Universiteit Leiden

Bachelor Computer Science & Datascience and Artificial Intelligence

Optimizing the Accuracy–Calibration Trade-off with Multi-Objective Genetic Algorithms in Transformer-Based Forecasting for the Hungarian Energy Balancing Market

Bence Valint

Thesis supervisor:
Hao Wang & Robin van der Laag

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

08/06/2026

Abstract

For successful probabilistic forecasting of the the Hungarian electricity balancing markets, models must balance accuracy and calibration. However, traditional hyperparameter optimization typically targets a single loss function defined by a single scalar objective. This paper applies a well-established evolutionary algorithm, Non-dominated Sorting Genetic Algorithm II to find a set of hyperparameters that jointly optimize model accuracy, mean absolute scaled error, and calibration error. The Temporal Fusion Transformer and PatchTST were tuned to forecast the Hungarian automatic Frequency Restoration Reserve upward activation price due to the volatility and seasonality of the target. At a 15-minute horizon both transformer models managed to outperform the baseline seasonal-naive and LightGBM models. Relative to LightGBM, the transformer models improved forecasting accuracy by $\approx 18-20\%$ at the 15 minute horizon while at longer horizons the LightGBM model stayed competitive among the complex transformer models. The evolutionary tuning resulted in a set of four configurations for the temporal fusion transformer model and a set of three configuration for the PatchTST transformer. Calibration error, measured as the empirical coverage for the 80% coverage interval, was reduced from 0.087 to 0.018 on the validation set for TFT and from 0.095 to 0.032 on the validation set for the PatchTST model. However, despite the strong validation calibration scores, these gains did not transfer over to the test set due to poor generalization. This failure points towards the 80% coverage error, calibration, being a non-proper objective. The evolutionary tuning overfit the validation set producing Pareto fronts that performed with a very low calibration error on the validation but a high calibration error over the test set. This paper suggests replacing the objective with a proper objective such as the Continuous Ranked Probability Score for future work.

1 Introduction

1.1 Motivation

Electricity markets are increasingly characterized by high volatility, strong non-linear dependencies and rapidly changing temporal dynamics driven by weather conditions, consumer behavior, and renewable energy developments [Mosquera-López and Nursimulu, 2019]. In such rapidly changing environments, accurate forecasting is essential for market participants, particularly in balancing mechanisms, where prediction errors can lead to great financial and operational risk.

Transformer-based forecasting models have gained a reputation for demonstrating strong performance in capturing long-range dependencies in time series data [Thundiyil et al., 2023]. However, their behavior is highly sensitive to architectural and training hyperparameters, and improvements in predictive accuracy often result in costs involving a reduction of calibration under distribution shifts or noisy conditions [Amit et al., 2025], [Balanya et al., 2024]. This creates a fundamental trade-off between calibration and accuracy that cannot be addressed with standard single-objective optimization approaches.

In the context of the Hungarian Energy Power Exchange (HUPX) and the transmission system operator (TSO), MAVIR, where operational decisions must remain stable under uncertain and fluctuating conditions, calibration is as critical as raw predictive performance. This motivates the need for systematic optimization strategies that explicitly account for multiple competing objectives rather than treating calibration as an implicit byproduct of accuracy.

To address this challenge, this work explored multi-objective optimization using genetic algorithms for transformer-based forecasting models. By balancing accuracy and calibration, the proposed approach aims to identify Pareto-optimal configurations that provide principled trade-offs between the

two main evaluation criteria; accuracy and calibration, thus improving reliability and performance in real-world energy market applications.

1.2 Problem Statement

HUPX and other energy markets face challenges in balancing sudden spikes in supply or demand that offset the grid load from equilibrium, as shown by a study conducted by Sebastian Schwenen and Karsten Neuhoff [Schwenen and Neuhoff, 2024]. The grid is defined as the underlying system that transfers electricity between consumers and suppliers. An imbalance refers to an inequality in demand and supply, where the market fails to reach equilibrium. A common example would be a large cloud that reduces solar-panel power generation, leading to a huge energy deficit. When such an imbalance occurs, the TSO must purchase additional electricity, balancing energy, from providers who can increase power output or consumption. The cost for this service is is the activation price. The costs differ based on the direction of balancing, upward activation refers to more energy input into the grid, while downward refers to consumers using up energy from the grid.
To help the TSO balance the grid, companies employ large scale batteries and scalable gas engines to be able to consume or produce more electricity depending on the needs of the TSO and in return get paid [Firoozi et al., 2020]. However, these services are expensive, and unforeseen imbalance events raise the prices of electricity for every electricity consumer.
The TSO's balancing can be split into two main groups: the automatic frequency restoration reserve (aFRR) and the manual frequency restoration reserve (mFRR) [Firoozi et al., 2020]. Both aFRR and mFRR are reserve energy products that the TSO reserves for balancing purposes. Manual reserves are much cheaper, but require early notice, with automatic reserves responding to imbalance events within seconds [European Commission, 2017]. Through early forecasting of imbalance events, manual reserves can be notified early to prepare to balance the market, reducing energy costs, and benefiting society as a whole.
The HUPX is currently divided into 15 minute intervals according to the Platform for the International Coordination of Automated Frequency Restoration and Stable System Operation (PICASSO) regulations [European Commission, 2017]. To address the problem of forecasting, the transformer model will predict the shortage costs, aFRR upwards activation costs, in the energy market from multi hour horizons. The multi-horizons are made up of 15 minute intervals starting from 15 to 240 minutes as shown in table 1. The multi-hour horizon was chosen because it allows almost all manual reserves to be able to respond depending on their minimum notice time, thus minimizing costs related to solving the upcoming shortage of energy. Transformer models will output a probability distribution of the aFRR upward activation price rather than a single point estimate, since the balancing costs of shortages can have a great random error due to availability limitations, as shown in a master thesis conducted by Finn Uijtewaal in Delft University [Uijtewaal, 2025].

Table 1: The table shows the sixteen forecast horizons. At every 15-minute forecast origin each model produces a path of 16 predictions, one per horizon step h, with lead time $15h$ minutes —- spanning 15 minutes to 4 hours ahead. Results and metrics in Chapter 7 are reported at the key horizons $h \in \{1, 4, 8, 12, 16\}$.

Step h	**Lead time**	**Step h**	**Lead time**
1	15 min	9	135 min
2	30 min	10	150 min
3	45 min	11	165 min
4	60 min (1 h)	12	180 min (3 h)
5	75 min	13	195 min
6	90 min	14	210 min
7	105 min	15	225 min
8	120 min (2 h)	16	240 min (4 h)

1.3 Research Questions

The increasing complexity and volatility of modern electricity markets require forecasting models that are not only accurate but also properly calibrated under rapidly changing environments and market conditions. Optimizing purely for predictive accuracy may perform poorly under noisy or shifting market conditions.

To address these challenges, this study investigates the application of multi-objective genetic optimization for transformer-based forecasting in the Hungarian energy balancing market. The research is guided by the following primary research question:

RQ: How effectively can multi-objective genetic algorithms optimize the trade-off between predictive accuracy and model calibration in transformer-based forecasting for the Hungarian energy balancing market?

To support the main research question, two guiding questions will support the research:

GQ1: How effective is transformer-based forecasting compared to baseline models in forecasting the Hungarian balancing market shortage prices?

GQ2: Which state-of-the-art transformer architectures are most applicable to forecast Hungarian balancing market shortage prices?

1.4 Contributions

This study makes multiple contributions to the fields of energy market forecasting and evolutionary hyperparameter tuning. First, the research presents one of the first systematic evaluations of state-of-the-art transformer-based forecasting architectures for shortage price prediction in the Hungarian energy balancing market, establishing a strong empirical benchmark against conventional baseline approaches. Moreover, the paper conducts a comparative analysis of multiple transformer architectures to identify which modeling strategies are most effective for capturing the temporal dynamics, volatility, and non-linear behavior of balancing market prices. The self-made multi-objective hyperparameter optimization framework based on genetic algorithms directly optimizes predictive accuracy and calibration, rather than treating calibration as an implicit secondary outcome. Furthermore, this paper provides insight and analysis of Pareto-optimal model configurations, depicting empirical information about trade-offs between forecast accuracy and calibration under real market uncertainty. Finally, the research presents a reproducible experimental framework, bundled with

model configurations, evaluation protocols, and implementation details, to support future research in transformer-based energy forecasting and multi-objective model optimization.

1.5 Thesis structure

The remainder of this thesis is organized as follows. Chapter 2 explores the necessary background information. This includes the operation of the aFRR balancing market, the fundamentals of multi-horizon and probabilistic time series forecasting, the evolution of transformer architectures for forecasting, and the principles of multi-objective optimization, with special attention to the NSGA-II algorithm and the existing literature on electricity price forecasting. Subsequently, Chapter 3 describes the available data, the ENTSO-E and Open-Meteo sources, the construction of the canonical 15-minute dataset, the handling of daylight-saving transitions and forecast-origin leakage, and the walk-forward cross-validation design. The next chapter presents the chosen forecasting models — the seasonal-naive and LightGBM baselines, and the two transformer architectures, the Temporal Fusion Transformer and PatchTST — together with the quantile loss used to produce probabilistic forecasts. The reasoning behind choosing the baseline and transformer models will further provide insight into the structure of such models and their properties that make them fit for prediction for the Hungarian balancing market. Chapter 5, the methodological core of the thesis, develops the multi-objective evolutionary approach to hyperparameter tuning. This chapter describes the accuracy–calibration trade-off, presents the NSGA-II algorithm in full detail, describes its from-scratch implementation, and applies it to both transformer models. Chapter 6 specifies the experimental setup, including the evaluation metrics and the statistical testing procedure. Chapter 7 reports the results and compares baselines, default-configuration transformers, and NSGA-II–tuned models on varying forecast horizons. Chapter 8 discusses the findings, their interpretation and meaning, and any limitations during the study. Finally, Chapter 9 concludes and outlines the directions for future work.

2 Background and Related Work

2.1 The FRR Mechanism

The frequency restoration reserve (FRR) is a key component in modern electricity markets, designed to restore frequency deviations of the system and maintain grid stability in real time [Firoozi et al., 2020]. It consists of the manual (mFRR) and automatic (aFRR) participants. In liberalized energy systems, including the HUPX operated by MAVIR, aFRR is automatically activated in response to imbalances between supply and demand and is the dominant source of balancing despite the expensive service, as shown by the ENTSO-E data. This is due to poor forecasts and unforeseen events that compensate the market.

Market participants submit bids for upward and downward regulation, and the system operator activates these bids based on the system's needs and merit order. The participants in the balancing market consist mainly of large scale batteries and gas turbines that can be turned up or down if they are operating [Uijtewaal, 2025]. The resulting imbalance settlement prices are highly volatile and depend on real-time system conditions, making them particularly challenging to forecast. This volatility motivates the use of advanced time-series forecasting models capable of capturing both short-term fluctuations and long-term dependencies.

2.2 Timeseries forecasting fundamentals

Time series forecasting is the action of predicting future values based on historical and current observations that are typically ordered in time. Formally, given a sequence $\{x_t\}_{t=1}^{T}$, the goal is to learn a function that maps past observations to future values x_{t+h} for some forecast horizon h.
Classical timeseries forecasting involves autoregressive models such as ARIMA and exponential smoothing methods, which assume linear dependencies and stationarity [Shumway and Stoffer, 2017]. However, due to strong nonlinearities, regime shifts, and external dependencies such as weather and demand, the effectiveness of classical models to forecast energy markets is limited. This limitation of classical models has motivated the adoption of deep learning approaches that can model complex temporal patterns without strong assumptions about data distributions.

2.3 Transformer Architecture for Forecasting

Transformer architectures have gained popularity as a powerful alternative to sequence modeling due to their ability to capture long-range dependencies using self-attention mechanisms [Zhao et al., 2026]. Unlike recurrent neural networks, such as the popular LSTM model, transformers process entire sequences in parallel and compute pairwise dependencies between all time steps, allowing for more efficient learning of global temporal relationships. However, this strength also means that transformer models require large amounts of data to work efficiently.
In the context of time series forecasting, transformer-based models adapt the original architecture by incorporating positional encoding and specialized attention mechanisms tailored to continuous valued inputs [Zhao et al., 2026]. Transformer models have a large spectrum of different adaptations with temporal fusion transformers and encoder-only architectures such as PatchTST demonstrating strong performance for high volatility forecasting and also forecasting with complex temporal dependencies.

2.4 Multi-Objective Optimization

Multi-objective optimization addresses problems involving multiple conflicting objectives that must be optimized simultaneously while also achieving a fair balance. Instead of optimizing for a single statistic and producing a single optimal solution, such methods aim to identify a set of Pareto-optimal solutions, where no objective can be improved without degrading another [Deb, 2001]. Pareto-optimal defines the allocation of resources between two or more objectives in such a manner that altering the resources in a way that benefits one objective must degrade the other objective. In the context of machine learning, common objectives include predictive accuracy, computational efficiency, and model calibration. Formally, the optimization problem can be expressed as the following equation:

$$\min_{\theta \in \Theta} \mathbf{F}(\theta), \quad \mathbf{F}(\theta) = (f_1(\theta), f_2(\theta), \ldots, f_k(\theta)) \tag{1}$$

where each component, f_i, represents a unique objective function, such as the mean absolute error or the sum of squared residuals. The outcome of the optimization function is a Pareto front that provides trade-off solutions for decision-makers that decide based on the importance of objectives.

2.5 NSGA-II in Literature

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) is one of the most widely used evolutionary algorithms for multi-objective optimization [Yusoff et al., 2011]. The algorithm maintains a population of candidate solutions and iteratively improves them through selection, crossover, and mutation. Selection decides which candidate solutions are to reproduce based on their fitness score. Crossover combines two parent solutions to produce a new offspring, and finally mutation introduces random changes to an individual's genes.
NSGA-II introduces non-dominated sorting to rank candidate solutions based on Pareto dominance and uses a crowding distance metric to preserve diversity along the Pareto front. This algorithm has risen in popularity due to its computational efficiency and the ability to approximate complex Pareto surfaces, allowing it to be a standard choice in engineering, hyperparameter optimization, and machine learning applications.
NSGA-II has already been recorded in applications of neural architecture search and hyperparameter tuning, particularly in complex settings where there are multiple competing objective functions [Gayibov and Gasimov, 2025].

2.6 Related Works on Electricity Price Forecasting and NSGA-II for Hyperparameter Tuning of Deep Learning Models

Electricity price forecasting has been widely studied and researched using both classical econometric models and modern deep learning approaches. Research indicates that modern deep architectures such as LSTMs, temporal convolutional networks, and transformer-based models are superior in capturing non-linear dependencies and external factors that might cause price fluctuations [Zhao et al., 2026].
However, there is a clear gap that has not yet been explored since most literature focuses on single-objective optimization, typically prediction error without considering any other objectives such as the empirical 80% coverage interval – calibration – which can be a very strong performance indicator, too. Similarly, NSGA-II has been applied in machine learning, but its usage in forecasting the electricity market remains limited.
Some literature further indicates that genetic algorithms for hyperparameter tuning of deep learning models outperform random search methods [Verma et al., 2021]. Nevertheless, multi-objective genetic algorithm based hyperparameter tuning of electricity price forecasting transformer-based models remains an unexplored area of research, especially in the context of the Hungarian energy balancing system.

3 Data

3.1 Sources

The European Network of Transmission System Operators for Electricity (ENTSO-E) is an EU foundation by European TSOs to enable cooperation and to collect and publish data about each country's grid. Formally, the goal of the organization is "to promote closer cooperation between European TSOs to support the implementation of EU energy policy and achieve Europe's energy & climate policy objectives, which change the very nature of the power system". The organization's website and API client allows users to access the data with a valid API key, which for this research was provided by Cesi Invest Kft. and Skyline Codeworks B.V.

Six different data fields were collected from the ENTSO-E platform. Firstly, the aFRR upwards and downwards prices, the primary target, the aFRR capacity that was used. Moreover, price data of electricity was also collected from the largest electricity market, the day-ahead market (DAM), along with the system load and final imbalance volume and settlement price of balancing.
Some weather data was also collected from the open-meteo historical weather API which provides a free source of weather archive. Temperature, wind speed at 100m, shortwave radiation, and cloud cover were chosen as collected data fields because they are a proxy for the weather drivers of renewable generation and demand.
From each data source, monthly Parquet chunks were saved for caching and reproducibility. Data were saved within the study period, from April 2023 to the end of April 2026.

Table 2: All the data sources used in the study can be found below. All series are retrieved for the Hungarian bidding zone (*HU*) and span April 2023 (First available aFRR price data points) to April 2026, aligned to a common 15-minute grid in the *Europe/Budapest* time zone.

Series	**Provider**	**Access method**	**Native res.**	**Units**	**Role**
aFRR activation prices (up, down)[a]	ENTSO-E TP	`query_activated_balancing_energy_prices`	15 min	HUF/MWh	Target
aFRR procured capacity[b]	ENTSO-E TP	`query_procured_balancing_capacity` (process type A47)	15 min	MW; HUF/MWh	Exogenous
Day-ahead electricity price	ENTSO-E TP	`query_day_ahead_prices`	60 min	EUR/MWh	Exogenous
Actual total load	ENTSO-E TP	`query_load`	15 min	MW	Exogenous
Imbalance settlement prices (long, short)	ENTSO-E TP	`query_imbalance_prices`	15 min	HUF/MWh	Exogenous
System imbalance volume	ENTSO-E TP	`query_imbalance_volumes`	15 min	MW (signed)	Exogenous
Weather: temperature, wind speed (100 m), shortwave radiation, cloud cover	Open-Meteo	Historical Weather API	60 min[c]	°C, m/s, W/m^2, %	Exogenous

[a] The same query also returns mFRR activation prices; these are dropped in preprocessing, as mFRR is rarely activated within the Hungarian activation record over the study period.
[b] Summarized from the accepted-bid stack into six per-period statistics: total procured volume, maximum accepted price, and volume-weighted average price, for each direction (up/down).
[c] Resampled to the 15-minute grid by linear interpolation.

3.2 Data Formatting

Due to different temporal resolutions, the data are aligned with the aFRR temporal resolution, 15-minutes. This is because according to the PICASSO regulations all markets are required to have 15-minute resolutions, thus this temporal resolution is future proof, and even the main exchange

the day-ahead market within the HUPX changed to a 15-minute resolution since October 1, 2025 [HUPX, 2025].
The canonical grid is built with a single timezone aware `DatetimeIndex` and every series is reindexed to it. The timezone for the grid was chosen to be the Hungarian timezone to enhance the interpretability of the dataset.
The last issue that has to be addressed when building the dataset is daylight saving handling since when switching back from daylight saving time each day no longer consists of 96 quarters and due to duplicate datetime rows an `AmbiguousTimeError` would be thrown. To solve this issue a timezone aware canonical index is used so it automatically emits 92 slots on the spring day and 100 on the fall day. Moreover, to ensure data consistency the weather data is requested in UTC timezone since it does not have daylight-saving time, thus every single day consists of 96 quarters. ENTSO-E already returns data in UTC format as well. As a final defense against misalignment a defensive deduplication is used at the boundary which drops duplicate timestamps before reindexing due to dst transitions. This is just an extra measure since due to UTC timezone each row has to have a unique datetime.

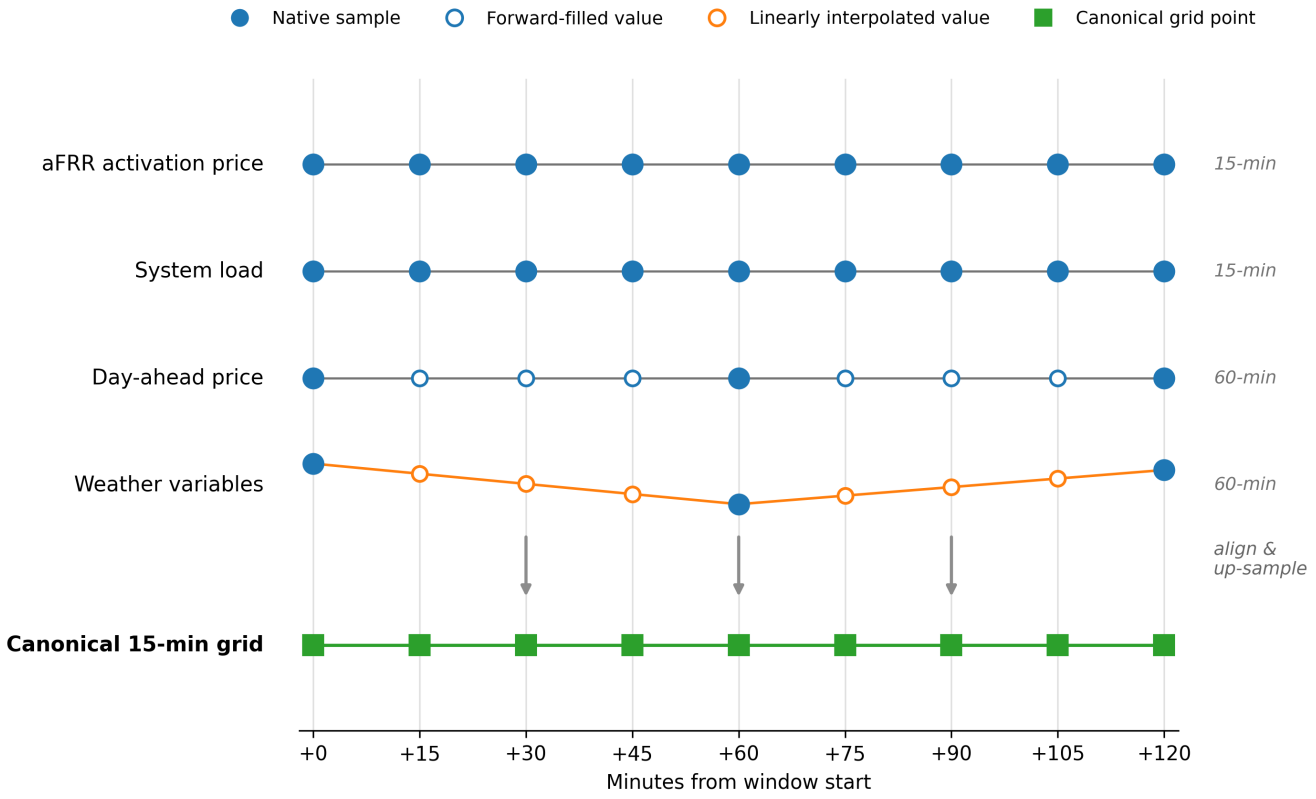

Figure 1: The alignment procedure of the raw input series onto the canonical 15-minute grid can be found above. AFRR activation price, system load, are in 15 minute resolution, thus they are mapped directly onto the grid; day-ahead price is forward-filled and weather is linearly interpolated, holding each native value constant until the next one. All series are indexed in the *Europe/Budapest* time zone. The aFRR procured capacity, published per procurement period is omitted for clarity but is aligned by the same forward-fill rule as day-ahead prices.

