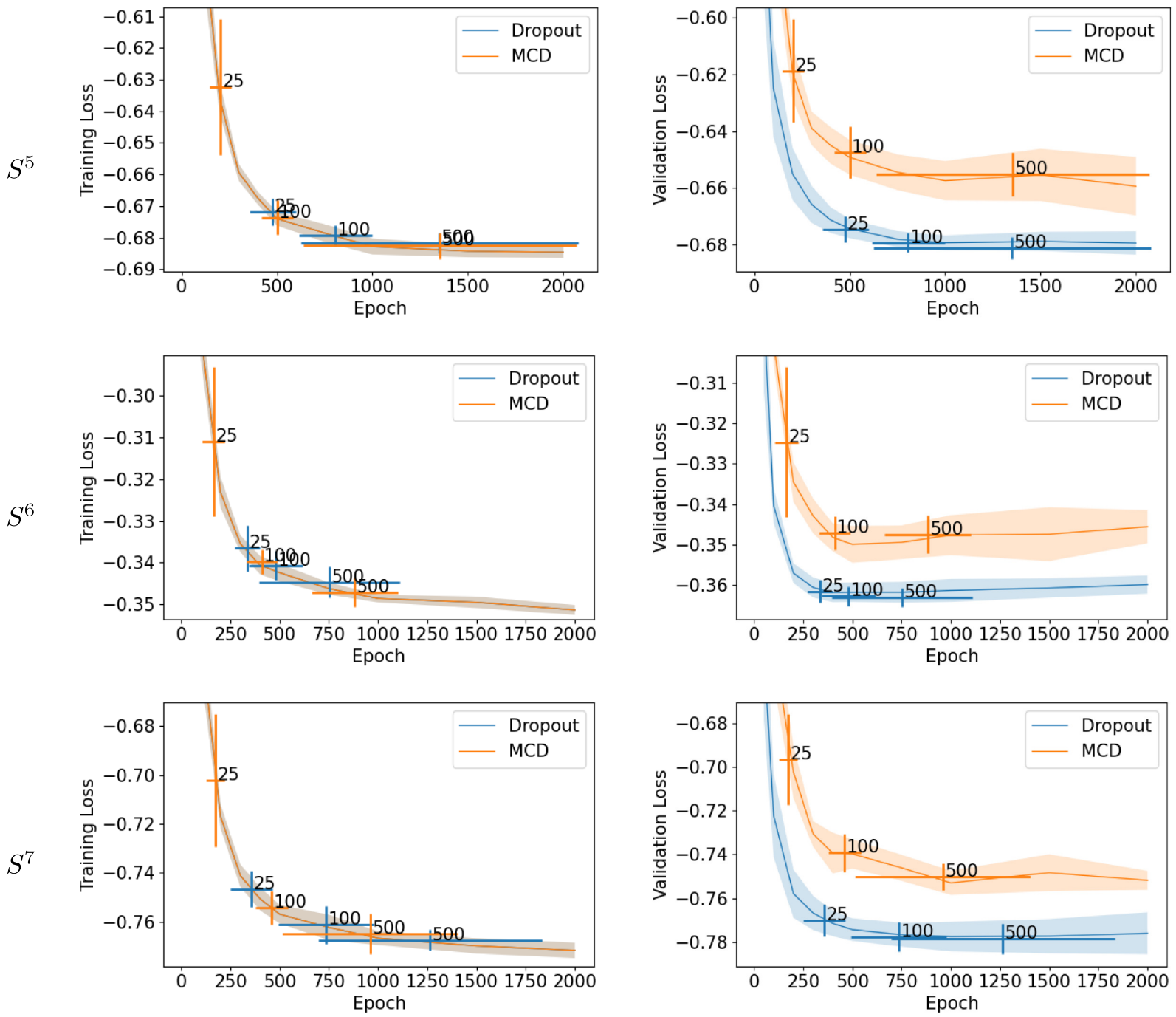


Figure 4.5: Training and validation losses of synthetic datasets, part 2. Shaded areas show standard deviation of values. Crosses show standard deviations when using given early stopping patience steps.

changes in the metric when the number of training epochs change. This can be related to the bias-variance trade-off: the MCD model is reducing the variance of the results, and this may be translated into having more bias. In the case of S^3, an AR process, similar conclusions may be obtained from the metrics, specially regarding the coverage, as it can be seen from CWC that MCD successfully reduced the failing cases that made this metric high when not using it for some repetitions. For S^4, an ARMA process, similar conclusions regarding CWC may be implied, although it is also observed an improvement on the quality of the PIs by the IS metric. This can be observed in the PICP and PINAW metrics, that show that the coverage is met widely by the MCD model, making the PIs wider, but the lower IS metric means that the linear penalty for observations laying out of the PI was bigger than the penalization due to the width of the PI. In S^5 dataset, a mixture of models, it can be seen that the coverage was hardly met by the original model, as shown by PICP, and translating into having big CWC values. This fact may be due to the mixture nature of this time series, where the adjusted model has to produce a PI which is good for the corresponding process of the current time step. In the case of the MCD model, it is able to better handle the model uncertainty, thus producing a wider PI, and reducing the CWC metric. This improvement is not clear for the IS metric, meaning that the gaining in coverage was at the expense of a wide PI, although having less variance in general, as expected. For S^6, an ARMA process with changing variance noise added, none of the two models is able to cover the expected 90% of the observations, while the MCD model is able to improve the quality of the PIs in terms of the IS metric and CWC metric. In this case, it is observed that for a few training epochs, the models produce PIs that meet the coverage, having small CWC, but the associated IS metric shows that then the PIs were too wide, having too big IS. Also, this fact is observed over the test set, which is unknown during training. Finally, S^7 dataset, a more complex mixture, shows that MCD produces PIs that have a lower IS and CWC metrics, in comparison with using traditional dropout, thus being able to better capture the complex underlying mixture, due to its ability to handle the model uncertainty.

In addition to PIs, other probabilistic forecasting tasks such as QR and empirical distribution exploration can be performed. For these tasks, the average PL and CRPS are shown in Figures 4.10 and 4.11. It can be seen that using MCD improved the test metrics for all datasets, while also reducing the standard deviation of the metrics, except for S^5 and S^6 datasets. In these cases, the metrics are reduced, but for a value that is not significant. This means that MCD is able to improve the model identification in most cases, but for the case of a simple mixture of models such as S^5 and for the case of adding a changing variance noise to an ARMA process, such as S^6, the improvement is not significant. Interestingly, for S^7, which is a more complex mixture, MCD worked better. This fact, added to the previous observation, may simply mean that, while MCD in general improves the behavior of the model, it may not be relevant in all cases, because sometimes the real model is extracted sufficiently well by the original model.

Additionally, point forecasting of the expected mean or median value can be done, for which the

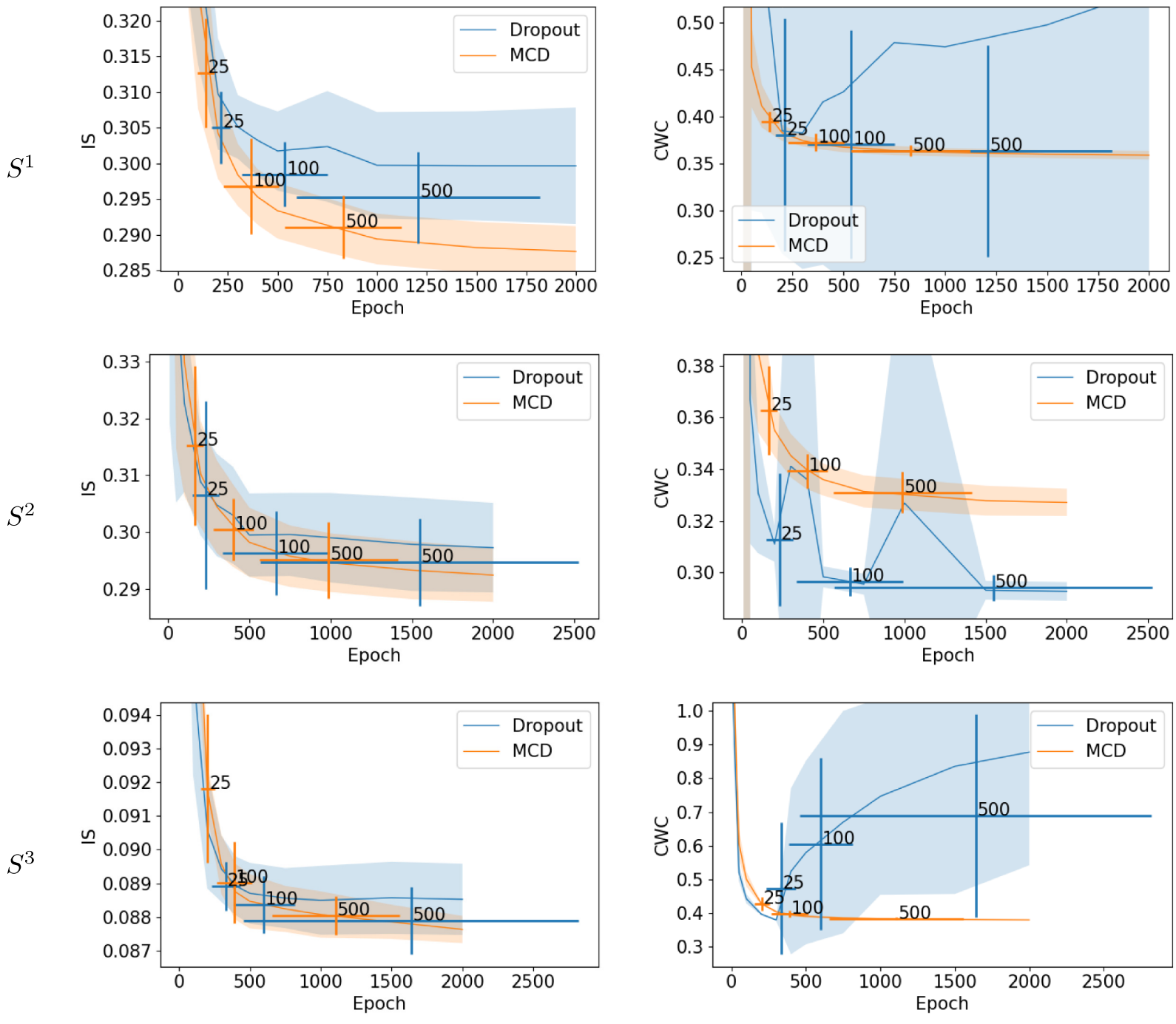


Figure 4.6: PIs metrics on synthetic datasets, part 1. Shaded areas show standard deviation of values. Crosses show standard deviations when using given early stopping patience steps.

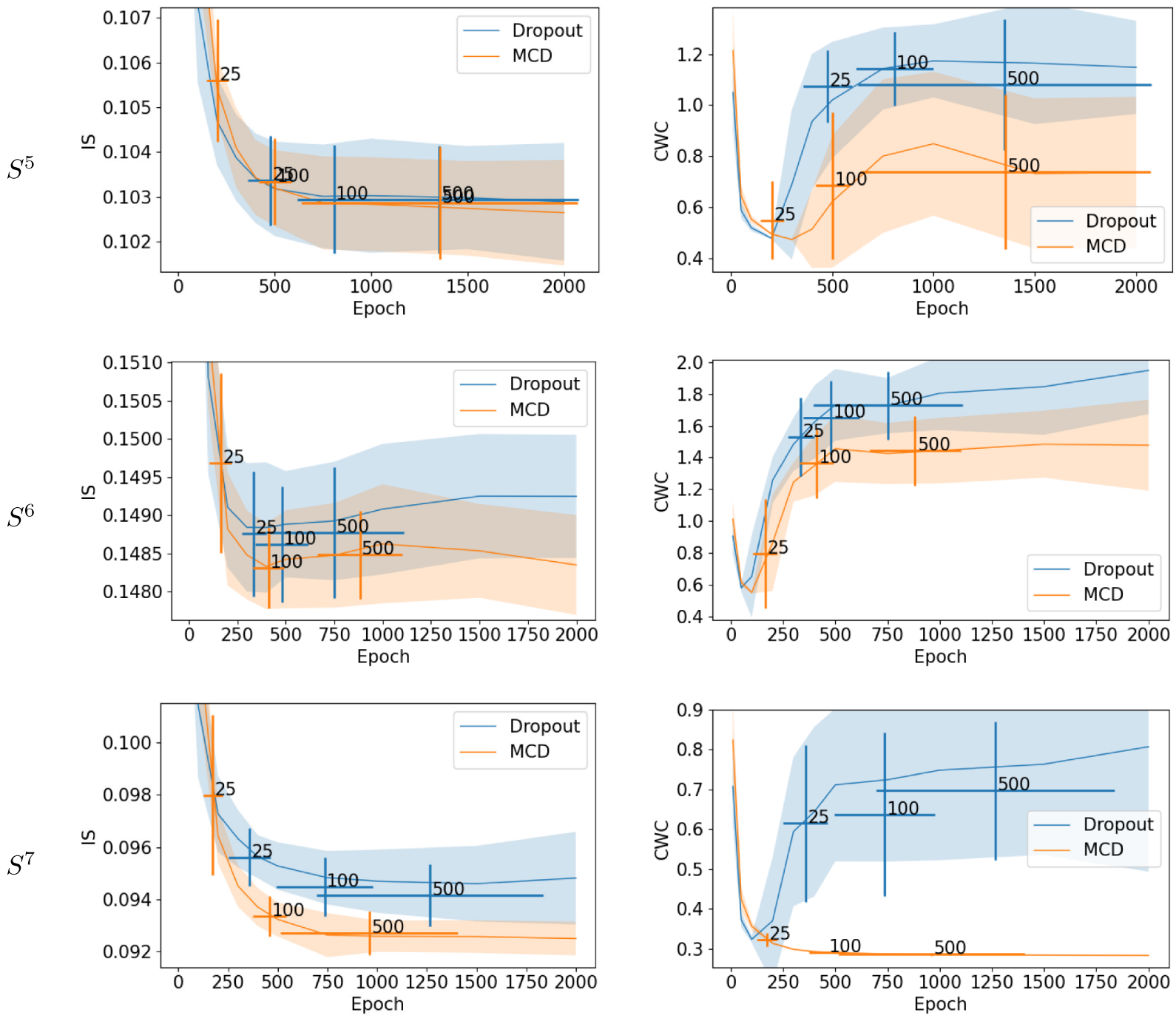


Figure 4.7: PIs metrics on synthetic datasets, part 2. Shaded areas show standard deviation of values. Crosses show standard deviations when using given early stopping patience steps.

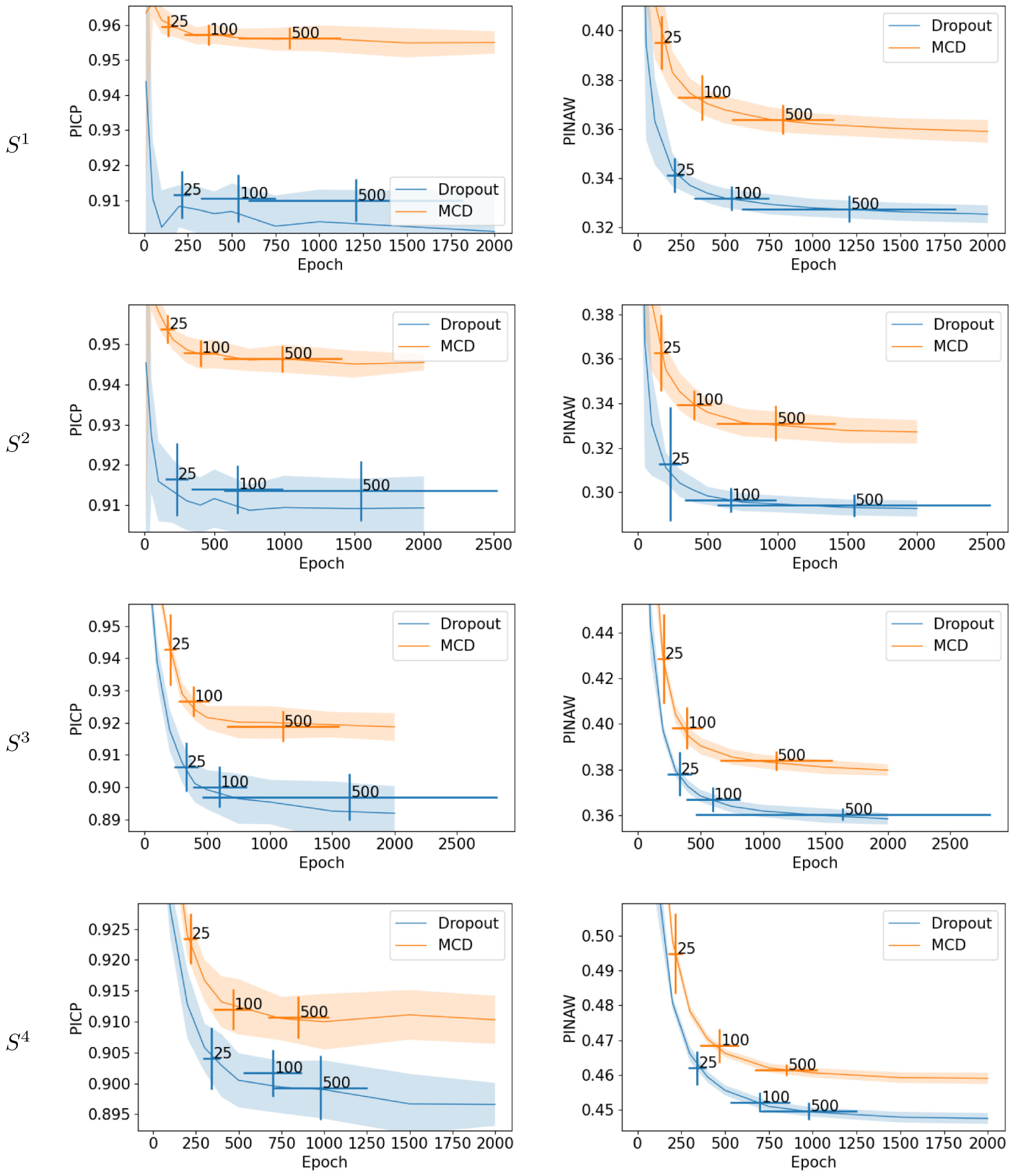


Figure 4.8: PIs coverage and width on synthetic datasets, part 1. Shaded areas show standard deviation of values. Crosses show standard deviations when using given early stopping patience steps.

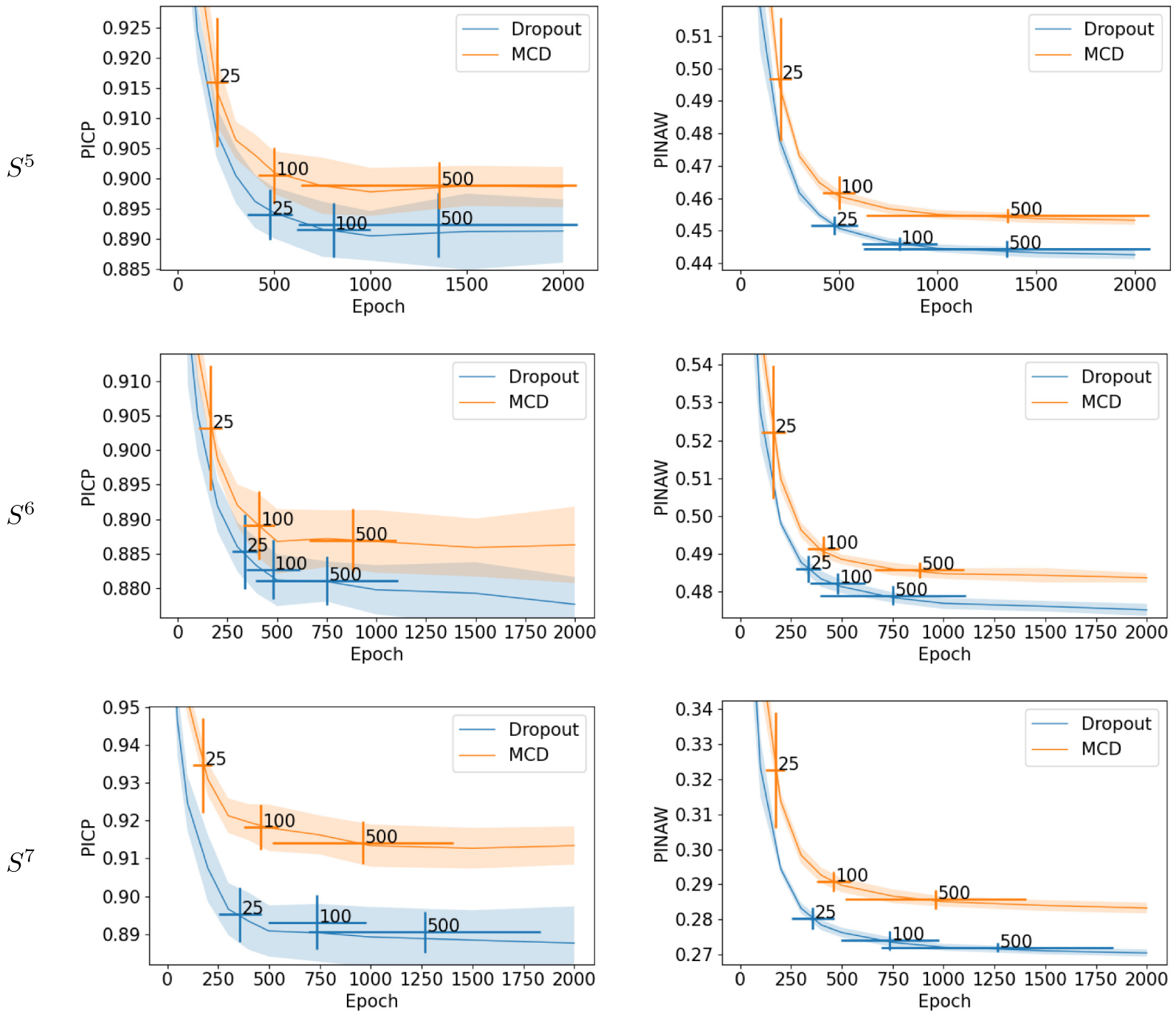


Figure 4.9: PIs coverage and width on synthetic datasets, part 2. Shaded areas show standard deviation of values. Crosses show standard deviations when using given early stopping patience steps.

