FACULTY OF ENGINEERING
AND ARCHITECTURE

Generative modeling of electricity imbalance prices for battery optimization

Master Thesis

Name: Victor Mylle

Promotors: prof. dr. ir. Chris Develder
prof. Bert Claessens

Supervisor: Jonas Van Gompel

Academic year: 2023–2024

GHENT
UNIVERSITY

Contents

1 Introduction

The electricity market is a complex system influenced by numerous factors. The rise of renewable energy sources adds to this complexity, introducing greater volatility compared to traditional energy sources. Renewables, with their unpredictable nature, exacerbate the challenge of maintaining a stable balance between supply and demand. This critical balance is managed by the transmission system operator, Elia, which utilizes reserves to mitigate any potential shortages or surpluses, directly influencing electricity prices.

(TODO: Market participants met flexible assets (Groot genoeg), zij willen grote winst maken. Elia moet minder eigen reserves gebruiken -> goedkoper voor iedereen)

Forecasting the imbalance price is vital for market participants engaged in buying or selling electricity. It enables them to make informed decisions on the optimal times to buy or sell, aiming to maximize their profits. However, current industry practices often rely on simplistic policies, such as adhering to a fixed price for transactions. This approach is not optimal and overlooks the potential benefits of adaptive policies that consider the forecasted imbalance prices.

The goal of this thesis is to generatively model the Belgian electricity market. This allows the reconstruction of the imbalance price for a given day which can then be used by other simple policies to make decisions on when to buy or sell electricity. These policies can then be compared to the current industry practices to assess their performance.

Forecasting the system imbalance will become increasingly important as the share of renewable energy sources continues to grow.

This thesis can be divided into two main parts. The first part focuses on modeling the Net Regulation Volume (NRV) of the Belgian electricity market for the next day. This modeling is conditioned on multiple inputs that can be obtained from Elia (TODO: add citation to the open data of Elia). The second part of the thesis focuses on optimizing a simple policy using the NRV generations for the next day. The policy tries to maximize profit by charging and discharging a battery and thereby buying and selling electricity on the market. Multiple models are trained and tested to model the NRV and compared to each other based on their profit optimization.

2 Electricity market

The electricity market consists of many different parties who all work together and want to make a profit in the end. An overview of the most important parties can be found in Table 1.

Party	Description
Producers	Generates electricty. The electricity can be generated using coal, nuclear energy, wind parks etc.
Consumers	Uses electricity. This can be normal households, companies but also industry.
Transmission system operator (TSO)	Party responsible for reliable transmission of electricity from generation plants to local distribution networks. This is done over the high-voltage grid. In Belgium, this party is Elia.
Distribution system operator (DSO)	Party responsible for the distribution of electricity to the end users. Here, the electricity is transported over the low-voltage grid.
Balancing responsible party (BRP)	These parties forecast the electricity consumption and generation of their clients. They make balanced nominations to Elia.
Balancing Service Provider (BSP)	Parties that provide the TSO (Elia) with balancing services. They submit Balancing Energy Bids to Elia. If needed, they will provide balancing energy at a set price.

Table 1: Overview of the most important parties in the electricity market

Elia, the Transmission system operator (TSO) in Belgium is responsible for keeping the grid stable. They do this by balancing the electricity consumption and generation. If there is an imbalance, Elia will use reserves to balance the grid. These reserves are expensive and are paid by the market participants. The prices paid for the activations of these reserves is called the imbalance price. Keeping the grid balanced is a very important but also a very difficult task. If the grid is not balanced, it can lead to blackouts but also other problems like damage to equipment and so on.

Balance responsible parties (BRPs) forecast the electricity consumption and generation of their portfolio to effectively manage the balance between supply and demand within the grid they operate in. They submit a daily balance schedule for their portfolio the day before to the trans-

mission system operator. This consists of the expected physical injections and offtakes from the grid and the commercial power trades. The power trades can be purchases and sales between BRPs or they can even be trades with other countries. BRPs must provide and deploy all reasonable resources to be balanced on a quarter-hourly basis. They can exchange electricity with other BRPs for the following day or the same day. There is one exception where a BRP can deviate from the balance schedule. This is when the grid is not balanced and they can help Elia to stabilize the grid. In this case, they will receive a compensation for their help. When a BRP deviates from the balance schedule in a way that destabilizes the grid, it will need to pay the imbalance price for the deviation.

The imbalance price is determined based on which reserves Elia needs to activate to stabilize the grid. The imbalance of a BRP is the quarter-hourly difference between total injections and offtakes from the grid. The Net Regulation Volume (NRV) is the net control volume of energy that Elia applies to maintain balance in the Elia control area. The Area Control Error is the current difference between the scheduled values and the actual values of power exchanged in the Belgian control area. The imbalance of the system (SI) is the Area Control Error minus the NRV. Using the System Imbalance, the imbalance price is calculated.

Elia, the Transmission System Operator (TSO) in Belgium, maintains grid stability by activating three types of reserves, each designed to address specific conditions of imbalance. These reserves are crucial for ensuring that the electricity supply continuously meets the demand, thereby maintaining the frequency within the required operational limits. The reserves include:

1) **Frequency Containment Reserve (FCR)**
FCR is a reserve that responds automatically to frequency deviations in the grid. The reserve responds automatically in seconds and provides a proportional response to the frequency deviation. Elia must provide a minimal share of this volume within the Belgian control area. This type of volume can also be offered by the BSPs.

2) **Automatic Frequency Restoration Process (aFRR)**
aFRR is the second reserve that Elia can activate to restore the frequency to 50Hz. The aFRR is activated when the FCR is not sufficient to restore the frequency. Every 4 seconds, Elia sends a set-point to the BSPs. The BSPs use this set-point to adjust their production or consumption. The BSPs have a 7.5-minute window to activate the full requested energy volume.

3) **Manual Frequency Restoration (mFRR)**
Sometimes the FCR and aFRR are not enough to restore the imbalance between generation and consumption. Elia activates the mFRR manually and the requested energy volume is to be activated in 15 minutes.

The order in which the reserves are activated is as follows: FCR, aFRR and mFRR. BSPs provide bids for the aFRR and mFRR volumes. The provided bids consist of the type (aFRR or mFRR), bid volume (MW), bid price (per MWh) and start price (per MWh). The start price is used to cover the costs of starting a unit.

Elia selects the bids based on the order of activation and then the price. The highest marginal price paid for upward or downward activation determines the imbalance price. This means that the last bid that is activated determines the imbalance price. This price is paid by the BRPs that are not balanced. The imbalance price calculation is shown in Table 2.

	System Imbalance	
Imbalance of the balance responsible party	**Positive**	**Negative or zero**
Positive	MDP - α	MIP + α
Negative	MDP - α	MIP + α

Table 2: Prices paid by the BRPs

The imbalance price calculation includes the following variables:
- MDP: Marginal price of downward activation
- MIP: Marginal price of upward activation
- α: Extra parameter dependent on System Imbalance

TODO: Add more information about the imbalance price calculation, alpha?

The imbalance price can be reconstructed given the bids of a certain quarter/day and the System Imbalance. During this thesis, the system imbalance is assumed to be almost the same as the Net Regulation Volume. This is a simplification but it is a good approximation. The goal of this thesis is to model the Net Regulation Volume which can then be used to reconstruct the imbalance price and to make decisions on when to buy or sell electricity.

3 Generative modeling

Simple forecasting of the NRV is often not accurate and defining a policy using this forecast will lead to wrong decisions. A better method would be to try to model the NRV and sample multiple generations of the NRV for a whole day. This can give a better understanding of the uncertainty of the NRV. Better decisions can then be made based on multiple generations of the NRV.

Generative modeling is a type of machine learning that is used to generate new data samples that look like the training data. The goal of generative modeling is to learn the true data distribution and use this distribution to generate new samples. Generative modeling is used in many different fields including image generation, text generation etc.

In this thesis, generative modeling can be used to model the NRV of the Belgian electricity market using different conditional input features like the weather, the load forecast etc. The model can then be used to generate new samples of the NRV.

There exist many different types of generative models. Some of the most popular ones are:

- Generative Adversarial Networks (GANs)
- Variational Autoencoders (VAEs)
- Normalizing Flows
- Diffusion models

3.1 Quantile Regression

Another method can be used to use any feedforward neural network as a generative model. This method is called quantile regression. This method enables the model to output values to reconstruct the distribution of the target variable instead of a single value for a quarter. This distribution can then be used to sample the NRV value for a quarter. The sampling allows for multiple full-day generations of the NRV.

When quantile regression is used, the model outputs the values for multiple quantiles for the target value of a certain quarter. A quantile is a statistical value of a random variable below which a certain proportion of observations fall. Figure 1 shows the cumulative distribution function of a normal distribution. The figure shows the 25th, 50th and 75th quantiles. The 25th quantile is the value below which 25% of the observations fall. In the example, this value is -0.67. The other quantiles work in the same way.

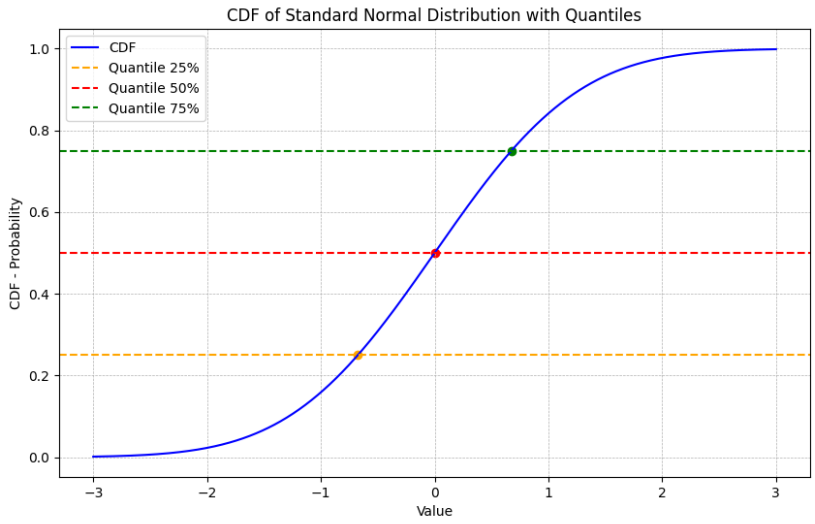


Figure 1: Example of quantiles

Using the outputted quantiles, the cumulative distribution function can be reconstructed and used to sample the NRV value for the quarter to predict. An example of the output of a quantile regression model is shown in figure 2. The output values of the different quantiles are plotted and these are interpolated to get the cumulative distribution function. In this thesis, the quantiles used are 1%, 5%, 10%, 15%, 30%, 40%, 50%, 60%, 70%, 85%, 90%, 95%, and 99%. These are

chosen to get a good approximation of the cumulative distribution function. More quantiles at the tails of the distribution are used because the edges of the distribution are more important for the imbalance price calculation. The outputted quantile values are then interpolated using cubic spline and samples can be drawn from the reconstructed cumulative distribution function.