3.3 Missing Data and Data Quality

During data collection, there have been some missing data points mainly due to ENTSO-E publication gaps. These gaps mainly involved aFRR data, and since the affected columns include the target variables, the beginning of the data was clipped to the earliest aFRR activation data points in April 2023.
To handle non-systematic missing rows, if the row involves a missing target, it is dropped. In regards to missing features, they are first forward filled then backward filled. Moreover, missing features are tracked through `*_was_nan` indicator columns so the missingness is not silently erased. Additionally, 8 sentinel observations were found in the aFRR price data that are excluded since they are unrealistic data points and sit outside of the data range.
The final data set consists of 108,096 rows and 20 columns. However, mFRR and wind direction columns are dropped, thus only 17 columns are used for modeling.

Table 3: The table below depicts the missing values per column in the canonical 15-minute dataset, before feature-level treatment. Target columns with missing values cause the affected rows to be dropped. Other feature columns are first forward-, then back-filled with an accompanying missingness indicator for statistics.

Column	Group	Missing	% Missing	Treatment
da_price_eur	Market price	0	0.00	Fill + indicator
load_mw	System state	0	0.00	Fill + indicator
imb_long	System state	0	0.00	Fill + indicator
imb_short	System state	0	0.00	Fill + indicator
afrr_price_up	Target	0	0.00	Rows dropped
afrr_price_dn	Target	0	0.00	Rows dropped
imb_volume_mw	Target / system state	0	0.00	Rows dropped
temperature_2m	Weather	0	0.00	Fill + indicator
wind_speed_100m	Weather	0	0.00	Fill + indicator
shortwave_radiation	Weather	0	0.00	Fill + indicator
cloud_cover	Weather	0	0.00	Fill + indicator
afrr_cap_up_total	aFRR capacity	1,248	1.15	Fill + indicator
afrr_cap_up_max_price	aFRR capacity	1,248	1.15	Fill + indicator
afrr_cap_up_wavg_price	aFRR capacity	1,248	1.15	Fill + indicator
afrr_cap_dn_total	aFRR capacity	1,248	1.15	Fill + indicator
afrr_cap_dn_max_price	aFRR capacity	1,248	1.15	Fill + indicator
afrr_cap_dn_wavg_price	aFRR capacity	1,248	1.15	Fill + indicator

3.4 Exogenous Predictors

The exogenous predictors consist of raw data that are linked with the price of the balancing market. These predictors involve DAM price, weather data, system load, and imbalance which are all directly linked to the price of balancing. Each exogenous predictor influences the aFRR activation price through merit-order mechanism: the day-ahead price anchors the cost level of balancing-energy bids; procured capacity volume and capacity prices determine the depth and cost of the reserve pool, hence how far and how steeply activation climbs the merit order. Finally, weather drives both the level of renewable infeed - and through forecast error - the imbalances that activation is called

upon to correct.

3.5 Feature Engineering

The feature engineering consists of two distinct groups of features. First, the calendar features that provide cyclical encodings such as hour, day-of-week, month as sin/cos pairs, `is_weekend`, `is_holiday`, and `is_dst`. These allow the model to learn cyclical patterns that repeat over certain periods of time.
In addition, lagged features are created for the imbalance volume, load, and imbalance prices since these features are unobservable at their contemporaneous timestamp. This issue was originally discovered upon LightGBM ranking `imb_long` most important spuriously.

3.6 Target Variables and Their Distributions

The three important targets for forecasting the imbalance in the energy market are the aFRR upward and downward correct price and the imbalance volume. This paper especially focuses on the upward correction price due to its difficulty to predict and high volatility, but the models can be adapted to predict the other target feature as well.
The upward activation costs have heavy right tails and frequent zeros and low values since a shortage occurs only $\approx 34\%$ of the time. There are occasional extreme spikes due to large unforeseen balancing that involves a steep climb over the merit orders.
The distribution of the target is balanced using a log1p transformation to approach a normal distribution since neural networks perform better on a normally distributed target variable. For evaluation, the target is inverse-transformed for fair reporting of statistics. After target transformation the dataset has a skew of 0.685 and excess kurtosis of -1.51.

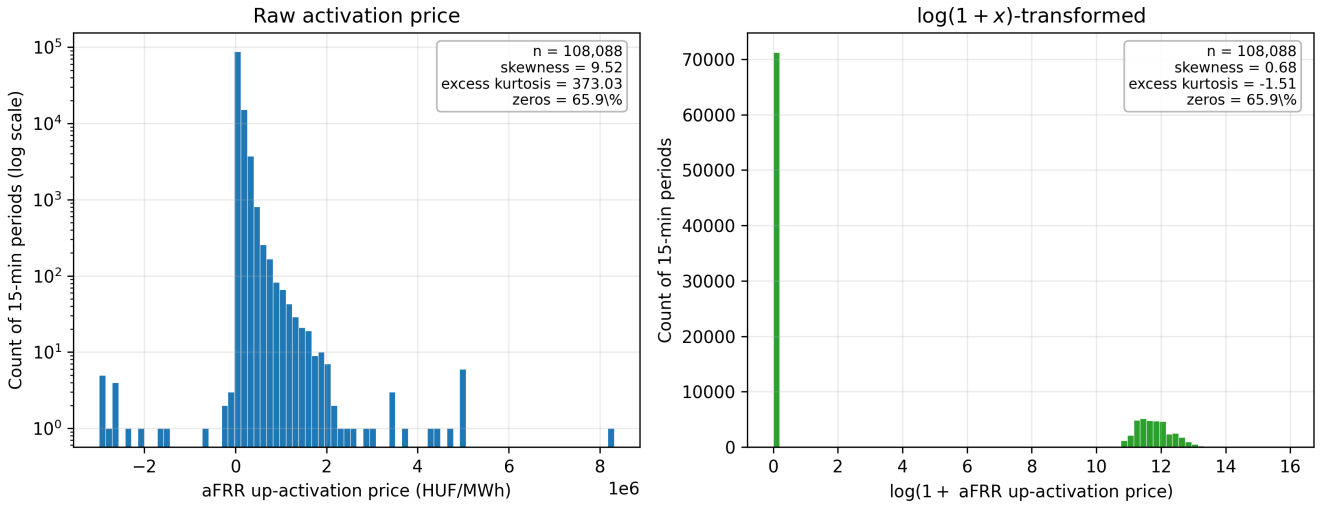

Figure 2: Distribution of the aFRR up-activation price over the study period. The left chart portrays the raw price with a logarithmic y-axis. The target is extremely heavy-tailed — skewness 9.52, excess kurtosis 373. The right chart shows the target distribution after a $\log(1+x)$ transformation. The transformation compresses the tail to near-symmetry — skewness 0.68. 65.9% of periods record no upward activation, so the target is zero-inflated and the transformation addresses the tail but not the spike at zero.

3.7 Exploratory Data Analysis

Before any model is specified, the target series and its candidate predictors are examined to characterize their statistical structure and to motivate the modeling choices that follow. Typically the exploratory data analysis has three main purposes, to identify periodic structure that any forecasting model must capture, expose non-stationarity that the cross-validation design must accommodate, and quantify the dependence between the target and the exogenous predictors.

3.7.1 Seasonality

aFRR activation prices inherit the strong periodic structure of the underlying power system. There are three main cycles that determine the market prices; the diurnal, weekly, and yearly cycles. The diurnal cycle explains the daily rhythm of electricity demand and renewable generation, with the largest changes in supply and demand occurring during the morning and evening ramps. The weekly cycle explains the weekday and weekend work cycle which is one of the main driving factors of demand. The third cycle, the yearly cycle further explains slow changes in the total renewable generation due to changes in weather which slowly moves the average price.

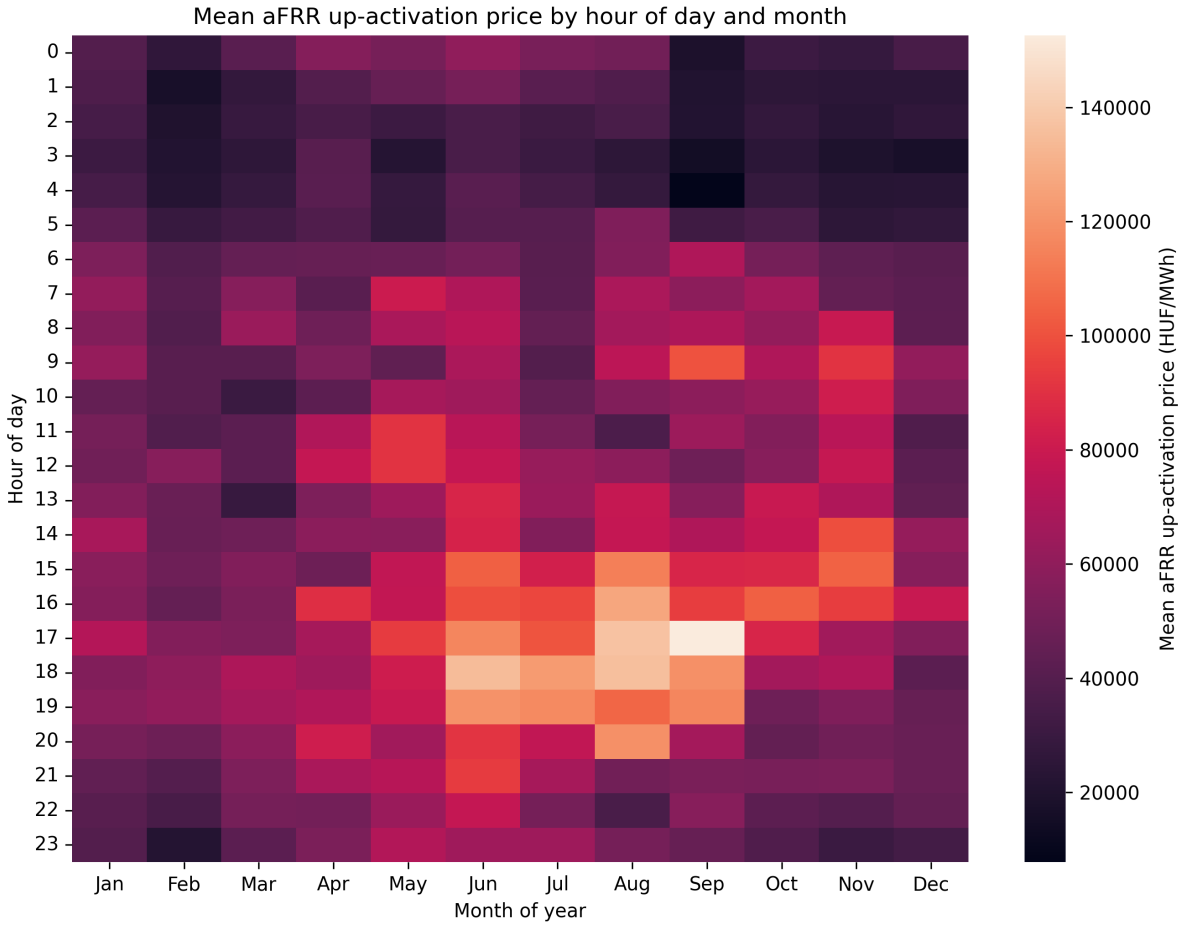

Figure 3: Mean aFRR up-activation price by hour of day and month of year, over the study period. The price is concentrated in the evening ramp — hours 15–20, peaking at 17:00 — and in the summer months, peaking in June. The overnight hours and the winter months are remarkably cheaper. Because the target is zero-inflated, each cell mean reflects both the frequency and the conditional level of activation.

Figure 3 shows the mean aFRR upward activation price as a heat map over hour-of-day versus month-of-year. The average activation price is the product of the frequency of activation and the conditional price given the activation where the activation is defined by $price \neq 0$:

$$E[\texttt{price}] = P(\texttt{activation}) \times E[\texttt{price}|\texttt{activation}] \tag{2}$$

Thus, a cell with a high mean may therefore reflect frequent low-cost or rare high-cost activation. These two cases can be separated by plotting the frequency separately. Important to note that the diurnal pattern is expressed in local `Europe/Budapest` timezone, therefore there is a daylight saving time switch.

3.7.2 Temporal Evolution and Regime Shifts

Figure 4 plots the entire target series over the study period, along with a 30-day rolling average and standard deviation. The study begins on April 2023, after the effects of the 2022 European energy-price crisis have mostly died out. It is clear that the series within the study window is not

stationary. Balancing prices visibly exhibit both shifts in level and pronounced volatility clustering, where steady periods are interrupted by bursts of extreme prices.

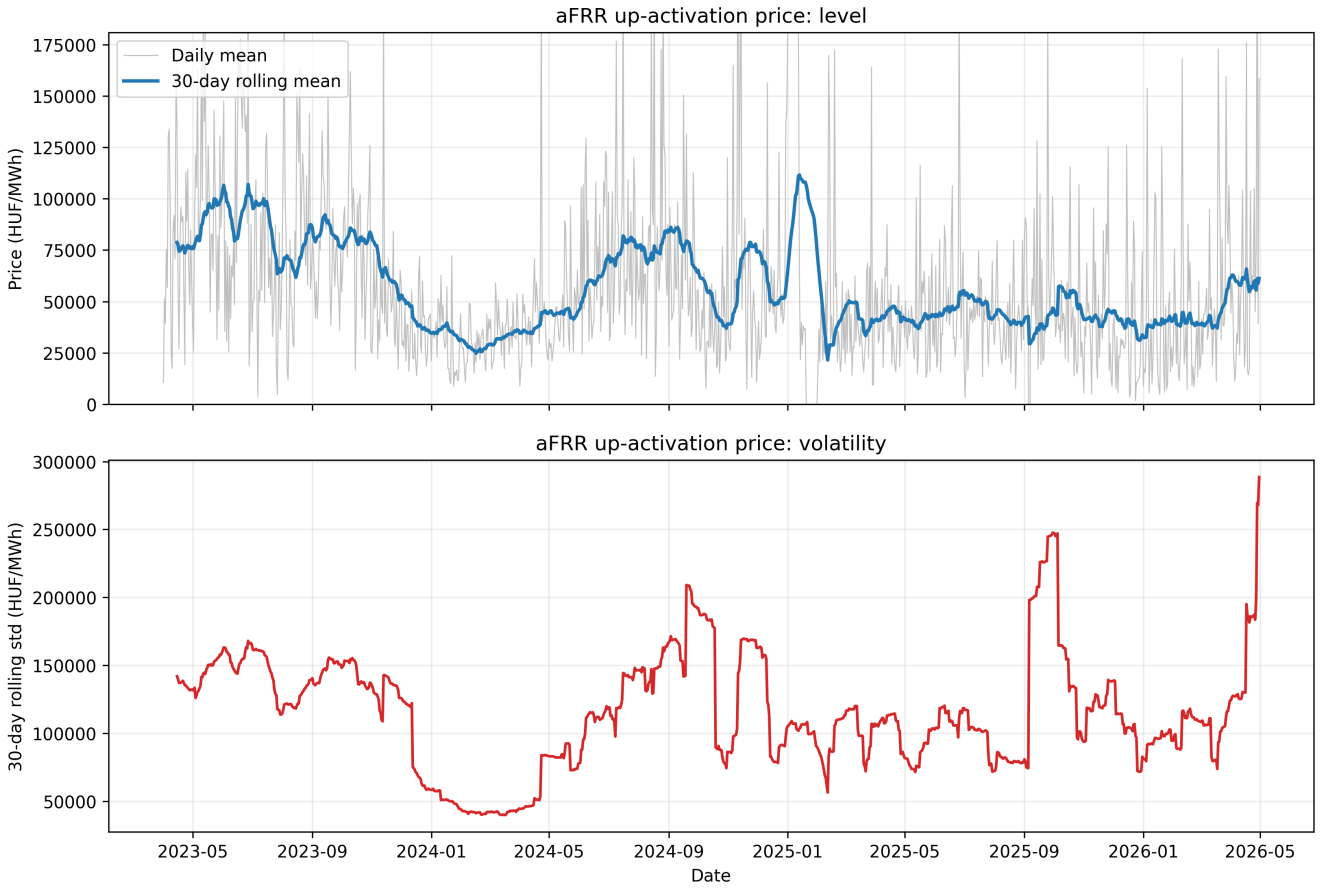

Figure 4: Evolution of the aFRR up-activation price over the study period. The top graph shows the daily mean price in grey and its 30-day rolling mean in blue. There are a total of 14 daily means above 180,880 HUF/MWh which are clipped for legibility. The bottom graph portrays the 30-day rolling standard deviation. The graph depicts the non-stationarity in the data. A calm regime starts off the study period in early 2024, a sharp level transient at the 2024/2025 turn, and steadily rising volatility through 2025–2026.

As shown in the figure above, there is a clear cyclical pattern in the aFRR up-activation prices over the study period. The activation costs peak daily in the afternoon ramp period which is around the 17 hour range, and over a year the activation costs peak in the summer months. Identifying these regimes is crucial for two distinct reasons. First, they determine how representative any single test window is of the others, and second they motivate the expanding-window cross-validation that is explained in section 3.8.

3.7.3 Autocorrelation

The autocorrelation function (ACF) and partial autocorrelation function (PACF) of the target series quantify its dependence on its own past and directly inform the length of the encoder window supplied to the transformer models. The figure below 5, shows both the ACF and PACF functions computed up to a lag of one week, 672 quarters.

Given the seasonality explained in section 3.7.1, it is expected that the ACF shows pronounced peaks in daily and weekly lags, with the daily peak larger of the two. Due to these large daily peaks it is justified to use a 24-hour encoder window, which is going to be used by both transformer architectures. The look-back must be long enough to contain at least one complete daily cycle, allowing the model to condition forecast on the same phase of the previous day. A shorter than the 96 steps would truncate the dominant periodicity, while a longer window significantly increases computational cost for a diminishing return.

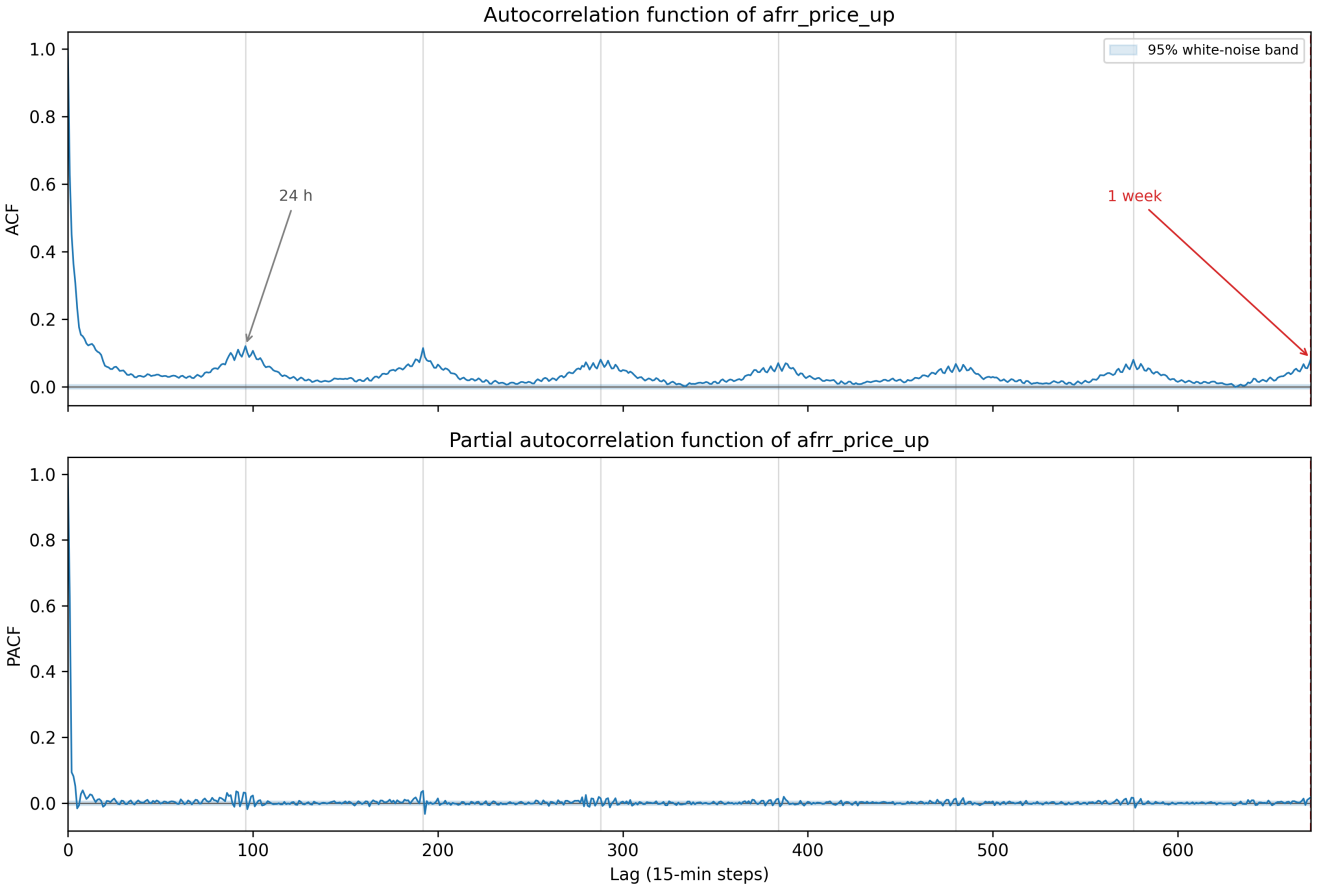

Figure 5: The top graph shows autocorrelation and the bottom depicts the partial autocorrelation of *afrr_price_up*, to a one-week lag. Grey guides mark the daily lag multiples. The autocorrelation function shows strong short-range dependence and a clear daily seasonal bump at lag 96, recurring at every daily multiple; the weekly term is comparatively weak. The partial autocorrelation function clearly cuts off sharply after the first lag.

It is important to note that since the ACF of the target series is zero-inflated, the ACF partly reflects the persistence of the activation state itself.

3.7.4 Cross-correlation with exogenous

Section 3.4 explained the economic mechanisms by which the DAM price, the procured capacity, and the weather affect the activation price. This section covers the corresponding statistical dependence on such exogenous predictors. Figure 6 shows the contemporaneous correlation matrix among the targets and the principal predictors.

