LG-TRANSFORMER: LLM-GUIDED DISCOVERY OF BOOSTED NOVEL TRANSFORMER MODELS FOR TIME-SERIES MODELING

Anonymous Authors

ABSTRACT

Transformer-based models have achieved strong results in long-term time-series forecasting, yet their architectures are still largely designed manually, leaving a vast combinatorial design space underexplored. To address this issue, we present **LG-Transformer**, an LLM-guided neural architecture search framework that automatically discovers high-performing Transformer variants for time-series modeling. Our method combines a pretrained large language model controller with a weight-sharing supernet to enable context-aware architecture generation and efficient candidate evaluation. The search space spans key encoder design choices, including network depth, attention configuration, normalization placement, activation function, feed-forward expansion ratio, and residual operators. After search, the selected architecture is retrained from scratch, yielding a standard forecasting model with no additional inference-time overhead. Experiments on widely used multivariate forecasting benchmarks, including ETT, Electricity, Traffic, Weather, and ILI, show that the architectures discovered by LG-Transformer consistently outperform strong manually designed baselines and achieve highly competitive results across diverse prediction settings. These results demonstrate that LLM-guided architecture search is an effective and practical route to better time-series Transformers, while lowering the barrier for small industrial practitioners and startups to adopt large-model-driven solutions.

1 INTRODUCTION

Time-series forecasting is one of the fundamental problems in machine learning, with broad applications in energy systems, transportation, finance, healthcare, and climate-related services (Wen et al., 2022). Its practical importance has driven sustained interest in models that can capture long-range temporal dependencies, complex cross-variate interactions, and non-stationary dynamics (Zeng et al., 2023; Tang et al., 2025; Wu et al., 2022). In recent years, Transformer-based methods have become a dominant paradigm for long-term forecasting due to their strong representation capacity and flexible sequence modeling ability. A series of representative models has substantially advanced this line of research, including Informer (Zhou et al., 2021), Autoformer (Wu et al., 2021), FEDformer (Zhou et al., 2022), Non-stationary Transformer (Liu et al., 2022b), PatchTST (Nie et al., 2023), Pyraformer (Liu et al., 2022a), and iTransformer (Liu et al., 2023). These models demonstrate that carefully designed Transformer variants can achieve strong forecasting performance across diverse benchmarks. However, the success of these methods also highlights a broader question: given the large combinatorial design space of forecasting Transformers, can we move beyond manual architecture engineering and automatically discover architectures that are better matched to the structure of time-series data?

Despite the strong empirical progress of existing forecasting Transformers, most current architectures remain largely hand-crafted. In practice, design choices such as encoder depth, attention configuration, normalization placement, activation functions, and residual operators are typically determined by expert intuition and iterative trial-and-error, and are often modified in isolation rather than explored jointly (So et al., 2019). This development paradigm has two important limitations. First, it only covers a small subset of the full architecture space and may therefore miss high-performing configurations that do not conform to standard design templates. Second, the best architectural pattern can vary across datasets with different dimensionalities, temporal resolutions, and forecasting horizons, making fixed manually designed models inherently less adaptive. Neural architecture search (NAS)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

offers a principled alternative for exploring large design spaces automatically (Zoph et al., 2017; Pham et al., 2018; Dong and Yang, 2019; Xu et al., 2020; Liu et al., 2019; Yan et al., 2026), but prior NAS research has mainly focused on image models (Wang et al., 2021) or generic operators (Serianni and Kalita, 2023), with limited attention to Transformer-based time-series forecasting. Moreover, conventional NAS controllers are usually lightweight recurrent or differentiable modules (Zhu et al., 2021), which are less suited to modeling structured dependencies among heterogeneous architectural decisions. As a result, existing approaches still fall short of a search framework that is both expressive in generation and efficient in evaluation for forecasting Transformers.

To address these limitations, we propose **LG-Transformer**, an **L**LM-**G**uided architecture search framework for automatically discovering high-performing Transformer models for time-series forecasting. Our key idea is to cast architecture discovery as a reinforcement learning problem in which a controller learns a policy over a structured Transformer search space, while a weight-sharing supernet provides efficient performance estimates in the spirit of one-shot NAS (Pham et al., 2018). Different from conventional NAS controllers, we instantiate the policy with a pretrained large language model, and condition generation on the search space specification. This design allows the controller to produce valid architecture candidates in a context-aware manner and to progressively shift toward promising regions of the search space. The search space covers both macro- and micro-level design dimensions of the Transformer encoder, including depth, attention heads, scaling, normalization placement, activation functions, feed-forward expansion ratios, and residual-path operators. After search, the best candidates are retrained from scratch, so that the final deployed model is a standard forecasting architecture and incurs no additional inference-time cost.

We evaluate LG-Transformer on a diverse set of widely used multivariate forecasting benchmarks, including ETT, Electricity, Traffic, Weather, and ILI. Experimental results show that the architectures discovered by our method consistently outperform strong manually designed baselines such as PatchTST (Nie et al., 2023), Time-LLM (Jin et al., 2024), FEDformer (Zhou et al., 2022), Autoformer (Wu et al., 2021), DLinear (Zeng et al., 2023), Informer (Zhou et al., 2021), Pyraformer (Liu et al., 2022a), and LogTrans (Li et al., 2019). In particular, LG-Transformer achieves the best performance on the large majority of evaluated forecasting settings, demonstrating that automated architecture discovery can surpass expert-designed Transformer variants for time-series modeling. Further ablation studies validate the importance of the persistent shared-parameter pool, the from-scratch derive stage, and the LLM-based controller. Overall, this paper makes the following contributions:

- We introduce LG-Transformer, a new LLM-guided NAS framework for time-series forecasting that combines context-aware architecture generation with efficient weight-sharing evaluation.
- We design a structured yet expressive Transformer encoder search space that exposes multiple coupled architectural factors important to forecasting performance.
- We show through extensive experiments that the discovered architectures achieve state-of-the-art or highly competitive results across diverse forecasting benchmarks, while preserving standard inference-time efficiency.

