Nuwa: Evaluation-Grounded Agentic Construction of Time Series Forecasting Systems

Fanda Fan[1*] **Yuxuan Yang**[1,2*] **Xiaorui Wang**[1,2] **Luqi Gong**[3] **Wentao Xiang**[1]
Xiang Ao[1] **Liming Song**[1] **Yinuo Zhu**[1] **Yutong Chen**[1] **Chunjie Luo**[1] **Jianfeng Zhan**[1,2†]
[1] Institute of Computing Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] Beijing University of Posts and Telecommunications
{fanfanda, aoxiang}@dojoduck.co {luochunjie, zhanjianfeng}@ict.ac.cn

Abstract

Time series forecasting has recently moved beyond static, single-pass prediction toward agentic methods that reason, use tools, execute workflows, and accumulate memory. However, most existing forecasting agents still optimize predictions or pipelines under largely implicit configurations, leaving the construction of the forecasting system itself underexplored. This limits their ability to attribute performance changes to specific decisions, such as which module, exogenous variable, or hyperparameter caused an improvement. We propose **Nuwa**, an evaluation-grounded agentic framework for constructing time series forecasting systems. Nuwa defines a structured design space over modules, exogenous variables, and hyperparameters, and performs Evaluation-Budgeted Construction that uses data meta-information, performance diagnostics, and memory to propose candidate sets, select budgeted subsets for controlled evaluation, attribute local effects, and decide whether to stop, accept, retry, switch subspace, or continue construction. The resulting design-level evidence is stored in a Design Memory, enabling reusable knowledge about which configurations work under which data conditions. Across approximately 15,000 experimental runs, Nuwa achieves consistent performance gains across module, exogenous-variable, and hyperparameter construction, with an overall average MSE reduction of 8.0% over the corresponding reference settings. By turning forecast-level feedback into attributable and transferable design evidence, Nuwa makes forecasting system construction an explicit objective for agentic time series forecasting. The project repository is available at https://github.com/BenchCouncil/Nuwa.

1 Introduction

Time series forecasting (TSF) underpins decision-making in domains such as energy, finance, healthcare, and transportation. The dominant research paradigm is model-centric: it improves forecasting by proposing new architectures or selecting among existing ones, while treating the surrounding pipeline as largely fixed. This paradigm has produced a rapid succession of specialized architectures [1, 2, 3, 4, 5, 6, 7], foundation models [8, 9, 10, 11, 12], and LLM-based forecasters [13, 14, 15]. Yet model-centric progress faces a structural limitation: forecasting accuracy often depends on factors outside the model architecture itself, including how inputs are transformed, which exogenous variables are included, and how training configurations are chosen. Recent large-scale studies show

