Journal Title
XX(X):2–22

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Enforcing Interpretability using Concept Bottleneck Transformer Layers for Time Series

Angela van Sprang[1], Erman Acar[1] and Willem Zuidema[1]

Abstract

By combining deep learning with symbolic reasoning, neuro-symbolic AI can yield more explainable models while preserving expressive capacity. Temporal data is a particularly important domain for these efforts, as it captures dynamic concepts that evolve over time. Time series are arguably the most general type of temporal data, yet neuro-symbolic approaches to time series transformers remain limited. In this work, we introduce a *Concept Bottleneck Transformer Layer* (CBTL), an encoder layer whose hidden representations are trained to encode predefined, human-interpretable concepts. We employ a Centered Kernel Alignment (CKA) regularizer to allow the learned representations to be similar, but not identical, to the concepts. We apply our framework to a common time series transformer (Autoformer) and use a simple linear surrogate model and time-of-day information as interpretable concepts. We evaluate on synthetic data and six long-term time series benchmarks and find that predictive performance remains largely unaffected while interpretability is substantially improved. In particular, using activation patching, we causally intervene on the CBTL and obtain evidence that its components represent the predefined concepts. This illustrates how high-level concepts can be localized in a dedicated transformer layer and directly intervened on, rather than relying solely on post-hoc explanations.

Keywords

Concept Bottleneck Models, Time Series Forecasting, Transformers, Interpretability by Design, Neuro-Symbolic AI

Prepared using sagej.cls *[Version: 2017/01/17 v1.20]*

Introduction

The fields of neuro-symbolic AI and mechanistic interpretability share closely aligned ambitions, yet approach them from different directions. Neuro-symbolic AI explicitly combines deep learning with symbolic reasoning, with the goal of grounding neural computations in symbolic representations. This can enhance model explainability. Examples include combining knowledge graphs with LLMs; including logic rules in a network or algorithm, or training neural networks with (fuzzy) logic operators (Diamantini et al. 2026; Manhaeve et al. 2018; Van Krieken et al. 2022). Other works emphasize the role of concepts as basic units in reasoning and build neuro-symbolic concept representations (Mao et al. 2025).

In contrast, mechanistic interpretability typically focuses on reverse engineering the internal mechanisms of deep learning models. It usually involves formulating and testing hypotheses on the granularity and type of internal representations that correspond to interpretable concepts. Different techniques focus on attention heads, circuits, or dictionary features to uncover structure in model representations (Zheng et al. 2025; Conmy et al.; Shu et al. 2025).

Despite their shared goal of bridging human-understandable concepts and model internals, there is only limited, and very recent, work that explicitly combines neuro-symbolic AI and mechanistic interpretability (Helff et al. 2025). In this work, we take a neuro-symbolic approach to enforce transformer interpretability, aiming not only to reveal causal mechanisms in deep learning models but also to integrate them as explicit, concept-level components in the architecture.

In particular, we propose a **symbolic training framework** for time series transformers using time series specific, yet domain-agnostic concepts, as shown in Figure 1. A key aspect of our training framework is to leave the model's architecture intact, while encouraging the learned representations of the **concept bottleneck transformer layer (CBTL)** to be similar, but not identical, to the interpretable concepts. The interpretable concepts are enforced with the similarity measure Centered Kernel Alignment (CKA; Kornblith et al. 2019), which is included in the loss function as a regularizer. The first concept is a simple, linear surrogate model and the second is time information (e.g. hour-of-day).

We apply our concept bottleneck framework to Autoformer (Wu et al. 2021), which is a common time series transformer. We perform extensive experiments on seven datasets in the domains of electricity, traffic, weather, illness, exchange rate and (physical) electricity transformers. We show that our setup results in models that are more transparent while

[1]University of Amsterdam, the Netherlands

Corresponding author:
Angela van Sprang
Email: a.v.vansprang@uva.nl

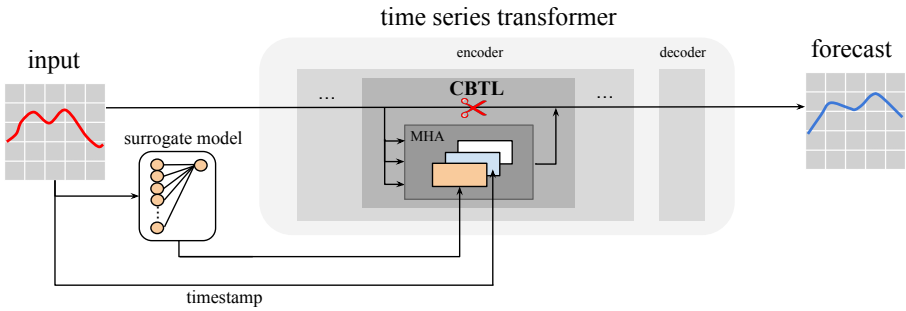


Figure 1. Overview of the symbolic training framework using a concept bottleneck transformer layer (CBTL). The CBTL is one encoder layer which is trained to be similar to predefined, interpretable concepts. The residual stream around the bottleneck is removed, such that all information passes through the bottleneck.

the overall performance remains largely unaffected. Furthermore, we causally test the obtained interpretations with an intervention study using activation patching.

Our contributions are summarized as follows:

1. We propose a novel symbolic training framework using a concept bottleneck transformer layer (CBTL) to enhance the interpretability of transformers for time series.
2. We demonstrate the feasibility of applying this framework to time series transformers by conducting extensive experiments on seven datasets for Autoformer, and identify interpretable concepts in each of these transformers.
3. We assess the faithfulness of the interpretability analysis by performing an activation patching experiment, and obtain evidence that the identified components (in the concept bottleneck transformer layer) indeed have the hypothesized unique and causal role in the predictions of the target model.

Background and Related Work

This paper combines and builds upon foundational works from different fields, including neuro-symbolic AI, CBMs, knowledge transfer with CKA and time series transformers. CBMs have been applied to time series before (Ferfoglia et al. 2024), but not with the same interpretable concepts. Likewise, the similarity index CKA has been used before to transfer knowledge between models (Tian et al. 2023), yet, to the best of our knowledge, it has not been used to construct a CBM. This makes our work a unique contribution at the intersection of (mechanistic) interpretability, concept learning, and neuro-symbolic AI.

Concept Bottleneck Models

Concept bottleneck models (CBMs; Koh et al. 2020) have emerged as promising interpretable models, but are not usually considered neuro-symbolic AI (Poeta et al. 2023). Here, the concept bottleneck constrains the model to first predict interpretable concepts, and then to use only these concepts in the final downstream task. They are shown to be useful in multiple applications, such as model debugging and human intervention. The bottleneck allows for explaining which information the model is using and when it makes an error due to incorrect concept predictions.

