Master thesis : Denoising diffusion probabilistic models applied to energy forecasting in power systems

Auteur : Hernandez Capel, Esteban
Promoteur(s) : Cornélusse, Bertrand; Dumas, Jonathan
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"
Année académique : 2021-2022
URI/URL : http://hdl.handle.net/2268.2/15989

Master thesis submitted in fulfillment of the requirements for the Degree of Master of Science (MSc) in Computer Science and Engineering

Denoising diffusion probabilistic models applied to energy forecasting in power systems

Author: Esteban Hernandez Capel
Supervisors: Bertrand Cornélusse and Jonathan Dumas

University of Liège - Faculty of Applied Sciences

Academic year 2021 - 2022

Abstract

Transition towards a carbon-neutral society by 2050 is one of the greatest challenges of this century. This goal will necessarily lead to a progressive increase in the part of renewable energy in the global energy mix. However, renewable energies are much more subject to uncertainty than conventional power plants. This uncertainty raises new challenges for the integration of renewable energy in the energy mix. Probabilistic forecasting has emerged as a solution to some of those problems as it provides a way to reduce this uncertainty.

This thesis aims to apply a new deep learning approach to the task of probabilistic forecasting. It is based on *denoising diffusion probabilistic models* (DDPM), a recent type of deep generative model. This new type of method has recently shown outstanding results with image generation. A lot of focus has been given to those models by the computer vision community. On the other hand, really few have been given for other types of applications such as time-series forecasting, and they have not yet been applied in the power system community at all.

In this thesis, the first implementation of DDPM for conditional probabilistic forecasting applied to power systems application is presented. Then, a demonstration of the competitiveness of the developed method is realized. This is done by comparing the quality and the value of the predictions with other state-of-the-art deep generative methods, namely, generative adversarial networks (GAN), variational auto-encoder (VAE), and normalizing flows (NF). One big advantage of the methods implemented through this thesis is the fact that they are able to deal with conditional data. The forecasts are weather-based forecasts and depend on external conditions instead of just relying on historical values. The empirical comparisons are realized across three different datasets from the Global Energy Forecasting Competition 2014. The assessment considered the quality of the generated forecasts as well as the actual value of using them. This thesis shows that not only DDPMs are competitive with other state-of-the-art deep generative models, but they are able to consistently outperform them.

Acknowledgments

First of all, I would like to thank Jonathan Dumas for his support and feedbacks during the year, as well as his availability whenever I was in doubt. This thesis is the result of a lot of discussions with him and it would not have been possible without him. I would also like to thank Prof. Bertrand Cornélusse for introducing me to the field of power systems a couple of years ago and for letting me enter the micro-grid team.

I also would like to thank Prof. Pierre Geurts, Prof. Gilles Louppe, Prof. Louis Wehenkel, Prof. Damien Ernst, and Prof. Ashwin Ittoo for their respective courses related to machine learning that helped me improve in that field and push my limits.

Finally, I would also like to thank my family, my girlfriend and my friends for their support, help, and availability during this thesis, and also during my entire academic journey in general.

To all those people: Thank you !

Contents

Chapter 1

Introduction

In order to mitigate climate change and reduce the amount of carbon emitted by human activity, the share of renewable generation in the energy mix will increase (M. Allen, 2019). However, renewable energy generation comes with uncertainty. It leads to increasing the challenges linked to operational predictability of modern power systems (Morales González et al., 2014). To overcome this issue, the development and deployment of forecasting techniques is essential. They allow to reduce this uncertainty and take better decisions (De Gooijer and Hyndman, 2006). The forecasts serve as input for decision-making tools. Namely, some optimization algorithms. Indeed, knowing the future of load or the solar or wind power generation in advance can help to take optimized decisions.

There exist several different forecasting families. They are mainly separated into two types: *point forecast* and *probabilistic forecast*. The advantage of probabilistic forecasting over point forecasting is that it can represent the uncertainty of the predictions. Both probabilistic and point forecast can be done with different techniques, from statistical methods to machine learning, as well as deep learning based methods.

In this thesis, we will explore *deep generative models* for the task of probabilistic forecasting. Indeed, the fact that renewable energy production are stochastic production units makes probabilistic forecast more suited than point forecast as one needs the additional information on the uncertainty for better decision making.

Deep generative models use neural networks to approximate high-dimensional probability distributions. Particularly, for this thesis, we will focus on the applicability of a new promising method in deep generative modeling called *denoising diffusion probabilistic models* (DDPM) (Sohl-Dickstein et al., 2015). Those methods have shown impressive results in the computer vision community (Dhariwal and Nichol, 2021) as well as in the natural language processing community (Kong et al., 2020; Popov et al., 2021), but have not yet been applied to power system applications, and rarely to time series in general (Rasul et al., 2021).

Deep generative models directly learn a generative process of the data. They are suited for probabilistic forecasting as one can generate a forecast in the form of a Monte-Carlo sample. Some of those methods have already shown effectiveness for the computation of probabilistic forecasts and received some attention from the power system community. For this thesis, we will focus on and compare the developed DDPM

to *generative adversarial networks* (GANs) (Goodfellow et al., 2014), *variational auto-encoders* (VAEs) (Kingma and Welling, 2013) and *Normalizing flows* (NFs) (Rezende and Mohamed, 2016). A general overview and comparison of the different deep generative models is proposed by both (Ruthotto and Haber, 2021) and (Bond-Taylor et al., 2021).

1.1 Related work on Deep generative models applied to power systems

A GAN has been introduced in (Chen et al., 2017) to produce wind and PV scenarios based on historical observations. It has been shown to outperform Gaussian-copula. Other scenarios generation GAN have been proposed in (Chen et al., 2018) and (Yuan et al., 2021). Following the improvements of GANs brought by other fields such as computer vision for instance, improved versions of GANs have also been proposed by (Jiang et al., 2021) and (Wang et al., 2020b) for wind and load scenarios generation. However, none of them made use on conditional information.

A VAE developed by Zhanga et al (Zhanga et al., 2018) has been shown to be able to capture the unstable behavior of wind and PV power generation. It has achieved good results on several metrics with really low computing power consumption. It has been compared to classical deep-learning methods such as Recurrent neural networks (RNN) [1] in (Dairi et al., 2020). It has been shown that the VAE outperforms the other methods in a consistent way. An improved version of VAEs has also been proposed by (Qi et al., 2020).

Finally, when it comes to normalizing flows, Ge et al. (2020) proposed a NF for daily load forecasting and compared it with GAN and VAE. However, most of the previous studies suffer from not using conditional weather information to improve the quality of the forecasts. Building on top of that, (Dumas et al., 2021) presented Normalizing flows for conditional probabilistic forecasting in power systems. Their models conditioned the output predictions to some weather forecast data. Furthermore, their study provides an empirical comparison between NFs, GANs, and VAEs.

1.2 Research gap and contribution

From there one have identified several research gaps. First, DDPM have never been introduced to the power system community. Then, most of the existing studies on generative models applied to power systems do not use conditional weather information to improve the forecast quality, and did only used past observations.

The main **contributions** of this thesis are :

1. For the power system community, provide an implementation of a new deep generative method able to handle conditional weather information to produce weather-

[1] Recurrent neural networks are a type of neural network commonly used with time series data, and widely used in power system applications. They are able to capture sequential patterns in the data.

based probabilistic forecasting for the load, PV generation, and wind power generation.

2. For the deep learning community, show another application where DDPM can compete with state-of-the-art deep generative models such as GANs, VAEs, and NFs.

3. Demonstrate the competitiveness of DDPM with respect to state-of-the-art models such as GANs, VAEs, and NFs.

4. Provide a fair comparison between those four deep generative models for both the quality and the value of the forecast. The quality assessment is based on complementary metrics and the value assessment is based on a simple case study taken from the methodology of (Dumas et al., 2021). The comparisons are performed thanks to the Global Energy Forecasting Competition 2014 (GEFCom 2014) datasets (Hong et al., 2016).

1.3 Organization

Chapter 2 gives some background and context about probabilistic forecasting in power systems and deep generative modeling in general. Then, chapter 3 and 4 give the formalization and architecture of the developed model. Chapter 5 describes thoroughly the state-of-the-art model used for the comparison with the developed DDPM. Afterward, chapter 6 elaborates on the methodology of evaluation of the models. Chapter 7 develop on some experiment realized on DDPM. Chapter 8 shows the results for both the quality and the value and compares the four deep generative models. Finally, chapter 9 provides limitations, potential improvements, and concludes the work. A graphical view of the overall methodology of the thesis is provided on figure 1.1.

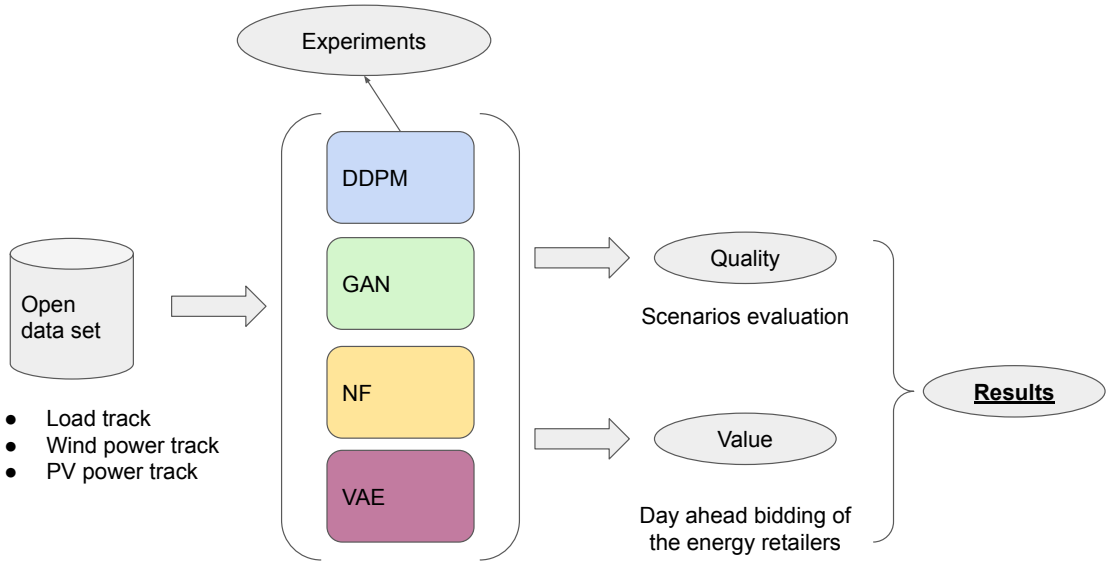


Figure 1.1: High level overview of the overall framework of the thesis.

Chapter 2

Background and problem statement

This section provides a problem statement of probabilistic forecasting through the lens of power systems. It also presents the deep generative models that will be used as comparison for the DDPM.

2.1 Probabilistic forecasting

Forecasting techniques can be separated into two categories: point forecast and probabilistic forecast. The former is widely used in the industry. But, it does not model the uncertainty of the predictions, it simply outputs the value for the quantity to be forecasted. However, modeling the uncertainty of the predictions is of great importance for any decision-making process as stated by Morales González et al. (2014). Indeed, having access to information about uncertainty would allow the deciders to balance the acceptable level of risk leading to better expected-return than with point forecast. Probabilistic forecast (Hong and Fan, 2016) aims at capturing and delivering information about the uncertainty, for instance by predicting the distribution over all possible realization of the random variable of interest (Gneiting, 2008; Hong and Fan, 2016), instead of one single value.

Capturing this uncertainty can be done in several ways. A naive way could be to have a deterministic model and add a residual error to its predictions in order to predict an interval. Another way could be to predict a complete description of a density function (cumulative density function or probability density function). This thesis will focus on *scenarios generation*, a subset of the probabilistic forecasting family. Its goal is to capture the uncertainty of the random variable of interest by generating a succession of its possible outcomes. For this thesis, we will explore load, photovoltaic (PV), and wind power generation forecasting.

Probabilistic forecasting can be classified into two different families based on how it is performed: *statistical methods* and *machine learning based methods*. Each presents some advantages over the other. On one hand, machine learning models are more robust and more user-friendly. They also are better for dealing with the possible non-linearity present in the data. But they can sometimes be hard to interpret. On the other hand, statistical methods are easily interpretable. They learn a relationship between several explanatory variables and a target variable (De Gooijer and Hyndman, 2006). However, they require more expert knowledge in order to formulate the interaction between the

variables. In general, statistical methods rely on strong assumptions which makes it hard for them to model underlying stochastic processes (Dumas et al., 2021). This thesis will mainly focus on machine learning based probabilistic forecasting and in particular deep learning methods.

2.2 Conditional multi output forecasting

The task we are interested in this work is *multi-output forecasting*. Formally, let us have a dataset $\mathcal{D} = \{\mathbf{x}^i, \mathbf{c}^i\}_{i=1}^N$ of N i.i.d sample from a joint distribution $p(\mathbf{x}, \mathbf{c})$ where $\mathbf{x}$ and $\mathbf{c}$ comes from two random variables X and C. X is the variable of interest, for instance, the load, the PV generation, or the wind generation, and C is a condition variable, such as weather forecasts for instance. Both have T periods per day,

$$\mathbf{c}^i := \left[c_1^i, \ldots, c_T^i\right]^\top \in \mathbb{R}^T \text{ sand } \mathbf{x}^i := \left[x_1^i, \ldots, x_T^i\right]^\top \in \mathbb{R}^T.$$

Our goal is to develop a model that generates multi-outputs weather based scenarios $\hat{\mathbf{x}} \in \mathbb{R}^T$ distributed under $p(\mathbf{x}|\mathbf{c})$. In this work, we will use deep generative models $p_\theta(.)$ whose purpose is to generate synthetic but realistic data $\hat{\mathbf{x}} \sim p_\theta(\mathbf{x}, \mathbf{c})$. We want $p_\theta(\mathbf{x}, \mathbf{c})$ to be as close to the real unknown distribution $p(\mathbf{x}, \mathbf{c})$. We are interested in models that compute sets of scenarios on a day-ahead basis.

2.3 Deep generative models

A generative model is a probabilistic model whose goal is to generate synthetic but realistic high-dimensional data. It serves as a simulator of the real data. Deep generative models use deep neural networks to learn the distribution of the data in order to achieve this goal. Deep generative models have received a lot of interest in the last few years. This has been made possible thanks to the development of new generative model techniques and frameworks, the increasing availability of data, as well as the application of promising neural network architectures to the task of generative modeling.

There exist several frameworks when it comes to deep generative modeling. The two most famous methods are *generative adversarial networks* (GAN) (Goodfellow et al., 2014) and *variational auto-encoder* (VAE) (Kingma and Welling, 2013). More recently, new classes of deep generative models have emerged. Among them, *normalizing flows* (NF) (Rezende and Mohamed, 2016), and more recently, *denoising diffusion probabilistic models* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020).

Deep generative models are suited for the task of probabilistic forecasting. Indeed, our goal is to develop a conditional distribution of the data given the conditional weather forecast in order to generate scenarios. Learning a distribution is exactly the purpose of deep generative models as long as data are available. Finally, one can derive quantiles from the scenarios, those quantiles can be used in robust optimization problems (Dumas et al., October 2021).

The next two chapters thoroughly describe the developed DDPM. Then, chapter 5 describes in depth the formalization behind GANs, VAEs, and NFs as well as their application for conditional multi-output forecasting in power systems. Afterward, a

complete comparison between the four deep generative models is performed in terms of quality and value (Thornes and Stephenson, 2001) in chapter 8.

Chapter 3

Denoising diffusion models

This section presents the main developed model: *Denoising diffusion probabilistic models* (DDPM). DDPM is a class of latent variable models. They have first been introduced by Sohl-Dickstein et al. (2015). Since then, they have slowly raised in popularity. Ho et al. (2020) have shown that they can achieve competitive results and scores on the task of image generation. They also proposed a modified version of the formulation of DDPM which has widely been adopted since then. DDPMs have gained a lot of popularity with Dhariwal and Nichol (2021), where a group of researchers from OpenAI presented a modified version of DDPM that was able to beat generative adversarial networks for the task of image generation. It has been an important breakthrough as GANs have been the state-of-the-art for image generation for a long time before that (Wang et al., 2020a).

3.1 Background and formalization

Formally, DDPM are a class latent variable model of the form:

$$p_\theta\left(\mathbf{x}_0\right) := \int p_\theta\left(\mathbf{x}_{0:T}\right) d\mathbf{x}_{1:T}$$

where, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T$ are latent variables. They have the same dimensionality as $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, which is a sample from the real data distribution. We have that $\mathbf{x}_0 \in \mathbb{R}^L$. With L is the dimensionality of the data. For the task of multi-output forecast L corresponds to the periods of the day. The dimension of the data was referred a T in the previous chapter, but in this chapter and in the following one, it will be referred a L in order to avoid confusion with the diffusion timesteps which is also called T. DDPM is made of two main parts:

1. The *reverse process*: It is defined by the joint distribution $p_\theta(\mathbf{x}_{0:T})$ which is a Markov chain with learned transitions. It is given by:

$$p_\theta\left(\mathbf{x}_{0:T}\right) := p\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) \tag{3.1}$$

   The transitions are given by the following equation where the mean and covariance matrix are learned and parameterize by θ:

$$p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) := \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right) \tag{3.2}$$

The goal of those transitions is to slowly remove noise and add structure to the samples.

2. The *forward process*: It is also called the *diffusion process*, it is defined with a Markov chain. It represents the posterior $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)$. The Markov chain progressively adds Gaussian noise to the data. The variance of the noise is added with respect to some noise schedule $\beta_1, \beta_2, ..., \beta_T$. The diffusion process is given by:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \tag{3.3}$$

and the transitions of the Markov chain by:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right). \tag{3.4}$$

The goal of those transitions is to slowly add Gaussian noise to the data until it becomes pure noise.

Thus, with those two processes, if one take $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$, the forward process starts from $\mathbf{x}_{t-1}$ and add some noise to it with the transition given by Eq (3.4) in order to get $\mathbf{x}_t$. On the other hand, the reverse process starts from $\mathbf{x}_t$ and tries to remove the noise that has been added in order to try to recover $\mathbf{x}_{t-1}$ through the transition given by (3.2). Performing T steps of the forward process with $t = 1, 2, ..., T$ convert a sample from the dataset into pure noise. On the other hand, performing T steps of the reverse process transforms noise into a sample of the input data distribution.

Thus, the idea behind diffusion models is to add progressively noise to the data distribution with the forward process in T steps, until obtaining pure Gaussian noise. Then, learn the reverse process in order to remove the noise from the samples. It would allow to start from noise and perform T steps in reverse in order to convert it to a data sample to achieve sampling.

One of the particularities of DDPM is that the forward process does not contain any learnable parameters. It is a fixed Markov chain that adds Gaussian noise in T steps according to a predefined noise schedule $\beta_1, \beta_2, ..., \beta_T$. Also, it has been shown by FELLER (1945) and Sohl-Dickstein et al. (2015) that if the transitions of the forward process are Gaussian and β_t is small, then the reverse process $p_\theta(\mathbf{x}_{t-1}|t)$ is also a Gaussian. That is why it make sense to chose $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to be a Gaussian in Eq (3.2). For this reason, the number of steps T must be as large as possible. Because the highest, T, the smallest the β_t will be, and the more the previous claim holds.

Training is done by maximizing the model's log-likelihood as it leads to a model that best explains the input data. However, it is intractable in practice because it would require marginalizing over all possible realization of the latent variables. However, the likelihood has a lower bound given by Jensen's inequality, given in equation (3.5). Thus in practice, the training procedure is done by optimizing the variational lower bound of the log-likelihood, which is the second term of equation (3.5) similarly to what is done with VAEs (Kingma and Welling, 2013), which leads to the following loss:

$$\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right]=\mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right)-\sum_{t>1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right]=: L. \tag{3.5}$$

One practical property of the forward process is that if the noise schedule β_t is known, it is possible to sample $\mathbf{x}_t$ at any arbitrary timestep t in closed form. As one knows the amount of noise in advance thanks to the noise schedule, we have that:

$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{\bar{\alpha}_t} \mathbf{x}_0,\left(1-\bar{\alpha}_t\right) \mathbf{I}\right) \tag{3.6}$$

with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$.

Intuitively, it means that as one knows the noise schedule, one can compute in closed form the "compound" noise from step 0 to step t, and thus directly sample any at any timestep t from the forward process. One could leverage this property for the training in order to train the reverse process at random time step t, this corresponds to random terms of the sum in Eq (3.5). It is more efficient than to use the transition given by equation (3.4) T times and follow the Markov chain.

The loss can be rewritten in terms of Kullback-Leibler divergences (KL divergence) (Perez-Cruz, 2008) as:

$$\mathbb{E}_q[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T}+\sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}] \tag{3.7}$$

Rewriting the loss with KL divergences has the advantage to reduce both variance and computing time as it avoids using Monte-Carlo estimates. The KL divergence directly compares two Gaussians which is computable in closed form. With the KL divergence, one compares $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ with the forward process posteriors $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ which are tractable if it is conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) \longrightarrow q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right) \tag{3.8}$$

With:

$$\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1-\bar{\alpha}_t} \mathbf{x}_0+\frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t} \mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t$$

To summarize, starting from the log-likelihood, one can express its variational lower bound. Afterward, by conditioning the forward process on $\mathbf{x}_0$, it has been shown that it is possible to compute its posterior in closed form which allows for efficient loss computation in terms of KL divergence.

Further details on the derivation that led to the following results can be found in Appendix B. Figure 3 illustrates the forward and reverse process with load generation.

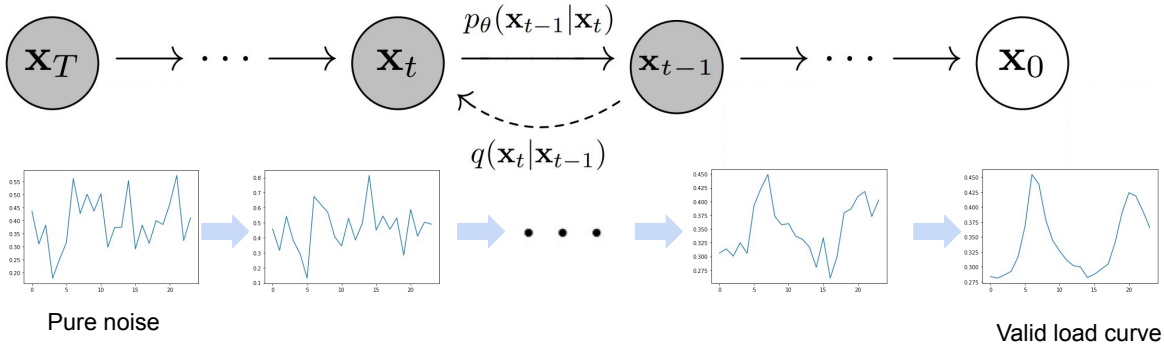


Figure 3.1: Illustration of the forward and reverse process of DDPM with an example of load scenario generation.
The forward process $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ start from a data sample $\mathbf{x}_0$ and slowly add gaussian noise to it until we have $\mathbf{x}_T$ as pure noise. The reverse process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ remove gradually the noise until recovering a valid load scenario.

