LIÈGE université

FACULTÉ DES SCIENCES APPLIQUÉES
Ecole d'ingénieurs et d'informaticiens

Thesis submitted in partial fulfilment of the requirements for the Degree of Master of Science (MSc) in Data Science and Engineering

Probabilistic load forecasting with generative models

Benjamin Delvoye
Advisor : Bertrand Cornélusse
Co-advisor : Jonathan Dumas

University of Liège
-
Faculty of Applied Sciences

Academic years 2020-2021

Abstract

Electric load forecasting is a central step for economic actors to plan supply purchases and for energy system operators to control networks. The marginal gain in forecast accuracy potentially leads to important resources saving for power companies involved. With this in mind, researchers in the field of power system shifted from point forecast to probabilistic forecast in order to produce more informative and reliable predictions, then to characterize uncertainty tied to the forecast. In turn, this work explores the introduction of information from an electrical grid topology into the probabilistic forecast framework, hoping to further improve load forecasters. Building upon normalizing flows, we apply graphical normalizing flows to the field of power system. Normalizing flows are a class of generative models that create a mapping between the distribution of interest and a known distribution. Graphical normalizing flows allow to add inductive bias by taking into account prescribed dependencies between random variables from the targeted multivariate distribution. In this particular study, we are interested in the forecast of day-ahead electric load from different connected zones. Experiments are conducted on a actual case study, hourly aggregated electric load collected over 14 years in all 8 states of New England, USA. After analysing the data, we investigate the performance of graphical normalizing flows when proposing various topology for the electrical network. We finally compare the results to other generative models, namely autoregressive normalizing flow and variational autoencoder. We observe tuned and trained graphical normalizing flow forecaster achieves slightly better result on various metrics than all other models globally. We show that GNF with well-identified independence introduces an inductive bias sufficient to improve model distributions and scenarios generated.

Contents

Note

This note sums up modifications brought to the master thesis document since last submission in January 2022. Overall, the whole document is reorganized. Each existing chapter (namely *Introduction*, *Related works*, *Case study*, *Experiments* and *Conclusion*) get deeply revised and improved. *Problem statement* and *Generative models* chapters are added to introduce necessary concepts related to the work. We complete the chapter *Related works* with recent methods and approaches from deep generative models. In the *Experiments* chapter, we integrate an hyperparameters search (HS) section and discuss the choice of heuristic for adjacency matrix of GNF. We define more scores, with pros and cons, and provide an extended discussion about results obtained with final models from HS. The *Conclusion* chapter summarizes the entire document and provide possible future leads to deepen the subject.

In more details:

- The hierarchical approach of zone has been fully dropped to concentrate on inter zone dependencies.
- In the introduction, we present two views of the followed workflow.
- Settings of the case study are clearer and a better analysis is provided about the data.
- Continuous rank probability score is fully detailed along with energy score and variogram score metrics.
- An hyperparameters search is conducted on optimizer parameters and for important hyperparameters of each models. In particular we introduce procedures to build heuristic of the adjacency matrix of GNF and discuss their relevance.
- We provide results through visual inspection of scenarios and numeric scores. All scores are discussed and models scores are compared against each other.

Acknowledgements

I would like to thank my advisor Pr. Bertrand Cornélusse for introducing me to the very interesting field of smart micro-grid and welcoming me within his team to conduct my master thesis. I am also deeply grateful to engD. Jonathan Dumas for its support and guidance throughout the extended period of time within which I wrote this document.

I wish to give a special appreciation to engr. Antoine Wehenkel, as an engineer and as a friend, that provided me precious feedback and advises while restructuring the final version of this work.

Finally, I would like to thank my family, whom never failed me, my friends and my girlfriend, Sara, whom inspires me to keep pushing my limits further.

Chapter 1

Introduction

Electric load demand forecasting is a crucial challenge for electrical grid operation, economic and political actors. Indeed its application ranges from energy purchasing, transmission and distribution, policymaking process, and much more [Hong, 2010] In particular, short term load forecast, such as day-ahead forecast, tackle problems linked to unit commitment, generation scheduling function or even security of power systems [Gross and Galiana, 1987]. In all of these applications, having access to accurate prediction hugely impacts companies involved in the sector. According to analysis on important electrical utilities, 1% in reduction in the average forecast error can save hundreds of thousands or even millions of dollars. [Hobbs et al., 1999, Alfares and Nazeeruddin, 2002].

Despite load forecasting playing a major role in the electrical industry for over a century [Hong, 2014], many challenges remain to be tackled. One of them is to properly characterize the uncertainty tied to electricity load forecast. Indeed, electricity load demand of a region being driven by many exogenous factors [Sangrody and Zhou, 2016, Santos et al., 2007], such as weather, economy and demography, it contains an important part of uncertainty. Up until recently, the electricity load forecast was deterministic, single valued predictions, and thus missing information about uncertainty of events [Gneiting and Katzfuss, 2014]. Instead, probabilistic forecast model the whole range of possible outcomes allowing for decision-makers to build better knowledge about demand behaviors. As described thoroughly in 2, several approaches exist to define uncertainty of forecast trough probabilistic methods: quantiles, error bars, intervals or full density functions, each of them having strengths and drawbacks.
Moreover, new opportunities arise from the quantity and quality of data collected over the years. During the last decade, a massive deployment of measurement devices took place at all aggregation levels in electrical networks [Gajowniczek and Ząbkowski, 2014]. Thus load forecasting field needs innovative approaches to apprehend and make sense of those database. Generative models are promising methods to crunch this new amount of data and synthesize efficiently underlying dynamics of load demand. Generative models learn to reduce the input data into a latent space from which it is expected to reconstruct new and unseen samples, close to the input data distribution. As such they are candidate of choice for scenario based forecast and density function estimators.

In this work, we propose to apply several generative models to forecast day-ahead electricity load demand in the region of New England, USA. The load demand consists in hourly timeseries aggregated at state level, providing information about eight related zones. The study particularly focuses on normalizing flows (NFs), a class of generative models. NFs create a mapping between the distribution of interest and a known and tractable distribution. They perform this mapping trough simple and invertible functions, constrained to allow fast computation of the change

of variable. We introduce the use of graphical normalizing flows (GNFs) in power system [Wehenkel and Louppe, 2021]. GNFs embed relation of topology from the grid and exploits it while issuing probabilistic forecasts. Indeed, as Wehenkel and Louppe [2020] demonstrated, some subclasses of normalizing flows naturally represent specific Bayesian Network (BN). BN models expert knowledge of independence between random variables through an adjacency matrix. By masking the input vector according to the adjacency matrix, GNF tackles the same problems as the corresponding BN with the advantage of learning the distributions from data. In this work, each node of the BN represents the aggregated electrical load demand of a zone at a given time of the day. Thus, edges of the BN model the spatio-temporal dependence between nodes. Formally, GNF adjacency matrix introduces an inductive bias on the dependence structure. Inductive biases are the set of hypothesises about the data that are induced through a specific model architecture. Alet et al. [2021] reminds the importance of inductive biases in neural networks and breakthroughs they allowed recently. As tool of comparison for normalizing flows, variational autoencoders (VAEs) are applied to the case study. VAEs are also generative model but compress the information into latent variables instead of explicitly transforming the density function [Kingma and Welling, 2019].
Given this introduction, we highlight research gaps we aim to fill as

- the thorough assessing of multi zonal load forecasting under the probabilistic framework.
- the exploration of dependence inductive bias in power system application.

We sum up the core of this work as the following:

- We review the concept of uncertainty representation and verify the pertinence of generative model in the frame of load forecasting.
- We provide deep analysis of the ISONE dataset.
- We introduce graphical normalizing flow in power system, allowing to add inductive bias about spatial dependence. We offer visual and quantitative comparisons with autoregressive normalizing flow and variational autoencoder based on results of experiments lead on real case study.

Following this introduction, Chapter 2 lay the base of the problem and defines a framework to tackle probabilistic forecasting. Chapter 3 reminds general goal of generative models then showcases models used in this work. The case study and data related to it are presented in Chapter 5. Chapter 6 is divided into 3 sections. The first introduce metrics proposed to assess forecasters. The second review the hyper-parameters search and loss scores associated to the grid search. In the third, we discuss results obtained on tuned forecasters of each model. Finally, Chapter 7 summarises the work and remind the main findings of the work. We end it by giving some hints about future work directions.

Figure 1.1 illustrates bottom-up workflow of the study core. Analysing the problem and generative models define objectives and narrow the scope of the research area. Related works inform about the state of targeted fields. They also provide intuitions about how to handle data and/or models. Settings introduce dataset exploited and its high level structure while exploratory data analysis (EDA) deepens the understanding of the raw data. Those parts indicates interesting data processing and problem statement defines data structure needed for our forecast framework. This data structure and its corresponding values characterize which metrics are required to measure models performances. Hyperparameters search is performed mainly based on loss scores the tested forecasters. We pick final models from HS and evaluate

them with metrics defined earlier. EDA and results are interdependent as a data analysis might lead to result interpretation as well as results might call for further EDA.

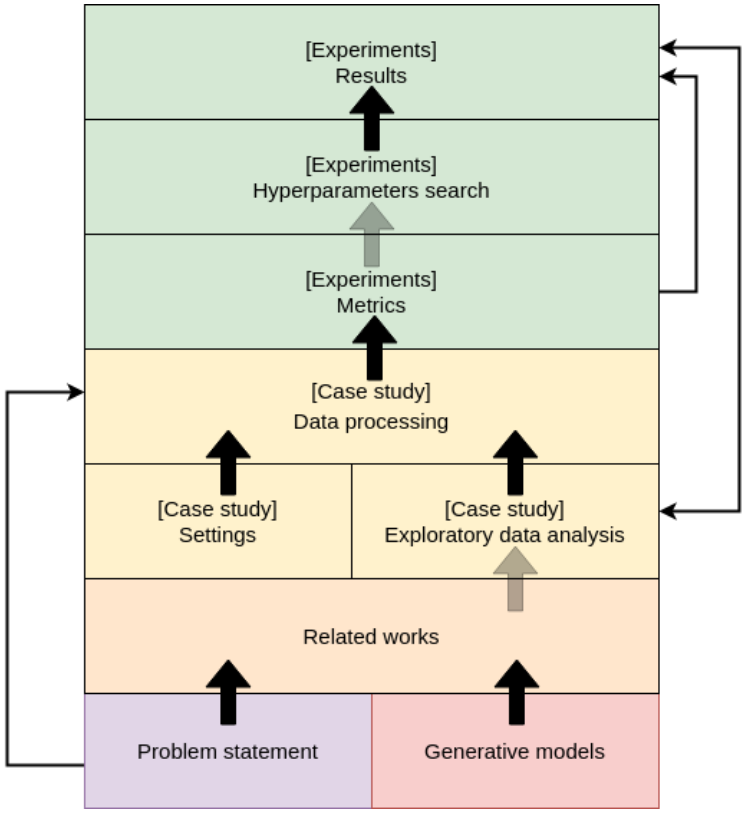


Figure 1.1: Illustration of the workflow applied to perform this study. Arrows indicate existing relation between different part of the work.

Focusing on green sections from Figure 1.1, Figure 1.2 represents the methodology applied to the Experiments chapter. Main algorithms are summarized, respective hyperparameters tuning are presented and metrics are noted. Division of the dataset for each step is also detailed.

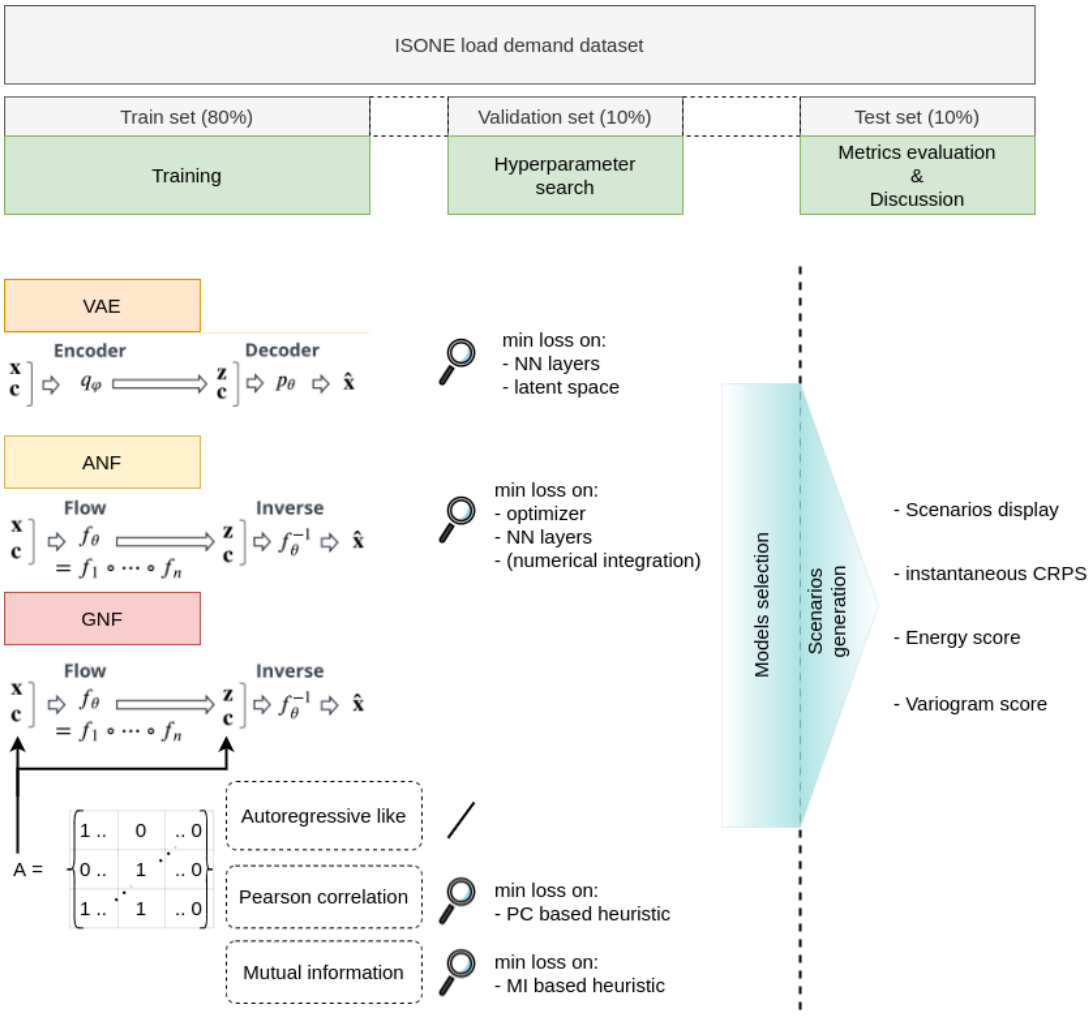


Figure 1.2: Summary of the "experiments" section methodology.

Chapter 2

Problem statement

This chapter develops concepts of load forecasting and probabilistic forecasting with related notion of uncertainty. Finally we propose a framework adapted to probabilistic load forecasting in a multizonal setting.

1 Probabilistic forecasting

In the early nineties, scientific disciplines making use of forecast progressively shifted from single-valued predictions to distributional or probabilistic forecast, following the trend of value estimation [Palmer et al., 1993, Gneiting and Katzfuss, 2014]. Nowadays, forecast practitioners agree that getting insight about an uncertain future ought to be probabilistic in nature. Either taking the form of probability distributions over future quantities or events [Gneiting, 2008] or at least through some statistics summarizing it [Morales et al., 2014], uncertainty must be quantified and communicated. Probabilistic forecast leads to great improvement in decision-making when measured on value scores [Mylne, 2002]. Indeed under probabilistic paradigm, deciders are able to adjust the level of risk acceptable to its application and yield higher expected return than with point value forecast [Taghavifard et al., 2009].

Loosely following Morales et al. [2014] notations, let us define a forecaster as a model g, aiming to produce an estimate, $\hat{e}_{t+k|\mathbf{\Omega}}$, from the stochastic process of interest $\mathbf{Y}$. The information set, $\mathbf{\Omega}$, gathers all data available to g up to time t, including past observations of $\mathbf{Y}$. The estimate is thus generated conditionally on $\mathbf{\Omega}$. k represents the lead time of the estimate. Forecasts are issued as series of consecutive values $\hat{\cdot}_{t+k|\mathbf{\Omega}}$, $k = 1, 2, \ldots, K$ for regularly spaced lead times, referred as the time resolution, up to the forecast horizon K.
The model g can be formulated either in the deterministic or in the stochastic process framework. Various methods exists to characterize the uncertainty inherent to forecast. This section successively defines mathematically some of those approaches along with their pros and cons.

1.1 Enriched deterministic forecast

A naive approach to characterize uncertainty is to decompose the forecast into two parts: one which is the deterministic forecast and the other that specify the residual error, ϵ, made by the model (see Figure. 2.1). By formulating an assumption on the distribution of the error, e.g. normal distribution, we may inform of the uncertainty through some statistics. With normal distribution, the uncertainty can be represented with the standard deviation learned from the

residual error of the model. From now, probabilistic models are denoted as g and deterministic ones as g'.

$$g(k, \mathbf{\Omega}) = g'(k, \mathbf{\Omega}) + \epsilon$$
$$g(k, \mathbf{\Omega}) = g'(k, \mathbf{\Omega}) + \mathcal{N}(0, \sigma^2)$$

Even though it results in a intuitive and simple representation of uncertainty, this method is very limited in term of information and potentially misleading if hypothesis about the distribution of the error is incorrect.

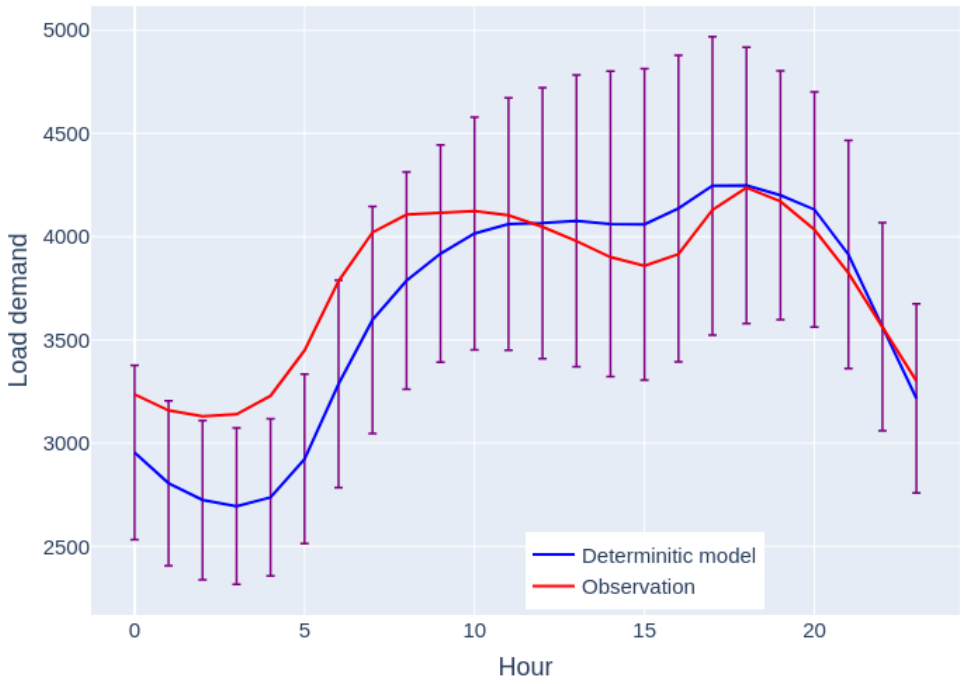


Figure 2.1: Illustration of error bars approach to uncertainty characterization. The blue line indicates the deterministic model result. Purple vertical bars show the uncertainty tied to the model under the form of standard deviation of errors from the deterministic model.

1.2 Interval forecast

A major flaw of the previous method was to rely on the knowledge of the error distribution to model the uncertainty. Interval forecast relaxes this hypothesis and makes use of quantile predictions. Based on the definition of the quantile of a cumulative distribution function, a quantile forecast $\hat{q}^{(\alpha)}_{t+k|\Omega}$ with a nominal level α is an estimate, issued at time t for lead time $t + k$, of the quantile $q^{(\alpha)}_{t+k}$ for the random variable Y_{t+k}, given a model g and the information set Ω., i.e.,

$$P[Y_{t+k} \leq \hat{q}^{(\alpha)}_{t+k|\Omega}] = \alpha$$

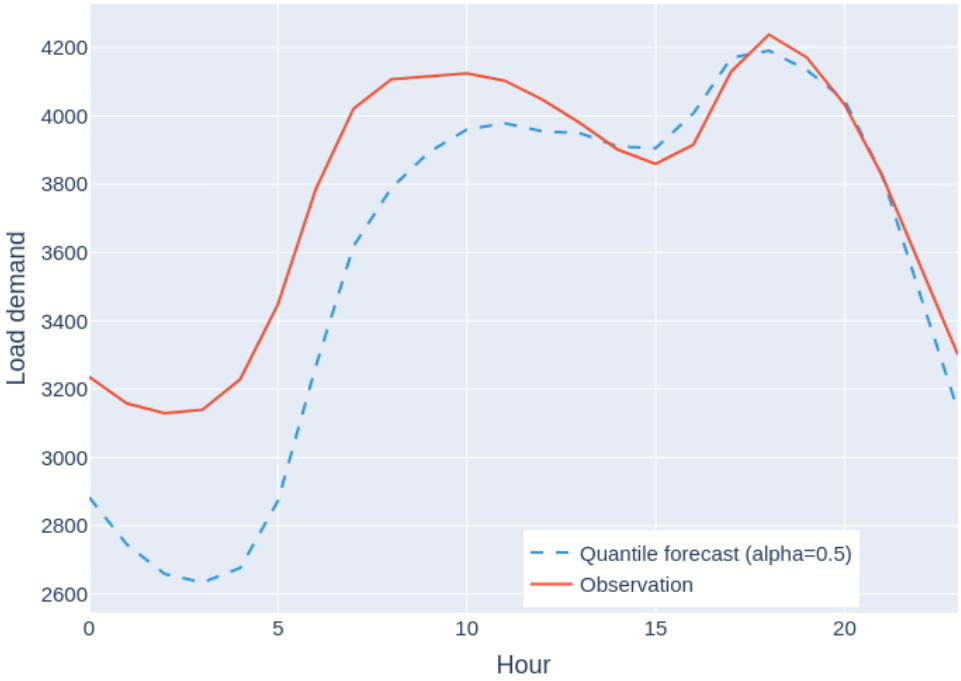


Figure 2.2: Quantile forecast of the aggregated electrical load demand in Cunnecticut, USA. A single day is observed and a forecast is issued at each hour of the day. The nominal level of the quantile forecast displayed reaches $\alpha = 0.5$.

Build upon quantile forecast, interval forecast provide a much better insight in term of uncertainty. A interval forecast $\hat{I}^{(\beta)}_{t+k|\Omega}$, issued at time t for time $t+k$, defines a range of potential values for Y_{t+k}, for a certain level of probability $(1-\beta)$, $\beta \in [0,1]$, its nominal coverage rate ,

$$P[Y_{t+k} \in \hat{I}^{(\beta)}_{t+k|\Omega}] = 1 - \beta$$

The interval $\hat{I}^{(\beta)}_{t+k|\Omega}$ must be defined by its lower and upper bounds, along with its centering.

$$\hat{I}^{(\beta)}_{t+k|\Omega} = [\hat{q}^{(\underline{\alpha})}_{t+k|\Omega},\ \hat{q}^{(\overline{\alpha})}_{t+k|\Omega}]$$

where these bounds are quantile forecast whose nominal levels $\underline{\alpha}$ and $\overline{\alpha}$ verify that $\overline{\alpha} - \underline{\alpha} = 1 - \beta$. Commonly, the interval is centered on the median such as $\underline{\alpha} = 1 - \overline{\alpha} = \beta/2$.

