TSGym: Design Choices for Deep Multivariate Time-Series Forecasting

Shuang Liang[1,*], **Chaochuan Hou**[1,*], **Xu Yao**[1], **Shiping Wang**[3],
Minqi Jiang[1,*,†], **Songqiao Han**[1,2,†], **Hailiang Huang**[1,2,†],
[1]AI Lab, Shanghai University of Finance and Economics [2]Ant Group
[3]MoE Key Laboratory of Interdisciplinary Research of Computation and Economics
{liangs1104,houchaochuan,yaoxu}@stu.sufe.edu.cn, shiping.wsp@antgroup.com,
{jiangminqi,han.songqiao,hlhuang}@shufe.edu.cn,

Abstract

Recently, deep learning has driven significant advancements in multivariate time series forecasting (MTSF) tasks. However, much of the current research in MTSF tends to evaluate models from a holistic perspective, which obscures the individual contributions and leaves critical issues unaddressed. Adhering to the current modeling paradigms, this work bridges these gaps by systematically decomposing deep MTSF methods into their core, fine-grained components like series-patching tokenization, channel-independent strategy, attention modules, or even Large Language Models and Time-series Foundation Models. Through extensive experiments and component-level analysis, our work offers more profound insights than previous benchmarks that typically discuss models as a whole.

Furthermore, we propose a novel automated solution called TSGym for MTSF tasks. Unlike traditional hyperparameter tuning, neural architecture searching or fixed model selection, TSGym performs fine-grained component selection and automated model construction, which enables the creation of more effective solutions tailored to diverse time series data, therefore enhancing model transferability across different data sources and robustness against distribution shifts. Extensive experiments indicate that TSGym significantly outperforms existing state-of-the-art MTSF methods. All code is publicly available on https://github.com/SUFE-AILAB/TSGym.

1 Introduction

Multivariate time series refer to time series data involving multiple interdependent variables, which are widely present in various fields such as finance [51], energy [5, 17], traffic [15, 69], and health [9, 29]. Among the numerous analysis tasks, multivariate time series forecasting (MTSF) attracts substantial attention from the research community due to its significant practical applications. Traditional approaches to MTSF are largely based on statistical methods [4, 72] and machine learning techniques [23, 44]. In recent years, deep learning (DL) has become the most active area of research for MTSF, driven by its ability to handle complex patterns and large-scale datasets effectively [59].

Early academic efforts of deep MTSF methods like RNN-type methods [67] are reported to struggle with capturing long-term temporal dependencies due to their inherent limitations of gradient vanishing or exploding problem [76, 78]. More recently, Transformer [56] shows significant potential, largely due to the effectiveness of its attention mechanisms in modeling temporal correlation [56, 62]. Consequently, attention mechanism has continuously been studied in MTSF, with a focus on adapting them to time series data, for instance, by exploiting sparsity inductive bias [33, 76], transforming time and frequency domains [78], and fusing multi-scale series [36]. While simpler MLP-based structures

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

emerged [70] offering alternatives to the established Transformer architecture in MTSF, notable modeling strategies like series-patching and channel-independent [45], significantly enhanced the performance of Transformer-based methods, thereby sustaining research interest in them. Building upon these developments, large time-series models including large language models (LLMs) [28, 79, 25] and time series foundation models (TSFMs) [27, 42] have recently been introduced, achieving promising results and fostering new research directions for MTSF. Alongside these advancements in model architectures, active research within the deep MTSF community also focuses on other critical topics, such as variable (channels) dependency modeling [45, 37, 75], series normalization methods [40, 19], and trend-seasonal decomposition [70, 38].

As the field of MTSF continues to diversify, existing studies typically address critical concerns about methodological effectiveness, either by conducting large-scale benchmarks [59, 52, 47] or performing model selection via AutoML [2, 20]. However, we identify three main challenges with these prevailing approaches: First, *the granularity of existing studies is insufficient*. Current benchmarking works evaluate or select models as a whole, which hinders a deeper understanding of the mechanisms that drive model performance. In AutoML, this lack of granularity prevents breakthroughs beyond the limits of existing models. Second, *the scope of existing studies is limited*. Current benchmarking and automated selection efforts are often confined to restricted model architectures or hyperparameters, without covering a broad range of data processing methods or feature modeling techniques. Third, *the range of existing studies is narrow*. Existing studies tend to cover only a subset of network architectures and often lack discussions on more diverse models, such as LLMs and TSFMs.

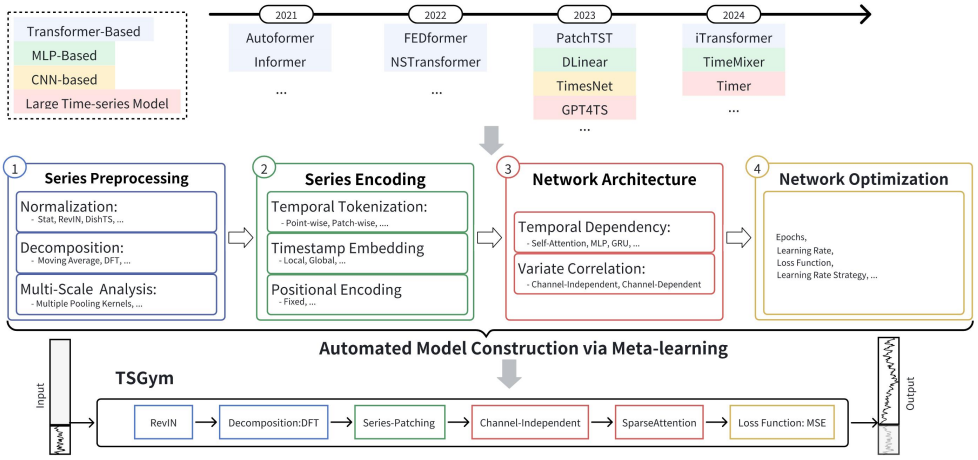


Figure 1: The Pipeline of Existing Multivariate Time Series Forecasting Methods.