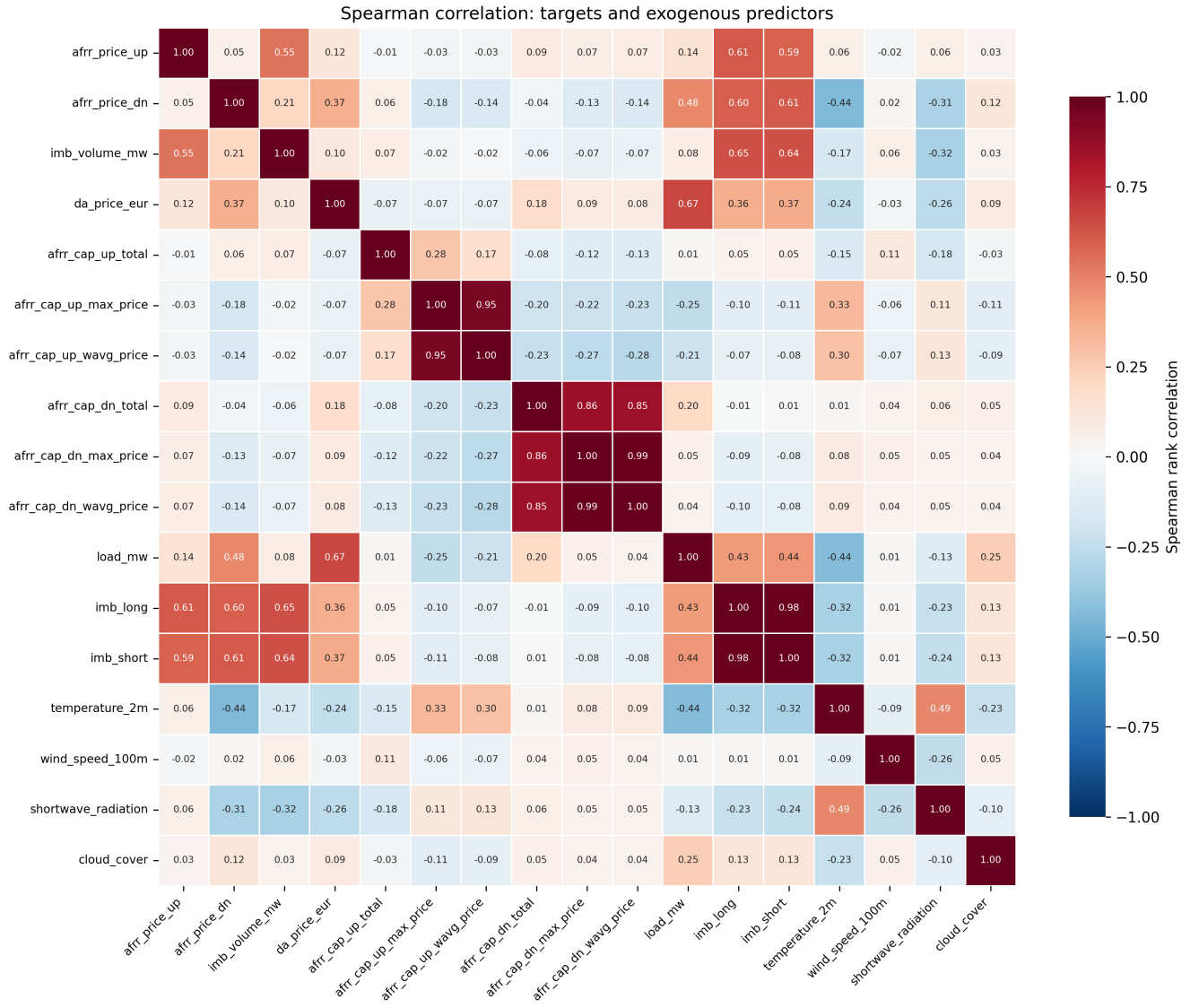

Figure 6: The figure shows the Spearman rank correlation among the targets and exogenous predictors over the study period. Rank correlation is used in place of Pearson because the price series are heavy-tailed and zero-inflated. Sentinel observations are excluded. The strongest correlates of *afrr_price_up* are the imbalance variables, which are observable only after the settlement period closes. These features need to be lagged to avoid data leakage. The known-future predictors show only weak monotonic association highlighting the difficulty of predicting the aFRR upward activation prices.

The lead-lag structure is of particular interest since a predictor that is correlated with the target itself at a positive lead conveys a lot of useful information for forecasting, whereas one correlated only contemporaneously is less useful and exploitable. The DAM price is published a day before delivery; thus, it is a known future input and anchors the balancing costs; therefore, it is expected to be positively associated with the activation price as an anchor of the balancing-energy cost level. System load is expected to be positively associated through system tightness. And lastly, weather is expected to indirectly correlate with the target through its effect on renewable energy generation.

3.8 Walk-forward Splits

The standard k-fold cross-validation partitions observations randomly into training and test folds, which is invalid for time-series data. Random partitioning places observations from after a given date in the training fold and observations from before in the test fold, so the model is allowed to learn from the future to predict the past. Since electricity-market series are strongly autocorrelated as shown in section 3.7.3, an observation and its immediate neighbors are similar. Thus, a randomly selected test point will almost always have a similar neighbor in the training fold therefore making the results biased. A valid evaluation must respect the flow of time in such a manner that every observation used to train a model must precede every observation used to test it.
Due to difficulties with splitting the data, this study adopts a walk-forward cross validation with an expanding training window. A total of three folds are used, and each fold trains from the start of the study period. Successive folds extend the training window forward in time, thus sliding the validation and test windows with it. The split configuration consists of a minimum of 24 months of training data, 2 months of validation and test data with a 4 month step between successive folds. 4 month step is the largest step that is permitted given the training, validation and test split due to the 37 months of total data. The exact boundaries can be found in table 4 below.

Table 4: The table highlights the walk-forward cross-validation folds. The training window expands from a fixed start (2023-04). The validation and test windows slide forward by a 4-month step per fold. All windows contain a 2-month validation and 2-month test set. Date intervals are half-open: a window labeled "a to b" covers $[a, b)$.

Fold	**Training window**	**Train length**	**Validation window**	**Test window**
1	2023-04 to 2025-04	24 months	2025-04 to 2025-06	2025-06 to 2025-08
2	2023-04 to 2025-08	28 months	2025-08 to 2025-10	2025-10 to 2025-12
3	2023-04 to 2025-12	32 months	2025-12 to 2026-02	2026-02 to 2026-04

Each fold partitions the data into three contiguous, non-overlapping roles. The validation window is used for model-selection decisions that must not touch the test data — early stopping and the NSGA-II hyperparameter search of Chapter 5. The test window is used only exactly once per model, for the final evaluation reported in Chapter 7. A visualization of the 3 folds can be seen below in figure 7.

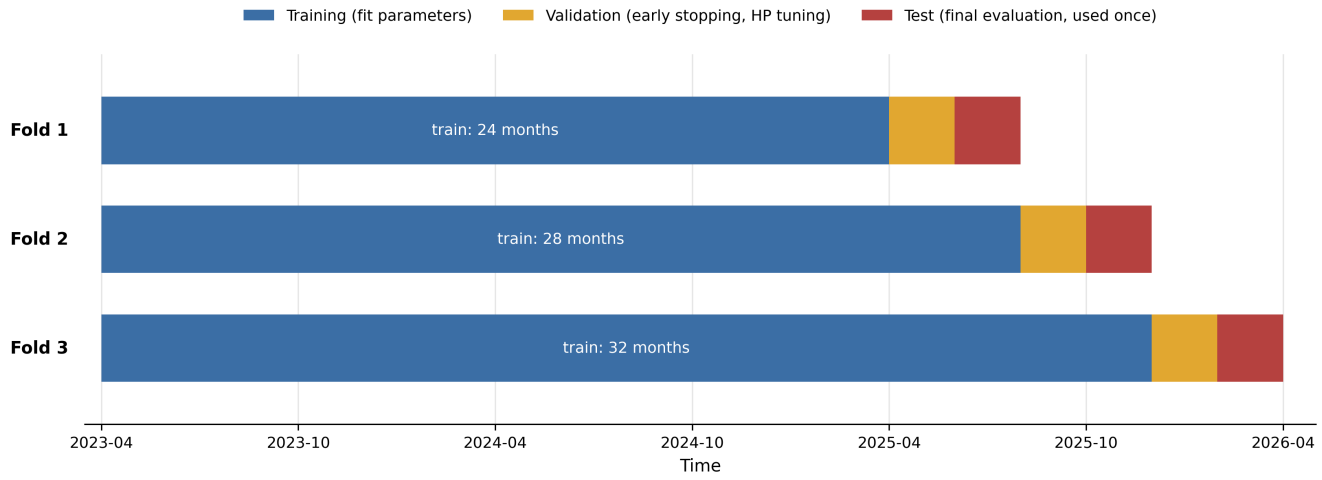

Figure 7: The walk-forward cross-validation scheme. Each fold is a contiguous split of the study period into training, validation, and test windows. The training window expands from a fixed start (2023-04); the 2-month validation and 2-month test windows slide forward by a 4-month step between folds. The test window of each fold is held out entirely and used exactly once, for the final evaluation reported in Chapter 7.

4 Forecasting Models

4.1 Baselines

First, to test the effectiveness of the transformer-based models two baseline models will be built that serve a bar for the transformer-based models to pass to justify their complexity. These baseline models also serve as the basis for the comparison of the statistics used throughout Chapter 7. After careful research and considerations the two most fit baseline models for high volatility timeseries forecasting are seasonal-naive and gradient-boosted tree models.

4.1.1 Seasonal-naive

Seasonal-naive forecasting is one of the simplest forms of forecasting for time-series targets. It relies on the assumption that the future will look like the most recent equivalent season of the past [Hyndman and Athanasopoulos, 2021]. Instead of simply taking the latest observed target, seasonal naive forecasting uses the value from the same season in the previous cycle. Due to the seasonality explained in section 3.7.1 there were two baseline models constructed with a daily season and a weekly season. However, for basis of comparison the daily model is used and the weekly model purely provides further support for arguments.
Formally, the forecast formula is defined as:

$$\hat{y}_{t+h} = y_{t+h-m} \tag{3}$$

where y_t is the observed value at time t, m is the seasonal period and $\hat{y}_{t+h}$ is the forecast for future period $t + h$. Despite the simplicity of the model, it works well in forecasting targets with strong repeating seasonal patterns, stable seasonal behavior over time and targets with limited trend or structural change [Hyndman and Athanasopoulos, 2021]. However, it struggles when there is a

strong upward/downward trend and during times of external shocks. As previously discussed, due to the strong seasonality within the balancing prices, the seasonal naive model serves as a strong baseline for comparison.

4.1.2 LightGBM

The second chosen baseline model is the light gradient boosting machine (LightGBM), which is in essence a gradient-boosted regression tree for supervised learning. It was originally developed by Microsoft as a fast, scalable implementation of gradient boosting [Ke et al., 2017]. The model builds an ensemble of decision trees sequentially with each new tree attempting to correct the mistakes made by the previous tree. Formally, the predictions can be modeled with the equation:

$$\hat{y} = T_1(x) + T_2(x) + T_3(x) + ... + T_n(x) \tag{4}$$

where T_i is a decision tree and each tree learns from the residual errors of the previous trees. Unlike the seasonal-naive model, the LightGBM benefits from the feature set which consists of lagged target values, calendar features, and lagged exogenous predictors. The model outputs point forecasts via LightGBM's default L2 objective. This model originally discovered the data leakage caused by the imbalance, `imb_long`, feature and served as motivation for the lagging rule. The input feature set also allows ranking of the features by importance and this is output for interpretability. Regarding the hyperparameters, since this model is simply a baseline model a sensible configuration was built and will not be optimized via NSGA-II. The configuration for the model can be found below:

Table 5: The table shows the hyperparameter configuration of the LightGBM baseline model. One model is trained per combination of (horizon, fold). Boosting iterations are capped at 1500 and the final count is selected by early stopping on the validation fold.

Hyperparameter	**Value**	**Description**
Objective and boosting		
`objective`	regression (L2)	Squared-error loss (LightGBM default)
`n_estimators`	1500	Maximum boosting iterations (early-stopped)
`learning_rate`	0.05	Shrinkage applied to each new tree
`num_leaves`	63	Maximum leaves per tree
`max_depth`	-1	Tree depth unconstrained
`min_child_samples`	20	Minimum observations per leaf
Sampling and regularisation		
`subsample`	0.9	Row (bagging) fraction per iteration
`subsample_freq`	1	Bagging applied every iteration
`colsample_bytree`	0.8	Feature fraction per tree
`reg_alpha`	0.0	L1 regularisation
`reg_lambda`	0.0	L2 regularisation
Training protocol		
early-stopping rounds	50	Halt if validation loss does not improve
`random_state`	42	Seed for reproducibility

4.2 TFT

4.2.1 Model Structure

The temporal fusion transformer (TFT) is a deep learning architecture designed specifically for multi-horizon time-series forecasting [Lim et al., 2020]. It was introduced in 2019 by Google and has proven to be a strong transformer-based model for seasonal forecasting. Unlike the simple baseline models, the TFT model can learn from historical time-series values, known future inputs such as holidays and weather forecasts, and static information.
TFT combines recurrent networks, LSTMs, for sequential pattern processing, while also utilizing the transformer attention mechanism to enable it to capture long-range dependencies. This allows the model to combine the LSTM model's power of capturing short-range dependencies while ensuring that long-range dependencies are not ignored either through the attention mechanism. The model also incorporates a feature selection network that allows one to choose important variables and finally a gating mechanism to suppress unnecessary components. The combination of these structures allows the model to be very effective at multi-horizon forecasting and capturing seasonal patterns.

4.2.2 Motivation for TFT Architecture

The reason for choosing this architecture is its strong alignment with the requirements of aFRR price forecasting. A key motivation is TFT's native handling of known-future covariates, which allow the model to incorporate information that is available across the entire forecast horizon [Lim et al., 2020]. These covariates include DAM market prices, calendar effects, weather forecasts, and other exogenous features. This prevents restricting the model inputs to the historical encoding stage, making the model particularly advantageous in electricity markets, where future calendar structure and scheduled market information often carry substantial predictive value.
A second motivation is TFT's support for direct multi-horizon forecasting. Rather than recursively generating one-step-ahead predictions as the vast majority of Transformer models do, TFT produces forecasts for the full prediction window in a single forward pass. Moreover, the naive quantile output layer aligns naturally with the probabilistic forecasting objective adopted in this paper, allowing the model to estimate predictive distributions and uncertainty intervals.
Interpretability is also an important point to consider. Unlike the majority of deep learning architectures that operate as a black-box model, TFT provides mechanisms for post-hoc analysis through the previously mentioned variable selection network and temporal attention weights. These components allow for in-depth analysis, such as examining which variables contribute most strongly to predictions, and which historical time periods receive the greatest attention. This is valuable for the analysis that is conducted in Chapter 7 and 8, where these model outputs are examined in detail.

4.3 PatchTST

4.3.1 Model Structure

Patch Time Series-Transformer, PatchTST, is a transformer-based forecasting model designed specifically for long-horizon multivariate time series forecasting. Yuqi Nie et al. were the first to introduce the model as an adaptation of transformer architectures for time-series data [Nie et al., 2023].
The central idea behind PatchTST is to divide an input time series into local patches, similar to the technique used in vision transformers [Nie et al., 2023]. The model groups consecutive observations

into fixed-length segments called patches, which are treated as input tokens for the transformer. This results in reduced sequence length, which improves computational efficiency and enable the model to capture local temporal patterns before modeling long-range dependencies. Formally; given a historical input sequence of length L, each variable is divided into overlapping or non-overlapping patches, depending on the model type, of length P, and stride S. Each patch thus becomes one transformer token allowing a reduced sequence length N from L given by the formula:

$$N \approx \frac{L - P}{S} + 1 \tag{5}$$

This allows for scalable attention computation for long input windows.
Another strength of PatchTST comes from channel independence. Instead of jointly embedding all variables at the input stage, each time-series channel is processed independently using the same shared transformer weights [Nie et al., 2023]. This means that each variable is patched separately and passed through the same encoder architecture, which reduces parameter count, improves generalization, and helps to prevent overfitting in high-dimensional settings.
Patch embedding also contributes to this model's success. Each patch is projected into a latent embedding space using a linear projection layer. This allows the raw temporal segments to be converted into dense vector representations suitable for transformer processing. Positional encoding is added to retain information about the temporal order of patches. The embedded patches are then passed through stacked transformer encoder blocks. Each encoder layer is made up of multi-head self-attention mechanisms that capture dependencies between distant temporal segments, feed-forward networks that learn non-linear feature transformations and residual connections and layer normalizations to improve optimization stability and gradient flow. Finally, the encoder output is passed to a prediction head which maps latent representations to the forecast horizon. In this study, the output head is configured for probabilistic forecasting to align with the previously mentioned nature of balancing price forecasting.

4.3.2 Motivation for PatchTST Architecture

The PatchTST model was used in this study due to its strong empirical performance in long-horizon forecasting tasks and its architectural simplicity compared to covariate-heavy forecasting models. There have been multiple records of the model achieving state-of-the-art results on several long-horizon benchmarks due to its patch-based tokenisation. The reduced sequence length lowers the quadratic computational cost of self-attention while providing each token with richer local semantic context. This results in the model being able to capture short-term temporal structure without the need of sacrificing long-range dependency modeling.
Another motivation for using this model is the use of the Reversible Instance Normalization (RevIN), which directly addresses non-stationarity in time-series data [Kim et al., 2022]. RevIN is a technique used in deep learning, especially in time-series forecasting, that resolves the issue of distributions. Distributional shifts and regime changes are common in the balancing market as discussed in section 3.7.2, and is a key challenge in aFRR price dynamics. Thus using RevIN the model is especially fit for the task of the forecasting the balancing market of Hungary.
PatchTST also allows for comparison with more complex forecasting frameworks such as the other transformer model, TFT. PatchTST is primarily a pure time-series model that does not rely on extensive exogenous feature engineering, unlike the TFT model. This deliberate difference between the models allows us to investigate whether forecasting performance for aFRR prices is driven more by sophisticated covariate integration and interpretability mechanisms or by strong sequence modeling capacity and architectural efficiency.

4.4 Loss Functions

The last important detail of the models is the loss function. To enable probabilistic forecasting, all transformer models in this study are trained using the quantile loss. Unlike conventional regression objectives, for example, mean squared error, which optimize a single point estimate, quantile loss allows the model to estimate conditional quantiles of the target distribution. This is important since it allows the prediction of uncertainty intervals alongside central forecasts. This can be defined as the equation:

$$L_q(y, \hat{y}_q) = \texttt{max}(q(y - \hat{y}_q), (q-1)(y - \hat{y}_q)) \tag{6}$$

where the target value is y, the predicted quantile is $\hat{y}_q$ and q is the quantile level where $q \in (0, 1)$. The asymmetric loss penalizes under- and over-prediction differently depending on the selected quantile level. This study uses the forecast quantile set:

$$q \in \{0.1, 0.5, 0.9\} \tag{7}$$

corresponding to the 10, 50, and 90 percent predictive intervals. The final training objective is computed as the average quantile loss, or pinball loss, across the three quantiles and across all 16 horizons.

Quantile loss is motivated by the stochastic nature of electricity market prices, particularly in the dominant aFRR market, where price spikes and regime changes are common. A single point forecast provides less value than a calibrated probabilistic forecast, since it does not provide information about uncertainty.

The target variable is scaled using a log1p transformation, as explained in section 3.6, to stabilize variance and reduce the influence of extreme price spikes. Therefore, the loss is computed in the log-transformed space during optimization. However, to increase interpretability and usability the model predictions are inverse-transformed to the original price scale. For fair comparisons with the base models, the median of the output probability can be taken as a single point and therefore allows all four models to be compared on the same grounds.

5 Multi-Objective Hyperparameter Tuning with NSGA-II

5.1 Motivation

After a smoke test of the transformer models it was clear that they were accurate as they already outperformed the baseline models with an $\texttt{MAE} \approx 37 - 38k$ vs $\texttt{LightGBM}46k$ but the trade-off is visible already. PatchTST under-covers its nominal 80% interval at longer horizons, while the TFT model systematically over-covers. While neither of the coverage errors are catastrophic, they leave room for improvement that a multi-objective search can exploit by trading a small amount of accuracy for an improvement in the calibration metric at each Pareto point.

Single-objective hyperparameter optimization (HPO) is usually conducted using grid search, random search, or Bayesian search on one scalar. All of these methods force a fixed accuracy-vs-calibration weighting before the trade-off is known. However, multi-objective hyperparameter tuning using Non-dominated Sorting Genetic Algorithm II (NSGA-II) allows for the recovery of the whole set of non-dominated configurations, the Pareto front, so the trade-off is explicit and configuration is chosen post hoc. Instead of returning one solution, the algorithm returns an entire Pareto front allowing the trade-off to be tuned exactly, and the algorithm also handles a mixed discrete or continuous search space without the need for gradients. Moreover, in rough search spaces with

multiple local minimums where standard HPO struggles to find the optimal solution, the NSGA-II still is able to find the Pareto front making it a good solution for multi-objective hyperparameter optimization [Yusoff et al., 2011]. The NSGA-II algorithm will be implemented from scratch as it is the central topic of the essay and is one of the main contributions of this thesis.

5.2 Multi-Objective Optimization Preliminaries

Multi-objective optimization finds the Pareto front where the only possibility to make one objective can be done with making at least another objective worse. The general problem minimizes the vector:

$$f(x) = (f_1(x), ..., f_m(x)) \tag{8}$$

where each $f_n(x)$ is an objective

5.2.1 Pareto

In multi-objective optimization, solutions are evaluated across multiple often conflicting objectives. A solution is said to Pareto dominate another solution if it is no worse than the other solution on every objective and strictly better on at least one objective. To formalize the definition, when $f(x) = (f_1(x), f_2(x), ..., f_m(x))$ then x dominates another solution y when $f_i(x) \leq f_i(y)$ for all objectives i, with at least one strict inequality. A set consisting of solutions that are not dominated by any other solution forms of the non-dominated set. The corresponding solutions in decision variable space form the Pareto-optimal set, while their mapped objective values form the Pareto front. The Pareto front represents the best achievable trade-offs between the competing objectives. This means that movements along the Pareto front reflect a trade-off surface where improving one objective degrades another. Within the Pareto front the ideal point is referred to as the hypothetical objective vector composed of the individually best achievable values for each objective, however this is generally not achievable in practice. Moreover, the nadir point is the point on the Pareto-optimal solutions where the worst objective values are achieved. These two points bound the Pareto front and support the interpretation of optimization results. The figure below illustrates the previously discussed concepts for two-objective optimization problem, with dominated solutions, dominance regions and resulting non-dominated Pareto front clearly depicted.

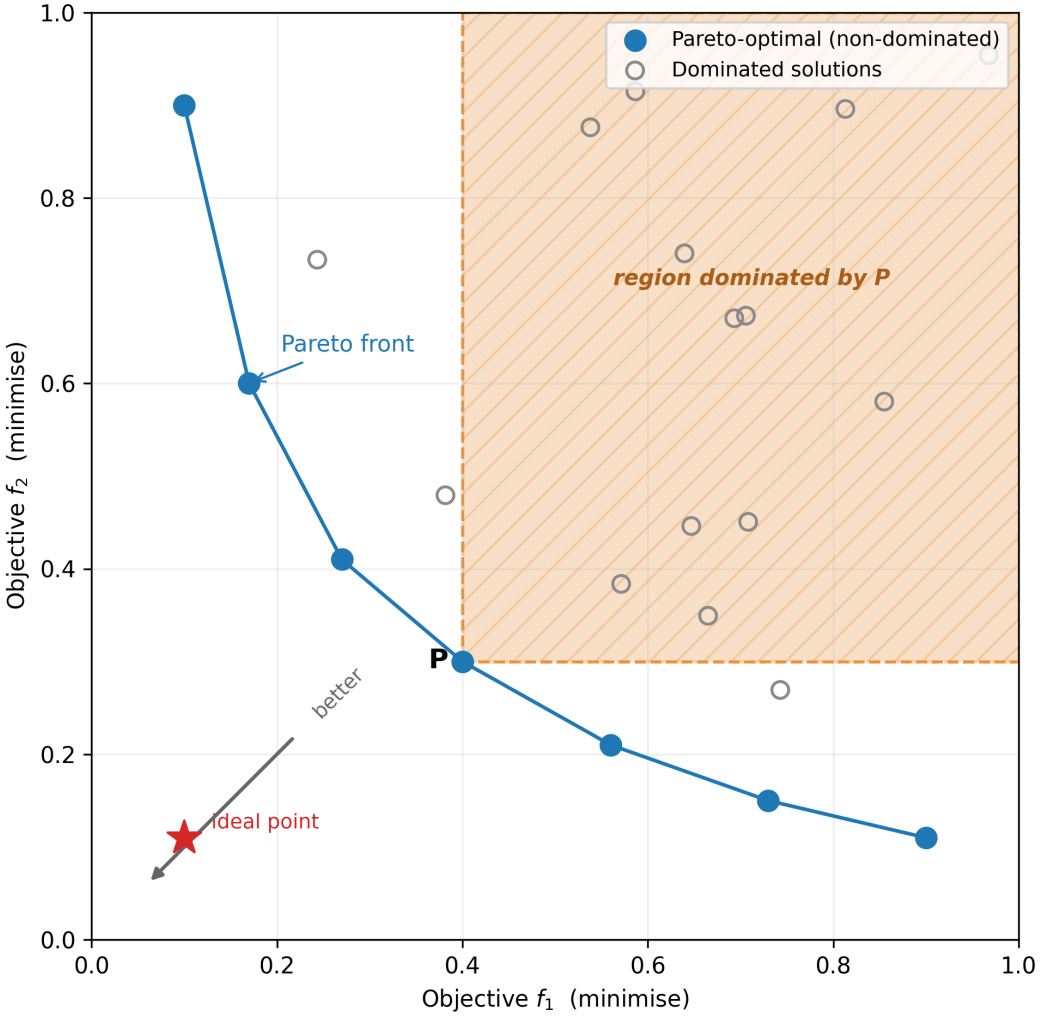

Figure 8: Illustration of Pareto dominance in a two-objective minimization problem. Each marker is a candidate solution with the full markers forming the Pareto front. A solution is *dominated* if another is no worse on both objectives and strictly better on at least one. The orange shaded region contains every solution dominated by point P. The *ideal point*, shown as a red star, combines the best attainable value of each objective and is generally not itself achievable.

