Winds of change: interpretable forecasting of Redispatch 2.0 measures

M. Basangova, V. Bolovaneanu,
A. Conda, C. Erlwein-Sayer, A. Melzer,
D. T. Pele, A. Petukhina, M. Phan

ASE Bucharest
HTW Berlin
COST 19130 FinAI
COST 26130 HiTEc
AI4EFIN

Redispatch

„A request issued by the transmission system operator (TSO) to power plans to adjust the real power they input in order to avoid or eliminate congestion. This method can be applied within or between control areas. “ (Transnetbw)

This talk on Quantinar

Relevance

- Grid Congestion Dynamics
- Challenges with Renewables
- Cost of Redispatch
- Legislative Adaptation

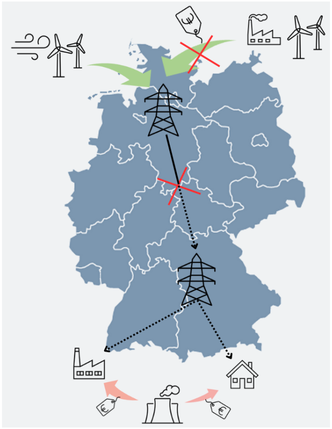
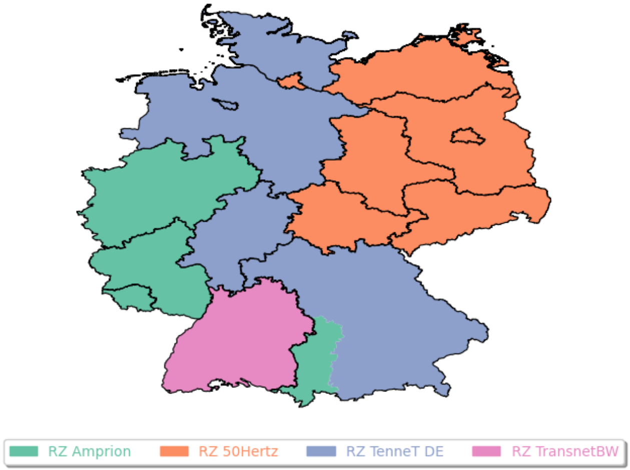


Figure 1: Redispatch Process

TSO in Germany



Electricity generation in Germany

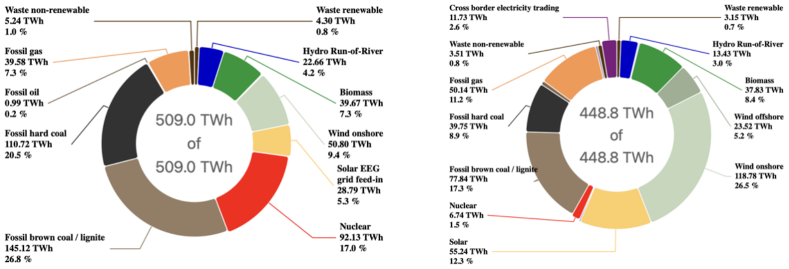


Figure 2: Public net electricity generation in Germany in 2013 (left) and 2023 (right), Source link

Redispatch costs evolution

Year	2020	2021	2022	2023	2024
50Hertz	80.23	106.30	325.83	273.30	211.56
Amprion	128.19	338.20	896.78	394.07	407.16
TenneT DE	708.75	724.89	1563.60	1748.62	972.58
TransnetBW	20.74	101.51	315.07	184.08	101.04

Table 1: Yearly redispatching costs in million € reported by each TSO. Source: ENTSO-E

State of the Art

- Staudt et al., 2018
  - Predict redispatch occurrence (not load) per power plant in Germany
  - Use multiple machine learning models, from Artificial Neural network to Random Forest
  - Integrate weather features, electricity price and production
- Gürses-Tran, Flamme, and Monti, 2020
  - Use RNN-based probabilistic model with parametric and non-parametric implementations to forecast 40h-ahead future congestion quantiles
  - Focus on a single power plant in Southern Sweden
  - Benchmark with naive benchmarks

State of the Art

- Billault-Chaumartin, Eising, and Motte-Cortés, 2020
  - Literature overview of papers regarding redispatch modeling in Germany
  - use hourly wind and PV feed-in, load and redispatch measures data between 2015 and 2019 to model redispatch direction (Up and Down) with Fast Fourier transformation
- Titz, Pütz, and Witthaut, 2024
  - Model for the hourly volume of redispatch and countertrade (Gradient Boosted Trees and SHAPley values) in the whole German transmission grid
  - Wind power generation in northern Germany emerged as the main driver, along with hydro run-of-river and neighboring country flows

