universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Application and Analysis of Generative AI Algorithms for Probabilistic Energy Forecasts

verfasst von | submitted by

Verena Alton BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von | Supervisor:	Dipl.-Ing. Dr.techn. Lukas Exl Privatdoz.

Zusammenfassung

Für den optimalen Betrieb von Energiesystemen sind probabilistische Prognosen essenziell, besonders im Angesicht des großflächigen Ausbaus volatiler erneuerbarer Energiequellen. Allerdings sind traditionelle Prognose-Modelle oftmals nur in der Lage, deterministische Prognosen zu erzeugen. Diese Arbeit untersucht daher die Anwendung von Generative Adversarial Networks (GANs) zur probabilistischen Vorhersage einer besonders relevanten Variable für Energiesysteme, der Day-Ahead-Strompreise. Drei verschiedene Conditional Time Series GAN-Modelle werden implementiert, wobei exogene Faktoren wie Stromlast und Prognosen der erneuerbaren Erzeugung als Features verwendet werden. Die Performance der Modelle wird mit traditionellen Prognosemethoden, etwa SARIMAX und LSTMs, verglichen, wobei einerseits etablierte Metriken wie der Continuous Ranked Probability Score (CRPS) als auch ein einfacher Optimierungsansatz für eine Handelsstrategie verwendet werden. Die Ergebnisse zeigen, dass GAN-Modelle die Basismodelle hinsichtlich Prognosepräzision und Zuverlässigkeit sowie der Optimierungsergebnisse übertreffen. Damit können sie potentiell wertvolle Einblicke für Marktteilnehmer und Netzbetreiber liefern.

Abstract

Probabilistic forecasting is essential for the optimal operation of energy systems, especially with the rollout of volatile renewable energy sources. However, traditional forecasting models are often not able to produce probabilistic but only deterministic forecasts. Therefore, this thesis investigates the application of Generative Adversarial Networks (GANs) for probabilistic forecasting of day-ahead electricity prices. Three different Conditional Time Series GAN (CTSGAN) models are implemented, that use exogenous features like the electricity load and renewable generation forecasts. The model's performances are evaluated against traditional forecasting methods, including SARIMAX and LSTMs, using standard metrics like the Continuous Ranked Probabilty Score (CRPS) as well as a simple optimisation approach for a trading strategy. The results demonstrate that CTSGANs outperform baseline models in terms of forecast sharpness and reliability as well as optimisation results, and can therefore provide valuable insights for energy market participants and grid operators.

Acknowledgements

I would like to thank the AIT Austrian Institute of Technology and my colleagues there, Peter Widhalm, Stefan Strömer and Klara Maggauer, for all their valuable support. I also thank my supervisor at the University of Vienna, Lukas Exl, for his excellent supervision.

Declaration

The large language model ChatGPT 4.0 from OpenAI (2024) was applied for linguistic enhancements in parts of the written text of this thesis, as English is not the first language of the Author.

Contents

1 Introduction

This chapter was linguistically enhanced with a large language model.

The transition to a more sustainable and decarbonized energy sector requires the large-scale rollout of renewable energy technologies like photovoltaic and wind power plants. However, the variability of these energy sources leads to significant challenges for power system operation. Many (partly interdependent) factors, including not only the integration of renewables but also fluctuating electricity demand, weather conditions, and market conditions lead to a high complexity and volatility of modern electricity systems. To ensure grid stability and to prevent system disturbances, it is necessary that electricity supply and electricity demand match exactly at all times. Therefore, accurate forecasts of different energy system variables are essential for system operators. They enable not only grid stability but also more efficient trading strategies, reduction of green house gas emissions, and an optimized operation of flexible energy assets. Chai, Li, Abedin, and Lucey (2024); Lago, Marcjasz, De Schutter, and Weron (2021); Maji, Shenoy, and Sitaraman (2023)

Predictions of energy system-relevant variables benefit a wide range of stakeholders, one example being biomass plants. Unlike other renewable energy sources, biomass plants have the can store and dispatch energy strategically. However, their operational constraints, such as the time required to convert raw biomass into usable biogas, needs forward-looking decision-making. Probabilistic forecasting of electricity prices can provide biomass plant operators with valuable insights to optimize their dispatch strategies, which leads to enhanced economical returns for the plant operators.

Recent advancements in artificial intelligence (AI) have shown the potential of generative models for probabilistic forecasting. Unlike traditional forecasting methods that generate single-point estimates, generative AI models can produce full probability distributions, where they can capture the inherent uncertainty and volatility of energy markets. This ability to model uncertainty is especially valuable in the context of electricity price forecasting, where market fluctuations are influenced by numerous exogenous factors. Gault (2024); Lago et al. (2021)

This thesis explores the application of generative AI models for day-ahead electricity price forecasting. By implementing and analysing generative machine learning models, we aim to assess whether these models can enhance forecasting accuracy, improve decision-making for energy market participants, and contribute to a more resilient and efficient energy system.

1.1 Background and Motivation

This master thesis contributes to the project *transpAIrent.energy*, which has two primary objectives: (1) to develop innovative generative AI algorithms to generate probabilistic forecasts for

energy system-relevant variables and publish them transparently on a publicly accessible platform, and (2) to use these forecasts to optimize flexible renewable energy assets, enhancing their economic, efficient, and sustainable operation.

The project is being carried out as part of the 2023 "AI for Green" call for proposals from the Federal Ministry for Climate Protection, Environment, Energy, Mobility, Innovation and Technology (BMK). The processing is carried out on behalf of the BMK by the Austrian Reseach Promotion Agency (FFG). The consortium of the project consists of the AIT Austrian Institute of Technology GmbH as project leader, the Projektplanung- Beratungs und Enticklungs GmbH, the B-SEC better secure KG and the UBIMET GmbH.

1.1.1 Project Motivation

The increasing complexity and volatility of modern electricity markets present significant challenges for system operators, energy producers, and consumers. These challanges can be addressed by utilizing artificial intelligence (AI) methods to model the complex dynamics of today's energy system. The *transpAIrent.energy* platform is being developed to provide open access to AI-based probabilistic forecasts for key energy system variables in Austria.

In the following paragraphs, three key challenges within the Austrian electricity system are outlined together with the corresponding contributions of the project:

Predictive Accuracy and Efficiency

Electricity markets show short-term fluctuations due to many different factors, including renewable energy generation, demand variations, and system requirements such as frequency restoration. Traditional forecasting methods often struggle to capture these complex interactions, which leads to suboptimal predictions. The project addresses this limitation by applying advanced generative AI techniques to enhance the probabilistic forecasting of key market parameters. By generating more accurate predictions, the resulting forecasts have the potential for market participants to make informed, data-driven decisions, which can then improve market efficiency and stability.

Transparency and Accessibility

Transparency in energy markets is fundamental to ensure efficiency, liquidity, and competition. Open access to market data reduces the potential for market power abuse and to make informed decisions. The *transpAIrent.energy* platform is developed to enhance transparency by providing publicly accessible short- to mid-term forecasts for different energy system variables, such as day-ahead electricity prices, CO_2 intensities and balancing energy activations. Additionally, as specific forecast data for Austria is currently limited, the platform improves data availability.

Proof of Concept and Impact Evaluation

While theoretical advancements in forecasting and optimization provide valuable academic insights, their practical use should be demonstrated through real-world applications. To validate the developed AI-based probabilistic forecasts, the project integrates them into a digital twin of a power plant and applies stochastic operational optimization strategies. By assessing the impact of these forecasts on the operation of flexible renewable energy assets, the project provides a quantitative evaluation of their benefits in terms of economic efficiency, environmental sustainability, and grid stability. The publication of results on an open platform ensures that also decentralized energy actors can use the results, including smaller market participants who typically lack access to advanced forecasting tools.

Contribution to Climate and Energy Goals

The project directly supports Austria's climate and energy objectives by developing innovative generative AI models for forecasting electricity prices, CO_2 intensities, and other system-relevant variables. These forecasts enable optimized operation of flexible renewable and agricultural energy systems, and promote a balance between economic viability and environmental impact. By reducing CO_2 emissions and improving resource efficiency, the project contributes to the transition towards a more sustainable and resilient energy system.

1.1.2 Project Structure

The bullet points below present the methodological approaches of the project:

1. **Data Collection, Evaluation, and Documentation:** Since generative AI methods rely on high-quality data, we assess the availability of relevant datasets, conduct rigorous quality and consistency evaluations, and apply correction mechanisms where necessary. Additionally, AI techniques are used to generate highly spatially and temporally resolved weather forecasts, which are important for optimizing renewable energy assets. To achieve transparency, all methods, code, and processed datasets are documented and unified into a central database, which is then publicly accessible via the *transpAIrent.energy* platform.

2. **Algorithm Development and Validation:** Based on the available datasets, we review, implement, and validate generative AI forecasting algorithms for energy system-relevant time series data. Forecast quality is assessed using standardized evaluation metrics and backtesting approaches. Transparent documentation and publication of methodologies and results on the platform ensure traceability and verifiability.

3. **Platform Development and Implementation:** A publicly accessible platform is implemented to maximize transparency and usability. Stakeholders are actively involved in its design to ensure accessibility and relevance. The platform serves as a repository for all data, methods, and results. Furthermore, an operational optimization configurator is developed to enable "low volume" renewable asset operators to assess the benefits of

our optimization strategies for their own installations, to enable decentralized renewable energy generation.

4. **Proof-of-Concept:** To quantitatively validate the AI-generated forecasts and evaluate their practical applicability, we develop a multi-objective stochastic operational optimization strategy for renewable energy assets. Experimental validation is conducted using a digital twin of three real-world biomass plants, as well as live deployments at the sites themselves. The optimization framework integrates multiple objectives, including CO_2 emissions reduction, cost minimization, and self-sufficiency. The results provide a quantitative benchmark for assessing the effectiveness of our approach and are transparently published on the platform.

5. **Project Management and Dissemination:** Continuous project monitoring and management ensure steady progress. Dissemination activities focus on promoting the added value of the project through transparent communication and publication of results. By optimizing the operation of renewable energy flexibilities, *transpAIrent.energy* contributes to a more sustainable energy future.

This master thesis' focus is on the part *Algorithm Development and Validation*, as models for generative AI forecasts are developed, implemented, and validated with standard metrics as well as simplified optimisation goals.

1.2 State of the Art

The target variable in this thesis is the day-ahead electricity price in Austria. As it is modelled as time series, general time series forecasting models can be interpreted as State of the Art.

1.2.1 Deterministic Models

A well-established and widely used model for time series forecasting is the Autoregressive Integrated Moving Average (ARIMA) model and its extension with exogenous variables (ARMAX) as well as seasonality (SARIMAX). They score with interpretability and comparatively good performance, especially considering their simplicity. However, they are not perfectly able to model all non-linear complexities of modern electricity markets. Nevertheless, they establish a solid foundation for forecasting and are widely used in practice according to Shah, Iftikhar, and Ali (2022). A SARIMAX model is used as a baseline model in this thesis, and therefore explained in detail in Section 1.4.1.

The strength of Deep neural networks (DNNs) is to capture non-linear and very complex patterns in data. However, they are not designed for sequential data like time series (see Lago et al. (2021); Tschora, Pierre, Plantevit, and Robardet (2022)).

Long short-term memory (LSTM) networks are more suitable for the given forecasting task than DNNs as they are additionally able to incorporate and model long-term trends in time series, according to Tschora et al. (2022). Therefore, an LSTM model serves as another baseline for comparison in this thesis and is further introduced in in Section 1.4.2.

Hybrid models combine different machine learning techniques and often lead to superior results. For example, models using convolutional neural networks (CNNs) together with LSTM networks seem very promising, as summarized in Lago et al. (2021). In this work, the approach of combining an LSTM with a GAN-based model is introduced and tested, as explained in Section 2.1.5.

Ensemble methods score with robustness and accuracy, as they use not only the predictions from one single model but from different models. There are various mechanisms, including stacking or bagging, that are frequently used in state-of-the-art forecasting systems, as pointed out by Lago et al. (2021).

1.2.2 Generative Models

A novel generative AI approach, WIAE-GPF (Weak Innovation AutoEncoder-based Generative Probabilistic Forecasting), has been proposed by X. Wang, Zhao, and Tong (2024) for probabilistic forecasting of electricity market signals. This method generates realistic future samples of multivariate time series data, including electricity prices, using the Wiener-Kallianpur innovation representation, which enhances interpretability compared to typical black-box models. WIAE-GPF serves as a non-parametric generalization of the Wiener/Kalman filter-based forecasting and uses a variational autoencoder (VAE) learning a latent representation of historical price data, capturing underlying patterns and dynamics. The trained VAE generates diverse sets of possible future price trajectories by sampling from the learned latent distribution. The approach has shown superior performance compared to classical techniques and other machine learning methods in tests on data from U.S. electricity markets.

In Lu, Qiu, Lei, and Zhu (2022), an improved GAN model called CTSGAN (Conditional Time Series GAN) is introduced for day-ahead electricity price forecasting. This model extends the Time Series GAN proposed in Smith and Smith (2020), which itself builds upon the Wasserstein GAN (WGAN) framework from Arjovsky, Chintala, and Bottou (2017). CTSGAN enables scenario-based forecasting by generating multiple plausible price trajectories, capturing the inherent uncertainty and volatility in electricity prices. The model has been validated with case studies from various regions of the Australian National Electricity Market (NEM), demonstrating effectiveness across diverse market conditions. This thesis implements the proposed CTSGAN model as GAN_0 (see Section 2.1.3) and conducts further tests and validation on the Austrian electricity market. Furthermore, modifications of the model concerning the architecture as well as training are proposed and implemented as GAN_1 (see Section 2.1.4) and evaluated against the original model as well State-of-the-Art baseline models.

1.3 Theoretical Foundations of AI

The following section provides a general overview on artificial intelligence (AI) with a focus on models that are essential for understanding both the generative AI model investigated in this thesis and the baseline models used for evaluation and comparison (LSTM network and multivariate linear regression, see Sections 1.4.2 and 1.4.3 respectively).

The primary references of this sections are the lecture notes from *Angewandtes Maschinelles Lernen* by Exl and Schaffer (2025) and from *Mathematical Foundations of Deep Learning* by Sourangshug (2025).

1.3.1 General Overview

AI refers to the interdisciplinary field of mathematics, computer science and data engineering that investigates algorithms and models that are able to learn from data. The applications are widely spread from simple regression tasks to complex problems like natural language processing. Let X be the input space and Y be the output space, then an AI system can be described as a function:

$$f : X \to Y. \tag{1.1}$$

AI tries to learn a mapping f that works not only on the given training data but also on unseen data.

A prominent subfield of AI is Machine Learning (ML), where the goal is to create algorithms that can learn from data and improve performance over time without being explicitly programmed. ML methods are commonly employed in data-driven modeling, where they learn relationships between variables in complex, nonlinear, and high-dimensional domains. Among ML methods, neural networks and generative models are becoming increasingly popular due to their success in areas such as image generation, language modeling, and time series forecasting.

ML techniques can be categorized in multiple ways based on how they learn from data, how they model the problem space, and how they process input data:

Supervision-Based Learning Machine learning algorithms can be classified based on the availability of labelled data:

- **Supervised Learning:** The model uses a labelled training dataset with pairs (x_i, y_i) to learn a function f from the input space X to output space Y. The objective is to minimize a loss function $L(f(x), y)$, commonly using empirical risk minimization:

$$\min_f \sum_{i=1}^{N} L(f(x_i), y_i) \tag{1.2}$$

  Examples for this kind of learning are linear regression as well as neural networks.

- **Unsupervised Learning:** No labelling is needed for this kind of learning, as the model purely learns from patterns in the data. For example, clustering means finding categories of data points with similar properties and assigning them as best as possible to these categories. Another example is dimension reduction, e.g. principal component analysis, where the model tries to eliminate dimensions with redundant information for a comprehensive presentation of data.

- **Semi-Supervised Learning:** In this approach only a small amount of the data is labelled whereas the bulk of the data is unlabelled. This is often beneficial to fully supervised learning regarding the high efforts needed to label data.

- **Reinforcement Learning (RL):** The model interacts with an environment by trial and error and gets punished or rewarded depending on the outcome. The goal is to identify the most rewarding actions and therefore achieving the maximum rewards.

Incremental vs. Batch Learning Based on how data is processed during training, learning can be classified as:

- **Batch Learning:** The model is trained using the entire dataset at once. This approach is computationally intensive and needs all data to be available upfront.

- **Incremental Learning:** Also known as online learning, this approach updates the model continuously as new data becomes available. Incremental learning is useful in real-time systems and streaming data scenarios.

1.3.2 Linear Regression

Linear regression aims to identify and model linear relationships between the input and target variables. Specifically, given an input vector $\mathbf{X} \in \mathbb{R}^n$ and an output variable $Y \in \mathbb{R}$, a linear regression model assumes a linear relationship of the form:

$$Y = \mathbf{w}^\top \mathbf{X} + b + \varepsilon,$$

where $\mathbf{w} \in \mathbb{R}^n$ is the vector of regression coefficients, $b \in \mathbb{R}$ is the intercept or bias, and ε is the residual error, often assumed to be Gaussian with zero mean and constant variance: $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

The model parameters $\mathbf{w}$ and b are estimated using a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, typically by minimizing the sum of squared residuals:

$$\min_{\mathbf{w},b} \sum_{i=1}^{m} (y_i - \mathbf{w}^\top \mathbf{x}_i - b)^2.$$

This approach is also known as ordinary least squares (OLS). The calculated linear approximation of the target function assumes uncorrelated and homoscedastic residuals.

While linear regression is very popular because of its interpretability and simplicity, it provides only point estimates for predictions and does not inherently quantify uncertainty. A natural extension to a probabilistic framework is the multivariate normal distribution model, which allows for both prediction and uncertainty estimation. This approach is implemented as a baseline model and discussed in more detail in Section 1.4.3.

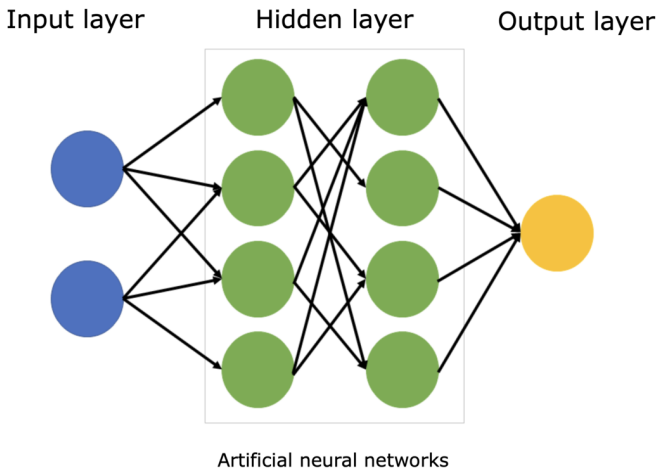

Figure 1.1: Architecture of a neural network. Image obtained from TseKiChun on Wikimedia Commons under the terms of the Creative Commons Attribution License (CC BY-SA 4.0) TseKiChun (2021)

1.3.3 Neural Networks

Neural networks are very prominent and widely used in the field of machine learning. In their structure they resemble biological neural networks as they consist of nodes that are connected to each other. These nodes consist of activation functions and are used to process input data and learn patterns from it.