One of the shortcomings of standard CBMs is that concept annotations are needed during training to learn the bottleneck, and concept labels do not necessarily contain all information needed to accurately perform the downstream task, and can therefore decrease the task accuracy (Mahinpei et al. 2021). Therefore, Zarlenga et al. (2022) propose Concept Embedding Models, where concepts are represented as vectors, such that richer and more meaningful concept semantics can be captured. CBMs and their variants are usually applied to the field of computer vision, and less frequently to natural language (Tan et al. 2024), graphs (Barbiero et al. 2023) or tabular data (Zarlenga et al. 2022). In principle, the methodology can be applied to time series as well, but defining high-level, meaningful concepts is challenging. Ferfoglia et al. (2024) use Signal Temporal Logic (STL) formulas as concept embeddings for time series to convert them into natural language, and use these concepts as bottleneck for anomaly detection.

A line of work on CBMs for time series was intiated by our earlier preprint (Sprang et al. 2025). Since then, several authors have adopted and extended this idea. For example, Forest et al. (2025) introduce a time series concept bottleneck where the concepts are learned by masking the input and Ma et al. (2025) introduce a bottleneck with a small set of physically motivated concepts.

Knowledge Transfer with Centered Kernel Alignment

Inspired by neuroscience, CKA measures the similarity between different representations from neural networks (Kornblith et al. 2019). By factoring out differences in scaling or orthogonal transformations, CKA captures intuitive notions of similarity between representations. To obtain the score, firstly, the similarity between every pair of examples in each representation separately is measured using a predefined kernel, and then the obtained similarity structures are compared. We use a linear kernel, which makes the CKA score defined as follows for representations X and Y using the Frobenius norm:

$$\mathrm{CKA}(X, Y) = \frac{\left\|Y^\top X\right\|_F^2}{\left\|X^\top X\right\|_F \left\|Y^\top Y\right\|_F}. \tag{1}$$

The CKA score can be used to transfer knowledge between different models when included in the loss function (Tian et al. 2023). In this work, the authors study knowledge distillation between a teacher and student model, and incorporate CKA into the loss function to transfer feature representation knowledge from the pretrained model to the incremental learning model (Parisi et al. 2019).

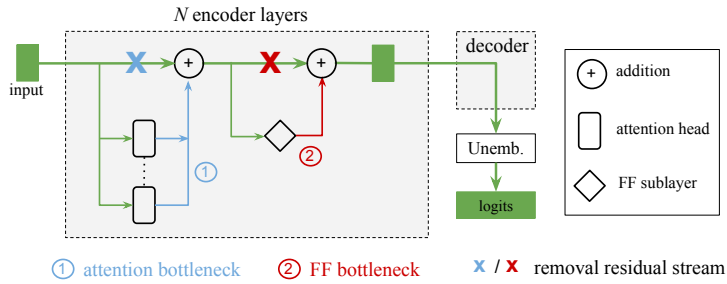


Figure 2. The CBTL can be on the attention (blue) or the feed-forward (red) module. Note that the residual connection is removed at the location of the bottleneck (and the residual stream thus interrupted). Visualisation inspired by Rai et al. 2024.

Time Series Transformers

Time series transformers for long-term time series forecasting, such as Autoformer, obtain two types of input: (1) data values $\boldsymbol{X} \in \mathbb{R}^{I \times d}$, and (2) timestamps $\boldsymbol{T} \in \mathbb{R}^{I \times 4}$. More specifically, they can be regarded as a function $f : \mathbb{R}^{I \times d} \times \mathbb{R}^{I \times 4} \times \mathbb{R}^{O \times 4} \rightarrow \mathbb{R}^{O \times d}$, where I is the number of input time steps, O is the number of future time steps, and d is the number of variables in the time series. The additional four dimensions of timestamps $\boldsymbol{T}$ represent four time features, namely hour-of-day, day-of-week, day-of-month, and day-of-year. The future timestamps are also provided, for which the model should forecast the future data values. Note that we explicitly introduce a notation for the timestamps to later define the CKA scores and the intervention.

Method

We propose a training framework to make any transformer model interpretable by including a bottleneck based on knowledge transfer with CKA (Kornblith et al. 2019), as shown in Figure 2. The main idea is that we assign one of the encoder layers to be the CBTL, whose representations are subject to a soft constraint of being as similar as possible to predefined interpretable concepts. To this end, we calculate CKA scores with the interpretable concepts, and include these scores in the loss function.

Loss Function

The loss function encourages the model to represent the interpretable concepts in the bottleneck layer. Therefore, we add a term $\mathcal{L}_{CKA}$ based on the CKA scores of the bottleneck and the interpretable concepts (Eq. 3). In particular, low similarity between the bottleneck and the interpretable concepts results in a higher value for $\mathcal{L}_{CKA}$. The total loss function $\mathcal{L}_{Total}$ (Eq. 2), then, is a weighted average of the Mean Squared Error

(MSE) loss $\mathcal{L}_{MSE}$ and the CKA loss $\mathcal{L}_{CKA}$:

$$\mathcal{L}_{Total} = (1-\alpha)\,\mathcal{L}_{MSE} + \alpha\,\mathcal{L}_{CKA}, \tag{2}$$

$$\mathcal{L}_{CKA} = 1 - \frac{1}{c}\sum_{i=1}^{c} CKA_i, \tag{3}$$

where α is a hyperparameter, c is the number of concepts, and $CKA_i \in [0,1]$ is the CKA score (using a linear kernel) between the model's representation and concept i (see Interpretability Analysis Section).

Interpretable Concepts in the Bottleneck

In this section, we describe how to calculate the CKA score to measure the presence of a concept. We refer to the Appendix for a more detailed description of the concept bottleneck framework.

Location bottleneck. We assign one encoder layer to be the bottleneck layer, because the encoder focuses on modelling seasonal information. Within the bottleneck layer, the latent representations can be taken from two different types of blocks: the attention block ($\tau = Att$) and the feed-forward block ($\tau = FF$). These two options are illustrated in Figure 2. We assign c interpretable concepts over the latent representations, with the goal of teaching the corresponding model component to represent the predefined interpretable concepts.

Since the attention block is multi-headed, different heads naturally form the components of the attention bottleneck. Moreover, the components need to be divided between the heads, which would be convenient when the number of heads is a multiple of the total number of concepts to maintain a uniform concept per head ratio. For the feed-forward bottleneck, we define the components to be slices from its output, such that stacking the components results in the original output.

Interpretable concepts. For the real-world time series, we use two domain-agnostic interpretable concepts which can be used for forecasting, namely: (1) a simple, human-interpretable surrogate forecasting model, (2) the input timestamps recorded with the time series. Note that each model token (time step) should map to each concept to calculate the CKA score.

1. We use a simple autoregressive model (AR) as a surrogate model, which predicts the next future value as a linear combination of its past values. This model is transparent, and the attribution of each input feature to the output can be simply interpreted by its weight. This concept can also be regarded as a baseline for the forecasting performance. The model is fit to the same training data as the transformer (with its order being the length of the input). We use its activations to calculate the CKA score.
2. We use the hour-of-day feature from the timestamps $\boldsymbol{T}$ as interpretable time concept, denoted by $\boldsymbol{T}_{hourofday}$. This provides the bottleneck with a simplified notion of time.

