

Accurate and Explainable Electrical Load Forecasting Using Time-Series Transformers

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Matthias Hertel

Tag der mündlichen Prüfung:

Referenten:

27.07.2026

Prof. Dr. Veit Hagenmeyer

Prof. Dr. Christoph Goebel

Matthias Hertel

Accurate and Explainable Electrical Load Forecasting Using Time-Series Transformers

Dissertation, 2026

DOI: 10.5445/IR/1000197399

Karlsruhe Institute of Technology (KIT)

Institute for Automation and Applied Informatics (IAI)

Department of Computer Science

Hermann-von-Helmholtz-Platz 1

76344 Eggenstein-Leopoldshafen

License Information

Chapter 6 and Appendix A: Adapted, with permission, from M. Hertel, L. Ambrosius, M. Treutlein, R. Mikut, and V. Hagenmeyer, “A comparison of local, cluster-specific and global Transformer models for forecasting electrical loads of individual buildings and substations”, *IEEE PowerTech 2025*, © 2025 IEEE.

All other chapters: © Matthias Hertel, 2026. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0), <https://creativecommons.org/licenses/by/4.0/>.

Statement on the usage of generative AI: The author used ChatGPT to improve parts of the writeup. After using this tool, the author reviewed and edited the generated output as needed and takes full responsibility for the content of this dissertation.

Abstract

Accurate electrical load forecasts are essential for the operation of an increasingly complex energy system, including for maintaining the balance between supply and demand, for preventing grid congestion, and for operating energy management systems (EMS). In addition to forecast accuracy, explainability is crucial for fostering user trust and meeting regulatory transparency requirements. This dissertation evaluates and extends time-series Transformers for accurate and explainable electrical load forecasting.

First, different strategies for training time-series Transformers on data from multiple metering units are compared. The results show that a global Transformer model generalizes well across time series and achieves lower forecast errors than local, multivariate, and cluster-specific models.

Second, a flexible Transformer architecture is introduced, that integrates architectural modifications from previous work and allows for an automated architecture optimization. This Transformer is benchmarked against established forecasting methods on three electrical load datasets representing the TSO level, the low-voltage feeder level and the client level. Transformer-based models outperform established methods across grid levels, reducing the forecast error by 6.6–10.7 % compared to the next-best method. The Time-Series Foundation Model (TSFM) Chronos-2 is competitive with specialized load forecasting models on datasets from low aggregation levels, but it is inaccurate for special events in the TSO data.

Third, the potential of the Transformer’s attention mechanism for explainable time-series forecasting is investigated. An analysis of individual attention heads reveals distinct functional patterns; however, their interpretability and the insights into feature dependencies remain limited. To address these limitations, a novel Transformer variant, termed SHAPformer, is proposed. SHAPformer uses attention manipulation to efficiently compute Shapley Additive Explanations (SHAP). The resulting explanations provide insights into the trained model, including seasonal patterns and dependencies on covariates. Furthermore, an adaptation of the SHAPformer approach to TSFMs is presented.

Fourth, the effectiveness of Transformer-based forecasts is evaluated in two downstream applications. In the BigDEAL peak load forecasting competition, a conditional Invertible Neural Network (cINN) outperforms the Transformer. In contrast, in a peak shaving application for an energy community comprising heat pumps and battery storage, Transformer-based forecasts enable more effective peak load reduction compared to alternative approaches.

Overall, this dissertation demonstrates the effectiveness of Transformers for electrical load forecasting and introduces methods to enhance model explainability based on the attention mechanism. These findings contribute to both the methodological advancement and the practical deployment of electrical load forecasting models in modern energy systems.

Danksagung

Ohne die zahlreichen Beiträge und die Unterstützung vieler Menschen wäre es mir nicht möglich gewesen, diese Dissertation zu schreiben. Daher möchte ich an dieser Stelle denjenigen einen Dank aussprechen, die mich in den letzten Jahren auf meinem Weg begleitet und unterstützt haben.

Ralf Mikut danke ich für die gute Betreuung sowohl auf fachlicher als auch auf persönlicher Ebene. Durch deine Unterstützung während meiner gesamten Promotion und dein detailliertes und ausführliches Feedback zur Dissertation hattest du einen großen Anteil am Gelingen dieser Arbeit.

Veit Hagenmeyer danke ich für die Möglichkeit, am Institut für Automation und angewandte Informatik (IAI) promovieren zu dürfen und für das in mich gesetzte Vertrauen, das mich auch durch die schwierigen Phasen der Promotion getragen hat.

Christoph Goebel danke ich für die Begutachtung der Dissertation, und Barbara Bruno, Raffaella Mirandola und Maïke Schwammbberger für die Komplettierung meines Prüfungskomitees.

Meinen Kolleg*innen – insbesondere Alexandra, Angelo, Benedikt, Benjamin, Christian, Ellen, Frederik, Henri, Hossein, Kaleb, Karin, Karthik, Leo, Leonard, Linus, Luca, Manuel, Marian, Markus, Max, Moritz, Nathalie, Nils, Oli, Patricia, Sebastian, Stefan, Steff, Thomas und Ulrich – danke ich für die schöne gemeinsame Zeit und den regelmäßigen Austausch zu Fach-, Promotions- und persönlichen Themen. Es war mir eine Freude, auf so vielen Ebenen gemeinsam mit euch und von euch lernen zu dürfen, die Höhen und Tiefen der Promotion gemeinsam zu durchleben, und ab und an zusammen auf einen Retreat zu fahren. Besonders freue ich mich, dass aus vielen der Kontakte Freundschaften fürs Leben wurden.

Den von mir am IAI betreuten Studierenden – Alexander, Jonathan, Lara, Lena und Manuel – danke ich für das mir entgegengebrachte Vertrauen und für ihre wertvolle geleistete Arbeit.

Meiner früheren Arbeitsgruppe an der Universität Freiburg – Claudius, Johannes, Natalie, Patrick, Sebastian und Theresa – und den dort von mir betreuten Studierenden danke ich für die Zusammenarbeit, die Unterstützung und das gemeinsame Entdecken neuer Themen. Hannah Bast danke ich dafür, dass ich so vieles von ihr lernen durfte, das die Grundlage für meine wissenschaftliche Laufbahn gelegt hat.

Meiner Familie und meinen Freund*innen danke ich für die vielen Momente, in denen ich die Arbeit auch mal loslassen konnte. Insbesondere danke ich Roland und Claudia, die mich immer gefördert und an mich geglaubt haben. Mira – danke, dass du mich auf den Boden der Tatsachen zurückbringst, wenn ich verzweifle, und dass du dieses Leben so schön und lebenswert machst.

Allen, die sich für eine bessere Welt einsetzen: danke, dass es euch gibt.

Publications

Part II: Training Strategies for Time-Series Transformers

- [110] M. Hertel, M. Beichter, B. Heidrich, O. Neumann, B. Schäfer, R. Mikut, and V. Hagenmeyer. “Transformer training strategies for forecasting multiple load time series”. In: *Energy Informatics* (2023). DOI: 10.1186/s42162-023-00278-z.
- [109] M. Hertel¹, L. Ambrosius¹, M. Treutlein, R. Mikut, and V. Hagenmeyer. “A comparison of local, cluster-specific and global Transformer models for forecasting electrical loads of individual buildings and substations”. *IEEE PowerTech 2025*. DOI: 10.1109/PowerTech59965.2025.11180482.

Part III: Benchmarking Electrical Load Forecasting Models

- [114] M. Hertel¹, S. Pütz¹, J. Kolar, B. Schäfer, R. Mikut, and V. Hagenmeyer. “A Benchmark for Electrical Load Forecasting Across Grid Levels: Time-Series Transformers Outperform Established Methods”. *DACH+ Energy Informatics 2026*. Preprint available at <https://doi.org/10.48550/arXiv.2607.15705>.

Part IV: Explainability of Time-Series Transformers

- [115] M. Hertel, S. Pütz, R. Mikut, V. Hagenmeyer, and B. Schäfer. “Explainable time-series forecasting with sampling-free SHAP for Transformers”. In: *Nature Communications* (2026). DOI: 10.1038/s41467-026-73243-5.
- [111] M. Hertel¹, A. Nikoltchovska¹, S. Pütz, R. Mikut, V. Hagenmeyer, and B. Schäfer, "Explainable Load Forecasting with Covariate-Informed Time Series Foundation Models". *ACM e-Energy 2026*. DOI: 10.1145/3744255.3811724.

Part V: Applications of Electrical Load Forecasts

- [105] B. Heidrich¹, M. Hertel¹, O. Neumann¹, V. Hagenmeyer, and R. Mikut. “Using conditional Invertible Neural Networks to perform mid-term peak load forecasting”. In: *IET Smart Grid* (2024). DOI: 10.1049/stg2.12169.
- [244] L. Semmelmann, M. Hertel, K. J. Kircher, R. Mikut, V. Hagenmeyer, and C. Weinhardt. “The impact of heat pumps on day-ahead energy community load forecasting”. In: *Applied Energy* (2024). DOI: 10.1016/j.apenergy.2024.123364.

¹ Shared first authors.

Publications not included in the dissertation

Publications on Energy Data Analytics

- [113] M. Hertel¹, S. Ott¹, B. Schäfer, R. Mikut, V. Hagenmeyer, and O. Neumann. “Evaluation of Transformer Architectures for Electrical Load Time-Series Forecasting”. In: *Proceedings 32. Workshop Computational Intelligence* (2022).
- [112] M. Hertel, S. Ott, O. Neumann, B. Schäfer, R. Mikut, and V. Hagenmeyer. “Transformer Neural Networks for Building Load Forecasting”. *Tackling Climate Change with Machine Learning: workshop at NeurIPS 2022*.
- [216] S. Pütz, H. El Ashhab, M. Hertel, R. Mikut, M. Götz, V. Hagenmeyer, and B. Schäfer. “Feasibility of Forecasting Highly Resolved Power Grid Frequency Utilizing Temporal Fusion Transformers”. *Proceedings of the 15th ACM e-Energy 2024*. DOI: 10.1145/3632775.3661963.
- [266] M. Treutlein, M. Schmidt, R. Hahn, M. Hertel, B. Heidrich, R. Mikut, and V. Hagenmeyer. “Generating peak-aware pseudo-measurements for low-voltage feeders using metadata of distribution system operators”. In: *IET Smart Grid* (2025). DOI: 10.1049/stg2.12210.
- [151] J. Kolar, M. Hertel, S. Meisenbacher, R. Mikut, and V. Hagenmeyer. “Reducing Imbalance Energy by Spatial Balance Area Forecasting”. In: *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems (e-Energy) 2025*. DOI: 10.1145/3679240.3734634.
- [153] A. Kreusel, M. Hertel, M. Noskiewicz, H. Maaß, R. Mikut, and V. Hagenmeyer. “Evaluating Time-Series Foundation Models for Cooling Demand Forecasting with Little Data”. In: *Proceedings 35. Workshop Computational Intelligence* (2025).
- [154] A. Kreusel, M. Hertel, M. Noskiewicz, F. Zahn, H. Maaß, R. Mikut, and V. Hagenmeyer. “Covariates Are the Key to Accurate Probabilistic Building Energy Forecasting with Time Series Foundation Models”. In: *Proceedings of the 2026 ACM Sustainability Week*. DOI: 10.1145/3765611.3815507.
- [149] M. Kleinebrahm et al. “Energy-Arena: A Dynamic Benchmark for Operational Energy Forecasting”. *International Conference on the European Energy Market (EEM) 2026*. DOI: 10.1109/EEM68581.2026.11589800.

Publications on Natural Language Processing

- [25] H. Bast, M. Hertel, and M. M. Mohamed. “Tokenization Repair in the Presence of Spelling Errors”. In: *Proceedings of the 25th Conference on Computational Natural Language Learning* (2021). DOI: 10.18653/v1/2021.conll-1.22.
- [27] H. Bast, M. Hertel, and N. Prange. “ELEVANT: A Fully Automatic Fine-Grained Entity Linking Evaluation and Analysis Tool”. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. DOI: 10.18653/v1/2022.emnlp-demos.8.

- [29] H. Bast, M. Hertel, and S. Walter. “Fast Whitespace Correction with Encoder-Only Transformers”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)* (2023). DOI: 10.18653/v1/2023.acl-demo.37.
- [26] H. Bast, M. Hertel, and N. Prange. “A Fair and In-Depth Evaluation of Existing End-to-End Entity Linking Systems”. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. DOI: 10.18653/v1/2023.emnlp-main.411.

Publication on Public Transport Route Planning

- [28] H. Bast, M. Hertel, and S. Storandt. “Scalable Transfer Patterns”. In: *2016 Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*. DOI: 10.1137/1.9781611974317.2.

Contents

I	Introduction	1
1	Motivation	2
1.1	Energy Transition	2
1.2	Energy Forecasting	3
1.3	Explainable Artificial Intelligence (XAI)	7
2	Related Work and Contributions	9
2.1	Electrical Load Forecasting	9
2.2	Time-Series Transformers	12
2.3	Training Strategies	13
2.4	Benchmarking of Load Forecasting Models	14
2.5	Explainability and Interpretability of Time-Series Forecasting Models	17
2.6	Applications of Electrical Load Forecasts	19
2.7	Dissertation Outline	20
3	Preliminaries	21
3.1	Time-Series Forecasting	21
3.2	Transformers	24
3.3	Shapley Additive Explanations (SHAP)	29
4	Datasets	33
4.1	Electricity	33
4.2	Ausgrid Solar Home	37
4.3	Ausgrid Substations	37
4.4	TransnetBW	37
4.5	FeederBW	38
4.6	Local Distribution Companies	38
4.7	Energy Community	39
4.8	Synthetic Dataset	39
II	Training Strategies for Time-Series Transformers	42
5	Enhancing Transformer Models by a Global Training Strategy	43
5.1	Methodology	44
5.2	Experimental Setup	47
5.3	Results	48
5.4	Discussion	51

5.5	Conclusion	52
6	Cluster-Specific Transformer Model Training	53
6.1	Methodology	54
6.2	Experimental Setup	56
6.3	Results	57
6.4	Discussion	62
6.5	Conclusion	63
III	Benchmarking Electrical Load Forecasting Models	64
7	Benchmarking of an Optimized Transformer Architecture Across Grid Levels	65
7.1	Methodology	66
7.2	Experimental Setup	69
7.3	Results	71
7.4	Discussion	79
7.5	Conclusion	81
IV	Explainability of Time-Series Transformers	82
8	Interpretable Forecasting Based on Attention Maps	83
8.1	Methodology	84
8.2	Experimental Setup	87
8.3	Results	87
8.4	Discussion	96
8.5	Conclusion	97
9	SHAPformer: Sampling-Free Explanations by Attention Manipulation	98
9.1	Methodology	99
9.2	Experimental Setup	103
9.3	Results	106
9.4	Discussion	115
9.5	Conclusion	117
10	An Efficient Algorithm for Explaining Time-Series Foundation Models with SHAP	118
10.1	Methodology	119
10.2	Experimental Setup	121
10.3	Benchmarking TSFMs on Recent Operational Load Data	121
10.4	Explaining TSFMs for Load Forecasting	123
10.5	Conclusion	127
V	Applications of Electrical Load Forecasts	129
11	Peak Load Forecasting for the BigDEAL Challenge	130
11.1	Experimental Setup	131

11.2 Results	134
11.3 Conclusion	135
12 Peak Shaving for Energy Communities With Heat Pumps	136
12.1 Experimental Setup	136
12.2 Results	138
12.3 Conclusion	140
VI Conclusion	141
13 Summary	142
14 Limitations and Outlook	144
Appendix	146
A Clustering	148
B Benchmarking	152
C Explainability	156
C.1 Transformer Explained by SHAP	156
C.2 Feature Attention	159
C.3 Attention Manipulation	159
C.4 Day Impacts in SHAPformer	159
Acronyms	166
Symbols	168
List of Figures	169
List of Tables	171
Bibliography	172

Part I

Introduction

1 Motivation

Svante Arrhenius discovered the greenhouse effect of atmospheric carbon dioxide (CO₂) in 1896 [15]. Meanwhile, the atmospheric concentration of CO₂ has risen from 284 parts per million (ppm) in the mid-19th century to 426 ppm in 2025 [227], primarily driven by the combustion of fossil fuels [228]. This rise in CO₂ concentration, alongside increases in other greenhouse gases such as methane and nitrous oxide, has led to a substantial warming of the Earth's climate system. Global mean surface temperatures for the period 2015–2024 were on average 1.26 °C higher than pre-industrial levels [185], and on average more than 1.5 °C higher for the years 2023–2025 [56].

Climate change has increasingly severe consequences, including rising sea levels, declining food and water security, more frequent and intense extreme weather events, ecosystem degradation, and biodiversity loss [42]. Above 1.5 °C global warming, the probability for cascading effects¹ increases with each additional increment of global warming [42].

To mitigate climate change, 194 states and the European Union (EU) adopted the Paris Agreement, committing to limit global warming to "well below 2 °C" and to "pursue efforts to limit the temperature increase to 1.5 °C above pre-industrial levels" [269]. However, current efforts are insufficient to meet these targets. The nationally determined contributions (NDCs) of the agreement's parties are projected to result in approximately 2.6 °C of global warming by 2100, while currently installed policies could lead to an even higher warming of around 3.1 °C [76]. These projections highlight the need for a rapid decarbonization across all sectors of human activity to avoid the most severe impacts of climate change.

1.1 Energy Transition

Decarbonization of the energy sector plays a central role in achieving net-zero greenhouse gas emissions, as the energy sector – a combination of electricity, heat, transport (including aviation and shipping), manufacturing, construction, buildings, other fuel combustion, and fugitive emissions – accounts for 77.2 % of global emissions [226]. However, the transition from fossil fuels to renewable energy sources introduces three major challenges:

- **Increased demand for electrical energy.** The electrification of heating, transport, and industry, together with the deployment of Power-to-X technologies, is expected to substantially increase electricity demand [175, 260].

¹ Cascading effects are consequences of climate change which lead to additional greenhouse gas emissions.

- **Increased volatility and uncertainty of electricity generation.** Photovoltaic (PV) and wind power are central to a fossil-free electricity supply. However, their generation is weather-dependent and thus volatile. While a reduction of the generation via curtailment is possible, the maximum available generation is constrained by weather conditions, which are uncertain in advance.
- **Increased decentralization of electricity generation and demand.** In the past, electricity generation was dominated by a small number of large fossil and nuclear power plants connected to the high-voltage grid. In contrast, renewable energy sources such as wind and PV systems are more decentralized, consisting of many smaller units often connected to medium- and low-voltage grids. At the same time, demand is becoming more decentralized, driven by the adoption of heat pumps and electric vehicle chargers.

These challenges complicate the operation of the energy system by reducing controllability and exposing limitations in existing grid infrastructure. A fundamental requirement is the continuous balance between electricity generation and demand to maintain system frequency [179]. At the same time, the grid must reliably transmit and distribute electricity while respecting the thermal limits of installed equipment. With increasing decentralization of both supply and demand, existing distribution grids are expected to reach their capacity limits [40].

Addressing these challenges requires a combination of grid expansion, grid reinforcement, increased deployment of energy storage, and greater demand-side flexibility. Storage systems enable temporal balancing by shifting energy from periods of surplus to periods of scarcity. Additionally, flexible loads such as heat pumps and electric vehicles can help align demand with supply, thereby reducing peak loads and alleviating grid stress. This flexibility can reduce the need for costly grid reinforcements [214, 263], which in Germany alone are projected to exceed 200 billion euros by 2045 [39].

1.2 Energy Forecasting

Forecasts are essential for planning measures to maintain grid frequency stability, ensuring that power flows remain within the operational limits of the power grid, and enabling the effective utilization of system flexibility. While forecasts are made on different timescales, ranging from short-term forecasts for hours to days into the future to long-term forecasts up to decades into the future [93], this dissertation focuses on short-term forecasts, which are relevant for the operation of the electrical grid. This section introduces two types of short-term energy forecasts: renewable energy forecasts and electrical load forecasts.

1.2.1 Renewable energy forecasts

Short-term renewable energy forecasts, including forecasts of the PV and wind power generation, are estimates of the available renewable energy at times in the near future. Renewable energy forecasts are relevant for transmission system operators (TSOs), who need to keep supply and demand in balance and to plan redispatch measures [36], and for distribution system operators (DSOs), who need to prevent equipment overloadings. By predicting peaks in renewable generation, grid operators can coordinate

flexibilities such as storage systems, electric vehicles, or controllable generation units. This allows for reducing curtailment of renewable energy and therefore for better renewable energy integration. In addition, accurate forecasts increase the profitability of renewable energy, because they reduce the risk of penalties when the sold energy cannot be delivered [289]. This is especially relevant for virtual power plants (VPPs) [302], which trade the aggregated energy from many small power plants.

Renewable energy forecasts are often based on weather forecasts [125]. Having a good weather forecast at hand, the task is to translate the predicted weather conditions and the forecast uncertainty into a (often probabilistic) forecast of electricity generation. Alternatives model inputs to weather forecasts are satellite images (to track clouds from above), or sky imaging cameras (to track clouds from the ground) [125].

1.2.2 Electrical load forecasts

Short-term electrical load forecasts are estimates of the electricity demand for times in the near future. Electrical load forecasts have various applications on different levels of the energy system. This section gives an overview on applications categorized into transmission grid level, distribution grid level, building level and residential level. Some of the applications require forecasts of the residual load, i.e., the load minus the renewable generation. The applications of electrical load forecasts usually involve a downstream optimization, e.g., of an appliance's schedule, based on the forecast.

Load forecast applications in the transmission grid The transmission grid operates at extra-high voltages and transmits power over long distances. The TSOs are in charge of keeping the overall system in balance, meaning that generation and demand match at every point in time to keep the frequency stable. The following applications exist at the transmission grid level.

- **Balancing supply and demand.** The TSOs have to maintain the frequency stability by keeping the electrical supply and demand in their control area in balance [38, 179]. Therefore, forecasts of both values are needed for predicting imbalances and planning balancing measures. These measures include the procurement of balancing energy and curtailment of renewable energy generation.
- **Congestion management [33].** Granular forecasts of the residual load at every grid node allow to estimate the grid state in the nearby future. When these grid states violate technical constraints of electrical transformers and power lines, redispatch measures can be planned to bring the system into a valid state. This can be supported by TSO-controlled energy storages to reduce the need for redispatch measures [71].
- **Cross-border transmission capacity estimation.** Load forecasts are used to define the available capacity for energy transmission between bidding zones of the electricity market [258].
- **Energy procurement.** Energy utilities buy the electricity needed by their customers in advance via long-term contracts or on the day-ahead and intraday markets [241]. Load forecasts are needed to estimate the amount of energy to procure.

- Energy trading [137]. Energy traders are entities that buy and sell energy – different from energy utilities that buy energy for their end customers. Load forecasts inform energy traders on when energy is needed, and can be used as indicators for future energy prices.

Load forecast applications in the distribution grid The distribution grid operates at high to low voltages and distributes power from the transmission grid to end customers or vice versa. Forecasting can help to foresee and prevent critical grid situations. In particular, load forecasts are used for the following applications in the distribution grid.

- Demand response [101], also called load shifting, means that electricity demand is shifted over time to match with the available generation. This can be achieved via price incentives.
- Demand-side management [201, 222, 265] means that the grid operator directly controls the generation and load based on a grid state estimation [104]. In Germany, Energiewirtschaftsgesetz (EnWG) §14a allows the DSO to control appliances in critical grid situations. In such situations, the DSO can reduce the power of heat pumps and electric vehicle chargers. To plan these measures in advance and inform the appliance owners, residual load forecasts at the level of substations or individual feeders are needed.
- Storage control [101, 142]. Storage systems such as batteries can be charged in times of high supply and low demand, and discharged when energy is needed. Storage systems in the low-voltage grid can help to avoid grid congestions in the form of violation of equipment constraints or voltage deviations.
- Co-ordinated electric vehicle (EV) charging [101]. EV chargers have high capacity and high simultaneity factors, as many people have similar daily schedules. Charging can be scheduled by the grid operator to avoid grid congestions. This application can be extended to vehicle-to-grid approaches, where the EV batteries are also discharged when energy is needed [222].
- Local electricity markets [101] are places where energy can be traded between neighbors of the same energy community, thereby consuming the energy close to where it is generated, which reduces grid expansion needs. Market participants require load forecasts to procure the needed energy.
- Balancing group forecasting [151]. The balancing group operators are obliged to deliver day-ahead forecasts to the transmission operator and are billed for inaccuracies in the form of balancing energy.
- Intervention effect estimation [101]. A comparison of the measured load with the expected load based on an accurate forecast model can be used as an estimate of the effect of an intervention such as a demand-response signal or price incentive.
- Anomaly detection [101]. The deviation from a forecast can indicate an anomaly, such as energy theft, measurement errors or changes in behavior.

Load forecast applications at building level At the level of buildings, which includes industrial buildings and public institutions such as research facilities, load forecasts can help to reduce electricity costs and increase consumption of locally generated solar power.

- Building energy management systems [204]. The use of flexibilities in buildings, including battery storages, heating, ventilation and air conditioning (HVAC), and EV charging is optimized based on forecasts of electrical load, heating and cooling demand, and renewable energy generation.
- Smart EV charging. In contrast to co-ordinated EV charging, where the grid operator controls the charging schedule, smart electric vehicle charging optimizes the self-consumption of locally produced renewable energy or the cost based on a dynamic electricity tariff. This application is explored in the Smart East project [84]. It can be extended to a vehicle-to-grid application where the vehicle owner makes profit by discharging the battery in times of high prices.
- Peak shaving [101]. Companies and research facilities in Germany do not only pay for the used electrical energy, but also pay a fee for their grid connection point based on the largest amount of power used in a 15-minute interval of a year. Forecasts can help with peak shaving, thereby reducing the grid connection fee and reducing the grid stress, as peaks can cause critical situations.
- Dispatchable feeder [285]. A dispatchable feeder is a schedule-based control algorithm that uses demand and storage flexibilities to follow a committed day-ahead schedule at the grid connection point.

Load forecast applications at residential level Households increasingly install PV panels and batteries, and energy utilities offer dynamic electricity tariffs, which give incentives to consume energy when it is abundant. Home energy management systems (HEMS) [23, 201] optimize the energy usage of flexible assets based on forecasts of the locally produced electricity, demand and the dynamic price, leading to reduced energy costs for the households.

1.2.3 Time-series Transformers

In 2017, Vaswani et al. [271] introduced the Transformer, a neural machine translation model based on the attention mechanism [24]. Since then, the Transformer has been adopted for a variety of tasks and data modalities, including contextualized word embeddings [66], large language models [219], computer vision [69], tabular data [120, 128], audio data [218], and multimodal learning tasks [132].

Given that Transformers excel at language modeling [219], they are also promising for time-series forecasting, which shares the same autoregressive structure. In both tasks, the next value in a sequence is computed on the basis of the preceding values. First approaches using Transformers for time-series forecasting report promising results [129, 294, 297]. However, other works claim that Transformers are incapable of capturing time-series structures and even simple linear models produce better forecasts [296]. Therefore, this dissertation studies if – and how – Transformers can be used effectively for electrical load forecasting.

The focus of this dissertation is electrical load forecasting, which is more suited to the Transformer architecture than renewable energy forecasting. Time-series Transformers follow an autoregressive structure, where forecasts are based on past observations. In contrast, renewable energy forecasts are often based on weather forecasts or satellite data instead of past observations [125].

1.2.4 Training strategies for multiple time series

In addition to the transition to renewable energy sources, the energy sector is transformed by the ongoing digitalization and evolution toward a smart grid [222]. In this context, Germany aims to equip 90 % of consumers having an annual electricity demand of at least 6 MWh or covered by §14a EnWG with smart meters until 2030². Currently, 26.1 % of these consumers are equipped with smart meters [37], whereas only 5.9 % of all consumers are equipped with smart meters [37]. In addition to smart meters for end consumers, distribution grid operators roll out measuring devices for grid substations, but only a fraction of the substations are currently monitored [266]. The increasing deployment of smart meters and measurement infrastructure at substations will increase the availability of fine-grained electrical load data in the near future.

When electrical load time series from multiple sources are available, different strategies for training forecasting models are possible, ranging from local models that are specialized for one time series to global models that learn to generalize across time series [95, 189]. Recently, Time-Series Foundation Models (TSFMs) [167] extended the global training strategy to training models on large datasets containing time series from different domains, as well as synthetic data. Part II evaluates training strategies for the Transformer in forecasting scenarios with multiple load time series, and Parts III and IV deal with the predictive performance and transparency of TSFMs.

1.2.5 Benchmarking of electrical load forecasting methods

A variety of time-series Transformer architectures have been proposed [282], but the evidence for their applicability to electrical load forecasting is limited (see the discussion of related work in Chapter 2). When a new forecasting method is proposed, it is best practice to compare it with established forecasting methods to assess whether it is an improvement over the state of the art [116]. Therefore, Part III compares Transformer models with established load forecasting methods. Because load forecasts are applied on different levels of the energy system, the comparison includes datasets from the TSO level to the client level.

1.3 Explainable Artificial Intelligence (XAI)

Machine learning and deep learning models, including Transformers, often involve a large number of parameters, making their internal decision processes opaque to humans. However, model transparency is

² Messstellenbetriebsgesetz (MSBG) §45.

valuable for several reasons. First, it can enhance user trust and facilitate the adoption of deep learning in real-world applications [61]. Second, it helps users assess the reliability of predictions and decide how to act on them [206]. Third, it assists developers in debugging and refining models – for example, by revealing spurious patterns (sometimes called "Clever Hans" effects) [159]. Fourth, it supports compliance with emerging regulations that demand transparency in "Artificial Intelligence" (AI) systems. The EU's AI Act, for instance, mandates transparency for applications involving humans or critical infrastructure, which includes energy systems [78]. These reasons motivate the development of XAI methods, which aim to improve the transparency of how machine learning and deep learning models operate [6]. While not always required to meet regulatory standards, XAI can support human oversight and improve transparency [206], and may justify future regulatory standards for transparency in AI systems. However, existing XAI methods are costly to compute, especially for large models like Transformers and TSFMs [30]. Therefore, Part IV proposes efficient XAI methods for time-series Transformers and TSFMs.

2 Related Work and Contributions

This chapter presents the status quo of Transformers and XAI for electrical load forecasting, and identifies open research questions, which are addressed in the subsequent chapters. Section 2.1 describes the state of the art in electrical load forecasting. Section 2.2 introduces previous work on time-series Transformers and Section 2.3 describes training strategies for time-series models. Together, these two sections motivate research about training strategies for time-series Transformers (Part II). Section 2.4 presents best practices for benchmarking load forecasting methods, as well as existing time-series forecasting benchmarks. This highlights the need for benchmarking Transformer architectures against established load forecasting methods (Part III). Section 2.5 presents related work on XAI for time-series forecasting, motivating research on explainability methods tailored to time-series Transformers and TSFMs (Part IV). Section 2.6 motivates the usage of Transformer-based load forecasts in down-stream applications (Part V). Finally, Section 2.7 gives the outline of the dissertation.

2.1 Electrical Load Forecasting

Electrical load forecasting is a very active research field, with more than a thousand publications per year¹. Despite the large number of publications, only a few methods are commonly used. The goal of this section is to identify these common methods, which will later be used for benchmarking. To identify relevant comparison methods, this section summarizes literature reviews, results of load forecasting competitions, and recent publications on electrical load forecasting.

2.1.1 Literature reviews

Hong [121] presents a history of electrical load forecasting from the 1940's to the mid-2010's. Early forecasting methods are based on the similar day method or on statistical models such as Exponential Smoothing and Autoregressive Integrated Moving Average (ARIMA). A review of low-voltage load forecasting [100] confirms that statistical methods, namely ARIMA, nonlinear regression and Exponential Smoothing, are still frequently used for load forecasting.

In the last decades, machine learning methods became increasingly popular for electrical load forecasting. Reviews on machine learning for electrical load forecasting largely agree on the most common methods [100, 103, 122, 125, 156, 237]. The review by Kuster et al. [156] from 2017 states that there is no single best-performing method; instead, the best method for a particular load forecasting task depends

¹ A Scopus search for publications with "load forecast*" in the title returns 1398 works published in the year 2025, evaluated on 14.04.2026.

on characteristics such as the aggregation level, location, data quality and forecast horizon. For short-term forecasts, Support Vector Regression (SVR), artificial neural networks (ANNs), and ARIMA were most common [156]. Scheidt et al. [237] conclude from their review of data analytics in the energy sector published in 2020 that ARIMA, ANNs, hybrid approaches based on ANNs, and SVR are the most common methods for short-term electrical load forecasting. A review on electrical load forecasting in the low-voltage grid published in 2021 by Haben et al. [100] finds that k-Nearest-Neighbor (kNN), regression trees, SVR, and ANNs are common methods. Next to fully connected ANNs, Convolutional Neural Networks (CNNs) [8] and long short-term memories (LSTMs) [152] are used as load forecasting models. Gasparin et al. [87] provide an overview of CNNs and LSTMs architectures for electrical load forecasting. In the newest review by Hasan et al. [103] from 2025, time-series Transformers [297] and Temporal Fusion Transformer (TFT) [90] are mentioned in addition to the methods contained in the previous reviews.

Based on these reviews, it can be concluded that only a limited set of established load forecasting methods exists. This set includes statistical methods (similar day method, Exponential Smoothing, ARIMA, nonlinear regression), machine learning methods (SVR, kNN, tree-based models), and deep learning methods (ANN, CNN, LSTM). The next sections describe which methods performed best in load forecasting competitions and in recent publications.

2.1.2 Load forecasting competitions

Load forecasting competitions evaluate many different approaches of participants from academia and industry under equal conditions. Therefore, they are good indicators of the state of the art in electrical load forecasting. Hong et al. [125] present a history of electrical load forecasting competitions. In one of the first competitions organized by Puget Sound Power and Light Company in the 1990s, a multiple linear regression model performed best [221]. In 2001, a competition by EUNITE was won by an SVR-based approach [49]. The Global Energy Forecasting Competitions (GEFComs) 2012, 2014, and 2017 [123, 124, 126] were dominated by multiple linear regression, ANNs [118, 145], SVRs [49], generalized additive models (GAMs) [82], and gradient boosting [126].

In addition to the models mentioned in the literature reviews, gradient boosting and GAMs are relevant comparison methods due to their strong performance in load forecasting competitions.

Table 2.1: Related work on benchmarking electrical load forecasting models.

Publication & year	Dataset(s)	Models (<u>best underlined</u>)
Amarasinghe et al. [8], 2017	IHEPC [89] (no weather)	<u>LSTM</u> , CNN, SVR, ANN, Boltzmann machine
Voß et al. [272], 2018	Pecan Street households [209]	<u>CNN</u> , <u>ANN</u> , LR, kNN
Groß et al. [96], 2021	59 buildings	<u>kNN</u> , <u>MLP</u> , SVR, ARIMA, LR, LSTM, RF
Gasparin et al. [87], 2022	IHEPC [89] (no weather), CBT [55] (no weather), GEFCom 2014 [124]	<u>GRU</u> , <u>CNN</u> , <u>MLP</u> , <u>LSTM</u> , RNN
Jain et al. [135], 2022	SCiBER Dataset [256] (no weather)	<u>LSTM</u> , <u>RF</u> , naive baseline, ARIMA
Choi and Lee [52], 2023	Simulated building	<u>MLP</u> , <u>LSTM</u> , CNN, RNN, GRU, Transformer
Dai et al. [58], 2023	Singapore and Switzerland	<u>CNN+LSTM+attention</u> , RNN, XGB
Emami et al. [74], 2023	BuildingsBench [74] (no weather)	<u>Transformer</u> (pre-trained), LR, RNN, LightGBM
Giacomazzi et al. [90], 2023	20 US and 70 German substations	<u>TFT</u> , <u>LSTM</u> , ARIMA
Ibrahim and Hossain [130], 2023	Smart-grid smart-city customer trial [19]	<u>LSTM</u> , GRU, RNN, MLP, Resnet
López Santos et al. [173], 2023	6 buildings in Spain	<u>TFT</u> , MLP, LSTM
Pelekis et al. [210], 2023	National load of Portugal	<u>N-BEATS</u> , MLP, LSTM, CNN, TFT
Shahare et al. [246], 2023	Real-world utility	<u>CNN-LSTM</u> , LSTM, CNN, ANN, SVR, ES
Shi et al. [250], 2023	City in China	<u>CNN+LSTM+attention</u> , LSTM, GB, RF
Upadhyay et al. [270], 2023	London Smartmeter [268]	<u>Transformer</u> , LSTM, GRU, CNN-LSTM, CNN, LightGBM, CatBoost, RF
Cen and Lim [43], 2024	Office building in Bangkok (no weather)	<u>PatchTST</u> , LSTM, GRU, CNN, Transformer, Informer, Autoformer
El-Azab et al. [20], 2024	ISO New England	<u>MLP</u> , <u>LSTM</u> , GRU, ANFIS
Jain and Gupta [134], 2024	Territory in India	<u>LSTM</u> , MLP, RNN, SVR, ensemble
Maleki et al. [182], 2024	Swedish utility	<u>CNN</u> , ARIMA, Prophet, XGB, RF
Meyer et al. [186], 2024	OPSD household data [287], Lower Saxony Dataset [240], East Midlands Dataset [193], Lothian Dataset [215] (all without weather)	<u>TimesFM</u> , PatchTST, Chronos, Transformer, TFT, iTransformer, Lag-Llama
Saravanan et al. [235], 2024	BuildingsBench [74] (no weather)	<u>Transformer</u> (pre-trained), TimesFM, Lag-LLama, Moirai, Chronos, PatchTST, TFT, LightGBM, RNN, LR
Tziolis et al. [267], 2024	University of Cyprus net load	<u>BNN</u> , RF, LR, RNN, XGB, kNN, GB, MLP, SVR
Waheed and Xu [273], 2024	IEEE dataset	<u>MLP</u> , SVR, RF, ARIMA
Zhang and Jánošík [298], 2024	Malaysian city [232]	<u>LightGBM</u> , CatBoost, XGB
Asghar Majeed et al. [16], 2025	Beijing load	<u>LSTM</u> , GRU, RNN, CNN
Moustati et al. [192], 2025	CBT [55] (no weather)	<u>LSTM</u> , ARIMA, SARIMA, CNN, SVR, ANN, kNN, decision trees
Lin et al. [170], 2025	London smart meters, Germany, Netherlands	<u>Chronos</u> , TimesFM, Moment, Lag-Llama, TimeGPT, GP, SVR
Moosbrugger et al. [191], 2025	London smart meters, demandlib	<u>Transformer</u> , LSTM, xLSTM, kNN
Katholnigg and Goebel [142], 2025	EMSx industrial data [161] (no weather)	<u>N-BEATS</u> , ensemble, TFT, LSTM, CNN
Papadopoulos et al. [207], 2026	Zero-emission buildings	<u>SVR</u> , <u>baseline</u> , XGBoost, CatBoost, LightGBM, N-HITS, LSTM, TFT, TiDE, KAN

2.1.3 Recent publications

Next, we analyze recent publications that compare load forecasting models. This analysis shows the latest developments since the literature reviews and competitions. Table 2.1 summarizes the analyzed publications by listing the datasets, models, and best-performing approaches from each publication. As stated by Kuster et al. [156], there is no single best method outperforming the others on all datasets. Deep learning methods achieve the best results in almost all publications. However, they are occasionally outperformed by kNN, LightGBM, RFs, and even by the naive baseline. Among the deep learning models, RNNs and their variants (LSTMs and GRUs) are often best, together with combinations of CNNs with LSTMs (i.e., CNN-LSTMs). All of these models are general deep learning models that can be used for different types of data, whereas models specific to time series (e.g., N-BEATS [202], DeepAR [234], Profile Neural Networks [106], TFT [168], N-HITS [44]) are less common in the electrical load forecasting literature. Transformer-based models become increasingly popular and usually outperform other models when included in the experiments. However, there are many variants of time-series Transformers, which are summarized in Section 2.2, and comparisons between them are rare in the load forecasting literature. TSFMs perform well in some newer works [170, 186], which shows their potential for electrical load forecasting.

Overall, the recent literature largely confirms the literature reviews in their method selection, but newer methods – including time-series Transformers and TSFMs – gain popularity. Due to the fact that no single method performs best on all datasets, it is a best practice to use automated machine learning (AutoML) methods to select models and optimize their hyperparameters [184].

2.2 Time-Series Transformers

Prior to the development of Transformer architectures for time-series forecasting, attention mechanisms were already incorporated into other time-series models. Lai et al. [158] propose LSTNet, which combines convolutional layers, recurrent neural networks, and attention to capture both short- and long-term dependencies. One of the first applications of the original Transformer architecture [271] to time-series forecasting is presented by Wu et al. [294] for influenza prevalence forecasting. A broader empirical comparison by Agarwal et al. [3] evaluates several deep learning models, including the Transformer, across multiple datasets, showing that Transformer-based approaches can be competitive, though not consistently superior.

Later, a variety of Transformer architectures for time-series forecasting were introduced, including LogSparse Transformer [165], Informer [300], Autoformer [292], TFT [168], FEDformer [301], and PatchTST [199]. More recently, Transformer-based TSFMs have been proposed [167], including TimeGPT [86], Chronos [12], TimesFM [60], Moirai [290], Lag-Llama [223], Chronos-2 [11], and Moirai 2.0 [171]. The Transformer architectures and selected TSFM architectures are described in Chapter 3.

2.3 Training Strategies

An increasing number of buildings and substations is equipped with measurement devices, leading to an increase in available load data from heterogeneous sources. When time series are recorded at multiple locations in the grid, several strategies for training load forecasting models can be employed.

2.3.1 Existing training strategies for multiple time series

When multiple time series of the same type are available, such as time series measured at different substations, feeders or buildings, there are several ways to leverage this data for load forecasting models.

Multivariate model A common strategy is to train one multivariate model, which gets the past observations from all time series as input and generates a forecast for all of them jointly. This approach is adopted by multiple works on time-series Transformers [172, 277, 292, 300, 301].

Local models A simple training strategy is to train one forecasting model per time series. These models are then called local models [101].

Global model A third training strategy is to train a global model, which predicts each time series independently, but is trained on the data from all time series [95, 101, 189]. This has the advantage that more training data is available than for local models, which can potentially result in a better model. Global load forecasting models are already used with CNNs [272], DeepAR [234], and N-BEATS [95].

Cluster-specific models² A compromise between the local and global training strategies can be achieved by clustering similar time series and training one model per cluster. For the clustering algorithm, the time series are represented as low-dimensional vectors. This representation can be based on features [81, 131, 160, 166, 217, 257, 288], average daily load diagrams [46, 247], or excerpts from the time series [83, 146]. K-means is the most common clustering algorithm [131, 146, 160, 166, 217, 247, 257, 288], and only few publications use other clustering algorithms such as greedy clustering [288], hierarchical clustering [83], fuzzy c-means (FCM) [81], affinity propagation [46], and k-medoids [257]. The cluster-specific training strategy has been shown to improve a variety of load forecasting models, including linear regression [288], SVR [81, 288], MLPs [81, 131, 146, 166, 217, 247, 288], and a combination of neural networks [46]. However, sometimes only a subset of the evaluated models is improved [46, 83, 160] or the model accuracy worsens [257]. Only one publication compares cluster-specific models with local and global models [276], while the other publications compare only with local models [160, 257] or only with global models [46, 81, 83, 131, 146, 166, 217, 247, 288]. Most publications use client-level data [46, 81, 83, 131, 146, 160, 166, 217, 247, 276, 288], whereas substation data [257, 276] and industry data [160] is underrepresented in the literature.

² This paragraph is a modified excerpt from Hertel et al. [109], © 2025, IEEE.

Foundation model TSFMs [167] such as TimeGPT [86], Chronos [12], TimesFM [60], Moirai [290], Lag-Llama [223], Tiny Time Mixers [72], Chronos-2 [11], Moirai 2.0 [171], as well as tabular foundation models [120] (which are also applicable to time series [127]), are pretrained on large time-series datasets including data from various domains and often synthetic data [11, 120]. This is an extension of the global training strategy towards pretraining on time series from different datasets. The previously mentioned TSFMs rely on Transformers, but there are also TSFMs based on other neural network architectures [18, 73, 291].

Transfer learning The idea of transfer learning is to train a model on a given dataset and task and then apply it to a different dataset or different task. Global forecasting models and TSFMs can be seen as transfer learning, because in both cases the model is trained on many time series and then applied to a specific time series of interest. According to the taxonomy of [117], the global training strategy is transductive, since the training task and the target task are the same. However, the data distributions for training and inference can be different. Two works [85, 195] use transfer learning for time-series Transformers by pre-training on one time series and fine-tuning on another, either applied to buildings [195] or districts [85]. In contrast to these works, the global training strategy trains a generalized model on the data from many time series instead of just one.

2.3.2 Research gap regarding Transformer training strategies

Previous work on time-series Transformers focuses on multivariate models or settings where only one time series is available [172, 277, 292, 300, 301]. Zeng et al. [296] show that simple linear models outperform multivariate Transformers and local Transformers and question the effectiveness of Transformers for time-series forecasting in general. However, the global training strategy and the cluster-specific training strategy may be advantageous for the Transformer, as they allow to train the model on more samples. On the other hand, global models need to generalize across heterogeneous time series, which increases the difficulty of the task. This leads to the first research question investigated in this dissertation:

Research question (RQ) 1

Does the forecast error of the Transformer improve by using global or cluster-specific training strategies in scenarios with multiple electrical load time series?

2.4 Benchmarking of Load Forecasting Models

This section summarizes best practices for benchmarking load forecasting models, motivates the need for dedicated benchmarking despite the availability of several large-scale time-series benchmarks, and identifies a gap in the existing evaluation of time-series Transformers.

2.4.1 Best practices for benchmarking

Multiple reviews on electrical load forecasting [100, 125, 237] and forecast evaluation [116] provide recommendations for benchmarking. The most important practices are summarized as follows:

1. **Experiments on multiple datasets:** Models should be evaluated on multiple datasets with varying characteristics (e.g., aggregation level, forecast horizon) to ensure robustness and generalizability. For reproducibility, datasets and code should be made publicly available.
2. **Comparison to baselines and state-of-the-art methods:** New methods should be compared against a diverse set of baselines, including simple, statistical, and deep learning models, to demonstrate improvements in accuracy, efficiency, or interpretability.
3. **Use of relevant features:** Forecasting models should incorporate informative features such as lagged load values, calendar variables, and weather data, and use sufficiently long contexts (e.g., at least one week to represent weekly seasonality).
4. **Hyperparameter optimization:** All models should be carefully tuned using a suiting validation set, as suboptimal hyperparameters can lead to misleading comparisons.
5. **Appropriate metrics and significance testing:** Multiple evaluation metrics should be reported and statistical significance tests should be used to support reliable comparisons between models.

2.4.2 Existing time-series forecasting benchmarks

Several large benchmark datasets have been proposed for the comparison of time-series forecasting methods. However, their applicability to electrical load forecasting is often limited due to differences in temporal resolution, availability of covariates, or domain characteristics.

Early large-scale benchmarks such as the ones used in the M4 and M5 forecasting competitions [180, 181] focus on univariate forecasting tasks with predominantly low-frequency data (i.e., daily or monthly resolution) and non-energy domains, which restricts their relevance for load forecasting.

The Monash Time Series Forecasting Archive [92] provides a collection of 25 datasets, including a subset of the Electricity dataset [229] and Australian electricity demand as the only two electrical load datasets. While this benchmark enables consistent evaluation across domains, it does not include covariates, limiting its suitability for realistic load forecasting scenarios.

BuildingsBench [74] specifically targets building-level load forecasting and combines large-scale synthetic data with real-world datasets. Although it includes calendar features, it lacks weather data, which is a key driver of electricity demand. The benchmark demonstrates that pretrained Transformer models outperform classical approaches in this setting.

Fev-bench [249] comprises 100 forecasting tasks, including 26 from the energy domain, many of which provide covariates. Fev-bench contains electrical load data from ENTSO-E and from the GEFComs 2012, 2014 and 2017. Chronos-2 [11] outperforms other TSFMs on this benchmark [249], but no comparisons are made to models trained for specific datasets.

GIFT-Eval [4] is a comprehensive benchmark covering multiple domains, frequencies, and forecast horizons. Aksu et al. [4] compare a wide range of statistical methods, deep learning models, Transformers, and TSFMs [4]. Recent results indicate that Chronos-2 achieves state-of-the-art performance on this benchmark [11]. However, GIFT-Eval contains only one electrical load dataset (the Electricity dataset).

While existing benchmarks indicate which models work well on a variety of datasets, their limitations regarding covariates and domain specificity motivate evaluations on diverse electrical load datasets.

2.4.3 Research gap regarding the evaluation of time-series Transformers for electrical load forecasting

Existing work on time-series Transformer architectures does not follow best practices for benchmarking. The works on LogSparse Transformer [165] and TFT [168] train global models and evaluate on multiple datasets, including the Electricity dataset [229], but use only one week as the test set. The works on Informer [300], Autoformer [292], FEDformer [301], DLinear [296], iTransformer [172], and TimeXer [277] only train multivariate and local Transformer models, but no global models. Informer [300] evaluates on four datasets, including a subset of the Electricity dataset and a weather dataset. They evaluate models for predicting the weather 720 hours ahead, a task that is impossible even for the best numerical weather prediction models. Autoformer [292] replaces the weather dataset by another dataset with the same name (without saying so explicitly in the paper). They fix the input length of all models to a suboptimal value of 96 time steps and they add a stock exchange rate dataset, where their model is outperformed by a simple consistency baseline [116]. FEDformer [301] adopts the setup from Autoformer without global training and with 96 input time steps. Zeng et al. [296] show that simple linear baselines outperform the multivariate and local Informer, Autoformer and FEDformer, thereby questioning the effectiveness of time-series Transformers in general, but they do not evaluate global Transformers. PatchTST [199] is a global model that outperforms the linear models, although not using relevant covariates such as calendar and weather features. Despite the fact that PatchTST [199] and our work done in parallel [110] show the importance of global training and long context, iTransformer [172] and TimeXer [277] continue the multivariate and local training and restrict the input length for all baselines to 96 time steps. Thereby, they claim to outperform linear models and PatchTST with their approaches, although their results are worse than the results reported in the original sources with longer context [199, 296].

Due to limited evaluations on electrical load forecasting datasets, rare use of covariates, and other suboptimal settings in related work (like short input lengths and multivariate training), it is not clear whether the architectural modifications proposed in the related work are beneficial for short-term electrical load forecasting under realistic settings. Therefore, this dissertation benchmarks different Transformer architectures against established load forecasting methods on multiple datasets and grid levels. The addressed research question is:

Research question (RQ) 2

Does an optimized Transformer architecture outperform established methods for electrical load forecasting on datasets from varying grid levels?

2.5 Explainability and Interpretability of Time-Series Forecasting Models

The term XAI is often used as a subordinate term for *interpretable* and *explainable* machine learning methods. In agreement with Flora et al. [79], this dissertation defines the two terms as follows:

- *Interpretable* machine learning models can be understood without additional tools, by analyzing the trained models themselves.
- *Explainable* machine learning methods are tools to create insights into the functioning of trained models, including models that are not interpretable by themselves.

This section first reviews existing XAI methods for general applications, as well as XAI methods applied to time series, forecasting, and Transformer models. It then motivates the need for explainable Transformer-based forecasting models.

2.5.1 Existing XAI methods

This section introduces general XAI methods and then reviews XAI for time series, XAI for Transformers, XAI for energy systems, and XAI for load forecasting.

General XAI methods Adadi and Berrada [2] review XAI methods that let the user understand the predictions of a black-box machine learning model. They note a tradeoff between inherent interpretability of a model and its predictive performance. Some models are inherently interpretable, such as linear models, additive models, and rule sets. As more complex and therefore more powerful models are less interpretable, they need to be explained by post-hoc methods. However, these methods are often costly to compute. Local Interpretable Model-Agnostic Explanations (LIME) [225], SHAP [176], Grad-CAM [242], and layer-wise relevance propagation (LRP) [22] are commonly used post-hoc methods. XAI methods are classified into whether they create global explanations, i.e. explanations of the model across many predictions, or create local explanations, i.e. explanations of individual predictions. Additionally, an XAI method can be model-specific, or model-agnostic such as LIME and SHAP.

XAI for time series Rojat et al. [230] review XAI for time-series forecasting, classification and clustering. For convolutional neural networks, class activation mappings [278], gradient-based methods [253], and perturbation-based methods [141] are used. For RNNs and Transformers, attention mechanisms [113, 140] and TFT [168] are used. Next, Rojat et al. [230] classify some forecasting techniques as explainable. This includes symbolic aggregate approximation (SAX) [245], which discovers time series patterns that are important for each class, and fuzzy cognitive maps [274], which allow to analyze the connection weights between concepts. Finally, methods based on representative examples are used, such as nearest-neighbor-based approaches [162] and time-series shapelets [295], where the interpretability is achieved by retrieval of the representative examples (neighbors or shapelets). The intended audience for the explanations ranges from model developers to end-users and decision makers. Most explanation

methods are not evaluated at all, but could be evaluated qualitatively by the end-users [230]. Quantitative evaluations are done by a comparison to existing attribution methods or perturbation analysis [238]. Rojat et al. [230] conclude that most methods highlight important input regions, which is understandable in computer vision, but can be hard to interpret for time series due to their unintuitive nature.

Theissler et al. [259] review XAI for time-series classification. They distinguish between time point-based explanations, subsequences-based explanations, instance-based explanations and counterfactual-based explanations. Time-point based explanation methods assign a score to each time point. The methods are attribution-based, such as gradient-based methods [278], LIME [225], and SHAP [176], or attention-based, e.g. [140]. Instead of time points, a score can also be assigned to a subsequence of the time series [98, 99, 245]. Instance-based explanations rely on the entire time series to derive an explanation. The explanation can be feature-based [88] or prototype-based, e.g. returning examples that are similar in the embedding space [62]. Counterfactual-based explanations [65] rely on minimal changes to a time series that lead to a change of the predicted class. The evaluation of XAI methods is done visually or by analyzing the explanation faithfulness, e.g. via perturbation analysis [239].

XAI for Transformers Transformers are the state of the art in natural language processing and computer vision and there are multiple XAI methods based on their attention mechanism. A common XAI method for Transformers is to visualize the model's attention maps. Such a visualization can be averaged across the individual attention-heads of all layers of the model, or it can be done for individual attention heads. Individual attention heads encode syntactic and semantic relationships between words in the masked language model Bidirectional Encoder Representations from Transformers (BERT) [53] and generative language models [138]. However, there is an ongoing debate whether attention scores are meaningful explanations of a model [34, 136, 286]. A difficulty of using attention for XAI is that embeddings in higher levels already aggregate information from the entire input. Attention rollout is a method which addresses this by computing the attention flow between layers [1]. Another approach is attention manipulation [64], where the model's attention mechanism is manipulated so that the model does not have access to some inputs or that the influence of some inputs is increased.

As Transformers are increasingly used for time-series tasks, there are also time-series XAI approaches based on them. A common approach is to visualize the model's attention scores [85]. LogSparse Transformer [165] and Informer [300] use the fact that the canonical time-series Transformer has sparse attention weights as a motivation for their sparse attention mechanisms. FEDformer [301] shows that the attention in the frequency-decomposed feature space is sparse and use it as an argument for the "compactness" of their time-series representation. TFT [168] uses feature selection networks to compute context-dependent feature importance and analyzes the attention maps to find important lag time steps.

XAI for energy and power systems Machlev et al. [178] review applications and methods of XAI for energy and power systems. The applications of XAI are grouped into three fields: power grid applications such as frequency analysis [155] and transient stability modeling [224], energy applications such as renewable energy forecasting [236] and imbalance forecasting [262], and building energy management

applications such as cooling load forecasting [283] and benchmarking of buildings [14]. The most dominant XAI method in these fields is SHAP, followed by LIME and attention methods [178].

XAI for load forecasting Baur et al. [30] present a review on explainable and interpretable electric load forecasting methods. They note an increasing trend in the number of publications on this topic. Apart from classically interpretable models, such as linear regressors [187] and decision trees [190], they mention time-series decomposition methods [233], fuzzy logic [255], and probabilistic modeling [122] as methods that add interpretability in load forecasting. They mention the computation of feature importance with tree-based methods such as XGBoost [164] and post-hoc methods such as LIME [97] and SHAP [293], and the analysis of attention scores [163] as approaches towards explainability for load forecasting. Neumann et al. [197] propose an alternative approach to XAI for load forecasting, attributing feature importance in the spatial domain rather than in the time domain. Baur et al. [30] discuss multiple open challenges in current research on interpretable and explainable load forecasting methods. One challenge is the difficulty to evaluate XAI methods due to the unavailability of ground truth explanations. To address this challenge, we provide a synthetic dataset with ground-truth explanations in Chapter 4. Another challenge is that time series are auto-correlated due to their sequential nature, a fact that is typically ignored by post-hoc methods [169]. We discuss in Chapter 9 how our method SHAPformer deals with this challenge.

2.5.2 Research gap regarding explainable time-series Transformers

A key limitation of post-hoc XAI methods like LIME and SHAP is that they need to run a model many times to generate an explanation, which makes them computationally expensive [157]. Baur et al. [30] conclude that this is the reason why these XAI methods have not been applied to Transformer-based load forecasting models yet. This motivates the third research question:

Research question (RQ) 3

How can the attention mechanism be leveraged for efficient interpretable and explainable electrical load forecasting?

A common approach in natural language processing (NLP) and Computer Vision is to visualize the attention weights and use them as a proxy for which inputs are important for the prediction. This approach is adopted for time-series Transformers in Chapter 8. Next, attention offers a way to suppress the access of the model to specific inputs by manipulating the attention weights. This mechanism is used in Chapter 9 to develop a new approach for efficient and sampling-free estimation of SHAP values, called SHAPformer. Then, an efficient SHAP algorithm for TSFMs is introduced in Chapter 10.

2.6 Applications of Electrical Load Forecasts

An accurate forecast in itself has no value for the energy transition. Instead, electrical load forecasts are used in down-stream applications that reduce costs or carbon emissions and thereby support the energy

transition. Two applications are investigated in this dissertation: forecasting peak loads in the Big Data Energy Analytics Laboratory (BigDEAL) challenge, and peak shaving for an energy community with heat pumps and battery storage. Thereby, this dissertation addresses the following research question:

Research question (RQ) 4

Are Transformer-based electrical load forecasts beneficial for down-stream peak forecasting and peak shaving applications?

2.7 Dissertation Outline

Chapter 1 motivated the need for electrical load forecasts due to climate change and the ongoing energy transition. Chapter 2 presented the status quo on Transformers and explainability in time-series forecasting and stated the research questions that are addressed in this dissertation. Chapter 3 introduces the preliminaries about time-series forecasting, Transformers and SHAP. Chapter 4 presents the datasets that are used throughout the dissertation, thereby concluding Part I.

Part II is about Transformer training strategies, thereby addressing RQ 1. It consists of Chapter 5 with a comparison of local, global and multivariate models, and Chapter 6 analyzing cluster-specific models.

Part III is about benchmarking. It compares Transformer architectures and benchmarks them with established methods in Chapter 7, thereby addressing RQ 2.

Part IV addresses the interpretability aspects of Transformer models, thereby addressing RQ 3. Chapter 8 analyzes the attention weights. Chapter 9 introduces SHAPformer, an efficient implementation of SHAP for Transformers. Chapter 10 extends the SHAPformer methodology to TSFMs.

Part V evaluates Transformer-based forecasts in two down-stream applications, thereby addressing RQ 4. Chapter 11 presents results from the BigDEAL challenge on peak load forecasting. Chapter 12 presents a peak shaving application in an energy community with heat pumps.

Part VI concludes the dissertation with a summary (Chapter 13), as well as limitations and an outlook on potential future work (Chapter 14).

3 Preliminaries

This chapter introduces the fundamentals of time-series forecasting, the Transformer architecture for text translation, its adaption for time-series forecasting, and SHAP as a model-agnostic explanation method. These methods build the foundation for the subsequent chapters.

3.1 Time-Series Forecasting

This section defines the task of time-series forecasting, introduces evaluation metrics, and briefly describes probabilistic forecasts. The notation used in this dissertation is an adaption from Meisenbacher [183].

3.1.1 Task definition

A time series $\mathbf{y}$ is a sequence of measurements $\mathbf{y} = (\mathbf{y}[1], \dots, \mathbf{y}[T])$ of length T . We assume the measurements to be equidistant, meaning that the time between $\mathbf{y}[t]$ and $\mathbf{y}[t + 1]$ is the same for all $t \in \{1, \dots, T - 1\}$.

The goal in time-series forecasting is to make a prediction at a time t about the values of $\mathbf{y}$ during the next H time steps. The parameter H is called the forecast horizon. This means that the values $\mathbf{y}[1, \dots, t]$ are known at time t , and a forecast $\hat{\mathbf{y}}_t[t + 1, \dots, t + H]$ is to be made, which is close to the ground truth $\mathbf{y}[t + 1, \dots, t + H]$.

An autoregressive forecasting model f often does not rely on all previous values, but only the last C values, where C is called the context length. That is, a solely autoregressive forecast is created by the formula:

$$\hat{\mathbf{y}}_t[t + 1, \dots, t + H] = f(\mathbf{y}[t - C + 1, \dots, t]) \quad (3.1)$$

A forecasting model may not only rely on the past values of the target time series $\mathbf{y}$, but can also incorporate covariates $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_E)$ (also called exogenous variables). Covariates are also time series, defined as $\mathbf{x}_i = \mathbf{x}_i[1, \dots, T]$. The covariates can be split into a set $\mathbf{X}_{past}$ of covariates that are only available in the past and a set $\mathbf{X}_{future}$ of covariates that are available in the past and the future, either because they are known (such as calendar features) or forecasted (such as weather variables). Then, a forecasting model has access to the past covariates until time t and to the future covariates until time $t + H$:

$$\begin{aligned} \hat{\mathbf{y}}_t[t + 1, \dots, t + H] = f(\mathbf{y}[t - C + 1, \dots, t], \\ \mathbf{X}_{past}[t - C + 1, \dots, t], \\ \mathbf{X}_{future}[t - C + 1, \dots, t + H]) \end{aligned} \quad (3.2)$$

3.1.2 Evaluation metrics

There are multiple metrics that are used to evaluate the accuracy of a forecasting model. The metrics are defined for forecasts of one target time series $\mathbf{y}$. To evaluate models on multiple time series, the metrics are computed per time series and then averaged. Let $\hat{\mathbf{Y}} = (\hat{\mathbf{y}}_{t_1}, \dots, \hat{\mathbf{y}}_{t_n})$ be the collection of n forecasts for $\mathbf{y}$ made at different time steps $t_1, \dots, t_n$.

Mean absolute error (MAE) The MAE measures the absolute differences between forecast and true values. It can be interpreted in the same unit as the time series:

$$\text{MAE}(\hat{\mathbf{Y}}, \mathbf{y}) = \frac{1}{n \cdot H} \sum_{t=t_1}^{t_n} \sum_{i=1}^H |\mathbf{y}[t+i] - \hat{\mathbf{y}}_t[t+i]| \quad (3.3)$$

Mean squared error (MSE) The MSE measures the squared differences between forecast and true values, thereby punishing larger deviations more. Its unit is the squared unit of the time series, making interpretation more difficult:

$$\text{MSE}(\hat{\mathbf{Y}}, \mathbf{y}) = \frac{1}{n \cdot H} \sum_{t=t_1}^{t_n} \sum_{i=1}^H (\mathbf{y}[t+i] - \hat{\mathbf{y}}_t[t+i])^2 \quad (3.4)$$

Root mean squared error (RMSE) The RMSE is the square root of the MSE, which has the same unit as the time series:

$$\text{RMSE}(\hat{\mathbf{Y}}, \mathbf{y}) = \sqrt{\text{MSE}(\hat{\mathbf{Y}}, \mathbf{y})} \quad (3.5)$$

Normalized mean absolute error (nMAE) The nMAE is the MAE divided by the mean of the absolute time series $\bar{y}_{\text{abs}} = \frac{1}{T} \sum_{t=1}^T |y[t]|$. This allows for a comparison of errors between time series of different magnitudes:

$$\text{nMAE}(\hat{\mathbf{Y}}, \mathbf{y}) = \frac{\text{MAE}(\hat{\mathbf{Y}}, \mathbf{y})}{\bar{y}_{\text{abs}}} \quad (3.6)$$

Normalized mean squared error (nMSE) Analog to the nMAE, the nMSE is the MSE normalized by the mean of the absolute time series:

$$\text{nMSE}(\hat{\mathbf{Y}}, \mathbf{y}) = \frac{\text{MSE}(\hat{\mathbf{Y}}, \mathbf{y})}{\bar{y}_{\text{abs}}} \quad (3.7)$$

Standardized mean absolute error (sMAE) The sMAE is the MAE computed on the standardized time series. Like the nMAE, the sMAE allows to compare between time series with different characteristics. Given the mean $\bar{y}$ and standard deviation σ of the true time series $\mathbf{y}$, the sMAE is defined as follows:

$$\text{sMAE}(\hat{\mathbf{Y}}, \mathbf{y}) = \frac{1}{n \cdot H} \sum_{t=t_1}^{t_n} \sum_{i=1}^H \left| \frac{\mathbf{y}[t+i] - \bar{y}}{\sigma} - \frac{\hat{\mathbf{y}}_t[t+i] - \bar{y}}{\sigma} \right| \quad (3.8)$$

Since the $-\bar{y}$ in the two parts of the subtraction cancel out, and the σ can be pulled out of the sum because it is always positive, the formula simplifies to the MAE divided by the time series' standard deviation:

$$\text{sMAE}(\hat{\mathbf{Y}}, \mathbf{y}) = \frac{\text{MAE}(\hat{\mathbf{Y}}, \mathbf{y})}{\sigma} \quad (3.9)$$

Standardized mean squared error (sMSE) Analog to the sMAE, the sMSE is the MSE computed on the standardized time series, with $\bar{y}$ and σ being the time series' mean and standard deviation:

$$\text{sMSE}(\hat{\mathbf{Y}}, \mathbf{y}) = \frac{1}{n \cdot H} \sum_{t=t_1}^{t_n} \sum_{i=1}^H \left(\frac{\mathbf{y}[t+i] - \bar{y}}{\sigma} - \frac{\hat{\mathbf{y}}_t[t+i] - \bar{y}}{\sigma} \right)^2 \quad (3.10)$$

Mean absolute percentage error (MAPE) The MAPE represents the average relative deviation of the forecast from the ground truth. It can be interpreted as a percentage of the ground truth values. However, it is undefined if one ground truth value equals zero, and ground truth values close to zero lead to large MAPE values:

$$\text{MAPE}(\hat{\mathbf{Y}}, \mathbf{y}) = \frac{1}{n \cdot H} \sum_{t=t_1}^{t_n} \sum_{i=1}^H \left| \frac{\mathbf{y}[t+i] - \hat{\mathbf{y}}_t[t+i]}{\mathbf{y}[t+i]} \right| \quad (3.11)$$

Coefficient of determination (R^2) The R^2 is defined via the squared residuals and the squared deviations from the time series' mean $\bar{y}$. A perfect forecast has an R^2 of one, while a forecast that always predicts the mean has an R^2 of zero. Negative values of R^2 are also possible and indicate that the forecast is worse than the mean prediction:

$$R^2(\hat{\mathbf{Y}}, \mathbf{y}) = 1 - \frac{\sum_{t=t_1}^{t_n} \sum_{i=1}^H (\mathbf{y}[t+i] - \hat{\mathbf{y}}_t[t+i])^2}{\sum_{t=t_1}^{t_n} \sum_{i=1}^H (\mathbf{y}[t+i] - \bar{y})^2} \quad (3.12)$$

3.1.3 Probabilistic forecasts

Probabilistic forecasts do not only provide an expected target value, but represent a probability distribution over possible outcomes, expressed as distribution parameters or quantiles. Probabilistic forecasts offer an estimate of the prediction's uncertainty and are thereby more informative than deterministic forecasts [91]. Such probabilistic forecasts can be created by conformal prediction [10], forecast ensembles [275], quantile regression [150], or generative models [212]. While these approaches are promising, this dissertation focuses on deterministic forecasts for two reasons:

1. Many down-stream applications incorporate an optimization based on deterministic forecasts. This includes the peak shaving application in Chapter 12.
2. Most publications on electrical load forecasting use deterministic models, but a systematic benchmarking of these models is currently lacking.

A benchmarking of deterministic forecasting models is the basis of future extensions to probabilistic forecasting. In addition, future work can evaluate the value of Transformer-based probabilistic forecasts in down-stream applications such as battery scheduling [13] and grid state estimation [94].

3.2 Transformers

This section presents the original Transformer [271] architecture, its adaption as a time-series forecasting model, and different time-series Transformer architectures.

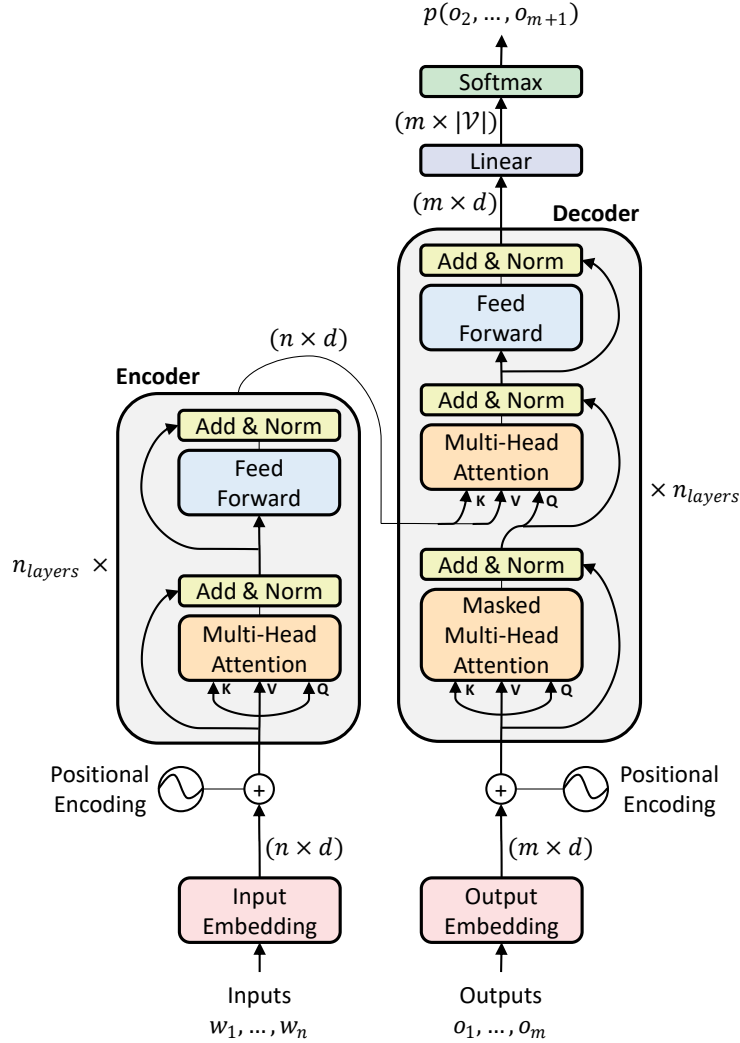


Figure 3.1: Architecture of a Transformer text translation model with input tokens $w_1, \dots, w_n$ and output tokens $o_1, \dots, o_m$, n_{layers} layers and hidden dimension d . The attention blocks receive keys K , values V and queries Q . Figure adapted from [271].