Architecture of Neural Networks A standard feedforward neural network is also known as multi-layer-perceptron, as multiple layers of linked activation functions form the basis of the architecture. It is a direct extension of single-layer-perceptrons, which consist of only one layer of nodes (activation functions). A multi-layer-perceptron has at least an input and output layer with an arbitrary amount of hidden layers in-between. See Figure 1.1 for a visualisation of a feed forward neural network architecture.

Mathematically, the output of a single neuron assuming a wholly connected layer is given by:

$$z_i^{(l)} = \sum_j w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)}, \tag{1.3}$$

where:

- $z_i^{(l)}$ is the value of the i-th neuron in layer l before applying the activation function ϕ,
- $w_{ij}^{(l)}$ are the weights in the weight matrix $W^{(l)}$ connecting neuron j in layer $l-1$ to neuron i in layer l,
- $a_j^{(l-1)}$ is the past activation value of neuron j from the previous layer,
- $b_i^{(l)}$ is the bias term.

Applying the activation function ϕ to $z_i^{(l)}$ gives then the final output of the neuron:

$$a_i^{(l)} = \phi(z_i^{(l)}). \tag{1.4}$$

Commonly used activation functions are:

$$\text{Sigmoid:} \quad \sigma(z) = \frac{1}{1+e^{-z}}, \tag{1.5}$$

$$\text{ReLU:} \quad \phi(z) = \max(0, z), \tag{1.6}$$

$$\text{Tanh:} \quad \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{1.7}$$

Training Neural Networks The training of neural networks involves optimization algorithms that aim to minimize a loss function $\mathcal{L}$ that measures the distance between the predictions from the neural network and the real data. The most popular optimization algorithm is the gradient descent in its various forms. Different loss functions $\mathcal{L}$ are used for different tasks: while regression tasks often utilize the simple mean squared error (MSE), classification tasks would rather apply the cross-entropy loss.

$$\text{MSE:} \quad \mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{1.8}$$

$$\text{Cross-entropy:} \quad \mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i). \tag{1.9}$$

The parameters (weights and biases) of the network are then updated using backpropagation. η presents the learning rate, an important hyper-parameter, that determines the step size of the updates:

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}}. \tag{1.10}$$

Training deep neural networks comes with several challenges:

- **Overfitting:** A model learning the training data almost perfectly has the problem that it probably will not generalize well to unseen data. This means, it learned the training data samples by heart without learning the actual distribution. This can be addressed with regularization methods, dropout, or early stopping.
- **Vanishing/Exploding Gradients:** Mathematically there is no restriction on how large or small the gradients can get in the back-propagation process. However, too small or large gradients are a problem for effective learning and can even result in an early termination of the learning process. Techniques to counter this problem include batch normalization as well as automatic learning rate adaptions.
- **Non-convex Optimization:** The loss surface in neural networks is highly non-convex, making it difficult to find the global minimum. Optimizers like Adam and RMSprop are ad-

vancements of the classical gradient descent incorporating different techniques to handle this problem.

- **Computational Cost:** Training large neural networks requires significant computational resources and time. Using GPUs and parallelization strategies helps to speed up training.

Hyper-parameter Tuning There are various hyper-parameters defining a neural network, which have to be selected very carefully to achieve the best performance of the model. Well-selected hyper-parameters are essential to counteract all of the training challenges discussed in the last paragraph. They include not only the learning rate or batch size, but also fundamental properties of the neural network like the number of layers or the type of activation function. Popular approaches for hyper-parameter tuning include grid search and random search.

Types of Neural Networks Many enhancements of standard feed-forward neural networks have been developed that specialise on different types of data:

- **Feedforward Neural Networks (FNNs):** Basic architecture where information flows in one direction, as discussed in this section.
- **Convolutional Neural Networks (CNNs):** Designed for spatial data, widely used in image processing.
- **Recurrent Neural Networks (RNNs):** Developed for sequential data, like language data and time series, they have loops between the layers. Discussed in the following section as primer for LSTMs.
- **Long Short-Term Memory Networks (LSTMs):** Based on RNNs but with a gating mechanism additionally to the memory loops. Better for long-term dependencies and less risk of vanishing gradients. Introduced more formally in Section 1.4.2 as they are used as baseline in this thesis.
- **Transformer Networks:** Based on self-attention mechanisms and suitable for sequential data with even longer dependencies than the LSTM. Mentioned as potential alternative generative AI model for probabilistic time series forecasting in Chapter 5, but a rigorous introduction would go beyond the scope of this thesis.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) adopt loops between layers to simulate a memory that gives information from the previous time step to the current time step. This is especially useful to handle sequential data like time series. See Figure 1.2 for a visualisation of the architecture of a recurrent neural network.

The key idea behind RNNs are the hidden states at each time steps that are updated from previous time steps. Let h_t be the hidden state at time step t, then its value depends not only

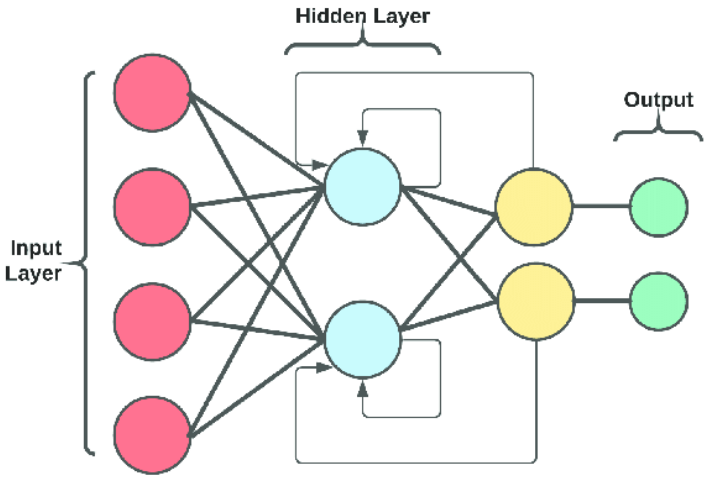

Figure 1.2: Architecture of a recurrent neural network. Image obtained from Fernando Xavier Arias on Research Gate under the terms of the Creative Commons Attribution License (CC BY-SA 4.0) Arias et al. (2022)

on the input x_t but also on the hidden state h_{t-1} of the previous time step $t-1$:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b), \tag{1.11}$$

where ϕ is again an activation function, W_h is the weight matrix for the hidden states, W_x is the weight matrix for the input, and b is a bias vector.

The final output y_t of time step t can then be calculated with an output weight matrix W_y, another bias term c and another activation function ψ:

$$y_t = \psi(W_y h_t + c). \tag{1.12}$$

A significant constraint for the use of RNNs is their difficulty with handling long-term dependencies. Vanishing or exploding gradients, as discussed in the challenges of training neural networks, often occur when RNNs are trained on long sequences. Therefore, enhanced architectures where developed to counter these limitations and offer possibilities to model also complex sequential data with long-term dependencies. These include Long Short-Term Memory Models (LSTMs), which are used as a baseline in this work and therefore more rigorously introduced in Section 1.4.2.

1.3.4 Generative AI Models

Generative Artificial Intelligence (generative AI) are machine learning models that are able to generate synthetic data samples. Unlike discriminative models, which estimate conditional probabilities $P(y|x)$, generative models learn the underlying joint probability distribution $P(x, y)$ or the marginal data distribution $P(x)$ directly. This ability allows them to not only generate new samples but also to make probabilistic inferences using Bayes' theorem as presented in

Exl and Schaffer (2025):

$$P(y|x) = \frac{P(x,y)}{P(x)}. \tag{1.13}$$

Let $x \in \mathbb{R}^n$ represent a data sample from an unknown true distribution $p_{\text{data}}(x)$. The objective of generative modeling is to approximate $p_{\text{data}}(x)$ with a model distribution $p_{\text{model}}(x;\theta)$, parametrized by θ. This is achieved by calculating the likelihood of the observed data and maximizing it:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \log p_{\text{model}}(x_i;\theta), \tag{1.14}$$

where N is the number of training samples.

Generative Modelling Approaches

The following section provides an overview of different modelling approaches for generative AI and is based on Feuerriegel, Hartmann, Janiesch, and Zschech (2024); Kalota (2024); Stryker and Scapicchio (2024).

Generative Adversarial Networks (GANs). GANs are a prominent class of generative AI models proposed by Goodfellow et al. (2014). The key mechanism of the GAN is that two separate neural networks work against each other: a generator G learns to produce synthetic data from random noise $z \sim p(z)$, and a discriminator D that tries to distinguish this generated data from real data. The joint training of the two networks is a competitive minimax game, where G wants to deceive D, and D learns to improve its classification. The training objective is defined as:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))], \tag{1.15}$$

so the generator is minimizing its loss function when it generates output that is categorized as real by the discriminator, i.e., samples that are similar to the real data. GANs are widely used for image creation and synthetic data generation and are investigated in this thesis concerning their ability to produce probabilistic forecasts, with further details provided in Section 2.1.1.

Variational Autoencoders (VAEs). VAEs, proposed by Kingma and Welling (2022), are probabilistic generative models that use a probabilistic encoder $q_\phi(z|x)$ together with a probabilistic decoder $p_\theta(x|z)$ to generate output from an input $x \in X$ via a latent space Z:

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\text{KL}}(q_\phi(z|x)\|p(z)), \tag{1.16}$$

where D_{KL} denotes the Kullback-Leibler divergence. VAEs are popular for their efficient latent space representation and sound stochastic foundation. This makes them particularly useful for tasks like image reconstruction and anomaly detection.

Normalizing Flows. Normalizing Flows are generative AI models that start with a base distribution, such as a standard Gaussian, and transform it via a bijection (the normalizing flow) into a target distribution. The normalizing flow is calculated with the rules of calculus for change-of-variable. In fact, given $z \sim p(z)$ and $x = f(z)$, the model can compute the exact likelihood of the generated data using the change-of-variables formula:

$$p(x) = p(z) \left| \det \left(\frac{\partial f^{-1}}{\partial x} \right) \right| . \tag{1.17}$$

Since the transformation is invertible, the density of of the target distribution can be evaluated. Therefore, explicit likelihoods are given by the model and don't have to be derived from repeated sampling. A challenge concerning normalizing flows is to find and define transformations that are effective and fulfil the mathematical requirements (e.g., maintaining a tractable Jacobian).

Diffusion Models. Diffusion models are a class of generative AI models where data is generated via the reversal of a diffusion process. During training, a stochastic process is used to add noise to data and therefore deprave it. The model then learns to reverse this process and denoise the data step-by-step. Applying the trained diffusion model on random noise will then produce data similar to the training set. Denoising Diffusion Probabilistic Models (DDPMs), for example, achieve state-of-the-art results in image synthesis. While diffusion models score with high stability in training and very good results concerning the quality of the data samples, the high amount of steps needed for sample generation is comparatively time consuming.

Challenges

Despite their success and recent popularity, generative AI models still face several challenges that limit their practical application. Training instability is a major issue in adversarial models like GANs, where a delicate balance between generator and discriminator convergence is needed. Mode collapse, where the generator produces only a few different outputs that work well to deceive the discriminator and doesn't cover the full target distribution, is another common problem. Furthermore, these models often require large datasets and significant computational resources to train. Improving the robustness, interpretability, and efficiency of generative models are therefore important research areas. Feuerriegel et al. (2024); Kalota (2024)

1.4 Baseline Models

Three baseline models following the State-of-the-Art are implemented to validate and compare the proposed GAN-based models: an ARMA-based SARIMAX model (Section 1.4.1), an LSTM model (Section 1.4.2) and a multivariate linear regression model (Section 1.4.3).

1.4.1 ARMA-based Model (SARIMAX)

The following section provides an overview of auto-regressive-moving average (ARMA) models as State-of-the-Art data-driven approach for time series modelling. The primary reference for this section is the book *Methoden der Zeitreihenanalyse* from Winfried Stier Stier (2001).

The key idea of the ARMA model is that it uses an autoregressive factor (AR) to account for past values together with a moving average factor (MA) to incorporate previously made forecasting errors. Let y_t be a stationary time series, then an ARMA(p, q) model is defined as:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t, \tag{1.18}$$

where:

- p is the order of the autoregressive component, which defines the number of prior values used to predict y_t,
- q is the order of the moving average component, which accounts for past forecasting errors in the prediction,
- ϕ_i are the autoregressive coefficients,
- θ_j are the moving average coefficients,
- ϵ_t is a white noise error term, which is typically assumed to be normally distributed with zero mean and constant variance.

Autoregressive Integrated Moving Average Model (ARIMA)

A requirement for the ARMA model is stationarity, i.e., the time series should have the same mean and finite variance over all time steps. Unfortunately, most time series do not fulfil this prerequisite of stationarity, including most energy-related variables. Therefore, the ARIMA(p, d, q) model differentiates the time series to obtain stationarity. This is introduced with an integration term d to the model:

$$y_t^* = \Delta^d y_t = (1 - B)^d y_t, \tag{1.19}$$

where B is the backward shift operator, such that $By_t = y_{t-1}$. The differenced series y_t^* is then modelled using the initial ARMA(p, q) process.

Seasonal ARIMA Model (SARIMA)

In many applications, including energy forecasting, time series data has decisive seasonal patterns that should be considered for accurate modelling. The SARIMA(p, d, q)(P,D,Q)$_s$ model extends ARIMA by incorporating seasonal components:

$$\Phi_P(B^s)(1 - B^s)^D y_t = \Theta_Q(B^s)\epsilon_t, \tag{1.20}$$

where:

- P is the seasonal order of the autoregressive component,
- D is the seasonal order of the differencing component,
- Q is the seasonal order of the moving average component,
- s is the seasonal period (e.g., 24 for hourly electricity prices in a daily cycle),
- $\Phi_P(B^s)$ is the seasonal autoregressive operator,
- $\Theta_Q(B^s)$ is the seasonal moving average operator.

SARIMAX: Incorporating Exogenous Variables

The SARIMA model is further extended to SARIMAX (Seasonal ARIMA with Exogenous Variables) to incorporate external factors that influence the time series. The SARIMAX model is defined as:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{k=1}^{K} \beta_k X_{t,k} + \epsilon_t, \tag{1.21}$$

where:

- $X_{t,k}$ represents the k-th exogenous variable at time t,
- β_k are the coefficients associated with the exogenous variables.

Exogenous variables in the context of electricity price forecasting may include weather conditions, demand forecasts, and fuel prices, among others. By incorporating such external factors, SARIMAX improves predictive accuracy compared to standard SARIMA models.

1.4.2 Long Short-Term Memory Model (LSTM)

As mentioned in Section 1.3.3, recurrent neural networks encounter severe challenges with longer-term dependencies in data. Therefore, Long Short-Term Memory (LSTM) networks were proposed by Hochreiter and Schmidhuber (1997) to counter these challenges with gating mechanisms. The architecture of an LSTM cell is more complex than a standard RNN cell, as it contains several components (gates) that allow it to manage and update the cell's internal state over time. Figure 1.3 visualizes the structure and components of a single LSTM cell, the details are explained in the following paragraphs.

The LSTM unit at time step t receives three inputs: the input data x_t, the hidden state from the previous time step h_{t-1}, and the cell state from the previous time step C_{t-1}. The computations within the LSTM cell are as follows:

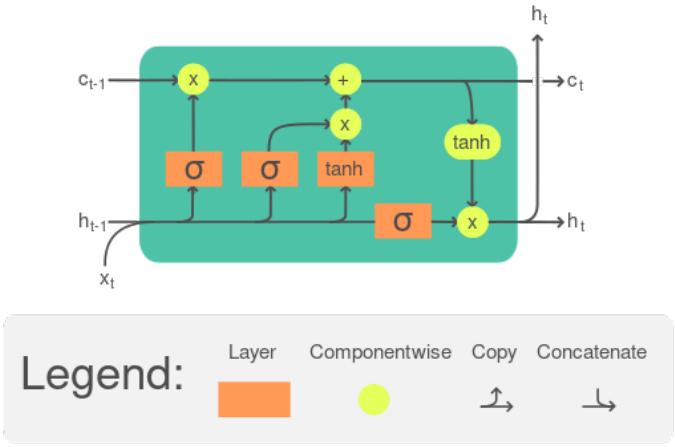

Figure 1.3: Schematic of an LSTM cell. Image obtained from Guillaume Chevalier on Wikimedia Commons under the terms of the Creative Commons Attribution License (CC BY-SA 4.0) Chevalier (2018)

1. **Forget Gate**:
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
The forget gate f_t calculates the proportion of the previous cell state that C_{t-1} should be kept. W_f is the weight matrix and b_f denotes the bias.

2. **Input Gate**:
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
The input gate i_t determines how much of the new input should be used to update the cell state. σ denotes the sigmoid and $\tanh$ the hyperbolic tangent activation function; W_i and W_C are the weight matrices for the input gate and candidate cell state; b_i and b_C the respective biases; and $\tilde{C}_t$ is the candidate cell state that will be added to the new cell state.

3. **Cell State Update**:
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$
The new cell state C_t combines what remains from the old cell state C_{t-1} after the forget gate with the value of the candidate cell state $\tilde{C}_t$ after the input gate.

4. **Hidden State**:
$$h_t = o_t \cdot \tanh(C_t)$$
The hidden state h_t is the final output of the LSTM cell at time step t.

For time-series forecasting, the LSTM is doing a step-wise processing of the input sequence. The hidden state h_t at each time step t captures information about the current input and past

inputs.

- **Input Sequence**: The input sequence $\{x_1, x_2, \ldots, x_T\}$ represents the historical data up to time T.
- **Output Sequence**: The output sequence $\{y_1, y_2, \ldots, y_T\}$ represents the predicted values, where each y_t is derived from the hidden state h_t.

The final prediction can be made using a fully connected layer that maps the hidden state h_t at the final time step (or at each time step) to the desired output y_t:

$$y_t = W_y \cdot h_t + b_y$$

Training an LSTM works very similar to training neural networks in general: with an optimization algorithm a loss function is minimized, e.g., the mean squared error (MSE), that calculates the difference between real and predicted values. The weights W_f, W_i, W_C, W_o, W_y and biases b_f, b_i, b_C, b_o, b_y are updated using backpropagation through time (BPTT), a specialised backpropagation algorithm suitable for sequence processing.

For further reading on LSTMs we refer to Hochreiter and Schmidhuber (1997).

1.4.3 Multivariate Linear Regression

A natural transition from regression models (Section 1.3.2) to a probabilistic approach is provided by the multivariate normal distribution. In particular, the conditional distribution of one set of variables given another in a multivariate normal framework closely resembles a regression model.

Given that $\mathbf{X}$ and $\mathbf{Y}$ follow a joint multivariate normal distribution, the conditional mean of $\mathbf{Y}$ given $\mathbf{X} = \mathbf{x}$ is given by:

$$\mu_{Y|X} = \mu_Y + \mathbf{\Sigma}_{YX}\mathbf{\Sigma}_{XX}^{-1}(\mathbf{x} - \mu_X)$$

This equation has a direct analogy to a linear regression model, where $\boldsymbol{\mu}_{\mathbf{Y}}$ acts as an intercept term, and $\mathbf{\Sigma}_{YX}\mathbf{\Sigma}_{XX}^{-1}$ serves as a regression coefficient matrix that linearly maps the input $\mathbf{X}$ to the output expectation $\mu_{Y|X}$. While standard regression models assume a fixed noise variance, the multivariate normal framework additionally provides a way to quantify predictive uncertainty through the conditional covariance:

$$\mathbf{\Sigma}_{Y|X} = \mathbf{\Sigma}_{YY} - \mathbf{\Sigma}_{YX}\mathbf{\Sigma}_{XX}^{-1}\mathbf{\Sigma}_{XY}.$$

Thus, this model generalizes linear regression by providing not only point predictions but also a full probabilistic characterization of the output distribution. For a more rigorous introduction see Helwig (2017).