Removal of residual connection. Any transformer layer contains residual connections around the attention and feed-forward blocks. To ensure that all information passes through the bottleneck, we remove the residual connection around the bottleneck, potentially at the cost of a loss in performance. Otherwise, any concept, including the interpretable concepts, can be passed through the residual connection and compromise the bottleneck.

In the scenario that the number of components is equal to the number of interpretable concepts ($c = 2$), the construction of the bottleneck limits learning domain-specific features from the data, other than the interpretable concepts. Therefore, we perform experiments where we allow an extra component in the bottleneck to not learn any predefined concept ($c = 3$). In other words, the extra component serves as a *side-channel* or *free component*, on which no CKA loss is calculated. The free component may partly restore the information lost by removing the residual connection, but with the advantage that we can monitor which information goes through it, and even visualize it (see Interpretability Analysis Section).

Implementation details.

In our experiments, we use transformer models with three encoder layers, of which the bottleneck layer is the second layer. Similar to the original Autoformer paper, we use one decoder layer, employ the Adam optimizer (Kingma and Ba 2017) with an initial learning rate of 10^{-4}, and use a batch size of 32. The training process is early stopped within 25 epochs. All experiments are repeated five times on different seeds, using hyperparameter $\alpha = 0.3$. Each model is trained on 1 Nvidia GeForce GTX 1080 Ti for approximately 30 minutes.

Experiments

We evaluate our framework on seven datasets, including synthetic and real-world data. The six real-world benchmarks consider the domains of energy, traffic, economics, weather, and disease (similar to Wu et al. 2021, see Table 1). These datasets are multivariate, and the task is to predict the future values of all variates. For example, the electricity dataset consists of hourly measurements of the electricity consumption of 321 customers from 2012 to 2014.

We apply our experiments to the Autoformer. First, we train a simple AR model on the same data, so that its outputs can be used to align the representations of the bottleneck. Then, we train the transformers with and without bottleneck, using different configurations for the bottleneck.

Synthetic Data

To show the general applicability of the bottleneck framework, we first train an Autoformer on a synthetic time series. In particular, we generate the dataset as the sum of different sines using the function f_{Total} with time t as follows:

$$f_{Total}(t) = f_1(t) + f_2(t) + f_3(t),$$

where:

$$f_1(t) = \sin(2\pi t),$$
$$f_2(t) = \frac{1}{2}\sin(4\pi t + \frac{\pi}{4}),$$
$$f_3(t) = \frac{1}{4}\sin(6\pi t + \frac{\pi}{2}) + \epsilon_t.$$

Note that all functions f_1, f_2 and f_3 follow a periodic structure, and f_3 contains random noise ϵ from a normal distribution with standard deviation of 0.2. See Figure 3 for a visualization of the functions.

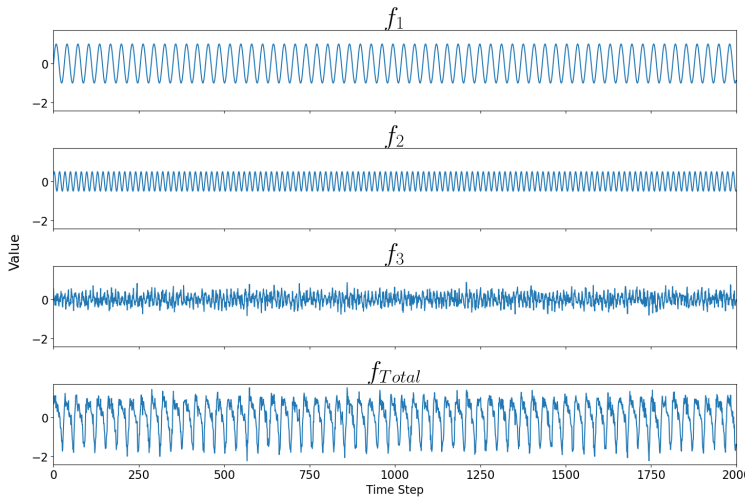
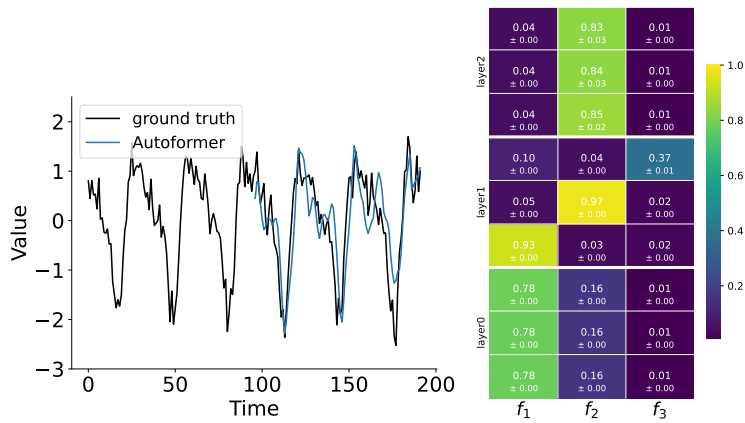


Figure 3. The synthetic time series dataset consists of three sines and a noise component.

Each concept in the bottleneck is defined as one of the underlying functions (i.e., f_1, f_2 or f_3), for which the ground-truth is known by construction. For hyperparameter $\alpha = 0.8$ (see Method Section), we find that the model is able to forecast well, while achieving very high similarity scores. That is, the model obtains a Mean Squared Error (MSE) of 0.36 ± 0.17 and Mean Absolute Error (MAE) of 0.46 ± 0.12 on 5 different seeds. See Figure 4 for a sample forecast on the test data and the CKA scores of the model's representations with the concept representations. The heads in the bottleneck `layer1` show high similarity for their respective concepts, e.g., a score of 0.93 for the head trained on f_1 (recall that CKA scores range from 0 for totally dissimilar to 1.0 for identical, although potentially scaled and rotated).

These results indicate that the model is successfully learning the intendend interpretable concepts (high CKA scores), while maintaining downstream performance. Interestingly, when using lower values for hyperparameter α, we find that the performance slightly decreases (Figure 5). This suggests that properly chosen concepts

improve the performance of the model, at least when the ground-truth underlying functions are known. Note that a low forecasting error cannot be expected for $\alpha = 1$, because in this edge case the loss function does not contain any term that represents the forecasting error.



(a) Forecast **(b)** CKA scores

Figure 4. The CBTL successfully captures the interpretable concepts, while maintaining high forecasting performance. The CKA scores are given for three heads per layer (vertically) across the three concept vectors (horizontally) regarding the attention CBTL Autoformer on synthetic data.

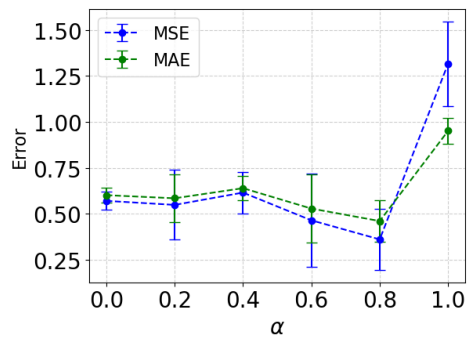


Figure 5. Increasing the weight of the CKA loss on the CBTL enhances model performance. The forecasting error on the synthetic dataset decreases for α, for an Autoformer with attention CBTL (across 5 seeds).