TODO: figure goes under 0, maybe use other values or other interpolation? + inverse the values to real values

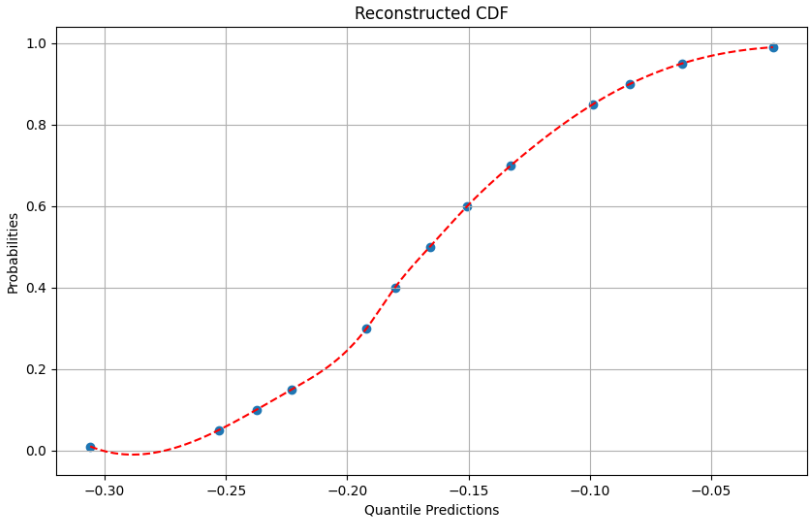


Figure 2: Example of quantile regression output for one-quarter of the NRV, showing interpolated values for quantiles at 1%, 5%, 10%, 15%, 30%, 40%, 50%, 60%, 70%, 85%, 90%, 95%, and 99%. These quantiles are used to reconstruct the cumulative distribution function.

The NRV value for a quarter can be sampled from the reconstructed cumulative distribution function. A full-day prediction for the NRV exists of 96 values. This means 96 cumulative distributions need to be reconstructed and samples need to be drawn from each of the distributions.

The quantile regression model is trained using the pinball loss function, also known as the quantile loss. The model outputs the quantile values for the NRV. The quantile values themselves are not available in the training data. Only the real NRV values are known. The loss function is defined as:

$$L_\tau(y, \hat{y}) = \begin{cases} \tau(y - \hat{y}) & \text{if } y \geq \hat{y} \\ (1 - \tau)(\hat{y} - y) & \text{if } y < \hat{y} \end{cases} \tag{1}$$

Where:

τ = Quantile of interest

y = Actual observed value of NRV

$\hat{y}$ = Predicted quantile value of NRV

The loss function works by penalizing underestimation and overestimation differently. When a quantile is predicted that is lower than or equal to the actual value, the loss is calculated as the difference between the actual value and the predicted quantile value multiplied by the quantile of interest. This means that underestimations for high quantiles are penalized higher than for lower quantiles.

When the quantile value prediction is higher than the real NRV value, the loss is calculated as the difference between the predicted quantile value and the real NRV multiplied by $(1 - \tau)$. This means that overestimations are penalized less for high quantiles of interest.

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{\tau \in T} L_\tau(y_i, \hat{y}_i) \tag{2}$$

Where:

N = Number of samples

T = Quantiles of interest

y_i = Actual observed value of NRV for sample i

$\hat{y}_i$ = Predicted quantile value of NRV for sample i

To calculate the pinball loss, the mean over the quantiles of interest and samples need to be taken. This gives a scalar loss value which can be used to do backpropagation. The lower this value, the better the NRV distribution is modeled.

3.2 Autoregressive vs Non-Autoregressive models

Two types of generative models exist, autoregressive and non-autoregressive models. Autoregressive models generate samples by sampling from the model one step at a time. The model generates the next value based on the previous values. This means that the model generates samples sequentially. Non-autoregressive models on the other hand generate samples in one step. The model generates the whole sample existing of multiple values at once. This means that the model can generate samples in parallel which can be done way faster than autoregressive models. The downside of non-autoregressive models is that the model itself is more complex and harder to train. It needs to predict all values at once which can be harder than predicting one value at a time.

The quantile regression method can be used with both types of models. The autoregressive model will only output the quantiles for the next quarter based on the given input features. The cumulative distribution function can be reconstructed from these and be used to sample the NRV value. To obtain a full-day sample, the model needs to be run 96 times sequentially. The sample for the next quarter depends on the sample of the previous quarter.

The non-autoregressive model will output the quantiles for all quarters of the day based on the input features. The cumulative distribution functions all need to be reconstructed and samples can be drawn from each of the distributions. When sampling from the distributions at once, the samples are independent of each other. The sample for the next quarter does not depend on the sample of the previous quarter which can result in some unrealistic samples.

The input features for autoregressive and non-autoregressive also differ. When forecasted features are used, the autoregressive model only uses the forecasted values for the next quarter while the non-autoregressive model uses the forecasted values for all quarters of the day. In theory, the autoregressive model should also be able to use forecasted values for quarters further in the future but this makes it harder to use in practice. When the last quarter of a day needs to be predicted, the forecasted values for the next day are needed which are not available. For simplicity, during this thesis, the autoregressive model will only be provided with the forecasted values for the next quarter.

3.3 Model Types

3.3.1 Linear Model

A simple linear model can be used as a baseline to compare the more complex models. This model assumes a linear relation exists between the input features and the output. The relationship is modeled using the following formula:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{3}$$

Where:

y = Output value

β_0 = Intercept

$\beta_1, ..., \beta_n$ = Coefficients

$x_1, ..., x_n$ = Input features

This model needs to be adapted to be used for quantile regression. The model needs to output the quantiles for the target value. This can be done by training multiple linear models for each of the quantiles. The model can be trained using the pinball loss function. The number of

parameters in this model is quite low which makes it easier and faster to train. The downside of this model is that it is very simple and might not be able to capture the complexity of the data. The number of parameters of this model is number of quantiles × (number of input features + 1).

$$\hat{y}_\tau = \beta_{0,\tau} + \beta_{1,\tau} x_1 + \beta_{2,\tau} x_2 + ... + \beta_{n,\tau} x_n \tag{4}$$

Where:

τ = Quantile of interest

$\hat{y}_\tau$ = Predicted quantile value for the target value

$\beta_{0,\tau}$ = Intercept for the quantile of interest

$\beta_{1,\tau}, ..., \beta_{n,\tau}$ = Coefficients for the quantile of interest

$x_1, ..., x_n$ = Input features

3.3.2 Non-Linear Model

A more complex model can be used to model the NRV. A feedforward neural network with multiple hidden layers and activation functions can be used. This model can then capture the non-linear relationships between the input features and the output. This model has more parameters and is harder to train than the linear model. The non-linear model also has some hyperparameters that need to be chosen like the number of hidden layers, the number of neurons in each layer, the activation function etc. The model can be trained to output the quantiles for the NRV based on the input features. The same pinball loss function can be used to train the model.

3.3.3 Recurrent Neural Network (RNN)

Another more complex model that can be used is a Recurrent Neural Network (RNN). The RNN can be used to model the NRV data because of the sequential nature of the input features. The RNN keeps a hidden state that is updated at every time step using the new input data. The hidden state contains information about the previous time steps and can be used to make predictions for the next time step. These models are used in multiple fields like natural language processing, time series forecasting etc.

The RNN model can be used to model the NRV data. The input features are structured in a way that the model can learn the sequential patterns in the data. The model can be trained to output the quantiles for the NRV based on the input features using the pinball loss function.

Multiple types of RNN models exist. The two most common types of RNNs are the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU). The GRU is a simpler version of

the LSTM. The GRU has fewer parameters which results in faster training times. The GRU still can capture long-term dependencies in the data and can achieve similar performance to the LSTM. The GRU model has two gates, the reset gate and the update gate. The reset gate determines how much of the past information to forget, and the update gate determines how much of the new information to keep.

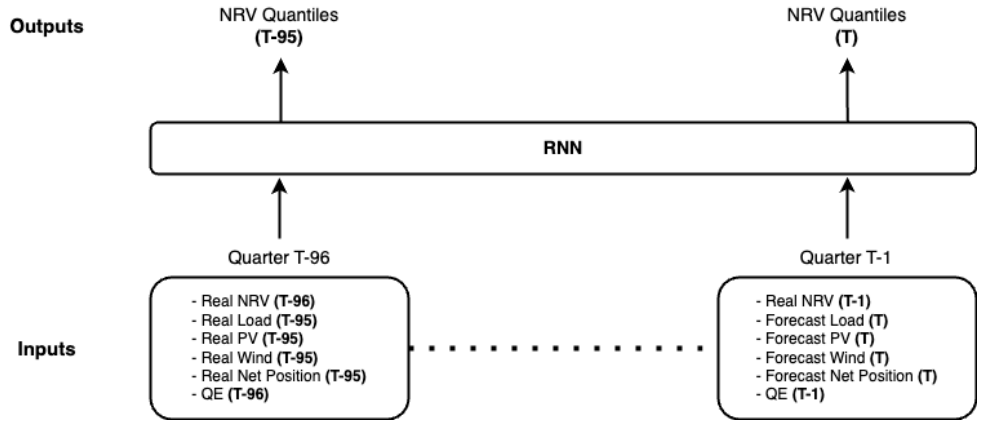


Figure 3: RNN model input and output visualization

The input features for the RNN model are carefully structured to capture the relevant information from the previous quarters and the forecasted values. Each input feature vector represents a quarter and consists of the following components:

- The actual NRV value from the current quarter (T-1), which provides the model with the historical context of the NRV.
- The forecasted or real values for the next quarter (T), including load, PV, wind, and net position. If the next quarter is not the quarter to predict, the real values for that quarter are used. If the next quarter is the quarter to predict, the forecasted values are used.
- A quarter embedding vector representing the current quarter (T-1). The embedding vector gives the model information about the time of day, which can help it learn the daily patterns in the NRV data.