3.2 Reparametrisation

The formalization presented in section 3.1 is the same as presented in the original paper of Sohl-Dickstein et al. (2015). However, Ho et al. (2020). proposed a new formulation motivated by the link between denoising diffusion models and denoising score matching (Song and Ermon, 2019). It allowed them to obtain a simpler model design as well as better empirical results.

First, if the variances β_t of the forward process are fixed constant (with no learnable parameters) then, the first term L_T in Eq. (3.7) is a constant and can thus be ignored during training.

The most general form of p_θ is given by $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)$, for $t = 1, 2, ..., T-1$. The first simplification done by Ho et al. (2020) is to fix the variance of the reverse process $\Sigma_\theta(\mathbf{x}_t, t)$ to a constant with no trainable parameters, but time-step dependent. Formally, $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2\mathbf{I}$. Ho et al. (2020) explored several values for σ_t but found it had not a considerable impact. Finally, they opted for $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.

Consequently, now we are only interested in the mean $\mu_\theta(x_t, t)$ of the reverse process. The terms L_{t-1} can be rewritten taking into account equation (3.8) as well as the fact that $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \sigma_t^2\mathbf{I}\right)$:

$$L_{t-1} = D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)$$

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \left\| \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t, \mathbf{x}_0 \right) - \boldsymbol{\mu}_\theta \left(\mathbf{x}_t, t \right) \right\|^2 \right] + C. \tag{3.9}$$

C is a constant and can thus be dropped during training. It can be seen from equation (3.9) that the most obvious parametrization of $\boldsymbol{\mu}_\theta$ is to predict $\tilde{\boldsymbol{\mu}}_t$ which is the mean of the forward process posterior. However, we can re-parameterize $\boldsymbol{\mu}_\theta$ to predict something else by using the re-parametrization trick, similarly to what is done with VAEs (Kingma and Welling, 2013). The idea is to remove the stochasticity into another independent variable. We re-parameterize $q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)$ such that: $q\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$ to obtain $\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. It leads to rewriting Eq (3.9) as :

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon} \right) - \boldsymbol{\mu}_\theta\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right), t\right) \right\|^2 \right]. \tag{3.10}$$

From the previous equation, it can be seen that $\mu_\theta(\mathbf{x}_t, t)$ must predict $\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)$. As $\mathbf{x}_t$ is always available during training, it makes sense to opt for this parametrization:

$$\boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right) = \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t\right)\right)\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right). \tag{3.11}$$

Where $\boldsymbol{\epsilon}_\theta$ is the actual function approximator. It predicts $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$. In other words, we have moved from a parametrization where in order to compute the transition of the reverse process to go from $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$, one would need to predict the mean and the variance of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to a parametrization where one predicts $\boldsymbol{\epsilon}$ which is the noise to remove to go from $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$. Thus, sampling $\mathbf{x}_{t-1} \sim p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$ becomes equivalent to computing $\mathbf{x}_{t-1}$ thanks to

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right) + \sigma_t \mathbf{z},$$

where z is sampled from a Normal distribution: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Finally, thanks to this re-parametrization, Eq (3.10) have now the following form and is the loss that will be used for training:

$$\mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t \left(1-\bar{\alpha}_t\right)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right) \right\|^2 \right]. \tag{3.12}$$

Also, Ho et al. (2020) compared this loss to a simplified version of it:

$$\mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right) \right\|^2 \right], \tag{3.13}$$

which only takes into account the absolute error between the added noise and the predicted noise. While this loss is simpler, they have shown better empirical results on computer vision applications using it. Removing the weights in Eq (3.12) corresponds to training on a weighted version of the variational bound on the log-likelihood instead of the actual variational lower bound on the log-likelihood. Also, there exist two ways of performing the diffusion process during training. Either, perform the entire T forward

diffusion steps for each data point, or sample randomly time steps for each data sample similar to the idea behind stochastic gradient descent but applied to the diffusion steps. The two ways of performing the diffusion process at training time can be found on Algorithm 1 and 2. On the other hand, the sampling procedure can be found on Algorithm 3. Experiments regarding the loss function and the training algorithms can be found in chapter 7.

Algorithm 1 Iterative training algorithm

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t = 0$
  while $t \leq T$ **do**
    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Gradient step on: $||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)||^2$
  end while

until converged

Algorithm 2 Stochastic training algorithm

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim Uniform(1, 2, .., T)$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Gradient step on: $||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)||^2$

until converged

To summarize, we have two valid parameterizations of DDPMs. First, one can train the reverse process with a function approximator $\boldsymbol{\mu}_\theta$ that predicts $\tilde{\boldsymbol{\mu}}_t$ which is the mean of the distribution that removes the noise. Then, it is also possible to have a function approximator $\boldsymbol{\epsilon}_\theta$ that predicts $\boldsymbol{\epsilon}$ which is the noise to remove at each time step in the reverse process. Both parameterizations make sense and are valid. Then Ho et al. (2020) derived the loss function and proposed a simplified version of it and empirically compared both. They found out that the latter is both simpler and lead to better performances on computer vision application. Chapter 7 will explore if that claim still holds for the task of probabilistic forecasting.

Algorithm 3 Sampling algorithm

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, .., 1$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})ift > 0$ else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)) + \sigma_t \boldsymbol{z}$
end for
return $\mathbf{x_0}$

In practice, one need a function approximator $\boldsymbol{\epsilon}_\theta$. It takes as input a noisy scenario and the noise step t and returns the noise, which is a tensor of the same size as the scenario. Subtracting this noise to the noisy scenario at time step t lead to the less noisy scenario $\mathbf{x}_{t-1}$. Each of those steps progressively leads the initial Gaussian noise

closer to a real valid scenario. The next chapter will detail further the choice of the architecture of $\boldsymbol{\epsilon}_\theta$.

Chapter 4

Architecture

This section describes the different parts of the architecture of $\boldsymbol{\epsilon}_\theta$: $\mathbb{R}^L \times \mathbb{N} \to \mathbb{R}^L$. It is a neural network based on DiffWave (Kong et al., 2020) and WaveNet (Oord et al., 2016a). This architecture is a 1-D adaptation of a popular architecture of generative model for images: PixelCNN (Oord et al., 2016b). This architecture has already been applied to diffusion models for the task of speech synthesis (Kong et al., 2020). It is based on dilated convolutions. Figure 4.1 summarizes the overall architecture. The network is composed of N residual layers. The next subsections describe individually the most important parts of the architecture.

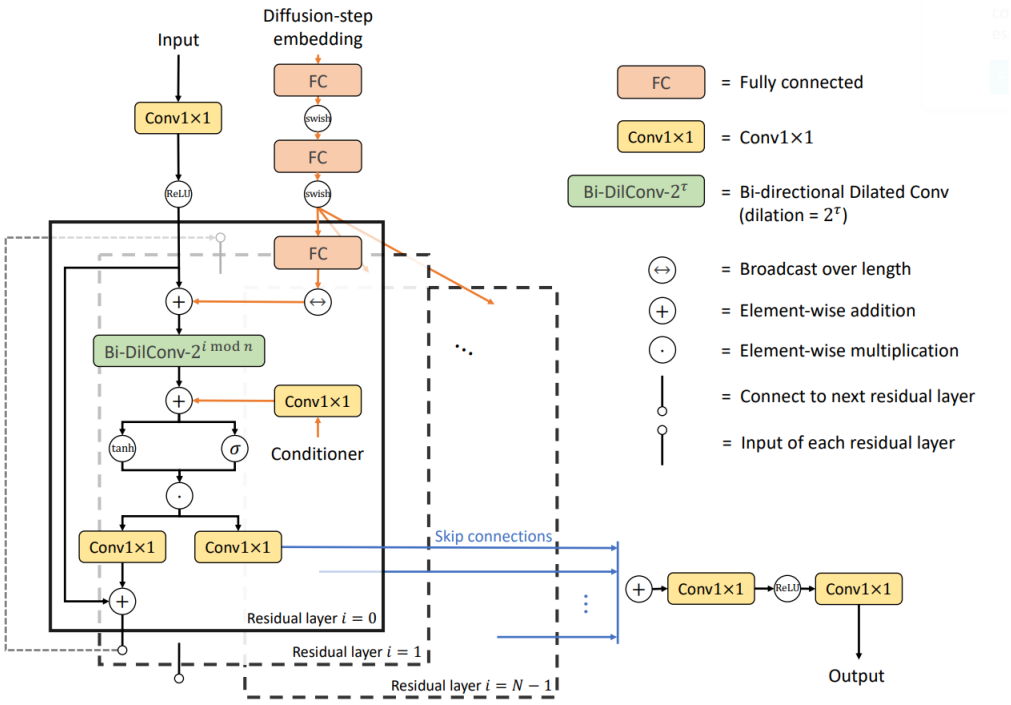


Figure 4.1: Overall network architecture of $\boldsymbol{\epsilon}_\theta$. Its different blocks are discussed throughout the next subsections.
Source: (Kong et al., 2020)

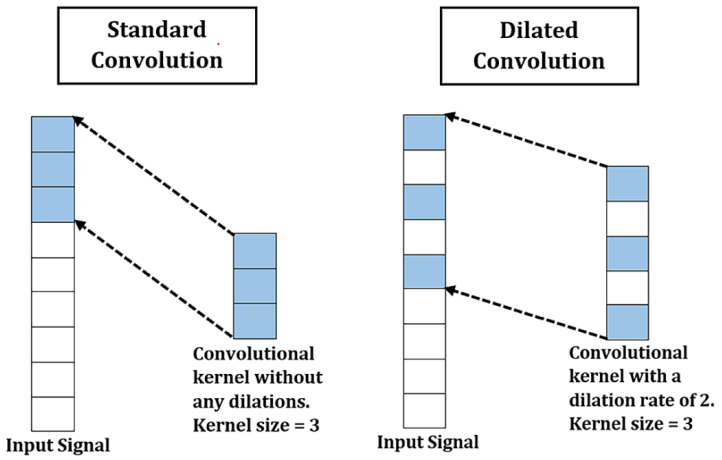


Figure 4.2: Comparison between a normal 1-D convolution and a 1-D dilate convolution with a dilation factor of 2. Both convolutions have a kernel size of 3.
Source: (Raja Sekaran et al., 2022)

4.1 Dilated convolution

The idea behind dilated convolution is to apply the convolution filter to a larger area than its size by skipping some of the input values. The kernel is applied on input elements equally spaced depending on the dilation factor. The idea is similar to pooling but here, it leads to an output of the same size as the input. One can note that normal convolutions can be seen as a particular case of dilated convolution with a dilation factor equal to 1. Dilated convolutions have already been applied to computer vision architectures for instance for image segmentation (Chen et al., 2014; Yu and Koltun, 2015b). The main advantage of dilated convolutions is the ability to significantly increase the receptive field while preserving input resolution. Figure 4.2 shows the difference between dilated and standard 1-D convolution.

4.2 Residual and skip connections

Residual (He et al., 2015) and skip connections are used throughout the network. Their goal is to prevent issues induce when training too deep networks such as exploding or vanishing gradient. It also speeds up the convergence of the network. The residual connections flow through each layer, while the skip connections flow directly to the output.

4.3 Residual layers

The network is composed of N residual layers. Each has a residual and a skip connection. The chosen configuration for this architecture is to double the dilation factor at each residual layer, i.e., $[1, 2, 4, .., 2^{N-1}]$. The idea is that by increasing exponentially the dilation factor with depth, it increases exponentially the receptive field (Yu and Koltun, 2015a). Furthermore, adding layers increases the overall capacity of the network.

4.4 Gated activation units

The chosen non-linearity comes in the form of a gated activation unit. It is a combination of sigmoid activation and hyperbolic tangent activation. It is inspired from Oord et al. (2016b). It has been shown by Oord et al. (2016a) that this non-linearity leads to better results than rectified linear activation function (RELU) (Hinton, 2010) for modeling time series data. The gated activation unit is given by

$$\tanh\left(W_{f,k} * \mathbf{x}\right) \odot \sigma\left(W_{g,k} * \mathbf{x}\right), \tag{4.1}$$

where W is a learnable convolution filter, f and g denote respectively filter and gate, $\sigma(.)$ is the sigmoid operator, and k is the layer index. Each residual layer has one gated activation unit.

4.5 Diffusion step embedding

One particularity of $\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)$ is that it takes two inputs. The first is the noisy sample $\mathbf{x}_t$ and the second is the time step t (three if we count the conditioner). It is important to include the time step in the model as $\boldsymbol{\epsilon}_\theta(\cdot, t)$ must lead to different outputs for different t. As incorporating the time step t to the model is similar to incorporating information about the position, it has been chosen to use position embedding as introduced by Vaswani et al. (2017) for the embedding of the time step. The time step t is converted to a vector containing sine and cosine of different frequencies. It is chosen to use a vector of dimension 64 for the embedding of the time step t. It is given by

$$t_{\text{embedding}} = \left[\sin\left(10^{\frac{0\times4}{31}}t\right), \cdots, \sin\left(10^{\frac{31\times4}{31}}t\right), \cos\left(10^{\frac{0\times4}{31}}t\right), \cdots, \cos\left(10^{\frac{31\times4}{31}}t\right)\right]. \tag{4.2}$$

Then, after the embedding is produced, it goes through a series of three fully connected layers. The first two are common for all the residual layers and thus share parameters. The third is layer-specific and is used to map the output of the first two onto an embedding vector that can be added to the input of every residual layer of the model.

4.6 Conditionality embedding

The model produces conditional outputs, meaning that it takes an additional input in the form of a condition vector. This condition vector is formed from an input feature vector. Similar to the diffusion embedding, this feature vector will go through 3 layers. The first two are fully connected layers common for all the residual layers and use RELU activation function, and the last one is a 1D convolution and is different for each residual layer. The conditioner block on figure 4.1 is obtained as the output of the two common fully connected layers from an input feature vector.

Chapter 5

Deep generative model baselines

This chapter presents the formalization behind GANs, VAEs, and NFs as well as their application for multi-output probabilistic forecasting. Those models will be used for the comparison with the proposed DDPM introduced in chapter 3.

5.1 Generative adversarial networks

Generative adversarial networks are a class of latent deep generative model. It has the particularity of training simultaneously two models in an adversarial way. A *generative* model G and a *discriminative* model D. The generative model G tries to capture the data distribution, and the discriminative model D represents the probability that a sample came from the dataset rather than being generated from G. The two models are trained simultaneously: D is trained to maximize the probability of assigning correct labels to both the real samples and those generated by G. On the other hand, G is trained by maximizing the probability of D making mistakes, i.e, maximizing $\log(1 - D(G(z)))$. Such a framework corresponds to a min-max two-player game with an objective value function V:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))].$$

Theoretically, in the space of arbitrary functions G and D, a unique solution exists, where G recovers the data distribution and D is equal to $\frac{1}{2}$ everywhere (Goodfellow et al., 2014), meaning that the discriminator is not able to distinguish between real and generated samples. If G and D are both neural networks, the entire system is trainable through backpropagation. In practice, both the generator and the discriminator are neural networks. The generator: g_θ with the set of parameters θ and the discriminator d_ϕ with another set of parameters ϕ.

GANs suffer from several issues. The main being mode collapse and the instability of training. A solution to those problems has been proposed in the form of Wasserstein GAN (WGAN) (Arjovsky et al., 2017) and has also been applied to power systems by Yuan et al. (2022) and Wang et al. (2020b). After that, an improved version of WGAN in the form of Wasserstein GAN with gradient penalty (WGAN-GP) proposed by Gulrajani et al. (2017) has emerged. Appendix C describes the history and improvements of GANs from their introduction as proposed by Goodfellow et al. (2014) to WGAN (Arjovsky et al., 2017) and until the WGAN-GP (Gulrajani et al., 2017) used in this thesis.

5.1.1 GAN for probabilistic forecast

GANs can be applied to the task of multi-output forecast. In this settings, the generator $g_\theta(\cdot) : \mathbb{R}^d \times \mathbb{R}^{|\mathbf{c}|} \to \mathbb{R}^T$ maps a latent vector $\mathbf{z} \in \mathbb{R}^d$ and a condition vector $\mathbf{c} \in \mathbb{R}^{|\mathbf{c}|}$ onto a sample $\hat{\mathbf{x}} \in \mathbb{R}^T$. The discriminator $d_\phi(.) : \mathbb{R}^T \times \mathbb{R}^{|\mathbf{c}|} \to [0, 1]$ [2]is a classifier that determines if a sample is a real sample from the dataset $\mathbf{x}$ or a generated sample $\hat{\mathbf{x}}$. After training, one uses the generator g_θ to produce the scenarios. Figure 5.1 sketch an overview of the conditional generative adversarial network with load scenarios generation.

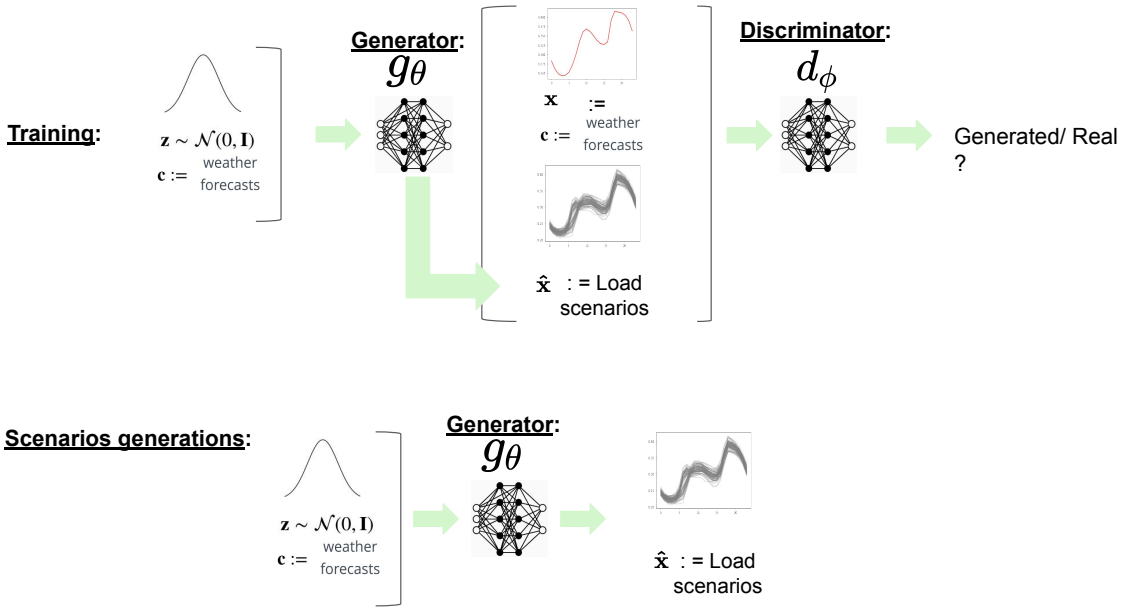


Figure 5.1: Overview of the conditional generative adversarial network on the case of load scenarios generation.
The generator g_θ generates load scenarios $\hat{\mathbf{x}}$ from samples from a normal distribution and conditional data in the form of weather forecast $\mathbf{c}$. The discriminator d_ϕ is a classifier that differentiate true data $\mathbf{x}$ from generated scenarios $\hat{\mathbf{x}}$. Both the generator and the discriminator are trained through a min-max problem. After training, one uses the generator g_θ to perform scenarios generation.

5.2 Variational auto-encoders

Variational auto-encoders (VAE) (Kingma and Welling, 2013) are a class of deep latent variable models composed of two distinct parts. An *encoder* and a *decoder*. The encoder takes as input a data sample and converts it to a latent representation $\mathbf{z}$ with smaller dimension d following some known distribution, for instance, a Gaussian. The decoder takes a latent representation $\mathbf{z}$ and return a reconstruction $\hat{\mathbf{x}}$. Both models are trained jointly with backpropagation. The goal is to minimize a lower bound on the likelihood. Indeed, maximum likelihood is intractable as it would require marginalizing over all possible realization of the latent variable $\mathbf{z}$. VAEs circumvent this problem by maximizing a variational lower bound of the likelihood.