Redispatch volume in Germany with Gradient Boosted Trees

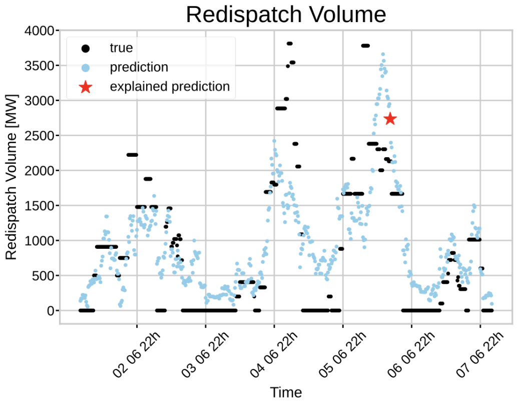


Figure 3: Redispatch volume in Germany from power grid features, Titz et al. 2024

Challenges

- Unbalanced supply and demand in Germany
- Impacts distribution grid, most importantly north - south
  - Higher concentration on renewable energies in Northern Germany because of wind farms
  - Electricity demand in the South is higher

$\Rightarrow$ leads to congestions

Objectives

- Develop a data-driven 24-hours ahead forecast of redispatch measures for each German TSO
- Identify drivers of congestion via feature importance tools
  - Leverage state-of-the-art machine learning models, such as Temporal Fusion Transformers (TFT) and N-BEATS, to analyze key factors influencing redispatch needs.
- Evaluate the economic impact of redispatch forecasting
  - Assess cost implications, including potential savings from optimized redispatch operations and reduced dependency on conventional balancing measures.

Outline

Model selection

- Targets for each TSO:
  - Total load of redispatch - up and down with a single model
- ML Algorithms
  - Regression for day-ahead (24-hour) forecasting
    1. TFT (Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting, Bryan Lim et al., 2021)
    2. N-BEATSx (Neural Basis Expansion Analysis Time Series with exogenous variables, Olivares et al., 2023)
    3. N-HiTS (Neural Hierarchical Interpolation for Time Series Forecasting, Chalu et al., 2022)
    4. Naive, ARIMA as benchmarks

▸ NBEATSx ▸ NHITS ▸ TFT

Training & Inference

- Regression for up and down load
- Compare results for many regression algorithms under different conditions
  - Input size
  - Scaling
  - Hyperparameters
  - Features

Training & Inference

- N-BEATSx
  - Simplicity and effectiveness compared to other deep learning models like RNNs or LSTMs on more irregular time series through repeated decomposition of the signal
  - Supports covariates (exogenous variables)
- N-HiTS
  - It builds upon the strengths of previous models like N-BEATS but introduces innovative features such as hierarchical pooling and interpolation to improve performance, efficiency, and interpretability
  - Suitable for long-horizon predictions
- TFT
  - An advanced deep learning model specifically designed for multi-horizon time series forecasting
  - Ability to capture complex temporal patterns while providing interpretability through attention and variable selection

Training & Inference - Benchmarks

Naive benchmark: Use last available time point: $\hat{y}_t = y_{t-24}$

ARIMA:

- Econometric approach widely employed by researchers and practitioners in fields such as energy and finance

Target processing

- 60% of redispatch interventions are performed by all 4 TSOs, and we split them equally
- We exclude interventions related to:
  - Test runs
  - Countertrading with Denmark and Norway bidding zones
  - Interventions with foreign TSOs

Target processing

- Input: total intervention length per event
- Split into equal 1h intervals, sum by hour $\Rightarrow$ hourly redispatch per TSO
- Final aggregation levels:
  - Hour interval
  - TSO
  - Direction: up / down

Features data

70 features

- Prices (USD)
  - Brent oil, natural gas, carbon emissions futures, CO2 certificates (daily)
  - day-ahead electricity DE, CH, CZ, NL, PL, AU (hourly)
- Weather
  - Average air temperature, wind velocity per TSO (hourly)
  - *Min, max, average wind velocity and air temperature per Bundesland* (hourly)
- Date
  - weekday, weekend, holidays, hour/day/month
- Electricity consumption and production forecasts (hourly)
  - electricity production from renewable and conventional sources
  - grid and residual load