The input feature structure is designed to provide the model with a comprehensive view of the previous quarters and the forecasted values for the current quarter. By incorporating both historical and forecasted information sequentially, the model can learn to predict the NRV quantiles for the next quarter more accurately.

3.4 Diffusion models

TODO: reference the paper The "Denoising Diffusion Probabilistic Models" (DDPM)

3.4.1 Overview

Diffusion models are a type of probabilistic model designed to generate high-quality, diverse samples from complex data distributions. The way this type of model is trained is unique. The model is trained to reverse an iterative noise process that is applied to the data. This process is called the diffusion process. The model denoises the data in each iteration. During the training, the model learns to reverse the diffusion process. A training sample is transformed into a noise sample by applying the diffusion process. The model is then trained to recover the original sample from the noise sample. The model is trained to maximize the likelihood of the data given the noise. By doing this, the model learns to generate samples from the data distribution. Starting from the noise, the model can generate samples that look like the data. The model can also be conditioned on additional information to generate samples that follow other distributions.

3.4.2 Applications

Diffusion models gained popularity in the field of computer vision. They are used for inpainting, super-resolution, image generation, image editing etc. The paper introducing "Denoising Diffusion Probabilistic Models" (DDPM) showed that diffusion models can achieve state-of-the-art results in image generation. This type of model was then applied to other fields like text generation, audio generation etc. The most popular application of diffusion models is still image generation. Many different models and products exist that make use of diffusion models to generate images. Some examples are DALL·E, Stable Diffusion, Midjourney, etc. These models can generate or edit images based on a given text description.

This method can also be applied to other fields like audio generation, text generation etc. In this thesis, diffusion models are explored to model time series data conditioned on additional information. A small example of the diffusion process is shown in Figure 4. An image of a cat is generated by starting from noise and iteratively denoising the image.

Figure 4: Example of the diffusion process. The image of a cat is generated by starting from noise and iteratively denoising the image.

3.4.3 Generation process

The generation process is quite different in comparison to other models. For example, GANs and VAE generate samples by sampling from a noise distribution and then transforming the noise into a sample that looks like the training data in one step using a generator network.

Diffusion models generate samples by starting from a noise distribution and then applying a series of denoising steps to the noise. The diffusion process consists of 3 main components: the forward process, the reverse process and the sampling process.

- **Forward process**
  This forward process is a Markov chain that starts from the data and applies a series of diffusion steps to the data. During this process, Gaussian noise is added to the data in each of the T time steps according to a variance schedule $\beta_1, ..., \beta_T$.

  $$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \text{with} \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

  This formula shows that the noisy data distribution after T diffusion steps is the product of the transition probabilities at each step t. The noise added in each time step is a Gaussian distribution with mean $\sqrt{1-\beta_t}\mathbf{x}_{t-1}$ and variance $\beta_t\mathbf{I}$. The variance schedule $\beta_1, ..., \beta_T$ is a hyperparameter that needs to be chosen or optimized during training.

- **Reverse process**
  The diffusion process must then be reversed. The model is trained to model the noise distribution given the data and timestep.

  $$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad \text{with} \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

  In the reverse process, each step aims to undo the diffusion by estimating what the previous, less noisy state might have been. This is done using a series of conditional Gaussian distributions $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. For each of these Gaussians, a neural network with parameters θ is used to estimate the mean $\mu_\theta(\mathbf{x}_t, t)$ and the covariance $\Sigma_\theta(\mathbf{x}_t, t)$ of the distribution. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is then the product the marginal distribution of the last timestep $p(\mathbf{x}_T)$ and the conditional distributions $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ for each timestep.

- **Training**
  TODO: explain better!
  The model training is done by optimizing the variational bound of the negative log-likelihood. This is also called the evidence lower bound (ELBO) in the context of generative models.

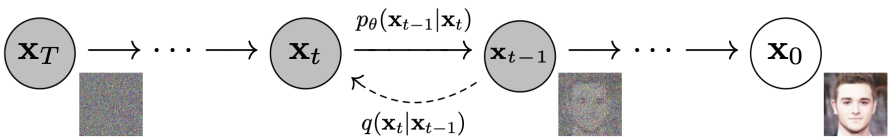
  $$\begin{aligned}
  \log p(x) \geq & \mathbb{E}_q \left[\log p_\theta(x_0|x_1)|x_1, x_0\right] \\
  & - D_{KL}\left(q(x_T|x_0)||p(x_T)\right) \\
  & - \sum_{t=2}^{T} \mathbb{E}_q \left[D_{KL}\left(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)\right)|x_t, x_0\right] \\
  = & L_0 - L_T - \sum_{t=2}^{T} L_{t-1}
  \end{aligned}$$

  The formula shows that maximizing the likelihood can be done by minimizing the KL divergence between the noise distribution and the data distribution for each timestep. After a lot of math, it can be proven that this can be simplified further to minimize the

mean squared error between the predicted noise by the model and the actual noise added in each timestep.

- **Conditioning**
  The model can be conditioned on additional information. This can be used to guide the generation process. In the context of image generation, this can be used to generate images of a certain class or with certain attributes. This requires some changes in the model architecture and training process. A simple way to condition the model is to add additional information to the input of the model. This can be done by concatenating the additional information to the input of the model. The model can then learn to generate samples that follow the distribution of the data conditioned on the additional information.

The diffusion process can be seen in Figure ??. The model is trained to reverse this process. Starting from the noise, the model learns to generate samples that look like the data.



TODO: fix citation

3.5 Evaluation

To evaluate the performance of the quantile regression models, multiple metrics can be used. The pinball loss itself can be used to compare models on the test set. Other metrics that can be used are the mean absolute error (MAE) and the mean squared error (MSE). This can be done by generating multiple full-day NRV samples for each day of the test set and calculating the error metrics for each of the samples. The mean can then be taken over the different samples to get a single value for the error metrics.

MAE does not consider the direction of the error. It is the average of the absolute differences between the predicted and actual values. The formula in our case with full-day NRV samples is:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{96}\sum_{j=1}^{96}|y_{ij} - \hat{y}_{ij}| \tag{5}$$

Where:

N = Number of samples

y_{ij} = Actual observed value of NRV for sample i and quarter j

$\hat{y}_{ij}$ = Sampled value of NRV for sample i and quarter j

MSE is more sensitive to outliers than MAE because it squares the error between the predicted and actual values. The formula in our case with full-day NRV samples is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{96}\sum_{j=1}^{96}(y_{ij} - \hat{y}_{ij})^2 \tag{6}$$

The MAE and MSE metrics do not compare the distribution of the NRV to the real NRV value but only take into account the sampled values. Evaluating the outputted distribution for the NRV must be done differently. The Continuous Ranked Probability Score (CRPS) can be used to evaluate the distribution to the real NRV value. The CRPS metric is used to evaluate the accuracy of the predicted cumulative distribution function. The CRPS can be seen as a generalization of the MAE for probabilistic forecasts. The formula for the CRPS is:

$$CRPS(F, x) = \int_{-\infty}^{\infty}(F(y) - \mathbb{1}(y - x))^2\, dy \tag{7}$$

Where:

F = Predicted cumulative distribution function

x = Real NRV value

$$\mathbb{1}(x) = \text{Heavyside function} = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

The mean CRPS can be calculated over the different days to get a single value. The lower this value, the better the NRV is modeled. The CRPS metric can be visualized as shown in figure 5. The CRPS is the area between the predicted cumulative distribution function and the Heavyside function. The lower the area between the curves, the better the NRV is modeled.

TODO: improve visualisation? -> echte NRV + y as cummulative prob

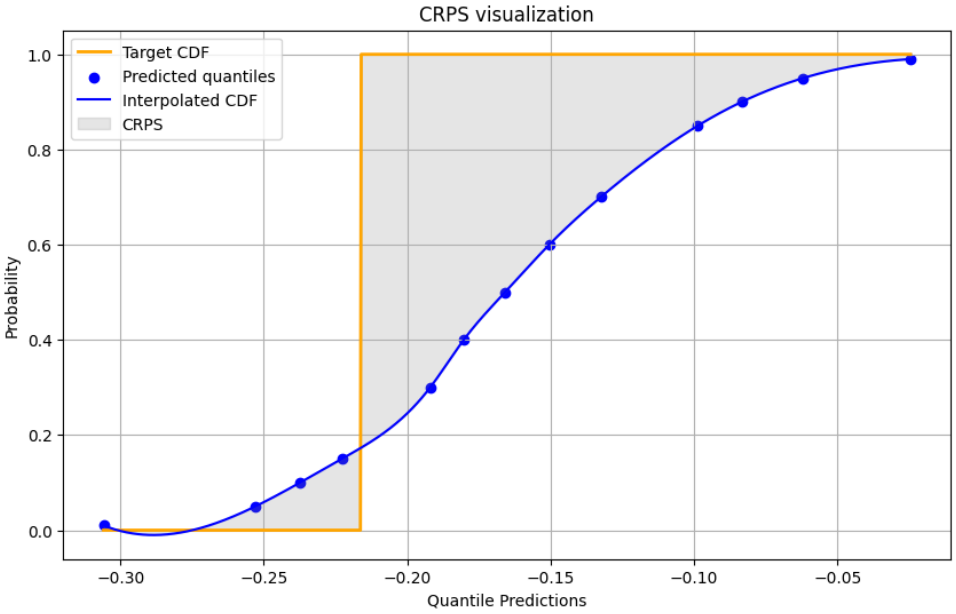


Figure 5: Visualization of the CRPS metric

4 Policies

Organizations that own a battery and are active in the electricity market have to make decisions on when to charge and discharge their battery. These decisions are based on the current state of the battery, the current state of the market, and the future state of the market. The future state of the market can be predicted using generative models like the ones discussed in previous sections. The organizations want to maximize their profit by buying electricity when it is cheap and selling electricity when it is expensive. The policies used decide when to charge and discharge the battery. Another important aspect of these policies is to keep the battery in a healthy state. Charging and discharging a battery too much can reduce its lifetime. The policies have to take this into account.