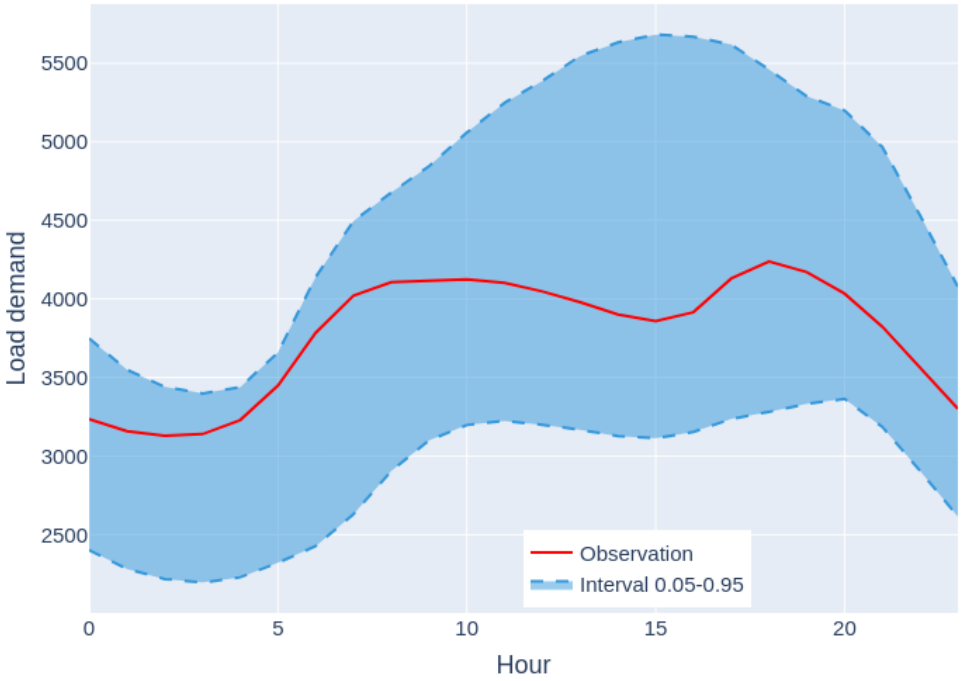


Figure 2.3: Interval forecast of the aggregated electrical load demand in Cunnecticut, USA. A single day is observed and a forecast is issued at each hour of the day. The nominal coverage rate of the interval forecast displayed reaches $\beta = 0.1$. It is build from the respective lower quantile with $\alpha = 0.05$ and upper quantile with $\alpha = 0.95$.

Through interval forecast, stakeholders are equipped to resonate more adequately to risk under uncertainty, having access to a range of possible outcomes associated to a confidence level. Yet the information about uncertainty is not fully complete as interval forecasts only inform the reader about a given coverage rate.

1.3 Density forecast

Density forecast give the whole information about each point of time in the future. A density forecast, $\hat{f}_{t+k|\Omega}$ (or $\hat{F}_{t+k|\Omega}$), issued at time t for time $t+k$ is a complete description of the probability density function (pdf), or of the cumulative distribution function (cdf), of Y_{t+k} given a model g and the information set, Ω.

Interval forecast can still be leveraged to create density forecast by overlaying multiple central prediction intervals with different coverage rates, $(1 - \beta \in 0.1,)$. Other methods allow to offer directly the whole estimated distribution.

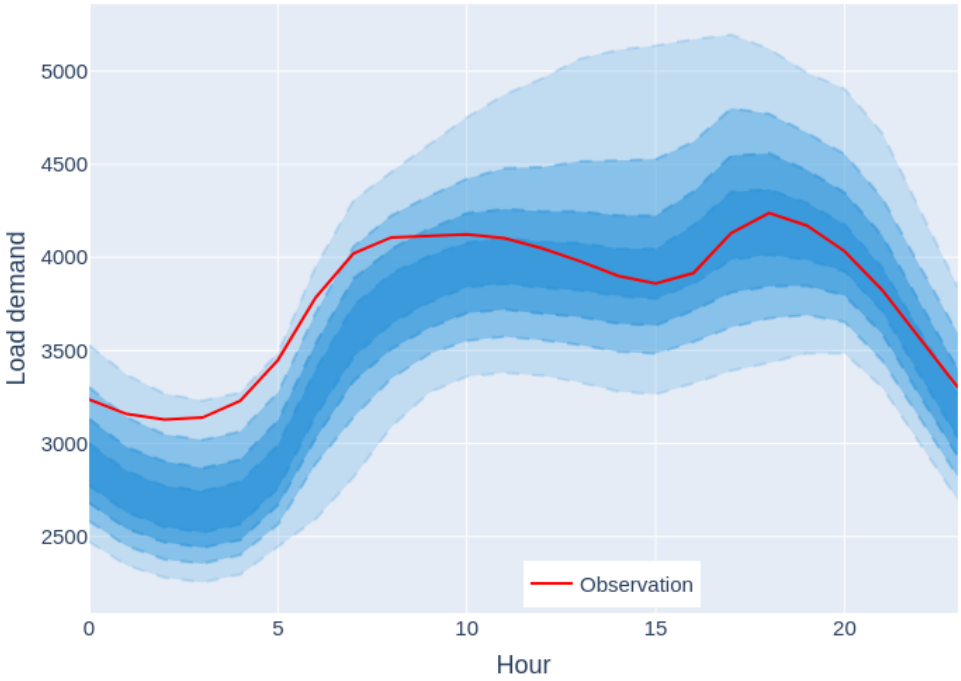


Figure 2.4: Density forecast of the aggregated electrical load demand in Cunnecticut, USA. A single day is observed and a forecast is issued at each hour of the day. The nominal coverage rates of the interval forecast displayed ranges from $\beta = 0.1$ to $\beta = 0.9$. Interval forecast are overlapped to create the density forecast.

Gneiting and Katzfuss [2014] resumes key characteristics of probabilistic forecast under the density forecast paradigm. According to Murphy and Winkler [1987], Gneiting et al. [2007], probabilistic forecasting should aim to maximize the sharpness of the predictive distributions, subject to calibration.

Calibration and dispersion

Calibration and dispersion concern statistical compatibility between probabilistic forecast, $\hat{F}_{t+k}$ and corresponding realizations, Y_{t+k}. Let's introduce the *probability integral tranform* (PIT) of a forecast cdf, $\hat{F}$, as the random variable $Z_F = F(Y)$. If F is continuous and $Y \sim F$, then Z_F is standard uniform. The PIT is the value that the predictive CDF attains at the observation. Then

1. The forecast F is marginally calibrated if $\mathbb{E}[F(y)] = P(Y \leq y),\ \forall y \in \mathbb{R}$.
2. The forecast F is probabilistically calibrated if its PIT Z_F has a standard uniform distribution.
3. The forecast F is overdispersed if $var(Z_F) \leq \frac{1}{12}$ and underdispersed if $var \geq \frac{1}{12}$.

A forecast that is ideal relative to some information set, Ω, is both marginally and probabilistically calibrated.

Sharpness

Sharpness concerns the concentration of the predictive distributions. A forecast distribution is expected to be as narrow as possible, as long as it is calibrated. Sharpness is extrinsic to the observations and are exclusively related to the forecast.

1.4 Scenarios forecast

Even if density forecast fully transmits the information about uncertainty, its representation is only able to convey marginal distribution of each lead time k from Y_{t+k}, or from any other variables jointly forecast. And so density forecast display no information about dependencies among variable and their tied uncertainty. Yet in term of realization, e.g. when forecasting timeseries, it is expected that consecutive lead time conserve their temporal dependence instead of being independently drawn from their marginal distribution. However communicating efficiently a full multivariate distribution is near impossible. To overcome this issue, forecast should also be delivered as trajectories. The following definition focuses on a single timeserie, but will be extended in the next section to fulfil the needed framework.

Defining the multivariate random variable $\mathbf{Z}_t = Y_{t+k},\ k = 1, \ldots, K$ and its multivariate cdf F_{Z_t}, scenarios issued at time t and for a set of K successive lead times, i.e., with $k \in 1, 2, \ldots, K$, are samples of $\hat{F}_{Z_t}$, namely the predicted cdf of Z_t. They consist in a set of J time trajectories,

$$\hat{z}_t^{(j)} = [\hat{y}_{t+1|\Omega}^{(j)}, \hat{y}_{t+2|\Omega}^{(j)}, \ldots, \hat{y}_{t+K|\Omega}^{(j)}],\ j \in 1, \ldots, J$$

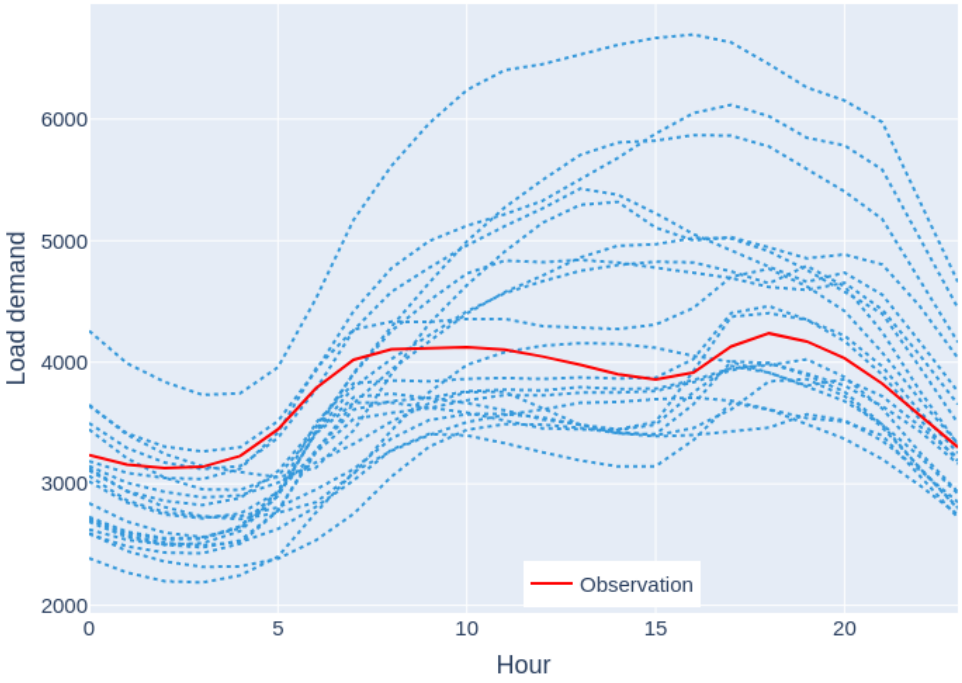


Figure 2.5: Scenarios forecast of the aggregated electrical load demand in Cunnecticut, USA. A single day is observed and multiple scenarios are issued at each hour of the day. Scenarios preserve correlation information about the studied process which can not be captured with quantile methods for uncertainties.

2 Probabilistic load forecasting

Based on the definition of probabilistic forecast, we specify *probabilistic load forecasting* in the frame of this work. We hope to jointly forecast the probabilistic load demand of k successive lead time at time t for N zones of a connected network, given the information set Ω. Now, the multivariate random variable $\mathbf{Z}_t$ is composed of $K \times N$ elements

$$\mathbf{Z}_t = Y_{t+k,\,n},\ k = 1, \ldots, K \text{ and } n = 1, \ldots, N$$

Trajectories can be sampled from $\hat{F_{\mathbf{Z}_t}}$, the estimated cdf of $\mathbf{Z}_t$, creating scenarios for connected timeseries, $\hat{z}^{(j)}$.

$$\hat{z}_t^{(j)} = \begin{bmatrix} \hat{y}^{(j)}_{t+1,1|\Omega} & \cdots & \hat{y}^{(j)}_{t+K,1|\Omega} \\ \hat{y}^{(j)}_{t+1,2|\Omega} & \cdots & \hat{y}^{(j)}_{t+K,2|\Omega} \\ \cdots & \cdots & \cdots \\ \hat{y}^{(j)}_{t+1,N|\Omega} & \cdots & \hat{y}^{(j)}_{t+K,N|\Omega} \end{bmatrix}$$

Chapter 3

Generative models

In this chapter, we review generative models and how we leverage it to produce probabilistic load forecast of load demand. Subsequently, we detail normalizing flows and variational autoencoders exploited in this work.

Generative modeling is a machine learning paradigm addressing the problem of generating unseen data sample similar to a targeted dataset. In term of learning methods, supervised learning aims at creating a mapping between a set of inputs x to an output y, given a labeled set of input-output pairs, while unsupervised learning focuses on extracting or synthesizing patterns from data of interest [Murphy, 2012]. Unsupervised learning is thus more appropriate to build generative model, as it offers the opportunity to draw latent information and reconstruct plausible data close to the original data distribution. Among the applications of generative modeling, we find

- **Parameter estimation**: to estimate parameters of a system, simulation based inference takes advantage of generative models to estimate the likelihood function of the parameters given observations [Cranmer et al., 2020, Green et al., 2020].
- **Image synthesis**: based on dataset of real world images, the model generates real looking pictures of faces, objects or scenes [Radford et al., 2015].
- **Data completion**: estimation of missing data in images or series can be recovered through generative models. It learns the structure of the joint distribution then draws missing information conditioned on context at run time [Yeh et al., 2017].

More formally, given a dataset D of i.i.d. examples $\{\mathbf{y}_i\}^n$, from a data distribution $p(y)$, a generative model describes an estimated distribution of the targeted data $\hat{p}_\theta(y)$. From this description, a generative model is expected to be able to produce in turn new and useen data points, $\hat{y}$, belonging to the initial distribution as presented on Figure 3.1 [Karpathy and Abbeel].

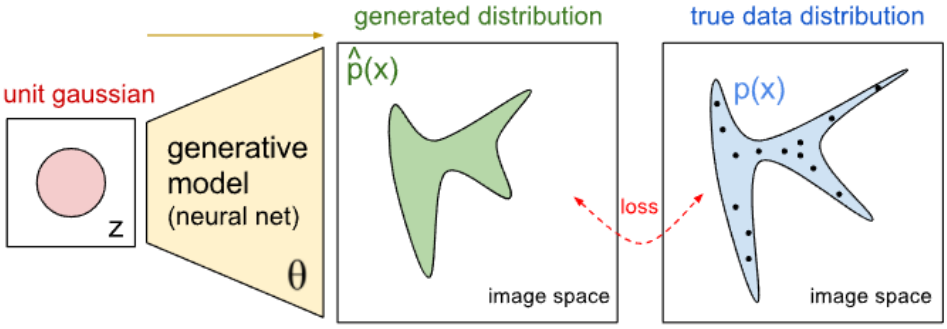


Figure 3.1: A generative model parametrized with θ maps a known distribution, here unit gaussian, to a modified probabilistic space estimating the targeted distribution, $\hat{p}_\theta(y)$. The model is usually trained by minimizing the difference between $p(y)$ and $\hat{p}_\theta(y)$.

Through this introduction of generative modeling we seek to present generative models as good fit to probabilistic forecast. Indeed they model implicitly the distribution of the random variable $Z_t = Y_{t+k}$, $k \in 1, \dots, K$, loads of interest at an horizon of K, by extracting relevant information in past observations. They can then generate plausible scenarios and informs of the uncertainty tied to future events.

1 Bayesian networks

We introduce Bayesian networks (BNs) as example for generative models and as a basis for methods presented further. Bayesian networks are probabilistic graphical models (PGMs) acknowledging relationships of independence within a set of variables.

More formally, a BN is a directed acyclic graph (DAG) encoding the independence between variables of a random vector. Given a random vector $\mathbf{x} = [x_1, \dots, x_d]^T \in \mathbb{R}^d$ distributed under p_x, a BN associated to $\mathbf{x}$ is a DAG with d vertices (nodes), being the elements of $\mathbf{x}$, and m edges. The absence of edge highlights conditional independence between groups of components. A BN is a valid representation of a random vector $\mathbf{x}$ iff it factorizes its density as

$$p_{\mathbf{x}}(\mathbf{x}) = \prod_{i=1}^{d} p(x_i|\mathcal{P}_i)$$

where $\mathcal{P}_i = \{j : A_{i,j}\} = 1$ denotes the set of parent of node i and $A \in \{0,1\}^d$ is the adjacency matrix of the BN (see Figure 3.2).

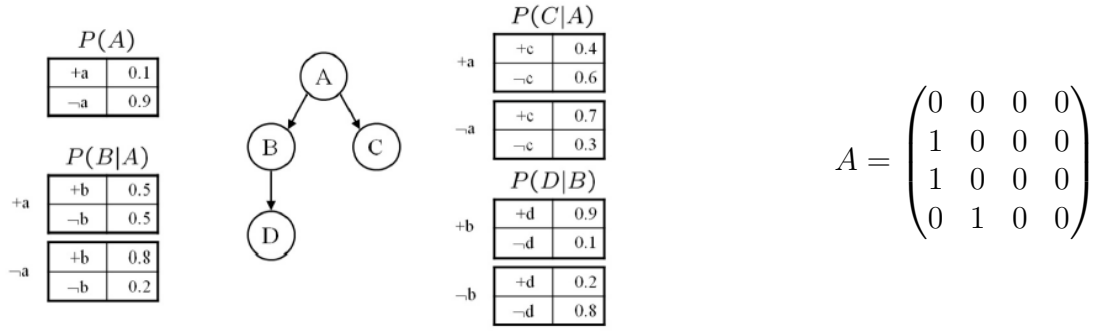


Figure 3.2: Generic schema of a Bayesian network. Variables of this BN encode binary event but BN easily extends to continous variables through *pdf* informations. A is the corresponding adjacency matrix with row and columns going from variable A to D (top - down, left - right).

Given the conditional distributions of variables and independencies required, we easily cast BN as a generative model. Indeed by sampling in the ancestors variables and propagating it in the BN, we now generate potentially unseen data at the DAG end [Doan et al., 2012]. BN is an intuitive and interpretable framework that can be integrated to learning algorithms when dealing with PGM and expert knowledge.

2 Normalizing flows

Normalizing flows were initially proposed by Tabak and Vanden-Eijnden [2010] for density estimation of observed samples. Kobyzev et al. [2020] and Papamakarios et al. [2021] perform comprehensive reviews of normalizing flows. All type of normalizing flows are constructed on the same flow-based model idea. Let $Y \in \mathbb{R}^d$ be a realization of random variables vector of interest and $Z \in \mathbb{R}^d$ be a realization of random variables vector sampled from a known and tractable distribution, p_z. Then normalizing flows are defined as sequence of k invertible transformations $\mathbf{f} := \mathbf{f}_1 \circ ... \circ \mathbf{f}_k$ from $\mathcal{Y}$ to $\mathcal{Z}$ (and thus also from $\mathcal{Z}$ to $\mathcal{Y}$ when using $\mathbf{f^{-1}}$). By the change of variable formula, $\mathbf{f}$ implicitly defines the probability density of $\mathcal{Y}$ as,

$$p(\mathbf{y}) = p_z(\mathbf{z})|J_{\mathbf{f(y)}}|$$
$$p(\mathbf{y}) = p_z(\mathbf{f}(\mathbf{y}))|J_{\mathbf{f(y)}}|$$

where $|J_{\mathbf{f}(y)}|$ is the determinant of the Jacobian of $\mathbf{f}$ in regard to $\mathbf{y}$. In the frame of density estimation, learnable parameters θ are introduced in $\mathbf{f}$ to let the function fit a suitable transformation to the data. The models are then trained by maximizing the likelihood of the parameters given a training dataset.

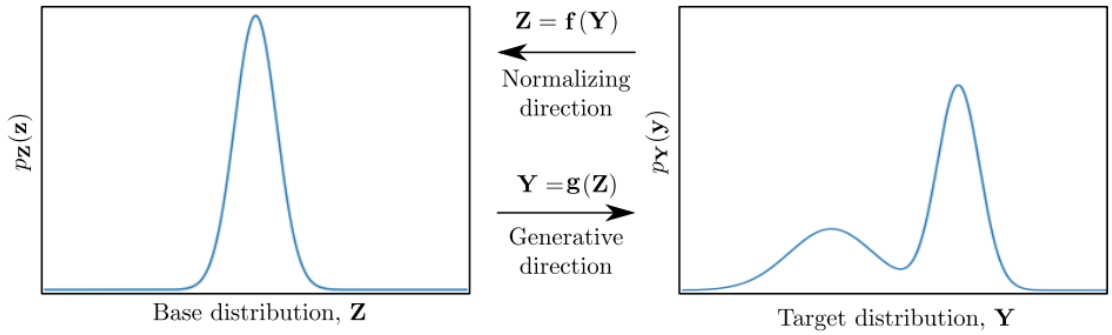


Figure 3.3: Change of variable representation. $\mathbf{f}$ pushes the target distribution $\mathcal{Y}$ to a known tractable distribution $\mathcal{Z}$. As $\mathbf{f}$ is defined as invertible, its inverse function, $\mathbf{f}^{-1}$ (or $\mathbf{g}$ to denote the *generative function*) can be exploited to generate new samples.[Kobyzev et al., 2020]

Normalizing flows appear in a broad range of flavor. Among others, we find planar or radial flow, distorting the distribution either in specific direction (planar) or around a point (radial) [Rezende and Mohamed, 2015]. In order to facilitate the practical setting of normalizing flows, most approaches look to break down its architecture into two main parts. The two problems to tackle in NFs are respectively to define an invertible function $\mathbf{f}$ then to compute efficiently its Jacobian determinant in regard to the input, $\mathbf{y}$, while preserving dependencies between variables. To do so, in one hand, the *normalizer* defines a set of invertible scalar functions, $f_1, \ldots, f_d$, each depending explicitly only on their corresponding entry, y_i, $i = 1, \ldots, d$. In the other hand, the *conditioner* role is to parametrize the *normalizer* through part of remaining variables while constraining the Jacobian to be at least lower triangular. This trick makes possible to compute the Jacobian determinant in $\mathcal{O}(d)$ instead of $\mathcal{O}(d^3)$. Plus, this format reduces the invertibility of each function to $\mathbb{R} \leftrightarrow \mathbb{R}$, as the conditioner do not need to be inverted at sampling time. Thus we write,

$$\mathbf{f}(\mathbf{y}) = [f^1(y_1; \mathbf{c}^1(\mathbf{y})), \ldots, f^d(y_d; \mathbf{c}^d(\mathbf{y}))]$$

where $\mathbf{c}^i$ are the conditioners. When designing a normalizing flow, it is important to verify the *normalizer* is monotonic to ensure the invertibility of the scalar functions. Indeed, as the Jacobian determinant indicates the local deformation of the base distribution in the probability space, it is prohibited to reach a value of zero. That case would represent an singularity of its deformation, making it non invertible.

Concerning the normalizer part, we choose to experiment with two different modules. First, affine normalizers [Dinh et al., 2014, Kingma and Dhariwal, 2018, Papamakarios et al., 2017] are transformations $\mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$ of the distribution defined by

$$f^i(y_i; \boldsymbol{c}^i) = y_i e^{\theta_1} + \theta_2$$

with θ_1, θ_2 as parameters computed by the conditioner $\mathbf{c}^i$. Affine normalizers are computationally efficient but lack expressiveness. To overcome this problem, multiple steps and permutations of NF are combined by piling up normalizer - conditioner functions. The second type are named monotonic normalizers. Huang et al. [2018] state that any neural network (NN) with positive weights and strictly monotone activation functions is itself strictly monotone. This type of NN is thus invertible and a valid normalizer. However training might reveal to be slower due to the constraints on the NN form. Wehenkel and Louppe [2019] propose a free form NN, noted F, with only restriction to remain strictly positive, called *unconstrained*

monotonic neural networks (UMNNs). Its integral is then assured to be monotonic and is expressed as,

$$f^i(y_i, \mathbf{c}^i) = \int_0^{y_i} F^i(t, \mathbf{c}^i)dt + \beta(\mathbf{c}^i)$$

where $\mathbf{c}^i \in \mathbb{R}^{|\mathbf{c}|}$. Also $F : \mathbb{R}^{|\mathbf{c}|+1} \to \mathbb{R}^+$ and $\beta : \mathbb{R}^d \to \mathbb{R}$ are two NN.

About conditioners, we focus in this work on a subclass of NFs using coupling function, which are the most widely used in flow-based architecture. They are introduced in the following section.

2.1 Autoregressive normalizing flows

As precised earlier, one role of the conditioner is to reduce the complexity of the Jacobian determinant computation. With autoregressive conditioners, $\mathbf{c}^i(\mathbf{y}) = \mathbf{h}^i([y_1, \ldots, y_{i-1}])$. Meaning the conditioner c_i is a function only depending on the $i-1$ first entries of the random vector, $\mathbf{y}$. This creates a lower triangular Jacobian which determinant is the product of its diagonal elements. For the implementation used in this work, Papamakarios et al. [2017] enforces the autoregressive structure on the autoencoder used to model conditioning values by assigning degrees to the units of each layer. Giving units of layers l and $l+1$ both assigned with degrees from 0 to $D-1$, unit with degree d of layer $l+1$ is prohibited from receiving inputs from units of the previous layer $l+1$ having higher degree. The output layer contains $D \times$ *out* units sequentially annotated by batch of *out* from 0 to $D-1$. The so called *masked autoregressive flow* (MAF) allows to perform density estimation of any datapoint $\mathbf{y}$ in a single forward pass. However the model requires D sequential passes to produce a sample from the learned density.