Formally, the encoder is a neural network $q_\varphi(.) : \mathbb{R}^T \times \mathbb{R}^{|\mathbf{c}|} \to \mathbb{R}^d$. It approximate the intractable posterior $p(\mathbf{z}|\mathbf{x}, \mathbf{c})$ and the decoder is also a neural network $p_\theta(.) : \mathbb{R}^d \times \mathbb{R}^{|\mathbf{c}|} \to$

$\mathbb{R}^T$. It approximates the likelihood $p(\mathbf{x}|\mathbf{z}, \mathbf{c})$ with $\mathbf{z} \in \mathbb{R}^d$. As stated, the log-likelihood is non tractable, thus its variational lower bound $\mathcal{L}_{\theta,\varphi}(\mathbf{x}, \mathbf{c})$ is used instead. We have

$$\begin{aligned}\log p_\theta(\mathbf{x} \mid \mathbf{c}) &= KL\left[q_\varphi(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) \| p(\mathbf{z} \mid \mathbf{x}, \mathbf{c})\right] + \mathcal{L}_{\theta,\varphi}(\mathbf{x}, \mathbf{c}), \\ &\geq \mathcal{L}_{\theta,\varphi}(\mathbf{x}, \mathbf{c}), \\ \mathcal{L}_{\theta,\varphi}(\mathbf{x}, \mathbf{c}) &:= \mathbb{E}_{q_\varphi(\mathbf{z}|\mathbf{x},\mathbf{c})}\left[\log \frac{p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}, \mathbf{c})}{q_\varphi(\mathbf{z} \mid \mathbf{x}, \mathbf{c})}\right],\end{aligned}$$

as the first term is a KL divergence (Perez-Cruz, 2008) and is therefore non-negative. The Kullback-Leibler (KL) divergence is a measure of distance between probability distributions. Furthermore, it is is intractable in this case, this is why we bound $\log p_\theta$ with $\mathcal{L}_{\theta,\varphi}(\mathbf{x}, \mathbf{c})$. Finally, $\mathcal{L}_{\theta,\varphi}(\mathbf{x}, \mathbf{c})$ can be re-written as

$$\mathcal{L}_{\theta,\varphi}(\mathbf{x}, \mathbf{c}) = \mathop{\mathbb{E}}_{q_\varphi(\mathbf{z}|\mathbf{x},\mathbf{c})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z}, \mathbf{c})\right] - \mathrm{KL}\left[q_\varphi(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) \| p(\mathbf{z})\right]. \tag{5.1}$$

The first term encourages distribution to place their mass on configurations of latent variables that explain the observed data. The second terms encourages $q_\varphi(\mathbf{z} \mid \mathbf{x}, \mathbf{c})$ to stay close to the prior. When it comes to optimization, as there are two sets of parameters, there are two sets of gradients to compute. First, the gradients w.r.t θ, $\nabla_\theta \mathcal{L}_{\theta,\varphi}$ are simple to obtain, they can be computed with the classical Monte Carlo integration. However, on the other hand, gradients w.r.t φ, $\nabla_\varphi \mathcal{L}_{\theta,\varphi}$ are difficult to obtain because they would require back-propagates through a stochastic node in the computation graph of the model, which is non-differentiable. To bypass this problem, Kingma and Welling (2013) proposed a *reparametrization trick*. The idea is to re-parameterize the variable $\mathbf{z}$ as a deterministic variable with a stochastic part ϵ with independent marginal $p(\epsilon)$ independent of $\mathbf{x}$ and φ such that the gradients can backpropagate through $\mathbf{x}$ and φ. The reparametrization is given by

$$\mathbf{z} = g_\varphi(\epsilon, \mathbf{x}). \tag{5.2}$$

It allows the gradients to backpropagate. Then the variational lower bound can be re-written as

$$\mathcal{L}_{\theta,\varphi}(\mathbf{x}, \mathbf{c}) = \mathop{\mathbb{E}}_{p(\epsilon)}\left[\log p_\theta(\mathbf{x} \mid g_\varphi(\epsilon, \mathbf{x}), \mathbf{c})\right] - \mathrm{KL}\left[q_\varphi(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) \| p(\mathbf{z})\right], \tag{5.3}$$

and can be estimated with Monte-Carlo integration as well.
As presented by Kingma and Welling (2013), it is often chosen for $p(\mathbf{z})$ to be an isotropic Gaussian, $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c})$ and $q_\varphi(\mathbf{z}|\mathbf{x}, \mathbf{c})$ to be both multivariate Gaussian with learnt parameters $\mu_\theta, \sigma_\theta$ and $\mu_\varphi, \sigma_\varphi$. It leads to

$$\begin{aligned}p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}), \\ p_\theta(\mathbf{x} \mid \mathbf{z}, \mathbf{c}) &= \mathcal{N}\left(\mathbf{x}; \mu_\theta, \sigma_\theta^2 \mathbf{I}\right), \\ q_\varphi(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) &= \mathcal{N}\left(\mathbf{z}; \mu_\varphi, \sigma_\varphi^2 \mathbf{I}\right), \\ \mu_\theta, \log \sigma_\theta^2 &= \mathrm{NN}_\theta(\mathbf{x}, \mathbf{c}), \\ \mu_\varphi, \log \sigma_\varphi^2 &= \mathrm{NN}_\varphi(\mathbf{z}, \mathbf{c}),\end{aligned}$$

and using the reparametrization trick,

$$\begin{aligned}\epsilon &\sim \mathcal{N}(0, \mathbf{I}), \\ \mathbf{z} &:= \mu_\varphi + \sigma_\varphi \epsilon.\end{aligned}$$

The choice of Gaussians and the reparametrization trick lead to $\mathcal{L}_{\theta,\varphi}$ being commutable and differentiable without the need for Monte-Carlo integration as the KL divergences simplify by

$$\mathrm{KL}\left[q_\varphi(\mathbf{z} \mid \mathbf{x}, \mathbf{c}) \| p(\mathbf{z})\right] = -\frac{1}{2}\sum_{j=1}^{d}\left(1 + \log \sigma_{\varphi,j}^2 - \boldsymbol{\mu}_{\varphi,j}^2 - \sigma_{\varphi,j}^2\right) \tag{5.4}$$

and the first term of 5.1 becomes

$$\mathop{\mathbb{E}}_{p(\epsilon)}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z}, \mathbf{c})\right] \approx -\frac{1}{2}\left\|\frac{\mathbf{x} - \boldsymbol{\mu}_\theta}{\sigma_\theta}\right\|^2. \tag{5.5}$$

5.2.1 VAE for probabilistic forecast

From there, one can perform scenarios generation simply by doing Monte-Carlo samples with the decoder $p_\theta(.)$ starting from a Gaussian input. There are no particular constraints on the architecture of NN_θ and NN_φ, they can be arbitrarily complex. For the rest of the thesis, both NN_θ and NN_φ are chosen to be simple fully connected neural networks. Figure 5.2 pictures the conditional VAE process with load scenarios generation as an example.

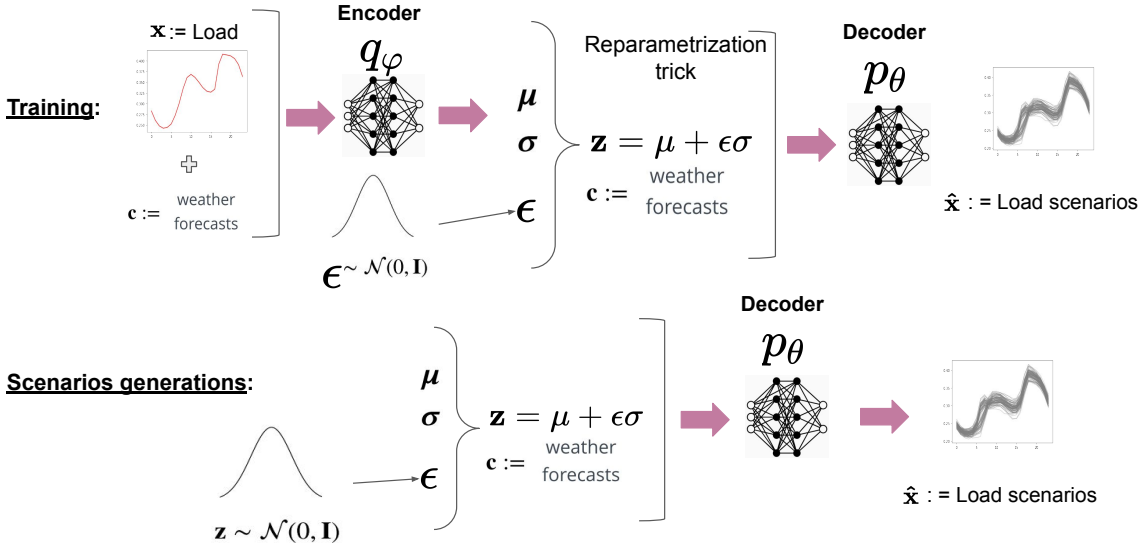


Figure 5.2: Overview of the conditional variational auto-encoder illustrated in the case of load scenarios generation.
The encoder q_φ takes the variable of interest $\mathbf{x}$ and maps it to a latent space $\mathbf{z}$ taking as secondary input weather forecast data $\mathbf{c}$. The decoder p_θ takes as input sample from the latent space and the condition variable $\mathbf{c}$ and outputs scenarios $\hat{\mathbf{x}}$ for the variable of interest.

5.3 Normalizing flows

Normalizing flows (Rezende and Mohamed, 2016) learns a complex probability distribution in a particular way. The idea is to start from an analytically known distribution,

such as a Gaussian for instance, and then, transform this probability density through a series of invertible mappings. This sequence of transformations is known as *flow*. A normalizing flow learns the flow. from a known density to the unknown density representing the data. Then, thanks to the invertibility of the transformation, one can model the unknown density of the data with the inverse transformation. In opposition to most deep generative models, NFs can be trained directly through maximum log-likelihood estimation and have direct access to the data set probability distribution. Following the flow from the know tractable density to the unknown density correspond to applying the rule of change of variable in a chain, once per mapping.

Formally, a normalizing flow is defined as a sequence of transformation :$f_k : \mathbb{R}^T \to \mathbb{R}^T, k = 1, \ldots, K$. Those transformations must be invertible. They are composed together to form an invertible mapping: $f_\theta := f_1 \circ \ldots \circ f_K : \mathbb{R}^T \to \mathbb{R}^T$. This composed function can be used to perform density estimation by using f_θ in order to map a data point $\mathbf{x} \in \mathbb{R}^T$ onto a latent vector $\mathbf{z} \in \mathbb{R}^T$ following a tractable density function p_z, for instance a Gaussian. The flow f_θ implicitly defines a density $p_\theta(\mathbf{x})$ thanks to the change of variable formula

$$p_\theta(\mathbf{x}) = p_z\left(f_\theta(\mathbf{x})\right) \left|\det J_{f_\theta}(\mathbf{x})\right|, \tag{5.6}$$

where J_{f_θ} is the Jacobian of f_θ regarding $\mathbf{x}$. The change of variables formula describes how to evaluate densities of a random variable that is a deterministic transformation from another variable. We have

$$Z = f_\theta(X) \tag{5.7}$$
$$X = f_\theta^{-1}(Z), \tag{5.8}$$

where X is the random variable of interest and Z is the latent random variable.
The model can be trained by minimizing the log-likelihood of the parameters θ given by $\sum_{i=1}^{N} \log p_\theta\left(\mathbf{x}^i, \mathbf{c}^i\right)$ over the dataset $\mathcal{D}$. Generally, f_θ can take any form, but some constraints must be fulfilled. First, it must define a bijection. Then it must admit input and output with the same dimensions. Finally, computation-wise, the determinant of the Jacobian in equation (5.6) must be efficiently computed and differentiable in order to scale with the data dimensionality. Indeed, the evaluation of the likelihood of a distribution parameterized by a normalizing flow requires computing (5.6) and its log determinant:

$$\begin{aligned}\log\left|\det J_{f_\theta}(\mathbf{x})\right| &= \log\left|\prod_{k=1}^{K} \det J_{f_{k,\theta}}(\mathbf{x})\right| \\ &= \sum_{k=1}^{K} \log\left|\det J_{f_{k,\theta}}(\mathbf{x})\right|.\end{aligned}$$

Still, with no further hypothesis on f_θ, the complexity of the log-determinant is $\mathcal{O}(K.T^3)$ which is intractable for large T. Thus, finding a way of reducing the computational cost of those operations is of crucial importance as the likelihood is repeatedly computed. For the following, the indices k are neglected by choosing a value of K equal to 1 to lighten the notations, it corresponds to a single-step flow.

There are several possible ways of implementing NFs in order to meet the constraints on the Jacobian as shown by Papamakarios et al. (2019), Kobyzev et al. (2021) and Kingma et al. (2016). For this thesis, inspiring from Dumas et al. (2021), it has been chosen to use *autoregressive* transformations (Kingma et al., 2016). In this configuration, f_θ is rewritten as a vector of scalar bijections

$$\begin{aligned} f_\theta(\mathbf{x}) &:= \left[f^1\left(x_1; h^1\right), \ldots, f^T\left(x_T; h^T\right)\right]^\top, \\ h^i &:= h^i\left(\mathbf{x}_{<i}; \varphi^i\right) \quad 2 \leq i \leq T, \\ \mathbf{x}_{<i} &:= \left[x_1, \ldots, x_{i-1}\right]^\top \quad 2 \leq i \leq T, \end{aligned}$$

with the $f^i(\cdot; h^i)$ are functions parametrized by a conditioner $h^i(\cdot; \varphi^i)$ whose values h^i only depends on the $i-1$ first values of $\mathbf{x}$. Formally, we have $f^i(\cdot; h^i) : \mathbb{R} \to \mathbb{R}$ and $h^i(\cdot; \varphi^i) : \mathbb{R}^{i-1} \to \mathbb{R}^{|h^i|}$ and θ is the union of all the φ^i.

The advantage of an autoregressive transformation f_θ is that the Jacobian of such transformations is defined by a lower triangular matrix and the computation of the log-determinant is now computed in $\mathcal{O}(T)$ instead of $\mathcal{O}(T^3)$ because the determinant of a lower triangular matrix is defined simply by the product of the elements of its diagonal. Its expression becomes

$$\begin{aligned} \log\left|\det J_{f_\theta}(\mathbf{x})\right| &= \log \prod_{i=1}^{T} \left|\frac{\partial f^i}{\partial x_i}\left(x_i; h^i\right)\right| \\ &= \sum_{i=1}^{T} \log \left|\frac{\partial f^i}{\partial x_i}\left(x_i; h^i\right)\right|. \end{aligned} \tag{5.9}$$

From there, different type of transformers f^i could be chosen: affine, non-affine, integration based, *etc.* Inspired again from Dumas et al. (2021), this work implements integration-based transformers that use Unconstrained Monotonic Neural Networks (UMNN) (Wehenkel and Louppe, 2019). UMNN is a universal density approximator of continuous random variables when combined with autoregressive functions. The UMNN consists of a neural network architecture that enables learning arbitrary monotonic functions. They parameterize the bijections f^i as

$$f^i\left(x_i; h^i\right) = \int_0^{x_i} \tau^i\left(x_i, h^i\right) dt + \beta^i\left(h^i\right), \tag{5.10}$$

with $\tau^i(\cdot; h^i) : \mathbb{R}^{|h^i|+1} \to \mathbb{R}^+$ is the integrated neural network that has a strictly positive output, $h^i \in \mathbb{R}^{|h^i|}$ the embedding from the conditioner, and $\beta^i(\cdot) : \mathbb{R}^{|h^i|} \to \mathbb{R}$ a neural network outputting a scalar. The T autoregressive embeddings of the flow are implemented using Masked autoregressive Network (MAF) introduced by Papamakarios et al. (2017). Finally, by putting everything together and by adding the weather conditional forecasts, the expression of the log-density is given by

$$\begin{aligned} \log p_\theta(\mathbf{x}, \mathbf{c}) &= \log p_z\left(f_\theta(\mathbf{x}, \mathbf{c})\right) \left|\det J_{f_\theta}(\mathbf{x}, \mathbf{c})\right| \\ &= \log p_z\left(f_\theta(\mathbf{x}, \mathbf{c})\right) + \sum_{i=1}^{T} \log \tau^i\left(x_i, h^i\left(\mathbf{x}_{<i}\right), \mathbf{c}\right) \end{aligned} \tag{5.11}$$

It is important to note that the previous derivations imply a single-step transformation f_θ. However, in practice, several transformations must be stacked in order to obtain a

high-capacity model. Indeed, the constraints imposed on the Jacobians reduces significantly the expressiveness. As a consequence, more transformations are needed.

5.3.1 NF for probabilstic forecast

The flow f_θ is trained to map the distribution of the variable of interest (PV, load, wind power) to a simple distribution such as a Normal distribution. Then, one can use the inverse transformation f_θ^{-1} in order to generate the scenarios from a sample of a Normal distribution and the weather condition forecast. An overview of the conditional NF process with load scenarios generation is given by figure 5.3.

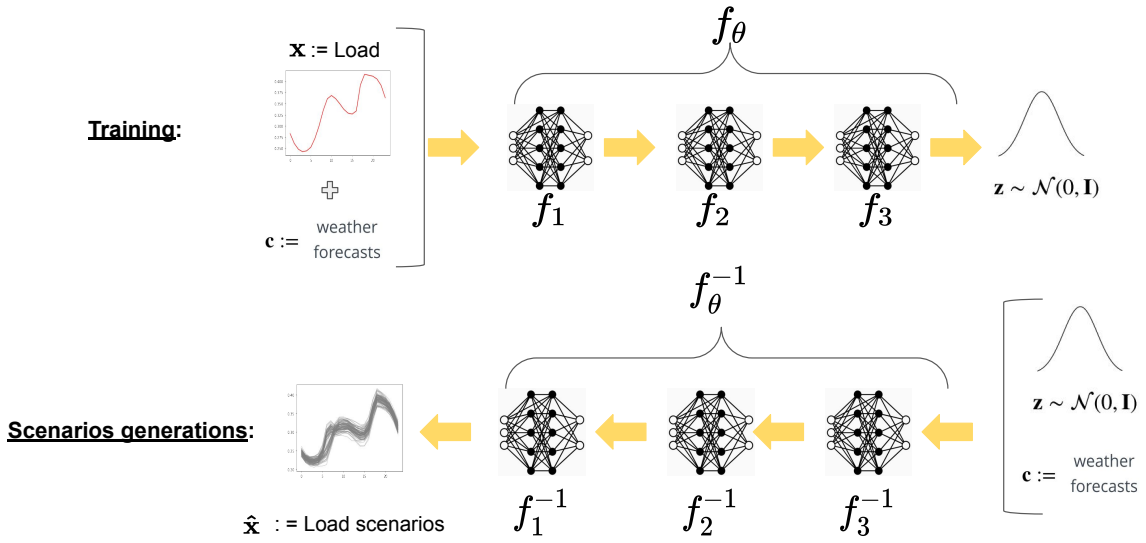


Figure 5.3: Overview of the conditional Normalizing flow represented in the case of load scenarios generation with a three step flow.
The model is trained through the log-likelihood of the parameters θ given the dataset. The flow f_θ defines a bijection between a variable of interest $\mathbf{x}$ (here the load) and a variable sampled from simple distribution (here a Normal distribution). Then, the scenarios are generated thanks to f_θ^{-1} that performs the opposite mapping from a Normal distribution and the weather forecast $\mathbf{c}$ to the scenarios $\hat{\mathbf{x}}$.

Chapter 6

Quality and value assessment

When it comes to assessing the predictions, one must consider two angles. First is the forecast quality. It corresponds to the ability to mimic the process involved. Forecasts quality represents the ability of the forecast to deliver information about the future. Forecasts quality is based on objective criteria. On the other hand, the forecast value represents the actual benefit of using the predictions in a decision-making process (Dumas et al., 2021). This chapter describes the case study, the different datasets used, as well as the framework for assessing the quality and the value of the predictions.

6.1 Case study

The quality and value assessment of the different models will be performed with the open access Global Energy Forecasting Competition 2014 (GEFCom 2014) datasets (Hong et al., 2016). It is composed of several tracks. For the case study, three of those are going to be used. The load track, the wind power track, and the solar power track. The load track contains hourly historical loads and hourly weather forecasts for one zone. The wind track contains the hourly wind power generation by timesteps of 24 hours in 10 different zones corresponding to 10 wind farms in Australia as well as weather forecasts. Similarly, the solar power track contains hourly solar power 24 hours ahead in three distinct zones corresponding to three solar power plants also located in Australia as well as weather forecasts data for the same periods.

Each of the three tracks is going to be separated into three parts in order to train and evaluate the models: the learning, validation, and testing sets. The learning set (LS) is used to train the models, the validation set (VS) is used to configure the hyper-parameters, and the testing set (TS) is used to compare the quality and value of the different generative models. The different models (DDPM, GAN, NF, VAE) will use some weather forecasts data as input and output day-ahead scenarios $\hat{\mathbf{x}} \in \mathbb{R}^T$ for each of the three tracks. Each of the tracks contains different information and different weather-based features will be extracted from them.

6.1.1 Load track

The features are composed of the weather forecasts. The track is composed of one zone and 25 stations. The load condition vector for each day d is given by

$$\mathbf{c}_d^{load} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{25}]. \tag{6.1}$$

and it's dimension is $D = 24 \times 25$.

Examples of load curves can be found in figure 6.1. Load curves are usually smooth curves with one or two peaks during the day. Those peaks correspond to moments of high power consumption, usually at the beginning of the morning or in the evening depending on the season.

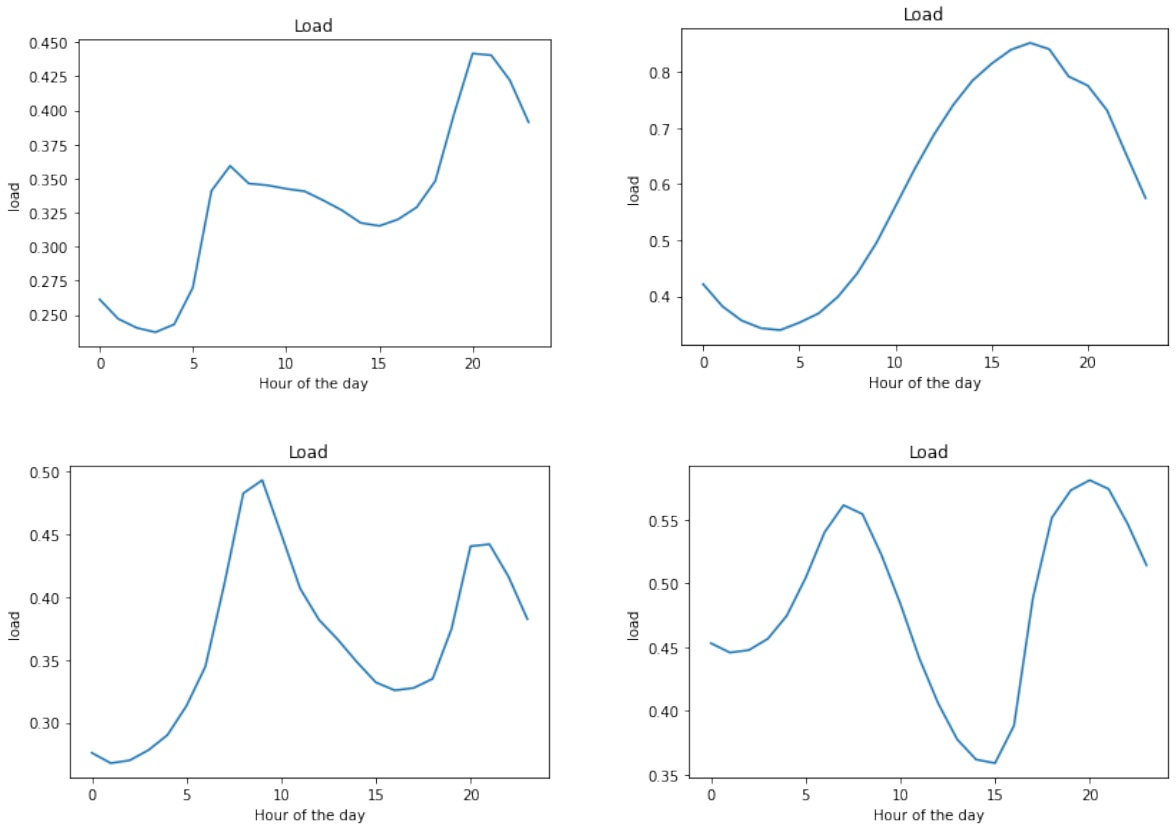


Figure 6.1: Examples of load curves from random days of the load track of the GEFCom 2014 dataset

6.1.2 Wind track

In this track, one has access to the zonal and meridional wind component at different altitude.

- $\mathbf{u}^{10}$: The zonal component of the wind at 10 meters.
- $\mathbf{u}^{100}$: The zonal component of the wind at 100 meters.
- $\mathbf{v}^{10}$: The meridional component of the wind at 10 meters.
- $\mathbf{v}^{10}$: The meridional component of the wind at 100 meters.

From there, 6 features inspired from Landry et al. (2016) are extracted. Those features are the wind speed $\mathbf{ws}^{10}, \mathbf{ws}^{100}$, the wind energy $\mathbf{we}^{10}, \mathbf{we}^{100}$ and wind direction $\mathbf{wd}^{10}, \mathbf{wd}^{100}$ at both altitudes. They are derived according to the following formulas

$$\begin{aligned}\mathbf{ws} &= \sqrt{\mathbf{u}+\mathbf{v}},\\ \mathbf{we} &= \frac{1}{2}\mathbf{ws}^3,\\ \mathbf{wd} &= \frac{180}{\pi}\arctan(\mathbf{u},\mathbf{v}).\end{aligned} \tag{6.2}$$

Also, this track contains 10 different zones. This information is incorporated into the wind condition vector by adding one-hot encoding variables $O_1, O_2, ..., O_{10}$. The wind feature vector for a given day d is given by

$$\mathbf{c}_d^{\text{wind}} = \left[\mathbf{u}_d^{10}, \mathbf{u}_d^{100}, \mathbf{v}_d^{10}, \mathbf{v}_d^{100}, \mathbf{ws}_d^{10}, \mathbf{ws}_d^{100}, \ \mathbf{we}_d^{10}, \mathbf{we}_d^{100}, \mathbf{wd}_d^{10}, \mathbf{wd}_d^{100}, O_1, \ldots, O_{10}\right], \tag{6.3}$$

It's dimension is $24 \times 10 + 10$.

