TiNAS: Adapting Time Series Forecasting Architectures with Language Models

Anonymous authors
Paper under double-blind review

Abstract

The architecture best suited to a forecasting task can vary with its temporal dynamics, variable dependencies, and forecast horizon. Nevertheless, most forecasting models adapt to new tasks by optimizing parameters within a fixed architecture, leaving their internal structure unchanged. We introduce TiNAS, a task-feedback-driven framework that extends forecasting adaptation from parameter estimation to architecture optimization. It preserves the PatchTST representation pathway while searching its encoder at a fine granularity, including normalization placement, attention configuration, local temporal operators, feed-forward design, and depth. An autoregressive LLM controller generates these decisions sequentially by conditioning on the forecasting task, preceding architectural choices, and observed performance, and learns its search policy from validation MSE rewards. A weight-sharing supernet and progressive architecture derivation strategy enable efficient exploration, after which the highest-ranked architectures are independently retrained from scratch and selected by validation performance. Across eight forecasting benchmarks, TiNAS achieves the lowest overall MSE and MAE and improves the average MSE of PatchTST on six. Controlled ablations further confirm the complementary roles of the search space, controller, and search strategy. These results show that task-specific structural adaptation can substantially extend the capability of an established forecasting backbone.

1 Introduction

Time series forecasting has benefited from architectural advances that encode distinct inductive biases for temporal dynamics, long range dependencies, and cross variable relations (Wu et al., 2021; Zhou et al., 2022; Liu et al., 2022c; Wu et al., 2023; Nie et al., 2023; Liu et al., 2024), highlighting model structure as an important factor in forecasting performance. Most forecasters, however, use a manually specified architecture across tasks and adapt only its parameters, even though forecasting settings differ in temporal patterns, variable interactions, and forecast horizons. Neural architecture search extends task adaptation to model structure by optimizing operators, topology, and model capacity (Zoph & Le, 2017; Pham et al., 2018). Recent studies have introduced this paradigm into time series forecasting (Deng & Lindauer, 2025; Jia, 2026; Cao et al., 2026), establishing task specific architecture discovery as a natural direction beyond fixed model design.

We keep the macro architecture of PatchTST fixed and search only within its Transformer encoder. Specifically, we randomly construct eight candidates (A1–A8) by varying fine-grained encoder choices, including normalization and attention, while retaining the same patch embedding and prediction head. Across the three forecasting tasks in Figure 1, both the best performing architecture and the ranking of all candidates change with the task. These ranking shifts reveal task specific architecture preferences within a shared forecasting backbone and motivate fine-grained search over encoder design. We use an autoregressive LLM controller to condition architectural choices on task attributes, architecture descriptions, and previous evaluation outcomes. Rather than using validation MSE only to rank completed candidates, our search feeds it back as a reward after each evaluation to update the controller's sampling policy.

TiNAS searches task specific Transformer encoders within the fixed PatchTST forecasting backbone with two convolutional branches. It retains the patch embedding, channel-independent modeling, and prediction head, while searching normalization, attention, and other fine-grained encoder choices.

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.

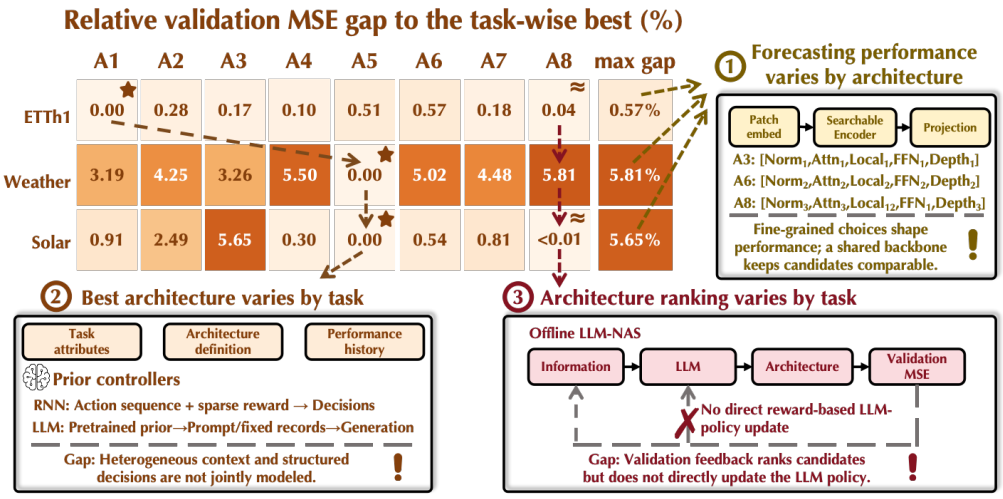

Figure 1: Relative validation MSE gaps of eight candidate architectures across forecasting tasks. Each value indicates the percentage increase over the lowest validation MSE among the eight candidates for the corresponding task.

The autoregressive LLM controller treats an architecture as a sequence of dependent decisions and conditions each decision on task attributes, architecture descriptions, and accumulated performance history. Each sampled subnet inherits weights from a shared supernet, and its validation MSE is converted into a reward that updates the controller's sampling policy. TiNAS periodically carries high reward candidates into full validation and retrains the strongest architectures from scratch.

Our main contributions are summarized as follows:

- We develop a fine grained encoder search space that preserves the PatchTST representation and explore it with an autoregressive LLM controller that integrates task and search context, using validation MSE to directly optimize its sampling policy.
- We introduce TiNAS, a unified system that extends PatchTST from a fixed forecasting architecture to a task adaptive model family by closing the loop between architecture generation, shared weight evaluation, validation feedback, and progressive candidate verification.
- Across eight benchmarks, TiNAS achieves the top two average MSE and MAE on four of them and lowers PatchTST's average MSE on six, demonstrating that task specific architecture adaptation can advance an established forecasting backbone without replacing its representation paradigm.

2 Related Work

Transformers for Time Series Forecasting Transformer forecasters encode temporal structure through efficient attention, multiscale or frequency modeling, and alternative tokenizations (Zhou et al., 2021; Liu et al., 2022b; Woo et al., 2022; Shabani et al., 2023; Chen et al., 2024; Zhang & Yan, 2023). PatchTST forms channel independent patch tokens, whereas iTransformer treats variables as tokens (Nie et al., 2023; Liu et al., 2024). These designs show that architectural bias is central to temporal modeling, yet each retains a fixed structure across tasks. This motivates adapting the PatchTST encoder at the task level while preserving its patch representation.

Neural Architecture Search Classical NAS already uses autoregressive controllers, validation rewards, weight sharing, and differentiable relaxation (Zoph & Le, 2017; Pham et al., 2018; Liu et al., 2019). Forecasting NAS assembles temporal modules or searches Transformer components with conventional optimizers (Chen et al., 2021; Deng & Lindauer, 2025; Jia, 2026; Zhang et al., 2025; Cao et al., 2026). LLM-guided NAS instead employs pretrained models for iterative generation, code mutation, reflection, or offline architecture ranking (Zheng et al., 2023; Chen et al., 2023; Nasir et al., 2024; Ji et al., 2025; Li et al., 2025; Hu et al., 2025). These methods generally do not train an

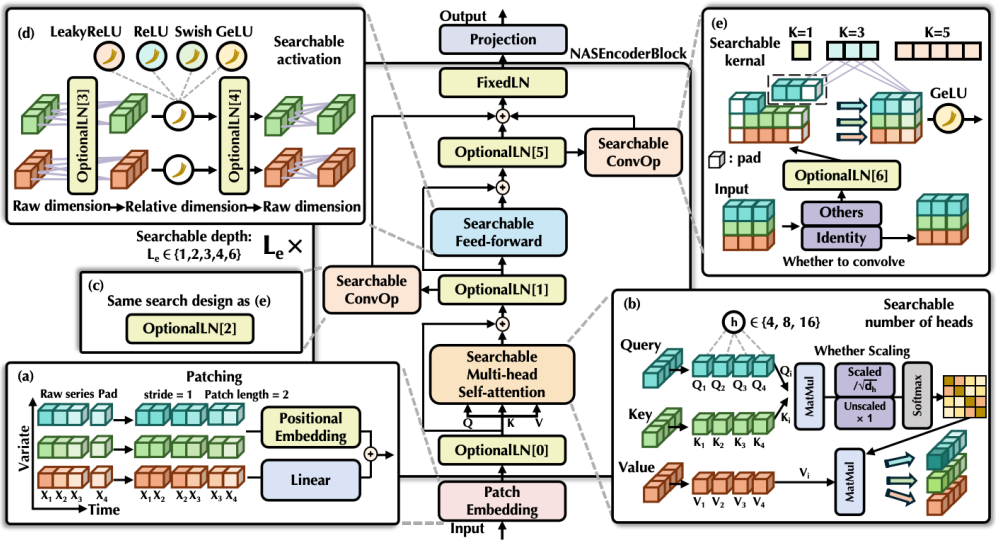

Figure 2: **Fine-grained encoder search space of TINAS.** Patch embedding and prediction remain fixed, while attention, feed-forward computation, local temporal branches, normalization placement, and depth are searchable. OptionalLN[$j-1$] corresponds to u_j.

autoregressive LLM as a stochastic policy using validation rewards from the current forecasting task. Moreover, shared-weight rankings can be unreliable (Chu et al., 2021; You et al., 2020). This leaves open how to combine a discrete search space built around PatchTST, online LLM policy learning, and progressive candidate validation within one forecasting NAS framework.

3 METHOD

Given a multivariate time series $\mathbf{X} \in \mathbb{R}^{L\times C}$ with context length L and C variables, the task is to predict $\mathbf{Y} \in \mathbb{R}^{H\times C}$ over a horizon of length H.

3.1 FINE-GRAINED ARCHITECTURE SEARCH SPACE

TINAS preserves the channel-independent patch embedding and prediction head of PatchTST while searching its encoder. Each variable is represented by N temporal patch tokens of dimension D. An architecture a in the search space $\mathcal{A}$ comprises fourteen decisions in five groups:

$$a = (\boldsymbol{u}, h, s, k_{\text{att}}, g, r, k_{\text{ffn}}, L_e), \qquad \boldsymbol{u} = (u_1, \ldots, u_7). \tag{1}$$

The groups cover attention configuration, feed-forward design, local temporal operators, normalization placement, and encoder depth. Figure 2 summarizes their positions; Appendix A.1 provides the complete specification and the remaining implementation details.

Attention and feed-forward choices Attention searches the head count $h \in \{4, 8, 16\}$, restricted to divisors of D, and a scaling indicator $s \in \{0, 1\}$. With head dimension $d_h = D/h$, attention logits use the multiplier $d_h^{-s/2}$: $s = 1$ applies standard scaling and $s = 0$ disables it. The FFN searches the width ratio $r \in \{0.25, 0.5, 1, 2, 4\}$ and activation $g \in \{\text{GELU}, \text{Swish}, \text{ReLU}, \text{LeakyReLU}\}$. Its hidden width is $D_r = \text{round}(rD)$.

