Effectively Leveraging Exogenous Information across Neural Forecasters

Andres Potapczynski
Work done at Amazon
andpotap@nyu.edu

Kin G. Olivares
Amazon
kigutie@amazon.com

Malcolm Wolff
Amazon
wolfmalc@amazon.com

Andrew Gordon Wilson
Amazon & New York University
rewgw@cims.nyu.edu

Dmitry Efimov
Amazon
defimov@amazon.com

Vincent Quenneville-Bélair
Amazon
quennv@amazon.com

Abstract

Research on neural networks for time series has mostly focused on developing models that learn patterns about the target signal without the use of additional auxiliary or exogenous information. In applications such as selling products on a marketplace, the target signal is influenced by these variables, and leveraging exogenous variables is important. In particular, knowing that a product would go into promotion would mostly likely generate a spike in its demand; and ignoring this information would degrade the forecasting ability of the models. In such applications, the exogenous information comes as a mixture of categorical and real variables on different scales. In this paper we develop a decoder method that leverages the time structure of exogenous information through structured state-space model layers and learns relationships between the variables through MLPs. We show that this decoder method can be applied to a wide variety of models such as `NBEATS`, `NHITS`, `PatchTST`, and `S4`, yielding notable performance improvements across a different datasets.

1 Introduction

Most of the current research on neural forecasters has centered on developing models that rely on the transformer architecture. Some of the research focuses on using large language models for zero-shot prediction as in Gruver et al. [8], Dooley et al. [7], `Chronos` [1], `TimeLLM` [10], and `LagLlama` [18]. Other methods aim to improve the performance of transformers for time series predictions, such as `PatchTST` [14], `iTransformer` [13], `MQT` [3], and `DFR` [17]. While another research thrust focuses on using transformers for long-horizon prediction with methods like `TiDE` [6], `Autoformer` [21], `FEDformer` [24], `Triformer` [5], `Pyraformer` [12], and `Informer` [23].

Unfortunately, all of the aforementioned models focus on learning specific patterns that occur in a target signal without the use of additional information (exogenous variables). Not using exogenous variables severely hampers the application of these recently developed models to important real-life applications. For example, the price of electricity is directly affected by factors such as the weather and system load, or the demand of a product is likely to spike if it is in promotion. Regardless of how accurate a neural forecaster might be at capturing time patterns in a target signal, it will fail to predict abrupt changes if it is not using the exogenous information that might explain the change.

Furthermore, the reliance of the aforementioned models on the transformer architecture makes the addition of exogenous information a non-trivial task. The models that rely on large language models tokenize the target signal, and this tokenization process is not directly applicable to exogenous categorical variables such as brand or geographic location. Moreover, most of the models use the attention mechanism in an idiosyncratic manner, and it is therefore unclear if each case requires an *ad-hoc* method to incorporate exogenous information.

In this work, we develop a universal method, which we call `RevPred` (Revisiting Predictions), to effectively incorporate exogenous information into different neural forecasters. The main idea is that once a neural forecaster makes a prediction, we will revisit this prediction with a decoder model that depends on the exogenous information. This decoder learns linear embeddings for categorical variables and it is able to exploit the time structure of the other exogenous variables through a mixture of state-space model (SSM) and feed-forward layers, inspired by Gu et al. [9]. It is worth noting that our work is not the first one to incorporate exogenous information to neural forecasters. However, it is the first to propose a universal method to do so, as most of the existing methods such as `TFT` [11], `TSMixer` [4], `TimeMixer` [20], `BitCN` [19] `NBEASTx` [15], and `NHITS` [2] rely on approaches that are specific to each model and hence not compatible with others.