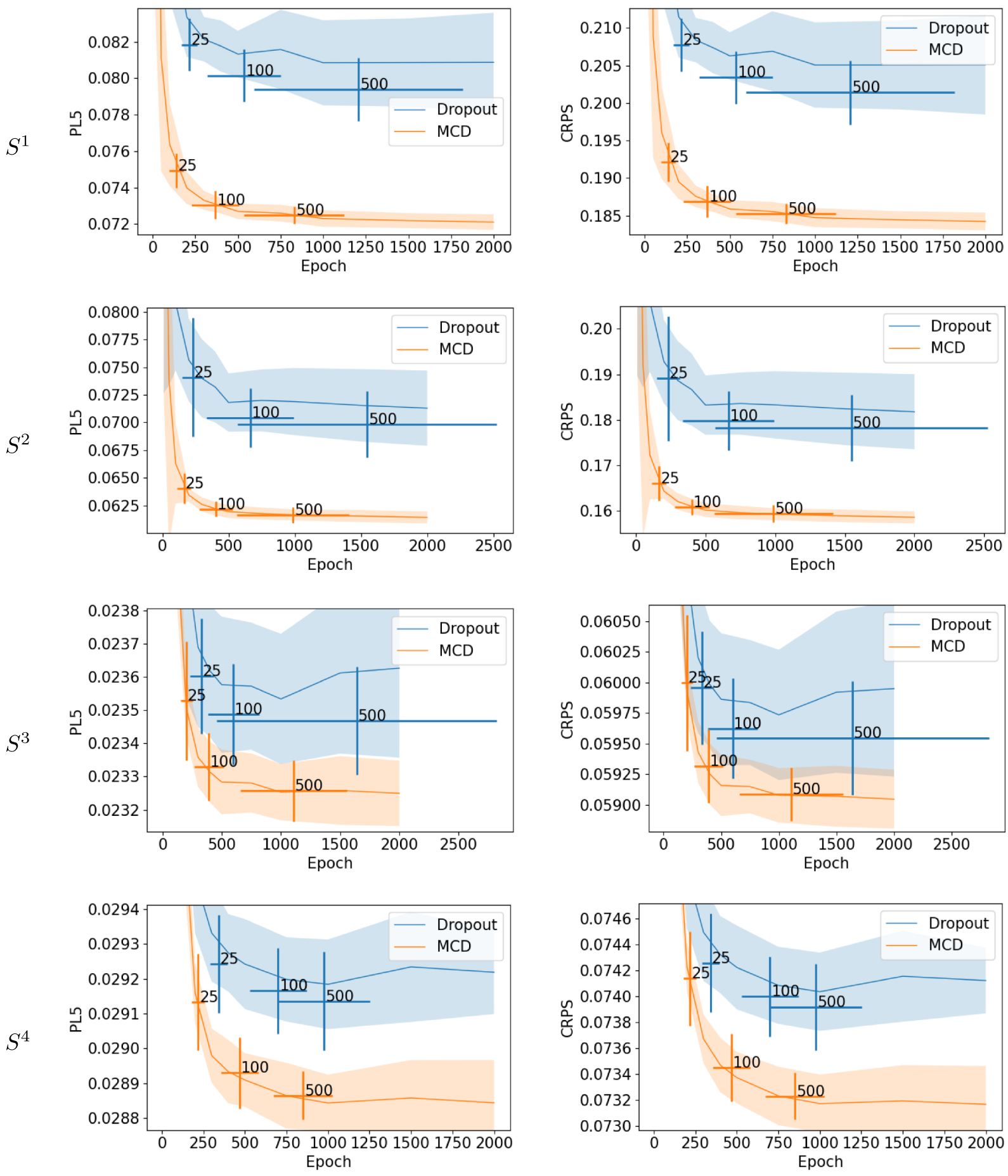


Figure 4.10: PL applied to quantiles 20%, 40%, 60% and 80%, and CRPS, part 1. Shaded areas and cross width and height show standard deviation of values.

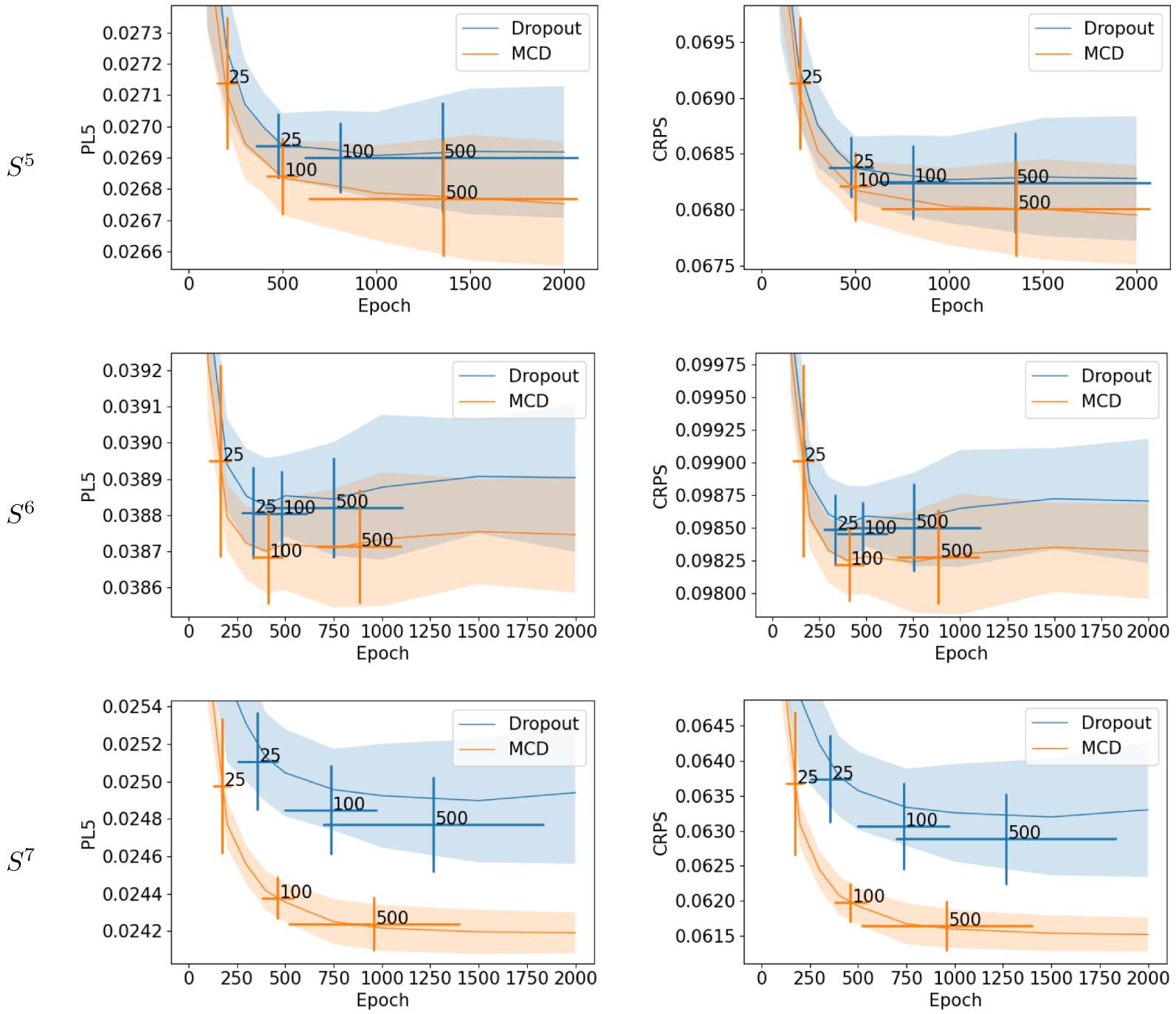


Figure 4.11: PL applied to quantiles 20%, 40%, 60% and 80%, and CRPS, part 2. Shaded areas and cross width and height show standard deviation of values.

average RMSE and MAE are shown in Figures 4.12 and 4.13. While the objective of this work was not towards generating a point forecasting model, it can be seen that the addition of MCD improved these point forecasting metrics as well, with similar results to the previously shown, where S^5 and S^6 did not show clear improvements. This is an important result, because it means that the robustness of the MCD validation procedure also helps to find better models in general.

As a conclusion for the synthetic datasets experiments, the addition of MCD makes the model to work better regarding the test metrics for one-step regression tasks, including probabilistic forecasting tasks such as PI estimation and QR, including series that present homoscedasticity or heteroscedasticity. The improvement seems to be consistent for different metrics and datasets, and, while the average difference may be small in some cases, the variance of the presented metrics is significantly lower when using MCD, which shows that it handles the model uncertainty, reducing the variance, and making the training procedure more robust. This reduced variance may be associated, in some cases, to bias, recalling the bias-variance trade-off.

4.2 Real datasets

4.2.1 Description

To assess the proposal, seven datasets of four representative probabilistic forecasting tasks were considered, summarized in Table 4.1, showing different global distribution of values, as shown in Figures 4.14 and 4.15. A brief description of each of them is given below:

- *B08*, *D05a*, *D08* and *E01*: These are four wind speed measurements datasets taken by prospecting towers located in different places of the Atacama Desert, in Northern Chile. Data can be downloaded from `http://walker.dgf.uchile.cl/Mediciones/`. Although the data collection is every 10 minutes, the datasets are converted to hourly averages. The distribution of values from these series varies, as it can be seen in Figure 4.14.

- *UCI individual household electric power consumption*: This dataset is composed of electrical load consumption measured every minute in a home in France for almost 4 years [DG17]. It presents many missing values, which were filled by copying the value at the same hour and minute from the previous day. For example, if the 10 am measurement of a day is missing, the value at 10 am of the previous day is used. In this work, the resolution was reduced to hourly values to be able to build models using the same architecture as the ones applied to the other datasets, taking the time series comprised by the hourly sum of each minute value as the input for the model. The distribution of values is highly two modal, as it can be seen in Figure 4.15.

- *GEFCom2014-L*: It corresponds to the power load of an electric utility in the US measured every

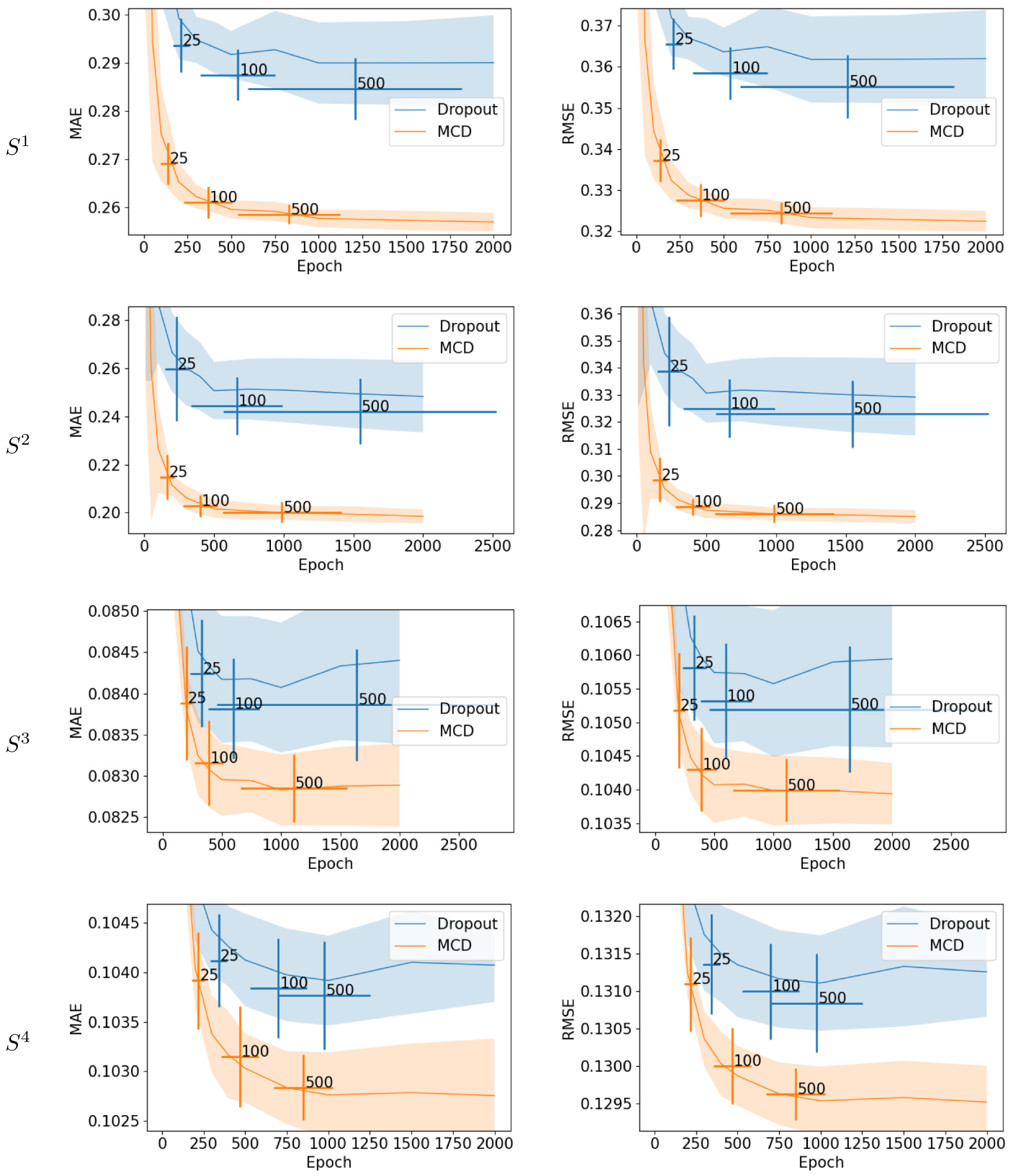


Figure 4.12: Error of the median prediction (MAE) and the mean prediction (RMSE), part 1. Shaded areas and cross width and height show standard deviation of values.

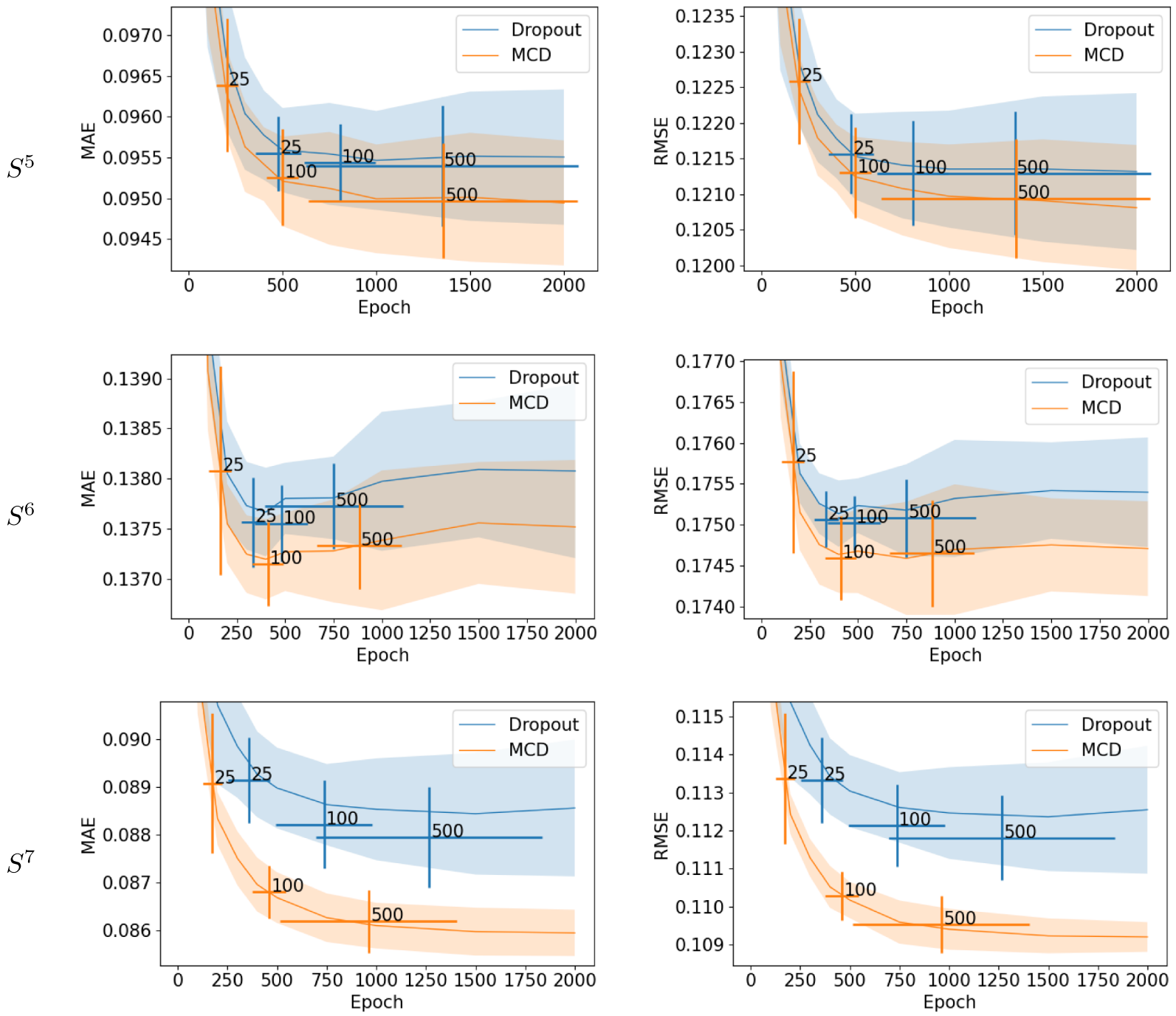


Figure 4.13: Error of the median prediction (MAE) and the mean prediction (RMSE), part 2. Shaded areas and cross width and height show standard deviation of values.

Table 4.1: Summary of seven datasets used in this research to validate the proposal.

Name	Domain	Resolution	Total	Mean	Min.	Max.	Missing (%)
B08	wind speed	1 hour	117215	6.72	0.00	17.0	0 (0%)
D05a	wind speed	1 hour	457155	7.20	0.00	26.1	0 (0%)
D08	wind speed	1 hour	320595	6.89	0.00	26.7	0 (0%)
E01	wind speed	1 hour	154266	7.78	0.00	25.5	0 (0%)
UCI ind. household	power load	1 min.	2075259	1.09	0.08	11.1	25979 (1.25%)
GEFCom2014-L	power load	1 hour	61344	146.18	16.10	317.5	0 (0%)
Canela 1	wind power	1 hour	39889	3.13	0.00	19.0	1 (0.000025%)

hour for 7 years, and it was made available for a Global Energy Forecasting Competition [HPF$^+$16]. Its distribution is smoother than the other series, with one dominant mode.

- *Canela 1*: This dataset consists of hourly values of power injected to the electrical grid by a wind power plant located in Chile. Data can be downloaded from `https://www.coordinador.cl/operacion/graficos/operacion-real/generacion-real/`. The distribution of this series is highly skewed, having many small values, and a more exponential distribution.

Before model evaluation, all datasets were divided into 5 blocks, and then each block sub-sequentially divided in training, validation and test set. This separation allows considering the behavior of the model on different regions of the data, instead of using only a fixed final test set. A small overlap between blocks was considered between them, to have more training data for each block, as shown in Figure 4.16. The test set of each block was left with 1000 hours, and 10% of the remaining block data as validation set. The data was normalized for each block, considering only the training set to fix the minimum and maximum normalization parameters.

A note regarding time series cross validation

Usually in supervised machine learning, it is assumed that samples are obtained independently and from an identical underlying distribution. When working with time series, these assumptions are broken. First, the distribution of data may be changing with time, as it has been explained during this text, and this work proposes a model that may use a different distribution from the same family for each time step. Second, and most importantly, the independence assumption is broken because the time series observed values may depend on previous values. This is handled in this work assuming stationarity and applying the Markov assumption, thus considering each sample as a different independent sample from the underlying distribution that is assumed can be learned from the given number of input time steps.