To bridge these gaps, we propose TSGym—a framework designed for the **Large-scale Evaluation**, **Analysis**, and **Automated Model Construction** in deep MTSF tasks. Rather than viewing models as unified entities, TSGym systematically deconstructs popular deep MTSF methods by organizing them into distinct design dimensions that cover the entire time series modeling pipeline (see Fig. 1 and Table 1). Through extensive experiments, TSGym conducts fine-grained, isolated evaluations of core components, thereby identifying key design dimensions/choices and valuable insights from the vast MTSF methods. Moreover, TSGym proposes the first component-level model construction in MTSF tasks, which effectively overcomes limitations in the previous automation methods by enabling more flexible and customized model designs tailored to data characteristics. Extensive experimental results indicate that the proposed TSGym generally outperforms existing SOTA methods. We summarize the key contributions of TSGym as follows:

Component-level evaluation of MTSF methods. We propose TSGym, the first large-scale benchmark that systematically decouples deep MTSF methods. By evaluating 16 design dimensions across 10 benchmark datasets, TSGym offers key insights to inform future development for MTSF.

Automated MTSF model construction. Leveraging meta-learning, TSGym develops models that outperform current SOTA methods, offering the MTSF community an effective, automated, and data-adaptive solution for model design.

Discussion on emerging large time-series models. TSGym broadens current MTSF scope by applying systematic evaluation and automated combination not only to well-established models like MLP and Transformer, but also to novel large time-series models like LLMs and TSFMs.

2 Related Work

2.1 Deep Learning-based MTSF

MTSF evolves from traditional statistical methods like ARIMA and Gaussian processes to modern deep learning approaches. Recurrent Neural Networks (RNNs) introduce memory mechanisms for sequential data but struggle with long-term dependencies. Temporal Convolutional Networks (TCNs) improve this by capturing multi-scale patterns, though their fixed window sizes limit global context. Transformers, using self-attention, enable long-range forecasting but introduce high computational complexity, leading to efficient variants like sparse attention [65] and patch-based models [45]. Multilayer Perceptrons (MLPs) regain attention as simple yet effective models [71], with numerous variants offering competitive performance [14, 68, 16, 38]. Leveraging NLP foundation models, LLM adaptation approaches use frozen backbones and prompt engineering [26, 79] or fine-tuning [11] to transfer pretrained knowledge. Simultaneously, pure TSFMs trained on large datasets achieve zero-shot generalization [42, 21], though constrained by Transformers' complexity. Our TSGym framework modularizes six core backbones—RNNs, CNNs, Transformers, MLPs, LLMs, and TSFMs—offering flexible, hybrid integration based on temporal dependencies and resource needs.

In recent advancements in MTSF, we summarize the design paradigm through a unified pipeline (Fig. 1), consisting of four stages: *Series Preprocessing→Series Encoding→Network Architecture→Network Optimization*. Additionally, several specialized modules are proposed to enhance predictive accuracy by addressing non-stationarity, multi-scale dependencies, and inter-variable interactions. We categorize these developments into 6 specialized modules:

(1) Normalization methods like RevIN [30] adjust non-stationary data, improving robustness against distribution shifts. **(2) Decomposition methods**, such as Autoformer [65]'s trend-seasonality separation, isolate non-stationary components, making the data more predictable by separating trends from seasonality. **(3) Multi-scale analysis** extracts temporal patterns across granularities, as in TimeMixer [58], capturing both high-frequency fluctuations and low-frequency trends through hierarchical resolution modeling. **(4) Temporal tokenization techniques** like PatchTST[45]'s subseries-level embedding represent time series hierarchically, improving the capture of complex temporal semantics. **(5) Temporal dependency** modeling through architectures like Transformers leverages self-attention to capture long-range dependencies, effectively modeling both short- and long-term relationships. **(6) Variate correlation learning**, exemplified by DUET [48], models inter-variable dependencies using frequency-domain metric learning, improving predictions by capturing interactions across variables.

To provide a more detailed categorization and comprehensive technical specifications, please refer to Appx. B. Due to the extensive focus and continuous evolution of these modules in MTSF research, TSGym strives to decouple and modularize these key modules, exploring their real contributions and enabling more flexible model structure selection and configuration.

2.2 Benchmarks for Time Series Forecasting

Recent time series forecasting benchmark studies [59, 52, 47, 41] have conducted large-scale experiments across a diverse range of datasets. However, most of these works treat current models as monolithic entities. TSlib[1] [59], one of the most popular repositories for time series analysis, provides a comprehensive survey and evaluates recent time series models across various time series analysis tasks. From the perspective of time series characteristics, BasicTS [52] analyzes model architectures and the strategy of treating channels (or variables) independently. With a more extensive experimental setup, TFB [47] additionally includes machine learning and statistical forecasting methods, and covers datasets from a broader range of domains. More recently, OpenLTM[2] [41] provides a system to evaluate Time Series Foundation Models as well as Large language Models for time series methods.

Although some surveys and benchmarks analyze the fine-grained components of time series models, their scope is often limited. Wen et al. [61] discuss various time series data augmentation techniques

[1] https://github.com/thuml/Time-Series-Library

[2] https://github.com/thuml/OpenLTM

and evaluate their effectiveness. Another survey [62] systematically reviews fine-grained components within Transformer-based architectures, but lacks broader coverage of model structures and experimental evaluations. To the best of our knowledge, TSGym is the first benchmark that not only provides component-level fine-grained analysis, but also conducts large-scale empirical evaluations.

2.3 AutoML for Time Series Forecasting

Current automated approaches for DL-based MTSF can be categorized into ensemble-based [53] and meta-learning-based [2, 20] methods. The former fits and integrates various models from a predefined pool with ensemble techniques, which inevitably incurs substantial computational cost. The latter leverages meta-features to characterize datasets and selects optimal models for the given datasets. However, both approaches operate at the model level and struggle to surpass the performance ceiling of existing methods. AutoCTS++ [66] achieves automated selection by searching over model architectures and hyperparameters, but its search space is limited in scope. In contrast, TSGym is the first framework to support automated selection over a wide range of fine-grained components for MTSF, extending beyond narrow model structures, hyperparameters, and data processing strategies.