In this thesis, a simple policy is used to optimize the profit made by charging and discharging a battery. The policy is based on the Net Regulation Volume (NRV) predictions for the next day. This shows the potential of using NRV predictions to optimize the policy. In the real world, more complex policies can be used to optimize the profit. These policies can be trained using reinforcement learning or other optimization techniques. Multiple baseline policies are defined to compare the performance of the policy based on NRV predictions.

The simple policy uses two thresholds to decide when to charge and discharge the battery based on the imbalance price. When the imbalance price is below the charging threshold, the battery is fully charged. When the imbalance price is above the discharging threshold, the battery is fully discharged again. This policy is very simple and does not take into account some important aspects.

4.1 Baselines

The most simple baseline policy is to define two fixed thresholds for charging and discharging the battery. These thresholds can be determined by the historical data of the imbalance price. The thresholds can be found by doing a simple grid search for the best thresholds. The thresholds that maximize the profit on the historical data are used as the fixed thresholds. During the optimization, a penalty parameter can be added to the profit function to penalize when the battery is charged or discharged too much.

Another baseline policy is to determine the thresholds for charging and discharging the battery based on the NRV of the previous day. This policy is based on the assumption that the NRV of the next day will be similar to the NRV of the previous day. The NRV of the previous day can be seen as the NRV prediction for the next day. The thresholds can then be determined by doing a simple grid search for the best thresholds over the NRV prediction. The same penalty parameter can be added to the profit function to reduce the charge cycles of the battery.

4.2 Policies based on NRV generations

The simple baseline policy can be used with the NRV predictions for the next day. First, multiple full-day NRV samples are generated using a generative model. Each of these samples will be seen as a prediction for the NRV of the next day. The charge and discharge thresholds are determined for each of these samples using a simple grid search like in the baseline policy. The mean is taken over all the thresholds to determine the final thresholds for the next day. This results in a policy that uses the NRV samples of the generative model. This policy also uses the penalty parameter to reduce the charge cycles of the battery.

5 Literature Study

5.1 Electricity Price Forecasting

Forecasting the electricity price is a challenging task that has been researched extensively. Knowing the future electricity price is crucial for market participants to make informed decisions and optimize their operations and profit. Already since the early 2000s, researchers have been trying to predict the electricity price. The first models were based on time series analysis, but with the rise of machine learning, more advanced models have been developed. A rise in publications on this topic can be observed since 2005. This is described in the literature review by (Weron, 2014). An overview is given of the evolution of the methods used for electricity price forecasting. A significant shift can be observed towards integrating machine learning techniques with traditional statistical methods. The earliest models were based on time series analysis involving methods like autoregression, moving averages and their combinations (ARMA, ARIMA). These methods are not always able to capture the complex patterns in the electricity price. Therefore, researchers started to use more advanced models like neural networks, support vector machines, and random forests. The combination of statistical and machine learning models is more accurate. The statistical models are used to capture the linear patterns, while the machine learning models are used to capture the more complex non-linear patterns. This results in a more accurate and robust model. The more recent paper (Poggi et al., 2023) compares the performance of statistical and machine learning methods for electricity price forecasting. The authors use ARIMA and SARIMA as statistical methods and XGBoost as a machine learning method. They also compare the performance of Long Short-Term Memory (LSTM) networks for electricity price forecasting.

Because forecasting the electricity price is a challenging task with a lot of uncertainty, other generative methods to model the electricity price were researched. Generative modeling is a type of unsupervised learning that can be used to generate new samples from the same distribution as the training data. This can be used to generate new electricity price samples. The authors of (Lu et al., 2022) use General Adversarial Networks (GANs) to generate new electricity price scenarios. They introduce a deep learning framework called Conditional Time Series Generative Adversarial Networks (CTSGAN) to generate electricity price scenarios. This enhances the traditional forecasting models by allowing the generation of a diverse set of potential future scenarios. This capability allows the modeling of the uncertainty in the electricity price. The authors show that the CTSGAN model outperforms traditional forecasting models in terms of forecasting accuracy. Other generative models like normalizing flows can also be used to generate new electricity price samples. The authors of (Dumas et al., 2022) use normalizing flows to generate new electricity price samples. They show that normalizing flow models for electricity price forecasting are more accurate in quality than other generative models like GANs and Variational Autoencoders (VAEs). Not a lot of research has been done on using diffusion models for electricity price forecasting. The authors of (**rasul_autoregressive_2021**), however, show that autoregressive diffusion models can be used for time series forecasting and achieve good results. They apply the model on multiple datasets which includes an elec-

tricity price dataset. The use of diffusion models for NRV modeling is further explored in this thesis.

Most research on forecasting for the electricity market focuses on the electricity price for consumers. Another important aspect of the electricity market is the imbalance price. Not many papers have been published on forecasting the imbalance price. One paper (Dumas et al., 2022) describes the forecasting of the imbalance price. They do not forecast the price itself but rather forecast the NRV and use this to reconstruct the imbalance price. This approach will also be used in this thesis.

TODO: more information?

5.2 Policies for Battery Optimization

6 Results & Discussion

As discussed in the background information, the imbalance prices are based on the Net Regulation Volume (NRV). This means that the imbalance prices can be reconstructed from the sampled NRV. Multiple baselines and models will be compared that forecast and model the NRV using different metrics. The data utilized in this thesis is provided by Elia. Elia makes a lot of data public and provides them in quarterly hour or minute intervals. The data used in this thesis is on a quarterly hourly basis. This makes the number of input features and output features way more manageable and makes the training more computationally efficient. A full-day sample of the NRV exists of 96 values. One value for every quarter. Further research could be done using smaller data intervals to see if this improves the models.

6.1 Data

Elia offers a lot of different data on their website (TODO: open data citation). They provide data for the following categories: (TODO: Relevant? or too much information?)

- Balancing
- Transmission
- Power generations
- Congestion management
- Load
- Studies

The data useful to model the NRV is scattered over multiple categories. The data used in this thesis is the following:

TODO: ask Jonas: add urls to the correct data? via citation?

- **Imbalance prices per quarter-hour (Historical data)**
  This dataset contains the NRV and system imbalance in a quarter-hour interval. The data is available from 01-01-2015 to the present day. The NRV is used as the target variable that needs to be modeled but can also be used as input features. The next day NRV modeling can be conditioned on the real NRV of the previous day.
- **Measured and forecasted total load on the Belgian grid (Historical data)**
  Elia publishes what the total load on the Belgian grid is. This data is also provided in a quarter-hour interval. This data consists of the real load for a certain quarter but also the different forecasted loads. There are day-ahead and week-ahead forecasts available. The total load on the Belgian grid can be used as input features for the NRV modeling. The data is also available from 01-01-2015 to the present day.

- **Photovoltaic power production estimation and forecast on Belgian grid (Historical)**
  The photovoltanic power production is also available in a quarter-hour interval. The production is also forecasted day-ahead and week-ahead. The data is provided for each of the provinces in Belgium. Forecasts are also available for the 3 Belgian regions (Flanders, Wallonia, Brussels) and the total Belgian production. The photovoltanic data has been provided since 01-04-2018 and is available to the present day.
- **Wind power production estimation and forecast on Belgian grid (Historical)**
  Just as the photovoltanic power production data, wind power production is available in a quarterly-hour interval for each of the provinces and regions in Belgium. This data also includes the real production and the forecasts. An additional column is available that shows if the power is generated offshore or onshore. During this thesis, the offshore and onshore data will be combined. The wind power production data has been provided since 01-01-2015 and is available to the present day.
- **Day-ahead implicit net position (Belgium's balance)**
  The day-ahead implicit net position shows the total amount of electricity that will be imported or exported to neighboring countries. The trades are done on the day-ahead market and are thus known in advance. This data is available in a quarter-hour interval and has been provided since 01-11-2020 and is available to the present day. The data before 01-11-2020 is also available but only in hourly intervals.

A lot of data is available but only the most relevant data needs to be used. Experiments will be done to identify which data and features improve the NRV modeling. The data will be split into a training and test set. The training dataset starts depending on which data features are used but ends on 31-12-2022. The test set starts on 01-01-2023 and ends on (TODO: check the end date). This makes sure enough data is available to train the models and the test set is large enough to evaluate the models. The year 2023 is chosen as the test set because it is the most recent data available when the thesis experiments were conducted. Using data from 2022 in the test set also does not make a lot of sense because the trained models would be used to predict the future. Data from 2022 is not relevant anymore to evaluate the models.

6.2 Quantile Regression

6.2.1 Linear Model

The simplest model to be trained for the NRV modeling is the linear model. The linear model is trained using the pinball loss function explained in the section above. The outputs of the model are values for the chosen quantiles. The linear model can be trained in an autoregressive and non-autoregressive way. Both methods will be compared to each other. The linear model is trained using the Adam optimizer with a learning rate of 1e-4. Early stopping is used with a patience of 5 epochs. The linear model is evaluated using the mean squared error (MSE), mean absolute error (MAE), and continuous ranked probability score (CRPS). The influence of the

input features is also evaluated by training the models with different input feature sets.