3.2.1 Transformer architecture

An overview of the Transformer architecture is given in Figure 3.1. The figure shows an encoder-decoder Transformer for neural machine translation. The left part is the encoder and the right part is the decoder.

The input to the encoder consists of n tokens $w_1, \dots, w_n$ in the input language. Each token represents a sub-word; that is, either a word or a part of a (longer) word. For each sub-word from a pre-defined vocabulary, an embedding vector of dimension d is maintained, where d is a hyperparameter. These token embeddings are randomly initialized and updated during the training of the model. The input tokens are embedded to n vectors of size d . A positional encoding is added to the vectors to enrich them with information about the order of the tokens in the input sentence.

Transformer encoder

The encoder consists of n_{layers} encoder layers. Each layer gets n vectors of size d as input and produces an output of the same size $n \times d$. Encoder layers use multi-head attention. The attention mechanism computes a weighted average of its input vectors. It is defined for a query vector $\mathbf{q} \in \mathbb{R}^d$, a set of key vectors $\mathbf{k}_1$ to $\mathbf{k}_n$ with each $\mathbf{k}_i \in \mathbb{R}^d$, and value vectors $\mathbf{v}_1$ to $\mathbf{v}_n$ with each $\mathbf{v}_i \in \mathbb{R}^d$, where $\mathbf{v}_i$ is associated with $\mathbf{k}_i$. An attention score a_i is computed for each key vector via the dot-product of the query vector with the key vector:

$$a_i = \frac{\mathbf{q}^T \mathbf{k}_i}{\sqrt{d}} \quad (3.13)$$

These scores are then softmaxed so that they become attention weights which sum up to 1:

$$\alpha_i = \frac{e^{a_i}}{\sum_{j=1}^n e^{a_j}} \quad (3.14)$$

where $e \approx 2.71828$ is the Euler number. The output of the attention mechanism is the weighted average of the value vectors:

$$\text{attention}(\mathbf{q}, \{\mathbf{k}_i | 1 \leq i \leq n\}, \{\mathbf{v}_i | 1 \leq i \leq n\}) = \sum_{i=1}^n \alpha_i \cdot \mathbf{v}_i \quad (3.15)$$

Note that the attention mechanism is applied for each of the n query vectors, and the formula can be written more compactly (and computed more efficiently by matrix operations) when the query vectors, key vectors and value vectors are combined to three matrices $\mathbf{Q} \in \mathbb{R}^{n \times d}$, $\mathbf{K} \in \mathbb{R}^{n \times d}$, $\mathbf{V} \in \mathbb{R}^{n \times d}$:

$$\text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} \right) \mathbf{V} \quad (3.16)$$

Note that since $\mathbf{Q}$ has dimensions $n \times d$ and $\mathbf{K}^T$ has dimensions $d \times n$, the result of $\mathbf{Q}\mathbf{K}^T$ has dimensions $n \times n$. The softmax operation is applied per row, so that the values of each row sum up to one. The division by a scalar and the softmax operation do not change the dimensions. The resulting $n \times n$ matrix after the softmax operation is called the attention map. Multiplying the attention map of dimensions

$n \times n$ with $\mathbf{Q}$ of dimensions $n \times d$ results in a $n \times d$ matrix. Therefore, the output of the attention mechanism has dimensions $n \times d$.

Multi-head attention applies n_{heads} attention mechanisms (called "attention heads") in parallel. Each attention head receives a part of each of the n vectors. More precisely, the i^{th} head receives the entries $i \cdot \frac{d}{n_{\text{heads}}}$ to $(i + 1) \cdot \frac{d}{n_{\text{heads}}} - 1$ of each vector, with $0 \leq i \leq n_{\text{heads}} - 1$. These n_{heads} smaller vectors of dimension $\frac{d}{n_{\text{heads}}}$ are transformed via three linear transformations into query vectors, key vectors and value vectors before the i^{th} attention head is applied to them. Each attention head returns a matrix of dimensions $n \times \frac{d}{n_{\text{heads}}}$, which are then concatenated to the result of the multi-head attention mechanism with dimensions $n \times d$. A visualization of multi-head attention can be found in [5].

The multi-head attention is followed by an "add and norm" operation, which consists of a residual connection and layer normalization [21]. Then, a feedforward network follows with two fully connected layers, and another "add and norm" operation. The output of the last encoder layer can be used as a context-enriched token embedding.

Transformer decoder

The decoder receives the tokens in the target language as input and predicts the next token in the target language based on the input text in the input language and the previous tokens in the target language. It gets m sub-words for the target words as input, which are embedded to a $m \times d$ matrix via learned token embeddings. A positional encoding is added to the embeddings to encode their sequential order.

The decoder also consists of n_{layers} layers. There are two major differences to the encoder layers. First, the self-attention in the decoder is masked. This means that future key vectors get an attention weight of 0, so that the model cannot use future token embeddings when predicting the next token during training (which would not work during inference where the next token is unknown and is precisely what the model has to predict). Second, there is not only a self-attention mechanism, where queries, keys and values come from the output embedding, but a second attention mechanism called cross-attention, where the queries come from the output embedding and the keys and values come from the encoder output. With the cross-attention mechanism, the decoder receives information from the encoder about the input text.

Finally, the output of the decoder consists of m vectors of dimension d , which are transformed into probabilities for the next word via a linear layer with $|\mathcal{V}|$ units where $\mathcal{V}$ is the vocabulary (i.e., the set of all sub-words in the target language) and a softmax operation.

3.2.2 Adaption of the Transformer as a forecasting model

Instead of translating from a sequence of sub-words in one language to a sequence of sub-words in another language, the Transformer can be used to "translate" from a sequence of past observations to a sequence of expected future observations. These expected future observations form the forecast. The adapted architecture is shown in Figure 3.2, with changes to the original architecture highlighted in red. The architecture builds on the original Transformer architecture [271] with the input and output layers adapted for time series [294].

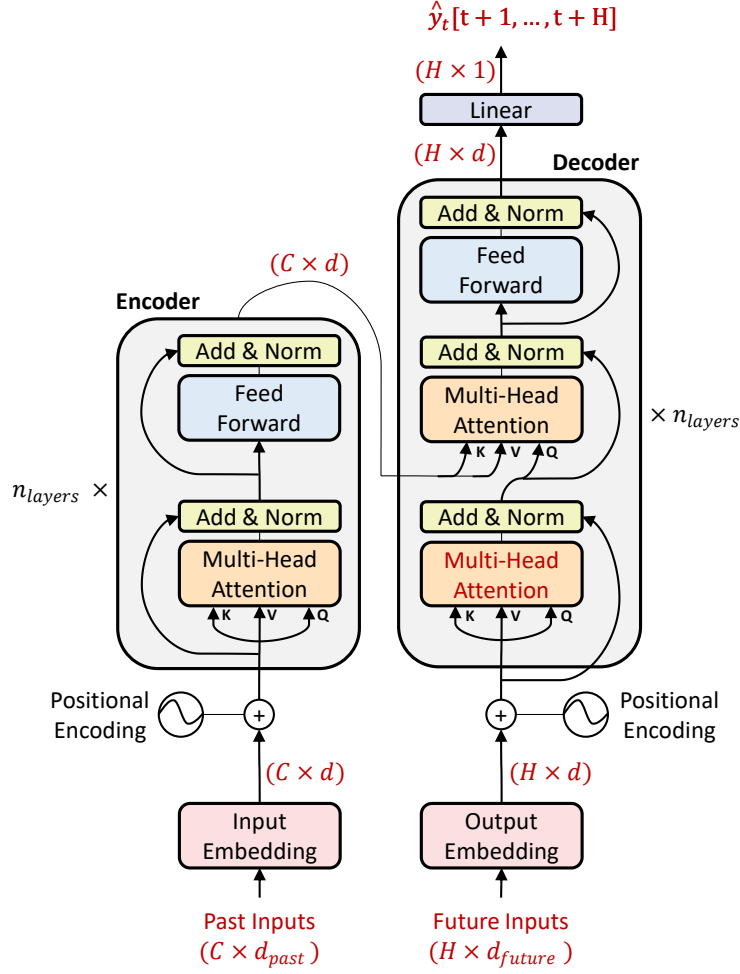


Figure 3.2: Architecture of the Transformer forecasting model with context length C , forecast horizon H , past input dimension d_{past} , future input dimension d_{future} and hidden dimension d . Changes to the original Transformer architecture from [271] are highlighted in red.

At a time step t , the input to the encoder consists of C vectors representing the C last known time steps (remember that C is the context length). That is, we have C input vectors of dimension d_{past} representing the last C time steps. These vectors contain information about the time steps, such as the observation y_i and potentially exogenous feature values. Usually, the input size d_{past} does not equal the model's hidden dimension d , so the input vectors are embedded by a linear layer with d units before passing them to the encoder.

The decoder receives H vectors representing the H future time steps (remember that H is the forecast horizon). These vectors have dimension d_{future} and contain information about the exogenous features that are known in the future. Unlike in the text translation Transformer, the cross-attention is not masked, as the future vectors only contain information that is already known at time step t , so using information of future time steps during inference is possible.

Finally, the H decoder output vectors of size d are transformed by a linear layer with a single unit into H scalar values that form the forecast $\hat{y}_t$ for the next H time steps.

3.2.3 Time-series Transformer architectures

Multiple publications propose Transformer architectures for time-series forecasting [282], including Transformer-based TSFMs [167]. The proposed architectural modifications can be classified into input representations, hybrid model combinations, partial Transformers, and variants of attention mechanisms.

Input representations The time-augmented Transformer [297] first builds two separate embeddings for the past load and the time covariates, and then adds them into a single embedding. PatchTST [199] combines consecutive time steps into patches, thereby reducing the effective sequence length. iTransformer [172] and TimeXer [277] build embeddings of entire covariate time series and use attention to fuse them with past observations. Chronos [12] represents observations as tokens, by binning observed values and one-hot-encoding the binned values. However, the successor model Chronos-2 [11] does not continue the tokenization approach, and instead mixes patching and covariate attention by creating patches for past observations and covariates, and applying attention in the time dimension and the feature dimension in an alternating fashion.

Hybrid models Multiple works combine Transformer models with other neural network layers. TFT [168] uses special feature selection networks, which generate interpretable feature weights, together with LSTM layers and a single Transformer encoder layer. Informer [300] uses a technique called "self-attention distilling" to reduce the sequence length after each encoder layer, which is based on max pooling that is typical for CNNs.

Partial Transformers Multiple works deviate from the encoder-decoder architecture introduced by Vaswani et al. [271]. TFT [168] uses an encoder-decoder LSTM network followed by a single Transformer encoder layer. Similarly, LogSparse Transformer [165] and Chronos-2 [11] are encoder-only Transformers. The encoder output can either be combined into a flat vector, or the last output vectors (which correspond to the future time steps) can be used in the final prediction layer to create the forecast [11].

Attention mechanisms Multiple modifications to the attention mechanism are proposed to reduce the time and space complexity or to enhance the model accuracy. LogSparse Transformer [165] introduces sparse attention patterns based on logarithmically increasing distance and calculates keys and queries with convolutional layers instead of dense layers to better capture local patterns. Informer [300] proposes ProbSparse attention, where attention is only performed between selected pairs of keys and queries, thereby reducing the complexity of the attention mechanism. FEDformer [301] performs a frequency decomposition of the input to each Transformer layer and applies attention on a subset of the frequency components. Autoformer [292] integrates trend-seasonality decomposition into the attention mechanism and selects pairs of keys and queries based on autocorrelation.

Chapter 7 proposes a flexible Transformer architecture integrating a variety of architectural modifications, and benchmarks it against the standard Transformer and established load forecasting methods.

3.3 Shapley Additive Explanations (SHAP)¹

SHAP [176] is a popular XAI framework based on Shapley values [248] from game theory. This section introduces Shapley values and how they are used by SHAP to explain machine learning models.

3.3.1 Shapley values

Shapley values are used in game theory to compute the contribution of a player to the outcome of a cooperative game [248]. The player's contribution is defined as the change in the game outcome when the player enters the game, averaged across all possible permutations of players entering the game. The Shapley value definition is the unique solution for the computation of players' contributions that fulfill the efficiency axiom (the sum of the contributions adds up to the outcome), the symmetry axiom (two identical players receive equal contributions), the dummy axiom (a player that does not contribute to any coalition gets a contribution of zero) and the additivity axiom (the contribution to the sum of two games equals the sum of the contributions of the two games) [176, 188].

3.3.2 Explaining machine learning models with SHAP

Shapley values are used in machine learning to compute the contribution of features to the prediction of a model [176, 188]. In this setting, the features are the players and the model prediction is the game's outcome. The exact Shapley values usually cannot be computed for computational reasons and because the full feature attributions are unknown, so it is not possible to integrate over them. Therefore, the Shapley values are estimated in practice and are then called SHAP values, in order to distinguish them from the exact Shapley values. The naive algorithm to compute SHAP values works as follows: For a set of features $\{x_1, \dots, x_n\}$, consider all permutations p_j , $1 \leq j \leq n!$. For a permutation p_j , go through all features in the order of the permutation. For a feature x_i , let S be the set of features that precede x_i in p_j . Then, the marginal contribution ϕ_i of x_i is defined as the change in model prediction when x_i is added to S :

$$\phi(x_i, p_j) = f(S \cup \{x_i\}) - f(S) \quad (3.17)$$

where $f(X)$ is the model prediction using a set of features X . The SHAP value for x_i is defined as the average marginal contribution of x_i in all permutations. It can be computed more efficiently by iterating over all 2^n feature subsets instead of all $n!$ feature permutations:

$$\text{SHAP}(x_i) = \sum_{S \subseteq V \setminus \{x_i\}} \frac{(n-1-|S|)! \cdot |S|!}{n!} \cdot (f(S \cup \{x_i\}) - f(S)) \quad (3.18)$$

The first part of the formula is a weight, defined as the fraction of permutations in which x_i gets added to S , and the second part is the marginal contribution of x_i , defined as the difference of the model prediction

¹ Section 3.3 is a modified excerpt from Hertel et al. [111].

with and without x_i . Note that to compute all feature contributions, it is necessary to compute the model predictions for all 2^n feature subsets $S \subseteq V$ (also called coalitions) [176, 188].

Usually, a machine learning model trained on n features requires all n features to be available at inference time and therefore does not support predictions based on feature subsets. A common approach for obtaining predictions based on a subset of features is to marginalize over the absent features using Monte Carlo integration. This can be achieved by sampling their values k times from a background dataset (e.g., the training data), evaluating the model for each resulting feature configuration, and averaging the corresponding predictions. Consequently, the model has to be run k times for each of the feature subsets. This is computationally expensive even for small feature sets. An efficient way to compute forecasts based on feature subsets without the need to sample from background data is introduced in Chapter 9.

3.3.3 PermutationSHAP

PermutationSHAP is a more efficient way of estimating SHAP values by calculating marginal contributions only for a subset of the coalitions [188]. It samples K random feature permutations. For each permutation p_j , it starts with the empty feature set and iteratively adds features in the order of the permutation. The marginal contribution of each feature is computed as the change in model prediction when adding x_i to S , i.e., $f(S \cup \{x_i\}) - f(S)$ as above, where S is the set of features preceding x_i in p_j . As in the naive algorithm, predictions based on feature subsets are estimated by a Monte Carlo integration over the absent features (by sampling their values from background data). Each permutation is used twice, where features are added once in the forward and once in the backward direction. The SHAP value estimate is then computed as the average marginal contribution of a feature over all sampled permutations.

PermutationSHAP is commonly used in practice, but it has two drawbacks:

1. It is costly to compute for large feature sets. Considering a forecasting problem with 168×7 past features and 168×6 future features (2 184 features in total), ten feature permutations (each run forward and backward), and 100 samples to estimate distributions of absent features, this makes $2\,184 \times 10 \times 2 \times 100 = 4\,368\,000$ model calls that are needed to compute a single local explanation of a forecast.
2. When feature dependencies are not considered during Monte Carlo sampling, unrealistic samples are created, which are out of distribution of the training data and can potentially lead to arbitrary model behavior. For example, when calendar features are sampled independently, it can happen that Monday midnight is followed by Friday noon. Other examples of unrealistic feature combinations are 30 °C air temperature in winter, quick jumps in the electrical load or temperature values, high air temperature or electrical load at night, and many more.

3.3.4 Other explainers

Since most models do not natively support inputs with absent features, several algorithms have been proposed to estimate SHAP values, including three approaches tailored to time-series models [32, 196, 299]:

- KernelSHAP [176] estimates SHAP values by fitting a weighted local linear model to outputs obtained from perturbed samples, where absent features are sampled from background data.
- TreeSHAP [177] is designed for tree-based models and exploits the tree structure to compute SHAP values efficiently.
- DeepSHAP [176] estimates SHAP values for neural networks by applying DeepLIFT [251] contribution rules repeatedly for a set of background samples.
- GradientSHAP is a method for differentiable models and is based on Expected Gradients [77]. It estimates SHAP values by averaging gradients evaluated at random points between the instance and randomly sampled background samples.
- TimeSHAP [32] is designed for time series. It selects relevant time steps and features which are then explained with KernelSHAP, where absent values are replaced with a baseline.
- WindowSHAP [196] and ShapTime [299] are SHAP algorithms for time series that estimate SHAP values over temporal windows instead of individual time steps, thereby reducing the number of model evaluations.

Most SHAP algorithms evaluate models on feature subsets by following one of two strategies [47]: (1) sampling absent features from a marginal or conditional distribution, or (2) replacing absent features with a predefined baseline value. Both approaches have limitations. Sampling is computationally intensive and may generate unrealistic inputs that lie outside the data distribution, resulting in off-manifold evaluations [80]. Potential replacement values for time series data are values of zero, the negative observed values or the values in reversed order (as in perturbation methods [239]). However, choosing a baseline value is not obvious and affects the resulting explanations. After standardization of the continuous time-series features, a value of zero represents the time series' mean. When a non-holiday is chosen as the baseline, non-holiday feature values will be attributed SHAP values of zero, as the feature value and the baseline value are the same. For the day-of-week feature, no meaningful sequence of weekdays can be chosen as a baseline. To address these limitations, Chapter 9 proposes a new SHAP algorithm that requires neither sampling nor baseline substitution.

3.3.5 Owen values

Owen values [203] are a variant of SHAP values for games with a coalitional structure. A partition of the features into coalitions $B_1, \dots, B_l$ must be given. The contributions of the coalitions are computed on a first level, and then broken down into contributions of the individual players on a second level. The formula is similar to Equation 3.18 for SHAP values, but now also considers the coalitions:

$$\text{Owen}(x_i) = \sum_{R \subseteq M \setminus \{k\}} \sum_{T \subseteq B_k \setminus \{x_i\}} \frac{1}{|M| \cdot |B_k|} \frac{1}{\binom{|M|-1}{|R|}} \frac{1}{\binom{|B_k|-1}{|T|}} (f(Q \cup T \cup \{x_i\}) - f(Q \cup T)) \quad (3.19)$$

where $M = \{B_1, \dots, B_l\}$ is the set of coalitions, B_k is the coalition containing x_i , and $Q = \bigcup_{r \in R} B_r$ is the union of features over a subset of coalitions R .

3.3.6 From local to global explanations

An explanation of a single model prediction is called a *local explanation*. In addition to local explanations, explanations of the model behavior over the full data distribution are desirable. Such explanations are called *global explanations*, which summarize many local explanations to give a broader picture.

Feature importance is a common form of global explanation in which a single importance value is assigned to each feature, indicating its overall influence on the model's predictions. To obtain feature importance values from local explanations, the absolute SHAP values of each feature are summed across m samples. The resulting sums are then normalized to obtain the relative feature importance as percentages.

Feature dependence plots provide a more detailed form of global model explanation. A separate dependence plot is generated for each feature, with each plot containing m points corresponding to the m local explanations. The feature values are shown on the x-axis, while the corresponding SHAP values are shown on the y-axis, thereby illustrating how the feature contributes to the model's predictions. The points can additionally be colored according to the value of another feature to visualize feature interactions.

4 Datasets

This chapter describes the datasets used in the dissertation. The datasets were selected to represent a variety of use cases on different grid levels, ranging from highly aggregated TSO data to less aggregated data from individual clients. All datasets are publicly available, allowing for future comparisons to the results of this dissertation. Table 4.1 presents an overview of the datasets and lists relevant statistics, such as the amount of missing data, the mean load per time series, and the daily, weekly and annual autocorrelation averaged across all time series.

Dataset	Time series	Duration	Missing	Weather	Mean load			Autocorrelation		
					min	median	max	1d	1w	1y
Electricity-321	321	1.1.12 – 31.12.14	1.1 %		4.2 kW	148.0 kW	50.1 MW	0.88	0.89	0.77
Electricity-328	328	1.1.11 – 31.12.14	14.0 %		5.5 kW	149.4 kW	50.1 MW	0.89	0.90	0.77
Electricity-287	287	1.1.11 – 31.12.14	14.0 %	✓	5.5 kW	152.6 kW	50.1 MW	0.89	0.89	0.77
Ausgrid solar home	299	1.7.10 – 30.6.13	0.1 %		0.2 kW	0.6 kW	1.5 kW	0.46	0.39	0.25
Ausgrid substations	152	1.1.16 – 31.12.19	0.6 %		0.3 MW	12.3 MW	60.0 MW	0.84	0.79	0.68
TransnetBW	1	1.1.15 – 31.12.25	-	✓	-	6.9 GW	-	0.73	0.89	0.69
FeederBW	200	1.4.23 – 31.3.25	-	✓	-4.7 kW	13.7 kW	41.9 kW	0.78	0.74	0.66
LDCs	3	1.1.15 – 31.12.18	-	✓	148.4 MW	238.2 MW	1061.2 MW	0.88	0.67	0.63
Energy community	1	3.1.19 – 31.12.20	-	✓	-	18.5 kW	-	0.93	0.86	0.82
Synthetic	120,000	336 hours	-	✓	-0.63	0.00	0.66	0.86	0.92	-

Table 4.1: Statistics of the datasets used in the dissertation. The mean load is given for the time series with the lowest load (min), median load (median) and highest load (max). The autocorrelation is given for lags of one day (1d), one week (1w) and one year (1y). The synthetic dataset has no unit.

4.1 Electricity

The Electricity dataset is derived from the UCI Electricity Load Diagrams repository¹. This dataset was first introduced by Rodrigues and Trindade [229] and consists of 370 electricity consumption time series collected between 2011 and 2014 from clients of an energy trading company in Portugal. The clients represent a diverse set of consumption profiles, including factories, offices, supermarkets, retailers, hotels, public buildings and other facilities, as well as a few medium-to-low-voltage substations and cogeneration plants [229]. Some clients entered the dataset after 2011; for these cases, electricity consumption prior to their creation is recorded as zero. Additionally, the dataset reflects daylight saving time adjustments: on the March clock-shift day, which contains only 23 hours, consumption values between 1:00 a.m. and

¹ <https://archive.ics.uci.edu/dataset/321/electricityloadaddiagrams20112014>, accessed 2026-01-23.

2:00 a.m. are zero for all series, while on the October clock-shift day, which has 25 hours, the values between 1:00 a.m. and 2:00 a.m. aggregate the consumption of two hours. We interpolate the missing values in March and divide the aggregated values in October by two.

The Electricity dataset has become a popular benchmark in the time-series forecasting literature, particularly for evaluating deep learning models. It has been used in previous works on Transformer architectures, including the LogSparse Transformer [165] and TFT [168]. In addition, the dataset is included in several benchmarking suites, such as GIFT-Eval [4] and BuildingsBench [74].

Despite its widespread use, our detailed inspection of the Electricity dataset reveals several issues with some of the contained time series. First, multiple time series contain missing values; an example is illustrated in Figure 4.1a. Second, we observe artifacts in the form of consecutive days or weeks with exactly identical consumption values, likely resulting from copy-paste imputation of missing data [281]. Such artifacts can make the forecasting task unrealistically easy, as even a trivial consistency baseline achieves zero error over these periods. Figure 4.1b shows such an artifact, where two weeks are exactly identical. Notably, missing values and imputation artifacts often appear in the last year of the series, which is commonly reserved for validation and testing. Finally, some time series are shorter than one year (see Figure 4.1c), which is problematic for training local forecasting models, as at least one full year of training data is typically required to expose the model to all seasonal effects.

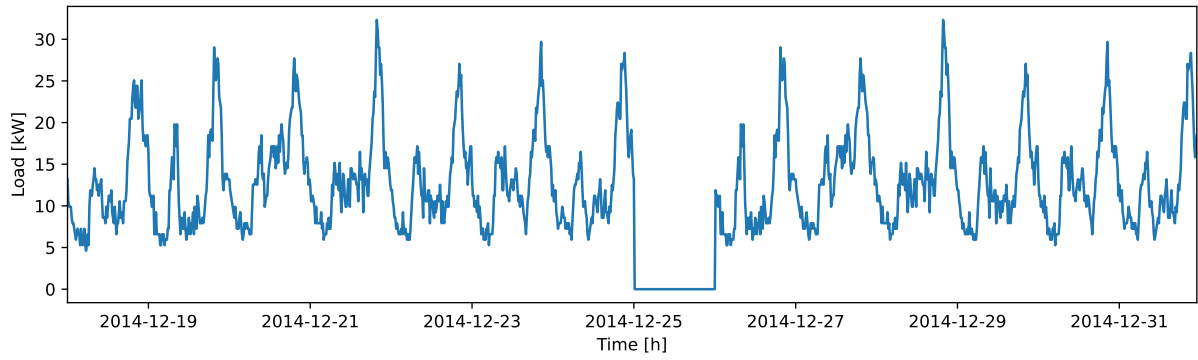
Despite the issues, we use the Electricity dataset because of its popularity in the literature and its significant size. The dataset is used for three different purposes in this thesis, each requiring a different dataset version:

1. For comparisons to related work [199, 292, 296, 300, 301], we use the same variant of the Electricity dataset as these works. As this version contains 321 time series, we denote it as Electricity-321.
2. For more realistic evaluations that do not suffer from imputation artifacts and short time series, we derive our own cleaned version of the Electricity dataset. This dataset contains 328 time series and is therefore denoted as Electricity-328.
3. The third application is benchmarking models with varying context length. A fair comparison requires that the time series have no missing values in the evaluation period. Therefore, we introduce Electricity-287, which contains the time series from Electricity-328 that have no missing values in the last year. As it is common to use weather data as exogenous features for load forecasting [100], Electricity-287 is enriched with weather data from the ERA5 reanalysis model [57].

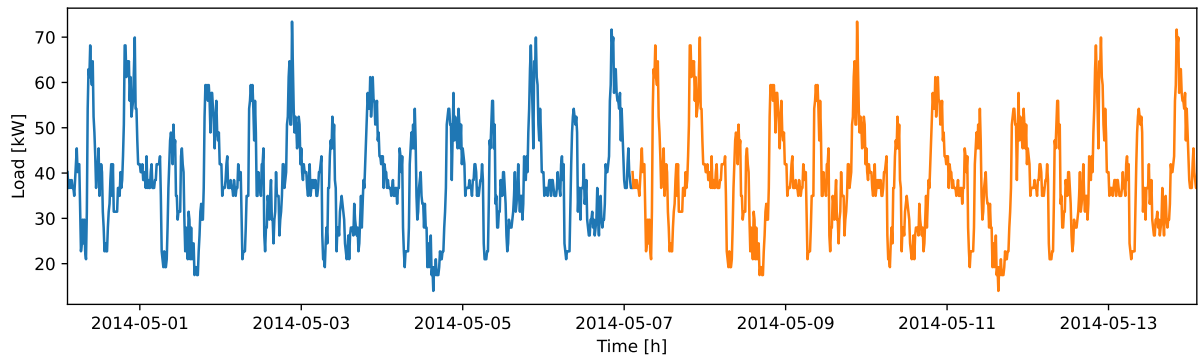
The preprocessing for each version of the datasets is described next.

4.1.1 Electricity-321

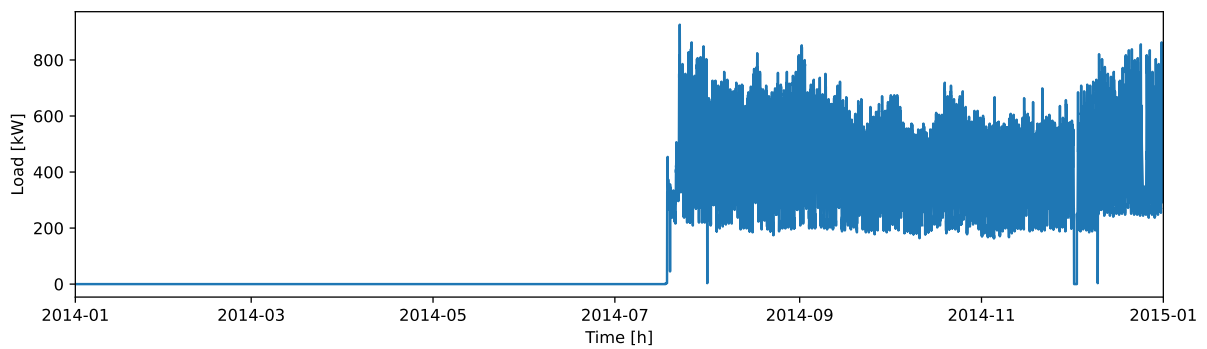
To address some of the shortcomings of the original Electricity dataset, Lai et al. [158] introduced a refined variant comprising 321 time series. In this version, the first year of data is removed, as many clients had not yet started reporting consumption during that period, and all time series containing missing values in the remaining time span are excluded. As a result, the retained series are complete and sufficiently long, which resolves the issues of short time series and missing values identified in the original dataset.



(a) **Missing values:** Cutout of time series MT_023 of the Electricity dataset with missing values on 2014-12-25.



(b) **Imputation artifact:** Cutout of time series MT_009 of the Electricity dataset, showing an imputation artifact of two weeks (blue and orange) with identical load values.



(c) **Short time series:** Cutout of time series MT_178 of the Electricity dataset, which is shorter than one year.

Figure 4.1: Examples of problematic time series in the Electricity dataset.

However, the imputation artifacts discussed above persist in this variant. This revised dataset has since been widely adopted in the literature and is used in several follow-up works [172, 199, 292, 296, 300, 301]. It is also included in the Monash Time Series Forecasting Archive [92].

4.1.2 Electricity-328

Ambrosius [9] introduces a cleaned version of the Electricity dataset, where imputation artifacts and time series with insufficient data are removed. To eliminate copy-paste imputation artifacts, identical consecutive weeks or days are recognized and only the first occurrence is kept, replacing all subsequent repetitions with not-a-number (NaN) values. In addition, for constant segments lasting longer than one hour, only the first value is kept while the other values are set to zero. To remove time series with insufficient data, time series are removed when they have less than two years of data, or when at least 10 % of their two-week windows contain NaN values. After these filtering steps, all zeros are linearly interpolated, and the original 15-minute measurements are aggregated to hourly values. After removing time series with insufficient data, the final dataset contains 328 preprocessed time series, each with at least two years of data at hourly resolution. The resulting series may still contain NaN values, which are handled during model training and evaluation by skipping examples that include NaN values in the input or the target. The dataset is available on GitHub².

4.1.3 Electricity-287

Finally, we further reduce the dataset to ensure a fair comparison across models with different context lengths. While NaN values in the validation and test period are unproblematic when all models share the same fixed context length because the same forecast windows are skipped for all models, this is no longer the case when context length itself is treated as a model hyperparameter. In such settings, the presence of missing values in the input can cause different numbers of target windows to be excluded depending on the context length. For example, if the observation at 1 January 2014, 00:00 is missing, a model with a one-week context skips all forecasts up to 8 January 2014, 00:00, whereas a model with a two-week context must skip additional windows, as the missing value remains within the input until 15 January 2014, 00:00. To avoid this source of inconsistency, we remove all time series that contain missing values in 2014, which is used for validation and testing. This final filtering step yields a dataset comprising 287 time series.

As weather data is commonly used in load forecasting models [100], we enrich the Electricity-287 dataset with weather data. For this purpose, hourly values of air temperature, global horizontal irradiance, precipitation and wind speed from the ERA5 reanalysis model [107] for Portugal are added to the dataset. Note that using weather measurements or reanalysis data for model comparisons on historical data is a common approach when historical weather forecasts are not available [100].

The dataset is available on GitHub³.

² <https://github.com/hertelm/time-series-clustering>

³ <https://github.com/hertelm/load-forecasting-benchmark>

4.2 Ausgrid Solar Home

The *Ausgrid solar home* dataset⁴ is used as a second dataset with load time series from individual clients. It contains solar generation and electrical load data from 300 clients of an Australian energy supplier. We use 299 of the clients because one client had missing data. The clients are private houses with rooftop solar systems. The time series range from July 2010 to June 2013. We only use the electrical load data, transformed into hourly resolution.

4.3 Ausgrid Substations

Many load forecasting applications deal with the electricity grid rather than individual consumers. Therefore, we use the *Ausgrid substation dataset*⁵ to represent substations. This dataset contains load measurements of 176 high-to-medium-voltage substations in Australia. We use the last four years of available data, ranging from 2016 to 2019, so that the time series have the same length as in the Electricity dataset, and apply the same preprocessing as for Electricity-328. After removing time series with insufficient data, the final dataset contains 152 preprocessed time series in hourly resolution.

4.4 TransnetBW

To represent applications on the TSO level, the aggregated load of the control area of the German TSO TransnetBW is used. The original dataset comes from the Open Power System Data time-series archive [287].⁶ It contains the electrical load in MW for every quarter of an hour from 2015 to 2019. In order to shorten the time series, we transform the data to hourly resolution by averaging every four consecutive values. For an extended version containing more recent data, we downloaded the load for the years 2020–2025 from the ENTSO-E Transparency Platform [75], which shows for example a much stronger influence of behind-the-meter PV than the 2015–2019 data due to the ongoing installation of PV panels.

Figure 4.2 shows the seasonal pattern and the weekly pattern for each year in the TransnetBW dataset. The upper part shows the electrical load curve smoothed with a window size of one week. An annual cycle can be observed, with higher load in winter than in summer. A strong effect of the Christmas period is visible, where the loads are lower due to extended holidays. The lower part shows the weekly load profile averaged for each year. A weekly cycle can be observed, with lower loads on the weekend. The dataset is affected by a concept drift with decreasing loads over the years, and an increasing drop during noon because of an increased behind-the-meter PV capacity.

As weather data is commonly for load forecasting [100], we enrich the TransnetBW dataset with weather data from the Baden-Württemberg region from the ERA5 reanalysis model [57].

⁴ <https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Solar-home-electricity-data>, downloaded 2023-06-15.

⁵ <https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Distribution-zone-substation-data>, downloaded 2024-03-12.

⁶ https://data.open-power-system-data.org/time_series/, accessed 2026-02-22.

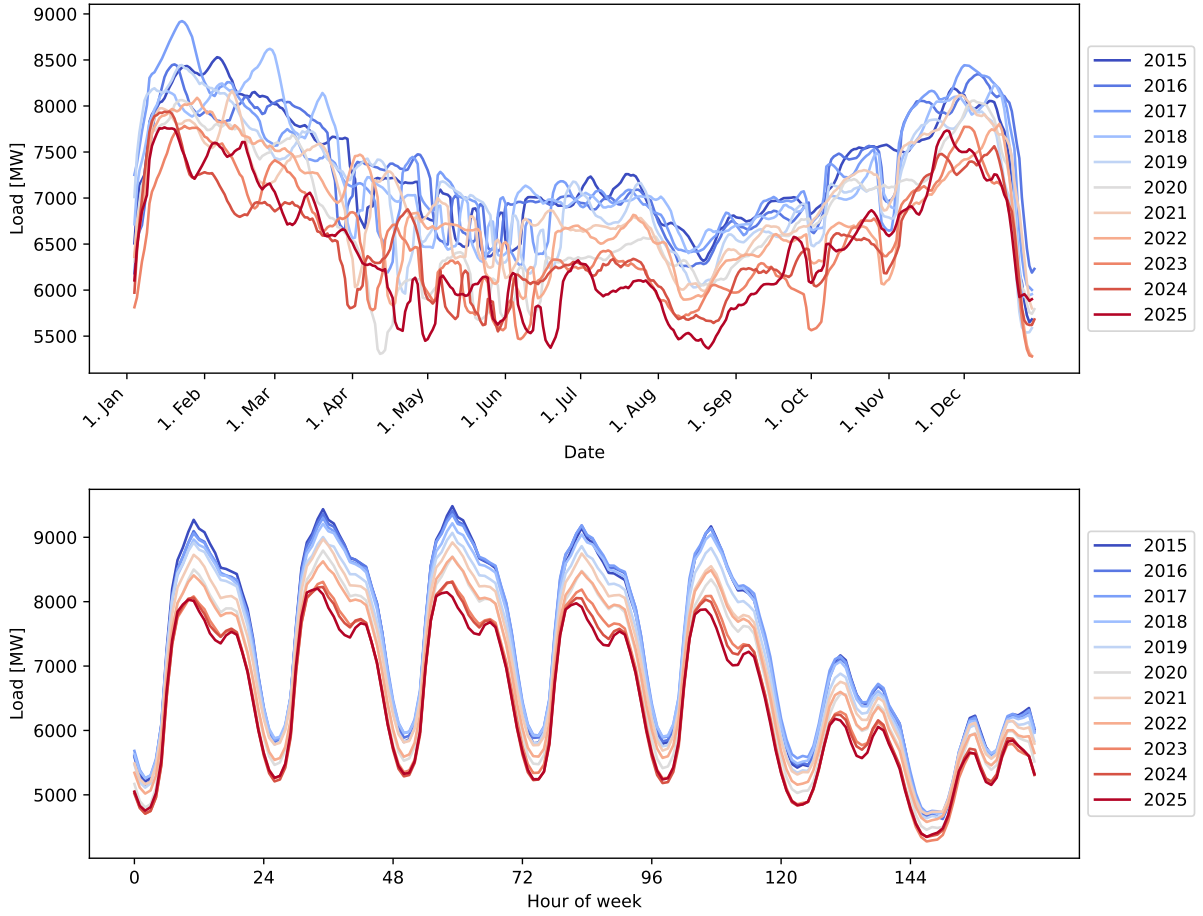


Figure 4.2: Seasonal pattern (top) and weekly pattern (bottom) for each year in the TransnetBW dataset.

4.5 FeederBW

The FeederBW dataset [264] comprises the electrical load from 200 low-voltage feeders in Baden-Württemberg in minutely resolution, together with weather data associated with the post code of each feeder. The length of the time series is two years, ranging from 1 April 2023 to 31 March 2025. We transform the load data into hourly resolution by averaging the minutely recorded values. We use irradiance, air temperature, precipitation, and a binary holiday feature as covariates.

4.6 Local Distribution Companies

Chapter 11 evaluates the Transformer on the data from the final of the BigDEAL challenge [252]. The dataset used in the final of the BigDEAL challenge comprises the hourly load of three local distribution companies (LDCs), together with temperature forecasts for six nearby weather stations for the years 2015–2018. It is further described in Chapter 11.

4.7 Energy Community

Chapter 12 evaluates the benefit of Transformer-based forecasts in a battery scheduling application for an energy community with 38 households. The data used in this study is originally from [240]. Semmelmann et al. [244] enriched the dataset with additional features. The additional features are the binary-encoded type of day (weekend, weekday, holiday), cyclically encoded hour of day and month, the ambient temperature as well as rolling averages of the ambient temperature with windows of 24 and 48 hours, the average load at the same hour of day over the last week, wind speed and humidity. The data from the beginning of 2019 until the end of 2020 is used for the experiments.

4.8 Synthetic Dataset⁷

As the presented datasets have no ground truth for the evaluation of XAI methods, we introduce a new synthetic dataset that can be used to verify SHAP-based explainability methods. As the dataset is synthetic, it allows to generate an arbitrary number of samples, which can be used to train accurate prediction models. Since we have access to the data generation process, we can use the SHAP permutation explainer on the generation function to calculate ground truth SHAP values.

We publish the synthetic dataset with the ground truth explanations, so that it can be used in the future to validate other explanation methods⁸.

The generation of the dataset and the ground truth SHAP values is described next.

4.8.1 Dataset generation

The dataset consists of 120,000 examples. Each example consists of two weeks of hourly values from the target variable and multiple covariates. The first week of the target variable is used as input to the model, and the model has to predict the second week of the target variable. We call the target variable "load", as in our real dataset. For the covariates, both weeks are used as input to the model. That is, we assume a perfect forecast of the covariates for the next week.

The generation of an example follows a multi-step procedure:

- A month $\in [1, 12]$, start weekday $\in [0, 6]$ and start hour $\in [0, 23]$ are sampled with uniform probability.
- A base load is sampled uniformly in the range $(-0.5, 0.5)$.
- The month has an additive effect on the base load. A value of $0.1 \cdot \sin(\text{month} \cdot 2\pi/12)$ gets added to the base load.

⁷ Section 4.8 is a modified excerpt from Hertel et al. [111].

⁸ The synthetic dataset with ground-truth explanations is available on Zenodo [108].

- A daily load curve of 24 values is generated as $\text{factor} \cdot (\sin(h - 0.5 \cdot \pi) + \alpha \cdot \sin(h \cdot 2\pi/24) + \beta \cdot \cos(h \cdot 2\pi/24))$, with $\text{factor} \in (0.5, 1)$, $\alpha, \beta \in (-0.5, 0.5)$ uniformly sampled.
- A uniformly sampled pattern of 24 values with mean 0 and standard deviation 0.1 is added to the daily load. We call the result the `day_pattern` and use it as the load curve for workdays.
- The Saturday pattern deviates from the workday pattern by a multiplicative factor and an additive deviation. It is generated as $s_1 \cdot \text{day_pattern} + \text{normal}(0, 0.1)$, with $s_1 \in (0.5, 0.9)$ uniformly sampled, and $\text{normal}(0, 0.1)$ generating 24 values from a normal distribution with mean 0 and standard deviation 0.1.
- The Sunday pattern is $s_2 \cdot \text{day_pattern} + \text{normal}(0, 0.1)$, with $s_2 \in (0.2, s_1)$.
- From the workday, Saturday and Sunday patterns, the 336 hourly values are produced by repeating the workday pattern, Saturday pattern and Sunday pattern, beginning from the start day of week and start hour of day. Each day has a 10% chance of being a holiday. On holidays, the Sunday pattern is always used, independent of the weekday.
- A temperature curve is generated by a random walk starting from a uniformly sampled value in $(-0.5, 0.5)$ and each step sampled from a normal distribution with mean 0 and standard deviation 0.02. The temperature has a multiplicative effect on the load. That is, the load is rescaled by the temperature as $\text{load} \cdot (0.5 + 0.5 \cdot \text{temperature})$.
- Finally, random noise is added to the load time series, sampled from a normal distribution with mean 0 and standard deviation 0.05.

Examples from the synthetic dataset are shown in Figure 4.3.

4.8.2 Ground-truth SHAP values

We calculate ground truth SHAP values for the validation of XAI methods. Because we have access to the data generation procedure, we can create explanations of the true data dependencies using SHAP. As SHAP is model agnostic, it can be used to explain an arbitrary function $f(x)$. Usually, in the context of XAI, f is a machine learning model, but we set f to the data generation process in order to calculate ground-truth explanations. The inputs x to the data generation process consist of the past load, hour of day, day of week, month, holiday, multiplier and two noise features, each represented as a time series of 168 past and – for all inputs except for the load – 168 future values, and the output is the target load of the next week.

We compute ground-truth explanations for the examples in the test set of the synthetic dataset. To do so, we use PermutationSHAP on the data generation process. For a given test example and a permutation, PermutationSHAP computes marginal contributions of the inputs based on subsets of the inputs. The inputs that are absent, i.e. not contained in the subset, are resampled 1000 times following the sampling process described above, and an alternative target load curve is generated using the data generation process and the resampled inputs. These 1000 alternative targets are then averaged, and the averaged values are used as the expected target load curve given the subset of the inputs. From the target load curves generated

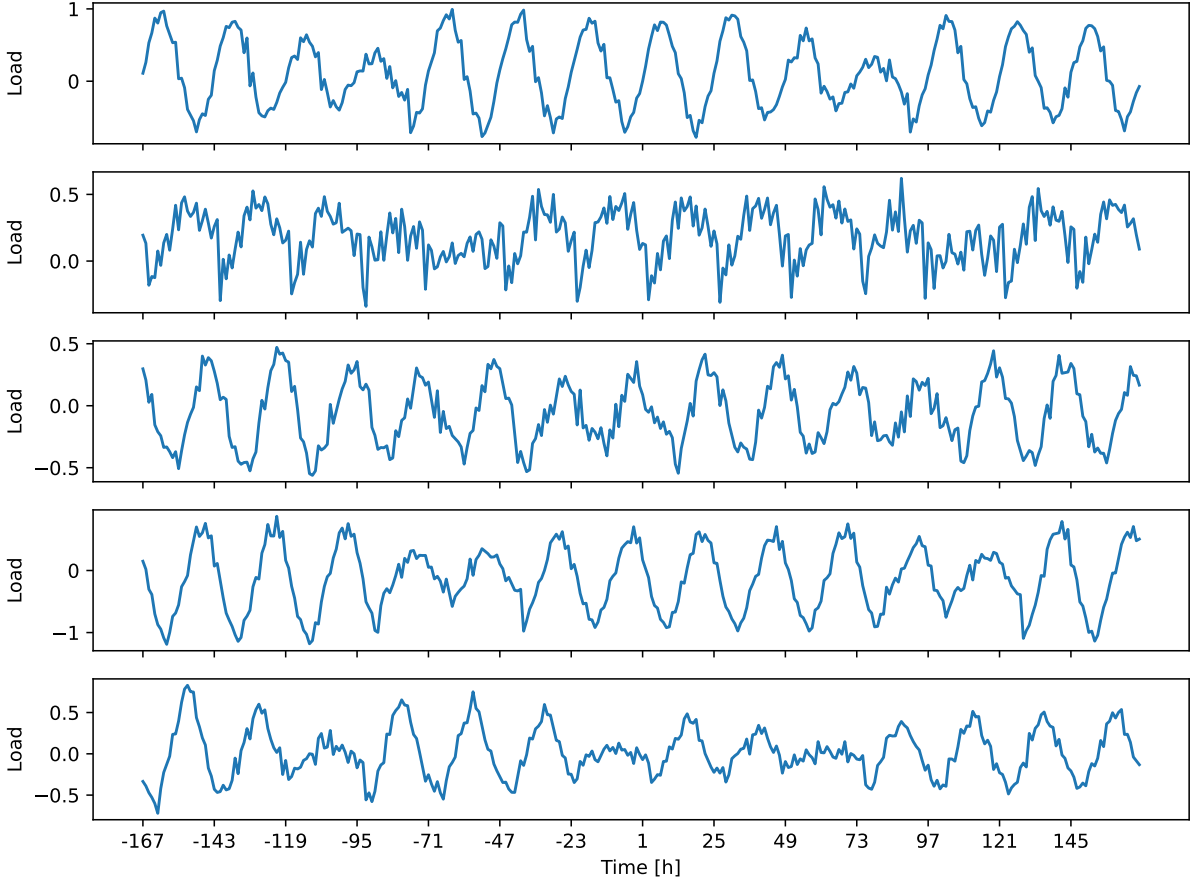


Figure 4.3: Examples from the synthetic dataset.

with different input subsets, marginal contributions and SHAP values are computed with the procedure described in Section 3.3.

While sampling alternative inputs, we make sure that no unrealistic combinations of inputs are generated. In particular, when an input x_1 depends on an input x_2 that is contained in the active subset, we do not resample x_1 . The following dependencies are respected:

- If the day of the week is in the active subset, the hour of day is not resampled, as it can be inferred from the day beginnings and endings.
- For the same reason, the hour of day is not resampled if the holiday feature is in the active subset and there is at least one holiday.
- If the load of the past week is in the active subset, all calendric information as well as the multiplier of the last week is not resampled (because they affect the load), as well as the load patterns for the workday, Saturday and Sunday (as changing any of these would affect the past load).

Note that the ground truth SHAP values are meant to be true to the data [48] under the assumptions stated above. However, a machine learning model could learn different patterns, so that explanations being true to the model instead of true to the data would deviate from our ground truth.

Part II

Training Strategies for Time-Series Transformers

5 Enhancing Transformer Models by a Global Training Strategy

Previous publication

Sections 5.1–5.4 are modified excerpts from:

[110] M. Hertel, M. Beichter, B. Heidrich, O. Neumann, B. Schäfer, R. Mikut, and V. Hagemeyer. "Transformer training strategies for forecasting multiple load time series", *Energy Informatics* 6.1 (2023), DOI: 10.1186/s42162-023-00278-z.^a

^a Licensed under a Creative Commons Attribution 4.0 International License, <https://creativecommons.org/licenses/by/4.0/>.

With the rollout of smart meters and substation measuring devices, fine-grained electrical load data becomes available for an increasing number of measuring points. When time series are recorded at N locations and a forecast is required for each of them, N target time series $\mathbf{y}_1$ to $\mathbf{y}_N$ exist. At a time t , N forecasts $\hat{\mathbf{y}}_{t,i}[t+1, \dots, t+H]$, $1 \leq i \leq N$ are created.

Part II compares training strategies for the Transformer load forecasting model in such scenarios with multiple load time series. Thereby, this part addresses RQ1:

Research question (RQ) 1

Does the forecast error of the Transformer improve by using global or cluster-specific training strategies in scenarios with multiple electrical load time series?

In Chapter 5, three training strategies are compared:

1. A *multivariate* model training strategy, where a single model forecasts all time series jointly.
2. A *local* model training strategy, where separate models forecast each time series.
3. A *global* model training strategy, where a single model is used to forecast each time series separately.

These training strategies have different advantages and disadvantages. The multivariate model can potentially learn dependencies between time series, but it represents a complex function with many inputs and outputs and has little training data available, which leads to a risk of overfitting. Many models need to be trained with the local strategy, each only using the data from one time series, but the models can specialize for each time series. The global model has large training data from all time series, but it needs to infer all time series characteristics from the input during inference.

Chapter 6 introduces a fourth training strategy based on time-series clustering and cluster-specific models.

5.1 Methodology

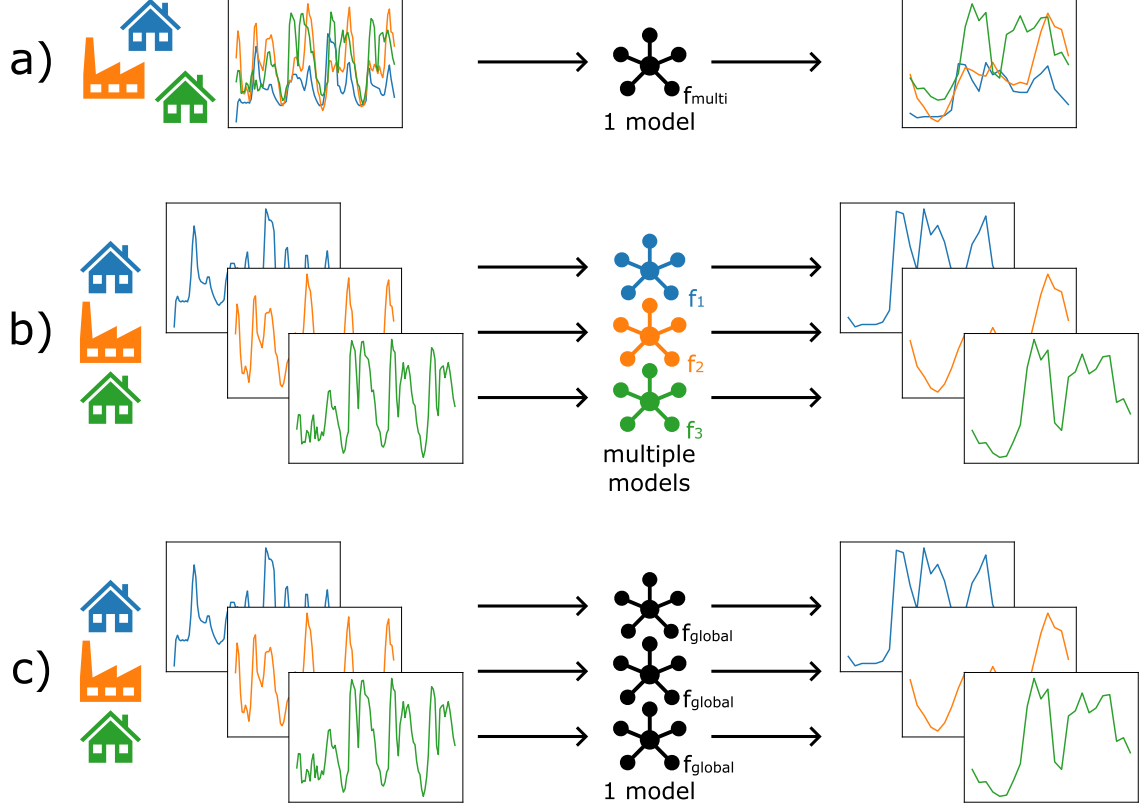


Figure 5.1: The three training strategies, with models depicted as networks. An example with three load time series, four days input and one day output is shown. (a) Multivariate: one model processes all load time series simultaneously; (b) local: separate models (blue, orange, green) process each load time series; (c) global: one model (black) processes all load time series one at a time. Figure from [110].

We compare multivariate, local and global Transformers. The training strategies are depicted in Figure 5.1 and are defined next.

- *Multivariate training strategy:* This strategy trains a single model f_{multi} that forecasts all time series jointly. In the input to the model, each time step is represented by a vector of size $N + E$, where N is the number of load time series and E is the number of exogenous variables. The model forecasts N values for the next H time steps, i.e., its output is a matrix of dimension $(N \times H)$. The multivariate model is defined as:

$$\begin{aligned}
 \hat{\mathbf{y}}_{t,1}[t+1, \dots, t+H], \dots, \hat{\mathbf{y}}_{t,N}[t+1, \dots, t+H] \\
 = f_{multi}(\mathbf{y}_1[t-C+1, \dots, t], \\
 \dots, \\
 \mathbf{y}_N[t-C+1, \dots, t], \\
 \mathbf{X}_{past}[t-C+1, \dots, t], \\
 \mathbf{X}_{future}[t-C+1, \dots, t+H])
 \end{aligned} \tag{5.1}$$

- *Local training strategy*: The local training strategy trains N separate models f_1 to f_N for the N time series, each using the training data from one time series. In the input to a local model, each time step is represented by a vector with $E + 1$ entries for the E calendar features and the electrical load value. The output of each local model is a vector of length H . During inference, the N separate forecasting models $f_1, \dots, f_N$ create one forecast each. The local models f_i are defined as follows:

$$\begin{aligned}\hat{\mathbf{y}}_{t,i}[t + 1, \dots, t + H] = f_i(\mathbf{y}_i[t - C + 1, \dots, t], \\ \mathbf{X}_{past}[t - C + 1, \dots, t], \\ \mathbf{X}_{future}[t - C + 1, \dots, t + H])\end{aligned}\quad (5.2)$$

- *Global training strategy*: The global training strategy trains a single model f_{global} that generalizes for all load time series. The model gets one load time series as input and generates a forecast for that load time series. In contrast to the local models, only one global model is trained, using the samples from all load time series, and this model is used to forecast all load time series separately. This results in N times as many training data for the global model as for a local model. To generate forecasts for all N time series, the global model is used N times with the history of one load time series as input. The output of the global model is a vector of length H , and the model is run N times to generate the forecasts for all N time series. The global model is defined as follows:

$$\begin{aligned}\hat{\mathbf{y}}_{t,i}[t + 1, \dots, t + H] = f_{global}(\mathbf{y}_i[t - C + 1, \dots, t], \\ \mathbf{X}_{past}[t - C + 1, \dots, t], \\ \mathbf{X}_{future}[t - C + 1, \dots, t + H])\end{aligned}\quad (5.3)$$

The Transformer architecture used in this chapter is shown in Figure 5.2. Details on the input and output dimensions, the number of models and the training data size for each training strategy are given in Table 5.1.

Table 5.1: Training strategy details for the Electricity-321 dataset with $N = 321$ load time series, 2.1 years training data and $E = 9$ time and calendar features. For the local models, *training data* is the amount of training data per model.

Training strategy	Models	Encoder input	Decoder input	Output	Training data
Multivariate	1	$C \times 330$	$H \times 330$	$H \times 321$	2.1 years
Local	321	$C \times 10$	$H \times 10$	$H \times 1$	2.1 years
Global	1	$C \times 10$	$H \times 10$	$H \times 1$	$321 * 2.1$ years

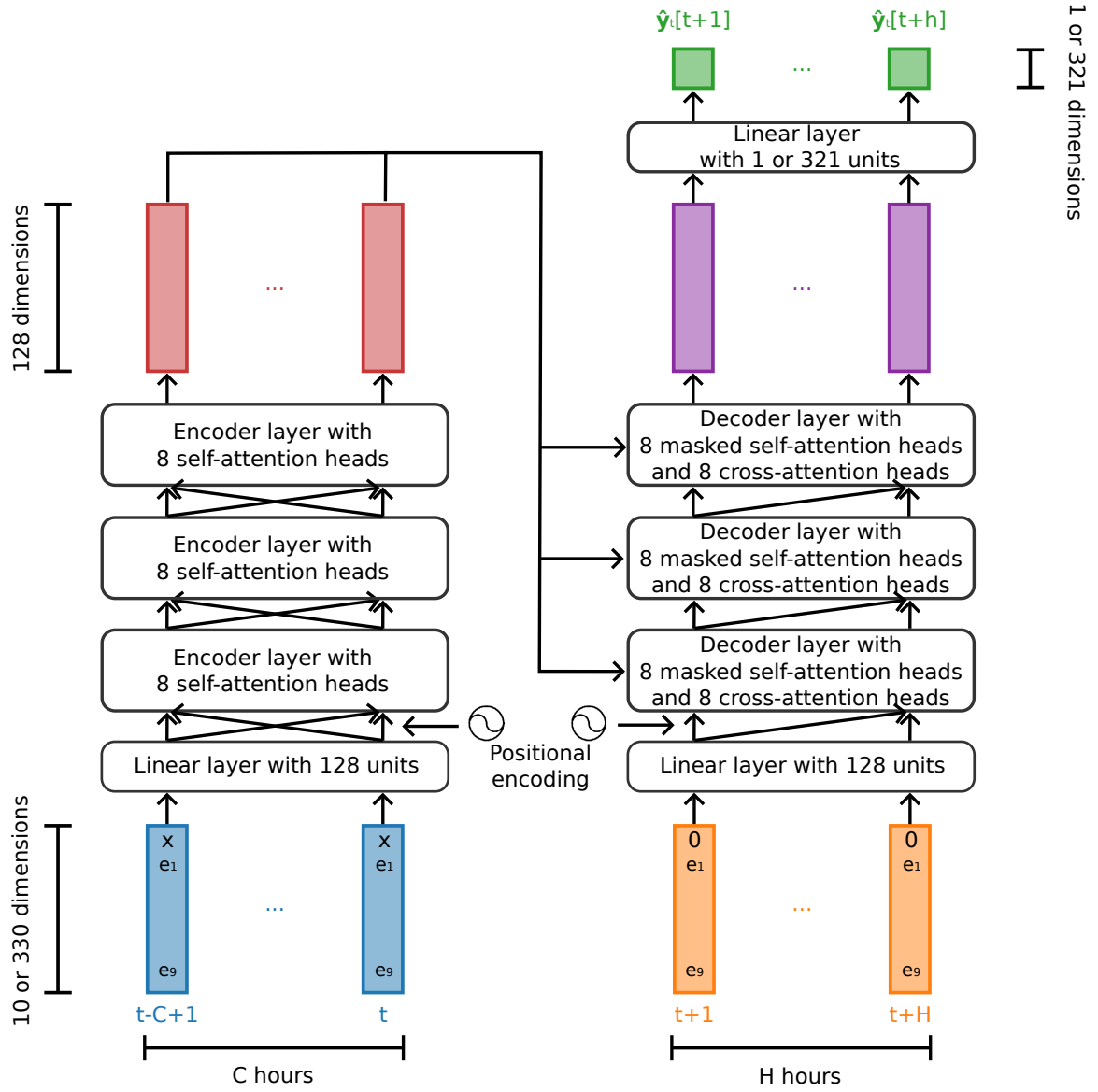


Figure 5.2: Architecture of the Transformer forecasting model. The input and output dimensions differ for the multivariate model and the local and global models. The dimensions shown in the figure refer to the Electricity-321 dataset with 321 time series and nine exogenous features. Figure adapted from [110].

5.2 Experimental Setup

This section introduces the datasets used in the experiments, the comparison methods and details on the experimental setup and model hyperparameters.

5.2.1 Datasets

As recommended in recent literature reviews on load forecasting [100, 125, 237], we conduct experiments on multiple datasets, namely the *Electricity-321* and the *Ausgrid solar home* datasets presented in Chapter 4. For both datasets we make a temporal split and use the first 70 % of each time series for training, the next 10 % for validation, and the last 20 % for testing, as in related work [199, 292, 296, 301]. The models are trained and evaluated with stride 1, i.e., making a forecast at every hour.

5.2.2 Comparison methods

We compare our models with Transformer models from related work [199, 292, 296, 300, 301], a persistence baseline, linear regression models, MLPs and LSTM networks.

- *Models from related work:* For Informer [300], Autoformer [292], FEDformer [301], PatchTST [199] and LTSF-Linear [296], we take the results reported in the publications where applicable, and run the code published with the papers otherwise. All hyperparameters except for the forecast horizon are left unchanged. Autoformer and FEDformer are multivariate models, whereas PatchTST and LTSF-Linear are global models.
- *Persistence baseline:* The persistence baseline takes the value from one week before the predicted hour as a forecast for the 24 hours and 96 hours horizons, and the value from one month before the predicted hour as the 720 hours forecast.
- *Linear regression:* For each load time series, we train a local linear regression model with H outputs. The input consists of load values of the past two weeks (336 hours) and the nine time and calendar features for the current hour t when the prediction is made. Differences to the global LTSF-Linear [296] are that the linear regression models are local models, their parameters are estimated with the ordinary least squares algorithm, and they use time and calendar features.
- *Multi-layer perceptron:* As for the linear regression, we train a local MLP for each load time series. The MLPs get the last 168 load values and the nine time and calendar features of the current time step t as input. Using more than 168 load values as input did not improve the results. Each MLP has two hidden layers with ReLU activation¹ and 1024 neurons per layer.
- *Long short-term memory:* We train multivariate, local and global LSTM [119] models. We use the same architecture as in [152], consisting of two LSTM layers with 20 units each and a linear prediction layer. Using larger models did not improve the results.

¹ <https://docs.pytorch.org/docs/stable/generated/torch.nn.ReLU.html>, accessed 2025-09-05.

5.2.3 Training details

All models are trained with the AdamW optimizer [174] using the mean squared error loss. We use a batch size of 128 and a learning rate of 0.0001 with 1000 warm-up steps and cosine decay with $\gamma = 0.8$. When testing different context lengths C , we find one week to be optimal for the multivariate Transformer and the local Transformers. For the global Transformer, the results improve with increasing context length until $C = 336$ (two weeks), and stay almost the same for $C = 720$ (one month). For Transformer models with two weeks input and one month output, the batch size has to be reduced to 64 due to the quadratic memory consumption of the model. For the multivariate Transformer, the batch size is set to 32 as in related work [292, 300, 301]. The validation error is evaluated every 10,000 training steps and at the end of every epoch. We use early stopping to end the training when no more improvement on the validation set is seen for ten evaluations. For the MLPs, the initial learning rate is set to 0.001 and decayed with $\gamma = 0.5$ after every epoch.

5.3 Results

This section presents the forecasting results and the computational cost of the three training strategies.

5.3.1 Forecast accuracy

Table 10.1 shows the MAE and MSE results on the two datasets. The metrics are computed on the standardized time series. Therefore, the MAE is the same as the sMAE, and the MSE is the same as the sMSE. On the Electricity-321 dataset, the global Transformer is the best model for the 24 hours horizon, and PatchTST is the best model for longer horizons. On the Ausgrid solar home dataset, PatchTST is the best model for all three horizons. The global Transformer beats the local Transformers and the multivariate Transformer across all tested horizons. On average, it reduces the MAE by 21.8% compared to the multivariate Transformer and by 12.8% compared to the local Transformers. Compared to the best local model, the linear regression, it reduces the MAE by 2.9%. Compared to the best multivariate model, FEDformer, it reduces the error by 15.4%. All multivariate models, including Informer, Autoformer, FEDformer and the multivariate Transformer, perform poorly and do not beat the persistence baseline with a lag of one week. The local linear regression models are slightly better than the global linear model, LTSF-Linear, on the Electricity-321 dataset, but it is vice versa on the Ausgrid solar home dataset. The MLP is in five out of six cases a bit worse than the linear regression, with a 1.5% larger MAE on average. The local LSTMs are better than the local Transformers, but the Transformer is better as a multivariate model and as a global model (except for the one month horizon on the Electricity-321 dataset). The forecast errors are lower on the Electricity-321 dataset than on the Ausgrid solar home dataset, which is more volatile as it contains more fine-grained time series from single private houses.

Table 5.2: Forecasting results on the Electricity-321 and Ausgrid solar home datasets, with 24, 96 and 720 hours forecast horizon. MV = multivariate, L = local, G = global. The best results are highlighted in bold and the best results per training strategy are underlined.

Model	Strategy	Context (days)	Electricity-321						Ausgrid solar home					
			24h		96h		720h		24h		96h		720h	
			MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
Informer [300]	MV	4	0.399	0.305	0.407	0.320	0.450	0.384	0.582	0.835	0.607	0.870	0.645	0.891
Autoformer [292]	MV	4	0.289	0.166	0.317	0.201	0.361	0.254	0.579	0.732	0.569	0.754	0.592	0.800
FEDformer [301]	MV	4	0.284	0.164	0.297	0.183	0.343	0.231	0.560	0.707	0.566	0.741	0.609	0.798
LSTM	MV	7	0.400	0.311	0.402	0.314	0.407	0.329	0.611	0.841	0.618	0.851	0.613	0.835
Transformer	MV	7	0.366	0.267	0.384	0.292	0.382	0.300	0.584	0.853	0.586	0.841	0.576	0.800
Persistence	L	-	0.279	0.214	0.279	0.214	0.447	0.490	0.647	1.163	0.647	1.163	0.717	1.353
Linear regression	L	14	0.203	0.103	0.233	0.133	0.296	0.194	0.496	0.604	0.524	0.659	0.565	0.725
MLP	L	7	0.199	0.097	0.236	0.135	0.308	0.212	0.499	0.627	0.532	0.693	0.567	0.749
LSTM	L	7	0.263	0.146	0.283	0.169	0.337	0.234	0.517	0.647	0.541	0.693	0.573	0.751
Transformer	L	7	0.256	0.144	0.289	0.186	0.354	0.271	0.535	0.721	0.563	0.776	0.583	0.802
LTSF-Linear [296]	G	14	0.209	0.110	0.237	0.140	0.301	0.203	0.490	0.598	0.515	0.647	0.553	0.705
PatchTST [199]	G	14	0.190	0.094	0.222	0.129	0.290	0.197	0.468	0.576	0.494	0.641	0.522	0.704
LSTM	G	7	0.207	0.106	0.239	0.139	0.302	0.202	0.491	0.603	0.525	0.667	0.559	0.719
Transformer	G	14	0.184	0.090	0.225	0.127	0.312	0.219	0.482	0.599	0.514	0.665	0.533	0.716

Table 5.3: Training times in hours, measured on a machine with a Nvidia 3090 RTX GPU.

Model	Electricity-321			Ausgrid solar home		
	24h	96h	720h	24h	96h	720h
Linear regression (local)	0.02	0.03	0.08	0.02	0.03	0.07
MLP (local)	0.42	0.42	0.36	0.40	0.39	0.39
LSTM (multivariate)	0.06	0.08	0.30	0.03	0.04	0.10
LSTM (local)	8.25	7.61	7.20	4.27	3.55	3.49
LSTM (global)	1.09	0.82	0.98	1.11	0.84	0.71
Transformer (multivariate)	0.19	0.23	0.88	0.10	0.09	0.39
Transformer (local)	14.20	16.82	102.09	8.33	9.74	62.53
Transformer (global)	3.42	2.00	9.86	3.85	2.85	6.77

5.3.2 Computational cost

The training times for the different models and training strategies are given in Table 5.3. The local Transformer models need by far the most time to train. Their training time increases sharply with longer forecast horizons. The multivariate Transformer trains fast and is even faster than the MLPs for short horizons. Training a global Transformer is much faster than training the many local Transformers but takes longer than the linear regression, MLP and the multivariate Transformer. The LSTM trains always faster than the Transformer with the same training strategy.

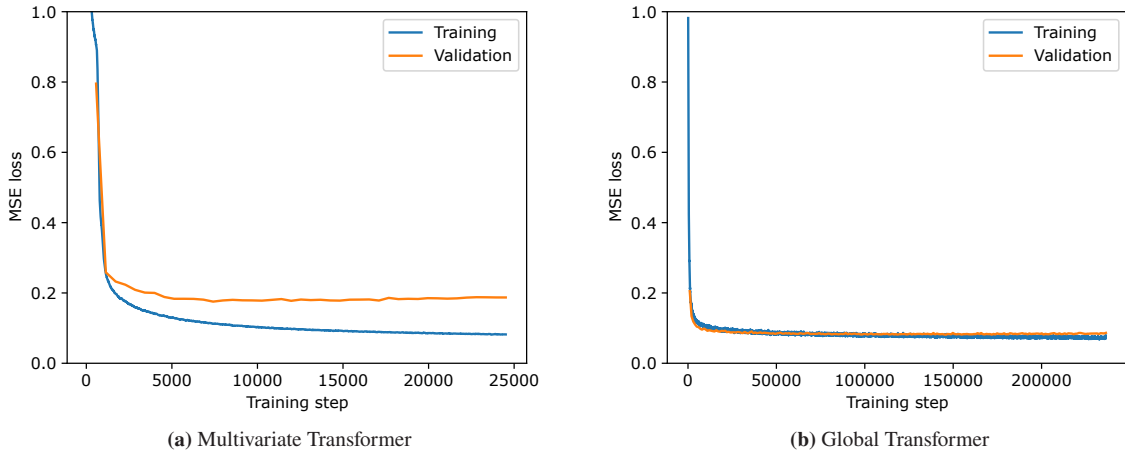


Figure 5.3: Training curves of the multivariate and the global Transformer. The multivariate Transformer overfits early, as can be seen by the gap between training and validation MSE. The global Transformer overfits much later and achieves a lower validation MSE.