A closely related work is our previous effort, ADGym[24], which is designed for tabular anomaly detection with model decomposition. Differently, TSGym deals with multivariate time series data, which presents more complex data processing design choices, such as series sampling, series normalization, and series decomposition. Besides, TSGym considers finer-grained model structures, such as various attention variants in Transformers, and broader network types, including LLMs and TSFMs. It is worth mentioning that the success of TSGym validates the universality of the model decomposition framework, marking an innovation and progression distinct from ADGym. Further details on the differences between two works can be found in Appx.F.

3 TSGym: *Benchmarking* and *Automating* Design Choices in Deep MTSF

3.1 Problem Definition for MTSF

In this paper, we focus on the common MTSF settings for time series data containing C variates. Given historical data $\chi = \{\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_C^t\}_{t=1}^{L}$, where L is the look-back sequence length and $\boldsymbol{x}_i^t$ is the i-th variate, the forecasting task is to predict T-step future sequence $\hat{\chi} = \{\hat{\boldsymbol{x}}_1^t, \ldots, \boldsymbol{x}_C^t\}_{t=L+1}^{L+T}$. To avoid error accumulation ($T > 1$), we directly predict all future steps, following [76].

3.2 Large Benchmarking towards Design Choices of Deep MTSF

Considering the above numerous methods proposing for MTSF tasks, the foremost priority involves decoupling the current state-of-the-art (SOTA) methods and further conducting large-scale benchmark to identify the core components that really drive the improvements in time-series forecasting.

Following the taxonomy of the previous study [62, 71], we decouple existing SOTA methods according to the standard process of MTSF modeling, while significantly expanding the diversity of the modeling pipeline. Based on the flow direction from the input to the output sequence, the **Pipeline** of TSGym includes: *Series Prepropcessing→Series Encoding→Network Architecture→Network Optimization*, as is demonstrated in Fig. 1. Moreover, we structure each pipeline step according to distinct **Design Dimensions**, where a DL-based time-series forecasting model can be instantiated by specified **Design Choices**, as is shown in Table 1.

Through the proposed design dimensions and choices, TSGym provides detailed description of time-series modeling pipeline, disentangling key elements within mainstream time-series forecasting methods and facilitating component-level comparison/automated construction. For example, TSGym includes multi-scale mixing module proposed in TimeMixer [58], Inverted Encoding method proposed in iTransformer [37], Channel-independent strategy and Series-Patching encoding used in PatchTST [45], various attention mechanism discussed in [62], and also LLM and TSFM network type choices that are often integrated without fully considering their interactions with other design dimensions. Detailed descriptions of all design choices are provided in Appx. G.1.

3.3 Automated construction MTSF models via TSGym

Overview. Differing from traditional methods that focus on selecting an off-the-shelf model, TSGym aims to customize models given the downstream MTSF tasks and data descriptions. Given a pre-defined conflict-free model set $\mathcal{M} = \{M_1, ..., M_m\}$, each model M_i is instantiated by the

Table 1: TSGym supports comprehensive design choices for deep time-series forecasting methods.

Pipeline	Design Dimensions	Design Choices
↓Series Preprocessing	Series Normalization	[None, Stat, RevIN, DishTS]
	Series Decomposition	[None, MA, MoEMA, DFT]
	Series Sampling/Mixing	[False, True]
↓Series Encoding	Channel Independent	[False, True]
	Sequence Length	[48, 96, 192, 512]
	Series Embedding	[Inverted Encoding, Positional Encoding, Series Patching]
↓Network Architecture	Network Type	[MLP, RNN, Transformer, LLM, TSFM]
	Series Attention	[Null, SelfAttn, AutoCorr, SparseAttn, FrequencyAttn, DestationaryAttn]
	Feature Attention	[Null, SelfAttn, SparseAttn, FrequencyAttn]
	d_model	[64, 256]
	d_ff	[256, 1024]
	Encoder Layers	[2, 3]
↓Network Training	Epochs	[10, 20, 50]
	Loss Function	[MSE, MAE, HUBER]
	Learning Rate	[1e-3, 1e-4]
	Learning Rate Strategy	[Null, Type1]

design choice combinations illustrated in Table 1. TSGym learns the mapping function from these automatically combined models to their associated forecasting performance on the training datasets, and generalize to the test dataset(s) to select the best model based on predicted results.

Meta-learning for automated MTSF model construction. Formally speaking, TSGym propose k design dimensions $\mathcal{DD} = \{DD_1, ..., DD_k\}$ for comprehensively describing each step of aforementioned pipeline in deep learning time-series modeling. Each design dimension DD_i represents a set containing elements of different design choices DC. By taking the Cartesian product of the sets $\mathcal{DD}$ corresponding to different design dimensions, we obtain the pool of all valid model combinations $\mathcal{M} = DD_1 \times DD_2 \times \cdots \times DD_k = \{(DC_1, DC_2, \ldots, DC_k) \mid DC_i \in DD_i, i = 1, 2, \ldots, k\}$. Considering the potentially large number of combinations and the computational cost, we randomly sampled $\mathcal{M}$ to $\mathcal{M}_s$, where $M_i = (DC_1 = RevIN, DC_1 = DFT, ..., DC_k = Type1) \in \mathcal{M}_s$, for example, which means M_i instantiates RevIN method to normalize input series, then decompose it to the seasonal and trend term. Subsequently, following the Series Encoding and Network Architecture constructing pipeline (as illustrated in Table 1), finally the Type1, i.e., a step decay learning rate strategy is employed to adjust the learning rate for updating the model parameters.

Suppose we have n training datasets $\boldsymbol{\mathcal{D}}_{\text{train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ and the number of sampled model combinations (i.e., the size of the set $\mathcal{M}_s$) is m, TSGym conducts extensive experiments on n historical training datasets to evaluate and further collect the forecasting performance of m model combinations. TSGym then acquire the MSE performance matrix $\boldsymbol{P} \in \mathbb{R}^{n \times m}$, where $\boldsymbol{P}_{i,j}$ corresponds to the j-th auto-constructed MTSF model's performance on the i-th training dataset. Since the difficulty of prediction tasks varies across training datasets, leading to significant differences in the numerical range of performance metrics. Directly using these metrics (e.g., MSE) as training targets of a meta-learner may result in overfitting on more difficult dataset(s). Therefore, we convert the performance metrics of $\mathcal{M}_s$ into their corresponding normalized ranking, where $\boldsymbol{R}_{i,j} = rank(P_{i,j})/m \in [0, 1]$ and smaller values indicate better performance on the corresponding dataset.