There is a big difference in the number of parameters between the autoregressive linear model and the non-autoregressive linear model. The autoregressive model only needs to output the NRV quantiles for one value while the non-autoregressive model needs to output the NRV quantiles for all the quarters of the day. Assuming thirteen quantiles are used, the autoregressive has 13 output parameters while the non-autoregressive model has 13 * 96 = 1248 output parameters. The total number of parameters for the autoregressive model is 13 * (number of input features + 1) while the total number of parameters for the non-autoregressive model is 13 * 96 * (number of input features + 1). Assuming only the NRV history of the previous day is used as input features, the autoregressive model has 1261 trainable parameters while the non-autoregressive model has 121056 parameters. This is a huge difference in the number of parameters and thus the complexity of the model.

	MSE		MAE		CRPS	
	AR	NAR	AR	NAR	AR	NAR
NRV	39222.41	41219.98	152.49	152.26	91.56	**73.97**
NRV + Load	39266.29	47045.17	152.54	163.24	90.36	79.72
NRV + Load + PV	37642.66	46404.63	149.90	161.82	89.34	79.74
NRV + Load + Wind	39284.68	48148.10	152.32	164.84	88.60	79.51
NRV + Load + PV + Wind	36134.87	50312.85	146.22	169.06	84.56	79.85
NRV + Load + Wind + NP	37890.66	49442.48	149.37	167.90	86.19	76.72
NRV + Load + PV + Wind + NP	**35725.42**	49132.26	**145.64**	167.37	83.30	78.75

Table 3: Linear model results

Comparing the results of the autoregressive and non-autoregressive linear models, it can be seen that the non-autoregressive model has a higher MSE and MAE on the test set. The CRPS is, however, lower for the non-autoregressive model. The CRPS is calculated using the outputted quantiles while the MSE and MAE are calculated by sampling from the reconstructed distributions. Because of error propagation in the autoregressive model, the outputted quantiles also contain more error which leads to a higher CRPS. The non-autoregressive model does not suffer from this problem. During the training of the autoregressive model, the model does not take into account that it will be used to generate full-day samples and thus the error is propagated. This is one possible explanation for the higher CRPS of the autoregressive model.

The MSE and MAE of the non-autoregressive model are higher than the autoregressive model. This can be explained by the fact that the non-autoregressive model does not take into account the previous sampled value. Sampling is done for every quarter of the day independently. This can lead to large differences between the sampled values and thus can increase the MSE and MAE. The autoregressive model does take into account the previous sampled value and can adapt its quantile predictions based on this value so a smoother and more accurate sample can

be generated.

Another thing to note is the influence of the input features on the non-autoregressive linear model. When increasing the number of input features, the evaluation metrics are a lot worse in comparison with only using the NRV history of the previous day. A reason for this behavior could be that the model is not able to capture the patterns in the data because of the huge amount of input parameters. When using the NRV, Load, Photovoltanic power production, Wind power production, and the Net Position as input features, the non-autoregressive model has an input size of 864. This increases the complexity of the model as well. The total number of trainable parameters becomes 1,079,520. This is a huge number of parameters and the model is not able to learn the patterns in the data anymore.

The performance of the autoregressive linear model, however, improves with the addition of more input features. When using the NRV, Load, Photovoltanic power production, Wind power production, and the Net Position as input features, the autoregressive model has an input size of 484. This is almost half the size of the non-autoregressive model. The total number of trainable parameters becomes 6,305 which is way less than the non-autoregressive model.

An important thing to note is that the autoregressive model needs an additional feature to know which quarter of the day it is modeling. The quarter of the day also influences the value of the NRV. This can easily be seen in Figure 6. The figure shows the mean and standard deviation of the NRV values over the quarter of the day. These values change over the day which means the quarter is very valuable information for the model. The non-autoregressive on the other hand does not need this information because it models all the quarters of the day at once.

Providing the autoregressive model with the quarter of the day can be done in multiple ways. The quarter of the day can be provided as a one-hot encoded vector. The cyclic nature of the quarter would not be captured using a one-hot encoded vector. The vectors for quarter 0 and quarter 95 would be very different while they should be very close to each other. Other methods exist that do take the cyclic property of the quarter into account. Trigonometric functions can be used to provide the quarter of the day information. The quarter of the day can be mapped to a sine and cosine value which can be used as input features. The sine and cosine values are calculated as follows:

$$\sin(\frac{2\pi}{96} \times \text{quarter}) \quad \text{and} \quad \cos(\frac{2\pi}{96} \times \text{quarter}) \tag{8}$$

The sine and cosine values are then concatenated with the input features. Another method that can be used is adding an embedding layer to the model. The discrete quarter of the day value can then be mapped to a vector. The embedding layer itself is learned during the training process which allows the model to learn patterns between quarters. The length of the embedding vector can be chosen and experimented with. The quarter-of-the-day information is then concatenated with the input features. Other information (eg. day of the week, month,

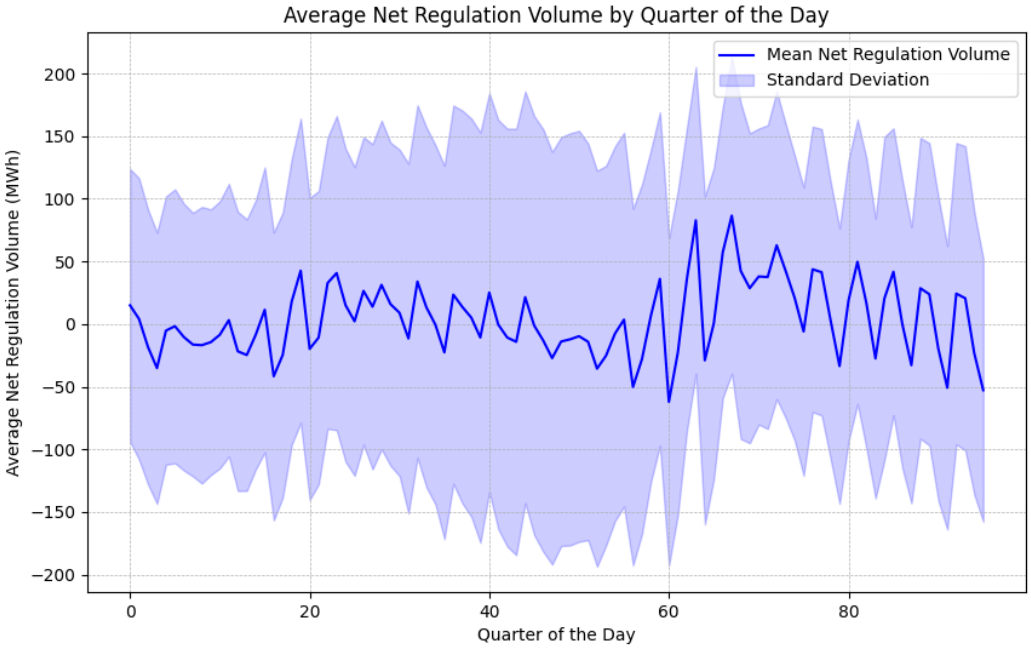


Figure 6: Mean and standard deviation of the NRV values over the quarter of the day

year) can also easily be added to the model using this method by just increasing the size of the embedding layer. The results of the linear model with the quarter information are shown in Table 4.

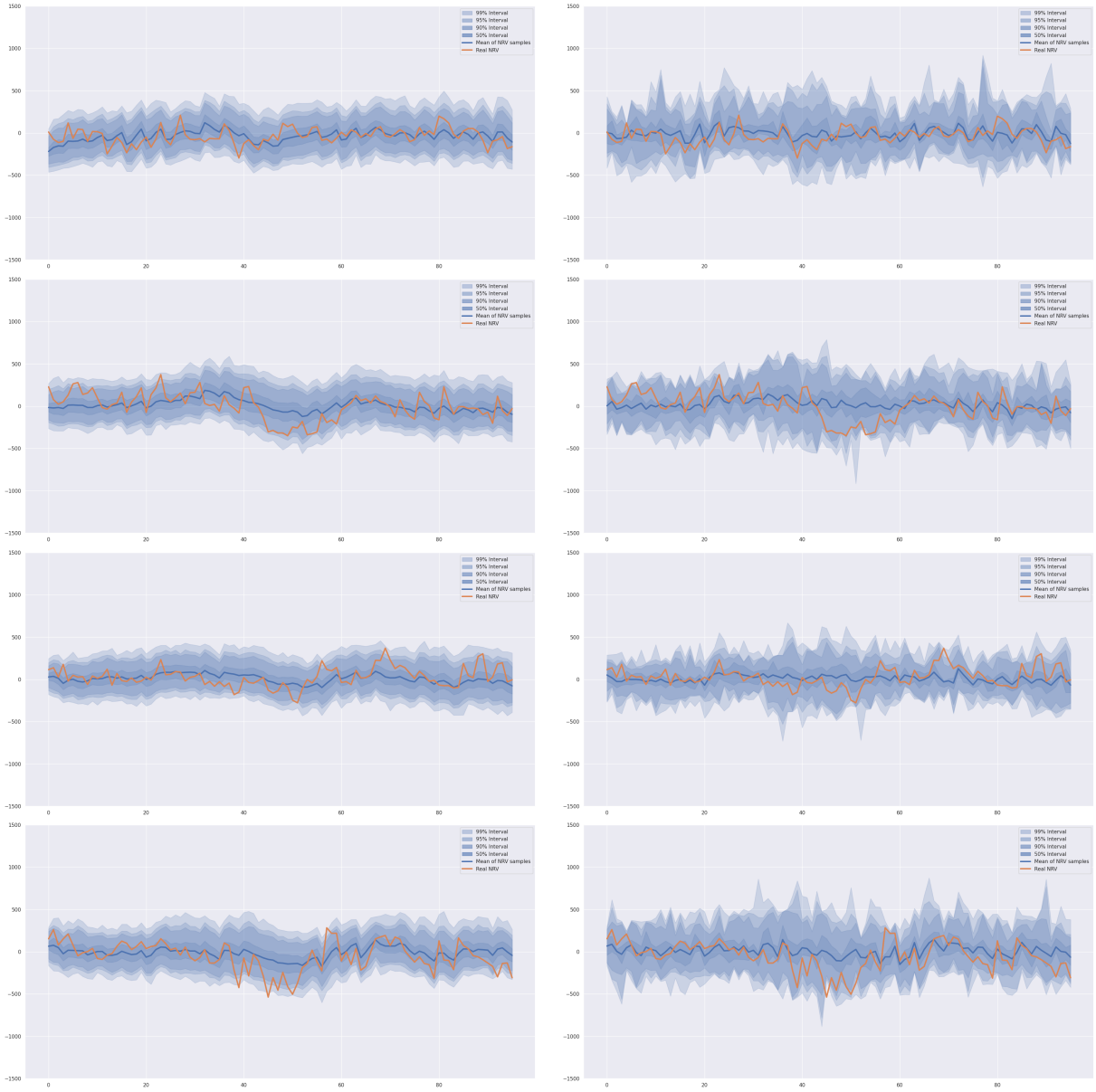
	MSE	MAE	CRPS
NRV	39222.41	152.49	91.56
NRV + QT	39069.96	152.06	90.90
NRV + QE **(2 dim)**	**38216.27**	**150.41**	**89.69**
NRV + QE **(5 dim)**	38617.17	151.20	89.72
NRV + QE **(8 dim)**	38423.30	150.89	89.81
NRV + Load + PV + Wind + NP	35725.42	145.64	83.30
NRV + Load + PV + Wind + NP + QT	34783.13	143.98	84.21
NRV + Load + PV + Wind + NP + QE **(2 dim)**	35746.01	146.01	85.54
NRV + Load + PV + Wind + NP + QE **(5 dim)**	**34031.71**	**142.29**	**79.99**