Real-world data

Table 1. Descriptions of the datasets, as given by Zhou et al. (2022) and shared online. 'Pred len' denotes the prediction length used in our experiments.

Dataset	Pred len	Description
Electricity	96	Hourly electricity consumption of 321 customers from 2012 to 2014.
Traffic	96	Hourly data from California Department of Transportation, which describes the road occupancy rates measured by different sensors on San Francisco Bay area freeways.
Weather	96	Recorded every 10 minutes for 2020 whole year, which contains 21 meteorological indicators, such as air temperature, humidity, etc.
Illness	24	Includes the weekly recorded influenza-like illness (ILI) patients data from Centers for Disease Control and Prevention of the United States between 2002 and 2021, which describes the ratio of patients seen with ILI and the total number of the patients.
Exchange rate	96	Daily exchange rates of eight different countries ranging from 1990 to 2016.
ETT	96	Data collected from electricity transformers, including load and oil temperature that are recorded every 15 minutes between July 2016 and July 2018.

Table 2 shows the performance of the Autoformer with our bottleneck on the benchmark datasets, compared to the AR surrogate model (i.e., the first interpretable concept) and Wu et al. (2021) (i.e., the original Autoformer model).

Table 2. Error scores of different Autoformer models. For both metrics, it holds that a lower score indicates a better performance, where the best results are **bold**, and the second-best are underlined.

	Att bottleneck		**FF bottleneck**		**No bottleneck**		**AR**		**Wu et al.**	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	0.231	0.338	0.207	0.320	0.280	0.368	0.497	0.522	**0.201**	**0.317**
Traffic	0.642	0.393	**0.393**	**0.377**	0.619	0.387	0.420	0.494	0.613	0.388
Weather	0.290	0.354	0.271	0.341	0.269	0.344	**0.006**	**0.062**	0.266	0.336
Illness	3.586	1.313	3.661	1.322	3.405	1.295	**1.027**	**0.820**	3.483	1.287
Exchange rate	0.195	0.323	0.155	0.290	0.152	0.283	**0.082**	**0.230**	0.197	0.323
ETT	0.177	0.282	0.174	0.280	0.155	0.265	**0.034**	**0.117**	0.255	0.339

We find that including a bottleneck (either **Att bottleneck** or **FF bottleneck**) outperforms **Wu et al.** for three datasets (traffic, exchange rate and ETT), and stays within 5% of the MSE and MAE for the other three datasets. Surprisingly, the surrogate AR model outperforms the other models for most datasets w.r.t. both MSE and MAE, even

though this model is very simple.* Visualizations of the forecasts from these models are shown in Figure 6.

We extended the analysis using different training set-ups, and obtained similar results. In particular, we repeated the experiments for different transformer architectures (vanilla tranformer and FEDformer), for different values of hyperparameter α and with excluding the free component.

Interpretability Analysis

To demonstrate the impact of the bottleneck on model interpretability, we first conduct a CKA analysis on the bottleneck layer with the corresponding interpretable concepts, and then visually demonstrate how each component contributes to the final forecast.

CKA Analysis To test the extent to which the bottleneck represents the interpretable concepts, we calculate the CKA scores of the model's representations with the concept representations. The scores of the feed-forward bottleneck on the electricity dataset are shown in Figure 7. Note that the bottom, middle and upper layer of `layer1` correspond to the AR, hour-of-day, and free component of the bottleneck, respectively.

The scores show that the representations in the bottleneck layer are much more similar to the intended concepts than the representations from the model without bottleneck: 0.80 for the AR model, and 0.99 for the hour-of-day feature, whereas the model without bottleneck does not show high similarity to the interpretable concepts. This indicates that the training framework can encourage the components to form representations that are perfectly similar to the interpretable concepts. Additionally, note that the CKA scores of other layers than the bottleneck layer are also higher in Fig. 7b, which indicates that these other model components also learn to represent the interpretable concepts. This does not affect the interpretability of the bottleneck layer itself (see Intervention Section below).

Component Visualizations Because the model components all read and write from the residual stream (Elhage et al. 2021), we can visualize the contributions to the final prediction of each component separately by applying the entire decoder to the component representations (Decoder Lens method, Langedijk et al. (2023)). This way, we obtain visualizations of the contributions of each component in the bottleneck, see Figure 8. We obtain the output from the full bottleneck by applying the decoder to the output of the bottleneck (after performing layer normalization). The output from each component individually is obtained by masking the other components with zero (close to the mean).

From Figure 8a and 8b we see that the different bottleneck components are similar to the concepts they were trained on. In particular, the first component shows a forecast with correct periodicity and few irregularities, similar to the actual forecast from the AR model. Likewise, the second component shows a periodicity to the actual hour-of-day

*Note that the phenomenon that simple models sometimes beat time series transformers (Zeng et al. 2022) has been observed before. There has been a vivid discussion about the relevance of these results, for instance here. These discussions are beyond the scope of our paper, which rather targets interpretability of time series transformers. For more information on the effect of AR as surrogate model, see Appendix.