Examples of wind power generation can be found in figure 6.2. It can be seen that wind power exhibit a high degree of variability. It is due to the more stochastic nature of wind in comparison to load or photovoltaic generation for instance.

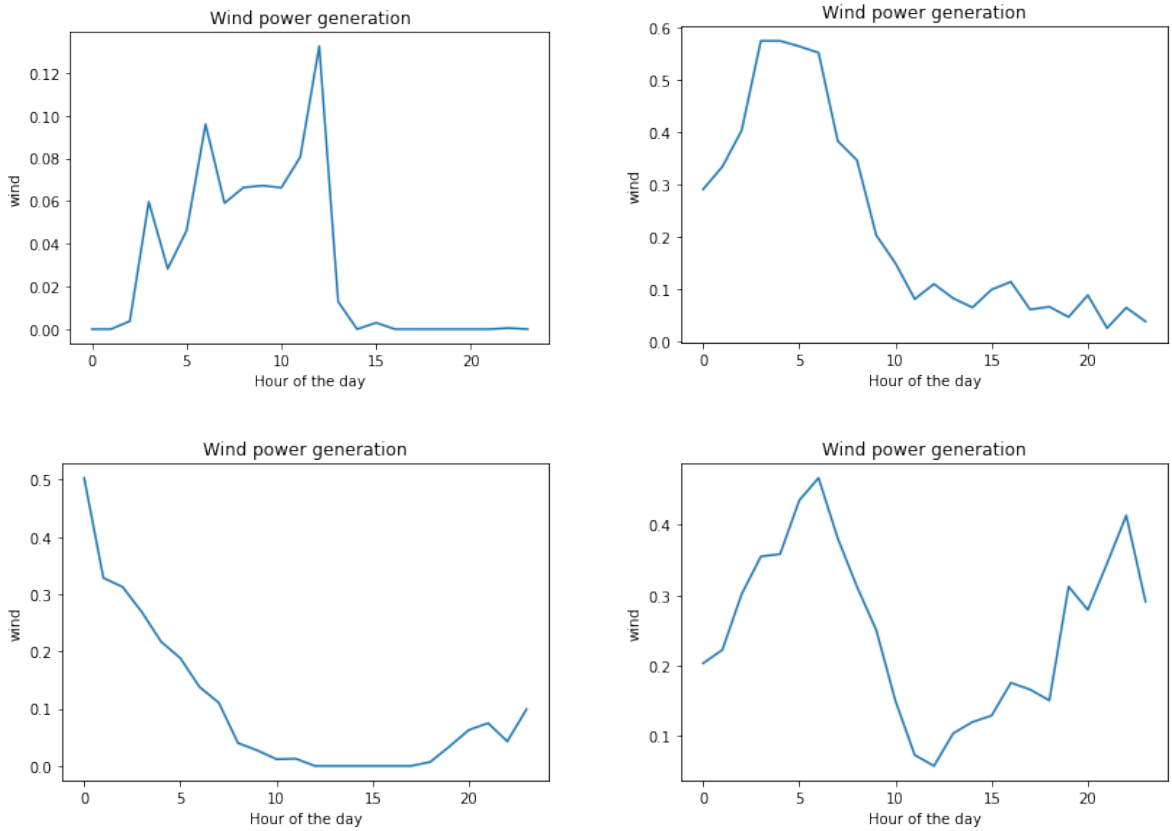


Figure 6.2: Examples of wind curves from random days of the wind track of the GEFCom 2014 dataset

6.1.3 Photovoltaic track

In this track, the wind temperature **T**, the solar irradiation **I**, and the relative humidity **rh** are available and selected as features. In addition, secondary features are

computed from them by computing $\mathbf{I}^2$ and $\mathbf{IT}$ inspired by Dumas et al. (2021). Like for the wind track, one hot encoding is used to take the different zones into account, leading to the PV feature vector for the day d

$$\mathbf{c}_d^{\mathrm{PV}} = \left[\mathbf{I}_d, \mathbf{T}_d, \mathbf{rh}_d, \mathbf{I}_d^2, \mathbf{IT}_d, O_1, O_2, O_3\right]. \tag{6.4}$$

Also, there are hours during a full 24 hours cycle in which the PV generation is always equal to 0 during all the days d and over all the zones. Those periods have been removed. It leads to a vector of dimension $5 \times 16 + 3$, where 16 is the number of hours per day with non-zeros PV generation in the dataset across all zones.

Examples of PV generation curves can be found in figure 6.3. One can see that PV generation shows more recognizable patterns than wind power. Indeed, The PV curve usually takes the form of an inverted bell with non-negative values only during the day (when there is sun).

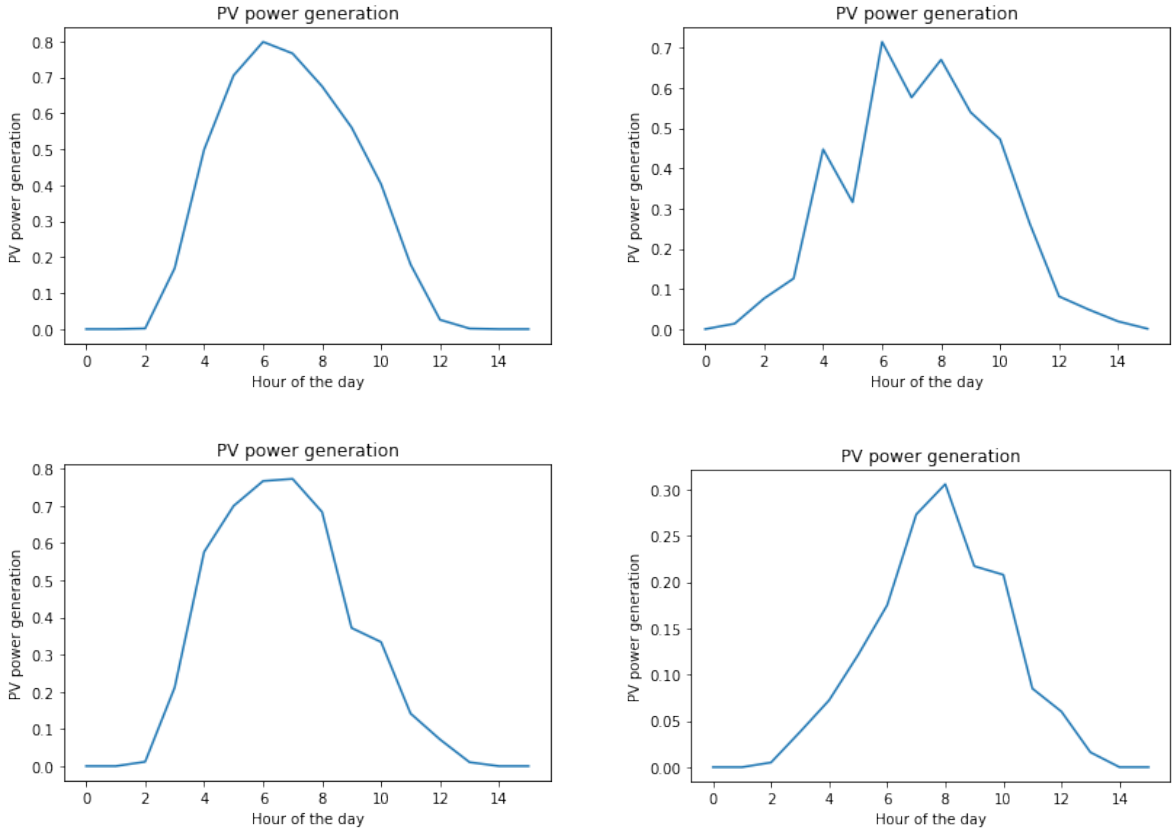


Figure 6.3: Examples of PV curves from random days of the PV track of the GEFCom 2014 dataset

6.2 Quality assessment

The task of evaluating deep generative models is not trivial. Indeed, no consensus about which metrics to use has been reached Theis et al. (2015). Rather, the best choice of metrics is task-dependent and must be carefully considered. Emphasis must be given towards choosing complementary metrics that capture different aspects of the considered phenomenon. Also, most of the work and thus the metrics proposed in the

literature to evaluate generative models are related to computer vision, meaning that they cannot be transposed to energy forecasting easily.

In this work, 6 metrics related to probabilistic forecast have been selected to assess the quality of the predictions.

6.2.1 Continuous ranked probability score

The *continuous ranked probability score* (CRPS) Gneiting and Raftery (2007) is an univariate metric. It is a quadratic measure of the difference between the forecast cumulative distribution function (CDF) and the empirical CDF of the observations. It compares the predictive marginals for each component of the random variable of interest. Formally, let $\mathcal{X}$ be a random variable, and let F be the CDF of $\mathcal{X}$ defined as $F(y) = P(\mathcal{X} \leq y)$. Let y be an observation and F, the CDF of an empirical probabilistic forecast, then the CRPS is given by

$$CRPS(F, y) = \int_{-\infty}^{\infty} [F(x) - \mathbb{1}(x \geq y)]^2 \mathrm{d}x,$$

where $\mathbb{1}$ is the identity function. It basically represents a generalization of the MAE from deterministic forecast to probabilistic forecast. As a consequence, it is negatively oriented (i.e the lower the better). It penalizes biased forecasts. Gneiting and Raftery (2007) showed that the previous integral form can be estimated by being re-written as

$$CRPS(F, x_k) = \mathbb{E}_{\mathcal{X}}|\mathcal{X} - x_k| - \frac{1}{2}\mathbb{E}_{\mathcal{X},\mathcal{X}'}|\mathcal{X} - \mathcal{X}'|, \tag{6.5}$$

where $\mathcal{X}$ and $\mathcal{X}'$ are two independent random variables distributed according to F. This is called the energy form of the CRPS because it corresponds to the one-dimensional case of the energy score.

In our case, the wind power, PV generation, and load are multivariate random variables of T dimensions: $\boldsymbol{x} \in \mathbb{R}^T$, where T is the number of time periods per day (usually 24). Thus, if one generate M scenarios per day of the testing set TS, one can considerate $\hat{x}^i_{d,k}$ to be the component k of the i^{th} scenario for the day d, and the CRPS per marginals $k = 1, 2, .., T$ is given by

$$CRPS_{d,k} = \frac{1}{M}\sum_{i=1}^{M}\left|\hat{x}^i_{d,k} - x_{d,k}\right| - \frac{1}{2M^2}\sum_{i,j=1}^{M}\left|\hat{x}^i_{d,k} - \hat{x}^j_{d,k}\right|,$$

as presented in Dumas et al. (2021). Then,

$$CRPS_k = \frac{1}{\#TS}\sum_{d \in TS} CRPS_{d,k}$$

is the average over the testing set. Finally, $CRPS_k$ is averaged over all time periods:

$$\overline{CRPS} = \frac{1}{T}\sum_{k=1}^{T} CRPS_k.$$

6.2.2 Quantile score

The quantile score (QS) gives information about the forecasts at different probability levels, i.e over-forecasting or under-forecasting. It is also negatively oriented. Let τ be the quantile index ($\tau = 0.01, .., 0.99$), let $\hat{\boldsymbol{x}}_d^q$ be the q^{th} quantile for the d^{th} day. The quantiles are computed from the set of M scenarios. Let $\hat{x}_{d,k}^q$ be the k^{th} time period of $\hat{\boldsymbol{x}}_d^q$. Then, the Quantile score is

$$\rho_q\left(\hat{x}_{d,k}^q, x_{d,k}\right) = \begin{cases} (1-\tau) \times \left(\hat{x}_{d,k}^q - x_{d,k}\right) & x_{d,k} < \hat{x}_{d,k}^q \\ \tau \times \left(x_{d,k} - \hat{x}_{d,k}^q\right) & x_{d,k} \geq \hat{x}_{d,k}^q, \end{cases}$$

as given by Dumas et al. (2021). One can see that ρ_q assigns asymmetric weights to positive and negative errors for each quantile. Then $\rho_q\left(\hat{x}_{d,k}^q, x_{d,k}\right)$ is averaged over the T time periods and over the entire training set

$$QS_d = \frac{1}{\#TS} \sum_{d \in TS} \frac{1}{T} \sum_{k=1}^{T} \rho_q\left(\hat{x}_{d,k}^q, x_{d,k}\right)$$

Finally, QS_d is averaged over all quantiles:

$$\overline{QS} = \frac{1}{99} \sum_{q=1}^{99} QS_d.$$

6.2.3 Reliability diagram

Reliability diagrams are a visual assessment that evaluates the quality of the quantiles derived from a set of scenarios. Generated quantiles are reliable if their positions are equal to those of the observed value. The distribution for a perfect reliable forecast is a 45° diagonal. The more the reliability diagram is close to that diagonal, the better the forecasts. Figures 6.4 and 6.5 shows examples of typical shapes for reliability diagrams. For sake of simplicity, when the quality metrics are gathered on a table, the reliability is measured with the mean absolute error between the reliability curve and the 45° diagonal.

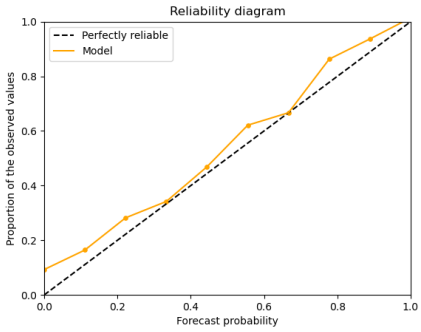


Figure 6.4: Example of reliability diagram with under-forecasting (forecast probability smaller than observed frequency)

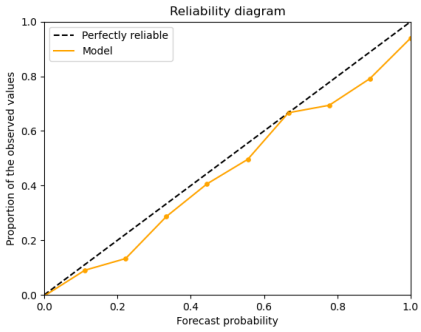


Figure 6.5: Example of reliability diagram with over-forecasting (forecast probability greater than observed frequency)

6.2.4 Energy score

The *energy score* (ES) is a commonly used scoring rule when the density is described by a finite number of samples (Golestaneh et al., 2016). It is a multivariate generalization of the CRPS. It has been introduced by Gneiting and Raftery (2007). Similar to the CRPS, it is negatively oriented. It is given by

$$\mathrm{ES}(F, \mathbf{x}) = \mathbb{E}_{\mathcal{X}} \|\mathcal{X} - \mathbf{x}\| - \frac{1}{2}\mathbb{E}_{\mathcal{X},\mathcal{X}'} \|\mathcal{X} - \mathcal{X}'\|, \tag{6.6}$$

where $\mathcal{X}$ and $\mathcal{X}'$ are independent random variables of distribution F and with finite first moment and $||.||$ is the Euclidean norm. One can notice that if we consider the marginals of $\mathbf{x}$ in Eq. (6.6), it becomes equivalent to the energy form of the CRPM given by Eq. 6.5. In practice, it is computed using the following:

$$\mathrm{ES}_d = \frac{1}{M}\sum_{i=1}^{M} \left\|\hat{\mathbf{x}}_d^i - \mathbf{x}_d\right\| - \frac{1}{2M^2}\sum_{i,j=1}^{M} \left\|\hat{\mathbf{x}}_d^i - \hat{\mathbf{x}}_d^j\right\|,$$

with the same notation as previously. Finally, the energy score is averaged over the testing set

$$\mathrm{ES} = \frac{1}{\#TS}\sum_{d \in TS} \mathrm{ES}_d$$

Golestaneh et al. (2016) investigates the ES as a scoring rule and determined that ES has a good discriminative power. However, it is a weak scoring rule when the only distinction between forecasts is the correlation between their components. The ES will be used to determine the performances comparably to how the MAE would be used for point forecasts.

6.2.5 Variogram score

Another class of proper scoring rule is the *variogram score* proposed by Scheuerer and Hamill (2015). It is based on the geo-statistical concept of variograms. In Scheuerer and Hamill (2015), the sensitivity of the variogram score to incorrectly predicted means, variances, and correlations in a number of examples with simulated observations and forecasts is studied. It is shown that this scoring rule is more discriminative to the correlation structure, thus it makes it a nice complement to the Energy score as it is where the ES is lacking discriminative power. The variogram score of order γ is given by

$$\mathrm{VS}_d = \sum_{k,k'}^{T} w_{kk'} \left(|x_{d,k} - x_{d,k'}|^\gamma - \mathbb{E}_P\, |\hat{x}_{d,k} - \hat{x}_{d,k'}|^\gamma\right)^2, \tag{6.7}$$

where $\hat{\mathbf{x}}_d \in \mathbb{R}^T$ and $x_{d,k}$ and $x_{d,k'}$ are respectively the k^{th} and the k'^{th} components of the random vector $\hat{\mathbf{x}}_d$ distributed according to P, and $\mathbf{x}_d$ is the multivariate observation. It is important to note that the γ^{th} absolute moment of P must exist. Also, $w_{kk'}$ are non-negative weights. Given a set of M scenarios $\{\mathbf{x}_d^i\}_{i=1}^M$ for each given day d, the expectation can be approximated by

$$\mathbb{E}_P\, |\hat{x}_{d,k} - \hat{x}_{d,k'}|^\gamma \approx \frac{1}{M}\sum_{i=1}^{M} \left|\hat{x}_{d,k}^i - \hat{x}_{d,k'}^i\right|^\gamma. \tag{6.8}$$

Then, the variogram score is averaged over the days of the testing set.

$$\mathrm{VS} = \frac{1}{\#TS} \sum_{d \in TS} \mathrm{VS}_d.$$

For the experiments, it has been chosen to use equal weights across the hours $w_{kk'} = 1$ and $\gamma = 0.5$ as suggested in Dumas et al. (2021).

6.2.6 Correlation matrix

The last considered quality metric is the correlation matrix between the scenarios given weather forecasts. Let $\{\mathbf{x}_d^i\}_{i=1}^M$ be a set of M scenarios for a given day d. This set can be seen as a $(M \times 24)$ matrix where each line is a scenario. The Pearson's coefficient between the scenarios can then be computed, which leads to a (24×24) matrix. This metric carries information about the variety of the scenarios for a given day. The higher the correlation, the smaller the variety, and the scenarios resemble each other, and reversely. This metric can also be averaged over all the days of the testing set to have more of a global insight.

6.3 Value assessment

The previously presented metrics measure the quality of the forecasts. However, a model with better quality metrics is not necessarily the most effective model to use (Hong et al., 2020). It is interesting to estimate the value of the different techniques to generate day-ahead scenarios. This is achieved by considering the day-ahead market scheduling of electricity aggregators such as energy retailers as shown by Toubeau et al. (2019); Dumas et al. (2021). The retailer's goal is to balance its portfolio on an hourly basis so as to avoid financial penalties in case of imbalance by exchanging the surplus or deficit of energy in the day-ahead electricity market. In the chosen formulation for the optimization problem (which is heavily inspired by the case study carried by Dumas et al. (2021)), the retailer is interested in day-ahead bidding. Its portfolio is composed of load, wind power generation, and PV generation. It also has access to a battery energy storage system (BESS) to manage its portfolio and minimize imports from the grid when day-ahead prices are restrictive. The methodology is summarized by figure 6.6.

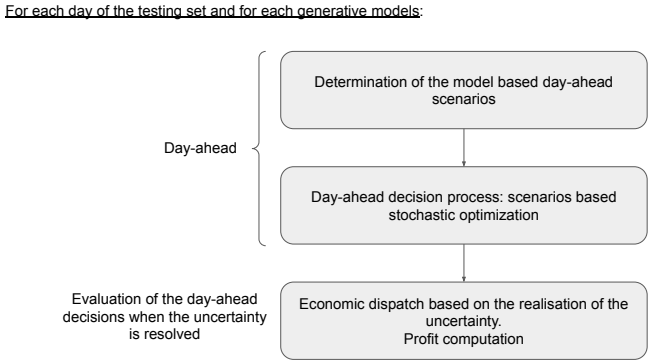


Figure 6.6: Methodology used to compare the value of the day ahead decisions of the different models.

The retailer bids on the day-ahead market by computing a planning based on stochastic optimization. Then the dispatch is computed thanks to the real observation of the forecasted variables, namely, the PV generation, wind power, and the load. Finally, the profits are computed and can be compared for each generative model. The first step of the methodology described in figure 6.6 is model-dependent, it consists of generating the scenarios with the different generative models for all the days of the testing set. The following sub-section describes the stochastic optimization problem and the dispatching.

6.3.1 Scenarios based stochastic optimization

A stochastic planner with a linear programming formulation and linear constraints is used. We have,

- e_t [MWh] the net energy retailer position for the t-th hour of the day.
- y_t [MWh] the realized net energy retailer position for the t-th hour of the day.
- π_t [€/ MWh] the clearing price in the day-ahead market for the t-th hour of the day.
- q_t, the posterior settlement price for negative imbalance $y_t < e_t$.
- λ_t, the posterior settlement price for positive imbalance $y_t > e_t$.

The retailer is assumed to be a price taker in the day ahead market. Indeed, usually, the capacity of an individual retailer is negligible in comparison to the whole market, thus π_t is chosen to be fixed and known. The imbalance prices are chosen to be random variables as they manifest volatility. Their expectation are noted as $\bar{q_t} = \mathbb{E}[q_t]$ and $\bar{\lambda_t} = \mathbb{E}[\lambda_t]$. e_t is a first stage variable and y_t is chosen to be a second stage variable because of the stochastic nature of the PV generation, wind generation, and load. The planner computes the day-ahead bids e_t. Those bids cannot be changed later when the uncertainty is solved. The dispatch decisions $y_{t,\omega}$ in scenario ω correspond to the second stage. The goal of the dispatch decisions is to avoid portfolio imbalances. Those imbalances are modeled by a cost function f^c. Thus, the stochastic planner's objective is to maximize:

$$J_S = \mathbb{E}\left[\sum_{t\in\mathcal{T}} \pi_t e_t + f^c\left(e_t, y_{t,\omega}\right)\right] \tag{6.9}$$

The expectation is taken over the PV generation, wind generation, load, and the random variables. Thanks to a scenarios-based approach, it is approximated by

$$J_S \approx \sum_{\omega\in\Omega} \alpha_\omega \sum_{t\in\mathcal{T}} \left[\pi_t e_t + f^c\left(e_t, y_{t,\omega}\right)\right] \tag{6.10}$$

with $\omega \in \Omega$, α_ω the probability of the scenario ω and $\sum_{\omega\in\Omega} \alpha_\omega = 1$. More details about the value assessment can be found in appendix E.