Some works have discussed this problem and the best way to assess models when these assumptions

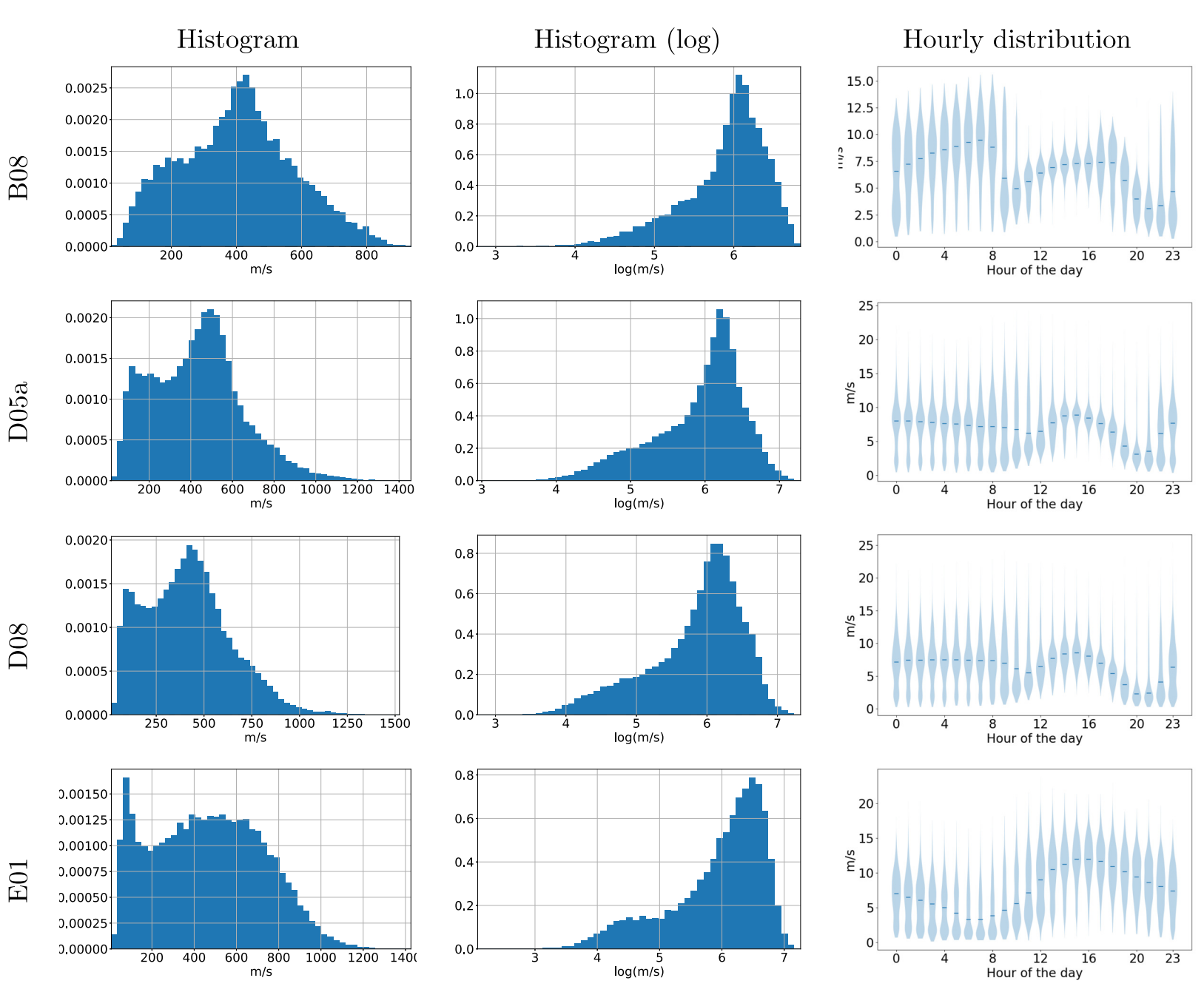


Figure 4.14: Histogram of wind speed values, histogram when adding 1 and applying logarithm, and violin plots showing the hourly distribution of data throughout the day.

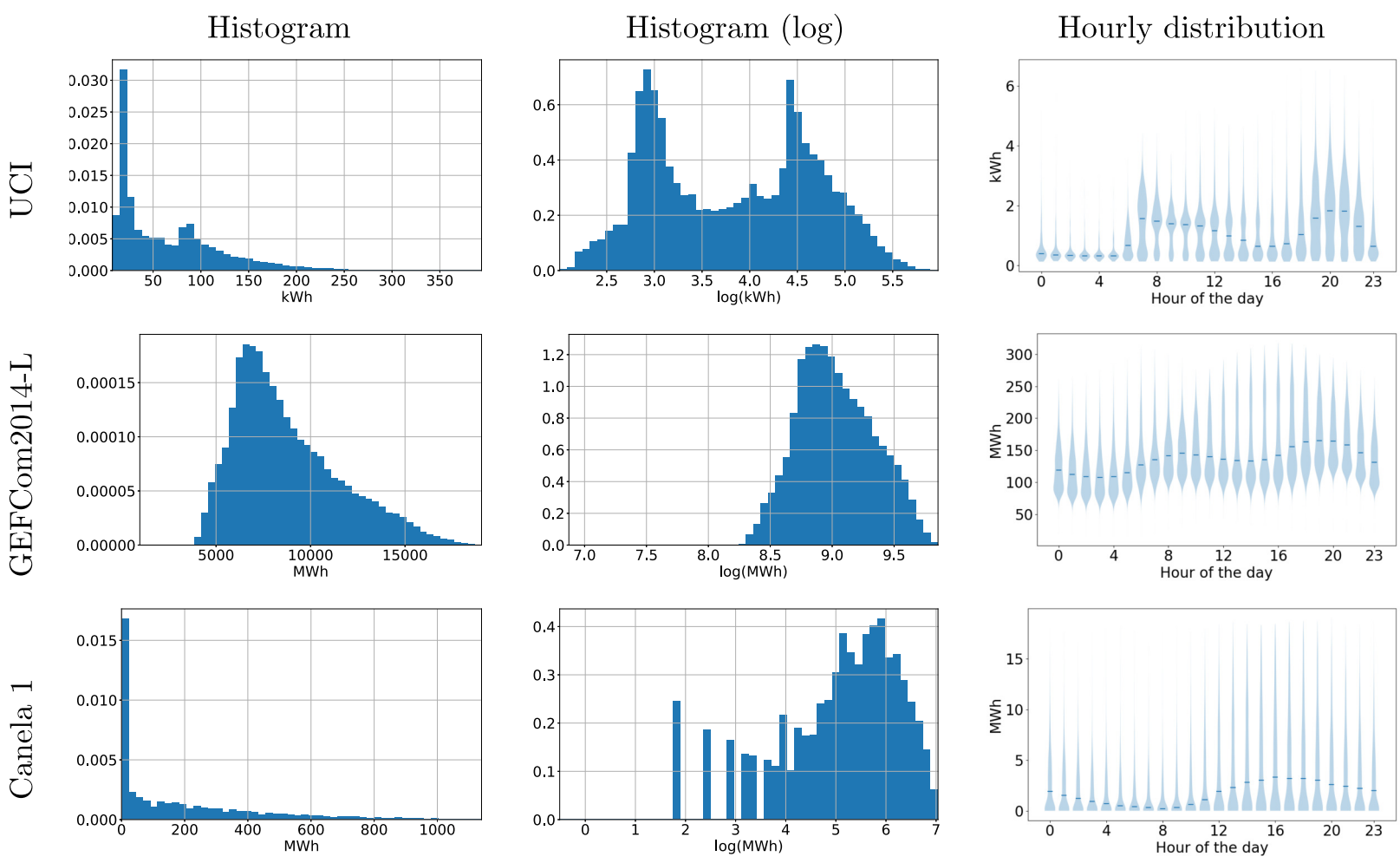


Figure 4.15: Histogram of power load values, histogram when adding 1 and applying logarithm, and violin plots showing the hourly distribution of data throughout the day.

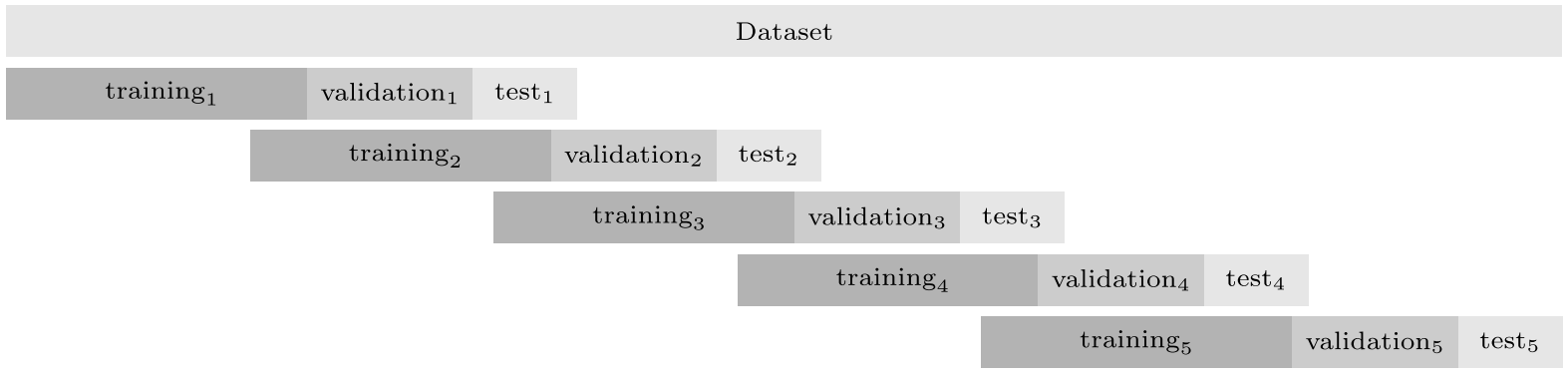


Figure 4.16: Each time series is divided into five blocks of training, validation and test sets.

are made. In this work, a test set is separated at the beginning, so it is not used neither for training model parameters nor hyperparameters tuning, which is done using a validation set, as mentioned by [MPK+05]. This test set is taken from the final time steps of the block and represents hold-out data that is used only when evaluating the final model, simulating the real life situation, where future data is not available. Afterward, the final 10% time steps of the remaining data are separated into a validation set, which is used to evaluate hyperparameters for a model trained based only on the training data set. This is the *out-of-sample* approach, as explained in [BHK18]. According to [CTSM17], out-of-sample validation is the most appropriate method when using real time series data, in comparison to k-fold cross validation, because it shows improved performance. Note that the approach of this work is different to *blocked* cross validation explained in [CTSM17], because in the evaluation pipeline, each block is treated independently, meaning that the first test set evaluates the first training and validation sets, the second test set evaluates the second training and validation sets, and so on, without merging the training and validation sets from previous blocks when evaluating test sets from following blocks. In [NMB17], a similar training, validation and test set separation procedure is performed, although they train the model in the training set, and then continue training using the validation set, adjusting the hyperparameters in an online fashion, which is different to this work, where the validation set is used to evaluate only a fixed hyperparameter value. They call their approach *rolling* cross validation. To select the best hyperparameters for the model, a random search is conducted, as explained in [BB12]. For each set of hyperparameters values, 5 training runs are conducted on each block, and the average validation loss is stored. At the end, the hyperparameters values that had the minimum average validation loss are selected, and the model is retrained considering all training and validation data, finally evaluating this retrained model over the test set, as explained by [BHK18]. The thresholds for explored hyperparameters values were selected after manual experimentation, allowing enough flexibility for the models to choose from a wide range of hyperparameters, having in mind that the goal of the work is more to compare methods to handle epistemic uncertainty and aleatoric uncertainty, rather than finding the specific hyperparameters values that work best.

4.2.2 Results

Recall that the hypothesis of this work includes the consideration of MCD as a variational inference technique for the epistemic uncertainty and the use of different distributions for the aleatoric uncertainty, for multistep probabilistic forecasting of energy time series.

To validate the first part, the proposed model using MCD was compared against similar neural network models using dropout only during training, and without using dropout at all. For the second part, the same model was compared when assuming different output distributions. Additionally, to explore how the model can be used for multistep forecasting, three cases where considered on how to perform multistep

Table 4.2: Range of hyperparameters explored through random search for re-inject model experiments.

Parameter	Min.	Max.	Scale
Hidden recurrent (LSTM) layers	1	3	Discrete
Hidden recurrent (LSTM) units per layer	10	100	Discrete
Learning rate	0.00001	0.1	Log
Weight decay	0.0001	0.1	Log
Dropout rate	0.1	0.5	Linear
Epochs	25	45	Linear

forecasting, that are explained below, looking towards having the best results.

Multistep forecasting with re-inject model

In this case, a neural network model is trained considering only one time step in the forecasted future for the training loss, similarly to the synthetic datasets case, but this model is used to perform multistep forecasting, first by performing a one-step prediction with the neural network model, and then creating a new input using the predicted value, to predict the second future time step, and iteratively continue until the whole horizon has been predicted. Thus, for each time step, the output is re-injected as input to the neural network. In this case, the model is called *re-inject* model. This procedure was already presented in Section 3.5.4 and, for this experiment, a normal distribution was assumed as the output of the model. Using it, PIs were estimated for 24 future time steps, thus for a whole day, from an input consisting of the values of the previous 48 hours. The resulting average metrics for all time steps are displayed in Figure 4.17, for wind speed datasets, and Figure 4.18, for the other datasets. The best set of hyperparameters values was chosen for each dataset and time step, with a random search of values, according to the values shown in Table 4.2.

It can be seen that, in general, MCD performs similarly to traditional dropout and no dropout for few time steps, but its handling of model uncertainty allows it to perform better for further time steps, having reduced IS. Looking at the PICP metric, it is clear that the coverage is higher using MCD, translating into more observations falling inside the PIs, except for B08 and D08 datasets. Interestingly, for these datasets, the IS is still low having a reduced PICP, meaning that, while having many observations out of the PI, its width was smaller than the linear penalization used by the metric. In general, as the coverage is still met for further time steps in the future, the CWC metric is also low in comparison with traditional or no dropout, being this difference bigger than in the case of IS, because it penalizes exponentially when the coverage is not met. In the case of dataset B08, the reduced coverage also makes CWC higher. For D05a dataset, CWC growths further for MCD, because unfortunately one of the repetitions had low coverage, which turned in an exponential penalty that, when averaged with the other repetitions, made

it grow large. This effect is also seen on the big standard deviations related to CWC in general. For the same D05a dataset, and for the UCI dataset, the average IS was similar for traditional and MCD models, meaning that, while the coverage was increased, it was due to a bigger PI width. Interestingly, for UCI dataset, IS and CWC metrics were similar for dropout and MCD. This could be related to the fact that the series has a distribution of values that is very different to a normal distribution, and it has two modes, as seen in Figure 4.15, making it difficult for MCD to improve a model that has a normal distribution output assumption. This is not the case of Canela 1 dataset, which also has a non-normal distribution, but has only one mode when looking at its logarithm distribution in the same figure.

To compare the models in detail, Tables 4.3, 4.4 and 4.5 show average metrics for 1, 12 and 24 steps ahead forecasting, including point forecasting metrics MAE and RMSE. While for 1 time step dataset B08 presents a notable improvement, this is not the case for the other datasets, as seen in Table 4.3, meaning that considering the model uncertainty for one-step ahead forecasting is not really necessary. Nevertheless, 12 step ahead results (Table 4.4) present mixed improvements, while 24 steps ahead results (Table 4.5) present a clear improvement using MCD. This means that, although not clear for short term prediction, considering model uncertainty with MCD makes the model more robust for medium or long term forecasting tasks. This effect was specially notable for dataset GEFCom2014-L, an electrical load consumption dataset, which has the smoothest distribution, closer to the normal distribution used by the model, as shown in Figure 4.15. Also, as shown by point prediction metrics, adding MCD does not reduce the performance of the model for point forecasting tasks, as shown by the MAE and RMSE metrics, meaning that models using MCD can safely be used for multistep point forecasting tasks as well.

In Table 4.6, average IS for all time steps is displayed for dataset GEFCom2014-L. It is clear that using MCD handles better the uncertainty for long term probabilistic forecasting tasks for this dataset. Also, standard deviation, in brackets, grows more slowly when using MCD than when not using it. This is in line with our consideration of the uncertainty of the model parameters, because it means that the model is more robust to changes in the random seed that controls the randomized parts of the training, like initialization of parameters and mini-batch selection.

As mentioned before, applying MCD to a neural network may be interpreted as simulating an ensemble of subnetworks that share parameters. To compare the proposed model against an ensemble, similar experiments as described were conducted using an ensemble of 20 similar neural networks using traditional dropout or no dropout at all, only for Canela 1 dataset, due to the very long time an ensemble takes to train. The results for all time horizons are displayed in Figure 4.19, where it can be seen that the ensemble behaves better than the other models, specially for long term forecasting, as expected. The coverage (PICP) is increased for all time steps, slightly making the PIs wider (PINAW), while keeping metrics IS and CWC low for PIs, and low RMSE for point prediction. The ensemble model always had the

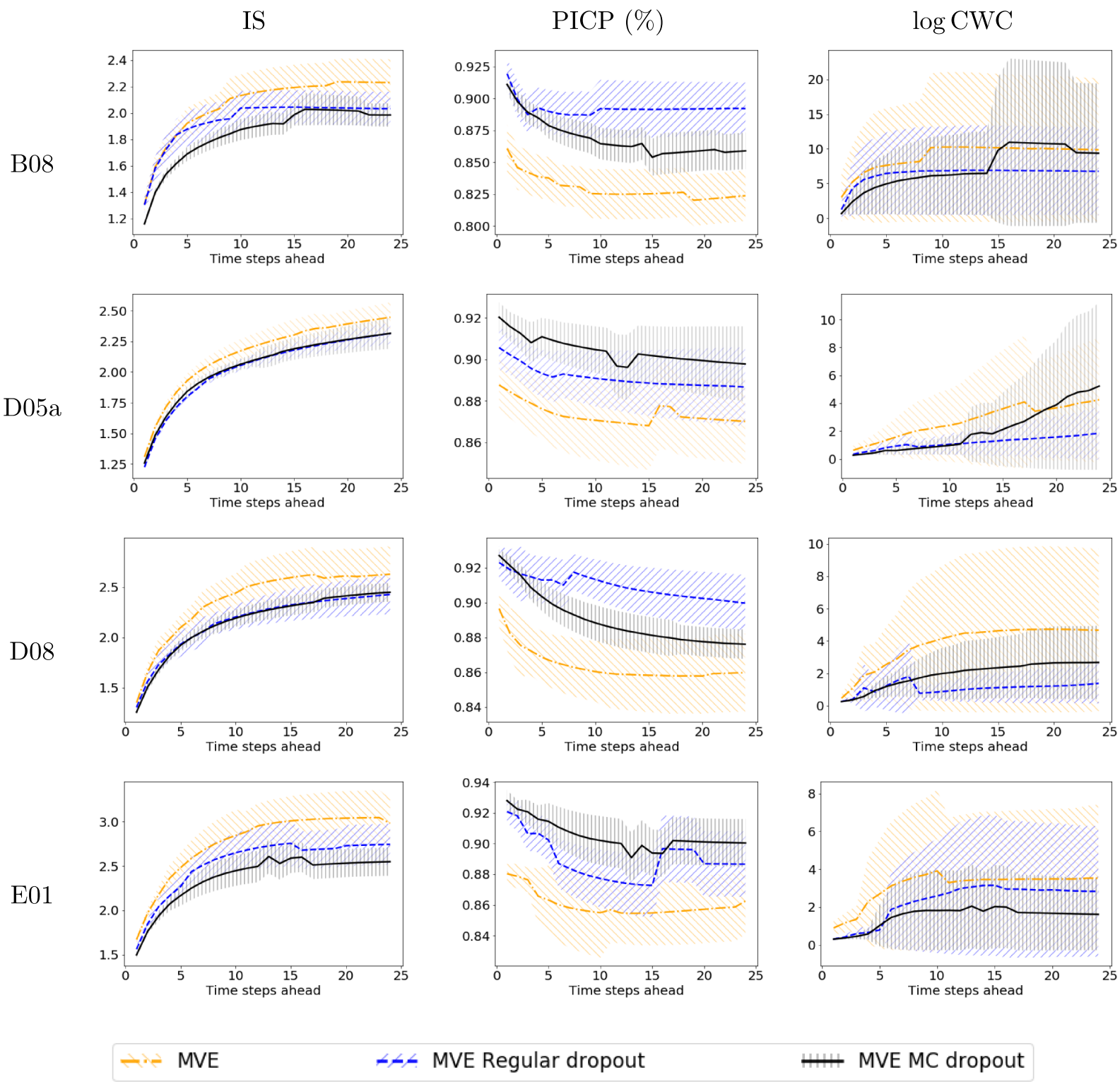


Figure 4.17: Average IS, PICP and CWC metrics, considering different prediction horizons, for wind speed datasets. Standard deviation of each metric is displayed as shaded areas.

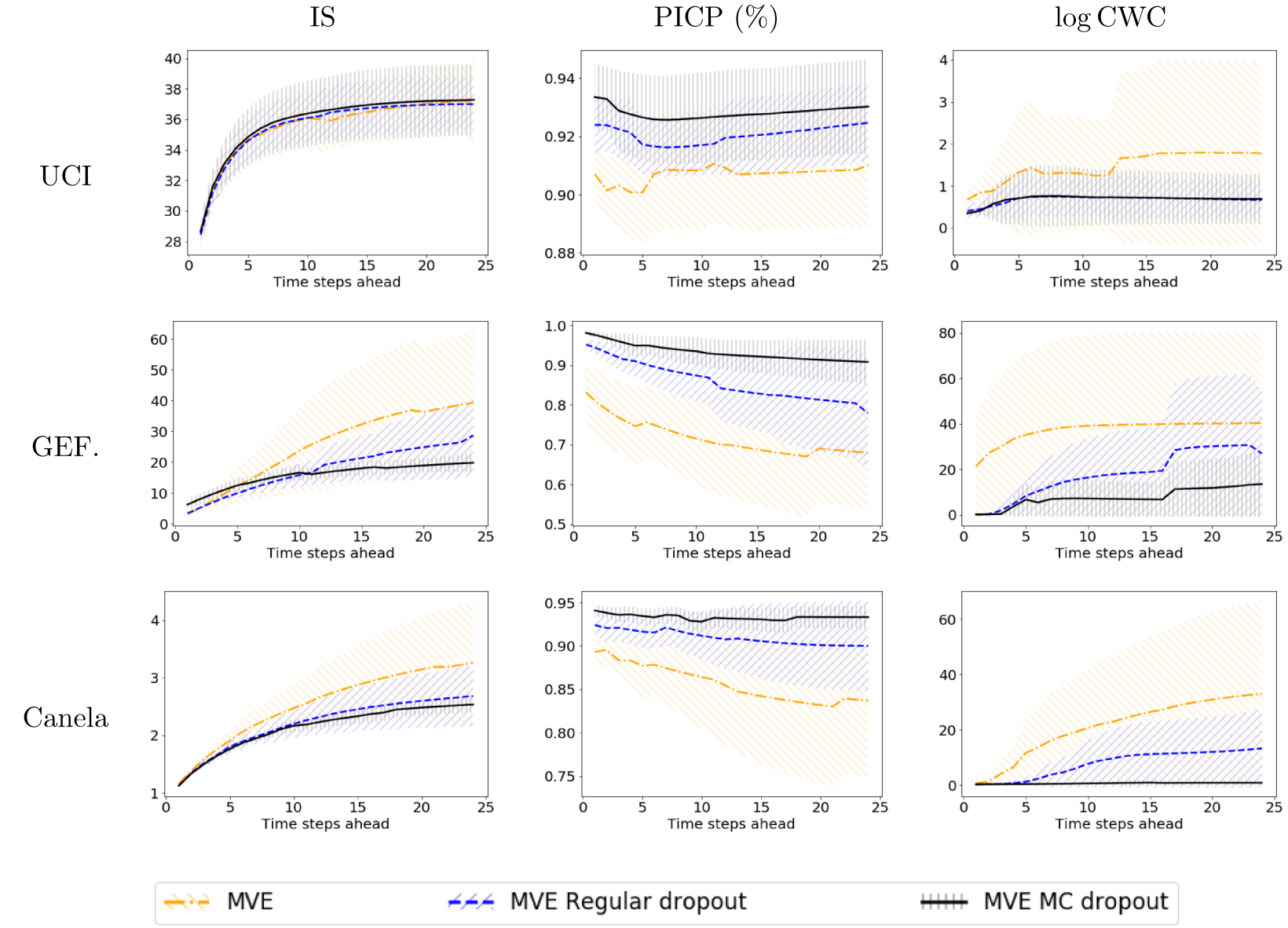


Figure 4.18: Average IS, PICP and CWC metrics, considering different prediction horizons, for electrical load and wind power datasets. Standard deviation of each metric is displayed as shaded areas.