1.5 Research Aim

While traditional forecasting models, including statistical and deep learning approaches, have shown strong performance in predicting day-ahead electricity prices, they often focus on deterministic forecasts and struggle to capture the uncertainties of electricity markets, as outlined by Lago et al. (2021). Nowotarski and Weron (2018) suggest that recent advancements in generative AI offer new possibilities for probabilistic forecasting by producing diverse future price scenarios. Given the potential of these methods, this study investigates three different GAN-based models, one of them being a State-of-the-Art conditional time series GAN from Lu et al. (2022) already tested on Australian data, and two enhanced versions based on this model. We are investigating their effectiveness in generating probabilistic day-ahead electricity price forecasts on Austrian data in two different scenarios and evaluate their impact on optimization in an electricity trading context.

1.5.1 Research Question

How effectively can generative AI models generate probabilistic forecasts for day-ahead electricity prices, and how do these forecasts compare to established deterministic and probabilistic methods in terms of different metrics and their impact on optimisation and decision making?

1.5.2 Research Approach

To address the research question, this study follows a structured methodology that encompasses data preprocessing, model implementation, training, evaluation, and comparative analysis. The data processing pipeline includes a train-test split, where two different train-test configurations are created and subsequently processed in the same manner (Section 3.2). This step is followed by peak-cutting to handle extreme values (Section 3.1.1), and standardization to ensure consistent input distributions (Section 3.1.2).

Three baseline forecasting models are implemented for comparison: a statistical approach using SARIMAX (Section 1.4.1), a deep learning-based LSTM model (Section 1.4.2), and a probabilistic baseline model based on multivariate linear regression (Section 1.4.3). Three generative AI models are propsed: a conditional time series GAN from literature (GAN_0, Section 2.1.3), a modified version with enhanced structure (GAN_1, Section 2.1.4), and a hybrid approach combining point predictions from the LSTM baseline with residual prediction using a conditional time series GAN (GAN_3, Section 2.1.5).

To optimize feature selection, a forward feature selection approach is applied using the LSTM model (Section 3.3). All models are trained with identical feature sets to ensure a fair comparison. Model performance is evaluated using standard forecasting metrics (Section 2.2) to answer the question of how effectively the models can generate probabilistic forecasts. This includes the Continuous Ranked Probability Score being able to compare deterministic and probabilistic models as generalization of the mean absolute error, the Negative Log-Likelihood to assess the similarity of the forecasted distribution to the real distribution, the Empirical Cov-

erage Probability to evaluate how well the forecasts cover the real values, and the Sharpness to assess the uncertainty estimation versus precision of the forecasts. Furthermore, a simplified optimization-based approach is implemented to assess the impact on optimization and decision making (Section 2.2.4). The results are then analyzed across both train-test splits, considering both quantitative performance metrics and their implications for decision-making in electricity trading (Chapter 4).

1.5.3 Tools & Libraries

The programming language Python (2025) was used to implement all models investigated in this thesis. The data processing relied heavily on the library pandas (2025). Statsmodels Perktold et al. (2024) and pmdarima (2025) where applied for the SARIMAX baseline, while all AI models (LSTM and GAN models) where developed with Tensorflow (2025) and Keras (2025). The multivariate linear regression baseline was built with functions of the NumPy (2025). The plotting of the data as well as the results was conducted with Plotly (2025) and Matplotlib (2025).

A Dell notebook with Intel Ultra 7 CPU and 32 GB of RAM was used for the training of the models, without additional GPU.

The large language model ChatGPT 4.0 from OpenAI (2024) was applied for some linguistic enhancements of the written text, as English is not the first language of the Author.

2 Methodology

This chapter was linguistically enhanced with a large language model.

2.1 Model

In this section, we introduce the generative model used in our study, the Generative Adversarial Network (GAN). Starting with the basic GAN we will then discuss an enhanced version, the Wasserstein GAN (WGAN). Then we will present the conditional time series GAN (CTSGAN), that builds up on the WGAN and includes exogenous conditions (features) as well as time series components. This model is then also used for our forecasts (titled as GAN_0), together with a modified version of it (GAN_1) and an ensemble model that consists of a deterministic model and the conditional time series GAN model (GAN_2).

2.1.1 Generative Adversarial Network (GAN)

Generative Adversarial Networks where first proposed by Goodfellow et al. (2014). As already briefly explained in Section 1.3.4, the key mechanism of the GAN is the competitive game between the generator G and the discriminator D. A noise vector z is sampled from a given prior distribution p_z (often Gaussian) and the generator maps it to the data space, where it tries to mimic the real distribution. On the other side, the discriminator attempts to categorize original data samples $x \sim p_{\text{data}}$ versus generated samples $\tilde{x} = G(z)$ correctly. The generator can then learn from the output of the discriminator how to produce more realistic samples that will deceive it and be categorized as real. This adversarial game is formulated as a minimax problem:

$$\min_G \max_D \; L(D,G) = \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_Z}\left[\log\left(1 - D\big(G(z)\big)\right)\right]. \tag{2.1}$$

When the discriminator is very proficient in categorizing the data samples correctly, but the generator has learned to generate fake samples that the discriminator nonetheless cannot differentiate from real data, an equilibrium is reached and the training is finished. To apply the trained GAN for data generation, the discriminator is not required any more but only the generator is needed to produce synthetic data. See Figure 2.1 for the basic architecture of the GAN.

Because of the rather high complexity of the GAN training, it faces several difficulties. One issue is vanishing gradients: when the discriminator gets too proficient and the generator can't keep up, it may provide little useful feedback to the generator. With that, the discriminator is impairing the generator's ability to learn from it and to generate better samples in the next training

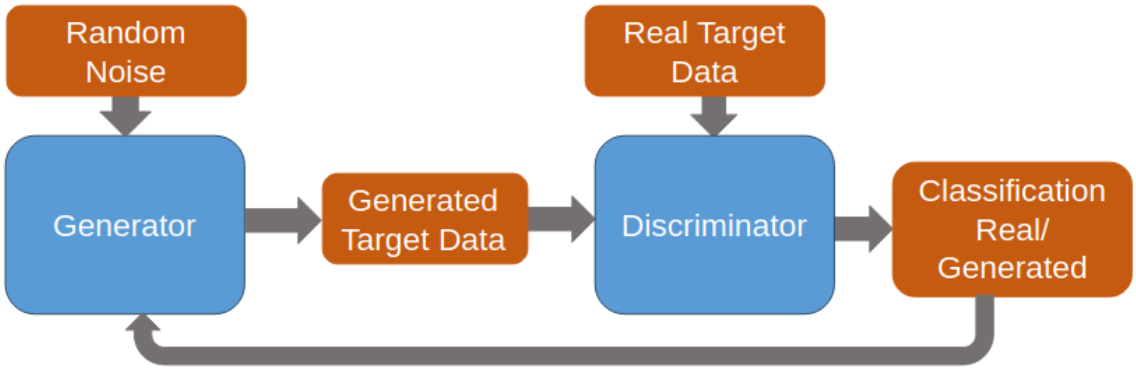

Figure 2.1: Architecture of the generative adversarial network (GAN) as suggested by Goodfellow et al. (2014).

epochs. Another well-known challenge of GAN training is *mode collapse*, where the generator does not cover the full target distribution but is focusing on only a few samples that successfully deceive the discriminator, as presented in Arjovsky and Bottou (2017). These challenges motivate alternative approaches, such as the Wasserstein GAN.

2.1.2 Wasserstein GAN

The Wasserstein GAN (WGAN), introduced by Arjovsky et al. (2017), addresses these training issues by replacing the Jensen-Shannon divergence with the Wasserstein distance. In WGAN, the discriminator is replaced by a critic function f_w that is constrained to be 1-Lipschitz. Let the generated distribution be p_g and the real distribution be p_r, then the training objective is to minimize their Wasserstein distance:

$$W(p_r, p_g) = \inf_{\gamma \in \Pi(p_r, p_g)} \mathbb{E}_{(x,\tilde{x})\sim\gamma} \left[\|x - \tilde{x}\|_2 \right], \tag{2.2}$$

where $\Pi(p_r, p_g)$ denotes the set of all joint distributions with marginals p_r and p_g. In practice, the WGAN objective is formulated as:

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x\sim p_r} \left[f_w(x) \right] - \mathbb{E}_{z\sim p_Z} \left[f_w\big(G(z)\big) \right], \tag{2.3}$$

and the generator minimizes this objective:

$$\min_{G} \mathbb{E}_{z\sim p_Z} \left[f_w\big(G(z)\big) \right]. \tag{2.4}$$

Key Architectural Differences

1. **Critic Instead of Discriminator:** In WGAN, the critic f_w outputs real-valued scores rather than probabilities. The Wasserstein distance between p_r and p_g is numerically approximated by the critic.

2. **Loss Function:** The loss functions in Eqs. (2.3) and (2.4) replace the log-likelihood losses used in standard GANs.

3. **Lipschitz Constraint:** To enforce the 1-Lipschitz condition required by the Wasserstein distance, early implementations used weight clipping. Later improvements, such as the gradient penalty proposed by Gulrajani, Ahmed, Arjovsky, Dumoulin, and Courville (2017), offer a smoother alternative (see Section 2.1.4).

Advantages

- **Improved Gradient Behavior:** By using the Wasserstein distance, WGAN provides smoother and more meaningful gradients even when the critic is well-trained. This helps to counter the vanishing gradient problem that is often a major difficulty in training the standard GAN.

- **Reduction of Mode Collapse:** The continuous and more informative loss values in WGAN help to reduce mode collapse as the generator is encouraged to cover a broader range of the data distribution.

- **Enhanced Stability and Convergence:** Overall, the use of the Wasserstein distance and the 1-Lipschitz constraint leads to more stable training dynamics and improved convergence.

Arjovsky et al. (2017)

2.1.3 Conditional Time Series GAN (GAN_0)

The conditional time series GAN, proposed by Lu et al. (2022) and labelled as GAN_0 in this thesis, extends the WGAN framework to time series forecasting by conditioning the generative process on exogenous variables (e.g., historical electricity prices, weather data, calendar information). To achieve this, GAN_0 incorporates an Embedding network $e(\cdot;\theta^{(e)})$ and a Recovery network $R(\cdot;\theta^{(R)})$, which first transform the input sequence and the conditioning variables into a latent space and then reconstruct a forecast sequence. The architecture of GAN_0 is illustrated in Figure 2.2.

For a given time step t, let the combined input at time t be denoted by $[X, y]_t$, where X represents exogenous features and y represents the target variable. The embedding network computes the latent representation:

$$h_t = e\left([X, y]_t, h_{t-1}; \theta^{(e)}\right), \tag{2.5}$$

and the recovery network reconstructs the forecast:

$$\tilde{y}_t = R\left(h_t; \theta^{(R)}\right). \tag{2.6}$$

The conditional time series GAN is trained using three loss functions:

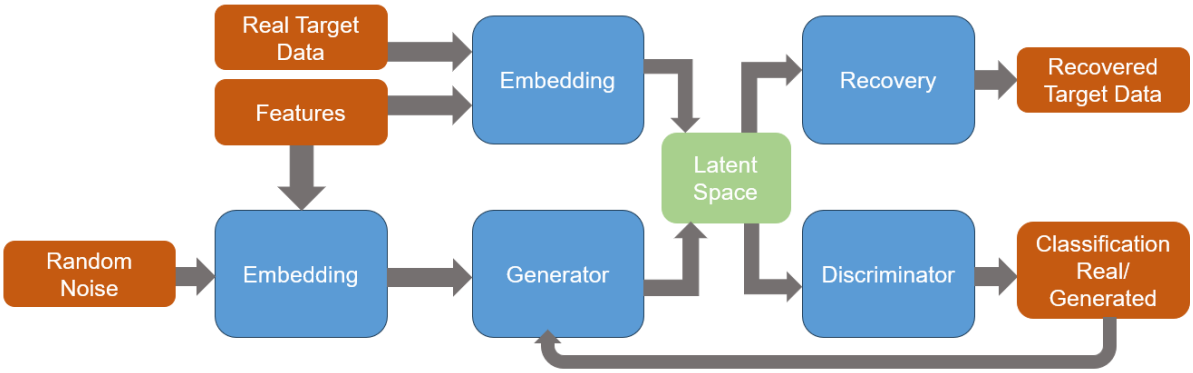

Figure 2.2: Architecture of the conditional time series GAN (GAN_0) as suggested by Lu et al. (2022).

1. **Reconstruction Loss:** This loss ensures that the embedding and recovery networks accurately reconstruct the input sequence:

$$L_R = \mathbb{E}_y \left[\sum_t \left\| R\left(e\left([X,y]_t, h_{t-1}; \theta^{(e)}\right); \theta^{(R)}\right) - [X,y]_t \right\|_2 \right]. \tag{2.7}$$

2. **Supervised Loss:** In the supervised training, the generator $G(\cdot;\theta^{(G)})$ is trained so that its output in the latent space matches that of the embedding network:

$$L_S = \mathbb{E}_{y,z} \left[\sum_t \left\| e\left([X,y]_t, h_{t-1}; \theta^{(e)}\right) - G\left(e\left([X,z]_t, h_{t-1}; \theta^{(e)}\right); \theta^{(G)}\right) \right\|_2 \right]. \tag{2.8}$$

3. **Unsupervised Loss (GAN Loss):** Finally, the unsupervised loss displays the adversarial training between the generator and a discriminator $D(\cdot;\theta^{(D)})$, which is the key idea of GAN models. The discriminator distinguishes between the latent representations computed by the embedding network e (from real data) and those generated by the generator G (from noise z):

$$\begin{aligned} L_U = \mathbb{E}_y &\left[\sum_t \log D\left(e\left([X,y]_t, h_{t-1}; \theta^{(e)}\right); \theta^{(D)}\right) \right] \\ &+ \mathbb{E}_z \left[\sum_t \log\left(1 - D\left(G\left(e\left([X,z]_t, h_{t-1}; \theta^{(e)}\right); \theta^{(G)}\right); \theta^{(D)}\right)\right) \right], \end{aligned} \tag{2.9}$$

To benefit from the stability introduced by the Wasserstein distance, the loss is simplified

to a linear form:

$$L_U = \mathbb{E}_y \left[\sum_t D\left(e\left([X,y]_t, h_{t-1}; \theta^{(e)}\right); \theta^{(D)}\right) \right] - \mathbb{E}_z \left[\sum_t \left(1 - D\left(G\left(e\left([X,z]_t, h_{t-1}; \theta^{(e)}\right); \theta^{(G)}\right); \theta^{(D)}\right)\right) \right]. \tag{2.10}$$

The training algorithm is presented in pseudo code in the appendix (algorithm 2).

Incorporating Temporal Dependencies To fully capture the sequential nature of time series data, Long Short-Term Memory (LSTM) layers are integrated into the networks. The LSTM layers allow the model to keep information across time steps, therefore capturing long-term dependencies in the provided features.

2.1.4 Modified Conditional Time Series GAN (GAN_1)

In our modified version of the conditional time series GAN, titled as GAN_1, we relax the constraint that the noise dimension must equal the output dimension. Instead of concatenating noise with the conditioning variables and feeding them through the pre-trained embedding network, we pass the conditions directly to the generator. This modification is illustrated in figure 2.3 and detailed in Algorithm 1.

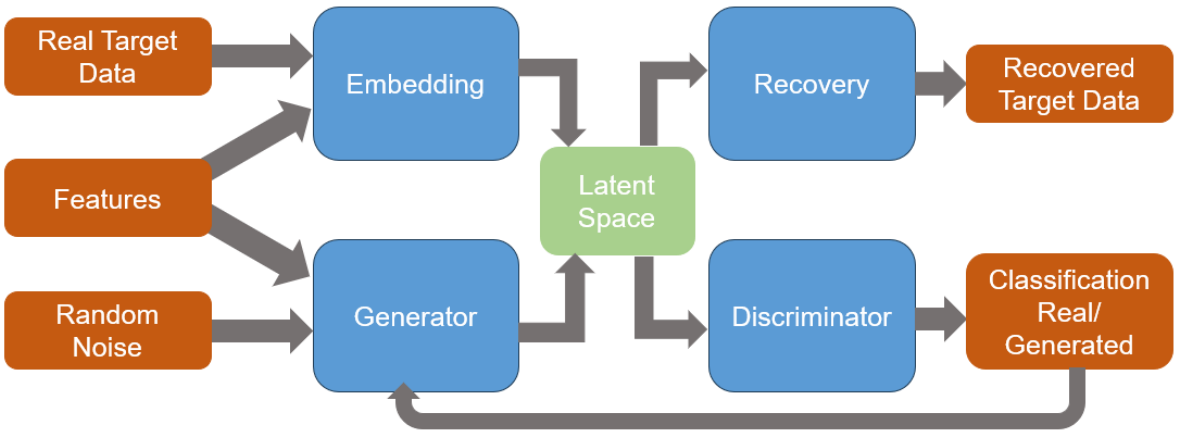

Figure 2.3: Architecture of the modified conditional time series GAN (GAN_1).

Additionally, we reduce the number of training epochs in the joint training that includes the discriminator to avoid/reduce mode collapse.

To further stabilise the training, we use a gradient penalty instead of weight clipping for the discriminator as described in the following paragraph.