Chapter 7

Experiments

This section will go through the different experiments primarily related to the hyperparameters of diffusion models. The methodology will be to fit a model on a learning set (either the load, PV, or wind track) and compare the different configurations of hyperparameters by testing on the validation set in order to gain insight about diffusion models parameters, and finally display the results on the testing set. The comparisons are performed by generating 100 scenarios per day of the testing set. The comparisons performed in this section are solely based on quality metrics. Then, section 8 uses both quality and value assessment in order to compare diffusion models to the different state-of-the-art deep generative models.

7.1 Training loss

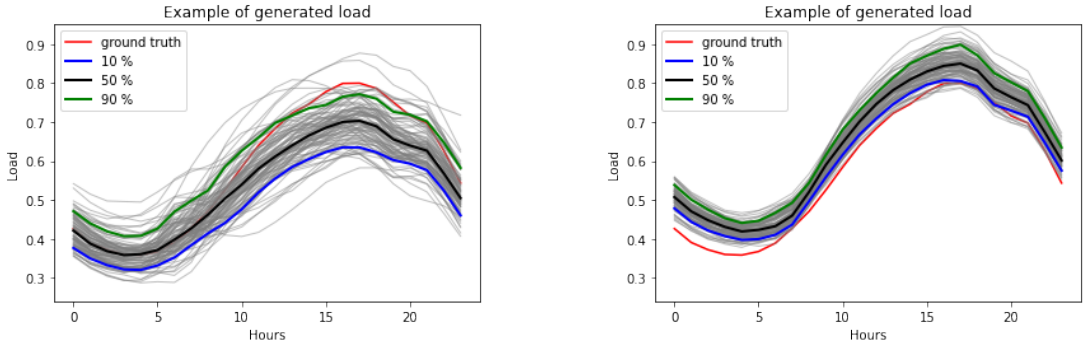
It has been shown empirically by Ho et al. (2020) that the simplified loss function (3.13) lead to better sample quality than the analytically derived loss function (3.12) for computer vision applications. The purpose of the following experiment is to assess if this result holds for time series data as well. Figures 7.1, 7.2, and 7.3 show examples of scenarios and the quantiles derived from the scenarios generated with models trained with the two different loss functions for respectively the load, PV, wind tracks. Table 7.1 summarizes the quality results for the three tracks with models trained with the two loss function. It can be seen that almost independently of the tracks, the simplified loss function lead to significantly better quality results. Thus, the simplified loss function is chosen for the rest of this thesis.

7.2 Training loop

The second experiment is related to the number of diffusion steps per sample during training. There are two possible ways of doing so. First, one could start from a sample and perform all the diffusion steps from 0 to T with that sample before passing to the next samples as on algorithm 1. On the other hand, one could perform only one random step of the diffusion process per sample as with algorithm 2, in this case, more epochs must be done in order to maintain the number of forward passes per sample constant. This is similar to the idea behind stochastic gradient descent but applied to the diffusion steps of the forward process. It is important to note that the only difference between the two algorithms is the order in which the model sees the noised samples of the diffusion process. In both cases, the model sees all the samples with all

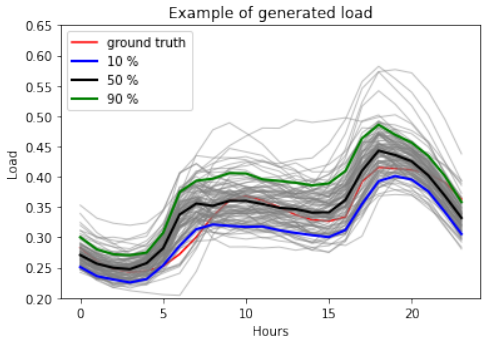
		Full loss function	Simplified loss function
Wind	ES	66.32	**62.05**
	VS	21.31	**19.39**
	QS	5.43	**5.12**
	CRPS	10.74	**10.16**
	MAE-r	8.71	**6.02**
PV	ES	35.72	**22.36**
	VS	9.54	**4.51**
	QS	2.0	**1.18**
	CRPS	3.95	**2.34**
	MAE-r	5.71	**3.74**
Load	ES	16.13	**11.03**
	VS	2.97	**1.53**
	QS	1.40	**0.97**
	CRPS	2.77	**1.92**
	MAE-r	**3.68**	18.72

Table 7.1: Comparison between models trained with analytically derived loss function or a simplified one. All the hyper-parameters remain constant between the two models except the loss function used for training.
The simplified loss function lead to better scores on almost all the quality metrics over the three tracks.

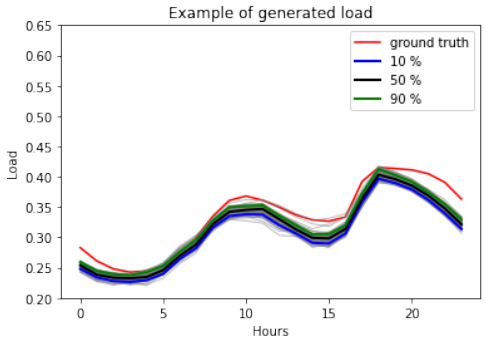


(a) Full loss function

(b) Simplified loss function



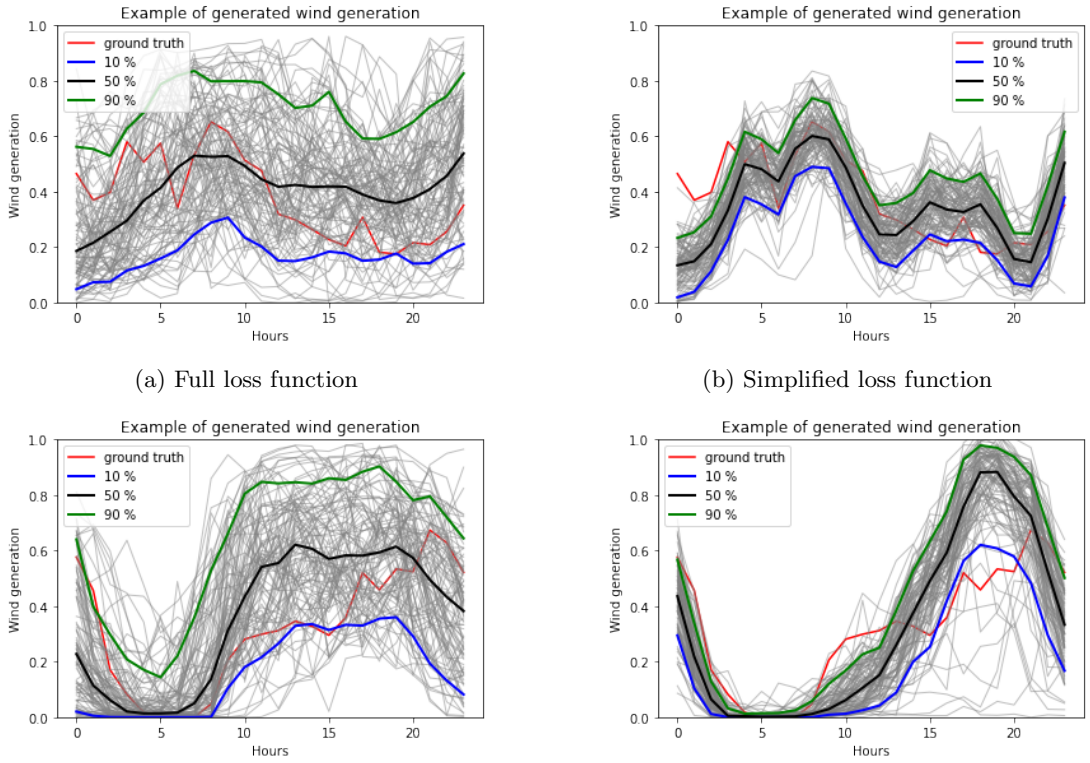
(c) Full loss function



(d) Simplified loss function

Figure 7.1: Examples of load scenarios with the two loss functions for two random days of the dataset.7.1a and 7.1b are scenarios for the same day, and 7.1c and 7.1d correspond to another day. The model trained with the simplified loss function tends to generate scenarios that exhibit less variability and more temporal correlation. The range of values of the scenarios is closer to the ground truth.

the diffusion steps (on expectation), only the order changes. Table 7.2 compares the two algorithms. If the number of forward passes is the same, none of the algorithms has



(a) Full loss function

(b) Simplified loss function

(c) Full loss function

(d) Simplified loss function

Figure 7.2: Examples of wind power scenarios with the two loss functions for two random days of the dataset.7.2a and 7.2b are scenarios for the same day, and 7.2c and 7.2d correspond to another day. The model trained with the simplified loss function tends to generate scenarios that exhibit much less variability and more temporal correlation. The range of values of the scenarios is also closer to the ground truth. Note: whatever the training loss, the wind power scenarios tends to be more difficult to predict than the PV power and the load. It is due to its more stochastic nature.

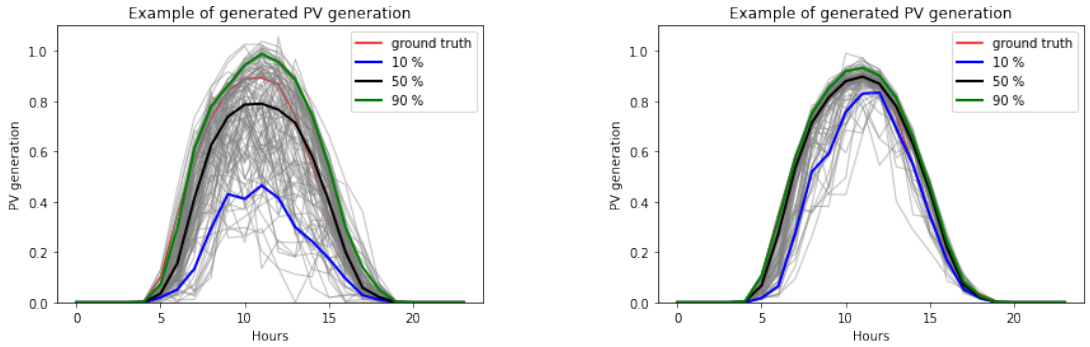
an edge when it comes to the computational complexity, but as can be seen in table 7.2, algorithm 2 outperforms algorithm 1. For the rest of the thesis, algorithm 2 is chosen.

7.3 Diffusion steps

Here, we are interested in exploring the relationship between the number of diffusion steps (for training and sampling) and the quality of the scenarios generated. The results for each track with several values for the number of diffusion steps are shown in table 7.3. Overall, as could be expected, higher diffusion steps tend to increase the quality of the generated scenarios. For the PV and load track, the best size for the diffusion process is equal to $T = 400$, whereas, for the wind track, it is equal to $T = 200$. Thus, those values for the diffusion process length are used for the rest of the thesis.

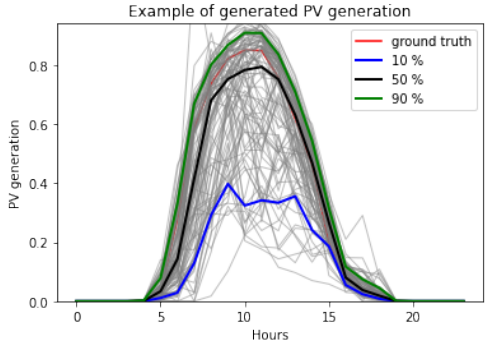
7.4 Sampling steps ablation

One of the main problems related to diffusion models is related to sampling. Indeed, usually generating a scenario with a model trained with T forward pass requires also using T forward pass at sampling time. It is particularly expensive in comparison to other models that only require one. The purpose of the following experiment is to

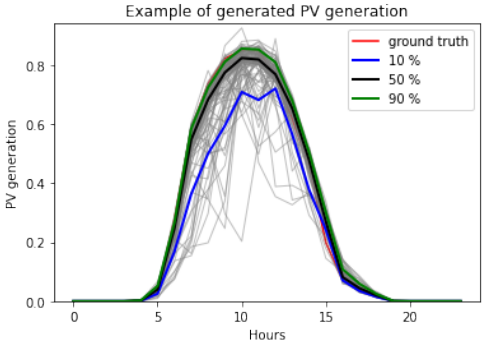


(a) Full loss function

(b) Simplified loss function



(c) Full loss function



(d) Simplified loss function

Figure 7.3: Examples of PV scenarios with the two loss functions for two random days of the dataset.7.3a and 7.3b are scenarios for the same day, and 7.3c and 7.3d correspond to another day. The model trained with the simplified loss function tends to generate scenarios that exhibit less variability and more temporal correlation. The range of values of the scenarios is also closer to the ground truth.

look at ways of using fewer steps at sampling time than the number of steps used at training time. The idea consists of generating scenarios with fewer diffusion steps than the number of steps used at training time to look at the marginal gain of using more steps at sampling time. The quality results can be found in table 7.4. The T_{infer} steps are uniformly spaced between 0 and T. It can be seen that the best quality is obtained by using the same number of steps at training and at sampling time. However, the marginal gain decreases such that for instance, the quality metrics obtained by using $T_{infer} = 0.8 \times T$ is not significantly better than by using $T_{infer} = T$.

		iterative training	stochastic training
Wind	ES	62.24	**55.79**
	VS	19.56	**17.47**
	QS	5.17	**4.56**
	CRPS	10.27	**9.04**
	MAE-r	4.13	**1.38**
PV	ES	36.05	**29.37**
	VS	9.77	**6.88**
	QS	2.05	**1.62**
	CRPS	4.05	**3.21**
	MAE-r	5.76	**3.77**
Load	ES	16.9	**11.66**
	VS	3.7	**1.87**
	QS	1.45	**1.02**
	CRPS	2.86	**2.01**
	MAE-r	**12.27**	15.48

Table 7.2: Comparison between iterative training and stochastic training.
The stochastic training consistently outperforms the iterative training on the quality metrics with the same number of forward passes.

		T = 50	T = 100	T = 200	T = 400
Wind	ES	69.20	66.40	**54.47**	61.84
	VS	22.34	21.03	**17.29**	19.57
	QS	5.74	5.50	**4.41**	5.14
	CRPS	11.35	10.87	**8.73**	10.17
	MAE-r	11.03	9.95	**3.84**	10.52
PV	ES	24.77	22.11	23.86	**21.60**
	VS	5.25	4.43	4.80	**4.16**
	QS	1.34	1.17	1.28	**1.14**
	CRPS	2.65	2.31	2.53	**2.26**
	MAE-r	9.84	**3.81**	7.59	8.09
Load	ES	10.81	11.07	10.30	**9.78**
	VS	1.58	1.53	**1.42**	1.49
	QS	0.94	0.97	0.89	**0.85**
	CRPS	1.87	1.93	1.75	**1.69**
	MAE-r	13.84	18.80	11.26	**9.43**

Table 7.3: Comparison of scenarios obtained with models trained with 4 total diffusion steps for both training and sampling (T=50, T=100, T=200, T=400). As could be expected, higher diffusion steps usually lead to better quality metrics.

		$\frac{T_{infer}}{T} = 1\%$	$\frac{T_{infer}}{T} = 5\%$	$\frac{T_{infer}}{T} = 10\%$	$\frac{T_{infer}}{T} = 20\%$	$\frac{T_{infer}}{T} = 50\%$	$\frac{T_{infer}}{T} = 80\%$	$\frac{T_{infer}}{T} = 100\%$
Wind	ES	103.65	99.77	94.01	85.18	65.71	56.05	**55.91**
	VS	35.26	33.57	33.72	28.14	21.14	17.74	**17.71**
	QS	8.92	8.56	8.11	7.24	5.45	4.57	**4.55**
	CRPS	17.64	16.94	16.05	14.32	10.80	9.03	**8.99**
	MAE-r	4.67	4.81	4.85	4.65	3.79	1.66	**1.62**
PV	ES	48.00	44.99	41.55	36.31	25.48	21.68	**21.37**
	VS	15.57	13.83	12.51	9.48	5.44	4.21	**4.11**
	QS	2.84	2.64	2.41	2.04	1.35	1.14	**1.13**
	CRPS	5.64	5.22	4.78	4.04	2.69	2.26	**2.23**
	MAE-r	4.03	4.02	3.91	3.90	**3.70**	7.91	7.75
Load	ES	42.75	37.68	32.45	24.32	12.96	9.73	**9.72**
	VS	13.98	11.45	8.90	5.47	1.95	1.48	**1.44**
	QS	3.60	3.14	2.71	2.05	1.14	0.85	**0.84**
	CRPS	7.11	6.24	5.38	4.07	2.24	1.68	**1.67**
	MAE-r	7.04	**7.01**	7.44	8.01	8.34	9.54	9.44

Table 7.4: Quality metrics using T_{infer} diffusion steps at sampling time with T_{infer} different from T. The best results are obtained by using $T_{infer} = T$. But the marginal gain of using more steps decreases a lot after using about 80% of the steps used at training time.

Chapter 8

Results

This section provides a comparison between the DDPM and the three other deep generative models (GAN, VAE, NF). The comparison is three-fold. First, based on quality metrics, they measure the ability of a model to capture the stochastic process in order to generate good scenarios. Second, based on the forecast value, it represents the actual financial benefit of using one model over the other in a decision-making process. Finally, subjectively, by inspecting the generated scenarios in order to get the intuition behind the predictions of each model.

8.1 Quality assessment

Several metrics have been considered for the quality assessment. The assessment is performed by generating 100 scenarios per day of the testing set. The different metrics are computed based on those 100 scenarios and the ground truth for each day of the testing set. The metrics are presented in section 6.2. Table 8.1 summarizes the results for the four models and the three tracks. It can be seen that for the wind track, the DDPM is the best performing deep generative model on all the quality metrics. For the PV track, the DDPM is the best performing model on all the metrics except the MAE-r, where the NF outperforms it. Finally, when it comes to the load, the NF is the best performing model on all the metrics except the variogram score for which DDPM achieves the best score.

Figures 8.1, 8.3, and 8.5 displays scenarios and the derived percentiles for the 4 models for respectively the load track, wind track and PV track. Figures 8.2, 8.4 and 8.6 illustrates the Pearson correlation between the 100 scenarios displayed on figures 8.1, 8.3, and 8.5. One can see that a tendency emerges. On one hand, we have GAN and VAEs, where the scenarios are really similar and do not cross each other. On the other end of the spectrum, there is the NF where the scenarios tend to show a high level of variation. In the middle of those extremes, we have the DDPM for which the scenarios also exhibit variability and cross each other, but to a smaller extent. This tendency is confirmed by the correlation matrices. It can be seen on figures 8.7g, 8.7h, and 8.7i that the NF demonstrates no time correlation at all. Figures 8.7a, 8.7b, 8.7c, 8.7d, 8.7e, and 8.7f exhibit a high temporal correlation for GAN and VAE. Finally, figures 8.7g, 8.7h, and 8.7i show that DDPM sits in the middle, with a moderate time correlation in comparison to GAN and VAE for the wind and the load forecasting, and a low correlation for the PV but still displays more time correlation than the NF. Similar plot with scenarios and correlation matrices for another day of the training set can be

		VAE	GAN	NF	DIFF
Wind	ES	54.82	60.52	56.71	**54.47**
	VS	17.87	19.87	18.54	**17.29**
	QS	4.45	4.95	4.58	**4.41**
	CRPS	8.80	9.79	9.07	**8.73**
	MAE-r	2.67	6.82	2.83	**1.35**
PV	ES	24.65	24.15	23.08	**21.60**
	VS	5.02	4.88	4.68	**4.16**
	QS	1.31	1.32	1.19	**1.14**
	CRPS	2.60	2.61	2.35	**2.26**
	MAE-r	9.04	4.94	**2.66**	8.06
Load	ES	15.11	17.96	**9.17**	9.76
	VS	1.66	3.81	1.63	**1.49**
	QS	1.39	1.52	**0.76**	0.8
	CRPS	2.74	3.01	**1.51**	1.69
	MAE-r	13.97	9.99	**7.70**	9.43

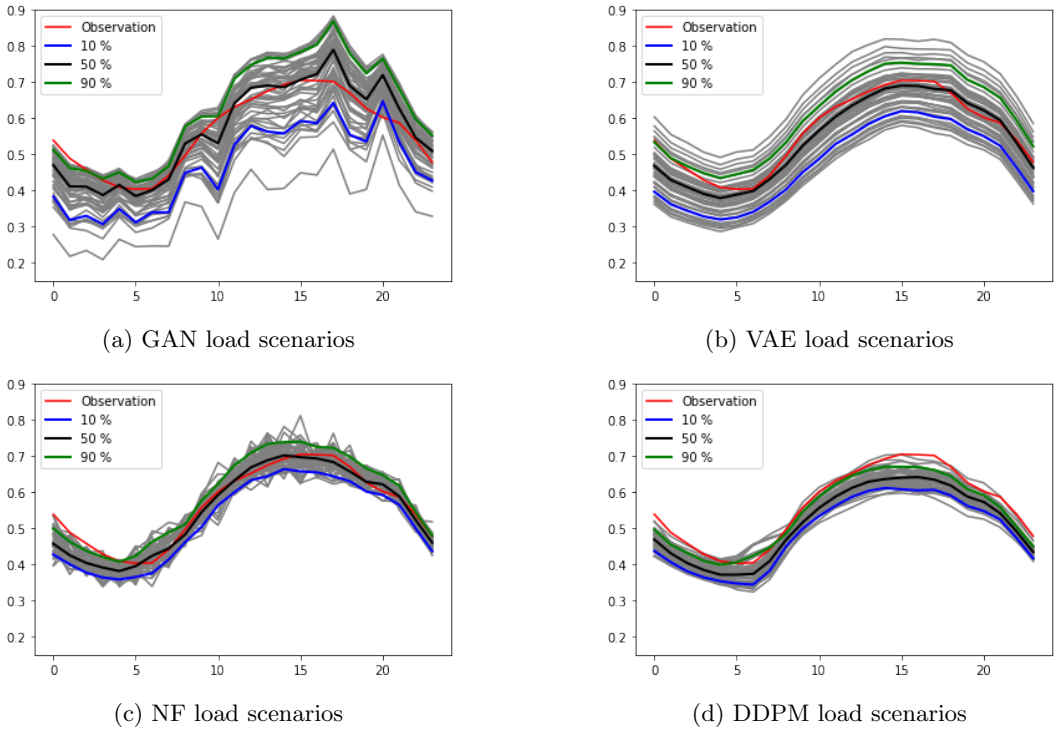
Table 8.1: Summary of the averaged quality metrics per model and per track (wind, PV, or load).
The CRPS is averaged over the 24 time periods, the QS is averaged over 99 percentiles. The MAE-r is a numerical expression associated with the reliability diagram, it is the mean absolute error between the reliability curve and the diagonal line.

found in appendix F.