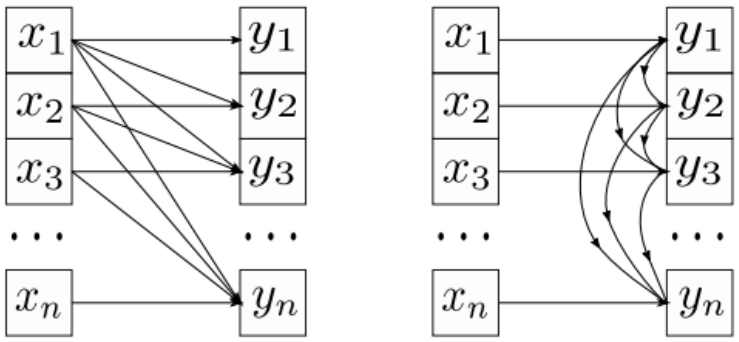


Figure 3.4: The left plot is the direct autoregressive flow while the right plot represent the generative process.

More importantly for this study, Wehenkel and Louppe observe in Wehenkel and Louppe [2020] autoregressive conditioners are valid encoding of Bayesan networks. The multivariate density $p(\mathbf{y}; \theta)$ induced by $\mathbf{f}(\mathbf{y}, \theta)$ and $p_z(\mathbf{z})$ now factorizes with the chain rule as a product of d univariate conditional densities,

$$p(\mathbf{y}; \theta) = p(y_1; \theta) \prod_{i=2}^{d} p(y_i | \mathbf{y}_{1:i-1}; \theta)$$

Then we identify $p(y_i|\mathbf{y}_{1:i-1}; \theta)$ as the coupling of each component z_i of the factored distribution of $\mathcal{Z}$, $p_z(z) = \prod_{i=1}^{d}$ with the corresponding functions f_i and c_i. If we denote $P_i = y_1, \ldots, y_{i-1}$, we

observe the relationship between autoregressive transformations and a BN with no statement of independence and defined by the ordering of the nodes. Stacking multiple autoregressive transformations with permutation of the nodes relax the arbitrary initialization of the vector. Built upon this knowledge, Wehenkel and Louppe [2021] proposes a deep probabilistic graphical model bridging Bayesian network and normalizing flow, allowing to induce expert knowledge about the dependencies of variables.

2.2 Graphical normalizing flow

Graphical normalizing flow is made of a conditioner allowing to introduce information about conditional independence between random variables contained in the input vector. Expert knowledge domain is then injectable into the NN structure as inductive bias. The graphical conditioner is defined as

$$\mathbf{c}^i(\mathbf{y}) = \mathbf{h}^i(\mathbf{y} \odot \mathbf{A}_{[i,:]})$$

With $\mathbf{A} \in \{0,1\}^{d \times d}$ is the adjacency matrix and where $\mathbf{A}_{[i,:]}$, the ith row of $\mathbf{A}$, serves as a mask on the input vector $\mathbf{y}$ (see Figure 3.5). As proved by Wehenkel and Louppe [2021], the topological ordering of any arbitrary Bayesian network leads to a lower triangular adjacency matrix. Thus, the Jacobian determinant from a transformation g based on conditioning factors $\mathbf{c}^i(y)$ selected by following a BN adjacency matrix remains efficient to compute.

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad \mathbf{c}(\mathbf{y}) \begin{cases} \mathbf{c}^1(\mathbf{y}) & = \mathbf{h}^1([0\ 0\ 0\ 0]^T) \\ \mathbf{c}^2(\mathbf{y}) & = \mathbf{h}^2([2\ 0\ 0\ 0]^T) \\ \mathbf{c}^3(\mathbf{y}) & = \mathbf{h}^3([2\ 0\ 0\ 0]^T) \\ \mathbf{c}^4(\mathbf{y}) & = \mathbf{h}^4([0\ 6\ 0\ 0]^T) \end{cases}$$

Figure 3.5: Illustration of the application of the adjacency matrix of the BN structure in 3.2 to GNF. Given a realization of a vector of random variables, e.g. $\mathbf{y} = [2\ 6\ 3\ 4]$, conditioners are functions of the masked vector.

In particular, the autoregressive normalizing flow presented above is a particular case of graphical normalizing flow. Indeed when masking the input vector with a lower triangular matrix filled with one, we recover the same independence structure as with ANF and lead to the same factorization.

In some cases expert knowledge is not sufficient to craft a correct adjacency matrix. Wehenkel and Louppe [2021] investigates an procedure based on *Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning* to learn the adjacency along with the the density at each node. As the procedure needs the matrix to be differentiable, the combinatorial optimization of scorebased learning of a DAG into a continuous optimization by relaxing the domain of A to real numbers instead of binary values. That is,

$$\begin{aligned} \max_{A \in \mathbb{R}^{d \times d}} \quad & F(A) \\ \text{s.t.} \quad & \mathcal{G}(A) \in \text{DAGs} \end{aligned} \iff \begin{aligned} \max_{A \in \mathbb{R}^{d \times d}} \quad & F(A) \\ \text{s.t.} \quad & w(A) = 0, \end{aligned}$$

where $\mathcal{G}(A)$ is the graph induced by the weighted adjancency matrix A and $F : \mathbb{R}^{d \times d} \to \mathbb{R}$ is the log-likelihood of the graphical NF plus a regularization term,

$$F(A) = \sum_{j=1}^{N} \log(p(\mathbf{y}^j; \theta)) + \lambda_{l_1} ||A||_1$$

where λ_{l_1} is an l_1-regularization coefficient and N is the number of training samples $\mathbf{x}^i$. The likelihood is computed as

$$p(\mathbf{y};\theta) = p_{\mathbf{z}}(\mathbf{g}(\mathbf{y};\theta)) \prod_{i=1}^{d} \left| \frac{\delta g^i(y_i; \mathbf{h}^i(\mathbf{y} \cdot A_{i,:}), \theta)}{\delta y_i} \right|$$

The function $w(A)$ that enforces the acyclicity is expressed as

$$w(A) := tr((I + \alpha A)^d) - d \propto tr\left(\sum_{k=1}^{d} \alpha^k A^k\right)$$

where $\alpha \in \mathbb{R}_+$ is an hyper-parameter that avoids exploding values for $w(A)$. Indeed as stated in , in case of positively valued A, an element (i, j) of A^k is non-null iff there exists a path going from node j to node i that is made of exactly k edges. Then $w(A)$ indicates to which extent A is cyclic as its second term accumulates the non-null values of A^k traces, estimating the number of existing paths from a node to itself. Augmented Lagrangian approach is exploited to solve the constrained optimization problem. This optimization procedure requires solving iteratively the following sub-problems:

$$\max_A \mathbb{E}_{\gamma_1,\gamma_2}[F(A)] - \lambda_t w(A) - \frac{\mu_t}{2} w(A)^2$$

where λ_t and μ_t respectively denote the Langrangian multiplier and penalty coefficients of the sub-problem t. They solve these optimization problems with mini-batch stochastic gradient ascent. Once $w(A) = 0$, the adjacency matrix is acyclic up to numerical errors. Finally they recover an exact DAG by thresholding the elements of A while checking for acyclicity with a path finding algorithm.

3 Variational autoencoder

Autoencoders (AEs) are a type of neural network architecture with purpose to learn a representation of a dataset in a lower dimension [Tschannen et al., 2018]. It learns to compress the data information into a latent vector in an unsupervised fashion through a two step process. A first NN, the encoder $f_\theta : \mathbb{R}^d \to \mathbb{R}^h$, takes the input vector and learns to reduce its dimension into a hidden layer of the NN. To verify the quality of compression, a second NN, the decoder $g_\phi : \mathbb{R}^h \to \mathbb{R}^d$, attempts to reconstruct the input vector from the latent vector [Ng et al., 2011]. The loss is a measure of the difference between the initial input and the reconstruction, such as mean square error. The autoencoder actually learns an estimation of the identity function constrained by the dimension of the latent vector.

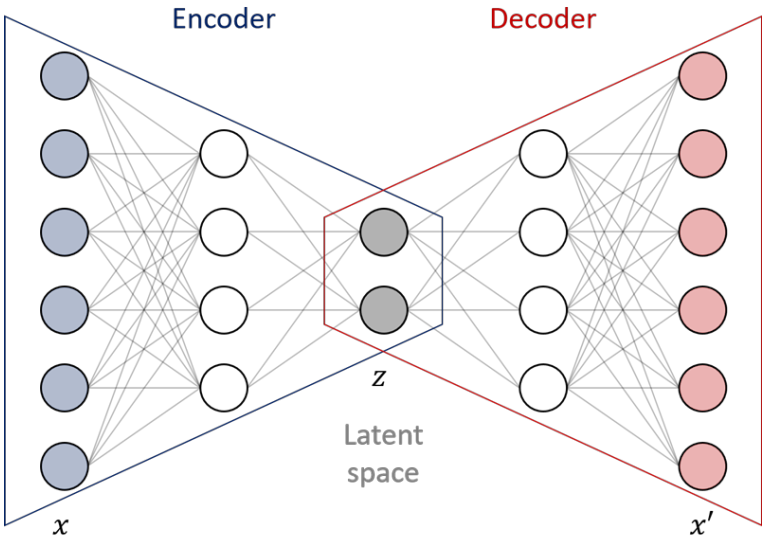


Figure 3.6: The encoder reduces the input data, x into latent variables $\mathbf{z}$ through successive hidden layers. In turn, the decoder produces a reconstruction of the data from z, x'.

In this work, we use variational autoencoders (VAEs) as generative models to contrast results of normalizing flows and variants presented above. VAEs are constructed on the same principle of latent information than AEs. Assuming about generative process that stochastically produces observations $\mathbf{x}$ given latent representation $\mathbf{z}$, VAEs aims at learning an estimation $p_\theta(\mathbf{x})$ of the input data distribution, $p(\mathbf{x})$. To learn the estimation, one has to maximize $p_\theta(\mathbf{x})$ with respect to θ. Given a prior probability distribution of the latent vector representing data in a lower dimension, $p(\mathbf{z})$, the distribution of interest can be written as,

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

New data points are generated by sampling from $p(\mathbf{z})$ then from $p_{theta}(\mathbf{x}|\mathbf{z})$. Yet to fully model $p(\mathbf{x})$, $\mathbf{z}$ should be marginalized out through the integration, which is very expansive to solve numerically. Monte Carlo simulations could approximate the integral but requires huge amount of samples to cover high dimensional latent space. To alleviate this problem, VAEs introduce a family of amortized variational distributions parametrized by ϕ, $\{q_\phi(\mathbf{z}|\mathbf{x})\}_\phi$. The logarithm of the marginal distribution is then approximated as

$$\begin{aligned}
\ln p_\theta(\mathbf{x}) &= \ln \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} && (3.1)\\
&= \ln \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} && (3.2)\\
&= \ln \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] && (3.3)\\
&\geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \ln\left[\frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] && (3.4)\\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\big[\ln p_\theta(\mathbf{x}|\mathbf{z}) + \ln p(\mathbf{z}) - \ln q_\phi(\mathbf{z}|\mathbf{x})\big] && (3.5)\\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\big[\ln p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\big[\ln q_\phi(\mathbf{z}|\mathbf{x}) - \ln p(\mathbf{z})\big]. && (3.6)
\end{aligned}$$

Through the Jensen's inequality (equation 3.4), we identify a lower bound on the log-likelihood we want to approximate, called Evidence Lower BOund (ELBO) [Ng et al., 2011]. The equation itself forms a auto encoder like model. $q_\phi(\mathbf{z}|\mathbf{x})$ would be the stochastic encoder and $p(\mathbf{x}|\mathbf{z})$ the

stochastic decoder. The first part of ELBO, $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\big[\ln p(\mathbf{x}|\mathbf{z})\big]$, is the *reconstruction error*. The second part, $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\big[\ln q_\phi(\mathbf{z}|\mathbf{x}) - \ln p(\mathbf{z})\big]$, correspond to a *regularizer* and coincides with the Kullback-Leibler divergence. Minimizing the negative ELBO encourages the decoder to reconstruct the data as good as possible. It also forces the NN encoder to learn parameters of latent distributions as close as possible to the prior distribution, $p(\mathbf{z})$.

As sampling is not a differentiable function, the backpropagation algorithm cannot be applied for optimization. The reparametrization trick solves this by dividing the problem into two part. First the encoder learns parameters for the posterior distribution. Then samples are drawn from a unit Gaussian and rescaled according to encoders outputs. The sampling process is not concerned by the optimization anymore and the backpropagation only concerns the latent parameters learned.

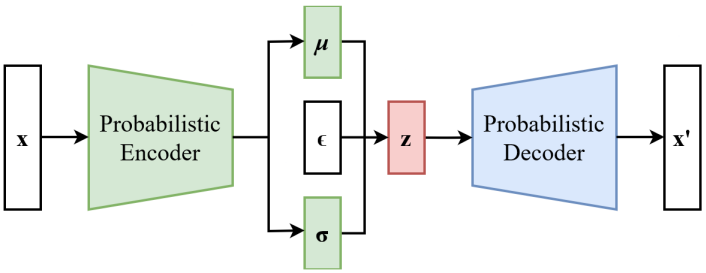


Figure 3.7: Adaptation of AEs to probabilistic paradigm. The encoder learns vectors of mean and variance. The reparametrization trick creates the latent vector of distribution, $p(z)$.

Chapter 4

Related works

In this section, classic approaches for load forecasting are first reviewed. In a second part, probabilistic modeling methods in the frame of load forecasting are discussed. Finally we present most recent applications of neural networks in power system and work related to our subject.

In [Hong and Fan, 2016], authors propose to classify the horizon of forecast into different categories. For this study, we focus on right edge of very short-term load forecast of the scale represented in Figure 4.1 and dig into methods and features related to it. Day-ahead load forecast aims at creating refined schedule for electricity production and energy stock bids.

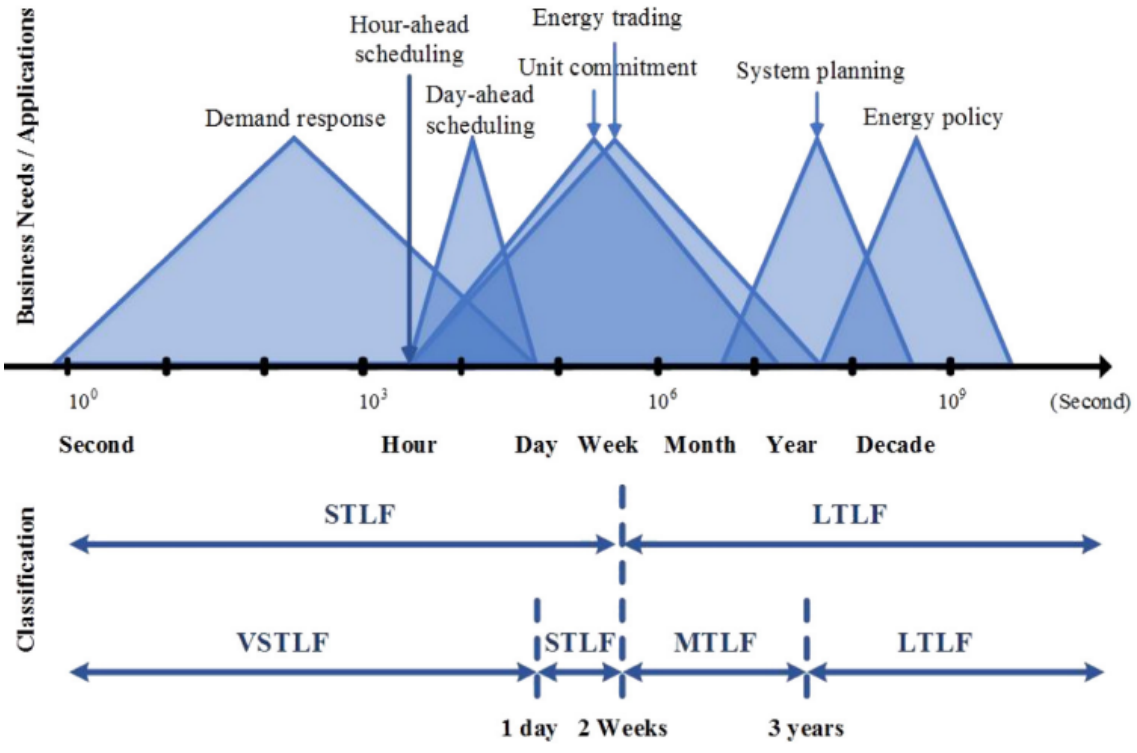


Figure 4.1: Different time span and application of load forecast.

Many researchers applied multiple variables regression to model the relationship between electrical load and explanatory variables, such as Charlton and Singleton [2014]. From an initial parametric model depending on day and temperature, they successively refine their model. For example, they propose to create subsets of the initial dataset and build distinct models on groups likely to present different load responses to clustered explanatory variables (zone,

seasons, weekdays, or weekend days, ..). Performing treatment of outliers from the dataset, noticing anomalies such as zero values, also lead to improvement. As their model extensively uses temperature, a combination of weather stations is studied to find the best fit for the energy usage of a specific zone. Scores were defined with the Weighted Root Mean Square Error.

Autoregressive models were also largely used to explore load forecasting. Auto Regressive Integrated Moving Average (ARIMA) is the most preferred method among AR for load forecast. However it is often enriched with supplementary non linear methods supposed to cope with non linearity ARIMA can't accurately model. Fard and Akbari-Zadeh [2014] propose an hybrid approach in which the Box-Jenkins ARIMA method is combined with Artificial Neural Network and wavelet transform to forecast residual terms of ARIMA and bring correction to the linear model. They demonstrate their hybrid model outperforms classic method of the field such as pure ARIMA, ANN, or Supported Vector Machine. However autoregressive model of this type do not make use of contextual data such as weather, demography or economy which may bring valuable information to spatio-temporal forecast of load.

As in many fields, load forecasting in the literature before the last decade concentrated on point forecast before shifting to probabilistic forecast. Indeed having access to a range of plausible outcomes is far more informative and accurate than concentrating the whole density into a single point. Yet, relevant insight about features selection and data analysis are of interest in the point forecast paradigm. Probabilistic forecasting refers to methods outputting forecasts in the form of scenarios, quantiles, intervals, or probability densities. Khoshrou and Pauwels [2019] propose scenario-based probabilistic load forecasting through the generation of temperature scenarios and ensemble of regression trees. Quantiles are then derived from the set of issued load forecasts. A popular method for probabilistic forecasting is quantile regression averaging. Liu et al. Liu et al. [2017] leverage a combination of point forecasts produced from sister models of regressions methods (mainly based on temperature and time information) to produce a weighted set of quantiles. The main motivation of this method is it leans on well-known methods of point forecast in the forecasting literature.
Quantiles forecast only offers partial information about the uncertainties of the load demand. Guo et al. Guo et al. [2018] combine deep learning model with quantile regression followed by a gaussian kernel density estimation to produce probability densities estimators of the load demand. The approach is motivated by the higher coverage in the probability of prediction interval.

Within deep learning methods, RNNs are very popular for energy forecasting applications. The recurrent behavior of the NN naturally applies to timeseries with long dependencies. Shi et al. [2017] proposes a pooling-based deep recurrent NN in the frame of short-term household load forecasting. Pooling batch of load profiles into the input allow to mitigate the high variance and volatility of a typical household. Their model outperforms statistical models such as ARIMA or classical RNN. A bidirectional long short term memory (BLSTM) architecture is implemented in [Toubeau et al., 2018] and make use of copula-based strategy to reconstruct cross-variable dependencies when generating scenarios. However, so far showcased models imperatively needed scenarios from exogenous variables if any context information was used. In the other hand, generative models allow to draw scenarios of the variable of interest directly. A model is said to be generative when it models the joint distribution of all its random variables. Notably, this class of model allows to generate new data from the probability structure learned in the training data. As such, generative models are interesting methods to produce load forecast in an electrical grid. Within this category deep generative

models has gained popularity. Variational Auto Encoder (VAE) [Kingma and Welling, 2013], Generative Adversarial Network (GAN) [Goodfellow et al., 2014] or Normalizing Flow (NF) [Tabak and Vanden-Eijnden, 2010] and variants performed very well in applications related to power systems. Among other studies we noticed the work from Dairi et al. [2020] that apply VAEs to photovoltaïc production forecast. Based on two PV plants in US and Algeria, they compare seven deep learning methods and highlight VAEs performances are on par with state of the art models. Furthermore they show VAEs systematically outperforms two considered baseline machine learning models. Concerning GANs, Yuan et al. [2021] proposes a progressive growing GAN (pg-GAN) to capture complex temporal dynamics and pattern correlations of wind power generation. Progressive growing procedure allow GAN to progressively learn from low resolution data to fine grained temporal series, improving quality, stability and variation. In the load demand area, Lan and Guo [2018] uses conditional GAN to create contextual based load demand forecast in various type of industry. Finally we found Ge et al. [2020] that explore the application of flow-based generative network to model daily load profiles in a distribution network. Dumas et al. [2022] goes further by thoroughly comparing the 3 methods cited above on a large range of metrics. Despite the low number of references in the energy forecasting area, it shows normalizing flow obtain decent scores and profit from various assets. It motivates practitioner to include NFs in their generative model toolkit.

However this study aims to take advantage of the geographic information. We hope to introduce inductive bias in NF applied to power system and measure gains obtained from it. Very few work about integration of expert knowledge (e.g. spatial and/or temporal dependencies) has been lead in the field of power system to this day. Most of approach integrate to the input some copula-based information for the model to reconstruct dependencies. In [Hu et al., 2017], authors use a Pair Copula approach to find correlation between errors of different wind farms and use it to improve forecast model of power generation. Yet this is different from constructing the model architecture around the introduction of expert knowledge. Ringsquandl et al. [2021] recognizes the potential benefits topology-based inductive bias could bring to power system problems. It proposes a graph neural network to apply on five different dataset for state estimation (SE). SE aims to approximate the voltages and phase angles at junction nodes, corresponding to busbars, of a power system by propagating information of injections and measurements across the system's network. Their main observation is that topology-aware GNN models for SE has net benefits when it comes to handling noisy labels. The inductive bias copes effectively with uncertain data but known topology.

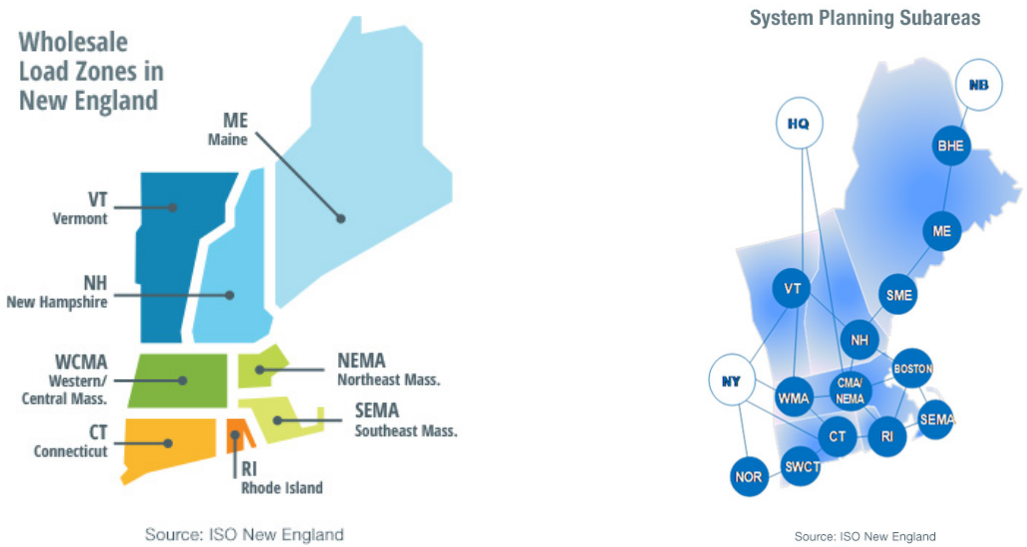
Chapter 5

Case study

In chapter 3 we defined a bunch of models that match some of the requirements for load probabilistic forecast. We now need a dataset highlighting gains obtained from the introduction of inductive bias. Concerning this study, we know independence might exist between some variables of the forecast random vector and use GNF to exploit it.

1 Settings

ISO-NE is an independent system operator (ISO) and the regional transmission organization (RTO) of power system and transmission lines in New England, USA. It is responsible for reliably managing the power sytem in the states of Connecticut (**CT**), Maine (**ME**), Massachusetts (**MASS**), New Hampshire (**NH**), Rhode Island (**RI**) and Vermont (**VT**). Another of ISO-NE duty is to fix prices, terms and conditions of the energy supply in New England.



(a) (b)

Figure 5.1: (5.1a) Map of New England with division of the territory into 8 aggregated load zones. We observe **MASS** is divided in turn into 3 sub zones. (5.1b) Map of New England representing the different transmission interfaces of load across the network.