Distinguished from previous model selection approaches [2, 3], TSGym decouples more recently MTSF methods (including MLP-Mixer-type, Transformer-based, LLM and TSFM models), and supports fine-grained model construction at the component level, rather than being constrained to a fixed, limited set of existing models, which enables significantly greater flexibility and effectiveness. Specifically, TSGym follows the idea of meta-learning to construct a meta-learner that learns the mapping function $f(\cdot)$ from training dataset $\mathcal{D}_i$ and model combination M_j, to the performance rankings $\boldsymbol{R}_{i,j}$, as is shown in Eq. 1. We leverage meta-features $\mathbf{E}_i^{meta}$ that capture multiple aspects such as statistical, temporal, spectral, and fractal features, and distribution shift metrics to fully describe the complex data characteristics of time series datasets. Learnable continuous embeddings $\mathbf{E}_j^{comp}$ are used to represent different model combinations and are updated through the gradient backpropagation of the meta-learner.

$$f(\mathcal{D}_i, M_j) = \boldsymbol{R}_{i,j}, f \ : \ \underbrace{\mathbf{E}_i^{meta}}_{\text{meta features}} \ , \ \underbrace{\mathbf{E}_j^{comp}}_{\text{component embed.}} \mapsto \boldsymbol{R}_{i,j} \ , \ i \in \{1, \ldots, n\}, \ j \in \{1, \ldots, m\} \quad (1)$$

We used a simple two-layer MLP as the meta-learner and trained it through a regression problem, thereby transferring the learned mapping to new test datasets. For a newcoming dataset (i.e., test dataset $\mathbf{X}_{\text{test}}$), we acquire the predicted relative ranking of different components using the trained $f(\cdot)$, and select top-1 (k) to construct MTSF model(s). Note this procedure is zero-shot without needing any neural network training on $\mathbf{X}_{\text{test}}$ but only extracting meta-features and pipeline embeddings. We show the effectiveness of the meta-predictor in §4.3.

4 Experiments

4.1 Experiment Settings

Datasets. Following most prior works [65, 64, 26], we adopt 9 datasets as experimental data for MTSF tasks, ETT (4 subsets), Traffic, Electricity, Weather, Exchange, ILI. And we utilize the M4 dataset for short-term forecasting tasks. The forecast horizon L for long-term forecasting is $\{96, 192, 336, 720\}$, while for the ILI dataset, it is $\{24, 36, 48, 60\}$. For short-term forecasting, the forecast horizons are $\{6, 8, 13, 14, 18, 48\}$. More details can be seen in Appx. A.

Baseline. We present a comprehensive set of baseline comparison experiments to demonstrate the superior performance of the pipelines automatically constructed by TSGym. Due to space limitations, the baseline methods presented in this section include the latest clustering-based approach DUET [48], time series mixing methods (TimeMixer [58]), MLP-based methods (MICN [57]), RNN-based methods (SegRNN [34]), CNN-based methods (TimesNet [64]), and Transformer-based methods (PatchTST [45], Crossformer [75], Autoformer [65]). We present experiments based on the complete baseline in the Appx. H.

Evaluation Metrics. We follow the experimental setup of most prior works, using Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics for long-term forecasting tasks, and using Symmetric Mean Absolute Percentage Error (SMAPE), Mean Absolute Scaled Error (MASE), and Overall Weighted Average (OWA) as metrics for short-term forecasting tasks. The mathematical formulas for these evaluation metrics are provided in the Appx. D.

Meta-predictor in TSGym. The meta-predictor is instantiated as a two-layer MLP and trained for 100 epochs with early stopping. The training process utilizes the Adam optimizer [31] with a learning rate of 0.001 and batch size of 512. See details in Appx. G.2.

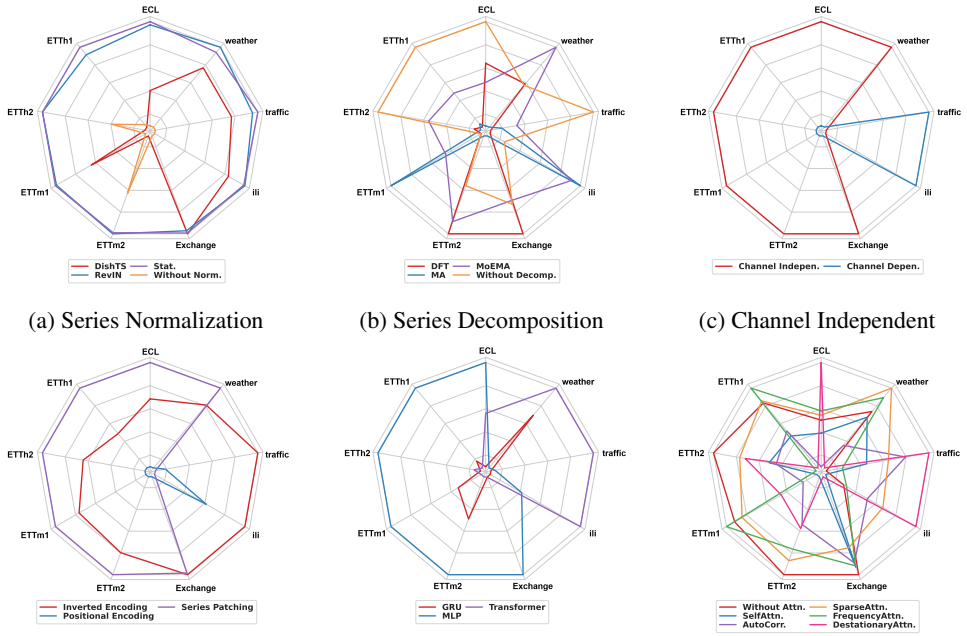
4.2 Large Evaluation on AD Design Choices

In this work, we perform large evaluations on the decoupled pipelines according to the standard procedure of MTSF methods. Such analysis is often overlooked in previous studies, and we investigate each design dimension of decoupled pipelines by fixing its corresponding design choice (e.g., Self Attention), and randomly sampling other dimensional design choices to construct MTSF pipelines.