2 Problem Formulation and Data Description

For a target signal $\mathbf{y}_{1:T} = (y_1, \dots, y_T)$ and some exogenous information $\mathbf{x}^{(s)}$, $\mathbf{x}^{(h)}_{1:T}$ and $\mathbf{x}^{(f)}_{1:T+H}$ we want to produce a forecast $\hat{\mathbf{y}}_{T+1:T+H} = \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{y}_{1:T}, \mathbf{x}^{(s)}, \mathbf{x}^{(h)}_{1:T}, \mathbf{x}^{(f)}_{1:T+H})$, where $\mathcal{F}_{\boldsymbol{\theta}}(\cdot)$ is a neural network and $\boldsymbol{\theta}$ its trainable parameters. Here $\mathbf{x}^{(s)}$ represents static exogenous information that is constant through time, such as the category or brand of a product. Additionally, $\mathbf{x}^{(h)}_{1:T}$ corresponds to exogenous information that is available in the same span of history $1, \dots, T$ but that it is unknown for $T+1, \dots, H$, such as current and historic weather measurements. Finally, $\mathbf{x}^{(f)}_{1:T+H}$ represents exogenous information that is known for both the historic and future horizon times, such as the promotional activity of a product. Most of the models in Section 1 produce forecasts only using $\mathbf{y}_{1:T}$ as $\hat{\mathbf{y}}_{T+1:T+H} = \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{y}_{1:T})$ which severely hampers the predictive capacity of $\mathcal{F}_{\boldsymbol{\theta}}(\cdot)$ as seen in Section 4.

Our goal is to exploit the exogenous information for forecasting. Unfortunately, the vast majority of time series benchmarks do not contain exogenous information complicating off-the-shelf comparisons. Nonetheless, there are two categories of publicly available data that we found suitable for our investigation: electric price forecasting and grocery demand forecasting.

For predicting electric prices [15] we have 5 datasets: the Nord pool (NP) market which is one of the largest European power markets containing hourly measurements from 2023-01-01 to 2018-12-24. The NP dataset comes with exogenous variables measuring the grid load and wind power. We then have the zonal prices for the COMED area of Pennsylvania, New Jersey and Maryland (PJM) containing hourly measurements from 2023-01-01 to 2018-12-14. The PJM dataset comes with exogenous measurements of the system load and zonal load. Next, we have the French electricity market (FR) containing hourly measurements from 2011-01-09 to 2016-12-31. The FR dataset contains exogenous measurements of system load and power generation. Then, we have the Belgian electricity market (BE) containing hourly measurements from 2011-01-09 to 2016-12-31. The BE dataset contains exogenous measurements of system load and power generation. Finally, we have the German electricity market (DE) containing hourly measurements from 2012-01-09 to 2017-12-31. The DE dataset contains exogenous measurements of zonal load and both solar and wind generation measurements. The dataset can be found here `https://zenodo.org/records/4624805`.

For grocery demand forecasting, we have the data from the ecuadorian Corporación Favorita (Favorita). This datasets consist of weekly unit demand across several products with indicators of promotional activity that we use as exogenous information. The dataset can be found here `https://www.kaggle.com/c/favorita-grocery-sales-forecasting`.

3 `RevPred`

We propose the `RevPred` decoder to facilitate the use of exogenous information for any neural forecaster. Intuitively, the decoder starts with the predictions of a given neural forecaster and then revisits this prediction based on the exogenous features. It does so by concatenating all the features (both target and exogenous) and then passing that through blocks of state-space model (SSM) layers and MLP mixer layers. The SSM layers leverage the time structure of the input and the MLP layers generate useful representations. Concretely, RevPred is constructed in the following manner. Given a neural forecaster model $\mathcal{F}_{\boldsymbol{\theta}}(\cdot)$ that only uses $\mathbf{y}_{1:T}$ as input, we then have that our RevPred decoder $\mathcal{D}_{\boldsymbol{\phi}}$ would use the predictions of $\mathcal{F}_{\boldsymbol{\theta}}$ and the rest of the exogenous information to create a refined prediction as

$$\hat{\mathbf{y}}_{T+1:T+H} = \mathcal{D}_{\boldsymbol{\phi}}(\mathcal{F}_{\boldsymbol{\theta}}(\mathbf{y}_{1:T}), \mathbf{y}_{1:T}, \mathbf{x}^{(s)}, \mathbf{x}^{(h)}_{1:T}, \mathbf{x}^{(f)}_{1:T+H}).$$

