Transformer Architectures in Time Series Analysis: A Review

Maohua Liu

School of Electrical and Computer Engineering, University of Georgia, Athens, Georgia, USA

1. UniTS

Different from all task-specific or domain-specific models, UniTS is a unified Transformer-based time series model that supports a universal task specification as well as domain specification, accommodating classification, forecasting, imputation, anomaly detection tasks as well as different domains such as human activity, healthcare, mechanical sensors and finance domains, as shown in Figure 1 [1]. Not only that, UniTS can accommodate time sequences with different length and variables/sensors. Surprisingly, UniTS still demonstrates superior performance compared to task-specific models and repurposed natural language-based LLMs. It is most like the counterpart of ChatGPT in time series analysis.

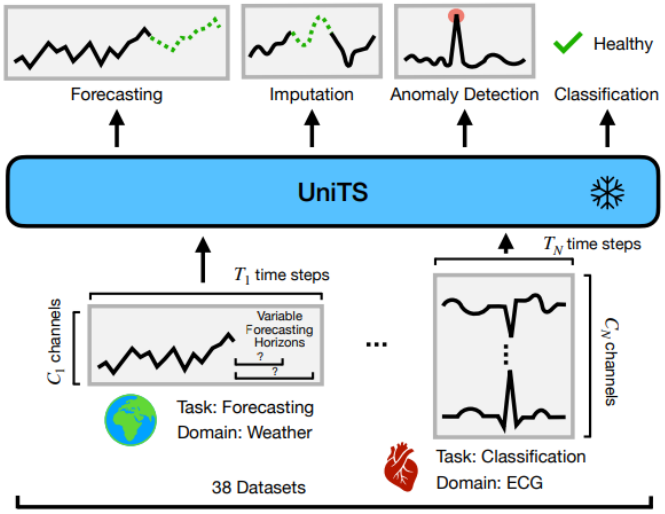

Figure 1: UniTS is a unified time series model that can process various tasks across multiple domains with shared parameters and does not have any task-specific modules [1].

It is not easy to build such a unified model because there are lots of challenges: (1) **Cross-domain temporal dynamics.** Unified models aim to learn general knowledge by co-training on diverse data sources, but time series data present wide variability in temporal dynamics across domains, along with heterogeneous data representations. This heterogeneity hinders the use of unified models developed for other domains. (2) **Diverging task specifications.** Diverging task specifications in time series data pose another challenge, as common tasks like forecasting and classification have fundamentally different objectives and may require varying specifications across datasets. (3) **Requirement for task-specific time series modules.** The distinct task-specific modules for each dataset in previous approaches are crucial for ensuring performance. However, Unified models employ shared weights across various tasks, hindering rapid adaptation to new tasks.

UniTS overcomes above challenges by employing: (1) **Unified token representation.** UniTS employs a prompting-based framework to convert various tasks into a unified token representation, creating a universal specification for all tasks. As shown in Figure 2(a), instead of using separate and different tokens for each head/task, UniTS adopts unified tokens within a unified token space. Different tokens decide different specific tasks, thereby complete each specific task by handling the unified output space, as shown in Figure 2(b,c). (2) **Flexible structure to accommodate diverse data.** UniTS utilizes self-attention across both sequence and variable dimensions to accommodate diverse data shapes, with a dynamic linear operator introduced to model dense relations between data points in sequences of any length. This approach allows UniTS to process multi-domain time series with diverse variables and lengths without requiring modification of the network structure, as shown in Figure 2(d). (3) **A unified reconstruction pretraining scheme.** A unified masked reconstruction pretraining scheme is introduced to handle both generative and recognition tasks within the unified model.

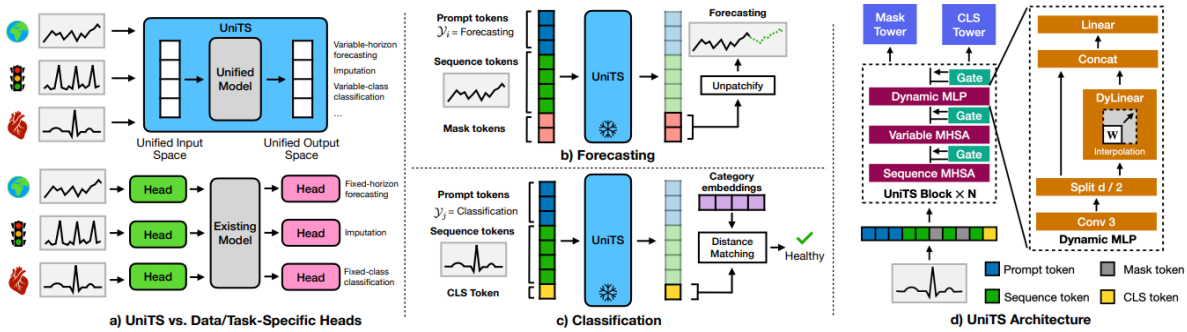

Figure 2: a) Compares UniTS to existing models, where UniTS model can process diverse inputs and achieve multiple time series tasks; previous models require separate modules for different tasks and input datasets. b) UniTS for forecasting; input is tokenized as described in-text and mask tokens are un-patchified to infer the forecast horizon. c) UniTS for classification; a CLS token is used to represent class information, and then compared to category tokens to get prediction class. d) Architecture of UniTS model [1].