2 Related Work

2.1 Time Series Forecasting with Transformers

Transformer-based models have become a dominant paradigm for long-term time series forecasting due to their strong capacity for modeling complex temporal dependencies (Wang et al., 2024). A substantial body of work has focused on adapting the original Transformer architecture to the statistical characteristics and computational challenges of temporal data. Informer (Zhou et al., 2021) improved scalability for long input sequences through ProbSparse attention, while Autoformer (Wu et al., 2021) incorporated series decomposition to explicitly model trend and seasonal components. FEDformer (Zhou et al., 2022) further explored frequency-domain representations to capture global temporal patterns more effectively, and Non-stationary Transformers (Liu et al., 2022b) addressed the distribution shifts and non-stationary behavior commonly observed in real-world time series. More recently, iTransformer (Liu et al., 2023) reformulated the tokenization perspective by treating variates as tokens, thereby emphasizing cross-variate dependency modeling rather than conventional attention over time steps.

Despite these advances, most existing Transformer variants for time series forecasting remain largely hand-crafted, with architectural choices determined by expert intuition and empirical trial-and-error. In practice, key design dimensions—such as attention formulations, normalization strategies, activation functions, and residual operators—are typically modified in isolation or combined within a limited set of manually designed patterns. Such a development pipeline makes it difficult to systematically explore the large combinatorial design space of forecasting architectures, and may therefore overlook model configurations that are better suited to different datasets, forecasting horizons, or variable regimes. This limitation motivates automated architecture discovery methods that can search more broadly and adaptively for effective Transformer designs in time series forecasting.

2.2 Neural Architecture Search

Neural Architecture Search (NAS) aims to automate architecture design by exploring large combinatorial model spaces under data-driven optimization objectives. Early NAS methods, such as NASNet (Zoph et al., 2017) and ENAS (Pham et al., 2018), showed that automatically discovered architectures can rival or surpass manually designed ones, while also highlighting the practical trade-off between search quality and computational cost. To improve search efficiency, differentiable and approximation-based methods were later introduced. In particular, DARTS (Liu et al., 2019) reformulated architecture search through continuous relaxation, enabling gradient-based optimization over discrete design choices, whereas GDAS (Dong and Yang, 2019) and PC-DARTS (Xu et al., 2020) further reduced the memory and optimization overhead associated with one-shot supernet training. Despite these advances, much of the NAS literature has focused on generic image backbones or standard neural operators, with comparatively limited attention to the architectural requirements of Transformer-based time-series forecasting.

3 Method

We present **LG-Transformer**, an LLM-guided neural architecture search framework for time-series forecasting. A prompt-conditioned LLM controller samples architectures from a structured encoder search space and is optimized using validation-based rewards. A weight-sharing supernet (Pham et al., 2018) enables efficient evaluation of sampled architectures during search. After search, the selected architecture is re-initialized and trained from scratch, while the controller and supernet are discarded. Figure 1 summarizes the overall pipeline.

3.1 Architecture Parameterization and Encoder Search Space

We represent each candidate Transformer encoder as a discrete architecture code

$$\theta = (D, \mathbf{z}, h, s, o_{\mathrm{attn}}, \sigma, \rho, o_{\mathrm{ffn}}) \in \mathcal{M}, \tag{1}$$

where $\mathcal{M}$ denotes the encoder search space. The parameterization combines a macro-level choice, encoder depth, with micro-level choices that determine the normalization, attention, feed-forward, and residual-path structure of the encoder block. A complete list of options is given in Table 4, with additional implementation details in Appendix B.1.

Depth and normalization. We search encoder depth $D \in \{1, 2, 3, 4, 6\}$ and seven binary LayerNorm switches $\mathbf{z} \in \{0, 1\}^7$, where each z_i enables or bypasses a predefined normalization site. This allows the controller to explore flexible normalization layouts beyond a fixed pre-norm or post-norm design. The same encoder block configuration is repeated for all D layers.

Attention and feed-forward design. Multi-head self-attention is parameterized by the number of heads $h \in \{2, 4, 8\}$ and a binary scaling switch $s \in \{0, 1\}$. When scaling is enabled, the attention logits use the standard dimension-dependent scaling before softmax; disabling it removes this scaling factor while leaving the remaining attention computation unchanged. The feed-forward network (FFN) is controlled by the activation $\sigma \in \{\mathrm{gelu}, \mathrm{swish}, \mathrm{relu}, \mathrm{leaky_relu}\}$ and the expansion ratio $\rho \in \{0.25, 0.5, 1, 2, 4, 8\}$. If the encoder embedding width is d_{model}, the FFN hidden width is determined by ρd_{model} (up to the implementation's integer rounding), so the search can vary both the nonlinearity and the capacity of the FFN.

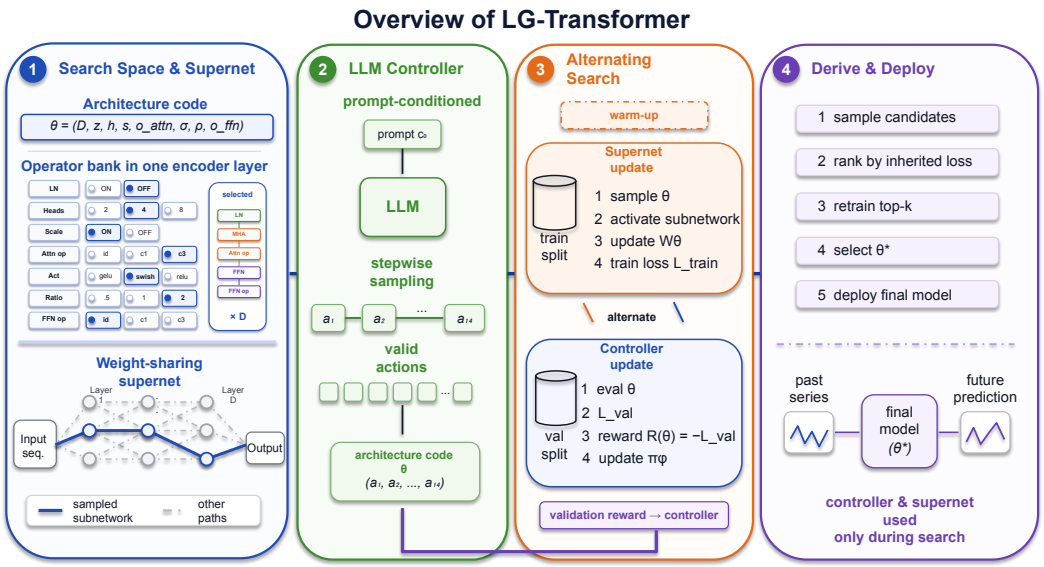


Figure 1: Overview of LG-Transformer. The framework combines a structured Transformer encoder search space with a weight-sharing supernet and a prompt-conditioned LLM controller. During search, supernet optimization on the training split alternates with controller updates driven by validation reward. The derived architecture is then retrained from scratch, while the controller and supernet are discarded at inference time. In the operator bank, cN denotes a temporal Conv1D operator with kernel size $N \in \{1, 3, 5\}$.