Algorithm 1 Training of GAN_1 (slightly modified from Lu et al. (2022))

Input: Batch size m, number of iterations for reconstruction and supervised training methods n, number of iterations for joint training n_j, learning rate β, clipping parameter c

Output: $\theta^{(e)}, \theta^{(R)}, \theta^{(G)}, \theta^{(D)}$

Update parameter for Embedding and Recovery

for iteration = $1, 2, ..., n$ **do**

Sample batch from historical data: $\{[X, y]^{(i)}\}_{i=1}^{m} \sim P_{[X,y]}$

Update Embedding and Recovery nets using gradient descent:

$g_{\theta^{(e)},\theta^{(R)}} \leftarrow \nabla_{\theta^{(e)},\theta^{(R)}}[\frac{1}{m}\sum_{i=1}^{m}\sqrt{(R(e([X, y]^{(i)})) - [X, y]^{(i)})^2}]$

$\theta^{(e)} \leftarrow \theta^{(e)} - \beta \cdot \text{RMSProp}(\theta^{(e)}, g_{\theta^{(e)}})$

$\theta^{(R)} \leftarrow \theta^{(R)} - \beta \cdot \text{RMSProp}(\theta^{(R)}, g_{\theta^{(R)}})$

end for

Update parameter for Generator only

for iteration = $1, 2, ..., n$ **do**

Sample batch from historical data: $\{[X, y]^{(i)}\}_{i=1}^{m} \sim P_{[X,y]}$

Sample batch from random noise (Gaussian or Uniform): $\{z^{(i)}\}_{i=1}^{m} \sim P_Z$

Replace target data $y^{(i)}$ from historical data with noise: $\{[X, z]^{(i)}\}_{i=1}^{m}$

Update Generator net using gradient descent:

$g_{\theta^{(G)}} \leftarrow \nabla_{\theta^{(G)}}[\frac{1}{m}\sum_{i=1}^{m}\sqrt{(e([X, y]^{(i)}) - G([X, z]^{(i)})^2}]$

$\theta^{(G)} \leftarrow \theta^{(G)} - \beta \cdot \text{RMSProp}(\theta^{(G)}, g_{\theta^{(G)}})$

end for

Joint training

for iteration = $1, 2, ..., n_j$ **do**

Update parameter for Generator only (twice more than Discriminator)

for iteration = $1, 2$ **do**

Sample batch from historical data: $\{[X, y]^{(i)}\}_{i=1}^{m} \sim P_{[X,y]}$

Sample batch from random noise (Gaussian or Uniform): $\{z^{(i)}\}_{i=1}^{m} \sim P_Z$

Replace target data $y^{(i)}$ from historical data with noise: $\{[X, z]^{(i)}\}_{i=1}^{m}$

Update Generator net using gradient descent:

$$g_{\theta^{(G)}} \leftarrow \nabla_{\theta^{(G)}}\left[\frac{1}{m}\sum_{i=1}^{m}\left(\lambda\sqrt{(e([X, y]^{(i)}) - G([X, z]^{(i)}))^2}\right.\right.$$
$$\left.\left.-(D(e([X, y]^{(i)})) - D(G([X, z]^{(i)})))\right)\right.$$

$\theta^{(G)} \leftarrow \theta^{(G)} - \beta \cdot \text{RMSProp}(\theta^{(G)}, g_{\theta^{(G)}})$

Update Embedding net using gradient descent:

$$g_{\theta^{(e)}} \leftarrow \nabla_{\theta^{(e)}}\left[\frac{1}{m}\sum_{i=1}^{m}\left(\nu\sqrt{(e([X, y]^{(i)}) - G([X, z]^{(i)}))^2}\right.\right.$$
$$\left.\left.+\sqrt{(R(e([X, y]^{(i)})) - [X, y]^{(i)})^2}\right)\right.$$

$\theta^{(e)} \leftarrow \theta^{(e)} - \beta \cdot \text{RMSProp}(\theta^{(e)}, g_{\theta^{(e)}})$

end for

Update parameter for Discriminator only

Sample batch from historical data: $\{[X, y]^{(i)}\}_{i=1}^{m} \sim P_{[X,y]}$

Sample batch from random noise (Gaussian or Uniform): $\{z^{(i)}\}_{i=1}^{m} \sim P_Z$

Replace target data $y^{(i)}$ from historical data with noise: $\{[X, z]^{(i)}\}_{i=1}^{m}$

Update Discriminator nets using gradient descent:

$g_{\theta^{(D)}} \leftarrow \nabla_{\theta^{(D)}}[\frac{1}{m}\sum_{i=1}^{m}(D(e([X, y]_t)) - D(G(e([X, z]_t))))] + \lambda \cdot (||\nabla_{\hat{h}} D(\hat{h})||_2 - 1)^2$

end for

Gradient Penalty To achieve the 1-Lipschitz constraint on the discriminator more smoothly than weight clipping, we incorporate a gradient penalty as proposed in Gulrajani et al. (2017). For an interpolated latent representation $\hat{h}$ defined by

$$\hat{h} = \epsilon\, h_{\text{real}} + (1-\epsilon)\, h_{\text{fake}}, \quad \epsilon \sim U(0,1), \tag{2.11}$$

the gradient penalty term is given by

$$L_{\text{GP}} = \lambda\, \mathbb{E}_{\hat{h}} \left[\left(\left\| \nabla_{\hat{h}} D\left(\hat{h}; \theta^{(D)}\right) \right\|_2 - 1 \right)^2 \right], \tag{2.12}$$

where λ is a hyper-parameter controlling the strength of the penalty.

A function f is 1-Lipschitz if for all x_1, x_2,

$$|f(x_1) - f(x_2)| \leq \|x_1 - x_2\|_2. \tag{2.13}$$

This implies that the norm of the gradient $\|\nabla f(x)\|_2$ should be at most 1 almost everywhere. Instead of hard-constraining this property via weight clipping—which may overly restrict model capacity—the gradient penalty softly enforces this constraint by penalizing deviations of $\|\nabla_{\hat{h}} D(\hat{h})\|_2$ from 1 at randomly sampled points $\hat{h}$ between real and generated samples.

In Gulrajani et al. (2017), this method has been shown to improve training stability and sample quality, as it allows the discriminator to retain more capacity while still satisfying the required theoretical assumptions for convergence in Wasserstein GANs.

In the modified architecture, the supervised and unsupervised losses become:

$$L_S = \mathbb{E}_{y,z} \left[\sum_t \left\| e\left([X,y]_t, h_{t-1}; \theta^{(e)}\right) - G\left([X,y]_t, h_{t-1}; \theta^{(G)}\right) \right\|_2 \right], \tag{2.14}$$

and

$$\begin{aligned} L_U = {} & \mathbb{E}_y \left[\sum_t D\left(e\left([X,y]_t, h_{t-1}; \theta^{(e)}\right); \theta^{(D)} \right) \right] \\ & - \mathbb{E}_z \left[\sum_t \left(1 - D\left(G\left([X,z]_t, h_{t-1}; \theta^{(G)}\right); \theta^{(D)} \right) \right) \right] \\ & + \lambda\, \mathbb{E}_{\hat{h}} \left[\left(\left\| \nabla_{\hat{h}} D\left(\hat{h}; \theta^{(D)}\right) \right\|_2 - 1 \right)^2 \right]. \end{aligned} \tag{2.15}$$

2.1.5 Combination of Deterministic and Generative Models (GAN_2)

In many time series forecasting tasks, state-of-the-art deterministic models (such as LSTMs) already provide strong baseline predictions. Inspired by approaches in the literature (e.g., Y. Wang, Hug, Liu, and Zhang (2020)), we propose to combine a deterministic model with our generative model to further capture and forecast the uncertainty in the residuals. This model is labelled

GAN_2 in this work.

The overall idea is to first obtain deterministic forecasts and then to model the residuals (i.e., the forecasting errors) with a generative model. More formally, let y_t denote the true value at time t, and let $\hat{y}_t$ be the forecast obtained from a deterministic model. We compute the residuals

$$r_t = y_t - \hat{y}_t. \tag{2.16}$$

The generative model (in our case, GAN_1) is then trained on the sequence of residuals $\{r_t\}$ rather than the original target values $\{y_t\}$.

The two-step procedure is described as follows:

Step 1: Deterministic Forecasting We train an LSTM network as a deterministic model on the training dataset. The LSTM produces forecasts $\hat{y}_t$ according to

$$\hat{y}_t = f_{\text{LSTM}}(X_t; \theta_{\text{LSTM}}), \tag{2.17}$$

where X_t represents the exogenous variables (and possibly historical data) at time t, and θ_{LSTM} are the parameters of the LSTM. Once trained, the deterministic model is used to predict on the training data, and the residuals are computed using Eq. (2.16).

Step 2: Generative Modeling of Residuals The computed residuals $\{r_t\}$ capture the aspects of the data that are not explained by the deterministic model. We then train the conditional time series GAN (specifically GAN_1) to generate these residuals conditioned on the same input features. The conditional time series GAN, with its embedding, recovery, and adversarial components (as detailed in Section 2.1.4), is modified to model the distribution of residuals. In other words, the generator G is trained to produce residuals $\tilde{r}_t$, so that the complete probabilistic forecast is given by:

$$\tilde{y}_t = \hat{y}_t + \tilde{r}_t, \tag{2.18}$$

where $\tilde{r}_t$ is the synthetic residual generated by the cTSGAN.

The training of the cTSGAN on residuals follows the same principles as described in Section 2.1.4, with the loss functions being computed on r_t rather than y_t. This two-stage approach allows the generative model to focus on modeling the uncertainty and variability that remains after the deterministic forecast has been subtracted, potentially leading to improved probabilistic forecasts.

Levelling Assistance

As preliminary testing showed that all GAN models have trouble to capture the level of the forecasting values (in contrast to the curve), the value of a past time step is used to determine the level/height of the forecasted time series. This is achieved by calculating the difference between the first time step of the time series produced by the GAN model and an eligible reference value (in our case the first time step of the previous day) and subtracting this difference from the whole forecasted time series to get the relative same level as the reference value.

It should be mentioned that this modification has an impact on the evaluation with standard metrics (see Section 2.2) but not on the evaluation with optimisation approaches (see Section 2.2.4), as they are using only the curve and not the level of the forecasts.

2.2 Evaluation Metrics

Evaluating probabilistic forecasting models requires metrics that assess both the accuracy of point predictions and the predicted uncertainty. Below, we present the metrics we used to evaluate our models following definitions from Georgii (2009) and suggestions concerning their relevance for forecasts from Bjerregård, Møller, and Madsen (2021); Kumar, Marks, Mou, Feng, and Liu (2019); Leung, Leutbecher, Reich, and Shepherd (2021); St-Aubin and Agard (2022).

2.2.1 Metrics for Point Forecasts

Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) calculates the mean of the squared difference between predictions and true values:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{2.19}$$

It penalizes larger errors more than smaller ones, making it sensitive to outliers. Georgii (2009); St-Aubin and Agard (2022)

Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is an alternative to RMSE that considers the absolute differences instead of squared differences:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{2.20}$$

MAE is less sensitive to large deviations than RMSE and allows for a meaningful interpretation. Georgii (2009); St-Aubin and Agard (2022)

2.2.2 Metrics for Distributions

Negative Log-Likelihood (NLL)

The Negative Log-Likelihood (NLL) measures how well a probabilistic prediction explains the real data. The calculation of the NLL depends on the assumed distribution of the predictions,

which is often considered Gaussian with mean μ and variance σ^2. Therefore, assuming a normal distribution $\mathcal{N}(\mu, \sigma^2)$ for predictions, the NLL is given by:

$$\text{NLL} = \frac{1}{2}\sum_{i=1}^{n}\left[\log(2\pi\sigma_i^2) + \frac{(y_i - \mu_i)^2}{\sigma_i^2}\right], \tag{2.21}$$

where y_i is the actual value, μ_i is the predicted mean, and σ_i is the predicted standard deviation. A lower NLL means that the predicted distributions assign high probability density to the actual observed values, implying better calibration of both mean and variance (uncertainty). However, NLL is sensitive to miscalibrated uncertainty estimates:

- Overconfident predictions (i.e., small σ but large errors) result in very large penalties.
- Underconfident predictions (i.e., large σ but small errors) may also be penalized, though less severely.

This makes NLL a valuable metric for evaluating both accuracy and uncertainty quantification in probabilistic forecasting models. Georgii (2009); Kumar et al. (2019)

Continuous Ranked Probability Score (CRPS)

The Continuous Ranked Probability Score (CRPS) can be used for deterministic as well as probabilistic forecasts, as it generalizes the Mean Absolute Error (MAE) for distributions. The cumulative distribution function (CDF) of the predictions is measured against the real CDF:

$$\text{CRPS}(F, y) = \int_{-\infty}^{\infty}(F(x) - \mathbb{1}_{x \geq y})^2 dx, \tag{2.22}$$

where $F(x)$ is the CDF of the predicted distribution, and $\mathbb{1}_{x \geq y}$ is the indicator function. A lower CRPS indicates a better probabilistic forecast, meaning the predicted distribution places more mass near the observed value while still reflecting the inherent uncertainty in the data. CRPS rewards distributions that balance sharpness and reliability. Bjerregård et al. (2021)

Coverage Probability

The Coverage Probability does not account for the whole predicted distribution but only for the lower and upper bounds of a prediction. With that, the amount of real values that are in-between these bounds is calculated. For a probabilistic forecast with predicted lower and upper bounds $[L_i, U_i]$, the empirical coverage probability (ECP) is defined as:

$$\text{ECP} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{y_i \in [L_i, U_i]\}, \tag{2.23}$$

where y_i is the observed value, and $\mathbb{1}$ is the indicator function, which equals 1 if y_i falls within the prediction interval and 0 otherwise.

An ideal probabilistic forecast should have an empirical coverage probability close to a desired nominal confidence level. If the coverage is too low, the model underestimates uncertainty, leading to overconfident predictions. Conversely, very high coverage can be a sign of underconfident predictions, meaning the uncertainty bounds are too wide and provide limited practical value. Georgii (2009)

Sharpness

Sharpness evaluates the concentration of the predicted distribution, independent of its accuracy. Given a probabilistic forecast with variance σ_i^2, a common measure to assess sharpness is the average predictive standard deviation:

$$\text{Sharpness} = \frac{1}{n}\sum_{i=1}^{n}\sigma_i. \tag{2.24}$$

A sharper prediction has a narrower spread, meaning high confidence in the forecast and low estimated uncertainty. However, sharpness is a trade-off between precision and coverage: a model that is too sharp but miscalibrated may result in insufficient coverage probability. In contrast, a forecast with a high spread (high sharpness) has a not very useful precision.

An optimal probabilistic forecast achieves both high sharpness and proper calibration, meaning it provides precise predictions while maintaining the correct uncertainty levels. Georgii (2009)

2.2.3 Metric Selection

The choice of metric depends on the application. RMSE and MAE evaluate point forecasts following suggestions from St-Aubin and Agard (2022), while NLL, CRPS, Coverage probability and Sharpness assess probabilistic performance. Minimizing NLL ensures well-calibrated uncertainty, while CRPS provides a general accuracy measure for probabilistic forecasts, as investigated by Bjerregård et al. (2021); Leung et al. (2021). Therefore, RMSE was used for tuning deterministic models like the LSTM, while NLL was used for tuning generative Models. For the evaluation of the models (see Chapter 4) we look at CRPS (which reduces to MAE for deterministic forecasts), NLL, Coverage Probability and Sharpness.

2.2.4 Optimization Approaches

Probabilistic forecasts are widely used in the context of stochastic optimization. As a comprehensive stochastic optimization run is computationally very expensive, we define simple optimization objectives to assess the potential of different forecasting models in their practical use.

Given time series forecasts $\hat{y}$, our goal is to identify the time steps with the highest and lowest predicted values to achieve the maximum difference (e.g., price difference): $\hat{\Delta}(t_{\text{high}}, t_{\text{low}}) =$

$\hat{y}(t_{\text{high}}) - \hat{y}(t_{\text{low}})$. We then compute the actual difference

$$\Delta(t_{\text{high}}, t_{\text{low}}) = y(t_{\text{high}}) - y(t_{\text{low}}) \tag{2.25}$$

where $y(t_{\text{high}})$ and $y(t_{\text{low}})$ represent the actual values at the forecasted maximum and minimum time steps, respectively. In a market context, this difference represents the achievable revenue, assuming the forecasted asset is bought at the lowest predicted value and sold at the highest. This approach can be applied to both deterministic forecasts and probabilistic ones, with the latter using the expected value of each time step.

To exploit the probabilistic nature of generative models, we introduce the following refinements that are only applicable to probabilistic forecasts and not deterministic forecasts:

- **Pessimistic Approach:** Instead of relying solely on expected values, we calculate a lower percentile of the possible differences and use this conservative estimate to determine the optimal time steps for buying and selling. This approach represents a risk-averse strategy by prioritizing scenarios with lower downside risk.
- **Score-based Approach:** A score is assigned to each potential Δ based on a weighted trade-off between expected revenue and variance:

  $$\text{score} = \alpha\mathbb{E}(\hat{\Delta}) - (1-\alpha)\sigma^2(\hat{\Delta}). \tag{2.26}$$

  The parameter $\alpha \in [0, 1]$ controls the balance between maximizing expected revenue ($\mathbb{E}(\hat{\Delta})$) and minimizing risk ($\sigma^2(\hat{\Delta})$). A higher α favors risk-taking, while a lower α leads to a more conservative strategy.
- **Probability of positive Revenue:** Instead of solely optimizing for expected revenue, we compute the probability that $\hat{\Delta} > 0$, considering all probabilistic samples. The selected $\hat{\Delta}$ is chosen to maximize both expected revenue and the probability of achieving a positive return. This approach is highly risk-averse, as it prioritizes scenarios with a near-certain positive outcome.

These optimization strategies allow for a more comprehensive evaluation of probabilistic forecasting models, bridging the gap between forecast accuracy and real-world applications.

3 Case Study: Day-ahead electricity prices

This chapter was linguistically enhanced with a large language model.

The models are evaluated on a data set of day-ahead electricity prices of Austria from 2019 to mid 2024. The resolution is in 15 minute steps, therefore a forecasting day has 96 data points. The data was gathered from the transparancy platform ENTSO-E (2025). All gathered data points are plotted in Figure 3.1 and the hourly mean of the data, presenting the average day curve with its typical morning peak and evening peak, is presented in Figure 3.2.

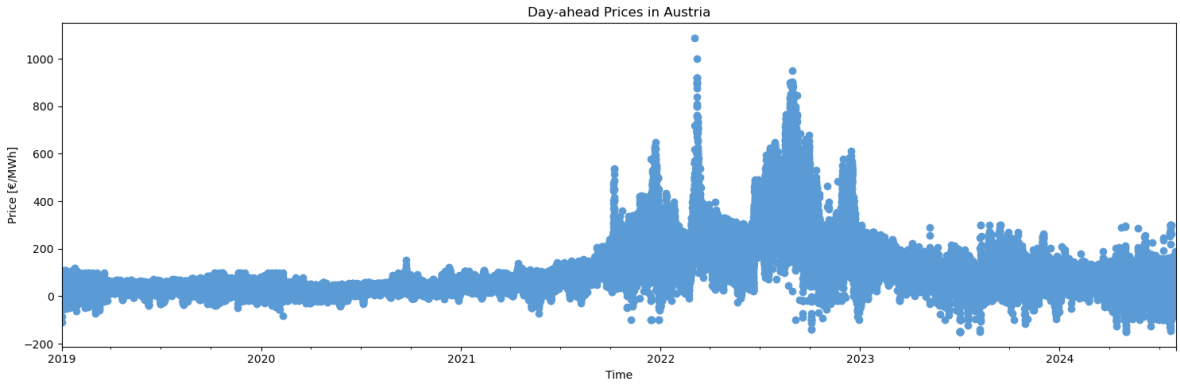

Figure 3.1: Day-ahead prices in Austria in €/MWh from 1.1.2019 to 31.7.2024 from ENTSO-E (2025)

3.1 Data Processing

3.1.1 Cutting Peaks

As proposed in Lu et al. (2022), we cut peaks in the training target data as we assume that outliers confuse the models more than provide valuable input. Therefore, peaks are defined as outliers in the top and bottom 2,5 percentiles of the data. To avoid loosing too many data points, the outliers are not removed but replaced with the top and bottom caps respectively. Figure 3.3 illustrated the unaltered distribution of day-ahead prices in a box-plot.

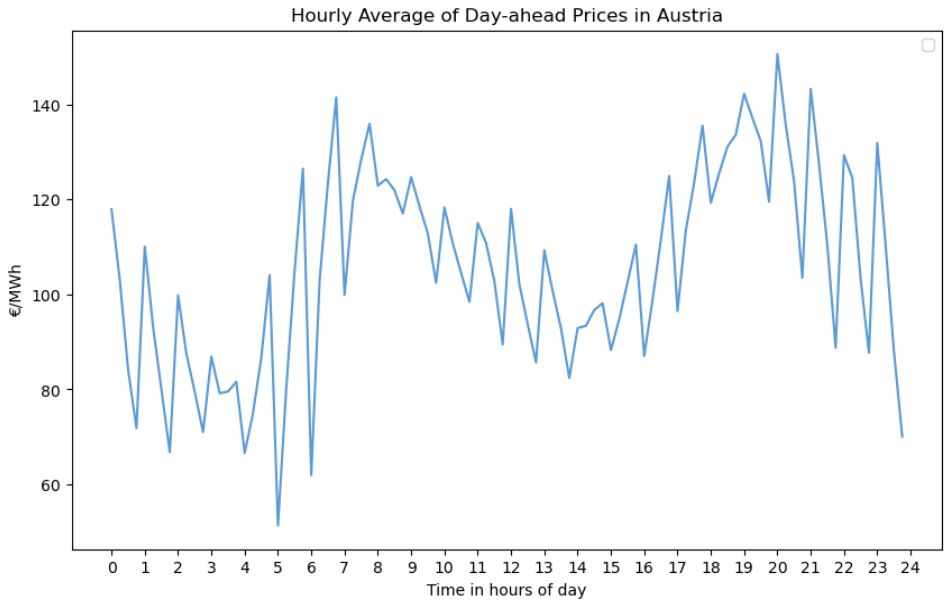

Figure 3.2: Average day curve of day-ahead prices in Austria.