As shown in Figure 2(d), UniTS has a typical yet different Transformer architecture. There are three important innovations in the architecture of UniTS. First is its unified and extended token space. Each token incorporates prompt, sequence and class, and this kind of extended token make it possible to build a unified model for time series analysis. Second is the variable Multi-Head Self-Attention (MHSA). The Variable MHSA can simultaneously capture global relations along both sequence and variable dimensions in diverse data domains. This approach accommodates variations in sequence length and the number of variables across different domains. Third is the dynamic MLP, one kind of self-adaptive self-attention mechanism. Unlike Sequence MHSA's similarity-based relation modeling, UniTS introduces Dylinear, a dynamic linear operator. Dylinear is designed to effectively model dense relations among tokens of varying sequence lengths across different data sources. UniTS is open source, please find its code at https://github.com/mims-harvard/UniTS

References:

[1] S. Gao, T. Koker, Q. Queen, T. Hartvigsen, T. Tsiligkaridis, and M. Zitnik, "UniTS: Building a Unified Time Series Model." arXiv preprint arXiv:2403.00131 (2024).

2. EEG-ConvTransformer and EEGformer

Both EEG-ConvTransformer and EEGformer are transformer-based architectures designed for EEG data analysis. They both utilize a combination of CNNs and self-attention mechanisms to extract local and global features efficiently, which is why they are discussed together. Despite their shared conceptual foundation, they have distinct structures, each of which is described separately below.

Figure 3 illustrates the architecture of EEG-ConvTransformer [2]. Initially, the input undergoes processing by a Local Feature Extractor (LFE) unit, which conducts initial feature extraction from the raw data. The output of the LFE unit is then fed into two consecutive ConvTransformer units, which further extract features from both spatial and temporal dimensions. In each ConvTransformer, the MultiHead Attention block focuses on extracting spatial features, while the Convolutional Feature Expansion block is responsible for extracting temporal features. The architecture of the ConvTransformer differs from that of a traditional transformer by replacing the traditional MLP block with the Convolutional Feature Expansion block, as the convolution module is more effective in processing EEG data than the MLP block. Subsequently, a Convolutional Encoder is used to encode the output of the ConvTransformer further. Finally, the encoder's output is flattened and used for classification. The name "EEG-ConvTransformer" reflects the integration of numerous convolution modules into its architecture, making it suitable for EEG data analysis.