Residual-path operators. Standard Transformer blocks typically use identity residual connections around the self-attention and FFN sublayers. While this design facilitates optimization, it does not explicitly model short-range temporal patterns along the residual path. Because local dependencies are often important in time-series forecasting, we extend the search space to include lightweight temporal transformations on these paths. Specifically, we independently search the operators applied to the post-attention and post-FFN residual paths:

$$o_{\text{attn}}, o_{\text{ffn}} \in \{\text{Identity}, \text{Conv1D}(k{=}1), \text{Conv1D}(k{=}3), \text{Conv1D}(k{=}5)\}. \tag{2}$$

Identity introduces no additional transformation on the corresponding residual path and recovers the conventional residual formulation. The remaining choices belong to the same temporal-convolution family and differ only in kernel size. Specifically, $\text{Conv1D}(k)$ applies a one-dimensional convolution along the temporal axis with kernel size k. The $k = 1$ operator performs a point-wise transformation at each time step without mixing neighboring observations, whereas $k = 3$ and $k = 5$ incorporate progressively wider local temporal contexts. The post-attention and post-FFN operators are selected independently, allowing the architecture to retain an identity mapping in one sublayer while introducing local temporal processing in the other. For compactness, Figure 1 denotes the convolutional family as cN, where $N \in \{1, 3, 5\}$ indicates the kernel size.

Together, these choices specify a candidate encoder architecture. LG-Transformer instantiates it within the weight-sharing supernet by activating the corresponding operations and repeating the selected encoder block for D layers. The resulting search space contains approximately 2.95M valid architectures. This design covers diverse Transformer variants while retaining a structured discrete search space that can be generated sequentially by the LLM-guided controller.

3.2 LLM-Guided Controller

We formulate architecture generation as a sequential policy over the discrete search space $\mathcal{M}$. Each candidate architecture θ is represented by an action sequence $\mathbf{a}_{1:T}$ with $T = 14$, consisting of seven LayerNorm decisions and seven additional choices for depth, attention, FFN design, and residual-path

operators. Given a base prompt c_0, the controller factorizes the policy as

$$\pi_\phi(\theta \mid c_0) = \prod_{t=1}^{T} \pi_\phi(a_t \mid c_t), \tag{3}$$

where c_t contains the base prompt together with previously sampled decisions.

Rather than generating free-form architecture descriptions, we use a pretrained LLM as the backbone of a constrained discrete controller. At each step t, the prompt state c_t is encoded by the LLM, and the final-token hidden state $\mathbf{u}_t$ is passed to a step-specific classification head defined over the valid action set $\mathcal{A}_t$:

$$\pi_\phi(a_t \mid c_t) = \mathrm{softmax}(W_t \mathbf{u}_t + \mathbf{b}_t), \qquad a_t \in \mathcal{A}_t. \tag{4}$$

After sampling a_t, its textual representation is appended to the prompt to construct c_{t+1}. Using step-specific heads keeps each prediction aligned with the valid options of a single search dimension, while appending previous decisions to the prompt allows later actions to condition on earlier architectural choices. This autoregressive procedure continues until all 14 decisions are made, ensuring that each sampled sequence corresponds to a valid architecture in $\mathcal{M}$.

After sampling $\theta \sim \pi_\phi(\cdot \mid c_0)$, the corresponding subnetwork is evaluated using inherited supernet weights and assigned a validation-based reward $R(\theta)$. We optimize the controller with a policy-gradient objective using an exponential moving-average baseline b and entropy regularization:

$$\mathcal{L}_{\mathrm{ctrl}} = -(R(\theta) - b) \sum_{t=1}^{T} \log \pi_\phi(a_t \mid c_t) - \beta \sum_{t=1}^{T} \mathcal{H}[\pi_\phi(\cdot \mid c_t)], \tag{5}$$

where β controls exploration. This update increases the probability of architecture decisions associated with stronger validation performance while maintaining exploration during search. As an exploratory ablation, we additionally evaluate a group-relative policy-gradient variant, with further analysis provided in Appendix B.2; the moving-baseline objective above is used as the default throughout our main experiments.

3.3 Training and Inference Procedure

LG-Transformer follows a search-to-deployment pipeline consisting of supernet training, controller optimization, architecture derivation, and final deployment. Let $\mathcal{W}$ denote the supernet parameters and $\mathcal{W}_\theta \subseteq \mathcal{W}$ the weights activated by architecture $\theta \in \mathcal{M}$. Following one-shot NAS (Pham et al., 2018), sampled architectures are evaluated using inherited weights during search rather than being trained independently from scratch.

Supernet warm-up and search. The weight-sharing supernet contains all operations in $\mathcal{M}$, while each sampled architecture activates only its corresponding subset $\mathcal{W}_\theta$. Before controller optimization, we warm up the supernet with the controller fixed to obtain meaningful inherited weights. The warm-up stage prevents early controller rewards from being dominated by poorly trained shared weights, making inherited-weight comparisons more informative. Search then alternates between supernet and controller updates. In the supernet step, a sampled architecture is trained on the training split by updating $\mathcal{W}_\theta$. In the controller step, sampled architectures are evaluated on the validation split without additional child-model training, using

$$R(\theta) = -\mathcal{L}_{\mathrm{val}}(\theta; \mathcal{W}_\theta), \tag{6}$$

as the reward for the policy-gradient update described above.