5.2.2 Decomposition

To solve such a complex problem, decomposition is required. This is done through scalarisation, which refers to combining objectives into a single scaler through weighted aggregation. Formally, this process can be defined by the equation:

$$\sum_i w_i f_i(x), \texttt{ where} w_i \geq 0 \tag{9}$$

Each weight w_i is a user-defined importance weight assigned to each objective. This process allows the complex problem to be simplified into a simple, standard single-objective optimization task, allowing the use of conventional optimization algorithms.
However, this approach has a significant problem for two reasons. First, it requires specification of weights that implicitly incorporate subjective preferences before the trade-off structure of the

problem is fully understood. The second reason is that this approach does not work in non-convex regions of the Pareto front. Classical single-objective optimization algorithms use gradient thus a non-convex region results in getting stuck in local minimums. This results in certain Pareto-optimal solutions that aren't found meaning that parts of the efficient frontier are excluded.
An alternative scalarisation method is called the Tchebycheff approach. This method of scalarisation minimizes the maximum weighted deviation from an ideal reference point [Lin et al., 2024]. Although it does outperform traditional approaches in non-convex solution spaces and therefore improve coverage of the Pareto front, it has a big limitation. It still relies on scalarisation and does not guaranty uniform exploration of the trade-off surface.
Thus, decomposition-based scalarisation methods are inadequate for the goal of this study, and therefore Pareto-based population methods need to be used. These methods approximate the entire Pareto front without the need for predefined weights. Therefore, Pareto-based methods provide a more complete representation of the trade-off structure and allow post hoc solution selection.

5.3 Evolutionary Algorithms

5.3.1 Fundamentals

Evolutionary Algorithms (EAs) are population-based metaheuristic optimization algorithms that were inspired by natural selection and biological evolution. Instead of iteratively improving a single candidate solution, EAs maintain a population of candidate solutions that improve and explore the search space through three fundamental mechanisms [Bäck and Schwefel, 1993]. Optimization begins with initialization of a population, followed by fitness evaluation of the candidate solutions; this process is the evaluation of the performance of a candidate solution according to the optimization objectives. Subsequently, the three fundamental operations take place – selection, crossover, and mutation. Selection is the process where higher-fitness solutions are preferentially chosen for reproduction. New candidate solutions are generated by crossover which is the process of combining parameters of parent solutions to generate a child candidate solution and finally by chance mutation happens which is the alteration of a gene of a child candidate solution. Each solution is represented using a genome, where each parameter is a gene. EAs, can also be tuned using selection pressure which has a large effect on effectiveness. Selection pressure determines how strongly better solutions are favored, which tunes the balance between exploration of new regions of the search space and exploitation of promising existing solutions. These properties allow EAs to be almost perfectly suited for multi-objective optimization (MOO), since the population naturally represents a diverse set of candidate trade-off solutions. This allows EAs to approximate the entire Pareto front in a single optimization run, rather then using scalarised single-objective optimization iterations.

5.3.2 Generational vs Steady-state schemes

There are two distinct methods for updating the population in EAs. In a generational approach, the entire population is updated with a newly generated set of child solutions at each iteration. This results in defined generations and synchronized population overrides. The other approach involves the replacement of a subset of solutions at the same time. This allows new solutions to enter the population, while most of the existing solutions remain unchanged within the population. This paper has chosen to incorporate the generational framework into the NSGA-II algorithm combined with elitist replacement, which is the process of taking the best performing individuals, the "elite", and copying them to the next generation. Through this strategy, it is ensured that high-quality non-dominated solutions are not discarded across generations, providing guaranteed

elitism while maintaining clean generation semantics and stable convergence behavior throughout the optimization process.

5.4 NSGA-II in detail

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) was first introduced by Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and Tanmoy Meyarivan in 2002 [Deb et al., 2002]. It gained quick reputation for its performance in multi-objective evolutionary algorithms, and therefore it is regarded as the canonical approach to Pareto-based optimization. As the name suggests, the algorithm employs non-dominated sorting, as well as crowding-distance diversity preservation and elitist population replacement to efficiently approximate the Pareto front. The pseudocode of the algorithm can be found below and in the following sections each critical parts of the NSGA-II algorithm will be investigated using pseudocode for support.

Algorithm 1 NSGA-II

Require: objective f, search space $\mathcal{S}$, population size N, generations G
Ensure: approximation of the Pareto front

$P \leftarrow N$ individuals sampled uniformly from $\mathcal{S}$
evaluate f for every individual in P
$\mathcal{F} \leftarrow \textsc{FastNonDominatedSort}(P)$
for all front $F_i \in \mathcal{F}$ **do**
    $\textsc{CrowdingDistance}(F_i)$
end for
for $g \leftarrow 1$ **to** G **do**
    $Q \leftarrow \emptyset$
    while $|Q| < N$ **do**
        $p_1 \leftarrow \textsc{TournamentSelect}(P)$; $p_2 \leftarrow \textsc{TournamentSelect}(P)$
        $(c_1, c_2) \leftarrow \textsc{Crossover}(p_1, p_2)$
        $\textsc{Mutate}(c_1)$; $\textsc{Mutate}(c_2)$
        add c_1 to Q; **if** $|Q| < N$ **then** add c_2 to Q
    end while
    evaluate f for every individual in Q
    $R \leftarrow P \cup Q$ ▷ combined pool, $|R| = 2N$
    $\mathcal{F} \leftarrow \textsc{FastNonDominatedSort}(R)$
    $P_{\text{next}} \leftarrow \emptyset$; $i \leftarrow 1$
    while $|P_{\text{next}}| + |F_i| \leq N$ **do** ▷ elitist $(\mu + \lambda)$ replacement
        $\textsc{CrowdingDistance}(F_i)$
        $P_{\text{next}} \leftarrow P_{\text{next}} \cup F_i$; $i \leftarrow i + 1$
    end while
    $\textsc{CrowdingDistance}(F_i)$
    sort F_i by crowding distance, descending
    $P_{\text{next}} \leftarrow P_{\text{next}} \cup F_i[1 : N - |P_{\text{next}}|]$
    $P \leftarrow P_{\text{next}}$
end for
return $\{\, p \in P : \text{rank}(p) = 1 \,\}$

5.4.1 Fast non-dominated sorting

The non-dominated sorting of the NSGA-II is a crucial component that partitions the population into a sequence of Pareto front $F_1, F_2, ...$ according to dominance rank [Yusoff et al., 2011]. This means that the first front F_1 contains all the non-dominated solutions, while F_2 contains the next set of non-dominated solutions after all individuals in earlier fronts have been accounted for, in this case F_1. As seen in the pseudocode below, in order for these fronts to get computed the algorithm maintains two structures for each solution p; a dominated set S_p, made up of all solutions that are dominated by p, and a domination count n_p, which is the count of solutions that dominate p. Through these two components the algorithm can compute the first Pareto front F_1 by selecting the individuals with $n_p = 0$. After this for each solutions q in front F_1, the domination set S_q is checked and for each solution q dominates the domination count n_i is decreased. The individuals whose domination count becomes zero, $n_p = 0$, are assigned to the next front during iteration, and this is repeated until no solutions are left.

Algorithm 2 FastNonDominatedSort

Require: population P
Ensure: ordered fronts $F_1, F_2, \ldots$

for all $p \in P$ **do**
  $S_p \leftarrow \emptyset$; $n_p \leftarrow 0$
  for all $q \in P \setminus \{p\}$ **do**
    if $p \prec q$ **then**
      $S_p \leftarrow S_p \cup \{q\}$ ▷ p dominates q
    else if $q \prec p$ **then**
      $n_p \leftarrow n_p + 1$
    end if
  end for
  if $n_p = 0$ **then**
    $\text{rank}(p) \leftarrow 1$; $F_1 \leftarrow F_1 \cup \{p\}$
  end if
end for
$i \leftarrow 1$
while $F_i \neq \emptyset$ **do**
  $H \leftarrow \emptyset$
  for all $p \in F_i$ **do**
    for all $q \in S_p$ **do**
      $n_q \leftarrow n_q - 1$
      if $n_q = 0$ **then**
        $\text{rank}(q) \leftarrow i + 1$; $H \leftarrow H \cup \{q\}$
      end if
    end for
  end for
  $i \leftarrow i + 1$; $F_i \leftarrow H$
end while
return $F_1, \ldots, F_{i-1}$

The computational complexity of fast non-dominated sorting is $O(MN^2)$, where M is the number

of objectives and N is the population size. This N^2 component is because each solution is compared to all solutions, resulting in N^2 pairwise comparisons. With each comparison there are up to M objective-wise comparisons giving the final efficiency of $O(MN^2)$

5.4.2 Crowding distance

NSGA-II employs an important measure called the crowding distance which is used to measure density to preserve diversity among the solutions within the same Pareto front. In essence it estimates the density of each solution through the distance to its neighbors using normalized distance between the two closest neighboring solutions and summing these contributions across all objectives [Yusoff et al., 2011]. To define it formally, for a solution i, the crowding distance is defined as the sum of normalized objective-wise distances to its adjacent solutions in sorted objective space. Solutions with minimum or maximum values in any objective are referred to as the boundary solutions and these solutions are assigned infinite crowding distance to ensure they are always preserved during selection, thus maintaining a diverse Pareto front.

Algorithm 3 CrowdingDistance

Require: front F, objective count M

$l \leftarrow |F|$
if $l \leq 2$ **then**
    $\text{dist}(p) \leftarrow \infty$ for all $p \in F$; **return**
end if
$\text{dist}(p) \leftarrow 0$ for all $p \in F$
for $m \leftarrow 1$ **to** M **do**
    sort F ascending by objective f_m
    $\text{dist}(F[1]) \leftarrow \infty$; $\text{dist}(F[l]) \leftarrow \infty$
    $f_m^{\min} \leftarrow f_m(F[1])$; $f_m^{\max} \leftarrow f_m(F[l])$
    if $f_m^{\max} = f_m^{\min}$ **then continue**
    end if
    for $k \leftarrow 2$ **to** $l - 1$ **do**
        $\text{dist}(F[k]) \mathrel{+}= \dfrac{f_m(F[k+1]) - f_m(F[k-1])}{f_m^{\max} - f_m^{\min}}$
    end for
end for

5.4.3 Crowded-comparison operator

Another important aspect of NSGA-II is the crowded-comparison operator which defines a partial ordering over solutions that guides both selection and survival [Yusoff et al., 2011]. The operator prioritizes Pareto front fitness, or rank, with the secondary aspect being the crowding distance. It is used to break the tie within the same front through selecting the solution with the greater crowding distance. The combination of these two criterion ensures that there is a balance between diversity and convergence toward the Pareto front. Finally, the crowded-comparison operator is employed during both parent selection, and population truncation.

5.4.4 Simulated Binary Crossover

Simulated Binary Crossover (SBX) is a real-coded recombination operator used in NSGA-II. It emulates the behavior of single-point crossover in binary representation while operating directly on continuous-valued decision variables. The operator generates offspring through a convex combination of parent solutions that are controlled through a spread factor β [Deb and Agrawal, 1995]. The distribution of offspring must also be controlled and its done through the distribution index η_c. Larger values of η_c produce offspring closer to the parent solution, leading to a reduced exploration variance and more exploitation. Formally, for a pair of parent variables x_1 and x_2, the spread factor β_q is factored using a probability distribution:

$$\beta_q = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, & u \leq 0.5 \\ (\frac{1}{2(1-u)})^{\frac{1}{\eta_c+1}}, & u > 0.5 \end{cases} \tag{10}$$

where u is a random variable sampled from a uniform distribution between 0 and 1. The resulting offspring are then constructed as symmetric perturbations around the parents using this spread factor.

As seen in the pseudocode below a bounded-SBX correction is applied to ensure that the offspring falls within predefined variable limits. This is done through clipping or resampling to remain within valid bounds, preserving feasibility without any distortion to the crossover distribution excessively near the edges of the search space.

Algorithm 4 Simulated binary crossover for one variable pair

Require: parents x_1, x_2, bounds $[x^\ell, x^u]$, distribution index η_c
Ensure: children c_1, c_2
if $|x_1 - x_2| < \epsilon$ **then return** x_1, x_2
end if
ensure $x_1 \leq x_2$ (swap if necessary)
for $k \in \{1, 2\}$ **do** ▷ one spread factor per side
  $d \leftarrow x_1 - x^\ell$ **if** $k = 1$ **else** $x^u - x_2$
  $\beta \leftarrow 1 + 2d/(x_2 - x_1)$
  $\alpha \leftarrow 2 - \beta^{-(\eta_c+1)}$
  $u \sim \mathcal{U}(0, 1)$
  if $u \leq 1/\alpha$ **then**
    $\beta_q^{(k)} \leftarrow (u\,\alpha)^{1/(\eta_c+1)}$
  else
    $\beta_q^{(k)} \leftarrow \big(1/(2 - u\,\alpha)\big)^{1/(\eta_c+1)}$
  end if
end for
$c_1 \leftarrow \frac{1}{2}\big[(x_1 + x_2) - \beta_q^{(1)}(x_2 - x_1)\big]$
$c_2 \leftarrow \frac{1}{2}\big[(x_1 + x_2) + \beta_q^{(2)}(x_2 - x_1)\big]$
return $\mathrm{clip}(c_1, x^\ell, x^u),\ \mathrm{clip}(c_2, x^\ell, x^u)$

Within this study, SBX is applied after the log transformation of the hyperparameters, to ensure that multiplicative-scale parameters are handled more appropriately and to ensure that crossover operations behave consistently across orders of magnitude. The logarithmic scale of the hyperparameters allows for greater stability and search efficiency for highly skewed parameter distributions.

5.4.5 Polynomial mutation

In terms of the mutation operator, NSGA-II utilizes polynomial mutation to introduce controlled variation into continuous decision variables. Unlike binary mutation, that simply flips discrete bits, polynomial mutation perturbs individual genes through the addition of small stochastic offsets. This allows the algorithm to explore nearby regions of the search space while simultaneously maintaining continuity. The perturbation can be controlled through the magnitude and distribution through the distribution index η_m. A larger value results in smaller mutations concentrated closer to the parent value, encouraging local exploitation, while a smaller value encourages exploration of the search space. Given a gene x, the mutation step can be defined by the mutation factor $\delta_q =:$

$$\delta_q = \begin{cases} (2u)^{\frac{1}{\eta_m+1}} - 1, & u < 0.5 \\ 1 - [2(1-u)]^{\frac{1}{\eta_m+1}}, & u \geq 0.5 \end{cases} \tag{11}$$

where, like the SBX formula, u is a random variable sampled from a uniform distribution between 0 and 1. The mutated gene is updated using perturbation while respecting the predefined variable bounds.

A per-gene mutation probability of $1/L$, where L is the genome length, is the standard recommendation for real-coded evolutionary algorithms [Deb, 2001]. This essentially means that on average one gene is mutated per offspring. This is enough to balance variation without excessively disrupting the inherited structure. Thus, this study will use the above mentioned standard probability of mutation. This finally gives the pseudocode:

Algorithm 5 Polynomial mutation for one variable

Require: value x, bounds $[x^\ell, x^u]$, distribution index η_m

$\delta_1 \leftarrow (x - x^\ell)/(x^u - x^\ell)$
$\delta_2 \leftarrow (x^u - x)/(x^u - x^\ell)$
$u \sim \mathcal{U}(0, 1)$
if $u < 0.5$ **then**
    $\delta_q \leftarrow \big[\, 2u + (1-2u)(1-\delta_1)^{\eta_m+1} \,\big]^{1/(\eta_m+1)} - 1$
else
    $\delta_q \leftarrow 1 - \big[\, 2(1-u) + 2(u-0.5)(1-\delta_2)^{\eta_m+1} \,\big]^{1/(\eta_m+1)}$
end if
return $\mathrm{clip}\big(x + \delta_q(x^u - x^\ell),\ x^\ell,\ x^u\big)$

5.4.6 Successive Halving

To further optimize the evolutionary algorithmic hyperparameter optimization successive halving was introduced. The algorithm starts with the training of N offspring, as shown in algorithm 6 below [Jamieson and Talwalkar, 2015]. The total resources are evenly spread between the offspring until the end of rung 0, where each offspring is trained to a maximum of $max\_epochs/4$. Next, rank the N configurations by Pareto front first and then by crowding distance to break the ties. The top $ceil(N/\eta)$ is kept where η is the halving rate. Next rung 1 begins where each of the remaining offspring are trained from scratch until a maximum of $max\_epochs/2$, and then they are ranked the same way as in rung 0, and the same formula is applied to truncate the offspring. Finally, the remaining survivors are trained to a maximum of max_epochs epochs and ranked. All of the candidate solutions regardless of elimination are then passed to the NSGA-II for the $(\mu + \lambda)$

replacement step discussed in section 5.4.7. The eliminated configurations are not deleted, however, they tend to lose the replacement tournament naturally. With a population size of 20, and halving factor of 2, and a maximum epoch of 20, successive halving saves around 38% computational resources. Without successive halving there would be $20 \times 20 = 400$ epochs per generation, while with successive halving this is reduced to $20 \times 5 + 10 \times 10 + 5 \times 20 = 250$ epochs.

Algorithm 6 Multi-objective successive halving (one NSGA-II generation)

Require: active set $A_0 = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$; rung budgets $r_0 \le r_1 \le \cdots \le r_{K-1}$; halving rate $\eta \ge 2$; fitness $f(\mathbf{x}, b) \in \mathbb{R}^M$ (minimised)
Ensure: per-rung objective records for every $\mathbf{x} \in A_0$

for $k \leftarrow 0$ **to** $K - 1$ **do**
  for all $\mathbf{x} \in A_k$ **do** ▷ train from scratch with budget r_k
    $\mathbf{o}(\mathbf{x}) \leftarrow f(\mathbf{x}, r_k)$
    record $(\mathbf{x}, k, r_k, \mathbf{o}(\mathbf{x}))$
  end for
  if $k = K - 1$ **then break**
  end if
  $\{F_1, F_2, \dots\} \leftarrow$ FastNonDominatedSort(A_k)
  for all fronts F_j **do**
    AssignCrowdingDistance(A_k, F_j)
  end for
  $n_{\text{keep}} \leftarrow \max\big(1, \lceil |A_k|/\eta \rceil\big)$
  sort A_k ascending by $\big(\text{rank}(\mathbf{x}), -\text{crowding}(\mathbf{x})\big)$
  $A_{k+1} \leftarrow$ first n_{keep} elements of sorted A_k ▷ rest keep $\mathbf{o}$ from rung k
end for

5.4.7 Elitist Replacement

The final component of NSGA-II that needs mentioning is the elitist population replacement. Using standard practices, NSGA-II utilizes an elitist replacement strategy, where the population of parents and offspring of size N are combined to form an intermediate population of size $2N$ [Yusoff et al., 2011]. Subsequently, the next generation is constructed through the addition of complete fronts until the population limit is reached. In scenarios where the final front exceeds the remaining capacity, the diversity of solutions is taken into account, and only the most diverse solutions are retained. The metric used for this assessment is the crowding distance discussed in section 5.4.2. Through this measure, convergence quality and front diversity are preserved during population truncation.

5.5 Application to Transformer Hyperparameter Optimization

5.5.1 Mixed-type Genome

Inherently, the hyperparameter space can be heterogeneous, especially for neural networks, making the application of NSGA-II challenging for transformer hyperparameter optimization. Heterogeneous refers to the search space containing a mixture of parameter types, such as integer, categorical, and logarithmically scaled continuous variables. As an example, in the TFT model's search space, parameters such as the dropout are continuous values defined by a bounded interval while layer

depth is integer-valued. A standard EA cannot treat these parameter types identically, due to different variation behavior and sampling.
To address this issue, it is crucial to define the domain of each hyperparameter by encoding its type and valid bounds. Each gene in the genome requires a parameter specification that describes sampling and variation during evolution. The four main types that need to be included are categorical, integer, float, and log-float. The categorical variables need to be a fixed set of discrete options, the integers need to be sampled uniformly from bounded integer intervals, the floats need to be sample uniformly over a continuous range, and finally the log-floats need to be uniformly sampled from the logarithmic space. This enables the use of different operators per value type. For continuous genes SBX and Polynomial Mutation is used while categorical and integer-valued genes are mutated using discrete operators such as uniform crossover. Finally, for the log-scaled values crossover and mutation are done in the logarithmic space to ensure there is no bias within the search space. Through this structure the NSGA-II optimization can be utilized for a heterogeneous hyperparameter space and therefore the above mentioned structure will be followed by this paper.

5.5.2 Objective Design

The problem that this essay addresses is formulated as a multi-objective minimization task through the NSGA-II framework. The first objective is accuracy for which Mean Absolute Scaled Error (MASE) will be used across the 5 key forecasting horizons; 15m, 1h, 2h, 3h, 4h. This is to ensure that scale-invariant evaluation of point prediction performance is fair between the transformers and the baseline models. The second objective is calibration quality which is calculated through the absolute deviation between empirical 80% coverage and its nominal target value of 0.8. Finally, this value is averaged across the same horizons. The formulas for the two objective functions can be found below:

$$\text{MASE} = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \text{MASE}_h, \qquad \mathcal{H} = \{1, 4, 8, 12, 16\} \tag{12}$$

$$\mathcal{L}_{\text{cal}} = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \left| \text{coverage}_{0.8,h} - 0.8 \right|, \qquad \mathcal{H} = \{1, 4, 8, 12, 16\} \tag{13}$$

Following standard machine-learning practice, both objectives are computed on the validation window of the chosen fold. The test window of each fold is held out entirely from the NSGA-II search and used exactly once, at final evaluation in Chapter 7, so that test scores remain an unbiased estimate of generalized performance.
These two objectives are a reflection of the trade-off discussed in section 5.1, where improving accuracy generally results in a loss of calibration, and vice versa. Although the loss function could include both these metrics it has many limitations, which is discussed in section 5.1. Through the usage of multiple competing objectives, the EA can explore a diverse set of Pareto-optimal solutions.
It is important to note that $\mathcal{L}_{\rfloor\dashv\updownarrow}$ as defined above, is considered a non-proper objective. A proper objective is defined by a forecaster minimizing their expected objective score by reporting their true belief. In other words, a coverage error of a fixed interval fails this criterion because the objective score depends on the observation falling in an interval and not the full predictive distribution [Gneiting and Raftery, 2007]. Technically, this means that instead of the model learning to minimize the objective function, it will just widen the prediction interval until the target coverage is met without any genuine understanding of the distribution forecast. This limitation will further be explored in section 8.2.