8.2 Value assessment

As explained earlier, the value assessment is done through a case study of an energy retailer having a portfolio with wind power, PV generation, load, and a battery energy storage system. The test set contains 50 days. In addition to that, the wind track is composed of 10 different zones, and the PV track is composed of 3 different zones. This lead to 1500 independent days for the simulations. The forecast value is evaluated in a two-step approach. First, compute the day ahead decisions through an optimization problem using the day-ahead scenarios generated by each generative model. Then, real-time dispatch is performed based on the true observations and the day-ahead decisions. The goal of the retailer is to balance the net power by importing or exporting from the main grid. The net power is given by the red curve in figure 8.8. Even though it is an arbitrary day of the testing set, it exhibits the general tendency being that the net is positive at noon when the PV generation is maximal, and the load is minimal. It is negative in the evening when the PV generation is minimal, and the load is maximal. Another issue that the retailers have to face is that usually, the spot day-ahead prices are often maximal at the end of the day when the load is maximum. To circumvent that, retailers try to save power by charging batteries during the day in order to prevent import in the evening. That's why forecasting is really important, as the better the forecasts, the better the planning.

Ultimately, the goal of the energy retailer is to maximize its net profit. It is computed as the profit minus the penalty. Table 8.2 displays the net profit for all the generative models. The model leading to the highest profit is the diffusion model with a 112 k€ net profit. It comes close to the normalizing flow with a 107 k€ profit. The oracle column corresponds to a forecast possessing knowledge of the future. It serves as a benchmark to show that there is still room for improvement. To conclude, when it comes to the forecast value, the DDPM slightly outperforms the NF, GAN, and VAE



(a) GAN load scenarios

(b) VAE load scenarios

(c) NF load scenarios

(d) DDPM load scenarios

Figure 8.1: Example of load scenarios generation for an arbitrary day of the testing set with the 4 generative models. 50 load scenarios with the derived percentile and the corresponding real observation.
The GAN and VAE scenarios almost look like a translation of one another, they are highly correlated. On the other hand, NF scenarios tend to cross each other and seem noisy, and DDPM scenarios are more similar to one another but seem to be less correlated to one another than those of GAN and VAE.

on this simple case study.

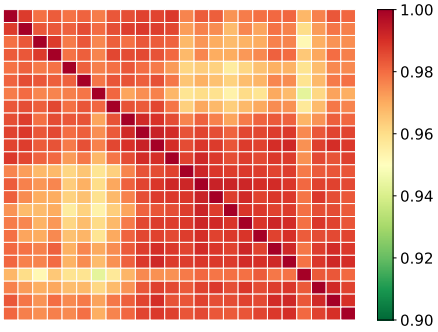
	GAN	VAE	NF	DDPM	Oracle
Net profit (k€)	93	97	107	**112**	298

Table 8.2: Total net profit. The stochastic planner achieved the highest profit using the PV, load, and wind scenarios generated by the DDPM with 112 k€. The oracle (O) corresponds to a forecast always equal to the actual observation. It represents having full knowledge of the future. Its profit is the highest achievable profit for a forecasting method.

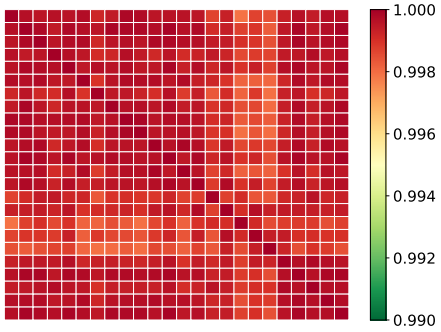
8.3 Comparison

This section summarizes the differences, advantages, and shortcomings of the different methods. First, figure 8.9 reminds the high-level working of each of the four deep generative models.

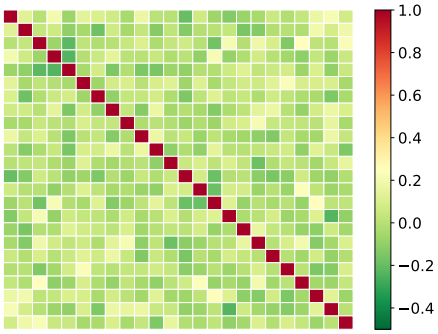
First, when it comes to the quality assessment, it has been seen that the DDPM and NF outperformed the GAN and VAE models. For the value assessment, the DDPM lead to the highest profit followed by the NF, the VAE, and the GAN. However, there are other aspects to mention for the adoption of those models in the power system community.



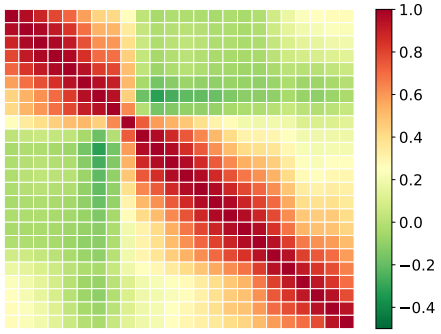
(a) Pearson time correlation matrix of GAN load model.



(b) Pearson time correlation matrix of VAE load model.



(c) Pearson time correlation matrix of NF load model.

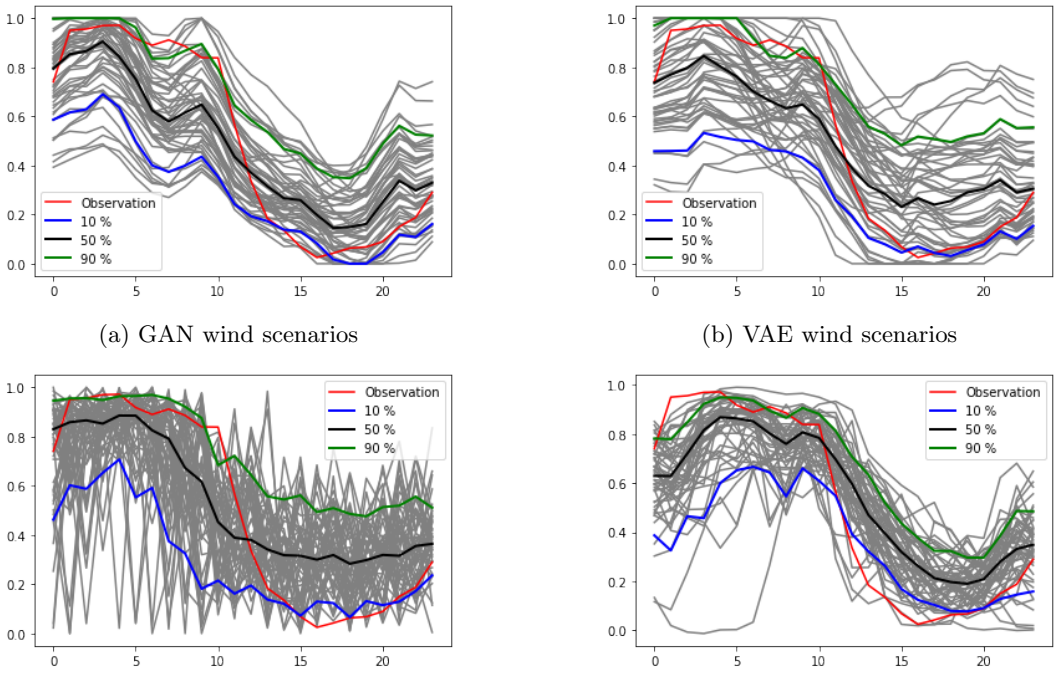


(d) Pearson time correlation matrix of DDPM load model.

Figure 8.2: Pearson correlation matrix associated with the load scenarios of the arbitrary day of figure 8.1. The matrices corroborate the intuition given by the scenarios shapes on figure 8.1. We have a really high correlation for both GANs and VAE for all the hours of the day, with a slight advantage for the VAE. On the other hand, the NF comes with no time correlation at all and the scenarios seem really noisy (but they all gravitate around the observation). Finally, the DDPM displays more correlation than the NF, but less than both GAN and VAE. With DDPM, some hours of the day are correlated.

Training and sampling speed For both the training and the sampling speed, the VAE is the fastest for training and for the generation of the scenarios, followed by the GAN, the NF and finally, the DDPM is significantly slower to train and especially slow to generate the samples due to its sequential nature. Also, DDPM and NF have the drawback of having latent spaces of the same dimension as the input dimension, which lead to expensive computations.

Hyper parameters When it comes to the stability of the model with respect to its hyperparameters, the GAN is the worst in the sense of a slight change in the hyperparameter can lead to a model generating poor samples. On the other hand, NF and DDPM models tend to be robust when it comes to their hyperparameters and overall easier to train. The VAE model sits close to NF and DDPM by being much easier to train than GAN. For DDPM, it might be explained by the fact that the latent space is easily interpretable, each latent variable is a noised version of the input. For NF it can be explained by the numerous constraints imposed on the transformations. Also, both NF and DDPM possess a latent space with the same dimensionality as the input data. On the other hand, GAN models are notoriously known for being difficult to train Salimans et al. (2016); Arjovsky and Bottou (2017).



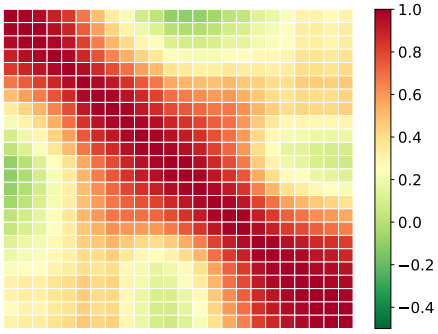
(a) GAN wind scenarios

(b) VAE wind scenarios

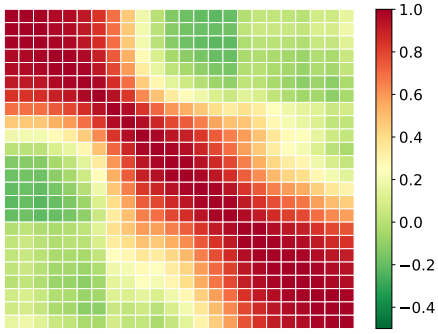
(c) NF wind scenarios

(d) DDPM wind scenarios

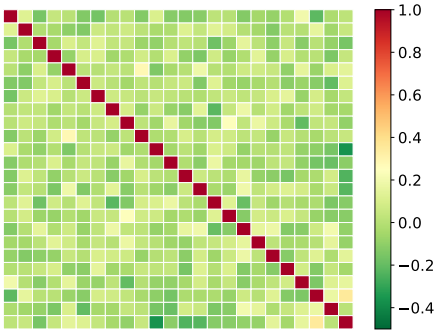
Figure 8.3: Example of wind scenarios generation for an arbitrary day of the testing set with the 4 generative models. 50 wind scenarios with the derived percentile and the corresponding real observation.
The GAN and VAE scenarios seem more correlated to each other. On the other hand, NF scenarios have a tendency to cross each other and it seems to have no correlation between the scenarios at all. DDPM scenarios are more similar to one another than NF's, but seem to be less correlated to one another than those of GAN and VAE.



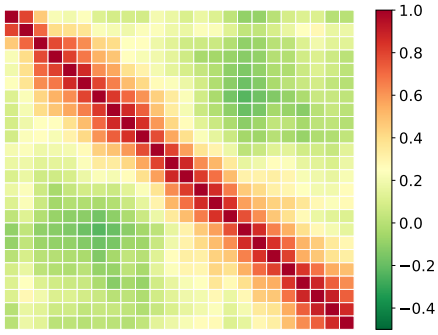
(a) Pearson time correlation matrix of GAN wind model.



(b) Pearson time correlation matrix of VAE wind model.

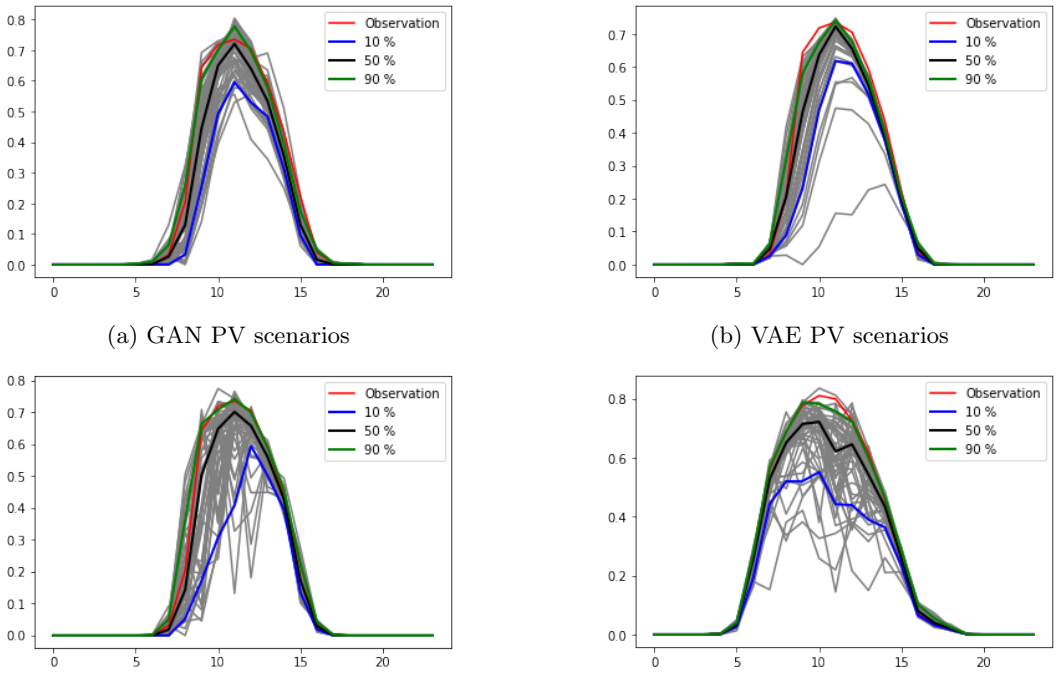


(c) Pearson time correlation matrix of NF wind model.



(d) Pearson time correlation matrix of DDPM wind model.

Figure 8.4: Pearson correlation matrix associated with the wind scenarios of the arbitrary day of figure 8.3. The matrices corroborate the intuition given by the scenarios shapes on figure 8.3. We have a really high correlation for both GANs and VAE for several hours of the day. On the other hand, the NF displays no time correlation at all and the scenarios seem noisier than the other models. Finally, the DDPM displays correlation for a couple of hours of the day but less than both GAN and VAE.



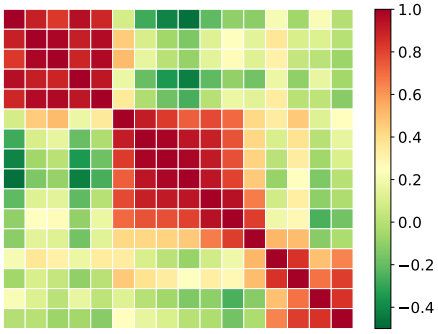
(a) GAN PV scenarios

(b) VAE PV scenarios

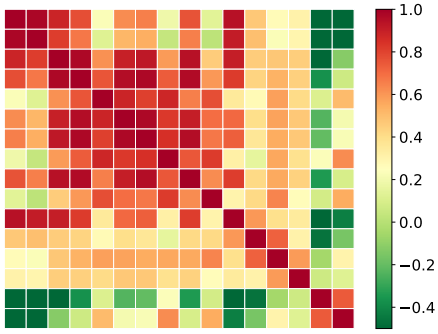
(c) NF PV scenarios

(d) DDPM PV scenarios

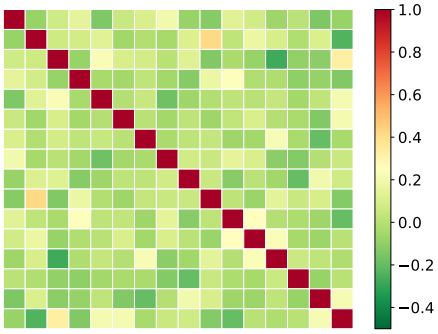
Figure 8.5: Example of PV scenarios generation for an arbitrary day of the testing set with the 4 generative models. 50 PV scenarios with the derived percentile and the corresponding real observation.



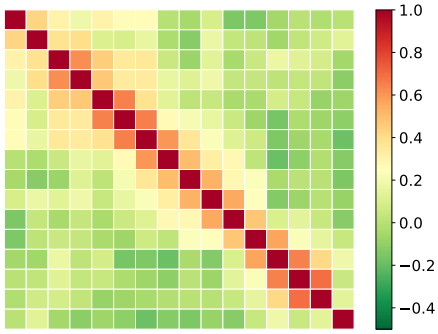
(a) Pearson time correlation matrix of GAN PV model.



(b) Pearson time correlation matrix of VAE PV model.

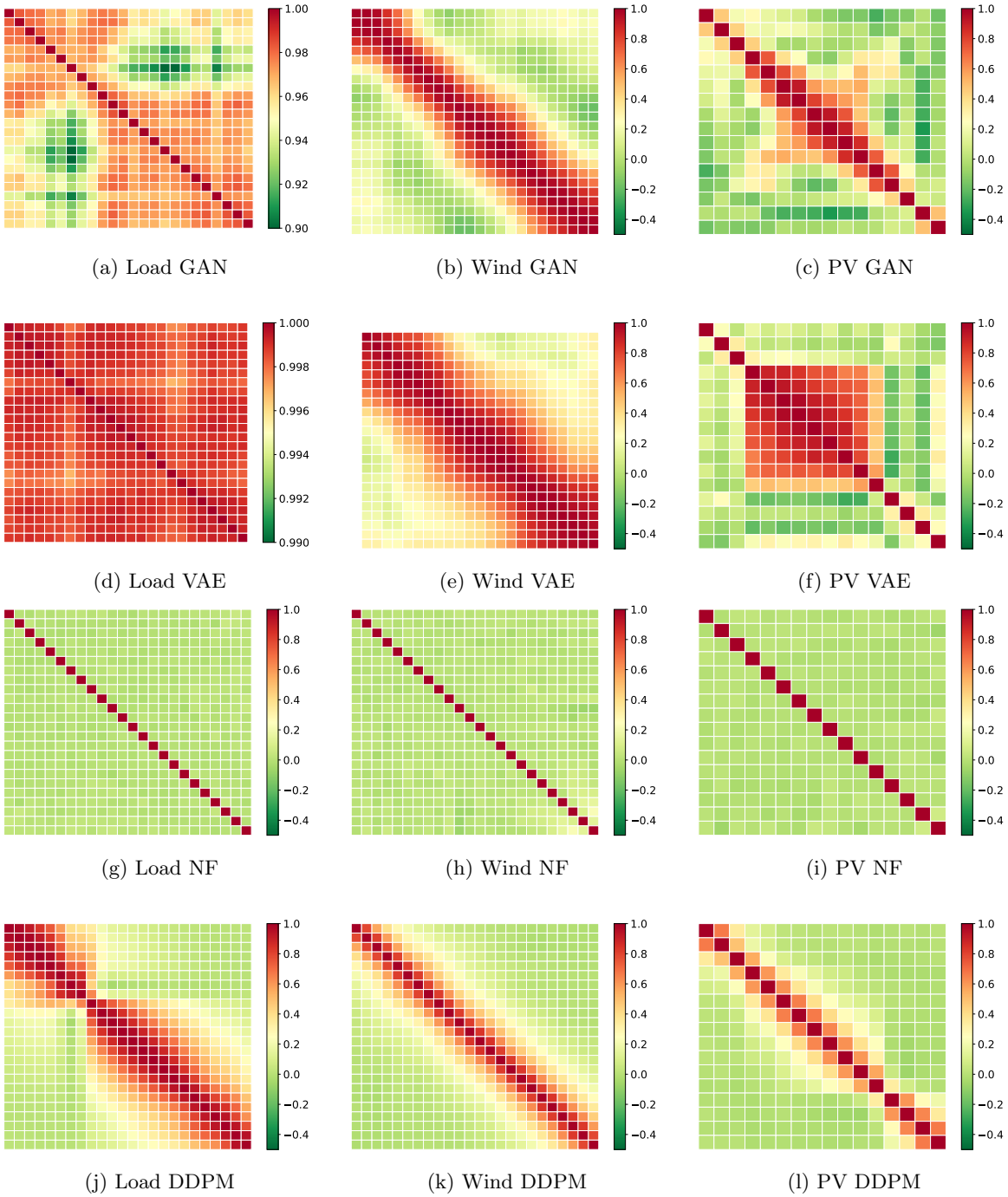


(c) Pearson time correlation matrix of NF PV model.



(d) Pearson time correlation matrix of DDPM PV model.

Figure 8.6: Pearson correlation matrix associated with the PV scenarios of the arbitrary day of figure 8.5. The matrices corroborate the intuition given by the scenarios shapes on figure 8.5. We have higher correlations for both GANs and VAE for some sparse hours of the day. Once again, there is a slight advantage for the VAE. On the other hand, the NF exhibit no time correlation at all, and the scenarios seem really noisy (but they all gravitate around the observation). Finally, the DDPM displays a little bit more correlation than the NF, but the difference is not significant.



(a) Load GAN (b) Wind GAN (c) PV GAN

(d) Load VAE (e) Wind VAE (f) PV VAE

(g) Load NF (h) Wind NF (i) PV NF

(j) Load DDPM (k) Wind DDPM (l) PV DDPM

Figure 8.7: Average correlation over the entire testing set.
Overall, VAE tends to produce highly correlated scenarios or partially correlated scenarios. GAN tends to produce moderately correlated scenarios. DDPM leads to a small correlation. Finally, NF leads to no correlation at all.

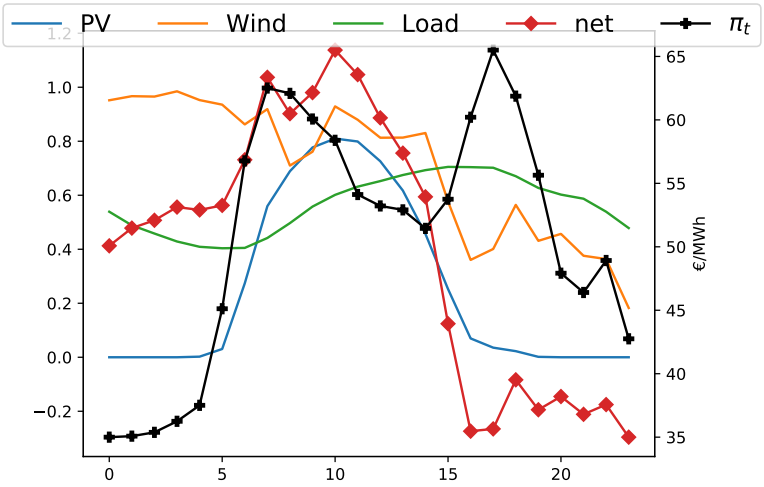


Figure 8.8: Illustration of the observation and net power of a random arbitrary day of the testing set. The energy retailer aims at balancing the power balance (net) of its portfolio. Its portfolio contains wind power generation, PV generation, load, and a battery storage system. The generated PV, wind power, and load scenarios from the testing set days are used as inputs of the stochastic day-ahead planner to compute the optimal bids.