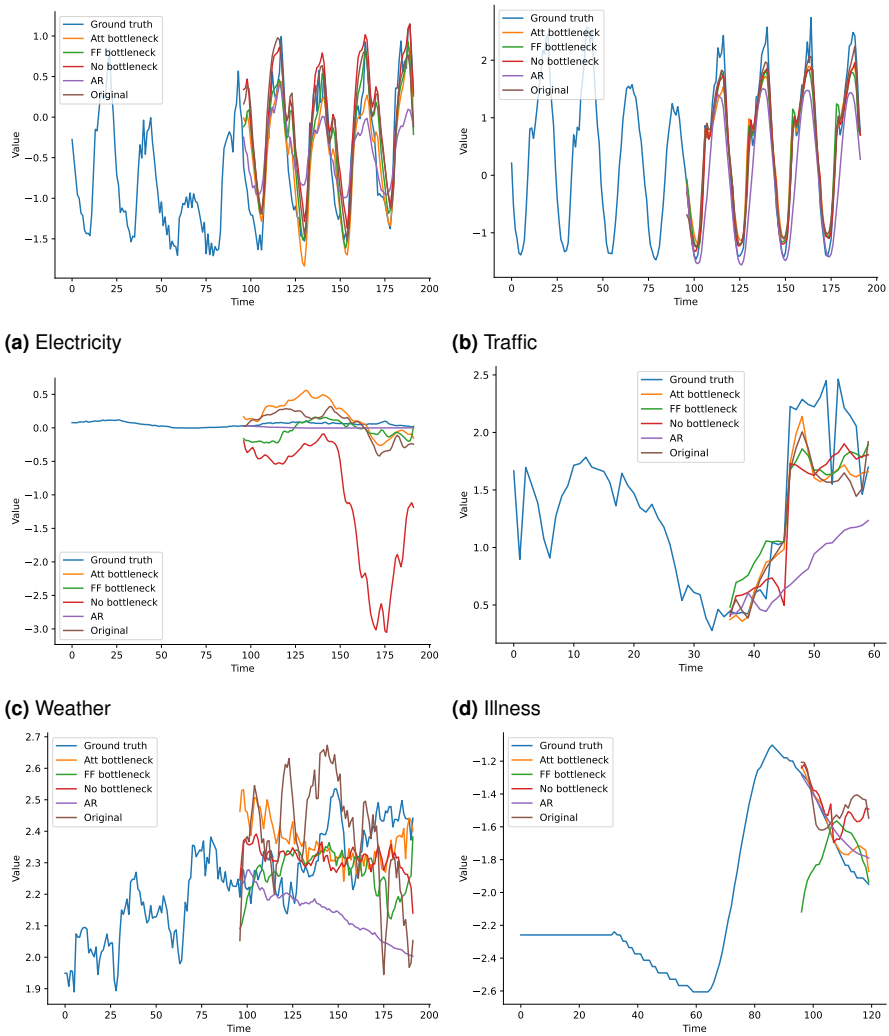
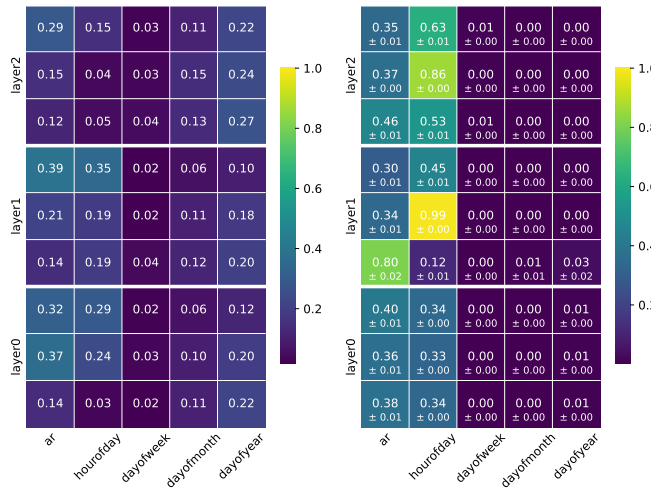


Figure 6. Models with or without CBTL obtain similar results. Forecasts of different models are given across datasets. The first part of the ground truth (shown in blue) is the input for the models.

feature. The third component is not trained to be similar to an interpretable concept, and seems to pick up on the high-frequency patterns in the data, e.g., the low, second peak in the forecast. This observation is further strengthened by Figure 8f, which shows that



(a) No bottleneck **(b)** Ours (FF bottleneck)

Figure 7. CBTL components learn representations that are similar to the interpretable concepts. CKA scores on different concepts for the encoder of the Autoformer without bottleneck and with FF bottleneck. The first component of `layer1` (lower row) of the CBTL is trained to be similar to AR, and the second component (middle row) to the hour-of-day concept. The scores are calculated on three batches of size 32 from the electricity test data. Recall that CKA is defined on a scale from 0 to 1, where 1 denotes perfect similarity.

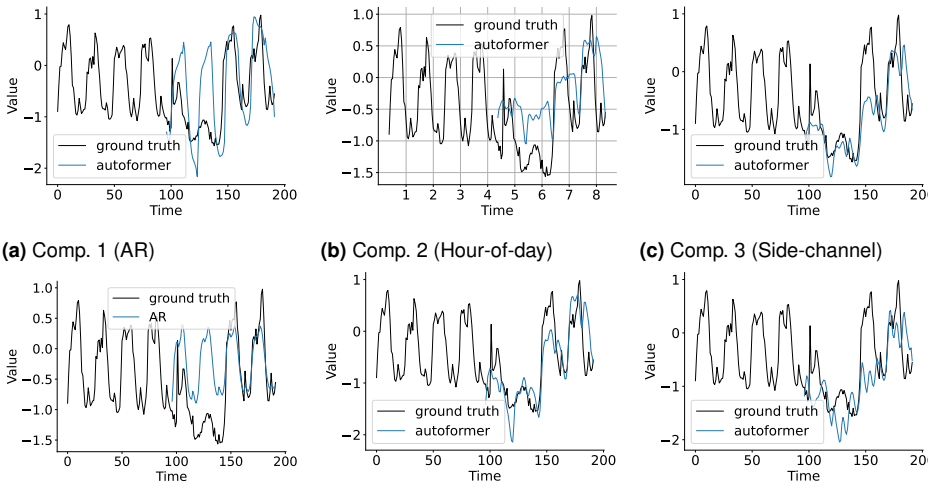
the final forecast consists of many high-frequency patterns when using only the third component from the bottleneck.

Intervention

The main benefit of interpreting trained models is gaining a deeper understanding and, possibly, more control of the model's behavior. This can be useful in the scenario of out-of-distribution data at inference time. If the data changes in features that can be interpreted in the model, it is feasible to intervene locally in these concepts to exclusively employ the model with data from its training distribution. Additionally, an intervention can be regarded as a causal interpretability test, where a successful intervention indicates a successful representation of the concept of interest.

To show such benefit of our framework, we perform activation patching (or causal tracing, Meng et al. 2023), where causal effects of hidden state activations are researched by evaluating the model on clean and corrupted inputs. We evaluate the trained model on data with shifted timestamps and compare it with performing an intervention on the shifted concept.

More specifically, we delay the input timestamps $\boldsymbol{T} \in \mathbb{R}^{I \times 4}$ with a fixed number of hours to obtain the shifted timestamps $\widetilde{\boldsymbol{T}}$, so that the learned patterns associated to the



(a) Comp. 1 (AR) **(b)** Comp. 2 (Hour-of-day) **(c)** Comp. 3 (Side-channel)

(d) AR **(e)** Full bottleneck **(f)** Final, using only comp. 3

Figure 8. The CBTL components individually predict their interpretable concept. Forecasts from individual bottleneck components by masking the other components with zero in 8a, 8b and 8c (FF bottleneck Autoformer on electricity data). The first half of the ground truth forms the input to the model. Note that the horizontal axes are the same across all figures, but Fig. 8b contains a grid of days instead of numbered hours. Fig. 8d shows the forecast made by the surrogate model AR; Fig. 8e shows the forecast of the entire layer (i.e., all components together), and 8f shows the forecast of the final layer when only the third component is used in the bottleneck layer. Note the difference between Figures 8c and 8f, where we decode from the bottleneck and the final layer, respectively.