ISO-NE presents a division (see Figure 5.1a) of the New England's transmission system for aggregated load zones. These zones load consumption are taken into account when computing the pricing in the wholesale electricity marketplace, along with generating units node. Related to this map, Figure 5.1b indicates a set of subareas linked between each others. This graph form a simplified model of load areas connected by the major transmission interfaces across the system. This model is generally used by the organisation for some high level studies of resource adequacy, operating-reserve requirements, production cost, and environmental emissions[1]. These maps are interesting information about the geographic situation and topological arrangement of the electrical grid in New England.

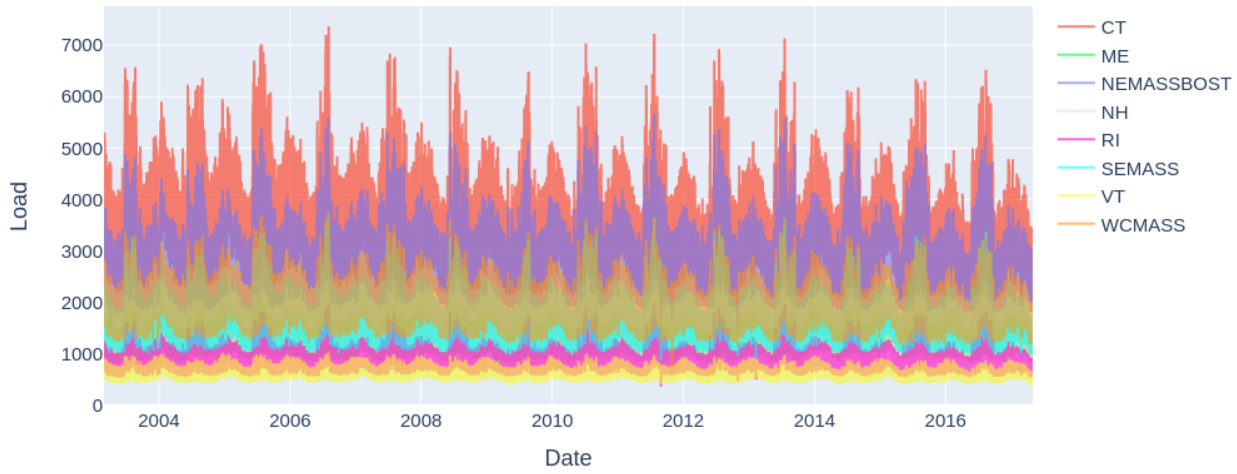


Figure 5.2: Each zone correspond to a different color. **CT** zone reaches highest average and peaks. It is observable data shows off cyclic behaviour at different frequencies.

ISO-NE gives access to historical data about a broad range of information related to the power system such as prices, loads, weather. As shown in Figure 5.2, for this work we retrieve 15 years of electrical aggregated load demand from the eight zones presented in Figure 5.1a. Data comes in the form of hourly load demand identified with the timestamp, allowing to derive various temporal information such as the day of the week, month, etc.. Records of dry bulb and dewpoint temperatures are also made available for each entry of the dataset. To this dataset we add information about U.S. federal holidays, to be able to discriminate special consumption days. This results in a set of data integrating *1 241 710 datapoints* of hourly load demand along with weather and calendar information distributed in 8 zones during 15 years. The core of the experiences will be to test if we can identify spatio-temporal independence among load demand and take advantage of them with GNFs.

2 Exploratory data analysis

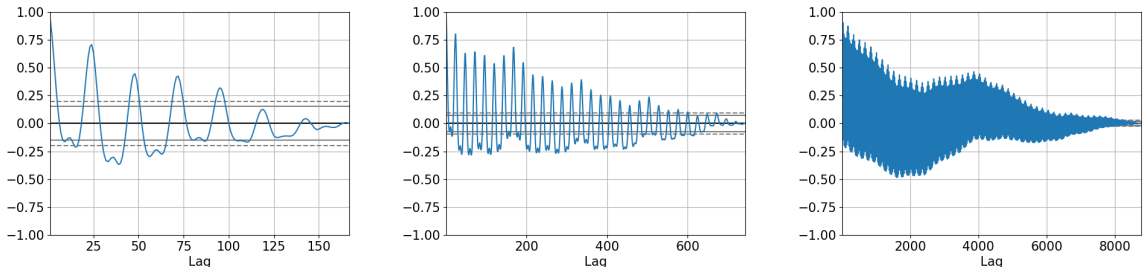
In this section we aim to create an intuition about data that populate our dataset and their dynamics. Having a good comprehension of the data content allows not only to coherently craft a model (feature selection, model selection, etc.) but also to spot and interpret more

[1] `https://www.iso-ne.com/about/key-stats/maps-and-diagrams`

easily any particular behavior of the final models.

2.1 Load analysis

First on Figure 5.2, presenting an overview of the studied load demand period, we spot two main findings. Load demand presents cyclic patterns at different frequencies that seems to match the seasons, the weeks and the days. Figure 5.3 confirms our intuition as it displays peaks of auto-correlation at different scale. We directly observe the repeated peaks with a period of $\sim$ 24 lags (hours). We also notice a pattern of negative correlations for lags within the period of a day. We discover on Figure 5.3b that lags corresponding to a week from the timestep t have higher correlation values. This means the day of the week impacts the load consumption behaviors (e.g. week-ends probably don't display the same patterns as week days). The same conclusion applies to yearly scale (Figure 5.3c). Calendar data thus brings valuable information to discriminate load distributions. Beyond those cyclic patterns, we know others context variables drive the load demand at a finer granularity, as learnt in [chap:exp]chapter 4.



(a) (b) (c)

Figure 5.3: Auto-correlation plots at different scale. Plots were performed on timeserie of the sum load from all zones. Lag k indicates the shifted timestep considered to compute the correlation in regard to the current load. Dashed straight lines and plain straight lines represent respectively the 95% and the 99% interval confidence bands. On the weekly scale plot (5.3a), we mostly observe 7 repeated peaks. On the monthly scale plot (5.3b), we notice a second frequency that repeat every week (168 timesteps). On the yearly scale plot (5.3c), we spot the periodical effect of seasons.

The second finding from 5.2 is the high correlations displayed among the different zones. Correlation heatmap among zones, Figure 5.4 confirms this trend. It indicates load demand across zones at time t tends to vary accordingly. It may suggest that a single global context vector (calendar information apart, which is naturally shared at time t) is enough to marginalize the load distribution for each zone. This seem to be coherent as the 8 zones are situated next to each other and are mostly impacted by the same external stresses such as weather, economy, policies and so on. Load distributions are then potentially independent when conditioned on the right context variables. In Figure 5.4 we also spot two zones, **VT** and **ME**, that are particularly less correlated to others. After deeper investigation we found those states were particularly less dense in population but with important economic activities (IBM, BioTek, etc.) [ACCD et al., 2016]. This might explain different load consumption patterns.

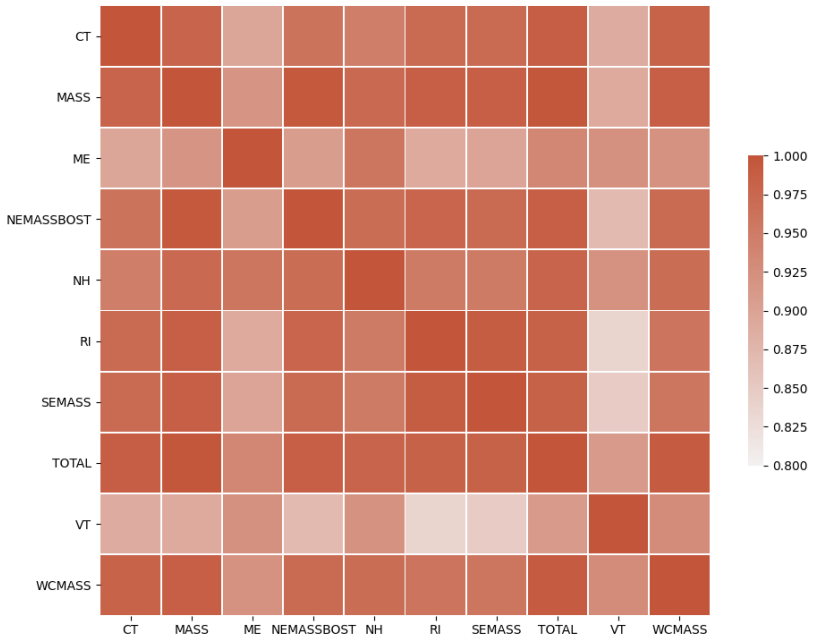
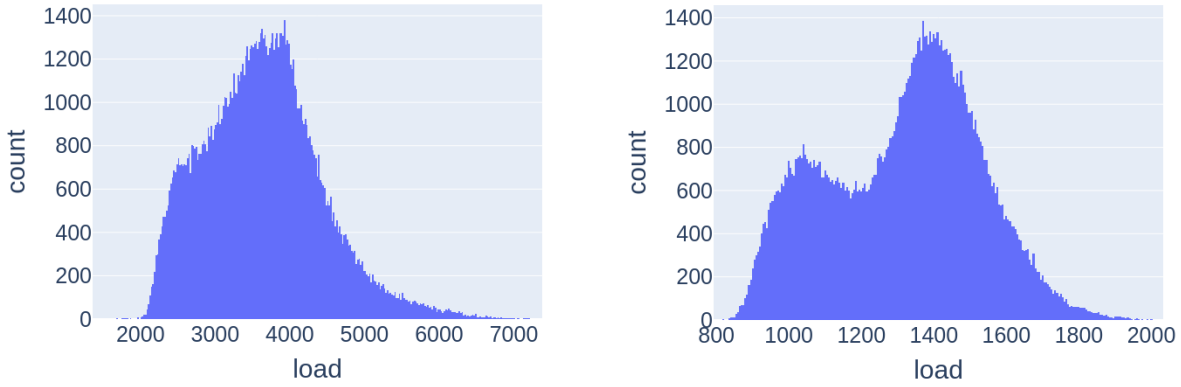


Figure 5.4: Pearson correlation among zones of the grid. To better visualize, the scale is shifted and begin at a value of 0.8. **VT** and **ME** are zones less correlated to others but still reach value around 0.8. Other zone attain correlations between 0.9 and 1.



(a) Histogram plot of the load in **CT**.

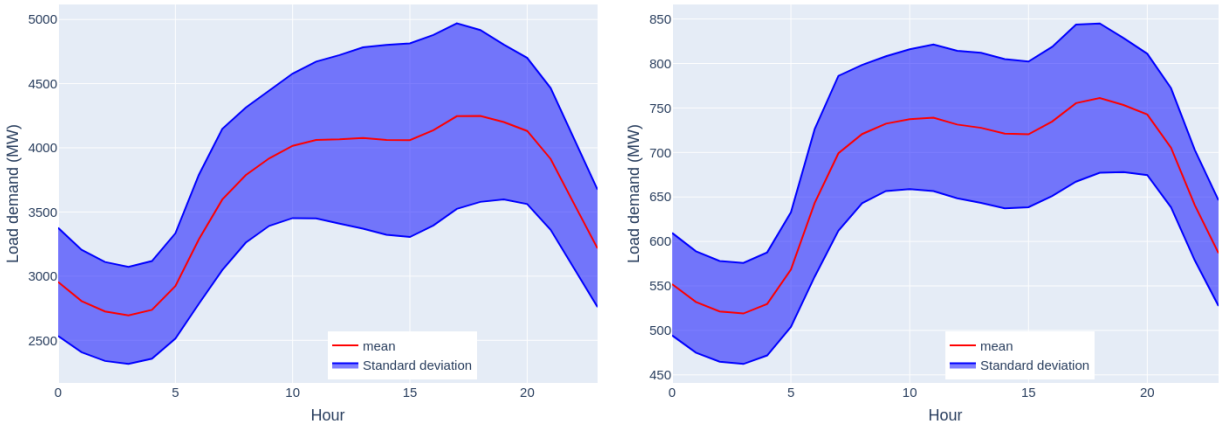
(b) Histogram plot of the load in **ME**.

Figure 5.5: Histograms of load demand for **CT** and **ME**. They are good representation of the main types of histograms from other zones. **ME** and **VT** (5.5a) present distributions that look like Gaussians mixture with two clear distinct regimes. The other zones (5.5b) display important concentrated data bulks around their respective medians but are highly right skewed.

We spot two main types of histogram regarding zones. **ME** and **VT** (Figure 5.5a) fall in the first type and present distributions mixture. It indicates some situations make the load demand switch from a regime to another. The second type of histogram usually present a plateau for lower values of load then a peak with high concentration of datapoint around the median. Finally the histograms tend to be strongly right skewed meaning the demand is less certain with context requiring higher load consumption. Also those two types of histogram

consolidate the result found on plot 5.4. **ME** and **VT** posses two regimes and don't always follow the same trend as other zones.

Once again, we display two different profiles to represent typical behaviors in different zones. On Figure 5.6, we spot standard deviation is higher when the load demand reached a plateau from approximately 9am to 7pm. Depending on the models capacity to condition load on context, this might lead to period of higher uncertainties when forecasting the day-ahead load demand. We also observe the fall around midday is more present in zones such as **VT** than other. This probably partly explains different regimes showcased in Figure 5.5a.

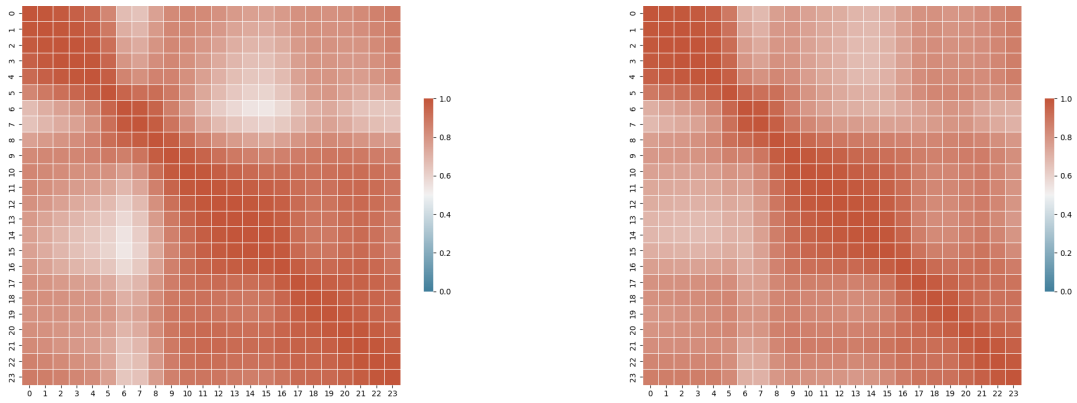


(a) Average load and std from CT.

(b) Average load and std from VT.

Figure 5.6: Average daily load profile along with their standard deviations. **CT** and **VT** are picked as two different typical load profiles being representative of other zones load profile. We mainly observe the decrease in load demand around midday is more marked on **VT** type, both in term of mean and std.

Concerning intraday correlation (see Figure 5.7), we observe some interesting behaviors. Naturally, successive hours of the day tend to follow the same trend. For **CT**, we even detect a submatrix running from 8am to 23pm of load demand very correlated with each other. We remark a band of lower correlation values around ∼3am to 7am, corresponding to the regime transition period. This means the rate at which we attain the plateau is not constant in regard to the plateau value. In Figure 5.7b, we notice the phenomenon is less visible in **VT** zone where all values seems to be smoothed.

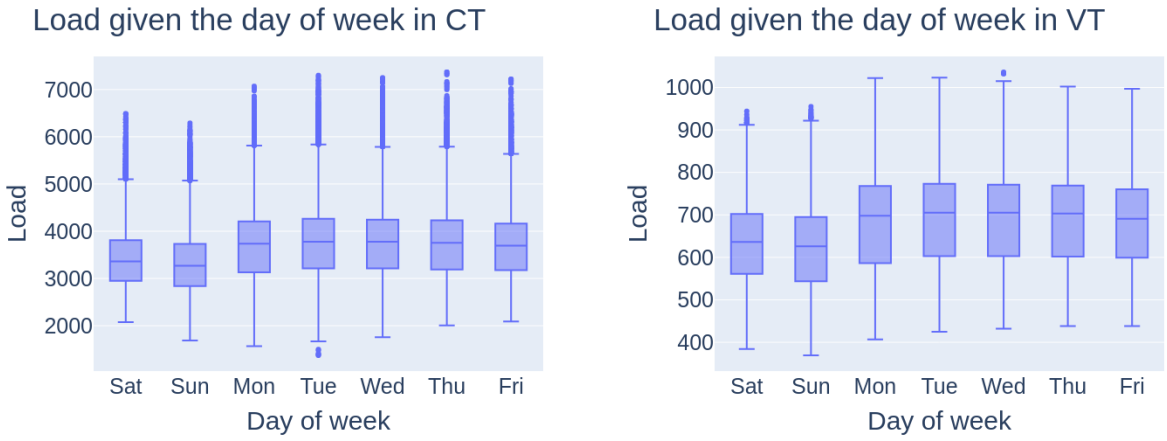


(a) Intraday correlation plot from **CT**. (b) Intraday correlation plot from **VT**.

Figure 5.7: Intraday correlation plot from two typical zones, namely **CT** and **VT**. In both zones, values around and on the diagonal are close to 1. Also in both situations, we remark a period of hours (∼3am to 7am) with lower correlations compared to the diagonal.

2.2 Explanatory variables analysis

Now that we discovered how load demand behaves when ignoring all external information. In this part, we break down load demand into multiple distributions given particular exogenous variables. We aim to verify if available context vector might have a significant impact on distributions discrimination.



(a) (b)

Figure 5.8: Boxplots of the load demand given the zone and the day. Two typical zones are picked as before. In both case we notice the influence of the day on statistics, in particular quartiles and skews are not the same for week days and week-ends.

Following the discussion above about Figure 5.3, we display boxplots for each day of a week for two different representative zones. We observe the week period signal found in Figure 5.3b is mainly drove by the difference between week days and week-ends. Within days of the week, we only spot slight differences in quartiles and outliers. Skews of **CT** boxplot reflects the distribution of the corresponding histogram. It informs us the skewness is not tied to a particular day of the week. Instead, looking at Figure 5.9, we understand temperature is a good

driving factor of this behavior. When detailing **CT** datapoints, we observe mild temperatures (50-70F) lead to minimum load demand responses. When reaching high temperature (from 80 to 100), load response are high and quite distant from the bulk of data. We explain this as buildings prevent extreme temperature within their walls with heating or air conditioning, leading to higher consumption. Slicing load responses per temperature, $T \sim 80$ has the most spread marginal distribution, presaging higher uncertainty for day forecast with those temperatures. Assessing distributions on left and right temperature ends is more difficult as we fall short of datapoints in those regions.

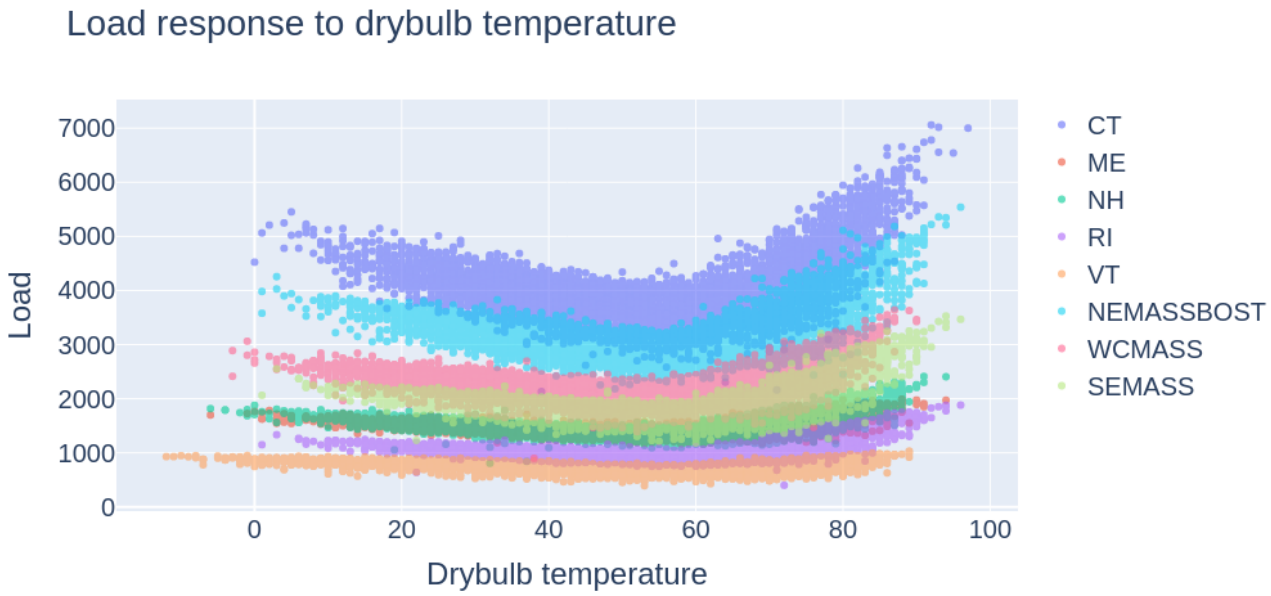
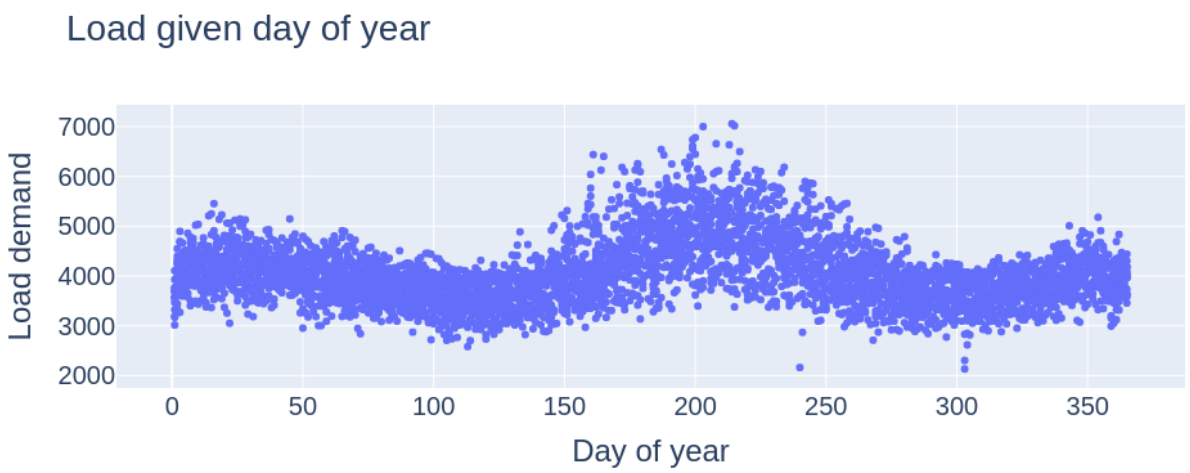
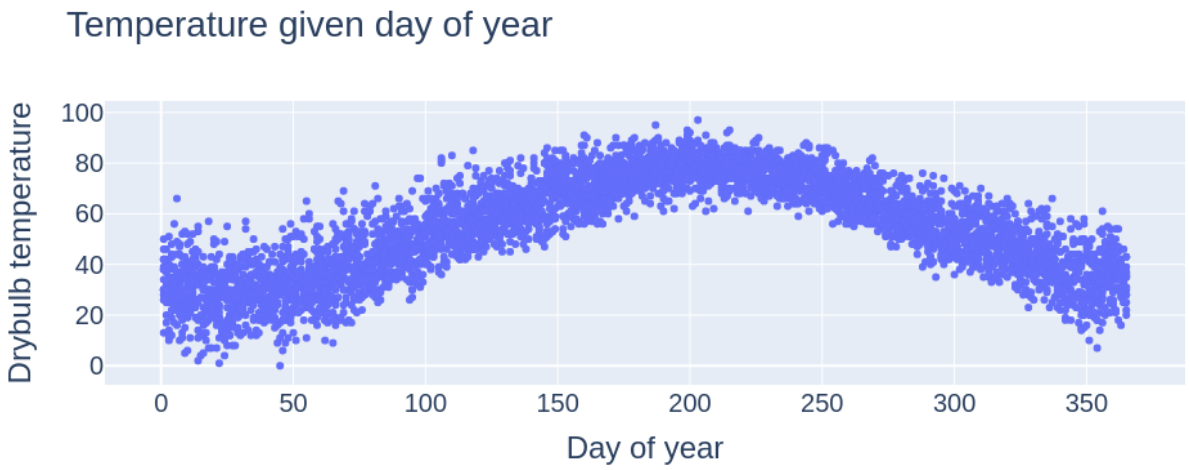


Figure 5.9: Load response (MW) to temperature in fahrenheit. Datapoints are seperated into zones. For most zones, we could characterize the response with a piecewise linear model. A first line with a slight negative slope is draw from 0F to 60F. A second line completes the response with a steeper positive slope. However **VT** seems to follow a simple linear response to the temperature.