Table 4.3: Average metrics for each dataset and dropout mode (N: No dropout, D: Traditional dropout, MCD: Monte Carlo Dropout), for 1 step ahead forecasting. Standard deviation in parenthesis. Big values of CWC are due to unmet coverage and its exponential penalty.

Set	Drop	CWC	PINAW (%)	PICP (%)	IS	MAE	RMSE
B08	N	19.517 (29.731)	30.2 (1.4)	86.1 (1.9)	8.07 (0.36)	1.60 (0.07)	2.02 (0.08)
	D	2.509 (1.787)	42.2 (2.2)	91.9 (1.2)	8.03 (0.37)	1.74 (0.06)	2.16 (0.07)
	MCD	**0.959** (**0.212**)	35.4 (0.4)	**91.1** (**0.7**)	**7.14** (**0.10**)	**1.58** (**0.03**)	**1.98** (**0.03**)
D05a	N	0.874 (0.425)	27.7 (1.2)	88.8 (1.3)	1.31 (0.02)	2.22 (0.11)	2.77 (0.13)
	D	0.414 (0.134)	27.7 (0.9)	**90.6** (**1.1**)	**1.22** (**0.02**)	**2.13** (**0.06**)	**2.66** (**0.08**)
	MCD	**0.312** (**0.038**)	29.8 (0.8)	92.0 (0.7)	1.26 (0.03)	2.18 (0.10)	2.70 (0.10)
D08	N	1.219 (0.588)	25.5 (0.8)	88.5 (1.2)	1.29 (0.02)	2.26 (0.11)	2.82 (0.12)
	D	0.559 (0.307)	25.8 (1.3)	89.8 (1.4)	**1.25** (**0.03**)	**2.22** (**0.04**)	**2.78** (**0.05**)
	MCD	**0.344** (**0.105**)	30.3 (4.8)	**92.1** (**1.5**)	1.34 (0.16)	2.28 (0.16)	2.83 (0.18)
E01	N	1.457 (0.442)	28.5 (0.6)	88.0 (0.7)	10.28 (0.22)	2.70 (0.12)	3.32 (0.17)
	D	**0.356** (**0.034**)	34.3 (0.8)	**92.1** (**0.7**)	9.62 (0.23)	2.74 (0.13)	3.34 (0.14)
	MCD	0.361 (0.035)	34.7 (0.5)	92.8 (0.7)	**9.22** (**0.19**)	**2.70** (**0.11**)	**3.27** (**0.11**)
UCI	N	0.968 (0.292)	36.6 (2.0)	**90.7** (**1.0**)	28.68 (0.91)	33.7 (2.7)	43.6 (2.7)
	D	0.509 (0.182)	37.9 (1.3)	92.4 (1.0)	**28.44** (**0.67**)	**32.9** (**2.0**)	**42.6** (**1.7**)
	MCD	**0.424** (**0.068**)	40.5 (2.7)	93.4 (1.1)	28.65 (0.92)	35.7 (3.2)	45.0 (3.4)
GEF	N	1.6×10^9 (3.7×10^9)	5.1 (0.5)	83.1 (7.4)	**3.03** (**0.72**)	26.1 (10.2)	31.4 (11.4)
	D	**0.081** (**0.009**)	8.1 (0.9)	**95.2** (**1.3**)	3.37 (0.30)	**20.7** (**5.0**)	**26.1** (**6.1**)
	MCD	0.158 (0.047)	15.8 (4.7)	98.1 (0.6)	6.20 (1.97)	503.7 (1086)	11479.7 (25618)
Can.	N	0.924 (0.411)	21.4 (1.4)	89.3 (1.3)	1.16 (0.03)	2.71 (0.56)	3.39 (0.56)
	D	0.261 (0.065)	23.4 (1.3)	**92.4** (**1.3**)	**1.12** (**0.02**)	**2.36** (**0.19**)	**3.00** (**0.14**)
	MCD	**0.253** (**0.009**)	25.3 (0.9)	94.1 (0.7)	1.13 (0.02)	2.47 (0.11)	3.07 (0.10)

Table 4.4: Average metrics for each dataset and dropout mode (N: No dropout, D: Traditional dropout, MCD: Monte Carlo Dropout), for 12 step ahead forecasting. Standard deviation in parenthesis. Big values of CWC are due to unmet coverage and its exponential penalty.

Set	Drop	CWC	PINAW (%)	PICP (%)	IS	MAE	RMSE
B08	N	28451 (47904)	42.3 (1.8)	82.5 (2.0)	13.32 (1.02)	1.61 (0.09)	2.04 (0.10)
	D	998 (656)	61.3 (3.3)	**89.2** (**2.2**)	12.57 (0.79)	1.93 (0.08)	2.37 (0.09)
	MCD	**598** (**366**)	45.4 (0.7)	86.3 (1.3)	**11.74** (**0.52**)	**1.58** (**0.05**)	**1.99** (**0.05**)
D05a	N	16.13 (23.83)	47.1 (2.3)	87.0 (2.0)	2.23 (0.11)	2.22 (0.11)	2.77 (0.13)
	D	**2.19** (**1.49**)	46.8 (1.9)	89.0 (1.4)	**2.11** (**0.05**)	**2.15** (**0.07**)	**2.68** (**0.08**)
	MCD	4.85 (6.13)	48.9 (2.1)	**89.7** (**1.5**)	2.12 (0.08)	2.19 (0.09)	2.71 (0.10)
D08	N	348.5 (746.2)	44.8 (3.2)	85.4 (3.5)	2.35 (0.21)	2.31 (0.17)	2.89 (0.20)
	D	126.8 (186.5)	44.3 (2.6)	86.2 (2.4)	2.28 (0.14)	**2.24** (**0.06**)	**2.79** (**0.07**)
	MCD	**12.6** (**18.7**)	47.7 (2.0)	**88.7** (**1.7**)	**2.19** (**0.06**)	2.25 (0.07)	2.79 (0.08)
E01	N	28.54 (31.42)	50.0 (1.8)	85.5 (2.0)	18.2 (1.6)	2.75 (0.15)	3.40 (0.17)
	D	18.72 (32.68)	53.1 (1.9)	87.5 (2.0)	16.6 (1.5)	2.72 (0.10)	3.31 (0.10)
	MCD	**5.14** (**7.16**)	55.1 (1.3)	**90.0** (**1.6**)	**15.4** (**1.1**)	**2.71** (**0.09**)	**3.28** (**0.11**)
UCI	N	2.49 (2.75)	47.5 (3.3)	**90.9** (**1.9**)	**35.9** (**1.8**)	34.3 (2.4)	44.0 (2.5)
	D	1.09 (0.67)	48.7 (3.4)	92.0 (1.3)	36.5 (1.8)	**33.1** (**2.2**)	**42.8** (**1.9**)
	MCD	**1.08** (**0.93**)	51.7 (5.2)	92.7 (1.6)	36.7 (2.2)	35.8 (3.2)	45.0 (3.4)
GEF	N	1.5×10^{17} (3.1×10^{17})	23.6 (2.0)	70.0 (12.8)	27.7 (16.8)	25.1 (8.8)	30.6 (9.9)
	D	5.3×10^{7} (1.2×10^{8})	31.3 (3.4)	84.2 (8.4)	19.1 (6.1)	21.3 (5.5)	26.7 (6.6)
	MCD	**1085** (**2421**)	37.9 (3.9)	**92.7** (**3.8**)	**16.6** (**2.0**)	**18.3** (**4.0**)	**23.2** (**5.3**)
Can.	N	8.6×10^{9} (1.9×10^{10})	44.8 (6.1)	85.4 (6.9)	2.66 (0.52)	2.62 (0.52)	3.31 (0.52)
	D	15732.43 (35176.26)	48.1 (6.5)	**90.8** (**3.4**)	2.32 (0.30)	**2.49** (**0.35**)	3.12 (0.35)
	MCD	**1.02** (**0.88**)	51.5 (4.2)	93.2 (1.1)	**2.23** (**0.12**)	2.50 (0.25)	**3.10** (**0.24**)

Table 4.5: Average metrics for each dataset and dropout mode (N: No dropout, D: Traditional dropout, MCD: Monte Carlo Dropout), for 24 step ahead forecasting. Standard deviation in parenthesis. Big values of CWC are due to unmet coverage and its exponential penalty.

Set	Drop	CWC	PINAW (%)	PICP (%)	IS	MAE	RMSE
B08	N	19131 (30056)	44.0 (1.9)	82.4 (2.0)	13.73 (1.08)	1.61 (0.08)	2.09 (0.10)
	D	**864 (555)**	61.7 (3.2)	**89.2 (2.1)**	12.52 (0.78)	1.93 (0.08)	2.41 (0.09)
	MCD	11718 (22479)	46.6 (1.0)	85.9 (1.4)	**12.22 (0.54)**	**1.57 (0.04)**	**2.04 (0.04)**
D05a	N	69.19 (86.98)	52.2 (2.7)	87.0 (2.0)	2.45 (0.12)	2.24 (0.09)	2.87 (0.11)
	D	**5.34 (5.76)**	50.6 (2.8)	88.7 (1.9)	**2.31 (0.09)**	**2.15 (0.07)**	**2.79 (0.09)**
	MCD	186.98 (405.49)	53.1 (2.6)	**89.8 (1.8)**	2.31 (0.12)	2.19 (0.09)	2.81 (0.11)
D08	N	96.5 (124.9)	49.5 (3.8)	85.7 (2.6)	2.53 (0.19)	2.33 (0.18)	2.99 (0.20)
	D	258.5 (417.2)	47.3 (3.3)	85.9 (2.6)	2.43 (0.17)	**2.24 (0.06)**	**2.88 (0.09)**
	MCD	**10.3 (6.9)**	50.7 (2.8)	**88.4 (1.8)**	**2.36 (0.07)**	2.27 (0.06)	2.91 (0.07)
E01	N	33.16 (47.38)	53.4 (2.5)	86.3 (2.2)	18.34 (1.71)	2.73 (0.15)	3.46 (0.19)
	D	16.03 (30.55)	58.4 (3.0)	88.7 (2.1)	16.88 (1.42)	2.80 (0.12)	3.47 (0.13)
	MCD	**4.04 (5.68)**	58.1 (1.2)	**90.0 (1.6)**	**15.68 (0.97)**	**2.72 (0.09)**	**3.35 (0.10)**
UCI	N	4.936 (7.938)	49.2 (5.8)	**91.0 (2.2)**	37.2 (2.7)	34.1 (3.3)	44.7 (3.0)
	D	**0.955 (0.500)**	50.3 (3.7)	92.5 (1.3)	**37.0 (1.7)**	**33.1 (2.2)**	**43.9 (1.9)**
	MCD	1.000 (0.814)	53.2 (5.2)	93.0 (1.6)	37.3 (2.3)	35.8 (3.2)	45.9 (3.4)
GEF	N	3.2×10^{17} (5.3×10^{17})	30.4 (4.0)	67.9 (14.3)	39.4 (23.5)	24.3 (7.5)	30.8 (8.6)
	D	5.0×10^{11} (9.9×10^{11})	41.0 (6.5)	77.9 (14.4)	28.7 (12.4)	23.4 (7.2)	30.2 (8.7)
	MCD	**723510** $\mathbf{(1.6 \times 10^6)}$	44.1 (5.5)	**90.8 (5.6)**	**19.8 (2.8)**	**18.0 (3.8)**	**23.6 (4.8)**
Can.	N	2.1×10^{14} (4.7×10^{14})	52.0 (11.5)	83.7 (8.9)	3.26 (1.08)	2.78 (0.72)	3.69 (0.68)
	D	573329.47 (1.3×10^6)	54.9 (10.4)	**90.0 (5.2)**	2.68 (0.52)	2.52 (0.39)	3.36 (0.39)
	MCD	**1.27 (0.87)**	58.5 (5.0)	93.3 (1.3)	**2.54 (0.14)**	**2.51 (0.20)**	**3.27 (0.16)**

Table 4.6: Average IS for GEFCom2014-L, comparing dropout modes. Standard deviation in parenthesis.

Steps	No Dropout	Traditional Dropout	MCD
1	**3.03 (0.72)**	3.37 (0.30)	6.20 (1.97)
2	**5.14 (1.67)**	5.15 (0.50)	7.99 (1.84)
3	7.43 (2.92)	**6.88 (0.79)**	9.60 (1.85)
4	9.84 (4.40)	**8.53 (1.17)**	11.07 (2.05)
5	12.32 (6.02)	**9.86 (1.56)**	12.45 (2.37)
6	14.21 (7.02)	**11.26 (2.02)**	13.27 (2.09)
7	16.59 (8.62)	**12.55 (2.50)**	14.27 (2.21)
8	18.91 (10.23)	**13.74 (2.97)**	15.14 (2.34)
9	21.15 (11.84)	**14.80 (3.40)**	15.90 (2.46)
10	23.80 (13.94)	**15.76 (3.78)**	16.55 (2.57)
11	25.79 (15.41)	16.62 (4.14)	**16.10 (1.85)**
12	27.65 (16.75)	19.11 (6.07)	**16.64 (2.01)**
13	29.30 (17.62)	19.91 (6.45)	**17.13 (2.17)**
14	30.86 (18.68)	20.65 (6.81)	**17.59 (2.32)**
15	32.29 (19.60)	21.34 (7.16)	**18.00 (2.46)**
16	33.60 (20.42)	21.98 (7.51)	**18.38 (2.60)**
17	34.80 (21.17)	23.05 (9.25)	**18.07 (2.14)**
18	35.91 (21.88)	23.69 (9.75)	**18.38 (2.24)**
19	36.93 (22.56)	24.30 (10.24)	**18.66 (2.34)**
20	36.32 (21.12)	24.88 (10.71)	**18.92 (2.44)**
21	37.15 (21.71)	25.42 (11.16)	**19.16 (2.53)**
22	37.93 (22.31)	25.94 (11.60)	**19.39 (2.62)**
23	38.66 (22.90)	26.45 (12.03)	**19.60 (2.71)**
24	39.35 (23.48)	28.71 (12.43)	**19.81 (2.80)**

lowest standard deviation, as expected, because the ensemble considers all sources of uncertainty, while the MCD proposal is the one having the lowest standard deviation after the ensemble. This is inline with the assumption that using MCD considers model uncertainty, thus reducing the uncertainty of the training procedure, reducing the variance, and finally making the technique more robust. In Table 4.7 the same metrics are detailed for some selected time steps, and also the average number of LSTM hidden layers and LSTM nodes of each layer. It can be seen that the ensemble model always has the smallest IS, being statistically different to MCD only for few time steps in terms of IS. As MCD disables some connections, the selection of hyperparameters chose networks with more hidden nodes, thus making the network more similar to those of an ensemble of models after applying dropout. It is important to note that when looking the difference between ensemble and MCD results, the CWC and RMSE difference is not significant for 1, 3 and 6 steps, meaning that for few time steps, MCD is a valid approach to consider the epistemic uncertainty, while for further time steps in the future, an ensemble continues performing better.

Multistep forecasting with auto-inject model

In this case, an auto-inject model was considered that, instead of being trained for one-step, it is trained to directly perform multistep forecasting, having the re-injection of the output as part of the model architecture, thus considering all future time steps when computing the training loss, as depicted in Figure 3.5. Other aspects of the network architecture, such as the Stacked LSTM, remained the same. In these experiments, early stopping was used, instead of a fixed number of epochs to stop training, and the hyperparameter selection was done over the values of Table 4.8. In addition to the use of MCD, these experiments focused on exploring how the output distribution assumption affects the forecasting metrics. To do so, one distribution was chosen and assumed for each experiment, from the following options: normal, log-normal, Gamma, and Weibull.

First, Table 4.9 displays the resulting metrics for all datasets when performing one time step forecasting. It can be seen that for most datasets, changing the output distribution did not make the model to perform significantly better in general, than when assuming a normal distribution, exception for UCI dataset, where other distributions worked better for most metrics, although not having big statistical significance. The results were much worse when assuming a Weibull distribution, not even able to converge in the case of Canela 1, represented as *nan* in the table. This means that the loss function was harder to optimize, and further work and fine-tuning is required to be able to get good results with it. The fact that the CWC and IS metrics may improve PIs for UCI dataset may be related to the distribution of its values, shown in Figure 4.15, that is very different to a normal distribution. Hence, changing the output distribution may be beneficial for one-step forecasting only for datasets that have distributions far from the normal one.

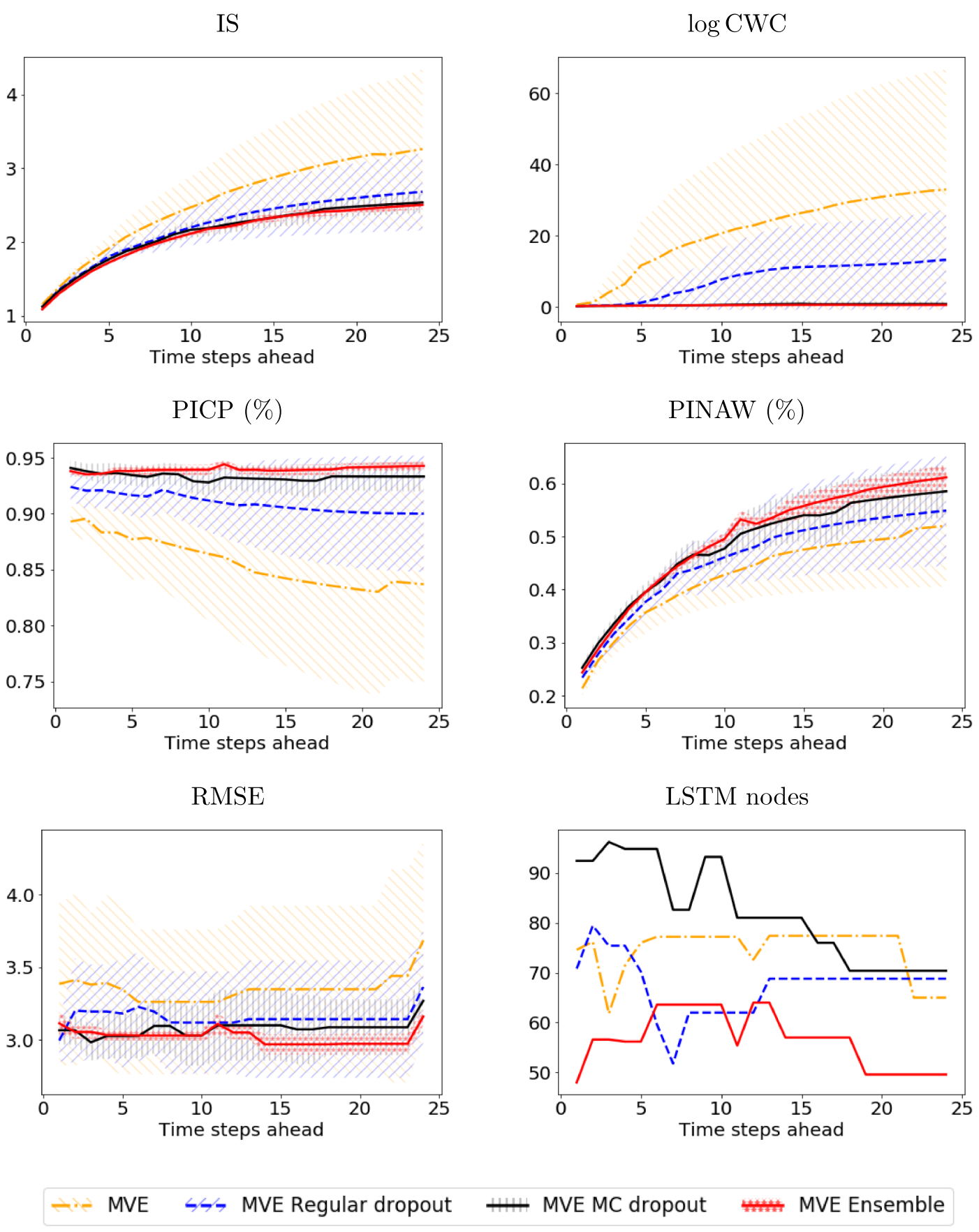


Figure 4.19: Average metrics for Canela 1 dataset, considering different prediction horizons, and considering an ensemble of models. Standard deviation of each metric is displayed as shaded areas.

Table 4.7: Metrics of best models for Canela 1 without dropout (N), with dropout (D), Monte Carlo Dropout (MCD) and an ensemble (E). Standard deviation in parenthesis. Colors represent statistical significance against closest value: **black** for p-value ≤ 0.05, **red** for ≤ 0.1, and **orange** for ≤ 0.2.