Searchable local temporal operators Two auxiliary branches operate on the attention and FFN residual states, respectively. Their independent choices $k_{\text{att}}, k_{\text{ffn}} \in \{0, 1, 3, 5\}$ select

$$\mathcal{C}_k(\mathbf{Z}) = \begin{cases} \mathbf{0}, & k = 0, \\ \gamma_k \,\text{Dropout}(\text{GELU}(\text{DWConv}_k(\mathbf{Z}))), & k \in \{1, 3, 5\}, \end{cases} \tag{2}$$

where depthwise convolution acts along the patch dimension and γ_k is a learnable scale. Choosing $k = 0$ removes the branch.

Normalization placement and block composition Seven bits $\boldsymbol{u} \in \{0,1\}^7$ control optional LayerNorm. Bits u_1, u_2, u_3 act before attention, after its residual, and inside its local branch, respectively. Bits u_4, u_5 act after the first FFN projection and after activation and dropout. Bits u_6, u_7 act after the FFN residual and inside its local branch. Let $\mathcal{N}_{\ell,j}^{(u_j)}$ apply LayerNorm when $u_j = 1$ and identity otherwise.

For block input $\mathbf{Z}_{\ell-1} \in \mathbb{R}^{N\times D}$, $\mathcal{U}_\ell$ denotes the attention residual mapping controlled by u_1, u_2. The FFN residual mapping $\mathcal{F}_\ell$ is controlled by u_4, u_5, u_6. Their composition is

$$\begin{aligned} \mathbf{U}_\ell &= \mathcal{U}_\ell(\mathbf{Z}_{\ell-1}), \qquad \mathbf{V}_\ell = \mathcal{F}_\ell(\mathbf{U}_\ell), \\ \mathbf{T}_\ell^{\text{att}} &= \mathcal{C}_{k_{\text{att}}}\left(\mathcal{N}_{\ell,3}^{(u_3)}(\mathbf{U}_\ell)\right), \qquad \mathbf{T}_\ell^{\text{ffn}} = \mathcal{C}_{k_{\text{ffn}}}\left(\mathcal{N}_{\ell,7}^{(u_7)}(\mathbf{V}_\ell)\right), \\ \mathbf{Z}_\ell &= \text{LN}_\ell^{\text{out}}\left(\mathbf{V}_\ell + \mathbf{T}_\ell^{\text{att}} + \mathbf{T}_\ell^{\text{ffn}}\right). \end{aligned} \tag{3}$$

Positions $j = 4, 5$ normalize the FFN hidden dimension D_r and use separate LayerNorm parameters for each candidate width; the remaining positions normalize the feature dimension D. The attention and FFN residual mappings are

$$\begin{aligned} \mathcal{U}_\ell(\mathbf{Z}) &= \mathcal{N}_{\ell,2}^{(u_2)}\Big(\mathbf{Z} + \mathcal{D}\Big(\mathcal{A}_{h,s}(\mathcal{N}_{\ell,1}^{(u_1)}(\mathbf{Z}))\Big)\Big), \\ \mathbf{E}_\ell &= \mathcal{N}_{\ell,4}^{(u_4)}\Big(\mathbf{U}\mathbf{W}_{1,\ell}^{(r)} + \mathbf{b}_{1,\ell}^{(r)}\Big), \\ \mathbf{G}_\ell &= \mathcal{N}_{\ell,5}^{(u_5)}(\mathcal{D}(g(\mathbf{E}_\ell))), \\ \mathcal{F}_\ell(\mathbf{U}) &= \mathcal{N}_{\ell,6}^{(u_6)}\Big(\mathbf{U} + \mathcal{D}\Big(\mathbf{G}_\ell\mathbf{W}_{2,\ell}^{(r)} + \mathbf{b}_{2,\ell}\Big)\Big). \end{aligned} \tag{4}$$

The attention-side local branch does not enter the FFN; both branches are fused only at the block output. The terminal LayerNorm remains fixed and is excluded from $\boldsymbol{u}$. Appendix A.1 gives the exact conditional normalization and the remaining implementation details.

Depth and weight sharing Depth is selected from $L_e \in \{1, 2, 3, 4, 6\}$, activating the first L_e blocks of the six-block supernet. Active blocks share the architecture decisions but retain distinct parameters. Head configurations share query, key, value, and output projections. FFN widths activate prefix slices of projections stored at the maximum width $4D$. Let ϕ denote the supernet parameters and $\phi_a \subseteq \phi$ the inherited parameters activated by architecture a.

3.2 Autoregressive LLM-Based Architecture Controller

We parameterize the controller by ψ as an autoregressive policy over the fourteen fields of Eq. (1), with one action per field. A separate action head scores each field, normalizing probabilities over its candidate values. The action a_t is drawn from

$$\begin{aligned} \mathbf{z}_t &= \mathbf{W}_t\mathbf{h}_t + \mathbf{b}_t, \qquad \pi_\psi(a_t \mid q, a_{<t}) = \text{Softmax}(c\tanh(\mathbf{z}_t/\tau)), \\ \mathbf{h}_t &= \text{LLM}_\psi(q \oplus \text{text}(a_{<t})), \end{aligned} \tag{5}$$

where q describes the task, legal values per field, and recent verification outcomes. The autoregressive policy models dependencies among architectural decisions by conditioning each choice on preceding fields. The pretrained backbone encodes task and architecture descriptions alongside validation feedback, providing a shared representation for the field-specific action heads. The backbone weights remain frozen, while LoRA adapters on its attention projections are optimized. The bound c limits the transformed logits, and τ controls their scale. Appendix A.2 reports the hyperparameters and prompt template, and Table 2 evaluates the controller components.

3.3 Architecture Search Procedure

Search comprises supernet warm-up, alternating supernet–controller optimization, and progressive derivation. Training data optimize the supernet and retrain finalists; validation data train the controller, rank architectures, and determine early stopping. The test set is used only once after selecting the final architecture and checkpoint.

Supernet warm-up For each training batch, we uniformly sample $a \in \mathcal{A}$ and update only ϕ_a, while keeping the controller fixed. This exposes subnetworks of different widths, operations, and depths to training before policy-guided sampling.

Alternating optimization Each search epoch alternates between supernet and controller updates. With ψ fixed, we construct a diverse policy-sampled architecture pool, supplement it with uniform samples when needed, and optimize activated parameters on training batches. We then freeze ϕ and evaluate newly sampled architectures on held-out validation batches.

For sampled architecture a_m, its reward and advantage are

$$R_m = -\mathcal{L}_{\text{val}}(\phi_{a_m}, a_m), \qquad A_m = R_m - b, \tag{6}$$

where b is an exponential moving average of previous rewards. For M samples, the controller objective is

$$\mathcal{L}_{\text{ctrl}} = -\frac{1}{MT} \sum_{m=1}^{M} \sum_{t=1}^{T} \left[A_m \log \pi_\psi(a_{m,t} \mid q, a_{m,<t}) + \lambda \mathcal{H}(\pi_\psi(\cdot \mid q, a_{m,<t})) \right], \tag{7}$$

where $\lambda \geq 0$ controls entropy-based exploration.

After each evaluation, we update

$$b \leftarrow \beta b + (1-\beta) R_m, \qquad 0 \leq \beta < 1, \tag{8}$$

and append the architecture and its validation MSE to the controller history.

Progressive architecture derivation At predefined epochs, the controller generates unique architectures that are ranked using inherited supernet weights. A two-stage protocol first evaluates all candidates on a fixed validation subset and then evaluates the strongest candidates on the full validation set. These candidates are retained in a persistent derivation pool.

At final derivation, historical and newly sampled candidates are reranked under the final supernet weights using the same protocol.

The highest-ranked candidates are independently retrained from scratch on the training set and reranked by validation error, with early stopping. The best architecture and checkpoint are evaluated once on the test set. Figure 6 and Algorithm 1 in Appendix A.3 provide the complete procedure.

4 Experiments

We evaluate the proposed framework on a diverse set of multivariate time series benchmarks and forecast horizons. Our experiments examine the detailed architecture search space and the LLM controller, as well as architecture selection, search strategy, and computational efficiency.

Datasets and setup We conduct experiments on eight widely used multivariate time series forecasting benchmarks covering several application domains: the four ETT variants (ETTh1, ETTh2, ETTm1, and ETTm2), Exchange, Weather, Electricity Consumption Load (ECL), and Solar-Energy. Following standard forecasting protocols, we fix the input sequence length to $L_{\text{in}} = 96$ and consider forecast horizons $L_{\text{out}} \in \{96, 192, 336, 720\}$ for every dataset. Appendix B provides further implementation and search details.

Table 1: Multivariate forecasting results with input length 96 and prediction lengths 96, 192, 336, and 720. Avg denotes the average over four horizons. The final TiNAS architecture selected for each task is reported in Appendix C.1.