In the following sections, we provide systematic conclusions based on long-term MTSF experimental results, addressing several gaps in the current MTSF research community. Specifically, different design choices are compared and demonstrated using a spider chart, where each vertex represents a dataset. Design choices that are closer to the vertices exhibit superior performance in their respective design dimensions. We analyze these components based on the different design dimensions, namely ***Series Preprocessing***, ***Series Encoding*** and ***Network Construction***. Finally, we evaluated the performance on four datasets under the fixed model architectures of LLMs or TSFMs, and discussed the effects of various design choices in the subsection on ***Large Time Series Models***. More detailed information, including the complete design choices and their performance on short-term time series forecasting tasks, is provided in the Appx. H.

Series Preprocessing. Based on the systematic evaluation of Series Preprocessing strategies in Fig. 2a and Fig. 2b, several key insights emerge. Series Normalization (Fig. 2a) proves universally effective, with RevIN and Stationary achieving the lowest MSE (75th percentile) across diverse datasets, establishing them as essential baselines for stabilizing non-stationary dynamics. In contrast, Series Decomposition (Fig. 2b) demonstrates varying effectiveness depending on the dataset. For example, decomposition methods improve performance on datasets like ETTm1 when using MA, while others, such as ETTh1, ECL, ETTh2, and traffic datasets, perform better without decomposition.

Series Encoding. Several clear observations emerge from the Series Encoding stage, as illustrated in Fig.2c and Fig.2d. First, as shown in panel Fig. 2c, channel-independent methods consistently



(a) Series Normalization (b) Series Decomposition (c) Channel Independent

(d) Series Tokenization (e) Network Backbone (f) Series Attention

Figure 2: Overall performance across key design dimensions. The results (MSE) are based on the 75th percentile across all forecasting horizons.

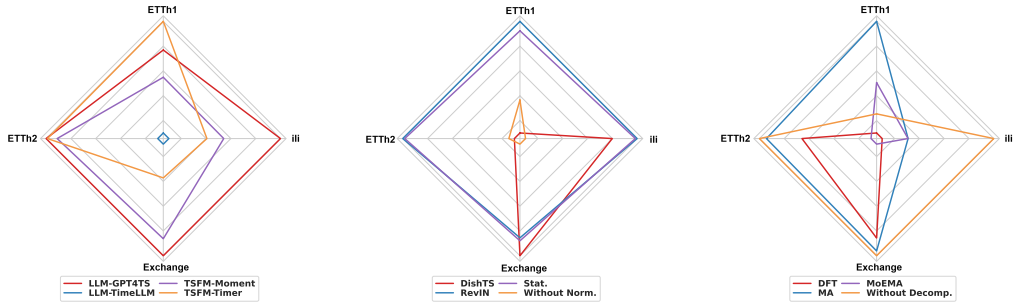
outperform channel-dependent ones, with the exception of traffic and ILI, highlighting the advantage of modeling each variable separately. Regarding tokenization strategies Fig. 2d, both patch-wise (sequence patching) and series-wise (inverted encoding) approaches outperform the traditional point-wise encoding. Among them, patch-wise encoding shows strong performance across most datasets, whereas inverted encoding proves particularly effective on the traffic dataset.

Network Construction. As is shown in Fig. 2e, we find that complex network architectures like the Transformer are not always necessary, which only performs better than MLP on weather, traffic and ILI datasets. This result is reasonable since more sophisticated DL backbone like Transformer and its variants often require specific hyperparameter combinations and tailored architectural designs to achieve satisfactory performance. We indicate that these observed results further emphasize the importance of automated model construction tailored to specific data characteristics. Indeed, we show in § 4.3 that including the Transformer architecture in the set of design choices would enhance the performance of constructed model via TSGym. Fig. 2f suggests that employing Attention mechanisms for modeling dependencies of input sequence does not offer significant advantages, which aligns with recent findings [70]. Additionally, the spider chart shows no significant performance differences were observed among different variants of attention mechanisms.

Large Time Series Models. We choose three design dimensions crucial to large time-series models for discussion, as is shown in Fig.3. The most surprising finding pertains to the model backbones. We observe that GPT4TS demonstrates stable and competitive performance across most datasets. In contrast, Time-LLM, which is also based on LLMs, exhibits opposite results. A highly plausible explanation is that time series embeddings processed through different methods struggle to align consistently with the embedding space of word representations, which is an important part in Time-LLM, resulting in suboptimal performance of Time-LLM under diverse experimental configurations.

4.3 Automatic Component Construction via TSGym

Extensive experimental results discussed above indicate that in deep time series modeling, most design choices are determined by data characteristics, meaning one-size-fits-all approaches are seldom effective. This, in turn, emphasizes the necessity of automated model construction.



(a) Network Backbone (b) Series Normalization (c) Series Decomposition

Figure 3: Overall performance of different design choices across 3 design dimensions (Network Backbone 3a, Series Normalization 2a, Series Decomposition 2b) when using LLMs or TSFMs. The results (MSE) are based on the 75th percentile across all forecasting horizons.

In this subsection, we compare the MTSF pipeline selected by TSGym with existing SOTA methods. Through large-scale experiments, we found that TSGym outperforms existing SOTA models in both long- and short-term MTSF tasks. Regarding algorithm efficiency, our experiments demonstrate that even when limited to a search pool of lightweight model structures, such as MLP and RNNs, TSGym can still achieve competitive results. We analyze the effectiveness of the pipelines automatically constructed by TSGym through five key questions as follows. Additional details, such as the results based on more metrics and more complex meta-features, can be found in the Appx.H.

Question 1: Is the model constructed by meta-predictor better than existing SOTA methods? Comprehensive forecasting results are presented in Table 2 and 3, where the best performances are highlighted in **red** and the second-best results are underlined. Compared with other state-of-the-art forecasters, TSGym demonstrates superior capability, especially in handling high-dimensional time series data. Its consistent top-ranking performance across multiple datasets underlines its robustness and effectiveness for complex multivariate forecasting tasks.