Step	Drop	IS	PICP (%)	PINAW (%)	CWC	RMSE	Layers	Nodes
1	N	1.16 (0.03)	89.3 (1.3)	21.4 (1.4)	0.92 (0.41)	3.39 (0.56)	2.2	74.6
	D	1.12 (0.02)	92.4 (1.3)	23.4 (1.3)	0.26 (0.06)	3.00 (0.14)	2.6	70.8
	MCD	1.13 (0.02)	94.1 (0.7)	25.3 (0.9)	0.25 (0.01)	3.07 (0.10)	2.6	92.4
	E	**1.10 (0.01)**	94.2 (0.3)	25.1 (0.3)	0.25 (0.00)	3.13 (0.07)	2.0	51.2
3	N	1.60 (0.06)	88.3 (2.1)	30.1 (2.4)	58.79 (122.32)	3.38 (0.49)	2.0	61.8
	D	1.52 (0.07)	92.1 (1.5)	31.7 (2.3)	0.52 (0.27)	3.20 (0.32)	2.2	75.4
	MCD	1.50 (0.04)	93.6 (0.9)	33.6 (1.3)	**0.36 (0.03)**	**2.98 (0.13)**	2.2	96.2
	E	**1.48 (0.01)**	93.9 (0.3)	33.7 (0.5)	0.38 (0.03)	3.07 (0.05)	2.2	57.0
6	N	2.06 (0.23)	87.8 (3.7)	37.2 (3.6)	8.1×10^5 (1.8×10^6)	3.26 (0.50)	2.2	77.2
	D	1.89 (0.13)	91.5 (2.0)	39.8 (3.2)	8.67 (18.00)	3.23 (0.37)	2.0	59.6
	MCD	1.87 (0.07)	93.3 (1.0)	41.7 (1.8)	0.48 (0.10)	3.03 (0.15)	2.0	94.8
	E	**1.82 (0.01)**	94.0 (0.4)	42.7 (0.8)	0.52 (0.05)	3.07 (0.05)	2.2	57.0
12	N	2.66 (0.52)	85.4 (6.9)	44.8 (6.1)	8.6×10^9 (1.9×10^{10})	3.31 (0.52)	1.8	72.6
	D	2.32 (0.30)	90.8 (3.4)	48.1 (6.5)	1.6×10^4 (3.5×10^4)	3.12 (0.35)	2.0	62.0
	MCD	2.23 (0.12)	93.2 (1.1)	51.5 (4.2)	1.02 (0.88)	3.10 (0.24)	2.4	81.0
	E	2.21 (0.03)	93.9 (0.3)	52.6 (0.9)	0.67 (0.05)	3.06 (0.04)	2.0	55.2
24	N	3.26 (1.08)	83.7 (8.9)	52.0 (11.5)	2.1×10^{14} (4.7×10^{14})	3.69 (0.68)	2.2	65.0
	D	2.68 (0.52)	90.0 (5.2)	54.9 (10.4)	5.7×10^5 (1.3×10^6)	3.36 (0.39)	2.6	68.8
	MCD	2.54 (0.14)	93.3 (1.3)	58.5 (5.0)	1.27 (0.87)	3.27 (0.16)	2.6	70.4
	E	2.51 (0.05)	94.0 (0.3)	60.2 (2.1)	**0.72 (0.06)**	**3.17 (0.05)**	1.8	48.2

Table 4.8: Range of hyperparameters explored through random search for auto-inject model experiments.

Parameter	Min.	Max.	Scale
Hidden recurrent (LSTM) layers	1	3	Discrete
Hidden recurrent (LSTM) units per layer	10	200	Discrete
Learning rate	0.00001	0.1	Log
Weight decay	0.0001	0.1	Log
Dropout rate	0.1	0.5	Linear

Table 4.9: Results for one-step forecasting, with standard dev. in parenthesis. PL-X metric is PL considering X number of quantiles. *nan* means the algorithm did not converge. Colors represent statistical significance against closest value: **black** for p-value ≤ 0.05, **red** for ≤ 0.1, and **orange** for ≤ 0.2.

Metric	Distribution	Canela 1	GEFCom2014-L	UCI	B08	D08
CRPS	Gamma	0.755 (0.033)	8.157 (0.919)	20.814 (0.503)	1.225 (0.134)	1.017 (0.150)
	Log normal	0.739 (0.026)	5.678 (0.446)	19.519 (0.659)	1.252 (0.117)	1.024 (0.139)
	Weibull	*nan* (*nan*)	11.381 (3.077)	31.057 (0.717)	1.825 (0.178)	1.736 (0.159)
	Normal	**0.689** (0.068)	5.601 (0.644)	19.188 (1.613)	1.156 (0.106)	1.125 (0.056)
PL-5	Gamma	0.294 (0.013)	2.351 (0.439)	7.770 (0.201)	0.461 (0.064)	0.379 (0.046)
	Log normal	0.291 (0.012)	2.156 (0.160)	**7.373** (0.529)	0.452 (0.045)	0.389 (0.054)
	Weibull	0.697 (0.645)	4.381 (1.166)	10.664 (0.264)	0.702 (0.072)	0.883 (0.203)
	Normal	**0.266** (0.023)	**1.999** (0.252)	7.732 (0.648)	**0.405** (0.018)	0.375 (0.016)
PL-10	Gamma	0.337 (0.015)	2.693 (0.482)	8.883 (0.233)	0.522 (0.073)	0.434 (0.054)
	Log normal	0.334 (0.014)	2.474 (0.177)	**8.398** (0.597)	0.513 (0.053)	0.449 (0.067)
	Weibull	0.826 (0.782)	5.051 (1.358)	12.276 (0.289)	0.807 (0.085)	1.026 (0.248)
	Normal	**0.305** (0.027)	**2.288** (0.278)	8.809 (0.731)	**0.460** (0.021)	0.432 (0.021)
PL-20	Gamma	0.358 (0.016)	2.860 (0.506)	9.426 (0.248)	0.553 (0.077)	0.462 (0.058)
	Log normal	0.354 (0.015)	2.629 (0.187)	**8.902** (0.631)	0.543 (0.056)	0.476 (0.070)
	Weibull	0.884 (0.842)	5.373 (1.449)	13.056 (0.301)	0.857 (0.091)	1.094 (0.268)
	Normal	**0.324** (0.029)	**2.429** (0.292)	9.338 (0.777)	**0.488** (0.022)	0.459 (0.023)
CWC	Gamma	0.345 (0.025)	939.948 (2096.924)	0.517 (0.069)	4.041 (4.915)	0.452 (0.111)
	Log normal	0.355 (0.016)	**0.269** (0.009)	0.535 (0.171)	**1.060** (0.477)	**0.375** (0.029)
	Weibull	1.657 (1.943)	0.614 (0.200)	0.919 (0.027)	1.286 (0.238)	1.575 (0.629)
	Normal	0.335 (0.041)	2.093 (4.205)	0.595 (0.179)	20.056 (13.212)	0.471 (0.104)
IS	Gamma	1.339 (0.071)	10.557 (1.578)	33.229 (1.334)	1.760 (0.196)	1.681 (0.313)
	Log normal	1.331 (0.037)	9.684 (0.329)	**30.461** (1.812)	1.844 (0.205)	1.548 (0.149)
	Weibull	4.750 (5.278)	21.063 (6.706)	53.772 (1.417)	3.227 (0.467)	4.847 (1.912)
	Normal	**1.259** (0.122)	**8.431** (0.613)	33.358 (2.546)	**1.592** (0.067)	1.552 (0.060)

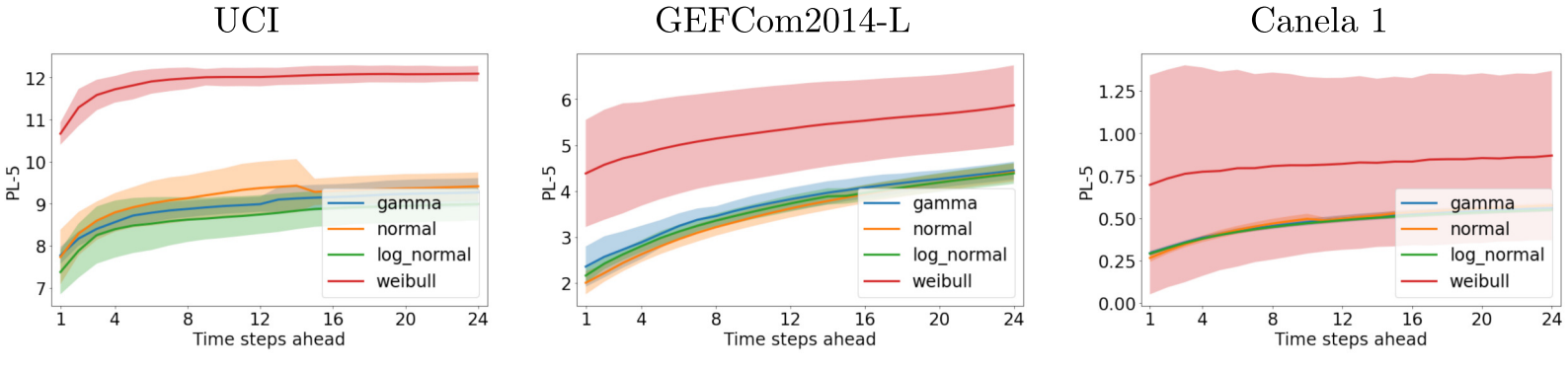


Figure 4.20: PL considering 5 quantiles (4 quantile cuts) for 24 time steps. Standard deviation shown as background color.

When considering multistep forecasting, the situation changes, and different output distributions worked better for each dataset, as shown in Table 4.10, where the metrics for 24 time steps forecasting are shown. In this case, for D08, Canela 1, GEFCom2014-L and UCI datasets, choosing a log-normal distribution presents an improvement in quantile performance metrics. In the case of B08 wind speed dataset, the normal distribution continues working best for 24-step forecasting. This was surprising at first, because the literature mostly uses a Weibull distribution to model the distribution of wind speed. To understand this, it can be seen in Figure 4.14 that B08 dataset presents a relatively symmetrical distribution, although this histogram considers the marginal distribution of all time steps, which may be different from the conditional distribution the model is learning. Interestingly as well, the PI generation was improved for GEFCom2014-L and B08, when using a Weibull distribution, according to the CWC metric, that is specially sensitive to PIs that do not match the desired nominal confidence. Thus, the Weibull distribution model is able to produce PIs that are closer to the desired confidence, and may be related to the fact that the distribution of data is non-symmetrical. As this does not happen for quantile metrics or CRPS, it means that the distribution that is assumed to produce better PIs may not be the closest one to the real distribution. From these observations, it is clear that the model behaves differently depending on the actual distribution of data and on the required task, so choosing the right distribution for the model requires both analyzing the data and studying how the model performs for one particular task. Moreover, it has to be noted that the statistical significance of the results for different distributions is reduced when multistep predictions are considered. Indeed, in Figure 4.20 it is shown how PL for 5 quantiles changes for intermediate time steps, where it can be seen that the separation among metrics gets shortened when considering more time steps, while all lines lay within the standard deviation of the metric of other lines. This fact may be produced because, while the exact shape of the distribution is important for few time steps, the errors accumulate in time, producing a more normal distribution for many time steps. It can be highlighted that the metrics are higher considering more time steps, which is expected due to more uncertainty of further time steps.

An interesting fact from these results is that the Weibull distribution did not perform well in most of

Table 4.10: Results for 24 steps forecasting, with standard dev. in parenthesis. PL-X metric is PL considering X number of quantiles. *nan* means the algorithm did not converge. Colors represent statistical significance against closest value: **black** for p-value ≤ 0.05, **red** for ≤ 0.1, and **orange** for ≤ 0.2.

Metric	Distribution	Canela 1	GEFCom2014-L	UCI	B08	D08
CRPS	Gamma	1.427 (0.044)	11.406 (0.342)	23.553 (0.682)	1.367 (0.119)	1.687 (0.030)
	Log normal	1.425 (0.020)	11.137 (0.558)	**22.094** (0.394)	1.278 (0.044)	**1.577** (0.024)
	Weibull	*nan* (*nan*)	15.148 (2.352)	34.787 (0.337)	1.801 (0.189)	1.787 (0.172)
	Normal	1.529 (0.102)	**10.753** (0.540)	24.153 (1.035)	1.263 (0.038)	1.614 (0.031)
PL-5	Gamma	0.560 (0.020)	4.444 (0.192)	9.269 (0.338)	0.525 (0.029)	0.635 (0.009)
	Log normal	**0.550** (0.011)	4.383 (0.228)	**8.989** (0.382)	0.503 (0.020)	**0.622** (0.008)
	Weibull	0.869 (0.498)	5.866 (0.869)	12.090 (0.184)	0.708 (0.081)	0.869 (0.120)
	Normal	0.576 (0.016)	4.404 (0.195)	9.413 (0.330)	**0.493** (0.011)	0.629 (0.010)
PL-10	Gamma	0.638 (0.022)	5.059 (0.216)	10.561 (0.377)	0.596 (0.032)	0.722 (0.010)
	Log normal	**0.626** (0.012)	4.983 (0.253)	**10.225** (0.429)	0.572 (0.023)	**0.707** (0.009)
	Weibull	1.019 (0.610)	6.739 (1.028)	13.898 (0.192)	0.814 (0.096)	1.010 (0.148)
	Normal	0.656 (0.018)	5.000 (0.217)	10.707 (0.370)	**0.559** (0.012)	0.715 (0.011)
PL-20	Gamma	0.677 (0.024)	5.373 (0.255)	11.197 (0.398)	0.632 (0.034)	0.765 (0.011)
	Log normal	**0.664** (0.013)	5.280 (0.266)	**10.834** (0.453)	0.605 (0.024)	**0.748** (0.009)
	Weibull	1.089 (0.659)	7.161 (1.101)	14.770 (0.202)	0.865 (0.103)	1.077 (0.160)
	Normal	0.696 (0.019)	5.295 (0.228)	11.343 (0.390)	**0.592** (0.012)	0.758 (0.012)
CWC	Gamma	0.637 (0.082)	843.004 (1882.076)	0.869 (0.231)	4.140 (5.648)	1.673 (1.026)
	Log normal	**0.591** (0.054)	44.269 (42.940)	0.915 (0.450)	63.170 (55.922)	2.160 (1.836)
	Weibull	1.866 (1.755)	**0.910** (0.279)	1.053 (0.149)	**1.595** (0.767)	1.562 (0.419)
	Normal	0.726 (0.197)	159.396 (159.245)	0.943 (0.406)	47.419 (54.285)	1.838 (0.847)
IS	Gamma	2.396 (0.052)	17.425 (0.866)	37.796 (0.951)	2.009 (0.082)	2.453 (0.110)
	Log normal	**2.339** (0.048)	17.100 (0.674)	**36.184** (1.097)	2.039 (0.120)	**2.345** (0.058)
	Weibull	5.351 (4.736)	26.913 (5.664)	58.097 (1.394)	3.367 (0.517)	4.798 (1.245)
	Normal	2.538 (0.093)	17.016 (0.803)	39.605 (2.545)	1.969 (0.071)	2.387 (0.060)

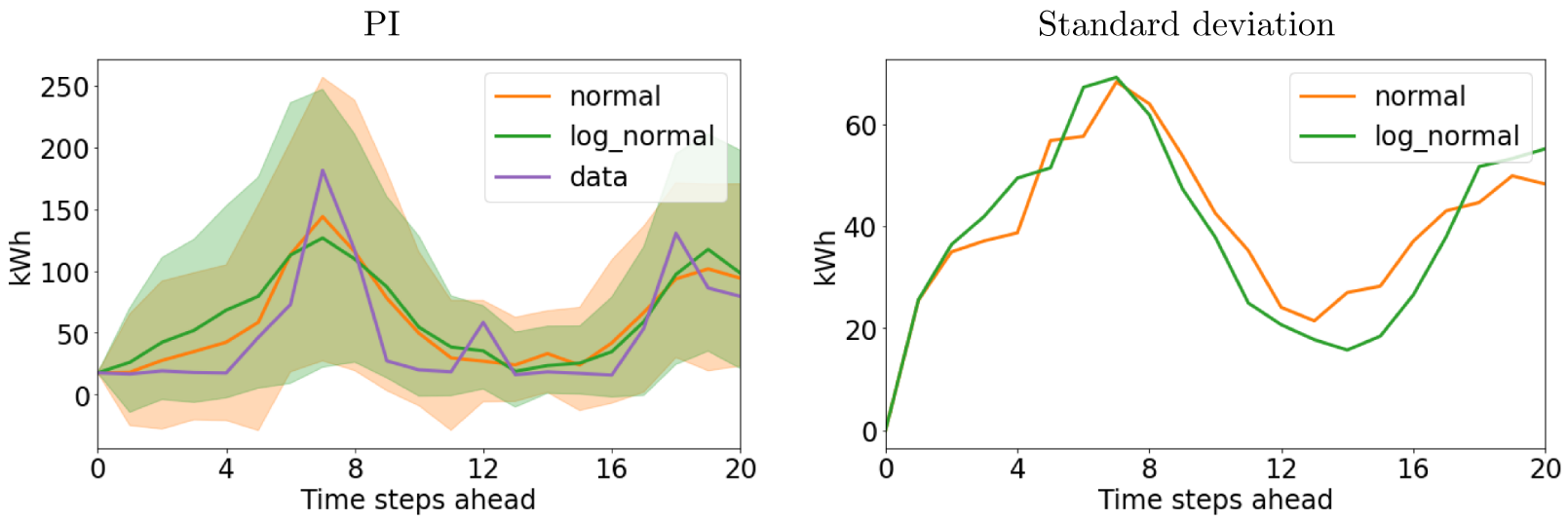


Figure 4.21: Left: Example of 90% confidence PIs generated for UCI dataset by normal and log-normal distributions. Mean predictions and real observations are shown as solid lines. Right: Empirical standard deviation of predictions for the same time steps, associated with the width of the PIs.

the experiments, in comparison with other distributions. This may be related to a difficulty of optimizing Eq. (3.6) by the gradient descent algorithm, as the derivatives of this function are highly sensitive to the values of λ and k, making the optimization unstable, even for small learning rate values. A future work can tackle this issue looking for a better parametrization of the log likelihood for this distribution.

In Figure 4.21, an example 24 time steps prediction for UCI dataset using the best models found for normal and log-normal distributions, corresponding to one sample of the test set, is shown. It can be seen that the model is correctly handling the heteroscedasticity of the series, showing a bigger uncertainty, thus wide PIs, for some time steps, while for some others, where the model is more certain, the PI is narrower. In this case, all real observations lay within the PIs, meaning that, while the mean prediction shown as a solid line is not exactly the same as the measured observation, the PIs are able to capture a region where the real observations occur. Also, it is highlighted in this figure that the normal distribution, as it is a symmetrical distribution that always assigns a non-negligible probability to negative values, produces PIs covering negative values, even when there is no negative values within this time series, related to power load. In contrast, the log-normal distribution is always positive, producing PIs that are skewed towards positive values, which can be seen in the figure as PIs being more towards the top of the chart, while the width of the PI remains roughly the same, as shown by the empirical standard deviation. Indeed, in Figure 4.22 the empirical distribution of a sample from these models is shown for one and 24 time step predictions. It is clear that in both cases the log-normal distribution produces values that are shifted to the right, meaning it does not assign probability to negative values. In the case of 24 time steps, a small probability lays within negative values, which is produced due to the accumulated errors within the intermediate time steps. An alternative that could be explored in future work is the use of truncated distributions, defined only for positive values.

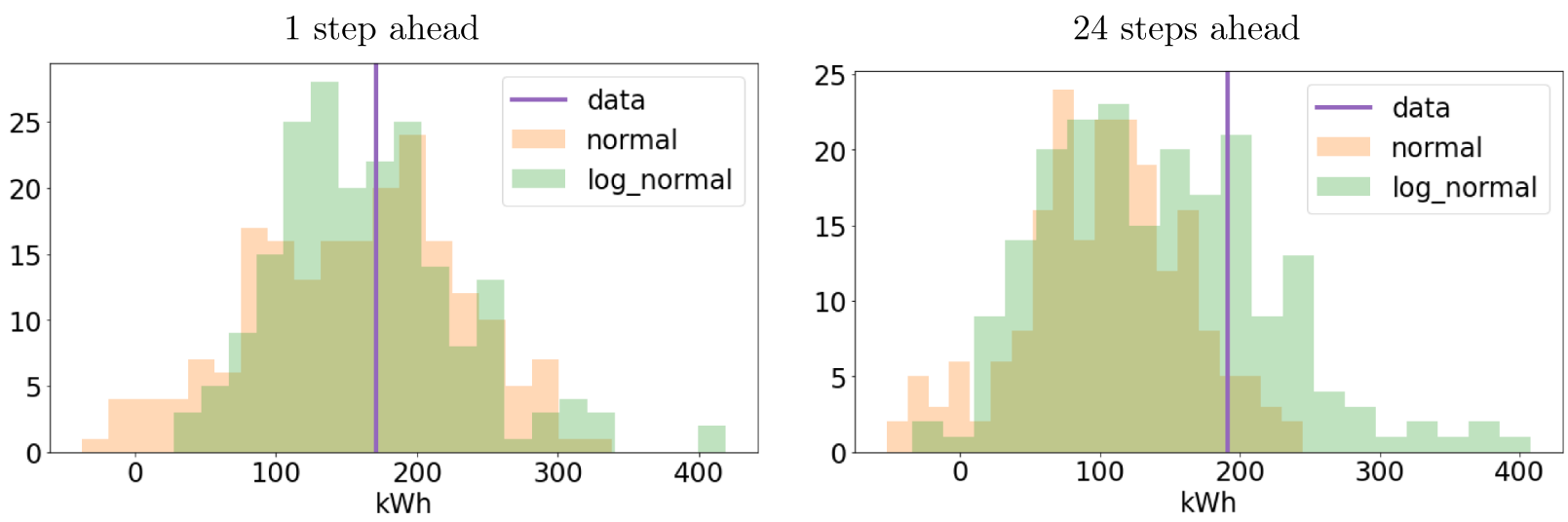


Figure 4.22: Examples of empirical distribution for 1 and 24 steps assuming normal and log-normal distributions for UCI dataset.

From Figure 4.15, it can be seen that the histograms of Canela 1 and UCI data show power distributions, so an experiment was run using a simple preprocessing step of adding 1 and applying logarithm to the original data before using training and evaluating the model, displaying the results in Figure 4.23. It can be seen that applying logarithm to the data improved the metrics for Canela 1 dataset, in comparison with not applying it, independently of which distribution is considered. Also, in Table 4.11 and Figure 4.24, it is shown the quantile estimation metrics for these preprocessed datasets, where it is clear that the Weibull distribution worked better for Canela 1, and log-normal for UCI, for multistep forecasting. In Figure 4.25, two samples of empirical distributions obtained by these models for Canela 1 are shown, where it can be seen that for step 24, the empirical distribution obtained using a Weibull distribution is more spread, allowing to capture that there are values that are not accumulated at the origin, as the one shown. Thus, this shows that, for power distributions, a preprocessing step may be still necessary, and that even after applying it, choosing a different distribution may lead to improved results.