3.1.2 Standardization

We assume that the target data follows a normal distribution and use normally distributed noise as input to the generator. To ensure an effective learning process, we standardise the training data rather than applying Min-Max normalisation as is often done in the literature (e.g. Lu et al. (2022)). This choice is motivated by the following considerations:

- **Aligning Input and Output Distributions**
  Since the generator receives input noise sampled from a standard normal distribution $\mathcal{N}(0, 1)$, standardizing the training data to have zero mean and unit variance helps the learning process. This transformation ensures that the model learns a mapping between similarly distributed spaces, reducing the complexity of the required transformation and improving convergence stability.

- **Enhancing Training Stability**
  Standardization helps stabilize the training of Generative Adversarial Networks by maintaining a consistent scale across features. This prevents large discrepancies in magnitude between input and output distributions, which can lead to issues such as vanishing gradients or slow convergence.

3.2 Energy Crisis and Train-Test Split

Looking at the data, a clear distortion can be seen from around September 2021 to January 2023: the energy crisis (see brown data points in Figure 3.4). The global energy crisis was the

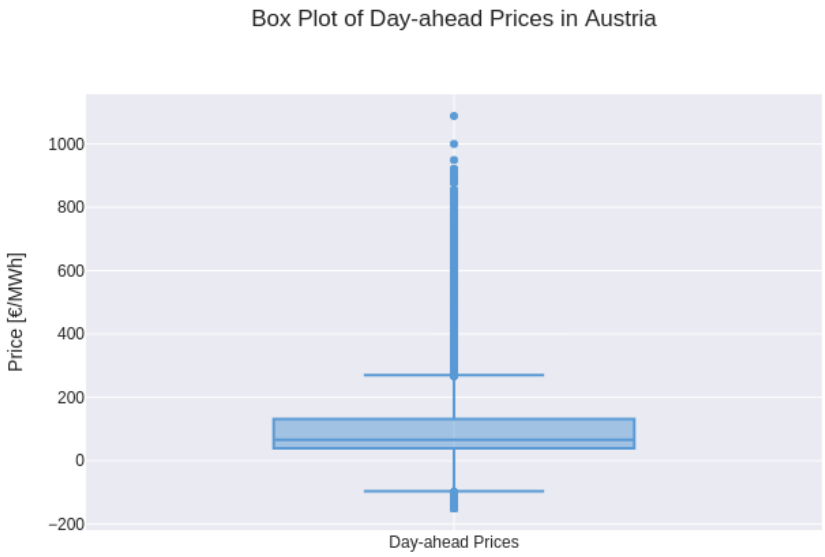

Figure 3.3: Box-plot of day-ahead prices in Austria from January 2019 to July 2024.

result of a combination of geopolitical tensions, supply chain disruptions, and overall uncertainties. A key factor has been the significant reduction in fossil fuel imports due to geopolitical conflicts, particularly the Russian-Ukrainian war, which resulted in a rather sudden decline in natural gas supplies to Europe. This caused high volatility and uncertainty in energy markets, leading to extreme price fluctuations in electricity markets, where prices reflect short-term supply and demand conditions. Moreover, policy interventions, such as price caps and emergency market regulations, have influenced market behaviour, sometimes distorting price signals. The resulting uncertainty and increased costs lead to significant challenges for grid operators, market participants, and policymakers. Förster, Harding, and Buhl (2024); Ghelasi and Ziel (2024)

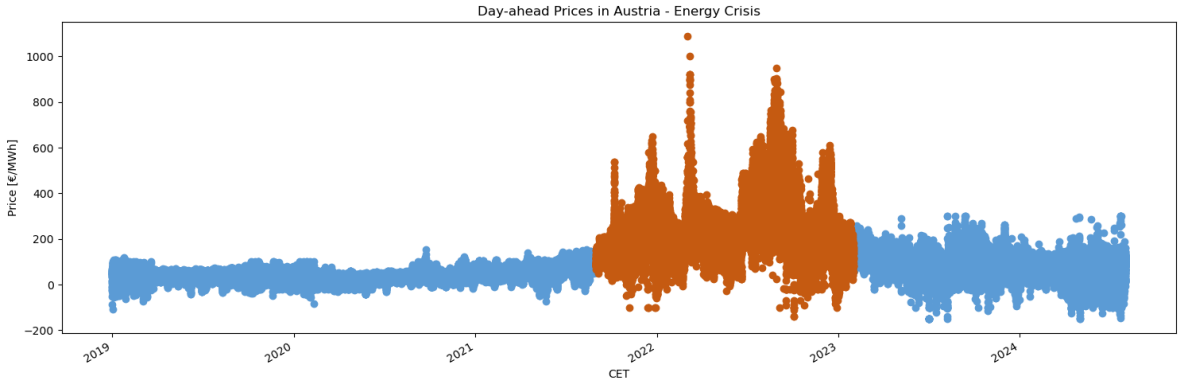

Figure 3.4: Day-ahead prices in Austria with energy crisis.

As these price distortions cannot be explained by regular features like weather or electricity demand, it suggests to remove the data for training to avoid confusing the models. However, the energy crisis also presents an interesting case study as it provides an extreme situation with a lot of uncertainty, which is inherently intriguing for probabilistic forecasting. Therefore, it was decided to investigate two different case studies: one with the energy crisis completely removed from the data and a standard train-test split, and one where the non-crisis data is used for training and the crisis-data is used for testing.

Case Study 1: Normal Mode - Dropping the Energy Crisis

In the case study *normal mode*, all data of the time period of the energy crisis (here defined as September 2021 to January 2023) is removed to avoid confusing the models with distortions that cannot be explained by the given features. The train-test-split is illustrated in Figure 3.5 with the training data in blue and the test data in dark gray. It should be noted that although not as extreme as in the energy crisis, the price variance after the energy crisis is still different and significantly higher than before the energy crisis. Therefore, the bulk of the training data, that is dated before the energy crisis, is still inherently different and not really representative for the test data.

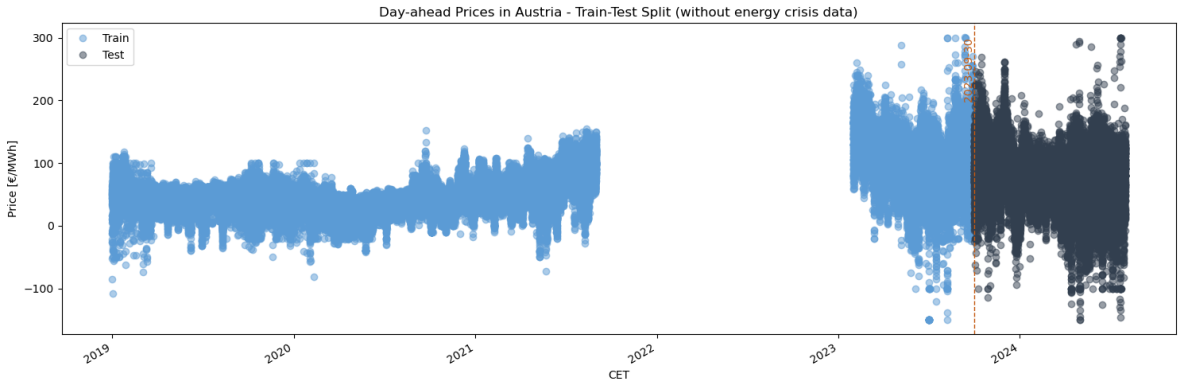

Figure 3.5: Train-test split of day-ahead prices data set in *normal mode* (without energy crisis).

Case Study 2: Crisis Mode - Using the Energy Crisis as Test Data

As mentioned above, the energy crisis presents an interesting case study to investigate the performance of forecasting models in extreme situations with lots of uncertainty. Therefore, all non-crisis data was used for the training, also the time after the energy crisis to have a sufficient amount of data points. The crisis-data was then used to test the models (see Figure 3.6 for the train-test-split).

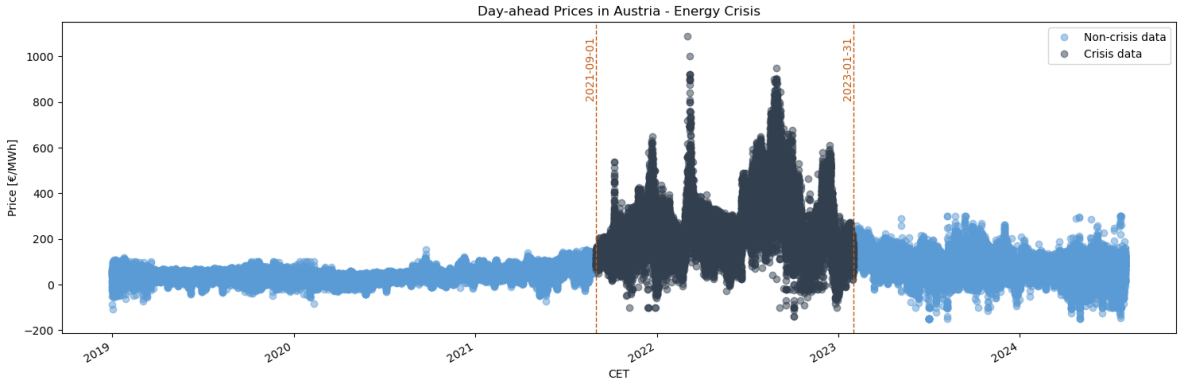

Figure 3.6: Train-test split of day-ahead prices data set in *crisis mode* (energy crisis as test data).

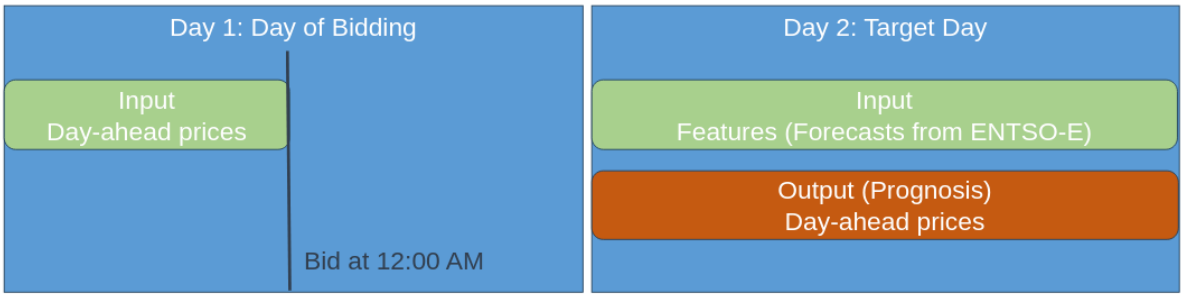

Figure 3.7: Forecasting setup: input and output for each forecasting day.

3.3 Feature Selection

Data used as features for the models were also gathered from ENTSO-E (2025). However, since the setup (see Figure 3.7) uses features of the target day, in this work only the data provided as forecasts are considered. This leaves the electricity load, wind production and solar production as well as time-related features. Additionally, temporal features like the type of day are given as input. A correlation analysis was conducted (see Figure 3.8) as well as forward features selection with the best baseline model, the LSTM, with the mean absolute error as validation loss (see Figure 3.9). The same features are then used for every model to maintain comparability.

Figure 3.8: Correlation of day-ahead prices with other features.

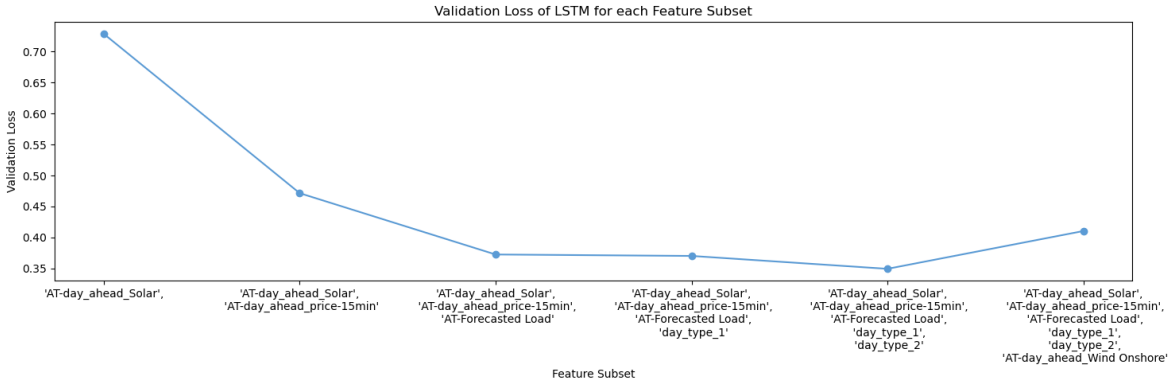

Figure 3.9: Result of forward feature selection using the LSTM baseline.

4 Results

This chapter was linguistically enhanced with a large language model.

For a visual evaluation, the forecasting days in the test data are plotted with the predictions of each of the models next to each other. The actual day-ahead prices are shown in brown, and the prediction of the model is shown in blue. The models in the top row are the baselines (SARIMAX, LSTM and multivariate linear regression) and the models in the bottom row the GAN models (GAN_0, GAN_1 and LSTM-cTSGAN ensemble model). Figure 4.1 exemplifies one forecasting day.

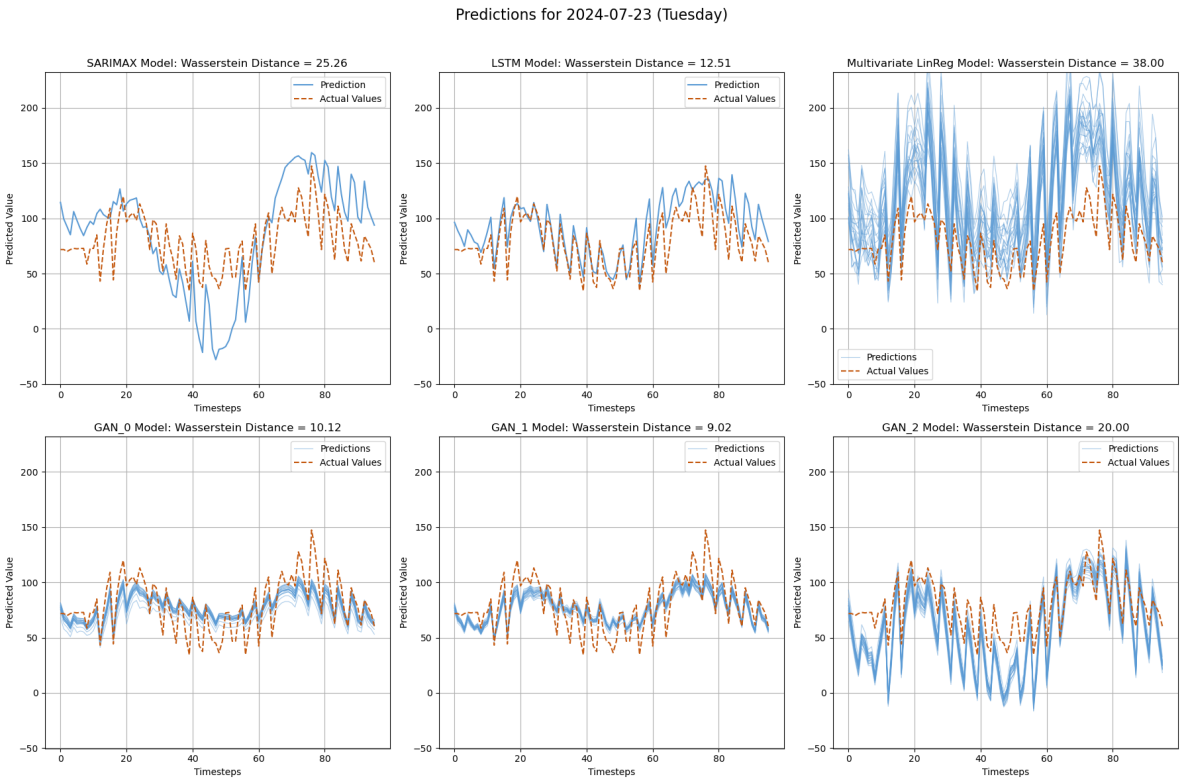

Figure 4.1: Example of one forecasting day: predictions for July 23rd 2024.

Comparing the probabilistic models, the multivariate linear regression baseline has by far the highest variance, while the GAN_0 has a very small variance. The variance of the other two GAN models is also small compared to the probabilistic baseline. This is probably a symptom of mode collapse in the GAN model training, as discussed in Section 5.1.3. However, the multivariate linear regression often fails to capture the altitude and/or the shape of the price curve,

although the shape is similar in most days with its noticeable morning and evening peak (see figure 3.2). The GAN models achieve mapping the shape of the day curve better than the multivariate linear regression.

To compare the performance of different models, we analyse four probabilistic evaluation metrics (a formal definitions of the metrics can be found in Section 2.2), averaged over all prediction days in the test datasets. For every metric the models are compared in both train-test-split modes, where the *normal mode* presents a data set without any crisis data and the *crisis mode* uses all non-crisis data for training and the crisis data for testing to investigate the performance of the models under unusual conditions. More details on the train-test split and mode definitions are provided in Section 3.2.

These four selected metrics capture different aspects of model performance. While all of them evaluate probabilistic forecasts, the Continuous Ranked Probability Score (CRPS) can also be applied to deterministic forecasts, where it reduces to the Mean Absolute Error (MAE). This allows for a direct comparison between probabilistic and point forecasting approaches.

As an overall result comparing both modes it can be seen that in all metrics, all models perform significantly worse in *crisis mode* than in *normal mode*, which is expected due to the test data being very different from the training data in *crisis mode*. However, comparing the models against each other in each of the modes shows interesting results.

4.1 Continuous Ranked Probability Score (CRPS)

The CRPS results are presented in Figure 4.2 and reveal notable patterns in both modes. In *normal mode*, where market conditions are stable, the LSTM baseline achieves the lowest CRPS, indicating the highest forecasting accuracy. The GAN-based models exhibit slightly higher scores, performing competitively but not outperforming the LSTM. The multivariate linear regression model has the second-highest CRPS, while the SARIMAX model yields the worst performance, suggesting that its linear assumptions are insufficient to capture electricity price dynamics.

In *crisis mode*, where the test data includes extreme market fluctuations from the energy crisis, the ranking of the models looks different. The three GAN-based models obtain the lowest CRPS values, which demonstrates that they are better able to capture the increased uncertainty during volatile market periods. This suggests that GANs effectively model the complex and non-stationary behavior of electricity prices under extreme conditions. In contrast, the LSTM baseline, which performed best under normal conditions, now exhibits a higher CRPS, indicating reduced robustness in handling sudden price shifts. The SARIMAX and linear regression models perform even worse, with the highest CRPS values, supporting the hypothesis that they are rather limited to adapt to crisis-induced deviations from historical patterns.