5.4 Discussion

Global training improves the Transformer On the two datasets, the global Transformer is superior to the multivariate and local Transformers. We hypothesize that this stems from the larger number of training samples used for the global model (see Table 5.1). Notably, the Transformer benefits from more training data, although the training data is more heterogeneous because it comes from different sources. The multivariate models on the other hand are prone to overfitting, as can be seen in Figure 5.3.

Best Transformer architecture PatchTST is the best model in five out of six cases. However, the difference to the global Transformer is small. This shows that the success of PatchTST is mainly a result of its global training strategy. Its improvement upon the global Transformer can be due to the patching mechanism, a better hyperparameter configuration, or the encoder-only architecture. Among the multivariate models, Autoformer and FEDformer give better results than the multivariate Transformer. A comparison of a flexible Transformer architecture with the standard Transformer and established load forecasting methods is done in Chapter 7.

Comparison with results from related work The global Transformer achieves a better result for short-term forecasting on the Electricity-321 dataset than related work [199, 292, 296, 300, 301], and achieves close results to the best model PatchTST for longer horizons and on the Ausgrid solar home dataset. However, to establish a state of the art for short-term and medium-term load forecasting, a comparison to other forecasting models must be undertaken, including models that are not based on the Transformer architecture and that are more sophisticated than the baselines. Such a comparison is done in Chapter 7, using electrical load datasets that include weather covariates to reflect realistic load forecasting setups.

Linear models are strong baselines As in related work [296], we observe that linear models are strong baselines. The linear regression is in five out of six cases the best local model and only outperformed by the local MLP for the one day horizon on the Electricity-321 dataset. No general answer can be given on whether the local linear regression models are better or the global LTSF-Linear is better, because each variant is better on one dataset.

Potential of transfer learning According to the definition of transfer learning in [213], the global training strategy can be seen as a transfer learning method, because the model must transfer knowledge between different types of buildings with different consumption patterns. Pre-training on other tasks than forecasting or on less similar data from domains other than electricity, as done with TSFMs [167], could improve the results. An advantage of the global model is that it can be applied to new time series without retraining. In [112] it was shown that the Transformer generalizes better to new time series than other approaches, but the forecasts are still better when training data from the target time series is available.

5.5 Conclusion

This chapter presents a comparison of three Transformer training strategies for load forecasting on two datasets with multiple years of data for multiple hundred clients. On both datasets, we find that the right training strategy is crucial to get good results, as the global Training strategy greatly improves the Transformer compared to the local and multivariate training strategies. This answers the first part of RQ 1: Transformers benefit from a global training strategy when multiple time series are available.

The comparison of Transformer training strategies reveals that the multivariate training strategy used in related work [292, 300, 301] is not optimal and it is better to use a global model instead. In particular, this answers the question raised by Zeng et al. [296] of whether Transformers are effective for time-series forecasting, showing that globally trained Transformers outperform the baselines from Zeng et al. [296] for short-term forecast horizons of one and four days. However, simple linear models give decent results for both short-term and medium-term horizons and train much faster than the Transformers. This motivates the benchmarking of established electrical load forecasting models across various datasets, evaluating their predictive performance and runtime, which is described in Chapter 7.

This chapter evaluates three strategies: local, global and multivariate training. A fourth strategy can be established by first clustering similar time series and then training one global model per cluster. This achieves a compromise between local and global models, as cluster-specific models have less training data than the global model, but may benefit from the training data being more homogeneous. This clustering-based approach is introduced and evaluated in the following chapter.

The good transferability of the global Transformer model across time series from different sources highlights the potential of TSFMs [167], which are pretrained on large time-series datasets. The benchmarking in Chapter 7 compares global Transformer models with TSFMs. Future work on fine-tuning TSFMs can build on the training strategies evaluated in this dissertation, because it is an open question whether TSFMs achieve the best results when fine-tuned as local, global, multivariate, or cluster-specific models.

6 Cluster-Specific Transformer Model Training

Previous publication

Sections 6.1–6.4 are modified excerpts from:

[109] M. Hertel^a, L. Ambrosius^a, M. Treutlein, R. Mikut, and V. Hagenmeyer. “A comparison of local, cluster-specific and global Transformer models for forecasting electrical loads of individual buildings and substations”. *IEEE PowerTech 2025*, DOI: 10.1109/Pow-erTech59965.2025.11180482. © 2025, IEEE.

^a Matthias Hertel and Lara Ambrosius contributed equally.

After having compared the local, global and multivariate Transformer training strategies in the previous chapter, this chapter analyzes a fourth training strategy based on clustering similar time series and training cluster-specific models. The assumption behind this training strategy is that it can potentially lead to better models than the local and global strategies, because cluster-specific models have more training samples available than local models, but the data is more homogeneous than the data for a global model. The cluster-specific strategy is commonly used for other forecasting models, as presented in Chapter 2. This chapter contributes an evaluation of the cluster-specific training strategy for Transformers.

With the cluster-specific training strategy, the N time series $\mathbf{y}_1$ to $\mathbf{y}_N$ are grouped into k clusters c_1 to c_k with every $\mathbf{y}_i \in c_j$ for exactly one j . Then, one model f_j is trained per cluster and used to forecast all time series belonging to that cluster:

$$\begin{aligned} \hat{\mathbf{y}}_{t,i}[t+1, \dots, t+H] = f_{j|\mathbf{y}_i \in c_j}(\mathbf{y}_i[t-C+1, \dots, t], \\ \mathbf{X}_{past}[t-C+1, \dots, t], \\ \mathbf{X}_{future}[t-C+1, \dots, t+H]) \end{aligned} \quad (6.1)$$

Time-series clustering could be beneficial in two ways, which are tested in this chapter. First, cluster-specific models could benefit from the more homogeneous data and outperform global models. Second, cluster-specific model selection could improve the results if the best model differs per cluster.

6.1 Methodology

The methodology for the cluster-specific training strategy consists of three steps:

1. Time-series clustering
2. Selection of the best clustering
3. Cluster-specific model training

Each step of the pipeline is explained in detail in the following.

6.1.1 Time-series clustering

First, each time series is represented as a low-dimensional vector which can be used for clustering. Since our datasets contain no metadata that could be used as features, we represent the time series as average weekly load diagrams (AWLDs), thereby including information about weekly variations of the load. The AWLD of a time series consists of 168 values, representing the mean loads of the 168 hours of the week.

Second, the AWLDs are clustered with **k-means clustering**. The number of clusters k is a hyperparameter, which we vary between $k = 2$ and $k = 7$. The algorithm starts with k randomly initialized cluster centers, and assigns each vector to the cluster with the closest center based on the Euclidean distance. Afterwards, the cluster centers are updated. The assignment and update steps are repeated until convergence or until a maximum of 100 iterations is reached. For each k , the algorithm is repeated 50 times with different initializations, and the best resulting clustering is selected based on four cluster validation indices (CVIs) (explained in the next section).

Next to k-means with Euclidean (EUC) distance, three other time-series clustering algorithms are used.

K-means with dynamic time warping (DTW) [70, 205] This algorithm uses DTW instead of the Euclidean distance to measure the distance between data points and cluster centers. DTW is designed for time series. It computes an optimal mapping of two time series, allowing for one-to-many mappings of measurements.

K-shape [208] K-shape follows the iterative process of k-means but uses the shape-based distance in order to form clusters with similar shapes invariant to scaling and shifting. The shape-based distance is based on the cross-correlation of two time series, one of which gets shifted such that the cross-correlation is maximized.

Fuzzy c-means with noise cluster [67] This algorithm allows for a partial assignment of data points to clusters, where the membership degree is a function of the Euclidean distance to the cluster center. The fuzzifier is chosen as $m = 1.15$. We use the fuzzy c-means package [67], which assigns each data point partially to the k clusters. We postprocess the result to get a hard assignment. If the distance of a data point to all cluster centers is larger than a manually defined threshold, it is assigned to the noise cluster.

Otherwise, it is assigned to the cluster with the closest cluster center. We chose 8.9 as the threshold by visual inspection of the AWLDs with the largest distance to the next cluster center. This leads to 21 time series from the Electricity-328 dataset belonging to the noise cluster, and four time series from the Ausgrid dataset respectively. The time series belonging to the noise cluster are forecasted with the global models.

6.1.2 Selection of the best clustering

Each algorithm runs 50 times with different initializations for $k = 2 \dots 7$. The best clustering is chosen for subsequent forecasting. The decision on the best clustering is based on a majority vote using the following four CVIs computed on the AWLDs of the entire time series. Each CVI is briefly explained. For a detailed explanation please refer to the cited sources.

- Within-cluster sum of squares (WCSS). It is the sum of the distances of all data points to their cluster center. Since the WCSS decreases with increasing k , the best clustering is selected based on the elbow method.
- Davies-Bouldin index [63]. It quantifies the average similarity of a cluster and its most akin cluster, taking into account the average within-cluster distance of both clusters divided by the distance of the cluster centers. The best clustering is selected based on the minimum value.
- Silhouette index [231]. It measures how well data points are separated from all other clusters in comparison to their own cluster. The score is based on compactness and separation values. Compactness is computed as the distance between points of the same cluster. Separation is computed as the distance between points and members of the nearest other cluster. The best clustering is selected based on the maximum Silhouette score.
- Calinski-Harabasz index [41]. It divides the between-cluster sum of squares (BCSS) by the WCSS, both normalized by their degrees of freedom. The best clustering is selected based on the maximum value.

6.1.3 Cluster-specific forecast model training

The models produce day-ahead forecasts ($H = 24$ hours) and the context length is set to one week ($C = 168$ hours) in order to provide all weekly characteristics to the model. The exogenous features are the hour of the day, day of the week and week of the year (all cyclically encoded) and workday, holiday, whether the previous day is a workday, whether the next day is a workday, as well as for the Christmas period and for daylight saving times (all binary encoded).

Transformer architecture The Transformer used in this chapter is based on the encoder-decoder architecture introduced in Chapter 3. The encoder gets 168 vectors as input, each of which represents one past time step. The vectors contain the load and exogenous features of the past time steps. The decoder gets H vectors as input, each of which represents a future time step. The vectors contain the exogenous

features of the next time steps. For both the encoder and the decoder, the vectors are fed through a linear layer first to increase their dimension to the model dimension d_{model} and a positional encoding is added. The encoder uses the self-attention mechanism to compute context-dependent embeddings of the input vectors. The decoder uses the cross-attention mechanism to retrieve information from the output of the encoder. The output of the decoder consists of 24 vectors which are fed into a linear layer of size 1 to produce 24 scalar values as the forecast for the next 24 hours.

For each of the k clusters, one cluster-specific model is trained on the training data from all time series belonging to the cluster. Early stopping is used to prevent overfitting.

Hyperparameters We train Transformer models with one, two and three encoder and decoder layers, $d_{model} = 128$, mean absolute error (MAE) loss and AdamW [174] with a batch size of 128 and a learning rate of 0.0005. Early stopping with a patience of five epochs is used to prevent overfitting. The hyperparameter configuration with the lowest nMAE on the validation data is used for the final evaluation on the test data.

6.2 Experimental Setup

This section describes the datasets and the comparison models used in the experiments.

6.2.1 Datasets

We conduct experiments on two publicly available datasets, one from the client level and one from the substation level, in order to represent various use cases on different aggregation levels. The Electricity-328 dataset contains load measurements of industrial or public electricity clients located in Portugal. The time series length is four years, ranging from 2011 to 2014. The *Ausgrid substation dataset* contains load measurements of high to medium voltage substations in Australia. We use the last four years of available data, ranging from 2016 to 2019.

Both datasets are split into training, validation and test periods, following a 70%-10%-20% scheme, i.e. the last 9.6 months are used for testing. Each time series is standardized to have mean 0 and standard deviation 1. The model predictions are back-transformed to the original scale before computing the evaluation metrics. The resulting datasets contain NaN values. These values are ignored when computing the AWLDs for clustering. For training and evaluating forecasting models, examples are skipped when a value is missing in the input or target.

6.2.2 Comparison methods

We compare the Transformer models with multiple baselines, namely a consistency baseline, linear regression and MLPs, each using the local, global and cluster-specific training strategy.

Consistency baseline The consistency baseline predicts the load value from one week before, i.e. $\hat{y}[t] = y[t - 168]$.

Linear Regression (LR) The linear regression gets the previous 168 hourly load values and the time and calendar features of the current timestamp t as input. It is fit with the ordinary least squares algorithm.

Multilayer perceptron (MLP) The MLP gets the same input as the linear regression. It consists of two hidden layers with 2048 units and an output layer with 24 units. Hidden layer sizes from 128 to 2048 units were tested and this was the best configuration on the validation data. The MLP is trained with the same optimizer and learning rate as the Transformer.

Local and global models Next to the cluster-specific LR, MLP and Transformer, these models are also tested as local and global models. Local models are trained on the training data from one time series and used to forecast that time series. Global models are trained on the training data from all time series in a dataset and used to forecast each time series separately.

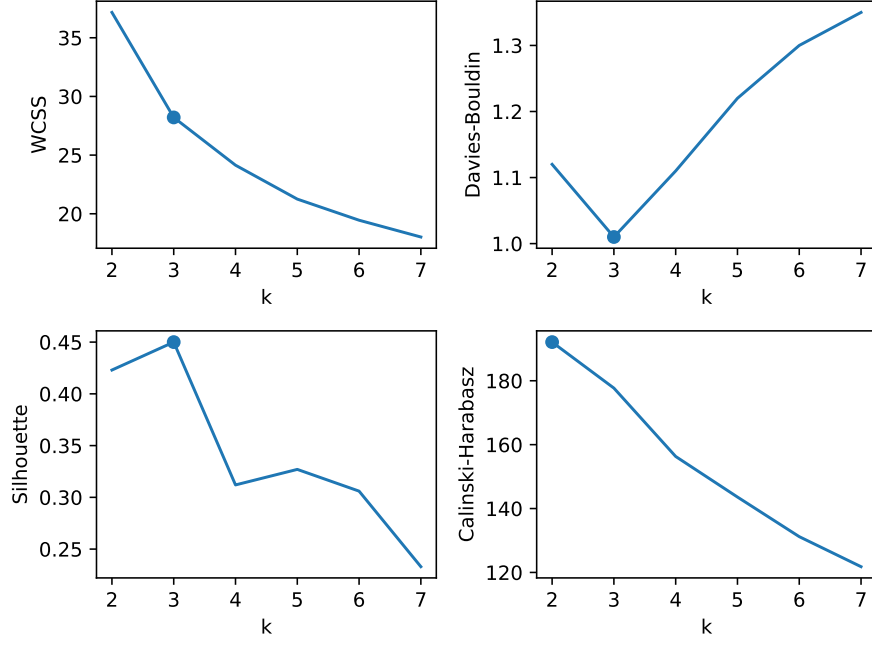
6.3 Results

This section describes (1) the clustering results, (2) the forecasting results, (3) the training runtimes, and (4) insights into the global Transformer model gained by analyzing its attention scores. Then, additional experiments based on the fuzzy c-means clustering are described.

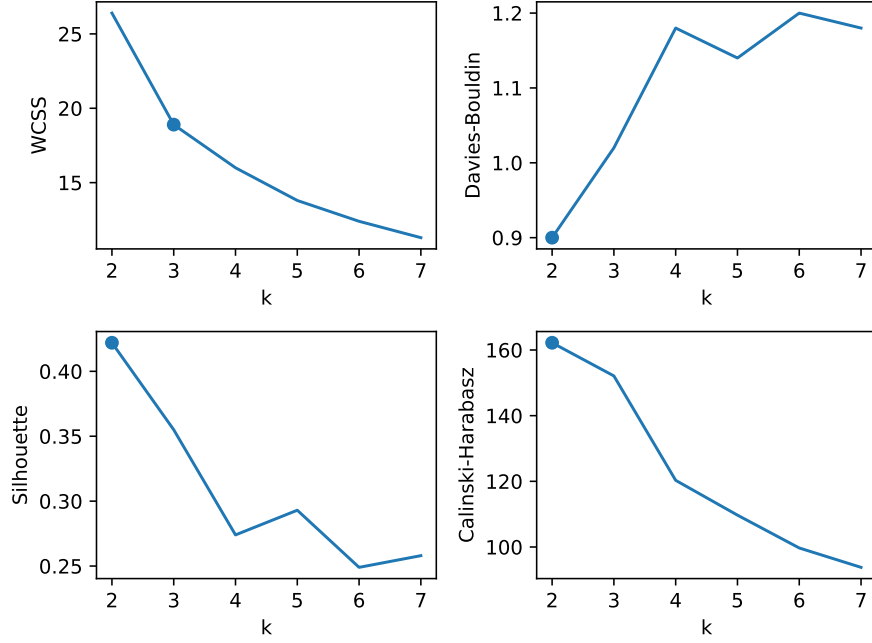
6.3.1 Clustering results

The CVIs for k-means with Euclidean distance and varying numbers of clusters k are shown in Figure 6.1. For Electricity-328, $k = 3$ clusters lead to the segmentation with the best CVIs, whereas on the Ausgrid substation dataset $k = 2$ clusters are optimal.

Figure 6.2 shows the AWLDs of the resulting clusters of the two datasets. The clusterings of both datasets have one cluster with a lower load on the weekend (cluster 1 in both cases). Clusters 1 and 2 on Electricity-328 and cluster 1 on Ausgrid have little variation of the load during daytime, whereas cluster 3 on Electricity-328 and cluster 2 on Ausgrid have a higher load in the evening and a more pronounced dip around noon.



(a) Electricity-328



(b) Ausgrid substations

Figure 6.1: CVIs for k-means with Euclidean distance and varying numbers of clusters k . The best k for each CVI is indicated by a dot. Figure adapted from [109], © 2025, IEEE.

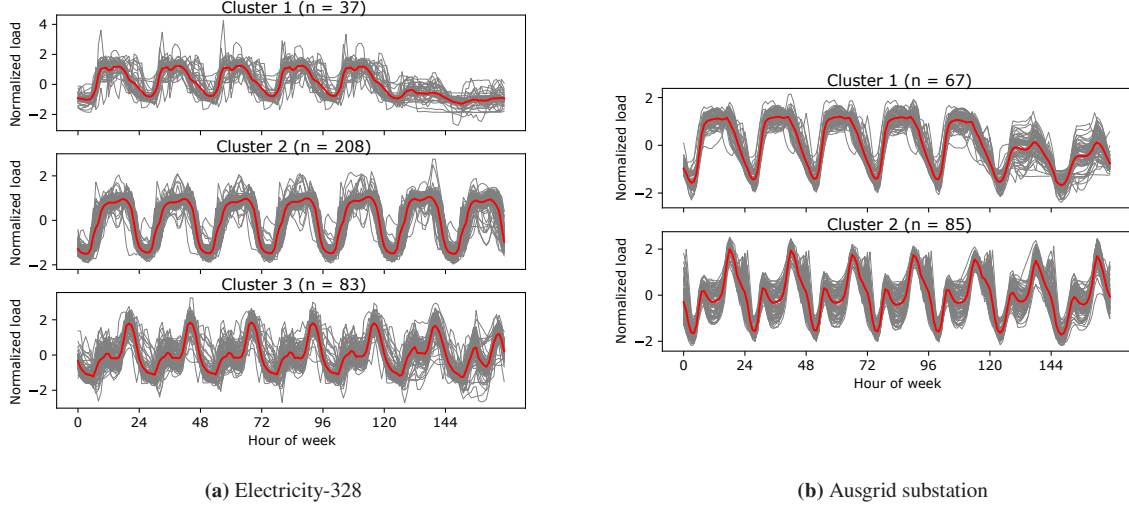


Figure 6.2: Resulting clusters on the two datasets. The cluster centers are shown in red and the AWLDs of the cluster members in gray. Figure adapted from [109], © 2025, IEEE.

6.3.2 Forecasting results

Table 6.1 shows the forecasting results on the test period of the Electricity-328 dataset, averaged across all time series and across five training runs with different initializations of the model weights, and Table 6.2 shows the same for the Ausgrid substation dataset. For each pair out of the ten models, we perform a two-sided t-test for independent samples based on the nMAE results of the five runs. It shows that all nMAE differences in Tables 6.1 and 6.2 are significant with $p < 0.01$. On both datasets, all models beat the consistency baseline. The best model overall and the best training strategy per model is the same for all metrics. For the linear regression, the local training strategy is best on both datasets, followed by the cluster strategy. For the MLP, the results on the two datasets differ. On Electricity-328, the cluster-specific MLP is the best MLP variant, whereas on the Ausgrid substation dataset the local MLP is better than the cluster-specific MLP. For the Transformer, the global strategy is the best on both datasets, followed by the cluster strategy and the local strategy. All models except for the cluster-specific Transformer and global Transformer perform better on the Ausgrid dataset than on Electricity-328. Overall, the global Transformer is the best model on Electricity-328, and the local MLP is the best model on the Ausgrid dataset.

6.3.3 Training runtimes

The training times of one training run are reported in Table 6.3. Notably, the linear regression trains in less than a minute. The MLP models are trained in less than one hour on both datasets. The Transformer models are up to 20 times slower to train than the MLP models.

Table 6.1: Metrics MAE [kW], nMAE [%], sMAE, RMSE [kW] and R^2 on the Electricity-328 dataset. The mean and standard deviation of five runs is reported. The best result overall is highlighted in bold and the best training strategy per model is underlined. Table from [109], © 2025, IEEE.

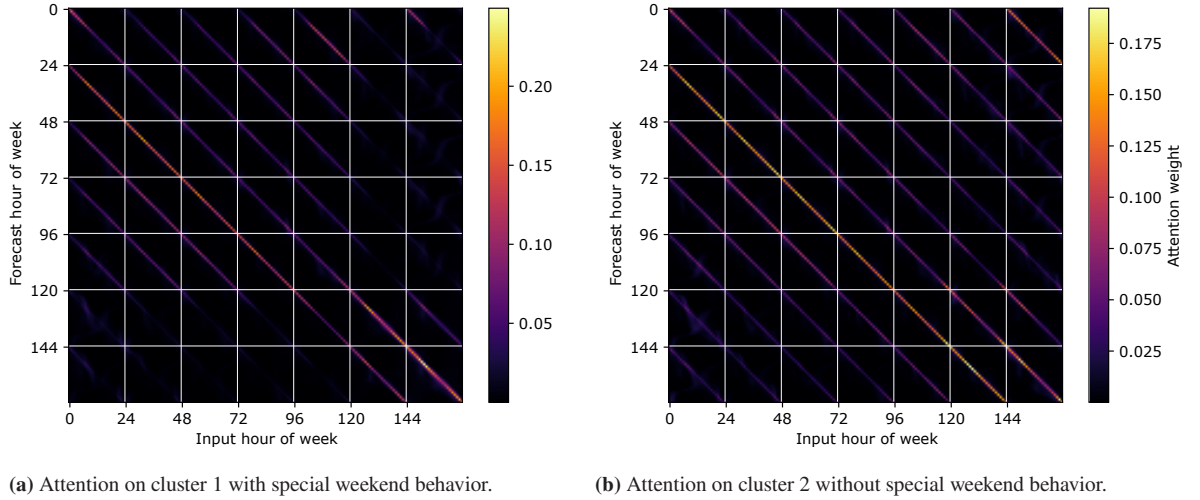
Model	MAE		nMAE		sMAE		RMSE		R^2	
Consistency baseline	63.0	±0.0	11.08	±0.00	0.274	±0.000	101.3	±0.0	0.75	±0.00
LR local	<u>42.8</u>	±0.0	<u>8.27</u>	±0.00	<u>0.204</u>	±0.000	<u>63.4</u>	±0.0	<u>0.88</u>	±0.00
LR cluster	44.5	±0.0	8.60	±0.00	0.210	±0.000	66.5	±0.0	0.87	±0.00
LR global	47.6	±0.0	8.94	±0.00	0.218	±0.000	71.0	±0.0	0.86	±0.00
MLP local	39.5	±0.0	7.44	±0.00	0.188	±0.000	60.3	±0.0	0.89	±0.00
MLP cluster	<u>38.5</u>	±0.0	<u>7.15</u>	±0.01	<u>0.180</u>	±0.000	<u>59.6</u>	±0.1	0.90	±0.00
MLP global	38.9	±0.0	7.23	±0.01	0.182	±0.000	60.1	±0.1	0.90	±0.00
Transformer local	49.2	±0.5	8.83	±0.08	0.223	±0.002	71.8	±0.6	0.86	±0.00
Transformer cluster	37.9	±0.2	6.97	±0.02	0.177	±0.001	60.1	±0.4	0.90	±0.00
Transformer global	37.4	±0.1	6.91	±0.01	0.176	±0.000	59.2	±0.2	0.90	±0.00

Table 6.2: Metrics MAE [kW], nMAE [%], sMAE, RMSE [kW] and R^2 on the Ausgrid dataset. The mean and standard deviation of five runs is reported. The best result overall is highlighted in bold and the best training strategy per model is underlined. Table from [109], © 2025, IEEE.

Model	MAE		nMAE		sMAE		RMSE		R^2	
Consistency baseline	1326	±0	10.90	±0.00	0.345	±0.000	2086	±0	0.62	±0.00
LR local	<u>947</u>	±0	<u>7.74</u>	±0.00	<u>0.245</u>	±0.000	<u>1394</u>	±0	<u>0.83</u>	±0.00
LR cluster	981	±0	7.99	±0.00	0.251	±0.000	1432	±0	0.82	±0.00
LR global	1018	±0	8.19	±0.00	0.259	±0.000	1483	±0	0.81	±0.00
MLP local	800	±3	6.69	±0.00	0.212	±0.002	1269	±6	0.85	±0.00
MLP cluster	803	±2	6.74	±0.01	0.214	±0.001	1278	±5	0.85	±0.00
MLP global	816	±11	6.83	±0.08	0.218	±0.002	1288	±9	0.85	±0.00
Transformer local	883	±5	7.40	±0.05	0.235	±0.001	1328	±5	<u>0.84</u>	±0.00
Transformer cluster	856	±5	7.19	±0.06	0.230	±0.002	1342	±8	0.83	±0.00
Transformer global	<u>840</u>	±8	<u>7.04</u>	±0.07	<u>0.225</u>	±0.002	<u>1320</u>	±15	<u>0.84</u>	±0.00

Table 6.3: Training times in hours, measured on a computer with a Nvidia 3090 RTX GPU. Table from [109], © 2025, IEEE.

Model	Electricity-328	Ausgrid substation
LR local	0.01	0.01
LR cluster	0.01	0.01
LR global	0.01	0.01
MLP local	0.43	0.20
MLP cluster	0.34	0.16
MLP global	0.21	0.10
Transformer local	3.31	2.93
Transformer cluster	4.39	2.31
Transformer global	4.06	2.24

**Figure 6.3:** Attention maps showing the cross-attention scores of the global Transformer on data subsets belonging to two clusters from the Electricity-328 test data (as shown in Figure 6.2). Figure adapted from [109], © 2025, IEEE.

6.3.4 Attention scores

The encoder of the Transformer generates 168 vectors that represent the 168 hours of the previous week. The decoder uses cross-attention to retrieve information from the encoder output. The cross-attention mechanism computes a weighted average of the encoder output vectors (or, more precisely, of linear transformations of these vectors). These cross-attention weights can be interpreted since they show which time steps from the input are relevant to create a prediction. Figure 6.3 shows a visualization of the global Transformer’s cross-attention weights, averaged for all samples from the time series belonging to cluster 1 (left) and cluster 2 (right) of the Electricity-328 dataset. The x-axis is the hour of the week in the input and the y-axis the hour of the week that is predicted. Hour 0 is Monday midnight and hour 167 is Sunday 11 p.m. The attention scores from all forecasts for the test set are collected, leading to attention scores for all forecast hours of the week, despite the forecast horizon being shorter than a week ($H = 24$). A bright pixel at position (row = i , column = j) indicates that the model uses the embedding from hour

of the week j in the input to predict hour of the week i in the output. The rows and columns are sorted from Monday midnight to Sunday 11 p.m. Both plots show strong diagonal patterns shifted by 24 hours. These diagonals indicate that the model attends the same hour of day on previous days. The strongest diagonal starts at position (row = 24, column = 0), i.e. the model attends the time step 24 hours ago. The second-strongest diagonal starts at (row = 0, column = 0), i.e. the model attends the time step one week ago. The plot (a) for cluster 1 shows a special weekend behavior of the model: when predicting a workday, the attention scores for the weekend are low. When predicting a weekend day, the attention scores for the workdays are low. This weekend behavior is not visible in panel (b) for cluster 2, where all days of the week show similar load patterns.

6.4 Discussion

Cluster-specific training does not improve the Transformer The global Transformer is better than the cluster-specific Transformer and the local Transformer on both datasets. Clustering does not improve the Transformer model in our experiments. Therefore, our assumption that the cluster-specific training strategy can improve the Transformer due to more homogeneous data than with global training does not hold. Instead, the global Transformer benefits from having more training data than the cluster-specific Transformers, which are only trained on the data from a subset of the dataset. The analysis of the attention scores from the global Transformer for the subsets of the time series belonging to different clusters shows that the global Transformer is able to adapt its internal behavior to different patterns in the time series, such as whether special weekend behavior is present. Note that this chapter focuses on clustering time series to train cluster-specific forecasting models. Other approaches include clustering within a time series, e.g. by clustering similar days [7, 45, 51, 133, 254, 261]. However, given that the global Transformer generalizes well across time series, we suspect that it is also able to generalize across samples.

Clustering can improve the MLP Cluster-specific training can lead to better models, but does not always do so. As in previous studies [46, 83, 160], the improvement is only observed for a subset of the models – the cluster-specific MLP on the Electricity-328 dataset is the only cluster-specific model that is better than both its local and global counterparts. In all other cases, the cluster-specific models are only better than one of the local and global counterparts, highlighting that they should always be compared to both variants in future studies.

Different models require different strategies The linear regression model cannot adapt to different patterns because of its few parameters and is therefore best used as a local model. The more complex Transformer can adapt to different patterns with the attention mechanism, and is therefore best used as a global model. For the MLP, either the cluster-specific or the local strategy gives the best results.

The best model is dataset-dependent On the Electricity-328 dataset, the global Transformer is the best model. However, on the Ausgrid substation dataset, the local MLP is better. This means that for a new dataset, multiple models need to be evaluated in order to select the best one. Further studies could

analyze which dataset characteristics influence the selection of the best model. The different sizes of the datasets do not explain why the global Transformer is best on the Electricity-328 dataset but not on the Ausgrid substation dataset, as the results on smaller variants of the Electricity-328 dataset are the same (see Appendix A).

MLPs can be favorable in practice MLPs give similar or better forecasts as the global Transformer with less training time. On the Ausgrid substation dataset, local MLPs are better than the global Transformer and train 11 times faster. On the Electricity-328 dataset, the global Transformer is 3.4% better than the best MLP, but trains eight times longer. Further studies are needed that evaluate the benefit of more accurate forecasts in downstream applications like building energy management, in order to estimate both computing-related carbon emissions during model training and avoided emissions due to better forecasts [139]. Depending on the downstream application, it may be more sustainable to use the MLP instead of the Transformer, thereby weighting forecast accuracy against energy usage in model training.

Cluster-specific model selection does not improve the results On both datasets, the overall best model achieves the best (or very similar to the best) result on each cluster (see Appendix A). This means that the overall results cannot be improved by selecting different models per cluster.

FCM does not identify the most difficult time series An advantage of FCM is that it creates a noise cluster containing the elements that differ from the cluster elements. Hypothetically, these unusual time series falling into the noise cluster could be more difficult to forecast. If that was the case, the time series with the largest distances to the closest cluster center would have larger nMAE errors than the rest. However, an analysis of the forecast error in relation to the distance to the closest cluster center reveals that this is not the case (see Appendix A). This means that FCM is not successful in identifying the difficult cases.

6.5 Conclusion

The experiments conducted in this chapter show that the global Transformer is able to adapt to heterogeneous patterns in the data and surpasses both cluster-specific and local Transformers. Although clustering does not improve the Transformer model, it can be beneficial for other forecasting models, particularly for the MLP on the Electricity-328 dataset. This indicates that the effectiveness of clustering depends on the specific forecasting model and dataset characteristics.

This concludes Part II about training strategies for time-series Transformers. The answer to RQ 1 is that the global training strategy is crucial to achieve the best load forecasts with Transformers in scenarios with hundreds of time series from low aggregation levels. The following part builds on this finding and evaluates global Transformers and established methods for load forecasting on varying grid levels.

Part III

Benchmarking Electrical Load Forecasting Models

7 Benchmarking of an Optimized Transformer Architecture Across Grid Levels

Previous publication

Sections 7.1–7.5 are modified excerpts from:

[114] M. Hertel^a, S. Pütz^a, J. Kolar, B. Schäfer, R. Mikut, and V. Hagenmeyer. “A Benchmark for Electrical Load Forecasting Across Grid Levels: Time-Series Transformers Outperform Established Methods”. *DACH+ Energy Informatics 2026*. Preprint available at <https://doi.org/10.48550/arXiv.2607.15705>.^b

^a Matthias Hertel and Sebastian Pütz contributed equally.

^b Licensed under a Creative Commons Attribution 4.0 International License, <https://creativecommons.org/licenses/by/4.0/>.

After having established in Part II that the Transformer is effective for electrical load forecasting when trained with the global strategy, this part benchmarks Transformer architectures against established forecasting models, thereby addressing RQ 2:

Research question (RQ) 2

Does an optimized Transformer architecture outperform established methods for electrical load forecasting on datasets from varying grid levels?

To answer RQ 2, the following three sub-questions are addressed:

1. Do the architectural modifications from related work improve the Transformer’s accuracy?
2. Does an optimized Transformer architecture outperform established load forecasting methods on datasets from varying grid levels?
3. Are zero-shot TSFMs beneficial over models trained from scratch, in terms of forecast accuracy or runtime?

To answer these research questions, our contributions are as follows:

1. We select and process three datasets for benchmarking of newly developed load forecasting methods across grid levels, representing a TSO control area, low voltage (LV) grid feeders and individual clients.
2. We study the effect of Transformer architectures on the forecast accuracy by introducing yet another Transformer (YAformer), a flexible Transformer architecture that integrates modifications from

architectures proposed in the literature and is optimized with hyperparameter optimization (HPO) techniques.

3. We compare the standard Transformer, YAformer and Chronos-2 with seven established forecasting methods and one baseline and highlight their advantages and disadvantages in terms of forecast accuracy and runtime.

The remainder of the chapter is structured as follows: Section 7.1 introduces YAformer and the comparison methods. Section 7.2 presents the experimental setup. Section 7.3 describes the results and Section 7.4 discusses them. Section 7.5 concludes the chapter and highlights potential future work.

7.1 Methodology

This section introduces YAformer, the comparison methods, and the hyperparameter optimization.

7.1.1 YAformer: a flexible Transformer architecture

We introduce YAformer, a flexible architecture that integrates modifications to the standard Transformer proposed in prior work. Its design is controlled by a set of hyperparameters, allowing to represent a variety of time-series Transformer variants. This allows us to systematically compare an optimized YAformer model with the standard Transformer, thereby quantifying the impact of architectural modifications on forecasting accuracy. An overview of the model architecture is given in Figure 7.1. It is an extension of the Transformer architecture from Section 3.2, where patching, CNN layers, LSTM layers, sparse attention and max pooling can optionally be added.

The basis of YAformer is the standard encoder-decoder Transformer [271] adapted for time series. The model gets the past load, past covariates and future covariates as input. For context length C , the encoder gets C vectors as input, each representing the load and covariates (weather data and calendar information) of one past time step. These vectors have one entry for the load and one for each covariate, so their size d_{past} equals the number of covariates plus one. The vectors go through CNN layers with d_{model} filters to embed them in the hidden dimension d_{model} . A stack of n_{layers} Transformer encoder layers creates context-aware embeddings, using self-attention with keys $\mathbf{K}$, values $\mathbf{V}$ and queries $\mathbf{Q}$ to retrieve information from the embedded input series. The encoder output has dimension $C \times d_{\text{model}}$. The decoder gets H vectors as input for a forecast horizon H , each representing the covariates of one future time step, so their size d_{future} equals the number of covariates. Like in the encoder, these vectors are embedded to dimension d_{model} via CNN layers. A stack of n_{layers} Transformer decoder layers create context-aware embeddings, using self-attention to retrieve information from the output of the previous layer and cross-attention to retrieve information about the past time steps from the encoder output. The decoder output has dimension $H \times d_{\text{model}}$ and goes through a stack of dense layers with rectified linear unit (ReLU) activation and a final linear output layer with one unit to create a point forecast of dimension $H \times 1$, containing the forecast values for the next H time steps.

The following architectural modifications can be added via hyperparameters:

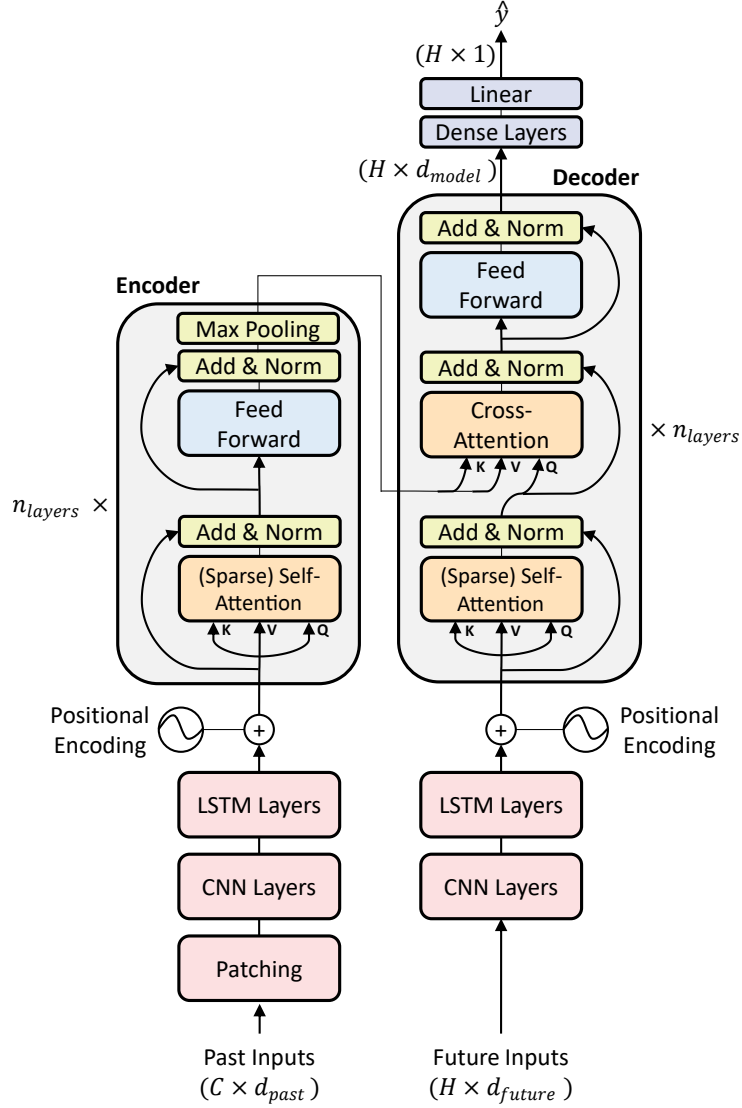


Figure 7.1: Overview of the YAformer architecture based on an encoder-decoder Transformer. The inputs go through patching, CNN and LSTM layers. Sparse self-attention and max pooling can be optionally added in the encoder and decoder layers. The last linear layer produces a point forecast for the next H time steps. Figure from [114].

- **Patching [199]:** Neighboring past input vectors are concatenated before going through the linear embedding layer. The patch size p determines the number of vectors that are concatenated. Since p vectors get concatenated, the encoder input is transformed from dimension $C \times d_{\text{past}}$ to $\frac{C}{p} \times (d_{\text{past}} \cdot p)$.
- **CNN layers [165]:** To encode local information, the linear embedding layer from the time-series Transformer presented in Section 3.2 is replaced by a stack of convolutional layers with d_{model} filters. The number of convolutional layers is controlled by a hyperparameter n_{conv} and the kernel width by a hyperparameter w_{conv} . If $w_{\text{conv}} = 1$, the CNN layers become linear layers.
- **LSTM layers [168]:** Before the Transformer layers in the encoder and decoder, a stack of LSTM layers processes the embedded vectors sequentially.

- Max pooling [300]: After each Transformer encoder layer, max pooling is used to reduce the length of the series. The pooling width $w_{\max}$ is controlled by a hyperparameter.
- Sparse attention [165]: Sparse attention restricts the self-attention mechanisms in the encoder and decoder to use information only from the vectors of previous time steps with an exponential distance to the embedded position (the vector itself and 1, 2, 4, 8, 16, ... steps before). A hyperparameter is used with YFormer to switch between full and sparse attention.

7.1.2 Comparison methods

The following forecasting methods are selected for comparison, including a number of established methods [100, 125, 237] as well as recently developed Transformer models [168, 271] and the TSFM Chronos-2 [11]. All models except for LightGBM are trained with the global strategy.

Persistence baseline The baseline predicts the load value from the same hour on the last observed day that has the same type as the predicted day, distinguishing between three types: workday, Saturday, and Sunday/holiday.

Linear model Like the models from Zeng et al. [296], the linear model gets a feature vector as input and predicts the next H values. The model consists of a single linear layer with $n_{\text{features}} \times H$ weights. It is a global model, meaning that the weights are shared for all time series. The feature vector is the concatenation of C past vectors, containing the load and covariates for the previous time steps, and H future vectors, only containing the covariates for the future time steps.

Light Gradient-Boosting Machine (LightGBM) LightGBM [143] is a machine learning method that builds an ensemble of decision trees using gradient boosting. It is designed to train efficiently and handle large datasets with high performance. For each time series, we train H individual models that predict the H future values based on the past observations and future covariates.

Multilayer perceptron (MLP) The MLP gets a flat vector as input, which is the concatenation of the C past and H future vectors. Multiple fully-connected layers with ReLU activation are stacked on top of each other. The final layer is a linear output layer with H units returning the forecast values.

Convolutional Neural Network (CNN) The CNN gets a sequence of $C + H$ vectors as input. The load value in the future vectors is set to zero as it is not known at prediction time. Multiple convolutional layers with a stride of one are stacked on top of each other. The number of filters and the filter width are controlled via hyperparameters. The output of the last convolutional layer is flattened and fed into a stack of fully-connected layers with ReLU activation, where the last layer has H units and linear activation to create the forecast values.

Long Short-term Memory (LSTM) A hyperparameter for the LSTM allows to switch between an encoder-decoder model and an encoder-only model. The inputs to the encoder and decoder are the same as with YAformer. The vectors get embedded to dimension d_{model} by linear layers. The encoder and decoder are bidirectional LSTMs. The last hidden vector of the encoder is fed to the decoder as its first hidden vector. The decoder output for each future time step goes through a stack of linear layers and a final output layer with one unit to create the H forecast values. In the encoder-only variant, all input vectors are fed to the encoder LSTM stack, whose output is flattened and goes through a stack of linear layers and a final output layer with H units to create the H forecast values.

N-HITS N-HITS [44] is a stack of MLPs, each of which returns a time series with a different time resolution. The forecast is finally computed as the sum of the returned (interpolated) time series.

Transformer We use a standard encoder-decoder Transformer. It gets the same inputs as YAformer, but does not make use of the architectural modifications introduced for YAformer.

Temporal Fusion Transformer (TFT) TFT [168] is an encoder-decoder model, receiving the same inputs as YAformer. The first layer of the encoder and decoder is a variable selection layer. It is followed by unidirectional LSTM layers and a single Transformer layer. Causal masking is used in the attention mechanism to prevent the Transformer layer from using information of time steps after the prediction time step.

Chronos-2 Chronos-2 [11] is a TSFM based on a Transformer with patched inputs. It is pretrained on a large dataset containing real time series data from various domains, as well as synthetic data with covariate dependencies. It is the best open model on the GIFT-Eval benchmark, particularly outperforming the TSFMs Moirai [290], TimesFM [60] and TabPFN-TS [127]. We tested multiple context lengths and the best results are achieved with $C = 8192$, the maximum context length supported by the model.

7.1.3 Hyperparameter optimization (HPO)

The architecture modifications of YAformer can be controlled via model hyperparameters. Note that the standard Transformer is a special case of YAformer with patch size one, kernel width one, zero LSTM layers, pooling width one and full attention. The hyperparameters of all models (YAformer and the comparison models) are optimized using Bayesian Optimization in weights and biases [35] with 100 hyperparameter configurations. The HPO of LightGBM ran less than 100 iterations because of long training times. The best hyperparameters found with the HPO are given in Appendix B.

7.2 Experimental Setup

This section describes the three datasets and the evaluation scheme used in the experiments.

7.2.1 Datasets

Table 7.1: Dataset characteristics.

Dataset	Aggregation	Time series	Period
TransnetBW	TSO control area	1	1.1.2015–31.12.2025
FeederBW	LV feeders	200	1.4.2023–31.3.2025
Electricity-287	Clients	287	1.1.2011–31.12.2014

The proposed benchmark contains three datasets from different load aggregation levels, thereby representing use cases in the transmission grid, the distribution grid, and for individual consumers. These datasets are described next. An overview of the datasets is given in Table 7.1. All three datasets are presented in detail in Chapter 4.

TransnetBW The TransnetBW dataset introduced in Chapter 4 comprises the electrical load of Baden-Württemberg in hourly resolution. We use air temperature, irradiance, precipitation, wind speed at 10 meters height, and a binary holiday feature as covariates. The years 2015–2022 are used as the training set, the year 2023 is used as the validation set for the HPO, and the years 2024 and 2025 are used as the test set.

FeederBW The FeederBW dataset introduced in Chapter 4 comprises the electrical load from 200 low-voltage feeders in Baden-Württemberg in hourly resolution, together with short-term weather forecasts associated with the post code of each feeder. The length of the time series is two years, ranging from 1.4.2023 to 31.3.2025. We take March 2023 as the validation set and the last twelve months (1.4.2024–31.3.2025) as the test set. We use irradiance, air temperature, precipitation, and a binary holiday feature as covariates.

Electricity-287 The Electricity-287 dataset introduced in Chapter 4 contains 287 load time series from individual clients of an energy trading company from Portugal in hourly resolution and combined with weather data from Portugal. The same test split as in GIFT-Eval [4] and Chronos-2 [11] is used to allow for a fair comparison to Chronos-2, which is trained on the training data from GIFT-Eval. Therefore, the last 10 % (8.8.2014 – 31.12.2014) are used as the test set, and we use the same portion of data (15.3.2014 – 7.8.2014) for the validation set. We use air temperature, irradiance, precipitation, wind speed, and a binary holiday feature as covariates.

7.2.2 Evaluation scheme

Forecasting task The task addressed in this work is to forecast the hourly loads for the next four days (96 hours) based on past observations and past and future covariates. We see this as representative for many applications, which often require planning more than one day ahead. Forecasts are generated with

a stride of one hour, i.e., at every hour, the next 96 values are predicted. For the datasets with multiple time series (FeederBW and Electricity-287), individual forecasts for every time series are made.

Model training The deep learning models are trained with a global training strategy, which was found to be superior to local and multivariate models (see Chapter 5), whereas LightGBM is trained as local models. In addition to the weather and holiday covariates, the models trained from scratch get covariates as input that are derived from the timestamp, namely the hour of day, day of week and month (all cyclically encoded as two features). All deep learning models are trained with AdamW [174] and the mean squared error loss function, one validation per epoch on TransnetBW and ten validations per epoch on the other datasets, using early stopping with a patience of ten validations without improvement.

Metrics We use the MAE (Equation 3.3) to evaluate the forecasts for the TransnetBW and FeederBW datasets. To account for the different magnitudes of the time series in the Electricity-287 dataset, the nMAE (Equation 3.6) is used instead of the MAE. The metrics are computed per time series on the entire test set and the mean of all time series is reported as the final metric value. Results for additional metrics are reported in Appendix B.

Retraining For the datasets with at least one year test period (TransnetBW and FeederBW), new models are trained every month, using the last month as validation data and everything before as training data. This allows the models to adapt to potential concept drifts. For the Electricity-287 dataset, a static split into training (1.1.2011–14.3.2014), validation (15.3.2014–7.8.2014) and test data (8.8.2014–31.12.2014) is used.

Statistical comparison For statistical evaluation of the observed performance differences, we apply Diebold–Mariano tests (DM tests) [68] on the per-series loss differentials between competing forecasting models. For each dataset and model, the absolute error per sample is averaged over the forecasting horizon, yielding one aggregated loss value per series and forecast origin. For each pair of models, we compute the corresponding loss differentials. For each series, we separately perform two one-sided DM tests, resulting in two p -values corresponding to the hypotheses that either model 1 or model 2 achieves significantly lower predictive loss. A model is counted as significantly better for a given series if the corresponding one-sided DM test rejects the null hypothesis at the 5% significance level. In addition to the DM tests per time series, we report p -values of a t -test and a permutation test comparing the overall performance of the best model with each other model in Appendix B.

7.3 Results

This section describes the forecasting results on the three datasets, the model runtimes, and the results from ablation studies.

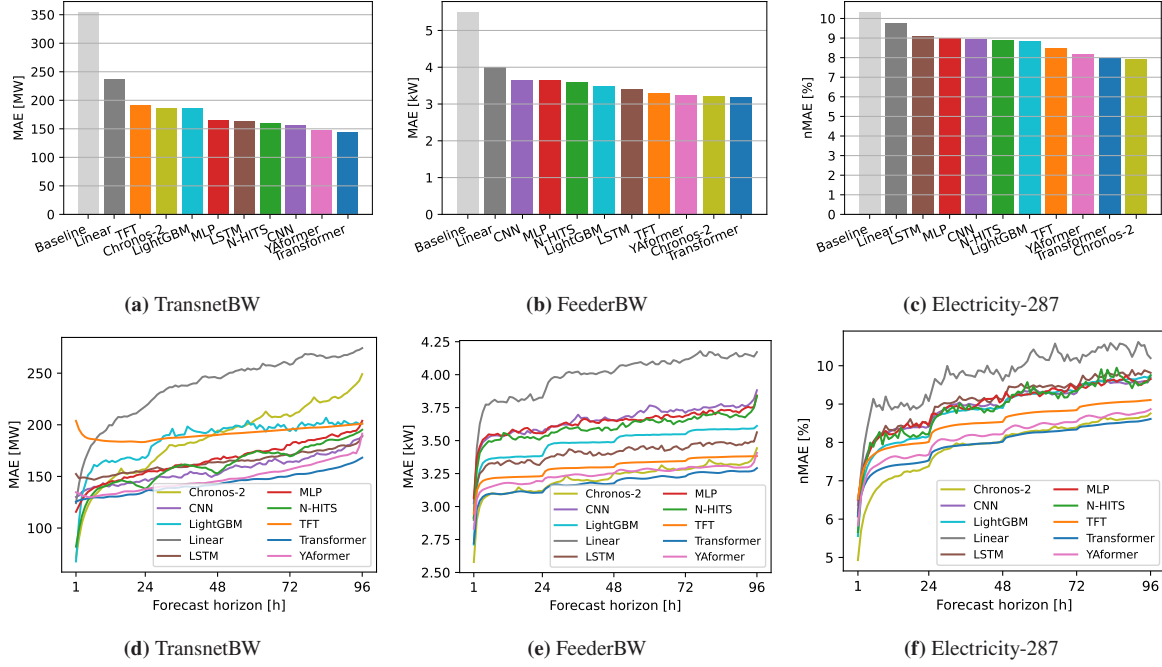


Figure 7.2: Benchmarking results on the three datasets. Figure from [114].

7.3.1 Model comparison

Figure 7.2 presents the forecast error of the evaluated models on the three datasets. All models outperform the baseline consistently. The improvement of the best model compared to the baseline is 59.6 % on TransnetBW, 42.3 % on FeederBW and 23.3 % on Electricity-287, indicating that it is easier to outperform a simple approach on highly aggregated data than on the more volatile LV feeder and client-level data.

The Transformer achieves the lowest forecast error on the TransnetBW and FeederBW datasets, whereas it is outperformed by Chronos-2 on the Electricity-287 dataset. Note that the training set of the Electricity-287 dataset is contained in the pretraining data of Chronos-2, so the evaluation on this dataset is not zero-shot.

On all three datasets, the best models are based on time-series Transformer architectures; namely, the two best models on TransnetBW (Transformer and YAformer) and the four best models on FeederBW and Electricity-287 (Transformer, Chronos-2, YAformer and TFT). The best Transformer model achieves 8.3 %, 6.6 %, and 10.7 % lower forecast errors than the best non-Transformer model (CNN, LSTM and LightGBM) on TransnetBW, FeederBW and Electricity-287, respectively. The Transformer outperforms YAformer on all three datasets, meaning that the architectural modifications do not improve the forecast error compared to that of a standard Transformer.

The forecast performance of Chronos-2 varies greatly across the three datasets. Chronos-2 is the best model on the Electricity-287 dataset, and second-best on the FeederBW dataset, but has a much larger MAE than the best models on the TransnetBW dataset. A closer look at the model performance over the test period in Figure 7.3a reveals that Chronos-2 has large errors around holidays (e.g., 1.1.2024, 28.3.2024, 1.5.2024, 3.10.2024, the Christmas period, 17.4.2025, 3.10.2025). Figure 7.4 shows four forecasts where Chronos-2 exhibits large errors. The beginning of January is characterized by low loads

due to winter holidays, and Chronos-2 fails to adjust the load to normal levels after the holiday period. Chronos-2 predicts a higher load on Good Friday (29.3.2024), potentially because the previous holiday (6.1.2024) did not affect the load because it was on a weekend. Ascension Day (9.5.2024) is predicted correctly, but Chronos-2 does not predict the long weekend. Before Christmas, many people take the full week off, and 24.12.2024 is not an official holiday, so Chronos-2 predicts a load that is too high. We evaluated Chronos-2 on TransnetBW with additional binary features for long weekends and school holidays, which reduces its MAE to 180 MW, but it is still outperformed by the deep learning models trained from scratch.

On the FeederBW dataset (Figure 7.3b), the forecast errors are higher in winter, when the load is generally higher, with spikes in April and around Christmas. In this case, the Transformer outperforms Chronos-2, especially from September to November. On the Electricity-287 dataset (Figure 7.3c), Chronos-2 outperforms the Transformer consistently, except for the Christmas period, where both models exhibit large errors.

The forecast errors evaluated for forecast horizons from one hour to four days are shown in Figure 7.2d-f. LightGBM is the best model for one-step forecasts on the TransnetBW dataset. Chronos-2 is good for short forecast horizons, outperforming all other models for two to six hours horizon on TransnetBW, for the first six hours on FeederBW, and for the first 31 hours on Electricity-287. The error of the Transformer increases less with longer horizons, making it the best model for long horizons.

The results of the DM tests to compare the models per time series are given in Figure 7.5. For each pair of models, the number in the figure is the fraction of time series where one model is significantly better than the other model with $p < 0.05$. The TransnetBW dataset contains only one time series, so the fraction of time series can only be 100 % or 0 %. Transformer and YAformer are not significantly different, and they outperform all other models significantly. On the FeederBW dataset, the result is mixed, with the Transformer outperforming Chronos-2 on 46 % of the time series, but vice versa on 20 % of the time series (and no significant difference on the remaining time series). On the Electricity-287 dataset, Chronos-2 has the best nMAE, but it outperforms the Transformer only on 14 % of the time series, whereas the result is vice versa on 39 % of the time series (and no significant difference on the remaining time series). The reason is that Chronos-2 is particularly good on difficult time series with a high nMAE, but it is outperformed on many time series with lower nMAE.

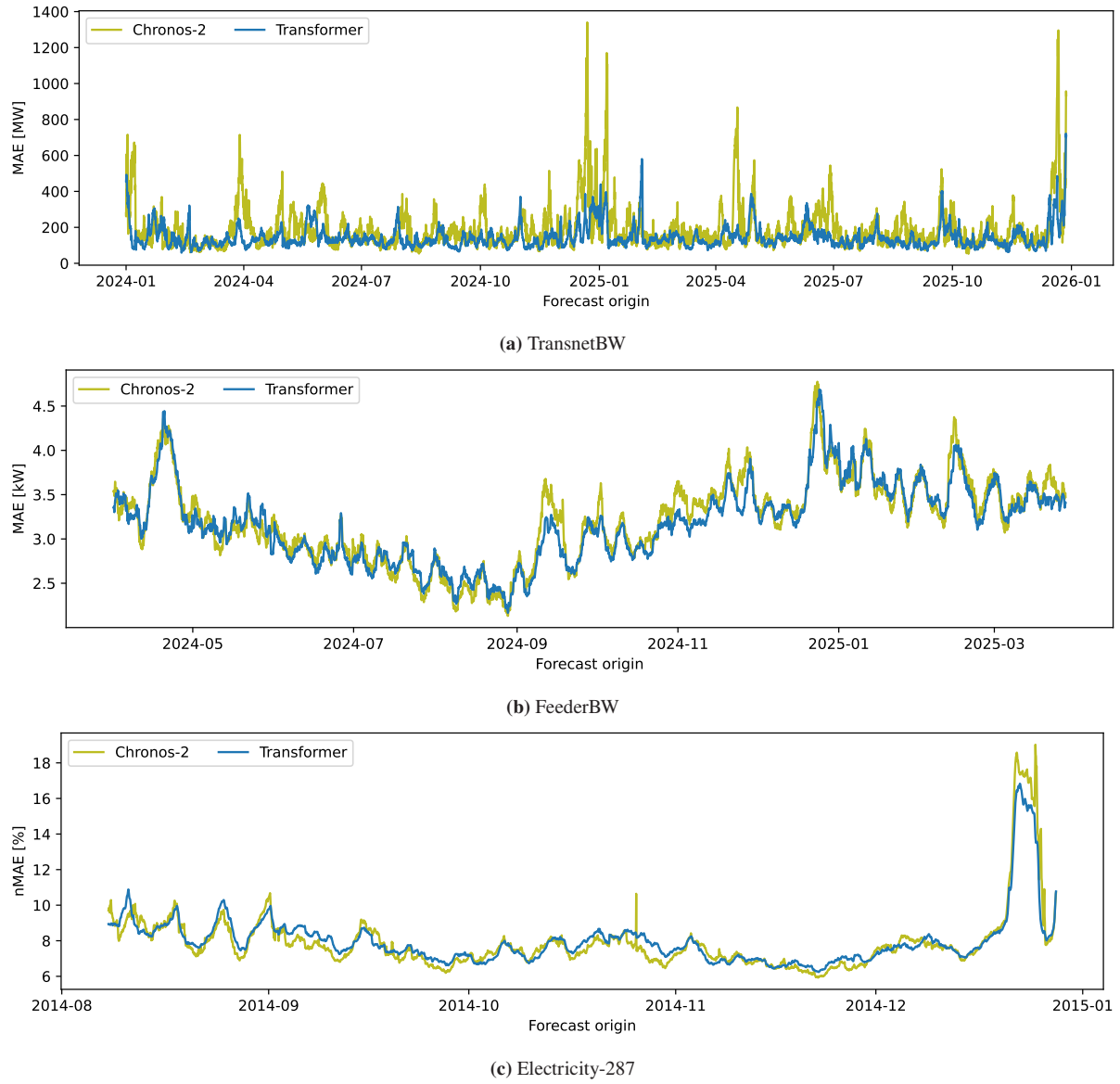
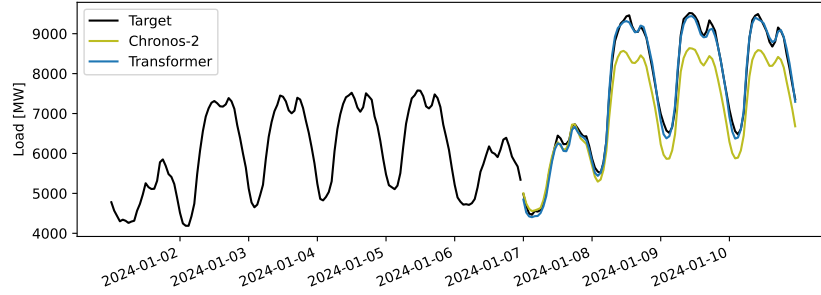
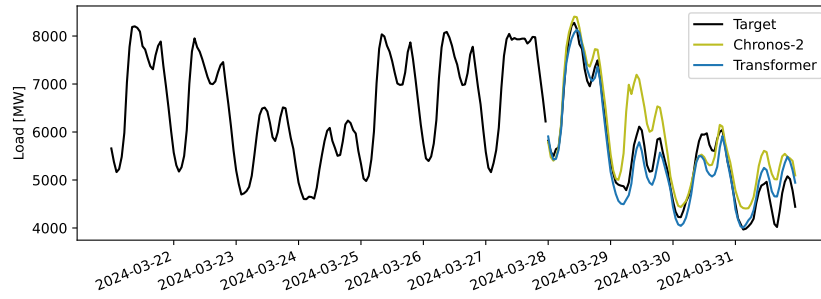


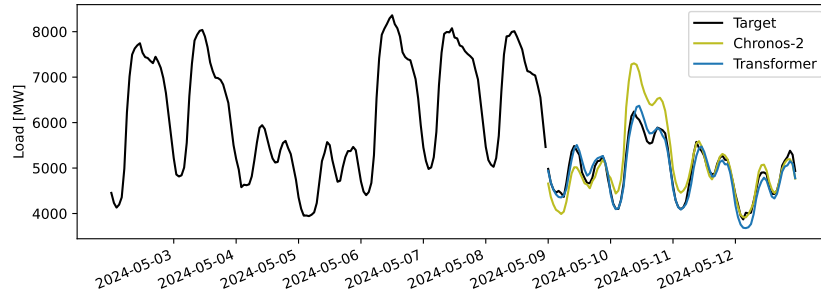
Figure 7.3: Comparison of the forecast error of the Transformer and Chronos-2 over time. Figure from [114].



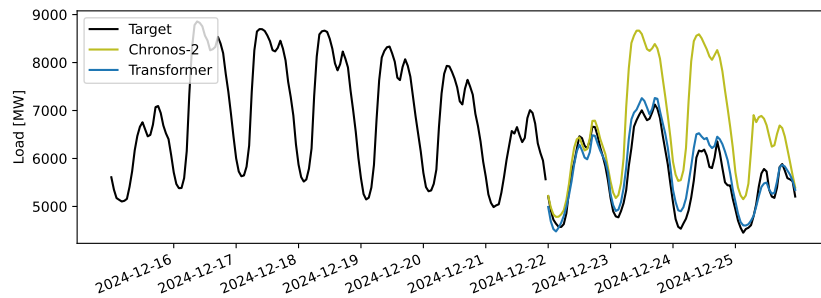
(a) Forecast from 7.1.2024.



(b) Forecast from 28.3.2024.



(c) Forecast from 9.5.2024.



(d) Forecast from 22.12.2024.

Figure 7.4: Exemplary forecasts of the Transformer and Chronos-2 on the TransnetBW data, showing cases where Chronos-2 exhibits large errors. Figure from [114].

Transformer -	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
YAformer -	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Chronos-2 -	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
TFT -	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
LSTM -	0.00	0.00	1.00	1.00	0.00	0.00	0.00	1.00	1.00
N-HITS -	0.00	0.00	1.00	1.00	0.00	0.00	1.00	1.00	1.00
CNN -	0.00	0.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00
MLP -	0.00	0.00	1.00	1.00	0.00	0.00	0.00	1.00	1.00
LightGBM -	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Linear -	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

(a) TransnetBW

Transformer -	0.61	0.46	0.85	0.95	0.98	0.99	0.98	0.85	0.99
YAformer -	0.01	0.27	0.49	0.81	0.96	0.99	0.97	0.76	0.99
Chronos-2 -	0.20	0.31	0.48	0.78	0.98	1.00	0.99	0.73	1.00
TFT -	0.01	0.01	0.17	0.65	0.92	0.97	0.94	0.64	0.98
LSTM -	0.00	0.01	0.02	0.05	0.94	0.96	0.94	0.47	0.98
N-HITS -	0.00	0.01	0.00	0.01	0.01	0.61	0.57	0.20	0.97
CNN -	0.00	0.01	0.00	0.00	0.01	0.01	0.09	0.14	0.90
MLP -	0.00	0.01	0.00	0.01	0.01	0.02	0.41	0.17	0.94
LightGBM -	0.03	0.07	0.04	0.13	0.28	0.55	0.65	0.61	0.83
Linear -	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.04

(b) FeederBW

Transformer -	0.23	0.39	0.72	0.91	0.95	0.97	0.93	0.67	0.98
YAformer -	0.05	0.20	0.51	0.74	0.85	0.91	0.86	0.57	0.95
Chronos-2 -	0.14	0.16	0.40	0.64	0.76	0.90	0.82	0.42	0.98
TFT -	0.00	0.02	0.07	0.59	0.46	0.79	0.56	0.29	0.89
LSTM -	0.00	0.00	0.01	0.01	0.18	0.33	0.21	0.14	0.53
N-HITS -	0.00	0.01	0.00	0.02	0.07	0.55	0.21	0.18	0.90
CNN -	0.00	0.00	0.00	0.02	0.02	0.01	0.01	0.14	0.59
MLP -	0.00	0.00	0.00	0.02	0.05	0.04	0.59	0.17	0.82
LightGBM -	0.02	0.06	0.07	0.13	0.28	0.43	0.60	0.46	0.76
Linear -	0.00	0.00	0.00	0.02	0.02	0.01	0.05	0.03	0.08

(c) Electricity-287

Figure 7.5: Statistical test results on the three datasets. A Diebold-Mariano test is performed per time series for each pair of models. The fraction of time series on which the model in the column is better than the model in the row with $p < 0.05$ is given. Figure from [114].

7.3.2 Runtime

Table 7.2: Runtime for one model training, one inference (one forecast for all time series), and annual total runtime ($12 \cdot \text{training} + 8760 \cdot \text{inference}$).

Model	TransnetBW			FeederBW			Electricity-287		
	Training [h:mm:ss]	Inference [ms]	Annual [h:mm:ss]	Training [h:mm:ss]	Inference [ms]	Annual [hh:mm:ss]	Training [hh:mm:ss]	Inference [ms]	Annual [hh:mm:ss]
Baseline	-	1.4	0:00:12	-	266.25	0:38:52	-	373.4	0:54:30
Chronos-2	-	78.3	0:11:26	-	11,160.7	27:09:27	-	16,015.6	38:58:16
CNN	0:03:53	5.4	0:47:29	0:36:06	28.1	7:17:19	1:42:37	35.8	20:36:37
LightGBM	0:18:42	1.2	3:44:36	10:04:12	269.7	121:29:46	50:20:28	239.3	604:40:32
Linear	0:05:59	5.2	1:12:39	0:37:22	38.5	7:17:19	1:11:35	38.9	14:28:16
LSTM	0:10:07	15.3	2:03:37	0:49:03	74.6	9:59:32	2:40:34	55.2	32:14:51
MLP	0:05:29	5.2	1:06:29	0:47:05	38.5	9:30:42	2:48:51	29.2	33:50:27
N-HITS	0:06:45	5.9	1:21:49	0:48:18	39.2	9:45:20	3:09:58	41.5	38:05:39
TFT	0:14:08	10.8	2:51:12	1:24:45	46.1	17:03:44	5:17:30	75.2	63:40:58
Transformer	0:11:40	6.4	2:21:00	1:34:26	72.5	19:03:48	5:03:10	60.6	60:46:50
YAformer	0:11:33	6.3	2:19:32	0:50:25	82.6	10:17:04	5:07:09	95.2	61:39:42

Training and inference runtimes for all models are given in Table 7.2¹. The training time is defined as the average time needed to train a global model. The inference time is defined as the average time needed to create a forecast for all time series in a dataset (one for TransnetBW, 200 for FeederBW, 287 for Electricity-287). The deep learning models trained by us are evaluated with a batch size equal to the number of time series, i.e., all time series are put into a single batch, whereas Chronos-2 is evaluated with a maximum batch size of 32, which resulted in the fastest throughput. We calculate the annual runtime in a scenario where the models are retrained every month and a forecast is made at every hour for the next 96 hours. This means that twelve models are trained and 8760 forecasts are made per year, so the annual runtime consists of twelve trainings and 8760 inference instances. This annual runtime is visualized in Figure 7.6, with the training part dashed. For the models trained by us, the annual runtime is dominated by the training time, which comprises almost all of the annual runtime. The inference time is fast, ranging between 5 ms and 95 ms per inference on the three datasets (see Table 7.2), making up only a small fraction of the annual runtime in Figure 7.6. Chronos-2 is used as a pretrained model, so it has zero training time in our scenario. This makes it the fastest model overall on the TransnetBW dataset, where only one time series needs to be forecast. On the other datasets, which have 200 and 287 time series, the slower inference time of Chronos-2 outweighs the fact that it has zero training time. The training of LightGBM takes long because it involves training individual models per time series and per forecast horizon (therefore, it is excluded from Figure 7.6).

¹ The runtimes are evaluated on a single high-performance computing (HPC) node with an NVIDIA A100 GPU with 40 GB VRAM, 19 CPU cores of an Intel Xeon Platinum 8368 and 128 GB RAM.

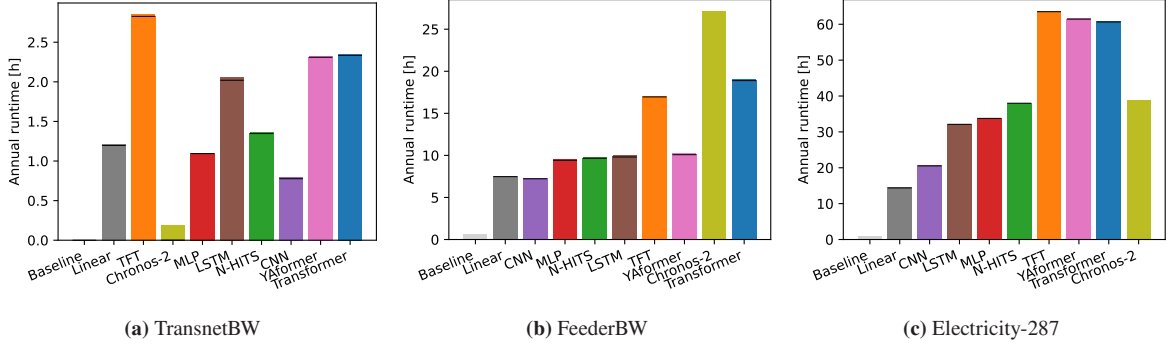


Figure 7.6: Annual total runtimes for twelve monthly model trainings and 8760 inferences on the three datasets. The training times are shown as dashed bars. Figure adapted from [114].