Models		TiNAS (Ours)		iTransformer (2024)		RLinear (2023)		PatchTST (2023)		Crossformer (2023)		TiDE (2023)		TimesNet (2023)		DLinear (2022a)		SCINet (2022)		FEDformer (2022)		Stationary (2022b)		Autoformer (2021)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.322**	**0.359**	0.334	0.368	0.355	0.376	0.329	0.367	0.404	0.426	0.364	0.387	0.338	0.375	0.345	0.372	0.418	0.438	0.379	0.419	0.386	0.398	0.505	0.475
	192	**0.365**	**0.384**	0.377	0.391	0.391	0.392	0.367	0.385	0.450	0.451	0.398	0.404	0.374	0.387	0.380	0.389	0.439	0.450	0.426	0.441	0.459	0.444	0.553	0.496
	336	**0.395**	**0.406**	0.426	0.420	0.424	0.415	0.399	0.410	0.532	0.515	0.428	0.425	0.410	0.411	0.413	0.413	0.490	0.485	0.445	0.459	0.495	0.464	0.621	0.537
	720	0.459	0.447	0.491	0.459	0.487	0.450	**0.454**	**0.439**	0.666	0.589	0.487	0.461	0.478	0.450	0.474	0.453	0.595	0.550	0.543	0.490	0.585	0.516	0.671	0.561
	Avg	**0.385**	**0.399**	0.407	0.410	0.414	0.408	0.387	0.400	0.513	0.495	0.419	0.419	0.400	0.406	0.403	0.407	0.486	0.481	0.448	0.452	0.481	0.456	0.588	0.517
ETTm2	96	**0.175**	**0.258**	0.180	0.264	0.182	0.265	**0.175**	0.259	0.287	0.366	0.207	0.305	0.187	0.267	0.193	0.292	0.286	0.377	0.203	0.287	0.192	0.274	0.255	0.339
	192	0.243	0.303	0.250	0.309	0.246	0.304	**0.241**	**0.302**	0.414	0.492	0.290	0.364	0.249	0.309	0.284	0.362	0.399	0.445	0.269	0.328	0.280	0.339	0.281	0.340
	336	0.307	0.347	0.311	0.348	0.307	**0.342**	**0.305**	0.343	0.597	0.542	0.377	0.422	0.321	0.351	0.369	0.427	0.637	0.591	0.325	0.366	0.334	0.361	0.339	0.372
	720	0.406	0.404	0.412	0.407	0.407	**0.398**	**0.402**	0.400	1.730	1.042	0.558	0.524	0.408	0.403	0.554	0.522	0.960	0.735	0.421	0.415	0.417	0.413	0.433	0.432
	Avg	0.283	0.328	0.288	0.332	0.286	0.327	**0.281**	**0.326**	0.757	0.611	0.358	0.404	0.291	0.333	0.350	0.401	0.571	0.537	0.305	0.349	0.306	0.347	0.327	0.371
ETTh1	96	0.382	0.404	0.386	0.405	0.386	**0.395**	0.414	0.419	0.423	0.448	0.479	0.464	0.384	0.402	0.386	0.400	0.654	0.599	**0.376**	0.419	0.513	0.491	0.449	0.459
	192	0.429	0.430	0.441	0.436	0.437	**0.424**	0.460	0.445	0.471	0.474	0.525	0.492	0.436	0.429	0.437	0.432	0.719	0.631	**0.420**	0.448	0.534	0.504	0.500	0.482
	336	0.464	**0.446**	0.487	0.458	0.479	**0.446**	0.501	0.466	0.570	0.546	0.565	0.515	0.491	0.469	0.481	0.459	0.778	0.659	**0.459**	0.465	0.588	0.535	0.521	0.496
	720	0.537	0.496	0.503	0.491	**0.481**	**0.470**	0.500	0.488	0.653	0.621	0.594	0.558	0.521	0.500	0.519	0.516	0.836	0.699	0.506	0.507	0.643	0.616	0.514	0.512
	Avg	0.453	0.444	0.454	0.447	0.446	**0.434**	0.469	0.455	0.529	0.522	0.541	0.507	0.458	0.450	0.456	0.452	0.747	0.647	**0.440**	0.460	0.570	0.537	0.496	0.487
ETTh2	96	0.295	0.342	0.297	0.349	**0.288**	**0.338**	0.302	0.348	0.745	0.584	0.400	0.440	0.340	0.374	0.333	0.387	0.707	0.621	0.358	0.397	0.476	0.458	0.346	0.388
	192	**0.370**	0.398	0.380	0.400	0.374	**0.390**	0.388	0.400	0.877	0.656	0.528	0.509	0.402	0.414	0.477	0.476	0.860	0.689	0.429	0.439	0.512	0.493	0.456	0.452
	336	0.423	0.437	0.428	0.432	**0.415**	**0.426**	0.426	0.433	1.043	0.731	0.643	0.571	0.452	0.452	0.594	0.541	1.000	0.744	0.496	0.487	0.552	0.551	0.482	0.486
	720	0.439	0.456	0.427	0.445	**0.420**	**0.440**	0.431	0.446	1.104	0.763	0.874	0.679	0.462	0.468	0.831	0.657	1.249	0.838	0.463	0.474	0.562	0.560	0.515	0.511
	Avg	0.382	0.408	0.383	0.407	**0.374**	**0.399**	0.387	0.407	0.942	0.684	0.611	0.550	0.414	0.427	0.559	0.515	0.954	0.723	0.437	0.449	0.526	0.516	0.450	0.459
ECL	96	0.159	0.245	**0.148**	**0.240**	0.201	0.281	0.181	0.270	0.219	0.314	0.237	0.329	0.168	0.272	0.197	0.282	0.247	0.345	0.193	0.308	0.169	0.273	0.201	0.317
	192	0.170	0.255	**0.162**	**0.253**	0.201	0.283	0.188	0.274	0.231	0.322	0.236	0.330	0.184	0.289	0.196	0.285	0.257	0.355	0.201	0.315	0.182	0.286	0.222	0.334
	336	0.189	0.274	**0.178**	**0.269**	0.215	0.298	0.204	0.293	0.246	0.337	0.249	0.344	0.198	0.300	0.209	0.301	0.269	0.369	0.214	0.329	0.200	0.304	0.231	0.338
	720	0.228	**0.309**	0.225	0.317	0.257	0.331	0.246	0.324	0.280	0.363	0.284	0.373	**0.220**	0.320	0.245	0.333	0.299	0.390	0.246	0.355	0.222	0.321	0.254	0.361
	Avg	0.187	0.271	**0.178**	**0.270**	0.219	0.298	0.205	0.290	0.244	0.334	0.251	0.344	0.192	0.295	0.212	0.300	0.268	0.365	0.214	0.327	0.193	0.296	0.227	0.338
Exchange	96	**0.086**	0.206	**0.086**	0.206	0.093	0.217	0.088	**0.205**	0.256	0.367	0.094	0.218	0.107	0.234	0.088	0.218	0.267	0.396	0.148	0.278	0.111	0.237	0.197	0.323
	192	**0.176**	**0.298**	0.177	0.299	0.184	0.307	**0.176**	0.299	0.470	0.509	0.184	0.307	0.226	0.344	**0.176**	0.315	0.351	0.459	0.271	0.315	0.219	0.335	0.300	0.369
	336	0.343	0.425	0.331	0.417	0.351	0.432	**0.301**	**0.397**	1.268	0.883	0.349	0.431	0.367	0.448	0.313	0.427	1.324	0.853	0.460	0.427	0.421	0.476	0.509	0.524
	720	0.868	0.706	0.847	**0.691**	0.886	0.714	0.901	0.714	1.767	1.068	0.852	0.698	0.964	0.746	**0.839**	0.695	1.058	0.797	1.195	0.695	1.092	0.769	1.447	0.941
	Avg	0.368	0.409	0.360	**0.403**	0.378	0.417	0.367	0.404	0.940	0.707	0.370	0.413	0.416	0.443	**0.354**	0.414	0.750	0.626	0.519	0.429	0.461	0.454	0.613	0.539
Weather	96	0.173	**0.213**	0.174	0.214	0.192	0.232	0.177	0.218	**0.158**	0.230	0.202	0.261	0.172	0.220	0.196	0.255	0.221	0.306	0.217	0.296	0.173	0.223	0.266	0.336
	192	0.219	**0.254**	0.221	**0.254**	0.240	0.271	0.225	0.259	**0.206**	0.277	0.242	0.298	0.219	0.261	0.237	0.296	0.261	0.340	0.276	0.336	0.245	0.285	0.307	0.367
	336	0.278	0.297	0.278	**0.296**	0.292	0.307	0.278	0.297	**0.272**	0.335	0.287	0.335	0.280	0.306	0.283	0.335	0.309	0.378	0.339	0.380	0.321	0.338	0.359	0.395
	720	0.353	**0.346**	0.358	0.347	0.364	0.353	0.354	0.348	0.398	0.418	0.351	0.386	0.365	0.359	**0.345**	0.381	0.377	0.427	0.403	0.428	0.414	0.410	0.419	0.428
	Avg	**0.256**	**0.278**	0.258	**0.278**	0.272	0.291	0.259	0.281	0.259	0.315	0.271	0.320	0.259	0.287	0.265	0.317	0.292	0.363	0.309	0.360	0.288	0.314	0.338	0.382
Solar-Energy	96	0.206	0.246	**0.203**	**0.237**	0.322	0.339	0.234	0.286	0.310	0.331	0.312	0.399	0.250	0.292	0.290	0.378	0.237	0.344	0.242	0.342	0.215	0.249	0.884	0.711
	192	0.235	**0.258**	**0.233**	0.261	0.359	0.356	0.267	0.310	0.734	0.725	0.339	0.416	0.296	0.318	0.320	0.398	0.280	0.380	0.285	0.380	0.254	0.272	0.834	0.692
	336	0.254	0.275	**0.248**	**0.273**	0.397	0.369	0.290	0.315	0.750	0.735	0.368	0.430	0.319	0.330	0.353	0.415	0.304	0.389	0.282	0.376	0.290	0.296	0.941	0.723
	720	0.254	**0.274**	**0.249**	0.275	0.397	0.356	0.289	0.317	0.769	0.765	0.370	0.425	0.338	0.337	0.356	0.413	0.308	0.388	0.357	0.427	0.285	0.295	0.882	0.717
	Avg	0.237	0.263	**0.233**	**0.262**	0.369	0.356	0.270	0.307	0.641	0.639	0.347	0.417	0.301	0.319	0.330	0.401	0.282	0.375	0.291	0.381	0.261	0.278	0.885	0.711
Overall Avg		**0.319**	**0.350**	0.320	0.351	0.345	0.366	0.328	0.359	0.603	0.538	0.396	0.422	0.341	0.370	0.366	0.401	0.544	0.515	0.370	0.401	0.386	0.400	0.490	0.475

Evaluation metrics We report mean squared error (MSE) and mean absolute error (MAE) over all test predictions, with lower values indicating better forecasting performance.

Baselines We compare our method with eleven representative forecasting models. The Transformer models include Autoformer (Wu et al., 2021), FEDformer (Zhou et al., 2022), Non-stationary Transformer (Liu et al., 2022c), Crossformer (Zhang & Yan, 2023), PatchTST (Nie et al., 2023), and iTransformer (Liu et al., 2024). We also include the linear and MLP models DLinear (Zeng et al., 2023), RLinear (Li et al., 2023), and TiDE (Das et al., 2023), together with the convolutional models SCINet (Liu et al., 2022a) and TimesNet (Wu et al., 2023).

4.1 Overall Forecasting Performance

Main results Table 1 summarizes the multivariate forecasting results over four forecast horizons. Across the 32 settings, TiNAS achieves the lowest overall MSE of 0.319 and MAE of 0.350. TiNAS achieves the lowest average MSE and MAE on ETTm1, and the lowest MSE with a tied MAE on Weather. Compared with PatchTST, TiNAS lowers the average MSE on six of eight datasets, with notable improvements on ECL and Solar. On Solar, TiNAS consistently outperforms PatchTST across all forecast horizons. The search space ablation separately evaluates the gains from searching beyond the fixed reference configuration.

Obtained TiNAS architectures We further examine whether the obtained configurations remain preferable as the forecast horizon changes. For each of ETTm1 and ETTh1, we retrain the four obtained configurations A1–A4, selected at horizons 96, 192, 336 and 720, at all four target horizons. Figure 3 shows that the configuration selected for the target forecast horizon achieves the lowest MSE in six of eight settings. No evaluated configuration is best across all four horizons on either dataset. These results suggest that the preferred configuration varies with the forecast horizon even within

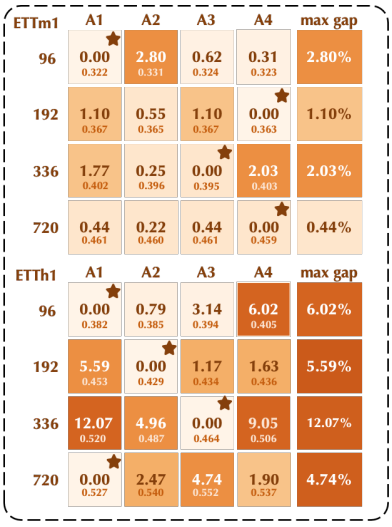

Figure 3: Evaluation across forecast horizons. Cells give the percentage MSE gap to the best configuration at each target horizon, with the test MSE below; stars mark the per-horizon best.