5.5.3 Search Space for TFT and PatchTST

The hyperparameter space for the transformers is made up of a combination of model specific constraints such as lstm layers for TFT and patch length for PatchTST and shared parameters such as learning rate. The complete parameter search space can be found in table 6 and table 7 below.

Table 6: The TFT hyperparameter search space explored by the NSGA-II tuning evolutionary algorithm. Continuous hyperparameters such as *dropout*, and, *learning_rate* are varied by simulated binary crossover and polynomial mutation. On the other hand, categorical and integer hyperparameters are varied by uniform crossover and random reset. The categorical menus are chosen so that every *(hidden_size, attention_head_size)* pair is valid — the head count always divides the hidden size — which removes the need for rejection sampling.

Hyperparameter	**Type**	**Domain**	**Description**
`hidden_size`	Categorical	{16, 32, 64}	Dimension of the model's hidden state
`attention_head_size`	Categorical	{1, 2, 4}	Number of interpretable attention heads
`hidden_continuous_size`	Categorical	{8, 16, 32}	Hidden size for continuous-variable processing
`lstm_layers`	Integer	{1, 2}	Number of LSTM encoder/decoder layers
`dropout`	Continuous	[0.05, 0.35]	Dropout rate
`learning_rate`	Log-continuous	$[10^{-4},\ 5 \times 10^{-3}]$	AdamW learning rate
`batch_size`	Categorical	{128, 256, 512}	Training batch size

Table 7: The PatchTST hyperparameter search space explored by the NSGA-II tuning evolutionary algorithm. Continuous hyperparameters such as *dropout*, *learning_rate*, and *weight_decay* are varied by simulated binary crossover and polynomial mutation. On the other hand, categorical and integer hyperparameters are varied by uniform crossover and random reset. Every *(d_model, n_heads)* pair is valid by construction, as the head count always divides the model dimension. The *stride* $\leq$ *patch_len* constraint is not menu-enforceable and is handled by clamping the stride at use time.

Hyperparameter	**Type**	**Domain**	**Description**
`d_model`	Categorical	{64, 128, 256}	Transformer embedding (model) dimension
`n_heads`	Categorical	{2, 4, 8}	Number of attention heads
`n_layers`	Integer	[2, 4]	Number of transformer encoder layers
`d_ff`	Categorical	{128, 256, 512}	Feed-forward inner dimension
`patch_len`	Categorical	{8, 16, 24}	Length of each input patch
`stride`	Categorical	{4, 8, 16}	Step between consecutive patches
`dropout`	Continuous	[0.05, 0.35]	Dropout rate
`learning_rate`	Log-continuous	$[10^{-4},\ 5 \times 10^{-3}]$	AdamW learning rate
`weight_decay`	Log-continuous	$[10^{-6},\ 10^{-3}]$	AdamW weight decay
`batch_size`	Categorical	{128, 256, 512}	Training batch size

5.5.4 Constraint Handling

Apart from the mixed types of hyperparameters there are other important issues that the NSGA-II algorithm must abide by. There are multiple constraints for parameters within neural networks,

for example within the attention mechanism of transformers, the number of head must divide the model dimension, and in the patching module the stride must be less then or equal to patch length. Due to these constraints the full independence of genes that is typically assumed in standard evolutionary representations is violated.
To make sure that the NSGA-II crossover and mutation compatibility is preserved, the implemented optimizer will enforce a constraint-preserving encoding strategy. Categorical design choices are structured so that only combinations satisfying the divisibility constraints are retained. Regarding inequality constraints, values are clamped at evaluation or decoding, to ensure feasibility. This way the underlying search space representation is also kept static, and therefore all constraints can be enforced while crossover and mutation compatibility is kept.
The reason behind this design choices is that alternative constraint-handling strategies such as rejection sampling and penalty-based fitness shaping, is due to efficiency reasons. Rejection sampling introduces inefficiency because it leads to frequent invalid offspring generation and thus wasted evaluations especially in the context of transformers where the search space can be tightly constrained. The penalty-based fitness shaping introduces artificial fitness gradients that can result in a mislead selection pressure away from genuinely promising regions.

5.6 Implementation

5.6.1 From-scratch Design

The multi-objective optimizer is implemented from scratch rather than using an established library such as Optuna. This decision was made because the EA is the methodological core of the thesis, and a self-made implementation allows full control over every operator and make the algorithm itself an explicit contribution of the work. The self-made algorithm also allows for the ablation studies in section 5.7. The implementation is split into small number of components with separate responsibilities. The `ParamSpec` type encodes the domain of a single hyperparameter, thus the search space can be expressed through a mapping from parameter names to `ParamSpec` objects. The second component is the `Individual` class, which encompasses a genome, its evaluation, rank, crowding, generation and a unique id. The third component is a single driver function `run_nsga2` that composes the ranking routines; fast non-dominated sorting, crowding-distance assignment, and variation operators; binary tournament selection, simulated binary crossover, and polynomial mutation, into the generational loop specialized optimization framework. It is important to note that due to the high complexity of the transformer models the tuning will be carried out on fold 3 only. The case study is largely computationally constrained and tuning on all three folds would result in a time and computational requirements that are unavailable.

5.6.2 State persistence and resumability

To make the algorithm efficient and flexible in case of crashes or errors a `state.json` file is written after each generation. Moreover, a `history.parquet` file is employed to log every evaluation with rank and crowding so no information is lost upon unexpected events. This allows the algorithm to resume from the last completed generation when re-run, which is essential for multi-hour searches. The reason behind the design is due to the computational load of the TFT architecture which took significantly more time to train then the PatchTST model, thus as a security measure state persistence and resumability was implemented.
Another important consideration was the handling of failed trials. To deal with this issue the algorithm is designed to catch errors in trials and to avoid the solution in future generations

objective values are set to infinite and the searching continues without problems.

5.6.3 Reproducibility

To make all the results and experiments in this study reproducible it was crucial for the algorithm to be deterministic. The RNG is seeded before runs to ensure the same RNG is achieved through different runs. Moreover the fold structure explained in section 3.8 is made deterministic and a fixed search space is set. These measures ensure that multiple runs yield the same results and thus the whole experiment is reproducible. The library free design further allows for an entire run to be reconstructible from the saved `history.parquet` file and through exact dependency versions.

5.7 Algorithmic Ablation Studies

The four ablation studies within the following section probe the from-scratch NSGA-II implementation, along the algorithm's three principal degrees of freedom; SBX distribution index η_c, the population-vs-generation trade-off under fixed budget, and convergence over generations. The algorithm will be compared to the simplest form of optimization, uniform random sampling. To measure hypervolume (HV) against ground truth rather than against the serach's own best-found point, and its $\sim$1 the studies will be conducted on the canonical ZDT1 benchmark rather than on the TFT/PatchTST objective [Zitzler et al., 2000]. μs evaluation cost enables 10-seed Monte Carlo averages that the real $\sim$10 h-per-run problem cannot afford. The real runs use the same code path, thus any conclusion about implementation behavior transfers directly to the real runs.

5.7.1 η_c Sensitivity

Figure 9 and table 8 sweep the SBX distribution index $\eta_c \in \{2, 5, 15, 30, 50\}$. The population size is fixed at 30 with 50 generations, 10 seeds each. The algorithm with an $\eta_c = 15$, the NSGA-II default, achieves 61.6% of the true Pareto-front hypervolume, but with the most explorative tested setting, $\eta_c = 2$, the algorithm reaches all the way to 94.9% of the hypervolume. This is because the ZDT1's Pareto front is smooth and connected. A small η_c generates children further from their parents allowing for diversity along a continuous front. Since in this paper there are discrete and log-uniform parameters, discussed in section 5.5.3, categorical genes are recombined using uniform crossover rather than SBX. Thus η_c sensitivity is muted, and a classical η_c of 15 is used for production runs.

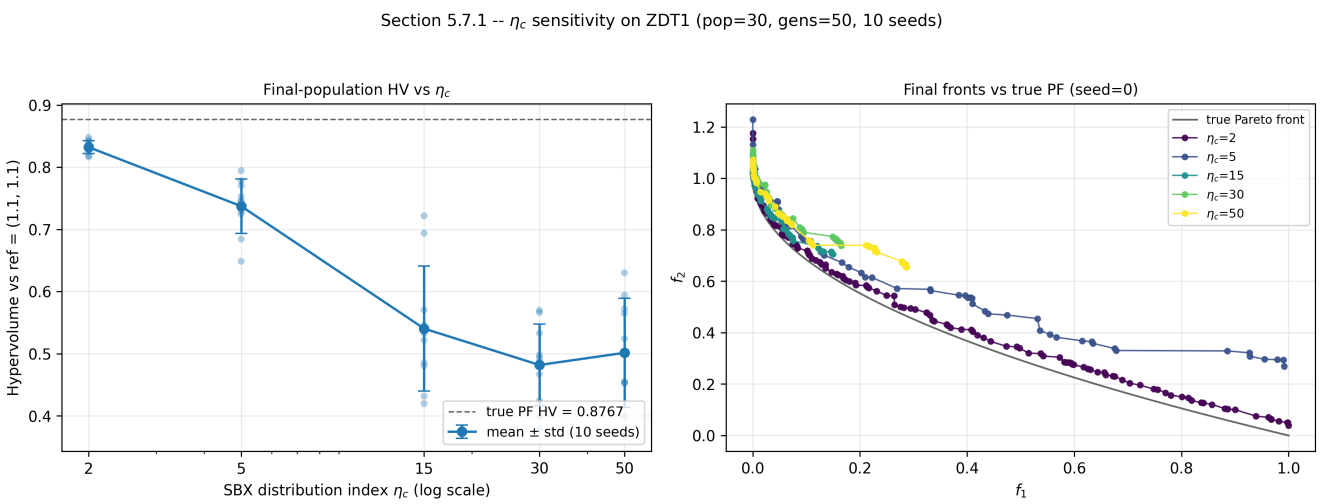

Figure 9: The graphs above show the η_c sensitivity on ZDT1. Left graph illustrates mean ± std final hypervolume across 10 seeds at five η_c values. The horizontal dashed line is the true Pareto-front hypervolume. On the other side the graph portrays the final fronts for seed 0 at each η_c, overlaid on the analytical true Pareto front.

Table 8: Final-population hypervolume of NSGA-II on the ZDT1 benchmark as a function of the SBX distribution index η_c. With a fixed population size of 30 and 50 generations, each row aggregates 10 independent seeds. True Pareto-front hypervolume 0.8767.

η_c	HV mean	HV std	HV / true PF	—front— mean
2	**0.8323**	0.0105	0.949	131.1
5	0.7374	0.0438	0.841	90.5
15	0.5405	0.1004	0.616	70.6
30	0.4819	0.0656	0.550	51.9
50	0.5015	0.0879	0.572	56.2

5.7.2 Population Size vs Generation Count Under Fixed Budget

With a fixed budget of ≈ 1500, six pop and gens pairs from $(10, 149)$ to $(100, 14)$ were swept, shown in figure 10 and table 9. The best configuration was found to be $(20, 74)$ with mean HV 0.563, with the worst configuration $(100, 14)$. Under a fixed budget, the NSGA-II benefits from many rounds of selection with a moderate population, than from a single large random initial sample, due to the elitist replacement step which requires multiple generations to refine the Pareto front. For the actual runs conducted in this paper, a fixed population of 20, and 5 generation is used along with successive halving, due to the high computation cost of training and evaluating models. This results in the real runs being well below the $(20, 74)$ configurations identified as optimal above. This is since each transformer trial takes ≈ 10-15 minutes for TFT, thus the optimal number of configurations would take an extensive time and computational costs. This constraint is examined in section 5.7.3 and 8.2.

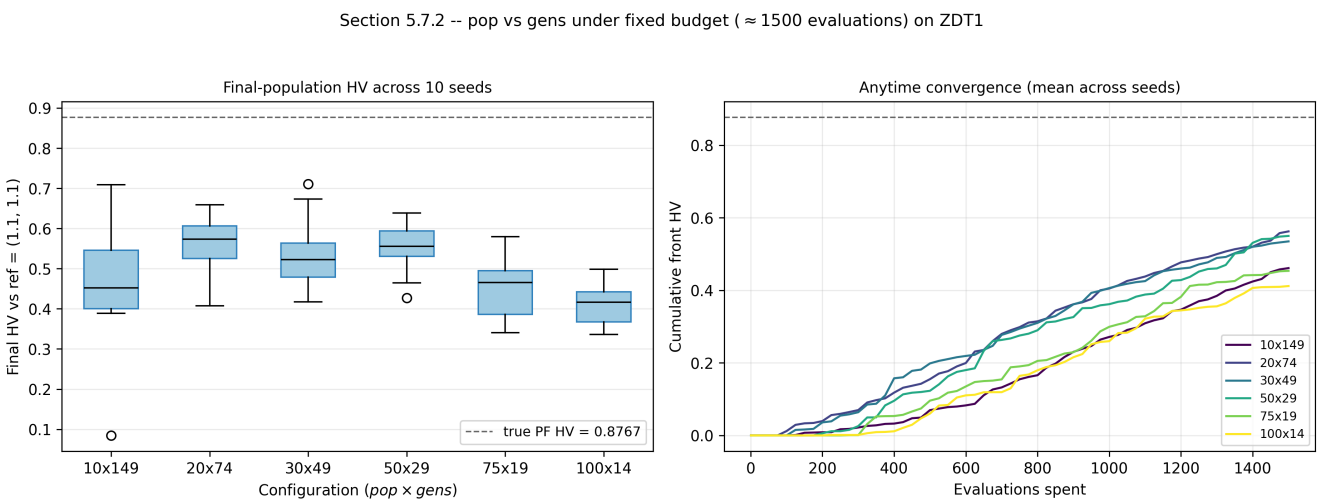

Figure 10: Population vs generations under fixed budget on ZDT1. The left plot shows boxplots of final hypervolume across 10 seeds per(population, generations) pair. The right plot shows the mean anytime convergence trace.

Table 9: Trade-off between NSGA-II population size and generation count under a fixed total budget of ≈ 1500 objective evaluations, on the ZDT1 benchmark. True Pareto-front hypervolume 0.8767. Each row aggregates 10 independent seeds with a fixed $\eta_c = 15$ and $p_{\text{cross}} = 0.9$.

pop	gens	budget	HV mean	HV std	HV / true PF	—front— mean
10	149	1500	0.4614	0.1726	0.526	50.5
20	74	1500	**0.5626**	0.0717	0.642	62.0
30	49	1500	0.5350	0.0963	0.610	68.0
50	29	1500	0.5501	0.0663	0.627	62.1
75	19	1500	0.4538	0.0801	0.518	40.7
100	14	1500	0.4119	0.0527	0.470	33.7

5.7.3 Convergence Diagnostics

For this section the synthetic-benchmark reference convergence on ZDT1 with the real-problem convergence on the aFRR price up search is used. Within figure 11, there are two panels deliberately on different scales; ZDT1 has known true Pareto front so HV ratio is meaningful. The real problem does not have a known front so only the absolute trajectory is plotted. On ZDT1, the algorithm reaches 60.8% of the true Pareto front hypervolume after 50 generations. The algorithm exhibits the gradual-improvement curve on a smooth multi-objective landscape. On the real PatchTST problem the cumulative HV moves from an initial 0.508 to 0.511 within 5 generations. Therefore the random initial population already captures most of the Pareto-relevant region, thus the NSGA-II's contribution over the remaining generations is to spread the front rather than push it closer to any optimum. Thus it is evident that a 20-sample random initialization is competitive with full NSGA-II procedure on objective-value terms.

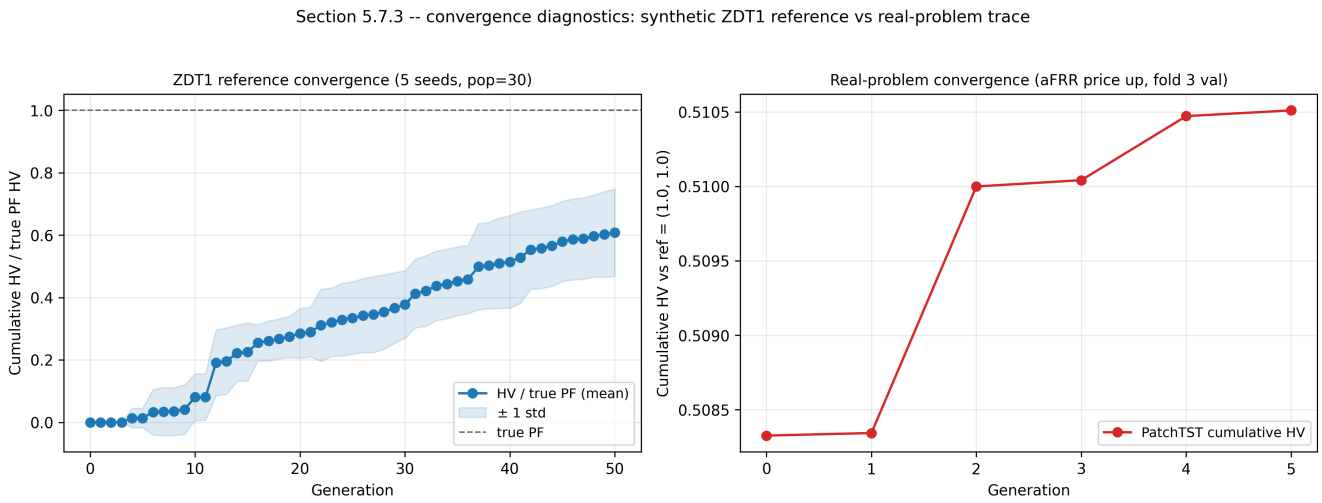

Figure 11: The left plot shows: ZDT1 reference HV / true PF HV per generation, mean ± std across 5 seeds. On the other side: the real-problem absolute cumulative HV per generation for the PatchTST aFRR price up search is visible.

Table 10: The graph shows the per-generation convergence on the synthetic ZDT1 benchmark aggregated across 5 seeds with population =30. On the real aFRR price up problem a single run per model can be seen below. For ZDT1, HV is reported both absolute and as a fraction of the analytical Pareto front HV, 0.8767 against reference $(1.1, 1.1)$. Real-problem HV is absolute against reference $(1.0, 1.0)$, in other words seasonal-naive accuracy and worst-possible calibration. the true Pareto front is unknown so no ratio is meaningful.

Series	gen	HV	HV / true PF	—front—
ZDT1 (synthetic reference)				
	0	0.0000	0.000	10.2
	1	0.0000	0.000	6.4
	2	0.0000	0.000	5.8
	3	0.0000	0.000	6.2
	4	0.0122	0.014	5.6
	5	0.0122	0.014	6.8
	6	0.0287	0.033	7.4
	7	0.0301	0.034	7.6
	8	0.0306	0.035	8.0
	9	0.0360	0.041	8.4
	10	0.0707	0.081	10.2
	11	0.0712	0.081	14.2
	12	0.1676	0.191	14.8
	13	0.1720	0.196	15.4
	14	0.1944	0.222	20.2
	15	0.1977	0.225	23.0

Continued on next page

Table 10 – *Continued from previous page*

Series	gen	HV	HV / true PF	—front—
	16	0.2237	0.255	25.8
	17	0.2289	0.261	27.6
	18	0.2347	0.268	26.8
	19	0.2405	0.274	28.8
	20	0.2503	0.285	27.6
	21	0.2539	0.290	30.4
	22	0.2730	0.311	30.8
	23	0.2816	0.321	29.4
	24	0.2886	0.329	29.6
	25	0.2935	0.335	29.8
	26	0.3000	0.342	26.4
	27	0.3030	0.346	24.8
	28	0.3100	0.354	27.4
	29	0.3216	0.367	28.0
	30	0.3310	0.378	27.4
	31	0.3619	0.413	28.4
	32	0.3692	0.421	28.6
	33	0.3836	0.438	31.6
	34	0.3889	0.444	33.4
	35	0.3968	0.453	33.2
	36	0.4013	0.458	35.6
	37	0.4375	0.499	39.8
	38	0.4406	0.503	43.2
	39	0.4472	0.510	45.4
	40	0.4510	0.514	47.4
	41	0.4633	0.528	44.2
	42	0.4856	0.554	49.6
	43	0.4894	0.558	51.8
	44	0.4958	0.565	54.4
	45	0.5085	0.580	58.0
	46	0.5141	0.586	59.4
	47	0.5164	0.589	60.6
	48	0.5236	0.597	61.6
	49	0.5281	0.602	66.0
	50	0.5331	0.608	70.4
aFRR price up, PATCHTST (single run)				
	0	0.5083	–	2
	1	0.5083	–	3
	2	0.5100	–	4
	3	0.5100	–	7
	4	0.5105	–	6
	5	0.5105	–	7

5.7.4 Random Search Baseline Comparison

Figure 12 and table 11 compare NSGA-II against uniform-random sampling at a budget of $1N = 500$. The fixed parameters for NSGA-II are pop $= 30$, gens $= 49$, $\eta_c = 15$. The from-scratch NSGA-II algorithm achieved a mean HV of 0.535 while random search achieved 0.000. The random-search HV is zero because ZDT1's $g(\mathbf{x}) = 1 + 9\,\overline{x_{2..n}}$ concentrates near 5.5 for $n = 30$ uniformly-random samples. Therefore $f_2 \geq g(1 - \sqrt{1/g}) \approx 3.16$ at every random draw – every random sample lies past the reference point $(1.1, 1.1)$ and contributes no dominated volume. To conclude the ablation studies it is clear that the NSGA-II manages to find the Pareto front through its non-dominated sort allowing for g-minimization. This motivates the use of NSGA-II for the real-problem search even if the per-generation improvement discussed in section 5.7.3 is subtle.