7.3.3 Ablations

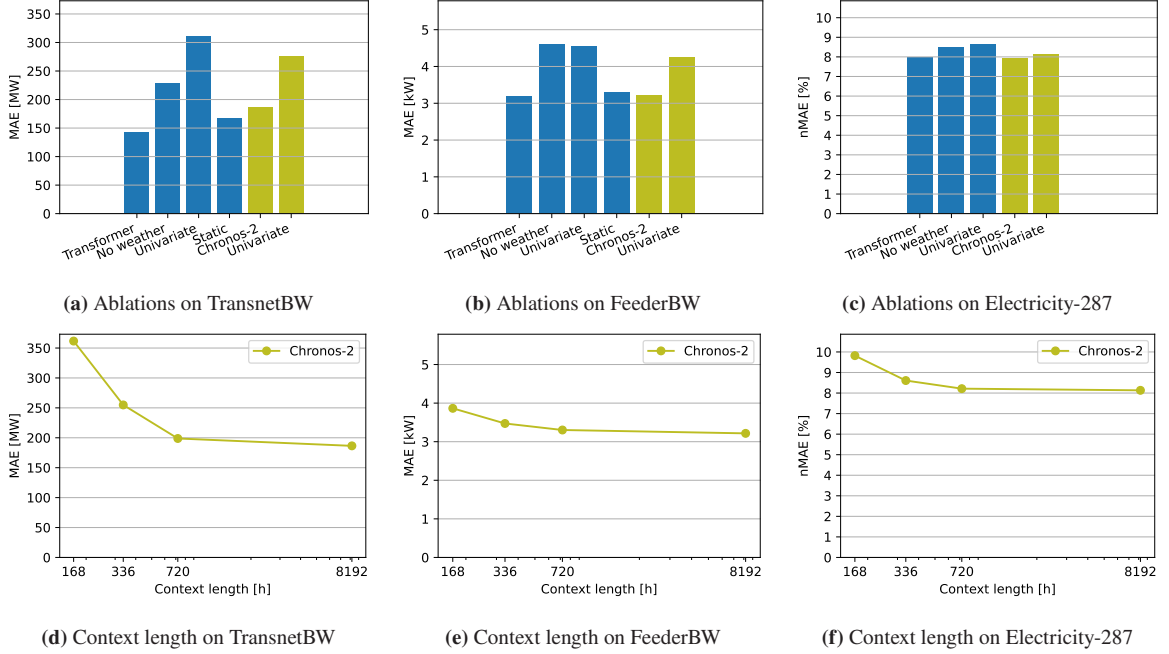


Figure 7.7: Ablation results on test data for models without weather data (*no weather*), without covariates (*univariate*), and without retraining (*static*), as well as reduced context length for Chronos-2. Figure from [114].

Figure 7.7 shows results of ablated models, where covariates are removed, static training is performed instead of monthly retraining, and the context length is altered.

Covariates Covariates are important to achieve the lowest forecast errors. This holds for all three datasets, as the Transformer with covariates is always better than the univariate Transformer. The same holds for Chronos-2, which also benefits from covariates. The improvement by the integration of covariates is large on the TransnetBW and FeederBW datasets, which are largely influenced by behind-the-meter photovoltaic power generation. The effect is smaller, but also important, on the Electricity-287 dataset, where some of the load is also driven by weather effects. Apart from weather covariates, calendar features

are important on the TransnetBW and Electricity-287 datasets, as the industrial activity is reduced during holidays, leading to a lower electrical load.

Continuous retraining On TransnetBW and FeederBW, a rolling evaluation is performed on a monthly basis, meaning that new models are trained every month. The previous month is used as validation data, and all data before that as training data. We compare the retrained Transformer model to a static Transformer model that is only trained once per year. The retrained model outperforms the static model on both datasets, achieving a 14.3 % lower error on TransnetBW and a 3.8 % lower error on FeederBW. This indicates that retraining helps the model to adapt to concept drifts due to newly installed appliances or changed consumption patterns.

Context length Figures 7.7d-f show the forecast error of Chronos-2 with one week, two weeks, one month context and the full context length of 8192 time steps. The forecast accuracy of Chronos-2 improves with longer contexts on all three datasets. With only one week context, Chronos-2 barely outperforms the baseline, but it quickly improves with longer contexts. Using the maximum context of 8192 time steps gives only a small improvement over using one month context.

7.4 Discussion

Best forecasting model Transformer-based models are effective for load forecasting on all tested grid levels. On the three datasets, the improvement over the best non-Transformer model is considerable, with 6.6–10.7 % lower errors for the best Transformer compared to the best other method. This includes datasets with different characteristics, such as highly aggregated TSO load and more diverse and volatile LV feeder loads and client loads, where time series from multiple hundred measurement points are available. After hyperparameter optimization, Transformer models are effective for all these data types.

Best Transformer architecture The standard Transformer outperforms the flexible YAformer after hyperparameter optimization on all three datasets. This shows that the architectural modifications introduced in YAformer – including patching, CNN layers, LSTM layers, max pooling and sparse attention – are not successful in reducing the forecast error. This finding is further strengthened by the fact that the standard Transformer also outperforms TFT, which is a Transformer with feature selection layers and LSTM layers. Although the standard Transformer is a subclass of YAformer, the larger hyperparameter search space of YAformer hinders the HPO in finding the optimal configuration. In fact, there is no guarantee that the Transformer configuration (or the configuration of any other model) after HPO is the global optimum, but we had to restrict the HPO to 100 evaluations per model for practical reasons.

Zero-shot Chronos-2 Chronos-2 delivers accurate zero-shot forecasts for diverse time series, but it has particular weaknesses with rare events and long horizons, which lead to larger overall errors on the TSO data. While Chronos-2 is very accurate for short forecast horizons between a few hours up to one day, it is outperformed by models trained particularly for the datasets at hand for longer horizons. On the

TransnetBW data, it exhibits large errors around holidays, where it needs to incorporate not only the lower industrial activity on the holiday, but also indirect effects such as long weekends and school holidays. Future work may improve the results from Chronos-2 by designing additional features or by fine-tuning the model.

Training and inference tradeoff The advantage of Chronos-2 is that no dataset-specific training is required. Training a model from scratch takes between half an hour and five hours, depending on the dataset and model. This computational cost is saved with Chronos-2. On the other hand, Chronos-2 is a very large model, resulting in slower inference times. Inference times for our models are below 100 ms for creating one forecast for all time series in a dataset, whereas Chronos-2 needs up to 21 seconds. Therefore, Chronos-2 is useful when inference time plays a minor role, e.g. because few time series need to be forecasted or forecasts are only made once per day. For continuous forecasts during the day and for scenarios with many time series, dataset-specific models can be advantageous.

Learnings from ablated models Long contexts, covariates and continuous retraining are important to achieve the lowest electrical load forecast errors. This means that univariate TSFMs will likely not succeed in achieving the lowest possible forecast errors. Comparisons on datasets without covariates, like the original UCI Electrical Load Diagrams, can indicate which models capture autocorrelated energy demand patterns, but are to be taken with caution because they do not show the full picture.

Potential for probabilistic forecasts Deterministic forecasts do not cover the uncertainty of a forecast. This is especially relevant for volatile time series like FeederBW and Electricity-287. On both datasets, it is difficult to make good point forecasts due to the volatile and unpredictable pattern of the load, so that even the best models sometimes exhibit large errors. In such cases, it can be beneficial to create probabilistic forecasts that cover the uncertainty in form of predicted distributions. Future work can extend our benchmarking to probabilistic forecasting models. Chronos-2 offers the capability to predict quantiles. The deep learning models from our work can be extended to probabilistic models by replacing the linear output layer with a probabilistic output layer and training the models with the pinball loss function.

Open data We agree with Haben et al. [100] and Hewamalage et al. [116] that benchmarking of electrical load forecasting methods on different public datasets, with comparable test splits and metrics, is necessary to establish the state of the art in electrical load forecasting. Therefore, we provide a comparison of a large set of methods on three datasets. We share our preprocessed datasets publicly, together with the predictions of all evaluated models, to allow for future comparisons to our results².

Experimental setup limitations The achieved forecast errors might be unrealistically low for real applications. For example, real-time measurements of past load values might become available only after

² <https://github.com/KIT-IAI/load-forecasting-benchmark>

a few hours or days, but are assumed to be available immediately in our experiments. In addition, we have used weather data that is not available at the forecast time in practice, such as reanalysis data or short-term weather forecasts. However, our primary goal is to compare models under fair conditions, which we achieve by using the same data for all.

7.5 Conclusion

This chapter presents a benchmark for electrical load forecasting across grid levels, consisting of three datasets that represent the TSO level, the LV grid feeder level and the client level. In a comparison of time-series Transformers and the TSFM Chronos-2 with established machine learning and deep learning methods for electrical load forecasting on the three datasets, Transformer-based models outperform the established methods across all grid levels, reducing the MAE (normalized for the Electricity-287 dataset) by 6.6–10.7% compared to the next-best method. This answers RQ 2 by showing that an optimized Transformer architecture outperforms a variety of established load forecasting methods across grid levels.

The standard time-series Transformer outperforms YAformer, a flexible Transformer architecture with patching, CNN layers, LSTM layers, max pooling, and sparse attention. This indicates that the standard architecture is already a strong model, and architectural modifications are not needed to outperform established methods. The modifications can even lead to worse accuracy as they enlarge the search space for the hyperparameter optimization (HPO). However, more architectures than the ones tested in this chapter are possible, which can potentially improve the results.

The TSFM Chronos-2 gives competitive forecasts in a zero-shot setting on the feeder and client levels. Chronos-2 is particularly accurate for short forecast horizons, but it is outperformed by specialized models on the TSO level and it is comparably slow in inference.

All our datasets and results are made publicly available to enable benchmarking of new approaches against our results, as well as to allow for an extension to probabilistic models in the future. We plan to integrate the best approaches into the Energy-Arena³ [149] for continuous performance comparisons under realistic operational conditions. Ensembling of forecast models has the potential to improve the results beyond the single-model forecasts evaluated in this chapter.

³ <https://energy-arena.org>, accessed 2026-05-18.

Part IV

Explainability of Time-Series Transformers

8 Interpretable Forecasting Based on Attention Maps

Machine learning models are often considered black boxes. Given an input, they generate an output, but the large number of internal parameters prohibits human understanding of how the output is generated. Interpretable forecast models are desirable, where such human understanding is possible. Such interpretations can inform the user on which models to trust and when to trust them, and they can reveal spurious patterns in the model, and thereby help developers with debugging and improving models.

With Transformers relying on the attention mechanism, an interpretation of the model can be generated without extra computational effort, by visualizing the attention weights of the model for a given input. This is a common approach in NLP [53] and computer vision [69]. For language models, it is shown that individual heads have distinct functions, so their attention patterns can be interpreted as, e.g., coreference resolution (attending previous mentions of the same entity), verb-object relations, or fixed positional offsets [53]. In computer vision, attention maps are used to highlight regions in an image that are important for its classification [69]. In time-series analysis, multiple publications have shown that the learned attention maps are sparse [165, 300, 301]. For example, TFT [168] learns to attend time steps with certain positional offsets, such as daily or weekly cycles. This observation motivates the use of attention mechanisms that are sparse by design [165, 300, 301]. When the time-series patterns change, the attention weights also change, which can be used to identify regimes and special events [168]. Wang et al. [279] use attention to pass information between load time series of multiple buildings and find that some buildings are attended more than others. Gao et al. [85] visualize attention maps of individual attention heads and find that certain times of the day are attended more, but leave the interpretation of other emerging patterns to future work.

To the best of our knowledge, there is no publication yet that analyzes how attention heads react to covariates. This chapter provides such an analysis and therefore aims to explain the internal functions of individual attention heads. Thus, this chapter addresses RQ 3 on interpretable Transformer forecasting models:

Research question (RQ) 3

How can the attention mechanism be leveraged for efficient interpretable and explainable electrical load forecasting?

The next chapter, Chapter 9, introduces SHAPformer, a more sophisticated method that leverages the attention mechanism to compute SHAP values efficiently. Afterwards, the SHAPformer algorithm is extended to explain the TSFM Chronos-2 in Chapter 10.

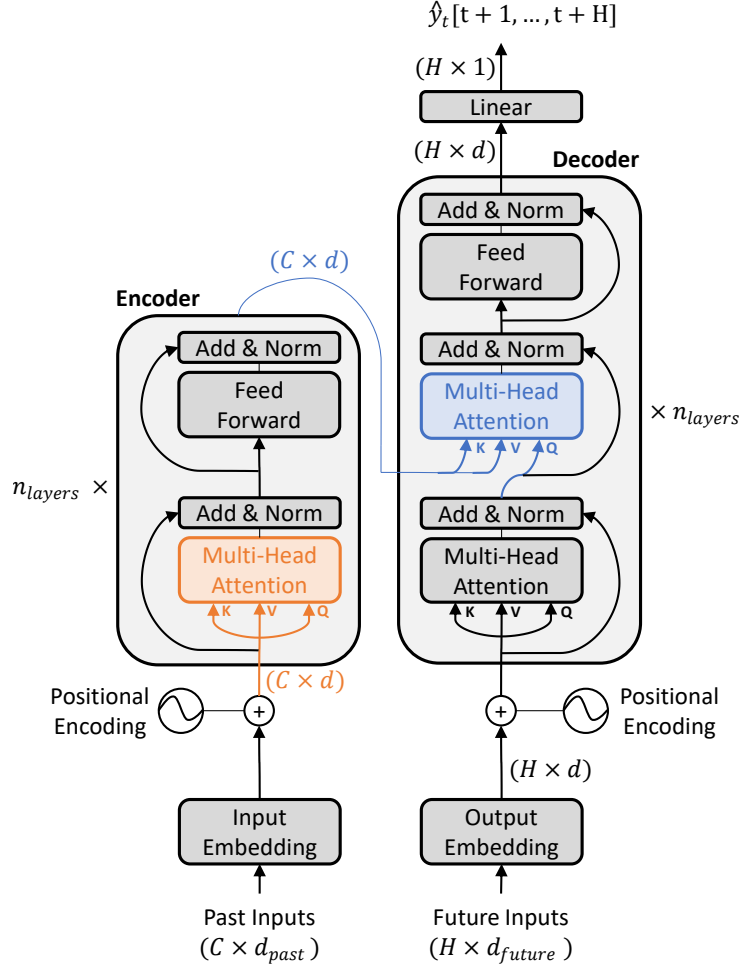


Figure 8.1: Transformer architecture with the encoder self-attention (orange) and cross-attention (blue) highlighted.

8.1 Methodology

This section describes how attention weights of the Transformer forecasting model are analyzed to obtain interpretable insights into its internal functioning. We focus on two attention mechanisms: encoder self-attention, where attention is applied within past time steps, and decoder cross-attention, where the decoder attends the encoder's representations of past time steps to retrieve information when generating forecasts. Attention weights are studied both as local explanations, i.e., attention maps for individual forecast instances, and as global explanations, obtained by averaging attention patterns across the entire test set. We further distinguish between analyzes of individual attention heads and aggregated analyzes in which attention weights are averaged across heads. Attention is examined from two complementary positional perspectives: with respect to positions in the input, and with respect to the hour of the week. These visualizations are complemented by quantitative analyzes, including correlations between attention patterns of different heads and correlations between attention weights and input feature values. Together, these analyzes characterize how input information is used within the model.

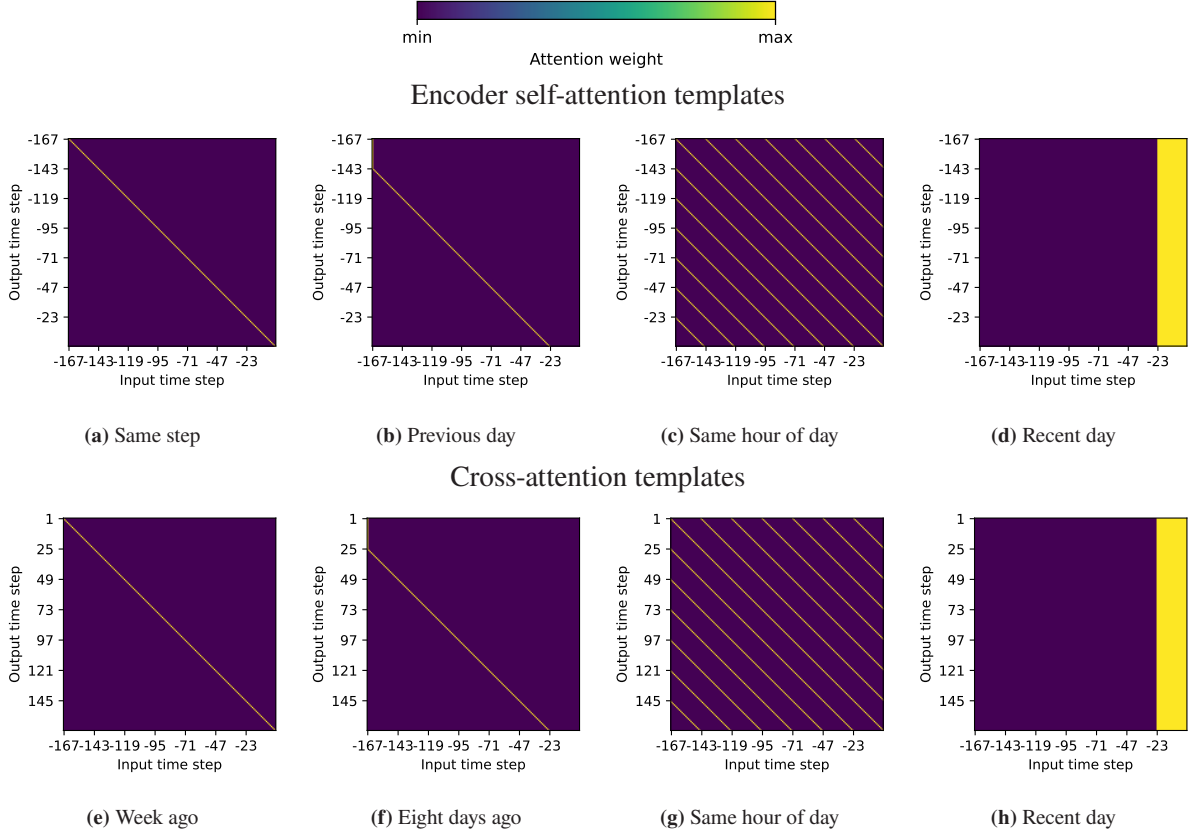


Figure 8.2: Attention templates. (a) Self-attention attending the embedded time step. (b) Self-attention attending the time step 24 hours ago. (c) Self-attention attending all time steps with the same hour of day. (d) Self-attention attending the most recent day. (e) Cross-attention attending the time step one week before the embedded time step. (f) Cross-attention attending the time step one week plus one day ago. (g) Cross-attention attending all past time steps with the same hour of day. (h) Cross-attention attending the day before the forecast origin.

Self-attention vs. cross-attention We analyze two attention mechanisms in the encoder-decoder Transformer: encoder self-attention and decoder cross-attention (see Figure 8.1). Encoder self-attention operates only on the historical input window and models dependencies among past time steps. Each past time step can attend to all other past time steps, allowing the model to form context-dependent representations of the history. In contrast, cross-attention in the decoder links the future time steps to the embedded past time steps. For each future time step, the decoder assigns attention weights to the past time steps represented in the encoder output, thereby selecting which parts of the history are most relevant for generating a particular prediction. While encoder self-attention reveals how the model organizes and compresses information within the past, cross-attention more directly reflects how past information is retrieved for forecasting.

Local vs. global explanations Attention weights are interpreted at two levels: local and global. Local explanations are attention maps associated with a single forecast instance. They show how the model distributes attention for one specific historical context and forecast situation. Such maps are useful for inspecting model behavior in individual cases, for example during unusual load patterns or special events. Global explanations, on the other hand, are obtained by aggregating attention weights across all forecast instances in the test set. By averaging attention maps over many inputs, systematic patterns become

visible. These global patterns characterize typical temporal dependencies learned by the model rather than behavior tied to a particular example.

Average vs. individual attention Finally, attention is analyzed either per head or after aggregation across heads. Individual attention heads can learn specialized patterns, such as focusing on specific temporal distances or covariate values. Studying heads separately therefore allows us to identify diverse strategies within the model and to assess whether different heads capture complementary types of temporal relationships. In contrast, averaging attention weights across all heads yields a more compact representation of overall model behavior. This aggregated view highlights dominant trends but may obscure head-specific specializations. Considering both perspectives enables us to combine detailed analysis with a high-level understanding of how attention is distributed.

Positional vs. calendrical analysis Attention maps are visualized from two complementary positional viewpoints. First, attention is analyzed with respect to positions in the model input. The encoder receives the last week of history, indexed from time step -167 to 0, and the decoder produces forecasts for the following week, indexed from 1 to 168. Attention maps in this representation therefore show how specific past time steps relate to other past steps (encoder self-attention) or to individual forecast steps (cross-attention). Synthetic examples illustrating the structure and interpretation of such maps are shown in Figure 8.2. Second, attention weights are aggregated according to the hour of the week, where indices 0 to 167 correspond to Monday 00:00 to Sunday 23:00. This representation abstracts from the position within a particular input window and instead highlights recurring weekly patterns, such as systematic attention to certain hours or days. Together, these two views allow us to distinguish position-specific effects from temporally recurring structures learned by the model.

Correlation between heads To quantify similarity between attention heads, we compute the correlation of their attention weights across the entire test set. For a given pair of attention heads h_1 and h_2 , we first collect their attention weights over all n instances. This results in n matrices of size $C \times C$ for both heads (more precisely, the matrices have size $C \times C$ for encoder self-attention and size $H \times C$ for cross-attention, but we set $C = H$). For each head, we flatten the matrices and concatenate the resulting vectors into one vector of length $n \cdot C^2$. We denote these vectors as $\mathbf{a}^{(h_1)}$ for head h_1 and $\mathbf{a}^{(h_2)}$ for head h_2 and calculate the similarity using the Pearson correlation coefficient:

$$r_{h_1, h_2} = \frac{\sum_{i=1}^{n \cdot C^2} \left(a_i^{(h_1)} - \bar{a}^{(h_1)} \right) \left(a_i^{(h_2)} - \bar{a}^{(h_2)} \right)}{\sqrt{\sum_{i=1}^{n \cdot C^2} \left(a_i^{(h_1)} - \bar{a}^{(h_1)} \right)^2} \sqrt{\sum_{i=1}^{n \cdot C^2} \left(a_i^{(h_2)} - \bar{a}^{(h_2)} \right)^2}}, \quad (8.1)$$

where $n \cdot C^2$ is the number of collected attention values and $\bar{a}^{(h)}$ denotes the mean attention weight of head h . High correlation indicates that two heads distribute attention in a similar manner across time, whereas low or negative correlation indicates that they capture different dependency patterns.

Correlation between heads and features To relate attention to input information, we compute correlations between attention weights and feature values. For each test instance i , we form vectors $\mathbf{x}_{f,i}$ containing the C past feature values and $\mathbf{a}_i^{(h)}$ containing the mean attention of head h to the C past time steps. The vectors of all n test instances get concatenated to vectors $\mathbf{a}^{(h)}$ and $\mathbf{x}_f$, each containing $n \cdot C$ values. The correlation between head h and the feature is computed as

$$r_{h,f} = \frac{\sum_{j=1}^{n \cdot C} (a_j^{(h)} - \bar{a}^{(h)}) (x_{f,j} - \bar{x}_f)}{\sqrt{\sum_{j=1}^{n \cdot C} (a_j^{(h)} - \bar{a}^{(h)})^2} \sqrt{\sum_{j=1}^{n \cdot C} (x_{f,j} - \bar{x}_f)^2}}, \quad (8.2)$$

where $\bar{a}^{(h)}$ denotes the mean attention weight and $\bar{x}_f$ the mean feature value. This measure captures whether higher attention weights are systematically associated with higher or lower values of a given feature, thereby indicating which types of input signals a head tends to focus on.

8.2 Experimental Setup

Dataset The dataset used in the experiments is the TransnetBW dataset presented in Chapter 4. Calendar features (hour of day, day of week, month, holiday) and weather features (temperature, precipitation) from the Copernicus reanalysis model [57] are used as exogenous features. In the absence of historical weather forecasts, the measured weather data is used as model input. The years 2015-2018 are used as training data, the first half of 2019 as validation data, and the second half of 2019 as test data. Explanations are generated for 1000 random samples from the validation data and the test data to represent all seasons.

Model We create an encoder-decoder Transformer model with $n_{layers} = 2$, $n_{heads} = 4$, $d_{model} = 128$, context length $C = 168$ hours and forecast horizon $H = 168$ hours. The model is trained with Adam [148] with a batch size of 128 and a learning rate of 0.0001 for a maximum of 100 epochs, with early stopping after 10 consecutive epochs without improvement on the validation data.

8.3 Results

This section presents the attention maps of the trained Transformer model. We generate global explanations by averaging the attention maps of 1000 samples. We visualize the self-attention and cross-attention combined for all heads, as well as separately for each head, and analyze the correlation between heads as well as their dependence on feature values.

8.3.1 Attention maps

Global explanation averaged over all heads

Figure 8.3 shows the self-attention averaged across all encoder heads (left), and the cross-attention averaged across all decoder heads (right), once per position relative to the last known time step (top) and once per hour of the week (bottom). Both the diagonal effect and the recency effect of the templates in Figure 8.2 are visible in the self-attention and cross-attention visualizations. For the self-attention by position (Figure 8.3a), a diagonal around pixels (i, i) is visible, indicating that the model attends the time step it encodes and few time steps around it. In addition, time steps that are multiples of 24 hours into the future are also attended. For the cross-attention by position (Figure 8.3b), similar but wider diagonals are visible. In both plots, it can be observed that the most recent time steps are attended more than time steps further in the past. For the self-attention by hour of week (Figure 8.3c), seven vertical block structures are visible, indicating that the model attends the time steps during the day more than time steps during the night. Sundays are attended more (vertical block to the right), especially for the computation of embeddings belonging to the Sunday. For the cross-attention by hour of week (Figure 8.3d), a grid structure is visible, indicating that nighttimes are always attended, whereas daytimes are only attended for forecasts during the day. The Sunday is attended more strongly than other days.

Global explanations of attention heads

Figures 8.4 and 8.5 show the attention maps of the individual heads, once per position in the input and once per hour of week. In the following, attention heads are named as *part.layer.head*, with *part* $\in \{\text{enc}, \text{dec}\}$ (encoder or decoder), *layer* $\in \{1, 2\}$ (first or second layer), and *head* $\in \{1, 2, 3, 4\}$ (four heads per layer). For the attention heads by position (Figure 8.4), diagonal structures are visible for many heads, indicating that the head looks at time steps with fixed offsets relative to the time step it embeds. Some heads are complementary to each other, e.g. *enc.1.1* attends the last day most strongly, whereas *enc.1.2* attends the last day the least. For the attention heads by hour of week (Figure 8.5), block structures are visible for all heads. Some heads attend only few time points, e.g. *enc.1.1* and *enc.1.4*, while others attend more broadly, e.g. *enc.2.1*. In addition to the block structure, a diagonal pattern is visible for head *dec.2.3*.

Figure 8.6 shows the mean attention based on the time lag between the key vector and the query vector. A positive time lag means that the model uses future information to embed a time step, whereas a negative value means that past information is used. Note that for cross-attention from the decoder to the encoder visualized in the third and fourth row, only past information can be used. All heads except for *dec.2.4* exhibit daily patterns. They attend time steps that are multiples of 24 hours apart from the embedded time step, resulting in 14 peaks in the attention graph. Head *enc.1.1* attends future values more than past values. Three encoder heads (*enc.1.4*, *enc.2.2*, *enc.2.4*) attend the embedded time step (lag 0) and the time steps around it more, whereas head *enc.1.2* attends these periods less. Heads *dec.1.2* and *dec.2.2* attend time steps one week ago (lag -168) more. Six decoder heads (*dec.1.1*, *dec.1.2*, *dec.1.4*, *dec.2.2*, *dec.2.3*, *dec.2.4*) attend recent values more than older values.

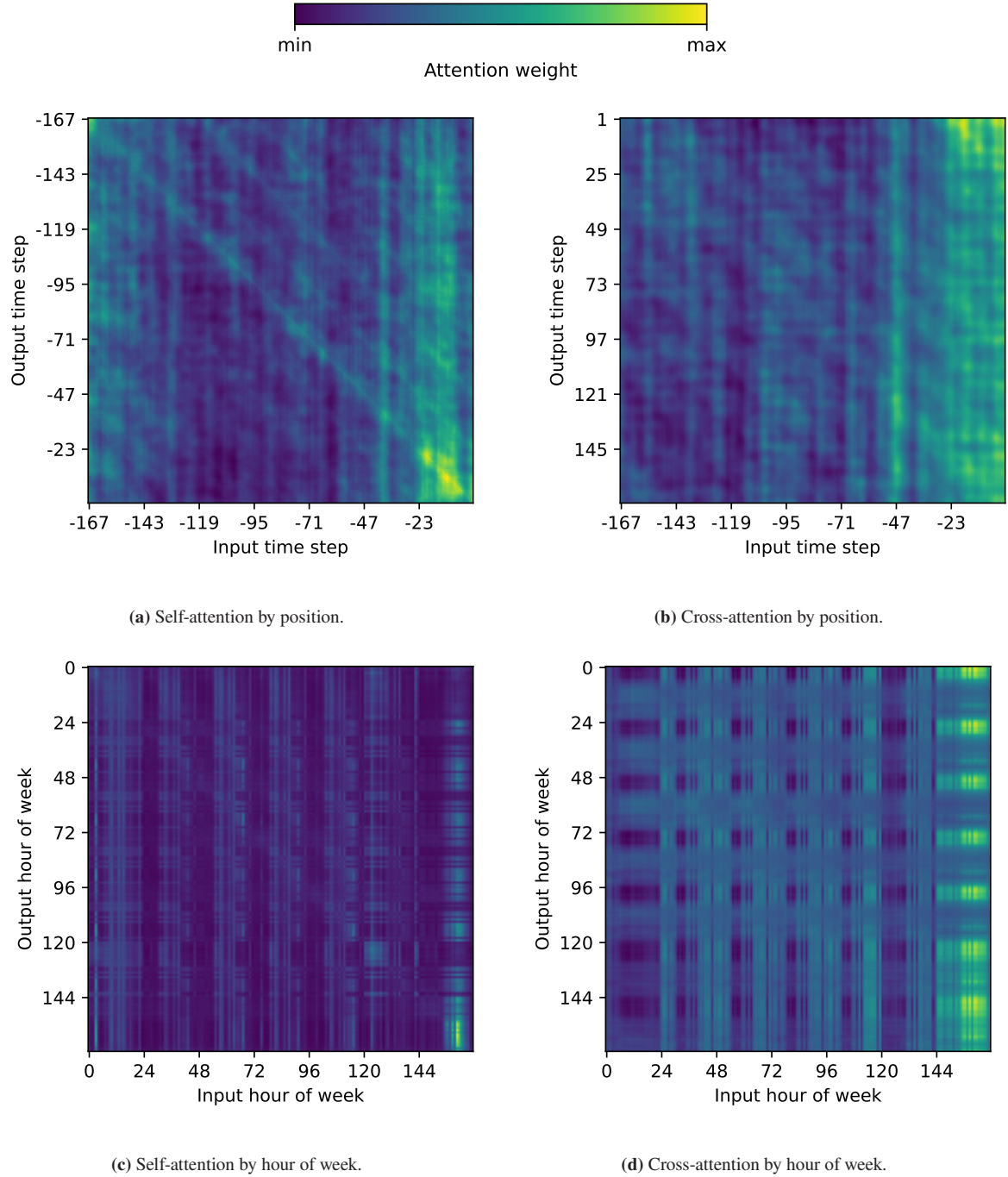


Figure 8.3: Attention maps of the self-attention and cross-attention averaged across all heads and 1000 samples of the TransnetBW dataset.

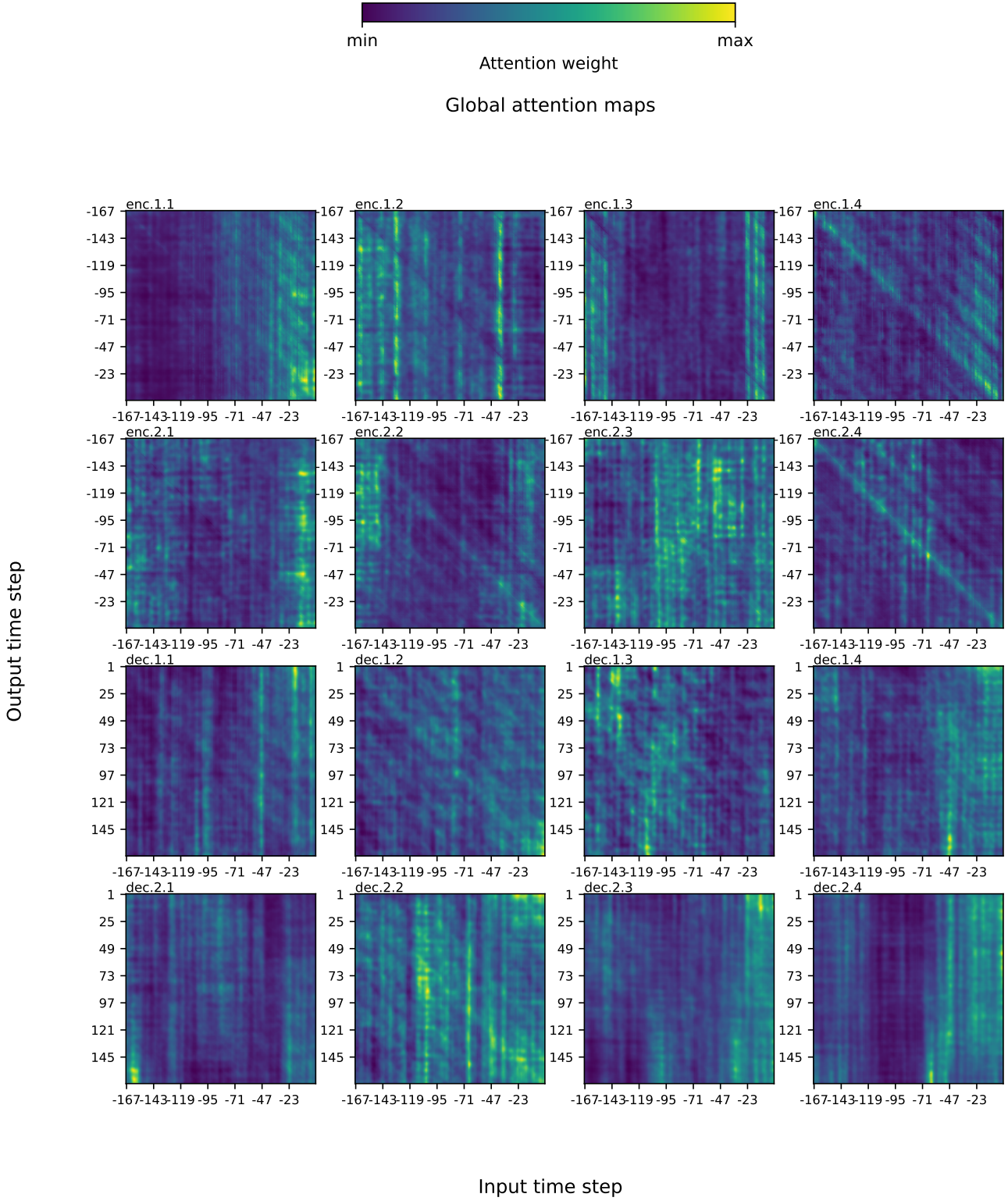


Figure 8.4: Attention maps of individual heads visualized by input position, analyzed on the TransnetBW dataset. Each row displays the heads of one Transformer layer.

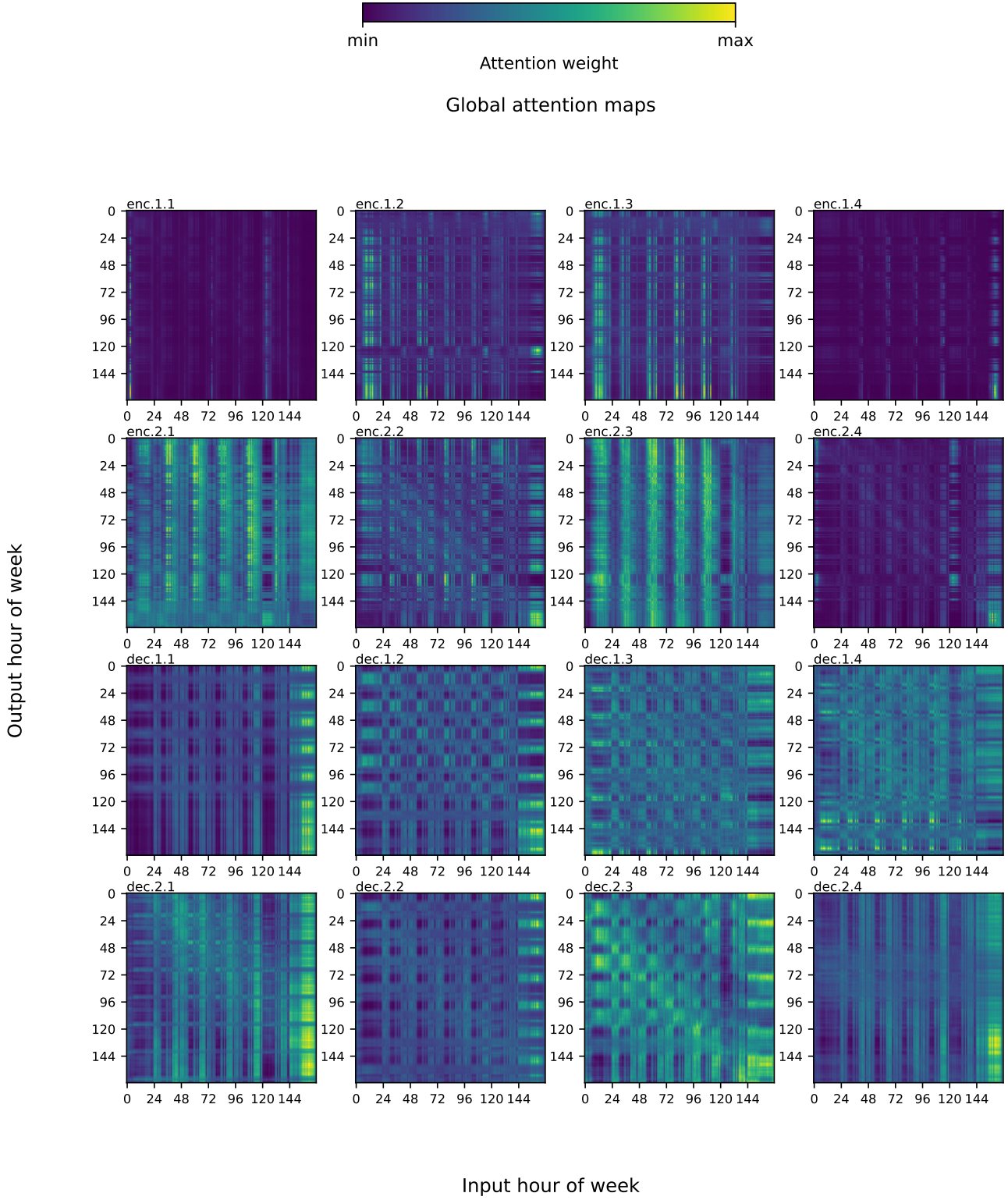


Figure 8.5: Attention maps of individual heads visualized by hour of week, analyzed on the TransnetBW dataset. Each row displays the heads of one Transformer layer.

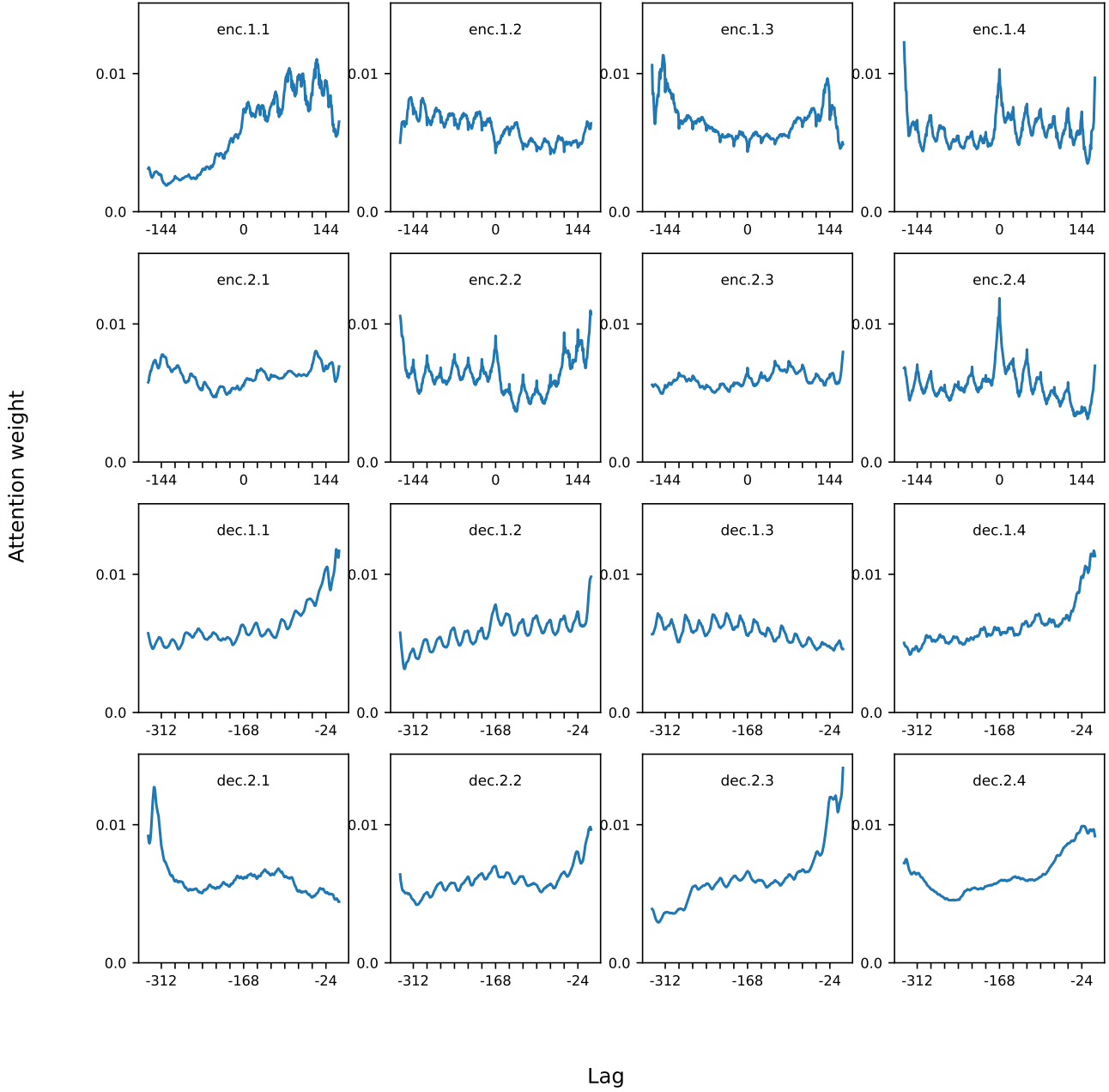


Figure 8.6: Attention weights of each head, displayed by lag (time difference between key vector and query vector) for the TransnetBW dataset. Each row displays the heads of one Transformer layer.

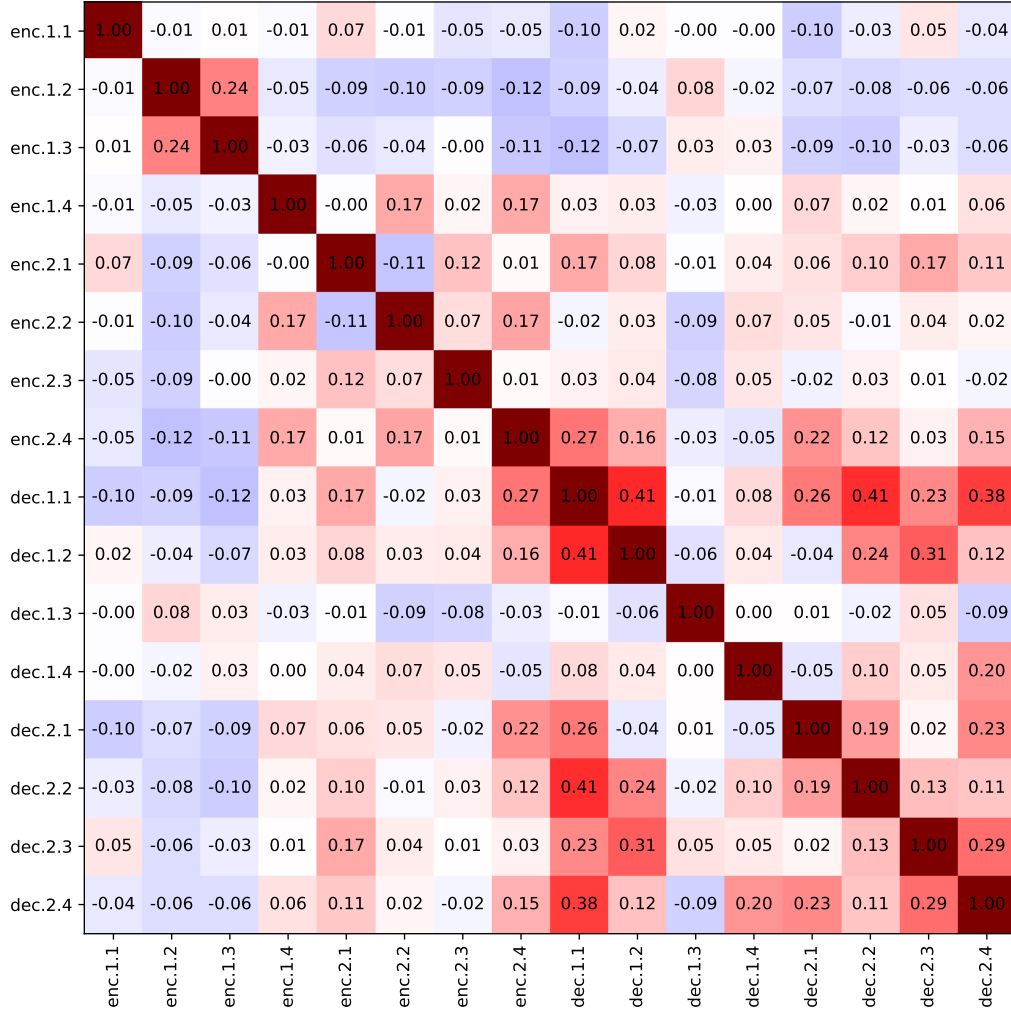


Figure 8.7: Correlation of attention heads of a Transformer with two encoder layers, two decoder layers and four heads per layer, calculated on the TransnetBW dataset.

Correlation between heads

In order to analyze whether different heads behave similarly – which would indicate that they are redundant and the number of heads in the model could potentially be reduced – we compute the correlation of the attention weights of the individual heads. Figure 8.7 shows the correlation between the attention heads evaluated on the 1000 randomly selected samples. There are heads that show a moderate correlation ($r > 0.3$), with the strongest correlation being 0.41 for *dec.1.1* with *dec.1.2* and *dec.2.2*. Only four pairs of heads exhibit moderate correlations ($r > 0.3$) and 25 pairs of heads exhibit weak correlations ($0.3 > r > 0.1$).

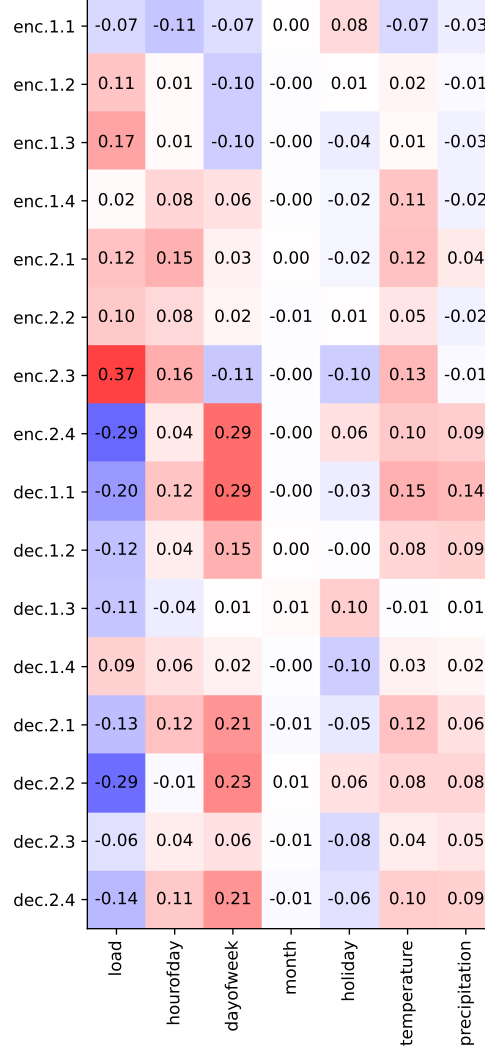


Figure 8.8: Correlation of the attention heads with features of the TransnetBW dataset.

8.3.2 Feature dependence of attention heads

Next, we analyze the dependence of the attention weights of individual heads on the input features. Figure 8.8 shows the correlation of the attention of each head with the features. All heads except *dec.2.3* are weakly correlated ($|r| > 0.1$) with at least one feature. Head *dec.2.3* is the only head that shows a moderate correlation ($(|r| > 0.3)$) with a feature. Head *dec.2.3*, which shows a strong recency effect (see Figure 8.6), is not correlated with the features.

Figure 8.9 gives an explanation for each head based on one feature. The feature with the strongest correlation is shown in most cases. However, sometimes a different feature is selected when a dependence of the attention head on that feature is visible in the dependence plot but not in the correlation values, which only show linear correlation. In the plots in Figure 8.9, a line indicates the mean attention weight

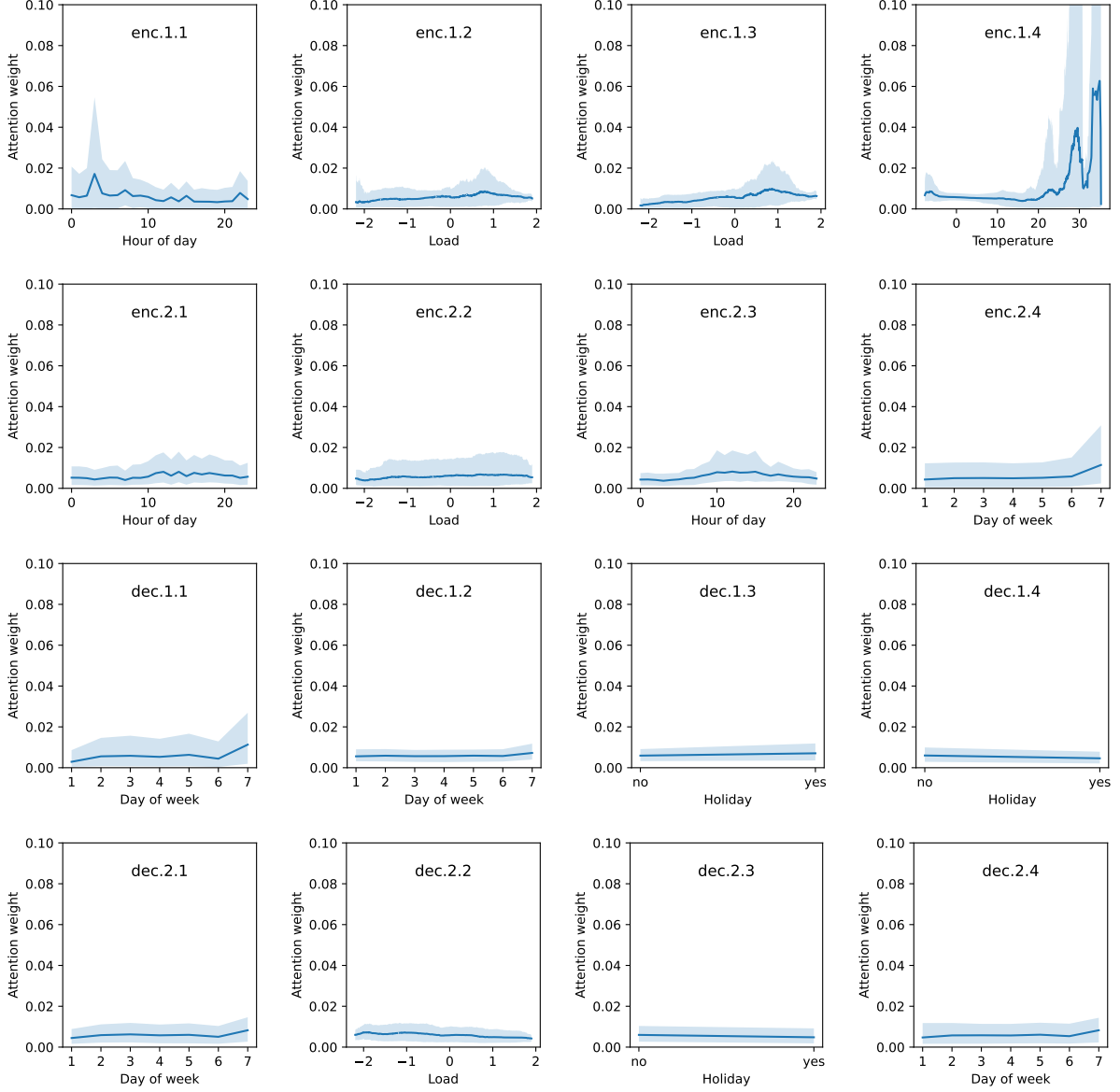


Figure 8.9: Dependence of the attention of the individual heads on the TransnetBW dataset. Each row shows the heads of one Transformer layer. The shaded area is the 90% percentile over 1000 samples of the test set.

for a rolling window (with width ± 1 for temperature and precipitation and ± 0.1 for the standardized load), and shaded areas represent the 90% percentiles of the same windows, calculated over the 1000 random samples from the TransnetBW test set.

Head *enc.1.1* attends the night more, whereas heads *enc.2.1* and *enc.2.3* attend the day more. Heads *enc.1.2*, *enc.1.3* and *enc.2.2* attend times with higher loads more, whereas head *dec.2.2* attends times with lower loads more. Head *enc.1.4* reacts to high temperatures. Heads *enc.2.4*, *dec.1.1*, *dec.1.2*, *dec.2.1* and *dec.2.4* attend Sundays more strongly than other days. Head *dec.1.3* attends holidays more strongly, whereas heads *dec.1.4* and *dec.2.3* attend non-holidays more strongly – however, the effect is weak in both directions.

8.4 Discussion

Attention heads have distinct functions. Notably, the analysis of individual attention heads reveals that the heads have distinct functions. The heads divide the task of retrieving relevant information among each other, as can be seen by the mostly weak correlation of individual heads' attention weights and the fact that dependencies on exogenous features vary greatly between heads. Some functions of heads can be interpreted – such as heads focusing on certain time lags, on the weekend, or on extreme values of past load or exogenous features. Such interpretations have the potential to improve the user's trust in the model, especially when the found interpretations are in line with expert domain knowledge, such as knowledge about daily and weekly repeating patterns, special weekend patterns or influences of exogenous features on the electrical load. For some heads this interpretation is complicated by the fact that multiple explanations are plausible – such as *enc.2.3* whose attention weights correlate negatively with past loads and positively with the day of the week. In such cases, it is not clear whether the head has multiple functions, or the multiple patterns are visible because the features are correlated. In addition, a meaningful interpretation based on feature values cannot be found for all heads, so that the model cannot be fully interpreted. Some heads behave similarly – indicated by the moderate positive correlation between their attention weights. This might indicate that these heads are redundant and a model with less attention heads could perform equally well. Indeed, the hyperparameter optimization in Chapter 7 showed no great influence of the number of heads on the model performance.

Single-layer attention maps do not show the full picture. Using the attention weights of a head as a proxy for the importance of the input features of the time step to the model prediction might be wrong. There is no guarantee that a larger attention weight signifies larger importance. In fact, the values of a vector that receives a low attention weight can result in a large influence on the model prediction due to the model's linear layers and nonlinearities. For all attention heads that come after the first encoder layer, attending a time step t does not necessarily mean that the input features of time step t are important, as the context-dependent embedding of time step t can contain information about the whole encoder input. A method called Attention Rollout [1] attempts to solve that problem by tracing the attention weights over all layers of the model, and could be tested for Transformer-based load forecasting models in the future. However, it is shown for language models [66] and translation models [24] that the attention weights are often meaningful when interpreting them based on the input words, which supports the idea of interpreting the forecasting model's attention weights based on the input features.

Attention does not show feature importance nor influence on the prediction. The attention maps only show which time steps are important for the model prediction, but do not reveal why these time steps are important or which features lead to that importance, nor can the attention maps be used to explain how the features of a time step, individually or jointly, affect the model prediction.

8.5 Conclusion

This chapter presents a first approach to RQ 3 by interpreting a time-series Transformer based on the attention weights. The analysis shows that individual attention heads have distinct functions, and some of the functions are interpretable, such as heads following 24-hours lags, attending holidays and weekend days, and reacting to extreme temperatures. The fact that these functions align with our domain knowledge about important features strengthens our trust into the trained model. However, the model interpretation based on the attention weights is limited in multiple respects. (1) A meaningful interpretation could not be found for all heads, while for others there are multiple possible interpretations. (2) The attention weights only reveal which time step embeddings are used by the model, but give no information about which features of these time steps are important, nor how the time steps (or more precisely, the features) influence the model prediction. Therefore, a more informative model interpretation based on attention manipulation and SHAP values is introduced in the next chapter.

9 SHAPformer: Sampling-Free Explanations by Attention Manipulation

Previous publication

Sections 9.1–9.5 are modified excerpts from:

[115] M. Hertel, S. Pütz, R. Mikut, V. Hagenmeyer, and B. Schäfer. “Explainable time-series forecasting with sampling-free SHAP for Transformers”. *Nature Communications* (2026), DOI: 10.1038/s41467-026-73243-5.^a

^a Licensed under a Creative Commons Attribution 4.0 International License, <https://creativecommons.org/licenses/by/4.0/>.

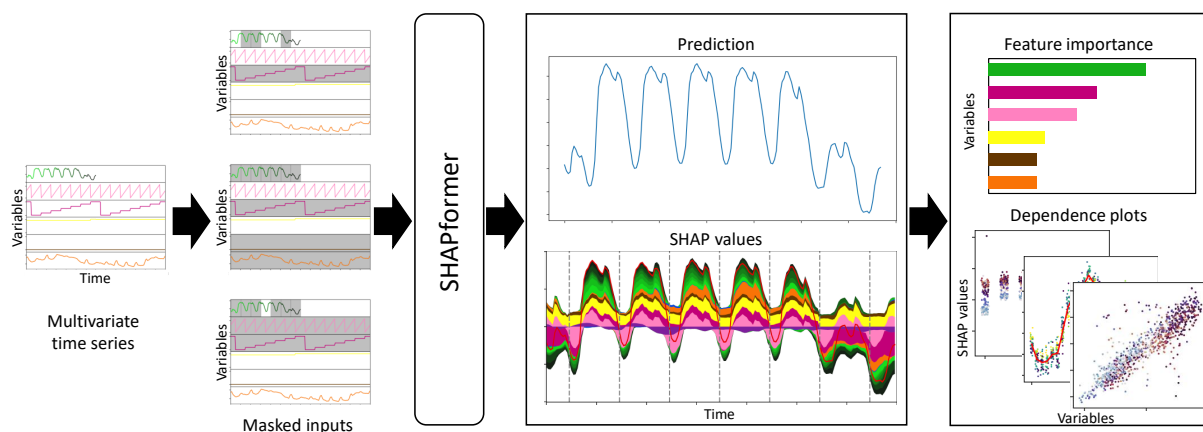


Figure 9.1: Overview of the proposed method. The model receives the past target and covariate time-series as input. The features are grouped as indicated by colors. SHAPformer makes predictions based on masked inputs. SHAP values are derived from the marginal feature contributions, defined as the difference of the prediction with and without a feature group. Multiple local explanations of forecasts get combined into global feature importance values and feature dependence plots. Figure from [115].

The previous chapter showed that explanations based on attention weights have limited capabilities. Most prominently, they are able to highlight important input time steps, but are not informative about how these time steps or individual covariates from these time steps, affect the forecast.

SHAP is a more informative explanation method, which allows to calculate feature importance, feature dependence plots, and local explanations of individual forecasts. However, existing SHAP algorithms are computationally expensive, because they require many model evaluations on sampled data. This is particularly problematic for complex models, such as Transformers, and many features, as often encountered in time-series forecasting.

In this chapter, we present a new algorithm for estimating SHAP values for Transformer-based time-series forecasting models, that is based on attention manipulation [64] and thereby not dependent on feature sampling. Our contributions are as follows:

1. We introduce SHAPformer, a Transformer-based forecasting model that enables efficient, sampling-free SHAP value estimation by grouping features and applying attention manipulation [64] to exclude feature groups during training and inference. This approach eliminates the need for sampling or setting baseline values. An overview of the method is shown in Figure 9.1.
2. We validate SHAPformer’s explanations on a synthetic dataset with known ground-truth explanations, demonstrating its ability to accurately capture the patterns underlying the data.
3. We show the benefits of SHAPformer applied to real-world electrical load data from the TSO TransnetBW. We find that SHAPformer is faster than alternative SHAP algorithms and it provides meaningful local and global insights.

9.1 Methodology

This section introduces two algorithms to estimate SHAP values for time series: WindowSHAP [196] and SHAPformer, a Transformer-based approach that leverages attention manipulation [64] to estimate SHAP values. Both algorithms build on Shapley values and SHAP, which are introduced in Section 3.3.

9.1.1 WindowSHAP

WindowSHAP [196] calculates SHAP values on the basis of groups of features instead of individual features, in order to reduce the number of model evaluations. Our instance of WindowSHAP uses seven groups for the past target, each consisting of 24 consecutive values for the past seven days, and one group for each exogenous variable. This makes 13 feature groups in total for the real-world data, and 14 feature groups in total for the synthetic data which exhibits one covariate more. As a result, the model is called $13 \times 10 \times 2 \times 100 = 26\,000$ times to create a local explanation of a forecast on the real-world data (and $14 \times 10 \times 2 \times 100 = 28\,000$ on the synthetic data), using 10 permutations and 100 background samples.

The features belonging to a group are sampled together, resulting in more realistic samples than with PermutationSHAP, which samples each feature individually. For a load feature group, 24 consecutive values from the training data are sampled, so that these 24 values form a smooth and realistic curve in themselves (but not necessarily realistic in the context of the other load values and exogenous features). The month values are increased by a constant, randomly chosen offset between 0 and 11 months. In the same way, the day-of-week and hour-of-day values are offset by 0 to 6 days, respectively 0 to 23 hours. For the temperature feature, the entire temperature curve is replaced by 336 consecutive values from the training data.

9.1.2 SHAPformer

We introduce SHAPformer, an explainable time-series forecasting model based on SHAP and Transformers. SHAPformer inherits the predictive strength of time-series Transformers and leverages the attention mechanism to calculate SHAP explanations more efficiently than with sampling-based algorithms. SHAPformer uses the same feature groups as WindowSHAP to reduce the number of model evaluations needed to calculate SHAP values. In addition, it eliminates the need for sampling many alternative inputs, by using attention manipulation to make predictions based on feature subsets. This section first introduces attention manipulation and then demonstrates how it is used in SHAPformer to calculate SHAP values efficiently.

Attention manipulation

SHAPformer uses attention manipulation [64] to restrict the model's access to feature groups that are not contained in a given coalition. Recall from Chapter 3 that the attention mechanism calculates a weighted sum $\sum_i \alpha_i \cdot \mathbf{v}_i$ of value vectors $\mathbf{v}_i$, where the weight α_i is determined by the similarity of a query vector $\mathbf{q}$ to key vectors $\mathbf{k}_i$ (Equation 3.14). Attention manipulation allows dropping out individual vectors $\mathbf{v}_i$ by setting their attention score to $a_i = -\infty$, so that the attention weight α_i becomes zero after the softmax operation. Thereby, the weight of vector $\mathbf{v}_i$ is set to zero, meaning that no information from $\mathbf{v}_i$ is contained in the attention output. Consequently, the following layers of the model have no access to the dropped-out vector $\mathbf{v}_i$.

SHAPformer architecture

Figure 9.2 shows the SHAPformer architecture, which is explained next.

Model inputs The model has a lookback size of 168 and a forecast horizon of 168 (one week in hourly resolution). The 168 past feature vectors are given to the encoder. Each past vector contains the load feature and exogenous features of one past time step. The 168 future feature vectors are fed to the decoder. Each future vector contains the exogenous features of one future time step.

Masked feature attention In the first layer, SHAPformer uses a masked feature attention mechanism to compute time step embeddings, as shown in Figure 9.3. This mechanism computes an embedding of the feature values belonging to one time step. For each variable (v_1 to v_n in Figure 9.3), the model maintains a learned embedding. There are two types of embeddings, one for categorical variables and one for continuous variables. Both embeddings transform a single value (a category for categorical variables or a scalar for continuous variables) into a vector of size d_{model} . For categorical variables – such as hour of day, day of week, month and holiday – with k possible values, k randomly initialized embedding vectors of size d_{model} are used and are modified during model training. For continuous variables $v_i \in \mathbb{R}$ – such as past load and weather data – a trainable linear layer with d_{model} units is used to embed the feature values. Next, the attention mechanism is used to calculate an embedding for a time

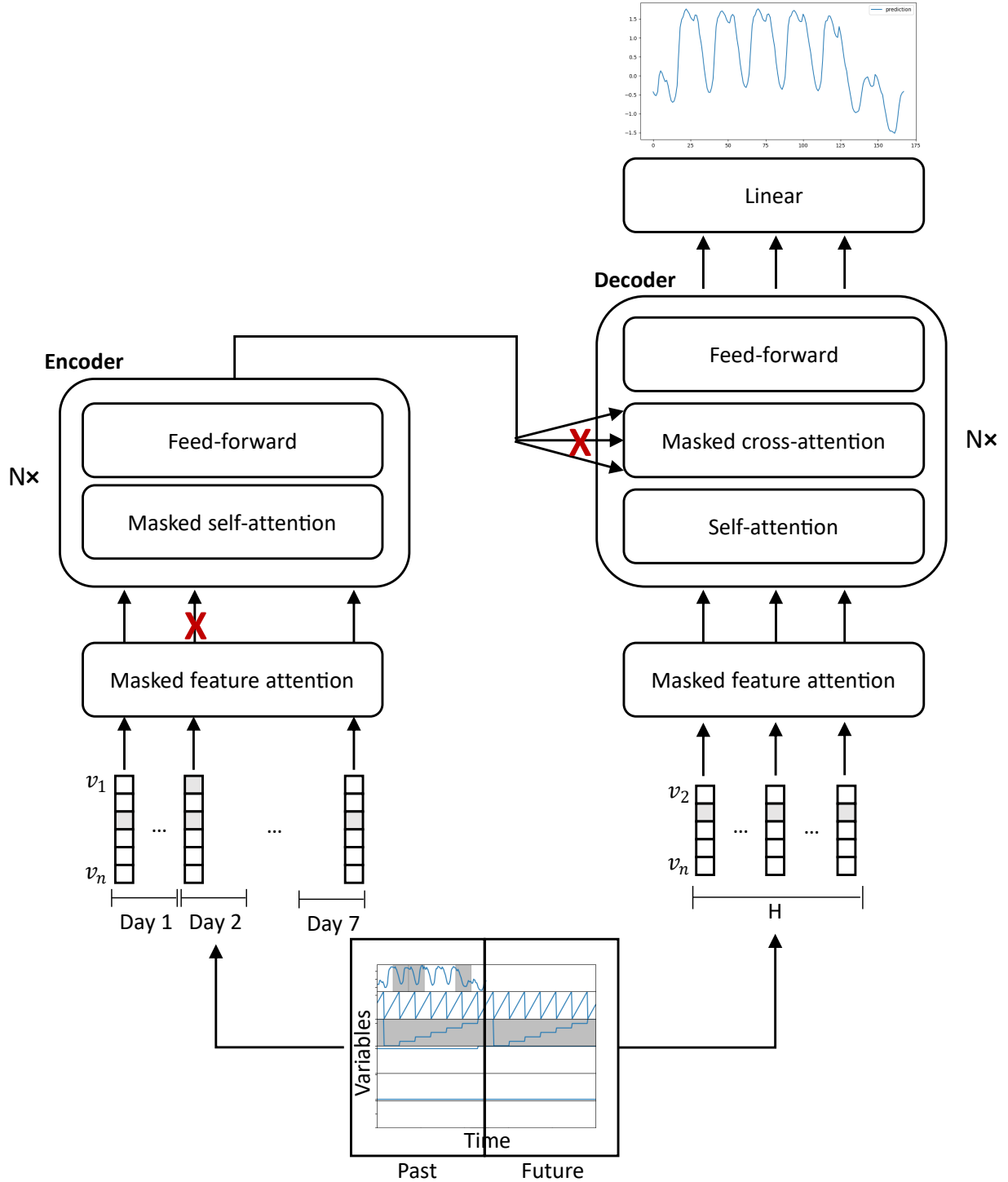


Figure 9.2: Overview of the SHAPformer architecture. N encoder and decoder blocks are stacked. Masked self-attention and masked cross-attention are used to drop out time steps (days 2, 3 and 6 in the example). Masked days and masked covariates are shaded. Attention weights of masked days are set to zero (red cross). Figure from [115].