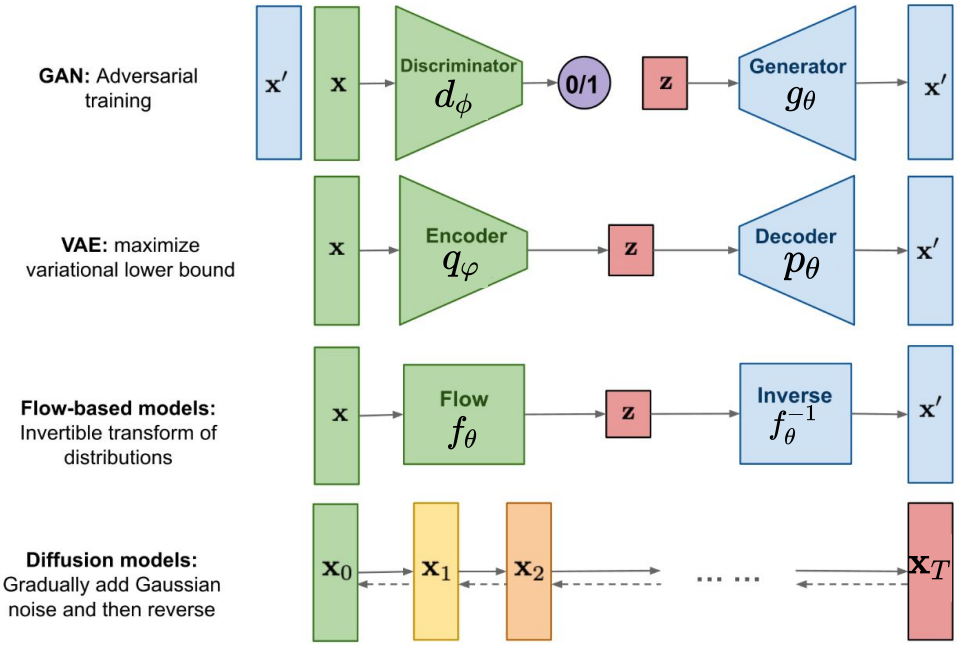


Figure 8.9: High level overview of the 4 deep generative models.
Source: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Chapter 9

Conclusion

This work provides a deep learning approach for the task of conditional probabilistic forecasting applied to power systems. A promising deep generative model has been investigated: denoising diffusion probabilistic models. Then, the developed DDPM has been thoroughly compared to state-of-the-art deep generative models, namely, generative adversarial networks, variational auto-encoders, and normalizing flows. All those models are able to integrate conditional weather information to enhance their predictions. The comparison is conducted in two ways. On one hand, based on objective quality metrics. On the other hand, based on the actual financial value of using the models in a decision-making process through the case study of an energy retailer. For both, the experiments were performed on the Global Energy Forecasting Competition 2014 open datasets. It has been shown that not only, DDPM is competitive with the other state-of-the-art deep generative models, but also that they are actually among the best performing deep generative models for this task based on both the quality and value of the scenarios generated. However, among the four models, DDPM was the slowest for both training and sampling. Finally, some experiments regarding DDPM parameters have been realized, and a suited deep learning architecture has been discussed.

Overall, this work adds a new tool to the machine learning toolbox of energy forecasting practitioners. And provides a fair comparison between 4 deep generative models on the task of probabilistic forecasting. However, the developed model still has some limitations and can be improved in several ways.

Limitation and further work Although the proposed DDPM already displayed impressive competitiveness with respect to other state-of-the-art deep generative models, some improvements are still possible. First, as seen in section 3.2 and proposed by Ho et al. (2020), the variance of the reverse process is chosen to be a fixed constant depending on the diffusion step with no learned parameters. However, more recent work (Nichol and Dhariwal, 2021; Dhariwal and Nichol, 2021) has shown interesting improvement in sample quality in computer vision applications by learning it instead.

Then, also proposed by Nichol and Dhariwal (2021) and corroborated in chapter 7, all the diffusion steps do not contribute equally to the quality of the samples. It is proposed by Nichol and Dhariwal (2021) to change the noise schedule in order to balance the relative importance of each diffusion step through the entire diffusion process.

One of the big shortcomings of diffusion models is their sampling speed. Indeed, generating one sample requires several successive forward passes in the network. This is expensive in comparison to other deed generative models that generally only require one. Some papers have proposed ways of reducing the sampling speed. Both Nichol and Dhariwal (2021) and Kong et al. (2020) proposed ways of using fewer sampling diffusion steps than used in training. Both are based on the simple idea of evenly distributing N diffusion steps between the T steps used at training and aligning the training variance schedule to the smaller one used at sampling.

Another improvement that could be brought to this work is regarding the used architecture, and particularly the conditioner. Currently, it has been chosen as a simple multi-layer perceptron. One way to improve the results could be to use instead a recurrent neural network. This would allow leverage the sequential nature of the condition weather vector because RNN are well suited for sequential data. Similarly, an attention network (Vaswani et al., 2017) could also fulfill this role.

Appendix A

Some additional considerations

A.1 Implementation details

The implementations have been conducted using Python 3.8 as well as the Pytorch (Paszke et al., 2019) and sklearn (Buitinck et al., 2013) python libraries. All the code and experiments will be made available at this github repository. The codes of the VAE, GAN and NF are based on codes available at this git gepository: `https://github.com/jonathandumas/generative-models` In addition, part of the code for the NF comes from `https://github.com/AWehenkel/Normalizing-Flows`

A.2 Comparative deep generative models hyper parameters

The hyper parameters for the GAN, VAE and NF have been chosen the same as in Dumas et al. (2021). They are reminded on the following table.

	Wind	PV	Load
Embedding Net	4×300	4×300	4×300
Embedding size	40	40	40
Integrand Net	3×40	3×40	3×40
Weight decay	5.10^{-4}	5.10^{-4}	5.10^{-4}
Learning rate	10^{-4}	5.10^{-4}	10^{-4}
Latent dimension	20	40	5
E/D Net	1×200	2×200	1×500
Weight decay	$10^{-3.4}$	$10^{-3.5}$	10^{-4}
Learning rate	$10^{-3.4}$	$10^{-3.3}$	$10^{-3.9}$
Latent dimension	64	64	256
G/D Net	2×256	3×256	2×1024
Weight decay	10^{-4}	10^{-4}	10^{-4}
Learning rate	2.10^{-4}	2.10^{-4}	2.10^{-4}

Table A.1: The first sub table displays the hyperparameters of the NF, The second sub table displays the hyperparameters of the VAE, and The third sub table displays the hyperparameters of the GAN.

Appendix B

Derivation and proof related to the loss function

This is the demonstration that allows to express the loss defined at equation (3.5) into the expression given by equation (3.7). This demonstration is taken from Ho et al. (2020).

$$
\begin{aligned}
L &= \mathbb{E}_q \left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)} \right] \\
&= \mathbb{E}_q \left[-\log p\left(\mathbf{x}_T\right) - \sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)} \right] \\
&= \mathbb{E}_q \left[-\log p\left(\mathbf{x}_T\right) - \sum_{t > 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)} - \log \frac{p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}{q\left(\mathbf{x}_1 \mid \mathbf{x}_0\right)} \right] \\
&= \mathbb{E}_q \left[-\log p\left(\mathbf{x}_T\right) - \sum_{t > 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)} \cdot \frac{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_0\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)} - \log \frac{p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}{q\left(\mathbf{x}_1 \mid \mathbf{x}_0\right)} \right] \\
&= \mathbb{E}_q \left[-\log \frac{p\left(\mathbf{x}_T\right)}{q\left(\mathbf{x}_T \mid \mathbf{x}_0\right)} - \sum_{t > 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)} - \log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right) \right] \\
&= \mathbb{E}_q \left[D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right) + \sum_{t > 1} D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right) - \log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right) \right]
\end{aligned}
$$

Appendix C

Generative adversarial networks formulations and improvements

GANs have received a lot of attention and improvement since their introduction by Goodfellow et al. (2014). THe orginal loss function was:

$$V(\phi, \theta) = \mathop{\mathbb{E}}_{\mathbf{x}} \left[\log d_\phi(\mathbf{x} \mid \mathbf{c})\right] + \mathop{\mathbb{E}}_{\hat{\mathbf{x}}} \left[\log\left(1 - d_\phi(\hat{\mathbf{x}} \mid \mathbf{c})\right)\right] \tag{C.1}$$

However, it is know that the divergences used in plain GANs are really unstable to train. A solution to this have been proposed in the form of Wasserstein GAN (Arjovsky et al., 2017). WGAN propose to use the Wasserstein-1 distance (also called Earth-Mover distance)

$$W_1(p, q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{C.2}$$

where $\Pi(p, q)$ is the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively p and q,$\gamma(x, y)$ indicates how much "mass" must be transported from x to y to tranform the distribution p into the distribution q. We have that $||x - y||$ is the cost of miving one unit of "mass" from x to y. However, Equation (C.2) is intractable because of the infimum. Thus, the Kantorovich-Rubinstein duality (Villani, 2008) is used by Arjovsky et al. (2017). It lead to the min-max problem of WGAN is expressed as

$$\theta^\star = \arg\min_\theta \max_{\phi \in \mathcal{W}} \mathop{\mathbb{E}}_{\mathbf{x}} \left[d_\phi(\mathbf{x} \mid \mathbf{c})\right] - \mathop{\mathbb{E}}_{\hat{\mathbf{x}}} \left[d_\phi(\hat{\mathbf{x}} \mid \mathbf{c})\right]. \tag{C.3}$$

In this configuration, the discriminator classifier is replace by a critic function $d_\phi(.) : \mathbb{R}^T \times \mathbb{R}^{|\mathbf{c}|} \to \mathbb{R}$, and we have $\mathcal{W}$ is the 1-Lipschitz space. WGAN uses a weight clipping strategy in order to enforces 1-Lipschitzness. But this strategy can lead to poor samples or even unability to converge. A solution to this problem have been proposed by Gulrajani et al. (2017) in the form of WGAN with gradient penalty. The idea is that knowing that a differentiable function is 1-Lipschtiz if and only if it has gradients with norm at most 1 everywhere, one can enforces 1-Lipschitzness by directly constraining the gradient norm of the critic's output with respect to its input. The obective for WGAN-GP is given by

$$\begin{aligned} V(\phi, \theta) &= -\left(\mathop{\mathbb{E}}_{\hat{\mathbf{x}}} \left[d_\phi(\hat{\mathbf{x}} \mid \mathbf{c})\right] - \mathop{\mathbb{E}}_{\mathbf{x}} \left[d_\phi(\mathbf{x} \mid \mathbf{c})\right] + \lambda \mathrm{GP}\right) \\ \mathrm{GP} &= \mathop{\mathbb{E}}_{\tilde{\mathbf{x}}} \left[\left(\left\|\nabla_{\tilde{\mathbf{x}}} d_\phi(\tilde{\mathbf{x}} \mid \mathbf{c})\right\|_2 - 1\right)^2\right] \end{aligned} \tag{C.4}$$

where $\tilde{\mathbf{x}}$ is sampled randomly along a straight line between pairs of points sampled from the data distribution and the generator distribution. Formally: $\tilde{\mathbf{x}} = \mathrm{a}\hat{\mathbf{x}} + (1-a)\mathbf{x}$ with $a \sim \mathbb{U}(0,1)$. This is motivated by the fact that the optimal critic contains straight lines with gradient norm 1 connecting coupled points from the two distributions (Gulrajani et al., 2017). It is shown empirically by Gulrajani et al. (2017) that, as enforcing the unit gradient norm constraint everywhere is intractable, enforcing it only along these straight lines seems sufficient. The value of λ have been chosen to 10.

Appendix D

Normalizing flows transformer additional information

One key aspect regarding normalizing flows is the choice of f_θ. Indeed, without assumptions, the computation of its Jacobian is intractable. That is why it has been chosen to use autoregressive transformers. The expression of the log determinant of the Jacobian in the autoregressive case is given by equation (5.9).

One of the simplest and most used idea of transformers are affine functions

$$f^i\left(x_i; h^i\right) = \alpha_i x_i + \beta_i$$

with$f^i(\cdot; h^i) : \mathbb{R} \to \mathbb{R}$ parameterized by $h^i = \{\alpha_i, \beta_i\}$, α_i controls the scale of the transformation and β_i the location of the transformation. As long as α_i is not equal to zero, then the transformation is invertible. This can be ensured by taking $\alpha_i = exp(\tilde{\alpha}_i)$, with $\tilde{\alpha}_i$ is a free parameter. This choice lead to a log-determinant expressed by

$$\log\left|\det J_{f_\theta}(\mathbf{x})\right| = \sum_{i=1}^{T} \log\left|\alpha_i\right| = \sum_{i=1}^{T} \tilde{\alpha}_i \tag{D.1}$$

because the derivative of the transformer with respect to x_i is equal to α_i.

Overall, affine autoregresive transformers are really simple and computationally efficient. However, they have a big shortcoming, they lack expressiveness. It lead to requiring a lot of stacked flows in order to model complex distribution. Also, theoretically not know if affine autoregresive flows are universal approximator (Papamakarios et al., 2019). That is why integral based transformation with UMNN are chosen for this work.

Appendix E

Value assessment additional information

The optimization problem contain both variables, parameters, and constraints. The parameters are:

Name of the parameters	Description
$e_t^{\min}, e_t^{\max}$	Minimum/maximum day-ahead bid
$y_t^{\min}, y_t^{\max}$	Minimum/maximum retailer net position [MWh].
$y_{\max}^{\text{dis}}, y_{\max}^{\text{cha}}$	BESS maximum (dis)charging power [MW].
$\eta^{\text{dis}}, \eta^{\text{cha}}$	BESS (dis)charging efficiency [-].
$s^{\min}, s^{\max}$	BES minimum/maximum capacity
$s^{\text{ini}}, s^{\text{end}}$	BESS initial/final state of charge [MWh].
π_t	Day-ahead price [€/MWh].
$\bar{q}_t, \bar{\lambda}_t$	Negative/positive imbalance price [€ /MWh].
Δt	Duration of a time period [hour].

Table E.1: Parameters of the value assessment optimization problem

The variables are:

Name	Range of values	Description
e_t	$\left[e_t^{\min}, e_t^{\max}\right]$	Day-ahead bid [MWh].
y_t	$\left[y_t^{\min}, y_t^{\max}\right]$	Retailer net position in scenario ω [MWh] .
$y_{t,\omega}^{\text{pv}}$	$[0, 1]$	PV generation [MW].
$y_{t,\omega}^{\text{w}}$	$[0, 1]$	Wind generation [MW] .
$y_{t,\omega}^{\text{cha}}$	$\left[0, y_{\max}^{\text{cha}}\right]$	BESS Charging power [MW].
$y_{t,\omega}^{\text{dis}}$	$\left[0, y_{\max}^{\text{dis}}\right]$	Discharging power [MW] .
$s_{t,\omega}$	$\left[s^{\min}, s^{\max}\right]$	BESS state of charge [MWh] .
$d_{t,\omega}^{-}, d_t^{+}$	$\mathbb{R}_+$	Short/long deviation [MWh] .
$y_{t,\omega}^{b}$	$\{0, 1\}$	BESS binary variable to prevent concurrent charge/discharge [-] .

Table E.2: Variables of the value assessment optimization problem

And the constraints are:

$$- d^{-}_{t,\omega} \leq - \left(e_t - y_{t,\omega}\right), \forall t \in \mathcal{T} \tag{E.1}$$

$$- d^{+}_{t,\omega} \leq - \left(y_{t,\omega} - e_t\right), \forall t \in \mathcal{T} \tag{E.2}$$

$$\frac{y_{t,\omega}}{\Delta t} = y^{\mathrm{pv}}_{t,\omega} + y^{\mathrm{w}}_{t,\omega} - y^{\mathrm{l}}_{t,\omega} + y^{\mathrm{dis}}_{t,\omega} - y^{\mathrm{cha}}_{t,\omega}, \forall t \in \mathcal{T} \tag{E.3}$$

$$y^{\mathrm{pv}}_{t,\omega} \leq \hat{y}^{\mathrm{pv}}_{t,\omega}, \forall t \in \mathcal{T} \tag{E.4}$$

$$y^{\mathrm{W}}_{t,\omega} \leq \hat{y}^{\mathrm{W}}_{t,\omega}, \forall t \in \mathcal{T} \tag{E.5}$$

$$y^{\mathrm{l}}_{t,\omega} = \hat{y}^{\mathrm{l}}_{t,\omega}, \forall t \in \mathcal{T} \tag{E.6}$$

$$y^{\mathrm{cha}}_{t,\omega} \leq y^{b}_{t,\omega} y^{\mathrm{cha}}_{\max}, \forall t \in \mathcal{T} \tag{E.7}$$

$$y^{\mathrm{dis}}_{t,\omega} \leq \left(1 - y^{b}_{t,\omega}\right) y^{\mathrm{dis}}_{\max}, \forall t \in \mathcal{T} \tag{E.8}$$

$$- s_{t,\omega} \leq -s^{\min}, \forall t \in \mathcal{T} \tag{E.9}$$

$$s_{t,\omega} \leq s^{\max}, \forall t \in \mathcal{T} \tag{E.10}$$

$$\frac{s_{1,\omega} - s^{\mathrm{ini}}}{\Delta t} = \eta^{\mathrm{cha}} y^{\mathrm{cha}}_{1,\omega} - \frac{y^{\mathrm{dis}}_{1,\omega}}{\eta^{\mathrm{dis}}}, \tag{E.11}$$

$$\frac{s_{t,\omega} - s_{t-1,\omega}}{\Delta t} = \eta^{\mathrm{cha}}\, y^{\mathrm{cha}}_{t,\omega} - \frac{y^{\mathrm{dis}}_{t,\omega}}{\eta^{\mathrm{dis}}}, \forall t \in \mathcal{T} \backslash \{1\} \tag{E.12}$$

$$s_{T,\omega} = s^{\mathrm{end}} = s^{\mathrm{ini}}\,. \tag{E.13}$$

The imbalance penalty is provided by constrains (E.1)-(E.2) assisted by the short and long term deviations variables. Equation (E.3) models the energy balance $\forall \omega \in \Omega$. $y^{\mathrm{pv}}_{t,\omega}$ and $y^{\mathrm{w}}_{t,\omega}$ are bounded by constraints (E.4) and (E.5), $\forall \omega \in \Omega$. $\hat{y}^{\mathrm{pv}}_{t,\omega}$ and $\hat{y}^{\mathrm{w}}_{t,\omega}$ are the PV and wind generation scenarios. Constrain (E.6) models the fact that the load is non-flexible, and it is a parameter $\forall \omega \in \Omega$ where $\hat{y}^{\mathrm{l}}_{t,\omega}$ and load scenarios. The BESS constraints are given by (E.7)-(E.10), and its dynamics are given by (E.11)-(E.13), $\forall \omega \in \Omega$. The optimization variables are $e_t, y_t, d^{-}_t, d^{+}_t, y^{\mathrm{pv}}_t,$ and $y^{\mathrm{w}}_t, y^{\mathrm{cha}}_t, y^{\mathrm{dis}}_t, s_t,$ and y^{b}_t

Then, once the bids e_t have been computed by the planner. The second stage variables must be computed. it is the **dispatching**. The second stage variables are computed given the the real observation of the PV, load and wind power generation. The dispatching is a special case of the formulation where e_t is now a parameter, and there is only one scenario where $\hat{y}^{\mathrm{w}}_{t,\omega}$, $\hat{y}^{\mathrm{pv}}_{t,\omega}$ and $\hat{y}^{\mathrm{l}}_{t,\omega}$ becomes there real values. The optimization variables are $y_t, d^{-}_t, d^{+}_t, y^{\mathrm{pv}}_t$, and y^{w}_t, $y^{\mathrm{cha}}_t, y^{\mathrm{dis}}_t, s_t$, and y^{b}_t.

Finally, there are two things to note. First, $\bar{q}_t$ and $\bar{\lambda}_t$ are chosen to be strictly positive. Then, the oracle (O) is a special case of the stochastic formulation. In that case, there is only one scenario where $y^{\mathrm{l}}_{t,\omega}$, $y^{\mathrm{pv}}_{t,\omega}$ and $y^{\mathrm{w}}_{t,\omega}$ becomes the actual values of the load, PV generation and wind power generation. It correspond to a perfect forecast, it is used for comparison. It is correspond to the best possible profit obtainable as it correspond to a perfect forecast that knows the future exactly.

Parameters values The values of the parameters are chosen according to Dumas et al. (2021). The value for the day ahead prices π_t is chosen equal to the day-ahead prices on *February 6*, 2020 of the Belgian day-ahead market. The negative and positive imbalances price are both chosen equal to $2\pi_t$. The BESS minimum capacity s^{min} is equal

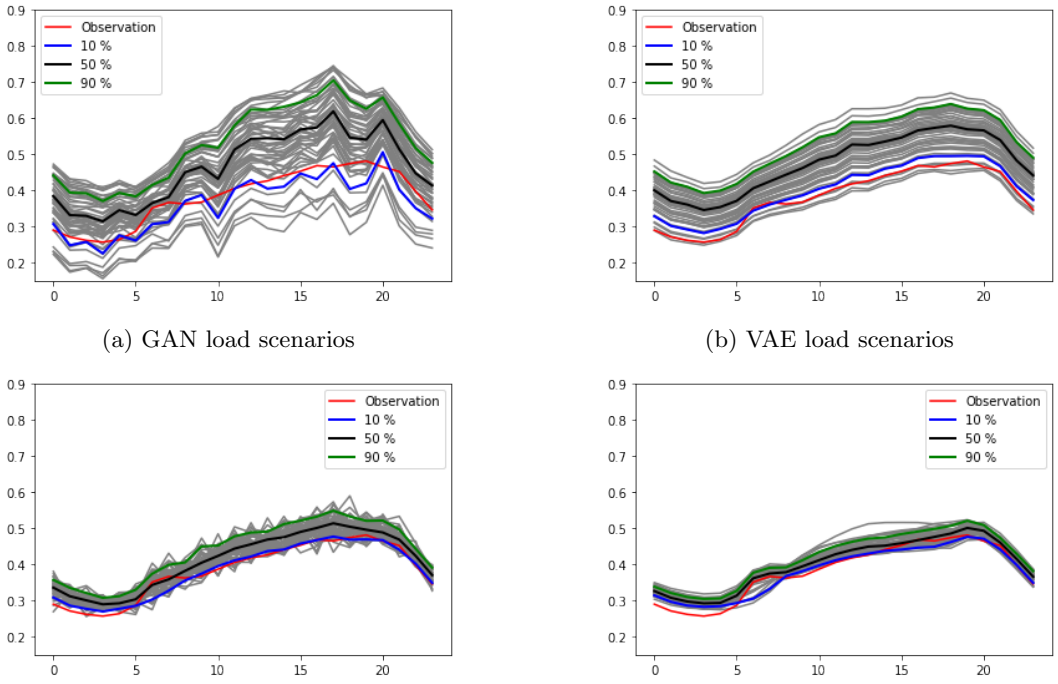
to 0 and the maximum capacity s^{max} is equal to 1. The battery needs 2 hours to fully charge or discharge, leading to $y^{dis}_{max} = y^{cha}_{max} = \frac{s^{max}}{2}$. Finally, the (dis)charging efficiencies are equal to 95%.