On plots 5.10, observing both figures allows us to the wide range's responses of load during high temperature periods is not linked to spread distribution of temperature. However, other weather factors we don't have access to (wind, relative humidity, etc.) may further explain these profiles. In our setting, we could expect higher drybulb temperature to bring higher uncertainties in the forecast.



(a) Response in load (MW) sliced at 12am to the day of year (0 - 364) for **CT**. The initial timestep is referenced on January 1st. Spring and autumn are the less demanding seasons and present narrower distributions. Load response in winter are higher but keep a narrow distribution profiles. In summer, load demand reaches the highest values and have much more wide distribution.



(b) Response in temperature (F) sliced at 12am to the day of year (0 - 364) for **CT**. The curve fits a parabola. In summer, temperature distributions are quite narrow. From summer to winter, distributions tend to spread then to shrink when approaching summer again.

Figure 5.10: Merging perspectives, we notice spread distribution from load response correspond to narrow temperature distributions for corresponding days of year. Having access to 15 years of data, each day distribution holds approximately 15 points.

3 Data processing

Raw data is retrieved from a public git repository[2]. The first data processing is to remove entries corresponding to daylight saving times as they are duplicates of the previous or next hours. A new dataset is derived by concatenating the 192 scaled values of 24 hours of a same day in each of the 8 zones into a vector. If any value is missing within a day, we drop the entire vector, which justified as the dataset is large enough and these entries are not numerous. Beside the load vector, we construct a context vector. It is formed from 192 values of scaled

[2] `https://github.com/camroach87/gefcom2017data`

temperature and from $2 \times 2 + 2$ values of calendar information. Indeed, we push into the context vector two cyclic information, namely day of year and month of the year, that we decompose into their trigonometric components. Decomposing cyclic variables into sin and cos signals transform entries into unique representation with equidistant values. Plus we add two binary variable, one indicating if the day is during the week-end (TRUE) or the week (FALSE) and the other if its a holiday (TRUE) or not (FALSE).

3.1 *Demand* scaling

Scaling the demand data has to be done as loads in different zones do not present the same order of magnitude. Normalization has been preferred over standardization as outliers are better represented in normalization with no bound to the resulting transformation. Normalization is done by subtracting the mean to initial values and dividing them by the standard deviation of the data.

3.2 Calendar encoding

In later experiments, a vector of context is added to help to condition the probability distributions learned by flow-based models. Indeed as observed on fig. 5.2, load demand is strongly subject to seasonality and presenting representations of calendar information could improve discrimination of the target distribution for a given day.

To keep the cyclicity and normalize it, the numeric value of the day of year (doy) is transformed into

$$doy_{\sin} = \sin(\frac{2\pi doy}{365})$$
$$doy_{\cos} = \cos(\frac{2\pi doy}{365})$$

Similar reasoning is applied to encode the month of the year (moy).

$$moy_{\sin} = \sin(\frac{2\pi moy}{12})$$
$$moy_{\cos} = \cos(\frac{2\pi moy}{12})$$

3.3 *drybulb temperature* scaling

Hourly drybulb temperature is also introduced in the context for each zone. Normalization is applied as to the demand load data.

Chapter 6

Experiments

1 Metrics

We introduce three metrics to evaluate performance in probabilistic forecasting of the electrical load demand.

Continuous Ranked Probability Score (CRPS)

CRPS is a usual metric to measure skill of probabilistic forecast. The measure was introduced in different papers ([Brown, 1974], [Matheson and Winkler, 1976]). CRPS is a quadratic measure of the discrepancy between the forecast CDF and the empirical CDF of y. It can be interpreted as a generalization of the mean absolute error in the point forecast setting. An instantaneous formulation of CRPS is given by

$$CRPS(F, y) = \int_{-\infty}^{+\infty} (F(x) - \mathbb{1}(x \geq y))^2 dx,$$

where F is the cumulative distribution function of the forecast while y is a realisation from the real distribution (see Figure 6.1). CRPS is a negative oriented measure, for which lower scores are better, and is included between 0 and $+\infty$. To achieve a score of 0, the forecast would have to be deterministic and exact for the sample cdf to superpose on the observation. However, as we operate in the probabilistic setting, we do not wish our forecasters to reach such a value. Yet as Gneiting and Raftery [2007] state, CRPS is a proper scoring rule, meaning the expected value of the score is minimized when the observation is drawn from the same distribution than the predictive distribution.

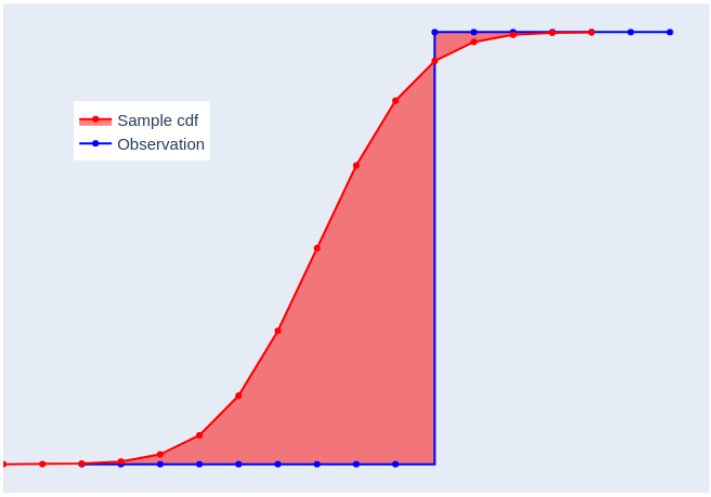


Figure 6.1: Illustration of the instantaneous CRPS. The measure is proportional to the filled area. The sample CDF describes the CDF obtained through generated forecast realisation.

According to [Zamo and Naveau, 2018], "the forecast CDF may be known, but an analytic formulation of the CRPS may not be derivable. In the latter case, one may be able to sample values from F." Thus the solution proposed in the paper cited above is to evaluate an estimator of the CRPS, making use of limited information made available by pair of realisation and ensemble of samples produced by the forecasts. Taillardat et al. [2016] propose a representation of CRPS based on L-moments, the *probability weighted moments* CRPS ($CRPS_{PWM}$).

$$CRPS_{PWM}(X, y) = \mathbb{E}_F|X - y| - \mathbb{E}_F\{X[2F(X) - 1]\},$$

An estimator, $\widehat{CRPS}_{PWM}$, is derived from this representation as,

$$\widehat{CRPS}_{PWM}(\{x_1, \ldots, x_M\}, y) = \frac{1}{M}\sum_{m=1}^{M}|x_m - y| + \hat{\beta}_0 - 2\hat{\beta}_1,$$

where $\hat{\beta}_0 = \frac{1}{M}\sum_{m=1}^{M} x_m$ and $\hat{\beta}_1 = \frac{1}{M(M-1)}\sum_{m=1}^{M}(m-1)x_m$. $\{x_1, .., x_M\}$ is an ensemble generated from the forecast distrbution.

In our setting, to further synthetize the information, the PWM CRPS estimator is averaged along the day for each zone. For each observable day of the test set, **10 scenarios** are generated with their context through models to create ensembles for each random variable. It is important to note CRPS calculates a measure based on marginal density of forecast. As CRPS is computed taking into account univariate realisations, this score can miss information contained in trajectories realisation. In the other hand CRPS being a generalization of the mean absolute error in the deterministic setting, it offers an intuitive measure. Plus it allows to examine a score for each variable if not averaged.

Energy score (ES)

Multivariate scores are proposed to measure performance of generated scenarios based on their full trajectories. The ES is a generalisation of CRPS in the multivariate setting, introduce by Gneiting and Raftery [2007]. The ES is calculated as,

$$ES = \frac{1}{M}\sum_{m=1}^{M}||\mathbf{x}_m - \mathbf{y}||_2 - \frac{1}{2M^2}\sum_{m'=1}^{M}\sum_{m=1}^{M}||\mathbf{x}_m - \mathbf{x}_{m'}||_2$$

With M as the size of the ensemble, $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ two independent trajectory realisations from the ensemble and $\mathbf{y}$ the real vector of observations. ES particularly discriminates models with correct variances but biased means. However compared to univariate score CRPS, ES identifies better forecaster respecting the probabilistic contract. Indeed the second term advantages models with large variance between scenarios while the first term penalizes scenarios far away from the true values. This lead models well calibrated with honest uncertainties to score better. Despite being a multivariate measure, ES do not consider dependence structure within scenarios generated.

Variogram score (VS)

Correlation structure holds a major role in spatio-temporal forecast as consecutive hours or places are higly dependent. VS brings better insight about the correlation between variables of the multivariate random vector [Scheuerer and Hamill, 2015]. Through VS, forecasters are not only assessed on the values but also on the correctness of correlations generated. The VS is defined as,

$$VS = \sum_{i,j=1}^{d} w_{i,j} \Big(|y_i - y_j|^\gamma - E_F |x_i - x_j|^\gamma \Big)^2,$$

with $w_{i,j}$ weighting the impact of each pair of element in the measure and γ defining the order of the measure. The expectation can be estimated with the ensemble of scenario generated,

$$E_F |x_i - x_j|^\gamma = \frac{1}{M} \sum_{m=1}^{M} |x_i^{(m)} - x_j^{(m)}|^\gamma.$$

w_{ij} is set to 1 for all i, j pair as we pay attention to error accross all hours and zones with no discrimination. VS effectively discriminate missestimated means, variances and correlations. In particular, error in correlation are embedded in VS through the differentiation between temporal difference of pairs from real observations and corresponding expected difference from generated samples.

2 Hyperparameters search

Before comparing final results, we aim to tune the hyperparameters of each different method. We perform a grid search on each set of hyperparameters and discuss their impact and limitations. To perform experiments, the dataset is split into 3 parts respecting chronological ordering (no shuffling). A train set is formed from 80% of the total dataset. In turn, the remaining 20% are divided equally into a validation set and a test set. As we forecast information, it is important to not shuffle the data and actually observe scores of future days (relatively to train set) at test time. As such we acknowledge load distribution might evolve with time and observe if our models are or are not able to model this transformation. Forecasters obtained during the grid search are run against the validation set. Based on their losses, we pick some of them out to perform further analysis on scores. Finally, models retained are compared against each other on the test set in the section Results. All optimization are executed with ADAM. In addition to hyperparameters relative to each algorithm, we look to improve loss for all models through the tuning of the learning rate and the weight decay of the optimizer.

During the hyperparameters search, models are trained for 50 epochs. As most models reached a plateau even before 50 epochs during pre-experiment phase, we agree on this value for resources

saving. Candidates for final models are retrained 50 epochs further. All training, validation and testing procedure are performed on a personal laptop (DELL XPS15 9560) mounted with CPU INTEL Core i7-7700HQ (8 cores clocking at 2.80GHz) and GPU (NVIDIA GeForce GTX 1050 Mobile).

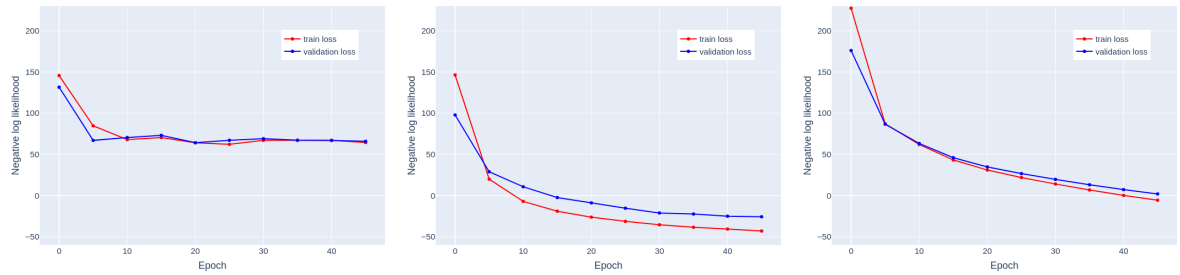
2.1 Autoregressive normalizing flows

With ANF plugged with affine normalizer, we observe behaviors of different models learnt with varying number of layers and neurons contained in the masked autoregressive network. We also tune the number of flows we pile on top of each other. Moreover when combining the autoregressive conditioner with monotonic normalizer, we have to search for the number of layers and neurons to include in the integrand MLP. We also test different numbers of conditioning inputs to the normalizer (corresponding to the output of the conditioner) and different numbers of step for the numerical evaluation of the integral.

Number stack	Conditioner layers	Learning rate	Weight decay
1	[200, 200, 200]	0.0001	0.01
2	[300, 300, 300]	0.001	0.05
3	[200, 200, 200, 200]	0.01	0.1
5	/	/	/

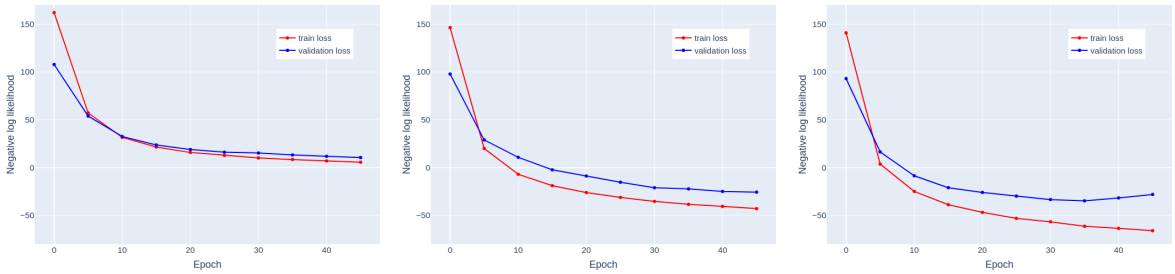
Table 6.1: Summary table of hyperparameter values tested with ANF with affine normalizer. Dictionaries of model hyperparameters are created from the permutation of values contained in respective set of hyperparameters. The number of elements in lists contained in *conditioner layers* expresses the number of layer while the actual numbers indicate the number of neurons in the respective layer.

We first screen losses obtained when varying the parameters of the optimizer. Keeping other hyperparameters constant, we observe the learning rate should be set around a typical value of 10^{-3}. If smaller, the convergence of the learning becomes to slow. In the other hand, setting it higher destabilizes the learning process as it probably jump over local minimum without converging. Concerning the weight decay, the better pick is situated around 10^{-1}. Higher values apply too much constraints on the weights to be as close to zero, reducing the expressiveness of the model. While lower ones lead models to overfit the training set as we see the training loss keeps on decreasing without the validation loss following it.



(a) (b) (c)

Figure 6.2: Loss scores for varying values of the optimizer learning rate, λ. (6.2a) With $\lambda = 0.01$, loss converges to a poor value. (6.2b) With $\lambda = 0.001$, score converges smoothly to a lower value, while (6.2c) with $\lambda = 0.0001$, the value tends to decrease smoothly but not fast enough. All other hyperparameters are fixed.



(a) (b) (c)

Figure 6.3: Loss scores for varying values of the optimizer weight decay, wd. (6.3a) With $wd = 0.5$, the model trained do not perform well enough due to strong constraints. (6.3b) With $wd = 0.1$, loss on validation set follows the same trend as on the training set. (6.3c) When $wd = 0.01$ however, the regularization is too loose and we observe an important gap between loss scores.

Concerning the conditioner layers, we notice models benefit better from increasing the number of neurons in each layer instead of increasing the number of layer in the masked autoregressive network. We hypothesize this outcome is due of the architecture of the MAN. Expanding a layer size allows for each degree of the autoregressive process to be included multiple times in this layer. This increase the expressiveness allocated to each degree. Finally we state that the main tuning factor is the number of flows piled. From one to three flows, the validation loss decreases approximately of a factor two for each additional flow. We spot that stacking more than 3 flows does not improve the loss anymore. This observation meet the theoretical result in Wehenkel and Louppe [2020] stating that stacking more than three steps of affine normalizer can't bring any new improvements.

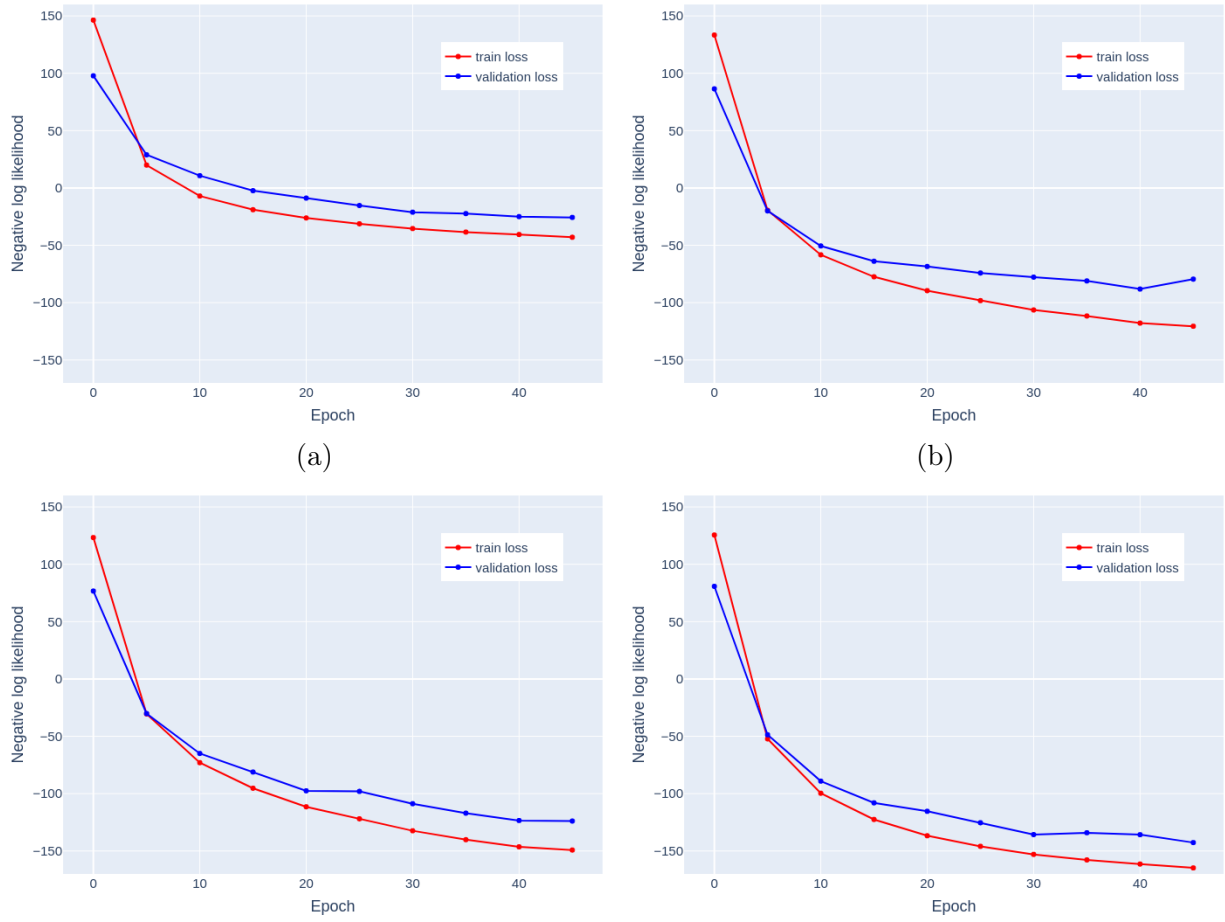


Figure 6.4: Loss scores for varying number of stacked flows. From (6.4a) to (6.4d), the number of flow is increased from 1 to 4. From 1 to 2 steps, the score is improved by a factor of 3. From 2 to 3 steps, the increase is approximately of a factor 2 while between 3 to 4 steps, the score remains of the same order.

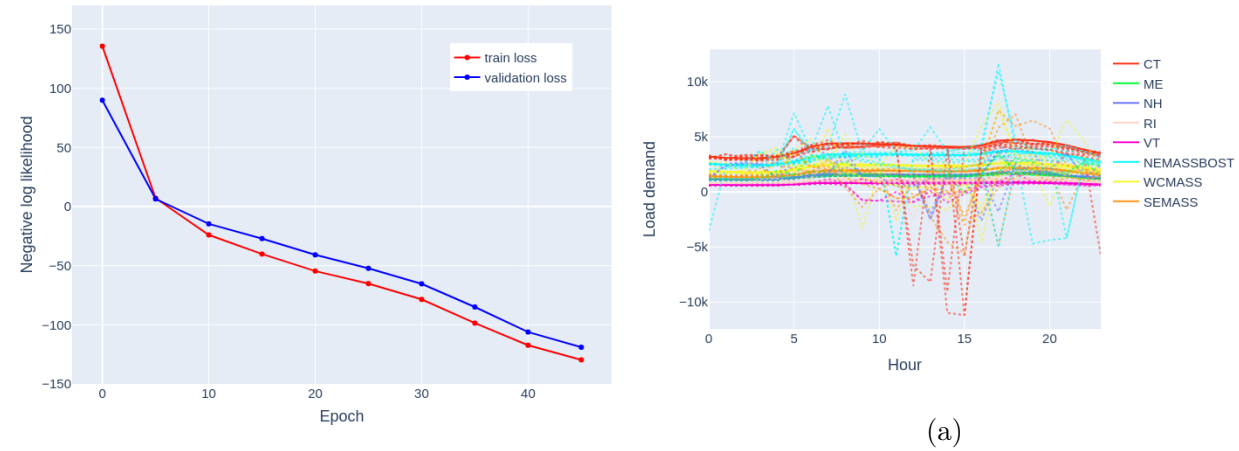
Based on insights provided by the discussion about ANF with affine normalizer, we iterate the search for ANF with monotonic normalizer. Learning rate and weight decay are locked on prescribed values found previously.

Number stack	Conditioner layers	Conditioner output	Normalizer layer	Integration evaluations
1	[200, 200, 200]	5	[100, 100, 100]	5
2	[300, 300, 300]	10	[200, 200, 200]	10
/	[200, 200, 200, 200]	20	[200, 200, 200, 200]	20

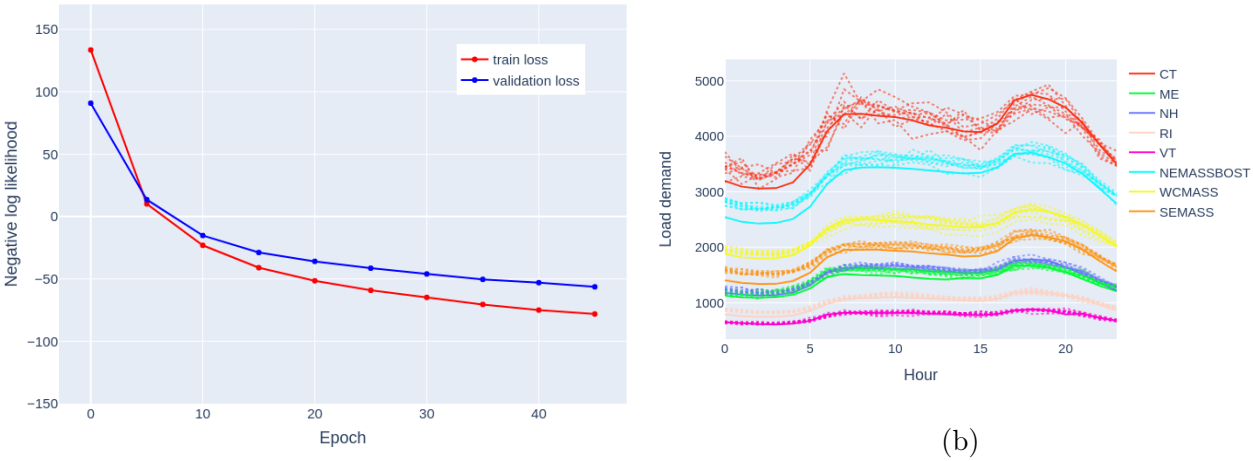
Table 6.2: Summary table of hyperparameter values tested with ANF with monotonic normalizer. Dictionaries of model hyperparameters are created from the permutation of values contained in respective set of hyperparameters. *Conditioner output* refers to the number of conditioning variables used as input to the normalizer. *Integration evaluations* indicates the number of evaluation for the numerical integration part of the normalizer. Search space is limited by the resources available.