Architecture screening and derivation. After search, inherited-weight validation performance is used to efficiently rank a pool of sampled architectures. Because shared-weight estimates may be biased by parameter co-adaptation, only a small set of top candidates is retained. These candidates are re-initialized and trained independently from scratch, and the final architecture is selected by

$$\theta^\star = \arg\min_{\theta \in \mathcal{C}} \mathcal{L}_{\mathrm{val}}(\theta; \widehat{\mathcal{W}}_\theta), \tag{7}$$

where $\mathcal{C}$ denotes the shortlisted candidate set and $\widehat{\mathcal{W}}_\theta$ the independently trained parameters.

Inference and deployment. At inference time, both the LLM controller and the weight-sharing supernet are discarded. Only the derived architecture $\theta^\star$ and its independently trained parameters $\widehat{\mathcal{W}}_{\theta^\star}$ are retained. Thus, LG-Transformer incurs additional computation only during architecture search, while test-time cost is determined solely by the selected forecasting model.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate LG-Transformer on eight widely used multivariate forecasting benchmarks covering diverse domains and temporal resolutions (Wu et al., 2022; Jin et al., 2024): ETTh1, ETTh2, ETTm1, ETTm2, Electricity, ILI, Traffic, and Weather. These datasets contain 7 to 862 variates, providing a broad testbed for studying architecture search under different dimensionalities and data regimes. We use chronological 70/10/20 train/validation/test splits following the protocol of MOMENT (Goswami et al., 2024). Dataset statistics are reported in Table 3 of Appendix B.

Search space. Each candidate encoder is specified by 14 discrete decisions, covering encoder depth, LayerNorm placement, attention heads, attention-score scaling, activation function, FFN expansion ratio, and post-attention/post-FFN residual-path operators. As summarized in Table 4, this design yields approximately 2.95M valid architectures while keeping the controller action space compact.

LLM controller. We use Qwen2.5-1.5B-Instruct as the backbone of the NAS controller. The controller does not generate free-form architecture descriptions; instead, it encodes a prompt containing the search-space specification and previous decisions. The final-token hidden state is fed to a step-specific linear head that predicts the next valid architectural choice. The controller makes 14 autoregressive decisions, appending each selected option to the prompt before the next step.

Search procedure. Each search epoch has two phases. In *Phase 1*, the controller samples one architecture, which is instantiated from the shared parameter pool and trained for one epoch; the updated active weights are then written back to the pool. Validation is checked every 50 optimization steps with early-stopping patience 25. In *Phase 2*, activated after a 4-epoch warm-up, the controller samples 16 architectures, implemented as eight controller-update steps with two architectures per step. These architectures are evaluated on the validation split using inherited weights without additional child-model training. The controller is updated with the policy-gradient objective defined in the Method section, using validation MSE as the reward signal. We run 25 search epochs for each search run.

Derive procedure. After search, we sample 256 architectures from the trained controller and rank them by validation MSE under inherited weights. The top 16 architectures are then re-initialized and trained from scratch for up to 200 epochs with early stopping. The architecture with the lowest validation MSE is selected as the final LG-Transformer model. For fair comparison, PatchTST is also trained under the same final-training protocol, but it is not included in the LG-Transformer candidate set.

Baselines. We compare LG-Transformer with strong forecasting baselines, including **PatchTST** (Nie et al., 2023), **Time-LLM** (Jin et al., 2024), **FEDformer** (Zhou et al., 2022), **Autoformer** (Wu et al., 2021), **DLinear** (Zeng et al., 2023), **Informer** (Zhou et al., 2021), **Pyraformer** (Liu et al., 2022a), and **LogTrans** (Li et al., 2019). These methods cover patch-based Transformers, LLM-based forecasting, frequency-domain models, decomposition-based models, linear baselines, and efficient attention variants.

Hyperparameters. Unless otherwise specified, all search runs use controller learning rate 3×10^{-5}, child-model learning rate 8×10^{-4}, and input length 512. The prediction horizons are $\{96, 192, 336, 720\}$ for ETT, Electricity, Traffic, and Weather, and $\{24, 36, 48, 60\}$ for ILI. We set $d_{\text{model}} = 16$ for the ETT datasets and $d_{\text{model}} = 128$ for the remaining datasets. Batch size is 16 for datasets with at most 21 variates, 8 for Electricity, and 4 for Traffic. All experiments use random seed 2024 and are conducted on a single NVIDIA A100 GPU with 48GB memory.

Evaluation metric. We report Mean Squared Error (MSE) and Mean Absolute Error (MAE) on the test set. Validation MSE is used for architecture derivation and checkpoint selection, while test metrics are used only for final comparison.

Table 1: Comparison of the strong baseline models and our methods on various datasets. For each row, the best MSE/MAE results are marked in bold, and the second-best MSE/MAE results are underlined.