Hourly Redispatch average load

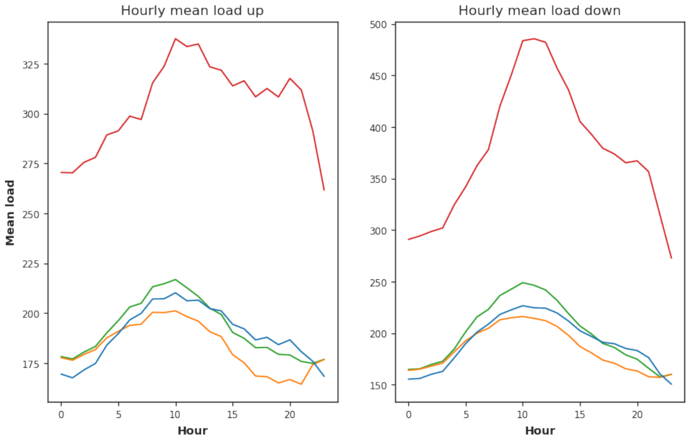


Figure 4: Hourly mean load for TenneT, 50Hertz, Amprion and TransnetBW

Weekly Redispatch total load

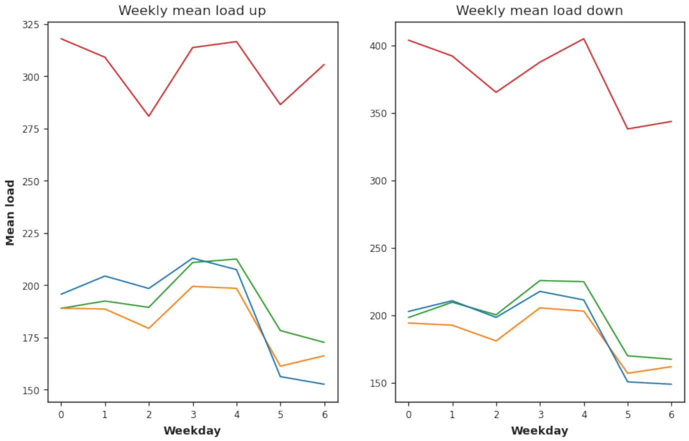


Figure 5: Weekly mean load for TenneT, 50Hertz, Amprion and TransnetBW

Redispatch patterns

- Usually, a sufficiently long input size is chosen to account for these seasonality or other patterns (e.g., Lim et al., 2021).
- As per Figure 4, 5 there are strong indications of hourly and more subtle for weekly patterns in the data.
- In our hyperparameter optimization, a 24-hour input size is chosen as the best for all models, even though we went as far as 1 week.

Affected plants

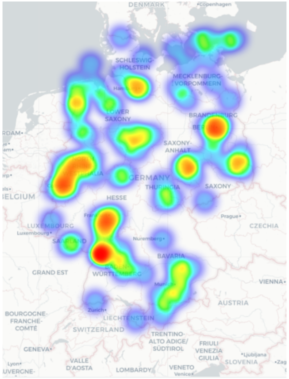


Figure 6: Affected units, Heat map based on total congestion load. Heat map video

Data split

Split	Time Period	*n*
Train	2020-01-01 to 2024-01-31	35,808
Validation	2024-02-01 to 2024-03-31	1,440
Test	2024-04-01 to 2025-01-31	7,344

Table 2: Data split for Regression training

- Rolling window training:
  - Training window: 49 months, validation window: 2 months, test window: 1 month
  - Total: 11 windows
  - Windows slide 1 month at a time

Training loss

- Early stopping on the validation dataset loss
- Mean Absolute Error (MAE) as loss
- Higher load is more unpredictable and models struggle more in this department

Training loss

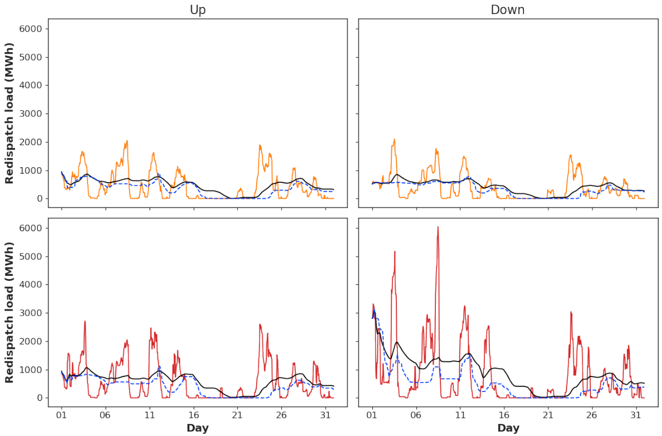


Figure 7: Redispatch load in January 2025 for TransnetBW and TenneT DE. MSE is proxied by 5-day rolling mean (solid) and MAE by 5-day rolling median (dashed)