Table 4: Autoregressive linear model results with time features

The results show that adding the quarter embedding to the model improves all evaluation metrics for the autoregressive linear model. The quarter embedding is a valuable feature for the model.

Some examples of the generated full-day NRV samples are shown in Figure **??**. The examples are taken from the test set. The figure shows the confidence intervals of the NRV generations

and the mean NRV prediction. The confidence intervals and mean are calculated based on 1000 generated full-day NRV samples. The samples were generated using the input features NRV, Load, Wind, PV, Net Position, and the quarter embedding for the autoregressive model.



(a) Autoregressive linear model

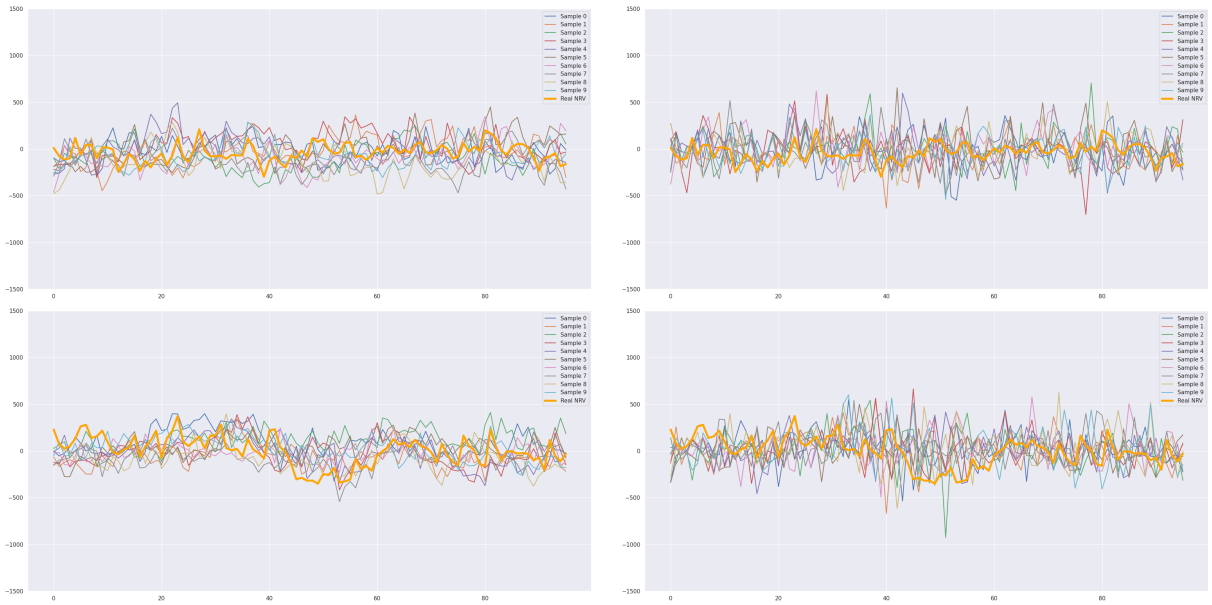
(b) Non-autoregressive linear model

Figure 7: Comparison of the autoregressive and non-autoregressive linear model samples.

When looking at the examples in Figure 7, it can be seen that the autoregressive linear model is already modeling the NRV quite well. The confidence intervals are quite small and the mean of the samples follows the trend of the real NRV. The mean of the samples, however, is way smoother than the real NRV. The real NRV has more peaks and fluctuations. The examples of the non-autoregressive model show another behavior. The confidence intervals are not as contained as the autoregressive model but fluctuates a lot more. A lot of peaks can be observed in the examples. The reason for this behavior is that the non-autoregressive model does not

take into account the previous sampled value. The sampled value of the next quarter is not dependent on the sampled value of the previous quarter. This can lead to a large difference between these values which results in samples with a high variance. The mean of the samples of the non-autoregressive model, however, does not follow the trend of the real NRV as well as the autoregressive model. The mean stays in a narrow range around zero.

Some samples for the examples from the test set are visualized in Figure 8. For the autoregressive model, the samples largely follow the trend of the real NRV while the non-autoregressive model has a lot of fluctuations and peaks. By visually looking at the samples themselves, the samples of the autoregressive model are more realistic than the samples of the non-autoregressive model.



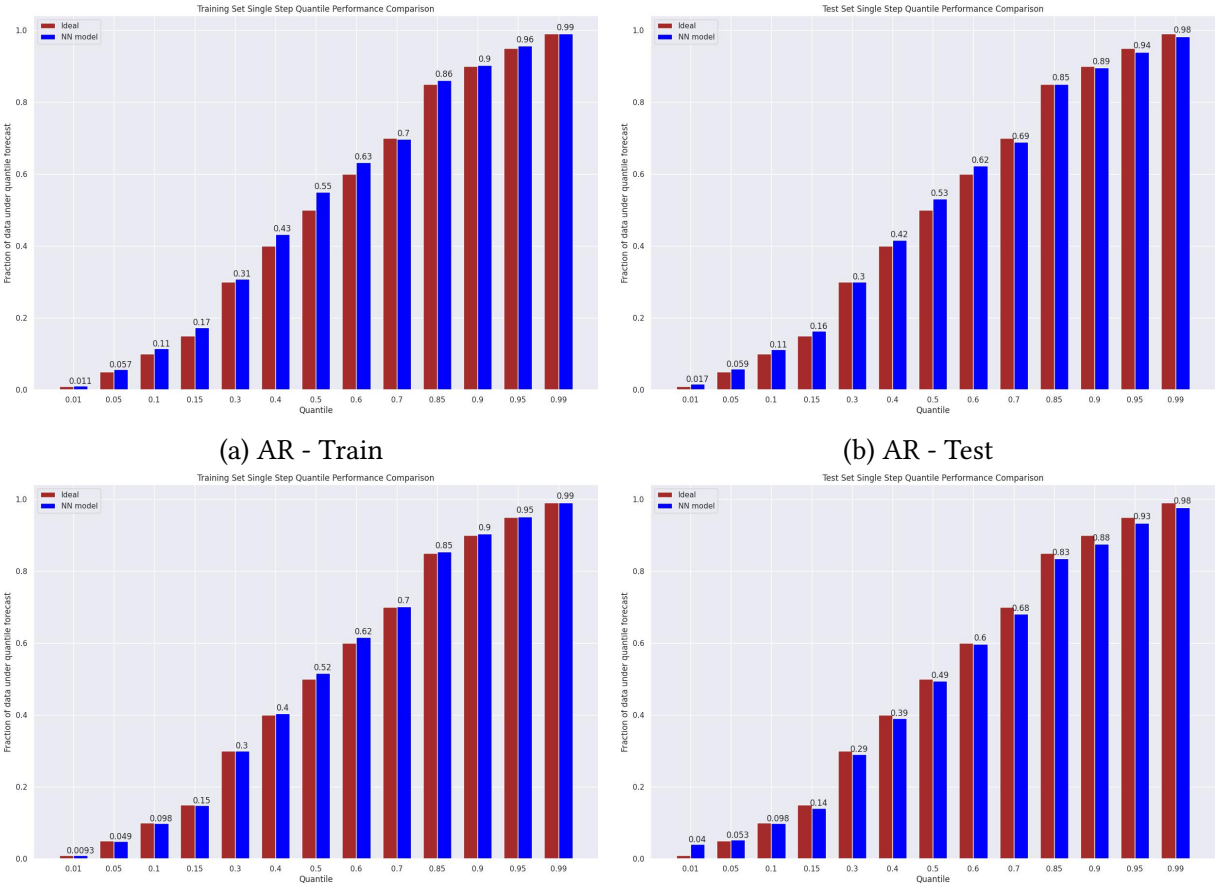
(a) Autoregressive linear model

(b) Non-autoregressive linear model

Figure 8: Samples for two examples from the test set for the autoregressive and non-autoregressive linear model. The real NRV is shown in orange.

Another way to evaluate the performance of the models is to look at the over/underestimation of the quantiles. For each day and every quarter in the test set, the quantiles are predicted by the model. Then for every quantile, it is checked how many times the real NRV is below the predicted quantile. For example, for the 10% quantile, around 10% of the real NRV values should be below the predicted quantile. This can be plotted for every quantile. These can be seen in Figure 9. The plots show the over/underestimation of the quantiles for the autoregressive and non-autoregressive linear models.