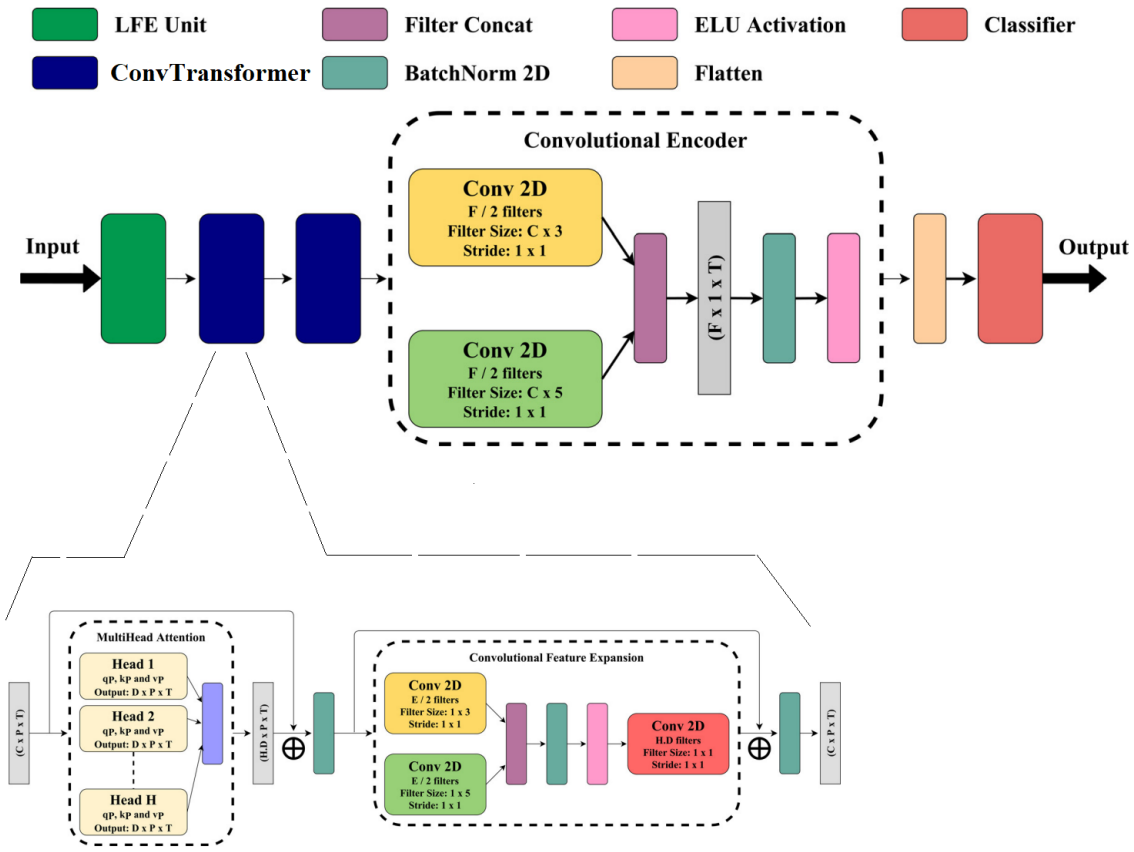

Figure 3: Architecture of EEG-ConvTransformer [2].

Figure 4 illustrates the architecture of EEGformer [3]. Similar to EEG-ConvTransformer, the input is first processed by a depth-wise CNN (1DCNN) unit to conduct preliminary feature extraction from the raw data. The output of the 1DCNN unit then passes through the EEGformer Encoder, which comprises three transformer units for feature extraction from different dimensions. One distinction from EEG-ConvTransformer is that EEGformer uses a traditional transformer architecture without modifications. Additionally, EEGformer employs three transformer units, whereas EEG-ConvTransformer uses two. Finally, the output of the encoder is passed through a decoder for specific tasks.

Both EEG-ConvTransformer and EEGformer emphasize the importance of CNNs in compressing EEG data, likely due to the low signal-to-noise ratio (SNR) of EEG data. Without compression, the transformer's ability to learn may be significantly hindered by noise. Both architectures primarily employ the transformer architecture for feature extraction, focusing more on the use of self-attention mechanisms than other transformer concepts such as tokenization and position embedding. Both architectures have demonstrated good performance in classification tasks across some datasets.

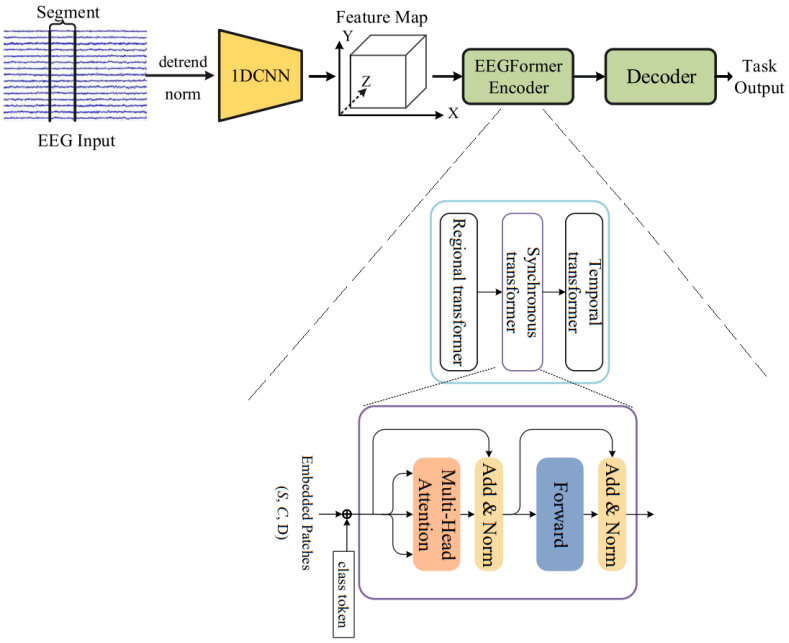

Figure 4: Architecture of EEGformer [3].

References:

[2] S. Bagchi, and D.R. Bathula, "EEG-ConvTransformer for single-trial EEG-based visual stimulus classification." Pattern Recognition 129 (2022): 108757.

[3] Z. Wan, M. Li, S. Liu, J. Huang, H. Tan, and W. Duan, "EEGformer: A transformer–based brain activity classification method using EEG signal." Frontiers in Neuroscience 17 (2023): 1148855.