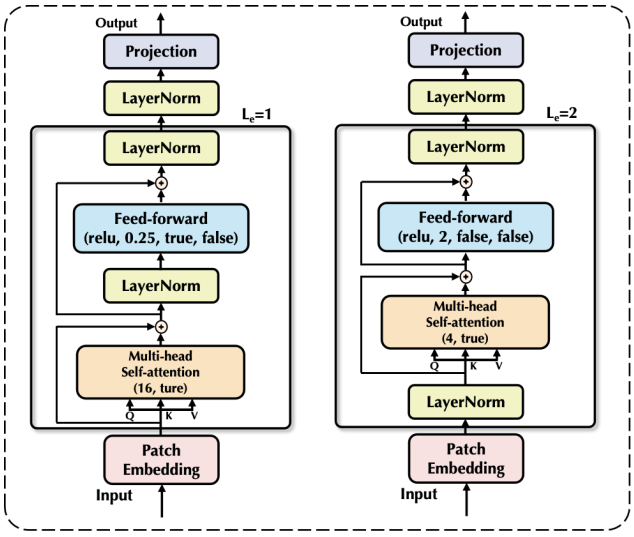

Figure 4: Architectures obtained for ETTh1 at horizon 336 (left) and ETTm1 at horizon 720 (right). The two configurations differ in encoder depth, attention heads, and normalization placement, and both disable the optional local convolution branches. Full configurations are provided in Appendix C.1.

the same dataset. The examples in Figure 4 illustrate the corresponding structural choices through differences in encoder depth as well as attention settings and normalization placement within the same PatchTST backbone.

Increasing lookback length Longer lookback windows expose a forecaster to broader temporal context, but prior work shows that additional history is not uniformly useful: conventional Transformer forecasters may plateau or deteriorate as inputs grow, whereas patch-based and inverted designs can better exploit expanded receptive fields (Zeng et al., 2023; Nie et al., 2023; Liu et al., 2024). This architecture-dependent behavior motivates us to test TiNAS beyond the standard $L_{\text{in}} = 96$ setting. We fix $L_{\text{out}} = 96$ and compare TiNAS with representative Transformer-based, linear, and convolutional baselines on Weather and ETTh2 for $L_{\text{in}} \in \{48, 96, 192, 336, 720\}$. Figure 5 summarizes how forecasting error varies with the available context. This controlled sweep complements the fixed-length benchmark by assessing robustness to input length while holding the forecast horizon constant.

Efficiency analysis At $L_{\text{in}} = L_{\text{out}} = 96$ on a single RTX 4090 GPU, the eight profiled tasks required 74.96–298.52 s per combined search epoch and 9.20–136.47 s per periodic-derivation call (Appendix C.6). Controller-optimization time varied little across tasks (42.04–43.44 s per epoch), whereas the shared-weight phase ranged from 32.92 to 255.36 s and accounted for most of the cross-task variation. The 1.546B-parameter controller keeps its LLM backbone frozen and optimizes only 2.242M LoRA and action-head parameters (0.145%); the shared supernet contains 19.718M trainable parameters. These short-run stage profiles are not complete search-to-deployment runtimes, and both components are discarded after architecture selection.

4.2 Ablation Studies

Controller ablation We ablate four controller components individually while retaining the same search space, supernet, reward, search budget, and derivation procedure. w/o Action Semantics replaces descriptive action labels with arbitrary codes while preserving the available architectural choices. w/o Previous Choices omits earlier choices in the current architecture from the prompt. w/o Performance History omits evaluated architectures and their validation MSE from the prompt but retains reward-based policy updates. w/o LoRA Updates freezes the LoRA parameters while continuing to train the action heads. In Table 2, Full Controller achieves the lowest average MSE and

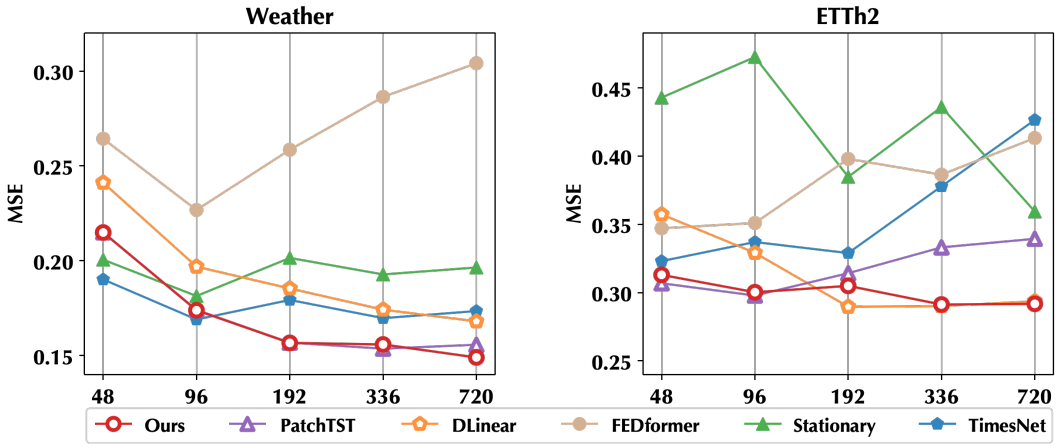

Figure 5: Forecasting performance with increasing lookback lengths. Test MSE of TiNAS and representative baselines on Weather (left) and ETTh2 (right) across lookback lengths from 48 to 720. Lower is better.

MAE on ETTh1 and Exchange and ties for the lowest averages on ETTm1. On Exchange, removing previous choices raises average MSE from 0.368 to 0.429, while freezing LoRA increases it to 0.413. Removing action semantics or performance history also increases average MSE on all three datasets. These results support the tested context and adaptation components, although their effects vary across datasets. Appendix C.2 reports results for individual forecast horizons.

Table 2: Controller ablation. Results are averaged over four forecast horizons, with results for individual forecast horizons reported in Appendix C.2. The best result is shown in bold and the second-best result is underlined.

Controller	w/o Action Semantics		w/o Previous Choices		w/o Performance History		w/o LoRA Updates		Full Controller	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	0.391	0.402	<u>0.387</u>	<u>0.400</u>	0.388	**0.399**	**0.385**	**0.399**	**0.385**	**0.399**
ETTh1	<u>0.454</u>	<u>0.445</u>	0.462	0.448	0.466	0.447	0.458	0.446	**0.453**	**0.444**
Exchange	0.389	<u>0.414</u>	0.429	0.435	<u>0.380</u>	<u>0.414</u>	0.413	0.425	**0.368**	**0.409**

Search space ablation We compare four design spaces with different searchable scopes: Fixed, Macro, Module, and Full. Fixed holds all fourteen decisions at the reference values in Appendix A.1. Macro searches feed-forward design, both local temporal operators, and encoder depth while fixing normalization placement, $h = 16$, and $s = 1$. Module searches attention and feed-forward design while fixing normalization placement, $k_{\text{att}} = k_{\text{ffn}} = 0$, and $L_e = 3$. Thus, Macro and Module isolate different decision groups rather than form a nested pair. Full searches all fourteen decisions. We also report Near-Full Avg., which averages five spaces that each fix one architectural group while searching all remaining groups. In Table 3, Full reduces the MSE over Fixed from 0.392 to 0.385 on ETTm1 and from 0.189 to 0.187 on ECL. The differences on Weather are small. The small gap between Near-Full Avg. and Full indicates limited average sensitivity to the five single-group restrictions on these datasets. Appendix C.4 reports per-horizon results and the five constituent restrictions.

Search strategy ablation We ablate three aspects of the search procedure: controller-guided proposals, periodic derivation, and candidate accumulation across derivation rounds. Uniform Sampling replaces the LLM controller while retaining the same search space and evaluation budget. w/o Periodic Derivation derives candidates only after search, whereas w/o Candidate Accumulation discards candidates retained from intermediate derivations before final selection. Uniform Sampling increases average MSE from 0.385 to 0.392 on ETTm1 and from 0.368 to 0.387 on Exchange.

Table 3: Search space ablation averaged over four forecast horizons. Parentheses list the groups searched by Macro and Module. Near-Full Avg. averages five search spaces that each fix one architectural group while searching all others. Results for individual forecast horizons and the five constituent restrictions are provided in Appendix C.4.

Search Space	Fixed		Macro (FFN+Local+Depth)		Module (Attention+FFN)		Near-Full Avg.		Full	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	0.392	0.404	0.391	0.402	0.386	0.400	0.387	0.400	**0.385**	**0.399**
ECL	0.189	0.275	0.188	0.273	0.188	0.272	0.188	0.273	**0.187**	**0.271**
Weather	0.257	0.279	**0.256**	0.279	0.257	0.279	**0.256**	**0.278**	**0.256**	**0.278**

The effect of candidate accumulation is most pronounced on Exchange, where Full + Retraining reduces average MSE by 7.5% relative to its ablation. Independent retraining performs better than inherited-weight fine-tuning on ETTm1 and Exchange, whereas fine-tuning is marginally better on Weather. The gains from controller guidance and candidate accumulation are therefore task dependent.

Table 4: Search strategy ablation. Results are averaged over four forecast horizons, with detailed settings and results for individual forecast horizons provided in Appendix C.5. The best result is shown in bold and the second-best result is underlined.

Search Strategy	Uniform Sampling		w/o Periodic Derivation		w/o Candidate Accumulation		Full + Fine-tuning		Full + Retraining	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	0.392	0.403	0.392	0.403	0.393	0.403	<u>0.387</u>	<u>0.402</u>	**0.385**	**0.399**
Exchange	0.387	0.419	<u>0.370</u>	**0.408**	0.398	0.420	0.424	0.428	**0.368**	<u>0.409</u>
Weather	<u>0.256</u>	<u>0.278</u>	<u>0.256</u>	0.279	0.257	0.279	**0.255**	**0.277**	<u>0.256</u>	<u>0.278</u>

5 Conclusion and Future Work

We presented TiNAS, a task-adaptive neural architecture search framework that extends time series forecasting adaptation from parameter optimization to architecture optimization while preserving the PatchTST representation pathway. TiNAS searches fine-grained encoder choices, including normalization placement, attention configuration, local temporal operators, feed-forward design, and encoder depth. Its autoregressive LLM controller generates dependent architectural decisions conditioned on task information, preceding choices, and validation history, and learns its sampling policy from validation MSE rewards. A weight-sharing supernet and progressive architecture derivation enable efficient candidate screening, after which shortlisted architectures are independently retrained from scratch. Across eight forecasting benchmarks and four forecast horizons, TiNAS achieved the top two average MSE and MAE on four datasets and reduced the average MSE of PatchTST on six. The ablations further support the complementary roles of the search space, controller, and search strategy, while the horizon-dependent architectures demonstrate the value of task-specific structural adaptation within an established forecasting backbone.

Limitations and future work TiNAS currently requires a separate search for each dataset–horizon task, and its supernet and LLM controller incur temporary search-time overhead. Its search space is also limited to the PatchTST encoder, while the evaluation focuses on regularly sampled multivariate point forecasting. Future work will investigate cross-task policy transfer, lightweight controllers or performance predictors, and multi-objective search over accuracy and deployment cost. Extending the search to patch construction and cross-variable modeling, and evaluating probabilistic forecasting, missing observations, and distribution shifts, are further directions.

Ethics statement

This work investigates neural architecture search for time series forecasting using publicly available benchmark datasets. It does not involve human subjects or the collection of new personal data. To the best of our knowledge, the proposed methodology raises no specific ethical concerns.

Reproducibility statement

The proposed search space, autoregressive controller, and architecture search procedure are described in the main text. Implementation details, hyperparameter settings, dataset preprocessing, and selected architecture configurations are provided in the Appendix. All experiments are conducted on publicly available forecasting benchmarks. We plan to release our implementation upon acceptance.