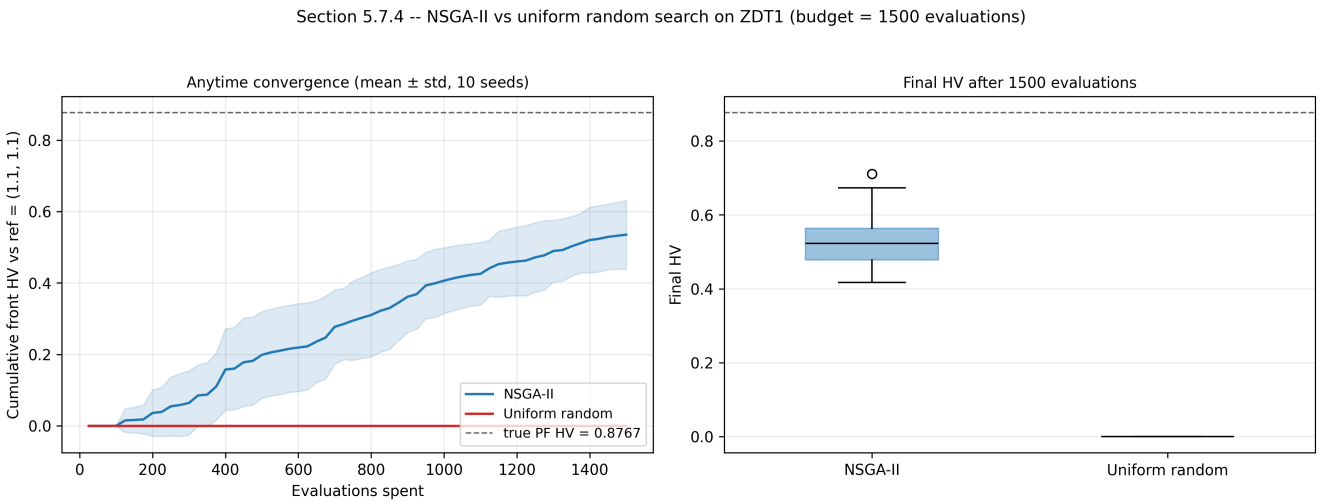

Figure 12: NSGA-II vs uniform random search on ZDT1. The left graph portrays the mean ± std anytime convergence over 1500 evaluations, 10 seeds each. On the other side the boxplot of final HV per method is visible. Random search lies on the $y = 0$ axis because every random sample's f_2 exceeds the reference point's threshold.

Table 11: NSGA-II versus uniform-random sampling on the ZDT1 benchmark at matched evaluation budget, $N = 1500$. Each row aggregates 10 independent seeds. NSGA-II uses $\eta_c = 15$, $p_{cross} = 0.9$, pop $= 30$, gens $= 49$. Two-sided Mann-Whitney U on the seed-level HV distributions: $p = 6.39 \times 10^{-5}$. ZDT1's auxiliary function $g(\mathbf{x}) = 1 + 9\,\overline{x_{2..n}}$ concentrates near 5.5 for $n = 30$ uniformly-random samples, so $f_2 \geq g \cdot (1 - \sqrt{1/g}) \approx 3.16$ at every random draw – every random sample lies outside the reference-bounded region $[0, 1.1) \times [0, 1.1)$, giving an empirical HV of exactly zero. The selection pressure NSGA-II applies on g-minimisation is what lets it find the Pareto front at all in this dimension. The contrast is therefore not subtle.

Method	HV mean	HV std	HV / true PF	—front— mean	wall (s)
NSGA-II	**0.5350**	0.0963	0.610	68.0	0.27
Uniform random	0.0000	0.0000	0.000	20.3	0.10

6 Experimental Setup

6.1 Walk-forward CV Protocol

The experimental evaluation of the models follows the expanding-window cross-validation protocol explained in section 3.8, table 4 and figure 7. This protocol serves two distinct purposes in this study; first, the final model evaluation. During evaluation, the models – seasonal-naive, LightGBM, Temporal Fusion Transformer, and PatchTST – are scored based on the reported performance metrics averaged across the three folds. This allows for a robust out-of-sample assessment. The second reason for the expanding-window cross-validation protocol is the hyperparameter optimization performed by NSGA-II which takes place only on one fold for computational tractability. The resulting Pareto-optimal configuration are re-evaluated across all three folds for the comparative analysis that is presented in section 7.1. At each 15 minute interval within the test window, a 16 step forecast trajectory is generated. Finally, the metrics are computed per horizon and then finally aggregated to put the model's performance into metrics. The models differ in how the 16 horizons are generated due to the differences in architecture; the transformers using one forward-pass, the LightGBM iterates through horizons and the seasonal naive performs lookup by horizon, as described in section 4.1.

6.2 Evaluation Metrics

In this study two complementary families of metrics are used to evaluate model performance: point forecasting metrics and probabilistic forecasting metrics. Point metrics can be reported for all models, including the baseline and transformer models, while the probabilistic metrics apply only to the transformer models. The point forecasting performance is measured through 3 distinct metrics all reflecting performance in distinct ways, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Scaled Error (MASE). MAE and RMSE are straightforward, and MASE is calculated by scaling the model's absolute forecast error by the in-sample mean absolute error of the daily seasonal-naive benchmark using a lag of 96 observation, as introduced by section 4.1.1. Therefore the MASE score can easily be interpreted; a value below 1 is indicative of that the model outperform the seasonal-naive baseline. In regards to the transformer model comparisons, probabilistic forecast quality is evaluated using pinball loss. It is averaged across the $\{0.1, 0.5, 0.9\}$ quantiles and across horizons, along with the empirical coverage of the nominal 80% prediction interval $[P10, P90]$. To measure calibration, the calibration error metric is used which is defined by the absolute deviation between observed interval coverage and the target value of 0.80. As described in section 5.5.2, this metric is also used in NSGA-II as one of the objectives. Since the transformer models output a probability distribution the median is taken as the point forecast, allowing for the computation of both point and probabilistic metrics. A complete summary and definitions is provided in table 12

Table 12: All the evaluation metrics' definition used in this study. All metrics are computed per horizon h and then averaged across the three cross-validation folds where applicable. Point metrics apply to every model, for a probabilistic model the point forecast is its median. Probabilistic metrics apply only to the transformer models.

Metric	Definition	Interpretation
Point metrics		
MAE	$\frac{1}{n}\sum_t \lvert y_t - \hat{y}_t \rvert$	Mean absolute error, in original units (HUF/MWh)
RMSE	$\sqrt{\frac{1}{n}\sum_t (y_t - \hat{y}_t)^2}$	Penalises large errors more heavily than MAE
MASE	$\frac{\text{MAE}}{Q}$	Error scaled by the seasonal-naive benchmark Q; MASE < 1 beats the daily seasonal-naive
Probabilistic metrics		
Pinball loss	$\frac{1}{\lvert\mathcal{Q}\rvert}\sum_{q\in\mathcal{Q}} \frac{1}{n}\sum_t \rho_q\left(y_t - \hat{y}_t^{(q)}\right)$	Proper score for quantile forecasts; averaged over $\mathcal{Q} = \{0.1, 0.5, 0.9\}$
Coverage_{80}	$\frac{1}{n}\sum_t \mathbf{1}\left[\hat{y}_t^{(0.1)} \le y_t \le \hat{y}_t^{(0.9)}\right]$	Empirical coverage of the 80% prediction interval; nominal target 0.80
Calibration error	$\lvert\text{Coverage}_{80} - 0.80\rvert$	Deviation from nominal coverage; one of the NSGA-II objectives (§5.5.2)

Notation: y_t is the observed value and $\hat{y}_t$ the point forecast at forecast point t; $\hat{y}_t^{(q)}$ is the predicted q-quantile; n is the number of forecast points at the horizon. The scaling factor $Q = \frac{1}{N-m}\sum_{t=m+1}^{N} \lvert y_t - y_{t-m} \rvert$ is the in-sample mean absolute error of the seasonal-naive forecast at the daily lag $m = 96$. The pinball loss function is $\rho_q(u) = \max(qu, (q-1)u)$, and $\mathbf{1}[\cdot]$ is the indicator function.

6.3 Statistical Testing

During a comparison of forecasting models, differences may arise in the collected metrics, described in section 6.2. Some of these differences are due to sampling variability rather than genuine performance differences; thus statistical significance testing is necessary before concluding whether one model outperforms the other. To perform statistical significance tests this study will employ the Diebold-Mariano test for pairwise comparison of forecasting models [Diebold and Mariano, 1995]. In essence, the test evaluates whether the differences in metrics are significant through an analysis of the time series of forecast-loss differentials. Formally, given two models with forecast errors $e_{1,t}$ and $e_{2,t}$, the loss differential is defined by the equation:

$$d_t = L(e_{1,t}) - L(e_{2,t}), \qquad \texttt{where } L(\cdot) \texttt{ is a chosen loss function} \tag{14}$$

The null hypothesis states that the expected loss differential satisfies $E[d_t] = 0$, suggesting that there is no difference in predictive performance between two models. The Diebold-Mariano test examines the null hypothesis while accounting for serial autocorrelation in the loss differential sequence [Diebold and Mariano, 1995]. This is crucial for multi-step forecasting settings. To examine the performance of the model at different horizons, the test will be conducted at each forecast horizon using test-fold forecast errors. The comparisons will include PatchTST vs TFT and the transformer models compared to the baseline models for both tuned and default model configurations. Due to a

tendency of the Diebold-Mariano test to reject the null hypothesis in small-sample environments, the Harvey–Leybourne–Newbold small sample correction is utilized for fair evaluation of significance [Harvey et al., 1997]. The Harvey–Leybourne–Newbold is a small sample scaling correction that improves the test's size in finite samples. To control false positives that might arise from multiple hypothesis tests across model pairs and horizons, a family-wise error correction, the Holm-Bonferroni will be applied [Abdi, 2010]. A limitation of the statistical test includes an assumption that there are covariance-stationary loss differentials. Therefore, structural break or strong non-stationarity in the evaluation period may affect validity.

6.4 Computational Infrastructure

All experiments in this study were performed on a personal computer equipped with a NVIDIA GeForce RTX 5080 GPU equipped with 16GB of GDDR7 GPU memory. Regarding the CUDA environment, CUDA 12.8 was used with all the PyTorch, Lightning, PyTorch Forecasting and LightGBM dependencies compiled for the CUDA 12.8 runtime. To ensure ease of use all the packages are handled using uv. To allow efficient computing on Windows 11, all experiments were conducted in Windows Subsystem for Linux 2 (WSL2) through the Ubuntu distribution. To utilize resources to the maximum and ensure efficiency the experiments used `bfloat16 mixed-precision training` along with multi-worker data loading. Even with all these efficiency measures due to the complex architecture of the Temporal Fusion Transformer, training time per trial can exceed 10 minutes thus the cost of full multi-fold evolutionary tuning would be unrealistic. This further motivates the decision for the NSGA-II hyperparameter optimization is performed on the validation window of a single fold, as described in sections 6.1 and 5.5.3. A complete hardware summary can be found in the table below:

Table 13: Hardware and software environment used for all experiments. Transformer training uses bf16 mixed precision on the GPU.

Component	**Specification**
Hardware	
CPU	Intel Core i9-14900K
GPU	NVIDIA GeForce RTX 5080 (16 GB GDDR7)
NVIDIA driver	610.47 (CUDA 13.3)
System memory	64 GB
Software	
Host operating system	Windows 11
Runtime environment	WSL2 (Ubuntu)
Python	3.12.9
Environment manager	uv 0.11.15
PyTorch	2.11.0 (CUDA 12.8 build)
Lightning	2.6.1
pytorch-forecasting	1.7.0
LightGBM	4.6.0
NumPy / pandas	2.4.4 / 2.3.3

6.5 Limitations to Complete Reproducibility

Although the necessary steps to achieve reproducibility described in section 5.6.3 it is important to acknowledge the limitations for 1:1 reproducibility. This is due to the residual non-determinism in certain GPU and CUDA operations, along with the minor variability introduced by parallel worker scheduling. However, the pipeline still remains functionally reproducible, with negligible differences between different runs and hardware. To support transparency and contributions, the full implemented code, experiment, configurations, and publicly available datasets will be made available in the appendices.

7 Results

7.1 Baseline Results

Upon training both the LightGBM and seasonal-naive baseline models, there was a clear difference in prediction accuracy. As expected, the seasonal-naive model was significantly outperformed by the boosted regression tree model, likely due to the ability of LightGBM to use the feature space. As shown in table 14 the difference in mean absolute error, MAE is significant with the LightGBM model achieving a 46 thousand MAE while the seasonal naive achieved a score of 71 thousand MAE at the 15 minute horizon. That is a $\approx 38\%$ improvement for the LightGBM model. At the other end of the horizon spectrum at 4 hours, the LightGBM model still performed significantly better with an MAE of 47.7 thousand compared to the 71 thousand of the seasonal naive, achieving a $\approx 33\%$ reduction in MAE.

Table 14: Test-set MAE for the seasonal-naive baselines and LightGBM on the aFRR price up target, averaged across the three walk-forward folds.

Target	**Model**	**15 min**	**1 h**	**2 h**	**3 h**	**4 h**
aFRR price up (HUF/MWh)	Naive (day)	71 011.2	71 039.2	71 026.8	71 018.6	71 042.8
	Naive (week)	71 524.1	71 527.7	71 543.2	71 574.9	71 593.8
	LightGBM	**46 204.5**	**47 629.0**	**47 684.7**	**47 712.8**	**47 733.2**

As mentioned in section 6.3, the Diebold-Mariano test needs to be conducted on the model statistics to conclude an actual meaningful difference. Since the baseline models do not have a calibration statistic due to their point estimate output, the MAE is used for comparison. As shown in table 15, the difference between the LightGBM and seasonal-naive is most likely meaningful, and therefore it can be concluded that the LightGBM model is more appropriate for time-series forecasting. This was expected as the seasonal-naive model is one of the simplest models and thus is a useful baseline, however, it lacks the ability to extract information from the feature space. The visualization of the differences within the MAE of the models can be seen in figure 13. When comparing the weekly and daily seasonal-naive models, the difference is found to be negligible, shown by the p-value of 1 in table 15.

Table 15: Diebold-Mariano test on the absolute-loss differential for the aFRR price up target, with the Harvey–Leybourne–Newbold small-sample correction. p-values are Holm–Bonferroni-adjusted within the target a family size $m = 15$ is used, three model pairs × five horizons. Bold p-values are significant at $\alpha = 0.05$ after FWER correction. The first-listed model has lower absolute loss in every comparison shown, so a significant p-value indicates that the first model is significantly more accurate than the second.

Target	Comparison	15 min	1 h	2 h	3 h	4 h
aFRR price up (HUF/MWh)	LightGBM vs Naive (day)	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
	LightGBM vs Naive (week)	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
	Naive (day) vs Naive (week)	1.000	1.000	1.000	1.000	1.000

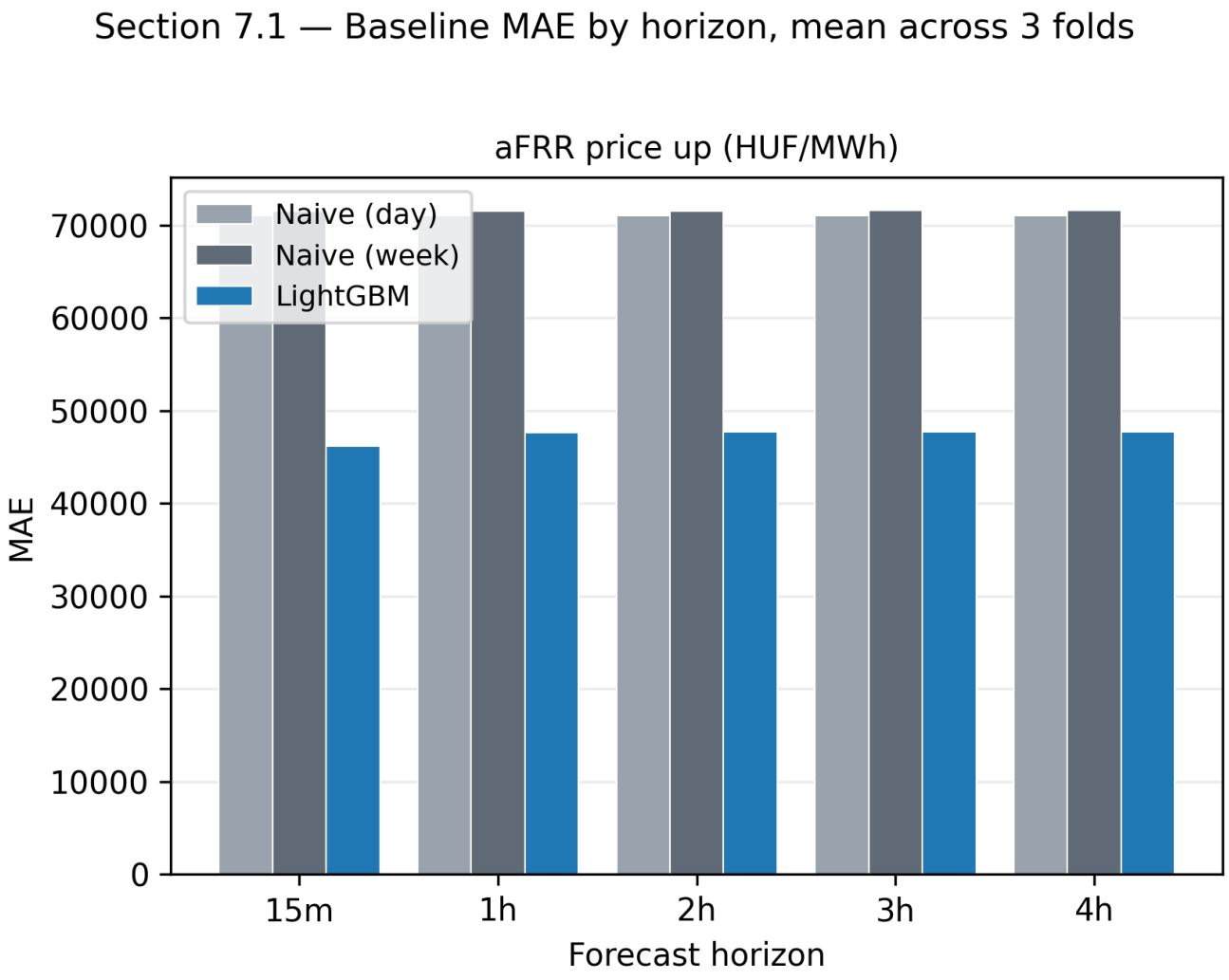

Figure 13: Test-set MAE of the two seasonal-naive baselines (*naive_day*, lag 96; *naive_week*, lag 672) and *LightGBM*, broken down by forecast horizon. Each bar is the mean across the three walk-forward folds.

7.2 Default-config Transformers

With a non-optimized hyperparameter space, the transformer models were already competitive by outperforming the LightGBM model at the 15m horizon, however, they have lacked behind over longer horizons. As shown in figure 14, the PatchTST was almost equal with the LightGBM for the other key horizons, while the TFT model showed worse performance.

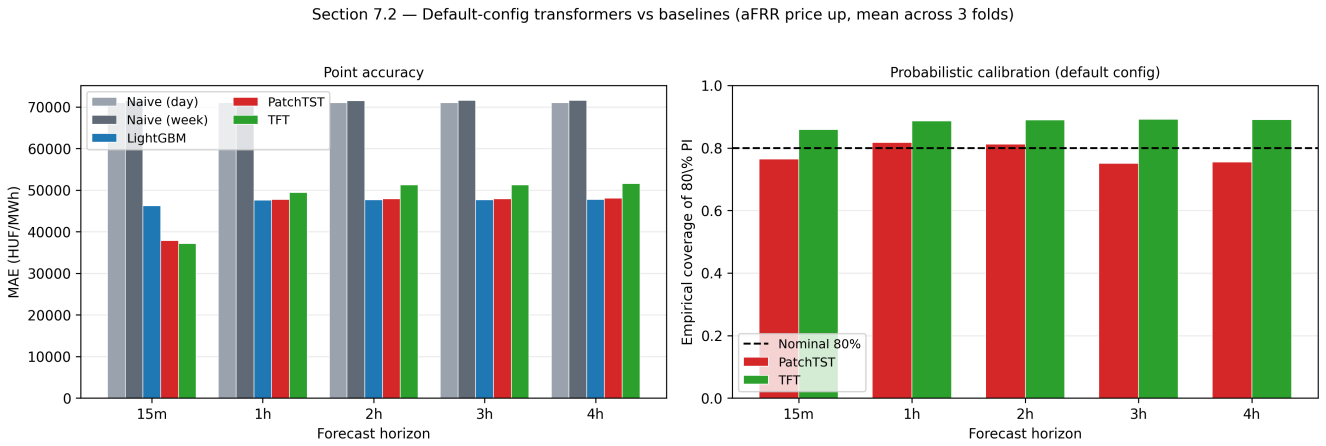

Figure 14: Default-config transformer performance on the aFRR price up target. The left graph shows the test-set MAE by forecast horizon for the two seasonal-naive baselines, LightGBM, and the two transformer models with default configurations. The right chart shows empirical coverage of the nominal 80% prediction interval for the two transformers.

Since the two transformer models have a probabilistic output it allows for comparison of the probabilistic metrics between the two out of the box transformers. In table 17 below, the calibration objective was measured between the models. It is important to note that the MAE and pinball loss are not the same metric as the pinball loss calculated using the formula:

$$\text{pinball} = \frac{1}{3} \sum_{q \in \{0.1,\, 0.5,\, 0.9\}} \frac{1}{N} \sum_{t} \rho_q\left(y_t - \hat{y}_t^{(q)}\right) \quad \text{where} \quad \rho_q(u) = \max\big(qu,\ (q-1)u\big) \tag{15}$$

Looking at the calibration errors, its clear that the PatchTST by default is calibrated better while the TFT model is poorly calibrated with errors almost reaching 0.1. These calibration errors further motivated the use of NSGA-II for hyperparameter tuning.

Table 16: Test-set MAE on the aFRR price up target for the two seasonal-naive baselines, LightGBM, and the two transformer models in their default configuration. Mean across the three walk-forward folds

Model	15 min	1 h	2 h	3 h	4 h
Naive (day)	71 011.2	71 039.2	71 026.8	71 018.6	71 042.8
Naive (week)	71 524.1	71 527.7	71 543.2	71 574.9	71 593.8
LightGBM	46 204.5	**47 629.0**	**47 684.7**	**47 712.8**	**47 733.2**
PatchTST	37 846.1	47 717.5	47 926.7	47 924.6	48 037.3
TFT	**37 194.7**	49 414.0	51 297.1	51 297.7	51 597.4

Table 17: Probabilistic metrics for the two default-config transformers on the aFRR price up target. The pinball loss is averaged over the $\{0.1, 0.5, 0.9\}$ quantiles. The metrics are averaged across the three walk-forward folds. A perfectly calibrated forecast has coverage error 0.

Model	Metric	15 min	1 h	2 h	3 h	4 h
PatchTST	Pinball loss	14 158.9	17 389.1	18 055.1	17 861.6	18 507.2
	coverage_{80}	0.765	0.818	0.813	0.751	0.755
	Calibration error	0.035	0.018	0.013	0.049	0.045
TFT	Pinball loss	13 335.5	16 420.3	16 969.7	16 979.9	17 061.9
	coverage_{80}	0.859	0.886	0.890	0.892	0.890
	Calibration error	0.059	0.086	0.090	0.092	0.090

7.3 NSGA-II Tuning

7.3.1 Pareto fronts

The evolutionary search algorithms for the Transformer models produced a 4 and 3-point Pareto front; both are clean monotone trade-off curves. As visible in figure 15, the PatchTST spans MASE 0.479-0.482 and calibration error surprisingly low at only 0.032-0.090. The trade-off is clear with a reduction in MASE resulting in calibration improvement. The second model, TFT, depicted the same trade-off with MASE spanning 0.469-0.478 and calibration error from 0.018 to 0.082, thus the model was strictly better then the PatchTST model on both extremes. However, it is important to note that the TFT is much more structurally complex and the training time was $\approx 4 - 5x$ of the PatchTST model. Moreover, as shown in table 18, the NSGA-II algorithm was able to improve on all extremes, proving that the algorithm is appropriate for an optimization task. All four extremes are found after the first two generation, depicting the success of the EA.