The optimization problems and the algorithms have been implemented using the python Gurobi library, using python 3.7 and Gurobi 9.5.1.

Appendix F

Additional scenarios and correlations with the 4 models

Here are another set of scenarios and correlation matrices for another random day of the testing set.



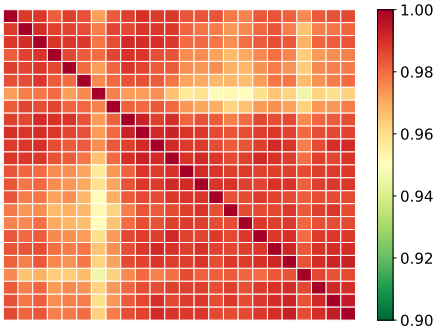
(a) GAN load scenarios

(b) VAE load scenarios

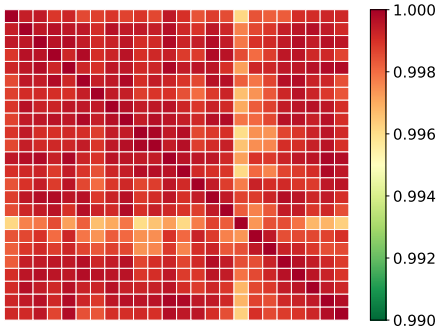
(c) NF load scenarios

(d) DDPM load scenarios

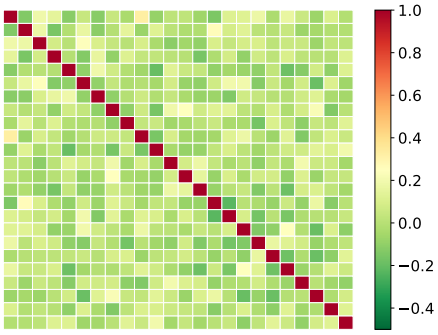
Figure F.1: Example of load scenarios generation for an arbitrary day of the testing set with the 4 generative models. 50 load scenarios with the derived percentile and the corresponding real observation.
The GAN and VAE scenarios almost look like a translation of one another, they are highly correlated. On the other hand, NF scenarios tend to cross each other and seem noisy, and DDPM scenarios are more similar to one another but seem to be less correlated to one another than those of GAN and VAE.



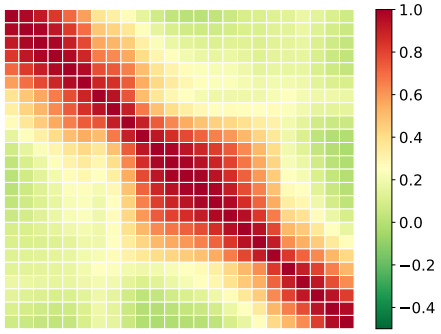
(a) Pearson time correlation matrix of GAN load model.



(b) Pearson time correlation matrix of VAE load model.

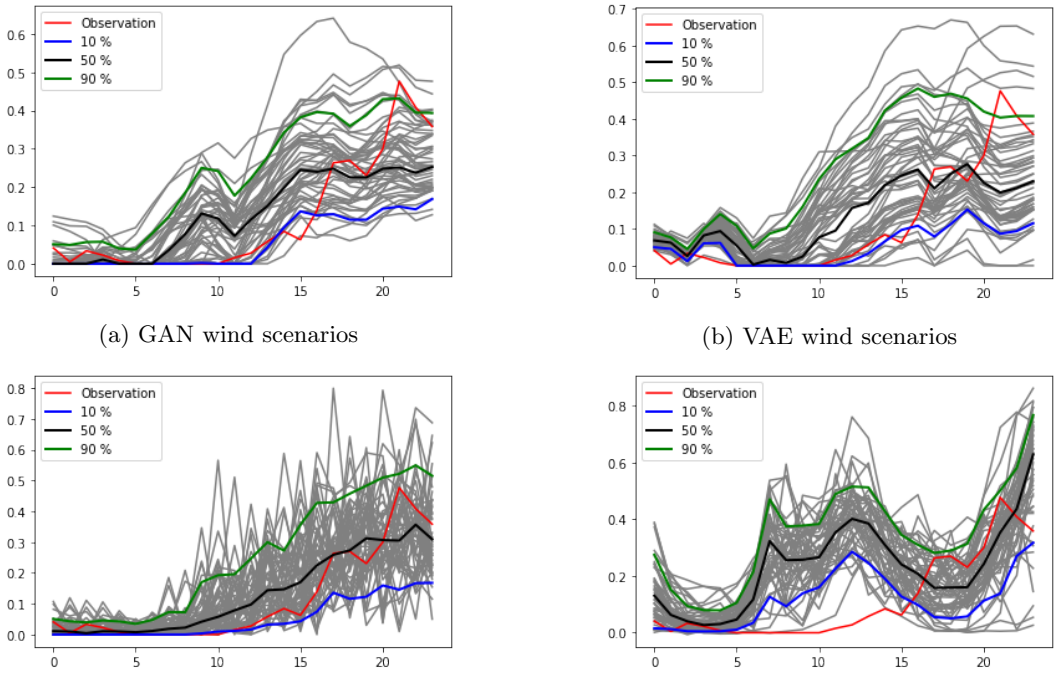


(c) Pearson time correlation matrix of NF load model.



(d) Pearson time correlation matrix of DDPM load model.

Figure F.2: Pearson correlation matrix associated with the load scenarios of the arbitrary day of figure F.1. The matrices corroborate the intuition given by the scenarios shapes on figure F.1. We have a really high correlation for both GANs and VAE for all the hours of the day, with a slight advantage for the VAE. On the other hand, the NF comes with no time correlation at all and the scenarios seem really noisy (but they all gravitate around the observation). Finally, the DDPM displays more correlation than the NF, but less than both GAN and VAE. With DDPM, some hours of the day are correlated.



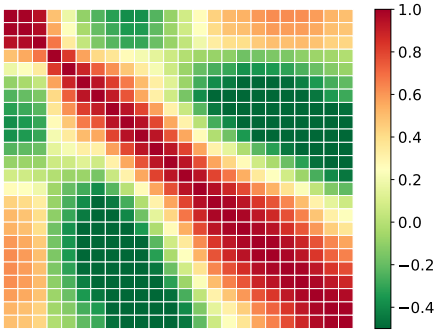
(a) GAN wind scenarios

(b) VAE wind scenarios

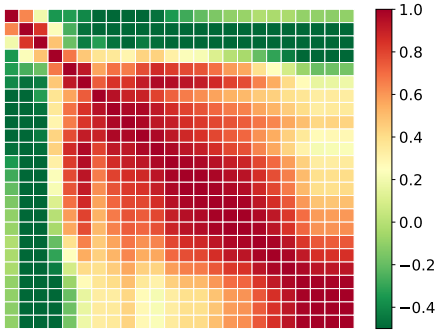
(c) NF wind scenarios

(d) DDPM wind scenarios

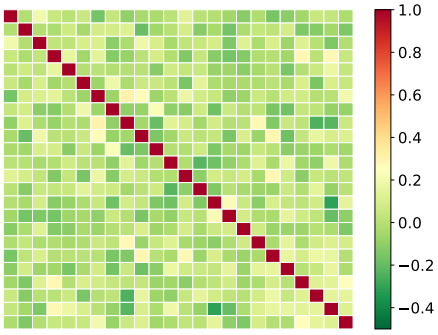
Figure F.3: Example of wind scenarios generation for an arbitrary day of the testing set with the 4 generative models. 50 wind scenarios with the derived percentile and the corresponding real observation.
The GAN and VAE scenarios seem more correlated to each other. On the other hand, NF scenarios have a tendency to cross each other and it seems to have no correlation between the scenarios at all. DDPM scenarios are more similar to one another than NF's, but seem to be less correlated to one another than those of GAN and VAE.



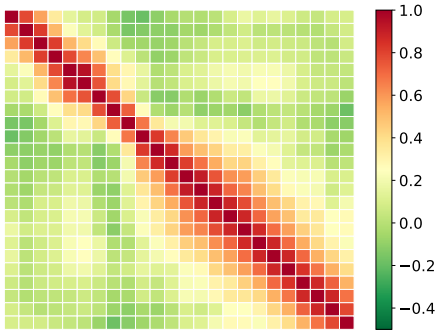
(a) Pearson time correlation matrix of GAN wind model.



(b) Pearson time correlation matrix of VAE wind model.

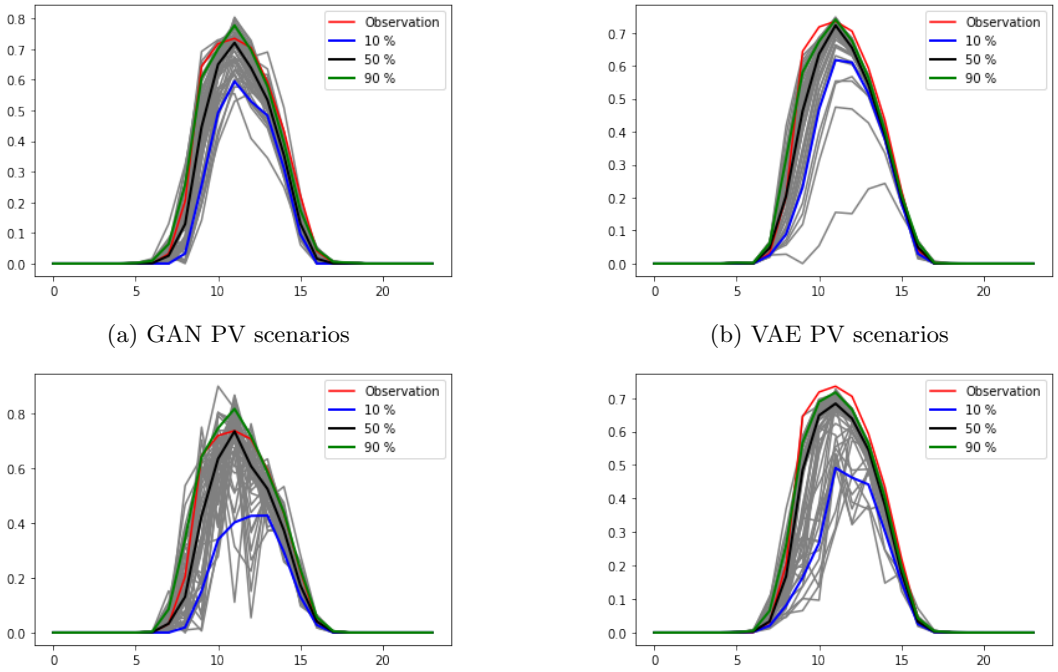


(c) Pearson time correlation matrix of NF wind model.



(d) Pearson time correlation matrix of DDPM wind model.

Figure F.4: Pearson correlation matrix associated with the wind scenarios of the arbitrary day of figure F.3. The matrices corroborate the intuition given by the scenarios shapes on figure F.3. We have a really high correlation for both GANs and VAE for several hours of the day. On the other hand, the NF displays no time correlation at all and the scenarios seem noisier than the other models. Finally, the DDPM displays correlation for a couple of hours of the day but less than both GAN and VAE.



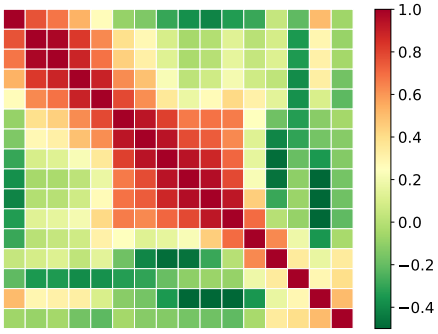
(a) GAN PV scenarios

(b) VAE PV scenarios

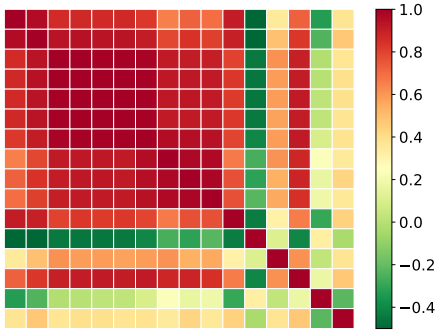
(c) NF PV scenarios

(d) DDPM PV scenarios

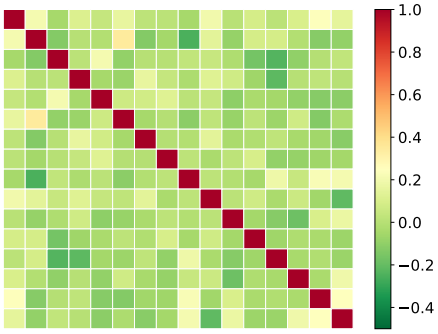
Figure F.5: Example of PV scenarios generation for an arbitrary day of the testing set with the 4 generative models. 50 PV scenarios with the derived percentile and the corresponding real observation.



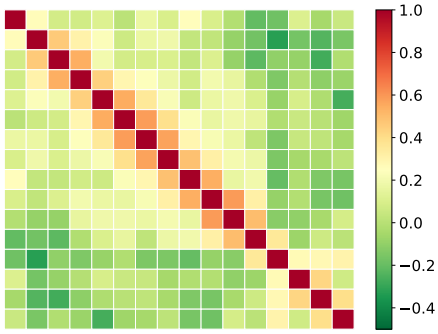
(a) Pearson time correlation matrix of GAN PV model.



(b) Pearson time correlation matrix of VAE PV model.



(c) Pearson time correlation matrix of NF PV model.



(d) Pearson time correlation matrix of DDPM PV model.

Figure F.6: Pearson correlation matrix associated with the PV scenarios of the arbitrary day of figure F.5. The matrices corroborate the intuition given by the scenarios shapes on figure F.5. We have higher correlations for both GANs and VAE for some sparse hours of the day. Once again, there is a slight advantage for the VAE. On the other hand, the NF exhibit no time correlation at all, and the scenarios seem really noisy (but they all gravitate around the observation). Finally, the DDPM displays a little bit more correlation than the NF, but the difference is not significant.

Bibliography

M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks, 2017. URL https://arxiv.org/abs/1701.04862.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017. URL https://arxiv.org/abs/1701.07875.

S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi: 10.1109/tpami.2021.3116668. URL https://doi.org/10.1109%2Ftpami.2021.3116668.

L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux. Api design for machine learning software: experiences from the scikit-learn project. 2013. doi: 10.48550/ARXIV.1309.0238. URL https://arxiv.org/abs/1309.0238.

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2014. URL https://arxiv.org/abs/1412.7062.

Y. Chen, Y. Wang, D. S. Kirschen, and B. Zhang. Model-free renewable scenario generation using generative adversarial networks. *CoRR*, abs/1707.09676, 2017. URL http://arxiv.org/abs/1707.09676.

Y. Chen, P. Li, and B. Zhang. Bayesian renewables scenario generation via deep generative networks. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2018. doi: 10.1109/CISS.2018.8362314.

A. Dairi, F. Harrou, Y. Sun, and S. Khadraoui. Short-term forecasting of photovoltaic solar power production using variational auto-encoder driven deep learning approach. *Applied Sciences*, 10(23), 2020. ISSN 2076-3417. doi: 10.3390/app10238400. URL https://www.mdpi.com/2076-3417/10/23/8400.

J. G. De Gooijer and R. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 22:443–473, 02 2006. doi: 10.1016/j.ijforecast.2006.01.001.

P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

J. Dumas, A. Wehenkel, D. Lanaspeze, B. Cornélusse, and A. Sutera. Deep generative modeling for probabilistic forecasting in power systems. *CoRR*, abs/2106.09370, 2021. URL https://arxiv.org/abs/2106.09370.

J. Dumas, C. Cointe, A. Wehenkel, A. Sutera, X. Fettweis, and B. Cornélusse. A probabilistic forecast-driven strategy for a risk-aware participation in the capacity firming market. *IEEE Transactions on Sustainable Energy*, October 2021. ISSN 1949-3029. doi: 10.1109/TSTE.2021.3117594. URL https://arxiv.org/abs/2105.13801.

W. FELLER. On the theory of stochastic processes, with particular reference to applications, 1945. URL https://digitalassets.lib.berkeley.edu/math/ucb/text/math_s1_article-21.pdf.

L. Ge, W. Liao, S. Wang, B. Bak-Jensen, and J. Pillai. Modeling daily load profiles of distribution network for scenario generation using flow-based generative network. *IEEE Access*, 8:77587–77597, Apr. 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2989350.

T. Gneiting. Editorial: Probabilistic forecasting. *Journal of the Royal Statistical Society Series A*, 171:319–321, 02 2008. doi: 10.1111/j.1467-985X.2007.00522.x.

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

F. Golestaneh, H. Gooi, and P. Pinson. Generation and evaluation of space-time trajectories of photovoltaic power. *Applied Energy*, 176:80–91, 2016. ISSN 0306-2619. doi: 10.1016/j.apenergy.2016.05.025.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans, 2017. URL https://arxiv.org/abs/1704.00028.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

G. E. Hinton. Rectified linear units improve restricted boltzmann machines vinod nair, 2010.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. URL https://arxiv.org/abs/2006.11239.

T. Hong and S. Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938, 2016. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2015.11.011.

T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. Hyndman. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. *International Journal of Forecasting*, 32(3):896–913, 2016. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2016.02.001.

T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour. Energy forecasting: A review and outlook. *IEEE Open Access Journal of Power and Energy*, 7, 10 2020. doi: 10.1109/OAJPE.2020.3029979.

C. Jiang, Y. Mao, Y. Chai, and M. Yu. Day-ahead renewable scenario forecasts based on generative adversarial networks. *International Journal of Energy Research*, 45(5):7572–7587, 2021. doi: https://doi.org/10.1002/er.6340. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/er.6340.

D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving variational inference with inverse autoregressive flow, 2016. URL https://arxiv.org/abs/1606.04934.

I. Kobyzev, S. J. Prince, and M. A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, nov 2021. doi: 10.1109/tpami.2020.2992934. URL https://doi.org/10.1109%2Ftpami.2020.2992934.

Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. Diffwave: A versatile diffusion model for audio synthesis, 2020. URL https://arxiv.org/abs/2009.09761.

M. Landry, T. P. Erlinger, D. Patschke, and C. Varrichio. Probabilistic gradient boosting machines for GEFCom2014 wind forecasting. *International Journal of Forecasting*, 32(3):1061–1066, 2016. doi: 10.1016/j.ijforecast.2016. URL https://ideas.repec.org/a/eee/intfor/v32y2016i3p1061-1066.html.

F. A.-D. M. B. P. B. M. B. S. B. M. B. I. C. A. C. e. a. M. Allen, P. Antwi-Agyei. Technical summary: Global warming of 1.5° c. an ipcc special report on the impacts of global warming of 1.5° c above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty, technical report, intergovernmental panel on climate change. 2019.

J. Morales González, A. Conejo, H. Madsen, P. Pinson, and M. Zugno. *Integrating Renewables in Electricity Markets: Operational Problems*. International Series in Operations Research and Management Science. Springer, 2014. ISBN 978-1-4614-9410-2. doi: 10.1007/978-1-4614-9411-9.

A. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models, 2021. URL https://arxiv.org/abs/2102.09672.

A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio, 2016a. URL https://arxiv.org/abs/1609.03499.

A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016b. URL https://arxiv.org/abs/1606.05328.

G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation, 2017. URL https://arxiv.org/abs/1705.07057.

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. 2019. doi: 10.48550/ARXIV.1912.02762. URL https://arxiv.org/abs/1912.02762.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

F. Perez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE International Symposium on Information Theory*, pages 1666–1670, 2008. doi: 10.1109/ISIT.2008.4595271.

V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech, 2021. URL https://arxiv.org/abs/2105.06337.

Y. Qi, W. Hu, Y. Dong, Y. Fan, L. Dong, and M. Xiao. Optimal configuration of concentrating solar power in multienergy power systems with an improved variational autoencoder. *Applied Energy*, 274:115124, 09 2020. doi: 10.1016/j.apenergy.2020.115124.

S. Raja Sekaran, Y. Pang, G. Ling, and O. Yin. Mstcn: A multiscale temporal convolutional network for user independent human activity recognition [version 2; peer review: 1 approved, 1 approved with reservations]. *F1000Research*, 10(1261), 2022. doi: 10.12688/f1000research.73175.2.

K. Rasul, C. Seward, I. Schuster, and R. Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. 2021. doi: 10.48550/ARXIV.2101.12072. URL https://arxiv.org/abs/2101.12072.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows, 2016.

L. Ruthotto and E. Haber. An introduction to deep generative modeling, 2021. URL https://arxiv.org/abs/2103.05180.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016. URL https://arxiv.org/abs/1606.03498.

M. Scheuerer and T. M. Hamill. Variogram-based proper scoring rules for probabilistic forecasts of multivariate quantities. *Monthly Weather Review*, 143(4):1321 – 1334, 2015. doi: 10.1175/MWR-D-14-00269.1. URL https://journals.ametsoc.org/view/journals/mwre/143/4/mwr-d-14-00269.1.xml.

J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf.

L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models, 2015. URL https://arxiv.org/abs/1511.01844.

J. Thornes and D. Stephenson. How to judge the quality and value of weather forecast products. *Meteorological Applications*, 8:307 – 314, 09 2001. doi: 10.1017/S1350482701003061.

J. Toubeau, J. Bottieau, F. Vallée, and Z. D. Grève. Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. *IEEE Transactions on Power Systems*, 34:1203–1215, 2019.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008. ISBN 9783540710509. URL https://books.google.be/books?id=hV8o5R7_5tkC.

L. Wang, W. Chen, W. Yang, F. Bi, and F. R. Yu. A state-of-the-art review on image synthesis with generative adversarial networks. *IEEE Access*, 8:63514–63537, 2020a. doi: 10.1109/ACCESS.2020.2982224.

Y. Wang, G. Hug, Z. Liu, and N. Zhang. Modeling load forecast uncertainty using generative adversarial networks. *Electric Power Systems Research*, 189:106732, 2020b.

A. Wehenkel and G. Louppe. Unconstrained monotonic neural networks. 2019. doi: 10.48550/ARXIV.1908.05164. URL https://arxiv.org/abs/1908.05164.

F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions, 2015a. URL https://arxiv.org/abs/1511.07122.

F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions, 2015b. URL https://arxiv.org/abs/1511.07122.

R. Yuan, B. Wang, Z. Mao, and J. Watada. Multi-objective wind power scenario forecasting based on pg-gan. *Energy*, 226:120379, 03 2021. doi: 10.1016/j.energy.2021.120379.

R. Yuan, B. Wang, Y. Sun, X. Song, and J. Watada. Conditional style-based generative adversarial networks for renewable scenario generation. *IEEE Transactions on Power Systems*, pages 1–1, 2022. doi: 10.1109/TPWRS.2022.3170992.

H. Zhanga, W. Hua, R. Yub, M. Tangb, and L. Dingc. Optimized operation of cascade reservoirs considering complementary characteristics between wind and photovoltaic based on variational auto-encoder. *MATEC Web of Conferences*, 246:01077, 01 2018. doi: 10.1051/matecconf/201824601077.