AI Use Statement

In this work, generative AI tools were used to assist with research methodology and experimental design, code development and debugging, and manuscript drafting and editing. All AI-assisted content was reviewed by the authors. The authors take full responsibility for the final content of this work, including its implementation, experimental results, and conclusions.

References

Qianying Cao, Shanqing Liu, Alan John Varghese, Jérôme Darbon, Michael S. Triantafyllou, and George Em Karniadakis. Automatic selection of the best neural architecture for time series forecasting. *Nature Communications*, 17(1):7058, 2026. doi: 10.1038/s41467-026-73687-9. URL https://doi.org/10.1038/s41467-026-73687-9.

Angelica Chen, David Dohan, and David So. EvoPrompting: Language models for code-level neural architecture search. In *Advances in Neural Information Processing Systems*, volume 36, pp. 7787–7817, 2023. doi: 10.52202/075280-0342. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/184c1e18d00d7752805324da48ad25be-Abstract-Conference.html.

Donghui Chen, Ling Chen, Zongjiang Shang, Youdong Zhang, Bo Wen, and Chenghu Yang. Scale-aware neural architecture search for multivariate time series forecasting. *arXiv preprint arXiv:2112.07459*, 2021. URL https://arxiv.org/abs/2112.07459.

Yushu Chen, Shengzhuo Liu, Jinzhe Yang, Hao Jing, Wenlai Zhao, and Guangwen Yang. A joint time-frequency domain transformer for multivariate time series forecasting. *Neural Networks*, 176: 106334, 2024. doi: 10.1016/j.neunet.2024.106334. URL https://doi.org/10.1016/j.neunet.2024.106334.

Xiangxiang Chu, Bo Zhang, and Ruijun Xu. FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12239–12248, 2021.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K. Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=pCbC3aQB5W.

Difan Deng and Marius Lindauer. Optimizing time series forecasting architectures: A hierarchical neural architecture search approach. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Ym2wqojm4e.

Yuxuan Hu, Jihao Liu, Ke Wang, Jinliang Zheng, Weikang Shi, Manyuan Zhang, Qi Dou, Rui Liu, Aojun Zhou, and Hongsheng Li. LM-searcher: Cross-domain neural architecture search with LLMs via unified numerical encoding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9408–9421. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.478. URL https://aclanthology.org/2025.emnlp-main.478/.

Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. RZ-NAS: Enhancing LLM-guided neural architecture search via reflective zero-cost strategy. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 27237–27254. PMLR, 2025. URL https://proceedings.mlr.press/v267/ji25a.html.

Haoyi Jia. STrans: Spontaneous architecture evolution for adaptive time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(27):22238–22246, 2026. doi: 10.1609/aaai.v40i27.39380. URL https://ojs.aaai.org/index.php/AAAI/article/view/39380.

Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*, 2023. URL https://arxiv.org/abs/2305.10721.

Zhe Li, Zhiwei Lin, and Yongtao Wang. CoLLM-NAS: Collaborative large language models for efficient knowledge-guided neural architecture search. *arXiv preprint arXiv:2509.26037*, 2025. URL https://arxiv.org/abs/2509.26037.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *The Seventh International Conference on Learning Representations*. OpenReview.net, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. SCINet: Time series modeling and forecasting with sample convolution and interaction. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5816–5828, 2022a. doi: 10.52202/068431-0421. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/266983d0949aed78a16fa4782237dea7-Abstract-Conference.html.

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 *The Tenth International Conference on Learning Representations*. OpenReview.net, 2022b. URL https://openreview.net/forum?id=0EXmFzUn5I.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, 2022c. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/4054556fcaa934b0bf76da52cf4f92cb-Abstract-Conference.html.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*. OpenReview.net, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

Muhammad Umair Nasir, Sam Earle, Christopher W. Cleghorn, Steven James, and Julian Togelius. LLMatic: Neural architecture search via large language models and quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1110–1118. ACM, 2024. doi: 10.1145/3638529.3654017. URL https://doi.org/10.1145/3638529.3654017.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4095–4104. PMLR, 2018. URL https://proceedings.mlr.press/v80/pham18a.html.

Mohammad Amin Shabani, Amir H. Abdi, Lili Meng, and Tristan Sylvain. Scaleformer: Iterative multi-scale refining transformers for time series forecasting. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=sCrnllCtjoE.

Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven C. H. Hoi. ETSformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022. URL https://arxiv.org/abs/2202.01381.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. GreedyNAS: Towards fast one-shot NAS with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1996–2005, 2020.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, 2023. doi: 10.1609/aaai.v37i9.26317. URL https://ojs.aaai.org/index.php/AAAI/article/view/26317.

Juyuan Zhang, Wei Zhu, and Jiechao Gao. Learning novel transformer architecture for time-series forecasting. *arXiv preprint arXiv:2502.13721*, 2025. URL https://arxiv.org/abs/2502.13721.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=vSVLM2j9eie.

Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. Can GPT-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970*, 2023. URL https://arxiv.org/abs/2304.10970.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325. URL https://ojs.aaai.org/index.php/AAAI/article/view/17325.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27268–27286. PMLR, 2022. URL https://proceedings.mlr.press/v162/zhou22g.html.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *The Fifth International Conference on Learning Representations*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

A METHOD DETAILS

A.1 SEARCH SPACE SPECIFICATION

Table 5 summarizes the encoder search space of TiNAS. The space consists of fourteen decisions in five groups, namely attention heads and scaling, feed-forward ratio and activation, the two local temporal branches, normalization placement, and encoder depth. The table reports the candidate values of every decision and the reference configuration, which fixes all decisions to the PatchTST encoder design. The Fixed baseline of the search space ablation in Table 3 keeps this configuration, and each restricted variant in Table 10 resets one group to it while the remaining groups stay searchable.

Table 5: Architecture search space of the encoder. Each row reports one of the fourteen decisions, its candidate values, and its value in the PatchTST-style reference configuration. A kernel size of 0 disables the corresponding local branch, $s = 1$ applies scaled dot-product attention, and $u_i = 1$ inserts LayerNorm at the listed position. The terminal normalization of each encoder block remains fixed.

Decision	Candidates	Reference
Attention		
attention heads h	$\{4, 8, 16\}$	16
scaled attention s	$\{0, 1\}$	1
Feed-forward		
width ratio r	$\{0.25, 0.5, 1, 2, 4\}$	1
activation g	GELU, Swish, ReLU, LeakyReLU	GELU
Local temporal		
kernel after attention k_{att}	$\{0, 1, 3, 5\}$	0
kernel after FFN k_{ffn}	$\{0, 1, 3, 5\}$	0
Normalization placement		
u_1 before attention	$\{0, 1\}$	0
u_2 after the attention residual	$\{0, 1\}$	1
u_3 inside the first local branch	$\{0, 1\}$	0
u_4 after the first FFN projection	$\{0, 1\}$	0
u_5 after the FFN activation and dropout	$\{0, 1\}$	0
u_6 after the FFN residual	$\{0, 1\}$	0
u_7 inside the second local branch	$\{0, 1\}$	0
Encoder depth		
depth L_e	$\{1, 2, 3, 4, 6\}$	3

The decisions are shared by all active encoder blocks, and each block keeps its own parameters. An architecture of depth L_e activates the first L_e of the six supernet blocks. When all three candidate head counts divide D, the product of the candidate counts gives $1{,}228{,}800$ configurations in total.

The fourteen decisions also form the action sequence of the controller. The controller prompt in Appendix A.2 presents these decisions in the same order with the same candidates, and the policy selects one value for one decision at each step and conditions on the preceding choices, that is, $\pi_\psi(a \mid q) = \prod_{t=1}^{T} \pi_\psi(a_t \mid q, a_{<t})$, factorizing as in Eq. (5) with $a_{<t} = (a_1, \ldots, a_{t-1})$.

Attention and feed-forward parameterization We make the remaining operator definitions underlying Eq. (3) explicit here. Let $\mathcal{D}$ denote dropout, with independently sampled masks at each occurrence. For $\mathbf{Z} \in \mathbb{R}^{N \times D}$, shared query, key, and value projections are partitioned into h heads of width $d_h = D/h$. Head i computes

$$\mathbf{O}_i^{(h,s)} = \mathcal{D}\Big(\mathrm{Softmax}\Big(d_h^{-s/2}\mathbf{Q}_i\mathbf{K}_i^\top\Big)\Big)\,\mathbf{V}_i, \qquad i = 1, \ldots, h. \tag{9}$$

Thus, $s = 1$ gives scaled attention and $s = 0$ leaves the logits unscaled. The multi-head mapping is

$$\mathcal{A}_{h,s}(\mathbf{Z}) = \mathrm{Concat}\Big(\mathbf{O}_1^{(h,s)}, \ldots, \mathbf{O}_h^{(h,s)}\Big)\,\mathbf{W}_O. \tag{10}$$

Projection biases are omitted from the notation. All head configurations reuse the same projection parameters within a block.

The base two-projection FFN, before optional normalization and dropout, is

$$\mathrm{FFN}_{r,g}(\mathbf{Z}) = g\Big(\mathbf{Z}\mathbf{W}_1^{(r)} + \mathbf{b}_1^{(r)}\Big)\,\mathbf{W}_2^{(r)} + \mathbf{b}_2, \qquad D_r = \mathrm{round}(rD). \tag{11}$$

The supernet stores projections at width $D_{\max} = 4D$. Candidates activate the following prefix slices, with block indices omitted:

$$\mathbf{W}_1^{(r)} = \mathbf{W}_1[:, 1{:}D_r], \qquad \mathbf{b}_1^{(r)} = \mathbf{b}_1[1{:}D_r], \qquad \mathbf{W}_2^{(r)} = \mathbf{W}_2[1{:}D_r, :]. \tag{12}$$

The second-projection bias $\mathbf{b}_2$ is shared in full.

Exact normalization. For position j in block ℓ, the conditional normalization is

$$\mathcal{N}_{\ell,j}^{(u_j)}(\mathbf{Z}) = \begin{cases} \mathrm{LN}_{\ell,j}(\mathbf{Z}), & u_j = 1, \\ \mathbf{Z}, & u_j = 0. \end{cases} \tag{13}$$

The positions $j = 4, 5$ normalize the FFN hidden dimension D_r with separate LayerNorm parameters for each candidate width; the remaining positions normalize the feature dimension D. The attention and FFN residual mappings are given in Section 3.1.

The two local branches use the operator in Eq. (2), with block- and branch-specific parameters. Their optional normalizations precede convolution. For nonzero k, depthwise convolution uses D groups, a bias, and padding $\lfloor k/2 \rfloor$. Each block–branch–kernel scale is initialized to 10^{-2}. For $k = 0$, the entire branch, including its optional LayerNorm, is skipped. As shown in Eq. (3), the FFN consumes only the attention residual state, and both auxiliary outputs are added at the block output before the fixed terminal LayerNorm.