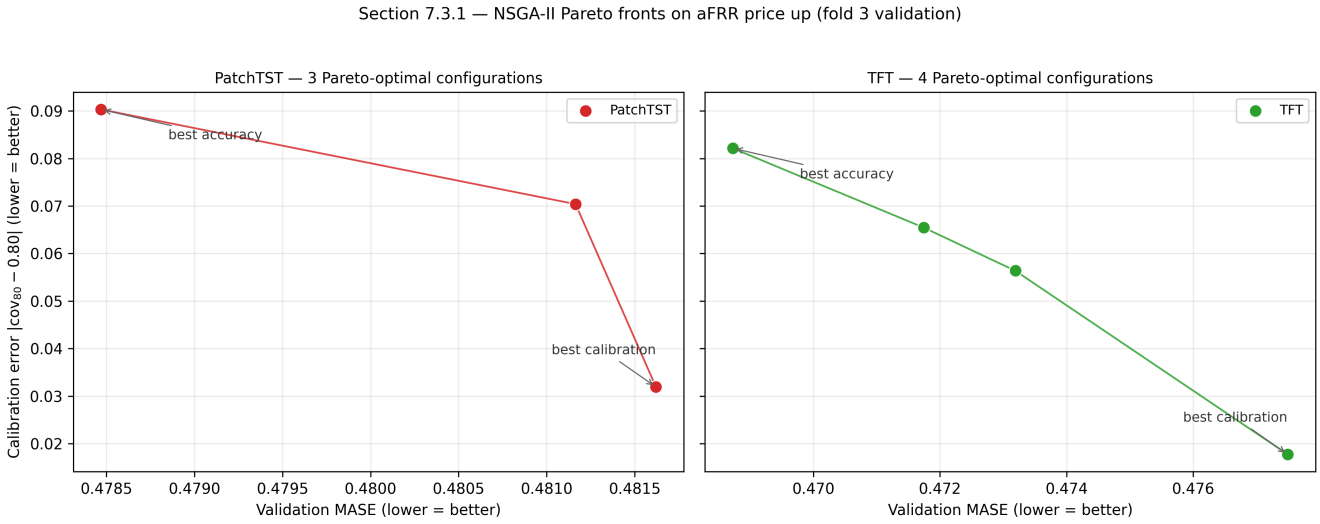

Figure 15: NSGA-II Pareto fronts on the aFRR price up target, evaluated on the fold-3 validation window. The left graph shows the PatchTST front (3 configurations), spanning validation MASE 0.479–0.482 and calibration error 0.032–0.090, while the right graph portrays the TFT front (4 configurations), spanning MASE 0.469–0.478 and calibration error 0.018–0.082

Table 18: Pareto-optimal hyperparameter configurations discovered by the NSGA-II search on the aFRR price up target. Objectives are evaluated on the validation window of fold 3. *eval_id* is the global trial identifier and *generation* is the generation in which the configuration was first evaluated.

Model	**eval id**	**gen**	**MASE**	**Cal. error**
PatchTST	56	2	0.478	0.090
	8	0	0.481	0.070
	72	3	0.482	0.032
TFT	104	5	0.469	0.082
	59	2	0.472	0.065
	43	2	0.473	0.056
	113	5	0.477	0.018

7.3.2 Convergence over generations

Figure 16, and table 19 portray the per-generation NSGA-II diagnostics for the two transformer models on fold-3 validation. There are three distinct noticeable patterns starting with the hypervolume growth for both models. PatchTST showed a 4.6% increase over 5 generations while the TFT model also exhibited an increase of 6.9% in hypervolume. The growth was monotone for both models with a single plateau for each. Thus it is clear that the NSGA-II algorithm is not just spreading the solutions along the Pareto front but also pushing the front forward like it is supposed to. The second pattern arises from Pareto front size growth. The front grows from 1 Pareto-optimal solution to 3 for PatchTST, and for TFT it grows from 2 to 4 Pareto-optimal solutions. This clearly highlights the crowding-distance operator and its succession in spreading the solutions along the front along the hypervolume push. The last pattern is within accuracy and calibration. While accuracy smoothly improves between generations, calibration rather shows step-jumps. This suggests that the MASE is well-learned and optimized while the calibration objective is dominated by lucky candidate solutions within new generations.

Table 19: Per-generation NSGA-II convergence diagnostics on the aFRR price up target. *evaluated* is the cumulative number of distinct trials with finite objectives; *front* is the size of the non-dominated front taken over all evaluations so far; *HV* is the 2-D hypervolume of that cumulative front against the fixed reference point $(1.0, 1.0)$. *Best MASE* and *best cal. err.* are the best single objective values seen up to and including the generation.

Model	gen	evaluated	front	HV	best MASE	best cal. err.
PatchTST	0	20	1	0.4823	0.481	0.070
	1	40	1	0.4823	0.481	0.070
	2	60	2	0.4848	0.478	0.070
	3	80	3	0.5047	0.478	0.032
	4	100	3	0.5047	0.478	0.032
	5	120	3	0.5047	0.478	0.032
TFT	0	20	2	0.4877	0.479	0.064
	1	40	4	0.5037	0.476	0.039
	2	60	3	0.5078	0.472	0.039
	3	80	3	0.5078	0.472	0.039
	4	100	3	0.5078	0.472	0.039
	5	120	4	0.5214	0.469	0.018

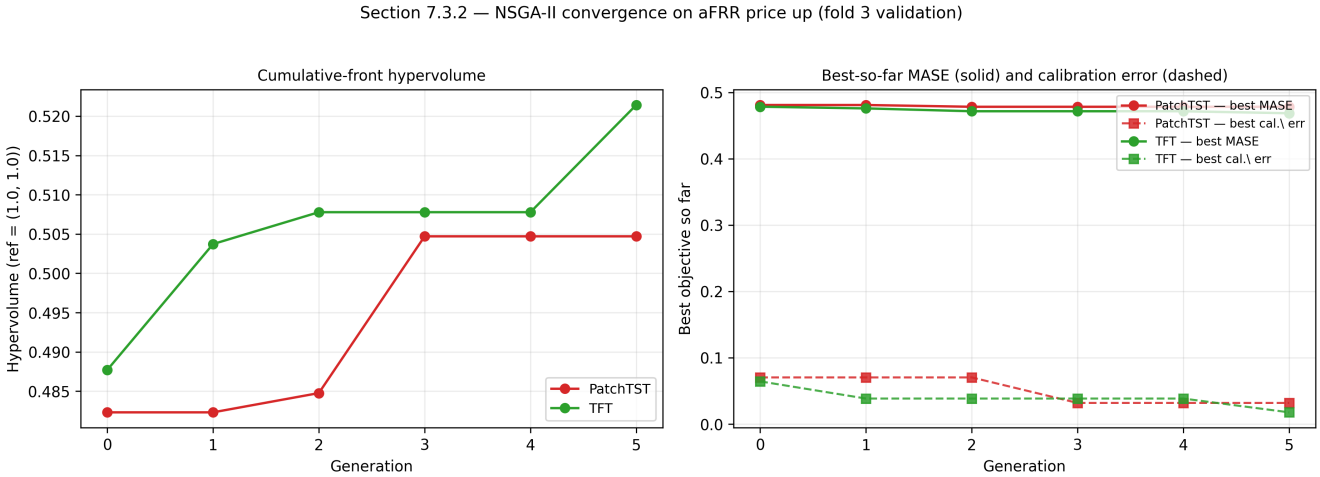

Figure 16: Per-generation NSGA-II convergence on the aFRR price up target, fold-3 validation, for both transformer searches. The left graph shows the cumulative-front hypervolume against reference point $(1.0, 1.0)$, which is the worst-case point (MASE = 1 matches the seasonal-naive baseline; calibration error = 1 is maximum miscalibration). PatchTST grows from 0.482 to 0.505 (+4.6%) and TFT from 0.488 to 0.521 (+6.9%) over five generations. The right graph illustrates the best-so-far MASE (solid) and best-so-far calibration error (dashed) per generation. MASE improves smoothly across generations for both models (PatchTST 0.481 → 0.479, TFT 0.479 → 0.469), whereas calibration error improves in discrete step-jumps – PatchTST drops from 0.070 to 0.032 at generation 3, TFT from 0.064 to 0.018 at generation 5 – consistent with the non-proper-objective dynamics discussed in Section 7.3.3.

7.3.3 Tuned vs default head-to-head

After the Pareto front was given by the NSGA-II tuning, the two extreme configurations per model; best accuracy and best calibration, were chosen and evaluated on the test set of fold 3, since the validation set of fold 3 was used for tuning too. The extremes for each model were chosen, and the results can be found in figure 17 and table 20. From the graph it is clear that the MASE between the best calibration and best accuracy models is very similar and the tuning shows a clear gain in accuracy within the TFT network. For calibration the gain from the tuning is not very apparent and will be discussed in section 8.2.

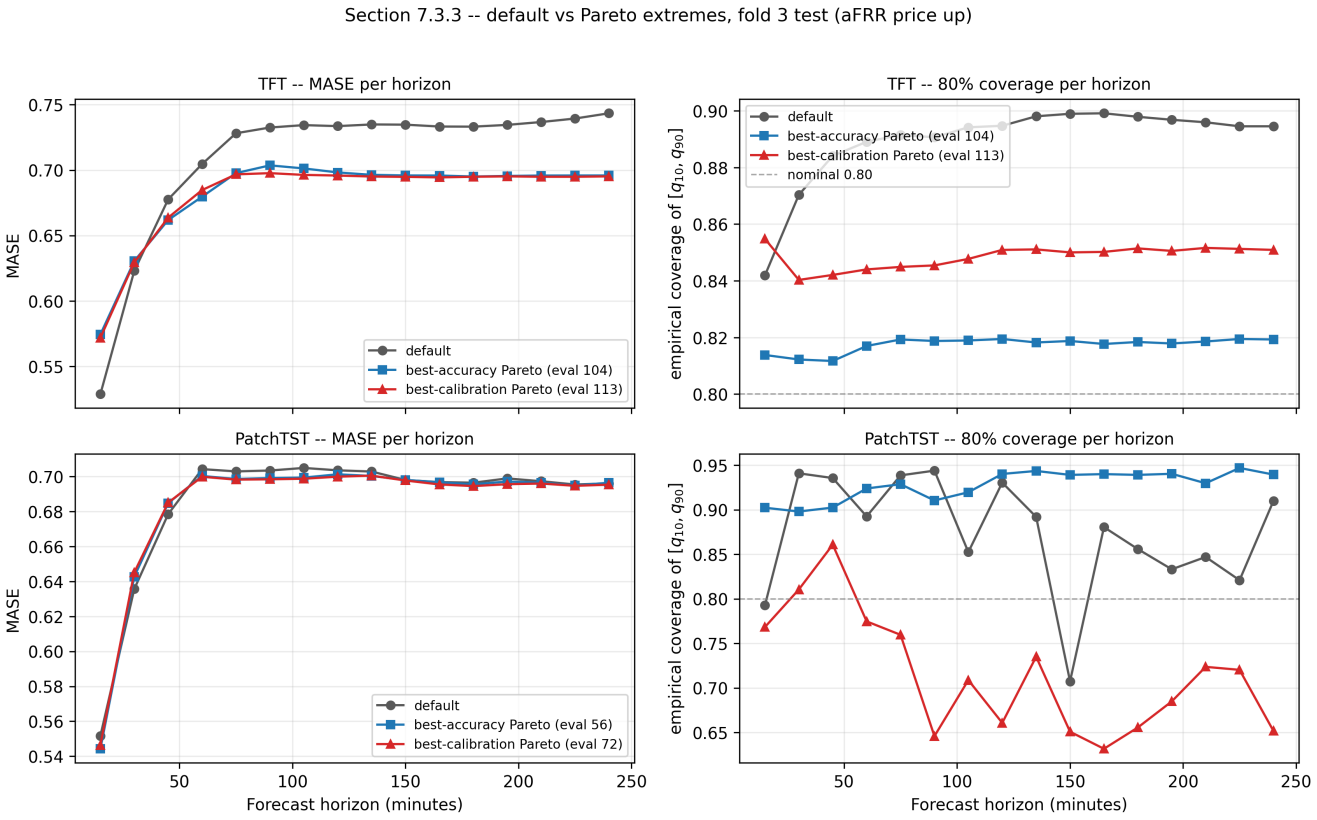

Figure 17: Per-horizon MASE, left column, and empirical 80% coverage, right column, on fold 3 test for the default configuration and the two NSGA-II Pareto extremes, on the aFRR price up target. Top row shows the TFT model, while the bottom row portrays the PatchTST model. The dashed grey line in each coverage panel marks the nominal level 0.80.

Table 20: Head-to-head on the aFRR price up target between the default configurations and the two NSGA-II Pareto-front extremes (best-accuracy and best-calibration). All six rows report metrics on the same window, fold 3 test, averaged over horizons $\{15\,\text{min}, 1\,\text{h}, 2\,\text{h}, 3\,\text{h}, 4\,\text{h}\}$. NSGA-II selected the extremes on fold 3 *validation*, so the test window is held out for the tuned configurations.

Model	Configuration	eval id	MASE	cov_{80}	Cal. error
PatchTST	default	–	0.670	0.876	0.079
	best-accuracy Pareto	56	0.668	0.929	0.129
	best-calibration Pareto	72	0.667	**0.702**	**0.098**
TFT	default	–	0.689	0.884	0.084
	best-accuracy Pareto	104	0.669	0.818	0.018
	best-calibration Pareto	113	0.668	**0.850**	**0.050**

7.4 Cross-model comparison

Figure 18 portrays the difference between the baseline models and the tuned vs defaul transformers. There is a surprising pattern with the transformers outperforming the baseline models at the 15 minute horizon, however, for longer horizons the transformers lose their edge over the LightGBM model. The reason why fold 3 was used for comparison is due to the tuning which used the validation set of fold 3 and using other folds would result in data leakage.
From table 21, both the point-accuracy and the probabilistic metrics are clear. As discussed earlier the default TFT performs the best on the 15 minute horzion with a $\approx 22\%$ improvement from the LightGBM model. The baseline is also slightly outperformed by the tuned TFT on the 1 hour horizon, and finally in the 2-4h horizons, the LightGBM ever so slightly outperforms the tuned TFT model, but with a negligible difference. Looking at the pinball error at every horizon the default TFT performs the best suggesting that the default parameters are very well fitted for the model. It is also interesting to note that the tuned PatchTST with the best accuracy achieves consistently higher calibration error compared to the default transformer.

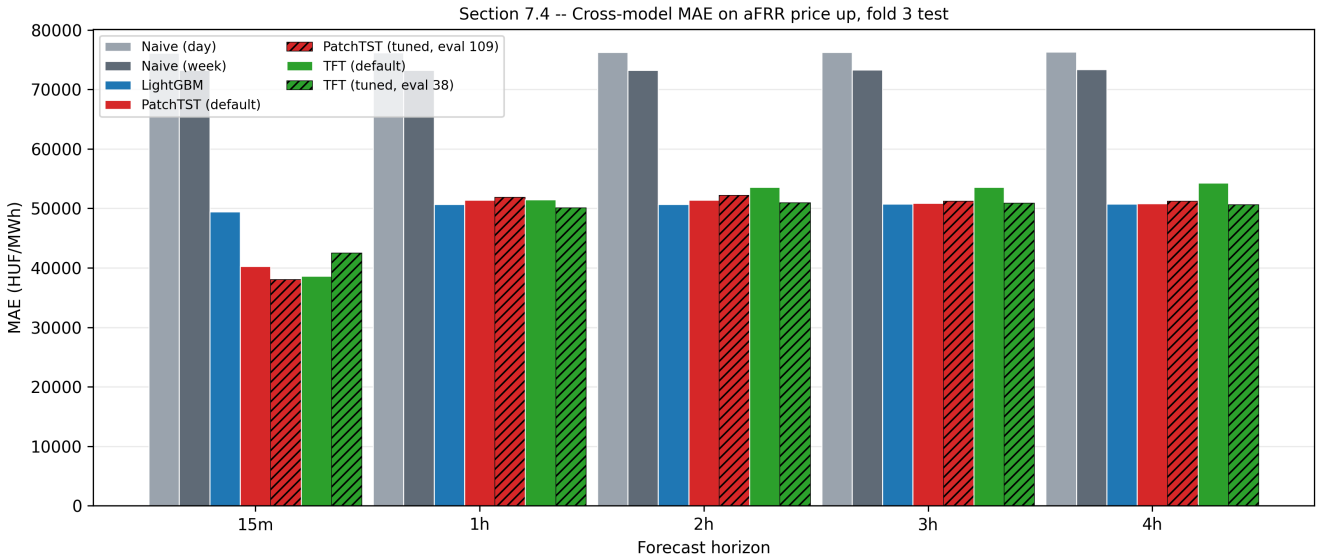

Figure 18: Cross-model test-set MAE on the aFRR price up target. To prevent data leakage the fold 3 test set is used across all models for metrics calculations.

Table 21: Cross-model summary on the aFRR price up target, evaluated on the *fold 3 test window only*. The tuned transformer rows are the best-accuracy NSGA-II Pareto extremes from Table 20, retrained on the fold 3 training window. Like mentioned in figure 18, the metrics were calculated on the test window of fold 3 to prevent data leakage due to the NSGA-II hyperparameter optimization.

Metric	Model	15 min	1 h	2 h	3 h	4 h
MASE	Naive (day)	1.028	1.028	1.029	1.030	1.030
	Naive (week)	0.988	0.989	0.989	0.990	0.990
	LightGBM	0.677	0.694	**0.694**	**0.695**	**0.695**
	PatchTST (default)	0.552	0.704	0.704	0.696	0.696
	PatchTST (tuned), eval 56	0.544	0.700	0.701	0.696	0.696
	TFT (default)	**0.529**	0.705	0.734	0.733	0.743
	TFT (tuned), eval 104	0.574	**0.680**	0.698	0.695	0.696
MAE	Naive (day)	76 123	76 186	76 222	76 277	76 305
	Naive (week)	73 228	73 256	73 244	73 312	73 340
	LightGBM	49 421	50 682	**50 688**	**50 756**	**50 782**
	PatchTST (default)	40 290	51 432	51 382	50 864	50 837
	PatchTST (tuned), eval 56	39 751	51 145	51 216	50 799	50 864
	TFT (default)	**38 617**	51 462	53 579	53 549	54 295
	TFT (tuned), eval 104	41 948	**49 640**	50 993	50 766	50 823
Pinball	PatchTST (default)	14 761	18 569	19 391	19 578	19 476
	PatchTST (tuned), eval 56	14 288	18 244	18 936	18 834	18 870
	TFT (default)	**13 593**	**17 394**	**18 112**	**18 144**	**18 320**
	TFT (tuned), eval 104	15 844	17 921	18 432	18 391	18 379
coverage_{80}	PatchTST (default)	0.793	0.892	0.930	0.856	0.910
	PatchTST (tuned), eval 56	0.902	0.924	0.940	0.939	0.939
	TFT (default)	0.842	0.889	0.895	0.898	0.895
	TFT (tuned), eval 104	0.814	0.817	0.819	0.818	0.819
Cal. err.	PatchTST (default)	**0.007**	0.092	0.130	0.056	0.110
	PatchTST (tuned), eval 56	0.102	0.124	0.140	0.139	0.139
	TFT (default)	0.042	0.089	0.095	0.098	0.095
	TFT (tuned), eval 104	0.014	**0.017**	**0.019**	**0.018**	**0.019**

7.5 Calibration Analysis

Looking at the transformer models it is clear from figure 19 that the default configurations for both models are poor and asymmetric. PatchTST under-covers the 80% calibration range while the TFT model over-covers. As shown in table 22, the PatchTST model achieves a mean coverage of 0.766 across the three folds, while the TFT model achieved an average coverage of 0.887. While as discussed in section 7.3.3, the NSGA-II evolutionary algorithm did manage to close most of the gap it was in an unexpected way. Moreover, while in TFT the gap was closed significantly, in the PatchTST model the calibration seems to be more stochastic and the tuning did not manage to close the gap significantly.

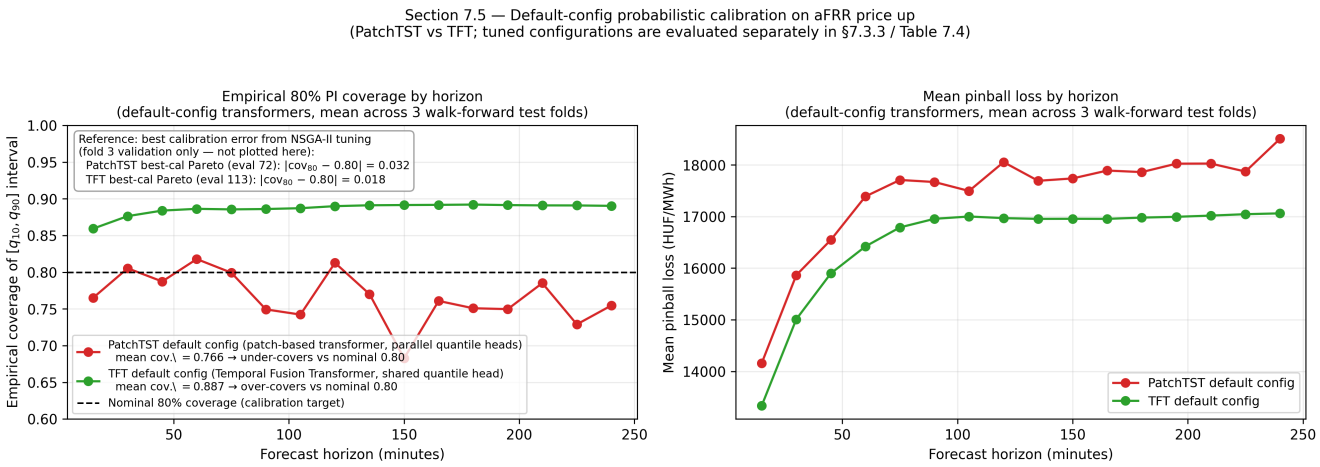

Figure 19: Probabilistic calibration on the aFRR price up target across all 16 test horizons (15-minute increments out to 4 hours), default configuration only. The left graph shows the empirical coverage of the 80% prediction interval against the nominal target. The right graph depicts the mean pinball loss across horizons.

Table 22: Default-config probabilistic calibration summary on the aFRR price up target. Coverage range and means are taken across all 16 test horizons (15 min to 4 h, in 15-minute increments) and the three walk-forward folds.

Model	min cov.	max cov.	mean cov.	mean cal. err.	mean pinball	best Pareto cal. err.
PatchTST	0.683	0.818	0.766	0.095	17 407	0.032 (eval 72)
TFT	0.859	0.892	0.887	0.087	16 522	0.018 (eval 113)

8 Discussion

8.1 PatchTST vs TFT

First examining table 23, of the default configuration transformers and interesting pattern appears. TFT show great performance at the 15-horizon to being significantly outperformed by LightGBM on longer horizons. On the other hand, although the PatchTST model achieves a worse accuracy than the LightGBM model it is never significantly outperformed by the LightGBM model showing stronger performance then the TFT. These results suggest that the TFT's covariate handling pays off less when its default config is suboptimal. The structural differences and their results are discussed in the next paragraph.