Method		Ours		PatchTST		Time-LLM		DLinear		FEDformer		Autoformer		Informer		Pyraformer		LogTrans	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.146**	**0.194**	0.149	0.198	0.148	0.199	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405	0.896	0.556	0.458	0.490
	192	**0.190**	**0.237**	0.194	0.241	0.196	0.237	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434	0.622	0.624	0.658	0.589
	336	**0.241**	0.280	0.245	0.282	0.243	**0.276**	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543	0.739	0.753	0.797	0.652
	720	**0.309**	**0.327**	0.314	0.334	0.317	0.340	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705	1.0004	0.934	0.869	0.675
Traffic	96	**0.357**	**0.245**	0.360	0.249	0.363	0.251	0.410	0.282	0.576	0.359	0.597	0.371	0.733	0.410	2.085	0.468	0.684	0.384
	192	**0.377**	**0.255**	0.379	0.256	0.383	0.260	0.423	0.287	0.610	0.380	0.607	0.382	0.777	0.435	0.867	0.467	0.685	0.390
	336	**0.386**	**0.260**	0.392	0.264	0.396	0.271	0.436	0.296	0.608	0.375	0.623	0.387	0.776	0.434	0.869	0.469	0.734	0.408
	720	**0.429**	**0.282**	0.432	0.286	0.435	0.289	0.466	0.315	0.621	0.375	0.639	0.395	0.827	0.466	0.881	0.473	0.717	0.396
Electricity	96	**0.124**	**0.213**	0.129	0.222	0.132	0.228	0.140	0.237	0.186	0.302	0.196	0.313	0.304	0.393	0.386	0.449	0.258	0.357
	192	**0.145**	**0.234**	0.147	0.240	0.146	0.237	0.153	0.249	0.197	0.311	0.211	0.324	0.327	0.417	0.386	0.443	0.266	0.368
	336	**0.161**	**0.256**	0.163	0.259	0.168	0.266	0.169	0.267	0.213	0.328	0.214	0.327	0.333	0.422	0.378	0.443	0.280	0.380
	720	**0.195**	0.292	0.197	**0.290**	0.201	0.304	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427	0.376	0.445	0.283	0.376
ILI	24	**1.307**	**0.742**	1.319	0.754	1.328	0.753	2.215	1.081	2.614	1.095	2.906	1.182	4.657	1.449	1.420	2.012	4.480	1.444
	36	**1.563**	**0.861**	1.579	0.870	1.578	0.873	1.963	0.963	2.516	1.021	2.585	1.038	4.650	1.463	7.394	2.031	4.799	1.467
	48	1.546	0.808	1.553	0.815	**1.539**	**0.804**	2.130	1.024	2.505	1.041	3.024	1.145	5.004	1.542	7.551	2.057	4.800	1.468
	60	**1.463**	**0.781**	1.470	0.788	1.483	0.796	2.368	1.096	2.742	1.122	2.761	1.114	5.071	1.543	7.662	2.100	5.278	1.560
ETTh1	96	**0.367**	**0.397**	0.372	0.400	0.373	0.399	0.375	0.399	0.376	0.415	0.435	0.446	0.941	0.769	0.664	0.612	0.878	0.740
	192	0.406	0.421	0.413	0.429	0.416	0.433	**0.405**	**0.416**	0.423	0.446	0.456	0.457	1.007	0.786	0.790	0.681	1.037	0.824
	336	**0.415**	**0.432**	0.422	0.440	0.423	0.442	0.439	0.443	0.444	0.462	0.486	0.487	1.038	0.784	0.891	0.738	1.238	0.932
	720	**0.437**	**0.452**	0.447	0.468	0.448	0.471	0.472	0.490	0.469	0.492	0.515	0.517	1.144	0.857	0.963	0.782	1.135	0.852
ETTh2	96	**0.270**	**0.333**	0.274	0.337	0.276	0.341	0.289	0.353	0.332	0.374	0.332	0.368	1.549	0.952	0.645	0.597	2.116	1.197
	192	**0.337**	**0.375**	0.341	0.382	0.346	0.393	0.383	0.418	0.407	0.446	0.426	0.434	3.792	1.542	0.788	0.683	4.315	1.635
	336	0.328	0.384	0.333	0.391	**0.326**	**0.380**	0.448	0.465	0.400	0.447	0.477	0.479	4.215	1.642	0.907	0.747	1.124	1.604
	720	**0.373**	**0.417**	0.379	0.422	0.376	**0.417**	0.605	0.551	0.412	0.469	0.453	0.490	3.656	1.619	0.963	0.783	3.188	1.540
ETTm1	96	**0.287**	0.342	0.293	0.346	0.290	**0.341**	0.299	0.343	0.326	0.390	0.510	0.492	0.626	0.560	0.543	0.510	0.600	0.546
	192	**0.330**	0.368	0.333	0.370	0.336	0.375	0.335	**0.365**	0.365	0.415	0.514	0.495	0.725	0.619	0.557	0.537	0.837	0.700
	336	**0.364**	**0.386**	0.369	0.392	0.374	0.403	0.369	0.386	0.392	0.425	0.510	0.492	1.005	0.741	0.754	0.655	1.124	0.832
	720	**0.411**	**0.415**	0.416	0.420	0.419	0.426	0.425	0.421	0.446	0.458	0.527	0.493	1.133	0.845	0.908	0.724	1.153	0.820
ETTm2	96	**0.161**	**0.251**	0.166	0.256	0.171	0.267	0.167	0.260	0.180	0.271	0.205	0.293	0.355	0.462	0.435	0.507	0.768	0.642
	192	**0.221**	**0.295**	0.223	0.296	0.227	0.309	0.224	0.303	0.252	0.318	0.278	0.336	0.595	0.586	0.730	0.673	0.989	0.757
	336	**0.269**	**0.323**	0.274	0.329	0.276	0.332	0.281	0.342	0.324	0.364	0.343	0.379	1.270	0.871	1.201	0.845	1.334	0.872
	720	**0.354**	**0.377**	0.362	0.385	0.365	0.394	0.397	0.421	0.410	0.420	0.414	0.419	3.001	1.267	3.625	1.451	3.048	1.328

4.2 Main Results

Table 1 compares **LG-Transformer** with representative forecasting baselines, including PatchTST (Nie et al., 2023), Time-LLM (Jin et al., 2024), FEDformer (Zhou et al., 2022), Autoformer (Wu et al., 2021), DLinear (Zeng et al., 2023), Informer (Zhou et al., 2021), Pyraformer (Liu et al., 2022a), and LogTrans (Li et al., 2019). Across the eight benchmarks and four prediction horizons per benchmark, LG-Transformer obtains the best MSE in 29 out of 32 settings. The MAE results show the same overall trend, with LG-Transformer either achieving the best value or remaining very close to the best competing method in most cases. These results indicate that the architectures discovered by the proposed search procedure provide consistently strong forecasting performance across heterogeneous datasets.

Compared with the strongest non-search Transformer baseline, PatchTST (Nie et al., 2023), LG-Transformer improves MSE on all reported settings in Table 1. The gains are consistent on Weather, Traffic, Electricity, ETTm1, and ETTm2, where LG-Transformer ranks first for every prediction horizon. For example, on Weather, MSE decreases from 0.149/0.194/0.245/0.314 to 0.146/0.190/0.241/0.309 for horizons 96/192/336/720, respectively; on Traffic, it decreases from 0.360/0.379/0.392/0.432 to 0.357/0.377/0.386/0.429. Similar improvements on Electricity and the ETTm benchmarks suggest that the searched encoder configurations transfer well across different variable counts and temporal resolutions.

The few non-best cases are limited and informative. On ETTh1, DLinear (Zeng et al., 2023) is marginally better at horizon 192, with MSE 0.405 versus 0.406 for LG-Transformer. On ETTh2, Time-LLM (Jin et al., 2024) obtains the best MSE at horizon 336, while LG-Transformer remains second. On ILI, Time-LLM is slightly better at horizon 48, whereas LG-Transformer is best at the other three horizons. Overall, Table 1 supports our central claim that automatically searched Transformer architectures can outperform carefully designed forecasting models. Since the LLM controller and weight-sharing supernet are used only during search, these gains do not introduce additional inference-time overhead beyond the final derived forecasting model.