To explore how this auto-inject model tackles epistemic uncertainty, the model was compared when using MCD and when using traditional dropout. In Table 4.12, results for one-step forecasting for Log-Normal distribution are shown, where mixed results are shown. For wind speed datasets, only CWC (related to PIs) was heavily improved. Instead, when looking the results for 24 time steps forecasting shown in Table 4.13, it is clear that the use of MCD presents an improvement for most datasets. As already mentioned for the re-inject model, this result indicates that the epistemic uncertainty is important when considering multistep forecasting, due to the fact that the errors accumulate, and considering the correct uncertainty in intermediate time steps, makes the model to perform better. Regarding the improvement for CWC metric for all time steps, it comes from the fact that CWC heavily punishes models that do not cover the required confidence, so a small lack of consideration of the correct uncertainty, makes the

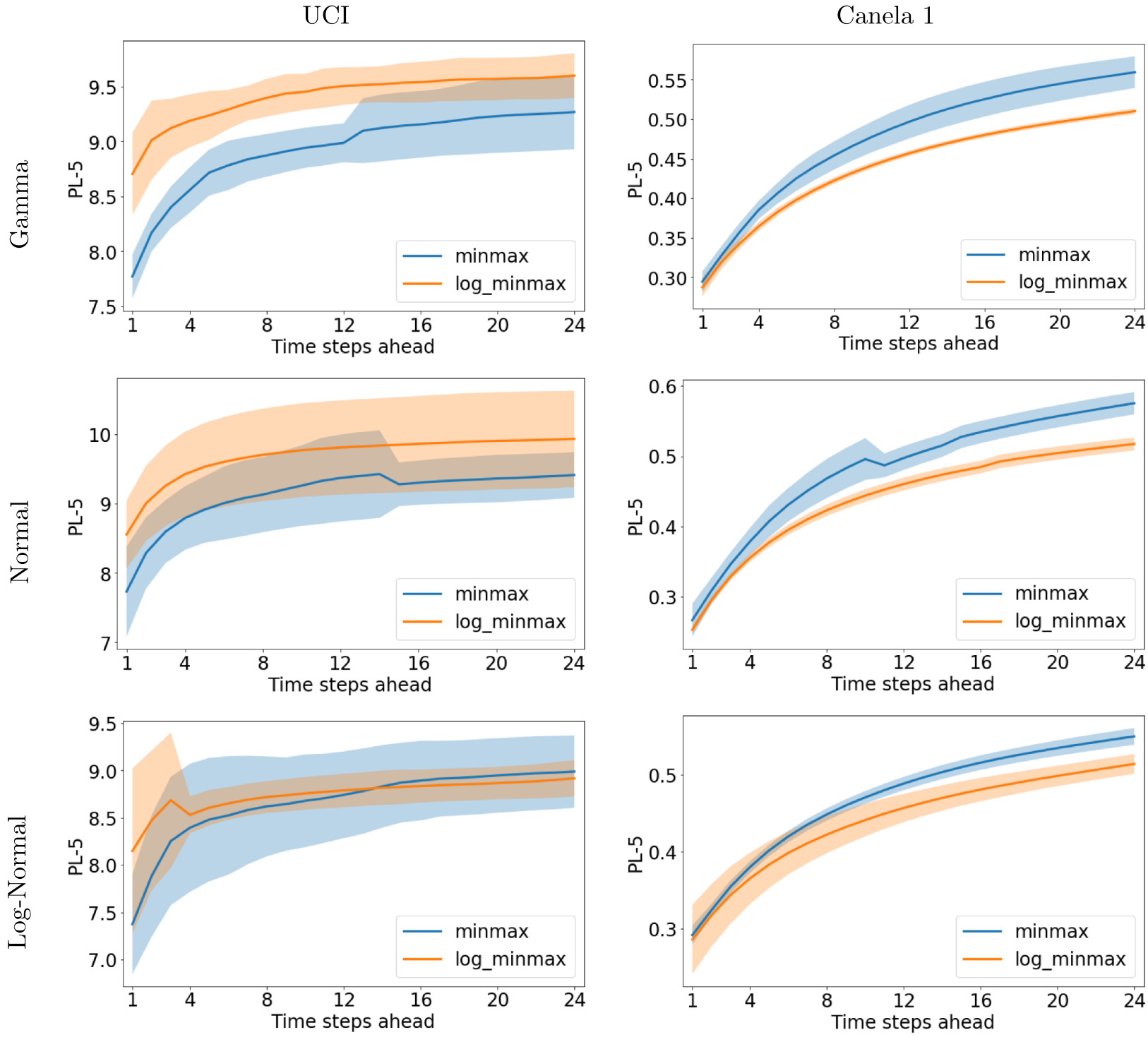


Figure 4.23: PL considering 5 quantiles for different distributions, considering preprocessing of Canela 1 and UCI datasets. Standard deviation shown as background color.

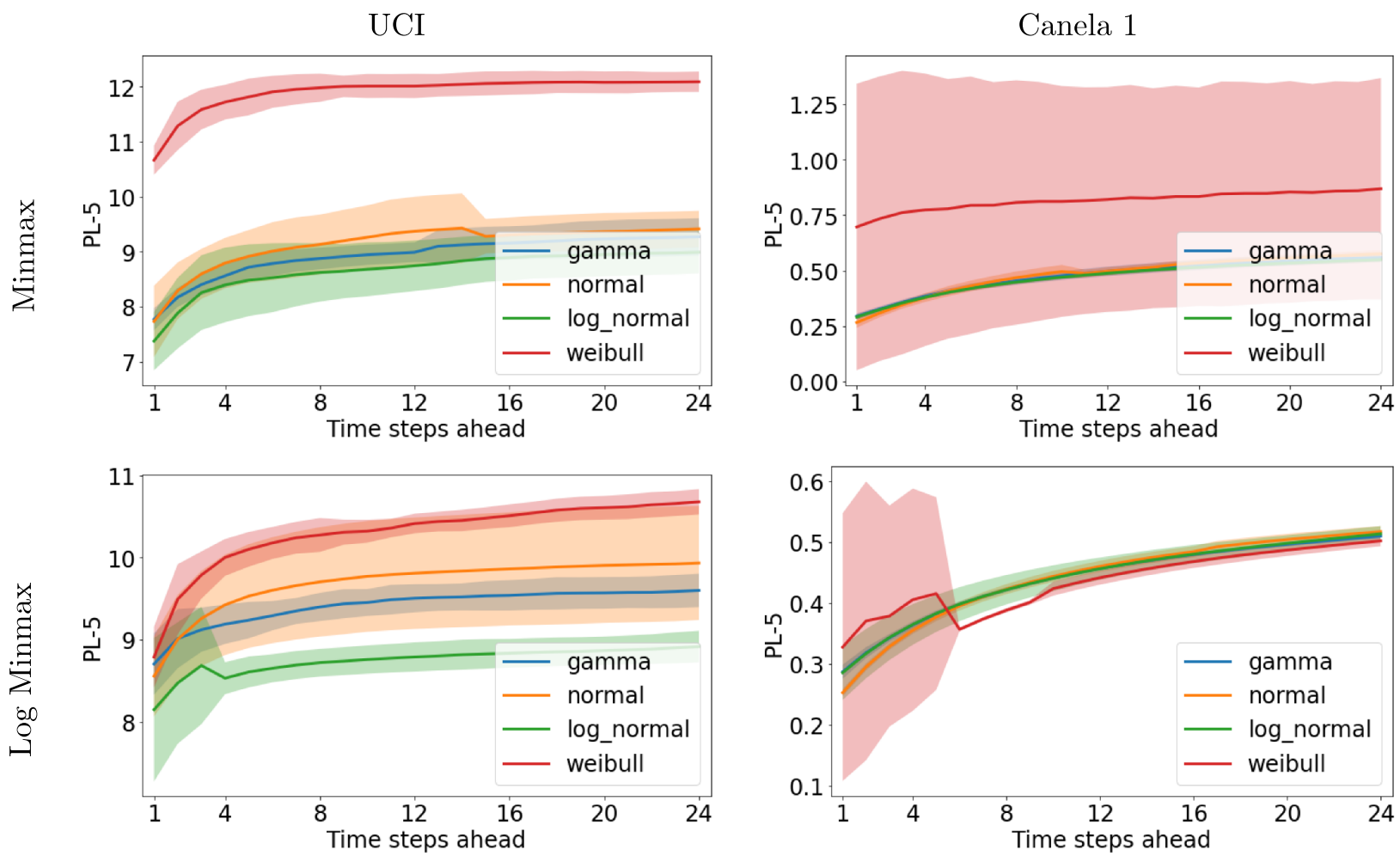


Figure 4.24: PL considering 5 quantiles for different distributions, considering preprocessing of Canela 1 and UCI datasets. Standard deviation shown as background color.

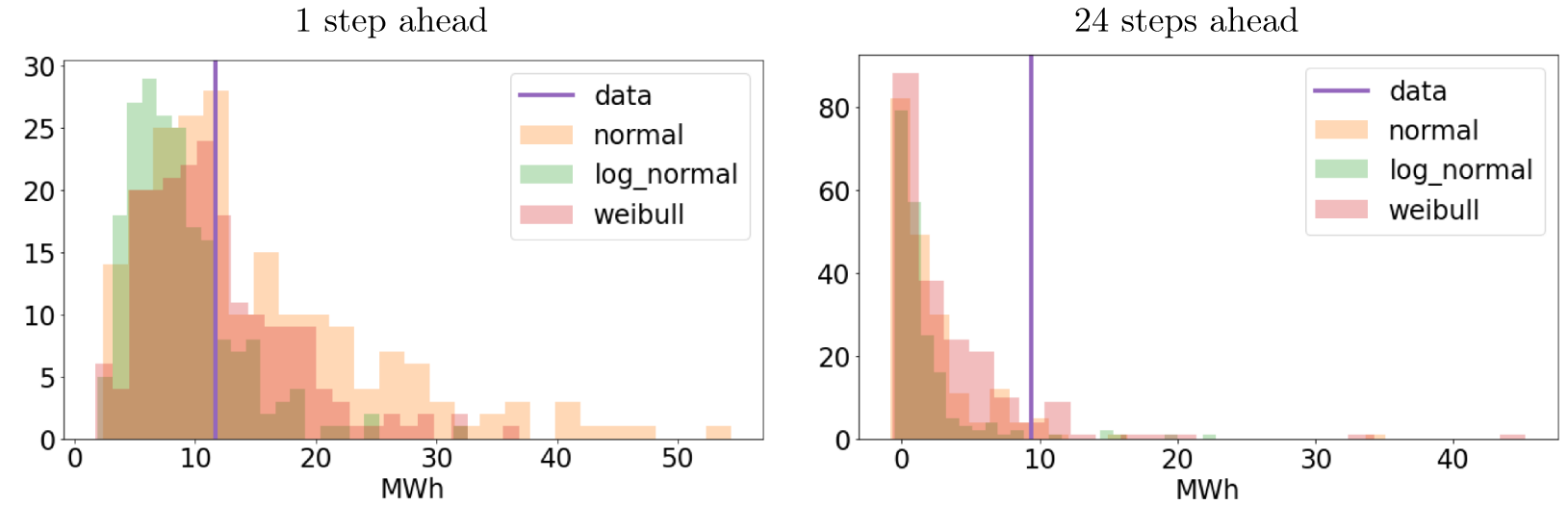


Figure 4.25: Examples of empirical distribution for 1 and 24 steps assuming different distributions for Canela 1 dataset, after applying logarithm preprocessing.

Table 4.11: Results for Canela 1 and UCI after preprocessing by adding one and applying logarithm, with standard dev. in parenthesis. PL-X metric is PL considering X quantiles. Highlighted values have p-value ≤ 0.05.

		Canela 1		UCI	
Metric	Distribution	1 step	24 steps	1 step	24 steps
PL-5	Gamma	0.287 (0.011)	0.510 (0.004)	8.704 (0.374)	9.600 (0.204)
	Log normal	0.286 (0.045)	0.515 (0.014)	8.147 (0.874)	**8.916** (0.192)
	Weibull	0.328 (0.220)	**0.502** (0.009)	8.788 (0.369)	10.681 (0.154)
	Normal	**0.252** (0.006)	0.517 (0.009)	8.556 (0.490)	9.935 (0.695)
PL-10	Gamma	0.337 (0.014)	0.592 (0.004)	9.968 (0.389)	10.982 (0.222)
	Log normal	0.331 (0.052)	0.597 (0.015)	9.568 (1.445)	**10.218** (0.219)
	Weibull	0.557 (0.611)	**0.576** (0.011)	10.086 (0.425)	12.232 (0.217)
	Normal	**0.289** (0.007)	0.591 (0.010)	9.754 (0.558)	11.297 (0.770)
PL-20	Gamma	0.363 (0.015)	0.633 (0.005)	10.599 (0.399)	11.674 (0.232)
	Log normal	0.354 (0.057)	0.640 (0.017)	10.942 (3.046)	**10.879** (0.236)
	Weibull	1.047 (1.558)	**0.612** (0.012)	10.771 (0.482)	13.041 (0.268)
	Normal	**0.307** (0.007)	0.629 (0.010)	10.347 (0.597)	11.975 (0.811)

PIs to be too narrow, thus not having the required coverage. This fact is important, because it shows that MCD is effectively considering the uncertainty of the auto-inject model. Then, after 5 repetitions of training using the same hyperparameters, but with different random seed, the metrics do not change significantly. To visualize the results for intermediate time steps, it is shown in Figure 4.26 how PL changed for four datasets, considering two distributions. It can be seen that the improvement of using MCD is consistent when evaluating more time steps, and that for Canela 1 dataset, the differences are not statistically significant.

Multistep forecasting with direct model

Both previous multistep models perform probabilistic forecasting by re-injecting samples of intermediate steps, each individual sample obtained from the estimated distribution parameters of each step. For the estimation of these parameters, only input past values and previous samples are used. This re-injecting procedure is done when using it to forecast test data. An ablation experiment was performed to check if this re-injection procedure was necessary, comparing to a simpler model that estimates the parameters of all future time steps at once, from a dense layer connected to the output of the LSTM layers, as depicted in Figure 4.27, thus ignoring the fact that values further in the future may depend on intermediate values as well, and assuming the dependence among them is only due to the model parameters. This experiment

Table 4.12: Comparison of metrics when using MCD and not using it, for one-step forecasting. Standard dev. in parenthesis. PL-X metric is PL considering X number of quantiles. Colors represent statistical significance: **black** for p-value ≤ 0.05, **red** for ≤ 0.1, and **orange** for ≤ 0.2.

Metric	Drop	Canela 1	GEFCom2014-L	UCI	B08	D08
CRPS	MCD	0.739 (0.026)	5.678 (0.446)	19.519 (0.659)	1.252 (0.117)	**1.024** (0.139)
	D	**0.672** (0.024)	5.587 (0.428)	19.142 (1.099)	**1.023** (0.062)	2.343 (0.816)
PL-5	MCD	0.291 (0.012)	**2.156** (0.16)	7.373 (0.529)	0.452 (0.045)	0.389 (0.054)
	D	**0.249** (0.011)	2.299 (0.297)	7.476 (0.422)	**0.409** (0.023)	**0.359** (0.026)
PL-10	MCD	0.334 (0.014)	2.474 (0.177)	8.398 (0.597)	0.513 (0.053)	0.449 (0.067)
	D	**0.283** (0.012)	2.617 (0.334)	8.497 (0.473)	**0.464** (0.026)	**0.408** (0.029)
PL-20	MCD	0.354 (0.015)	2.629 (0.187)	8.902 (0.631)	0.543 (0.056)	0.476 (0.070)
	D	**0.300** (0.013)	2.773 (0.352)	8.999 (0.500)	**0.491** (0.027)	**0.433** (0.031)
CWC	MCD	**0.355** (0.016)	**0.269** (0.009)	**0.535** (0.171)	**1.06** (0.477)	**0.375** (0.029)
	D	0.500 (0.209)	449.157 (999.084)	2.428 (2.763)	3.097 (1.277)	0.395 (0.118)
IS	MCD	1.331 (0.037)	9.684 (0.329)	30.461 (1.812)	1.844 (0.205)	1.548 (0.149)
	D	**1.095** (0.038)	**8.961** (0.992)	30.151 (1.746)	**1.602** (0.169)	**1.425** (0.082)

Table 4.13: Comparison of metrics when using MCD and not using it, for 24 steps forecasting. Standard dev. in parenthesis. PL-X metric is PL considering X number of quantiles. Colors represent statistical significance: **black** for p-value ≤ 0.05, **red** for ≤ 0.1, and **orange** for ≤ 0.2.

Metric	Drop	Canela 1	GEFCom2014-L	UCI	B08	D08
CRPS	MCD	1.425 (0.020)	11.137 (0.558)	**22.094** (0.394)	1.278 (0.044)	**1.577** (0.024)
	D	**1.398** (0.035)	11.121 (0.296)	23.427 (0.900)	1.284 (0.031)	2.688 (0.846)
PL-5	MCD	0.550 (0.011)	4.383 (0.228)	**8.989** (0.382)	**0.503** (0.020)	**0.622** (0.008)
	d	0.553 (0.012)	4.425 (0.189)	9.267 (0.393)	0.535 (0.013)	0.635 (0.019)
PL-10	MCD	0.626 (0.012)	4.983 (0.253)	**10.225** (0.429)	**0.572** (0.023)	**0.707** (0.009)
	D	0.628 (0.014)	5.018 (0.209)	10.513 (0.421)	0.605 (0.014)	0.720 (0.021)
PL-20	MCD	0.664 (0.013)	5.280 (0.266)	**10.834** (0.453)	**0.605** (0.024)	**0.748** (0.009)
	D	0.665 (0.015)	5.310 (0.220)	11.127 (0.437)	0.640 (0.015)	0.762 (0.022)
CWC	MCD	**0.591** (0.054)	44.269 (42.940)	0.915 (0.450)	**63.170** (55.922)	**2.160** (1.836)
	D	0.820 (0.210)	86.171 (102.602)	1.123 (0.451)	387.339 (364.161)	6.615 (3.747)
IS	MCD	2.339 (0.048)	17.100 (0.674)	**36.184** (1.097)	2.039 (0.120)	**2.345** (0.058)
	D	2.327 (0.048)	17.413 (0.783)	37.884 (1.727)	2.081 (0.084)	2.386 (0.055)

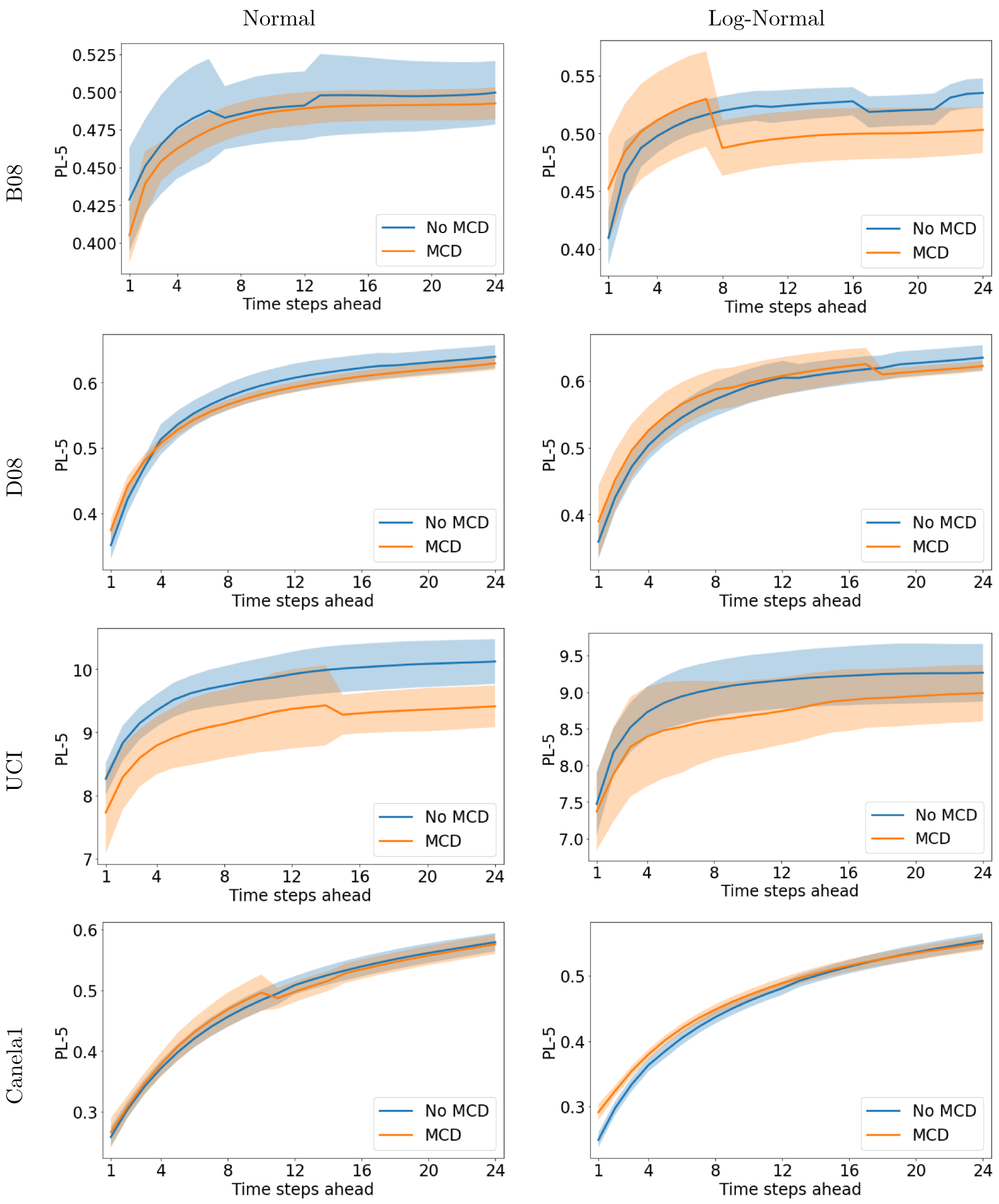


Figure 4.26: PL considering 5 quantiles for two distributions, considering MCD or not. Standard deviation shown as background color.