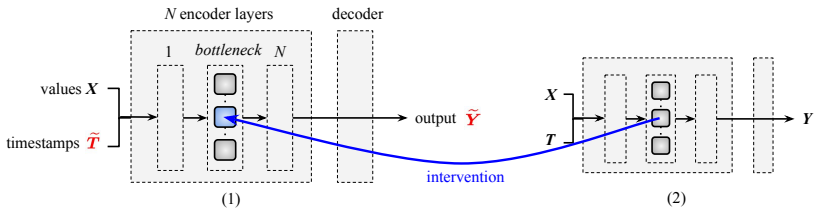
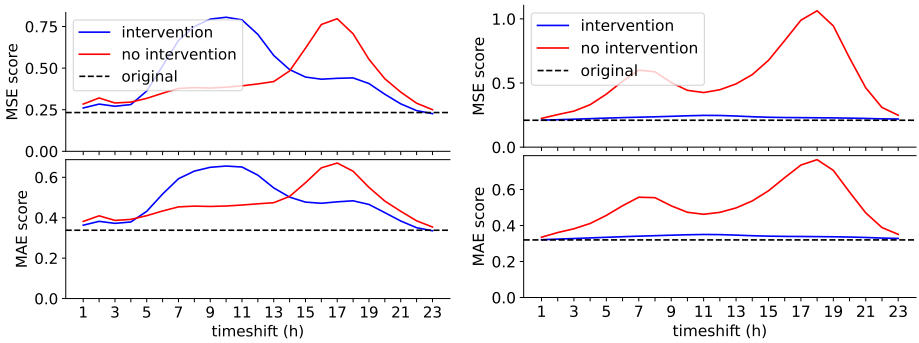


Figure 9. Activation patching allows for mechanistically testing the interpretations of the CBTL. In the intervention experiment we run the model on time-shifted input (1), but replace the activations of the hour-of-day component with those from a run (2) on unshifted timestamps $\boldsymbol{T}$.

hour-of-day feature are misleading. We run the model on both types of timestamps, and perform an intervention in the bottleneck by substituting the activations based on the shifted time with the activations based on the original, see Figure 9 for an overview.

We perform the intervention experiment with the electricity dataset, and perform shifts of up to and including 23 hours. We compare the performance of the intervention with out-of-the-box performance of the same model on the shifted dataset (see Figure 10).



(a) Att CBTL **(b)** FF CBTL

Figure 10. The hour-of-day concept is successfully learned by (only) the CBLT. The dashed line represents the performance on the data without timeshift. The intervention experiment shows the performance of the model on electricity data with shifted timestamps. The intervention experiment retrieves the original performance almost perfectly.

Remarkably, the intervention on the feed-forward CBTL achieves similar results to the original performance for all timeshifts. This indicates that the bottleneck can effectively learn to represent the hour-of-day concept in the dedicated bottleneck component. Most interestingly, the models only utilise this interpretable concept in the CBTL and *not* in other encoder layers (because the experiment only intervenes in the bottleneck).

Discussion and Conclusions

In this work, we propose a training framework based on Concept Bottleneck Models to enforce interpretability of time series transformers. We introduce a new loss function based on the similarity score CKA of the model's representations and interpretable concepts. We apply our framework to Autoformer using synthetic data and six benchmark datasets. Our results indicate that the overall performance remains unaffected, while the model's components become more interpretable. Additionally, it becomes possible to perform a local intervention when employing the model after a temporal data shift.

The main limitation of our concept bottleneck framework is that interpretable concepts have to be decided on before training, which might require domain knowledge. Representations for these concepts have to be available during training. However, domain-agnostic concepts such as the AR surrogate model and hour-of-day information are sufficient. Additionally, our framework increases computational complexity. This might be problematic if the size of the architecture increases.

An interesting direction for future research would be to optimize the number and type of interpretable concepts in the bottleneck, and extend the framework to other

modalities. We trained mostly using two domain-agnostic concepts (AR and hour-of-day), but including more concepts, possibly domain-specific, would be very interesting. For example, one could consider choosing speech and music concepts for audio time series. Additionally, the framework should also work for transformers in other modalities, e.g., language and vision, although these models are usually of larger size. We hope our work contributes to a deeper understanding of (time series) transformers and their behavior in different fields. In particular, recent progress in the field of mechanistic interpretability is based on the observation that the residual stream of the transformer encourages modular solutions, which enables localized concepts or specialized circuitry to perform a specific task. Instead of relying on post-hoc localization of these concepts, our paper presents a demonstration that we can encourage locality of concepts, without a significant loss in performance.

Regarding societal impact, this work enables transparent time series forecasting models, which enable explainable forecasts. However, in the case of malicious use, biases could be included in the models. Harm could be prevented by developing mechanistic interpretability techniques for bias detection in time series models.

References

Barbiero P, Ciravegna G, Giannini F, Zarlenga ME, Magister LC, Tonda A, Lio P, Precioso F, Jamnik M and Marra G (2023) Interpretable Neural-Symbolic Concept Reasoning. In: Proceedings of the 40th International Conference on Machine Learning. PMLR, pp. 1801–1825. URL https://proceedings.mlr.press/v202/barbiero23a.html.

Conmy A, Mavor-Parker AN, Lynch A, Heimersheim S and Garriga-Alonso A (????) Towards Automated Circuit Discovery for Mechanistic Interpretability .

Diamantini C, Mele A, Mircoli A, Potena D, Rossetti C and Storti E (2026) A Graph RAG Approach to Enhance Explainability in Dataset Discovery. Data Science and Engineering 11(1): 30–52. DOI:10.1007/s41019-025-00313-x. URL https://doi.org/10.1007/s41019-025-00313-x.

Elhage N, Nanda N, Olsson C, Henighan T, Joseph N, Mann B, Askell A, Bai Y, Chen A and Conerly T (2021) A mathematical framework for transformer circuits. Transformer Circuits Thread 1(1): 12.

Ferfoglia I, Saveri G, Nenzi L and Bortolussi L (2024) ECATS: Explainable-by-design concept-based anomaly detection for time series. URL http://arxiv.org/abs/2405.10608. ArXiv:2405.10608 [cs].

Forest F, Wei A and Fink O (2025) When, How Long and How Much? Interpretable Neural Networks for Time Series Regression by Learning to Mask and Aggregate. DOI:10.48550/arXiv.2512.03578. URL http://arxiv.org/abs/2512.03578. ArXiv:2512.03578 [cs.LG].

Helff L, Härle R, Stammer W, Friedrich F, Brack M, Wüst A, Shindo H, Schramowski P and Kersting K (2025) ActivationReasoning: Logical Reasoning in Latent Activation Spaces. DOI:10.48550/arXiv.2510.18184. URL http://arxiv.org/abs/2510.18184. ArXiv:2510.18184 [cs.LG] version: 1.

Kingma DP and Ba J (2017) Adam: A Method for Stochastic Optimization. DOI:10.48550/arXiv.1412.6980. URL http://arxiv.org/abs/1412.6980. ArXiv:1412.6980 [cs].

Koh PW, Nguyen T, Tang YS, Mussmann S, Pierson E, Kim B and Liang P (2020) Concept Bottleneck Models. In: Proceedings of the 37th International Conference on Machine Learning. PMLR, pp. 5338–5348. URL https://proceedings.mlr.press/v119/koh20a.html.