The decoder operates as follows. First, it embeds any categorical variable into a vector through an index to vector look-up. Thus, we can now assume that all the exogenous variables are embedded as real vectors, $\mathbf{x}^{(s)} \in \mathbb{R}^{D_s}$, $\mathbf{x}^{(h)}_{1:T} \in \mathbb{R}^{T \times D_h}$ and $\mathbf{x}^{(f)}_{1:T+H} \in \mathbb{R}^{(T+H) \times D_f}$. Second, it constructs three variables $\tilde{\mathbf{y}}_{1:T+H}$, $\tilde{\mathbf{x}}^{(s)}_{1:T+H}$, and $\tilde{\mathbf{x}}^{(h)}_{1:T+H}$ as follows

$$\begin{aligned}
\tilde{\mathbf{y}}_{1:T} &= \mathbf{y}_{1:T} \quad \text{and} \quad \tilde{\mathbf{y}}_{T+1:T+H} = \mathcal{F}_{\boldsymbol{\theta}}\left(\mathbf{y}_{1:T}\right), \\
\tilde{\mathbf{x}}^{(s)}_{t} &= \mathbf{x}^{(s)} \quad \text{for all } t = 1, \ldots, T+H \\
\tilde{\mathbf{x}}^{(h)}_{1:T} &= \mathbf{x}^{(h)}_{1:T} \quad \text{and} \quad \tilde{\mathbf{x}}^{(h)}_{T+1:T+H} = \mathbf{W}\mathbf{x}^{(h)}_{1:T}
\end{aligned}$$

where $\mathbf{W} : \mathbb{R}^{T} \to \mathbb{R}^{H}$ is a linear projection applied to the time axis. Once the three variables are constructed, the decoder then concatenates all the variables to create $\mathbf{z}_{1:T+H} = (\tilde{\mathbf{y}}_{1:T+H}, \tilde{\mathbf{x}}^{(s)}_{1:T+H}, \tilde{\mathbf{x}}^{(h)}_{1:T+H}, \mathbf{x}^{(f)}_{1:T+H}) \in \mathbb{R}^{(T+H) \times D}$ where $D = 1 + D_s + D_h + D_f$. To produce a forecast, the decoder applies, in blocks, a structured state-space model layer (SSM) [9] on the time axis and then a two-layered MLP on the dimensionality axis. In other words, $\mathbf{z}_{1:T+H} \leftarrow \texttt{MLP}(\texttt{SSM}(\mathbf{z}_{1:T+H}))$ repeatedly. A visual representation of this process can be found on Appendix A.1. Finally, the decoder makes a prediction by applying two linear projections, one on the time axis $\mathbf{W}^{\text{time}} : \mathbb{R}^{T+H} \to \mathbb{R}^{H}$ and one on the feature axis $\mathbf{W}^{\text{feat}} : \mathbb{R}^{D} \to \mathbb{R}$ such that

$$\hat{\mathbf{y}}_{T+1:T+H} = \mathbf{W}^{\text{time}}\, \mathbf{W}^{\text{feat}} \mathbf{z}_{1:T+H}.$$

We chose the combination of a SSM and MLP layers both to exploit the time structure of the input and to be parameter efficient. Similar to multi-head attention (MHA), SSM layers are sequence to sequence models that aim to find what previous parts of a sequence are important to predict the next parts of it. In contrast to MHA, SSMs are not permutation invariant but rather place larger emphasis on the most recent times when making a prediction. As argued in Zeng et al. [22], using MHA has many drawbacks for time series forecasting which SSMs do not posses. In terms of parameter efficiently, a simple approach that applies an MLP to the flattened and concatenated input would incur in $\mathcal{O}((T+H)^2 D^2)$ memory and compute whereas our decoder uses $\mathcal{O}\left((T+H)\log(T+H)D\right)$ for the SSM layer and $\mathcal{O}(D^2(T+H))$ for the MLP layer which, in total, is a substantial reduction in number of parameters.

We then train both neural network models $\mathcal{F}_{\boldsymbol{\theta}}$ and $\mathcal{D}_{\boldsymbol{\phi}}$ by minimizing the MAE loss

$$L\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) = \frac{1}{H} \sum_{t=T+1}^{T+H} \left| y_t - \hat{y}_t(\boldsymbol{\theta}, \boldsymbol{\phi}) \right|.$$

4 Results

Electric Price Forecasting In Table 1 we show the performance of adding the RevPrev decoder to multiple popular and performant neural forecasters. Namely, NBEATS [16, 15], NHITS [2], PatchTST [14] and S4 [9]. The first two models are based on MLP architectures and do have a variant that incorporates exogenous information (we compare against that as well). PatchTST is a transformer based model and S4 is an state-space based model. For this task we split the train and test sets on a 80% and 20% respectively.