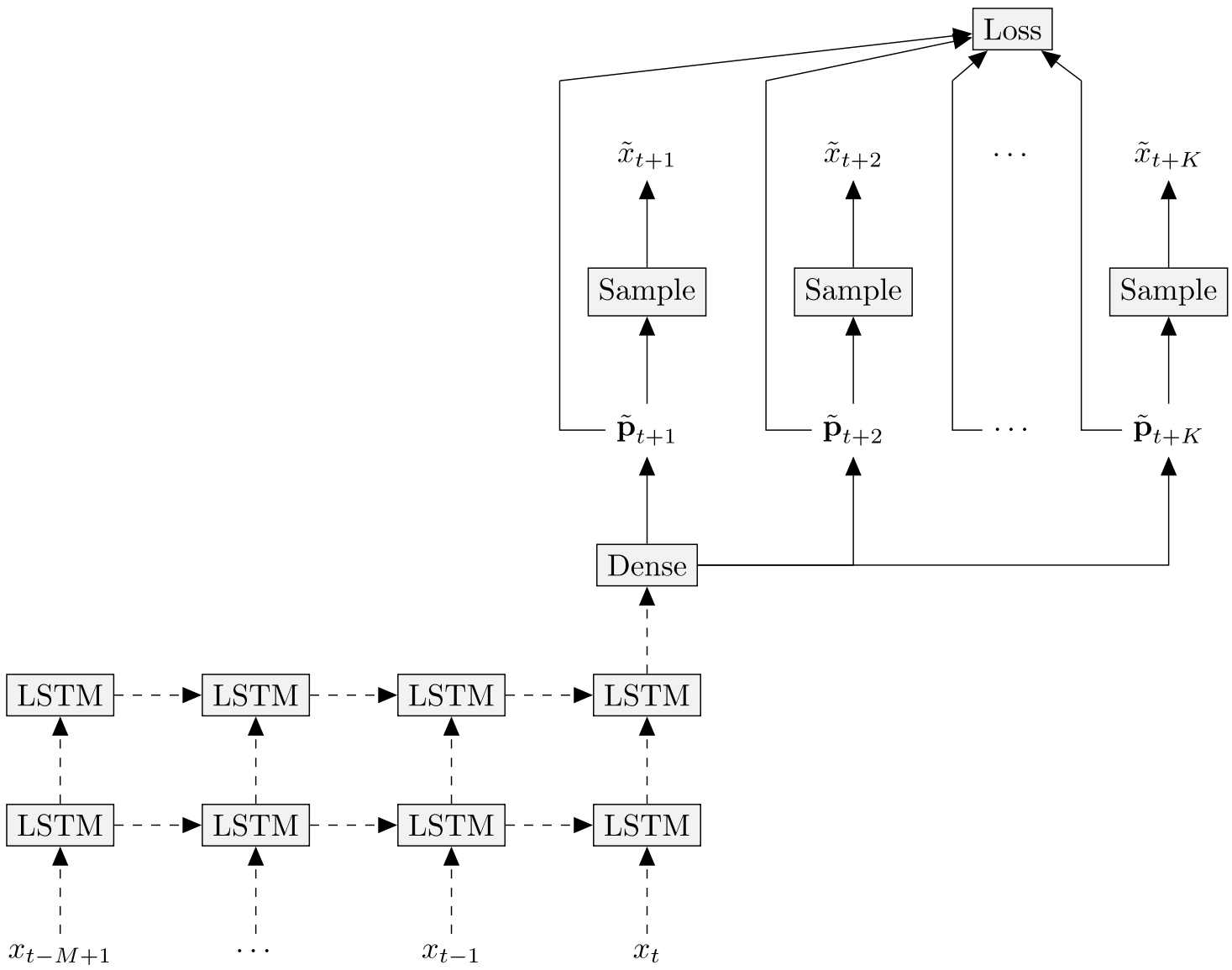
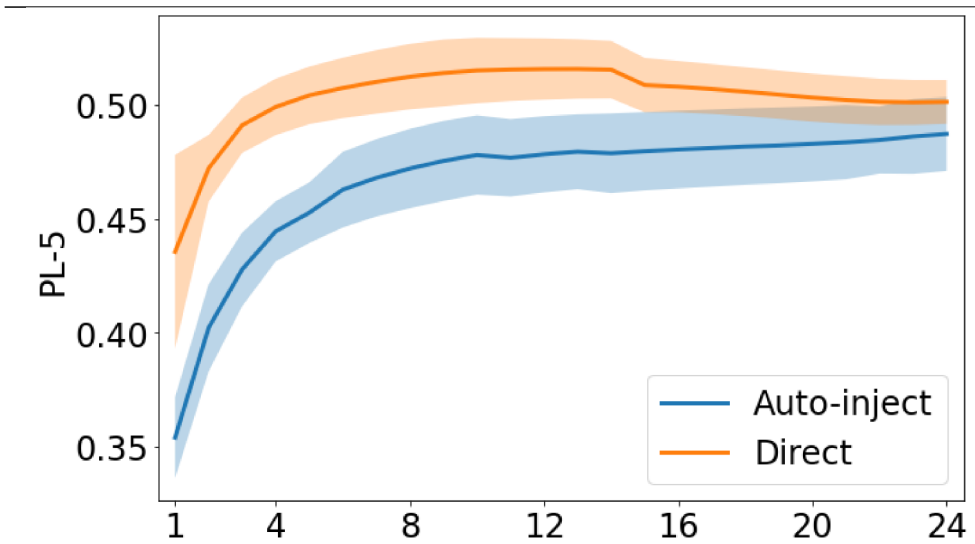


Figure 4.27: Simpler forecasting network that directly estimates the parameters of all future time steps.

was performed for B08 dataset assuming a normal output distribution, and results are summarized in Figure 4.28. It is clear that the re-injection of samples during training and test was important for this forecasting task. Interestingly, the difference between models gets shorter when further time steps in the future are considered. This may be related to the fact that the longer the time difference between input and forecasted values, less is known from the actual distribution of those values, and then they may be modeled by a distribution with parameters that are fixed and determined during training, instead of trying to estimate a distribution that depends on previous intermediate values, that, as being sampled, may be very different from the actual values that will occur.

Sensitivity to hyperparameters

An important part of the forecasting network architecture are the LSTM units and layers. It was studied how the forecasting performance changes when different values for the LSTM related hyperparameters are considered, for B08 dataset when assuming a normal output distribution. The results are depicted in



Step	Auto-inject	Direct
1	**0.353** (0.019)	0.435 (0.042)
2	**0.403** (0.019)	0.472 (0.015)
4	**0.445** (0.013)	0.499 (0.012)
6	**0.463** (0.016)	0.507 (0.013)
8	**0.473** (0.016)	0.512 (0.014)
10	**0.479** (0.016)	0.515 (0.014)
12	**0.478** (0.014)	0.516 (0.013)
16	**0.480** (0.015)	0.508 (0.011)
20	**0.483** (0.015)	0.503 (0.010)
24	**0.486** (0.015)	0.501 (0.010)

Figure 4.28: PL considering 5 quantiles for B08 dataset for different models. Standard deviation shown as background color and between parenthesis.

Figure 4.29. According to the left chart, increasing the number of LSTM units per layer seems to improve the performance, although only for the first few forecasted time steps. Further time steps in the future do not show a clear change. Interestingly, a few units highly reduces the performance, increasing the PL. This means that having only few units does not allow the neural network to learn correct features required to forecast using the last dense layer. It may be related to the fact that dropout is used, that disables some connections, and therefore more units are needed to learn the features robustly. Secondly, regarding the number of LSTM layers, results show that the best performance was obtained when only one LSTM layer was used, and that adding layers is harmful. This means that, for this particular forecasting task, a simple LSTM layer works best, in comparison with a stacked LSTM. This may be associated with dropout, as disabling connections can make deep learning harder, as features of the second layer depend on features extracted by the first layer.

Finally, it was checked how the performance changed when using different values for the dropout probability, as shown in Figure 4.30. The results show that the best dropout values were in between 0.2 and 0.3, making clear that the adjustment of this hyperparameter is fundamental to get the best performance from the model. Increasing the dropout probability seems to hurt the performance specially in the first few time steps, probably because then it is harder for the network to learn correct features, and also because it adds too much noise when forecasting. Using small values (less than 0.2 in the chart), seems to make the network not able to properly work for forecasting, possibly because it does not estimate the epistemic uncertainty correctly, or simply because of overfitting.

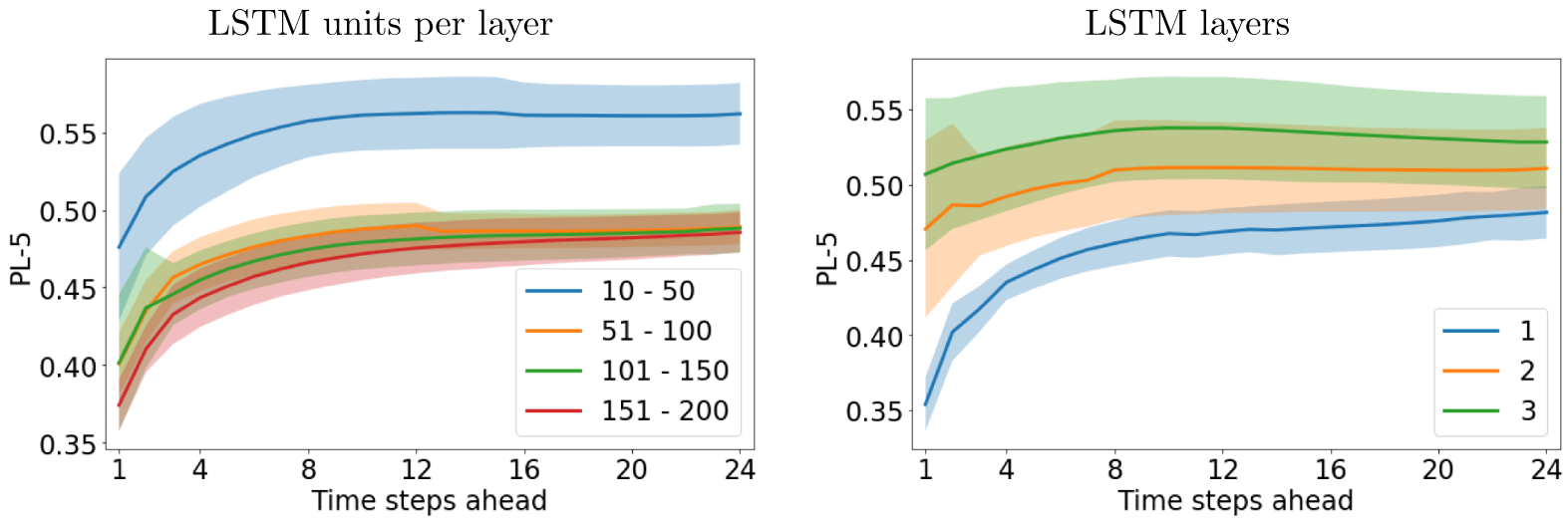
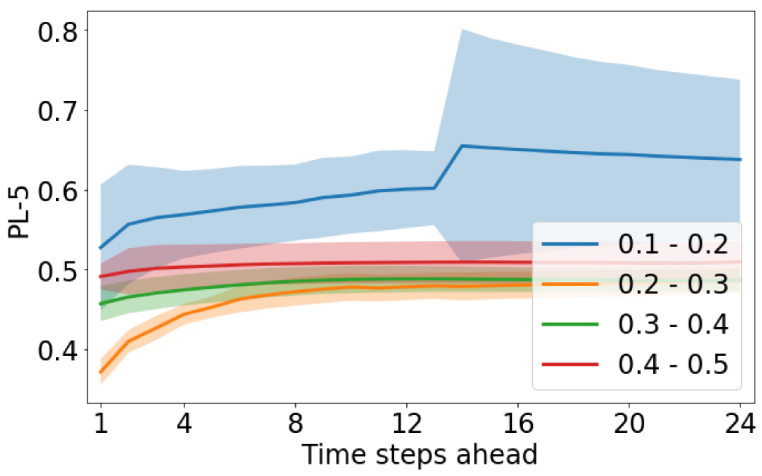


Figure 4.29: PL considering 5 quantiles for B08 dataset when considering different hyperparameters related to LSTM layers. Standard deviation shown as background color.



Step	0.2 — 0.3	0.3 — 0.4	0.4 — 0.5
1	**0.371** (0.016)	0.457 (0.022)	0.491 (0.016)
2	**0.410** (0.014)	0.465 (0.020)	0.498 (0.029)
4	**0.444** (0.013)	0.474 (0.020)	0.503 (0.028)
6	**0.463** (0.017)	0.481 (0.019)	0.506 (0.026)
8	**0.472** (0.017)	0.485 (0.018)	0.507 (0.026)
10	**0.478** (0.017)	0.487 (0.017)	0.508 (0.026)
12	**0.478** (0.017)	0.488 (0.016)	0.509 (0.026)
16	**0.480** (0.017)	0.487 (0.015)	0.509 (0.026)
20	**0.483** (0.016)	0.486 (0.014)	0.508 (0.026)
24	**0.486** (0.016)	**0.486** (0.013)	0.509 (0.026)

Figure 4.30: PL considering 5 quantiles for B08 dataset, considering different dropout levels. Low values were omitted from the table. Standard deviation shown as background color and between parenthesis.

4.3 Publications

The results presented in this thesis were included in the following published works:

- [SAVA19]: This conference paper showed the importance of the epistemic uncertainty for probabilistic forecasting, in particular for PI estimation, and how MCD can be incorporated to handle it, and compared a model that used it against other that did not.
- [SAVA20]: This journal paper extended the previous conference one, adding an additional wind power dataset and an additional consumer electrical load dataset in the assessment, adding a thorough review of the state-of-the-art, performing a deeper quantitative and qualitative comparison between models, and comparing the proposed model to an ensemble of models, for one dataset.
- [SVA23]: This journal paper extends the model to tackle different multi-step probabilistic forecasting tasks, and to better handle the aleatoric uncertainty, by assuming different output distributions. Also, it shows that a RNN that forecasts one step at a time and re-injects samples for intermediate time steps performs better than directly forecasting all future time steps without a recurrent approach. Finally, it adds evidence regarding the importance of considering the epistemic uncertainty.

4.4 Reproducibility and replicability

To improve the replicability of the experiments described in this work, the source code of the expermients has been publicly shared in the following GitHub repository: `https://github.com/cserpell/param_prob_forec`. Furthermore, the analysed datasets are available in the following repositories:

- UC Irvine Machine Learning Repository: Individual household electric power consumption `https://archive-beta.ics.uci.edu/dataset/235/individual+household+electric+power+consumption`.
- Global Energy Forecasting Competition 2014: Load forecasting data `http://blog.drhongtao.com/2017/03/gefcom2014-load-forecasting-data.html`.
- Wind and Solar resource measurement campaign (Chilean Energy Ministry): `http://walker.dgf.uchile.cl/Mediciones/`.
- Chilean National Electrical Coordinator Operational Data: Real operation and real generation: `https://www.coordinador.cl/operacion/graficos/operacion-real/generacion-real/`.

The experiments were conducted in the following computational infrastructure:

- Applied Computational Intelligence (INCA) Lab servers.
- CCTVal High Performance Computing cluster.

- NLHPC (ECM-02) supercomputing infrastructure.

The exact reproducibility of all experiments depends on random seeds that are set as a parameter of the programs used. As the results are shown as average and standard deviations using different random seeds, slightly different results may be obtained when reproducing them with other random seeds. It is expected that the results using MCD would be more similar to those in this work, as it handles the epistemic uncertainty.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this work, a deep learning model based on a recurrent neural network was proposed, to perform multistep probabilistic forecasting, that considers two kinds of uncertainties present in a deep learning probabilistic forecasting task: the aleatoric and the epistemic uncertainty. For the former, it is capable of assuming different families of output distributions, and, for the second, a variational technique was used, in particular Monte Carlo Dropout.

An extensive review of the state-of-the-art of probabilistic forecasting using deep learning techniques applied to the electricity domain was conducted and summarized in the theoretical framework of this document.

In general, the results show that the model, trained using the maximum likelihood criterion, is a valid approach for different kinds of time series probabilistic forecasting tasks, such as prediction intervals estimation, quantile regression and generating samples of the empirical distribution, which may be used to generate future scenarios as inputs for other systems.

The experiments that compared similar models using Monte Carlo dropout, using traditional dropout, and not using dropout at all, showed that considering the epistemic uncertainty is important for probabilistic forecasting. In fact, it seems to be specially important for multistep forecasting, as the uncertainty gets accumulated when considering more time steps as the prediction horizon. When the epistemic uncertainty is considered, a better estimation of the uncertainty may be established for those time steps further in the future. These results are summarized in Figures 4.17, 4.18 and 4.26. Furthermore, the results show that Monte Carlo dropout makes the neural network model to behave more similarly to an ensemble of neural network models, as the results associated with Figure 4.19 show. This reinforces the idea that Monte Carlo dropout is handling the model uncertainty.

Additionally, the experiments show that the choice of the output distribution is important, because the assumption of a normal distribution is not correct for all situations, specially for multistep forecasting, as noted in Figure 4.20. Which is the best distribution to choose depends on the dataset and the task. Experiments detailed in Table 4.10 showed that it is possible to even choose a distribution dissimilar to the correct data distribution to help for a prediction interval estimation task, when not much detail of the distribution is needed, but the coverage of the interval has to be covered. In the case of the Weibull distribution, the model was not able to produce satisfactory results, showing that further work is required to use it. While in principle the idea of choosing a distribution may alleviate the need for preprocessing the data, simple preprocessing steps were still necessary in some cases to get better results, as shown in Figure 4.23. This happened for a wind power dataset, that has power distribution in general, thus indicating other distributions different to those used in this study may be required for them.

Moreover, while the initial motivation was to handle the uncertainty for probabilistic forecasting tasks, it is concluded by results in Tables 4.3, 4.4 and 4.5 that the addition of Monte Carlo Dropout does not affect the point forecasting capabilities of the models. In some cases, it can even lead to improved average results, as the training procedure is more consistent.

Finally, to validate that the proposed multistep forecasting procedure considers the uncertainty that gets accumulated in intermediate time steps, a comparison of the proposed model with a direct forecasting model was performed. The direct model estimates the distribution of a future value directly, without considering the relationship that may exist among intermediate values and that future value. The results showed that the direct model was in fact not sufficient for the task, as summarized in Figure 4.28, concluding that the proposed sampling and re-injection procedure of this work is in fact necessary.

To conclude, this work shows that Monte Carlo Dropout, as a variational technique, can be used for time series probabilistic forecasting to include the epistemic uncertainty, and that changing the normality assumption usually done by many models is possible and may lead to improved results.

5.2 Future work

One line of future work could go for explore other variational techniques to handle the uncertainty of the model parameters. In particular, Monte Carlo Dropout makes the outputs of the network stochastic via sampling, a feature that may be problematic for tasks that require better interpretability of the model parameters and outputs. For example, other variational techniques are associated with assumptions that are smooth on the parameters distributions, such as Concrete Dropout, or assuming normal distributions. While in this work the deep architecture used is a recurrent one, Monte Carlo Dropout and other variational techniques may be used for other kind of networks, such as feedforward, convolutional, extreme learning machines, and others. Future work could extend these known architectures to handle the epistemic uncertainty, and perform probabilistic forecasting with them.

Additionally, other models could be used for aleatoric uncertainty that allow more flexible distributions, like mixture of distributions. Generally, mixtures of Gaussians are used in the literature, that may in principle model any distribution, while having more parameters that the base distributions included in this work. There are other more complex models based on neural networks that are also able to model other distributions, that may be interesting to explore, such as normalizing flows.

An important direction for further research is extending the model to handle multi-variate or multi-view time series, which may present dependencies among multiple variables. Modelling multivariate distributions is an important area of research in neural networks, where handling uncertainty is an important aspect where contributions can be made. This direction would also allow making predictions using exogenous variables which have to be forecasted as well.

This work focused on the results of probabilistic forecasting, without considering the direct extraction of latent features that may be used as a representation of the current state of a system. This is a big area of research that has decades of history, with names such as filtering, system identification, and others. It continues to be relevant, as recent neural network models worked for long time as black boxes, and currently there is a need to not only get good predictions, but also useful representations and extract information regarding the state of electrical systems.

Finally, by the side of applications, this work focused on time series of the energy domain. Many other important areas of society may benefit from this kind of research, where uncertainty is an everyday feature to consider when making decisions.

Bibliography

[BB12] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, volume 37, pages 1613–1622, Lille, 2015. PMLR.

[BHH+18] Jonathan D Black, Alex Hofmann, Tao Hong, Joseph Roberts, and Pu Wang. Weather Data for Energy Analytics: From Modeling Outages and Reliability Indices to Simulating Distributed Photovoltaic Fleets. *Power and Energy Magazine*, 16(3):43–53, 2018.

[BHK18] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction. *Computational Statistics & Data Analysis*, 120(April):70–83, 2018.

[Bis94] Christopher M. Bishop. Mixture Density Networks. Technical report, Neural Computing Research Group, 2 1994.

[BOSK15] Mehmet Bilgili, Arif Ozbek, Besir Sahin, and Ali Kahraman. An overview of renewable electric power capacity and progress in new technologies in the world. *Renewable and Sustainable Energy Reviews*, 49:323–334, 2015.

[BRGR21] Johannes Bracher, Evan L. Ray, Tilmann Gneiting, and Nicholas G. Reich. Evaluating epidemic forecasts in an interval format. *PLoS computational biology*, 17(2):e1008618, 2021.

[CKW20] M. Chudý, S. Karmakar, and W. B. Wu. Long-term prediction intervals of economic time series. *Empirical Economics*, 58(1):191–222, 2020.