Table 2: Short-term forecasting task on M4. The results are averaged from several datasets under different sample intervals. See Table in Appendix for the full results.

Models	TSGym (ours)	TimeMixer	MICN	TimesNet	PatchTST	DLinear	Crossformer	Autoformer	SegRNN
OWA	**0.872**	0.884	0.984	0.907	0.965	0.922	8.856	1.273	1.007
SMAPE	12.013	**11.985**	13.025	12.199	12.848	12.511	>30	16.392	13.509
MASE	**1.575**	1.615	1.839	1.662	1.738	1.693	>10	2.317	1.823

Table 3: Long-term forecasting task. The past sequence length is set as 36 for ILI and 96 for the others. All the results are averaged from 4 different prediction lengths, that is $\{24, 36, 48, 60\}$ for ILI and $\{96, 192, 336, 720\}$ for the others. See Table in Appendix for the full results.

Models	TSGym (Ours)		DUET [48]		TimeMixer [58]		MICN [57]		TimesNet [64]		PatchTST [45]		DLinear [71]		Crossformer [75]		Autoformer [34]		SegRNN [65]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.357**	**0.383**	0.407	0.409	0.384	0.399	0.402	0.429	0.432	0.430	0.390	0.404	0.404	0.407	0.501	0.501	0.532	0.496	0.388	0.404
ETTm2	**0.261**	**0.319**	0.296	0.338	0.277	0.325	0.342	0.391	0.296	0.334	0.288	0.334	0.349	0.399	1.487	0.789	0.330	0.368	0.273	0.322
ETTh1	0.426	0.440	0.433	0.437	0.448	0.438	0.589	0.537	0.474	0.464	0.454	0.449	0.465	0.461	0.544	0.520	0.492	0.485	**0.422**	**0.429**
ETTh2	**0.358**	**0.400**	0.380	0.403	0.383	0.406	0.585	0.530	0.415	0.424	0.385	0.409	0.566	0.520	1.552	0.908	0.446	0.460	0.374	0.405
ECL	**0.170**	0.265	0.179	**0.262**	0.185	0.273	0.186	0.297	0.219	0.314	0.209	0.298	0.225	0.319	0.193	0.289	0.234	0.340	0.216	0.302
Traffic	**0.435**	**0.313**	0.797	0.427	0.496	0.313	0.544	0.320	0.645	0.348	0.497	0.321	0.673	0.419	1.458	0.782	0.637	0.397	0.807	0.411
Weather	**0.229**	**0.268**	0.252	0.277	0.244	0.274	0.264	0.316	0.261	0.287	0.256	0.279	0.265	0.317	0.253	0.312	0.339	0.379	0.251	0.298
Exchange	0.410	0.431	**0.322**	**0.384**	0.359	0.402	0.346	0.422	0.405	0.437	0.381	0.412	0.346	0.414	0.904	0.695	0.506	0.500	0.408	0.423
ILI	2.233	1.015	2.640	1.018	4.502	1.557	2.938	1.178	**2.140**	0.907	2.160	**0.901**	4.367	1.540	4.311	1.396	3.156	1.207	4.305	1.397
1st Count	**11**		3		0		0		1		1		0		0		0		2	

Specifically, TSGym achieves the lowest MSE and MAE on a total of 11 occasions, reflecting its strong generalization ability over both medium and long forecasting horizons. While some baseline models, like DUET, PatchTST and SegRNN, occasionally show competitive results on certain datasets. As for short-term forecasting tasks, both TSGym and TSMixer demonstrate competitive performance, with TSGym outperforming on most evaluation metrics.

Question 2: Is TSGym with lightweight architecture better than existing SOTA methods?

In the previous section, we compared TSGym using the full component pool with SOTA and found that TSGym outperforms SOTA on 66.7% of the datasets evaluated. In this ablation experiment Table

4, we specifically compare the `-Transformer` configuration of TSGym with DUET. Remarkably, even after removing Transformer-related components from the TSGym component pool and retaining only the more computationally efficient MLP- and RNN-based models, TSGym still outperforms DUET on the majority of datasets. This demonstrates the robustness and efficiency of TSGym's architecture and highlights the strong predictive power of the simplified MLP-based design.

Question 3: Does the training strategies bring significant improvement for TSGym?

Following Table 4, we find that the introduction of the resampling method enhances TSGym's meta-predictor performance across 2 datasets, improving both robustness and accuracy. The `+AllPL` configuration, which trains on datasets with varying prediction lengths and transfers this knowledge to a test set with a single prediction length, further improves generalization, with the best performance observed on the ETTh1 dataset. Additionally, removing the Transformer component (`-Transformer`) leads to performance gains on certain datasets, suggesting that a simplified MLP- or RNN-based architecture can be more effective in specific scenarios. These results highlight the flexibility of TSGym's design and the potential benefits of customizing the component pool to suit dataset characteristics.

Table 4: Ablation study evaluates the removal of Transformer-based components and different training strategies, with results averaged over all prediction lengths, and the final row shows how often TSGym variants outperform DUET.

Models	**TSGym**		**-Transformer**		**+Resample**		**+AllPL**		**DUET**	
Metric	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTm1	**0.357**	**0.383**	0.363	0.388	0.361	0.384	0.367	0.391	0.407	0.409
ETTm2	0.261	**0.319**	0.274	0.330	**0.260**	0.320	0.265	0.320	0.296	0.338
ETTh1	0.426	0.440	0.441	0.450	0.432	0.440	**0.424**	**0.434**	0.433	0.437
ETTh2	0.358	0.400	0.358	0.401	**0.357**	**0.398**	0.361	0.404	0.380	0.403
ECL	**0.170**	0.265	0.174	0.273	0.170	0.265	0.185	0.277	0.179	**0.262**
Traffic	0.435	0.313	**0.415**	**0.293**	0.429	0.307	0.429	0.306	0.797	0.427
Weather	0.229	0.268	**0.226**	**0.265**	0.229	0.267	0.234	0.271	0.252	0.277
Exchange	0.410	0.431	0.409	0.435	0.399	0.430	0.377	0.411	**0.322**	**0.384**
ILI	2.233	**1.015**	2.437	1.053	2.698	1.114	**2.173**	1.020	2.640	1.018
Better Count	**14/18**		**12/18**		**12/18**		**12/18**			

Question 4: Does large time-series models bring significant improvement for TSGym?