Regression performance

▸ Metrics

TSO	metric	Direction	ARIMA	Naive benchmark	N-BEATSx	NHiTS	TFT
50Hertz	Huber	down	0.1837	0.1152	0.0579	**0.0541**	0.0562
		up	0.1852	0.1211	0.061	**0.0593**	0.0613
	R^2	down	-1.35	-0.26	**0.57**	0.57	0.52
		up	-1.27	-0.27	**0.59**	0.58	0.5
Amprion	Huber	down	0.1803	0.1099	0.0556	**0.0514**	0.0568
		up	0.1769	0.1214	0.0588	**0.0557**	0.0593
	R^2	down	-1.45	-0.25	0.58	**0.6**	0.53
		up	-1.23	-0.28	0.61	**0.62**	0.55
TenneT DE	Huber	down	0.1241	0.115	0.0582	0.0571	**0.057**
		up	0.1892	0.1358	0.0637	0.0626	**0.0602**
	R^2	down	-0.23	-0.3	0.49	0.49	**0.5**
		up	-1.17	-0.4	0.56	0.55	**0.58**
TransnetBW	Huber	down	0.2028	0.1128	0.0557	**0.0533**	0.0631
		up	0.1914	0.1202	0.0569	**0.0541**	0.0645
	R^2	down	-1.67	-0.21	0.61	**0.61**	0.45
		up	-1.3	-0.27	0.62	**0.63**	0.45

Table 3: Prediction metrics, 24-hour forecast horizon

Regression performance

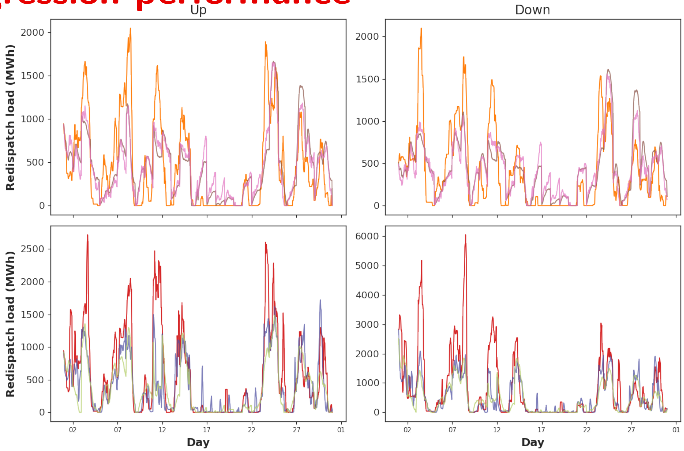


Figure 8: Redispatch load out-of-sample prediction in January 2025 for TransnetBW, with N-BEATSx, NHiTS predictions, and TenneT DE, with TFT, NHiTS predictions.

SHAP GradientExplainer — N-HiTS

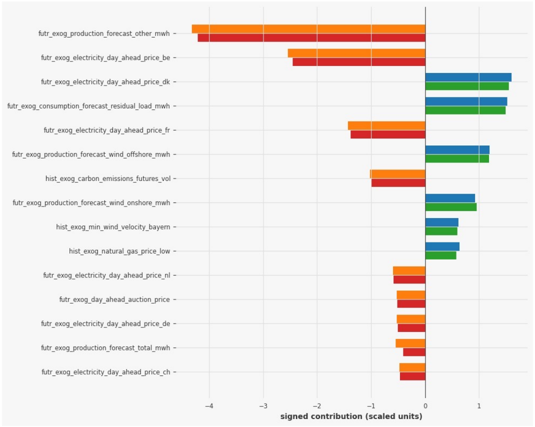


Figure 9: SHAP values from `shap.GradientExplainer`: signed, scaled feature contributions.

SHAP GradientExplainer — TFT

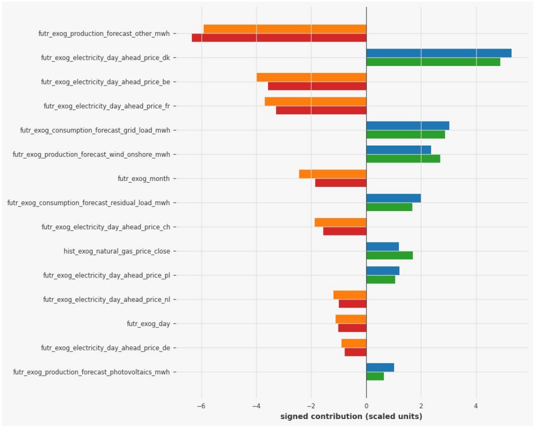


Figure 10: SHAP values from `shap.GradientExplainer`: signed, scaled feature contributions.