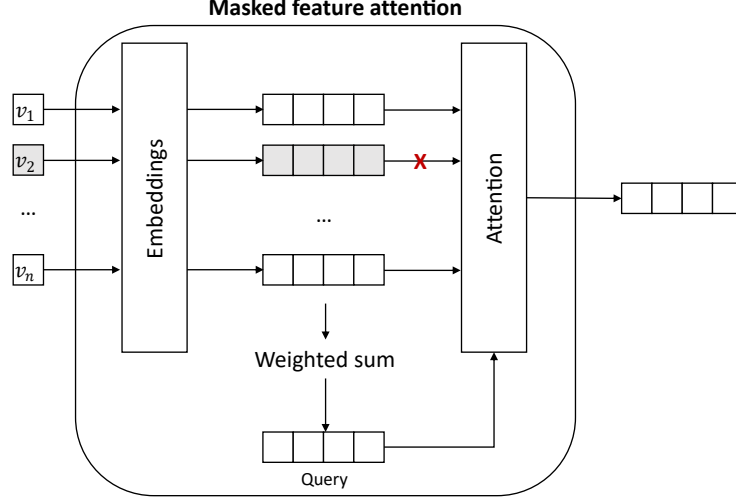


Figure 9.3: The masked feature attention mechanism. The embedding mechanism for one time step is shown. Out of the features v_1 to v_n , v_2 is masked. Each feature is embedded to a vector of dimension d_{model} . Feature v_2 is dropped out by setting its weight in the query calculation and the attention mechanism to zero (red cross). Figure from [115].

step, combining the feature embeddings for that time step. A query vector is computed as a weighted sum of the embedded variables, where dropped-out variables get zero weight and all others weight one. By setting the weight of dropped-out variables to zero, we ensure that they do not influence the attention mechanism. Then, attention is used to compute the time step embedding, based on the query vector and the embedded feature values as key vectors. In the masked feature attention mechanism, the attention weight of absent variables is set to zero, so that the model has no access to their embeddings (v_2 is masked in the example in Figure 9.3). The output of the masked feature attention mechanism is a vector of size d_{model} representing the available features for one time step. With this masking approach, SHAPformer can be run on subsets of the variables. After calculating the time step embeddings, a positional encoding is added to the embedding vectors to maintain their order.

Masking of time steps The past load is split into seven feature groups, each representing the 24 values of one day. To drop out load feature groups that are absent from a coalition, the model's access to the 24 time steps belonging to the feature group is restricted. This is achieved by using masked self-attention in the encoder and masked cross-attention in the decoder, as explained next.

Encoder The encoder gets 168 embeddings for the 168 past time steps as input. Multiple Transformer encoder layers are stacked, each consisting of self-attention and linear layers – both equipped with add and norm mechanisms (not shown in the figure). The self-attention is masked, i.e. the attention weight for absent time steps is set to zero, so that they are not used to calculate the attention output.

Decoder The decoder gets 168 vectors for the 168 future time steps as input. It consists of multiple Transformer decoder layers using self-attention and cross-attention to access the encoder output. The cross-attention is masked, i.e. the attention weight for absent time steps is set to zero, so that the decoder has no access to them (days 2, 3 and 6 in the example in Figure 9.2). Finally, a linear layer is used as

prediction head to transform the 168 output vectors of the decoder into 168 scalar values, which are the forecast for the next 168 time steps.

Masked training During model training, random masks are sampled for every example that is given to the model, so that each variable and each day has a 50 % chance of being masked. Thereby, the model learns to create robust forecasts using subsets of the features. Without masked training, the model could potentially behave arbitrarily when a feature is absent at inference time.

Owen values We calculate Owen values (Equation 3.19) in SHAPformer, based on a coalitional structure where the seven feature groups for the past load belong to one coalition, and each covariate is one coalition. Thereby, SHAPformer treats the past load equal to the covariates, and then distributes the past load's SHAP values among the seven input days. Note that when a load feature group is absent, the other features of the correspondent time steps are also absent, as an effect of the masked self-attention in the encoder and masked cross-attention in the decoder, which restrict the model's access to the masked time steps entirely. We assume that this has only a minor effect on the Owen values, because the past covariates are mostly used as context information for the past load, meaning that when the load of a day is not available to the model, the effect of covariates of the same day on the forecast is negligible.

9.2 Experimental Setup

We evaluate SHAPformer on the synthetic dataset and the TransnetBW dataset, both introduced in Chapter 4. The TransnetBW dataset is enriched with calendar features (hour of day, day of week, month and holiday) and weather data (temperature, precipitation) from the ERA5 reanalysis model [57]. The years 2015–2018 are used for training, the first half of 2019 for validation, and the second half of 2019 for testing. The forecasting models, XAI approaches and evaluation metrics used in the experiments are described next.

9.2.1 Forecasting models

The predictive performance of SHAPformer is compared to a persistence baseline, a linear regression, XGBoost [50], a standard time-series Transformer, and TFT [168]. All models have one week context length and forecast the next 168 hourly values.

Persistence baseline The persistence baseline predicts the value from one week before the predicted time step. It thereby respects daily and weekly seasonalities, but no exogenous dependencies.

Linear Regression The linear regression model is based on the following features: the load from 168 time steps before the predicted time step, the hour of the day (sine and cosine encoded), the day of the week (sine and cosine encoded), the month (sine and cosine encoded), whether it is a holiday (binary),

and the temperature and precipitation values. As the model is linear, an explanation can be derived from the model coefficients.

XGBoost Regressor XGBoost uses the same features as the linear regression. The default hyperparameters (100 estimators and a maximum depth of three) are used. The model is able to return feature importance values, but no information is available on how the features impact the prediction.

Time-series Transformer The time-series Transformer has the same model architecture as SHAPformer, but the model is trained without masking, so that it always receives the full information about the last week and exogenous features for the next week.

Temporal Fusion Transformer (TFT) TFT [168] is an encoder-decoder model based on an LSTM encoder and an LSTM decoder, followed by a single Transformer layer. It uses causal masking in the Transformer layer to prevent the model from using information of time steps after the prediction time step. We use the same features as for SHAPformer for both the encoder and the decoder, and no static covariates. Crucially, the model’s first layer is a variable selection network, which computes two sets of variable weights – one for the encoder and one for the decoder – that can be interpreted as feature importance values.

Table 9.1: The chosen hyperparameters for the different models.

Model	Hyperparameter	Synthetic	TransnetBW
Temporal Fusion Transformer	d_{model}	256	1024
	batch size	4	16
	heads	8	8
	optimizer	AdamW	AdamW
	learning rate	0.00067	0.00017
	decay rate	0.37	0.93
Transformer	layers	8	2
	d_{model}	64	128
	heads	4	2
	optimizer	AdamW	Adam
	batch size	16	64
	learning rate	0.00015	0.00010
SHAPformer	decay rate	1.00	1.00
	layers	7	2
	d_{model}	512	128
	heads	2	2
	optimizer	AdamW	Adam
	batch size	16	64
	learning rate	0.00010	0.00001
	decay rate	0.96	1.00

9.2.2 Hyperparameter optimization

The hyperparameters of TFT, the Transformer and SHAPformer are optimized using Bayesian Optimization in Weights and Biases (wandb) [35]. For SHAPformer on real-world data, the same hyperparameters are used as for the Transformer, but with a lower learning rate, which stabilizes the masked training. The selected hyperparameters are given in Table 9.1. Adam [148] and AdamW [174] are used as training algorithms, using the MSE loss function.

9.2.3 XAI approaches

Next to WindowSHAP [196] and SHAPformer introduced in Section 9.1, we use a number of established XAI approaches for comparison. PermutationSHAP, KernelSHAP and WindowSHAP are used to explain the time-series Transformer. In addition, two alternative explanation approaches based on the trained SHAPformer model are evaluated: feature attention and attention manipulation (AtMan).

PermutationSHAP PermutationSHAP is introduced in Chapter 3. It calculates SHAP values for individual features, which we sum up to generate SHAP values for feature groups. To estimate the runtime, we set the number of permutations to the default value 10 and we use 100 background samples. As this is not feasible to run for many samples, we reduce the number of permutations to two when calculating local and global explanations.

KernelSHAP KernelSHAP is an alternative to PermutationSHAP, which calculates local SHAP explanations by fitting a local linear model to outputs obtained from perturbed samples. Like with PermutationSHAP, 100 background samples are used to calculate SHAP values for individual features, which are then summed to generate SHAP values for feature groups.

Feature attention We use the feature attention mechanism of the trained SHAPformer model to estimate feature importance. The attention score of a feature embedding is used as proxy of the feature values' importance. Since the encoder and the decoder have separate feature attention mechanisms, we get two distinct sets of feature importances, once for the past features and once for the future features. As the attention scores sum up to one, the retrieved feature importances also sum up to one.

Attention manipulation (AtMan) We use attention manipulation [64] on the trained SHAPformer model as an alternative to calculating the SHAP values based on forecasts made with all feature subsets. Attention manipulation only considers the feature subsets with exactly one variable dropped out, and is therefore more efficient than the SHAP calculation. The absolute difference between the full prediction and the prediction with one variable dropped out is used to calculate the variable's importance. The returned importance values are normalized so that they sum up to one.

9.2.4 XAI evaluation metrics

On the synthetic data, the explanations from the XAI methods are evaluated using the feature importance error and the local explanation error.

Feature importance error This metric evaluates the feature importance values given as global explanations. An XAI method calculates a feature importance value for each variable. The feature importance error is defined as the MAE of the feature importance values compared to the ground-truth feature importance values.

Local explanation error This metric evaluates the SHAP values of local explanations. Therefore, it is only applicable to the SHAP algorithms. A local explanation consists of $H \cdot n_{\text{variables}}$ SHAP values for forecast horizon H and $n_{\text{variables}}$ variables. The local explanation error is defined as the MAE of the local explanations compared with the local ground-truth explanations.

9.3 Results

This section presents the results of the experiments in three steps. First, the forecast accuracy and the runtime of SHAPformer are compared to established forecasting models and XAI approaches. Second, SHAPformer’s explanations are validated on synthetic data with known ground-truth explanations. Third, SHAPformer is applied to real-world data from TransnetBW to gain insights into the trained forecasting model.

Table 9.2: Comparison of forecast models. Root mean squared error (RMSE) on the test set, training time, and inference time per forecast on the synthetic and real-world datasets. The best model is highlighted in bold and the second-best in italic. Zeros occur due to rounding.

Model	Forecast error (RMSE)		Training time		Runtime / forecast	
	Synthetic	TransnetBW	Synthetic	TransnetBW	Synthetic	TransnetBW
	-	[MW]	[h]	[h]	[ms]	[ms]
Persistence baseline	0.152	652.3	-	-	0.6	1.2
Linear Regression	0.149	553.7	0.00	0.00	0.0	<i>0.3</i>
XGBoost	0.119	387.0	<i>0.01</i>	0.00	0.0	0.1
TFT	0.059	390.8	5.20	1.84	8.4	13.3
Transformer	0.059	263.1	0.90	0.26	11.9	4.7
SHAPformer	0.060	265.9	10.55	3.46	10.8	4.7

Table 9.3: Forecast errors on the real-world data from TransnetBW, evaluated with five different metrics. Means and standard deviations over five runs are reported. The best model is highlighted in bold and the second-best in italic.

Model	MAE (scaled)	MSE (scaled)	MAE [MW]	RMSE [MW]	MAPE [%]
Persistence baseline	0.255 $\pm$ 0.000	0.177 $\pm$ 0.000	395.5 $\pm$ 0.0	652.3 $\pm$ 0.0	6.17 $\pm$ 0.00
Linear Regression	0.255 $\pm$ 0.000	0.128 $\pm$ 0.000	395.0 $\pm$ 0.0	553.7 $\pm$ 0.0	6.19 $\pm$ 0.00
XGBoost	0.177 $\pm$ 0.000	0.062 $\pm$ 0.000	274.2 $\pm$ 0.0	387.0 $\pm$ 0.0	4.18 $\pm$ 0.00
TFT	0.163 $\pm$ 0.016	0.065 $\pm$ 0.016	251.9 $\pm$ 24.3	390.8 $\pm$ 49.9	3.94 $\pm$ 0.41
Transformer	0.127 $\pm$ 0.005	0.028 $\pm$ 0.002	197.3 $\pm$ 8.3	263.1 $\pm$ 9.1	2.98 $\pm$ 0.12
SHAPformer	<i>0.131</i> $\pm$ 0.006	<i>0.029</i> $\pm$ 0.002	<i>203.3</i> $\pm$ 8.9	<i>265.9</i> $\pm$ 9.6	<i>3.09</i> $\pm$ 0.13

Table 9.4: Comparison of XAI methods. The feature importance error and local explanation error are evaluated on the synthetic data with ground-truth explanations.

XAI method	SHAP values	Runtime / explanation		Feature importance error [%]	Local explanation error
		Synthetic [s]	TransnetBW [s]		
TFT	no	0.01	0.03	8.04	-
Feature attention	no	0.01	0.00	7.45	-
AtMan	no	0.08	0.03	9.08	-
KernelSHAP	approx.	252.03	136.65	11.09	0.063
PermutationSHAP	approx.	1124.16	484.34	7.98	0.056
WindowSHAP	approx.	7.84	3.54	5.41	0.049
SHAPformer	exact	21.90	0.60	1.64	0.016

9.3.1 Fast calculation of exact SHAP values while maintaining forecast quality

Forecast accuracy SHAPformer achieves a forecast accuracy that is competitive with the best compared models (Table 9.2). All evaluated models outperform the persistence baseline, which simply predicts the value from one week earlier. The Transformer achieves the lowest forecast error on both datasets, with SHAPformer close behind, showing only about a 1% higher error. On the real-world dataset, both the Transformer and SHAPformer outperform Linear Regression, XGBoost [50], and TFT. The larger root mean square error (RMSE) of TFT on the real-world data stems from (a) a larger variance across experiment repetitions (although SHAPformer and the Transformer outperform TFT in all five runs), and (b) from large forecast errors before Christmas, where TFT’s causal attention prohibits the model from using the holiday feature of the later days when predicting the first days in the forecast horizon.

Forecasting results for five metrics evaluated on the real-world dataset from TransnetBW are given in Table 9.3. We observe that SHAPformer is close to the best model, the Transformer, regardless of the chosen metric. Note that the forecast errors are higher than in the benchmarking (Chapter 7), because the forecast horizon is extended to one week and the models are not continuously retrained.

Runtime The runtime of different XAI methods for explaining a single forecast is reported in Table 9.4 (evaluated on a machine with an AMD EPYC 7402P processor, 128 GB RAM and an NVIDIA 3090 RTX GPU with 24 GB VRAM). SHAPformer generates explanations more than $50\times$ faster than PermutationSHAP on the synthetic dataset and more than $800\times$ faster on the real-world dataset, achieving runtimes of less than one second per explanation. Remarkably, SHAPformer achieves this speedup despite computing exact SHAP values based on all coalitions of feature groups, whereas PermutationSHAP approximates SHAP values using only ten feature permutations. SHAPformer’s speedup comes partly from feature grouping, as WindowSHAP achieves an approximately $140\times$ speedup using the same strategy. In addition, SHAPformer leverages attention manipulation to exclude absent feature groups, eliminating the need for sampling and repeated model evaluations, which makes the evaluation of all coalitions computationally feasible while enabling an additional speedup. However, SHAPformer’s inference speed advantage comes at the cost of increased training time (Table 9.2), which is between two and thirteen times longer than that of TFT or the standard Transformer. Inference times of both the Transformer and SHAPformer are higher on the synthetic dataset than on the real-world dataset, as the models are larger (see Section 9.2.2 for the hyperparameters) and the synthetic dataset includes one additional covariate. SHAPformer is faster than WindowSHAP on the real-world dataset with 13 feature groups, but slower on the synthetic dataset with 14 feature groups due to the exponential growth in the number of evaluated coalitions in SHAPformer.

TFT and feature attention are the fastest explanation methods, because they require only one model evaluation to produce an explanation. However, they are not directly comparable to the SHAP-based approaches, as they provide only feature importance values but do not indicate how the features influence the predictions, making them less informative than SHAP-based approaches.

9.3.2 Successful validation of the explanations on synthetic data

Having demonstrated SHAPformer’s forecast accuracy and fast inference, we validate its explanations using synthetic data with known ground truth explanations. The synthetic time-series dataset incorporates daily, weekly, and annual seasonality, along with dependencies on covariates. Because the data generation process supports resampling of past target and covariate values, we can apply SHAP to compute ground truth explanations. Details of the dataset and ground truth construction are provided in Chapter 4.

Feature importance SHAPformer’s global feature importance is close to the ground truth, except that it underestimates the importance of the month feature (Figure 9.4). In contrast, PermutationSHAP and WindowSHAP deviate from the ground truth, particularly for key features such as the past target (called "load"), the hour of the day, and the day of the week. TFT is more difficult to interpret, as it produces separate sets of feature importance values for past and future features. Nonetheless, it is observable that neither of the sets resembles the ground truth, nor does a linear combination of the two. Notably, both SHAPformer and PermutationSHAP filter out the two noise covariates effectively, while the other methods assign them importance values greater than zero, although the target variable is independent of the noise covariates. We calculate the feature importance error, defined as the mean absolute error of the feature importance values shown in Figure 9.4 with respect to the ground truth, and report it in Table 9.4. We find that SHAPformer closely matches the ground truth feature importance, deviating only by 1.64

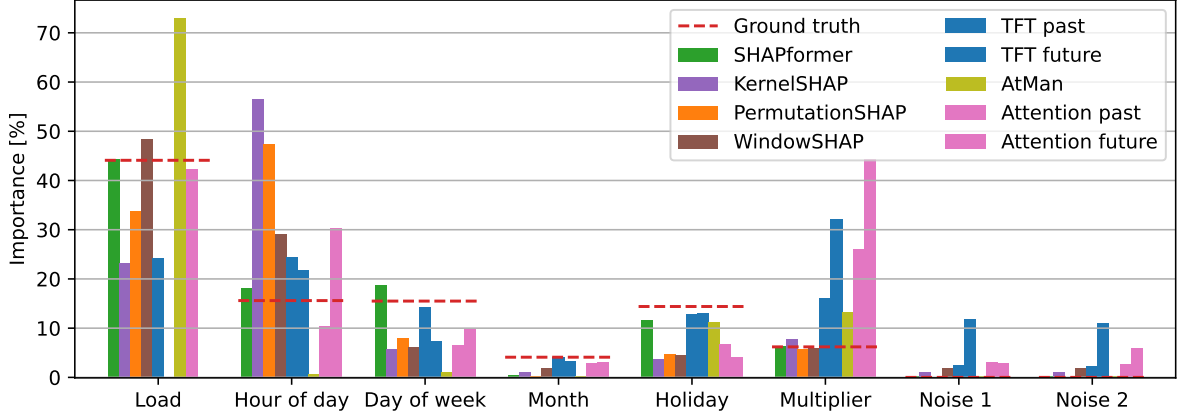


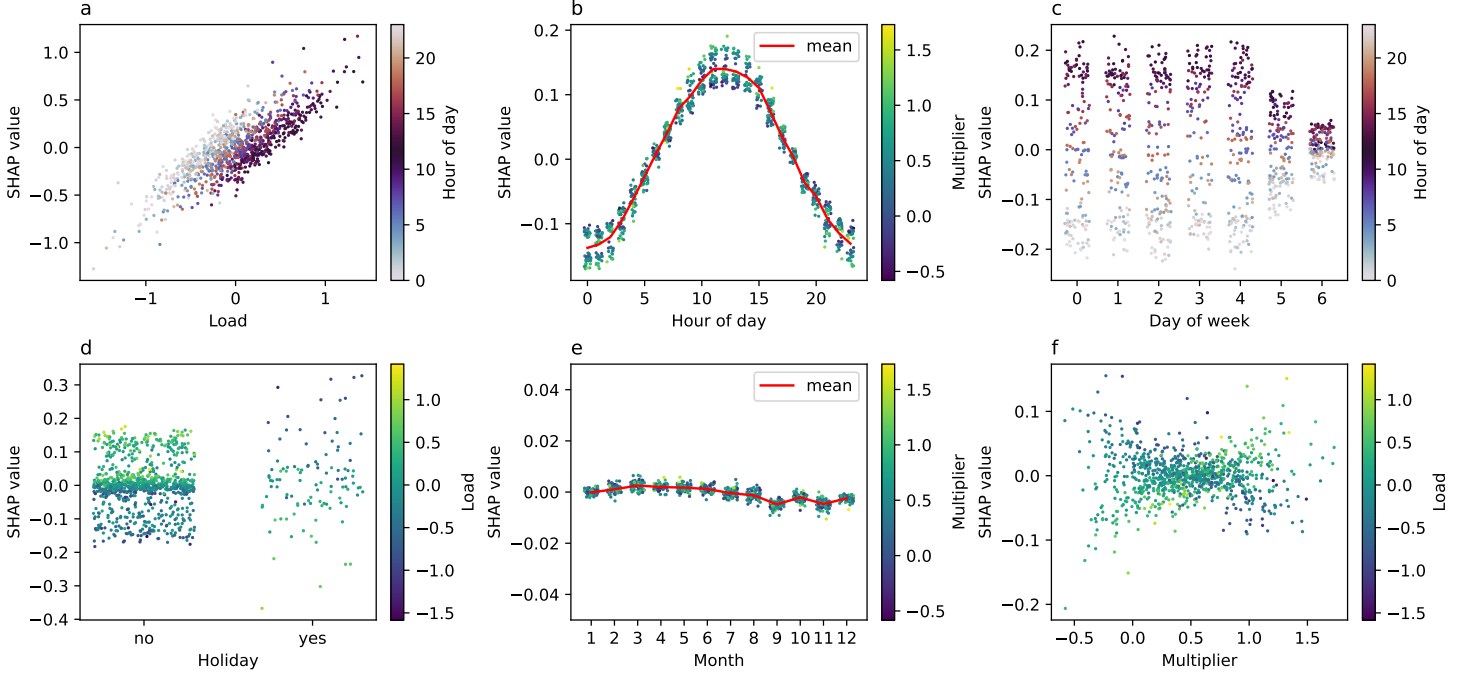
Figure 9.4: Feature importance on the synthetic test data. Attention uses the attention weights in the feature attention mechanism, separate in the encoder (attention past) and decoder (attention future). The ground truth is calculated by applying SHAP to the data generation process.

percentage points, whereas the feature importance error of the second-best WindowSHAP is more than three times larger. Since ground-truth explanations are available for all samples in the synthetic dataset, they can also be used to evaluate the local explanations of the SHAP methods. Table 9.4 reports the local explanation error, defined as the mean absolute error of the local explanations averaged across all samples, forecast horizons and feature groups. Again, we find that SHAPformer best matches the ground truth, resulting in a three- to four-times lower local explanation error than the other SHAP algorithms.

Feature dependence Next, we compare SHAPformer’s learned feature dependencies (Figure 9.5A) with the ground truth (Figure 9.5B). For these plots, m samples result in m points, each with the feature value on the x-axis and the SHAP value on the y-axis. A positive SHAP value indicates that a feature increases the model’s prediction relative to the mean prediction, whereas a negative SHAP value indicates a decrease. A sample results in 168 SHAP values for the 168 predicted time steps, but only the SHAP values for the 24-hour forecast (i.e. the 24th value in the forecast horizon) are shown in the dependence plots. Similarly, there are multiple feature values that could be shown on the x-axis (168 for the past load and 336 for each of the covariates), of which we chose the load of one week before the predicted time step for the load plot, and the future value of the predicted time step for all other plots. The color of the points indicates an interacting variable.

In general, the feature dependencies of SHAPformer align well with the ground truth, except that it underestimates the importance of the month. In panel (a) of Figure 9.5, the previous week’s load shows a linear effect on the prediction. A given past load results in a higher SHAP value at night (bright points) than during the day (dark points), because the load is expected to be lower at night. In panel (b), the dependence on the hour of the day forms a half-sine wave, which is exactly the daily pattern used during data generation. This interacts with the multiplier: negative multipliers (dark blue points) decrease the effect of the hour of the day, whereas positive multipliers (light green to yellow) increase its effect. Panel (c) reflects the multiplicative shrinking effect used in the data generation to model weekends (days five and six), with SHAP values closer to zero than on weekdays – negative at night and positive during the day. Panel (d) shows that holidays shrink the predicted load by increasing negative past loads and

A SHAPformer dependence plots



B Ground truth dependence plots

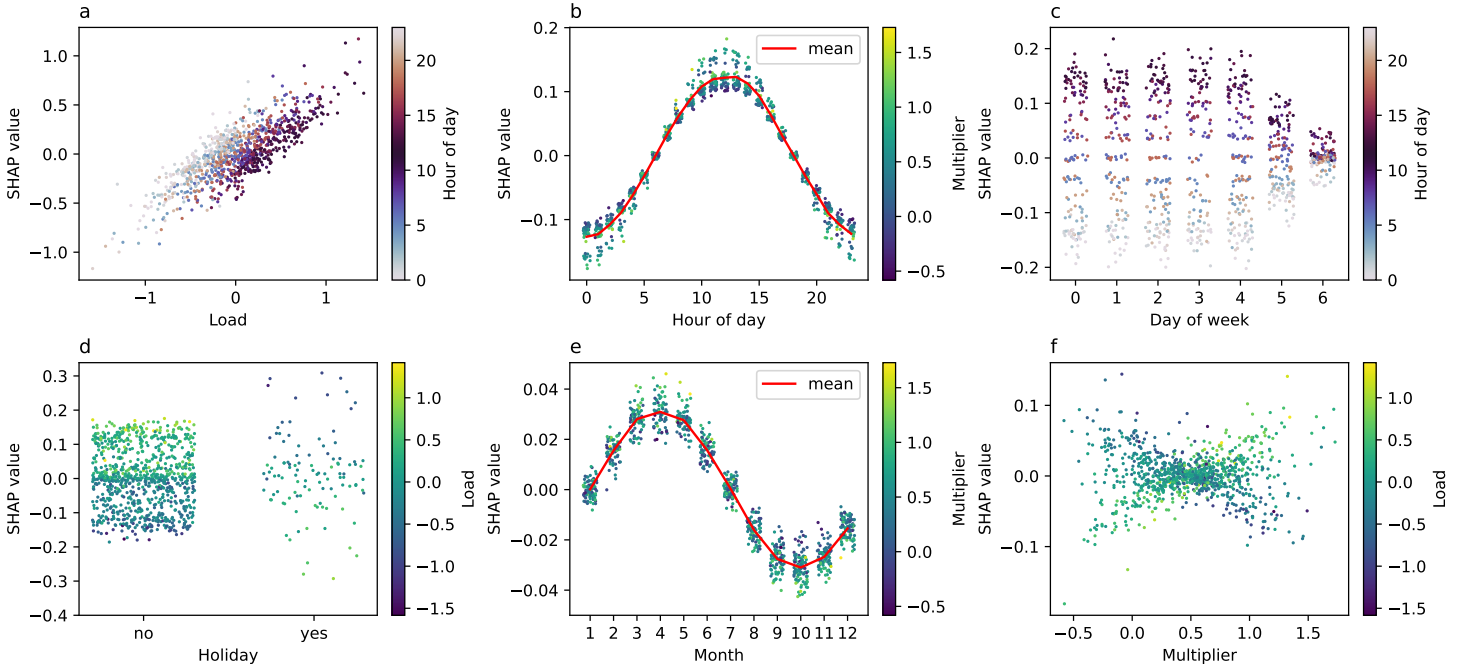
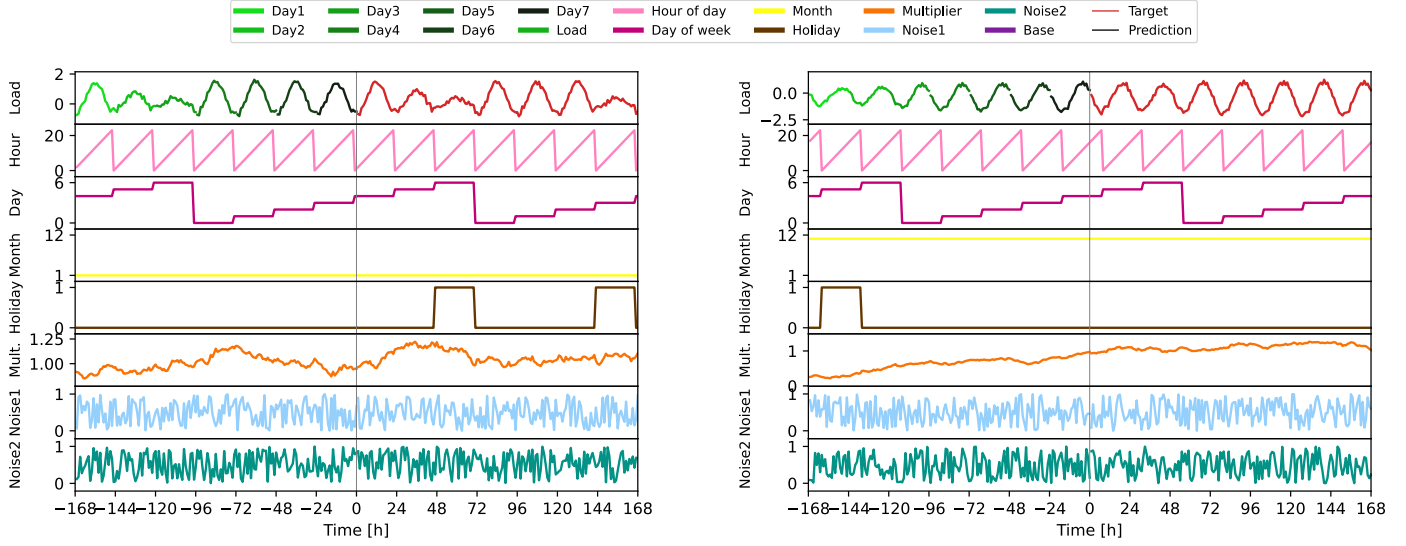


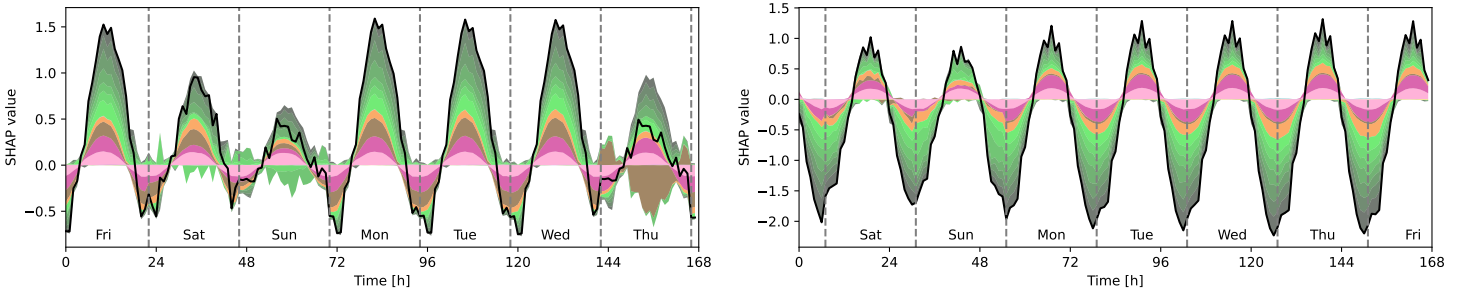
Figure 9.5: SHAPformer dependence (A) and ground truth dependence (B) on the six most important features of the synthetic dataset. The strongest interacting feature is shown in color (interactions mean the SHAP value depends on the interacting variable). For discrete variables, noise was added in the x-direction for visibility reasons. Figure from [115].

A Synthetic examples

Feature groups & target



B SHAPformer explanations



C Ground truth

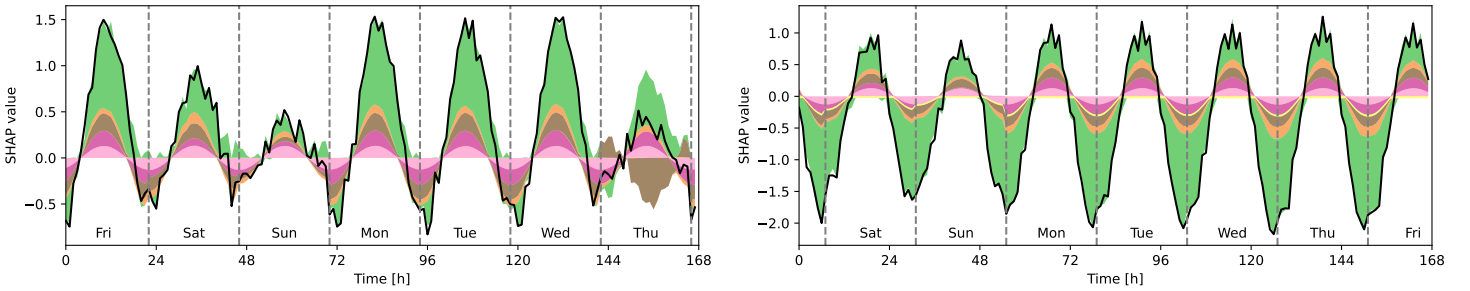


Figure 9.6: Local explanations of two different synthetic examples (left and right). Left: the holiday feature affects the last predicted day. Right: The increasing multiplier has an increasing effect. Figure from [115].

reducing positive ones, while non-holidays exhibit a reversed but less pronounced effect. In panel (e), the month feature has a sinusoidal effect in the ground truth, which SHAPformer partially captures, though with a reduced amplitude. Finally, panel (f) shows an X-shaped pattern for the multiplier feature: higher multipliers amplify positive previous loads and reduce negative ones, whereas lower multipliers have the opposite effect. Together, these patterns confirm that SHAPformer captures the dependencies present in the data.

Local explanations SHAPformers local explanations are also consistent with the ground truth. Local explanations of two synthetic examples are shown in Figure 9.6. On the left-hand side, the forecast horizon starts on a Friday. The hour of day, day of week, and holiday features influence the prediction in a pattern resembling the base load curve: a half-sine shape with higher loads during the day and lower loads at night. In this case, Sunday is also a holiday, so the holiday and day-of-week effects are both small compared to typical weekdays, resulting in a lower overall load. The previous Sunday (input day 3) exerts a negative influence on the predicted Sunday, as its particularly low load leads to a reduced forecast for the following week. The final day of the forecast horizon is also a holiday, where the holiday feature compresses the load curve – reducing positive daytime loads and amplifying negative nighttime loads. The ground truth looks similar to the SHAPformer explanation. Only the effect of the seven days is summarized, so the effect of input day 3 on the Sunday is not visible. On the right-hand side, the prediction starts on a Friday evening and the load is affected by a temperature increase. This is visible in the explanation of the forecast, where the temperature SHAP values increase in magnitude over time. In comparison to the ground truth, SHAPformer underestimates the effect of the least influential feature, the month, as observable on the right-hand side example.

9.3.3 Insights into load forecasts on real-world TSO data

With SHAPformer validated on synthetic data, we examine its explanations on the last twelve months of the real-world dataset. In this case, no ground truth is available, so the quality of the explanations must be assessed using domain knowledge.

Feature importance SHAPformer identifies the past load as the most important feature (Figure 9.7), followed by the day of the week and the hour of the day. The month, temperature, and holiday features contribute less and show similar levels of importance, while precipitation has almost no importance. The explanations of PermutationSHAP and WindowSHAP are similar to each other but differ from SHAPformer’s explanations, emphasizing the hour of the day and the day of the week, and assigning much lower importance to the past load than SHAPformer. For TFT and feature attention, the interpretation is more difficult, as they output two distinct sets of feature importances – one for past inputs and one for future covariates – with an unclear relationship between them. In TFT’s case, the most important past feature is the load, while the hour, temperature, and the day of the week are the most important future features.

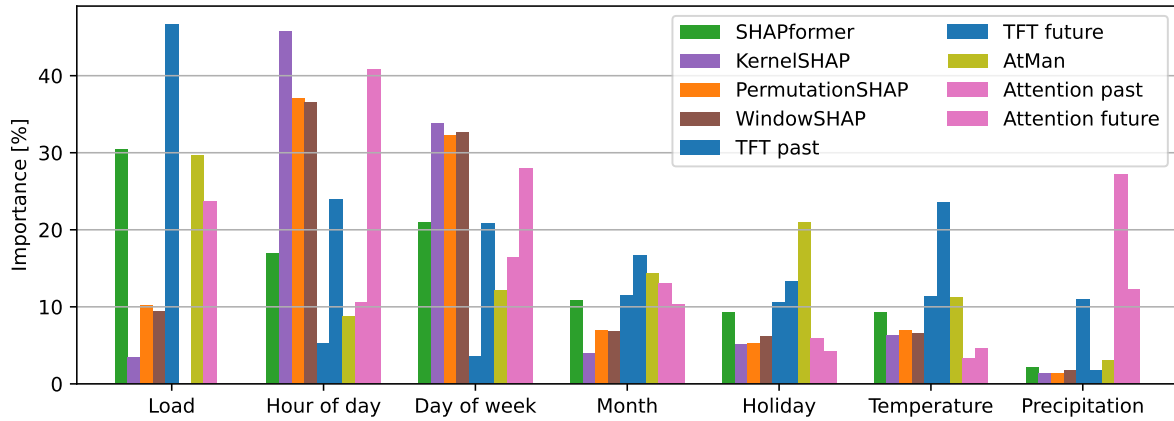
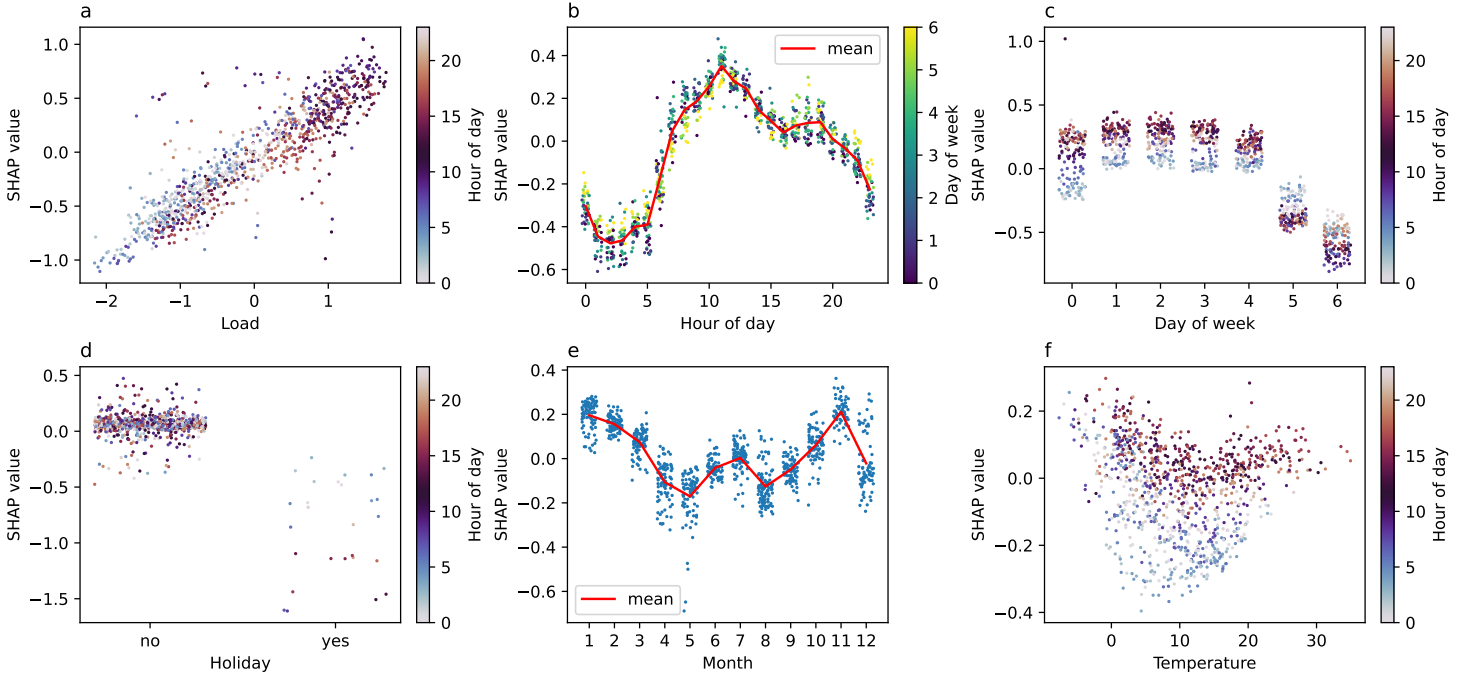


Figure 9.7: Feature importance scores by SHAPformer, a Transformer model explained using PermutationSHAP and WindowSHAP, Temporal Fusion Transformer (TFT), attention manipulation (AtMan) and feature-wise attention on the TransnetBW dataset. Feature-wise attention uses the weights from the feature attention mechanism, separate in the encoder (attention past) and decoder (attention future). Figure from [115].

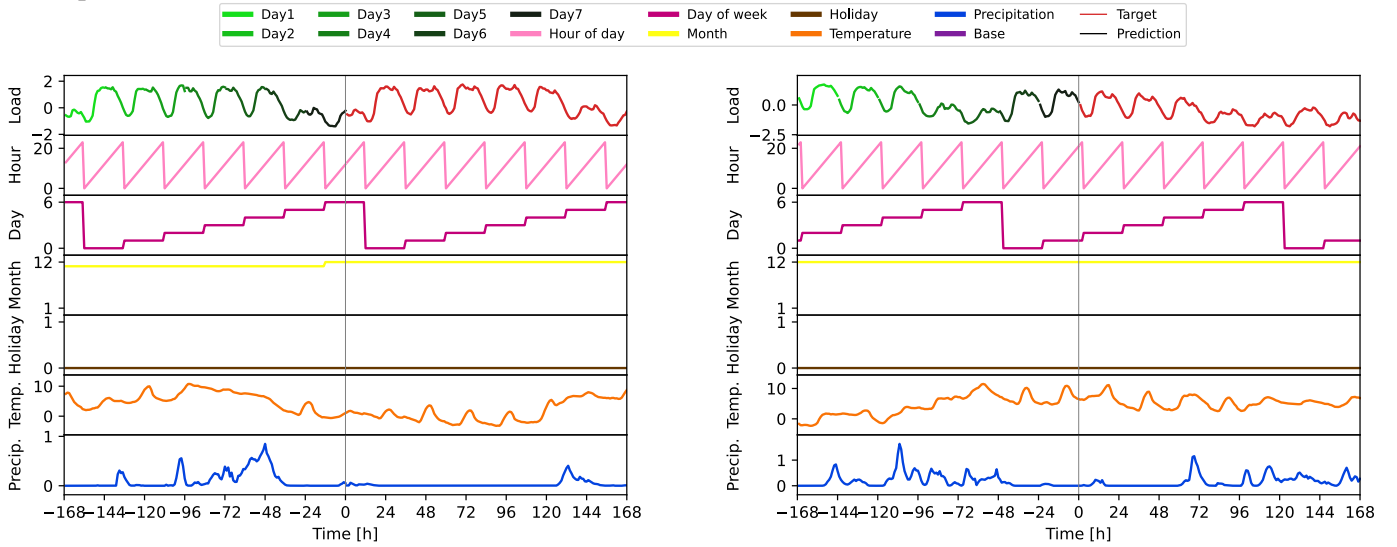
Feature dependence Figure 9.8 displays SHAPformer’s learned feature dependencies. SHAP values are expressed in terms of standardized electrical load, where a value of 1.0 corresponds to one standard deviation (approximately 1550 MW). In panel (a), a linear relationship between the previous week’s load and the forecasted load is observed. Notably, SHAP values for a given load value are higher at night (bright points) – likely because a load considered low during the day can be unusually high at night. Panel (b) shows the typical daily load pattern, with lower values at night and two peaks around noon and in the evening. On Sundays (yellow points), SHAP values tend to be closer to zero. Panel (c) illustrates that the predicted load is lower on Saturdays and Sundays, especially during the day (dark points). Holidays, shown in panel (d), consistently reduce the predicted load, although the effect is weaker at night (bright points). In panel (e), the month feature reflects a seasonal pattern with higher loads in winter, except in December, likely due to reduced industrial activity during the Christmas period. Finally, panel (f) shows that temperature influences the predicted load on cold days, with particularly strong increases when daytime temperatures are below 15 °C or nighttime temperatures are below 0 °C.

Local explanations Figure 9.8 presents local SHAPformer explanations for two forecasts in December. In both cases, the hour-of-day feature lowers the predictions at night and shows two daily peaks – mirroring the patterns observed in the dependence plots. The day-of-week feature reduces the predictions on weekends, as expected. Interestingly, in the left-hand example, the month feature has a positive effect despite being in December – likely because the previous week falls in November, which is represented in the month feature of the encoder input. In contrast, the right-hand example, which is later in December, shows a strongly negative month effect. The holiday feature has a positive impact during weekdays, suggesting that the model associates December with lower loads in general due to the extended holiday season, except during weeks without holidays. In the left-hand example, the colder temperatures in the early forecast days increase the prediction, while the warmer temperatures in the last two days decrease it. In the right-hand example, the temperature is more stable and has a consistently positive effect on the prediction. The effect of the past load also differs between the two examples: in the left-hand example, the past load increases the prediction on weekdays, whereas in the right-hand example it decreases the

A Dependence plots



B Example data



C Local explanations

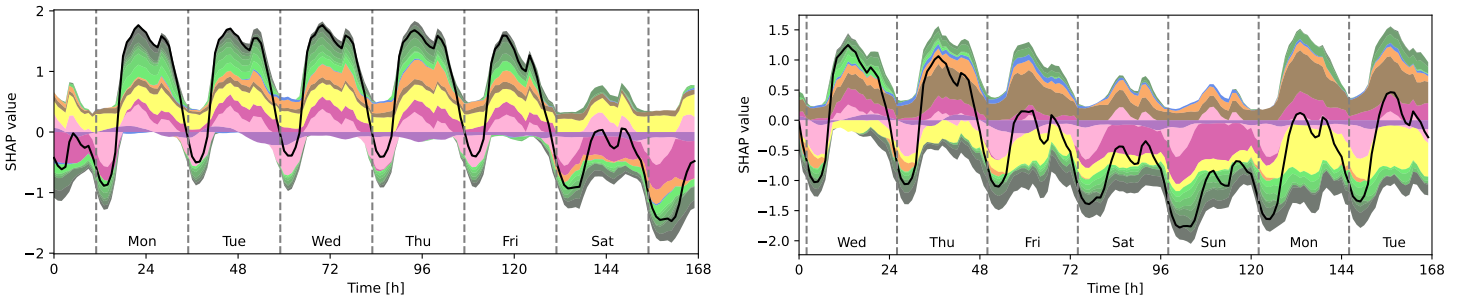


Figure 9.8: A: Dependence plots from SHAPformer on the TransnetBW data. B+C: Local explanations of examples from the real-world load data of TransnetBW. The start of the forecast horizon on the left-hand side is December 1, 2019, 13:00 and on the right-hand side December 17, 2019, 22:00. Figure from [115].

prediction from days 3 to 7 – especially influenced by the last two input days, which show a lower average load than the first days.

Overall, the observed dependence patterns align well with domain knowledge, reinforcing confidence in SHAPformer’s predictions. The December pattern – where a lower load is treated as the default and non-holidays as exceptions – is unexpected, but can be plausibly explained by the extended holiday period during that time.

Detailed results of the Transformer model explained with PermutationSHAP and WindowSHAP, feature attention and AtMan, as well as an analysis of the SHAP values of the input days evaluated with SHAPformer, are given in Appendix C.

9.4 Discussion

Efficient SHAP for Transformers We introduce SHAPformer, an accurate and efficient approach for explainable time-series forecasting. SHAPformer generates SHAP explanations for Transformer-based models several orders of magnitude faster than the established PermutationSHAP. This significant speedup results from two key mechanisms: grouping input features to reduce the number of coalitions and manipulating the attention weights to estimate marginal contributions without sampling. Both mechanisms contribute to reduced runtime. WindowSHAP, which applies feature grouping but not attention manipulation, is faster than PermutationSHAP. However, only SHAPformer achieves inference times below one second per explanation.

Validation of the explanations by SHAPformer SHAPformer has been successfully validated on synthetic data, as it closely reproduces the ground truth in terms of feature importance, dependence patterns, and feature interactions. It effectively filters out irrelevant noise features and captures key relationships, including daily patterns, multiplicative effects of holidays and weekends, and temperature interactions. One exception is the month feature, whose importance SHAPformer appears to underestimate, likely because its influence in the data generation process is small and comparable to the noise added to the target time series (see Section 4.8). However, note that the ground truth is created on the basis of multiple assumptions, such as that the ground truth should resemble SHAP values and it should respect dependencies between features (e.g., that the hour of the day can be inferred from the jumps in the day-of-the-week feature). Other explanations might be valid under different assumptions, especially when the goal is to explain a trained model that has learned different dependencies. Our evaluation against ground-truth explanations measures the correctness of the generated explanations with respect to the data generation process. Thereby, it complements other correctness tests, like perturbation of the most important features [194]. We also checked the plausibility of the explanations retrieved from SHAPformer on the TransnetBW data, using our domain knowledge about influences of external factors on the electric load. In the future, the usability of the explanations and their influence on user trust can be tested in user studies.

True to the data Existing SHAP algorithms are either true to the model or true to the data [48]. Conditional sampling tends to produce explanations that are true to the data, while off-the-manifold sampling generates explanations that better reflect the model’s internal behavior [80]. We argue that SHAPformer achieves both: it is true to the model and true to the data. This is made possible by its sampling-free design and training strategy, which builds robustness to absent features. In the presence of correlated features, SHAPformer learns to predict accurately using any of the correlated inputs, distributing the predictive contribution among them. This contrasts with models trained on the full feature set, which may rely on arbitrary feature combinations, leading to explanations that misrepresent underlying data relationships. SHAPformer’s strong alignment with the ground-truth explanations on synthetic data supports this claim.

Insights into the trained model Applying SHAPformer to real-world load data provides valuable insights into how different features influence the model’s predictions. The most important predictor is the load from the previous week, followed by the day of the week and the hour of the day. Other features – such as month, temperature, and holidays – also contribute, while precipitation has minimal impact. This contrasts with the Transformer model explained by PermutationSHAP and WindowSHAP, which emphasizes temporal features more heavily and downplays the role of the past load. This difference highlights how global explanations can help differentiate between models with similar forecast accuracy but different learned dependencies. In our view, a forecasting model can reasonably rely on both calendar-based inputs or lagged load values to estimate typical load patterns. SHAP values are instrumental in uncovering which of these cues the model relies on. Given that SHAPformer aligns more closely with the ground-truth explanations on synthetic data, we are confident that its explanations on real-world data are likewise more faithful to the underlying data relationships than those from the comparison methods.

Limitations Despite its advantages, SHAPformer has several limitations. First, SHAPformer’s fast inference comes at the cost of increased training time. Training with masked inputs requires the model to learn a more complex structure across varying feature subsets, and since it only sees partial inputs in each iteration, the effective amount of training data per epoch is reduced. Nevertheless, for applications where many forecasts need to be explained, the reduced inference time justifies the additional training effort. Second, the computational complexity of SHAPformer scales exponentially with the number of feature groups, because the model is evaluated for all 2^N subsets of N feature groups. In our setup with seven input days and six or seven covariates (depending on the dataset), this full enumeration is computationally feasible and desirable, as it yields the most accurate SHAP estimates. However, in scenarios with more feature groups, such as longer input sequences or additional covariates, this exhaustive approach may become impractical. Future work can address this issue by estimating approximate SHAP values based on a subset of the feature groups. Third, there are cases where the assumption that past covariates affect the forecast only in combination with past observations does not hold (e.g., when a past holiday indicates a long weekend with reduced load). A future version of SHAPformer may mask feature groups only based on masked feature attention instead of masked self-attention and cross-attention to give improved explanations in these rare cases. Fourth, feature importance and local explanations can vary across training runs. For five repeated model trainings on the real-world dataset, the standard deviation of

the feature importance values is 1.60 percentage points (averaged across features). Fifth, the empirical evaluation is limited to one synthetic dataset and one real-world electrical load dataset (and an additional electricity price dataset in [115]), since each dataset requires domain understanding to verify explanations, and we leave the evaluation on more datasets to future work. Finally, the synthetic benchmark relies on several assumptions: the ground-truth explanations are defined using SHAP values derived from the data-generating process, they assume that the learned model faithfully represents this process, and they respect dependencies between features (e.g., the hour-of-day can be inferred from the day-of-week feature which indicates day changeovers). While these assumptions allow controlled evaluation, faithful explanations for models that do not follow these assumptions will differ from the ground truth.

9.5 Conclusion

This chapter introduces SHAPformer, an accurate time-series forecasting model with the ability to calculate SHAP values efficiently without sampling from background data. Our analysis reveals that SHAPformer’s explanations are faster to compute, as well as more accurate than the explanations of PermutationSHAP, KernelSHAP and WindowSHAP. Thereby, this chapter answers RQ 3 by demonstrating that attention manipulation can be used to estimate faithful SHAP values efficiently. The model explanations on real-world data from TransnetBW allow for interesting insights, such as the model following typical daily and weekly patterns and holiday effects, as well as a special model behavior in December due to the extended Christmas season.

In the future, other models than Transformers can also be trained on feature subsets to enable fast sampling-free explanations at inference time. Training models in this way not only facilitates explainability but may also improve robustness to known anomalies and missing values. By manipulating attention or using equivalent mechanisms to ignore unreliable inputs, such models can adapt more effectively to real-world data imperfections. One option to drop out a variable is setting its value to zero. This could be misleading in some cases, since the zero often has a meaning (e.g., after standardization, zero is the variable’s mean value). A value different from all observed values could be used (e.g., a large negative value). Alternatively, a binary flag can be introduced for each variable, indicating whether this variable is absent. The model can then learn to distinguish between a normal zero value and zero representing a dropped-out variable. Then, SHAP values can be computed efficiently without sampling. A drawback is the potentially higher training time, as observed with SHAPformer.

We release SHAPformer as a Python package to make it available to researchers and practitioners¹. This will facilitate its application and evaluation across a range of domains beyond electrical load forecasting. Given SHAPformer’s successful validation on synthetic data, we are confident that it will generalize well to other use cases.

¹ SHAPformer is available on GitHub: <https://github.com/KIT-IAI/SHAPformer>

10 An Efficient Algorithm for Explaining Time-Series Foundation Models with SHAP

Previous publication

This chapter is a modified excerpt of:

[111] M. Hertel^a, A. Nikoltchovska^a, S. Pütz, B. Schäfer, R. Mikut and B. Hagenmeyer. “Explainable Load Forecasting with Covariate-Informed Time Series Foundation Models”. *ACM e-Energy* 2026. DOI: 10.1145/3744255.3811724.^b

^a Matthias Hertel and Alexandra Nikoltchovska contributed equally.

^b Licensed under a Creative Commons Attribution 4.0 International License, <https://creativecommons.org/licenses/by/4.0/>.

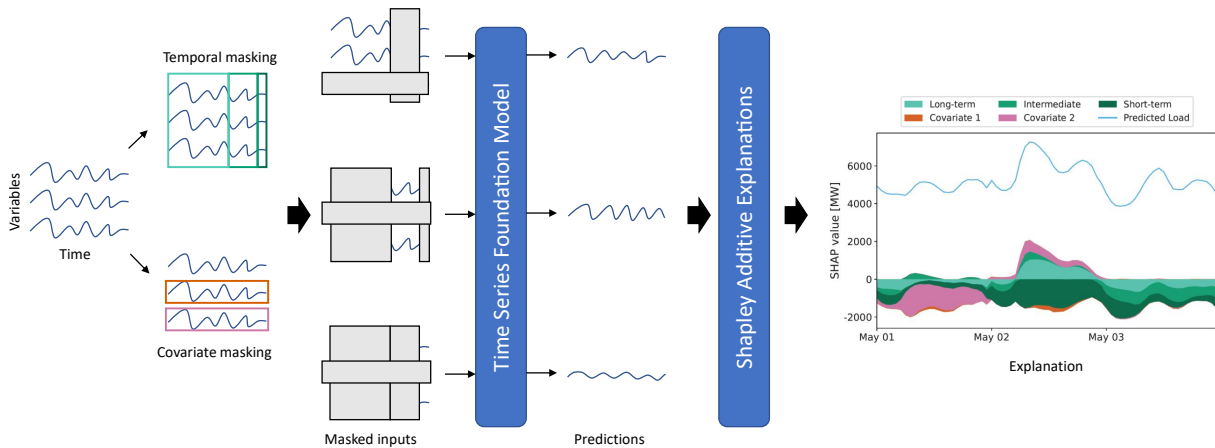


Figure 10.1: Overview of our approach to efficiently compute exact Shapley Additive Explanations (SHAP) for covariate-informed Time Series Foundation Models (TSFMs). We begin with historical hourly load observations accompanied by weather and calendar covariates. Our masking strategy operates along temporal and covariate dimensions – selectively removing historical time steps or entire covariates to create coalition samples. Each masked sample is then processed independently through Chronos-2 and TabPFN-TS to generate forecasts. Finally, SHAP values are computed by comparing predictions across coalitions, attributing the forecast to specific periods in the context and/or individual covariates. Figure from [111].

Time-Series Foundation Models (TSFMs) [167] have recently emerged as a new class of large-scale deep learning models, built predominantly on Transformer architectures and pretrained on large and diverse collections of time series data. Unlike conventional deep learning approaches for forecasting, which are typically trained from scratch on task-specific datasets, TSFMs leverage large-scale pretraining to achieve strong zero-shot and few-shot generalization across tasks and domains. More recent architectures, such as Chronos-2 [11] and TabPFN-TS [127], extend this paradigm by supporting the integration of covariates –

additional time-varying features that influence the target (sometimes also referred to as exogenous features or auxiliary information) – which is particularly relevant for energy system applications.

Like previous work [11, 127], we define models as *univariate* when they forecast a single time series, whereas *multivariate* models forecast multiple time series together. Models that incorporate covariates in addition to past values from the target time series are called *covariate-informed*.

Despite their promising performance, covariate-informed TSFMs remain largely opaque, and their increasing complexity exacerbates the challenge of understanding and trusting their predictions. Developing and applying suitable XAI methods is therefore a key requirement for the deployment of TSFMs in operational energy system contexts. Post-hoc explainability methods such as SHAP [176] offer theoretically grounded insights into model behavior, but their computational cost poses a major barrier when applied to large-scale time series models with long input horizons and multiple covariates.

To address this gap, we introduce an efficient SHAP-based explainability algorithm tailored to TSFMs and demonstrate its applicability to load forecasting, a key challenge in the energy domain [144]. We apply our approach to the TSFM Chronos-2 [11] (and TabPFN-TS [127] in [111]) using operational electrical load data from Baden-Württemberg, Germany. The overall procedure of our approach is illustrated in Figure 10.1. We make the following key contributions:

1. We evaluate Chronos-2 for electrical load forecasting on recent operational data from the TSO TransnetBW, and benchmark its performance against models trained from scratch.
2. We propose an efficient SHAP algorithm for TSFMs. Like SHAPformer in Chapter 9, the algorithm for explaining TSFMs uses temporal and covariate masking to enable the estimation of SHAP values without the need for sampling from background data.
3. We empirically demonstrate that Chronos-2 makes meaningful use of covariates, such as calendar effects and weather variables, and that the resulting explanations are consistent with established domain knowledge.

The remainder of this chapter is structured as follows: The proposed approach for efficiently computing SHAP-based explanations is introduced in Section 10.1, followed by a description of the experimental setup in Section 10.2, including the dataset, preprocessing steps, TSFM configurations and evaluation metrics. Section 10.3 assesses the performance of Chronos-2 on recent operational load data. Section 10.4 presents the explainability analysis, examining TSFM behavior through global feature importance patterns and local explanations of individual predictions. Finally, Section 10.5 draws conclusions and discusses future research directions.

10.1 Methodology

Chronos-2 Chronos-2 [11] is a pretrained foundation model for time series forecasting capable of handling univariate, multivariate, and covariate-informed forecasting tasks. The model architecture is based on a Transformer encoder, receiving past observations and past and future covariates, and a final quantile prediction layer. For covariate-informed forecasting, Chronos-2 builds patches for each variable

and uses group attention to incorporate information from the covariates. This design allows the model to learn cross-channel dependencies while leveraging its pretrained representations for zero-shot and few-shot forecasting scenarios.

Model inputs In contrast to classical machine learning models, TSFMs are flexible in the context length and covariates provided to the model. This allows to evaluate them on subsets of the data without sampling absent features from background data. A TSFM evaluated with context length C and forecast horizon H receives the past C values of the target time series as input (denoted as $X \in \mathbb{R}^C$), together with an arbitrary number of covariates Z_i . These covariates can either be available for the past only ($Z_i \in \mathbb{R}^C$), or extend into the forecast horizon ($Z_i \in \mathbb{R}^{C+H}$) when they are known in advance, such as calendar features or weather forecasts.

Feature grouping Computing exact SHAP values requires to evaluate the TSFM with 2^n feature subsets for n features. As this is not feasible for large amounts of features, we instead group the features into N subsets, with $N \ll n$, and evaluate the TSFM only 2^N times. We incorporate two grouping mechanisms: *Temporal grouping* splits the past load time series into consecutive patches, in order to analyze the effect of the load in different past periods. In particular, we introduce one group for the last day (hours -24 to -1 relative to the time the prediction is made), short-term (hours -168 to -25), intermediate (hours -672 to -169) and long-term (hours < -672) periods. *Covariate grouping* builds one group for each covariate, which is used to analyze the effect of the covariate on the prediction. In total, we analyze four temporal groups and $|Z|$ covariate groups.

Masking for Chronos-2 To evaluate a given feature subset, the groups not belonging to it are *masked*, i.e., withheld from the model input. *Temporal masking* works in two ways. To mask the first k values of the context window, the context length gets reduced to $C - k$, so that the masked values are not contained in the context. To mask intermediate or the most recent values, the load values are set to not-a-number (NaN), which indicates missing values. Internally, Chronos-2 sets missing values to zero and uses a special token to distinguish between missing values and measured zeros. *Covariate masking* works by removing masked covariates entirely from the model input. This is possible because Chronos-2 is pretrained on varying sets of covariates. When all data is masked so that the model input is empty, a base prediction is made. For this, we use the mean load from the previous 48 weeks, which corresponds to the model’s maximum input length.

The masking in SHAPformer and the masking for Chronos-2 have in common that two types of masks are applied: temporal masking of past time periods and covariate masking. For Chronos-2, the number of past windows is reduced to four (last day, short-term, intermediate and long-term) due to the larger computational cost of the model compared to SHAPformer. The covariate masking of the two approaches uses different mechanisms, which both have the effect that the model’s access to a covariate is restricted: the attention weights are manipulated in SHAPformer to represent different covariate sets, whereas Chronos-2 is provided with different covariate sets in the input.

10.2 Experimental Setup

Dataset The TransnetBW dataset for the period January 2015 – September 2025 is used for the experiments. The solar irradiance and the outside temperature are used as covariates. In practice, the reanalysis data is not available at prediction time, but it is a common approach to use weather data as perfect forecasts in the absence of historical weather forecasts [100]. A third covariate is used to encode holidays. It is set to 1 for Sundays and public holidays, and to 0 else. The last year of data (October 2024 – September 2025) is used as test data for the experiments, the year before (October 2023 – September 2024) as validation data and everything before (January 2015 – September 2023) as training data to train models other than TSFMs from scratch.

Forecasting task & metrics To evaluate forecast performance, a forecast is made at every hour in the test set with a horizon of 24 hours. We evaluate the models as point forecasters, taking the median of the predicted distribution as the forecast. Our focus is on evaluating and explaining point predictions, as these are the primary outputs used in operational forecasting decisions. Extending the analysis to probabilistic forecasts is left for future work. We use three different metrics to evaluate the forecast accuracy: MAE, RMSE and MAPE. To explain models, we generate explanations of the forecasts made at midnight, so that each hour in the test set is predicted exactly once and we do not get multiple explanations for the same hour. Note that the concatenation of the explained forecasts comprises the entire year without overlaps.

Chronos-2 We evaluate Chronos-2 with the default hyperparameters in a zero-shot setting. Chronos-2 is a probabilistic model returning quantile predictions, which we turn into point forecasts by taking the median of the predicted distribution. We test different context lengths ranging from one week to the model’s maximum of 8192 hours. We report forecasting results for a univariate Chronos-2 and for Chronos-2 with the three covariates described above, in order to quantify their contribution.

Non-TSFM models We compare the TSFMs to a simple baseline and to Transformer models trained from scratch on the TSO data. The baseline predicts the observed load from the last day of the same type, distinguishing between workday, Saturday and Sunday/holiday as the types. The *Transformer* architecture is inspired by preliminary experiments on the TransnetBW dataset [113]. It is an encoder-decoder model with two layers each and a hidden dimension of 128 and four attention heads, trained from scratch with two weeks context length using the AdamW optimizer with a batch size of 128, a learning rate of 0.0001 and early stopping with ten epochs patience. We report results for two Transformer variants with different amounts of training data, namely one year (October 2022 – September 2023) and the full training data comprising 8.75 years (January 2015 – September 2023).

10.3 Benchmarking TSFMs on Recent Operational Load Data

In this section, we analyze the forecast accuracy and the model runtimes to evaluate the applicability of TSFMs to load forecasting.

Table 10.1: Forecasting results on the test set, with the best result per metric highlighted in bold and second best in italics. Chronos-2 is evaluated in both univariate (univ.) and covariate-informed settings.

Model	Training data	MAE [MW]	RMSE [MW]	MAPE [%]
Baseline	—	325.9	451.2	5.24
Transformer	1 year	213.4	318.9	3.32
Transformer	8.75 years	150.2	206.8	2.34
Chronos-2 (univ.)	—	214.0	356.9	3.40
Chronos-2	—	<i>156.2</i>	<i>229.1</i>	<i>2.44</i>

Forecast accuracy Table 10.1 reports the forecasting results on the test set. All models outperform the baseline across all error metrics. The Transformer trained on the full data achieves the lowest errors, while the same architecture trained on one year of data shows substantially higher errors. Chronos-2 achieves lower errors than the Transformer trained on one year of data. Compared to the one-year Transformer, Chronos-2 reduces the MAE by 26.8 %. When trained on the full data, the Transformer is better than the TSFMs, but only by a small margin. Incorporating covariates leads to an MAE improvement of 27.0 % compared to the univariate setting, highlighting the importance of incorporating exogenous information.

Despite the strong overall forecasting performance of Chronos-2, it encounters difficulties on specific days in proximity to public holidays, including October 4, December 23, January 7, June 20, and June 23. This suggests potential for improvement through feature engineering, such as incorporating covariates for long weekends, school holidays, or other periods of reduced economic activity. For the complete year of forecasts made daily at midnight, see Figure 10.4.

Overall, the results highlight the importance of integrating covariates into TSFMs for accurate load forecasting. TSFMs are especially useful in scenarios with little training data, as they outperform a Transformer trained from scratch on one year of data. When lots of data is available, models trained from scratch can achieve better results than zero-shot TSFMs, but the performance difference is small and the TSFMs have the advantage that they are easy to use without requiring any training, which makes them appealing in practice.

Runtimes Table 10.2 reports the runtimes for model training, inference and explanation, evaluated on a machine with an AMD EPYC 7402P processor, 128 GB RAM and a NVIDIA GeForce RTX 3090 with 24 GB VRAM. Training the Transformer from scratch on the full training dataset takes less than 20 minutes. This effort is not necessary when TSFMs are used, which are pretrained on large diverse datasets and applicable zero-shot to new data without the need for further training. Chronos-2 has more parameters than the Transformer trained from scratch, and therefore needs more time to generate a prediction. However, both models are very efficient and allow to predict in a few milliseconds. To generate an explanation, Chronos-2 is run with all $2^7 = 128$ combinations of the seven temporal and covariate masks. Explaining a forecast takes less than $128 \times$ the prediction time, because masking reduces the context length and number of covariates in the model input.

Table 10.2: Training runtime, runtime to generate a single prediction, and runtime to generate a single explanation.

Model	Training data	Training [min]	Prediction [ms]	Explanation [s]
Transformer	8.75 years	19.2	6.8	—
Chronos-2	—	—	65.6	5.12

10.4 Explaining TSFMs for Load Forecasting

Having established that TSFMs show competitive performance, we next investigate the SHAP-based global and local explanations of Chronos-2 to understand how the provided data influences the predictions. Results for TabPFN-TS are reported in [111].

10.4.1 Global explanations

Feature importance Table 10.3 presents the feature importance for Chronos-2, calculated as the percentage of the absolute SHAP values over the test set. The past load is by far the most important feature, with 89 % total importance. For Chronos-2, the four temporal windows have similar importance, despite their different lengths. This indicates that recent information is important for the prediction, as the last day alone is almost as important as the whole long-term context. The holiday is the most important covariate, followed by temperature, and irradiance is the least important.

Table 10.3: Feature importance on the test set.

Feature Group	Importance
Past load	
Long-term (weeks ≤ -5)	23.64 %
Intermediate (weeks -4 to -2)	23.39 %
Short-term (days -7 to -2)	19.88 %
Last day (day -1)	22.30 %
Covariates	
Holiday	4.51 %
Temperature	3.55 %
Irradiance	2.74 %

Covariate dependence In order to analyze how the covariates are used, Figure 10.2 shows the SHAP values of Chronos-2 in dependence of the three covariates. The x-axis shows the feature values at the predicted hour (or the difference to the value 24 hours before) and the y-axis shows the corresponding

SHAP values. A positive SHAP value indicates that the feature increased the prediction, and a negative SHAP value indicates a decreased prediction. The dot color is used to show an interacting variable.

- *Holiday dependence:* The SHAP values are shown for six categories of days: Sunday, Monday, a workday following a workday ($\circ/\circ$), a workday following a holiday ($\bullet/\circ$), a holiday following a workday ($\circ/\bullet$), and a holiday following a holiday ($\bullet/\bullet$). When the previous day was a holiday and the next day is no holiday ($\bullet/\circ$), the holiday feature increases the predicted load. When the next day is a holiday ($\circ/\bullet$ and $\bullet/\bullet$), it decreases the load instead. The effect is stronger during the day (dark dots) than during the night (bright dots), where the base load is similar on all weekdays. This aligns well with our expectation that the load is reduced on holidays due to less industrial and economic activity, and jumps back to a normal level after the holiday. Chronos-2 is only slightly affected by the holiday feature on Sundays and Mondays, indicating that Chronos-2 infers the weekly pattern from the past load.
- *Temperature dependence:* For temperatures below 15 °C, Chronos-2 reacts to temperature differences with respect to the previous day. When the temperature increases with respect to the temperature 24 hours ago (red dots), the predicted load decreases, resulting in negative SHAP values. When the temperature decreases (blue dots), the opposite effect is observed, resulting in positive SHAP values. This can be plausibly explained by the increased use of electric heating on cold days. At warmer temperatures, the magnitude of the SHAP values is smaller and the effect of the temperature difference is less pronounced.
- *Irradiance dependence:* The lower panels show the SHAP values depending on the difference in the irradiance from the day before. Unsurprisingly, the difference is zero at night (bright dots), because the sun does not shine. During the day (dark dots), an almost linear pattern is visible, with the slope directly quantifying how much the predicted load changes per unit change in irradiance. The models use the irradiance feature to react to changing weather conditions – when the irradiance decreases (negative Δ irradiance), the predicted load increases (positive SHAP value), and vice versa. This can be explained by the fact that the TSO load is affected by behind-the-meter photovoltaic, so that a higher irradiance decreases the effective load.