Table 5 evaluates the impact of incorporating LLM and TSFM into the base TSGym framework. It is evident that both the introduction of LLM and TSFM consistently improve forecasting accuracy compared to the baseline TSGym configuration. This demonstrates the effectiveness of leveraging advanced model architectures and fusion strategies to further enhance the predictive performance of TSGym, particularly on complex multivariate time series datasets.

Table 5: Ablation study of TSGym incorporating LLM and TSFM in 4 datasets. The average results of all prediction lengths are listed here.

Models	**TSGym**		**+LLM**		**+TSFM**	
Metric	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1	0.442	0.440	0.442	0.438	**0.432**	**0.437**
ETTh2	0.411	0.429	**0.364**	**0.402**	0.376	0.409
Exchange	0.673	0.522	0.503	0.466	**0.374**	**0.411**
ILI	2.682	1.112	**2.470**	**1.063**	2.860	1.163

5 Conclusions, Limitations, and Future Directions

To advance beyond holistic evaluations in multivariate time-series forecasting (MTSF), this paper introduced TSGym, a novel framework centered on fine-grained component analysis and the automated construction of specialized forecasting models. By systematically decomposing MTSF pipelines into design dimensions and choices informed by recent studies, TSGym uncovers crucial insights into component-level forecasting performance and leverages meta-learning method for the automated construction of customized models. Extensive experimental results indicate that the MTSF models constructed by the proposed TSGym significantly outperform current MTSF SOTA solutions—demonstrating the advantage of adaptively customizing models according to distinct data characteristics. Our results show that TSGym is highly effective, even without exhaustively covering all SOTA components, and TSGym is made publicly available to benefit the MTSF community.

Future efforts will focus on expanding TSGym's range of forecasting techniques with emerging techniques and refining its meta-learning capabilities by incorporating multi-objective optimization to balance predictive performance against computational costs, especially for large time-series models, while also broadening its applicability across diverse time series analysis tasks.

References

[1] Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[2] M. Abdallah, R. Rossi, K. Mahadik, S. Kim, H. Zhao, and S. Bagchi. Autoforecast: Automatic time-series forecasting model selection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 5–14, 2022.

[3] M. Abdallah, R. A. Rossi, K. Mahadik, S. Kim, H. Zhao, and S. Bagchi. Evaluation-free time-series forecasting model selection via meta-learning. *ACM Transactions on Knowledge Discovery from Data*, 2025.

[4] B. Abraham and J. Ledolter. *Statistical methods for forecasting*. John Wiley & Sons, 2009.

[5] F. M. Alvarez, A. Troncoso, J. C. Riquelme, and J. S. A. Ruiz. Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1230–1243, 2010.

[6] I. Ashrapov. Tabular gans for uneven distribution, 2020.

[7] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[8] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[9] C. Bui, N. Pham, A. Vo, A. Tran, A. Nguyen, and T. Le. Time series forecasting for healthcare diagnosis and prognostics with the focus on cardiovascular diseases. In *6th International Conference on the Development of Biomedical Engineering in Vietnam (BME6) 6*, pages 809–818. Springer, 2018.

[10] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[11] C. Chang, W.-C. Peng, and T.-F. Chen. Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *CoRR*, 2023.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[13] P. Chen, Y. ZHANG, Y. Cheng, Y. Shu, Y. Wang, Q. Wen, B. Yang, and C. Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] S.-A. Chen, C.-L. Li, S. O. Arik, N. C. Yoder, and T. Pfister. TSMixer: An all-MLP architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023.

[15] R.-G. Cirstea, B. Yang, C. Guo, T. Kieu, and S. Pan. Towards spatio-temporal aware traffic time series forecasting. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2900–2913. IEEE, 2022.

[16] A. Das, W. Kong, A. Leach, S. K. Mathur, R. Sen, and R. Yu. Long-term forecasting with tiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023.

[17] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] W. Fan, P. Wang, D. Wang, D. Wang, Y. Zhou, and Y. Fu. Dish-ts: a general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 7522–7529, 2023.

[20] R. Fischer and A. Saadallah. Autoxpcr: Automated multi-objective model selection for time series forecasting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 806–815, 2024.

[21] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski. MOMENT: A family of open time-series foundation models. In *Forty-first International Conference on Machine Learning*, 2024.

[22] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

[23] A. D. Hartanto, Y. N. Kholik, and Y. Pristyanto. Stock price time series data forecasting using the light gradient boosting machine (lightgbm) model. *JOIV: International Journal on Informatics Visualization*, 7(4):2270–2279, 2023.

[24] M. Jiang, C. Hou, A. Zheng, S. Han, H. Huang, Q. Wen, X. Hu, and Y. Zhao. Adgym: Design choices for deep anomaly detection. *Advances in Neural Information Processing Systems*, 36:70179–70207, 2023.

[25] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[26] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[27] M. Jin, Q. Wen, Y. Liang, C. Zhang, S. Xue, X. Wang, J. Zhang, Y. Wang, H. Chen, X. Li, S. Pan, V. S. Tseng, Y. Zheng, L. Chen, and H. Xiong. Large models for time series and spatio-temporal data: A survey and outlook. *CoRR*, abs/2310.10196, 2023.

[28] M. Jin, Y. Zhang, W. Chen, K. Zhang, Y. Liang, B. Yang, J. Wang, S. Pan, and Q. Wen. Position paper: What can large language models tell us about time series analysis. *arXiv e-prints*, pages arXiv–2402, 2024.

[29] S. Kaushik, A. Choudhury, P. K. Sheron, N. Dasgupta, S. Natarajan, L. A. Pickett, and V. Dutt. Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in big data*, 3:4, 2020.

[30] 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*, 2021.

[31] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] C. Li, X. Li, L. Feng, and J. Ouyang. Who is your right mixup partner in positive and unlabeled learning. In *International Conference on Learning Representations*, 2022.

[33] 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. *Advances in neural information processing systems*, 32, 2019.