Multiple observations can be made when looking at the quantile performances in Figure 9. The fraction of the real NRV values that are below the predicted quantiles is very close to the expected fraction for the non-autoregressive model on the training set. The autoregressive model has a bit more trouble in the quantile range of 0.4 to 0.6. There, the model underesti-



(a) AR - Train

(b) AR - Test

(c) NAR - Train

(d) NAR - Test

Figure 9: Over/underestimation of the quantiles for the autoregressive and non-autoregressive linear models. Both the quantile performance for the training and test set are shown. The plots are generated using the input features NRV, Load, Wind, PV, Net Position, and the quarter embedding (only for the autoregressive model).

mates the quantiles. This means the model is predicting the quantile values too high which results in a bigger fraction of the real NRV under the quantile prediction. The test set shows a similar behavior for the autoregressive model with an additional small overestimation at the 0.95 and 0.99 quantiles. The non-autoregressive model has another behavior on the test set. There it can be observed that the model is underestimating the quantiles in the quantile range of 0.15 to 0.99. The reason for this different behavior in comparison with the training set can be overfitting.

Overall, the linear model is a good baseline to compare more complex models. It is, however, not able to capture the complex patterns in the data. In particular, the non-autoregressive model has a lot of trouble when more input features are added and the complexity of the model increases.

6.2.2 Non-Linear Model

Adding nonlinearity to the model can be done by adding some non-linear activations between linear layers. This improves the model's ability to learn more complex patterns in the data. The model is trained the same way as the linear model for quantile regression using the pinball loss. Because a non-linear model is more complex, it is more prone to overfitting the training data. Because of this, dropout layers are added to the model to prevent overfitting.

The architecture of the non-linear model is illustrated in Table 5. The autoregressive model begins with an input layer that converts the quarter of the day into an embedding. This layer concatenates the other input features with the quarter embedding. These combined features are then processed through a sequence of layers:

- Linear layer: Transforms input features to higher-dimensional space defined by hidden size.
- ReLU Activation Function: Introduces non-linearity to the model to learn complex patterns. This also helps with the vanishing gradient problems with deep neural networks.
- Dropout Layer: Regularizes the model to prevent overfitting. During training, random neurons are set to zero.

This sequence of layers is repeated N times to increase the depth of the model and enhance its ability to learn complex patterns. The final layer of the network is a linear layer that outputs the quantiles for the NRV prediction. For an autoregressive model, this is just the quantiles for a single quarter, whereas for a non-autoregressive model, the quantiles for every quarter of the day are outputted. The number of outputs is then the number of quarters in a day multiplied by the number of quantiles used.

Layer (Type)	**Output Shape**
Only for autoregressive model	
Time Embedding (Embedding)	[B, Input Features Size + Time Embedding Size]
Repeated Block (N times)	
Linear (Linear)	[B, Hidden Size]
ReLU (Activation)	[B, Hidden Size]
Dropout (Regularization)	[B, Hidden Size]
Linear (Linear)	[B, Number of quantiles]

Table 5: Non-linear Quantile Regression Model Architecture

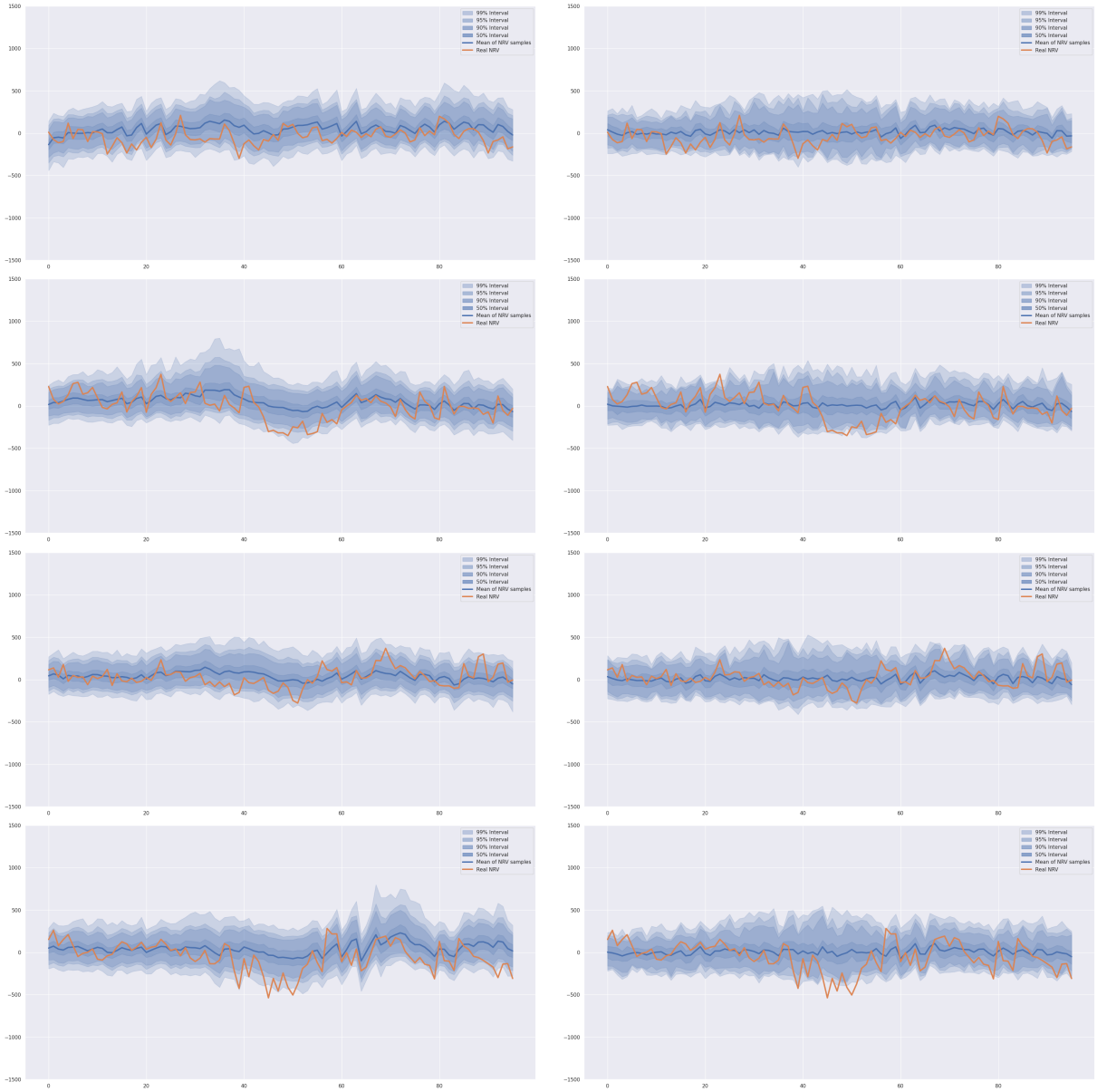
While this non-linear model is still quite simple, it offers the flexibility in tuning a limited set of hyperparameters. The hidden size of the linear layers and the number of layers can be experimented with, which can significantly influence the model's performance. The experiments

are executed with the same quantiles as the linear model. Multiple experiments are executed with different hyperparameters and input features. All results are shown in the Table 6.

Features	Layers	Hidden Size	MSE		MAE		CRPS	
			AR	NAR	AR	NAR	AR	NAR
NRV								
	2	256	38117.43	41574.38	147.55	153.83	86.42	75.61
	4	256	37817.78	40200.92	146.90	152.00	85.63	74.37
	8	256	36346.57	38746.81	144.80	148.82	84.51	74.55
	16	256	38624.83	39328.47	148.61	149.19	87.05	75.38
NRV + Load + PV + Wind								
	2	256	42983.21	42950.17	156.65	156.88	92.15	76.21
NRV + Load + PV + Wind + Net Position + QE (dim 5)								
	2	256	37785.49	42828.61	146.99	157.03	85.22	76.36
	4	256	34232.57	42588.16	139.78	157.20	80.14	73.75
	8	256	**32447.41**	40541.92	**137.24**	151.60	**79.22**	75.52
	2	512	44281.20	44018.79	158.63	159.06	91.82	77.99
	4	512	34839.79	41999.79	140.67	154.86	80.21	75.70
	8	512	34925.46	39774.38	141.11	150.62	81.11	74.67

Table 6: Non-linear quantile regression model results. All the models used a dropout of 0.2 .

The same behavior as the linear model is observed when looking at the metric differences between the autoregressive and non-autoregressive models. The autoregressive model performs better in terms of MSE and MAE, while the non-autoregressive model performs better in terms of CRPS. The results also give insight into the importance of the input features and hyperparameters. The addition of more input features improves the performance of the autoregressive model. The non-autoregressive model, on the other hand, does not benefit from this. The metrics are worse when more input features are added. This was also seen in the linear model. A reason for this behavior could be that the non-autoregressive model is not able to learn the complex patterns in the large set of input features. The non-autoregressive is provided with all the values for each quarter for which the quantiles need to be predicted. This increases the input size with 96 values each time a new forecast feature is added. Capturing patterns in this large input space can be a challenging task.