SHAP GradientExplainer — N-BEATSx

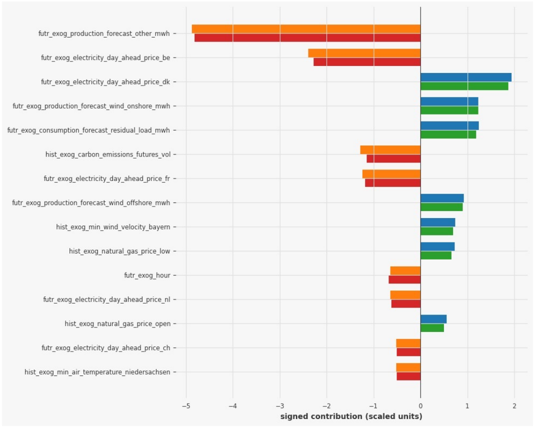


Figure 11: SHAP values from `shap.GradientExplainer`: signed, scaled feature contributions.

Feature importance

- Our features improve results by an average of 25 - 30% for MAE, RMSE and 50% for R^2, as can be seen below.

	nbeatsx			nhits			tft		
metric	mae	r2_score	rmse	mae	r2_score	rmse	mae	r2_score	rmse
down	0.29	-0.66	0.27	0.28	-0.51	0.23	0.25	-0.52	0.20
up	0.31	-0.58	0.35	0.35	-0.58	0.34	0.32	-0.50	0.26

Table 4: % of metric improvement when training with covariates vs without, TenneT DE, 24-hour forecast horizon

How much can we save?

- We want to test how prior knowledge of the redispatched load can impact costs
- A redispatch will occur regardless, but we can assume TSOs get to prepare and use cheaper energy sources
- We consider the allocation proportion $w_{t,s}^{orig}$ for source s used currently (which can be obtained from historical data)
- We propose a more efficient allocation portfolio $w_{t,s}^{prep}$ for correctly forecasted load

How much can we save?

- Goal: impact of prior knowledge of redispatched load on costs
- Redispatch inevitable $\Rightarrow$ TSOs prepared, cheaper energy sources available
- Baseline: historical allocation shares $w_{t,s}^{\text{orig}}$ for source s
- Proposal: prepared allocation $w_{t,s}^{\text{prep}}$ for correctly forecasted load

Metric details

How much can we save?

Model	Direction	TenneT DE	50Hertz	Amprion	TransnetBW	Overall
ARIMA	down	1825.8184	809.0985	769.1791	768.2001	4172.2962
	up	1178.2924	898.0889	910.2123	898.7217	3885.3153
Naive	down	1823.3757	806.9543	768.3125	763.7659	4162.4084
	up	1175.5876	895.7827	909.2220	893.7985	3874.3909
N-BEATSx	down	1794.3270	791.1348	752.5827	747.1530	4085.1975
	up	1145.2180	879.7501	893.6025	877.4403	3796.0109
NHiTS	down	**1793.0305**	789.8338	**751.5551**	**746.5174**	**4080.9367**
	up	**1144.6920**	878.5729	**892.6505**	**876.8414**	**3792.7569**
TFT	down	1793.9658	**789.1730**	752.3021	746.8719	4082.3127
	up	1144.8478	**877.5082**	893.2411	877.5763	3793.1733
Benchmark	down	1798.5516	791.0386	751.9573	748.1743	4089.7218
	up	1148.1821	879.9662	893.8082	879.2032	3801.1597

Table 5: Economic implications in million €, out-of-sample period, 24-hour forecast horizon; source loads are taken proportional to historical data

How much can we save? - Conclusions

- In this setup, our top models are more efficient that the actual operation, with NHiTS reaching a 0.21% reduction in total costs, equivalent to almost €9 million

Attention analysis

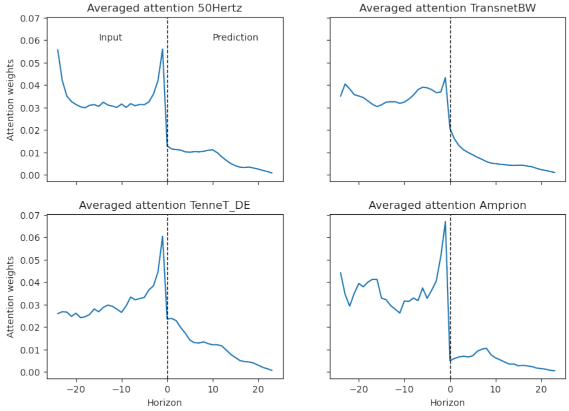


Figure 12: Average TFT daily attention across all horizons, out-of-sample, 24-hour forecast horizon. X axis represents the time horizon, negative values are model inputs. The dashed line indicates where TFT starts predicting.