4.3 Ablation Study

Framework ablations. We ablate three core design choices of **LG-Transformer** on ETTh1 and ETTm1 with prediction horizon 96; results are shown in the upper block of Table 2. The full model achieves the best overall performance across the two datasets. **LG-Transformer-1** resets

the shared-weight pool at each controller-training epoch, leading to consistently worse MSE and MAE. This indicates that persistent shared weights provide more stable inherited-weight estimates and more informative feedback throughout search. **LG-Transformer-2** initializes final training from inherited supernet weights instead of retraining the derived architecture from scratch. Its weaker performance suggests that shared weights are effective for candidate screening but may introduce co-adaptation bias when transferred directly to stand-alone training. **LG-Transformer-3** replaces the MSE-based controller reward with an MAE-based reward. Although this variant remains competitive, it is slightly worse overall, supporting MSE as the default search objective. Together, these results support the three-stage design of LG-Transformer: persistent shared-weight evaluation during search, validation-driven controller optimization, and independent from-scratch training for final architecture selection.

Table 2: Ablation results of LG-Transformer with prediction horizon 96. The upper block studies key framework components, the middle block compares controller architectures, and the final block compares controller objectives. Bold values indicate the best result within each comparison block.

Method	ETTh1		ETTm1	
	MSE	MAE	MSE	MAE
LG-Transformer	**0.367**	**0.397**	**0.287**	**0.342**
LG-Transformer-1	0.378	0.409	0.297	0.351
LG-Transformer-2	0.373	0.402	0.291	0.344
LG-Transformer-3	0.367	0.398	0.289	0.343
LSTM	0.371	0.399	0.293	0.345
Transformer	0.374	0.404	0.295	0.351
GPT-2	0.371	0.399	0.292	0.346
Qwen-2.5 1.5B Instruct	0.367	0.397	0.287	0.342
Qwen-2.5 1.5B Base	0.369	0.400	0.290	0.344
Qwen-2.5 7B Instruct	0.366	0.397	0.288	0.344
Qwen-2.5 72B Instruct	**0.366**	**0.395**	**0.286**	**0.340**
Qwen-2.5 1.5B + REINFORCE	**0.367**	**0.397**	**0.287**	**0.342**
Qwen-2.5 1.5B + GRPO	0.383	0.412	0.297	0.352

Controller architectures. We further study the effect of the controller backbone by replacing the default **Qwen2.5-1.5B-Instruct** model with an LSTM, a randomly initialized Transformer decoder, GPT-2, Qwen2.5-1.5B-Base, Qwen2.5-7B-Instruct, and Qwen2.5-72B-Instruct, while keeping the remaining search and derivation procedure unchanged. As shown in the middle block of Table 2, pretrained language-model controllers generally outperform lightweight or randomly initialized alternatives. Instruction tuning improves the 1.5B controller over its base counterpart, whereas increasing the model size provides only small and non-monotonic gains. We therefore use Qwen2.5-1.5B-Instruct as a practical trade-off between search quality and computational cost.

Controller objectives. We additionally compare REINFORCE and GRPO using the same Qwen2.5-1.5B-Instruct controller and equal controller sample budgets. GRPO normalizes validation rewards within groups of four architectures, whereas REINFORCE uses a moving reward baseline. On ETTh1, REINFORCE achieves lower MSE/MAE than GRPO (0.367/0.397 versus 0.383/0.412). REINFORCE also performs better on ETTm1 (0.287/0.342 versus 0.297/0.352). Under this evaluation, GRPO does not improve forecasting performance over the default moving-baseline objective, so we retain REINFORCE as the default controller optimization strategy.

Input-window robustness. We evaluate LG-Transformer with input lengths $L \in \{32, 64, 96, 128, 256, 512\}$ on ETTh1 and ETTm1 at prediction horizon 96. Performance generally improves as more historical context is provided, with the gains gradually saturating at longer input windows. Across all evaluated input lengths, LG-Transformer consistently outperforms PatchTST, indicating that its advantage is not tied to a particular look-back setting. This also suggests that the discovered architectures remain effective as the amount of available historical context varies substantially. Detailed results are shown in Figure 3.

4.3.1 Case Studies and Visualizations

To complement the aggregate comparisons in Table 1, Figure 2 presents a controlled qualitative replay across all four ETT datasets. Both models use an input window of 512 and predict the next 192 points, with only the final 96 input points shown for visualization. We use one fixed LG-Transformer architecture and one fixed PatchTST configuration across datasets, with checkpoint selection based only on validation performance. Therefore, this experiment is intended as a controlled cross-dataset case study rather than a reconstruction of the dataset-specific checkpoints used in Table 1.

Displayed OT-channel samples are selected using a model-independent criterion: target curvature closest to the dataset-specific 75th percentile. LG-Transformer achieves lower sample-level MSE on ETTh1 (0.0304 vs. 0.0364), ETTh2 (0.2123 vs. 0.2579), and ETTm2 (0.0319 vs. 0.0361), while PatchTST performs better on the selected ETTm1 case (0.0113 vs. 0.0156). The examples therefore illustrate complementary behavior rather than uniform pointwise dominance. Both models track the smoother oscillatory pattern in ETTm2 relatively well, whereas the sharper and higher-frequency variations in ETTh1, ETTh2, and ETTm1 are more challenging and tend to be partially smoothed. These cases complement, rather than replace, the aggregate test metrics.