[34] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, and H. Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting. *arXiv preprint arXiv:2308.11200*, 2023.

[35] M. Liu, A. Zeng, M. Chen, Z. Xu, Q. Lai, L. Ma, and Q. Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.

[36] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. 2022.

[37] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[38] Y. Liu, C. Li, J. Wang, and M. Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. *Advances in neural information processing systems*, 36:12271–12290, 2023.

[39] Y. Liu, H. Wu, J. Wang, and M. Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in neural information processing systems*, 35:9881–9893, 2022.

[40] Y. Liu, H. Wu, J. Wang, and M. Long. Non-stationary transformers: Rethinking the stationarity in time series forecasting. In *NeurIPS*, 2022.

[41] Y. Liu, H. Zhang, C. Li, X. Huang, J. Wang, and M. Long. Timer: Generative pre-trained transformers are large time series models. In *Forty-first International Conference on Machine Learning*.

[42] Y. Liu, H. Zhang, C. Li, X. Huang, J. Wang, and M. Long. Timer: Generative pre-trained transformers are large time series models. In *Forty-first International Conference on Machine Learning*, 2024.

[43] J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

[44] R. P. Masini, M. C. Medeiros, and E. F. Mendes. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.

[45] 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.

[46] X. Qiu, H. Cheng, X. Wu, J. Hu, C. Guo, and B. Yang. A comprehensive survey of deep learning for multivariate time series forecasting: A channel strategy perspective. *arXiv preprint arXiv:2502.10721*, 2025.

[47] X. Qiu, J. Hu, L. Zhou, X. Wu, J. Du, B. Zhang, C. Guo, A. Zhou, C. S. Jensen, Z. Sheng, et al. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *arXiv preprint arXiv:2403.20150*, 2024.

[48] X. Qiu, X. Wu, Y. Lin, C. Guo, J. Hu, and B. Yang. Duet: Dual clustering enhanced multivariate time series forecasting. KDD '25, page 1185–1196, New York, NY, USA, 2025. Association for Computing Machinery.

[49] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. In *ICML*, pages 4393–4402, 2018.

[50] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K. Müller, and M. Kloft. Deep semi-supervised anomaly detection. In *ICLR*. OpenReview.net, 2020.

[51] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[52] Z. Shao, F. Wang, Y. Xu, W. Wei, C. Yu, Z. Zhang, D. Yao, T. Sun, G. Jin, X. Cao, et al. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[53] O. Shchur, A. C. Turkmen, N. Erickson, H. Shen, A. Shirkov, T. Hu, and B. Wang. Autogluon–timeseries: Automl for probabilistic time series forecasting. In *International Conference on Automated Machine Learning*, pages 9–1. PMLR, 2023.

[54] M. Tan, M. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen. Are language models actually useful for time series forecasting? *Advances in Neural Information Processing Systems*, 37:60162–60191, 2024.

[55] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[57] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*, 2023.

[58] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[59] Y. Wang, H. Wu, J. Dong, Y. Liu, M. Long, and J. Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024.

[60] Y. Wang, H. Wu, J. Dong, G. Qin, H. Zhang, Y. Liu, Y. Qiu, J. Wang, and M. Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[61] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu. Time series data augmentation for deep learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4653–4660, 8 2021.

[62] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[63] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. C. H. Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.

[64] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

[65] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *NeurIPS*, 2021.

[66] X. Wu, X. Wu, B. Yang, L. Zhou, C. Guo, X. Qiu, J. Hu, Z. Sheng, and C. S. Jensen. Autocts++: zero-shot joint neural architecture and hyperparameter search for correlated time series forecasting. *The VLDB Journal*, 33(5):1743–1770, 2024.

[67] P. T. Yamak, L. Yujian, and P. K. Gadosey. A comparison between arima, lstm, and gru for time series forecasting. In *Proceedings of the 2019 2nd international conference on algorithms, computing and artificial intelligence*, pages 49–55, 2019.

[68] K. Yi, Q. Zhang, W. Fan, S. Wang, P. Wang, H. He, N. An, D. Lian, L. Cao, and Z. Niu. Frequency-domain mlps are more effective learners in time series forecasting. *Advances in Neural Information Processing Systems*, 36:76656–76679, 2023.

[69] Y. Yin and P. Shang. Forecasting traffic time series with multivariate predicting method. *Applied Mathematics and Computation*, 291:266–278, 2016.

[70] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? 2023.

[71] 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*, volume 37, pages 11121–11128, 2023.

[72] G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[73] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[74] T. Zhang, Y. Zhang, W. Cao, J. Bian, X. Yi, S. Zheng, and J. Li. Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures, 2022.

[75] Y. Zhang and J. Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

[76] 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*, volume 35, pages 11106–11115, 2021.

[77] T. Zhou, Z. Ma, Q. Wen, L. Sun, T. Yao, W. Yin, R. Jin, et al. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in neural information processing systems*, 35:12677–12690, 2022.

[78] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, 2022.

[79] T. Zhou, P. Niu, L. Sun, R. Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We systematically present the research topic, background, methodology, and conclusions of this paper in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly state the limitations of this paper in the "Conclusions, Limitations, and Future Directions" section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: the paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have open-sourced all the code and data. Anyone can directly reproduce our experimental results using the predefined scripts provided.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have open-sourced all the code and data, including component-level evaluations and the meta-learning based automated selection model. All code and datasets are publicly available at `https://github.com/SUFE-AILAB/TSGym`.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide detailed descriptions of the datasets and experimental settings, the evaluation metrics used, the baselines, and the training hyperparameters of our proposed meta predictor in the Experiment Settings and Appendix sections.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We present the error bars in the box plots shown in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: provide sufficient information on the computer resources needed to reproduce the experiments in the Appendix Section.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics, adhering to ethical standards in methodology, transparency, and data usage.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In the Introduction section, we discuss the broad applicability and value of the research topic addressed in this paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper outlines the safeguards implemented for the responsible release of data and models in the experiment settings section , ensuring that they are protected against potential misuse.

    Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators and original owners of all assets (e.g., code, data, models) used in the paper are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide detailed descriptions of the code and experimental settings in the text.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.