A.2 Architecture Controller

The controller produces the fourteen decisions of an architecture in a fixed order. The prompt assembled for each decision step has four parts. The first part describes the forecasting task with the dataset, the number of variables, the input length, and the horizon. The second part restates the decisions and candidate values summarized in Table 5 as a list of fields, one field per decision. The third part lists recently verified architectures with their validation MSE, and the history grows whenever the controller phase evaluates a sampled architecture (Appendix A.3). The fourth part collects the choices made in the preceding steps, and every new choice is appended before the next step starts. The prompt assembled at the first decision step appears below, with the history excerpt shortened and the middle normalization fields omitted. The field names in the listing follow the notation of Table 5, and the released code stores the same decisions under the corresponding encoder keys.

```
You are a Neural Architecture Search controller.
Design a PatchTST encoder for the forecasting task below by choosing one
value for each decision in order.

Task: ETTm1, 7 variables, input = 96, horizon = 192
Objective: minimize validation MSE.

Search space:
u1 before attention: 0 | 1
u2 after the attention residual: 0 | 1
... (u3 to u7 follow the same format)
h attention heads: 4 | 8 | 16
s scaled attention: 0 | 1
k_att kernel after attention: 0 | 1 | 3 | 5
g FFN activation: gelu | swish | relu | leaky_relu
r FFN width ratio: 0.25 | 0.5 | 1 | 2 | 4
k_ffn kernel after FFN: 0 | 1 | 3 | 5
L_e encoder depth: 1 | 2 | 3 | 4 | 6

Recent verified architectures (lower validation MSE is better):
u = 0010000, h = 16, s = 1, k_att = 1, g = relu, r = 0.25,
k_ffn = 5, L_e = 3 | val mse = 0.2871
...
```

```
Select the architecture expected to achieve the lowest validation MSE.
Your choices so far:
u1 = 0
u2 = 1
...
```

At each step the full context is encoded by the LLM, and the last-token hidden state of its final layer serves as the policy state $\mathbf{h}_t$, that is, the last token of $q \oplus \text{text}(a_{<t})$ in Eq. (5). Fourteen linear heads, one per decision, map $\mathbf{h}_t$ to the candidate logits of the corresponding field, which follow the transformation of the same equation. During architecture search we sample with softmax temperature $\tau = 5.0$ and logit bound $c = 0.5$, and candidate derivation uses temperature 1.0 and bound 1.0.

The backbone is Qwen2.5-1.5B with frozen pretrained weights. LoRA adapters with rank 16 and scaling factor 32 are attached to the query and value projections, and no dropout is used. Only the LoRA parameters and the action heads receive gradient updates, with separate Adam learning rates of 10^{-4} and 3.5×10^{-4}, and the controller gradient is clipped to a norm of 2. The total and trainable parameter counts of the controller are reported in Appendix C.6.

A.3 Complete Search Procedure

The main text describes the roles and objectives of the three search stages. Figure 6 makes the feedback and parameter-update paths explicit, whereas Algorithm 1 records their execution order. Here, we report only the budgets and settings needed to instantiate this workflow.

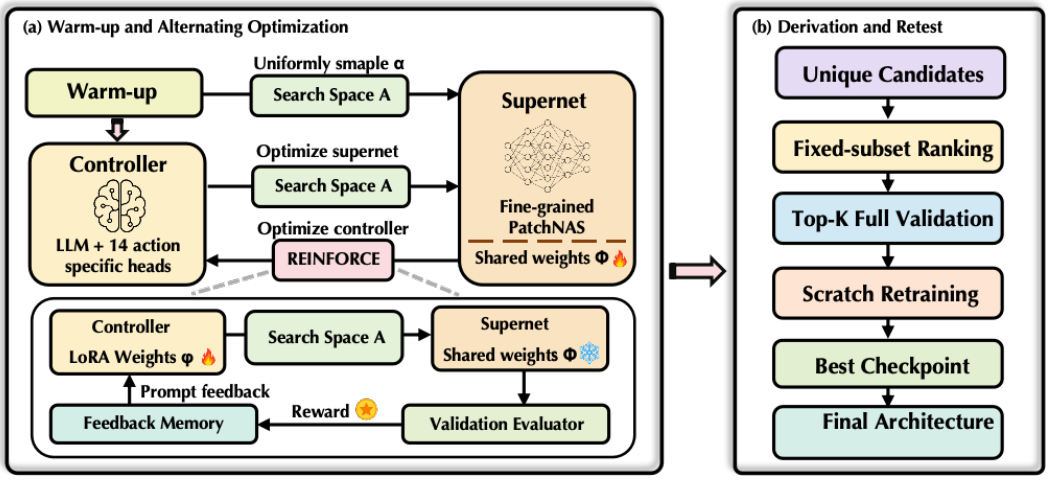

Figure 6: **Architecture search procedure of TiNAS.** (a) Uniform supernet warm-up followed by alternating optimization of the shared supernet weights and LLM controller. (b) Progressive architecture derivation, independent retraining from scratch, and final architecture selection.

The complete search schedule contains 40 epochs: 10 for uniform supernet warm-up and 30 for alternating optimization. During warm-up, one architecture is sampled uniformly for each training batch, and only its activated shared supernet weights are updated. In each subsequent shared-weight phase, the controller draws 64 architecture samples. Duplicate configurations are removed, uniform samples are added only when fewer than 16 unique architectures remain, and the shuffled pool is cycled across training batches.

Controller optimization then proceeds with the shared supernet fixed. Each epoch performs five controller updates, each aggregating four independently sampled architectures on the same four validation batches, thereby comparing their rewards under matched validation conditions. The reward baseline uses an exponential moving-average decay of 0.95, and the entropy coefficient is 1×10^{-4}. Each prompt retains at most five distinct high-reward records from the current and preceding search epochs as performance history.

Architecture derivation is performed after every five epochs of alternating optimization, with the final derivation at epoch 30. Each round collects 16 unique controller samples, ranks them on a fixed subset of eight evenly spaced validation batches, and reevaluates the leading 10 on the complete validation set. Both the shared supernet and controller remain fixed during derivation. The fully evaluated candidates from periodic rounds accumulate in the historical derivation pool $\mathcal{Q}$. During final derivation, $\mathcal{Q}$ is merged with 16 new candidates, deduplicated, and reranked under the final supernet weights, placing candidates discovered at different search stages under a common inherited-weight state. The leading three architectures are independently retrained from scratch for at most 20 epochs using a learning rate of 1×10^{-4}, a cosine schedule, and validation-based early stopping with patience 5. Their best checkpoints are reranked by validation MSE, validation MAE, and inherited derivation rank before the selected checkpoint is evaluated once on the test set. This separation confines inherited weights to candidate screening; final model selection relies on independently trained checkpoints.

Algorithm 1 TiNAS: Architecture Search and Selection

Require: Training, validation, and test sets D_t, D_v, D_e; search space $\mathcal{A}$; controller prompt q
Require: Warm-up and search epochs W, E; controller steps C; sampling budgets M, N, N_0
Require: Derivation schedule Γ; candidate budgets L, K, R
Ensure: Selected architecture a^* and trained weights w^*
Initialize supernet weights ϕ, controller ψ, an unset baseline b, and an empty history in q
Set $\mathcal{Q} \leftarrow \varnothing$ and fix an evenly spaced validation subset $V \subset D_v$
Stage I: Supernet warm-up
for $e = 1, \ldots, W$ **do**
    For each batch in D_t, uniformly sample $a \in \mathcal{A}$ and update its active weights to minimize training MSE
end for
Stage II: Alternating optimization
for $e = 1, \ldots, E$ **do**
    Draw N architectures from $\pi_\psi(\cdot \mid q)$ and deduplicate to form $\mathcal{S}$
    Add unique uniform samples until $|\mathcal{S}| \geq N_0$; shuffle $\mathcal{S}$
    Fix ψ; cycle through $\mathcal{S}$ across training batches and update active weights to minimize MSE
    for $k = 1, \ldots, C$ **do**
        Fix ϕ and select shared validation batches $B \subset D_v$
        Sequentially draw and evaluate M architectures on B, updating the history in q after each evaluation
        Update ψ using the mean eligible gradient from equation 7 and running baseline b
    end for
    if $e \in \Gamma$ **then**
        Fix ϕ and ψ; draw L unique candidates from $\pi_\psi(\cdot \mid q)$
        Rank candidates on V using inherited weights; evaluate the best K on D_v
        Add these K candidate configurations to $\mathcal{Q}$
    end if
end for
Stage III: Final selection and training
Draw L new unique controller candidates; merge with $\mathcal{Q}$ and deduplicate
Rank all candidates on V using final supernet weights
Evaluate the best K on D_v and retain the best R
Train each retained architecture from scratch with validation-based early stopping
Rank the best checkpoints by validation MSE, MAE, and inherited derivation rank; select (a^*, w^*)
Evaluate (a^*, w^*) once on D_e
return (a^*, w^*)

B ADDITIONAL EXPERIMENTAL DETAILS

Table 6 reports the settings that define each dataset–horizon run. Here, D is the backbone model dimension, p_{drop} is the model dropout rate, B is the training and validation batch size, and η_{sh}, η_{LoRA}, η_{head}, and η_{rt} are the learning rates for the shared supernet, controller LoRA parameters, structured action heads, and scratch retraining, respectively.

Table 6: Run-level hyperparameters used for the 32 dataset–horizon tasks in Table 1. Learning rates are reported in units of 10^{-4}. The epoch tuple $\boldsymbol{e} = (E_{\mathrm{w}}, E_{\mathrm{s}}, E_{\mathrm{r}})$ denotes the warm-up, total search (including warm-up), and maximum scratch-retraining epochs, respectively.

Task		**Backbone**		**Optimization**					**Epochs**
Dataset	**H**	D	p_{drop}	B	η_{sh}	η_{LoRA}	η_{head}	η_{rt}	$\boldsymbol{e}$
ETTm1	96	512	0.1	32	1	1	3.5	1	10/40/20
	192	128	0.1	32	1	1	3.5	1	10/40/20
	336	128	0.1	32	1	1	3.5	1	10/40/20
	720	512	0.1	32	1	1	3.5	1	10/40/20
ETTm2	96	128	0.1	32	1	1	3.5	1	10/40/20
	192	128	0.1	32	1	1	3.5	1	10/40/20
	336	128	0.1	32	1	1	3.5	1	10/40/20
	720	512	0.1	32	1	1	3.5	1	10/40/20
ETTh1	96	512	0.1	32	1	1	3.5	1	10/40/20
	192	256	0.1	32	1	1	3.5	1	10/40/20
	336	256	0.1	32	1	1	3.5	1	10/40/20
	720	256	0.1	32	1	1	3.5	1	10/40/20
ETTh2	96	128	0.1	32	1	1	3.5	1	10/40/20
	192	128	0.1	32	1	1	3.5	1	10/40/20
	336	128	0.1	32	1	1	3.5	1	10/40/20
	720	128	0.1	32	1	1	3.5	1	10/40/20
ECL	96	512	0.1	8	1	1	3.5	1	10/40/20
	192	512	0.1	8	1	1	3.5	1	10/40/20
	336	512	0.1	8	1	1	3.5	1	10/40/20
	720	512	0.1	8	1	1	3.5	1	10/40/20
Exchange	96	512	0.1	16	1	1	3.5	1	10/40/20
	192	512	0.0	16	1	1	3.5	1	10/40/20
	336	128	0.1	32	1	1	3.5	1	10/40/20
	720	512	0.1	16	1	1	3.5	1	10/40/20
Weather	96	512	0.1	128	1	1	3.5	1	10/40/20
	192	512	0.1	128	1	1	3.5	1	10/40/20
	336	512	0.1	128	1	1	3.5	1	10/40/20
	720	512	0.1	128	1	1	3.5	1	10/40/20
Solar-Energy	96	512	0.1	16	1	1	3.5	1	10/40/20
	192	512	0.1	16	1	1	3.5	1	10/40/20
	336	512	0.1	16	1	1	3.5	1	10/40/20
	720	512	0.1	16	1	1	3.5	1	10/40/20

C Additional results

C.1 Selected architectures

Table 7 reports the task-specific encoder selected for each of the 32 dataset–horizon settings evaluated in Table 1. The seven-bit LayerNorm mask $\boldsymbol{u} = (u_1, \ldots, u_7)$ is read from left to right. Entries u_1–u_3 place LayerNorm before attention, after the attention residual, and inside the post-attention temporal branch, respectively. Entries u_4–u_7 place it after the first FFN projection, after activation and dropout, after the FFN residual, and inside the post-FFN temporal branch. For the binary fields u_i and s, 0 denotes disabled and 1 denotes enabled; s controls attention scaling, while h gives the number of attention heads. The fields k_{att} and k_{ffn} give the local-operator kernel sizes after attention and the FFN, respectively. Here, $k = 0$ disables the branch, whereas $k \in \{1, 3, 5\}$ selects a depthwise convolution with the indicated kernel size. Finally, g, r, and L_e denote the FFN activation, FFN expansion ratio, and encoder depth, respectively.