Attention analysis - Conclusions

- TFT pays less and less attention when it is about to forecast. It is more "interested" in covariates immediately before the prediction interval.
- Such a behaviour could indicate that it does not find any good patterns and it tries to emulate the benchmark strategy (use latest data points)
- We confirm the descending attention with dwindling average performance on the forecast horizon - first 7 hours are much better predicted than the last

Conclusions

- Redispatch data presents some patterns that can be investigated by Machine Learning models
- NHITS and TFT outperform other econometric and benchmarks to forecast day-ahead redispatch load
- With the right over-prediction weight (not defined in the German law), we can achieve lower redispatch costs
- Main drivers:
  - negative: Production forecast others, DA price BE
  - positive: DA Price DK, onshore wind forecast

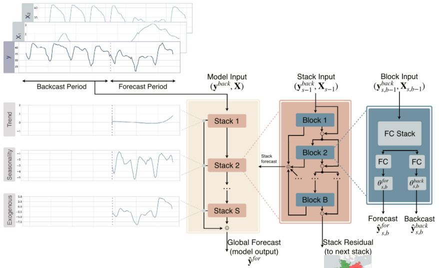
Next steps

- Refine per-source loads using available data in the economic cost calculations
- Better account of the over-prediction weight
- Feature importance for NHITS/NBEATSx

N-BEATS Description

- Introduced in Oreshkin et al., 2020
- Black-box or interpretable architecture
- Supports transfer learning
- Decomposes signal into *backcast* (history) and *forecast* (prediction)

- Two main building blocks:
  - *Blocks*: capture specific information using Fully-Connected (FC) layers and a learnable linear projection
  - *Stacks*: a collection of blocks



Back to "Pipeline"

NBEATSx Model Architecture

$$
\begin{aligned}
h_{s,b} &= \mathrm{MLP}_{s,b}\left(y_{\mathrm{back},s,b-1}, X_{s,b-1}\right) \\
\theta_{\mathrm{back},s,b} &= \mathrm{Linear}_{\mathrm{back}}\left(h_{s,b}\right) \\
\theta_{\mathrm{for},s,b} &= \mathrm{Linear}_{\mathrm{for}}\left(h_{s,b}\right)
\end{aligned}
$$

Basis Expansion:

$$
\begin{aligned}
\hat{y}_{\mathrm{back},s,b} &= V_{\mathrm{back},s,b}\theta_{\mathrm{back},s,b} \\
\hat{y}_{\mathrm{for},s,b} &= V_{\mathrm{for},s,b}\theta_{\mathrm{for},s,b}
\end{aligned}
$$

NBEATSx Model Architecture

Stack and Residual Update:

$$y_{\text{back},s,b+1} = y_{\text{back},s,b} - \hat{y}_{\text{back},s,b}$$

$$\hat{y}_{\text{for},s} = \sum_{b=1}^{B} \hat{y}_{\text{for},s,b}$$

where:

- $y_{\text{back},s,b}$ is the backcasted time series input.
- $X_{s,b}$ represents the exogenous variables used in the block.
- $h_{s,b}$ is the hidden representation in block b of stack s.
- $\theta_{\text{back},s,b}$ and $\theta_{\text{for},s,b}$ are the expansion coefficients.
- $V_{\text{back},s,b}$ and $V_{\text{for},s,b}$ are the learned basis functions.

NBEATSx

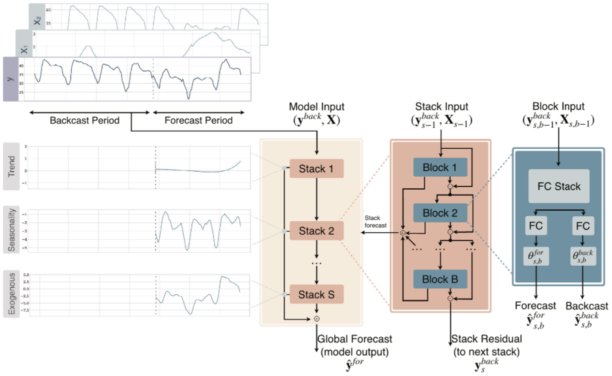
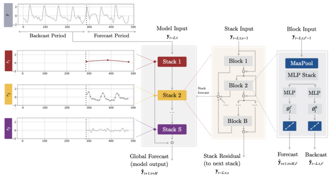


Figure 13: NBEATS with exogenous variables, Olivares et al. 2024

N-HiTS Description

- Introduced in Challu et al., 2022
- An evolution of N-BEATS by:
  - Included kernel pooling at each block's entry point
  - Regularized basis function forms by hierarchical interpolation and multi-rate data sampling