Kornblith S, Norouzi M, Lee H and Hinton G (2019) Similarity of Neural Network Representations Revisited. URL http://arxiv.org/abs/1905.00414. ArXiv:1905.00414 [cs, q-bio, stat].

Langedijk A, Mohebbi H, Sarti G, Zuidema W and Jumelet J (2023) DecoderLens: Layerwise Interpretation of Encoder-Decoder Transformers. URL http://arxiv.org/abs/2310.03686. ArXiv:2310.03686 [cs].

Ma H, Gao J and Tran MN (2025) Signals, Concepts, and Laws: Toward Universal, Explainable Time-Series Forecasting. DOI:10.48550/arXiv.2508.01407. URL http://arxiv.org/abs/2508.01407. ArXiv:2508.01407 [cs.LG].

Mahinpei A, Clark J, Lage I, Doshi-Velez F and Pan W (2021) Promises and Pitfalls of Black-Box Concept Learning Models. URL http://arxiv.org/abs/2106.13314.

ArXiv:2106.13314 [cs].

Manhaeve R, Dumančić S, Kimmig A, Demeester T and Raedt LD (2018) DeepProbLog: Neural Probabilistic Logic Programming. DOI:10.48550/arXiv.1805.10872. URL http://arxiv.org/abs/1805.10872. ArXiv:1805.10872 [cs.AI].

Mao J, Tenenbaum JB and Wu J (2025) Neuro-Symbolic Concepts. DOI:10.48550/arXiv.2505.06191. URL http://arxiv.org/abs/2505.06191. ArXiv:2505.06191 [cs.AI].

Meng K, Bau D, Andonian A and Belinkov Y (2023) Locating and Editing Factual Associations in GPT. DOI:10.48550/arXiv.2202.05262. URL http://arxiv.org/abs/2202.05262. ArXiv:2202.05262 [cs].

Parisi GI, Kemker R, Part JL, Kanan C and Wermter S (2019) Continual lifelong learning with neural networks: A review. Neural Networks 113: 54–71. DOI:10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

Poeta E, Ciravegna G, Pastor E, Cerquitelli T and Baralis E (2023) Concept-based Explainable Artificial Intelligence: A Survey. DOI:10.48550/arXiv.2312.12936. URL http://arxiv.org/abs/2312.12936. ArXiv:2312.12936 [cs].

Rai D, Zhou Y, Feng S, Saparov A and Yao Z (2024) A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models. DOI:10.48550/arXiv.2407.02646. URL http://arxiv.org/abs/2407.02646. ArXiv:2407.02646 [cs].

Shu D, Wu X, Zhao H, Rai D, Yao Z, Liu N and Du M (2025) A Survey on Sparse Autoencoders: Interpreting the Internal Mechanisms of Large Language Models. In: Christodoulopoulos C, Chakraborty T, Rose C and Peng V (eds.) Findings of the Association for Computational Linguistics: EMNLP 2025. Suzhou, China: Association for Computational Linguistics. ISBN 979-8-89176-335-7, pp. 1690–1712. DOI:10.18653/v1/2025.findings-emnlp.89. URL https://aclanthology.org/2025.findings-emnlp.89/.

Sprang Av, Acar E and Zuidema W (2025) Interpretability for Time Series Transformers using A Concept Bottleneck Framework. DOI:10.48550/arXiv.2410.06070. URL http://arxiv.org/abs/2410.06070. ArXiv:2410.06070 [cs.LG].

Tan Z, Cheng L, Wang S, Yuan B, Li J and Liu H (2024) Interpreting Pretrained Language Models via Concept Bottlenecks. In: Yang DN, Xie X, Tseng VS, Pei J, Huang JW and Lin JCW (eds.) Advances in Knowledge Discovery and Data Mining. Singapore: Springer Nature. ISBN 978-981-97-2259-4, pp. 56–74. DOI:10.1007/978-981-97-2259-4_5.

Tian S, Li W, Ning X, Ran H, Qin H and Tiwari P (2023) Continuous transfer of neural network representational similarity for incremental learning. Neurocomputing 545: 126300. DOI:10.1016/j.neucom.2023.126300. URL https://www.sciencedirect.com/science/article/pii/S092523122300423X.

Van Krieken E, Acar E and Van Harmelen F (2022) Analyzing Differentiable Fuzzy Logic Operators. Artificial Intelligence 302: 103602. DOI:10.1016/j.artint.2021.103602. URL https://linkinghub.elsevier.com/retrieve/pii/S0004370221001533.

Wu H, Xu J, Wang J and Long M (2021) Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In: Advances in Neural Information Processing Systems, volume 34. Curran Associates, Inc., pp. 22419–22430. URL https://proceedings.neurips.cc/paper_files/paper/

2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.

Zarlenga ME, Barbiero P, Ciravegna G, Marra G, Giannini F, Diligenti M, Shams Z, Precioso F, Melacci S, Weller A, Lio P and Jamnik M (2022) Concept Embedding Models: Beyond the Accuracy-Explainability Trade-Off. URL http://arxiv.org/abs/2209.09056. ArXiv:2209.09056 [cs].

Zeng A, Chen M, Zhang L and Xu Q (2022) Are Transformers Effective for Time Series Forecasting? URL http://arxiv.org/abs/2205.13504. ArXiv:2205.13504 [cs].

Zheng Z, Wang Y, Huang Y, Song S, Yang M, Tang B, Xiong F and Li Z (2025) Attention heads of large language models. Patterns 6(2). DOI:10.1016/j.patter.2025.101176. URL https://www.cell.com/patterns/abstract/S2666-3899(25)00024-8.

Zhou T, Ma Z, Wen Q, Wang X, Sun L and Jin R (2022) FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. URL http://arxiv.org/abs/2201.12740. ArXiv:2201.12740 [cs, stat].

Appendix

Formalization of Concept Bottleneck Framework

Any time series transformer obtains two types of input: (1) data values $\boldsymbol{X} \in \mathbb{R}^{I \times d}$, and (2) timestamps $\boldsymbol{T} \in \mathbb{R}^{I \times 4}$. The transformer consists of an encoder and a decoder, which are both constructed from one or multiple layers. Any encoder layer contains two sub-layers: a multi-head attention mechanism (Att) and a fully connected neural network (FF). Every sub-layer contains a residual connection around it. More specifically, the output $\boldsymbol{X}^\ell$ of any encoder layer ℓ is:

$$\begin{aligned} \boldsymbol{X}^\ell &= \text{Encoder}(\boldsymbol{X}^{\ell-1}) \\ &= \text{LayerNorm}(\text{FF}(\boldsymbol{S}^\ell) + \boldsymbol{S}^\ell), \\ \boldsymbol{S}^\ell &= \text{LayerNorm}(\text{Att}(\boldsymbol{X}^{\ell-1}) + \boldsymbol{X}^{\ell-1}), \end{aligned}$$

where

$$\begin{aligned} \text{FF}(\mathbf{x}) &= \max(0,\, \mathbf{x}\boldsymbol{W}_1 + \mathbf{b}_1)\, \boldsymbol{W}_2 + \mathbf{b}_2, \\ \text{Att}(\mathbf{x}) &= \boldsymbol{W}_0 \cdot \text{Concat}\left(\text{h}_1(\mathbf{x}),\, \ldots,\, \text{h}_h(\mathbf{x})\right). \end{aligned}$$

For future reference, we denote the output of the feed-forward module as follows: $\text{FF}(\boldsymbol{S}^\ell) = \boldsymbol{Z}^\ell \in \mathbb{R}^{d_1 \times d_2}$. We omit the definition of the decoder, because our bottleneck framework does not include it. Note that the exact implementation of each (sub-)layer depends on the type of transformer.