Regarding the conditioner MAN, results point to the same conclusion as when combining affine normalizer. It is still preferable to increase the number of units in each layer than increasing the number of layer. Focusing on the integrand MLP used as part of monotonic normalizer, we

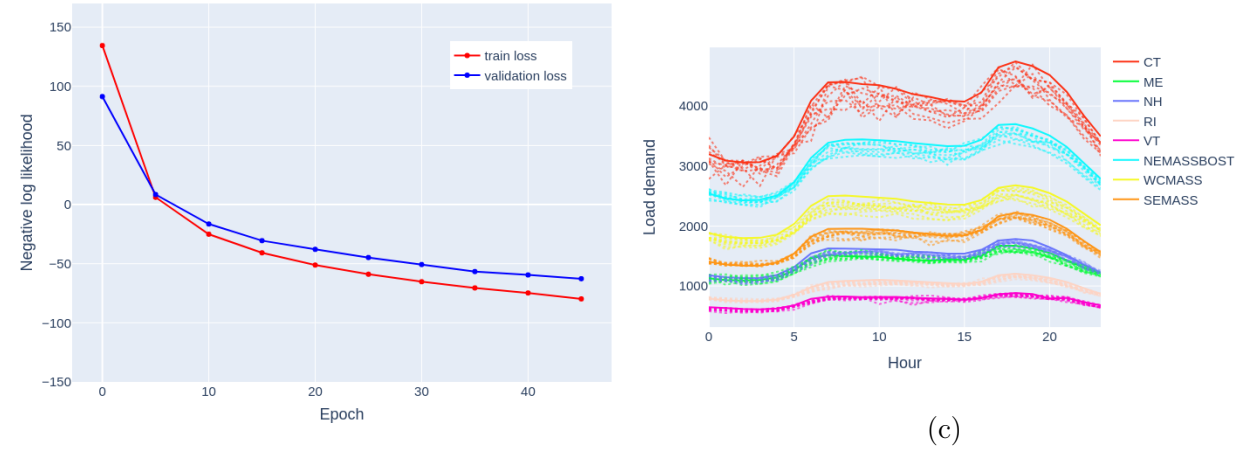
do not face the same symptom as for MAN. Instead deeper networks gain advantage over wider ones. To optimize time resource, we look to minimize the number of evaluation step required to compute the integral. While 5 steps lead to very good result in terms of loss, we find really poor fit when generating scenarios. Using 10 steps or 20 steps results sensitively to the same loss. We thus use 10 integration steps for further analysis. Finally we also record that the normalizer performs equally when having access to a limited number of conditioning values or a higher number. In our search, five values per variable of the random vector was enough to condition the normalizer.



(a)



(b)



(c)

Figure 6.5: Each row displays results for a different number of evaluation steps the numerical integration of the NN. The left plots give the loss scores while the right plots pin up generated scenarios for a given day of the validation set. (6.5a) reflects that with only 5 steps, transformations learned by the normalizer hit accurately the known distributions but are actually non invertible. We reach a sufficient number of evaluations with 10 steps, as 20 steps does not increase the precision of the integral evaluation.

2.2 Graphical normalizing flows

As we highlighted in the previous section, affine normalizer requires at least three stacks of the NF to reach a level of universal density estimator. Yet stacking multiple flows for GNF is not interesting as it would break dependencies set up by the adjacency matrix. GNF then looses its capacity to introduce induced biais in the learning and generating processes. Therefore we advise to combine GNF with monotonic normalizer. As we saw, its capacity as universal density estimator is independent of the number of combined flows but its performance improves with the NNs capacities it relies on. In this section we consider the adjacency matrix as a hyperparameter and explore methods to define an optimal one. Many matrices are candidates to model dependencies between spatio-temporal variables, yet we would like a matrix able to reduce as many spurious relations without loss of information. First we verify GNF with a lower triangular matrix filled with ones reaches similar results as ANF. We compare their loss at training time on Figure 6.6. Surprisingly we find autoregressive like GNF performs better than ANF by a factor 2. This result is probably a consequence of the different architecture of the conditioner. Indeed to apply the mask corresponding to A, the implementation of GNF used in this work is not build over a MAN. Instead d forward passes are performed on a MLP. The ith pass masks the input vector by the line i of A (with $i : 0 \rightarrow d$). The hot encoding of the variable concerned is also passed as context to discriminate the conditioner function c^i. Oppositely to MAN, all neurons in each layer are now available to compute the c^i conditioner outputs. A tradeoff has to be done between memory (higher for ANF conditioner) and time (higher for GNF conditioner).

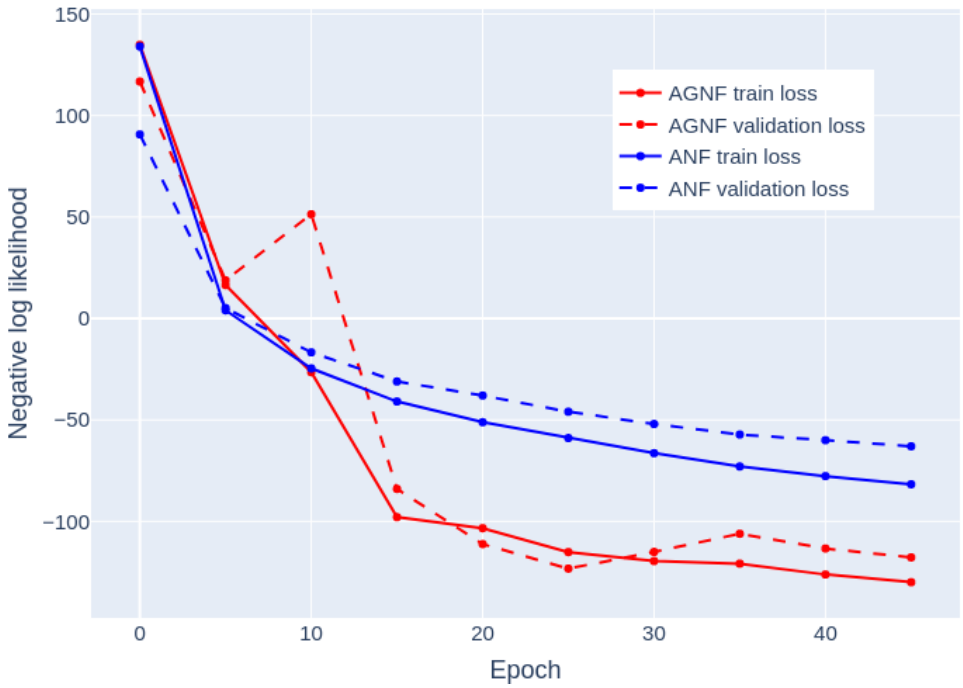


Figure 6.6: Comparison between autoregressive like GNF and autoregressive NF. Both scores didn't converge perfectly yet after 50 epochs but we observe a same trend. GNF achieves a better score, probably due to different architectures of the conditioner.

The best method available to obtain A would be to apply the algorithm proposed by Wehenkel and Louppe [2021] which optimizes conjointly the model parameters and the adjacency matrix (see procedure in chapter 3, under GNF). Yet this method introduces several additional

hyperparameters which require careful fine-tuning. If not set precisely, the learning procedure shows off unstable behavior. Due to limited resources, we decide to skip this approach and propose different heuristics of the adjacency matrix.

Pearson correlation based matrix

The first heuristic limits its search for dependencies between variables of interest to linear correlations. Pearson correlation (PC) is the ration between the covariance of two variables and the product of their standard deviation. Its result is a value between 1 and -1. The Pearson correlation between variables X and Y is denoted $\rho_{X,Y}$ and is defined by

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$
$$\rho_{X,Y} = \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{\mathbb{E}[X^2] - (\mathbb{E}[X])^2}\sqrt{\mathbb{E}[Y^2] - (\mathbb{E}[Y])^2}}$$

In practice we estimate $\rho_{X,Y}$ on the data from the training set,

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

With $\bar{x}$ and $\bar{y}$ being the estimated mean of the respective variables sample.
To actually create the adjacency matrix, we first compute the pair-wise correlation matrix and takes its absolute value-wise. Then we set a threshold and replace all values below it by 0 in the pair-wise correlation matrix and all above by 1. We finally mask the symmetric matrix to obtain a lower triangular matrix as represented on Figure 6.7. This method suggests that variable pairs with correlation values below the threshold are conditionally independent. We define a set of thresholds to test if imposing a varying restrictive structure on the joint probability density model impacts positively the forecast.

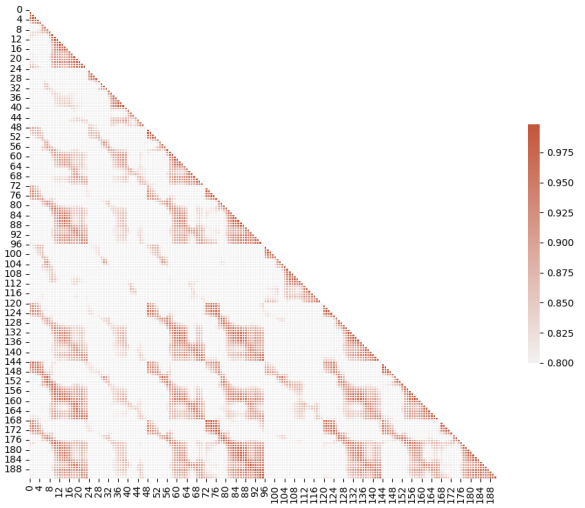
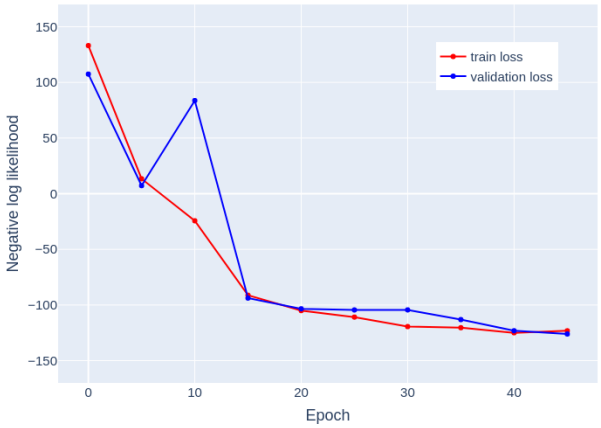


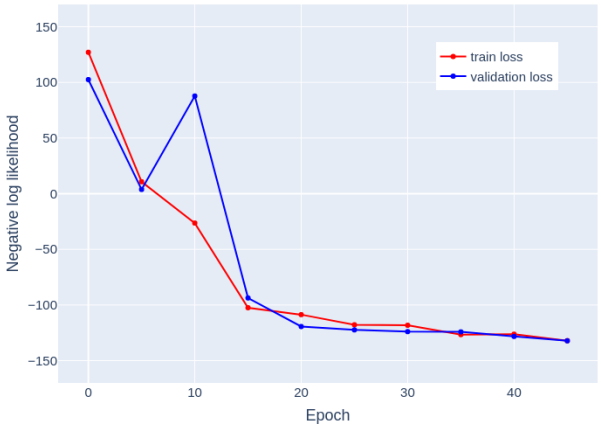
Figure 6.7: Triangular matrix filled with Pearson correlation coefficients. PC coefficients are computed for each pair of variables in the input vector. A similar pattern repeats itself for almost every 24×24 inner matrix. This correspond to correlations of close hours of the days in different zones.

However PC based adjacency matrix may impose more conditional independence between variables than existing in reality. Indeed, we know dynamics of load demand are more complex than linear relationships. Yet PC heuristic would systematically deny any possible dependence to variables that are correlated otherwise than linearly.

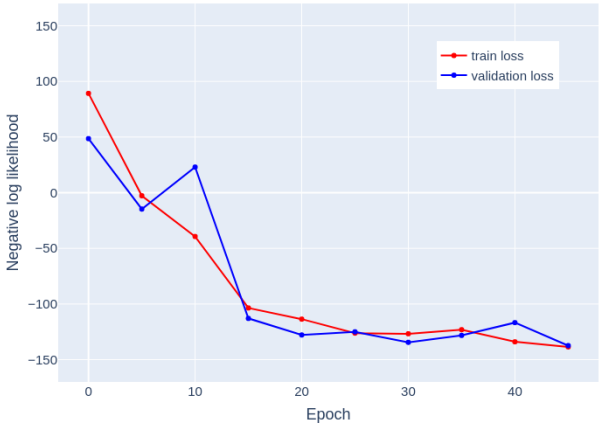
Three different threshold values are tested, **0.2, 0.5, 0.9**. We observe on Figure 6.8 that setting more restrictive conditional independence lead to better loss results than fully autoregressive matrix. Even though PC based heuristic is quite naive, we observe autoregressive factorisation of the joint probability density model, as it is done with ANF, is not optimal in this context. Enforcing specific conditional independence through the adjacency matrix reduces noise from unnecessary information flow.



(a) Threshold at 0.2



(b) Threshold at 0.5



(c) Threshold at 0.9

Figure 6.8: Loss scores of GNF with PC heuristic. Different thresholds are set to compare results of their corresponding adjacency matrices. All the resulting learning converge but the model setting the more conditional independence, with threshold at 0.9, performs sensitively better than the other. At epoch 0, this model already has an advantage over models with less restrictive PC.

Mutual information based matrix

As discussed previously, PC heuristic is limited as it only detects linear correlations. To overcome this problem we propose a second heuristic based on mutual information. First, entropy of a random variable, $H(X)$, measures in *bits* the quantity of information X holds. The more X is predictable, the less it is said to contain information. The mutual information

(MI) between two random variables, $I(X, Y)$, characterizes the quantity of shared information. It measures how much knowing one of those variable reduces the uncertainty about the other. In other words, MI is an estimation of the degree of dependence between two variables. Oppositely to PC, MI takes into account all forms of correlations. MI is closely related to the entropy of the corresponding variables and is defined as,

$$I(X, Y) = H(X) + H(Y) - H(X, Y)$$
$$H(X) = -\mathbb{E}[\log_2 p(X)]$$

Indeed, as represented on Figure 6.9, if we addition two set of information from random variables that overlap, the overlapping part will be present twice. By subtracting the joint set of those variables, then the common part remains once and corresponds to the mutual information.

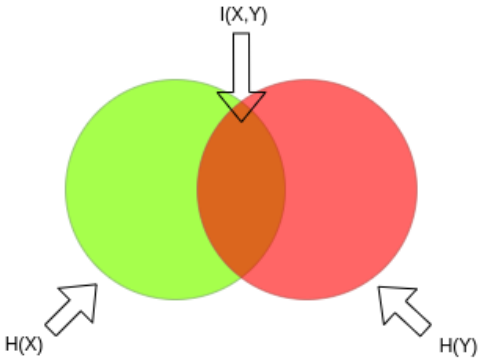


Figure 6.9: Representation of random variables information as sets. The joint entropy, $H(X, Y)$, is the contour area of the joint sets. The mutual information is the overlapping region of the two sets.

Nevertheless computing the mutual information of continuous random variables is non trivial. It would require to evaluate the integrals over X and Y of their marginal probability densities and of their joint probability density.

$$I(X, Y) = \int_X \int_Y P_{X,Y}(x, y) \log_2 \Big(\frac{P_{X,Y}(x, y)}{P_X(x) P_Y(y)} \Big) dxdy$$

To create the heuristic, we instead propose to estimate distributions through discretization and counting of variables values in the training set. We then apply the discrete version of the mutual information,

$$I(X, Y) = \sum_X \sum_Y P_{X,Y}(x, y) \log_2 \Big(\frac{P_{X,Y}(x, y)}{P_X(x) P_Y(y)} \Big)$$

From it, we create a matrix of pair-wise MI values. We set a threshold and replace all values below it by 0 and all others by 1 in the matrix. We finally mask the symmetric matrix to obtain a lower triangular matrix as represented on Figure 6.10. We test different value of MI as threshold. As MI has no upper bound, we rely on its distribution of values in the lower triangular matrix to pick thresholds. We take the median ($q^{MI}_{0.5} = 0.8$), the third quartile ($q^{MI}_{0.75} = 1.1$) and the 0.95 quantile ($q^{MI}_{0.95} = 1.7$).

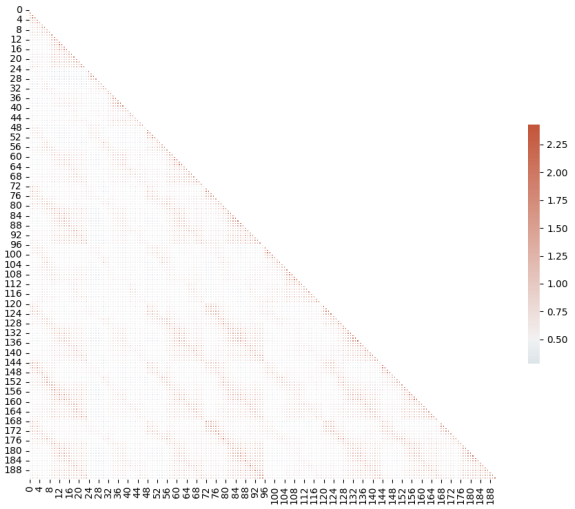
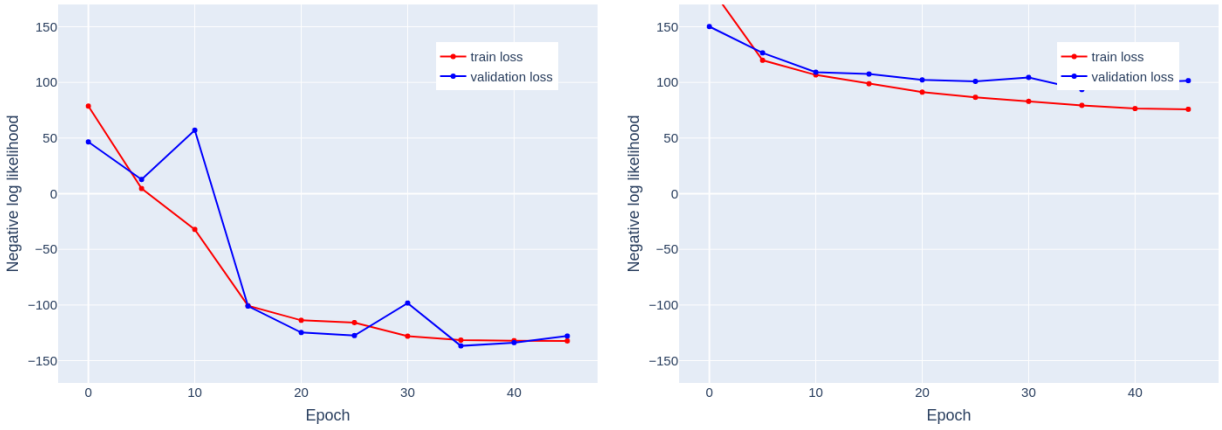


Figure 6.10: Triangular matrix filled with mutual information values. Mutual information values are computed for each pair of variables in the input vector. We observe quite the same patterns than for the PC based matrix meaning linear correlation are probably predominant concerning dependencies.

We underline the important arbitrary choice made on the bounds of bins used to discretize. The number of bins and the number of samples per bin, as well as the bin intervals, impact greatly the form of the discrete distributions. After some tests, we arbitrarily fixed the number of bins to 50 with linear distribution of the thresholds. In a sens we provide a first naive model of the data distribution we aim to forecast that we further refine with GNF.



(a) $I = 0.8$ (b) $I = 1.1$

Figure 6.11

We observe that too restrictive matrices, stating independence from MI values below 1.1 bits as example, result in poor models of the joint probability density (see Figure 6.11b). We remind that even if mutual information is a pertinent measure of dependence between variables, estimating it for continuous real variables is not a trivial task. Choice of discretization and procedure greatly impacts the final output. From about 0.8 bits of MI, we find back loss scores obtained with the PC heuristic (see Figure 6.11a). Differentiating their adjacency matrices in

Figure 6.12, we observe MI based heuristic formulates less independent variables than PC based heuristic with matrices leading to the same minimal loss.

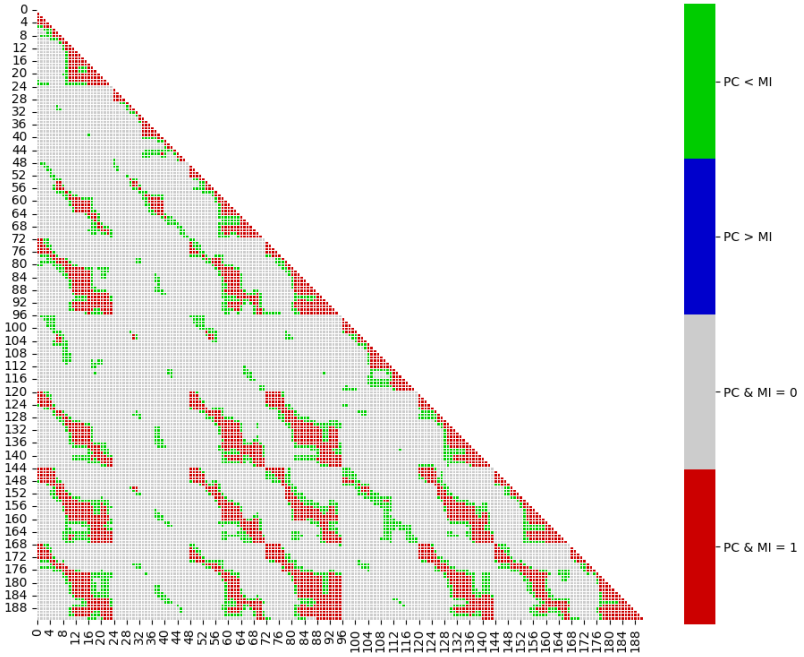


Figure 6.12: Differentiation of the adjacency matrices when $r = 0.9$ for PC and $I = 0.8$ for MI. Grey cells indicate common belief of independences. Greens cells are only considered as independent by PC heuristic.

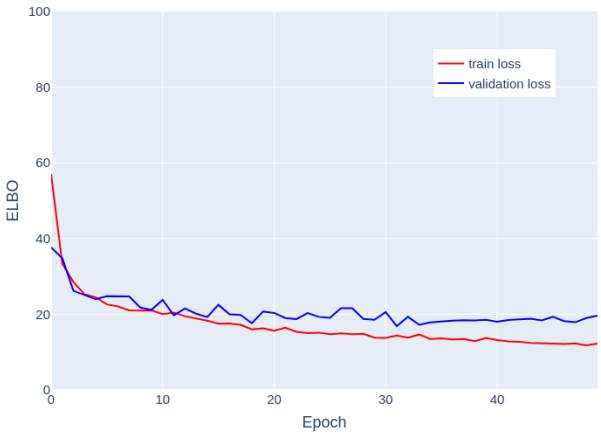
2.3 Variational autoencoder

To be fair we also tune some parameters of the VAEs. We keep the values related to the optimizer as we picked them for ANF and GNF. We specifically tune the number of dimension in latent space, the architecture of the MLP for the encoder and decoder respectively.

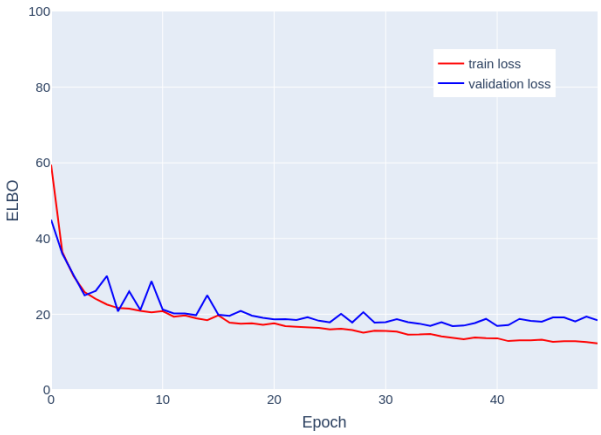
Latent dimension	Encoder layers	Decoder layers
10	[200, 200, 200]	[200, 200, 200]
50	[200, 200, 200, 200]	[200, 200, 200, 200]
100	[300, 300, 300]	[300, 300, 300]

Table 6.3: Summary table of hyperparameters values tested with VAEs. Dictionaries of model hyperparameters are created from the permutation of values contained in respective set of hyperparameters.

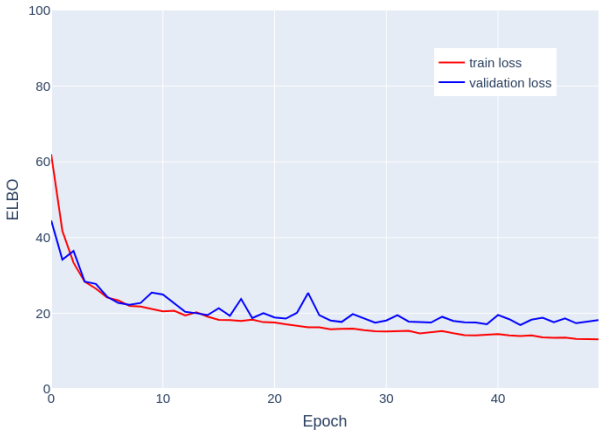
No tuning lead to real gain in term of loss score. For fixed encoder and decoder, the ELBO remains the same when increasing the size of the latent space as observed on Figure 6.13. Yet further exploring VAEs with smaller MLP shows latent space with higher dimension improve slightly the ELBO (see Figure 6.14). But once again no real enhancement of the model is done by changing the MLPs.