Table 7: Task-specific TiNAS architectures selected for the 32 dataset–horizon tasks in Table 1. Each row reports the final encoder configuration.

Task		Norm.	Attention		Local Operators		FFN		Depth
Dataset	**H**	$\boldsymbol{u}$	h	s	k_{att}	k_{ffn}	g	r	L_e
ETTm1	96	0000001	8	1	5	5	GELU	0.25	2
	192	0011100	16	1	5	0	ReLU	0.25	2
	336	0010010	4	1	5	5	LeakyReLU	0.25	4
	720	1000010	4	1	0	0	ReLU	2	2
ETTm2	96	0011000	16	1	5	1	Swish	4	1
	192	1010011	16	0	3	3	LeakyReLU	0.5	1
	336	1010011	16	0	3	3	LeakyReLU	0.5	1
	720	1100110	4	1	3	1	Swish	1	6
ETTh1	96	0010010	4	1	0	0	ReLU	1	6
	192	0011011	16	1	5	5	GELU	0.25	1
	336	0111010	16	1	0	0	ReLU	0.25	1
	720	0010001	16	1	1	1	Swish	4	2
ETTh2	96	0110000	16	1	5	5	Swish	4	6
	192	0010011	16	1	5	0	ReLU	0.25	1
	336	0000001	16	1	1	3	LeakyReLU	4	1
	720	0011011	16	1	5	1	ReLU	1	1
ECL	96	1011011	16	0	5	1	ReLU	2	6
	192	1010010	16	1	5	0	GELU	1	6
	336	0110011	8	1	0	3	ReLU	1	6
	720	0110101	16	1	5	3	GELU	1	6
Exchange	96	1011000	4	1	5	5	LeakyReLU	1	2
	192	0011010	4	0	0	1	LeakyReLU	0.25	1
	336	1000001	4	1	0	5	ReLU	1	1
	720	1011001	16	0	0	5	GELU	1	1
Weather	96	0011001	16	1	0	0	Swish	1	3
	192	1011010	16	1	5	1	GELU	1	6
	336	1111011	8	1	5	1	Swish	0.25	6
	720	1100100	16	0	0	0	LeakyReLU	0.5	3
Solar-Energy	96	1000000	16	1	5	1	ReLU	0.25	3
	192	1000001	16	1	5	0	LeakyReLU	2	6
	336	0110011	16	1	5	0	ReLU	4	6
	720	0010011	16	1	5	5	Swish	4	3

C.2 Controller ablation analysis

Table 8 reports the per-horizon results for the controller variants summarized in Table 2. The effect of each component varies with the forecasting task and horizon, but Full Controller yields the strongest aggregate performance across datasets. On Exchange at horizon 720, Full Controller achieves an MSE of 0.868, compared with 0.964 for w/o Action Semantics and 1.062 for w/o Previous Choices. w/o Performance History and w/o LoRA Updates obtain 0.891 and 1.030, respectively. In this setting, removing either previous choices or LoRA updates increases MSE relative to Full Controller. Individual ablations can still match or outperform Full Controller at isolated horizons. For example, w/o Action Semantics achieves an MSE of 0.520 on ETTh1-720, compared with 0.537 for Full Controller. The aggregate advantage therefore coexists with task- and horizon-specific exceptions.

C.3 Search Dynamics of Shortlisted Architectures

To examine search stability, we track the mean validation MSE of the top ten candidates at five periodic derivation rounds (search epochs 5, 10, 15, 20, and 25). At each round, 16 unique architectures are sampled, and the leading ten are evaluated on the full validation set using inherited supernet weights. As shown in Figure 7, ETTm1 and Weather exhibit relatively stable candidate performance across most horizons, although ETTm1-96 shows a moderate increase. Electricity, in contrast, exhibits gradual reductions at several horizons, indicating further improvements in candidate quality as search progresses. These trajectories describe task-dependent changes in inherited-weight validation

Table 8: Controller ablation results across forecast horizons (96, 192, 336, 720) with input length 96. Avg denotes the average over the four forecast horizons. For each forecast horizon and Avg row, the best result is shown in bold and the second-best distinct result is underlined; ties receive the same marking.

Controller		w/o Action Semantics		w/o Previous Choices		w/o Performance History		w/o LoRA Updates		Full Controller	
Dataset	**Horizon**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTm1	96	0.329	0.366	0.325	0.364	0.329	0.365	**0.320**	0.360	0.322	**0.359**
	192	0.370	0.388	0.367	0.386	0.368	**0.383**	0.369	0.386	**0.365**	0.384
	336	0.402	0.409	0.397	0.407	0.396	0.408	0.398	0.408	**0.395**	**0.406**
	720	0.462	0.443	0.458	0.443	0.460	**0.440**	**0.451**	**0.440**	0.459	0.447
	Avg	0.391	0.402	0.387	0.400	0.388	**0.399**	**0.385**	**0.399**	**0.385**	**0.399**
ETTh1	96	0.395	0.407	0.387	0.406	0.392	0.403	0.385	**0.400**	**0.382**	0.404
	192	0.434	0.434	0.437	0.432	0.432	**0.429**	**0.428**	**0.429**	0.429	0.430
	336	0.466	**0.446**	0.471	0.451	**0.463**	0.447	0.466	0.447	0.464	**0.446**
	720	**0.520**	**0.491**	0.551	0.503	0.578	0.508	0.551	0.506	0.537	0.496
	Avg	0.454	0.445	0.462	0.448	0.466	0.447	0.458	0.446	**0.453**	**0.444**
Exchange	96	**0.085**	**0.202**	0.086	0.204	0.089	0.206	**0.085**	0.203	0.086	0.206
	192	**0.171**	**0.294**	0.200	0.317	0.198	0.315	0.195	0.312	0.176	0.298
	336	**0.337**	**0.421**	0.366	0.440	0.340	0.423	0.343	0.425	0.343	0.425
	720	0.964	0.740	1.062	0.777	0.891	0.712	1.030	0.761	**0.868**	**0.706**
	Avg	0.389	0.414	0.429	0.435	0.380	0.414	0.413	0.425	**0.368**	**0.409**

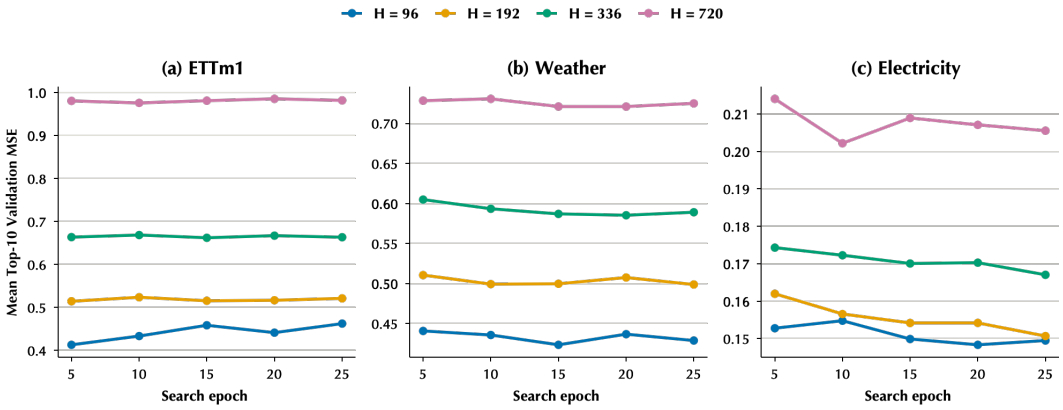

Figure 7: **Search dynamics of shortlisted architectures.** Mean validation MSE of the top ten candidates across derivation checkpoints on (a) ETTm1, (b) Weather, and (c) Electricity, evaluated with inherited supernet weights. Lower is better.

performance across derivation rounds. Across derivation rounds, the top ten candidate architectures can change, while continued supernet training updates the weights used to evaluate them. Changes in mean validation MSE therefore reflect both changes in the selected architectures and changes in their inherited weights.

C.4 Search space ablation analysis

Table 9 expands the average results in Table 3 across all forecast horizons. Near-Full Avg. averages five near-full search spaces at each horizon. Each space fixes one architectural group to the PatchTST-style reference configuration while retaining search over all remaining groups. Table 10 reports these five constituent restrictions separately. The restricted models remain close to Full, and a restriction

can occasionally yield a marginally better result at an isolated horizon. The small average gaps indicate limited sensitivity to these single-group restrictions in the evaluated tasks.

Table 9: Search space ablation across forecast horizons with input length 96. Near-Full Avg. averages the five single-group restrictions in Table 10. Parentheses list the groups searched by Macro and Module. Avg denotes the average over forecast horizons 96, 192, 336, and 720. For each forecast horizon and Avg row, the best result is shown in bold and the second-best distinct result is underlined; ties receive the same marking.