[CLR96] George Chryssolouris, Moshin Lee, and Alvin Ramsey. Confidence interval prediction for neural network models. *IEEE Transactions on Neural Networks*, 7(1):229–232, 1996.

[CTSM17] Vitor Cerqueira, Luis Torgo, Jasmina Smailović, and Igor Mozetič. A comparative study of performance estimation methods for time series forecasting. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 529–538, Tokyo, 2017.

[CVG+14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1724–1734, Doha, Qatar, 2014.

[Cyb89] G. Cybenko. Approximation by Superpositions of a Sigmoid Function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[CZD+18] Lilin Cheng, Haixiang Zang, Tao Ding, Rong Sun, Miaomiao Wang, Zhinong Wei, and Guoqiang Sun. Ensemble recurrent neural network based probabilistic wind speed forecasting approach. *Energies*, 11(8), 2018.

[DDP15] Mohini P. Darji, Vipul K. Dabhi, and Harshadkumar B. Prajapati. Rainfall forecasting using neural network: A survey. In *2015 International Conference on Advances in Computer Engineering and Applications, ICACEA 2015*, pages 706–713, Ghaziabad, India, 2015.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[DG17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[Egi16] Egil Martinsson. *WTTE-RNN: Weibull Time to Event Recurrent Neural Network*. PhD thesis, University of Gothenburg, 2016.

[FKG10] Martin Felder, Anton Kaifel, and Alex Graves. Wind power prediction using mixture density recurrent neural networks. In *European Wind Energy Conference and Exhibition 2010, EWEC 2010*, volume 5, pages 3417–3424, 2010.

[FRHC21] Alessandro Fanfarillo, Behrooz Roozitalab, Weiming Hu, and Guido Cervone. Probabilistic Forecasting using Deep Generative Models. *Geoinformatica*, 25:127–147, 2021.

[FvA14] Otto Fabius and Joost R. van Amersfoort. Variational recurrent auto-encoders, 2014.

[GA14] John Geweke and Gianni Amisano. Analysis of Variance for Bayesian Inference. *Econometric Reviews*, 33(1-4):270–288, 2014.

[Gam17] John Cristian Borges Gamboa. Deep learning for time-series analysis, 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning. MIT Press, 2016.

[GBW+19] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic Forecasting with Spline Quantile Function RNNs. *Proceedings of Machine Learning Research*, 89:1901–1910, 2019.

[GG15] Yarin Gal and Zoubin Ghahramani. On Modern Deep Learning and Variational Inference. In *Advances in Approximate Bayesian Inference workshop, Neural Information Processing Systems (NIPS)*, volume 2, pages 1–9, 2015.

[GG16a] Yarin Gal and Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 1019–1027, Barcelona, 2016. Curran Associates, Inc.

[GG16b] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, volume 48, pages 1050–1059, New York, NY, 2016. PMLR.

[GK14] Tilmann Gneiting and Matthias Katzfuss. Probabilistic Forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014.

[Gra11] Alex Graves. Practical Variational Inference for Neural Networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, pages 2348–2356, Granada, 2011. Curran Associates Inc.

[Gra13] Alex Graves. Generating sequences with recurrent neural networks, 2013.

[GRK+21] Adèle Gouttes, Kashif Rasul, Mateusz Koren, Johannes Stephan, and Tofigh Naghibi. Probabilistic time series forecasting with implicit quantile networks, 2021.

[GWYK18] Dahua Gan, Yi Wang, Shuo Yang, and Chongqing Kang. Embedding based quantile regression neural network for probabilistic load forecasting. *Journal of Modern Power Systems and Clean Energy*, 6(2):244–254, 2018.

[HD97] J. T. Gene Hwang and A. Adam Ding. Prediction Intervals for Artificial Neural Networks. *Journal of the American Statistical Association*, 92(438):748–757, 1997.

[HF16] Tao Hong and Shu Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938, 2016.

[HGD+19] Sue Ellen Haupt, Mayte Garcia Casado, Michael Davidson, Jan Dobschinski, Pengwei Du, Matthias Lange, Timothy Miller, Corinna Möhrlen, Amber Motley, Rui Pestana, and John Zack. The use of probabilistic forecasts: Applying them in theory and practice. *IEEE Power and Energy Magazine*, 17(6):46–57, 2019.

[HKKH+02] Eduard Hofer, Martina Kloos, Bernard Krzykacz-Hausmann, Jörg Peschke, and Martin Woltereck. An approximate epistemic uncertainty analysis approach in the presence of epistemic and aleatory uncertainties. *Reliability Engineering and System Safety*, 77(3):229–238, 2002.

[HL18] Yaoyao He and Haiyan Li. Probability density forecasting of wind power using quantile regression neural network and kernel density estimation. *Energy Conversion and Management*, 164(January):374–384, 2018.

[HLSK17] Kostas Hatalis, Alberto J. Lamadrid, Katya Scheinberg, and Shalinee Kishore. Smooth Pinball Neural Network for Probabilistic Forecasting of Wind Power, 2017.

[HNM14] Ashraf Ul Haque, M. Hashem Nehrir, and Paras Mandal. A hybrid intelligent model for deterministic and quantile regression approach for probabilistic wind power forecasting. *IEEE Transactions on Power Systems*, 29(4):1663–1672, 2014.

[HPF+16] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob J. Hyndman. Probabilistic energy forecasting: Global Energy Forecasting Competition 2014 and beyond. *International Journal of Forecasting*, 32(3):896–913, 2016.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2:359–366, 1989.

[HU97] Sepp Hochreiter and J Urgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[KD09] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? Does it matter? *Structural Safety*, 31(2):105–112, 2009.

[KFKN16] Abdollah Kavousi-Fard, Abbas Khosravi, and Saeid Nahavandi. A new fuzzy-based combined prediction interval for wind power forecasting. *IEEE Transactions on Power Systems*, 31(1):18–26, 2016.

[KG17] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

[KH01] Roger Koenker and Kevin F. Hallock. Quantile Regression. *Journal of Economic Perspectives*, 15(4):143–156, 2001.

[KKHN18] H. M. Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. Neural Network-Based Uncertainty Quantification: A Survey of Methodologies and Applications. *IEEE Access*, 6:36218–36234, 2018.

[KKKF+21] H. M. Dipu Kabir, Abbas Khosravi, Abdollah Kavousi-Fard, Saeid Nahavandi, and Dipti Srinivasan. Optimal uncertainty-guided neural network training. *Applied Soft Computing*, 99:106878, 2021.

[KKNN20] H. M. Dipu Kabir, Abbas Khosravi, Darius Nahavandi, and Saeid Nahavandi. Uncertainty Quantification Neural Network from Similarity and Sensitivity. In *IJCNN 2020*, pages 1–12, Glasgow, 2020.

[KN13] Abbas Khosravi and Saeid Nahavandi. Combined nonparametric prediction intervals for wind power generation. *IEEE Transactions on Sustainable Energy*, 4(4):849–856, 2013.

[KN14] Abbas Khosravi and Saeid Nahavandi. An optimized mean variance estimation method for uncertainty quantification of wind power forecasts. *International Journal of Electrical Power and Energy Systems*, 61:446–454, 2014.

[KNC10] Abbas Khosravi, Saeid Nahavandi, and Doug Creighton. Construction of optimal prediction intervals for load forecasting problems. *IEEE Transactions on Power Systems*, 25(3):1496–1503, 2010.

[KNC13] Abbas Khosravi, Saeid Nahavandi, and Doug Creighton. Prediction intervals for short-term wind farm power generation forecasts. *IEEE Transactions on Sustainable Energy*, 4(3):602–610, 2013.

[KNCA11] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F. Atiya. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE Transactions on Neural Networks*, 22(3):337–346, 2011.

[KPB18] Andreas Kamilaris and Francesc X. Prenafeta-Boldú. Deep Learning in Agriculture: A Survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018.

[KSW15] Diederik P. Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 2575–2583, Montreal, 2015. Curran Associates, Inc.

[KW13] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. Technical report, Universiteit van Amsterdam, 2013.

[LALP19] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2019.

[LJX+19] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, pages 5243—-5253. Curran Associates, Inc., 2019.

[LLP+20] Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7498–7512. Curran Associates, Inc., 2020.

[LNHW17] Bidong Liu, Jakub Nowotarski, Tao Hong, and Rafał Weron. Probabilistic load forecasting via quantile regression averaging of sister forecasts. *IEEE Transactions on Smart Grid*, 8(2):730–737, 2017.

[LSS20] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A General Framework for Uncertainty Estimation in Deep Learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[LW17] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, volume 70, pages 2218–2227, Sydney, 2017. PMLR.

[LZM+20] Chenyu Liu, Xuemin Zhang, Shengwei Mei, Shaowei Huang, and Deming Xia. Optimal Bound Based Ensemble Approach for Probabilistic Wind Power Forecasting. In *IEEE PES GM*, 2020.

[MB07] L. Magnano and J. W. Boland. Generation of synthetic sequences of electricity demand: Application in South Australia. *Energy*, 32(11):2230–2243, 2007.

[MDM18] Tawfek Mahmoud, Z. Y. Dong, and Jin Ma. An advanced approach for optimal wind power generation prediction intervals by using self-adaptive evolutionary extreme learning machine. *Renewable Energy*, 126:254–269, 2018.

[MKvA+21] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. Deep deterministic uncertainty: A simple baseline, 2021.

[MPK+05] Henrik Madsen, Pierre Pinson, George Kariniotakis, Henrik Aa Nielsen, and Torben S. Nielsen. Standardizing the performance evaluation of short-term wind power prediction models. *Wind Engineering*, 29(6):475–489, 2005.

[MYL+14] Zhongxian Men, Eugene Yee, Fue-Sang Lien, Zhiling Yang, and Yongqian Liu. Ensemble Nonlinear Autoregressive Exogenous Artificial Neural Networks for Short-Term Wind Speed and Power Forecasting. *International Scholarly Research Notices*, 2014:1–16, 2014.

[MYL+16] Zhongxian Men, Eugene Yee, Fue Sang Lien, Deyong Wen, and Yongsheng Chen. Short-term wind speed and power forecasting using an ensemble of mixture density neural networks. *Renewable Energy*, 87:203–211, 2016.

[MYLL07] Kittipong Methaprayoon, Chitra Yingvivatanapong, Wei-Jen Lee, and James R. Liao. An Integration of ANN Wind Power Estimation Into Unit Commitment Considering the Forecasting Uncertainty. *IEEE Transactions on Industry Applications*, 43(6):1441–1448, 2007.

[NBD18] Jyotirmayee Naik, Ranjeeta Bisoi, and P. K. Dash. Prediction interval forecasting of wind speed and wind power using modes decomposition based low rank multi-kernel ridge regression. *Renewable Energy*, 129:357–383, 2018.

[NMB17] William B. Nicholson, David S. Matteson, and Jacob Bien. VARX-L: Structured regularization for large vector autoregressions with exogenous variables. *International Journal of Forecasting*, 33(3):627–651, 2017.

[NTS13] Nikolay Nikolaev, Peter Tino, and Evgueni Smirnov. Time-dependent series variance learning with recurrent mixture density networks. *Neurocomputing*, 122:501–512, 2013.

[NW94] David A. Nix and Andreas S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pages 55–60, Orlando, FL, 1994. IEEE.

[OHM+20] Luis Oala, Cosmas Heiß, Jan Macdonald, Maximilian März, Wojciech Samek, and Gitta Kutyniok. Interval neural networks: Uncertainty scores, 2020.

[OLB+18] Aäron Van Den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, George Van Den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, and Dan Belov. Parallel WaveNet : Fast High-Fidelity Speech Synthesis. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80, pages 3918–3926, Stockholm, 2018. PMLR.

[PBL+17] Nick Pawlowski, Andrew Brock, Matthew C. H. Lee, Martin Rajchl, and Ben Glocker. Implicit weight uncertainty in neural networks, 2017.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[PPM17] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 2338–2347, Long Beach, CA, 2017. Curran Associates, Inc.

[PZBN18] Tim Pearce, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *35th International Conference on Machine Learning, ICML 2018*, volume 9, pages 6473–6482, 2018.

[QSK12] Hao Quan, Dipti Srinivasan, and Abbas Khosravi. Construction of neural network-based prediction intervals using particle swarm optimization. *Proceedings of the International Joint Conference on Neural Networks*, pages 10–15, 2012.

[QSK14] Hao Quan, Dipti Srinivasan, and Abbas Khosravi. Short-term load and wind power forecasting using neural network-based prediction intervals. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):303–315, 2014.

[QSK15] Hao Quan, Dipti Srinivasan, and Abbas Khosravi. Incorporating Wind Power Forecast Uncertainties Into Stochastic Unit Commitment Using Neural Network-Based Prediction Intervals. *IEEE Transactions on Neural Networks and Learning Systems*, 26(9):2123–2135, 2015.

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International*

Conference on Machine Learning (ICML 2015), volume 37, pages 1530–1538, Lille, 2015. PMLR.

[RSS+20] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows, 2020.

[SAR+19] Jonathan Sarochar, Ipsita Acharya, Hugo Riggs, Aditya Sundararajan, Longfei Wei, Temitayo Olowu, and Arif I. Sarwat. Synthesizing Energy Consumption Data Using a Mixture Density Network Integrated with Long Short Term Memory. In *IEEE Green Technologies Conference*, volume 2019-April, pages 1–4. IEEE, 2019.

[SAVA19] Cristián Serpell, Ignacio A. Araya, Carlos Valle, and Héctor Allende. Probabilistic Forecasting Using Monte Carlo Dropout Neural Networks. In Ingela Nyström, Yanio Hernández Heredia, and Vladimir Milián Núñez, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 387–397, Havana, 2019. Springer International Publishing.

[SAVA20] Cristián Serpell, Ignacio A. Araya, Carlos Valle, and Héctor Allende. Addressing model uncertainty in probabilistic forecasting using Monte Carlo dropout. *Intelligent Data Analysis*, 24(S1):185–205, 2020.

[SBSC+19] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank Gaussian copula processes. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 2019.

[SFGJ20] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[SH12] George Sideratos and Nikos D. Hatziargyriou. Probabilistic wind power forecasting using radial basis function neural networks. *IEEE Transactions on Power Systems*, 27(4):1788–1796, 2012.

[Sha19] Novin Shahroudi. *Probabilistic Forecasting with Monte-Carlo Dropout in Neural Networks*. PhD thesis, University of Tartu, 2019.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SLD18a] Zhichao Shi, Hao Liang, and Venkata Dinavahi. Direct Interval Forecast of Uncertain Wind Power Based on Recurrent Neural Networks. *IEEE Transactions on Sustainable Energy*, 9(3):1177–1187, 2018.

[SLD18b] Zhichao Shi, Hao Liang, and Venkata Dinavahi. Wavelet neural network based multiobjective interval prediction for short-term wind speed. *IEEE Access*, 6:63352–63365, 2018.

[SLP16] Nitin Anand Shrivastava, Kunal Lohia, and Bijaya Ketan Panigrahi. A multiobjective framework for wind speed prediction interval forecasts. *Renewable Energy*, 87:903–910, 2016.

[SSB+18] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks, 2018.

[SVA23] Cristián Serpell, Carlos Valle, and Héctor Allende. Multi-step probabilistic forecasting model using deep learning parametrized distributions. *Soft Computing*, 2023.

[TAS18] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. *35th International Conference on Machine Learning, ICML 2018*, 11:7824–7833, 2018.

[Tay00] James W. Taylor. A Quantile Regression Approach to Estimating the Distribution of Multiperiod Returns. *Journal of Forecasting*, 19:299–311, 2000.

[TBVD19] Jean Francois Toubeau, Jeremie Bottieau, Francois Vallee, and Zacharie De Greve. Deep Learning-Based Multivariate Probabilistic Forecasting for Short-Term Scheduling in Power Markets. *IEEE Transactions on Power Systems*, 34(2):1203–1215, 2019.

[vASJ+21] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty, 2021.

[vASTG20] Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty Estimation Using a Single Deep Deterministic Neural Network. In *Proceedings of the 37th International Conference on Machine Learning*, Vienna, Austria, 2020.

[VBS+17] Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008, 2017.

[VFM18] Julian Vossen, Baptiste Feron, and Antonello Monti. Probabilistic Forecasting of Household Electrical Load Using Artificial Neural Networks. In *2018 IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–6, 2018.

[WCQL16] Wenzu Wu, Kunjin Chen, Ying Qiao, and Zongxiang Lu. Probabilistic short-term wind power forecasting based on deep neural networks. In *2016 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–8, Beijing, 2016. IEEE.

[WLCC19] Bing Wang, Wei Li, Xianhui Chen, and Haohao Chen. Improved Chicken Swarm Algorithms Based on Chaos Theory and Its Application in Wind Power Interval Prediction. *Mathematical Problems in Engineering*, 2019, 2019.

[WLW+17a] Can Wan, Jin Lin, Jianhui Wang, Yonghua Song, and Zhao Yang Dong. Direct Quantile Regression for Nonparametric Probabilistic Forecasting of Wind Power Generation. *IEEE Transactions on Power Systems*, 32(4):2767–2778, 2017.

[WLW+17b] Huai-zhi Wang, Gang-qiang Li, Gui-bin Wang, Jian-chun Peng, Hui Jiang, and Yi-tao Liu. Deep learning based ensemble approach for probabilistic wind power forecasting. *Applied Energy*, 188:56–70, 2017.

[WNL+19] Jianzhou Wang, Tong Niu, Haiyan Lu, Wendong Yang, and Pei Du. A Novel Framework of Reservoir Computing for Deterministic and Probabilistic Wind Power Forecasting. *IEEE Transactions on Sustainable Energy*, pages 1–1, 2019.

[WSM+19] Yuyang Wang, Alex Smola, Danielle C. Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 11460–11475, 2019.

[WTNM17] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A Multi-Horizon Quantile Recurrent Forecaster. In *31st Conference on Neural Information Processing Systems (NIPS 2017), Time Series Workshop*, Long Beach, CA, 2017.

[WWL+16] Huai-zhi Wang, Gui-bin Wang, Gang-qiang Li, Jian-chun Peng, and Yi-tao Liu. Deep belief network based deterministic and probabilistic wind speed forecasting approach. *Applied Energy*, 182:80–93, 2016.

[WXO+14] Can Wan, Zhao Xu, Jacob Ostergaard, Zhao Yang Dong, and Kit Po Wong. Discussion of 'combined nonparametric prediction intervals for wind power generation'. *IEEE Transactions on Sustainable Energy*, 5(3):1021, 2014.

[WXP13] Can Wan, Zhao Xu, and Pierre Pinson. Direct interval forecasting of wind power. *IEEE Transactions on Power Systems*, 28(4):4877–4878, 2013.

[WXP+14a] Can Wan, Zhao Xu, Pierre Pinson, Zhao Yang Dong, and Kit Po Wong. Optimal prediction intervals of wind power generation. *IEEE Transactions on Power Systems*, 29(3):1166–1174, 2014.

[WXP+14b] Can Wan, Zhao Xu, Pierre Pinson, Zhao Yang Dong, and Kit Po Wong. Probabilistic forecasting of wind power generation using extreme learning machine. *IEEE Transactions on Power Systems*, 29(3):1033–1044, 2014.

[WZC+18] Yi Wang, Ning Zhang, Qixin Chen, Daniel S. Kirschen, Pan Li, and Qing Xia. Data-Driven Probabilistic Net Load Forecasting With High Penetration of Behind-the-Meter PV. *IEEE Transactions on Power Systems*, 33(3):3255–3264, 2018.

[WZT+18] Yi Wang, Ning Zhang, Yushi Tan, Tao Hong, Daniel S. Kirschen, and Chongqing Kang. Combining Probabilistic Load Forecasts. *IEEE Transactions on Smart Grid*, pages 1–10, 2018.

[XH16] Jingrui Xie and Tao Hong. GEFCom2014 probabilistic electric load forecasting: An integrated solution with forecast combination and residual simulation. *International Journal of Forecasting*, 32(3):1012–1016, 2016.

[XHLK17] Jingrui Xie, Tao Hong, Thomas Laing, and Chongqing Kang. On Normality Assumption in Residual Simulation for Probabilistic Load Forecasting. *IEEE Transactions on Smart Grid*, 8(3):1046–1053, 2017.

[YCZ21] Shi Yu, Yuxin Chen, and Hussain Zaidi. Ava: A financial service chatbot based on deep bidirectional transformers. *Frontiers in Applied Mathematics and Statistics*, 7, 2021.

[ZBKM19] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in Variational Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2019.

[ZL16] Florian Ziel and Bidong Liu. Lasso estimation for GEFCom2014 probabilistic electric load forecasting. *International Journal of Forecasting*, 32(3):1029–1037, 2016.

[ZL17] Lingxue Zhu and Nikolay Laptev. Deep and Confident Prediction for Time Series at Uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110, New Orleans, LA, 2017.

[ZLY+20] Hao Zhang, Yongqian Liu, Jie Yan, Shuang Han, Li Li, and Quan Long. Improved Deep Mixture Density Network for Regional Wind Power Probabilistic Forecasting. *IEEE Transactions on Power Systems*, 35(4):2549–2560, 2020.

[ZQG+20] Wenjie Zhang, Hao Quan, Oktoviano Gandhi, Ram Rajagopal, Chin-Woo Tan, and Dipti Srinivasan. Improving Probabilistic Load Forecasting using Quantile Regression NN with Skip Connections. *IEEE Transactions on Smart Grid*, 3053(c):1–8, 2020.

[ZQS19] Wenjie Zhang, Hao Quan, and Dipti Srinivasan. An Improved Quantile Regression Neural Network for Probabilistic Load Forecasting. *IEEE Transactions on Smart Grid*, 10(4):4425–4434, 2019.

[ZWW+15] Guoyong Zhang, Yonggang Wu, Kit Po Wong, Zhao Xu, Zhao Yang Dong, and Herbert Ho Ching Iu. An Advanced Approach for Construction of Optimal Wind Power Prediction Intervals. *IEEE Transactions on Power Systems*, 30(5):2706–2715, 2015.

[ZYI+19] Jinhua Zhang, Jie Yan, David Infield, Yongqian Liu, and Fue-sang Lien. Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and Gaussian mixture model. *Applied Energy*, 241(January):229–244, 2019.