Overall, the covariate dependence of Chronos-2 aligns well with our expectations derived from domain knowledge about influences of holidays, temperature and irradiance on the TSO load, indicating that Chronos-2 makes meaningful use of the covariates.

10.4.2 Local explanations

Next, we examine the local explanations for both past load windows and covariates to give further insights into when particular covariates are used by Chronos-2. We highlight four representative examples in Figure 10.3 that illustrate the model’s behavior across different scenarios.

- *Post-holiday effects* (Figure 10.3 (a)): On January 6, a public holiday in Baden-Württemberg, the holiday feature shows a negative SHAP value, indicating that the model appropriately reduces the predicted load. The following day (January 7) marks the first workday after an extended holiday

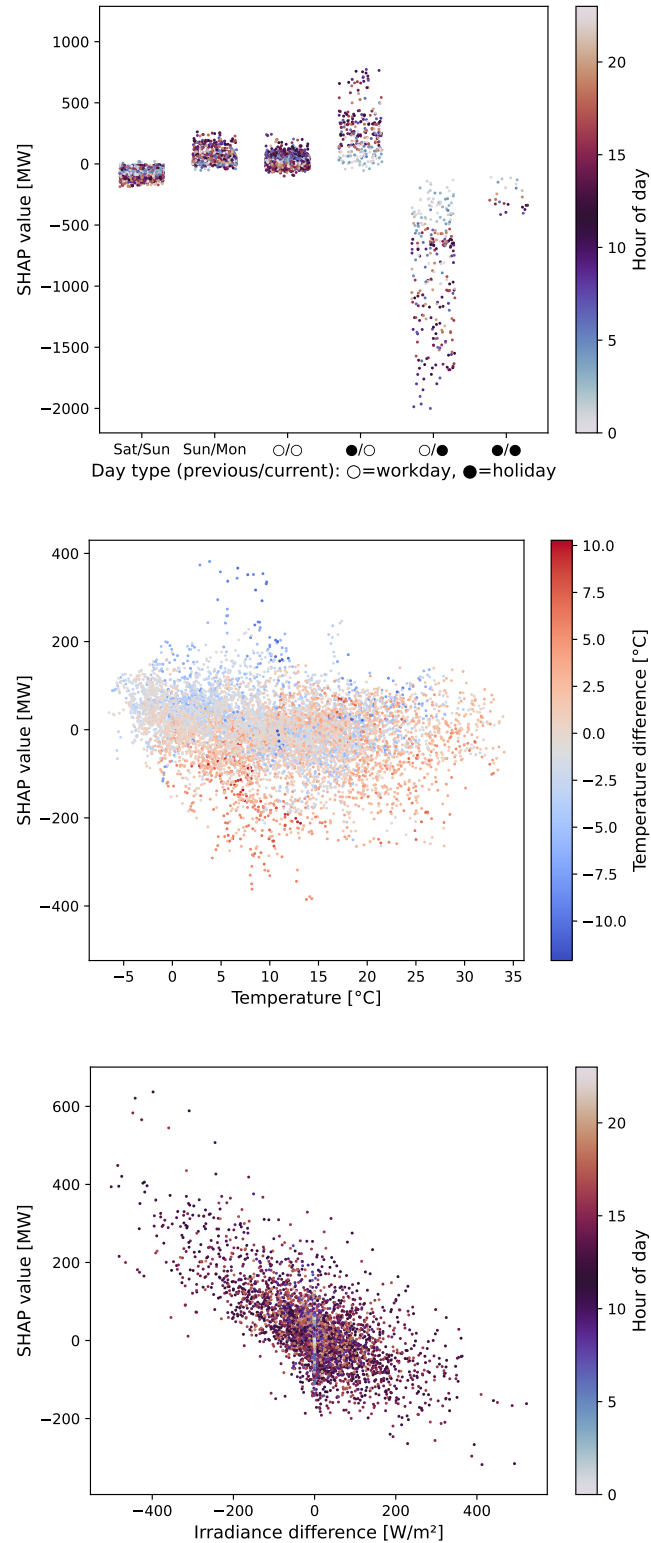


Figure 10.2: Dependence of Chronos-2 on the holiday covariate (top), the temperature covariate (middle) and the irradiance difference to the previous day (bottom). The dot color indicates an interacting variable. The predicted load is decreased on holidays (top), increased when the outside temperature drops on cold days (middle), and increased when PV generation drops due to lower irradiance (bottom). Figure adapted from [111].

period. While the holiday feature still plays a role by contributing positively to increase predicted load (reflecting the return to workdays), the previous day has a negative influence, resulting in an underestimation of the load on this transition day. By January 8, the model generates more accurate predictions by using January 7 as a reference – indicated by the large influence of the last day on January 8. This shows how the model uses the immediate post-holiday day as a marker for the end of the Christmas period.

- *Temperature effects* (Figure 10.3 (b)): Rising temperatures on February 20-21 produce negative SHAP values for the temperature feature, appropriately reducing the predicted load, which aligns with the inverse temperature-load relationship we would expect from domain knowledge that a part of the heating load is electric.
- *Holiday effects* (Figure 10.3 (c)): The May 1 example illustrates the impact of holidays. The holiday feature decreases the load on May 1 (a Thursday) and increases it on May 2 to account for the return to normal working patterns the following day. Similar to January 7, the previous day negatively impacts the prediction. This shows that the holiday feature helps the model to predict that the low load on the previous day is not due to a permanent change in consumption levels, but the load will increase again on the next workday.
- *Irradiance effects* (Figure 10.3 (d)): On August 20, a lower solar irradiance compared to the previous day leads to adjustments in the midday load curve. This indicates that the model captures the dependency between solar generation and load. Reduced behind-the-meter photovoltaic output lowers self-consumption and increases the net load, which is reflected in a higher prediction.

An analysis of the complete test period in Figure 10.4 reveals several broader patterns in the importance of covariates. Covariates contribute most strongly on specific days rather than uniformly over time. In particular, their importance increases when temperature or irradiance differs substantially from the previous day, or when the preceding or following day is a holiday. Furthermore, the previous day's load is especially influential for forecasts on Tuesdays, while its importance is markedly lower on Mondays and weekends. This pattern indicates that the model captures weekly cyclical patterns in electricity consumption behavior, distinguishing between workdays and weekends.

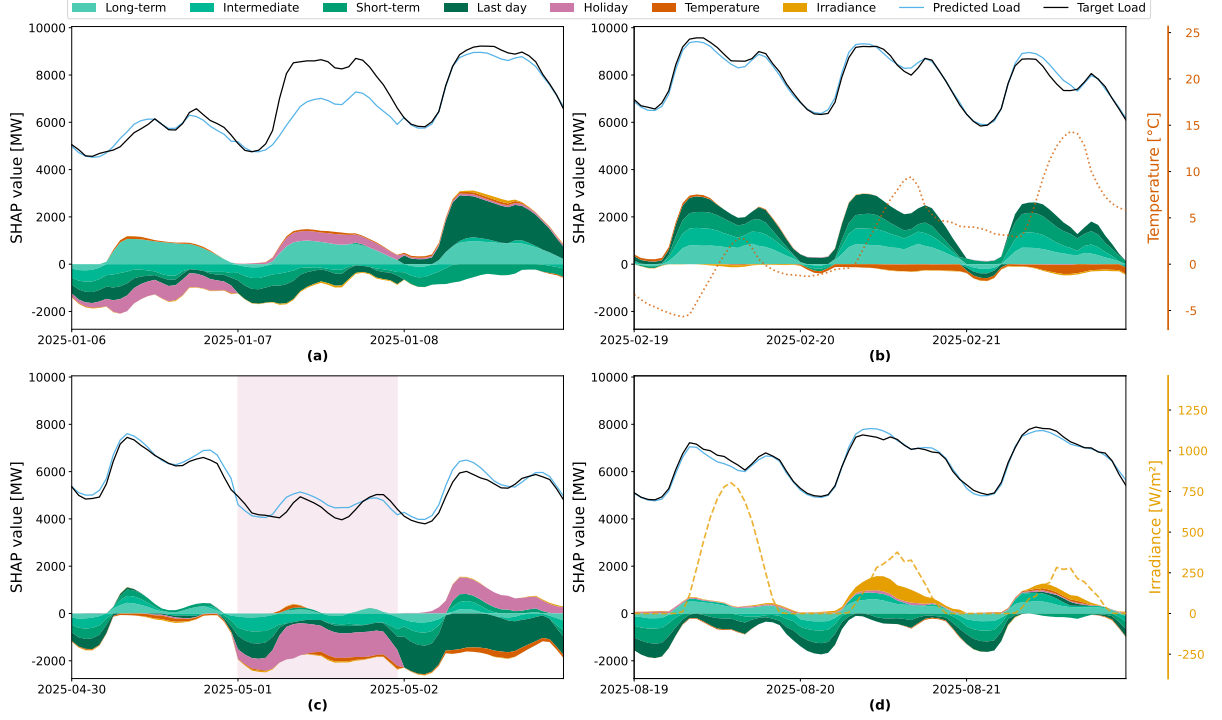


Figure 10.3: Local SHAP explanations for Chronos-2 predictions on selected days showing feature contributions to load forecasts: (a) post-holiday effects (January 6-8), (b) temperature responses (February 20-21), (c) holiday patterns (May 1-2), and (d) irradiance influence (August 20). Positive (negative) SHAP values correspond to an increase (decrease) in the predicted load. Figure from [111].

10.5 Conclusion

This chapter proposes an efficient algorithm to calculate SHAP explanations for Transformer-based TSFMs and applies it to explain electrical load forecasts from Chronos-2. TSFMs like Chronos-2 [11] and TabPFN-TS [127] use attention in the time dimension and in the variables dimension. Since the attention mechanism (Equation 3.14) is defined for an arbitrary amount of input vectors, TSFMs can be run with varying covariate sets and varying context lengths. Our proposed SHAP algorithm leverages this property by evaluating the model on subsets of the full input information, which allows to calculate SHAP values without sampling from background data. With our two XAI approaches SHAPformer (Chapter 9) and SHAP for TSFMs (Chapter 10), we can answer RQ 3: the attention mechanism allows to efficiently calculate sampling-free SHAP explanations for Transformer models, including Transformer-based TSFMs.

Our global and local explanations on the TransnetBW data reveal that Chronos-2 uses covariates in ways that align with established domain knowledge of energy consumption patterns, including calendar effects, temperature dependencies and behind-the-meter PV generation. Our work demonstrates that TSFMs can satisfy both the accuracy and explainability demands of critical infrastructure applications while preserving their zero-shot advantages.

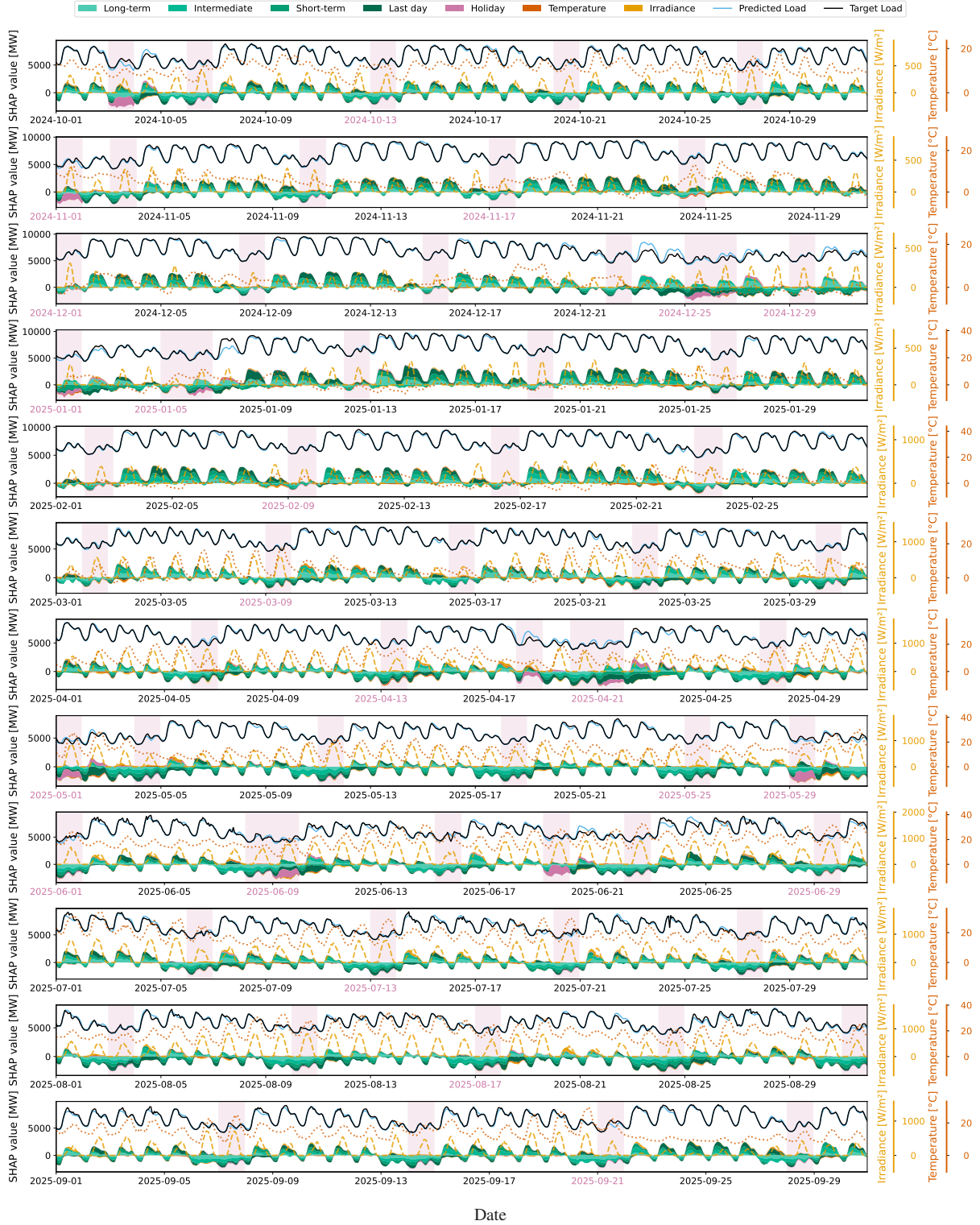


Figure 10.4: Chronos-2 predictions and local SHAP feature attributions across the full test set, with one panel per month. Each panel shows: actual load (black), model predictions (blue); SHAP values indicating feature contributions to load forecasts, where positive (negative) values correspond to increased (decreased) predicted load; actual covariate values for temperature and irradiance (right y-axis), with Sundays and holidays marked in pink. Figure from [111].

Part V

Applications of Electrical Load Forecasts

11 Peak Load Forecasting for the BigDEAL Challenge

Previous publication

The results in this chapter are partially published in:

[105] B. Heidrich^a, M. Hertel^a, O. Neumann^a, V. Hagenmeyer^b, and R. Mikut^b. “Using conditional Invertible Neural Networks to perform mid-term peak load forecasting”. *IET Smart Grid*, 2024, DOI: 10.1049/stg2.12169.^c

^a These authors contributed equally: Benedikt Heidrich, Matthias Hertel, Oliver Neumann.

^b These authors contributed equally: Veit Hagenmeyer, Ralf Mikut.

^c Licensed under a Creative Commons Attribution 4.0 International License, <https://creativecommons.org/licenses/by/4.0/>.

A forecast must be used for operations in a down-stream application to create a benefit for the energy transition. Part V introduces two peak forecasting applications and analyzes whether Transformer-based forecasts lead to superior down-stream results. Chapter 11 compares the Transformer to the results from the BigDEAL challenge on peak load forecasting [252]. Chapter 12 compares different forecast models in a battery control task for an energy community with heat pumps (HPs). Thereby, Part V addresses RQ 4:

Research question (RQ) 4

Are Transformer-based electrical load forecasts beneficial for down-stream peak forecasting and peak shaving applications?

Due to the ongoing energy transition, an increase in decentral electricity consumption and generation is expected, as more PV, EVs, and HPs are installed. These devices have a high simultaneity, as all PV plants in a region feed into the grid when the sun is shining, many EV owners have similar daily schedules, and HPs have similar control algorithms depending on the outside temperature. This can lead to increased feed-in peaks and demand peaks. For the grid operators, the peaks cause problems, as they lead to overloading the infrastructure such as electrical transformers and lines. To plan peak shaving measures, accurate forecasts of the peak time, peak magnitude and peak shape are needed. This is why the BigDEAL load forecasting challenge 2022 focused on peak load forecasting. Our conditional Invertible Neural Network (cINN)-based approach [105] performed best in two out of three tracks, resulting in a second place out of 78 participating teams in the overall ranking. In this chapter, a Transformer is benchmarked against the results of the challenge in order to analyze whether it improves peak load forecasting accuracy.

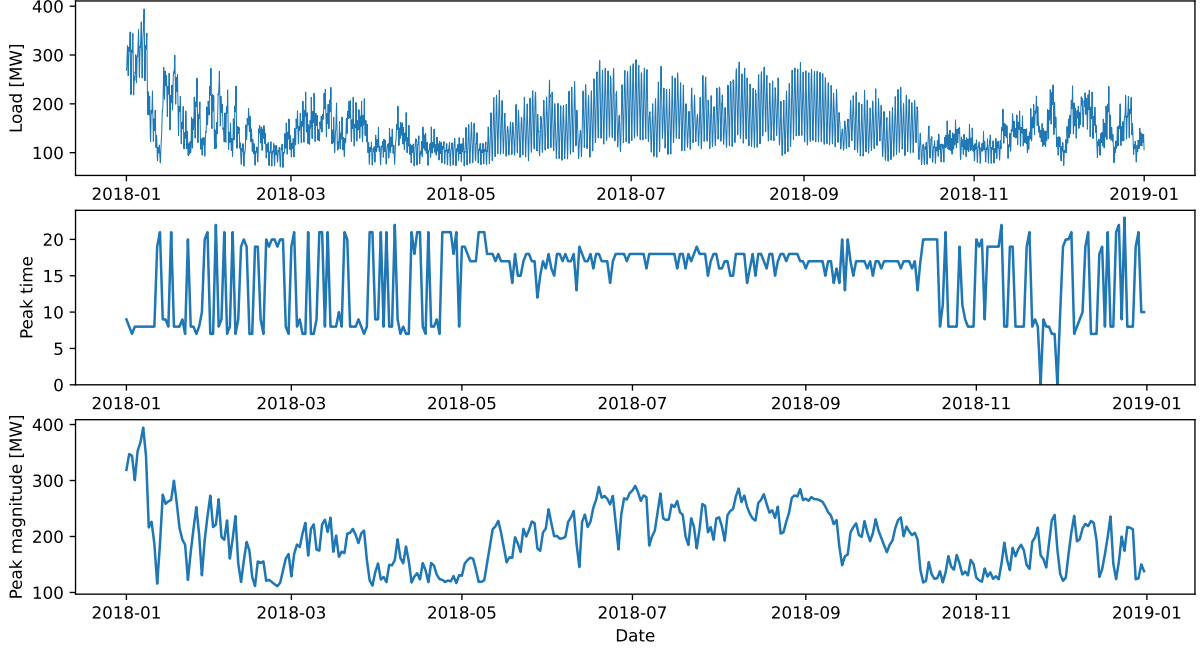


Figure 11.1: LDC 1 test data: the hourly load of one year (top) and the peak times (middle) and magnitudes (bottom).

11.1 Experimental Setup

This section describes the dataset and the metrics used in the BigDEAL challenge, as well as the models used for peak forecasting.

11.1.1 Dataset

The dataset used in the final of the BigDEAL challenge is the local distribution companies (LDCs) dataset introduced in Chapter 4. It comprises the hourly load of three LDCs, together with temperature forecasts for the years 2015-2018 for six nearby weather stations, of which we take the average. We enrich the dataset with features for the hour, month and weekday (all sine and cosine encoded) as well as holidays (binary encoded). The year 2018 is used as test data, split over six rounds spanning between one and three months. Figures 11.1, 11.2 and 11.3 show the load, peak time and peak magnitude for the three LDCs in the test period.

11.1.2 Metrics

For a given day i , we denote the 24 predicted hourly load values as a vector $\hat{\mathbf{y}}_i = (\hat{y}_{i,1}, \dots, \hat{y}_{i,24})$ and the actual hourly load values as a vector $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,24})$. In the following, we present metrics used by the organizers of the BigDEAL challenge to evaluate the performance of the forecasting method regarding the three tasks [252].

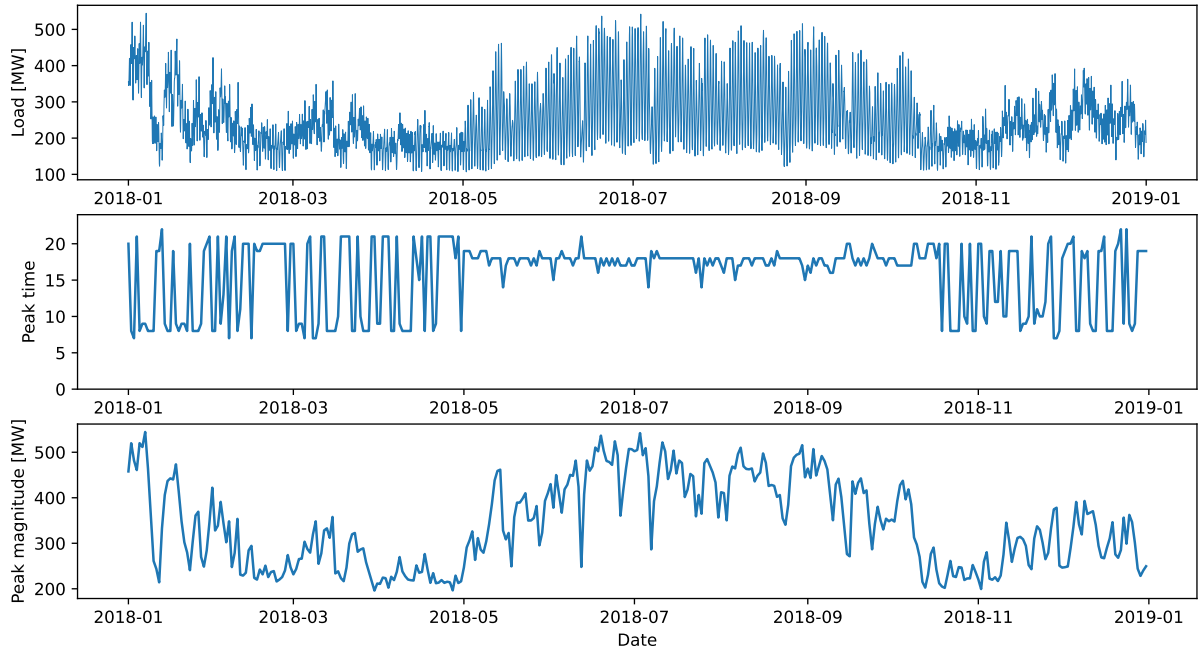


Figure 11.2: LDC 2 test data: the hourly load of one year (top) and the peak times (middle) and magnitudes (bottom).

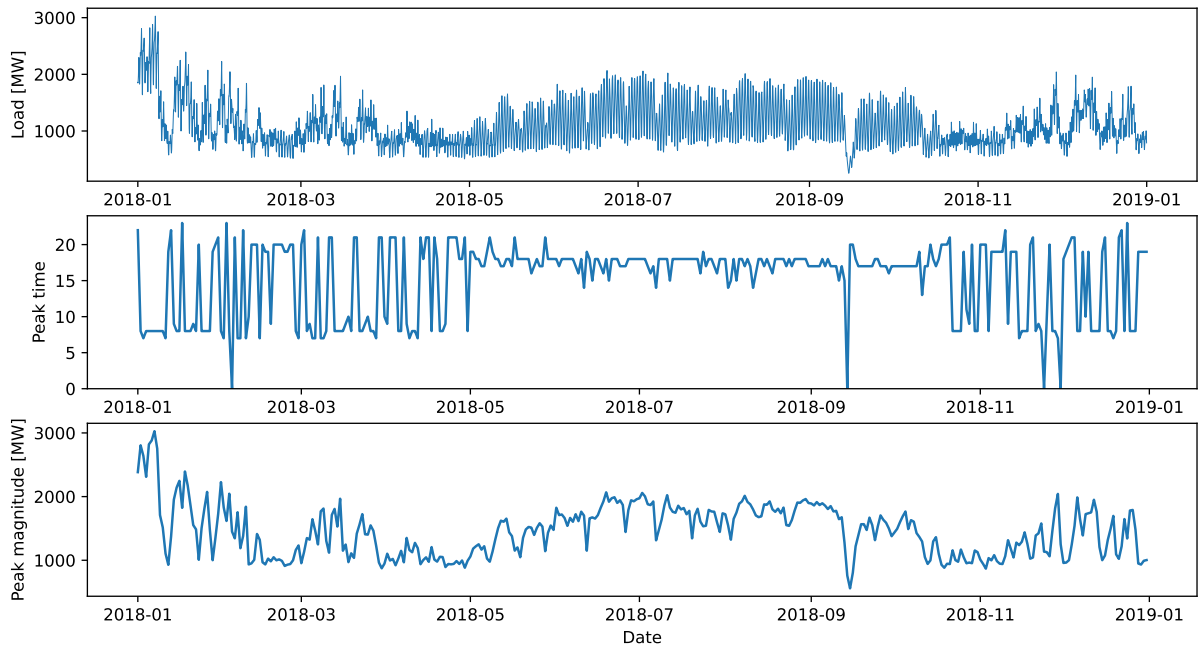


Figure 11.3: LDC 3 test data: the hourly load of one year (top) and the peak times (middle) and magnitudes (bottom).

Magnitude The challenge uses the mean absolute percentage error (MAPE) to evaluate the magnitude predictions. Let n be the number of days in the test data, $m_i = \max_j(y_{i,j})$ the ground truth peak magnitude and $\hat{m}_i$ the predicted peak magnitude for day i . The MAPE is defined as follows:

$$\text{MAPE}(\hat{m}, m) = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{m}_i - m_i}{m_i} \right| \quad (11.1)$$

Timing A weighted and capped mean absolute error (wMAE) is used to evaluate the peak timing predictions. The weight for the error of a day is one if the forecast is off by one hour, two if the forecast is off by two to four hours, and the error is capped at ten if the forecast is off by five hours or more. Let $p_i = \text{argmax}(y_i)$ be the ground truth peak magnitude and $\hat{p}_i$ the predicted peak magnitude for day i . The wMAE is defined as follows:

$$\text{wMAE} = \frac{1}{n} \sum_{i=1}^n \text{weighted}(|\hat{p}_i - p_i|) \quad (11.2)$$

with

$$\text{weighted}(x) = \begin{cases} x & \text{for } x \in \{0, 1\} \\ 2x & \text{for } x \in \{2, 3, 4\} \\ 10 & \text{for } x \in \{5, 6, \dots, 23\} \end{cases} \quad (11.3)$$

Note that the values of $x \in \{0, 1, \dots, 23\}$ are discrete.

Shape The calculation of the shape metric consists of three steps. First, the predicted loads and ground truth loads are normalized by the daily peaks

$$\bar{\mathbf{y}}_i = \mathbf{y}_i / \max_j(y_{i,j}) \quad (11.4)$$

$$\bar{\hat{\mathbf{y}}}_i = \hat{\mathbf{y}}_i / \max_j(\hat{y}_{i,j}). \quad (11.5)$$

Second, the sum of the absolute errors in the period ranging from two hours before the actual peak to two hours after the actual peak is computed for every day. Third, the daily sums of errors are averaged across all days to obtain the shape metric:

$$\text{shape}(\bar{\hat{\mathbf{y}}}, \bar{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^n \sum_{j=p_i-2}^{p_i+2} |\bar{\hat{y}}_{i,j} - \bar{y}_{i,j}| \quad (11.6)$$

where $p_i = \text{argmax}(\mathbf{y}_i)$ is the peak timing of day i .

11.1.3 Forecasting models

Transformer model The Transformer model is an encoder-decoder model with a context size of 168 hours and a forecast horizon of 24 hours. A forecast is made at midnight for the following day. The

Table 11.1: Features of the models evaluated on the BigDEAL data.

Model	Past load	Weather data	Calendar data
Persistence	yes	no	no
SVR	no	yes	yes
NN	no	yes	yes
cINN	no	yes	yes
Transformer	yes	yes	yes

Table 11.2: BigDEAL results.

Model	Magnitude			Timing			Shape		
	LDC1	LDC2	LDC3	LDC1	LDC2	LDC3	LDC1	LDC2	LDC3
Persistence	12.161	10.258	12.474	3.467	2.941	3.198	0.205	0.194	0.204
SVR	5.116	6.013	5.185	1.546	1.448	1.205	0.099	0.123	0.093
NN	4.041	4.376	4.555	1.638	1.113	1.189	0.105	0.111	0.091
cINN	3.462	3.855	4.143	1.180	0.934	0.790	0.061	0.084	0.051
Ensemble	3.521	3.681	4.052	1.438	1.229	1.153	0.062	0.085	0.064
Transformer	4.518	5.243	4.202	1.452	1.544	1.047	0.068	0.091	0.063

hour with the highest predicted load defines the predicted peak time and predicted peak magnitude. The model hyperparameters are optimized on a held-out validation set of the data from LDC 1, using Bayesian optimization in Weights and Biases [35] over 100 iterations.

Comparison models A cINN, a SVR and a neural network (NN) were developed for the BigDEAL challenge [105]. These models do not use past observations, as the task in the BigDEAL challenge was to forecast between one month and three months into the future, so that no recent load values were available. The different model inputs are summarized in Table 11.1. In addition to the individual models, an ensemble of the cINN, the SVR and the NN was used in the challenge. The ensemble is a linear combination of the predictions from the three models, with the optimal model weights estimated on a held-out validation set. The comparison results for the SVR, NN, cINN, and the ensemble are taken from [105]. In addition, a persistence baseline is introduced, which predicts the same load curve, peak timing and peak magnitude as observed the day before.

11.2 Results

The results for the three metrics evaluated on the three LDCs are shown in Table 11.2. All models beat the persistence baseline on all LDCs and metrics by a large margin. The Transformer is better than the SVR and the NN on one LDC for the magnitude task, on two LDCs for the timing task and on all three LDCs for the shape tasks. However, the cINN achieves better results than the Transformer consistently

on all tasks and LDCs. The ensemble improves the results only on two magnitude tasks compared to the best result of a single model.

11.3 Conclusion

The Transformer is competitive to the SVR and the NN. The Transformer outperforms the SVR and the NN on the shape task for all three LDCs and on the other tasks for one LDC. However, the comparison is unequal, because the Transformer has access to the past load values, whereas the other models only rely on exogenous features. The fact that the Transformer does not improve much on the results from the exogenous-only models speaks for a weak autocorrelation of the peak magnitude and timing. This is further supported by the fact that the persistence baseline, which only uses the past load values, results in large errors.

The cINN succeeds at peak load forecasting based on covariates only. Despite only relying on exogenous features, the cINN produces accurate peak magnitude, timing and shape forecasts on the data from the three LDCs, resulting in a second place overall in the BigDEAL challenge and beating the persistence baseline and the Transformer model consistently, even though these approaches use recent observations of the past load. There are two possible explanations for the strong performance of the cINN compared to the Transformer. First, the peak load in the datasets used in the BigDEAL challenge depends more on the weather than on the past observations, as can be seen by the fact that models without past observations outperform the persistence baseline. This may lead to a bad Transformer model, which relies on an autoregressive architecture. Second, the cINN is a generative model, whereas the compared models are deterministic forecasting models that estimate the expected value. The exact timing of the peaks may be difficult to predict, so that the expected value for an hour is lower than the peak load. In contrast, the cINN is trained to generate realistic instances, which contain a peak, although at varying positions. Further investigations are needed to verify if the strong peak forecasting performance of the cINN can be reproduced on other data, including time series with a stronger autoregressive relationship. In the future, it can be tested whether the cINN is also successful in time-series forecasting instead of peak forecasting, and whether it benefits from incorporating recent observations.

This chapter provides evidence that the cINN is more suited than the Transformer for peak load forecasting, thereby presenting a negative result for RQ 4. However, the Transformer is competitive to the established SVR and NN, and evidence on more datasets is needed to analyze whether the strong peak forecasting performance of the cINN is only achieved on the BigDEAL dataset or generalizes to other applications.

12 Peak Shaving for Energy Communities With Heat Pumps

Previous publication

This chapter is based on results from:

[244] L. Semmelmann^a, M. Hertel^a, K. J. Kircher, R. Mikut, V. Hagenmeyer, and C. Weinhardt. “The impact of heat pumps on day-ahead energy community load forecasting”. *Applied Energy*, 2024, DOI: 10.1016/j.apenergy.2024.123364.^b

^a Matthias Hertel developed the Transformer model. Leo Semmelmann developed the other models and the peak shaving application, which are described in his dissertation [243].

^b Licensed under a Creative Commons Attribution 4.0 International License, <https://creativecommons.org/licenses/by/4.0/>.

In order to reduce greenhouse gas emissions from the building sector and thereby help combating climate change, the heating supply in residential buildings is increasingly electrified by the adoption of heat pumps. This results in a change in the electrical load patterns, as the heat pumps cause a large share of a household’s energy consumption during cold days. In order to use locally produced electricity and reduce the need for grid reinforcements, battery storage systems can be used to reduce the daily peak electricity demand of a household. Local energy communities of multiple households can trade or share energy between them and operate a storage system together. This chapter evaluates the Transformer on data from such an energy community and evaluates the money saved by the energy community when relying on Transformer-based forecasts compared to forecasts from other models.

12.1 Experimental Setup

This section describes the dataset, the forecasting models and the peak shaving application.

12.1.1 Dataset

The energy community dataset described in Chapter 4 is used for the experiments. In addition to the past load, the dataset is enriched with binary features encoding the type of day (weekend, weekday, holiday), cyclical features for the hour of day and month, the ambient temperature, rolling averages of the ambient temperature with windows of 24 and 48 hours, the average load at the same hour of day over the last week, wind speed, and humidity.

Three variants of the dataset are used: a variant comprising only the household load (HH), a variant comprising only the heat pump load (HP), and a variant comprising the combined load from the households and heat pumps (Comb).

The dataset comprises two years of data, ranging from the beginning of 2019 until the end of 2020. For the hyperparameter optimization, the first half of 2019 is used for training, and the second half of 2019 for validation. For the final evaluation, a monthly rolling evaluation is performed. This means that separate models are trained for each of the 12 months of 2020, each of them using the previous 12 months as training data.

12.1.2 Forecasting models

A Random Forest, an LSTM network and a Transformer are used as forecasting models.

Each model is evaluated in two variants: (1) directly predicting the total load of the energy community, and (2) as a Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN) model. The CEEMDAN decomposes the time series into multiple component time series. For each component, a separate model is trained that gets the past values of the component as input and forecasts that component. To generate a load forecast from the component forecasts, the sum of the component forecasts is computed.

The model and training hyperparameters are optimized using Bayesian Optimization with wandb [35]. The selected hyperparameters for the Transformer model on the household load (HH), the heat pump load (HP), and the total load (Comb) are given in Table 12.1.

Table 12.1: Results of the Bayesian hyperparameter optimization process for the Transformer on the household (HH) dataset, the heat pumps only (HP) and the total load (Comb)

Model	Parameter	Values	HH	HP	Comb
Transformer	n_layers	(1,6)	1	1	1
	d_model	[16,32,64,128,256,512,1024]	128	1024	128
	n_heads	[1,2,4,8]	8	1	2
	batch_size	[16,32,64,128,256,512]	16	64	16
	learning_rate	(0.0001,0.1)	0.0008	0.0001	0.0014

12.1.3 Peak shaving application

To evaluate the value of the forecasts in a down-stream application, a battery management system is simulated for the energy community. The goal is to reduce the annual peak load of the energy community by charging and discharging a battery based on the day-ahead load forecast. This can result in a lower needed capacity for the community's grid connection point and in lower charges for the community's annual peak power, thereby effectively reducing the electricity cost of the inhabitants. At the beginning

of each day, a battery schedule for that day is computed using the mixed-integer linear program solver Gurobi. The objective of the optimization is to minimize the squared sum of the predicted load plus the battery charging operation. The square operation punishes large loads or infeeds, thereby targeting the peak load more than lower loads. The optimization problem has multiple constraints:

1. The battery's maximum power cannot be exceeded.
2. The state of charge at time t is defined by the previous state of charge plus the battery charging operation at time t .
3. The state of charge is always between zero (empty) and one (maximum capacity).

A mathematical formulation of the optimization problem is given in [244]. After the solver has terminated, the optimized charging schedule for the battery energy storage system (BESS) is applied to the actual observed loads during the day. Finally, it is evaluated how much lower the annual peak, i.e. the highest load during the year, is in comparison to the annual peak load without using the BESS.

12.2 Results

This section describes the forecasting results and the peak shaving results. The loads are always positive, so the standard metrics MAPE, MAE, RMSE and R^2 can be used to evaluate forecast accuracy.

Forecasting results The forecasting results on the total household load without heat pumps are given in Table 12.2. The Transformer achieves the best RMSE and R^2 among all models. However, the Random Forest achieves a better MAPE and MAE. The Transformer does not benefit from the CEEMDAN decomposition.

The forecasting results on the total household load without heat pumps are given in Table 12.3. The best result across all metrics is achieved with a Transformer model. The Transformer benefits from the CEEMDAN decomposition in terms of RMSE, but not for the other metrics.

The results on the total load as a combination of the household load and the heat pump load are given in Table 12.4. For each model, two ways of forecasting the total load exist: (1) predicting the household load and the heat pump load separately and adding up the two forecasts to create the forecast of the total load, and (2) forecasting the combined load directly. The best forecasts are achieved with a Transformer model and the separate forecasting strategy for the MAPE, MAE and the R^2 . Only for the RMSE, the Transformer with CEEMDAN decomposition and the direct forecasting strategy is better.

Peak shaving results Using the Transformer-based forecasts in the peak shaving simulation, a peak reduction by 5.7 kW is achieved. This is about 0.7 kW more than the second-best model (the LSTM Combined), and 57% of the peak reduction that can be achieved theoretically with a perfect forecast. The full results for all models are given in [244].

Table 12.2: Household load forecasting results for the test data without heat pumps (2020).

	MAPE [%]	MAE [W]	RMSE [W]	R^2
Random Forest	12.84	959.35	1362.61	0.74
Random Forest CEEMDAN	15.51	1093.85	1481.39	0.69
XGB	13.11	969.79	1362.03	0.74
XGB CEEMDAN	14.53	1036.19	1421.26	0.71
LSTM	14.08	1041.63	1447.21	0.70
LSTM CEEMDAN	17.12	1161.97	1522.47	0.67
Transformer	13.19	965.39	1352.30	0.74
Transformer CEEMDAN	15.60	1049.55	1373.21	0.73

Table 12.3: Heat pump load forecasting results for the test data (2020).

	MAPE [%]	MAE [W]	RMSE [W]	R^2
Random Forest	52.20	2054.40	2756.48	0.88
Random Forest CEEMDAN	97.54	2985.40	3897.60	0.76
XGB	49.17	2064.44	2817.45	0.88
XGB CEEMDAN	70.06	2617.10	3458.96	0.81
LSTM	70.51	2240.22	2908.37	0.87
LSTM CEEMDAN	49.32	1860.33	2482.39	0.90
Transformer	33.03	1602.49	2280.83	0.92
Transformer CEEMDAN	47.93	1690.77	2226.14	0.92

Table 12.4: Forecasting results for the whole energy community, including household and heat pump loads, for the test data (2020).

	MAPE [%]	MAE [W]	RMSE [W]	R^2
Random Forest: Separate	16.79	2377.94	3137.00	0.87
XGB: Separate	16.48	2375.88	3168.89	0.87
LSTM CEEMDAN: Separate	16.47	2314.98	3029.79	0.88
Transformer: Separate	13.43	2014.11	2743.59	0.90
Transformer: Combined	13.76	2062.61	2776.97	0.90
Transformer CEEMDAN: Separate	16.16	2150.27	2754.22	0.90
Transformer CEEMDAN: Combined	14.95	2089.65	2736.40	0.90

12.3 Conclusion

This chapter evaluates the Transformer in a peak shaving application for an energy community equipped with heat pumps and battery storage. The Transformer outperforms the compared methods in terms of forecast error, and the better forecasts translate to a more effective peak load reduction in the down-stream battery scheduling application. Since energy communities pay for their annual peak load, a more effective peak load reduction reduces this cost and enables faster amortization of the battery installation cost. Thus, this chapter provides positive evidence for RQ 4 on the effectiveness of Transformer-based forecasts in down-stream applications.

In the future, the Transformer can be extended to a probabilistic model, and battery dispatch algorithms considering the uncertainty in the prediction can be evaluated. An evaluation of the cINN presented in Chapter 11 for the energy community peak shaving application would allow to assess whether the results from the BigDEAL challenge generalize to another peak load forecasting and peak shaving application.

Part VI

Conclusion

13 Summary

The ongoing transition to renewable energy requires accurate electrical load forecasts for applications on all levels of the power grid, ranging from energy management systems for individual clients via state estimation via demand shifting in the distribution grid to balancing supply and demand in the transmission grid. Explainability of load forecasting models is key to foster user trust, develop robust models, and fulfill transparency requirements such as those posed in the EU’s AI Act [78]. This dissertation analyzes the usage of time-series Transformers to enable accurate and explainable load forecasting.

Part II For scenarios in which multiple load time series are recorded, such as measurements from smart meters or sensors installed in substations, the global training strategy is essential to achieve accurate load forecasts with Transformers. Zeng et al. [296] showed that linear models are better than the multivariate Transformers used in related work [292, 300, 301], thereby questioning the effectiveness of Transformers for time series. Our results show that with the global training strategy, Transformers are effective for electrical load forecasting, outperforming linear models and multivariate models. The global Transformer is able to handle heterogeneous data from time series with different characteristics. Cluster-specific model training does not improve the results, despite the fact that the clustered data is more homogeneous. An analysis of the attention weights per cluster reveals that the attention mechanism allows the model to adapt to time series with different characteristics, such as with or without a special weekend pattern. This finding motivates the usage of Transformers for Time-Series Foundation Models (TSFMs) [167], which extend the global training strategy to training on large numbers of time series from different domains to achieve generalized forecasting capabilities.

Part III Time-series Transformers are more accurate than established load forecasting methods across grid levels. In Part III, ten forecasting methods are compared on three datasets, representing the client level, the LV feeder level, and the TSO level. Time-series Transformers reduce the forecast error by 6.6–10.7% compared to the best non-Transformer model. Architectural modifications, including patching, CNN layers, LSTM layers, max pooling and sparse attention, are not needed to achieve strong forecast performance and even degrade forecast performance due to a larger hyperparameter optimization (HPO) search space. Repeated model retraining helps to adapt to concept drifts, and the inclusion of covariates is key to achieve the best results – aspects that are often overlooked in the time-series literature. The TSFM Chronos-2 achieves competitive results on two datasets, but it has difficulties around special events in the TSO data that are not directly encoded as covariates, such as long weekends and extended Christmas holidays. Chronos-2 can be used zero-shot without training on a particular dataset, but this advantage comes at the cost of slower inference than with the global Transformer, overall resulting in comparable annual runtimes on datasets with hundreds of time series.

Part IV Three explainability methods for time-series Transformers are proposed in Part IV. An analysis of individual attention heads reveals that they have distinct functions, such as attending to most recent observations, to specific times of the day or days of the week, or to periods with high temperatures. However, this approach only highlights relevant time steps and does not inform how individual model inputs affect the prediction. To address this limitation, this dissertation introduces SHAPformer, an accurate and explainable forecasting model based on SHAP and Transformers. SHAPformer leverages attention manipulation [64] during training and inference to allow to generate forecasts based on subsets of the input features. This eradicates the need for sampling that is inherent to most SHAP algorithms, resulting in a $50\text{--}1000\times$ speedup over PermutationSHAP, while maintaining the strong predictive performance of the Transformer. The explanations from SHAPformer closely resemble the underlying patterns on synthetic data with known ground-truth explanations. Applied to empirical load data, the explanations match with domain knowledge and reveal relevant insights, such as the past load being the most important predictor and a special model behavior during the Christmas period. Next, the SHAPformer algorithm is extended to TSFMs. TSFMs offer the flexibility to forecast based on different contexts and covariate sets, thereby allowing to calculate SHAP values efficiently without sampling. The resulting explanations for Chronos-2 on electrical load data show that recent observations influence the predictions most and changes in the covariate values let the model adapt to changing conditions, e.g., reduced behind-the-meter PV generation due to weaker solar radiation.

Part V In order to be valuable for the energy transition, load forecasts must be used in down-stream applications in the energy system. A peak load forecasting and a peak shaving application are analyzed in Part V. In the global BigDEAL peak load forecasting challenge, a cINN proves to be more accurate than the Transformer, resulting in a first place in two out of three tracks of the competition, and a second place in the overall ranking. Possibly, the Transformer is not beneficial due the weak autoregressive structure in the data, or generative models like the cINN might be superior for peak forecasting because they generate realistic samples instead of predicting the expected value. In a peak shaving application for an energy community equipped with heat pumps and a battery storage, the increased accuracy of the Transformer over established methods translates to more effective peak reduction.

Code and data availability Publicly available code from this dissertation allows the usage of our approaches in future research and practical applications¹. All datasets used in the dissertation are publicly available. In particular, the benchmarking datasets and the results from all approaches in Part III are made publicly available, so that newly developed methods can be compared to our results in the future.

Short summary Overall, this dissertation demonstrates that time-series Transformers are highly effective for electrical load forecasting on a variety of datasets, and the attention mechanism offers a way to calculate SHAP explanations efficiently, thus improving model transparency over mere attention visualization approaches.

¹ Training strategies: <https://github.com/KIT-IAI/transformer-training-strategies>; YAformer and benchmarking: <https://github.com/hertelm/load-forecasting-benchmark>; SHAPformer: <https://github.com/KIT-IAI/SHAPformer>; SHAP for TSFMs: <https://github.com/KIT-IAI/SHAP4TSFMs>.

14 Limitations and Outlook

Limited evidence This dissertation provides evidence for Transformers being more accurate than established forecasting models on a variety of datasets, as Transformers outperform the comparison methods on six out of eight datasets used throughout the dissertation. Transformer-based models perform best on all variants of the Electricity dataset used in various chapters, on the Ausgrid solar home dataset used in Chapter 5, on the synthetic dataset used in Chapter 9, on the TransnetBW dataset and the FeederBW dataset used in Chapter 7, and on the energy community dataset used in Chapter 12. However, there are also opposite results, as the Transformer is outperformed by local multilayer perceptrons (MLPs) on the Ausgrid substation dataset used in Chapter 6 and by a conditional Invertible Neural Network (cINN) on the BigDEAL dataset used in Chapter 11. Such inconsistencies are in line with previous findings that there is no single best-performing method and the results depend on dataset characteristics [156]. Given that the TSFM Chronos-2 outperforms a Transformer trained on a smaller variant of the TransnetBW dataset in Chapter 10, we expect that TSFMs are especially useful in setups with little training data. Future work can extend the benchmarking to more datasets with varying characteristics and sizes to provide more evidence for the model comparison.

Method selection limitations We took care to select the most relevant comparison methods for benchmarking, including statistical methods, machine learning methods, and deep learning methods. However, the fields of time-series forecasting in general and electrical load forecasting in particular are large and a variety of methods exist. Methods that proved to be competitive in related work that we did not compare with include GAMs, SVR, kNN, DeepAR [234], and Profile Neural Networks [106]. Future work can evaluate new methods on our benchmark, including the recently introduced TiDE [59], TSMixer [73], xLSTM [31], MAMBA [280], and cINN [105]. Evaluations of the cINN on more datasets are desirable to analyze whether its success in the BigDEAL peak load forecasting challenge generalizes to other forecasting tasks. While our comparisons focused on model architectures, a forecasting pipeline also includes data selection, preprocessing and postprocessing. We took care to select relevant input features, including past observations, calendar features, and weather features, but only limited feature selection and feature engineering was done. We cannot rule out that feature selection, feature engineering, or other preprocessing such as weather data transformations [198], instance normalization [147], target transformation [11] or time series decomposition [151, 301] could improve some models more than others. We did hyperparameter optimizations for all models in Chapter 7, but there is no guarantee that the search converges to the optimal hyperparameters after 100 trials. We benchmarked single models, which can be combined to model ensembles in the future, potentially leading to better results overall. To address these limitations of the evaluations in the dissertation, we publish our benchmarking datasets and results to allow for future comparisons of other methods. In addition, we plan to integrate the Transformer and

the Transformer-based TSFM Chronos-2 into the *Energy-Arena* [149] for continuous benchmarking on operational data from ENTSO-E.

Transformer architecture limitations There are architectural modifications that may improve the Transformer. We tested patching, CNN layers, LSTM layers, LogSparse attention, and max pooling in Chapter 7. We patched in the time dimension, but patching can also be done per covariate [11]. Next to LogSparse attention, ProbSparse attention and attention based on autocorrelation are used in the literature. We introduced the CNN layers before the first Transformer layer, whereas convolutional attention [165] uses CNN layers to compute the keys and queries of the attention mechanism. Other architectures perform decompositions after every layer [301]. While our experiments show that the standard Transformer performs well and is able to use local context as well as to select relevant inputs (i.e., it is not improved by the CNN layer and sparse attention which are designed for these purposes), we cannot rule out that other architectural modifications improve the forecast error.

Experimental setup limitations The forecast errors achieved in this dissertation are lower than what can be expected in real applications. We assumed that recent observations of the electrical load are available immediately, which might not be the case in practice due to delayed data transmission. In addition, we used weather estimates from reanalysis models or short-term weather forecasts as model inputs, which are not available in reality when the forecast is made. Assuming such perfect weather forecasts is common in the literature when no historical weather forecasts are available for backtesting models [100]. However, the inaccuracy and uncertainty of real weather forecasts will translate to increased load forecast errors in practice. All datasets used throughout the dissertation are in hourly resolution, and sub-hourly forecasts will result in larger errors due to increased noise.

Potential for probabilistic forecasts Electrical loads depend on human behavior which cannot accurately be predicted. While stochastic effects often cancel out in highly aggregated data such as TSO loads, they limit the predictability of less aggregated data. Therefore, the achieved errors of the best models on the FeederBW dataset and the Electricity-287 dataset in Chapter 7 are large in relation to the mean load (i.e., the best nMAE on the two datasets is 19.2 % and 7.9 %, respectively, see Table B.3). Probabilistic forecasts can reflect the uncertainty in the electrical load. This is particularly relevant for risk-managing applications such as peak forecasting [265]. The Transformer and the other deep learning models evaluated in this dissertation can be extended to probabilistic forecasting by adapting their output layer to predict quantiles and training the models with the pinball loss. Thereby, the benchmarking from Chapter 7 can be extended to probabilistic models in the future. The explanation methods from Part IV can not only be used to explain the predicted expected value, but can be extended in the future to explain the sources of forecast uncertainty.

Runtime of SHAP approaches The runtime of the algorithm to compute exact SHAP explanations with SHAPformer and TSFMs scales exponentially with the number of feature groups, which makes it costly when many groups of past observations or many covariates are considered. Instead of evaluating

the models on all coalitions of feature groups, a more efficient algorithm could estimate the SHAP values from a subset of the coalitions. In future works, these methods can be used to explain other forecasts than load forecasts, and to explain the uncertainty of probabilistic models [200, 211].

Future work on accurate and explainable TSFMs The training strategies from Part II are not only relevant for training models from scratch, but also for fine-tuning TSFMs. Future work can evaluate whether the best results are achieved with fine-tuning locally (i.e. per time series) or globally (i.e. per dataset). In addition, we identify several promising research directions at the intersection of XAI with TSFMs. Applying our methodology to other energy forecasting tasks – including low-voltage load forecasting, price forecasting, and renewable energy forecasting – would demonstrate the generalization ability of our approach across diverse operational contexts and lead to insights into covariate-dependencies of these parameters. Since our approach essentially only requires models to accept masked inputs, a capability inherent to most TSFM architectures, it is largely model-agnostic and can be easily generalized to other foundation models. Applying our SHAP adaptation to other TSFMs, both univariate (e.g. [17]) and covariate-informed (e.g. [54]), would therefore provide broader insights into how different architectural choices affect feature attribution patterns in TSFMs. Computational efficiency could be substantially improved by estimating SHAP values from subsets of feature group combinations rather than exhaustively evaluating all 2^N coalitions – a particularly important consideration for slower TSFMs and scenarios with larger feature sets. Extending our framework to explain the uncertainty in probabilistic forecasts would address a critical gap for risk-sensitive operational decisions [200]. Finally, analyzing how fine-tuning affects feature attribution patterns could reveal what domain-specific knowledge is acquired during adaptation and inform best practices for model deployment in specialized forecasting contexts.

Future work on Transformer-based forecasts and their applications The peak shaving application in Chapter 12 shows that Transformer-based load forecasts can be beneficial for down-stream applications, but a more accurate forecast does not always translate to better down-stream results [142, 284]. Therefore, more studies are needed to evaluate the value of Transformer-based load forecasts for the applications listed in Chapter 1. The Energy Lab [102] facilitates research on load forecasting applications before deploying them on a large scale. Future work may apply time-series Transformers to other energy time-series tasks, e.g., to model building dynamics [220].

Outlook Overall, this dissertation lays the foundation for evaluations of electrical load forecasting models on a large benchmark representing various grid levels, demonstrating improved performance through a Transformer trained with a global strategy. This benchmark can be extended to more datasets, new models, and probabilistic forecasts in future work. In addition, two XAI methods for Transformer models and TSFMs are proposed to enhance model transparency in future research and practical applications.

Appendix

A Clustering

Reduced dataset size of Electricity-328¹

The results on the two datasets differ, in the sense that the global MLP and global Transformer are better than the best local model on the Electricity-328 dataset, but not on the Ausgrid substation dataset. A possible explanation is that global models benefit from more training data, and are therefore better on the Electricity-328 dataset, which contains about twice as many time series as the Ausgrid substation dataset. To test whether the larger size of Electricity-328 is the reason for the better results of the global models on this dataset, we evaluate the global models on smaller variants of the Electricity-328 dataset. We create two subsets of Electricity-328, each of which contains half of the time series, while maintaining the relative sizes of the three clusters. A global MLP and a global Transformer are trained and evaluated on each subset. The final results shown in Table A.1 are obtained by averaging the results from the two subsets. We observe that the results on the smaller datasets are almost identical to the results on the full dataset. Therefore, the different dataset sizes do not explain the different results on the two original datasets.

Table A.1: Forecasting results [% nMAE] on the Electricity-328 test set with half and full dataset size. Table from [109], © 2025, IEEE.

Dataset	Global MLP	Global Transformer
Electricity-328 full	7.23	6.91
Electricity-328 half	7.23	6.92

Clustering algorithms²

Next, we analyze whether the choice of the cluster algorithm affects the results for the cluster-specific training strategy. The results of cluster-specific models based on the different clustering algorithms are shown in Table A.2. We note that the forecasting results are almost identical, irrespective of the chosen clustering algorithm. A visual inspection of the cluster centers and cluster members shows a high similarity of the resulting clusterings, which explains the almost identical forecasting results.

¹ This subsection is an excerpt from Hertel et al. [109], © 2025, IEEE.

² This subsection is an excerpt from Hertel et al. [109], © 2025, IEEE.

Table A.2: nMAE results with the cluster-specific strategy using different clustering algorithms and linear regression (LR), multilayer perceptron (MLP) and Transformer (Tr.) models. Table from [109], © 2025, IEEE.

Algorithm	Electricity-328			Ausgrid solar home		
	LR	MLP	Tr.	LR	MLP	Tr.
K-means EUC	8.60	7.15	6.97	8.02	6.74	7.13
K-means DTW	8.63	7.15	7.03	8.01	6.76	7.22
K-shape	8.55	7.15	7.01	7.99	6.77	7.16
FCM	8.62	7.17	7.05	8.05	6.75	7.16

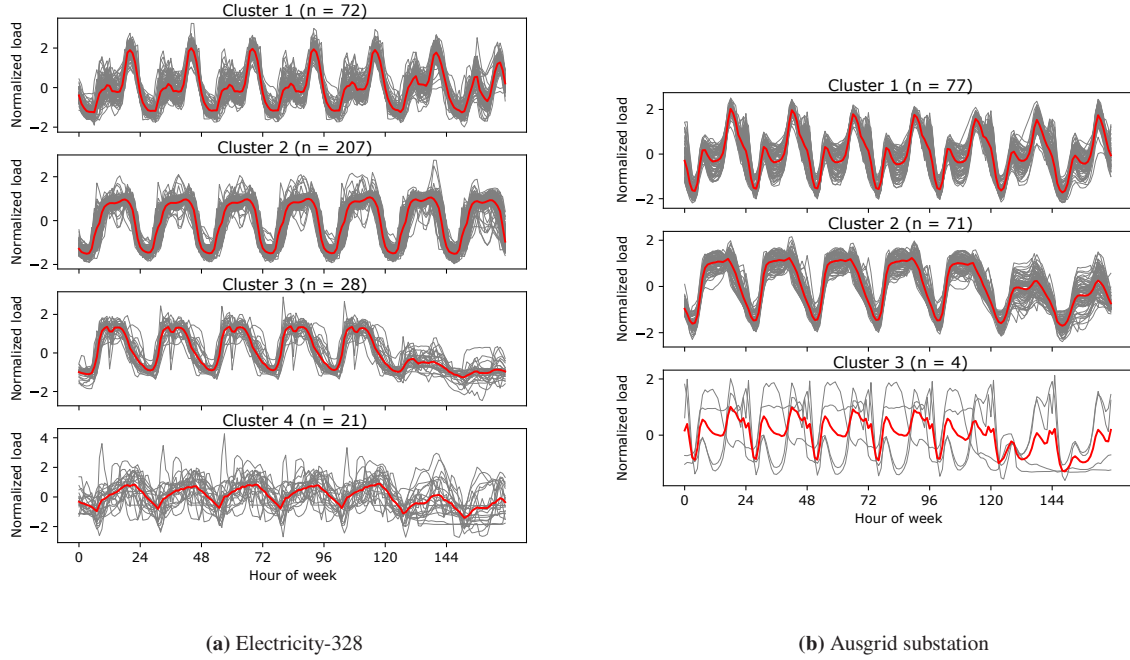


Figure A.1: Resulting clusters from FCM on the two datasets. The cluster centers are red and the AWLDs of the cluster members gray. The cluster on the bottom is the noise cluster.

Results per cluster

If different models are best on the different clusters, the overall results can be improved by selecting the best model per cluster and using it to forecast the time series belonging to that cluster. To test if this strategy results in improved forecasts, the models are evaluated on a per-cluster basis in the following. This is done using the FCM clustering result, which has the advantage that time series that differ from the rest are filtered out and put into a noise cluster. For the noise cluster, no cluster-specific model is trained, because the time series in that cluster are inhomogeneous. Instead, the global model is used to forecast the time series belonging to the noise cluster.

The clustering results of FCM are shown in Figure A.1. The cluster centers are similar to those resulting from the k-means algorithm. However, with FCM a noise cluster emerges, containing 21 time series for the Electricity-328 dataset and four time series for the Ausgrid substation dataset.

Table A.3: Forecasting results [% nMAE] per cluster on the Electricity-328 test set. The clusters are shown in Figure A.1.

Model	Cluster 1	Cluster 2	Cluster 3	Cluster 4
LR local	9.54	6.23	14.01	16.41
LR cluster	9.85	6.31	14.42	19.41
LR global	9.98	6.61	15.61	19.41
MLP local	8.82	5.84	11.39	13.24
MLP cluster	8.67	5.56	10.40	13.60
MLP global	8.71	5.62	10.56	13.60
Transformer local	10.36	7.16	13.51	14.81
Transformer cluster	8.86	5.48	10.06	12.28
Transformer global	8.53	5.44	9.82	12.28

Table A.4: Forecasting results [% nMAE] per cluster on the Ausgrid substation test set. The clusters are shown in Figure A.1.

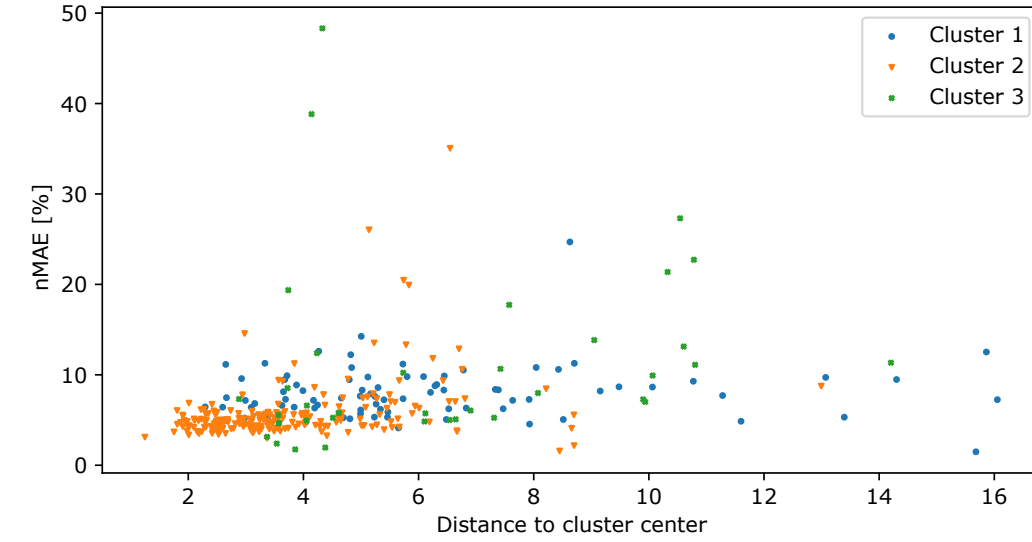
Model	Cluster 1	Cluster 2	Cluster 3
LR local	8.86	6.09	15.44
LR cluster	8.92	6.52	18.25
LR global	9.10	6.63	18.25
MLP local	7.90	5.03	12.80
MLP cluster	7.98	5.02	13.74
MLP global	8.22	5.26	13.74
Transformer local	8.85	5.54	14.09
Transformer cluster	8.71	5.26	11.84
Transformer global	8.38	5.15	11.84

To analyze whether the forecasting results can be improved by selecting different forecast models for each cluster, the forecasting results per cluster are given in Table A.3 for the Electricity-328 dataset and in Table A.4 for the Ausgrid substation dataset. On the Electricity-328 dataset, the global Transformer is the best model for all clusters. On the Ausgrid substation dataset, the local MLP is the best model on cluster 1 and almost the best model on cluster 2. On the noise cluster, the global Transformer achieves a better result, but the effect on the overall results is small, as the noise cluster only contains four time series. As one would expect, the results are on average worse on the noise cluster than on the other clusters. However, an analysis of the errors per time series in the next section reveals that the time series with the largest errors do not belong to the noise cluster.

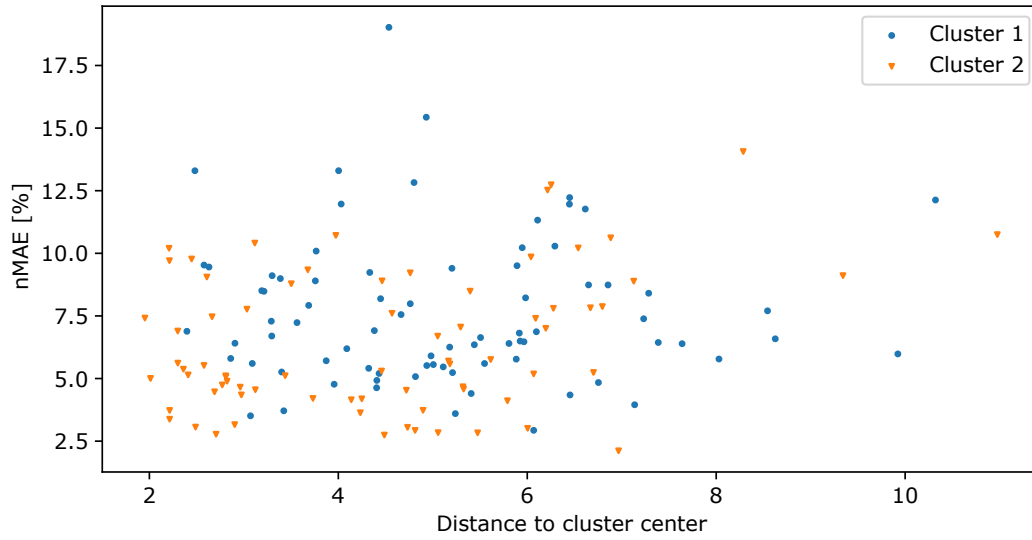
Forecast error vs. distance to cluster center

The goal of FCM is to filter out the time series that do not fit into the clusters because they comprise different patterns than the rest. FCM can thereby be used to identify unusual time series, which might be more difficult to forecast. Figure A.2 shows an analysis of the forecast error per time series in dependence

of the distance of the time series' AWLD to the closest cluster center resulting from FCM. On both datasets, the time series with the largest errors are not the ones with the largest distance to the closest cluster center, which fall into the noise cluster.



(a) Electricity-328



(b) Ausgrid substation

Figure A.2: Analysis of forecast error vs. distance to the closest cluster center.

B Benchmarking

Hyperparameters

The model sizes and the hyperparameters of the best configurations found with the HPO are given in Tables B.1 and B.2.

Table B.1: Number of trainable parameters per model on the three datasets.

Model	TransnetBW	FeederBW	Electricity-287
Linear	940,128	456,288	940,128
MLP	2,966,624	14,127,200	6,457,440
CNN	8,263,904	7,710,304	3,692,768
LSTM	13,855,456	117,325,408	117,327,456
N-HITS	7,308,894	48,654,500	24,865,431
TFT	195,828	696,810	741,748
Transformer	1,078,657	3,304,961	279,361
YAformer	4,350,849	2,823,297	8,608,769

Additional metrics

Results for four metrics on the test sets of the three benchmarking datasets are given in Table B.3.

Statistical tests

Results for a one-sided paired permutation test and a one-sided paired t-test comparing the best model with each other model are reported in Table B.4. Monthly MAEs are used as samples for TransnetBW, monthly MAEs of every time series are used as samples for FeederBW, and nMAEs per time series are used as samples for Electricity-287, thereby respecting the retraining schemes of the three datasets.

Table B.2: Model hyperparameter search range and the best configurations found with a Bayesian Optimization HPO for each dataset.

Hyperparameter	Search range	TransnetBW	FeederBW	Electricity-287
<u>Linear</u>				
Context length C	168, 336, 720	720	336	720
Batch size	128, 256, 512	512	512	512
Learning rate	(0.0001, 0.001)	0.00012	0.00010	0.00017
Dropout	(0.0, 0.2)	0.15	0.08	0.05
<u>MLP</u>				
Context length C	168, 336, 720	336	336	336
Layers	1, 2, 3	2	2	2
Units	128, 256, 512, 1024, 2048	512	2048	1024
Batch size	128, 256, 512	512	256	512
Learning rate	(0.0001, 0.001)	0.00019	0.00018	0.00012
Dropout	(0.0, 0.2)	0.00	0.12	0.09
<u>CNN</u>				
Context length C	168, 336, 720	336	168	336
Patch size	1, 4, 8, 24	8	24	24
Kernel width	3, 5, 7	3	7	5
Convolutional layers	1, 2, 3	1	1	1
Max pooling width	1, 2, 4, 8	1	1	1
Dense layers	1, 2, 3	2	1	2
Dense units	128, 256, 512, 1024	1024	512	1024
Batch size	128, 256, 512	256	128	512
Learning rate	(0.0001, 0.001)	0.00079	0.00016	0.00014
Dropout	(0.0, 0.2)	0.00	0.09	0.12
<u>LSTM</u>				
Context length C	168, 336, 720	720	720	720
Architecture	encoder, encoder-decoder	encoder	encoder	encoder
Patch size	1, 4, 8, 24	4	4	4
LSTM layers	1, 2, 3	2	3	3
d_{model}	32, 64, ..., 1024	128	512	512
Dense units	128, 256, 512, 1024	512	1024	1024
Dense layers	1, 2, 3	1	1	1
Batch size	128, 256, 512	256	128	128
Learning rate	(0.0001, 0.001)	0.00080	0.00037	0.00012
Dropout	(0.0, 0.2)	0.19	0.04	0.08
<u>N-HITS</u>				
Context length C	168, 336, 720	168	336	720
Blocks	(1,1,1), (2,2,2), (3,3,3)	(1,1,1)	(2,2,2)	(3,3,3)
Kernel size	(1,1,1), (2,2,2), (4,4,4), (2,2,1), (8,4,1), (16,8,1)	(2,2,1)	(8,4,1)	(16,8,1)
Downsample	(1,1,1), (24,12,1), (40,20,1), (60,8,1), (168,24,1), (180,60,1)	(40,20,1)	(180,60,1)	(60,8,1)
Dense units	32, 64, 128, ..., 1024	512	1024	256
Batch size	128, 256, 512	256	512	128
Learning rate	(0.0001, 0.001)	0.00025	0.00014	0.00010
<u>TFT</u>				
Context length C	168, 336, 720	720	336	720
d_{model}	32, 64, 128, 256, 512	32	64	64
n_{heads}	1, 2, 4, 8	1	4	4
Batch size	128, 256, 512	128	128	128
Learning rate	(0.0001, 0.001)	0.00049	0.00010	0.00035
Dropout	(0.0, 0.2)	0.04	0.03	0.02
<u>Transformer</u>				
Context length C	168, 336, 720	720	720	720
n_{layers}	1, 2, 3, 4	1	3	4
d_{model}	32, 64, 128, 256, 512	256	256	64
n_{heads}	1, 2, 4, 8	1	4	4
Dense layers	1, 2, 3	1	2	1
Dense units	64, 128, 256, 512	64	256	128
Batch size	128, 256, 512	128	512	512
Learning rate	(0.0001, 0.001)	0.00028	0.00017	0.00033
Dropout	(0.0, 0.2)	0.17	0.10	0.07
<u>YAformer</u>				
Context length C	168, 336, 720	336	720	720
Patch size	1, 4, 8, 24	1	1	1
w_{conv}	1, 3, 5	5	1	5
LSTM layers	0, 1, 2	0	1	0
n_{layers}	1, 2, 3, 4	1	1	2
d_{model}	32, 64, 128, 256, 512	512	256	512
n_{heads}	1, 2, 4, 8	2	1	8
Attention	full, sparse	full	sparse	full
Max pooling width	1, 2, 4	4	1	1
Dense layers	1, 2, 3	2	2	1
Dense units	64, 128, 256, 512	128	128	256
Batch size	128, 256, 512	256	128	512
Learning rate	(0.0001, 0.001)	0.00014	0.00038	0.00056
Dropout	(0.0, 0.2)	0.05	0.11	0.10

Table B.3: Test results on the three datasets evaluated with the mean absolute error (MAE), the root mean squared error (RMSE), the normalized mean absolute error (nMAE) and the normalized root mean squared error (nRMSE). The best result is highlighted in bold.