Concept Bottleneck Transformer Layer

We assign one encoder layer to be the bottleneck and construct it such that it contains c latent representations or *components*, i.e., $(\boldsymbol{H}_i)_{i=1}^c$. Depending on the bottleneck type τ, these latent representations are either taken from the attention mechanism or the feed-forward module. More specifically:

$$\boldsymbol{H}_i = \begin{cases} \text{h}_i(\mathbf{x}) & \text{if bottleneck type } \tau = \text{Att}, \\ \boldsymbol{Z}_i & \text{if bottleneck type } \tau = \text{FF}. \end{cases}$$

Since the attention block is multi-headed, different heads naturally form the components of the attention bottleneck. For the feed-forward bottleneck, we define the components to be slices (in d_1) from its output $\boldsymbol{Z}$, such that stacking the components results in the original output.

Note that the residual connection around the corresponding bottleneck component is removed, and that each component $\boldsymbol{H}_i$ should represent a predefined interpretable concept.

Intervention

In the intervention experiment, we shift the time stamps $\boldsymbol{T}$ to obtain $\widetilde{\boldsymbol{T}}$. The key aspect of the experiment is to run the transformer on the shifted time stamps $\widetilde{\boldsymbol{T}}$, and replace the

input representations $\widetilde{\boldsymbol{X}}^{b-1}$ of the bottleneck layer b with $\boldsymbol{X}^{b-1}$ (based on $\boldsymbol{T}$), but only in the component that represents the time concept.

More specifically, if type $\tau = \text{Att}$, we intervene on the attention block in the bottleneck as follows:

$$\text{Att}(\mathbf{x}, \widetilde{\mathbf{x}}) = \boldsymbol{W}_0 \cdot \text{Concat}\left(\text{h}_1(\widetilde{\mathbf{x}}),\, \text{h}_2(\mathbf{x}),\, \text{h}_3(\widetilde{\mathbf{x}})\right),$$

and, if type $\tau = \text{FF}$, as follows:

$$\text{FF}(\mathbf{x}, \widetilde{\mathbf{x}}) = \text{Stack}(\widetilde{\boldsymbol{Z}}_1, \boldsymbol{Z}_2, \widetilde{\boldsymbol{Z}}_3).$$

In both functions we make use of the fact that the time concept is represented in the second component, and there are three components in total. This intervention can be done in the bottleneck only, because, by construction, the location of its concept representations is known.

Detailed results

Table 3. Performance of different models in Mean Squared Error (MSE) and Mean Absolute Error (MAE). The bottlenecks **do** contain a free component ($c = 3$), and use AR as surrogate model. The model with no bottleneck is an original Autoformer of similar size. For all datasets, the shortest prediction lengths from Wu et al. (2021) are used, see Table 1. The standard deviation is determined using five different seeds.

Free component	**Att bottleneck**		**FF bottleneck**		**No bottleneck**		**AR**		**Wu et al.**	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	0.231 ± 0.009	0.338 ± 0.005	0.207 ± 0.005	0.320 ± 0.005	0.280 ± 0.165	0.368 ± 0.111	0.497	0.522	**0.201** ± 0.003	**0.317** ± 0.004
Traffic	0.642 ± 0.022	0.393 ± 0.013	**0.393** ± 0.013	**0.377** ± 0.006	0.619 ± 0.015	0.387 ± 0.005	0.420	0.494	0.613 ± 0.028	0.388 ± 0.012
Weather	0.290 ± 0.027	0.354 ± 0.020	0.271 ± 0.016	0.341 ± 0.011	0.269 ± 0.000	0.344 ± 0.000	**0.006**	**0.062**	0.266 ± 0.007	0.336 ± 0.006
Illness	3.586 ± 0.241	1.313 ± 0.040	3.661 ± 0.237	1.322 ± 0.050	3.405 ± 0.208	1.295 ± 0.044	**1.027**	**0.820**	3.483 ± 0.107	1.287 ± 0.018
Exchange rate	0.195 ± 0.029	0.323 ± 0.025	0.155 ± 0.010	0.290 ± 0.013	0.152 ± 0.003	0.283 ± 0.003	**0.082**	**0.230**	0.197 ± 0.019	0.323 ± 0.012
ETT	0.177 ± 0.003	0.282 ± 0.004	0.174 ± 0.006	0.280 ± 0.005	0.155 ± 0.004	0.265 ± 0.002	**0.034**	**0.117**	0.255 ± 0.020	0.339 ± 0.020

Table 4. Performance on different datasets, where the bottlenecks **do not** contain a free component (**$c = 2$**). AR is used as surrogate model in the bottlenecks. The model with no bottleneck is an original Autoformer of similar size. For all datasets, the shortest prediction lengths from Wu et al. (2021) are used, see Table 1. The standard deviation is determined using five different seeds.

No free component	**Att bottleneck**		**FF bottleneck**		**No bottleneck**		**AR**		**Wu et al.**	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	0.224 ± 0.006	0.332 ± 0.003	0.206 ± 0.009	0.321 ± 0.009	0.202 ± 0.006	0.318 ± 0.007	0.497	0.522	**0.201** ± 0.003	**0.317** ± 0.004
Traffic	0.629 ± 0.023	0.394 ± 0.015	0.627 ± 0.031	0.392 ± 0.025	0.613 ± 0.018	**0.378** ± 0.007	**0.420**	0.494	0.613 ± 0.028	0.388 ± 0.012
Weather	0.281 ± 0.025	0.348 ± 0.018	0.260 ± 0.015	0.333 ± 0.013	0.257 ± 0.004	0.332 ± 0.005	**0.006**	**0.062**	0.266 ± 0.007	0.336 ± 0.006
Illness	3.966 ± 0.296	1.401 ± 0.073	3.721 ± 0.268	1.351 ± 0.053	3.585 ± 0.331	1.333 ± 0.070	**1.027**	**0.820**	3.483 ± 0.107	1.287 ± 0.018
Exchange rate	0.208 ± 0.026	0.333 ± 0.022	0.158 ± 0.009	0.293 ± 0.009	0.152 ± 0.006	0.284 ± 0.007	**0.082**	**0.230**	0.197 ± 0.019	0.323 ± 0.012
ETT	0.178 ± 0.011	0.283 ± 0.007	0.174 ± 0.01	0.283 ± 0.009	0.165 ± 0.004	0.274 ± 0.004	**0.034**	**0.117**	0.255 ± 0.020	0.339 ± 0.020