(a) $d_z = 10$

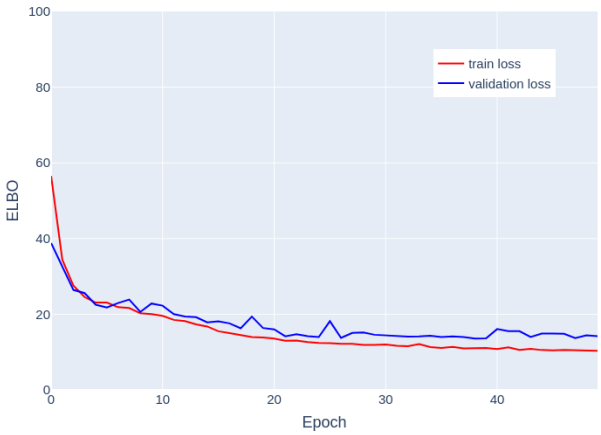


(b) $d_z = 50$

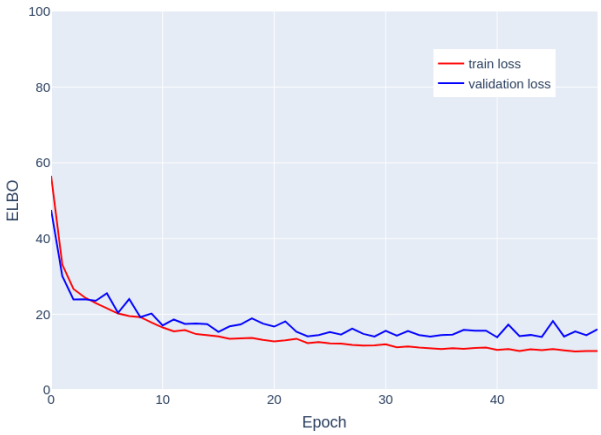


(c) $d_z = 100$

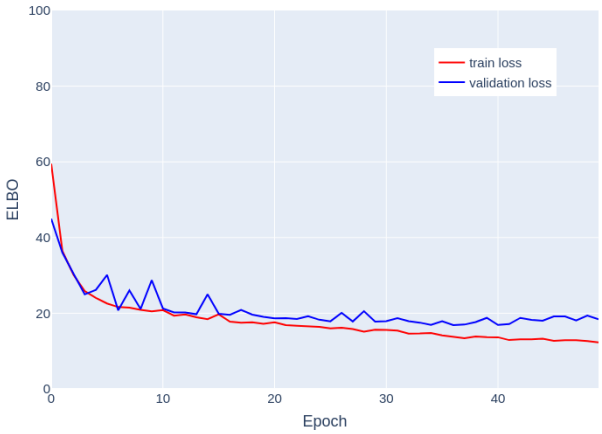
Figure 6.13: ELBO of VAE with varying size of latent space. Encoder and decoder MLP are both fixed with structure [200, 200, 200, 200].



(a) $d_{enc,dec} = [200, 200, 200]$



(b) $d_{enc,dec} = [300, 300, 300]$



(c) $d_{enc,dec} = [200, 200, 200, 200]$

Figure 6.14: ELBO of VAE with varying size of encoder and decoder MLP. Latent space dimension is fixed to 50.

Even if ELBO and likelihood are thightly related, interpreting a comparison of their values is not straightforward and might lead to confusing conclusions. In the next section we pick most performing models in terms of loss value for each method and compare them against various scores and visualizations.

3 Results

In this section, we pick a final model for each method (see table 6.4) and retrain them for 50 epochs further. We then generate scenarios with context of days from the **test set**. Based on these scenarios, we display visualizations of forecasted days, correlations in those forecasts and compute scores to compare. For each of those results, a discussion is developed to explore the behaviors of models, their strengths and shortcomings.

IG									
VAE	enc: [300, 300, 300]		ls: 50		dec: [300, 300, 300]				
ANF	c. MAN: [300, 300, 300]	c.o.: 20	n. MLP: [200, 200, 200, 200]	n.i.: 10					
AR GNF	c. MLP: [300, 300, 300]	c.o.: 20	n. MLP: [200, 200, 200, 200]	n.i.: 10	A: lower triangular				
PC GNF	c. MLP: [300, 300, 300]	c.o.: 20	n. MLP: [200, 200, 200, 200]	n.i.: 10	A (PC threshold): 0.9				
MI GNF	c. MLP: [300, 300, 300]	c.o.: 20	n. MLP: [200, 200, 200, 200]	n.i.: 10	A (MI threshold): 0.8				

Table 6.4: Summary table of models used for results and discussions. c. stands for *conditioner*, c.o. for *conditioner output*. n. stands for *normalizer* and n.i. for *normalizer integration*.

3.1 Days forecast

To begin with we observe appearances of generated scenarios with each method. Two different days from the test set were selected as they seemed representative of the performances. **Ten scenarios** are drawn and are displayed along with the true value of the corresponding zone during the 24 hours of the picked day. It results into $10 \times 8 \times 24 = 1920$ values to evaluate. From Figure 6.15 to Figure 6.20, each color represent a different zone of the electrical grid. Plain lines indicate observed values from test set and dashed lined are generated scenarios with models.

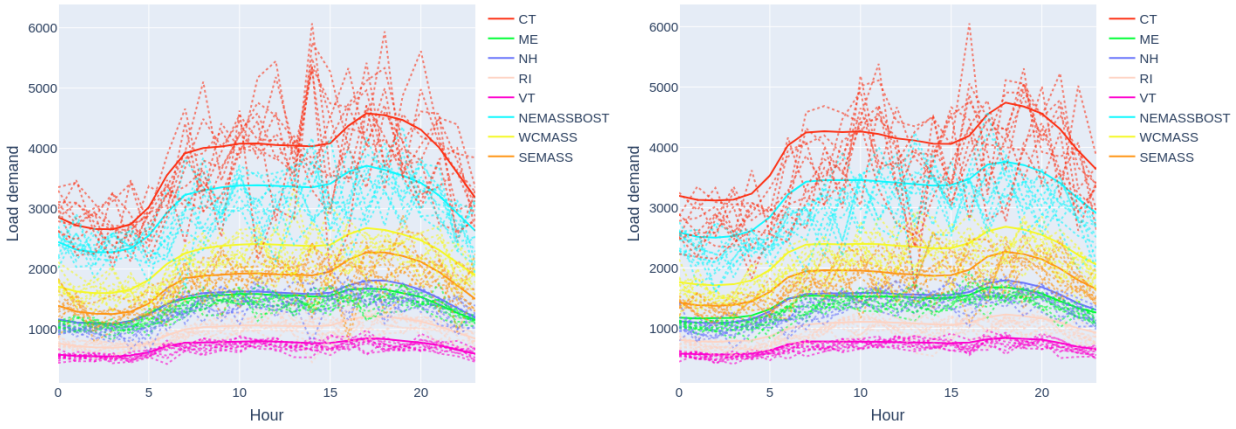


Figure 6.15: Scenarios generated with **independent Gaussians**. Means of samples appear to be well calibrated most of the time on the true values. Variance is however very high. As expected, lack of dependence between variables lead to chaotic prediction of time series.

We propose independent Gaussians (IG) (see Figure 6.15) as benchmark and contrast to other models. It allows to highlight interesting results or drawbacks of other forecasters. As we generate normalized value for each model, creating scenarios for independent Gaussians sums up to draw value from marginal normal distribution $\mathcal{N}(\mu = 0, \sigma = 1)$. It is then enough to shift and scale the values by the mean and the standard deviation from each variable of the training set. The high variance thus directly reflect the wide of the true distribution (at training time) when no dependence between variables nor context (explenatory variables) are taken into

account. Zones **RI** and **VT** are characterized by low absolute variance compared to the other zones (see discussion about demography and economy in chapter 5). Those zones don't bring much added value to the analysis as more information or model capacity won't improve their results by a large margin.

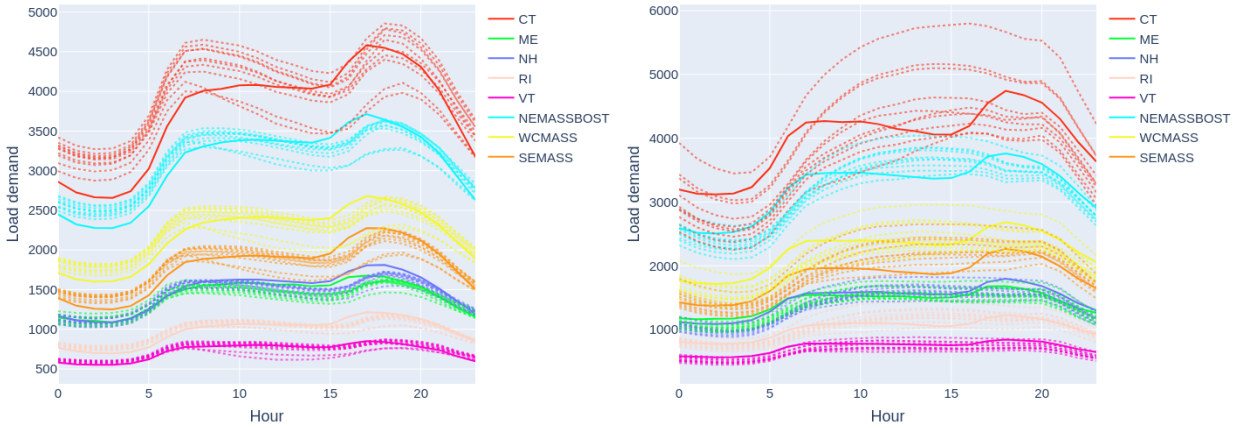


Figure 6.16: Scenarios generated with **VAE**. VAE forecasters create smooth scenarios within each zone in term of timeseries. The set of scenarios sampled from the model seems to respect the probabilistic contract and form coherent forecasts.

When comparing both figures on plot 6.16, we observe VAE forecaster produce highly correlated scenarios. It also seem to correctly respect the probalistic contract, a.k.a. aim to maximize the sharpness of the predictive distributions, subject to calibration. On the left plot, the range of scenarios is quite narrow and true trajectories fall almost during the whole day within the forecast values. On the right plot, scenarios covered a larger range of value but still ensure the true trajectories to be most of the time within the scenarios set. This translate the learning in the model of the uncertainties tied to given context.

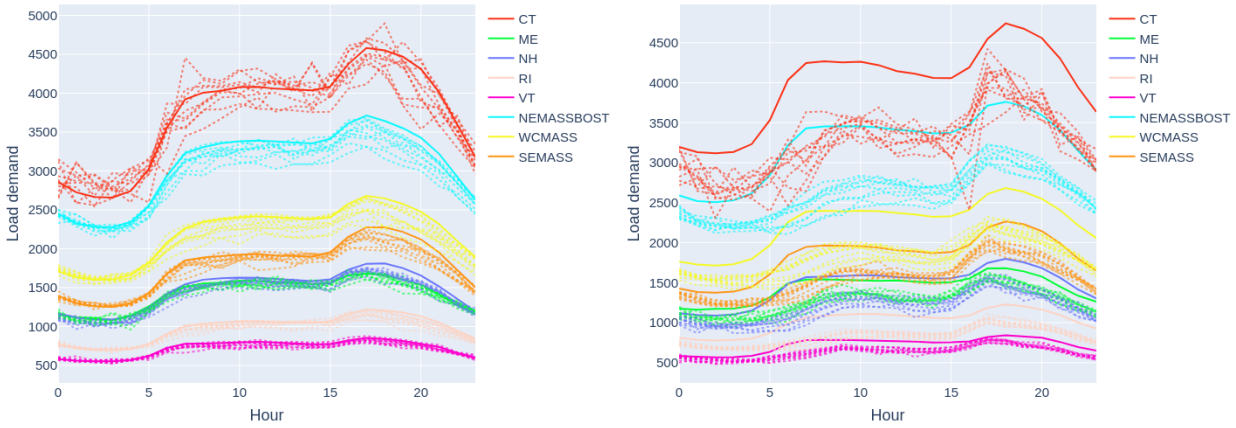


Figure 6.17: Scenarios generated with **ANF**. We observe scenarios do not follow strong correlation between each other nor across timesteps. Forecasts set is very narrow within each zone for most situations. The left plot shows some context lead to ill calibrated forecast when using NF forecaster.

As seen on Figure 6.17, scenarios miss the information about correlation between timesteps while a multidimensional forecaster should be able to reconstruct this information. The architecture of the NF conditioner is probably responsible for this as each conditioning value

is at best handled by 2 neurons in each layer. We impute the behavior of the offset between the mean of the predictions set and the true observation to the same reason. We estimate the normalizer found good set of transformations but still fail due to bad conditioning values. When observed, the offset error is not zone specific but concern the whole vector of predictions.

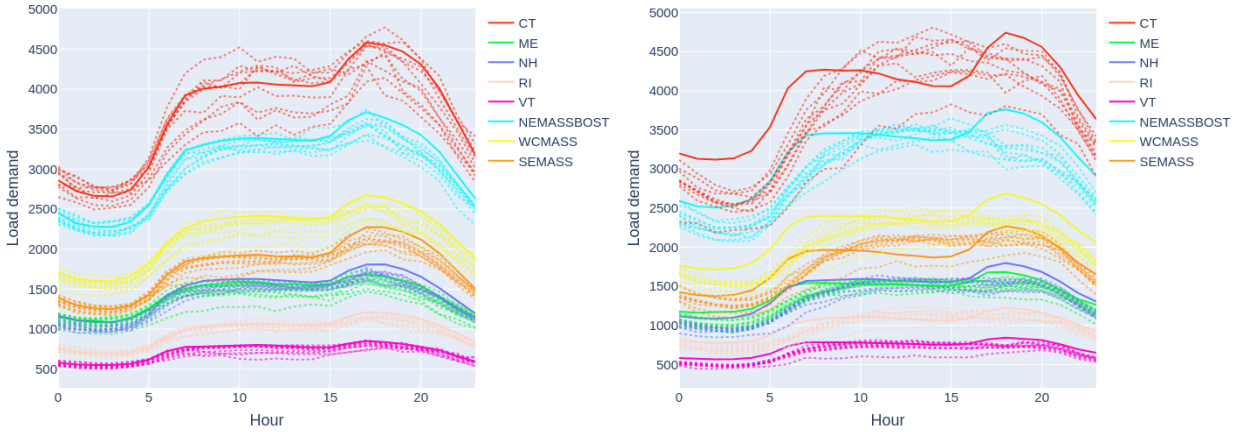


Figure 6.18: Scenarios generated with **autoregressive like GNF**. Scenarios are now highly correlated both between each other but are also coherent along the timesteps. Calibration is acceptable but not perfect for every context.

Comparing behaviors between ANF (Figure 6.17 and AR GNF (Figure 6.18), we spot a different results for two models that were expected to be relatively close as ANF corresponds to ARGNF equipped with a tailored autoregressive adjacency matrix. ARGNF is better at modeling correlations but also generate better fit for the scenarios set. As already discussed in hyperparameters search section, while the normalizer is the same, the conditioner use two different architecture leading to a tradeoff between time and memory usage. Our hyperparameter search concentrated on same order of memory usage, but didn't receive much attention on computation time limitation. As such, ARGNF ended up to be favoured as its procedure use the same order of memory but requires much more passes and thus time.

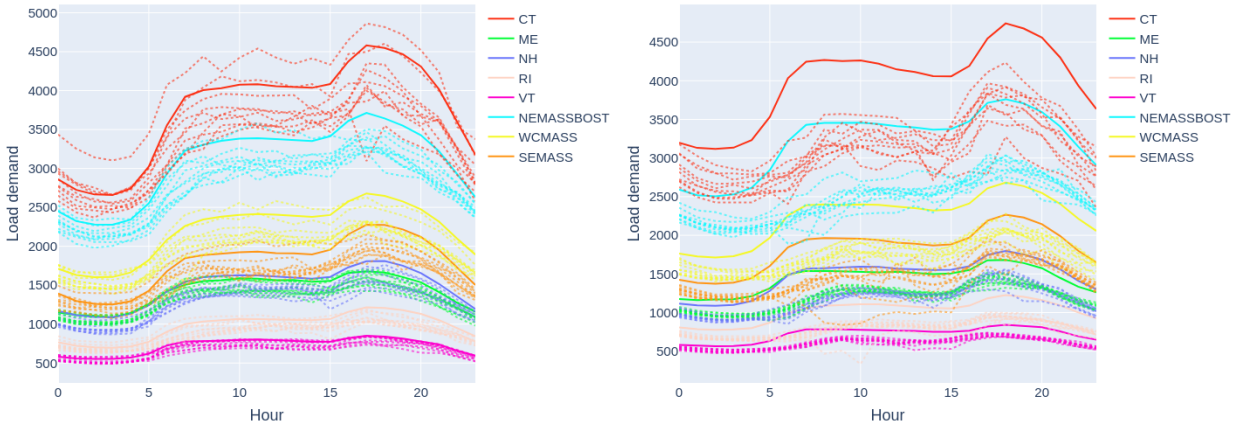


Figure 6.19: Scenarios generated with **PC GNF**. Scenarios are correlated enough but some values are completely off the distribution. On the right plot, we observe again the offset compared to the true observation. Yet forecasted trends follow correctly the observation.

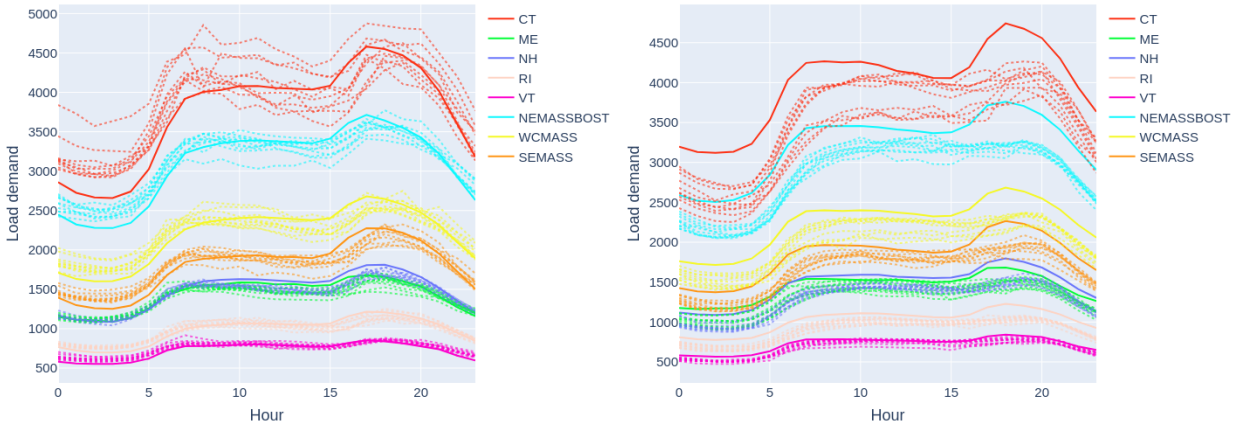


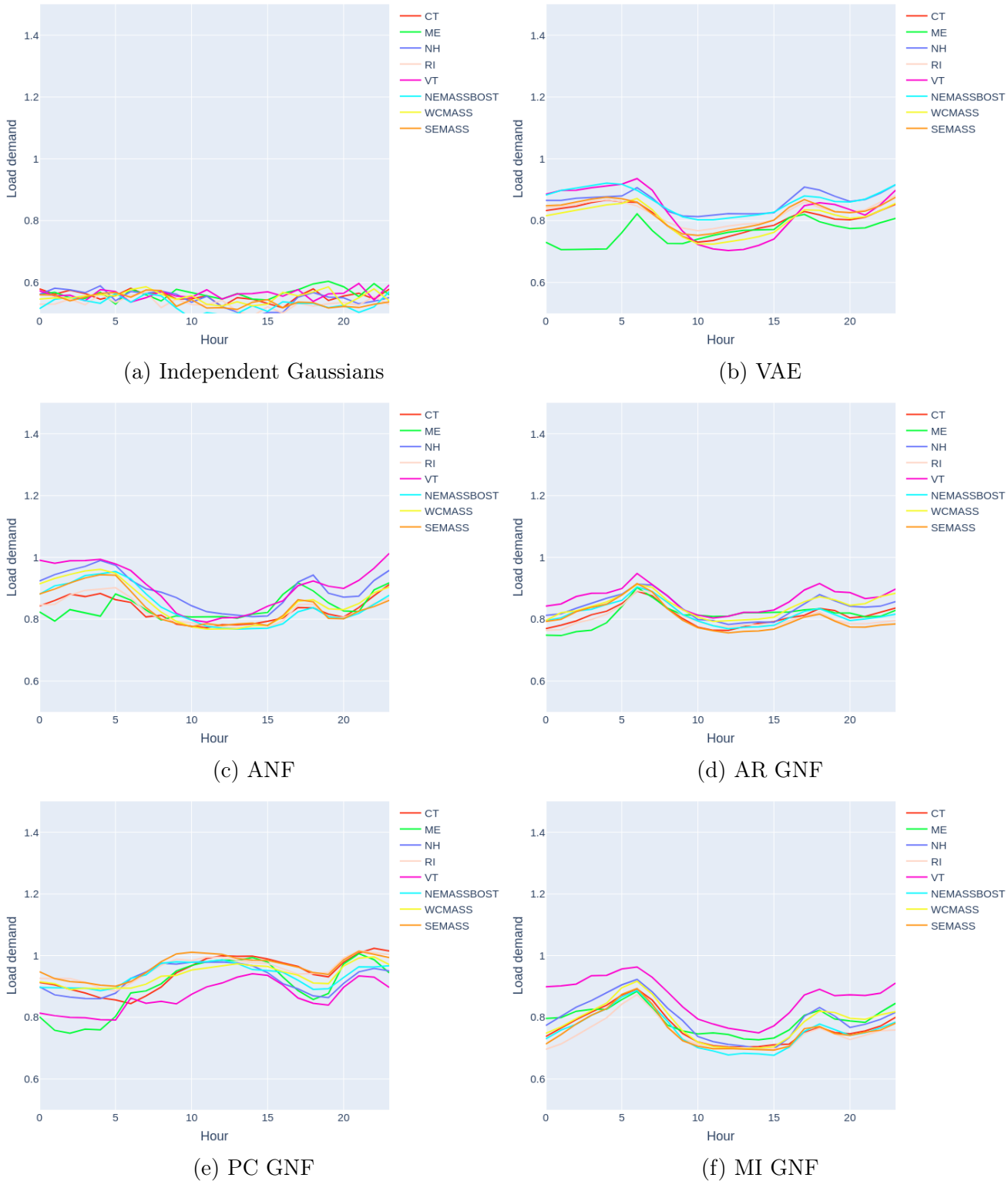
Figure 6.20: Scenarios generated with **MI GNF**. We observe values that seem to be out of distribution. Forecast set calibration is good on the left plot but still don't hit the target for the second day.

We now compare both proposed heuristic against each other. It seems **MI** heuristic improve both sharpness and calibration compared to **PC** heuristic. As discussed, **MI** is supposed to be able to detect a broader range of dependencies between variable. Some independence restriction imposed by **PC** might not be forced with **MI**, leading to better performances of the forecaster. It is interesting to observe GNFs produce relatively smooth scenarios, playing their roles as regulator.

Throughout following paragraphs, we detail forecasts quality and accuracy based on different metrics. We compare intuition obtained through screening batch of scenarios generated with the different models against the measures.

3.2 Scores

CRPS



(a) Independent Gaussians

(b) VAE

(c) ANF

(d) AR GNF

(e) PC GNF

(f) MI GNF

Figure 6.21: Hourly mean CRPS error computed for each zone along the day. Each model is displayed. For most of them, hourly CRPS presents the same patterns with a first peak of error around 6am and a second one around 5pm. **VT** tends to display the highest relative error for almost all models. **NH** does not either obtain a good measure of the marginals.

On Figure 6.21, we observe hourly average CRPS for each zone and each plot represent a

different forecaster. On Figure 6.21a, we notice independent Gaussians scores particularly well on CRPS compared to other models. This result is not surprising as CRPS is more sensible to the calibration than to the sharpness, meaning variance is not as penalized as bias. Moreover CRPS doesn't take into account quality of modeled correlation between dimensions, a parameter that constrain other models. We also spot error signal across timesteps and zones is mainly noise, oppositely to other models displaying certain patterns at given time or zone. VAE, ANF, AR GNF and MI GNF follow quite the same patterns with slight variation in score values. The two error peaks correspond to hours of the day when most people change places from work to home, probably causing larger uncertainties and increasing the metric value. Unexpectedly, PC GNF seems to invert this behavior but has an overall higher CRPS.