These results highlight the trade-off between model stability and adaptability. While traditional time series models may perform adequately under stable conditions, probabilistic generative models such as GANs can provide a significant advantage when forecasting electricity prices in highly uncertain environments.

The next three metrics are only calculated for the probabilistic models as they are not suitable for deterministic forecasts.

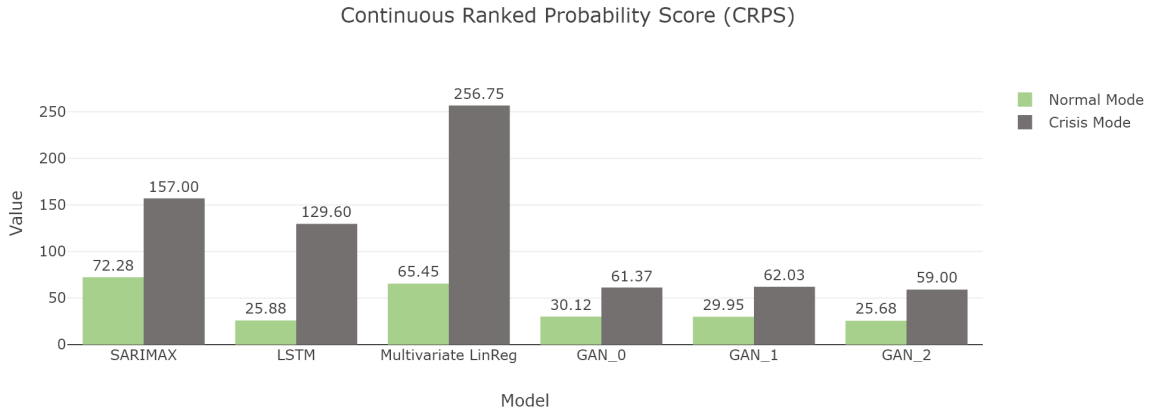

Figure 4.2: Continuous Ranked Probability Score (CRPS) of averaged over all test data. Reduces to Mean Absolute Error (MAE) with deterministic forecasts.

4.2 Negative Log-Likelihood (NLL)

The Negative Log-Likelihood (NLL) results highlight more pronounced differences between the two forecasting modes, as can be observed in Figure 4.3. Across both settings, the multivariate linear regression baseline achieves the lowest (best) NLL, meaning it produces relatively well-calibrated probabilistic forecasts. Conversely, the GAN_0 exhibits the highest (worst) NLL, suggesting that its predictive distributions frequently assign low probability density to the actual observed values and penalties for the low variance are applied.

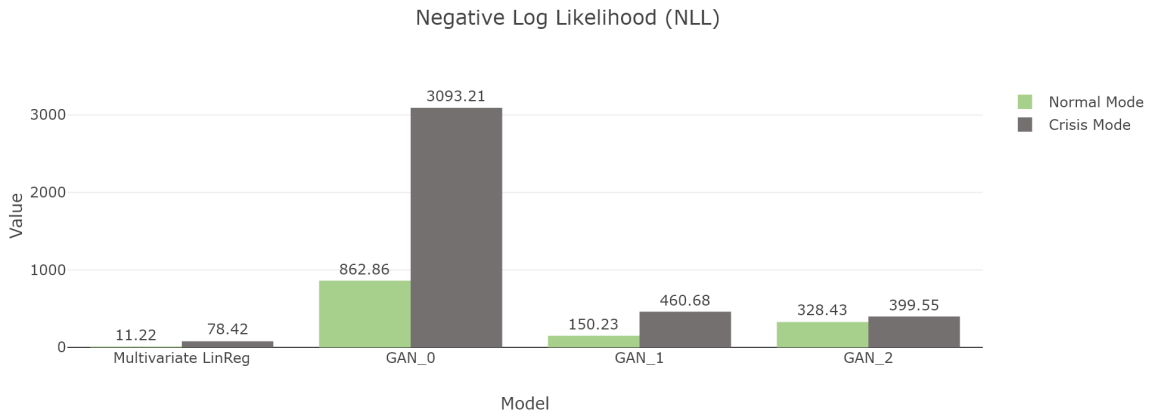

Figure 4.3: Negative Log Likelihood (NLL) averaged over all test data.

For the other GAN models, performance varies significantly between modes. In *normal mode*, the GAN_1 has a slightly higher NLL than the GAN_2 model, indicating that it may be slightly less well-calibrated in stable market conditions. However, in *crisis mode*, the GAN_1 model sig-

nificantly improves, achieving an NLL value close to that of the best-performing multivariate linear regression. Meanwhile, the GAN_2 model exhibits a substantial increase in NLL, making it the second worst-performing model after the GAN_0 in this setting.

For day-ahead electricity price forecasting, a lower NLL indicates a model's ability to assign high probability to actual prices. That the GAN models have significantly worse performance regarding the NLL as opposed to the CRPS is probably due to their low variance, which is strongly punished in the NLL calculation.

4.3 Coverage Probability

The Coverage Probability, which assesses how often observed prices fall within the predicted uncertainty intervals, reveals further contrasts between the two forecasting modes (see Figure 4.4).

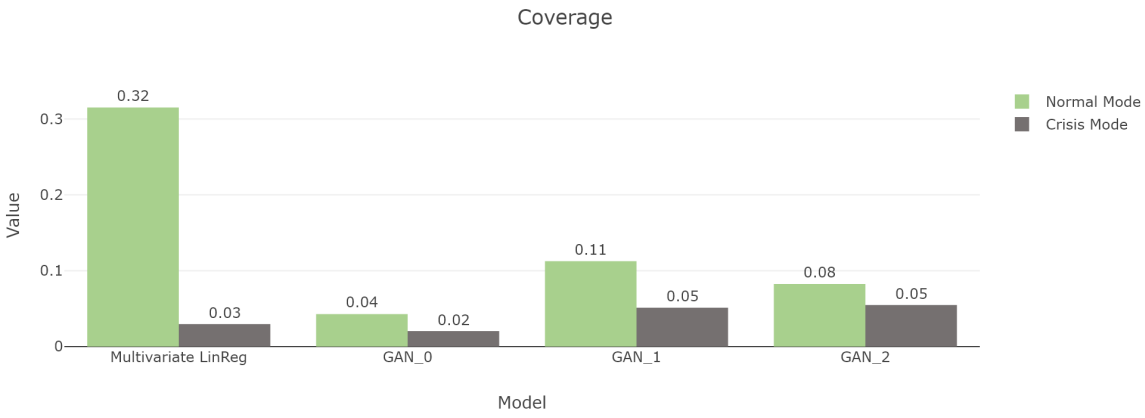

Figure 4.4: Coverage of forecasts averaged over all test data.

In *normal mode*, the multivariate linear regression baseline achieves by far the highest and best coverage probability (approximately 32%), suggesting that its prediction intervals are well-calibrated to the observed market variability. The three GAN models, however, exhibit much lower coverage probabilities (around 4-11%), with the GAN_0 performing the worst, followed by the GAN_1, and finally the GAN_2 achieving the highest (best) coverage among the GAN-based models.

In *crisis mode*, however, all models exhibit a drastic drop in coverage probability, failing to exceed 6%. The GAN_2 performs the best with nearly 6% coverage, closely followed by the GAN_1 model. The multivariate linear regression model drops to just 3%, while the GAN_0 ranks last with under 2% coverage.

In electricity price forecasting, low coverage probability suggests that prediction intervals are too narrow, underestimating market uncertainty. The sharp drop in coverage during *crisis mode* highlights the difficulty of correctly quantifying uncertainty in volatile market conditions.

While the multivariate linear regression model performs well in stable periods, its drop in crisis mode suggests that its intervals fail to adapt dynamically to changing conditions.

The GAN-models low coverage probability is likely a symptom of a rather low variance in the forecasted distributions. This could be because of mode collapse or limited representative training data (see Sections 5.1.3 and 5.1.5).

4.4 Sharpness

The Sharpness of a model's predictive distributions remains relatively stable across both modes and corresponds closely to the variance observed in visual forecast evaluations, as presented in Figure 4.5. The GAN_0 has the lowest sharpness, followed closely by GAN_1 and GAN_2. The multivariate linear regression model exhibits significantly higher sharpness, meaning its predictive distributions are wider.

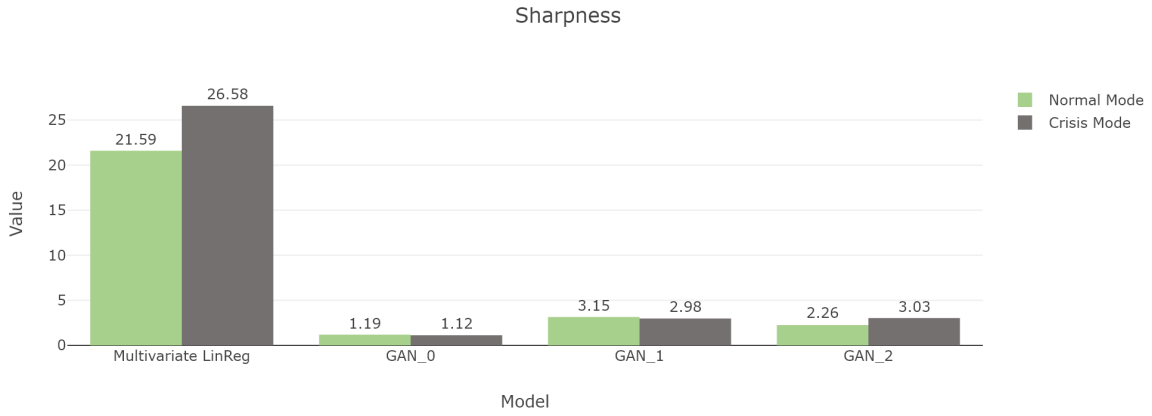

Figure 4.5: Sharpness of forecasts averaged over all test data.

For electricity price forecasting, sharpness reflects the concentration of probabilistic forecasts. A model with excessively high sharpness (low variance) may be overconfident, while a model with low sharpness (high variance) may lack useful precision. The results indicate that the GAN-based models provide relatively concentrated distributions, while the multivariate linear regression model produces more dispersed predictions, which may contribute to its higher coverage probability but lower overall accuracy.

4.5 Optimisation

As the forecasts are intended to be used for optimisations, they are evaluated against a simple optimisation goal that simulates the purchase of electricity at the time of the day where the price is the lowest and the sale of it at the time where the price is the highest. This is realised with four different strategies, three of them using distributions and therefore being only

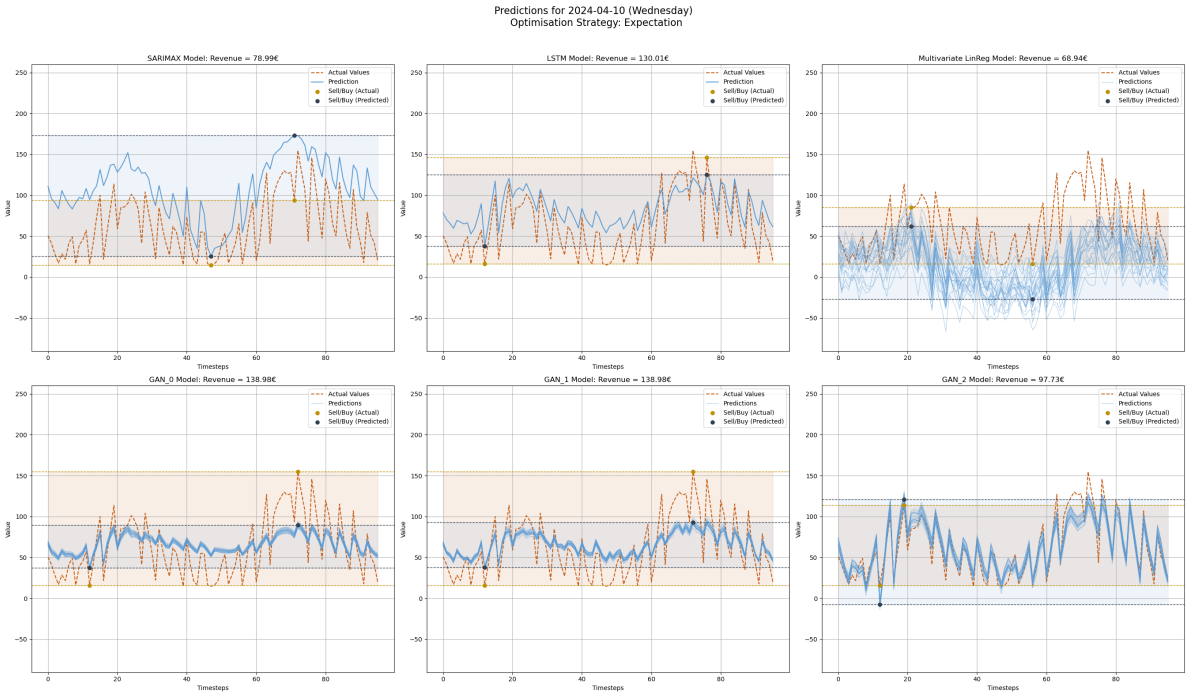

Figure 4.6: Example of one forecasting day with optimisation: the blue-grey points are the two chosen time steps with respective heights of the forecasted prices, the orange points are the same time steps but with the height of the actual prices. The coloured area between the points is the respective achieved revenue with those chosen time steps, where the blue-grey area is again the forecasted revenue and the orange area the actual revenue.

suitable for the probabilistic forecasts. For details on the optimisation see Section 2.2.4. An example illustrating the optimisation of the different models in one exemplary forecasting day is presented in Figure 4.6.

The results are presented as achieved revenues averaged over all test data as well as boxplots showing the distribution of revenues. For the sake of readability the results are shown separately for *normal mode* (see Figure 4.7 and Figure 4.8) and *crisis mode* (see Figure 4.9 and Figure 4.10).

4.5.1 Normal Mode

In *normal mode*, all models perform similarly well regarding the optimisation revenues. The GAN_2 model achieves in average the highest revenue of around 111€, followed by the LSTM baseline at 106€ and the two other GAN models at 96-99€. The SARIMAX baseline obtains on average 95€ and the multivariate linear regression baseline 85-89€ depending on the optimisation strategy (see Figure 4.7).

Differences between the strategies are rather small in the averages. Looking at the daily results, there are some differences but no clear trend suggesting that one strategy is significantly

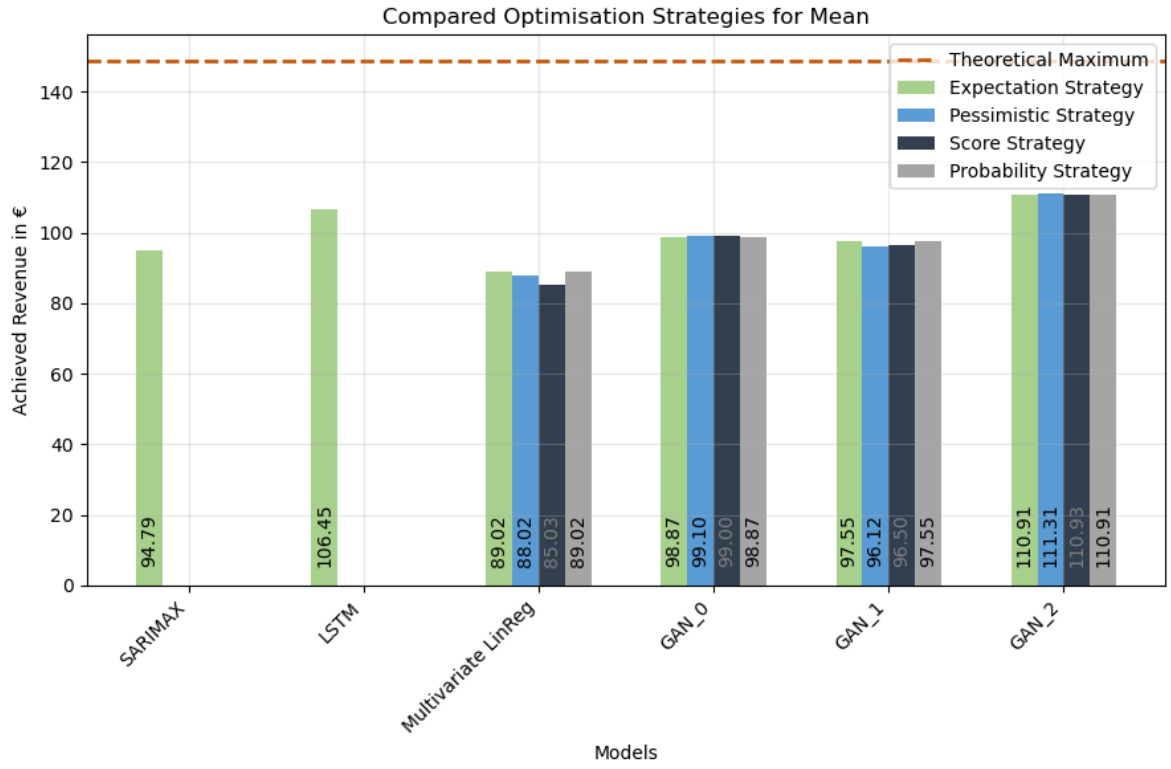

Figure 4.7: Averaged achieved revenues comparing optimisation strategies in *normal mode*.

better or worse than the others. The magnitude of differences in the strategies seems to correlate with the magnitude of variance of the forecasting distributions of the models, which is also intuitively understandable. Therefore, the multivariate linear regression baseline shows more difference between strategies than the GAN_0 with its very low variance.

Looking at the distributions of the revenues in Figure 4.8, not only the averages are of interest from an economical perspective but also how often the models incur losses (negative revenues). Here it can be seen that the two deterministic baselines SARIMAX and LSTM as well as the GAN_2 model never obtain losses, while the other two GAN models incur a few losses over all test data days. The multivariate linear regression model accounts for the highest amount of losses.

4.5.2 Crisis Mode

In *crisis mode*, the results differ again significantly, as can be seen in Figure 4.9. Here, all three GAN models achieve far higher revenues than the baselines. The GAN_0 performs the best with an average revenue of 192€, closely followed by the GAN_1 (170-171€) and the former best-performer, the GAN_2, with 157€. The baselines are all significantly worse: the LSTM achieves 125€ on average, the multivariate linear regression 31-32€ and the SARIMAX model has a negative average revenue with -52€.