3. Mamba

Since 2024, there has been a growing interest in using Mamba for time series analysis. Researchers have published around four papers on this topic, including "Is Mamba Effective for Time Series Forecasting?" [4], "TimeMachine: A Time Series is Worth 4 Mambas for Long-term Forecasting" [5], "MambaStock: Selective State Space Model for Stock Prediction" [6], and "SiMBA: Simplified Mamba-Based Architecture for Vision and Multivariate Time Series" [7]. These studies consistently show that Mamba surpasses previous models in both accuracy and efficiency for time series analysis.

Mamba was developed to enhance the efficiency of the Transformer architecture. While the self-attention mechanism in Transformers is powerful, it comes with a high computational cost due to its quadratic complexity. In contrast, Mamba can model long-sequence dependencies like the Transformer but with a near-linear computational complexity, resulting in significantly higher speed and efficiency.

As a State Space Model (SSM), Mamba's architecture shares some similarities with the Transformer but also has distinct differences. For instance, as depicted in Figure 5, Mamba's block resembles that of the Transformer, but its underlying mechanism is entirely different. Unlike the Transformer, Mamba does not run the same attention mechanism of Transformer. Instead, its core concept is based on compressing continuous time series using orthogonal polynomials [8], with the "state space" referring to the space constructed from dimensions corresponding to the orthogonal polynomial bases. This approach gives Mamba a natural advantage in modeling continuous data like time series, as all orthogonal polynomial bases are continuous. For more information, readers are encouraged to consult the following papers [9] [10] [11].

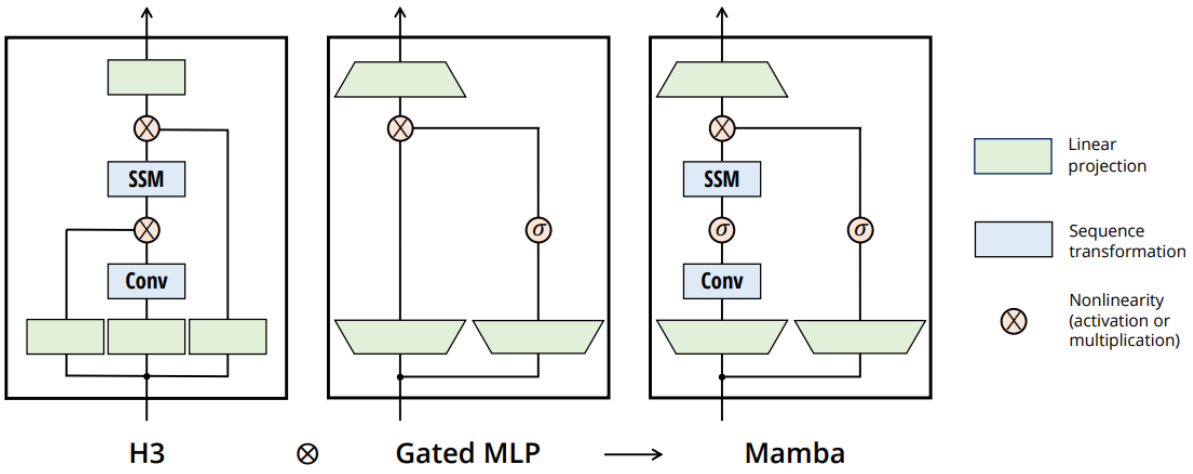

Figure 5: H3 block is the basis of most SSM architectures, with the ubiquitous MLP block of modern neural networks. Instead of interleaving these two blocks, we simply repeat the Mamba block homogenously. Compared to the H3 block, Mamba replaces the first multiplicative gate with an activation function. Compared to the MLP block, Mamba adds an SSM to the main branch.

References:

[4] z. Wang, F. Kong, F. Shi, M. Wang, H. Zhao, D. Wang, and Y. Zhang, "Is Mamba Effective for Time Series Forecasting?." arXiv preprint arXiv:2403.11144 (2024).

[5] A. M. Atik, and Q. Cheng, "TimeMachine: A Time Series is Worth 4 Mambas for Long-term Forecasting." arXiv preprint arXiv:2403.09898 (2024).

[6] Z. Shi, "MambaStock: Selective state space model for stock prediction." arXiv preprint arXiv:2402.18959 (2024).

[7] P. N. Badri, and V. S. Agneeswaran, "SiMBA: Simplified Mamba-Based Architecture for Vision and Multivariate Time series." arXiv preprint arXiv:2403.15360 (2024).

[8] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections." Advances in neural information processing systems 33 (2020): 1474-1487.

[9] A. Gu, and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces." arXiv preprint arXiv:2312.00752 (2023)

[10] A. Gu, K. Goel, and C. Re, "Efficiently modeling long sequences with structured state spaces." arXiv preprint arXiv:2111.00396 (2021).

[11] D.Y. Fu, T. Dao, K.K. Saab, A.W. Thomas, A. Rudra, and C. Ré, "Hungry hungry hippos: Towards language modeling with state space models." arXiv preprint arXiv:2212.14052 (2022).