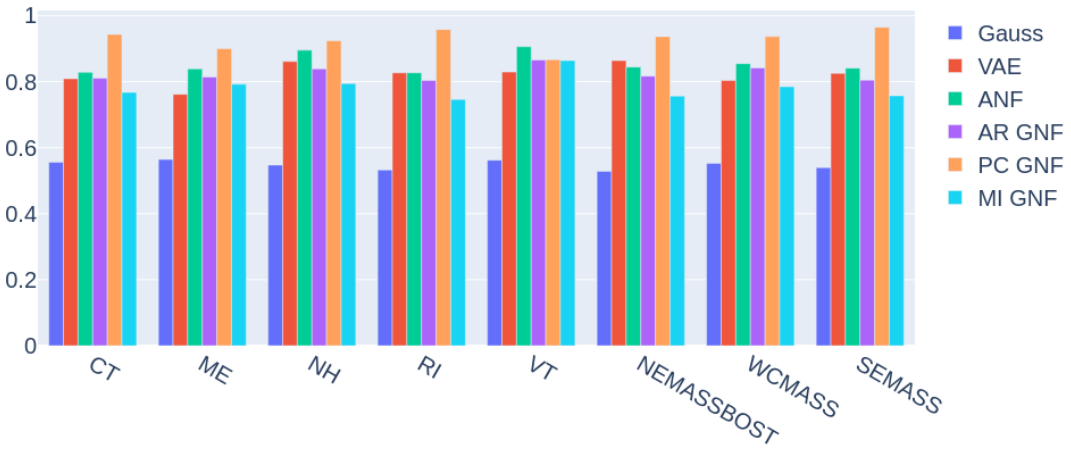


Figure 6.22: Average CRPS per zone for each method. Results have been standardize per zone. *Independent Gaussians* excluded, the best average CRPS score goes to *MI GNF* and the worst correspond to *PC GNF*.

Figure 6.22 showcases a resume of CRPS scores averaged along the day per zone. PC GNF performs clearly worst on marginal densities estimation than the rest of the model, with no regard to the zone. MI GNF scores slightly better than other tested forecasters indicating probable better calibration.

Energy score

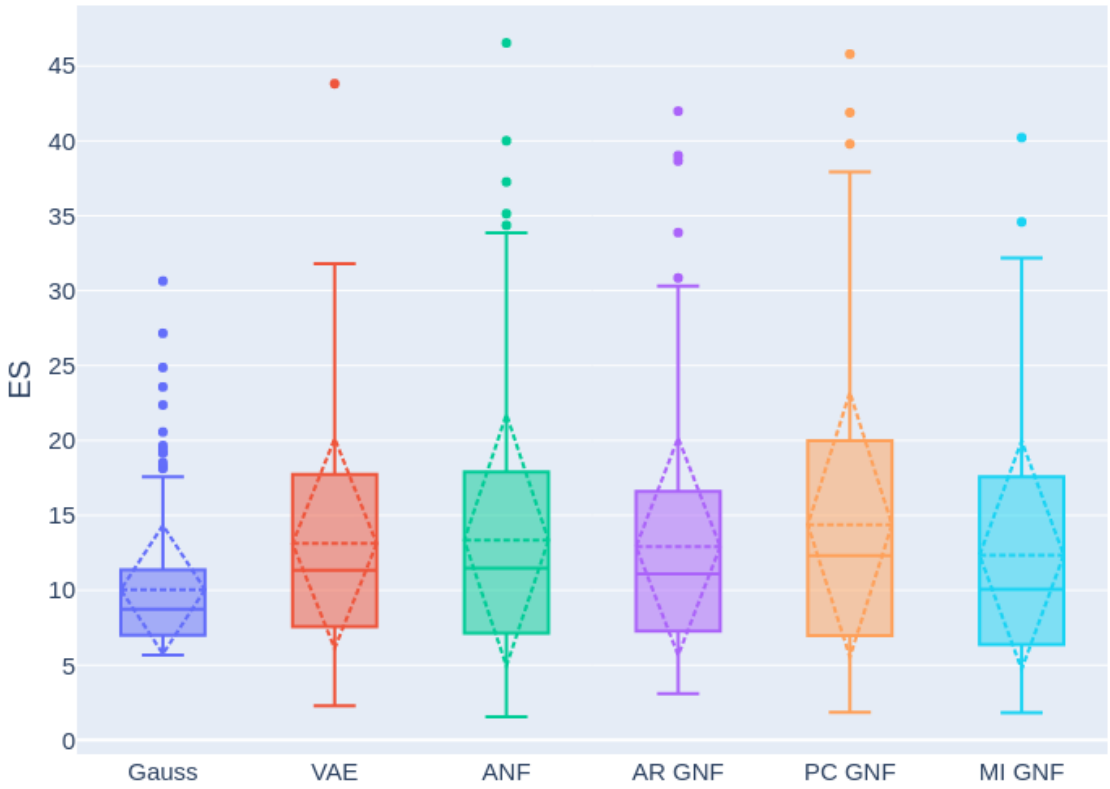


Figure 6.23: Boxplot of the energy score for the different models. Each datapoint is the error of a day from the test set. Horizontal dash line indicates the mean and vertices formed by dash lines on vertical axis represent the variance. Quartiles are displayed with skews and outliers.

Figure 6.23 displays results of energy scores of each method. We first notice the important gap between the median and the mean present for all forecasters. Distributions of errors are relatively narrow but are equipped with large upper skew. This translate bulks of errors around the median but large error for some datapoints that shift the mean towards the upper skew. We even observe multiple outliers. This might indicate some day context lead to poor scenarios generation. Identifying those type of context and/or proposing a special treatment to them for a real environment application might be an important asset. Comparing the models we note independent Gaussians is still the model scoring the best. But as discussed in the metrics section, ES still does not consider correlation. ES strongly reflects the magnitude of errors obtained with CRPS but allows to correctly compute a summary statistic per day and to observe the distribution of error.

Variogram score

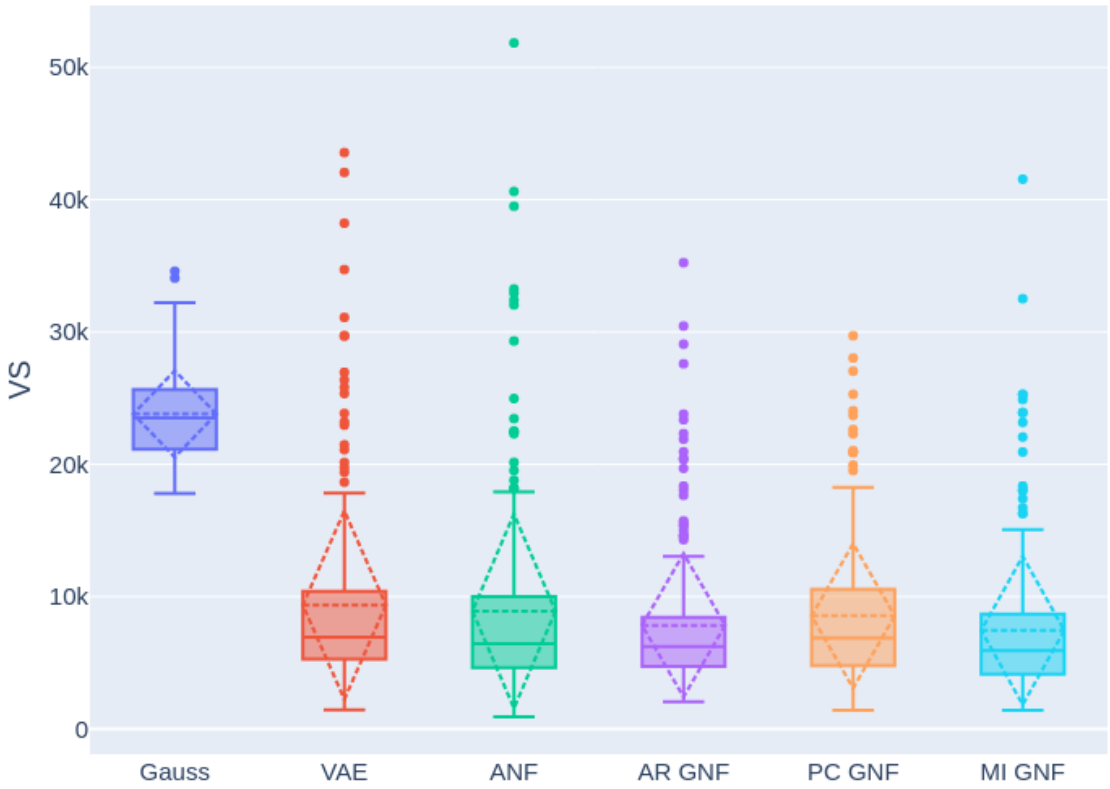


Figure 6.24: Boxplot of the variogram score for the different models. Each datapoint is the error of a day from the test set. Horizontal dash line indicates the mean and vertices formed by dash lines on vertical axis represent the variance. Quartiles are displayed with skews and outliers.

Through VS metric we appreciate a new perspective of the forecasters. When observing Figure 6.24, we directly spot IG got far last in VS compared to other models. In some model (VAE, ANF), the gap between the median and the mean get even more important. GNF based forecasters are not affected by this trend. In particular, AR GNF and MI GNF VS error distributions are quite narrower and appear less skewed than other models. Lower VS means AR GNF and MI GNF generate scenarios that capture better the correlation structure between variables of the targeted multivariate distribution. Absolute high values of error present in this graph are due to the transitions from a zone to another in the concatenated vectors of 192 values. To better understand which model performs better on VS, we desegregate the vectors into zone. Table 6.5 sums up VS per zone for each model. MI GNF achieves the best mean score in every zone. Its variances are also on pair or even lower across all models and zones.

	Gauss	VAE	ANF	AR GNF	PC GNF	MI GNF
CT	402.91 ± 76.8	101.46 ± 102.36	94.45 ± 90.38	80.52 ± 69.13	94.39 ± 85.72	**75.95 ± 68.1**
ME	358.8 ± 76.7	108.56 ± 88.36	114.48 ± 83.62	104.15 ± 78.18	124.44 ± 94.98	**98.5 ± 74.36**
NH	382.86 ± 75.17	107.44 ± 116.7	99.35 ± 95.82	93.28 ± 85.48	99.22 ± 93.19	**82.2 ± 80.31**
RI	411.55 ± 84.04	80.93 ± 74.83	76.78 ± 66.89	66.79 ± 58.84	77.12 ± 68.37	**62.92 ± 55.03**
VT	394.75 ± 83.43	76.23 ± 59.8	79.77 ± 59.01	74.41 ± 51.23	87.01 ± 63.03	**67.91 ± 48.99**
NEMASSBOST	404.6 ± 73.46	86.65 ± 86.07	80.76 ± 70.53	70.58 ± 62.19	75.95 ± 65.43	**66.52 ± 61.7**
WCMASS	398.58 ± 77.64	92.59 ± 93.42	86.73 ± 83.33	76.77 ± 63.78	89.32 ± 85.9	**70.62 ± 64.46**
SEMASS	411.73 ± 80.19	91.1 ± 89.95	89.1 ± 87.26	76.66 ± 72.33	86.05 ± 84.15	**73.29 ± 70.96**

Table 6.5: Variogram scores divided per zone.

A summary table of scores is proposed in table 6.6. Overall those results indicate that MI GNF and its corresponding dependence restrictions achieve to improve slightly better forecasts results. Calibration of marginal distributions (CRPS) along with the correlation formed by generated spatio-temporal series (VS) lead to gain in performance. Set of scenarios generated seems also to be good estimate of the data distribution (ES).

	averaged CRPS	ES	VS
Gauss	0.55	10.05	23799.67
VAE	0.82	13.12	9352.42
ANF	0.85	13.34	8895.8
AR GNF	0.82	12.92	7825.14
PC GNF	0.93	14.38	8535.47
MI GNF	**0.78**	**12.35**	**7433.24**

Table 6.6: Summary table of numerical results. We show MI GNF is above all other models on mean scores. Gaussian scores are displayed but not taken into account.

Chapter 7

Conclusion

During the next decades, climate change will probably be the biggest challenge mankind has to face. Average temperatures rise and its main driving factors, greenhouse gases, call to rethink the way we power our activities. Electricity is an important lever to exploit in order to engage the transition. Indeed electricity is already deeply rooted in our lifestyle, and will continue to expand as proved to be an efficient source of energy. From many possible problems that can be optimized (production, transmission, micro management and so on), this work focuses on the accuracy and quality of day ahead load forecast at a region scale. Day ahead load forecast allow for efficient grid operations such as optimal dispatching, minimal production or accurate electricity pricing wholesale.

Throughout this study, we presented three different generative models to be load forecaster, namely variational autoencoder, autoregressive normalizing flow and graphical normalizing flow. The normalizing flows family being quite new among generative processes, one of this work objective was to introduce them in the frame of power system and forecasting. Furthermore graphical normalizing flow was specifically picked to introduce a new type of inductive bias when learning multivariate distribution of load with NF. Adjacency matrix fed to GNF applies restrictions on forecasted variables dependencies between each other. This matrix can either be crafted by hand with expert knowledge, be derived through an heuristic of the true dependencies structure or with joint optimization along data distributions. The load demand being potentially independent from a zone to another and/or from certain period of the day, we expected the application of GNF to bring some asset to its forecaster.

The forecasters task were to generate 24 successive hours of load demand from 8 zones in the region of New England, USA. To do so chosen generative models learn patterns of the multivariate distribution at training time. At test time, models are asked to draw scenarios from learned distribution to assess forecasters performances. A context vector is also appended to the load vector. The context vector is formed of weather and calendar information and allow for models to condition output scenarios on exogenous variables. To form the context vector and better understant the data, we performed various analysis of the case study. The analysis was decomposed into three parts: the settings of the case study, the exploratory data analysis and the processing of the data to build the input and context vectors. The EDA contains a part about load itself with auto-correlation through time and zones, histograms, etc.. The second part concerns analysis of exogenous variables such as weather and calendar information.

Before assessing forecasters, we performed for each a hyper-parameters search through grid search. Optimizer parameters were tuned along with hyper-parameters search of autoregressive

normalizing flow and were conserved for other models. More importantly for GNF, we also tuned the adjacency matrix as we considered it as an hyper-parameter. Two heuristics, Pearson correlation and mutual information, were picked to estimate the degree of pair-wise dependence within load variables from the training set. To create adjacency matrices, a threshold was associated to the heuristic values. Each value of the heuristic was replaced by 0 if under the defined threshold, or to 1 if above. A list of thresholds were proposed for each heuristic to tune the restricted dependencies in the adjacency matrix. Final models were constructed by selecting the best performing forecasters on the validation set out of the grid search.
During experiments, various metrics were applied to compare final forecasters. Interpreting the different measures, and assessing distributions of scores, we discussed results obtained. First we observed for each forecaster and scores that means were a bit higher than medians. This behavior means a small number of outliers tended to increase the mean compared to the bulk of the scores values. Secondly we noticed ANF and autoregressive GNF did not perform the same while they were expected to. We suspect their different architectures to be responsible of it. To sum up, we concluded mutual information based GNF performed slightly better on median scores than the other forecasters. For most of the metrics it also produced narrower distribution of score. Indeed its marginal distributions were both better at calibration and respecting the probabilistic contract of forecasting. Even with a really small difference compared to autoregressive GNF on energy scores, mutual information based GNF seems to generate scenarios that are better at reconstructing the correlation found between true observations. In the other hand, Pearson correlation based GNF achieves the lowest scores across all forecasters. This shows us *correct* assumptions on independence between variables lead to forecast with better performance.

Under the light of those results, we believe inductive bias induced by GNF actually give a slight advantage to its forecaster. However we hypothesize the case study used was not the best fit to highlight GNF capacity as generative model in presence of independence. As steps within load demand timeseries are often highly dependent, the optimal adjacency matrix is probably not far from the lower triangular matrix, corresponding to the autoregressive GNF. Leads to explore further the application of GNF in power system include the test of other case studies with better known independence, e.g. hierarchy of load in transmission system. Adjacency matrix could then be crafted from the system topology with some independent zones. GNF would allow forecast of upper nodes for which the probabilistic relation with bottom nodes need to be learnt. More generally, concerning GNF applied to timeseries, the modification of the underlying neural network into recurrent neural network could lead to interesting results. Gates of recurrent cells could be blocked by the adjacency matrix. Finally, we encourage practitioners to develop even further the number of metrics used to asses probabilistic forecasters. As we discovered throughout the experiments, various metrics are necessary to grasp partial facet of the forecast quality. Other measures could still be added or developed to increase the forecast comprehension.

Chapter 8

Appendix

Normalizing Flows codes

Codes containing implementation of normalizing flows used in this work are installed from git repository available at `https://github.com/AWehenkel/Normalizing-Flows`.

Variational autoencoders codes

Codes containing implementation of variational autoencoders used in this work are inspired from codes available on git repository at `https://github.com/jonathandumas/generative-models`

EDA and experiments codes

Codes containing EDA and experiments presented in this document are available on git repository at `https://github.com/bendelv/Inge-M2TFE_load-forecast-GNF`.

Bibliography

Agency of Commerce ACCD, The Vermont CEDS Committee Community Development, and the Garnet Consulting Group. 2020 comprehensive economic development strategy. 2016.

Ferran Alet, Maria Bauza, Kenji Kawaguchi, Nurullah Giray Kuru, Tomás Lozano-Pérez, and Leslie Kaelbling. Tailoring: encoding inductive biases by optimizing unsupervised objectives at prediction time. *Advances in Neural Information Processing Systems*, 34, 2021.

Hesham K Alfares and Mohammad Nazeeruddin. Electric load forecasting: literature survey and classification of methods. *International journal of systems science*, 33(1):23–34, 2002.

T. A. Brown. Admissible scoring systems for continuous distributions. 1974.

Nathaniel Charlton and Colin Singleton. A refined parametric model for short term load forecasting. *International Journal of Forecasting*, 30(2):364–368, 2014. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2013.07.003. URL `https://www.sciencedirect.com/science/article/pii/S0169207013000794`.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Abdelkader Dairi, Fouzi Harrou, Ying Sun, and Sofiane Khadraoui. Short-term forecasting of photovoltaic solar power production using variational auto-encoder driven deep learning approach. *Applied Sciences*, 10(23), 2020.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

AnHai Doan, Alon Halevy, and Zachary Ives. 7 - data matching. In *Principles of Data Integration*, pages 173–207. 2012.

Jonathan Dumas, Antoine Wehenkel, Damien Lanaspeze, Bertrand Cornélusse, and Antonio Sutera. A deep generative model for probabilistic energy forecasting in power systems: normalizing flows. *Applied Energy*, 305:117871, 2022.

Abdollah Kavousi Fard and Mohammad-Reza Akbari-Zadeh. A hybrid method based on wavelet, ann and arima model for short-term load forecasting. *Journal of Experimental & Theoretical Artificial Intelligence*, 26(2):167–182, 2014. doi: 10.1080/0952813X.2013.813976.

Krzysztof Gajowniczek and Tomasz Ząbkowski. Short term electricity forecasting using individual smart meter data. *Procedia Computer Science*, 35:589–597, 2014.

Leijiao Ge, Wenlong Liao, Shouxiang Wang, Birgitte Bak-Jensen, and Jayakrishnan Radhakrishna Pillai. Modeling daily load profiles of distribution network for scenario generation using flow-based generative network. *IEEE Access*, 8:77587–77597, 2020.

Tilmann Gneiting. Probabilistic forecasting. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, pages 319–321, 2008.

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

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Stephen R. Green, Christine Simpson, and Jonathan Gair. Gravitational-wave parameter estimation with autoregressive neural network flows. *Phys. Rev. D*, 102:104057, 2020.

George Gross and Francisco D Galiana. Short-term load forecasting. *Proceedings of the IEEE*, 75(12):1558–1573, 1987.

Zhifeng Guo, Kaile Zhou, Xiaoling Zhang, and Shanlin Yang. A deep learning model for short-term power load and probability density forecasting. *Energy*, 160:1186–1200, 2018. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2018.07.090.

Benjamin F Hobbs, Suradet Jitprapaikulsarn, Sreenivas Konda, Vira Chankong, Kenneth A Loparo, and Dominic J Maratukulam. Analysis of the value for unit commitment of improved load forecasts. *IEEE Transactions on Power Systems*, 14(4):1342–1348, 1999.

Tao Hong. *Short term electric load forecasting*. North Carolina State University, 2010.

Tao Hong. Energy Forecasting: Past, Present, and Future. *Foresight: The International Journal of Applied Forecasting*, (32):43–48, 2014.

Tao Hong and Shu Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938, 2016. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2015.11.011. URL `https://www.sciencedirect.com/science/article/pii/S0169207015001508`.

Wei Hu, Yong Min, Yifan Zhou, and Qiuyu Lu. Wind power forecasting errors modelling approach considering temporal and spatial dependence. *Journal of Modern Power Systems and Clean Energy*, 5(3):489–498, 2017.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR, 2018.

Andrej Karpathy and Pieter Abbeel. Generative models. `https://openai.com/blog/generative-models/`. Accessed: 2022-03-13.

Abdolrahman Khoshrou and Eric J. Pauwels. Short-term scenario-based probabilistic load forecasting: A data-driven approach. *Applied Energy*, 238:1258–1268, 2019. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2019.01.155. URL `https://www.sciencedirect.com/science/article/pii/S0306261919301412`.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. doi: 10.1109/TPAMI.2020.2992934.

Jian Lan and Qinglai Guo. Demand side data generating based on conditional generative adversarial networks. *Energy Procedia*, 152:1188–1193, 10 2018.

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

James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.

Juan M Morales, Antonio J Conejo, Henrik Madsen, Pierre Pinson, and Marco Zugno. Renewable energy sources—modeling and forecasting. In *Integrating Renewables in Electricity Markets*, pages 15–56. Springer, 2014.

Allan H Murphy and Robert L Winkler. A general framework for forecast verification. *Monthly weather review*, 115(7):1330–1338, 1987.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Kenneth R. Mylne. Decision-making from probability forecasts based on forecast value. *Meteorological Applications*, 9(3):307–315, 2002.

Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

TN Palmer, F Molteni, R Mureau, R Buizza, P Chapelet, and J Tribbia. Ensemble prediction. In *Proc. ECMWF Seminar on Validation of models over Europe*, volume 1, pages 21–66. European Centre for Medium-Range Weather Forecasts Reading, United Kingdom, 1993.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Martin Ringsquandl, Houssem Sellami, Marcel Hildebrandt, Dagmar Beyer, Sylwia Henselmeyer, Sebastian Weber, and Mitchell Joblin. Power to the relational inductive bias: Graph neural networks in electrical power grids. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1538–1547, 2021.

Hossein Sangrody and Ning Zhou. An initial study on load forecasting considering economic factors. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2016.

PJ Santos, AG Martins, and AJ Pires. Designing the input vector to ann-based models for short-term load forecast in electricity distribution systems. *International Journal of Electrical Power & Energy Systems*, 29(4):338–347, 2007.

Michael Scheuerer and Thomas M Hamill. Variogram-based proper scoring rules for probabilistic forecasts of multivariate quantities. *Monthly Weather Review*, 143(4):1321–1334, 2015.

Heng Shi, Minghao Xu, and Ran Li. Deep learning for household load forecasting—a novel pooling deep rnn. *IEEE Transactions on Smart Grid*, 9(5):5271–5280, 2017.

Esteban G Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

MT Taghavifard, K Khalili Damghani, and R Tavakkoli Moghaddam. Decision making under uncertain and risky situations. In *Enterprise Risk Management Symposium Monograph Society of Actuaries*, 2009.

Maxime Taillardat, Olivier Mestre, Michaël Zamo, and Philippe Naveau. Calibrated ensemble forecasts using quantile regression forests and ensemble model output statistics. *Monthly Weather Review*, 144(6):2375–2393, 2016.

Jean-François Toubeau, Jérémie Bottieau, François Vallée, and Zacharie De Grève. Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. *IEEE Transactions on Power Systems*, 34(2):1203–1215, 2018.

Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in neural information processing systems*, 32, 2019.

Antoine Wehenkel and Gilles Louppe. You say normalizing flows I see bayesian networks. *CoRR*, abs/2006.00866, 2020. URL `https://arxiv.org/abs/2006.00866`.

Antoine Wehenkel and Gilles Louppe. Graphical normalizing flows. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 37–45, 13–15 Apr 2021.

Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5485–5493, 2017.

Ran Yuan, Bo Wang, Zhixin Mao, and Junzo Watada. Multi-objective wind power scenario forecasting based on pg-gan. *Energy*, 226:120379, 2021.

Michaël Zamo and Philippe Naveau. Estimation of the Continuous Ranked Probability Score with Limited Information and Applications to Ensemble Weather Forecasts. *Mathematical Geosciences*, 50(2):209–234, 2018. URL `https://hal.archives-ouvertes.fr/hal-02976423`.