Grocery Sales In this task we want to predict the weekly demand volume of roughly 3K different products from 2014 to 2016. In contrast to the previous task, we do not have high frequency hourly observations but rather weekly. However, the main challenge is that we now the model has to fit the different patterns present in the myriad of products. In Figure 1 we show that also RevPrev consistently improves the performance across various neural forecasters although in a less dramatical manner as seen for the electric price forecasting task. For this task we as well the train and test sets on a 80% and 20% respectively.

	NBEATS			NHITS			PatchTST		S4	
	-	Or	RP	-	Or	RP	-	RP	-	RP
BE	5.0627	2.3038	2.0129	5.2694	3.2859	2.1017	4.9244	2.4149	4.4321	2.4781
DE	9.1409	5.1628	2.2588	11.2157	2.2127	2.2133	11.4733	1.8857	12.7094	2.0194
FR	6.3306	2.7559	2.4095	6.8629	2.2568	2.6892	6.4160	3.0004	5.7751	2.8840
NP	0.8485	0.7971	0.3823	0.8090	0.9209	0.2763	1.0315	0.7320	0.5603	0.8314
PJM	1.0921	0.7844	0.4410	1.1839	0.5832	0.4858	1.1401	0.4588	0.9140	0.4263

Table 1: `RevPrev` **significantly reduces the test MAE across several neural forecasters**. We report unnormalized test MAE (lower is better) across 5 electric price forecasting datasets for 4 different neural forecasters. The symbol - refers to the model variant that does not use exogenous information. `Or` refers to the original variant of the model that uses exogenous information [15, 2] and `RP` stands for the model variant that uses the `RevPrev` decoder.

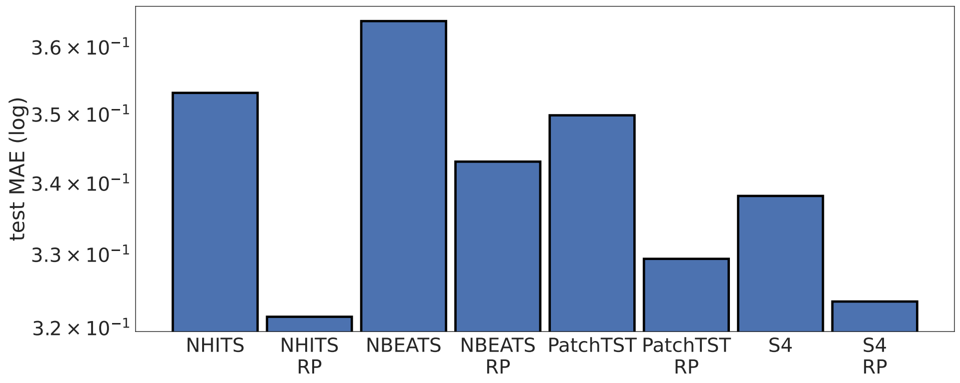

Figure 1: `RevPrev` **also improves performance of various neural forecasters for Favorita**. We report test MAE (lower is better) for the task of forecasting the national weekly demand. The legend `RP` implies that the model is using the `RevPrev` decoder. For `NHITS` and `NBEATS` we use the model variant that allows for exogenous variables.

5 Conclusion

A pervasive and major shortcoming of the current research in large neural network models for time series is to largely ignore the use of exogenous information that might influence severely the behavior of the time series. As seen in Section 4, ignoring exogenous information hinders the predictive ability of a model, regardless of its complexity. Yet, it is unclear how to best incorporate the exogenous information in these large neural network models. To circumvent this limitation we propose our `RevPred` decoder which can be easily applied to any neural forecaster and which improves empirical performance.

Although not evidenced by the experiments, `RevPred`'s main strength of being easily applied as a decoder at the end of the forecasting process could also be an important limitation. It seems plausible that incorporating the exogenous information earlier in the forecasting process might improve performance but that would also require developing an *ad-hoc* process that is not widely applicable. To strengthen `RevPred`'s appeal, it would be worth applying this decoder to more neural forecasters and also investigating other parameter efficient decoder structures beyond `SSM`s.

References

[1] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the Language of Time Series. *arXiv:2403.07815*, 2024.

[2] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2023.

[3] Kevin C. Chen, Lee Dicker, Carson Eisenach, and Dhruv Madeka. MQTransformer: Multi-Horizon Forecasts with Context Dependent Attention and Optimal Bregman Volatility. In Maria Florina Balcan and Marina Meila, editors, *In: Proceedings of 8th SIGKDD International Workshop on Mining and Learning From Time Series - Deep Forecasting: Models, Interpretability, and Applications (KDD' 22)*. ACM. Association for Computing Machinery, 8 2022. URL `https://kdd-milets.github.io/milets2022/papers/MILETS_2022_paper_9623.pdf`.