- Unique parameters from N-BEATS:
  - *Pooling*: control how pooling is done (max, mean, etc.), and how much the inputs are shrinked
  - *Interpolation type*: three were proposed in the paper (linear, cubic, and nearest neighbor)



NHITS Architecture

NHITS model consists of multiple blocks using hierarchical interpolation and multi-rate sampling:

$$
\begin{aligned}
h_\ell &= \mathrm{MLP}_\ell\left(\mathrm{MaxPool}\left(y_{t-L:t,\ell}, k_\ell\right)\right) \\
\theta_f^\ell &= \mathrm{Linear}_f(h_\ell) \\
\theta_b^\ell &= \mathrm{Linear}_b(h_\ell)
\end{aligned}
$$

Hierarchical Interpolation:

$$
\begin{aligned}
\hat{y}_{\tau,\ell} &= g(\tau, \theta_f^\ell), \quad \forall \tau \in \{t+1, ..., t+H\} \\
\hat{y}_{\tau,\ell} &= g(\tau, \theta_b^\ell), \quad \forall \tau \in \{t-L, ..., t\}
\end{aligned}
$$

NHITS Architecture

Final Forecast Assembly:

$$\hat{y}_{t+1:t+H} = \sum_{\ell=1}^{L} \hat{y}_{t+1:t+H,\ell}$$

$$y_{t-L:t,\ell+1} = y_{t-L:t,\ell} - \tilde{y}_{t-L:t,\ell}$$

where $y_{t-L:t,\ell}$ is the input time series segment with lagged observations, k_ℓ is the kernel size for multi-rate sampling, h_ℓ is the hidden representation in block ℓ. θ_f^ℓ and θ_b^ℓ are the forward and backward interpolation coefficients, $g(\tau, \theta_f^\ell)$ is the hierarchical interpolation function mapping latent features to future predictions, $\hat{y}_{t+1:t+H}$ is the final assembled forecast by summing the block-wise outputs.

NHITS Model Illustration

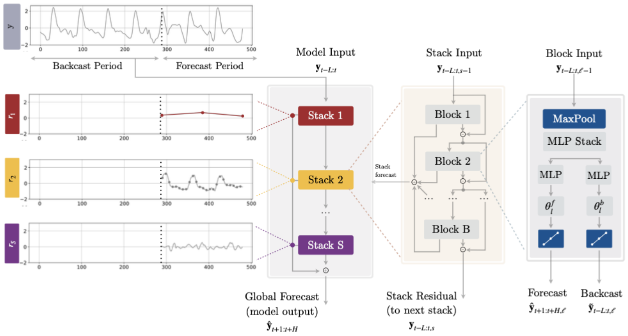
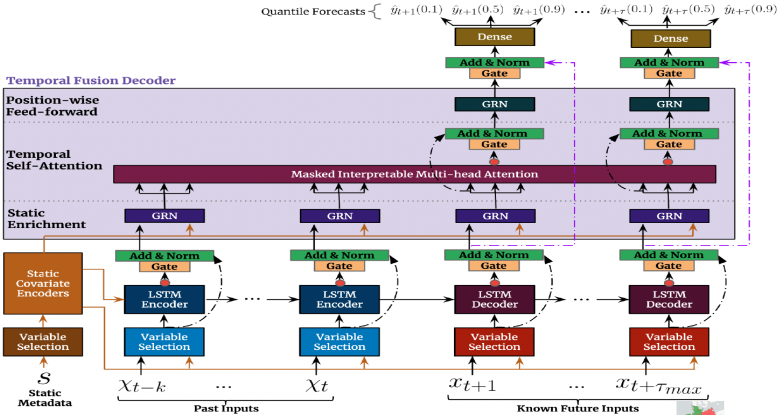


Figure 14: NHITS model with multi-rate input sampling and hierarchical interpolation.

▸ Back to "Pipeline"

TFT Description

- Introduced in Lim et al., 2021


Quantile Forecasts
$\hat{y}_{t+1}(0.1)$ $\hat{y}_{t+1}(0.5)$ $\hat{y}_{t+1}(0.9)$ … $\hat{y}_{t+\tau}(0.1)$ $\hat{y}_{t+\tau}(0.5)$ $\hat{y}_{t+\tau}(0.9)$
Dense
Add & Norm
Gate
Temporal Fusion Decoder
Position-wise Feed-forward
GRN
Temporal Self-Attention
Masked Interpretable Multi-head Attention
Static Enrichment
Static Covariate Encoders
LSTM Encoder
LSTM Decoder
Variable Selection
s
Static Metadata
χ_{t-k} … χ_t
Past Inputs
x_{t+1} … $x_{t+\tau_{max}}$
Known Future Inputs