Model	TransnetBW				FeederBW				Electricity-287			
	MAE [MW]	RMSE [MW]	nMAE [%]	nRMSE [%]	MAE [kW]	RMSE [kW]	nMAE [%]	nRMSE [%]	MAE [kW]	RMSE [kW]	nMAE [%]	nRMSE [%]
Baseline	355.3	487.1	5.49	7.53	5.51	9.24	33.5	51.8	56.1	566.5	10.35	16.79
Chronos-2	186.5	280.0	2.88	4.33	3.22	4.91	19.4	28.4	42.2	407.0	7.92	12.95
CNN	156.5	211.1	2.42	3.26	3.66	5.41	22.2	31.3	47.3	430.2	8.97	13.34
LightGBM	186.0	259.8	2.87	4.01	3.85	5.92	23.4	33.9	43.6	377.3	8.86	13.08
Linear	237.7	322.2	3.67	4.98	3.99	6.00	24.2	34.4	54.1	508.7	9.76	14.08
LSTM	163.8	217.5	2.53	3.36	3.41	5.05	20.6	29.2	48.2	453.1	9.10	13.77
MLP	166.0	222.4	2.57	3.44	3.64	5.40	22.0	31.1	47.9	438.3	8.98	13.25
NHITS	160.5	215.7	2.48	3.33	3.60	5.32	21.8	30.7	47.5	445.3	8.92	13.33
TFT	191.3	259.5	2.96	4.01	3.30	5.01	20.0	28.9	44.1	413.3	8.51	13.47
Transformer	143.6	191.7	2.22	2.96	3.18	4.82	19.2	27.8	40.6	378.9	7.99	12.56
YAformer	148.4	199.8	2.29	3.09	3.24	4.90	19.6	28.4	42.7	415.1	8.18	12.77

Model	TransnetBW		FeederBW		Electricity-287	
	PT	TT	PT	TT	PT	TT
Baseline	0.000	0.000	0.000	0.000	0.000	0.000
Chronos-2	0.000	0.000	0.000	0.000	b.m.	b.m.
CNN	0.001	0.001	0.000	0.000	0.000	0.000
LightGBM	0.000	0.000	0.000	0.000	0.000	0.000
Linear	0.000	0.000	0.000	0.000	0.000	0.000
LSTM	0.000	0.000	0.000	0.000	0.000	0.000
MLP	0.000	0.000	0.000	0.000	0.000	0.000
N-HITS	0.000	0.000	0.000	0.000	0.000	0.000
TFT	0.000	0.000	0.000	0.000	0.000	0.000
Transformer	b.m.	b.m.	b.m.	b.m.	0.201	0.191
YAformer	0.120	0.117	0.000	0.000	0.003	0.006

Table B.4: P-values of a paired permutation test (PT) and paired t-test (TT) comparing the best model (b.m.) with each other model on the three datasets. Results that are not significantly worse than the best model with $p < 0.05$ are marked in bold.

C Explainability

C.1 Transformer Explained by SHAP

This section shows the global and local explanations for the Transformer model created with PermutationSHAP and WindowSHAP on the two datasets.

C.1.1 Synthetic data

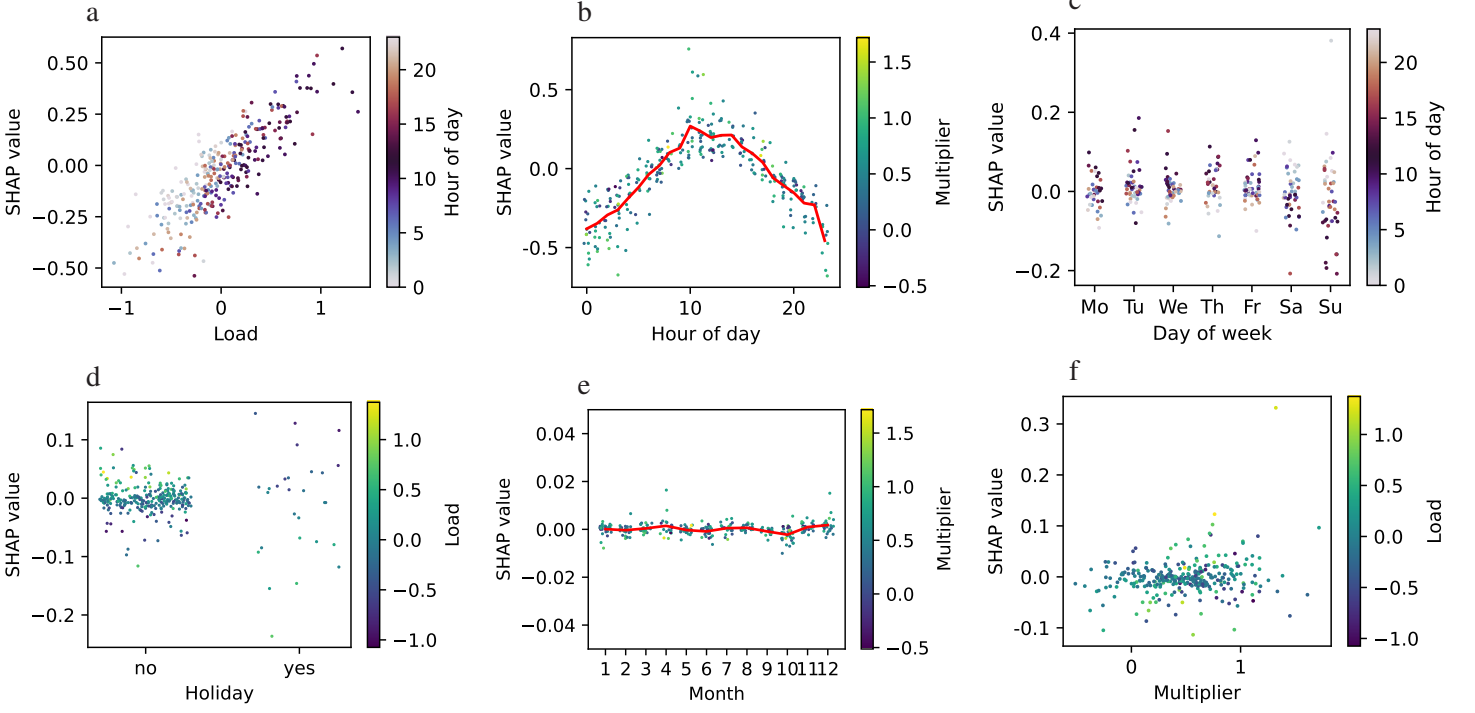
The global explanations of the Transformer model created with PermutationSHAP and WindowSHAP on synthetic data are shown in Figure C.1. The global explanations of PermutationSHAP are calculated for 300 instead of 1000 forecasts due to its long runtime. Similar patterns as the global explanations created with SHAPformer occur. Namely, a linear effect of the past load is visible in panel (a), a typical daily load curve in (b), stretching effects of the workdays and shrinking effects of the weekend days and holidays in (c) and (d), month has almost no impact as visible in (e) and the multiplicative effect of the multiplier feature is visible in (f). However, the magnitude of the SHAP values differs from the ground truth presented in Figure 9.5, which also leads to deviations in the feature importance values, as shown in Figure 9.4.

Two local explanations per approach are shown in Figure C.2. The expected effects are visible in general, such as the day-and-night effect of the hour of day feature, which repeats every day (with different magnitudes for weekend days and holidays), as well as the shrinking effects of the day-of-week feature on weekend days and holidays, and the increasing effect of the multiplier in the right-hand example. However, both explainers overestimate the effect of the hour-of-day feature and underestimate the effect of the past days.

C.1.2 TransnetBW data

The global explanations of the Transformer model created with PermutationSHAP and WindowSHAP on the TransnetBW data are shown in Figure C.3. The dependence on the past load is much less clear than with SHAPformer. However, the dependence on hour of day, day of week and holiday shows clear and interpretable patterns, such as a typical daily load curve, and lower predicted loads on the weekend and on holidays. The dependence on the month reveals that lower loads are predicted in summer months, and low loads are predicted in December – as it is also the case with SHAPformer. The dependence on the temperature shows increasing loads for cold and hot outside temperatures.

A PermutationSHAP



B WindowSHAP

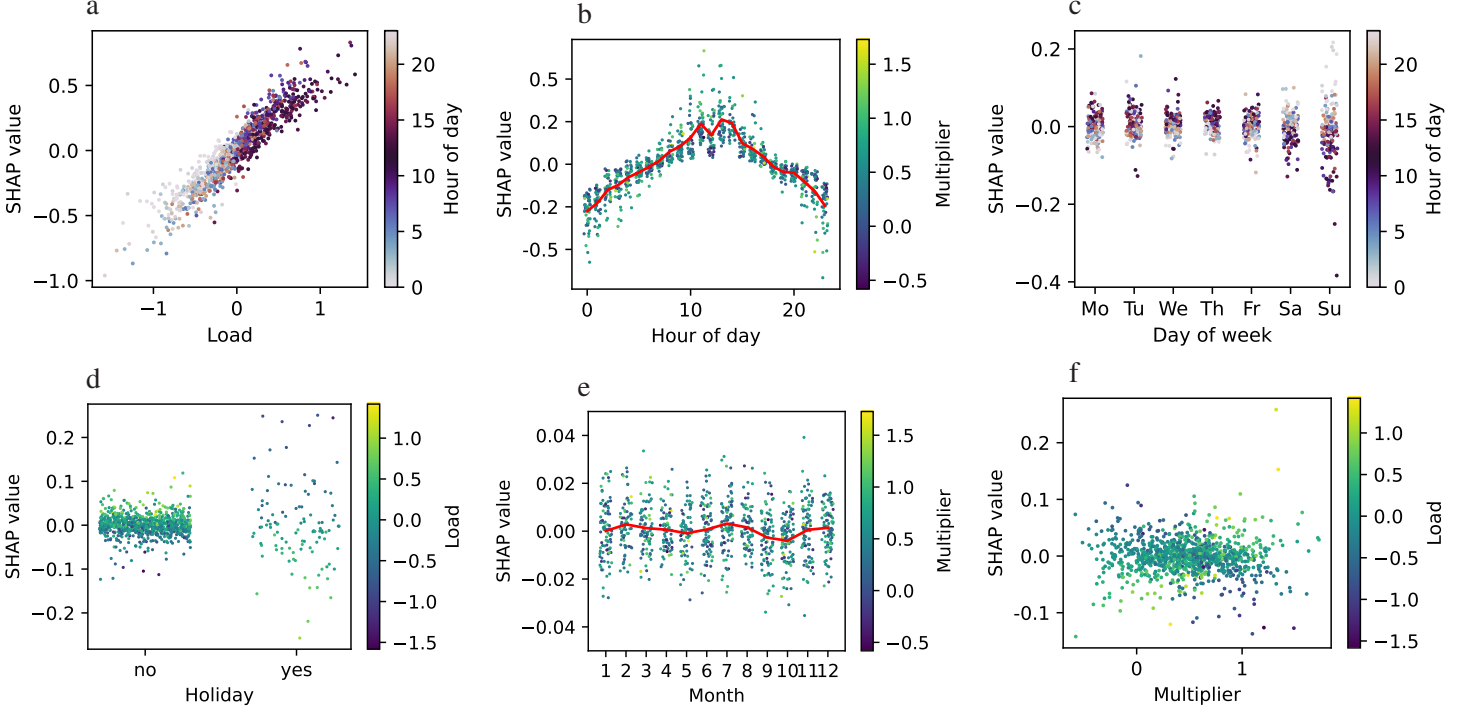
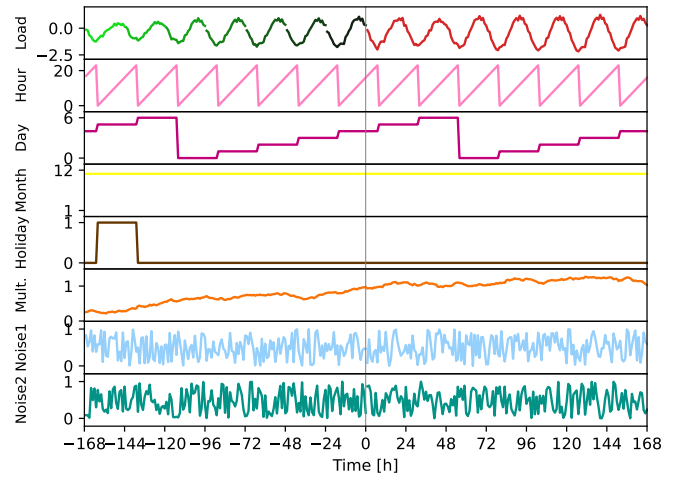
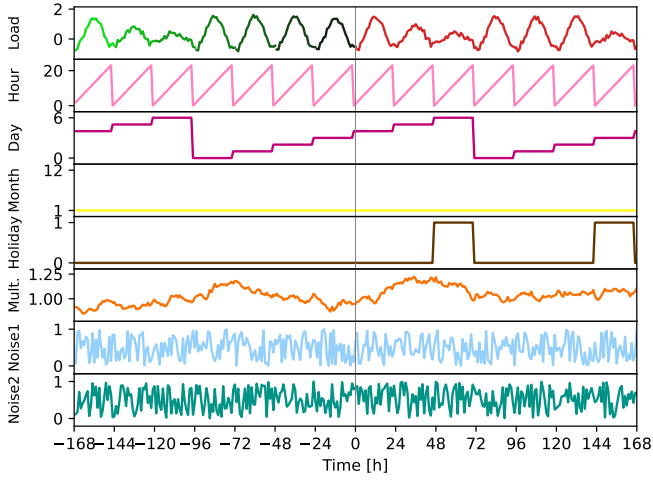
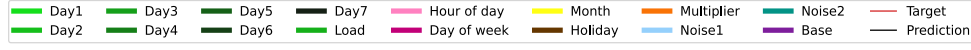


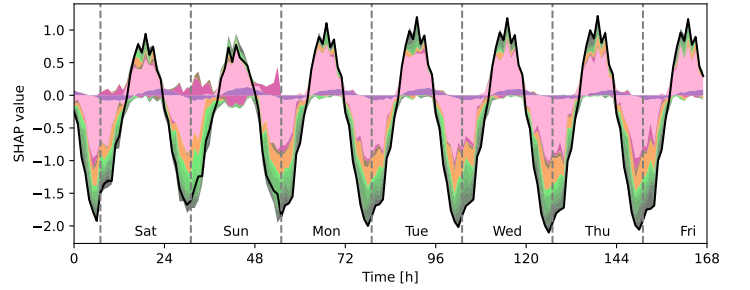
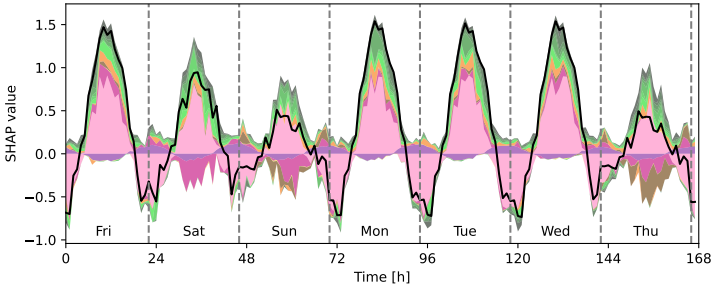
Figure C.1: Global explanations of the Transformer model on the synthetic data, generated with PermutationSHAP and WindowSHAP. Figure from [115].

A Synthetic examples

Feature groups & target



B Explanations from PermutationSHAP



C Explanations from WindowSHAP

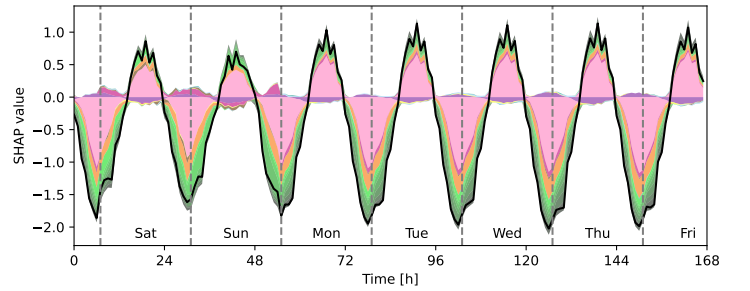
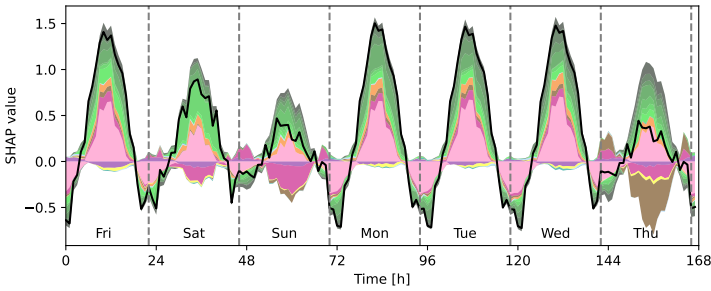


Figure C.2: Local explanations of synthetic examples, created with PermutationSHAP and WindowSHAP. Figure from [115].

Two local explanations on the TransnetBW data are shown in Figure C.4. The explanations generated with the two methods have a high similarity. On the left-hand side, weekly and daily effects are visible, as well as an increased load due to cold outside temperatures. On the right-hand side, the December effect leads to a lower predicted load. However, different from SHAPformer's explanation, the holiday feature has only a small effect.

C.2 Feature Attention

The feature attention plots are shown in Figure C.5. Note that the load feature is not present for future time steps and its attention score is therefore zero. On the synthetic data, the past load and the future multiplier are attended most. On both datasets, the day-of-week feature is attended at weekend days. Similarly, the holiday feature is attended when it is a holiday. On the TransnetBW data, the past load and precipitation are attended, as well as the future hour-of-day feature.

C.3 Attention Manipulation

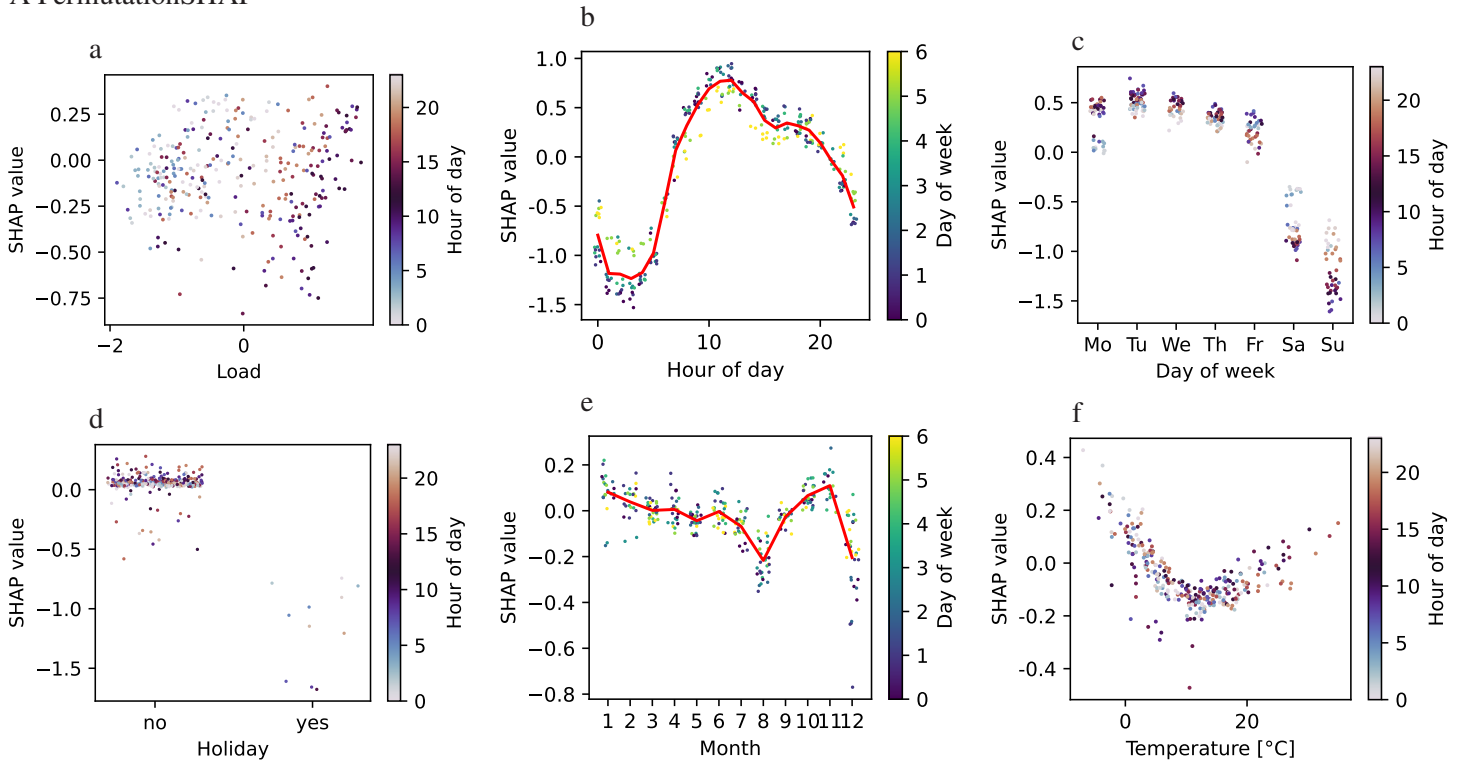
Attention manipulation uses the trained SHAPformer model, but only drops out one feature or day at a time, and evaluates the impact on the forecast. Examples are shown in Figure C.6.

- (a) Dropping out the holiday feature, the model predicts a normal workday pattern instead for the last day in the forecast horizon. Dropping out the past load leads to an average weekly diagram as the prediction, as the model can no longer estimate the base load.
- (b) Dropping out the multiplier, the model does not scale the forecast according to the increasing multiplier.
- (c) Dropping out the temperature in a cold week in December, the model predicts a lower load.
- (d) Dropping out the holiday feature in the week before Christmas, the model has no information whether it is before or after Christmas, therefore predicts a low load (it could potentially be after Christmas where the load is low). On the other hand, dropping out the month information leads to a higher predicted load. Dropping out the past load also leads to an increase in the predicted load.

C.4 Day Impacts in SHAPformer

Impacts of past weekdays on predicted weekdays are shown in Figures C.7 and C.8 for the synthetic data and the TransnetBW data, respectively. A linear effect is visible for all pairs of days for the synthetic data, with the strongest impact (importance of 0.058) for Sundays in the input on Sundays in the forecast horizon. The importance of past days is in general higher on the TransnetBW data than on the synthetic data. Here, the strongest impact is also from previous Sundays to predicted Sundays (with an importance of 0.144).

A PermutationSHAP



B WindowSHAP

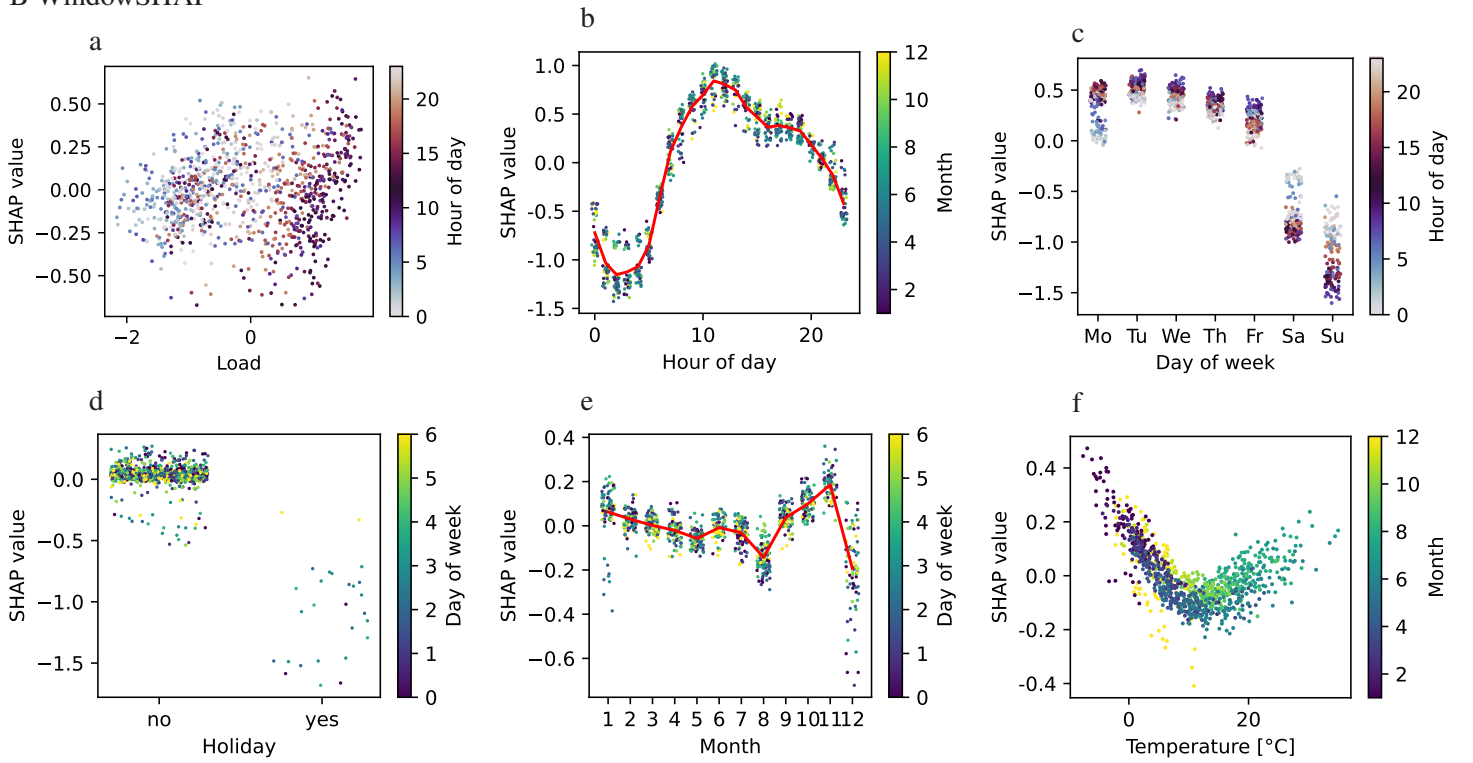
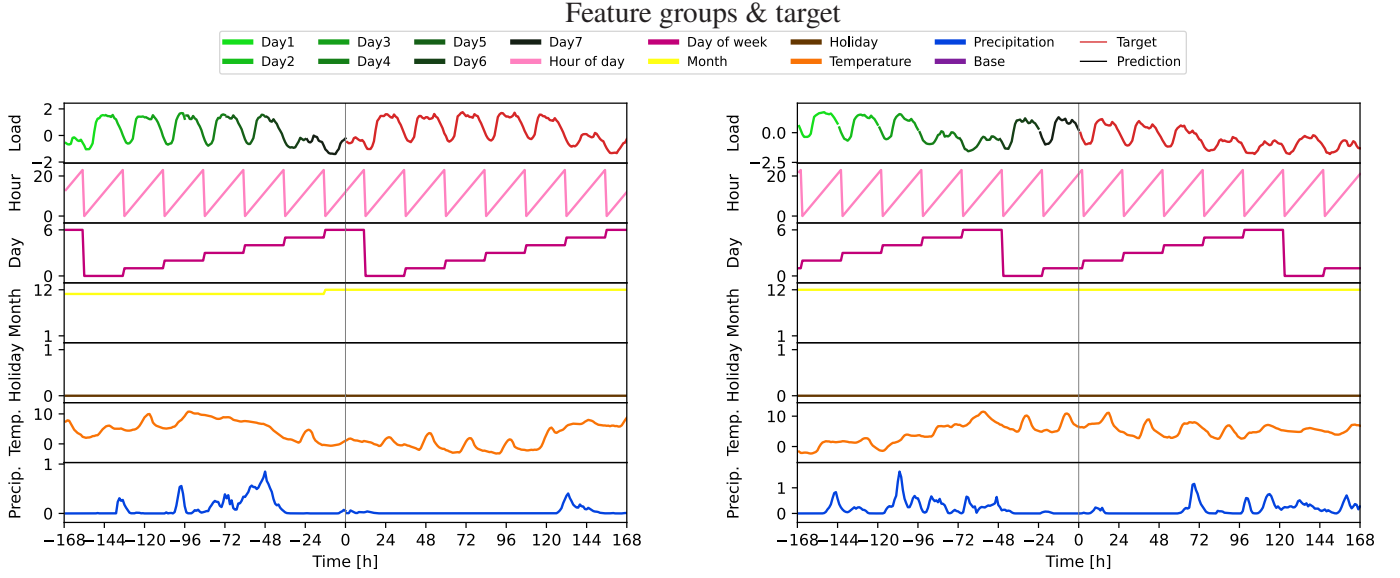
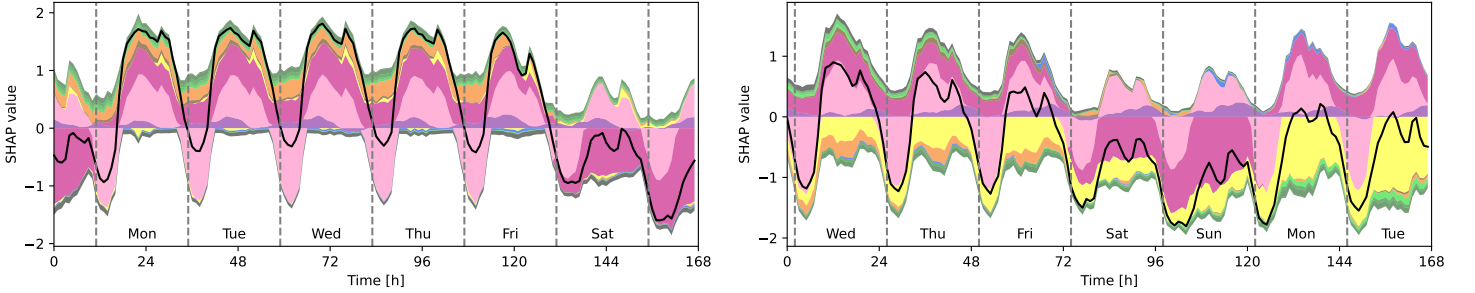


Figure C.3: Dependence plots of the Transformer model on TransnetBW data, created with PermutationSHAP and WindowSHAP. Figure from [115].

A Data



B PermutationSHAP



C WindowSHAP

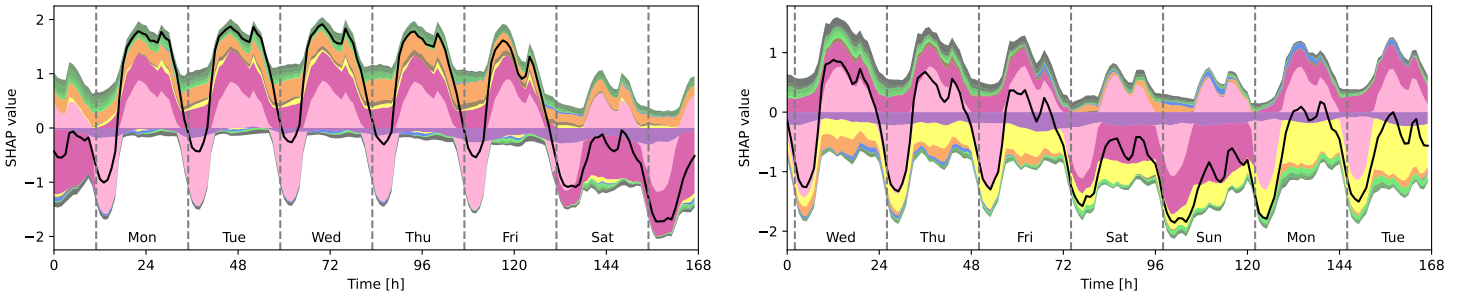


Figure C.4: Local explanations of TransnetBW examples created with PermutationSHAP and WindowSHAP for the same examples as in the main text: December 1, 2019, 13:00 (left) and December 17, 2019, 22:00 (right). Figure from [115].

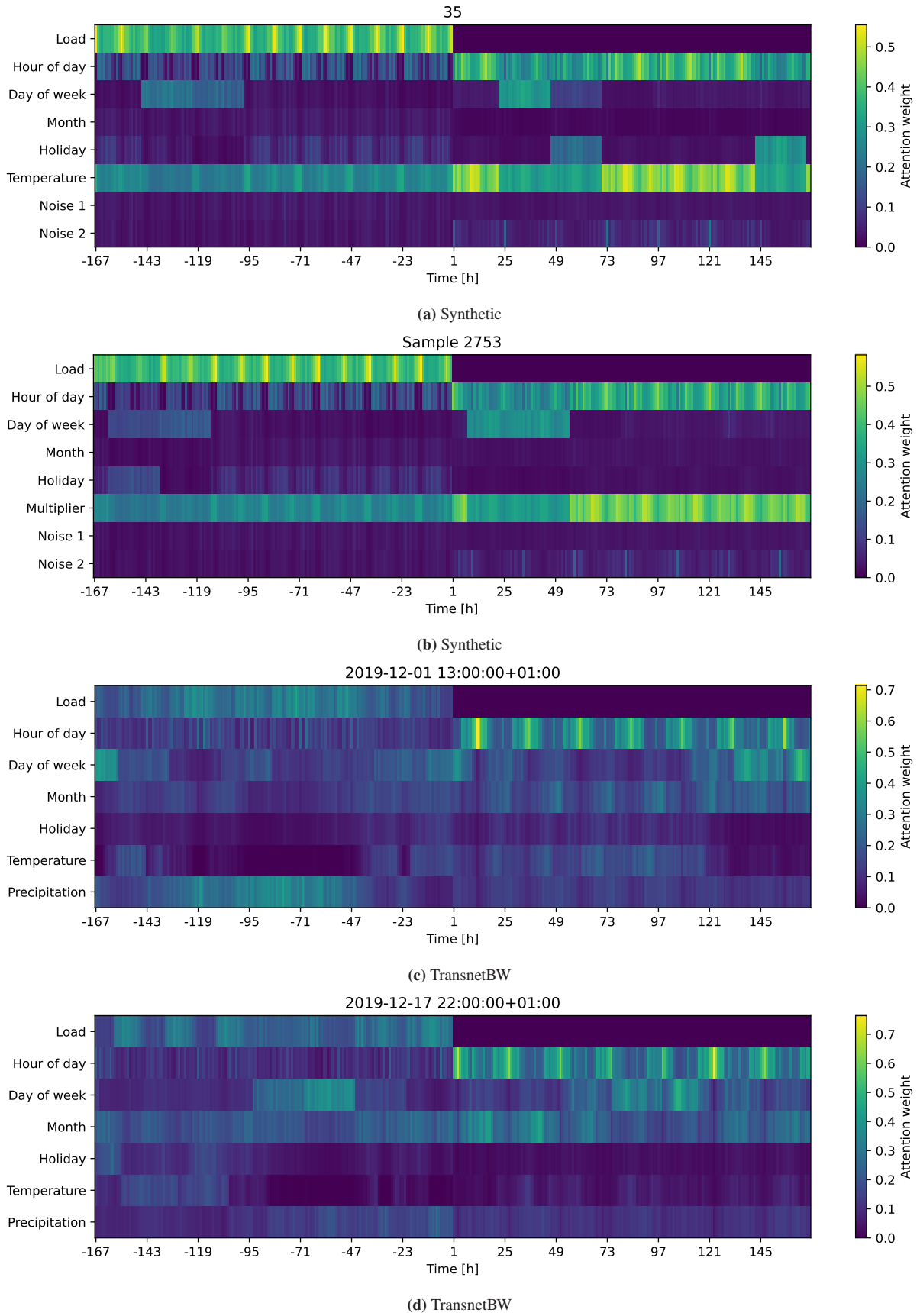
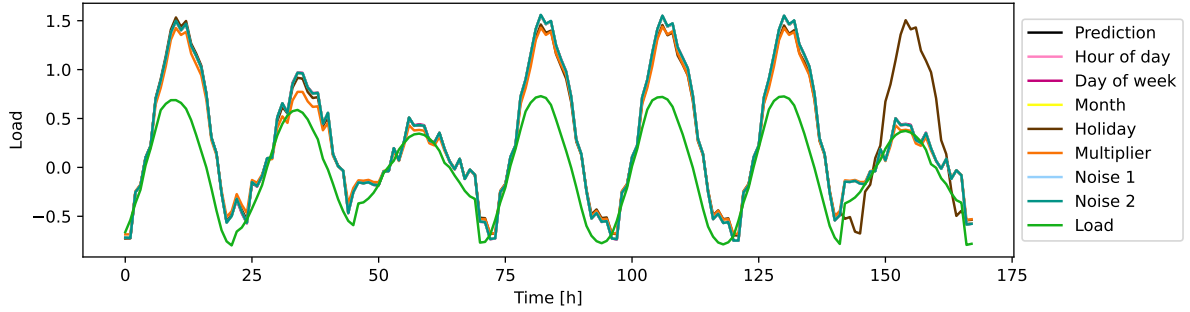
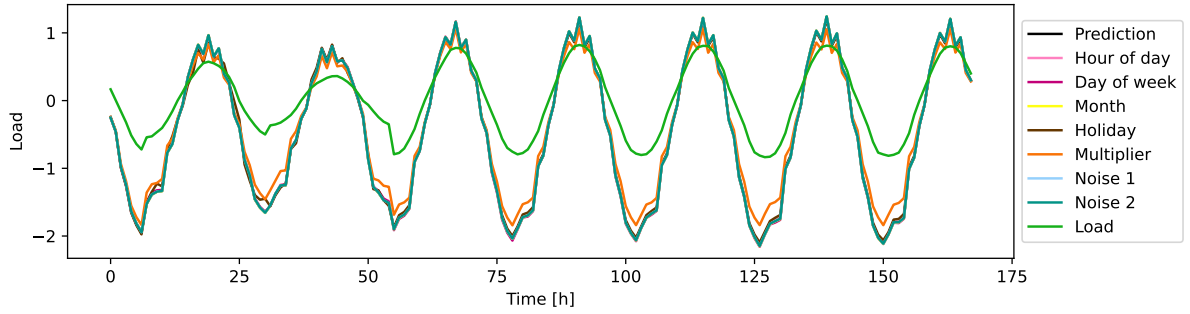


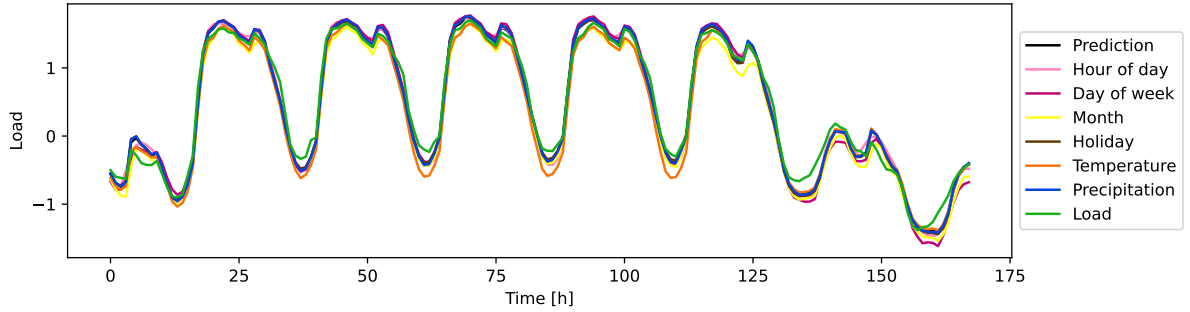
Figure C.5: Feature attention maps.



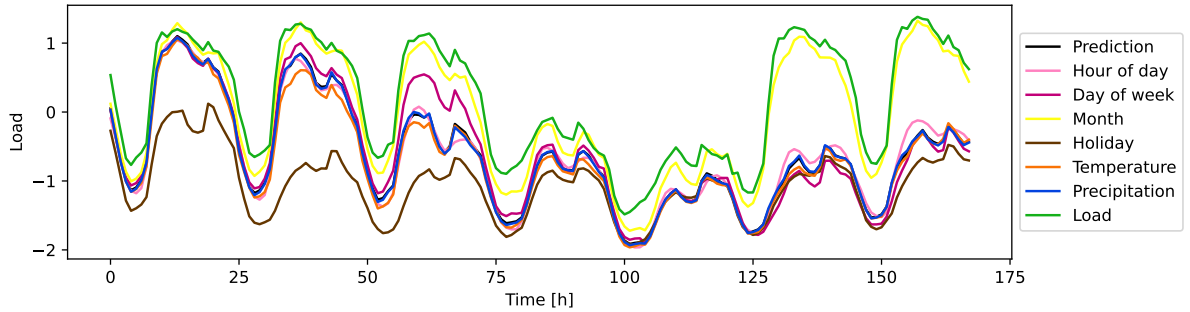
(a) Synthetic



(b) Synthetic



(c) TransnetBW: December 1, 2019, 13:00.



(d) TransnetBW: December 17, 2019, 22:00.

Figure C.6: Attention manipulation plots for the same examples as in the main text.

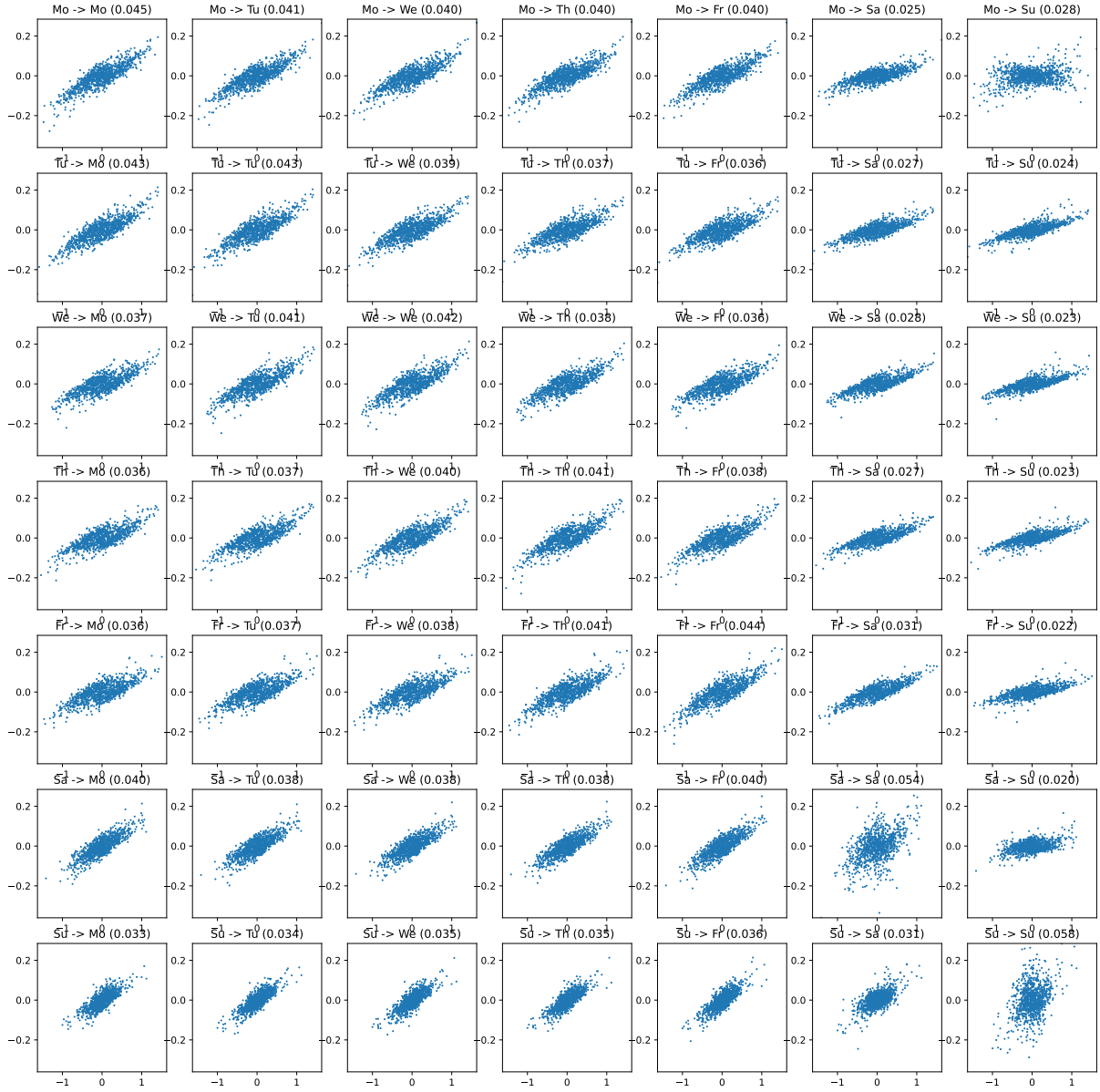


Figure C.7: Day to day impacts on the synthetic dataset. Each panel shows the impact of one day of the past week on one day of the next week, with the past load on the x-axis and the SHAP value on the y-axis.

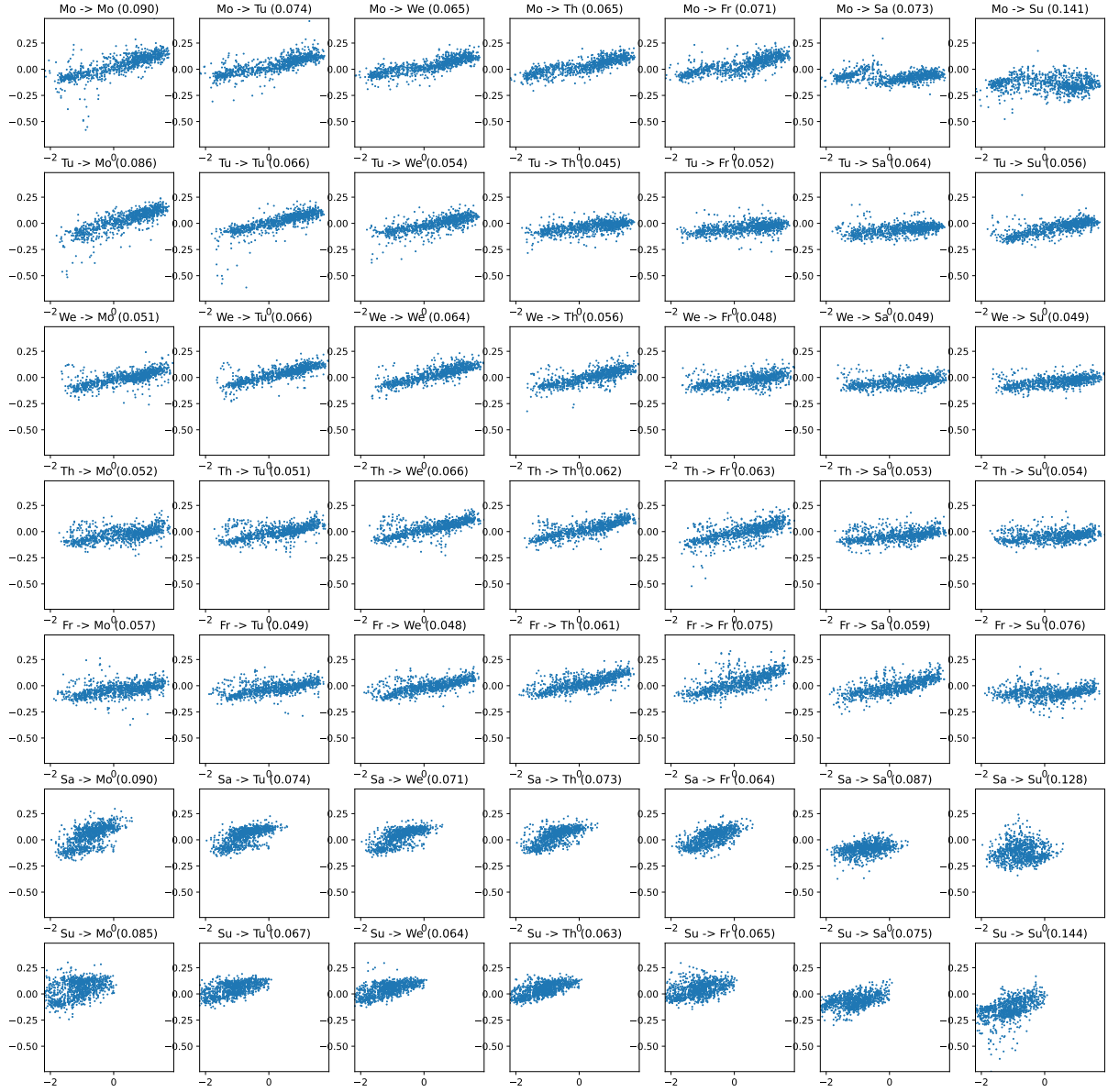


Figure C.8: Day to day impacts in the TransnetBW dataset. Each panel shows the impact of one day of the past week on one day of the next week, with the past load on the x-axis and the SHAP value on the y-axis.

Acronyms

AI	"Artificial Intelligence"	GEFCom	Global Energy Forecasting Competition
ANFIS	adaptive neural fuzzy inference system	GP	Gaussian Process
ANN	artificial neural network	GRU	Gated Recurrent Unit
ARIMA	Autoregressive Integrated Moving Average	HEMS	home energy management system
AtMan	attention manipulation	HP	heat pump
AutoML	automated machine learning	HPC	high-performance computing
AWLD	average weekly load diagram	HPO	hyperparameter optimization
BERT	Bidirectional Encoder Representations from Transformers	HVAC	heating, ventilation and air conditioning
BESS	battery energy storage system	IEEE	Institute of Electrical and Electronics Engineers
BigDEAL	Big Data Energy Analytics Laboratory	IHEPC	individual household electric power consumption dataset
BNN	Bayesian Neural Network	ISO	independent system operator
CBT	smart metering electricity customer behaviour trials	KAN	Kolmogorov Arnold Network
CEEMDAN	Complete Ensemble Empirical Mode Decomposition with Adaptive Noise	kNN	k-Nearest-Neighbor
cINN	conditional Invertible Neural Network	LDC	local distribution company
CNN	Convolutional Neural Network	LightGBM	Light Gradient-Boosting Machine
CVI	cluster validation index	LIME	Local Interpretable Model-Agnostic Explanations
DM test	Diebold–Mariano test	LR	Linear Regression
DSO	distribution system operator	LRP	layer-wise relevance propagation
DTW	dynamic time warping	LSTM	Long Short-term Memory
EnWG	Energiewirtschaftsgesetz	LV	low voltage
EU	European Union	MAE	mean absolute error
ES	Exponential Smoothing	MAPE	mean absolute percentage error
EV	electric vehicle	MLP	multilayer perceptron
FCM	fuzzy c-means	MSBG	Messstellenbetriebsgesetz
GAM	generalized additive model	MSE	mean squared error
GB	gradient boosting	NaN	not-a-number
		NDC	nationally determined contribution
		NLP	natural language processing
		nMAE	normalized mean absolute error
		nMSE	normalized mean squared error

nRMSE	normalized root mean squared error	SHAP	Shapley Additive Explanations
NN	neural network	sMAE	standardized mean absolute error
OPSD	Open Power System Data	sMSE	standardized mean squared error
ppm	parts per million	SVR	Support Vector Regression
PV	photovoltaic	TFT	Temporal Fusion Transformer
ReLU	rectified linear unit	TSFM	Time-Series Foundation Model
RF	random forest	TSO	transmission system operator
RMSE	root mean squared error	VPP	virtual power plant
RNN	recurrent neural network	XAI	Explainable Artificial Intelligence
RQ	research question	XGB	extreme gradient boosting
SARIMA	Seasonal Autoregressive Integrated Moving Average	XGBoost	Extreme Gradient Boosting Regressor
SAX	symbolic aggregate approximation	xLSTM	Extended Long Short-Term Memory
		YAformer	yet another Transformer

Symbols

Time series

$\mathbf{y}$	time series
$\mathbf{y}_i$	i^{th} time series
N	number of time series
T	time series length
C	context length
H	forecast horizon
$\hat{\mathbf{y}}_t$	forecast made at time t for time series $\mathbf{y}$
$\hat{\mathbf{y}}_{i,t}$	forecast made at time t for time series $\mathbf{y}_i$
f	model
$\mathbf{x}_i$	covariate
E	number of covariates
$\mathbf{X}$	set of all covariates
$\mathbf{X}_{past}$	set of covariates known in the past
$\mathbf{X}_{future}$	set of covariates known in the past and the future
$\hat{\mathbf{Y}}$	list of forecasts
$\bar{y}$	mean load
σ	standard deviation of the load
r	Pearson correlation
p_i	peak time of day i
$\hat{p}_i$	predicted peak time of day i
m_i	peak load of day i
$\hat{m}_i$	predicted peak load of day i

Transformer and Yformer

w_i	input word
o_i	output word

d	vector dimension
d_{past}	past vector dimension
d_{future}	future vector dimension
d_{model}	hidden dimension
$\mathbf{q}$	query vector
$\mathbf{k}_i$	key vector
$\mathbf{v}_i$	value vector
a_i	attention score
α_i	attention weight
$\mathbf{Q}$	query matrix
$\mathbf{K}$	key matrix
$\mathbf{V}$	value matrix
n_{layers}	number of layers
n_{heads}	number of attention heads per layer
$\mathcal{V}$	vocabulary
p	patch size
w_{conv}	convolution filter width

Shapley Additive Explanations (SHAP)

x_i	feature
V	set of features
S	coalition (subset of features)
ϕ_i	marginal contribution of x_i
p_j	feature permutation
B_i	pre-defined coalition
M	set of pre-defined coalitions
R	subset of pre-defined coalitions

List of Figures

3.1	Architecture of a Transformer text translation model	24
3.2	Architecture of the Transformer forecasting model	27
4.1	Examples of problematic time series in the Electricity dataset.	35
4.2	Seasonal pattern and weekly pattern for each year in the TransnetBW dataset.	38
4.3	Examples from the synthetic dataset.	41
5.1	The multivariate, local and global training strategies	44
5.2	Architecture of the Transformer forecasting model	46
5.3	Training curves of the multivariate and the global Transformer.	50
6.1	CVIs for k-means with Euclidean distance and varying numbers of clusters k	58
6.2	Resulting clusters on the Electricity-328 and Ausgrid substation datasets	59
6.3	Attention maps showing the cross-attention scores of the global Transformer per cluster.	61
7.1	Overview of the YAformer architecture.	67
7.2	Benchmarking results on the three datasets.	72
7.3	Comparison of the forecast error of the Transformer and Chronos-2 over time.	74
7.4	Exemplary forecasts of the Transformer and Chronos-2 on the TransnetBW data, showing cases where Chronos-2 exhibits large errors.	75
7.5	Statistical test results on the three datasets.	76
7.6	Annual total runtimes.	78
7.7	Ablation results.	78
8.1	Transformer architecture with the encoder self-attention (orange) and cross-attention (blue) highlighted.	84
8.2	Attention templates.	85
8.3	Attention maps of the self-attention and cross-attention averaged across all heads and 1000 samples of the TransnetBW dataset.	89
8.4	Attention maps of individual heads visualized by input position, analyzed on the TransnetBW dataset. Each row displays the heads of one Transformer layer.	90
8.5	Attention maps of individual heads visualized by hour of week, analyzed on the TransnetBW dataset.	91
8.6	Attention weights of each head, displayed by lag for the TransnetBW dataset.	92
8.7	Correlation of attention heads of a Transformer with two encoder layers, two decoder layers and four heads per layer, calculated on the TransnetBW dataset.	93
8.8	Correlation of the attention heads with features of the TransnetBW dataset.	94
8.9	Dependence of the attention of the individual heads on the TransnetBW dataset.	95
9.1	Overview of the SHAPformer approach.	98
9.2	Overview of the SHAPformer architecture.	101
9.3	The masked feature attention mechanism.	102
9.4	Feature importance on the synthetic test data.	109
9.5	Dependence on the six most important features of the synthetic dataset.	110

9.6	Local explanations of two different synthetic examples.	111
9.7	Feature importance on TransnetBW.	113
9.8	Dependence plots and local explanations from SHAPformer for TransnetBW data.	114
10.1	Overview of our approach to efficiently compute exact Shapley Additive Explanations (SHAP) values for covariate-informed Time Series Foundation Models (TSFMs).	118
10.2	Dependence of Chronos-2 on the covariates.	125
10.3	Local SHAP explanations for selected Chronos-2 predictions.	127
10.4	Chronos-2 predictions and local SHAP feature attributions across the full test set.	128
11.1	LDC 1 test data: the hourly load of one year (top) and the peak times (middle) and magnitudes (bottom).	131
11.2	LDC 2 test data: the hourly load of one year (top) and the peak times (middle) and magnitudes (bottom).	132
11.3	LDC 3 test data: the hourly load of one year (top) and the peak times (middle) and magnitudes (bottom).	132
A.1	Resulting clusters from FCM on the two datasets	149
A.2	Analysis of forecast error vs. distance to the closest cluster center.	151
C.1	Global explanations of the Transformer model on the synthetic data.	157
C.2	Local explanations of synthetic examples, created with PermutationSHAP and WindowSHAP.	158
C.3	Dependence plots of the Transformer model on TransnetBW data, created with PermutationSHAP and WindowSHAP.	160
C.4	Local explanations of TransnetBW examples created with PermutationSHAP and WindowSHAP.	161
C.5	Feature attention maps.	162
C.6	Attention manipulation plots for the same examples as in the main text.	163
C.7	Day to day impacts on the synthetic dataset.	164
C.8	Day to day impacts in the TransnetBW dataset.	165

List of Tables

2.1	Related work on benchmarking electrical load forecasting models.	11
4.1	Statistics of the datasets used in the dissertation.	33
5.1	Training strategy details	45
5.2	Forecasting results on the Electricity-321 and Ausgrid solar home datasets.	49
5.3	Training times in hours, measured on a machine with a Nvidia 3090 RTX GPU.	50
6.1	Forecasting results on Electricity-328.	60
6.2	Forecasting results on the Ausgrid substation dataset.	60
6.3	Training time in hours.	61
7.1	Dataset characteristics.	70
7.2	Runtime for one model training, one inference, and annual total runtime.	77
9.1	The chosen hyperparameters for the different models.	104
9.2	Comparison of forecast models.	106
9.3	Forecast errors on the real-world data.	107
9.4	Comparison of XAI methods.	107
10.1	Forecasting results on the test set.	122
10.2	Training runtime, runtime to generate a single prediction, and runtime to generate a single explanation.	123
10.3	Feature importance on the test set.	123
11.1	Features of the models evaluated on the BigDEAL data.	134
11.2	BigDEAL results.	134
12.1	Results of the Bayesian hyperparameter optimization process for the Transformer on the household (HH) dataset, the heat pumps only (HP) and the total load (Comb)	137
12.2	Household load forecasting results for the test data without heat pumps (2020).	139
12.3	Heat pump load forecasting results for the test data (2020).	139
12.4	Forecasting results for the whole energy community, including household and heat pump loads, for the test data (2020).	139
A.1	Forecasting results [% nMAE] on the Electricity-328 test set with half and full dataset size. Table from [109], © 2025, IEEE.	148
A.2	nMAE results for different cluster algorithms.	149
A.3	Forecasting results per cluster on the Electricity-328 test set.	150
A.4	Forecasting results per cluster on the Ausgrid substation test set.	150
B.1	Number of trainable parameters per model on the three datasets.	152
B.2	Model hyperparameter search range and the best configurations found with a Bayesian Optimization HPO for each dataset.	153
B.3	Additional metrics on the three datasets.	154
B.4	P-values of a paired permutation test and paired t-test.	155

Bibliography

- [1] S. Abnar and W. Zuidema. “Quantifying Attention Flow in Transformers”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020, pp. 4190–4197. DOI: 10.18653/v1/2020.acl-main.385.
- [2] A. Adadi and M. Berrada. “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6 (2018), pp. 52138–52160. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2870052.
- [3] K. Agarwal, L. Dheekollu, G. Dhama, A. Arora, S. Asthana, and T. Bhowmik. “Deep Learning based Time Series Forecasting”. In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Dec. 2020, pp. 859–864. DOI: 10.1109/ICMLA51294.2020.00140.
- [4] T. Aksu, G. Woo, J. Liu, X. Liu, C. Liu, S. Savarese, C. Xiong, and D. Sahoo. *GIFT-Eval: A Benchmark For General Time Series Forecasting Model Evaluation*. Nov. 2024. DOI: 10.48550/arXiv.2410.10393.
- [5] J. Alammar. *The Illustrated Transformer*. 2018. URL: <https://jalammar.github.io/illustrated-transformer/> (visited on 09/24/2025).
- [6] S. Ali et al. “Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence”. In: *Information Fusion* 99 (Nov. 2023), p. 101805. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2023.101805.
- [7] H. H. Aly. “A proposed intelligent short-term load forecasting hybrid models of ANN, WNN and KF based on clustering techniques for smart grid”. In: *Electric Power Systems Research* 182 (May 2020), p. 106191. ISSN: 03787796. DOI: 10.1016/j.epsr.2019.106191.
- [8] K. Amarasinghe, D. L. Marino, and M. Manic. “Deep neural networks for energy load forecasting”. In: *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. June 2017, pp. 1483–1488. DOI: 10.1109/ISIE.2017.8001465.
- [9] L. Ambrosius. *Analysis of the Impact of Upstream Time-Series Clustering on Short-Term Load Forecasting*. Master’s thesis. 2023.
- [10] A. N. Angelopoulos and S. Bates. *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*. 2021. DOI: 10.48550/ARXIV.2107.07511.
- [11] A. F. Ansari et al. *Chronos-2: From Univariate to Universal Forecasting*. 2025. DOI: 10.48550/ARXIV.2510.15821.
- [12] A. F. Ansari et al. *Chronos: Learning the Language of Time Series*. 2024. DOI: 10.48550/ARXIV.2403.07815.

- [13] R. R. Appino, J. Á. González Ordiano, R. Mikut, T. Faulwasser, and V. Hagenmeyer. “On the use of probabilistic forecasts in scheduling of renewable energy sources coupled to storages”. In: *Applied Energy* 210 (Jan. 2018), pp. 1207–1218. ISSN: 03062619. DOI: 10.1016/j.apenergy.2017.08.133.
- [14] P. Arjunan, K. Poolla, and C. Miller. “EnergyStar++: Towards more accurate and explanatory building energy benchmarking”. In: *Applied Energy* 276 (Oct. 2020), p. 115413. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2020.115413.
- [15] S. Arrhenius. “XXXI. *On the influence of carbonic acid in the air upon the temperature of the ground*”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41.251 (Apr. 1896), pp. 237–276. ISSN: 1941-5982, 1941-5990. DOI: 10.1080/14786449608620846.
- [16] M. Asghar Majeed, S. Phichaisawat, F. Asghar, and U. Hussan. “Data-Driven Optimized Load Forecasting: An LSTM-Based RNN Approach for Smart Grids”. In: *IEEE Access* 13 (2025), pp. 99018–99031. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3576303.
- [17] A. Auer, R. Parthipan, P. Mercado, A. F. Ansari, L. Stella, B. Wang, M. Bohlke-Schneider, and S. S. Rangapuram. *Zero-Shot Time Series Forecasting with Covariates via In-Context Learning*. 2025. DOI: 10.48550/ARXIV.2506.03128.
- [18] A. Auer, P. Podest, D. Klotz, S. Böck, G. Klambauer, and S. Hochreiter. *TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning*. Nov. 2025. DOI: 10.48550/arXiv.2505.23719.
- [19] Australian Government Department of Climate Change, Energy, the Environment and Water. *Smart-Grid Smart-City Customer Trial Data*. 2015. URL: <https://data.gov.au/data/dataset/smart-grid-smart-city-customer-trial-data>.
- [20] H.-A. I. El-Azab, R. Swief, N. H. El-Amary, and H. Temraz. “Machine and deep learning approaches for forecasting electricity price and energy load assessment on real datasets”. In: *Ain Shams Engineering Journal* 15.4 (Apr. 2024), p. 102613. ISSN: 20904479. DOI: 10.1016/j.asej.2023.102613.
- [21] J. L. Ba, J. R. Kiros, and G. E. Hinton. “Layer Normalization”. In: *NeurIPS 2016* (Aug. 2016). URL: https://openreview.net/forum?id=BJLa_ZC9 (visited on 08/14/2025).
- [22] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLOS ONE* 10.7 (Oct. 2015). ISSN: 1932-6203. DOI: 10.1371/journal.pone.0130140.
- [23] A. Q. H. Badar and A. Anvari-Moghaddam. “Smart home energy management system – a review”. In: *Advances in Building Energy Research* 16.1 (Jan. 2022), pp. 118–143. ISSN: 1751-2549, 1756-2201. DOI: 10.1080/17512549.2020.1806925.
- [24] D. Bahdanau, K. Cho, and Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: <http://arxiv.org/abs/1409.0473>.

- [25] H. Bast, M. Hertel, and M. M. Mohamed. “Tokenization Repair in the Presence of Spelling Errors”. In: *Proceedings of the 25th Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Nov. 2021, pp. 279–289. DOI: 10.18653/v1/2021.conll-1.22.
- [26] H. Bast, M. Hertel, and N. Prange. “A Fair and In-Depth Evaluation of Existing End-to-End Entity Linking Systems”. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023, pp. 6659–6672. DOI: 10.18653/v1/2023.emnlp-main.411.
- [27] H. Bast, M. Hertel, and N. Prange. “ELEVANT: A Fully Automatic Fine-Grained Entity Linking Evaluation and Analysis Tool”. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Dec. 2022, pp. 72–79. DOI: 10.18653/v1/2022.emnlp-demos.8.
- [28] H. Bast, M. Hertel, and S. Storandt. “Scalable Transfer Patterns”. In: *2016 Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, Dec. 2015, pp. 15–29. DOI: 10.1137/1.9781611974317.2.
- [29] H. Bast, M. Hertel, and S. Walter. “Fast Whitespace Correction with Encoder-Only Transformers”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Association for Computational Linguistics, 2023, pp. 389–399. DOI: 10.18653/v1/2023.ac1-demo.37.
- [30] L. Baur, K. Ditschuneit, M. Schambach, C. Kaymakci, T. Wollmann, and A. Sauer. “Explainability and Interpretability in Electric Load Forecasting Using Machine Learning Techniques – A Review”. In: *Energy and AI* 16 (May 2024), p. 100358. ISSN: 2666-5468. DOI: 10.1016/j.egyai.2024.100358.
- [31] M. Beck et al. *xLSTM: Extended Long Short-Term Memory*. 2024. DOI: 10.48550/ARXIV.2405.04517.
- [32] J. Bento, P. Saleiro, A. F. Cruz, M. A. Figueiredo, and P. Bizarro. “TimeSHAP: Explaining Recurrent Models through Sequence Perturbations”. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD ’21. Association for Computing Machinery, Aug. 2021, pp. 2565–2573. ISBN: 978-1-4503-8332-5. DOI: 10.1145/3447548.3467166.
- [33] M. Bernecker, M. Gebhardt, S. B. Amor, M. Wolter, and F. Müsgens. “Quantifying the impact of load forecasting accuracy on congestion management in distribution grids”. In: *International Journal of Electrical Power & Energy Systems* 168 (July 2025), p. 110713. ISSN: 0142-0615. DOI: 10.1016/j.ijepes.2025.110713.
- [34] A. Bibal, R. Cardon, D. Alfter, R. Wilkens, X. Wang, T. François, and P. Watrin. “Is Attention Explanation? An Introduction to the Debate”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, May 2022, pp. 3889–3900. DOI: 10.18653/v1/2022.ac1-long.269.
- [35] L. Biewald. *Experiment Tracking with Weights and Biases*. 2020. URL: wandb.com.

-
- [36] Bundesministerium für Wirtschaft und Energie. *Redispatch 3.0*. URL: <https://www.bundeswirtschaftsministerium.de/Redaktion/EN/Artikel/Digital-World/GAIA-X-Use-Cases/redispatch-30.html> (visited on 09/30/2025).
 - [37] Bundesnetzagentur. *Roll-out intelligenter Messsysteme*. 2026. URL: <https://www.bundesnetzagentur.de/DE/Fachthemen/ElektrizitaetundGas/NetzzugangMesswesen/Mess-undZaehlwesen/iMSys/artikel.html> (visited on 09/28/2026).
 - [38] Bundesnetzagentur. *Transmission system operators*. 2025. URL: <https://www.smard.de/page/en/wiki-article/6138/205528/transmission-system-operators> (visited on 08/05/2025).
 - [39] Bundesnetzagentur. *Update: Verteilernetze bis 2045*. 2025. URL: <https://www.smard.de/page/home/topic-article/444/215544/update-verteilernetze-bis-2045> (visited on 04/25/2026).
 - [40] H. K. Çakmak and V. Hagenmeyer. “Using Open Data for Modeling and Simulation of the All Electrical Society in eASiMOV”. In: *2022 Open Source Modelling and Simulation of Energy Systems (OSMSES)*. Apr. 2022, pp. 1–6. DOI: 10.1109/OSMSES54027.2022.9769145.
 - [41] T. Calinski and J. Harabasz. “A dendrite method for cluster analysis”. In: *Communications in Statistics - Theory and Methods* 3.1 (1974), pp. 1–27. ISSN: 0361-0926. DOI: 10.1080/03610927408827101.
 - [42] K. Calvin et al. *IPCC, 2023: Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, H. Lee and J. Romero (eds.)]*. IPCC, Geneva, Switzerland. Tech. rep. Intergovernmental Panel on Climate Change (IPCC), July 2023. URL: <https://doi.org/10.59327/IPCC/AR6-9789291691647> (visited on 05/13/2025).
 - [43] S. Cen and C. G. Lim. “Multi-Task Learning of the PatchTCN-TST Model for Short-Term Multi-Load Energy Forecasting Considering Indoor Environments in a Smart Building”. In: *IEEE Access* 12 (2024), pp. 19553–19568. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3355448.
 - [44] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski. “NHITS: Neural Hierarchical Interpolation for Time Series Forecasting”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.6 (June 2023), pp. 6989–6997. ISSN: 2374-3468. DOI: 10.1609/aaai.v37i6.25854.
 - [45] M. Chaouch. “Clustering-Based Improvement of Nonparametric Functional Time Series Forecasting: Application to Intra-Day Household-Level Load Curves”. In: *IEEE Transactions on Smart Grid* 5.1 (Jan. 2014), pp. 411–419. ISSN: 1949-3053, 1949-3061. DOI: 10.1109/TSG.2013.2277171.
 - [46] H. Chen, S. Wang, S. Wang, and Y. Li. “Day-ahead aggregated load forecasting based on two-terminal sparse coding and deep neural network fusion”. In: *Electric Power Systems Research* 177 (Dec. 2019), p. 105987. ISSN: 03787796. DOI: 10.1016/j.epsr.2019.105987.
 - [47] H. Chen, I. C. Covert, S. M. Lundberg, and S.-I. Lee. “Algorithms to estimate Shapley value feature attributions”. In: *Nature Machine Intelligence* 5.6 (June 2023), pp. 590–601. ISSN: 2522-5839. DOI: 10.1038/s42256-023-00657-x.