Search Space		Fixed		Macro (FFN+Local+Depth)		Module (Attention+FFN)		Near-Full Avg.		Full	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.334	0.371	0.328	0.364	0.323	0.364	0.326	0.362	**0.322**	**0.359**
	192	0.372	0.387	0.366	0.386	**0.365**	0.385	0.367	0.387	**0.365**	**0.384**
	336	0.400	0.411	0.406	0.411	0.399	0.408	0.396	**0.406**	**0.395**	**0.406**
	720	0.463	0.448	0.463	0.445	**0.457**	**0.441**	0.460	0.446	0.459	0.447
	Avg	0.392	0.404	0.391	0.402	0.386	0.400	0.387	0.400	**0.385**	**0.399**
ECL	96	0.162	0.249	0.161	0.248	0.160	**0.245**	0.161	0.247	**0.159**	**0.245**
	192	0.174	0.262	**0.170**	0.256	0.171	**0.255**	0.171	0.258	**0.170**	**0.255**
	336	0.189	0.278	0.189	0.276	**0.187**	**0.273**	0.189	0.276	0.189	0.274
	720	0.229	0.310	0.230	0.310	0.234	0.314	0.230	0.311	**0.228**	**0.309**
	Avg	0.189	0.275	0.188	0.273	0.188	0.272	0.188	0.273	**0.187**	**0.271**
Weather	96	**0.172**	**0.213**	0.173	**0.213**	0.176	0.214	0.175	0.215	0.173	**0.213**
	192	0.224	0.258	0.221	0.257	0.220	0.255	0.220	0.255	**0.219**	**0.254**
	336	0.277	0.298	0.277	0.298	**0.276**	**0.296**	**0.276**	**0.296**	0.278	0.297
	720	0.355	0.348	**0.352**	0.347	0.355	0.349	0.353	0.347	0.353	**0.346**
	Avg	0.257	0.279	**0.256**	0.279	0.257	0.279	**0.256**	**0.278**	**0.256**	**0.278**

C.5 Search strategy ablation analysis

We provide the complete results of the search strategy ablation in Table 11. Uniform Sampling replaces the autoregressive LLM controller while retaining the same search space and evaluation budget. w/o Periodic Derivation samples candidates only at the final search stage. w/o Candidate Accumulation performs intermediate derivations but excludes their candidates from final selection. Full progressively retains intermediate candidates for final selection. We further evaluate Full by applying either inherited-weight fine-tuning or independent retraining from scratch to the short-listed architectures. The horizon-wise results reveal that the advantage of the complete strategy is concentrated on the more challenging settings. In particular, on Exchange-720, Full + Retraining obtains an MSE of 0.868, compared with 0.934 for Uniform Sampling and 0.988 for w/o Candidate Accumulation. On Weather, by contrast, all variants remain within 0.002 average MSE, indicating that this dataset is relatively insensitive to the search-strategy choice. All variants use the same experimental settings and evaluation budget.

C.6 Efficiency Analysis

We measure the stage-level cost of architecture search independently on each of the eight forecasting benchmarks. All profiles use $L_{\text{in}} = L_{\text{out}} = 96$, $D = 512$, seed 2021, and a single NVIDIA RTX 4090 GPU. The batch size is 16 except on ECL, where it is 8. For each task, the profiler executes five consecutive warm-up epochs, five consecutive search epochs, and five repeated periodic-derivation calls; the first observation is retained in every summary. Within a search epoch, the shared-weight phase includes controller-based candidate-pool construction and one complete supernet-training epoch; it is followed by controller optimization under the fixed validation-feedback budget. We time the two phases separately and form the combined value by adding the paired measurements before computing the mean and sample standard deviation.

Table 10: Results for the five single-group restrictions underlying Near-Full Avg. Each variant fixes the named architectural group to the PatchTST-style reference configuration while searching all other groups. No Local fixes both local temporal branches as disabled. Full searches every group. For each forecast horizon and Avg row, the best result is shown in bold and the second-best distinct result is underlined; ties receive the same marking.

Restriction		Fixed Depth		Fixed Norm		Fixed Attention		No Local		Fixed FFN		Full	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.328	0.364	0.326	0.363	**0.320**	0.360	0.328	0.362	0.326	0.363	0.322	**0.359**
	192	0.377	0.392	**0.363**	**0.384**	0.364	0.385	0.367	0.387	0.365	0.385	0.365	**0.384**
	336	0.400	0.406	0.400	0.408	0.394	0.406	**0.393**	**0.404**	0.394	0.405	0.395	0.406
	720	0.464	0.449	0.457	0.445	0.463	0.449	0.461	0.445	**0.454**	**0.442**	0.459	0.447
	Avg	0.392	0.403	0.387	0.400	**0.385**	0.400	0.387	0.400	**0.385**	**0.399**	**0.385**	**0.399**
ECL	96	0.162	0.249	0.160	0.247	0.161	0.247	0.162	0.248	0.161	0.246	**0.159**	**0.245**
	192	0.172	0.259	0.171	**0.255**	0.170	**0.255**	0.171	0.267	**0.169**	**0.255**	0.170	**0.255**
	336	0.190	0.278	**0.187**	**0.273**	0.189	0.275	0.189	0.275	0.191	0.277	0.189	0.274
	720	0.232	0.312	0.230	0.311	**0.228**	0.310	0.230	0.312	0.230	0.311	**0.228**	**0.309**
	Avg	0.189	0.275	**0.187**	0.272	**0.187**	0.272	0.188	0.276	0.188	0.272	**0.187**	**0.271**
Weather	96	0.176	0.216	0.176	0.215	0.174	0.214	0.175	**0.213**	0.176	0.215	**0.173**	**0.213**
	192	0.220	0.255	0.220	0.255	0.220	0.255	0.221	0.256	0.220	0.255	**0.219**	**0.254**
	336	0.278	0.298	0.277	0.297	**0.275**	**0.295**	0.276	0.296	**0.275**	**0.295**	0.278	0.297
	720	0.353	**0.346**	0.356	0.348	0.353	0.347	0.353	**0.346**	**0.352**	**0.346**	0.353	**0.346**
	Avg	0.257	0.279	0.257	0.279	**0.256**	**0.278**	**0.256**	**0.278**	**0.256**	**0.278**	**0.256**	**0.278**

Table 11: Search strategy ablation across forecast horizons (96, 192, 336, 720) with input length 96. Avg denotes the average over the four forecast horizons. For each forecast horizon and Avg row, the best result is shown in bold and the second-best distinct result is underlined; ties receive the same marking.

Search Strategy		Uniform Sampling		w/o Periodic Derivation		w/o Candidate Accumulation		Full + Fine-tuning		Full + Retraining	
Dataset	Horizon	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.331	0.366	0.332	0.369	0.333	0.369	**0.321**	**0.359**	0.322	**0.359**
	192	0.373	0.389	0.376	0.394	0.366	**0.384**	**0.365**	0.387	**0.365**	**0.384**
	336	0.400	0.410	0.397	0.408	0.409	0.412	0.396	0.409	**0.395**	**0.406**
	720	0.463	**0.445**	0.461	0.447	0.464	0.446	0.466	0.451	**0.459**	0.447
	Avg	0.392	0.403	0.392	0.403	0.393	0.403	0.387	0.402	**0.385**	**0.399**
Exchange	96	0.088	0.210	0.084	0.202	**0.083**	**0.201**	0.085	0.206	0.086	0.206
	192	0.188	0.308	0.183	0.305	0.183	0.306	0.191	0.311	**0.176**	**0.298**
	336	**0.337**	**0.421**	0.347	0.426	0.339	0.423	0.349	0.426	0.343	0.425
	720	0.934	0.735	**0.865**	**0.700**	0.988	0.748	1.071	0.768	0.868	0.706
	Avg	0.387	0.419	0.370	**0.408**	0.398	0.420	0.424	0.428	**0.368**	0.409
Weather	96	**0.172**	0.214	0.175	0.217	0.174	0.215	0.173	**0.213**	0.173	**0.213**
	192	0.220	**0.254**	0.221	0.256	0.221	0.257	0.222	0.256	**0.219**	**0.254**
	336	0.277	0.297	0.276	0.296	0.276	0.297	**0.274**	**0.295**	0.278	0.297
	720	0.354	0.347	0.353	0.346	0.355	0.347	**0.352**	**0.345**	0.353	0.346
	Avg	0.256	0.278	0.256	0.279	0.257	0.279	**0.255**	**0.277**	0.256	0.278

The timing boundary includes data-loader iteration, device transfers, phase-mode switches, model computation, and metric reduction. It excludes component and dataset initialization, result serialization, post-search retraining or fine-tuning, and test evaluation. The repeated derivation calls use the same post-search model and controller state while resetting the initial candidate pool; they therefore profile current-round periodic derivation rather than final derivation over the accumulated historical pool. Consequently, Table 12 reports short-run stage measurements and does not extrapolate them into a total cost for the complete 40-epoch search and selection procedure.

Table 12: Stage-level wall-clock cost of TiNAS on the eight forecasting benchmarks. Values are mean $\pm$ sample standard deviation over five consecutive executions, with all observations retained. The combined search time is the paired sum of the shared-weight and controller-optimization phases and excludes periodic derivation.

Dataset	Batch	Warm-up (s/epoch)	Shared-weight phase (s/epoch)	Controller optimization (s/epoch)	Combined search (s/epoch)	Periodic derivation (s/call)
ETTh1	16	6.68 ± 0.39	36.31 ± 3.87	42.21 ± 1.96	78.52 ± 5.69	13.14 ± 0.55
ETTh2	16	6.29 ± 0.20	36.80 ± 3.34	42.97 ± 0.83	79.77 ± 3.33	13.15 ± 0.41
ETTm1	16	28.02 ± 0.46	59.29 ± 3.18	42.16 ± 1.68	101.45 ± 4.56	31.36 ± 1.87
ETTm2	16	27.90 ± 0.19	60.04 ± 3.24	42.37 ± 1.16	102.41 ± 4.05	29.97 ± 2.40
Exchange	16	3.68 ± 0.24	32.92 ± 4.01	42.04 ± 1.94	74.96 ± 5.04	9.20 ± 0.08
Weather	16	30.94 ± 0.74	64.18 ± 2.81	42.34 ± 1.06	106.52 ± 3.63	22.14 ± 1.25
ECL	8	188.64 ± 1.02	255.36 ± 11.31	43.15 ± 0.46	298.52 ± 11.58	136.47 ± 15.94
Solar-Energy	16	160.60 ± 0.88	222.90 ± 17.16	43.44 ± 0.60	266.34 ± 17.05	109.92 ± 11.34

The combined search epoch ranges from 74.96 ± 5.04 s on Exchange to 298.52 ± 11.58 s on ECL, and periodic derivation ranges from 9.20 ± 0.08 to 136.47 ± 15.94 s per call on the same tasks. In contrast to the task-dependent shared-weight phase, the controller-optimization phase occupies a narrow range of 42.04–43.44 s per epoch across all datasets, consistent with its fixed sampling and update budget. Because dataset size, channel count, and batch size are not normalized across tasks, the cross-dataset values characterize the actual task workloads rather than hardware-independent scaling.

Table 13 separately reports total and trainable parameter counts for the search-time components. The 19.718M parameters of the shared supernet are trainable. The controller contains 1,545.956M parameters in total, of which the frozen LLM backbone accounts for 1,543.714M. Only the 2.179M LoRA parameters and 0.063M action-head parameters are optimized, giving 2.242M trainable controller parameters, or 0.145% of the controller. The supernet and controller are used only for search and are not required to execute the selected forecasting architecture. The table therefore quantifies transient search-time parameter overhead; the size of the task-specific selected forecasting architecture is outside the parameter-counting configuration used for these profiles. Parameter counts are not used to infer peak memory, FLOPs, or inference latency.

Table 13: Parameter accounting for the search-time components of TiNAS. Counts are identical across the eight profiled tasks.

Component	Total (M)	Trainable (M)	Trainable (%)
Shared supernet	19.718	19.718	100.000
LLM backbone	1,543.714	0	0.000
LoRA adapters	2.179	2.179	100.000
Action heads	0.063	0.063	100.000
Controller total	1,545.956	2.242	0.145