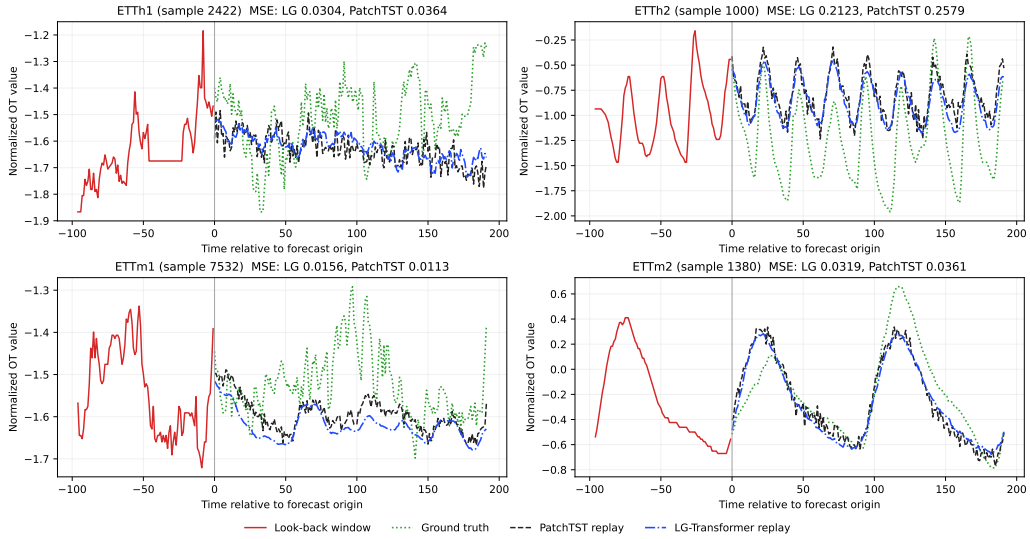


Figure 2: Controlled replay examples on ETTh1, ETTh2, ETTm1, and ETTm2 with look-back length 512 and prediction horizon 192. Only the final 96 input points are shown. Samples are selected independently of model error using the OT-channel target curvature closest to the dataset-specific 75th percentile. Titles report sample-level MSE.

5 Conclusion

We presented **LG-Transformer**, an LLM-guided neural architecture search framework for time-series forecasting that combines a pretrained LLM controller with a weight-sharing supernet to discover effective Transformer encoder architectures. Across eight multivariate forecasting benchmarks, LG-Transformer achieves the lowest MSE in 29 of 32 settings, while ablation studies validate the key design choices in the search and derivation pipeline. Since the controller and supernet are used only during search and discarded after architecture derivation, the final forecasting model introduces no additional inference-time overhead. These results demonstrate the potential of LLM-guided architecture search for automated time-series model design and motivate future exploration of broader search spaces and more challenging forecasting settings.

AI USE STATEMENT

Generative AI tools were used for formatting assistance and for drafting a schematic framework figure. The authors remain responsible for, and have reviewed, the scientific content, claims, citations, experiments, and final presentation.

REFERENCES

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

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, 2024. URL https://arxiv.org/abs/2310.01728.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

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. In *International Conference on Learning Representations*, 2022a.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35: 9881–9893, 2022b.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pages 4095–4104, 2018.

Aaron Serianni and Jugal Kalita. Training-free neural architecture search for rnns and transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2522–2540, 2023.

David So, Quoc Le, and Chen Liang. The evolved transformer. In *International conference on machine learning*, pages 5877–5886. PMLR, 2019.

Huanling Tang, Yulin Wang, Yu Zhang, Quansheng Dou, and Mingyu Lu. Ts-mixer: A lightweight text representation model based on context awareness. *Expert Systems*, 42(2):e13732, 2025.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. *arXiv preprint arXiv:2108.04392*, 2021.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Chen Wang, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

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*, 34:22419–22430, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020.

Yu Yan, Liping Mo, and Yangbin Tan. Neural architecture search from a natural language processing perspective: a survey. *Artificial Intelligence Review*, 2026.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11121–11128, 2023.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

Wei Zhu, Xiaoling Wang, Yuan Ni, and Guotong Xie. Autotrans: Automating transformer design via reinforced architecture search. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 169–182. Springer, 2021.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2017.

A Broader impact

LG-Transformer automates the design of Transformer encoders for time-series forecasting. Its main potential benefit is improved predictive accuracy in applications where forecasts support planning and resource allocation, such as energy management, transportation, weather-related services, and industrial operations. Because the LLM controller and the weight-sharing supernet are used only during search, the final derived model is a standard forecasting network and does not introduce LLM-related inference overhead.

The method may also inherit and amplify risks common to data-driven forecasting systems. In high-stakes settings, inaccurate forecasts can affect operational decisions and lead to financial loss, service degradation, or inequitable allocation of resources. These risks can be exacerbated when historical records are noisy, incomplete, or biased. Moreover, optimizing architectures on validation data does not establish robustness under distribution shifts, rare events, or abrupt non-stationarity. Deployment should therefore include domain-specific validation, stress testing, calibration analysis, and appropriate human oversight.

There are computational and environmental costs. Weight sharing, following the one-shot NAS paradigm (Pham et al., 2018), reduces the cost of evaluating candidate architectures, but the search stage is still more expensive than training a single manually specified model. The use of a pretrained LLM controller further increases search-time resource use. This cost is most justifiable when the resulting architecture can be reused across repeated deployments, related datasets, or settings where forecasting gains have clear operational value.

Table 3: Dataset statistics.

Dataset	Variates	Rows	Frequency	Domain
ETTh1	7	17,420	Hourly	Electricity transformer
ETTh2	7	17,420	Hourly	Electricity transformer
ETTm1	7	69,680	15-min	Electricity transformer
ETTm2	7	69,680	15-min	Electricity transformer
Electricity	321	26,304	Hourly	Electricity consumption
ILI	7	966	Weekly	National illness
Traffic	862	17,544	Hourly	Highway traffic flow
Weather	21	52,696	10-min	Meteorological

Improved forecasting capability is application-dependent and can be used beneficially or harmfully. For example, it may help optimize public infrastructure, but it could also support surveillance, market manipulation, or exclusionary automated decisions if integrated into opaque systems. Practitioners should document the intended use case, data provenance, evaluation protocol, and known failure modes, especially when forecasts trigger downstream automated actions.

Overall, **LG-Transformer** is intended as a tool for automated model design rather than an autonomous decision-making system. Responsible use requires careful dataset governance, privacy-aware data handling, transparent reporting of search and evaluation settings, and independent validation in the target environment, particularly when forecasts may affect safety-critical or socially consequential decisions.

B Appendix: Experiments

B.1 Datasets and Search Space

Datasets. The dataset statistics are presented in Table 3.