-
- [48] H. Chen, J. D. Janizek, S. Lundberg, and S.-I. Lee. *True to the Model or True to the Data?* June 2020. DOI: 10.48550/arXiv.2006.16234.
 - [49] B.-J. Chen, M.-W. Chang, and C.-J. Lin. “Load forecasting using support vector Machines: a study on EUNITE competition 2001”. In: *IEEE Transactions on Power Systems* 19.4 (Nov. 2004), pp. 1821–1830. ISSN: 1558-0679. DOI: 10.1109/TPWRS.2004.835679.
 - [50] T. Chen and C. Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.
 - [51] G. Chitalia, M. Pipattanasomporn, V. Garg, and S. Rahman. “Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks”. In: *Applied Energy* 278 (Nov. 2020), p. 115410. ISSN: 03062619. DOI: 10.1016/j.apenergy.2020.115410.
 - [52] W. Choi and S. Lee. “Performance evaluation of deep learning architectures for load and temperature forecasting under dataset size constraints and seasonality”. In: *Energy and Buildings* 288 (June 2023), p. 113027. ISSN: 03787788. DOI: 10.1016/j.enbuild.2023.113027.
 - [53] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. *What Does BERT Look At? An Analysis of BERT’s Attention*. June 2019. DOI: 10.48550/arXiv.1906.04341.
 - [54] B. Cohen et al. *This Time is Different: An Observability Perspective on Time Series Foundation Models*. 2025. DOI: 10.48550/ARXIV.2505.14766.
 - [55] Commission for Energy Regulation. *CER Smart Metering Project - Electricity Customer Behaviour Trial*. 2012. DOI: 10.57702/XKS7KDXD.
 - [56] Copernicus. *Copernicus: 2025 was the third hottest year on record*. Jan. 2026. URL: <https://climate.copernicus.eu/copernicus-2025-was-third-hottest-year-record> (visited on 04/25/2026).
 - [57] Copernicus Climate Change Service. *Climate and energy indicators for Europe from 1979 to present derived from reanalysis*. 2020. DOI: 10.24381/CDS.4BD77450.
 - [58] Y. Dai, X. Yang, and M. Leng. “Optimized Seq2Seq model based on multiple methods for short-term power load forecasting”. In: *Applied Soft Computing* 142 (July 2023), p. 110335. ISSN: 15684946. DOI: 10.1016/j.asoc.2023.110335.
 - [59] A. Das, W. Kong, A. Leach, S. K. Mathur, R. Sen, and R. Yu. “Long-term Forecasting with TiDE: Time-series Dense Encoder”. In: *Transactions on Machine Learning Research* (May 2023). ISSN: 2835-8856. URL: <https://openreview.net/forum?id=pCbC3aQB5W> (visited on 05/26/2026).
 - [60] A. Das, W. Kong, R. Sen, and Y. Zhou. *A decoder-only foundation model for time-series forecasting*. Feb. 2024. DOI: <https://doi.org/10.48550/arXiv.2310.10688>.
 - [61] A. Das and P. Rad. *Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey*. June 2020. DOI: 10.48550/arXiv.2006.11371.

- [62] S. Das, P. Xu, Z. Dai, A. Endert, and L. Ren. “Interpreting Deep Neural Networks through Prototype Factorization”. In: *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, Nov. 2020, pp. 448–457. ISBN: 978-1-7281-9012-9. DOI: 10.1109/ICDMW51313.2020.00068.
- [63] D. L. Davies and D. W. Bouldin. “A Cluster Separation Measure”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1.2* (Apr. 1979), pp. 224–227. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.1979.4766909.
- [64] B. Deiseroth, M. Deb, S. Weinbach, M. Brack, P. Schramowski, and K. Kersting. “ATMAN: Understanding Transformer Predictions Through Memory Efficient Attention Manipulation”. In: *Advances in Neural Information Processing Systems 36* (Dec. 2023), pp. 63437–63460. DOI: <https://dl.acm.org/doi/10.5555/3666122.3668891>.
- [65] E. Delaney, D. Greene, and M. T. Keane. “Instance-Based Counterfactual Explanations for Time Series Classification”. In: *Case-Based Reasoning Research and Development*. Springer International Publishing, 2021, pp. 32–47. ISBN: 978-3-030-86957-1. DOI: 10.1007/978-3-030-86957-1_3.
- [66] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (2019). URL: <https://aclanthology.org/N19-1423.pdf> (visited on 05/19/2026).
- [67] M. Dias, A. Florêncio, dirk, and D. Nicholson. *omadson/fuzzy-c-means: v1.7.2*. Mar. 2024. DOI: 10.5281/ZENODO.3066222.
- [68] F. X. Diebold and R. S. Mariano. “Comparing Predictive Accuracy”. In: *Journal of Business & Economic Statistics* 13.3 (July 1995), pp. 253–263. ISSN: 0735-0015, 1537-2707. DOI: 10.1080/07350015.1995.10524599.
- [69] A. Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. Oct. 2020. URL: <https://openreview.net/forum?id=YicbFdNTTy> (visited on 08/13/2025).
- [70] “Dynamic Time Warping”. In: *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer, 2007, pp. 69–84. ISBN: 978-3-540-74047-6 978-3-540-74048-3. DOI: 10.1007/978-3-540-74048-3_4.
- [71] J. Eickmann, T. Drees, J. D. Sprey, and A. Moser. “Optimizing Storages for Transmission System Operation”. In: *Energy Procedia*. 8th International Renewable Energy Storage Conference and Exhibition (IRES 2013) 46 (Jan. 2014), pp. 13–21. ISSN: 1876-6102. DOI: 10.1016/j.egypro.2014.01.153.
- [72] V. Ekambaram, A. Jati, P. Dayama, S. Mukherjee, N. H. Nguyen, W. M. Gifford, C. Reddy, and J. Kalagnanam. *Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series*. 2024. DOI: 10.48550/ARXIV.2401.03955.

-
- [73] V. Ekambaram, A. Jati, N. Nguyen, P. Sinthong, and J. Kalagnanam. “TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting”. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’23. Association for Computing Machinery, Aug. 2023, pp. 459–469. ISBN: 979-8-4007-0103-0. DOI: 10.1145/3580305.3599533.
 - [74] P. Emami, A. Sahu, and P. Graf. “BuildingsBench: A Large-Scale Dataset of 900K Buildings and Benchmark for Short-Term Load Forecasting”. In: *37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks*. Oct. 2023. DOI: <https://doi.org/10.48550/arXiv.2307.00142>.
 - [75] ENTSO-E. *Transparency platform*. 2026. URL: <https://transparency.entsoe.eu>.
 - [76] U. N. Environment. *Emissions Gap Report 2024 | UNEP - UN Environment Programme*. Section: publications. Oct. 2024. URL: <https://www.unep.org/resources/emissions-gap-report-2024> (visited on 05/16/2025).
 - [77] G. Erion, J. D. Janizek, P. Sturmfels, S. M. Lundberg, and S.-I. Lee. “Improving performance of deep learning models with axiomatic attribution priors and expected gradients”. In: *Nature Machine Intelligence* 3.7 (July 2021), pp. 620–631. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00343-w.
 - [78] European Commission. *Artificial Intelligence Act*. 2024. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L_202401689#art_13 (visited on 05/19/2026).
 - [79] M. Flora, C. Potvin, A. McGovern, and S. Handler. *Comparing Explanation Methods for Traditional Machine Learning Models Part I: An Overview of Current Methods and Quantifying Their Disagreement*. Nov. 2022. DOI: 10.48550/arXiv.2211.08943.
 - [80] C. Frye, D. d. Mijolla, T. Begley, L. Cowton, M. Stanley, and I. Feige. “Shapley explainability on the data manifold”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=OPyWRrcjVQw> (visited on 07/25/2025).
 - [81] X. Fu, X.-J. Zeng, P. Feng, and X. Cai. “Clustering-based short-term load forecasting for residential electricity under the increasing-block pricing tariffs in China”. In: *Energy* 165 (Dec. 2018), pp. 76–89. ISSN: 03605442. DOI: 10.1016/j.energy.2018.09.156.
 - [82] P. Gaillard, Y. Goude, and R. Nedellec. “Additive models and robust aggregation for GEFCom2014 probabilistic electric load and electricity price forecasting”. In: *International Journal of Forecasting* 32.3 (July 2016), pp. 1038–1050. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2015.12.001.
 - [83] K. Gajowniczek and T. Ząbkowski. “Simulation Study on Clustering Approaches for Short-Term Electricity Forecasting”. In: *Complexity* 2018 (2018), pp. 1–21. ISSN: 1076-2787, 1099-0526. DOI: 10.1155/2018/3683969.
 - [84] J. Galenzowski, S. Waczowicz, S. Meisenbacher, R. Mikut, and V. Hagenmeyer. “A Real-World District Community Platform as a Cyber-Physical-Social Infrastructure Systems in the Energy Domain”. In: *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. BuildSys ’23. Association for Computing Machinery, Nov. 2023, pp. 434–441. ISBN: 979-8-4007-0230-3. DOI: 10.1145/3600100.3626347.

-
- [85] J. Gao, W. Hu, D. Zhang, and Y. Chen. *TgDLF2.0: Theory-guided deep-learning for electrical load forecasting via Transformer and transfer learning*. Oct. 2022. DOI: 10.48550/arXiv.2210.02448.
 - [86] A. Garza, C. Challu, and M. Mergenthaler-Canseco. *TimeGPT-1*. 2023. DOI: 10.48550/ARXIV.2310.03589.
 - [87] A. Gasparin, S. Lukovic, and C. Alippi. “Deep learning for time series forecasting: The electric load case”. In: *CAAI Transactions on Intelligence Technology* 7.1 (Mar. 2022), pp. 1–25. ISSN: 2468-2322, 2468-2322. DOI: 10.1049/cit2.12060.
 - [88] D. Gay, R. Guigourès, M. Boullé, and F. Clérot. “Feature Extraction over Multiple Representations for Time Series Classification”. In: *New Frontiers in Mining Complex Patterns*. Springer International Publishing, 2014, pp. 18–34. ISBN: 978-3-319-08407-7. DOI: 10.1007/978-3-319-08407-7_2.
 - [89] A. B. Georges Hebrail. *Individual Household Electric Power Consumption*. 2006. DOI: 10.24432/C58K54.
 - [90] E. Giacomazzi, F. Haag, and K. Hopf. “Short-Term Electricity Load Forecasting Using the Temporal Fusion Transformer: Effect of Grid Hierarchies and Data Sources”. In: *Proceedings of the 14th ACM International Conference on Future Energy Systems*. ACM, June 2023, pp. 353–360. ISBN: 979-8-4007-0032-3. DOI: 10.1145/3575813.3597345.
 - [91] T. Gneiting and M. Katzfuss. “Probabilistic Forecasting”. In: *Annual Review of Statistics and Its Application* 1.1 (Jan. 2014), pp. 125–151. ISSN: 2326-8298, 2326-831X. DOI: 10.1146/annurev-statistics-062713-085831.
 - [92] R. W. Godahewa, C. Bergmeir, G. I. Webb, R. Hyndman, and P. Montero-Manso. “Monash Time Series Forecasting Archive”. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. Aug. 2021. URL: <https://openreview.net/forum?id=wEc1mgAjU-> (visited on 02/16/2026).
 - [93] J. Á. González Ordiano, S. Waczowicz, V. Hagenmeyer, and R. Mikut. “Energy forecasting tools and services”. In: *WIREs Data Mining and Knowledge Discovery* 8.2 (Mar. 2018). ISSN: 1942-4787, 1942-4795. DOI: 10.1002/widm.1235.
 - [94] J. Á. González-Ordiano et al. “Probabilistic forecasts of the distribution grid state using data-driven forecasts and probabilistic power flow”. In: *Applied Energy* 302 (Nov. 2021), p. 117498. ISSN: 03062619. DOI: 10.1016/j.apenergy.2021.117498.
 - [95] M. Grabner, Y. Wang, Q. Wen, B. Blažič, and V. Štruc. “A Global Modeling Framework for Load Forecasting in Distribution Networks”. In: *IEEE Transactions on Smart Grid* 14.6 (Nov. 2023), pp. 4927–4941. ISSN: 1949-3061. DOI: 10.1109/TSG.2023.3264525.
 - [96] A. Groß, A. Lenders, F. Schwenker, D. A. Braun, and D. Fischer. “Comparison of short-term electrical load forecasting methods for different building types”. In: *Energy Informatics* 4.3 (Sept. 2021), p. 13. ISSN: 2520-8942. DOI: 10.1186/s42162-021-00172-6.
 - [97] T. A. Grzeszczyk and M. K. Grzeszczyk. “Justifying Short-Term Load Forecasts Obtained with the Use of Neural Models”. In: *Energies* 15.5 (Jan. 2022). Number: 5, p. 1852. ISSN: 1996-1073. DOI: 10.3390/en15051852.

-
- [98] R. Guidotti, A. Monreale, F. Spinnato, D. Pedreschi, and F. Giannotti. “Explaining Any Time Series Classifier”. In: *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*. Oct. 2020, pp. 167–176. DOI: 10.1109/CogMI50398.2020.00029.
 - [99] M. Guillemé, V. Masson, L. Rozé, and A. Termier. “Agnostic Local Explanation for Time Series Classification”. In: *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. Nov. 2019, pp. 432–439. DOI: 10.1109/ICTAI.2019.00067.
 - [100] S. Haben, S. Arora, G. Giasemidis, M. Voss, and D. Vukadinović Greetham. “Review of low voltage load forecasting: Methods, applications, and recommendations”. In: *Applied Energy* 304 (Dec. 2021), p. 117798. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2021.117798.
 - [101] S. Haben, M. Voss, and W. Holderbaum. *Core Concepts and Methods in Load Forecasting: With Applications in Distribution Networks*. Springer International Publishing, 2023. ISBN: 978-3-031-27851-8 978-3-031-27852-5. DOI: 10.1007/978-3-031-27852-5.
 - [102] V. Hagenmeyer et al. “Information and Communication Technology in Energy Lab 2.0: Smart Energies System Simulation and Control Center with an Open-Street-Map-Based Power Flow Simulation Example”. In: *Energy Technology* 4.1 (Jan. 2016), pp. 145–162. ISSN: 2194-4288, 2194-4296. DOI: 10.1002/ente.201500304.
 - [103] M. Hasan, Z. Mifta, S. J. Papiya, P. Roy, P. Dey, N. A. Salsabil, N.-U.-R. Chowdhury, and O. Farrok. “A state-of-the-art comparative review of load forecasting methods: Characteristics, perspectives, and applications”. In: *Energy Conversion and Management: X* 26 (Apr. 2025), p. 100922. ISSN: 2590-1745. DOI: 10.1016/j.ecmx.2025.100922.
 - [104] T. Haug and C. Goebel. “DSML: A Comprehensive Framework for Evaluating ML Methods for Distribution System State Estimation”. In: *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems*. ACM, June 2025, pp. 122–135. ISBN: 979-8-4007-1125-1. DOI: 10.1145/3679240.3734629.
 - [105] B. Heidrich, M. Hertel, O. Neumann, V. Hagenmeyer, and R. Mikut. “Using conditional Invertible Neural Networks to perform mid-term peak load forecasting”. In: *IET Smart Grid* 7.4 (2024), pp. 460–472. ISSN: 2515-2947. DOI: 10.1049/stg2.12169.
 - [106] B. Heidrich, M. Turowski, N. Ludwig, R. Mikut, and V. Hagenmeyer. “Forecasting energy time series with profile neural networks”. In: *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*. ACM, June 2020, pp. 220–230. ISBN: 978-1-4503-8009-6. DOI: 10.1145/3396851.3397683.
 - [107] H. Hersbach et al. “The ERA5 global reanalysis”. In: *Quarterly Journal of the Royal Meteorological Society* 146.730 (July 2020), pp. 1999–2049. ISSN: 0035-9009, 1477-870X. DOI: 10.1002/qj.3803.
 - [108] M. Hertel. *Synthetic electrical load forecasting dataset with ground-truth SHAP explanations*. Apr. 2026. DOI: <https://doi.org/10.5281/zenodo.19485917>.
 - [109] M. Hertel, L. Ambrosius, M. Treutlein, R. Mikut, and V. Hagenmeyer. “A comparison of local, cluster-specific and global Transformer models for forecasting electrical loads of individual buildings and substations”. In: *2025 IEEE Kiel PowerTech*. IEEE, June 2025, pp. 1–8. ISBN: 979-8-3315-4397-6. DOI: 10.1109/PowerTech59965.2025.11180482.

- [110] M. Hertel, M. Beichter, B. Heidrich, O. Neumann, B. Schäfer, R. Mikut, and V. Hagenmeyer. “Transformer training strategies for forecasting multiple load time series”. In: *Energy Informatics* 6.1 (Oct. 2023), p. 20. ISSN: 2520-8942. DOI: 10.1186/s42162-023-00278-z.
- [111] M. Hertel, A. Nikoltchovska, S. Pütz, B. Schäfer, R. Mikut, and V. Hagenmeyer. “Explainable Load Forecasting with Covariate-Informed Time Series Foundation Models”. In: *Proceedings of the 17th ACM International Conference on Future and Sustainable Energy Systems*. ACM, June 2026, pp. 612–626. ISBN: 979-8-4007-2011-6. DOI: 10.1145/3744255.3811724.
- [112] M. Hertel, S. Ott, O. Neumann, B. Schäfer, R. Mikut, and V. Hagenmeyer. “Transformer Neural Networks for Building Load Forecasting”. In: *Tackling Climate Change with Machine Learning: workshop at NeurIPS 2022*. 2022. URL: <https://www.climatechange.ai/papers/neurips2022/43> (visited on 09/30/2025).
- [113] M. Hertel, S. Ott, B. Schäfer, R. Mikut, V. Hagenmeyer, and O. Neumann. “Evaluation of Transformer Architectures for Electrical Load Time-Series Forecasting”. In: *Proceedings 32. Workshop Computational Intelligence*. Vol. 1. 2022, p. 93. URL: https://library.oapen.org/bitstream/handle/20.500.12657/59840/external_content.pdf?sequence=1#page=105 (visited on 10/23/2023).
- [114] M. Hertel, S. Pütz, J. Kolar, B. Schäfer, R. Mikut, and V. Hagenmeyer. *A Benchmark for Electrical Load Forecasting Across Grid Levels: Time-Series Transformers Outperform Established Methods*. July 2026. DOI: 10.48550/arXiv.2607.15705.
- [115] M. Hertel, S. Pütz, R. Mikut, V. Hagenmeyer, and B. Schäfer. “Explainable time-series forecasting with sampling-free SHAP for Transformers”. In: *Nature Communications* 17.1 (May 2026), p. 4723. ISSN: 2041-1723. DOI: 10.1038/s41467-026-73243-5.
- [116] H. Hewamalage, K. Ackermann, and C. Bergmeir. “Forecast evaluation for data scientists: common pitfalls and best practices”. In: *Data Mining and Knowledge Discovery* 37.2 (Mar. 2023), pp. 788–832. ISSN: 1384-5810, 1573-756X. DOI: 10.1007/s10618-022-00894-5.
- [117] Y. Himeur, M. Elnour, F. Fadli, N. Meskin, I. Petri, Y. Rezgui, F. Bensaali, and A. Amira. “Next-generation energy systems for sustainable smart cities: Roles of transfer learning”. In: *Sustainable Cities and Society* 85 (Oct. 2022), p. 104059. ISSN: 2210-6707. DOI: 10.1016/j.scs.2022.104059.
- [118] H. Hippert, C. Pedreira, and R. Souza. “Neural networks for short-term load forecasting: a review and evaluation”. In: *IEEE Transactions on Power Systems* 16.1 (Feb. 2001), pp. 44–55. ISSN: 1558-0679. DOI: 10.1109/59.910780.
- [119] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.
- [120] N. Hollmann, S. Müller, K. Eggenberger, and F. Hutter. *TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second*. Sept. 2023. DOI: 10.48550/arXiv.2207.01848.
- [121] T. Hong. “Energy Forecasting: Past, Present, and Future”. In: *Foresight: The International Journal of Applied Forecasting* 32 (2014), pp. 43–48. (Visited on 08/11/2025).

-
- [122] T. Hong and S. Fan. “Probabilistic electric load forecasting: A tutorial review”. In: *International Journal of Forecasting* 32.3 (July 2016), pp. 914–938. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2015.11.011.
 - [123] T. Hong, P. Pinson, and S. Fan. “Global Energy Forecasting Competition 2012”. In: *International Journal of Forecasting* 30.2 (Apr. 2014), pp. 357–363. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2013.07.001.
 - [124] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman. “Global Energy Forecasting Competition 2014 and beyond”. In: *International Journal of Forecasting* 32.3 (July 2016), pp. 896–913. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2016.02.001.
 - [125] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour. “Energy Forecasting: A Review and Outlook”. In: *IEEE Open Access Journal of Power and Energy* 7 (2020), pp. 376–388. ISSN: 2687-7910. DOI: 10.1109/OAJPE.2020.3029979.
 - [126] T. Hong, J. Xie, and J. Black. “Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting”. In: *International Journal of Forecasting* 35.4 (Oct. 2019), pp. 1389–1399. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.02.006.
 - [127] S. B. Hoo, S. Müller, D. Salinas, and F. Hutter. *From Tables to Time: How TabPFN-v2 Outperforms Specialized Time Series Forecasting Models*. May 2025. DOI: 10.48550/arXiv.2501.02945.
 - [128] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin. *TabTransformer: Tabular Data Modeling Using Contextual Embeddings*. Dec. 2020. DOI: 10.48550/arXiv.2012.06678.
 - [129] P. C. Huy, N. Q. Minh, N. D. Tien, and T. T. Q. Anh. “Short-Term Electricity Load Forecasting Based on Temporal Fusion Transformer Model”. In: *IEEE Access* 10 (2022), pp. 106296–106304. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3211941.
 - [130] I. A. Ibrahim and M. Hossain. “Short-term multivariate time series load data forecasting at low-voltage level using optimised deep-ensemble learning-based models”. In: *Energy Conversion and Management* 296 (Nov. 2023), p. 117663. ISSN: 01968904. DOI: 10.1016/j.enconman.2023.117663.
 - [131] M. Imani and H. Ghassemian. “Electrical Load Forecasting Using Customers Clustering and Smart Meters in Internet of Things”. In: *2018 9th International Symposium on Telecommunications (IST)*. IEEE, Dec. 2018, pp. 113–117. ISBN: 978-1-5386-8274-6. DOI: 10.1109/ISTEL.2018.8661071.
 - [132] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. *Perceiver: General Perception with Iterative Attention*. June 2021. DOI: 10.48550/arXiv.2103.03206.
 - [133] H. Jahangir, H. Tayarani, S. S. Gougheri, M. A. Golkar, A. Ahmadian, and A. Elkamel. “Deep Learning-Based Forecasting Approach in Smart Grids With Microclustering and Bidirectional LSTM Network”. In: *IEEE Transactions on Industrial Electronics* 68.9 (Sept. 2021), pp. 8298–8309. ISSN: 0278-0046, 1557-9948. DOI: 10.1109/TIE.2020.3009604.
 - [134] A. Jain and S. C. Gupta. “Evaluation of electrical load demand forecasting using various machine learning algorithms”. In: *Frontiers in Energy Research* 12 (June 2024), p. 1408119. ISSN: 2296-598X. DOI: 10.3389/fenrg.2024.1408119.

-
- [135] M. Jain, T. AlSkaif, and S. Dev. “Are deep learning models more effective against traditional models for load demand forecasting?” In: *2022 International Conference on Smart Energy Systems and Technologies (SEST)*. Sept. 2022, pp. 1–6. DOI: 10.1109/SEST53650.2022.9898424.
 - [136] S. Jain and B. C. Wallace. “Attention is not Explanation”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, June 2019, pp. 3543–3556. DOI: 10.18653/v1/N19-1357.
 - [137] G. Jeong, S. Park, J. Lee, and G. Hwang. “Energy Trading System in Microgrids With Future Forecasting and Forecasting Errors”. In: *IEEE Access* 6 (2018), pp. 44094–44106. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2861993.
 - [138] J.-y. Jo and S.-H. Myaeng. “Roles and Utilization of Attention Heads in Transformer-based Neural Language Models”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July 2020, pp. 3404–3417. DOI: 10.18653/v1/2020.acl-main.311.
 - [139] L. H. Kaack, P. L. Donti, E. Strubell, G. Kamiya, F. Creutzig, and D. Rolnick. “Aligning artificial intelligence with climate change mitigation”. In: *Nature Climate Change* 12.6 (June 2022), pp. 518–527. ISSN: 1758-6798. DOI: 10.1038/s41558-022-01377-7.
 - [140] F. Karim, S. Majumdar, H. Darabi, and S. Chen. “LSTM Fully Convolutional Networks for Time Series Classification”. In: *IEEE Access* 6 (2018), pp. 1662–1669. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2779939.
 - [141] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff. *ConvTimeNet: A Pre-trained Deep Convolutional Neural Network for Time Series Classification*. May 2019. DOI: 10.48550/arXiv.1904.12546.
 - [142] M. Katholnigg and C. Goebel. “The Impact of Load Forecasting Model Choice on the Economic Operation of Industrial Energy Storage”. In: *Proceedings of the 12th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. ACM, Nov. 2025, pp. 12–22. ISBN: 979-8-4007-1945-5. DOI: 10.1145/3736425.3772098.
 - [143] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. “LightGBM: a highly efficient gradient boosting decision tree”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Curran Associates Inc., 2017, pp. 3149–3157. ISBN: 978-1-5108-6096-4. URL: <https://dl.acm.org/doi/10.5555/3294996.3295074> (visited on 04/14/2026).
 - [144] A. R. Khan, A. Mahmood, A. Safdar, Z. A. Khan, and N. A. Khan. “Load forecasting, dynamic pricing and DSM in smart grid: A review”. In: *Renewable and Sustainable Energy Reviews* 54 (Feb. 2016), pp. 1311–1322. ISSN: 13640321. DOI: 10.1016/j.rser.2015.10.117.
 - [145] A. Khotanzad, R. Afkhami-Rohani, and D. Maratukulam. “ANNSTLF - Artificial Neural Network Short-Term Load Forecaster - generation three”. In: *IEEE Transactions on Power Systems* 13.4 (Nov. 1998), pp. 1413–1422. ISSN: 08858950. DOI: 10.1109/59.736285.

-
- [146] H. Kim, S. Park, and S. Kim. “Time-series clustering and forecasting household electricity demand using smart meter data”. In: *Energy Reports* 9 (Dec. 2023), pp. 4111–4121. ISSN: 23524847. DOI: 10.1016/j.egy.2023.03.042.
 - [147] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo. “Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift”. In: *International Conference on Learning Representations*. Oct. 2021. URL: <https://openreview.net/forum?id=cGDAkQo1C0p> (visited on 05/18/2026).
 - [148] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. Jan. 2017. DOI: 10.48550/arXiv.1412.6980.
 - [149] M. Kleinebrahm et al. “Energy-Arena: A Dynamic Benchmark for Operational Energy Forecasting”. In: *2026 22nd International Conference on the European Energy Market (EEM)*. IEEE, June 2026, pp. 1–6. ISBN: 979-8-3195-3554-2. DOI: 10.1109/EEM68581.2026.11589800.
 - [150] R. Koenker. *Quantile regression*. Repr. Econometric Society monographs 38. Cambridge Univ. Press, 2009. ISBN: 978-0-521-60827-5 978-0-521-84573-1.
 - [151] J. Kolar, M. Hertel, S. Meisenbacher, R. Mikut, and V. Hagenmeyer. “Reducing Imbalance Energy by Spatial Balance Area Forecasting”. In: *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems*. E-Energy ’25. Association for Computing Machinery, June 2025, pp. 666–676. ISBN: 979-8-4007-1125-1. DOI: 10.1145/3679240.3734634.
 - [152] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang. “Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network”. In: *IEEE Transactions on Smart Grid* 10.1 (Jan. 2019), pp. 841–851. ISSN: 1949-3061. DOI: 10.1109/TSG.2017.2753802.
 - [153] A. Kreusel, M. Hertel, M. Noskiewicz, H. Maaß, R. Mikut, and V. Hagenmeyer. “Evaluating Time-Series Foundation Models for Cooling Demand Forecasting with Little Data”. In: *Proceedings 35. Workshop Computational Intelligence*. Nov. 2025, pp. 301–322.
 - [154] A. Kreusel, M. Hertel, M. Noskiewicz, F. Zahn, H. Maaß, R. Mikut, and V. Hagenmeyer. “Covariates Are the Key to Accurate Probabilistic Building Energy Forecasting with Time Series Foundation Models”. In: *Proceedings of the 2026 ACM Sustainability Week*. ACM, June 2026, pp. 172–182. ISBN: 979-8-4007-2199-1. DOI: 10.1145/3765611.3815507.
 - [155] J. Kruse, B. Schäfer, and D. Witthaut. “Exploring deterministic frequency deviations with explainable AI”. In: *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. Oct. 2021, pp. 133–139. DOI: 10.1109/SmartGridComm51999.2021.9632335.
 - [156] C. Kuster, Y. Rezgui, and M. Mourshed. “Electrical load forecasting models: A critical systematic review”. In: *Sustainable Cities and Society* 35 (Nov. 2017), pp. 257–270. ISSN: 22106707. DOI: 10.1016/j.scs.2017.08.009.
 - [157] M. Kuzlu, U. Cali, V. Sharma, and Ö. Güler. “Gaining Insight Into Solar Photovoltaic Power Generation Forecasting Utilizing Explainable Artificial Intelligence Tools”. In: *IEEE Access* 8 (2020), pp. 187814–187823. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3031477.

-
- [158] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. “Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR ’18. Association for Computing Machinery, June 2018, pp. 95–104. ISBN: 978-1-4503-5657-2. DOI: 10.1145/3209978.3210006.
 - [159] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller. “Unmasking Clever Hans predictors and assessing what machines really learn”. In: *Nature Communications* 10.1 (Mar. 2019), p. 1096. ISSN: 2041-1723. DOI: 10.1038/s41467-019-08987-4.
 - [160] P. Laurinec and M. Lucká. “Clustering-based forecasting method for individual consumers electricity load using time series representations”. In: *Open Computer Science* 8.1 (July 2018), pp. 38–50. ISSN: 2299-1093. DOI: 10.1515/comp-2018-0006.
 - [161] A. Le Franc, P. Carpentier, J.-P. Chancelier, and M. De Lara. “EMSx: a numerical benchmark for energy management systems”. In: *Energy Systems* 14.3 (Aug. 2023), pp. 817–843. ISSN: 1868-3967, 1868-3975. DOI: 10.1007/s12667-020-00417-5.
 - [162] Y.-H. Lee, C.-P. Wei, T.-H. Cheng, and C.-T. Yang. “Nearest-neighbor-based approach to time-series classification”. In: *Decis. Support Syst.* 53.1 (Apr. 2012), pp. 207–217. ISSN: 0167-9236. DOI: 10.1016/j.dss.2011.12.014.
 - [163] C. Li, Z. Dong, L. Ding, H. Petersen, Z. Qiu, G. Chen, and D. Prasad. “Interpretable Memristive LSTM Network Design for Probabilistic Residential Load Forecasting”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 69.6 (June 2022), pp. 2297–2310. ISSN: 1558-0806. DOI: 10.1109/TCSI.2022.3155443.
 - [164] M. Li and Y. Wang. “Power load forecasting and interpretable models based on GS_XGBoost and SHAP”. In: *Journal of Physics: Conference Series* 2195.1 (Feb. 2022), p. 012028. ISSN: 1742-6596. DOI: 10.1088/1742-6596/2195/1/012028.
 - [165] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan. “Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. DOI: <https://dl.acm.org/doi/10.5555/3454287.3454758>.
 - [166] Y. Li, P. Guo, and X. Li. “Short-Term Load Forecasting Based on the Analysis of User Electricity Behavior”. In: *Algorithms* 9.4 (Nov. 2016), p. 80. ISSN: 1999-4893. DOI: 10.3390/a9040080.
 - [167] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen. “Foundation Models for Time Series Analysis: A Tutorial and Survey”. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Aug. 2024, pp. 6555–6565. DOI: 10.1145/3637528.3671451.
 - [168] B. Lim, S. O. Arik, N. Loeff, and T. Pfister. “Temporal Fusion Transformers for interpretable multi-horizon time series forecasting”. In: *International Journal of Forecasting* 37.4 (Oct. 2021), pp. 1748–1764. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.03.012.
 - [169] B. Lim and S. Zohren. “Time-series forecasting with deep learning: a survey”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379.2194 (Feb. 2021), p. 20200209. DOI: 10.1098/rsta.2020.0209.

-
- [170] N. Lin, D. Yun, W. Xia, P. Palensky, and P. P. Vergara. “Comparative Analysis of Zero-Shot Capability of Time-Series Foundation Models in Short-Term Load Prediction”. In: *2025 IEEE Kiel PowerTech*. IEEE, June 2025, pp. 1–6. ISBN: 979-8-3315-4397-6. DOI: 10.1109/PowerTech59965.2025.11180251.
 - [171] C. Liu et al. *Moirai 2.0: When Less Is More for Time Series Forecasting*. Feb. 2026. DOI: 10.48550/arXiv.2511.11698.
 - [172] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. “iTTransformer: Inverted Transformers Are Effective for Time Series Forecasting”. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: <https://openreview.net/forum?id=JePfAI8fah> (visited on 11/25/2025).
 - [173] M. López Santos, S. Díaz García, X. García-Santiago, A. Ogando-Martínez, F. Echevarría Camarero, G. Blázquez Gil, and P. Carrasco Ortega. “Deep learning and transfer learning techniques applied to short-term load forecasting of data-poor buildings in local energy communities”. In: *Energy and Buildings* 292 (Aug. 2023), p. 113164. ISSN: 03787788. DOI: 10.1016/j.enbuild.2023.113164.
 - [174] I. Loshchilov and F. Hutter. *Decoupled Weight Decay Regularization*. Jan. 2019. DOI: 10.48550/arXiv.1711.05101.
 - [175] G. Luderer et al. *Die Energiewende kosteneffizient gestalten: Szenarien zur Klimaneutralität 2045*. Tech. rep. Artwork Size: 106 pages, 33,7 MB Medium: application/pdf. Potsdam Institute for Climate Impact Research, Mar. 2025, 106 pages, 33, 7 MB. DOI: 10.48485/PIK.2025.003.
 - [176] S. M. Lundberg and S.-I. Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4768–4777. URL: <https://dl.acm.org/doi/proceedings/10.5555/3295222>.
 - [177] S. M. Lundberg et al. “From local explanations to global understanding with explainable AI for trees”. In: *Nature Machine Intelligence* 2.1 (Jan. 2020), pp. 56–67. ISSN: 2522-5839. DOI: 10.1038/s42256-019-0138-9.
 - [178] R. Machlev, L. Heistrene, M. Perl, K. Y. Levy, J. Belikov, S. Mannor, and Y. Levron. “Explainable Artificial Intelligence (XAI) techniques for energy and power systems: Review, challenges and opportunities”. In: *Energy and AI* 9 (Aug. 2022), p. 100169. ISSN: 2666-5468. DOI: 10.1016/j.egyai.2022.100169.
 - [179] J. Machowski, Z. Lubosny, J. Bialek, and J. R. Bumby. *Power system dynamics: stability and control*. Third edition. John Wiley, 2020. ISBN: 978-1-119-52634-6.
 - [180] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. “M5 accuracy competition: Results, findings, and conclusions”. In: *International Journal of Forecasting*. Special Issue: M5 competition 38.4 (Oct. 2022), pp. 1346–1364. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.11.013.
 - [181] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. “The M4 Competition: 100,000 time series and 61 forecasting methods”. In: *International Journal of Forecasting*. M4 Competition 36.1 (Jan. 2020), pp. 54–74. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.04.014.

-
- [182] N. Maleki, O. Lundström, A. Musaddiq, J. Jeansson, T. Olsson, and F. Ahlgren. “Future energy insights: Time-series and deep learning models for city load forecasting”. In: *Applied Energy* 374 (Nov. 2024), p. 124067. ISSN: 03062619. DOI: 10.1016/j.apenergy.2024.124067.
 - [183] S. Meisenbacher. “Automated machine learning for time series forecasting in smart grid applications”. PhD thesis. Karlsruhe Institute of Technology, 2025. DOI: 10.5445/IR/1000181215.
 - [184] S. Meisenbacher, M. Turowski, K. Phipps, M. Rätz, D. Müller, V. Hagenmeyer, and R. Mikut. “Review of automated time series forecasting pipelines”. In: *WIREs Data Mining and Knowledge Discovery* 12.6 (Nov. 2022), e1475. ISSN: 1942-4787, 1942-4795. DOI: 10.1002/widm.1475.
 - [185] Met Office Hadley Centre - HadCRUT5 - with major processing by Our World in Data. *Annual temperature anomalies relative to the pre-industrial period*. URL: <https://ourworldindata.org/grapher/temperature-anomaly> (visited on 08/29/2025).
 - [186] M. Meyer, D. Zapata, S. Kaltenpoth, and O. Müller. *Benchmarking Time Series Foundation Models for Short-Term Household Electricity Load Forecasting*. Oct. 2024. DOI: 10.48550/arXiv.2410.09487.
 - [187] A. A. Mir, K. Ullah, Z. A. Khan, F. Bashir, T. U. R. Khan, and A. Altamimi. “Short Term Load Forecasting for Electric Power Utilities: A Generalized Regression Approach Using Polynomials and Cross-Terms”. In: *Engineering Proceedings* 12.1 (2021). Number: 1, p. 21. ISSN: 2673-4591. DOI: 10.3390/engproc2021012021.
 - [188] C. Molnar. *Interpreting machine learning models with SHAP: a guide with Python examples and theory on Shapley values*. First Edition. Christoph Molnar c/o MUCBOOK, Heidi Seibold, 2023. ISBN: 979-8-8577-3444-5.
 - [189] P. Montero-Manso and R. J. Hyndman. “Principles and algorithms for forecasting groups of time series: Locality and globality”. In: *International Journal of Forecasting* 37.4 (Oct. 2021), pp. 1632–1653. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2021.03.004.
 - [190] J. Moon, S. Park, S. Rho, and E. Hwang. “Interpretable Short-Term Electrical Load Forecasting Scheme Using Cubist”. In: *Computational Intelligence and Neuroscience* 2022.1 (2022). _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1155/2022/6892995>, p. 6892995. ISSN: 1687-5273. DOI: 10.1155/2022/6892995.
 - [191] L. Moosbrugger, V. Seiler, P. Wohlgenannt, S. Hegenbart, S. Ristov, and P. Kepplinger. *Load Forecasting for Households and Energy Communities: Are Deep Learning Models Worth the Effort?* Jan. 2025. DOI: 10.48550/arXiv.2501.05000.
 - [192] I. Moustati, N. Gherabi, B. E. Elbaghazaoui, and M. Amnai. “Benchmarking Time-Series Forecasting Models for Smart Meter Data”. In: *Advances in Intelligent Systems and Digital Applications*. Vol. 1486. Series Title: Lecture Notes in Networks and Systems. Springer Nature Switzerland, 2025, pp. 104–112. ISBN: 978-3-031-95329-3 978-3-031-95330-9. DOI: 10.1007/978-3-031-95330-9_11.
 - [193] D. Murray, L. Stankovic, and V. Stankovic. “An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study”. In: *Scientific Data* 4.1 (Jan. 2017), p. 160122. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.122.

-
- [194] M. Nauta et al. “From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI”. In: *ACM Comput. Surv.* 55.13s (July 2023), 295:1–295:42. ISSN: 0360-0300. DOI: 10.1145/3583558.
 - [195] M. Nawar, M. Shomer, S. Faddel, and H. Gong. “Transfer Learning in Deep Learning Models for Building Load Forecasting: Case of Limited Data”. In: *SoutheastCon 2023*. Apr. 2023, pp. 532–538. DOI: 10.1109/SoutheastCon51012.2023.10115128.
 - [196] A. Nayebi, S. Tipirneni, C. K. Reddy, B. Foreman, and V. Subbian. “WindowSHAP: An efficient framework for explaining time-series classifiers based on Shapley values”. In: *Journal of Biomedical Informatics* 144 (Aug. 2023), p. 104438. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2023.104438.
 - [197] O. Neumann, M. Beichter, B. Heidrich, N. Friederich, V. Hagenmeyer, and R. Mikut. “Intrinsic Explainable Artificial Intelligence Using Trainable Spatial Weights on Numerical Weather Predictions”. In: *The 15th ACM International Conference on Future and Sustainable Energy Systems*. ACM, June 2024, pp. 551–559. ISBN: 979-8-4007-0480-2. DOI: 10.1145/3632775.3662161.
 - [198] O. Neumann, M. Turowski, R. Mikut, V. Hagenmeyer, and N. Ludwig. “Using weather data in energy time series forecasting: the benefit of input data transformations”. In: *Energy Informatics* 6.1 (Nov. 2023), p. 44. ISSN: 2520-8942. DOI: 10.1186/s42162-023-00299-8.
 - [199] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. “A Time Series is Worth 64 Words: Long-term Forecasting with Transformers”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: <https://openreview.net/forum?id=Jbdc0vT0co1> (visited on 08/13/2025).
 - [200] A. Nikoltchovska, S. Pütz, X. Li, V. Hagenmeyer, and B. Schäfer. “Probabilistic and Explainable Machine Learning for Tabular Power Grid Data”. In: *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems*. ACM, June 2025, pp. 213–231. ISBN: 979-8-4007-1125-1. DOI: 10.1145/3679240.3734623.
 - [201] E. Okoyomon and C. Goebel. “EnergyOS: Designing a Multi-Application EMS Platform”. In: *The 15th ACM International Conference on Future and Sustainable Energy Systems*. ACM, June 2024, pp. 237–255. ISBN: 979-8-4007-0480-2. DOI: 10.1145/3632775.3661961.
 - [202] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. “N-BEATS: Neural basis expansion analysis for interpretable time series forecasting”. In: *International Conference on Learning Representations*. Sept. 2019. URL: <https://openreview.net/forum?id=r1ecqn4YwB> (visited on 08/12/2025).
 - [203] G. Owen. “Values of Games with a Priori Unions”. In: *Mathematical Economics and Game Theory*. Springer, 1977, pp. 76–88. ISBN: 978-3-642-45494-3. DOI: 10.1007/978-3-642-45494-3_7.
 - [204] L. N. Palaniswamy, N. Munzke, C. Kupper, and M. Hiller. “Optimized Energy Management of a Solar and Wind Equipped Student Residence with Innovative Hybrid Energy Storage and Power to Heat Solutions”. In: *Proceedings of the International Renewable Energy Storage Conference (IRES 2022)*. Vol. 16. Series Title: Atlantis Highlights in Engineering. Atlantis Press International

- BV, 2023, pp. 363–382. ISBN: 978-94-6463-155-5 978-94-6463-156-2. DOI: 10.2991/978-94-6463-156-2_24.
- [205] K. Paliwal, A. Agarwal, and S. S. Sinha. “A modification over Sakoe and Chiba’s dynamic time warping algorithm for isolated word recognition”. In: *Signal Processing* 4.4 (July 1982), pp. 329–333. ISSN: 01651684. DOI: 10.1016/0165-1684(82)90009-3.
- [206] C. Panigutti et al. “The role of explainable AI in the context of the AI Act”. In: *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’23. Association for Computing Machinery, June 2023, pp. 1139–1150. ISBN: 979-8-4007-0192-4. DOI: 10.1145/3593013.3594069.
- [207] D. N. Papadopoulos, S. P. Menci, and J. D. Fernandez. “Day-ahead net load forecasting in ZEBs: A comparative ML study with novel feature engineering and simulation-based evaluation”. In: *Energy and Buildings* 363 (July 2026), p. 117563. ISSN: 03787788. DOI: 10.1016/j.enbuild.2026.117563.
- [208] J. Paparrizos and L. Gravano. “k-Shape: Efficient and Accurate Clustering of Time Series”. In: *ACM SIGMOD Record* 45.1 (June 2016), pp. 69–76. ISSN: 0163-5808. DOI: 10.1145/2949741.2949758.
- [209] *Pecan Street Dataport*. 2018. URL: <https://www.pecanstreet.org/dataport/>.
- [210] S. Pelekis, I.-K. Seisopoulos, E. Spiliotis, T. Pountridis, E. Karakolis, S. Mouzakitis, and D. Askounis. “A comparative assessment of deep learning models for day-ahead load forecasting: Investigating key accuracy drivers”. In: *Sustainable Energy, Grids and Networks* 36 (Dec. 2023), p. 101171. ISSN: 23524677. DOI: 10.1016/j.segan.2023.101171.
- [211] K. Phipps. “Quantifying and Interpreting Uncertainty in Time Series Forecasting”. PhD thesis. Karlsruhe Institute of Technology, 2024.
- [212] K. Phipps, B. Heidrich, M. Turowski, M. Wittig, R. Mikut, and V. Hagenmeyer. “Generating probabilistic forecasts from arbitrary point forecasts using a conditional invertible neural network”. In: *Applied Intelligence* 54.8 (Apr. 2024), pp. 6354–6382. ISSN: 0924-669X, 1573-7497. DOI: 10.1007/s10489-024-05346-9.
- [213] G. Pinto, Z. Wang, A. Roy, T. Hong, and A. Capozzoli. “Transfer learning for smart buildings: A critical review of algorithms, applications, and future perspectives”. In: *Advances in Applied Energy* 5 (Feb. 2022), p. 100084. ISSN: 2666-7924. DOI: 10.1016/j.adapen.2022.100084.
- [214] Priyadarshan, C. Crozier, K. Baker, and K. Kircher. *Distribution Grids May Be a Barrier To Residential Electrification*. Aug. 2025. DOI: 10.48550/arXiv.2410.04540.
- [215] M. Pullinger et al. “The IDEAL household energy dataset, electricity, gas, contextual sensor data and survey data for 255 UK homes”. In: *Scientific Data* 8.1 (May 2021), p. 146. ISSN: 2052-4463. DOI: 10.1038/s41597-021-00921-y.
- [216] S. Pütz, H. El Ashhab, M. Hertel, R. Mikut, M. Götz, V. Hagenmeyer, and B. Schäfer. “Feasibility of Forecasting Highly Resolved Power Grid Frequency Utilizing Temporal Fusion Transformers”. In: *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. e-Energy ’24. Association for Computing Machinery, 2024, pp. 447–453. ISBN: 979-8-4007-0480-2. DOI: 10.1145/3632775.3661963.

-
- [217] F. L. Quilumba, W.-J. Lee, H. Huang, D. Y. Wang, and R. L. Szabados. “Using Smart Meter Data to Improve the Accuracy of Intraday Load Forecasting Considering Customer Behavior Similarities”. In: *IEEE Transactions on Smart Grid* 6.2 (Mar. 2015), pp. 911–918. ISSN: 1949-3053, 1949-3061. DOI: 10.1109/TSG.2014.2364233.
 - [218] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. *Robust Speech Recognition via Large-Scale Weak Supervision*. Dec. 2022. DOI: 10.48550/arXiv.2212.04356.
 - [219] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. “Language Models are Unsupervised Multitask Learners”. In: *OpenAI* (2019). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
 - [220] F. Raisch, T. Krug, C. Goebel, and B. Tischler. “GenTL: A General Transfer Learning Model for Building Thermal Dynamics”. In: *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems*. ACM, June 2025, pp. 322–333. ISBN: 979-8-4007-1125-1. DOI: 10.1145/3679240.3734589.
 - [221] R. Ramanathan, R. Engle, C. W. J. Granger, F. Vahid-Araghi, and C. Brace. “Short-run forecasts of electricity loads and peaks”. In: *International Journal of Forecasting* 13.2 (June 1997), pp. 161–174. ISSN: 0169-2070. DOI: 10.1016/S0169-2070(97)00015-0.
 - [222] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. “Putting the ‘smarts’ into the smart grid: a grand challenge for artificial intelligence”. In: *Commun. ACM* 55.4 (Apr. 2012), pp. 86–97. ISSN: 0001-0782. DOI: 10.1145/2133806.2133825.
 - [223] K. Rasul et al. *Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting*. 2023. DOI: 10.48550/ARXIV.2310.08278.
 - [224] C. Ren, Y. Xu, and R. Zhang. “An Interpretable Deep Learning Method for Power System Transient Stability Assessment via Tree Regularization”. In: *IEEE Transactions on Power Systems* 37.5 (Sept. 2022), pp. 3359–3369. ISSN: 1558-0679. DOI: 10.1109/TPWRS.2021.3133611.
 - [225] M. T. Ribeiro, S. Singh, and C. Guestrin. “‘Why Should I Trust You?’: Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. Association for Computing Machinery, Aug. 2016, pp. 1135–1144. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778.
 - [226] H. Ritchie, P. Rosado, and M. Roser. *Breakdown of carbon dioxide, methane and nitrous oxide emissions by sector*. 2020. URL: <https://ourworldindata.org/emissions-by-sector> (visited on 08/04/2025).
 - [227] H. Ritchie, P. Rosado, and V. Samborska. *Data Page: Annual concentration of atmospheric carbon dioxide*. 2025. URL: <https://ourworldindata.org/grapher/co2-long-term-concentration> (visited on 05/16/2025).
 - [228] H. Ritchie and M. Roser. *CO emissions*. 2021. URL: <https://ourworldindata.org/co2-emissions> (visited on 09/01/2026).
 - [229] F. Rodrigues and A. Trindade. “Load forecasting through functional clustering and ensemble learning”. In: *Knowledge and Information Systems* 57.1 (Oct. 2018), pp. 229–244. ISSN: 0219-3116. DOI: 10.1007/s10115-018-1169-y.

-
- [230] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Díaz-Rodríguez. *Explainable Artificial Intelligence (XAI) on Time Series Data: A Survey*. Apr. 2021. DOI: 10.48550/arXiv.2104.00950.
 - [231] P. J. Rousseeuw. “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of Computational and Applied Mathematics* 20 (Nov. 1987), pp. 53–65. ISSN: 03770427. DOI: 10.1016/0377-0427(87)90125-7.
 - [232] H. J. Sadaei, P. C. De Lima E Silva, F. G. Guimarães, and M. H. Lee. “Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series”. In: *Energy* 175 (May 2019), pp. 365–377. ISSN: 03605442. DOI: 10.1016/j.energy.2019.03.081.
 - [233] N. Saengngam and U. Tonggumnead. “Predicting Thailand Electricity Load Demand with Modified Fourier Series and Seasonal-Trend Decomposition Methods Using LOESS Transformation”. In: *Applied Science and Engineering Progress* 16.2 (2023). DOI: 10.14416/j.asep.2022.02.011.
 - [234] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. “DeepAR: Probabilistic forecasting with autoregressive recurrent networks”. In: *International Journal of Forecasting* 36.3 (July 2020), pp. 1181–1191. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.07.001.
 - [235] H. K. Saravanan, S. Dwivedi, and P. Arjunan. “Analyzing the Performance of Time Series Foundation Models for Short-term Load Forecasting”. In: *Proceedings of the 11th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. BuildSys ’24. Association for Computing Machinery, 2024, pp. 237–238. ISBN: 979-8-4007-0706-3. DOI: 10.1145/3671127.3698708.
 - [236] S. Sarp, M. Kuzlu, U. Cali, O. Elma, and O. Guler. “An Interpretable Solar Photovoltaic Power Generation Forecasting Approach Using An Explainable Artificial Intelligence Tool”. In: *2021 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. Feb. 2021, pp. 1–5. DOI: 10.1109/ISGT49243.2021.9372263.
 - [237] F. v. Scheidt, H. Medinová, N. Ludwig, B. Richter, P. Staudt, and C. Weinhardt. “Data analytics in the electricity sector – A quantitative and qualitative literature review”. In: *Energy and AI* 1 (Aug. 2020), p. 100009. ISSN: 26665468. DOI: 10.1016/j.egyai.2020.100009.
 - [238] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim. “Towards A Rigorous Evaluation Of XAI Methods On Time Series”. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. Oct. 2019, pp. 4197–4201. DOI: 10.1109/ICCVW.2019.00516.
 - [239] U. Schlegel and D. A. Keim. “A Deep Dive into Perturbations as Evaluation Technique for Time Series XAI”. In: *Explainable Artificial Intelligence*. Springer Nature Switzerland, 2023, pp. 165–180. ISBN: 978-3-031-44070-0. DOI: 10.1007/978-3-031-44070-0_9.
 - [240] M. Schlemminger, T. Ohrdes, E. Schneider, and M. Knoop. “Dataset on electrical single-family house and heat pump load profiles in Germany”. In: *Scientific Data* 9.1 (Feb. 2022), p. 56. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01156-1.
 - [241] A. J. Schwab. *Elektroenergiesysteme: Smarte Stromversorgung im Zeitalter der Energiewende*. Berlin, Heidelberg: Springer, 2022. ISBN: 978-3-662-64773-8 978-3-662-64774-5. DOI: 10.1007/978-3-662-64774-5.

-
- [242] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.
 - [243] L. Semmelmann. “Dynamic Tariff Adoption in Electricity Markets: Bridging Policy, Technology and Behavior”. PhD thesis. Karlsruhe Institute of Technology, 2025. DOI: 10.5445/IR/1000181556.
 - [244] L. Semmelmann, M. Hertel, K. J. Kircher, R. Mikut, V. Hagenmeyer, and C. Weinhardt. “The impact of heat pumps on day-ahead energy community load forecasting”. In: *Applied Energy* 368 (Aug. 2024), p. 123364. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2024.123364.
 - [245] P. Senin and S. Malinchik. “SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model”. In: *2013 IEEE 13th International Conference on Data Mining*. IEEE, Dec. 2013, pp. 1175–1180. ISBN: 978-0-7695-5108-1. DOI: 10.1109/ICDM.2013.52.
 - [246] K. Shahare, A. Mitra, D. Naware, R. Keshri, and H. Suryawanshi. “Performance analysis and comparison of various techniques for short-term load forecasting”. In: *Energy Reports* 9 (Mar. 2023), pp. 799–808. ISSN: 23524847. DOI: 10.1016/j.egy.2022.11.086.
 - [247] A. Shahzadeh, A. Khosravi, and S. Nahavandi. “Improving load forecast accuracy by clustering consumers using smart meter data”. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2015, pp. 1–7. ISBN: 978-1-4799-1960-4. DOI: 10.1109/IJCNN.2015.7280393.
 - [248] Shapley, Lloyd S. “A Value for n-person Games”. In: *Contributions to the Theory of Games. Annals of Mathematical Studies. Princeton University Press*. 28 (1953), pp. 307–317. DOI: <https://doi.org/10.1515/9781400881970-018>.
 - [249] O. Shchur, A. F. Ansari, C. Turkmen, L. Stella, N. Erickson, P. Guerron, M. Bohlke-Schneider, and Y. Wang. *fev-bench: A Realistic Benchmark for Time Series Forecasting*. 2025. DOI: 10.48550/ARXIV.2509.26468.
 - [250] H. Shi, K. Miao, and X. Ren. “Short-term load forecasting based on CNN-BiLSTM with Bayesian optimization and attention mechanism”. In: *Concurrency and Computation: Practice and Experience* 35.17 (Aug. 2023), e6676. ISSN: 1532-0626, 1532-0634. DOI: 10.1002/cpe.6676.
 - [251] A. Shrikumar, P. Greenside, and A. Kundaje. “Learning important features through propagating activation differences”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML’17. JMLR.org*, Aug. 2017, pp. 3145–3153. URL: <https://dl.acm.org/doi/10.5555/3305890.3306006> (visited on 02/27/2026).
 - [252] S. Shukla and T. Hong. “BigDEAL Challenge 2022: Forecasting peak timing of electricity demand”. In: *IET Smart Grid* 7.4 (Aug. 2024), pp. 442–459. ISSN: 2515-2947, 2515-2947. DOI: 10.1049/stg2.12162.
 - [253] S. A. Siddiqui, D. Mercier, M. Munir, A. Dengel, and S. Ahmed. “TSViz: Demystification of Deep Learning Models for Time-Series Analysis”. In: *IEEE Access* 7 (2019), pp. 67027–67040. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2912823.

- [254] G. Sideratos, A. Ikonopoulou, and N. D. Hatziaargyriou. “A novel fuzzy-based ensemble model for load forecasting using hybrid deep neural networks”. In: *Electric Power Systems Research* 178 (Jan. 2020), p. 106025. ISSN: 03787796. DOI: 10.1016/j.epsr.2019.106025.
- [255] Q. Song and B. S. Chissom. “Fuzzy time series and its models”. In: *Fuzzy Sets and Systems* 54.3 (Mar. 1993), pp. 269–277. ISSN: 0165-0114. DOI: 10.1016/0165-0114(93)90372-0.
- [256] P. Staudt, N. Ludwig, J. Huber, V. Hagenmeyer, and C. Weinhardt. “SCiBER: A new public data set of municipal building consumption”. In: *Proceedings of the Ninth International Conference on Future Energy Systems*. ACM, June 2018, pp. 618–621. ISBN: 978-1-4503-5767-8. DOI: 10.1145/3208903.3210281.
- [257] D. Syed, H. Abu-Rub, A. Ghayeb, S. S. Refaat, M. Houchati, O. Bouhali, and S. Banales. “Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid With Clustering and Consumption Pattern Recognition”. In: *IEEE Access* 9 (2021), pp. 54992–55008. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3071654.
- [258] TenneT. *Forecasts: usefulness and necessity*. 2021. URL: https://netztransparenz.tennet.eu/fileadmin/user_upload/Company/Publications/Technical_Publications/English/Forecasts_SO_FEB2021_EN_def.pdf (visited on 08/06/2025).
- [259] A. Theissler, F. Spinnato, U. Schlegel, and R. Guidotti. “Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions”. In: *IEEE Access* 10 (2022), pp. 100700–100724. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3207765.
- [260] C. Thelen, H. Nolte, M. Kaiser, P. Jürgens, P. Müller, C. Senkpiel, and C. Kost. “Wege zu einem klimaneutralen Energiesystem - Bundesländer im Transformationsprozess”. In: *Fraunhofer ISE* (2024). DOI: 10.24406/publica-4564.
- [261] M. Torabi, S. Hashemi, M. R. Saybani, S. Shamshirband, and A. Mosavi. “A Hybrid clustering and classification technique for forecasting short-term energy consumption”. In: *Environmental Progress & Sustainable Energy* 38.1 (Jan. 2019), pp. 66–76. ISSN: 1944-7442, 1944-7450. DOI: 10.1002/ep.12934.
- [262] J.-F. Toubreau, J. Bottieau, Y. Wang, and F. Vallée. “Interpretable Probabilistic Forecasting of Imbalances in Renewable-Dominated Electricity Systems”. In: *IEEE Transactions on Sustainable Energy* 13.2 (Apr. 2022), pp. 1267–1277. ISSN: 1949-3037. DOI: 10.1109/TSTE.2021.3092137.
- [263] M. Tretschock et al. *Gutachten zur Weiterentwicklung der Strom-Verteilnetze in Nordrhein-Westfalen auf Grund einer fortschreitenden Sektorenkopplung und neuer Verbraucher*. Tech. rep. 2021. URL: https://www.wirtschaft.nrw/sites/default/files/documents/210609_nrw_verteilnetzstudie_final.pdf (visited on 08/20/2025).
- [264] M. Treutlein, P. Bothe, M. Schmidt, R. Hahn, O. Neumann, R. Mikut, and V. Hagenmeyer. *Real-world energy data of 200 feeders from low-voltage grids with metadata in Germany over two years*. Feb. 2026. DOI: 10.48550/arXiv.2602.03521.
- [265] M. Treutlein, B. Kaas, and et al. “Low Voltage Peak Load Forecasting with Probabilistic Time Series Foundation Models”. In: *Under Review*. 2026.

-
- [266] M. Treutlein, M. Schmidt, R. Hahn, M. Hertel, B. Heidrich, R. Mikut, and V. Hagenmeyer. “Generating peak-aware pseudo-measurements for low-voltage feeders using metadata of distribution system operators”. In: *IET Smart Grid* 8 (Feb. 2025). DOI: 10.1049/stg2.12210.
 - [267] G. Tziolis, J. Lopez-Lorente, M.-I. Baka, A. Koumis, A. Livera, S. Theocharides, G. Makrides, and G. E. Georghiou. “Direct short-term net load forecasting in renewable integrated microgrids using machine learning: A comparative assessment”. In: *Sustainable Energy, Grids and Networks* 37 (Mar. 2024), p. 101256. ISSN: 23524677. DOI: 10.1016/j.segan.2023.101256.
 - [268] UK Power Networks. *Smartmeter Energy Consumption Data in London Households*. 2015. URL: <https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households>.
 - [269] UNFCCC. *Paris Agreement*. 2015. URL: <https://unfccc.int/documents/184656>.
 - [270] A. Upadhyay, D. Garg, and M. Singh. “Short Term Load Forecasting for Smart Grids Using Apache Spark and a Modified Transformer Model”. In: *Computing and Informatics* 42.1 (2023), pp. 75–97. ISSN: 2585-8807. DOI: 10.31577/cai_2023_1_75.
 - [271] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. URL: <https://dl.acm.org/doi/10.5555/3295222.3295349> (visited on 10/25/2023).
 - [272] M. Voß, C. Bender-Saebelkampff, and S. Albayrak. “Residential Short-Term Load Forecasting Using Convolutional Neural Networks”. In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. Oct. 2018, pp. 1–6. DOI: 10.1109/SmartGridComm.2018.8587494.
 - [273] W. Waheed and Q. Xu. “Data-driven short term load forecasting with deep neural networks: Unlocking insights for sustainable energy management”. In: *Electric Power Systems Research* 232 (July 2024), p. 110376. ISSN: 03787796. DOI: 10.1016/j.epsr.2024.110376.
 - [274] J. Wang, Z. Peng, X. Wang, C. Li, and J. Wu. “Deep Fuzzy Cognitive Maps for Interpretable Multivariate Time Series Prediction”. In: *IEEE Transactions on Fuzzy Systems* 29.9 (Sept. 2021), pp. 2647–2660. ISSN: 1941-0034. DOI: 10.1109/TFUZZ.2020.3005293.
 - [275] L. Wang, S. Mao, B. M. Wilamowski, and R. M. Nelms. “Ensemble Learning for Load Forecasting”. In: *IEEE Transactions on Green Communications and Networking* 4.2 (June 2020), pp. 616–628. ISSN: 2473-2400. DOI: 10.1109/TGCN.2020.2987304.
 - [276] Y. Wang, Q. Chen, M. Sun, C. Kang, and Q. Xia. “An Ensemble Forecasting Method for the Aggregated Load With Subprofiles”. In: *IEEE Transactions on Smart Grid* 9.4 (July 2018), pp. 3906–3908. ISSN: 1949-3053, 1949-3061. DOI: 10.1109/TSG.2018.2807985.
 - [277] Y. Wang et al. *TimeXer: Empowering Transformers for Time Series Forecasting with Exogenous Variables*. Nov. 2024. DOI: 10.48550/arXiv.2402.19072.
 - [278] Z. Wang, W. Yan, and T. Oates. “Time series classification from scratch with deep neural networks: A strong baseline”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. May 2017, pp. 1578–1585. DOI: 10.1109/IJCNN.2017.7966039.

-
- [279] Z. Wang, Z. Zhu, G. Xiao, B. Bai, and Y. Zhang. “A Transformer-Based Multi-Entity Load Forecasting Method for Integrated Energy Systems”. In: *Frontiers in Energy Research* 10 (July 2022). ISSN: 2296-598X. DOI: 10.3389/fenrg.2022.952420.
 - [280] Z. Wang, F. Kong, S. Feng, M. Wang, X. Yang, H. Zhao, D. Wang, and Y. Zhang. *Is Mamba Effective for Time Series Forecasting?* 2024. DOI: 10.48550/ARXIV.2403.11144.
 - [281] M. Weber, M. Turowski, H. K. Çakmak, R. Mikut, U. Kühnapfel, and V. Hagenmeyer. “Data-Driven Copy-Paste Imputation for Energy Time Series”. In: *IEEE Transactions on Smart Grid* 12.6 (Nov. 2021), pp. 5409–5419. ISSN: 1949-3061. DOI: 10.1109/TSG.2021.3101831.
 - [282] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. “Transformers in Time Series: A Survey”. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Aug. 2023, pp. 6778–6786. ISBN: 978-1-956792-03-4. DOI: 10.24963/ijcai.2023/759.
 - [283] S. Wenninger, C. Kaymakci, and C. Wiethe. “Explainable long-term building energy consumption prediction using QLattice”. In: *Applied Energy* 308 (Feb. 2022), p. 118300. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2021.118300.
 - [284] D. Werling, M. Beichter, B. Heidrich, K. Phipps, R. Mikut, and V. Hagenmeyer. “The Impact of Forecast Characteristics on the Forecast Value for the Dispatchable Feeder”. In: *Companion Proceedings of the 14th ACM International Conference on Future Energy Systems*. ACM, June 2023, pp. 59–71. ISBN: 979-8-4007-0227-3. DOI: 10.1145/3599733.3600251.
 - [285] D. Werling, B. Heidrich, H. K. Çakmak, and V. Hagenmeyer. “Towards line-restricted dispatchable feeders using probabilistic forecasts for PV-dominated low-voltage distribution grids”. In: *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*. ACM, June 2022, pp. 395–400. ISBN: 978-1-4503-9397-3. DOI: 10.1145/3538637.3538868.
 - [286] S. Wiegrefe and Y. Pinter. “Attention is not not Explanation”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Nov. 2019, pp. 11–20. DOI: 10.18653/v1/D19-1002.
 - [287] F. Wiese et al. “Open Power System Data – Frictionless data for electricity system modelling”. In: *Applied Energy* 236 (Feb. 2019), pp. 401–409. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2018.11.097.
 - [288] T. K. Wijaya, M. Vasirani, S. Humeau, and K. Aberer. “Cluster-based aggregate forecasting for residential electricity demand using smart meter data”. In: *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, Oct. 2015, pp. 879–887. ISBN: 978-1-4799-9926-2. DOI: 10.1109/BigData.2015.7363836.
 - [289] S. Witherspoon and W. Fadrhonc. *Machine learning can boost the value of wind energy*. Feb. 2019. URL: <https://deepmind.google/blog/machine-learning-can-boost-the-value-of-wind-energy/> (visited on 04/26/2026).
 - [290] G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo. *Unified Training of Universal Time Series Forecasting Transformers*. May 2024. DOI: 10.48550/arXiv.2402.02592.

-
- [291] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. *TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis*. Apr. 2023. DOI: 10.48550/arXiv.2210.02186.
 - [292] H. Wu, J. Xu, J. Wang, and M. Long. “Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 22419–22430. URL: <https://dl.acm.org/doi/10.5555/3540261.3541978> (visited on 10/25/2023).
 - [293] K. Wu, C. Xu, M. Zhang, X. Hu, Y. Jin, and Z. Qian. “RCM: Residue-aware Consolidation for Heterogeneous MLaaS Cluster”. In: *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. Nov. 2022, pp. 24–31. DOI: 10.1109/IPCCC55026.2022.9894301.
 - [294] N. Wu, B. Green, X. Ben, and S. O’Banion. *Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case*. Jan. 2020. DOI: 10.48550/arXiv.2001.08317.
 - [295] L. Ye and E. Keogh. “Time series shapelets: a novel technique that allows accurate, interpretable and fast classification”. In: *Data Mining and Knowledge Discovery* 22.1 (Jan. 2011), pp. 149–182. ISSN: 1573-756X. DOI: 10.1007/s10618-010-0179-5.
 - [296] A. Zeng, M. Chen, L. Zhang, and Q. Xu. “Are Transformers Effective for Time Series Forecasting?”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.9 (June 2023), pp. 11121–11128. ISSN: 2374-3468. DOI: 10.1609/aaai.v37i9.26317.
 - [297] G. Zhang, C. Wei, C. Jing, and Y. Wang. “Short-Term Electrical Load Forecasting Based on Time Augmented Transformer”. In: *International Journal of Computational Intelligence Systems* 15.1 (Aug. 2022), p. 67. ISSN: 1875-6883. DOI: 10.1007/s44196-022-00128-y.
 - [298] L. Zhang and D. Jánošík. “Enhanced short-term load forecasting with hybrid machine learning models: CatBoost and XGBoost approaches”. In: *Expert Systems with Applications* 241 (May 2024), p. 122686. ISSN: 09574174. DOI: 10.1016/j.eswa.2023.122686.
 - [299] Y. Zhang, Q. Sun, D. Qi, J. Liu, R. Ma, and O. Petrosian. “ShapTime: A General XAI Approach for Explainable Time Series Forecasting”. In: *Intelligent Systems and Applications*. Springer Nature Switzerland, 2024, pp. 659–673. ISBN: 978-3-031-47721-8. DOI: 10.1007/978-3-031-47721-8_45.
 - [300] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.12 (May 2021), pp. 11106–11115. ISSN: 2374-3468. DOI: 10.1609/aaai.v35i12.17325.
 - [301] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. “FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting”. In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, June 2022, pp. 27268–27286. URL: <https://proceedings.mlr.press/v162/zhou22g.html> (visited on 10/25/2023).
 - [302] A. Zurborg. “Unlocking Customer Value: The Virtual Power Plant”. In: *worldPower* (2010). URL: https://www.energy.gov/sites/default/files/oeprod/DocumentsandMedia/ABB_Attachment.pdf (visited on 09/24/2025).