Table 23: Diebold-Mariano test on the absolute-loss differential for the aFRR price up target. This table is for the default-config transformer models, three walk-forward folds pooled at the timestamp level. The HLN small-sample correction is applied per horizon and p-values are Holm–Bonferroni-adjusted. Bold p-values are significant at $\alpha = 0.05$ after FWER control. "↓ M1" means model 1 has lower paired MAE on this window (so a significant cell supports the claim that model 1 is more accurate); "↓ M2" is the reverse.

Comparison	**15 min**	**1 h**	**2 h**	**3 h**	**4 h**
TFT vs LightGBM	<**0.001** (↓ M1)	**0.010** (↓ M2)	<**0.001** (↓ M2)	<**0.001** (↓ M2)	<**0.001** (↓ M2)
PatchTST vs LightGBM	<**0.001** (↓ M1)	0.777 (↓ M2)	0.520 (↓ M2)	0.448 (↓ M2)	0.090 (↓ M2)
TFT vs PatchTST	0.333 (↓ M1)	**0.006** (↓ M2)	<**0.001** (↓ M2)	<**0.001** (↓ M2)	<**0.001** (↓ M2)

When looking at the tuned models, although the differences between PatchTST and TFT are tiny they still depict the architectural differences. PatchTST divides the input sequence into patches and applies attention across these patches allowing it to capture recent trends efficiently and learn local seasonality and short-term dependencies. Moreover this patching mechanism allows for less distraction from distant information giving the model an edge on short term forecasting. However as the horizon grows the recent observations become less informative and future calendar effects become the dominant source of information. TFT's strength lies in its handling of static features and known future inputs. Moreover, as the horizons increase variable selection becomes more important since some features might no longer supply meaningful making the TFT more fit for longer horizons. Through variable selection networks and gating mechanisms the TFT is able to more precisely predict the aFRR upward activation price from a longer horizon compared to the PatchTST model. This is also supported by the Diebold-Mariano test conducted in table 24 below indicating that PatchTST outperforms the TFT on the 15 minute horizon but lacks in the longer horizons.

Table 24: Diebold-Mariano test on the absolute-loss differential for the aFRR price up target. Since the two models were tuned on fold 3 the scores are evaluated on the fold 3 test set. HLN small-sample correction is applied per horizon; p-values are Holm–Bonferroni-adjusted across the 5-test family (one model pair × five reporting horizons). Bold p-values are significant at $\alpha = 0.05$ after FWER control. "↓ M1" / "↓ M2" indicate which model has lower paired MAE on this window.

Comparison	**15 min**	**1 h**	**2 h**	**3 h**	**4 h**
TFT (104) vs PatchTST (56)	<**0.001** (↓ M2)	**0.029** (↓ M1)	1.000 (↓ M1)	1.000 (↓ M1)	1.000 (↓ M1)

8.2 NSGA-II efficiency

The from scratch designed NSGA-II algorithm proved useful for transformer tuning. With great effort to adapt it to hyperparameter spaces with different parameters types and restrictions, the algorithm managed to find a clear Pareto optimal solution for both transformers as shown in figure 15. Moreover, it is clear that the model has achieved better solutions throughout generations since all extreme solutions appeared in later generations proving the success of the algorithm. The crowding distance has managed to explore the Pareto front as discussed in section 7.3.2.
It is worth noting the difference in ablation studies and the real-problem convergence. Section 5.7.4 portrays NSGA-II significantly outperforming random search on ZDT1, however as results show in

section 5.7.3, the real problem only experience a small hypervolume growth from 0.508 to a total of 0.511 over the five generations. These results do not pose a contradiction but they do not show the significant advantage of exploring the hypervolume as claimed in section 5.7.4. This is since ZDT1 is a benchmark ablation and has structural properties that allow for more hypervolume growth. This does not carry over to the transformer hyperparameter space, where a randomly sampled configuration can already produce a finite and reasonable solution.
It is important to note that although the clear fronts the calibration objective had a clear issue, since a very low calibration error is attainable for the wrong reasons. The search has many evaluations to find a solution with a low calibration error and due to the validation consisting of two months of data, the evolutionary algorithm tends to overfit and is not able to generalize well. Thus, in the test set of fold 3 which is the subsequent next two months, the model with the greatest calibration on the validation actually performs poorly on the test set, since seasonal patterns and other features that cause the model to overfit are no longer as important in different seasons. Solutions to this problem include increasing the validation set and making the training set less, or using a different objective such as CRPS. A proper objective is one where the forecaster cannot improve their expected score by reporting anything other than their true belief; coverage of a fixed interval fails this because the score depends only on a single quantile rather than the whole distribution [Gneiting and Raftery, 2007].

8.3 Limitations

Throughout the study there has been multiple limitations that had to be overcame. Perhaps the greatest one was that the tuned configurations were only validated on a single fold. This led to the problem of overfitting and decreased generalization over the test set. This decision was made due to the computational requirements of using all three folds for training, and validation. Another limitation is that only the aFRR upwards price was investigated, which is the hardest to predict, however downward activation price and imbalance volume were not investigated and are crucial market factors for balancing. Another limitation was that MAVIR's balancing quantity is not publicly available and therefore the imbalance volume as used as a proxy for that feature. Moreover, the 6 generations of NSGA-II also provided some limitations since only 120 evaluations were performed which limits the accuracy of the hyperparameter optimization.

8.4 Operational Implication

This study used 5 different key horizons to investigate the success of transformers on solving energy imbalances. Each horizon is for different market participants, for example, the 15 minute horizon allows solar powered farms to tweak their production to help minimize the imbalance, while longer horizons allow participants within the manual frequency reserve to react to future imbalances. The probabilistic output further allows for risk evaluation for market participants and risk based planning. The study found that for longer horizons the LightGBM performed equally well compared to the transformers highlighting the intrinsically hard market.

9 Conclusion and Future Work

9.1 Final Contributions

The study conducted in this thesis has provided three distinct contributions. First a clean reproducible benchmarking pipeline for the Hungarian aFRR price forecasting, including the data, splits, metrics, configurations and seeds found in appendix C. Moreover, the paper provides a head-to-head TFT-vs-PatchTST evaluation on a non-stationary balancing-market target, with both point and probabilistic metric evaluations. The differences were also tested through Diebold-Mariano test to show statistical significance. Additionally, an empirical demonstration was provided of NSGA-II with a non-proper calibration objective and its deceptive Pareto fronts. To avoid future errors similar to the on in this paper a solution was provided with reasoning behind it to replace the calibration objective.

9.2 Future Work

To extend this essay there are five different suggestions for future work. First off all the most important one is to run the NSGA-II algorithm with a 3 fold validation averaging allowing for greater generalization and less overfitting. Second, is rerunning the NSGA-II algorithm with a proper second objective such as the discussed CRPS as the objective for calibration. This can be done with the same generation budget, fold, and paired comparison of the resulting Pareto fronts can be conducted to validate this paper's claim about the flaws of the calibration objective design. Additionally, the models can be retrained with the imbalance volume or downward aFRR activation as targets to evaluate the model's performance on the other two crucial factors within the balancing market. Another interesting suggestion for future work is to pre-train the transformer models on a different country's balancing market such as Austria, and fine tune on the Hungarian balancing market to allow for greater data usage and this can be made into a follow-up paper as well. The final suggestion for future work is regime-aware modeling. The Ukrainian war in 2024 has caused massive changes in the energy market and thus regime feature or a pre-/post-2024 split could allow for the modeling of external factors such as wars.

References

[Abdi, 2010] Abdi, H. (2010). Holm's sequential bonferroni procedure. In Salkind, N. J., editor, *Encyclopedia of Research Design*, pages 1–7. SAGE Publications, Thousand Oaks, CA. Encyclopedia entry on multiple comparison correction and familywise error control.

[Amit et al., 2025] Amit, N., Reingold, O., and Rothblum, G. N. (2025). Accuracy vs. accuracy: Computational tradeoffs between classification rates and utility. *arXiv preprint arXiv:2505.16494*. Version 1, 22 May 2025.

[Balanya et al., 2024] Balanya, S. A., Maroñas, J., and Ramos, D. (2024). Adaptive temperature scaling for robust calibration of deep neural networks. *Neural Computing and Applications*, 36:8073–8095.

[Bäck and Schwefel, 1993] Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23.

[Deb, 2001] Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK.

[Deb and Agrawal, 1995] Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Syst.*, 9.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Diebold and Mariano, 1995] Diebold, F. X. and Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263.

[European Commission, 2017] European Commission (2017). Commission regulation (eu) 2017/2195 establishing a guideline on electricity balancing. Official Journal of the European Union, L 312, 28 November 2017. Electricity Balancing Guideline (EB GL).

[Firoozi et al., 2020] Firoozi, H., Khajeh, H., and Laaksonen, H. (2020). Optimized operation of local energy community providing frequency restoration reserve. *IEEE Access*, 8:180558–180575.

[Gayibov and Gasimov, 2025] Gayibov, A. and Gasimov, V. (2025). Interactive evolutionary nsga-ii with machine learning for efficient hyperparameter optimization of convolutional neural networks. In *2025 6th International Conference on Problems of Cybernetics and Informatics (PCI)*, pages 1–5.

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

[Harvey et al., 1997] Harvey, D., Leybourne, S., and Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2):281–291.

[HUPX, 2025] HUPX (2025). 15-minute mtu available on the hupx day-ahead market. HUPX news announcement on the implementation of 15-minute market time units (MTU) in the SDAC day-ahead market.

[Hyndman and Athanasopoulos, 2021] Hyndman, R. J. and Athanasopoulos, G. (2021). Forecasting: Principles and practice (3rd ed.) — simple forecasting methods. Accessed 2026-05-19.

[Jamieson and Talwalkar, 2015] Jamieson, K. and Talwalkar, A. (2015). Non-stochastic best arm identification and hyperparameter optimization. *CoRR*, abs/1502.07943.

[Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Kim et al., 2022] Kim, T., Kim, J., Tae, Y., Park, C., Choi, J., and Choo, J. (2022). Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

[Lim et al., 2020] Lim, B., Arik, S. O., Loeff, N., and Pfister, T. (2020). Temporal fusion transformers for interpretable multi-horizon time series forecasting.

[Lin et al., 2024] Lin, X., Zhang, X., Yang, Z., Liu, F., Wang, Z., and Zhang, Q. (2024). Smooth tchebycheff scalarization for multi-objective optimization. *arXiv preprint arXiv:2402.19078*.

[Mosquera-López and Nursimulu, 2019] Mosquera-López, S. and Nursimulu, A. (2019). Drivers of electricity price dynamics: Comparative analysis of spot and futures markets. *Energy Policy*, 126:76–87.

[Nie et al., 2023] Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers.

[Schwenen and Neuhoff, 2024] Schwenen, S. and Neuhoff, K. (2024). Renewable energy and equilibrium hedging in electricity forward markets. *The Energy Journal*, 45(5):105–123.

[Shumway and Stoffer, 2017] Shumway, R. H. and Stoffer, D. S. (2017). *ARIMA Models*, pages 75–163. Springer International Publishing, Cham.

[Thundiyil et al., 2023] Thundiyil, S., Shalamzari, S., Picone, J., and McKenzie, S. (2023). Transformers for modeling long-term dependencies in time series data: A review. In *2023 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–5.

[Uijtewaal, 2025] Uijtewaal, F. (2025). Forecasting afrr bid ladders in the dutch imbalance market. Master's thesis, Delft University of Technology. Accessed 2026-05-18.

[Verma et al., 2021] Verma, S., Pant, M., and Snášel, V. (2021). A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems. *IEEE Access*, 9:57757–57791.

[Yusoff et al., 2011] Yusoff, Y., Ngadiman, M. S., and Zain, A. M. (2011). Overview of nsga-ii for optimizing machining process parameters. *Procedia Engineering*, 15:3978–3983.

[Zhao et al., 2026] Zhao, J., Chu, F., Xie, L., Che, Y., Wu, Y., and Burke, A. F. (2026). A survey of transformer networks for time series forecasting. *Computer Science Review*, 60:100883.

[Zitzler et al., 2000] Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195.

A Hyperparameter Search Space

Table 25: NSGA-II run parameters shared by the TFT and PatchTST tuning

Parameter	**Value**
population_size	20
n_generations	5
η_c (SBX distribution index)	15.000
η_m (polynomial mutation index)	20.000
p_{cross} (per-pair)	0.9
seed	42
tuning fold	3
objective 1	mean MASE (validation)
objective 2	$\lvert\text{cov}_{80} - 0.80\rvert$ (validation)
successive halving	enabled (rate $\eta = 2$)

Table 26: TFT hyperparameter search domain. Categorical menus are chosen so every drawn *(hidden_size, attention_head_size)* pair satisfies the divisibility constraint.

Parameter	**Type**	**Domain**
`hidden_size`	categorical	{16, 32, 64}
`attention_head_size`	categorical	{1, 2, 4}
`hidden_continuous_size`	categorical	{8, 16, 32}
`lstm_layers`	int	$\{1, 2\}$
`dropout`	float	[0.05, 0.35]
`learning_rate`	log_float	[0.0001, 0.005], log-uniform
`batch_size`	categorical	{128, 256, 512}

Table 27: PatchTST hyperparameter search domain. Categorical menus are chosen so every drawn *(d_model, n_heads)* pair satisfies divisibility. *Stride* paramter is clamped to $\leq$ *patch_len* to guarantee a valid patch tokenisation.

Parameter	**Type**	**Domain**
`d_model`	categorical	{64, 128, 256}
`n_heads`	categorical	{2, 4, 8}
`n_layers`	int	$\{2, 3, 4\}$
`d_ff`	categorical	{128, 256, 512}
`patch_len`	categorical	{8, 16, 24}
`stride`	categorical	{4, 8, 16}
`dropout`	float	[0.05, 0.35]
`learning_rate`	log_float	[0.0001, 0.005], log-uniform
`weight_decay`	log_float	$[1e-06, 0.001]$, log-uniform
`batch_size`	categorical	{128, 256, 512}

B Full Pareto Front Configurations

Table 28: Full NSGA-II Pareto front for TFT on fold 3 validation. Both objectives (MASE and $|\text{cov}_{80} - 0.80|$) are means over the five reporting horizons. Hyperparameter columns use the same names as the search space in Table 26.

eval id	gen	MASE	cal. err.	hidden_size	attention_head_size	hidden_continuous_size	lstm_layers	dropout	learning_rate	batch_size
104	5	0.469	0.082	64.000	4.000	8.000	1.000	0.2094	0.0004367	128.000
59	2	0.472	0.065	32.000	4.000	8.000	2.000	0.1913	0.0009252	128.000
43	2	0.473	0.056	32.000	2.000	32.000	2.000	0.1654	0.002246	512.000
113	5	0.477	0.018	32.000	1.000	8.000	1.000	0.1965	0.0009338	128.000

Table 29: Full NSGA-II Pareto front for PatchTST on fold 3 validation. Both objectives (MASE and $|\text{cov}_{80} - 0.80|$) are means over the five reporting horizons. Hyperparameter columns use the same names as the search space in Table 27.

eval id	gen	MASE	cal. err.	d_model	n_heads	n_layers	d_ff	patch_len	stride	dropout	learning_rate	weight_decay	batch_size
56	2	0.478	0.090	128.000	8.000	4.000	128.000	16.000	16.000	0.1378	0.0001448	5.444e-05	128.000
8	0	0.481	0.070	256.000	8.000	4.000	256.000	16.000	16.000	0.2842	0.0006021	5.084e-05	128.000
72	3	0.482	0.032	64.000	8.000	4.000	512.000	16.000	16.000	0.1366	0.0001592	4.981e-06	128.000

C Code Repository Layout and Reproducibility Instructions

The repository for the code can be found here.

Table 30: Runtime environment used for every training and tuning run reported in this thesis. CUDA passthrough into the WSL2 guest is transparent to PyTorch.

Component	**Value**
Operating system	Linux 5.15.167.4-microsoft-standard-WSL2 (WSL2 guest on Windows 11)
CPU	Intel Core i9-14900K, 32 logical cores @ 6.0 GHz boost
GPU	NVIDIA GeForce RTX 5080 (16 GB VRAM), CUDA 13.3 driver
RAM	48 GiB available to the WSL2 guest (64 GiB host)
Python	3.12.9 (CPython)
Package manager	uv 0.11.15 with locked `uv.lock`
Random seed (NSGA-II, LightGBM)	42

Table 31: Top-level dependency declarations from *pyproject.toml*. Exact pinned versions are recorded in *uv.lock* and recreated deterministically by *uv sync*.

Dependency specification
`numpy>=1.26`
`pandas>=2.2`
`polars>=1.0`
`pyarrow>=15.0`
`scipy>=1.13`
`scikit-learn>=1.5`
`statsmodels>=0.14`
`torch==2.11.0`
`lightning>=2.3`
`pytorch-forecasting>=1.0.0`
`optuna>=3.6`
`lightgbm>=4.3`
`retry-requests>=2.0`
`httpx>=0.27`
`hydra-core>=1.3`
`omegaconf>=2.3`
`mlflow>=2.13`
`python-dotenv>=1.0`
`tqdm>=4.66`
`matplotlib>=3.9`
`seaborn>=0.13`
`plotly>=5.22`
`torchvision==0.26.0`
`torchaudio==2.11.0`

```
aFRR_forecast/
  data/                  # raw + interim parquets (not in git)
    raw/                 # ENTSO-E + Open-Meteo monthly chunks
    interim/             # dataset_15min.parquet (merged grid)
  notebooks/             # EDA notebook
  outputs/               # trained checkpoints, predictions (not in git)
  reports/               # result parquets (not in git)
  scripts/               # entry points
  src/afrr/              # library code
    data/                # ENTSO-E + weather fetchers
    features/            # feature engineering
    models/              # TFT factory, PatchTST module
    training/            # cv splits, runners, evaluation
    tuning/              # NSGA-II + per-arch search spaces
    utils/               # utility files such as seed utilites
    evaluation/          # dm test and metric evaluation
  tests/                 # pytest suite
  pyproject.toml         # dependency manifest
  uv.lock                # locked dependency versions
  README.md              # summary of the repository
  .env.example           # example enviornment
```

Figure 20: Project repository layout.

D Per-Fold Detailed Metric Tables

Table 32: Per-fold MAE on the aFRR price up target. Sections 7.1 and 7.2 average these across folds, where applicable. The unaveraged values are reported so per-fold variability and any single-fold anomalies are traceable.

Fold	Model	15 min	1 h	2 h	3 h	4 h
1	Naive (day)	68 623	68 612	68 500	68 387	68 444
	Naive (week)	69 037	69 011	69 093	69 120	69 176
	LightGBM	46 584	48 228	48 311	48 328	48 387
	PatchTST (default)	36 816	46 850	48 011	48 371	48 792
	TFT (default)	40 731	54 235	56 468	56 623	56 657
2	Naive (day)	68 287	68 319	68 359	68 391	68 380
	Naive (week)	72 308	72 316	72 293	72 293	72 265
	LightGBM	42 609	43 977	44 055	44 054	44 031
	PatchTST (default)	36 432	44 870	44 386	44 538	44 482
	TFT (default)	32 236	42 545	43 845	43 721	43 841
3	Naive (day)	76 123	76 186	76 222	76 277	76 305
	Naive (week)	73 228	73 256	73 244	73 312	73 340
	LightGBM	49 421	50 682	50 688	50 756	50 782
	PatchTST (default)	40 290	51 432	51 382	50 864	50 837
	TFT (default)	38 617	51 462	53 579	53 549	54 295

Table 33: Per-fold RMSE on the aFRR price up target. Sections 7.1 and 7.2 average these across folds, where applicable. The unaveraged values are reported so per-fold variability and any single-fold anomalies are traceable.

Fold	Model	15 min	1 h	2 h	3 h	4 h
1	Naive (day)	135 595	135 582	135 514	135 445	135 480
	Naive (week)	138 783	138 761	138 808	138 823	138 857
	LightGBM	104 507	108 280	108 750	108 893	108 942
	PatchTST (default)	89 952	99 974	105 432	106 494	107 330
	TFT (default)	89 661	102 490	106 764	106 969	106 997
2	Naive (day)	166 922	166 931	166 945	166 955	166 952
	Naive (week)	170 096	170 095	170 081	170 081	170 075
	LightGBM	122 997	125 921	126 012	126 013	126 003
	PatchTST (default)	110 134	122 691	125 857	126 145	126 266
	TFT (default)	107 805	122 160	125 363	125 302	125 581
3	Naive (day)	159 555	159 633	159 652	159 685	159 698
	Naive (week)	158 528	158 556	158 559	158 595	158 609
	LightGBM	123 634	128 183	128 263	128 309	128 323
	PatchTST (default)	101 173	121 426	128 139	128 049	128 187
	TFT (default)	102 105	121 134	125 827	125 672	125 730

Table 34: Per-fold MASE on the aFRR price up target. Sections 7.1 and 7.2 average these across folds, where applicable. The unaveraged values are reported so per-fold variability and any single-fold anomalies are traceable.

Fold	Model	15 min	1 h	2 h	3 h	4 h
1	PatchTST (default)	0.492	0.626	0.642	0.646	0.652
	TFT (default)	0.544	0.725	0.755	0.757	0.757
2	PatchTST (default)	0.495	0.610	0.603	0.605	0.604
	TFT (default)	0.438	0.578	0.596	0.594	0.596
3	PatchTST (default)	0.552	0.704	0.704	0.696	0.696
	TFT (default)	0.529	0.705	0.734	0.733	0.743

Table 35: Per-fold pinball loss on the aFRR price up target. Sections 7.1 and 7.2 average these across folds, where applicable. The unaveraged values are reported so per-fold variability and any single-fold anomalies are traceable.

Fold	Model	15 min	1 h	2 h	3 h	4 h
1	PatchTST (default)	13 914	16 686	17 342	17 161	18 170
	TFT (default)	14 109	16 828	17 359	17 395	17 408
2	PatchTST (default)	13 802	16 913	17 432	16 846	17 876
	TFT (default)	12 305	15 039	15 438	15 400	15 458
3	PatchTST (default)	14 761	18 569	19 391	19 578	19 476
	TFT (default)	13 593	17 394	18 112	18 144	18 320

Table 36: Per-fold empirical 80% prediction interval coverage on the aFRR price up target. Sections 7.1 and 7.2 average these across folds, where applicable. The unaveraged values are reported so per-fold variability and any single-fold anomalies are traceable.

Fold	Model	15 min	1 h	2 h	3 h	4 h
1	PatchTST (default)	0.857	0.703	0.741	0.673	0.655
	TFT (default)	0.822	0.855	0.857	0.857	0.854
2	PatchTST (default)	0.646	0.858	0.767	0.724	0.699
	TFT (default)	0.914	0.915	0.919	0.922	0.922
3	PatchTST (default)	0.793	0.892	0.930	0.856	0.910
	TFT (default)	0.842	0.889	0.895	0.898	0.895

E ENTSO-E API Access Notes

Since the ENTSO-E API key was privately provided it is not included in the paper but an API key can be freely obtained on the ENTSOE Transparency Platform website.