Architecture search space. Table 4 lists the complete set of discrete choices used by the controller. Each candidate is represented by 14 decisions: seven binary LayerNorm switches and one decision for each of the remaining seven parameter groups (attention heads, attention scaling, attention residual-path operator, activation function, FFN expansion ratio, FFN residual-path operator, and encoder depth).

The seven LayerNorm decisions correspond to predefined normalization sites in the encoder template and are independently enabled or bypassed. These sites span different locations around the self-attention and FFN sublayers, allowing the controller to explore flexible normalization layouts beyond a fixed pre-norm or post-norm design. The attention-scaling switch controls whether the attention logits are scaled by $1/\sqrt{d_k}$ before softmax. For the FFN, the expansion ratio ρ sets the hidden width relative to the encoder embedding width, i.e., $d_{\text{ff}} \approx \rho d_{\text{model}}$, while σ selects the activation function.

For both the self-attention and FFN residual branches, **Identity** introduces no additional temporal transformation. The remaining alternatives are one-dimensional temporal convolutions, written as $\mathrm{Conv1D}(k)$, with $k \in \{1, 3, 5\}$. Here, $k = 1$ applies a point-wise transformation independently at each time step, while $k = 3$ and $k = 5$ additionally incorporate increasingly wider local temporal context. Thus, these choices correspond to different receptive-field settings within the same Conv1D operator family. Figure 1 uses the compact labels `c1`, `c3`, and `c5` for $k = 1, 3, 5$. The attention and FFN residual-path operators are selected independently, allowing different local transformations on the two residual branches. The same encoder block configuration is repeated across the D layers specified by the depth decision.

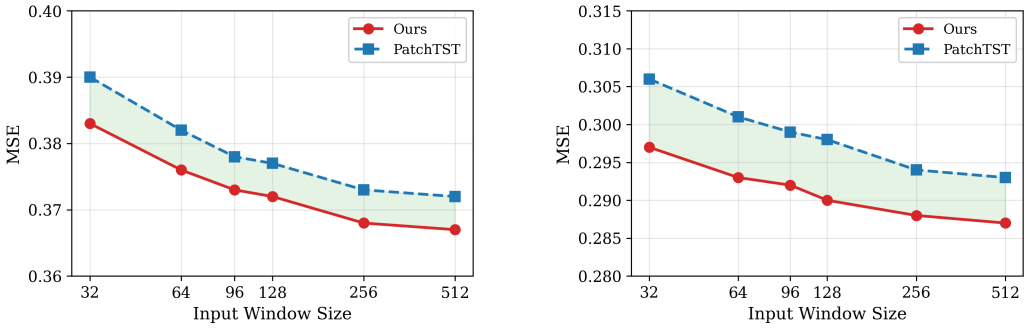
B.2 Additional Experimental Results

Input-window robustness. We further evaluate LG-Transformer under varying input window sizes on ETTh1 and ETTm1 with prediction horizon 96. As shown in Figure 3, performance generally improves as more historical context is provided, with diminishing gains at longer input windows.

Table 4: Architecture search space. Each candidate architecture is specified by 14 decisions across 8 parameter groups.

Parameter	Options	# Choices
LayerNorm positions (7 slots)	{True, False} each	$2^7 = 128$
Attention heads	{2, 4, 8}	3
Attention scaling	{True, False}	2
Attention residual-path op	Identity; Conv1D ($k = 1, 3, 5$)	4
Activation function	{gelu, swish, relu, leaky_relu}	4
FFN expansion ratio	{0.25, 0.5, 1, 2, 4, 8}	6
FFN residual-path op	Identity; Conv1D ($k = 1, 3, 5$)	4
Encoder depth	{1, 2, 3, 4, 6}	5
Total		≈2.95M

LG-Transformer consistently outperforms PatchTST across all evaluated window sizes, indicating that its advantage is not tied to a particular look-back setting.



(a) ETTh1 (b) ETTm1

Figure 3: Performance comparison under varying input window sizes with prediction horizon 96.

Controller-objective analysis. We additionally study the optimization objective used for the LLM-guided controller. Besides the default moving-baseline REINFORCE update, we evaluate a group-relative policy-gradient variant that normalizes rewards within groups of four sampled architectures. The controller backbone, search space, sampling budget, supernet training procedure, and architecture derivation protocol are otherwise kept unchanged. As reported in Table 2, GRPO does not improve over the default REINFORCE objective under our current evaluation.

Additional qualitative analysis. We provide an additional ETTh1 replay example to complement the matched cross-dataset case studies in the main text. Figure 4 shows a within-model diagnostic from the independently replayed $H = 96$ pipeline. The example is selected using a fixed criterion based on target curvature and prediction error rather than visual inspection.

The replayed LG-Transformer captures the overall level and the broad downward trajectory near the end of the forecast horizon, with sample-level MSE 0.0300 and MAE 0.1466. However, it smooths several short-lived fluctuations and underestimates some sharp intermediate excursions. This example highlights a remaining failure mode: the searched model can preserve coarse trajectory structure while attenuating higher-frequency local variation.

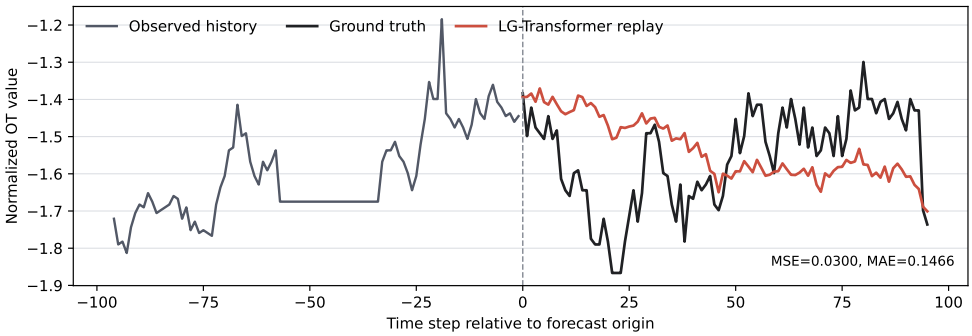


Figure 4: Independent ETTh1/96 replay example on the normalized OT channel. The final 96 points of the 512-step input window are shown before the forecast origin. This is a within-model diagnostic; a matched PatchTST prediction was not available for this replay sample.