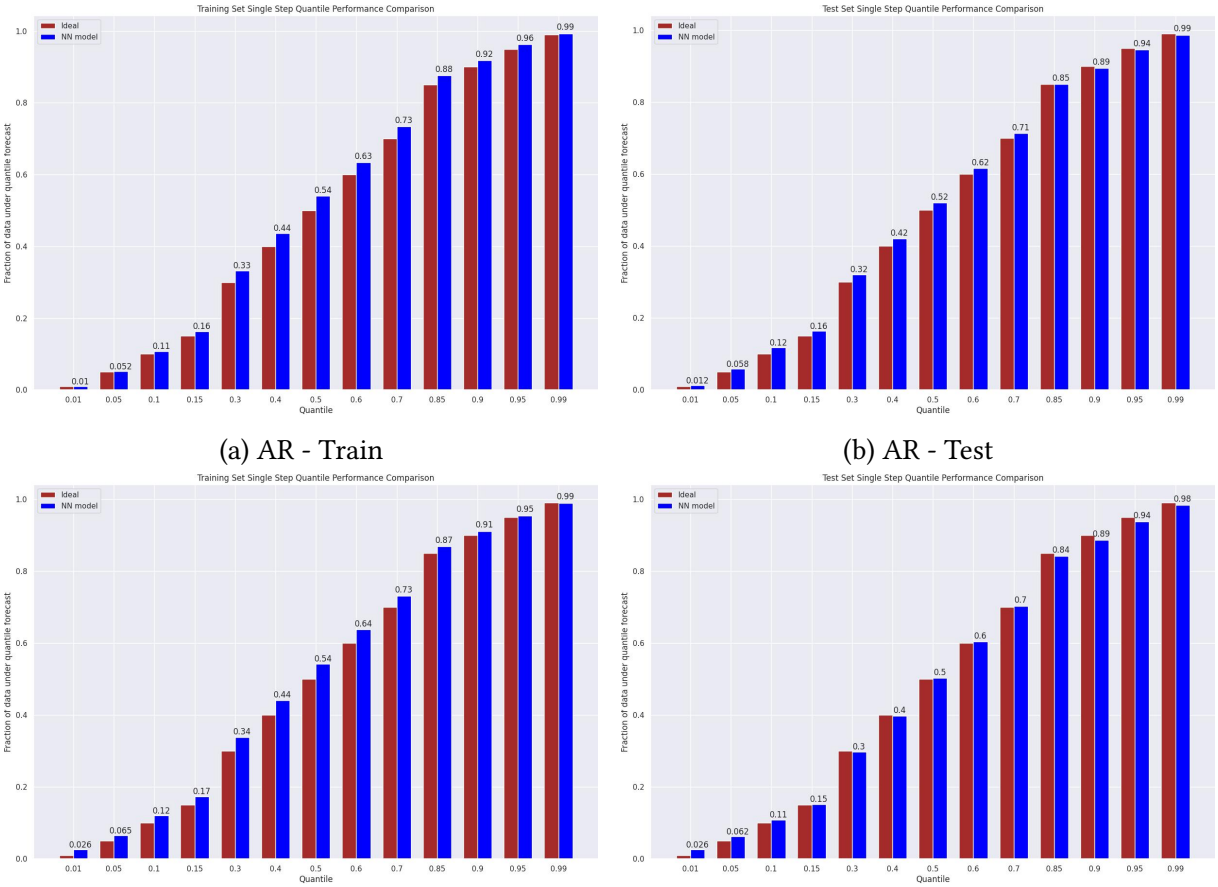
(a) Autoregressive linear model

(b) Non-autoregressive linear model

Figure 10: Comparison of the autoregressive and non-autoregressive non-linear model examples.

The examples from the test set for the non-linear model are shown in Figure 10. A big difference can be observed between the examples of the autoregressive and non-autoregressive models. The autoregressive model examples follow the actual NRV trend more closely than the non-autoregressive model. The mean of the samples generated by the non-autoregressive model is around zero for every quarter of the day. No clear trend can be observed in the samples. This is a clear indication that the non-autoregressive model is not able to learn the patterns in the data despite having a lower CRPS.

The plots in Figure 11 show the over/underestimation of the quantiles outputted by the non-linear models. Looking at the plots for the autoregressive models, the observation can be



(a) AR - Train

(b) AR - Test

(c) NAR - Train

(d) NAR - Test

Figure 11: Over/underestimation of the quantiles for the autoregressive and non-autoregressive non-linear models. Both the quantile performance for the training and test set are shown. The plots are generated using the input features NRV, Load, Wind, PV, Net Position, and the quarter embedding (only for the autoregressive model).

made that the fraction of the real NRV values under the quantiles is too big most of the time in comparison with the ideal fraction. This means the model is estimating the quantiles too high which results in a bigger fraction of NRV values below this value. The model overestimates the quantiles. The non-autoregressive model also suffers from this problem for the training set. For the test set, the lower quantiles are estimated too high and the higher quantiles are estimated too low. The quantiles in the middle are estimated quite accurately.

6.2.3 GRU Model

Another popular architecture to model sequential data is a recurrent neural network. There exist two main types of recurrent neural networks, the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU). The GRU is a simplified version of the LSTM, which has fewer parameters and is computationally less expensive. The GRU model can be trained for quantile regression in the same way as the linear and non-linear models using the pinball loss. There is, however, a difference in how the input data is structured and provided to the model. For linear and non-linear models, the data is provided in the shape of $(batch\_size, num\_features)$.

The recurrent neural network, on the other hand, expects the input data to be structured as $(batch\_size, time\_steps, num\_features\_per\_timestep)$. This is also explained in the background section about the recurrent neural network.

The GRU model architecture to predict the NRV quantiles is shown in Table 7. The model starts with an embedding layer that converts the quarter of the day into an embedding. This layer concatenates the other input features with the quarter embedding. The input of the TimeEmbedding is of shape (Batch Size, Time Steps, Input Features Size). The output of this layer is then passed to the GRU layer. The GRU layer outputs the hidden state for every time step. This results in a tensor of shape (Batch Size, Time Steps, Hidden Size). Only the last hidden state is relevant for the prediction of the NRV quantiles for the next quarter. The last hidden state should contain all the necessary information from the previous quarters to make the prediction. The last hidden state is then passed through a linear layer to output the quantiles for the NRV prediction. The input and output of the model depend if the model is trained using an autoregressive or non-autoregressive way. The non-autoregressive variant of the GRU model has two days worth of time steps. This results in $92 * 2$ time steps. The model then needs to output ($96 *$ number_of_quantiles) NRV quantile values.

TODO: Zielige visualisatie van model nu

Layer (Type)	**Output Shape**
Time Embedding	[B, Time Steps, Input + Time Embedding Size]
GRU	[B, Time Steps, Hidden Size]
Last state of GRU passed [B, Hidden Size]	
Linear	[B, Number of quantiles]

Table 7: GRU Model Architecture

Multiple experiments are conducted to find which hyperparameters and input features work best for the GRU model. The results of the GRU model are shown in Table 8.

Features	Layers	Hidden Size	MSE		MAE		CRPS	
			AR	NAR	AR	NAR	AR	NAR
NRV								
	2	256	39838.35	40097.62	150.81	150.37	85.04	76.12
	4	256	39506.55	39968.96	149.81	150.04	85.46	76.07
	8	256	37747.11	40400.37	146.67	151.03	83.67	76.59
	2	512	39955.79	40917.24	150.77	152.04	87.88	76.06
	4	512	43301.13	39954.62	156.73	150.14	89.78	76.25
	8	512	37681.71	40379.14	146.62	151.05	83.08	76.42
NRV + Load								
	2	256	33202.80	38427.91	138.02	147.27	79.62	84.17
	4	256	33600.73	38984.44	138.62	147.91	81.03	85.91
	8	256	32828.61	38343.98	136.82	146.44	79.42	84.22
	2	512	35979.57	41496.77	144.16	153.53	83.50	88.26
	4	512	32334.73	38000.40	135.92	146.10	78.82	83.99
	8	512	35177.39	41104.28	141.79	152.13	83.79	89.13
NRV + Load + PV + Wind								
	4	256	31594.55	39872.46	134.11	149.34	77.52	85.91
	8	256	31481.22	39704.37	133.45	148.59	77.26	85.62
	4	512	31368.31	39024.27	134.02	147.91	76.58	84.18
	8	512	34566.66	42397.86	140.13	154.00	82.09	89.87
NRV + Load + PV + Wind + Net Position + QE (5 dim)								
	4	256	30130.37	39906.53	130.92	149.78	75.02	84.88
	8	256	28560.67	37675.15	127.77	145.39	73.77	83.37
	4	512						
	8	512	27421.85	35238.98	125.32	141.02	72.73	80.92

Table 8: Autoregressive GRU quantile regression model results. All the models used a dropout of 0.2 .

TODO: Talk about the results + show plots of the samples with GRU model. Talk about the over/underestimation of the quantiles for the models. Plots have been made for this.

TODO: non autoregressive stuff?

6.3 Diffusion

Another type of model that can be used to generatively model the NRV is the diffusion model. This type of model is very popular for image generation. In the context of images, the diffusion model is trained by iteratively adding noise to a training image until there is only noise left. From this noise, the model tries to reverse the diffusion process to get the original image back. To sample new images using this model, a noise vector is sampled and iteratively denoised by the model. This process results in a new image.

This training process can also be used for other data types. An image is just a 2D grid of data points. A time series can be seen as a 1D sequence of data points. The diffusion model can thus be trained on the NRV data to generate new samples for a certain day based on a given input.

Once the diffusion model is trained, it can be used efficiently to generate new samples. The model can generate samples in parallel, which is not possible with autoregressive models. A batch of noise vectors can be sampled and passed through the model in one batch to generate the new samples. The generated samples contain the 96 NRV values for the next day without needing to sample every quarter sequentially.

TODO: Visualization of the diffusion model in the context of the NRV data.

7 Policies for battery optimization

7.1 Baselines

7.2 Policies using NRV predictions

References

Dumas, J., Wehenkel, A., Lanaspeze, D., Cornélusse, B., & Sutera, A. (2022). A deep generative model for probabilistic energy forecasting in power systems: Normalizing flows. Applied Energy, 305, 117871. https://doi.org/10.1016/j.apenergy.2021.117871

Lu, X., Qiu, J., Lei, G., & Zhu, J. (2022). Scenarios modelling for forecasting day-ahead electricity prices: Case studies in australia. Applied Energy, 308, 118296. https://doi.org/10.1016/j.apenergy.2021.118296

Poggi, A., Di Persio, L., & Ehrhardt, M. (2023). Electricity price forecasting via statistical and deep learning approaches: The german case [Number: 2 Publisher: Multidisciplinary Digital Publishing Institute]. AppliedMath, 3(2), 316–342. https://doi.org/10.3390/appliedmath3020018

Weron, R. (2014). Electricity price forecasting: A review of the state-of-the-art with a look into the future. International Journal of Forecasting, 30(4), 1030–1081. https://doi.org/10.1016/j.ijforecast.2014.08.008