TFT Model Architecture

Mathematical Formulation:

$$\begin{aligned} h_t &= \text{LSTM}\left(y_{t-L:t}, X_{t-L:t}\right) \\ \theta_t^{\text{enc}} &= \text{VariableSelection}(X_t) \\ \theta_t^{\text{dec}} &= \text{MultiHeadAttention}(Q, K, V) \end{aligned}$$

Self-Attention Mechanism:

$$A(Q, K) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

$$H = \sum_{h=1}^{H} A(QW_Q^{(h)}, KW_K^{(h)})VW_V^{(h)}$$

TFT Model Architecture

Final Forecasting Step:

$$\hat{y}_{t+1:t+H} = \sum_{b=1}^{B} \hat{y}_{t+1:t+H,b}$$

where:

- $y_{t-L:t}$ represents the past observations.
- $X_{t-L:t}$ contains past and future covariates.
- h_t is the hidden state representation.
- θ_t^{enc} and θ_t^{dec} are feature selection weights.
- $A(Q, K)$ computes the attention scores.
- H is the multi-head attention output.
- $\hat{y}_{t+1:t+H}$ is the final forecast.

Back to "Pipeline"

Metric Definition

- Our metric is defined as:

$$Cost_{t,s}^{forecast} = \underbrace{min(y_t, \hat{y}_t) \cdot P_{t,s}^{prep}}_{\text{Correct prediction}} + \underbrace{max(y_t - \hat{y}_t, 0) \cdot P_{t,s}^{orig}}_{\text{Under prediction}} + \underbrace{max(\hat{y}_y - y_t, 0) \cdot P_{t,s}^{prep}}_{\text{Over prediction}}$$

where c_t^{cur} are yearly average curtailment costs per resource are available on Netztransparenz, and $P_{t,s}^{prep} = w_{t,s}^{prep} \cdot c_{t,s}^{cur}, P_{t,s}^{orig} = w_{t,s}^{orig} \cdot c_{t,s}^{cur}$

Metric Definition

- We compare it to the actual

$$Cost_{t,s}^{actual} = y_t P_{t,s}^{orig}$$

- Results are aggregated per time and source

$$Cost^{forecast} = \sum_t \sum_s Cost_{t,s}^{forecast}$$

$$Cost^{actual} = \sum_t \sum_s Cost_{t,s}^{actual}$$

▸ Back to Results

Metric Limitations

- One problem is that we cannot correctly compare with our chosen benchmark because of the over-forecasting part
- To see this, notice that

  $$min(y, \hat{y}) + max(y - \hat{y}, 0) = y$$

  therefore the first two terms alone add up to the initial load; the third (which occurs often enough in practice) compensates for the improvement in w^{prep}.
- Therefore, we set it so that ARIMA and the naive benchmark have higher costs that the top ML models (very close to 0)

Metric Limitations

- There is no curtailment cost for conventional energy sources (hard coal, lignite, CHP)
- We use the renewable average cost as a proxy

Load per source

- Existing historical weights/structure of redispatch sources as $w_{2024}^{orig} = w_{2025}^{orig} = (0.58, 0.37, 0.05)$, inspired by 2021 - 2023 historical data
- Proposed weights vector changing grid reserve (the most expensive source for April - August 2024) and renewables (the most expensive source for September 2024 - January 2025) $w_{2024}^{prop} = (0.58, 0.42, 0), w_{2025}^{prop} = (0.58, 0.32, 0.1)$

Back to Results

Regression performance metrics

- Let $y_t, \hat{y}_t$ the actual and forecasted load at time t.
- We test prediction accuracy by computing:

$$R^2 = 1 - \frac{\sum_t (y_t - \hat{y}_t)^2}{\sum_t (y_t - \overline{y})^2}$$

$$H_{t,\delta} = \begin{cases} 0.5(\mathbf{y}_t - \hat{\mathbf{y}}_t)^2 & \text{if } |\mathbf{y}_t - \hat{\mathbf{y}}_t| < \delta \\ \delta \cdot (|\mathbf{y}_t - \hat{\mathbf{y}}_t| - 0.5\delta) & \text{otherwise} \end{cases}$$

▸ Back to Results

Regression performance metrics

- R^2 shows the performance difference in MSE between the model and a baseline where the mean is used as a predictor
- Huber loss H compromises MAE and MSE; for small errors, it uses the latter, else the former
- δ is chosen as the smallest difference in MAE during training (the error under which MAE alone cannot get)

▸ Back to Results