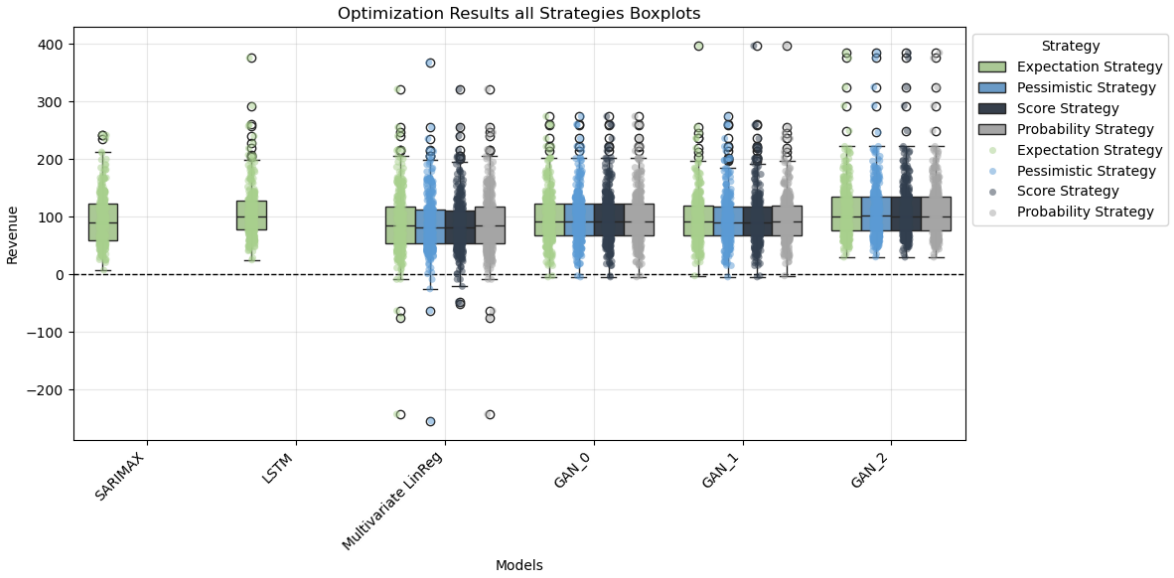

Figure 4.8: Achieved revenues in box-plots comparing optimisation strategies in *normal mode*.

The box-plots in Figure 4.10 of the revenues reveals that the GAN models make either zero losses (GAN_0 and GAN_1) or only very few (GAN_2 model), while the baselines perform significantly worse, achieving losses very often.

These results suggest that the GAN models are indeed capable of capturing uncertainties and performing well in extreme situations.

4.6 Feature Influence

As the performance of different models is very different throughout the test data days, it was tried to identify correlations between the optimisation revenues of the models with different features, including features that were not used in the model training. Again, the two modes are presented individually for the sake of readability. *Normal mode* is presented in Figure 4.11 and *crisis mode* in Figure 4.12.

4.6.1 Normal Mode

In *normal mode*, the GAN_0 (yellow) shows similar correlations as the GAN_1 (dark grey), while the GAN_2 (grey) has more in common with the LSTM baseline (orange). GAN_0 and GAN_1 show stronger performance on days with high load and high variance over the day of load as well as on workdays (day type 0). On days with high variance of electricity production from run of river generators they perform weaker, as well as on Saturdays (day type 1), Sundays (day type 2) and holidays (day type 2). The weekday correlation could be explained by the availability of training data: there is more data on work days available than on weekends and holidays,

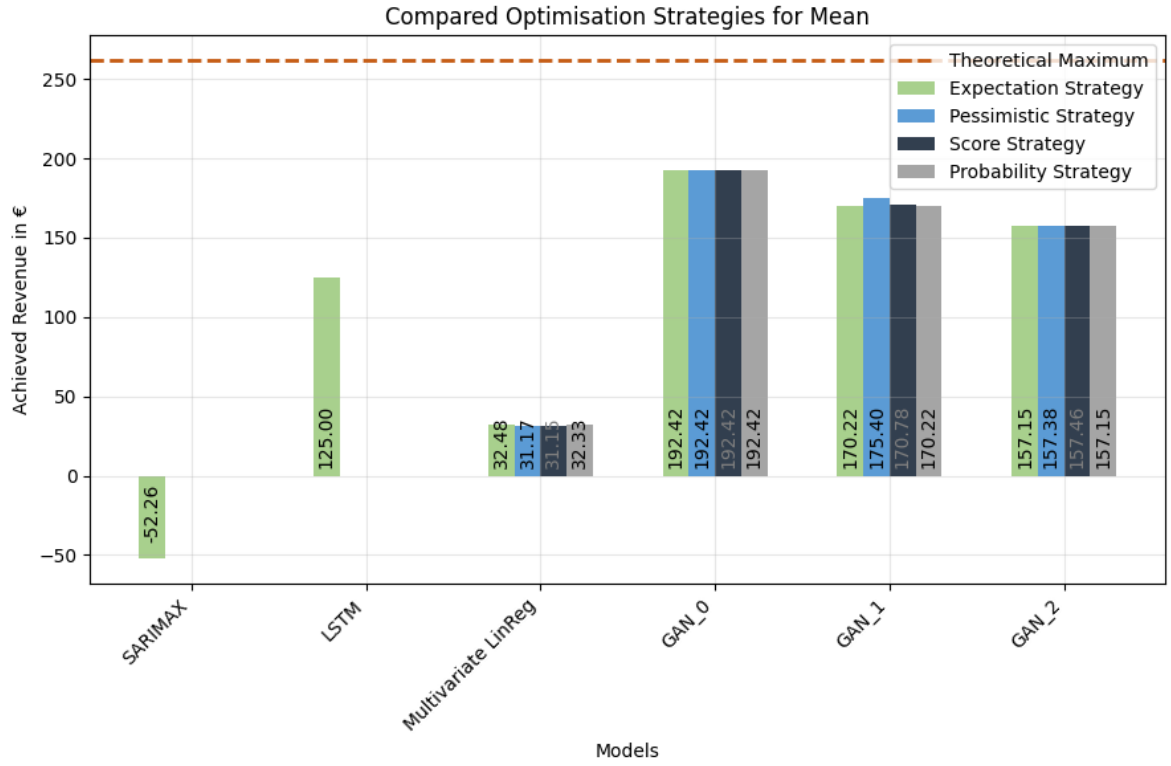

Figure 4.9: Averaged achieved revenues comparing optimisation strategies in *crisis mode*.

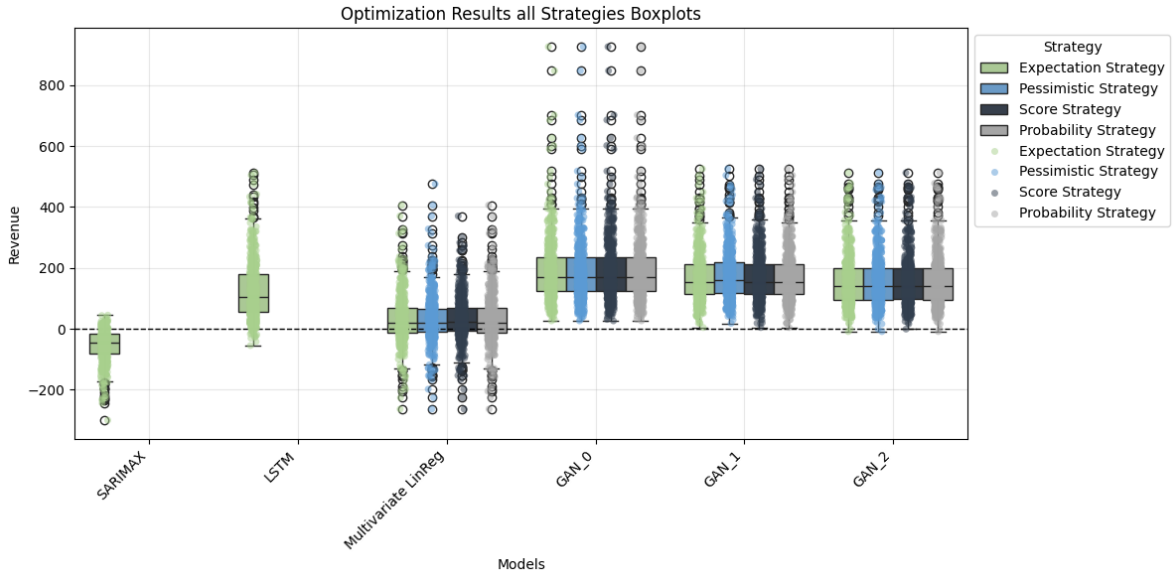

Figure 4.10: Achieved revenues in box-plots comparing optimisation strategies in *crisis mode*.

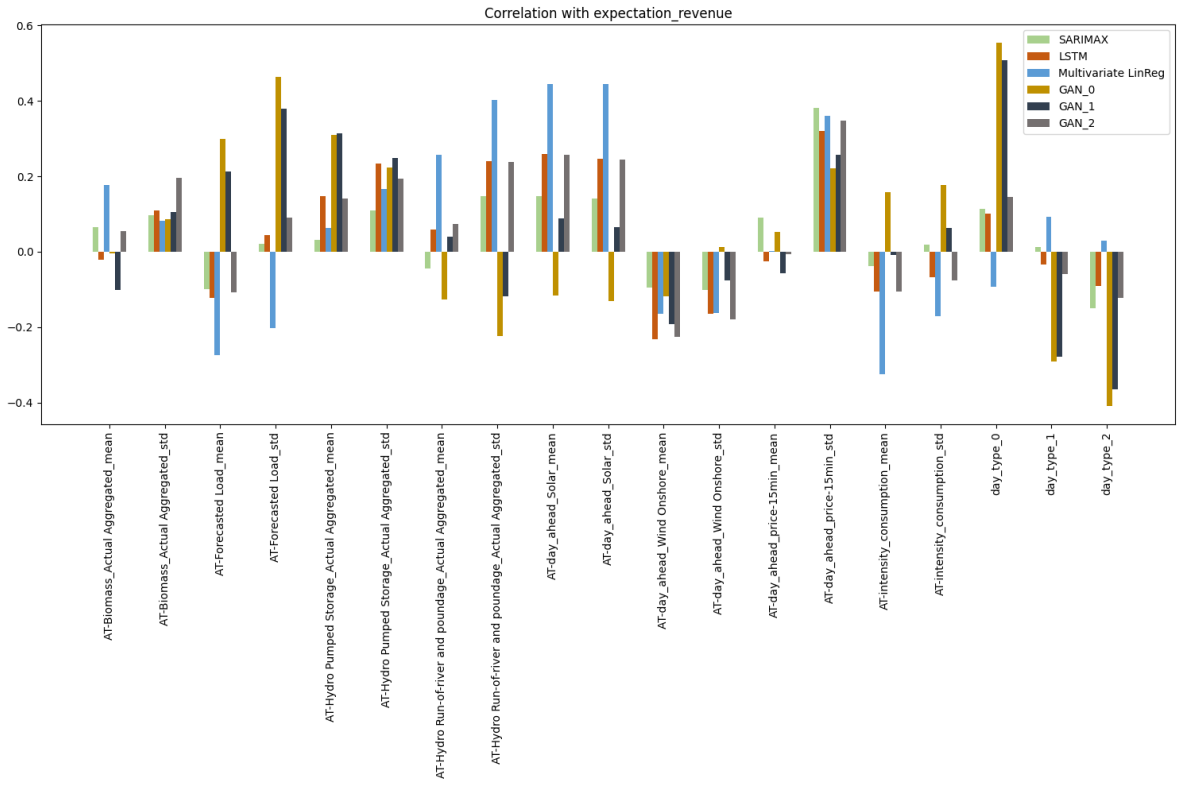

Figure 4.11: Influence of different features on optimisation revenues in *normal mode*.

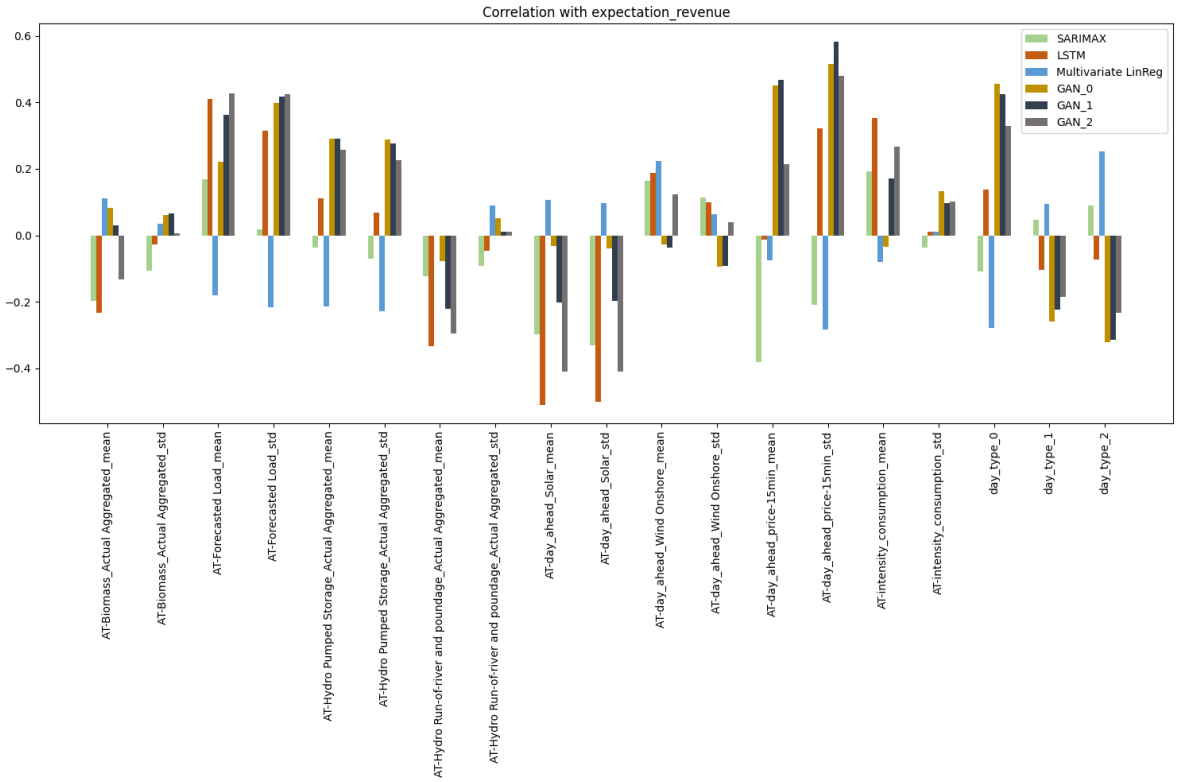

Figure 4.12: Influence of different features on optimisation revenues in *crisis mode*.

and the GAN models with their high complexity are notorious for needing a lot of training data compared to other models.

The GAN_2 model performance correlates similar to the baselines. High electricity production from solar generation as well as its variance correlates positively with the optimisation revenues, while a high CO_2-intensity correlates negatively.

4.6.2 Crisis Mode

In crisis mode, the correlations of the GAN models are all similar to each other as well as to the LSTM baseline. They perform better on days with high load and high variance in load as well as on days with high electricity production and variance of hydro pumped storage generators. In this mode, all GAN models as well as the LSTM baseline perform better on workdays and worse on Saturdays, Sundays and holidays.

5 Discussion

This chapter was linguistically enhanced with a large language model.

The results suggest that the three GAN-based models perform competitively against traditional baselines, particularly in scenarios with high uncertainty such as those observed during the energy crisis. However, several limitations and challenges remain, that could be addressed in future research.

5.1 Challenges and Limitations

5.1.1 Levelling Assistance

As described in the methodology (Section 2.1.5), a level correction mechanism was introduced to align the forecasted price levels with historical observations. This adjustment was necessary because the GAN models struggled to accurately capture the absolute level of electricity prices, despite their ability to model the shape and distribution of the forecast effectively. In contrast, baseline models did not require this explicit adjustment.

A potential reason for this limitation is that the GANs may not have fully learned the dependency between past price levels and future price trajectories. Increasing the amount of training data could enhance their ability to infer the appropriate price altitude directly from input features, particularly since the day-ahead prices from the previous day—also used for level correction—are already included in the feature set.

5.1.2 Training Instability

A well-documented challenge in GAN-based modeling is the instability of the adversarial training process Arjovsky and Bottou (2017). Unlike traditional models, which optimize a single loss function, GANs require a delicate balance between generator and discriminator training. In this study, while both the reconstruction and supervised training phases were consistently stable, the adversarial training phase frequently encountered convergence issues.

In particular, the discriminator's loss function often became non-differentiable, leading to numerical instability and the occurrence of NaN (Not a Number) values, causing premature termination of training. While adaptive learning rate strategies provided limited improvement, further research is needed to develop more robust training mechanisms, such as gradient regularization, spectral normalization, or alternative loss functions to address this issue.

5.1.3 Mode Collapse

Mode collapse, a well-known issue in GANs (rigorously investigated in Arjovsky and Bottou (2017)), happened particularly often with the GAN_0 model, where the generated distributions showed low variance and therefore failed to capture the full uncertainty inherent in electricity price forecasts. While the modifications introduced in the GAN_1 significantly improved performance, some degree of mode collapse still persisted.

This limitation is especially critical in probabilistic forecasting, where capturing uncertainty is essential for risk-aware decision-making. Future research could explore diversity-promoting regularization techniques, such as minibatch discrimination or latent space perturbation, to achieve greater variability in generated forecasts.

5.1.4 Limited Benefits of Stochastic Optimization

The presence of mode collapse directly impacted the benefits of stochastic optimization strategies. In cases where the generated distributions lacked sufficient variance, different optimization strategies produced nearly identical results, reducing their practical value in decision-making.

This issue was particularly evident in the GAN_0, whereas the GAN_1 exhibited significantly improved diversity, leading to meaningful differences in optimization outcomes. For instance, the pessimistic optimization strategy yielded a lower average revenue in normal market conditions, but a higher revenue during crisis periods compared to other strategies (see Figures 4.7 and 4.9). These findings suggest that further refinements in mode collapse mitigation could enhance the robustness of stochastic optimization approaches in electricity price forecasting.

5.1.5 Limited Representative Data

Another possible reason for the underestimated variance of the GAN models could lie in the limited representative data availability, meaning training data that represents the same market conditions as the test data. Although more than 5 years of data was available, the market conditions and therefore price curves are dramatically different at the time before, during and after the energy crisis (see Figure 3.1). While this is used in *crisis mode* to test performance in uncertain scenarios, it could explain the rather average or even low performance in *normal mode*, as the bulk of training data comes from before the energy crisis with inherently lower variance in prices as opposed to the test data that is only from after the energy crisis (see Figure 3.5). The GAN-models with their high complexity are particularly sensitive to a lack of representative training data compared to the more simple baseline models.

5.1.6 Feature Set Limitations

Due to time constraints during model development, certain key predictive features, such as weather forecasts, were not incorporated into the dataset. Given their strong correlation with renewable electricity generation (e.g., solar, wind, hydro) and electricity demand patterns, the inclusion of weather forecasts could potentially improve model accuracy and generalization.

5.1.7 Forecasting Setup

The forecasting setup, as defined in Figure 3.7, was designed to include only those features that are available in advance as forecasts—except for the previous day's day-ahead price, which was included as a key historical feature. This design choice ensures that the model only relies on information that would be realistically available at the time of prediction.

Alternative setups could incorporate features from past days or weeks, allowing the model to use longer historical trends. However, feature selection analysis indicated that the previous day's electricity price was by far the most important predictor, therefore the extension of other features with time offsets may provide only limited additional benefits.

5.2 Future Directions

Despite these challenges, the results demonstrate the potential of GAN-based models for probabilistic electricity price forecasting, particularly in volatile market conditions. Future research should focus on:

- Enhancing training stability through improved loss functions and optimization techniques.
- Developing mechanisms to reduce mode collapse and improve probabilistic calibration.
- Expanding the feature set to include weather forecasts and other relevant exogenous variables.
- Investigating other generative AI models like diffusion models and transformers for probabilistic forecasting.

By addressing these aspects, generative AI models could become a robust alternative to conventional forecasting methods, providing higher accuracy and more reliable uncertainty quantification in electricity markets.

6 Conclusion

This chapter was linguistically enhanced with a large language model.