[4] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. TSMixer: An All-MLP Architecture for Time Series Forecasting. *Transactions on Machine Learning Research*, 2023.

[5] Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting. *arXiv:2204.13767*, 2022.

[6] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term Forecasting with TiDE: Time-series Dense Encoder. *arXiv:2304.08424*, 2023.

[7] Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. ForecastPFN: Synthetically-Trained Zero-Shot Forecasting. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[8] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large Language Models Are Zero-Shot Time Series Forecasters. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[9] Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces. *International Conference on Learning Representations (ICLR)*, 2022.

[10] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. *International Conference on Learning Representations (ICLR)*, 2024.

[11] Bryan Lim, Sercan ö. Arik, Nicolas Loeff, and Tomas Pfister. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 2021.

[12] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, and Schahram Dustdar. Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. *International Conference on Machine Learning (ICML)*, 2022.

[13] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. *International Conference on Learning Representations (ICLR)*, 2024.

[14] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *International Conference on Learning Representations (ICLR)*, 2023.

[15] Kin G. Olivares, Cristian Challu, Grzegorz Marcjasz, Rafal Weron, and Artur Dubrawski. Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx. *International Journal of Forecasting*, 2023.

[16] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *International Conference on Learning Representations (ICLR)*, 2020.

[17] Vincent Quenneville-Belair, Malcolm Wolff, Brady Willhelme, Dhruv Madeka, and Dean Foster. Distribution-free multi-horizon forecasting and vending system. In *KDD 2023 International Workshop on Mining and Learning from Time Series (MileTS)*, 2023. URL `https://www.amazon.science/publications/distribution-free-multi-horizon-forecasting-and-vending-system`.

[18] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting. *arXiv:2310.08278*, 2023.

[19] Olivier Sprangers, Sebastian Schelter, and Maarten de Rijke. Parameter-efficient deep probabilistic forecasting. *International Journal of Forecasting*, 2023.

[20] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. *International Conference on Learning Representations (ICLR)*, 2024.

[21] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[22] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers Effective for Time Series Forecasting? *arXiv:2205.13504*, 2022.

[23] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

[24] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. *International Conference on Machine Learning (ICML)*, 2022.

A Appendix

A.1 Depiction of decoder process

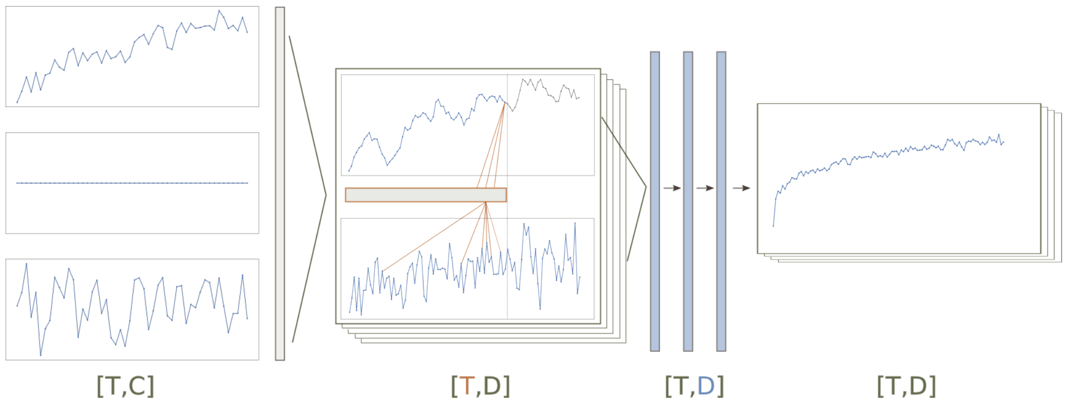

Figure 2: **Depiction of a block in the `RevPrev` decoder**. From left to right. First `RevPrev` receives different time series information, both the target sequence and the exogenous information. Then `RevPrev` uses an `SSM` layer to find the relationships of the information across time. Next `RevPrev` mixes the feature information with a `MLP` layer creating another representation with both time structure that gets passed to the next block.