This study investigated the effectiveness of generative AI models in producing probabilistic forecasts for day-ahead electricity prices and assessed their performance against established deterministic and probabilistic forecasting methods. Specifically, three Conditional Time Series Generative Adversarial Networks were evaluated—one from literature (GAN_0) and two enhanced versions (GAN_1 and GAN_2)—in comparison to three baseline models: SARIMAX, LSTM, and multivariate linear regression. The evaluation was conducted in two different market conditions: normal operation without data from the energy crisis (*normal mode*) and a crisis scenario with extreme price fluctuations as experienced in the energy crisis (*crisis mode*). Furthermore, the study examined how these probabilistic forecasts perform in an electricity trading context.

6.1 Effectiveness of Generative AI in Probabilistic Forecasting

The results indicate that GAN-based models provide significant advantages in probabilistic forecasting, particularly under extreme market conditions. In *normal mode*, the deterministic LSTM baseline achieved the highest accuracy, as measured by the Continuous Ranked Probability Score (CRPS), while the GAN models showed competitive but slightly lower performance. However, in *crisis mode*, the GAN-based approaches outperformed the baselines, suggesting higher robustness in uncertain scenarios. Among them, GAN_1 and GAN_2 showed the best overall performance, achieving low CRPS scores and modelling uncertainty relative effectively. In contrast, the GAN_0 model showed lower sharpness, indicating a tendency to underestimate price variability. However, it must be noted that all GAN models produce relatively low variance in their forecasts suggesting an underestimated uncertainty. This could be a reason of mode collapse (see Section 5.1.3).

While the multivariate linear regression model achieved the best Negative Log-Likelihood (NLL) scores, it displayed the highest variance and often failed to accurately capture the shape and magnitude of price fluctuations. Similarly, the SARIMAX model struggled in *crisis mode*, showing a significant drop in forecasting accuracy. These findings suggest that traditional statistical models are less suited for highly volatile market conditions compared to generative AI models.

6.2 Optimisation-Based Evaluation

When evaluating forecasting results in an economic context, all models performed similarly well in *normal mode*, with the GAN_2 achieving the highest revenue in an electricity trading simulation. The deterministic baselines (SARIMAX and LSTM) and the GAN_2 model never resulted in negative revenues, while the cTSGAN models incurred very few losses. However, in *crisis mode*, the GAN models significantly outperformed the baselines, achieving far higher revenues and demonstrating strong robustness against extreme price fluctuations. The SARIMAX model, in contrast, performed the worst, even resulting in negative average revenue.

These results highlight the potential of generative AI in providing actionable probabilistic forecasts that improve economic decision-making in volatile electricity markets.

6.3 Feature Influence and Model Behavior

Feature correlation analysis revealed that the performance of the GAN models in *normal mode* was positively influenced by high load levels and workdays, likely due to the availability of more training data. In contrast, their performance decreased on weekends and holidays, which could be due to a sensitivity to data sparsity. The GAN_2 and baseline models correlated more strongly with renewable energy production patterns and CO_2-intensity.

In *crisis mode*, all GAN models exhibited similar correlation patterns, aligning closely with the LSTM baseline.

6.4 Answering the Research Question

To answer the research question — *How effectively can generative AI models generate probabilistic forecasts for day-ahead electricity prices, and how do these forecasts compare to established methods?* — the results demonstrate that GAN-based models show potential in effective probabilistic forecasting, particularly in volatile market conditions. While traditional models perform sufficiently well in stable scenarios, they struggle under crisis conditions, where GANs provide more reliable forecasts which also enhances the economical benefits. The robust outputs of GANs allow for better risk assessment and decision-making for electricity market participants.

6.5 Future Work

Further research should focus on refining the calibration of GAN-based forecasts to improve coverage probability and reliability. Expanding the dataset to include a broader range of crisis periods as well as non-crisis periods and incorporating additional exogenous factors could enhance model generalizability. Additionally, integrating generative AI into real-world energy market operations and optimization strategies could show further insights into their practical use and potential benefits.

List of Figures

Bibliography

Arias, F., Zambrano Nuñez, M., Guerra-Adames, A., Tejedor, N., & Vargas-Lombardo, M. (2022, January). Sentiment Analysis of Public Social Media as a Tool for Health -Related Topics. *IEEE Access*, *10*, 1–1. doi: 10.1109/ACCESS.2022.3187406

Arjovsky, M., & Bottou, L. (2017, January). *Towards Principled Methods for Training Generative Adversarial Networks.* arXiv. Retrieved 2025-02-17, from http://arxiv.org/abs/1701.04862 (arXiv:1701.04862 [stat]) doi: 10.48550/arXiv.1701.04862

Arjovsky, M., Chintala, S., & Bottou, L. (2017, July). Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 214–223). PMLR. Retrieved 2024-09-10, from https://proceedings.mlr.press/v70/arjovsky17a.html (ISSN: 2640-3498)

Bjerregård, M. B., Møller, J. K., & Madsen, H. (2021, June). An introduction to multivariate probabilistic forecast evaluation. *Energy and AI*, *4*, 100058. Retrieved 2025-04-18, from https://www.sciencedirect.com/science/article/pii/S2666546821000124 doi: 10.1016/j.egyai.2021.100058

Chai, S., Li, Q., Abedin, M. Z., & Lucey, B. M. (2024, January). Forecasting electricity prices from the state-of-the-art modeling technology and the price determinant perspectives. *Research in International Business and Finance*, *67*, 102132. Retrieved 2024-10-20, from https://www.sciencedirect.com/science/article/pii/S0275531923002581 doi: 10.1016/j.ribaf.2023.102132

Chevalier, G. (2018, May). *English: Schematic of the Long-Short Term Memory cell, a component of recurrent neural networks.* Retrieved 2025-04-15, from https://commons.wikimedia.org/wiki/File:LSTM_Cell.svg

ENTSO-E. (2025). *ENTSO-E Transparency Platform.* Retrieved 2024-11-11, from https://transparency.entsoe.eu/

Exl, L., & Schaffer, S. (2025). *Angewandtes Maschinelles Lernen*. University of Vienna: Research Platform MMM & Institut für Mathematik.

Feuerriegel, S., Hartmann, J., Janiesch, C., & Zschech, P. (2024, February). Generative AI. *Business & Information Systems Engineering*, *66*(1), 111–126. Retrieved 2025-02-25, from https://doi.org/10.1007/s12599-023-00834-7 doi: 10.1007/s12599-023-00834-7

Förster, R., Harding, S., & Buhl, H. U. (2024, February). Unleashing the economic and ecological potential of energy flexibility: Attractiveness of real-time electricity tariffs in energy crises. *Energy Policy*, *185*, 113975. Retrieved 2025-03-05, from https://www.sciencedirect.com/science/article/pii/S0301421523005608 doi: 10.1016/j.enpol.2023.113975

Gault, C. (2024, April). *How predictive AI is helping smooth the path to net zero.* Retrieved 2024-06-17, from https://www.smart-energy.com/industry-sectors/

energy-grid-management/how-predictive-ai-is-helping-smooth-the-path-to-net-zero/

Georgii, H.-O. (2009, August). Stochastik: Einführung in die Wahrscheinlichkeitstheorie und Statistik. In *Stochastik*. De Gruyter. Retrieved 2025-04-18, from https://www.degruyterbrill.com/document/doi/10.1515/9783110215274/html

Ghelasi, P., & Ziel, F. (2024, August). *From day-ahead to mid and long-term horizons with econometric electricity price forecasting models.* arXiv. Retrieved 2025-03-05, from http://arxiv.org/abs/2406.00326 (arXiv:2406.00326 [stat]) doi: 10.48550/arXiv.2406.00326

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014, June). *Generative Adversarial Networks.* arXiv. Retrieved 2024-11-03, from http://arxiv.org/abs/1406.2661 (arXiv:1406.2661) doi: 10.48550/arXiv.1406.2661

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017, December). *Improved Training of Wasserstein GANs.* arXiv. Retrieved 2025-02-05, from http://arxiv.org/abs/1704.00028 (arXiv:1704.00028 [cs]) doi: 10.48550/arXiv.1704.00028

Helwig, N. E. (2017). *Mutlivariate Linear Regression.* Twin Cities. Retrieved from http://users.stat.umn.edu/~helwig/notes/mvlr-Notes.pdf

Hochreiter, S., & Schmidhuber, J. (1997, November). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. Retrieved 2024-08-31, from https://doi.org/10.1162/neco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Kalota, F. (2024, February). A Primer on Generative Artificial Intelligence. *Education Sciences*, *14*(2), 172. Retrieved 2025-02-25, from https://www.mdpi.com/2227-7102/14/2/172 (Number: 2 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/educsci14020172

Keras. (2025). *Keras: Deep Learning for humans.* Retrieved 2025-04-28, from https://keras.io/

Kingma, D. P., & Welling, M. (2022, December). *Auto-Encoding Variational Bayes.* arXiv. Retrieved 2025-04-15, from http://arxiv.org/abs/1312.6114 (arXiv:1312.6114 [stat]) doi: 10.48550/arXiv.1312.6114

Kumar, A., Marks, T. K., Mou, W., Feng, C., & Liu, X. (2019). UGLLI Face Alignment: Estimating Uncertainty with Gaussian Log-Likelihood Loss. In (pp. 0–0). Retrieved 2025-04-18, from https://openaccess.thecvf.com/content_ICCVW_2019/html/SDL-CV/Kumar_UGLLI_Face_Alignment_Estimating_Uncertainty_with_Gaussian_Log-Likelihood_Loss_ICCVW_2019_paper.html

Lago, J., Marcjasz, G., De Schutter, B., & Weron, R. (2021, July). Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy*, *293*, 116983. Retrieved 2024-10-20, from https://www.sciencedirect.com/science/article/pii/S0306261921004529 doi: 10.1016/j.apenergy.2021.116983

Leung, T. Y., Leutbecher, M., Reich, S., & Shepherd, T. G. (2021). Forecast verification: Relating deterministic and probabilistic metrics. *Quarterly Journal of the Royal Meteorological Society*, *147*(739), 3124–3134. Retrieved 2025-04-18, from https://onlinelibrary.wiley.com/doi/abs/10.1002/qj.4120 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qj.4120) doi: 10.1002/qj.4120

Lu, X., Qiu, J., Lei, G., & Zhu, J. (2022, February). Scenarios modelling for forecasting

day-ahead electricity prices: Case studies in Australia. *Applied Energy*, *308*, 118296. Retrieved 2024-06-12, from https://www.sciencedirect.com/science/article/pii/S0306261921015555 doi: 10.1016/j.apenergy.2021.118296

Maji, D., Shenoy, P., & Sitaraman, R. K. (2023, June). Multi-Day Forecasting of Electric Grid Carbon Intensity Using Machine Learning. *ACM SIGEnergy Energy Informatics Review*, *3*(2), 19–33. Retrieved 2024-06-12, from https://dl.acm.org/doi/10.1145/3607114.3607117 doi: 10.1145/3607114.3607117

Matplotlib. (2025). *Matplotlib — Visualization with Python.* Retrieved 2025-04-28, from https://matplotlib.org/

Nowotarski, J., & Weron, R. (2018, January). Recent advances in electricity price forecasting: A review of probabilistic forecasting. *Renewable and Sustainable Energy Reviews*, *81*, 1548–1568. Retrieved 2024-06-10, from https://www.sciencedirect.com/science/article/pii/S1364032117308808 doi: 10.1016/j.rser.2017.05.234

NumPy. (2025). *NumPy.* Retrieved 2025-04-28, from https://numpy.org/

OpenAI. (2024, March). *Introducing ChatGPT.* Retrieved 2025-04-28, from https://openai.com/index/chatgpt/

pandas. (2025). *pandas - Python Data Analysis Library.* Retrieved 2025-04-28, from https://pandas.pydata.org/

Perktold, J., Seabold, S., Sheppard, K., ChadFulton, Shedden, K., jbrockmendel, … Halchenko, Y. (2024, April). *statsmodels/statsmodels: Release 0.14.2.* Zenodo. Retrieved 2025-04-28, from https://zenodo.org/doi/10.5281/zenodo.593847 doi: 10.5281/ZENODO.593847

Plotly. (2025). *Plotly.* Retrieved 2025-04-28, from https://plotly.com/python/

pmdarima. (2025). *pmdarima: ARIMA estimators for Python — pmdarima 2.0.4 documentation.* Retrieved 2025-04-28, from https://alkaline-ml.com/pmdarima/

Python. (2025, April). *Welcome to Python.org.* Retrieved 2025-04-28, from https://www.python.org/

Shah, I., Iftikhar, H., & Ali, S. (2022). Modeling and Forecasting Electricity Demand and Prices: A Comparison of Alternative Approaches. *Journal of Mathematics*, *2022*(1), 3581037. Retrieved 2024-10-28, from https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/3581037 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2022/3581037) doi: 10.1155/2022/3581037

Smith, K. E., & Smith, A. O. (2020, June). *Conditional GAN for timeseries generation.* arXiv. Retrieved 2024-10-18, from http://arxiv.org/abs/2006.16477 (arXiv:2006.16477) doi: 10.48550/arXiv.2006.16477

Sourangshug, G. (2025). *Mathematical Foundations of Deep Learning*. Bangalore: Department of Civil Engineering, Indian Institute of Science Bangalore.

St-Aubin, P., & Agard, B. (2022, December). Precision and Reliability of Forecasts Performance Metrics. *Forecasting*, *4*(4), 882–903. Retrieved 2025-04-18, from https://www.mdpi.com/2571-9394/4/4/48 (Number: 4 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/forecast4040048

Stier, W. (2001). *Methoden der Zeitreihenanalyse*. Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved 2025-03-15, from http://link.springer.com/10.1007/978-3-642

-56709-4 doi: 10.1007/978-3-642-56709-4

Stryker, C., & Scapicchio, M. (2024, March). *What is Generative AI? | IBM.* Retrieved 2025-02-25, from https://www.ibm.com/think/topics/generative-ai

Tensorflow. (2025). *TensorFlow.* Retrieved 2025-04-28, from https://www.tensorflow.org/

Tschora, L., Pierre, E., Plantevit, M., & Robardet, C. (2022, May). Electricity price forecasting on the day-ahead market using machine learning. *Applied Energy*, *313*, 118752. Retrieved 2024-07-18, from https://www.sciencedirect.com/science/article/pii/S0306261922002057 doi: 10.1016/j.apenergy.2022.118752

TseKiChun. (2021, November). *English: Neural network explain.* Retrieved 2025-04-11, from https://commons.wikimedia.org/wiki/File:Neural_network_explain.png

Wang, X., Zhao, Q., & Tong, L. (2024, September). *Probabilistic Forecasting of Real-Time Electricity Market Signals via Interpretable Generative AI.* arXiv. Retrieved 2024-10-20, from http://arxiv.org/abs/2403.05743 (arXiv:2403.05743) doi: 10.48550/arXiv.2403.05743

Wang, Y., Hug, G., Liu, Z., & Zhang, N. (2020, December). Modeling load forecast uncertainty using generative adversarial networks. *Electric Power Systems Research*, *189*, 106732. Retrieved 2024-08-20, from https://www.sciencedirect.com/science/article/pii/S0378779620305356 doi: 10.1016/j.epsr.2020.106732

Appendix

Algorithm 2 Training of GAN_0 (from Lu et al. (2022))

Input: Batch size m, number of iterations for each training method n, learning rate β, clipping parameter c

Output: $\theta^{(e)}, \theta^{(R)}, \theta^{(G)}, \theta^{(D)}$

Update parameter for Embedding and Recovery

for iteration = $1, 2, ..., n$ **do**

Sample batch from historical data: $\{[X, y]^{(i)}\}_{i=1}^{m} \sim P_{[X,y]}$

Update Embedding and Recovery nets using gradient descent:

$g_{\theta^{(e)},\theta^{(R)}} \leftarrow \nabla_{\theta^{(e)},\theta^{(R)}}[\frac{1}{m}\sum_{i=1}^{m}\sqrt{(R(e([X, y]^{(i)})) - [X, y]^{(i)})^2}]$

$\theta^{(e)} \leftarrow \theta^{(e)} - \beta \cdot \text{RMSProp}(\theta^{(e)}, g_{\theta^{(e)}})$

$\theta^{(R)} \leftarrow \theta^{(R)} - \beta \cdot \text{RMSProp}(\theta^{(R)}, g_{\theta^{(R)}})$

end for

Update parameter for Generator only

for iteration = $1, 2, ..., n$ **do**

Sample batch from historical data: $\{[X, y]^{(i)}\}_{i=1}^{m} \sim P_{[X,y]}$

Sample batch from random noise (Gaussian or Uniform): $\{z^{(i)}\}_{i=1}^{m} \sim P_Z$

Replace target data $y^{(i)}$ from historical data with noise: $\{[X, z]^{(i)}\}_{i=1}^{m}$

Update Generator net using gradient descent:

$g_{\theta^{(G)}} \leftarrow \nabla_{\theta^{(G)}}[\frac{1}{m}\sum_{i=1}^{m}\sqrt{(e([X, y]^{(i)}) - G(e([X, z]^{(i)}))^2}]$

$\theta^{(G)} \leftarrow \theta^{(G)} - \beta \cdot \text{RMSProp}(\theta^{(G)}, g_{\theta^{(G)}})$

end for

Joint training

for iteration = $1, 2, ..., n$ **do**

Update parameter for Generator only (twice more than Discriminator)

for iteration = $1, 2$ **do**

Sample batch from historical data: $\{[X, y]^{(i)}\}_{i=1}^{m} \sim P_{[X,y]}$

Sample batch from random noise (Gaussian or Uniform): $\{z^{(i)}\}_{i=1}^{m} \sim P_Z$

Replace target data $y^{(i)}$ from historical data with noise: $\{[X, z]^{(i)}\}_{i=1}^{m}$

Update Generator net using gradient descent:

$$g_{\theta^{(G)}} \leftarrow \nabla_{\theta^{(G)}}\left[\frac{1}{m}\sum_{i=1}^{m}\left(\lambda\sqrt{(e([X, y]^{(i)}) - G(e([X, z]^{(i)})))^2}\right.\right.$$
$$\left.\left.-(D(e([X, y]^{(i)})) - D(G(e([X, z]^{(i)}))))\right)\right]$$

$\theta^{(G)} \leftarrow \theta^{(G)} - \beta \cdot \text{RMSProp}(\theta^{(G)}, g_{\theta^{(G)}})$

Update Embedding net using gradient descent:

$$g_{\theta^{(e)}} \leftarrow \nabla_{\theta^{(e)}}\left[\frac{1}{m}\sum_{i=1}^{m}\left(\nu\sqrt{(e([X, y]^{(i)}) - G(e([X, z]^{(i)})))^2}\right.\right.$$
$$\left.\left.+\sqrt{(R(e([X, y]^{(i)})) - [X, y]^{(i)})^2}\right)\right]$$

end for

Update parameter for Discriminator only

Sample batch from historical data: $\{[X, y]^{(i)}\}_{i=1}^{m} \sim P_{[X,y]}$

Sample batch from random noise (Gaussian or Uniform): $\{z^{(i)}\}_{i=1}^{m} \sim P_Z$

Replace target data $y^{(i)}$ from historical data with noise: $\{[X, z]^{(i)}\}_{i=1}^{m}$

Update Discriminator nets using gradient descent:

$g_{\theta^{(D)}} \leftarrow \nabla_{\theta^{(D)}}[\frac{1}{m}\sum_{i=1}^{m}(D(e([X, y]_t, h_{t-1})) - D(G(e([X, z]_t, h_{t-1}))))]$

$g_{\theta^{(D)}} \leftarrow \text{clip}(\theta^{(D)}, -c, c)$

end for