*Equal contributions.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

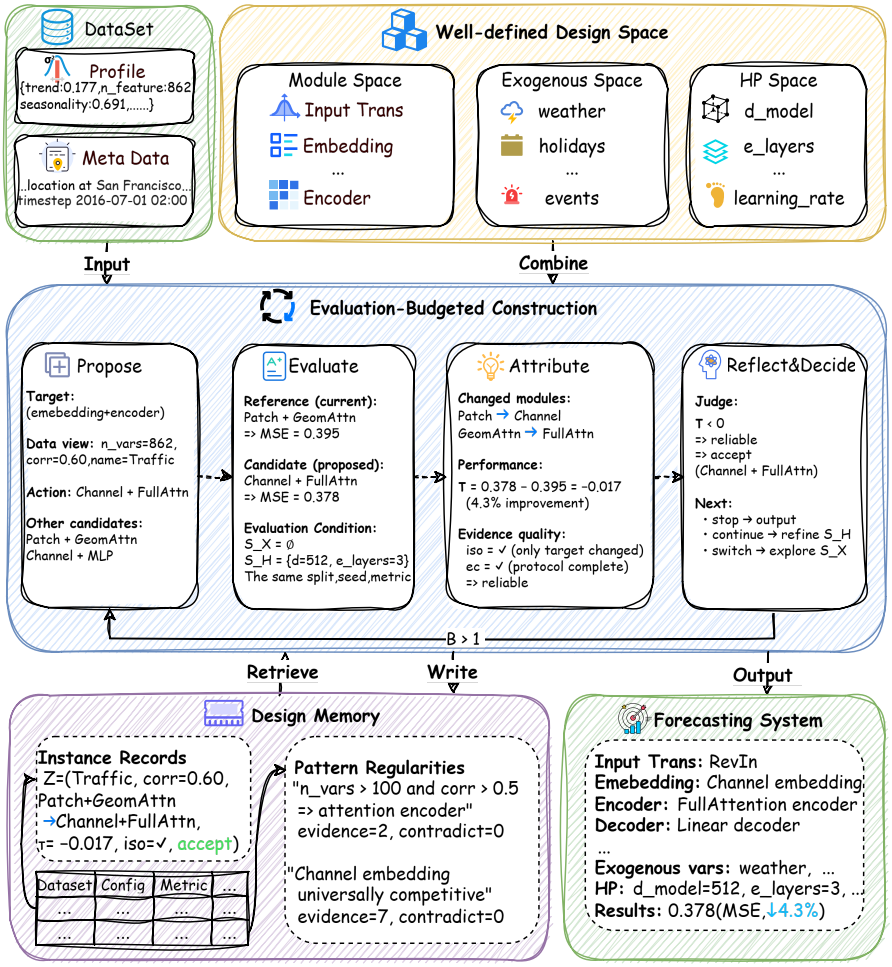

Figure 1: Overview of Nuwa. Nuwa constructs a time series forecasting system by operating over a structured design space, grounding each design action in controlled evaluation, attributing local effects, and using reflection to decide whether to stop or continue construction. Attributed traces are accumulated as reusable design-level evidence in memory.

that no single architecture consistently dominates across datasets and horizons [16, 17], and even hyperparameter choices can substantially alter method rankings [16]. These observations suggest that improving TSF requires reasoning beyond individual models.

Recent agentic TSF methods respond to this limitation by moving forecasting beyond static, single-pass prediction [18]. Instead of directly mapping historical observations to future values, they introduce multi-turn reasoning, tool use, workflow execution, reflection, and memory. Reasoning-centric agents such as AlphaCast [19] refine forecasts through contextual reasoning and reflective verification; workflow-centric agents such as CastFlow [20] coordinate planning, action, forecasting, and reflection; tool-centric systems such as TimeSeriesScientist [21] and AutoML-Agent [22] automate parts of the forecasting pipeline. These works mark an important shift from passive prediction toward active forecasting agents.

However, most existing agents still optimize the forecasting process rather than the forecasting system. Reasoning- and workflow-centric agents mainly refine predictions under a given forecasting setup, while tool-centric agents automate model-level pipelines whose design choices remain implicit and entangled. As a result, they optimize only part of the variables that determine performance. For example, ignoring exogenous variables can impose an inherent performance ceiling in externally driven time series [23], while treating an architecture as a monolithic model obscures the effects of input transformations, embeddings, encoders, decoders, and training configurations. This partial,

model-level view makes evaluation feedback coarse: an agent may observe that one run performs better than another, but cannot determine which design choice caused the improvement. Without a well-defined, modular design space, such feedback is difficult to attribute, difficult to reuse, and therefore weak as context for subsequent optimization.

We argue that the next step of agentic TSF is *evaluation-grounded forecasting system construction*. A forecasting agent should not only reason about future values, but also construct the forecasting system itself by selecting data, modules, and training configurations under controlled evaluation. In this view, evaluation is not merely a final assessment step; it is the harness that turns design actions into attributable evidence, enabling the agent to understand what worked, why it worked, and where the knowledge may transfer.

Based on this view, we propose **Nuwa**[3], an evaluation-grounded agentic framework for constructing time series forecasting systems. Nuwa consists of three components. First, a structured **Design Space** decomposes forecasting systems into controllable subspaces, including Module Space, Exogenous Space, and Hyperparameter Space. Second, **Evaluation-Budgeted Construction** iteratively performs PROPOSE → EVALUATE → ATTRIBUTE → REFLECT & DECIDE: it uses data meta-information, performance diagnostics, and memory to propose candidate sets in the three subspaces, selects budgeted subsets for controlled evaluation, and converts performance feedback into design-level evidence. Third, a **Design Memory** stores instance-level evaluation records and pattern-level regularities, enabling reusable knowledge about which design choices work under which data conditions. Together, these components make forecasting system construction attributable, controllable, and transferable across datasets.

Our contributions are summarized as follows:

- ❶ We propose evaluation-grounded forecasting system construction as a new direction for agentic TSF, making the construction of the system itself an explicit agent objective.
- ❷ We define a structured design space covering modules, exogenous variables, and hyperparameters, enabling forecasting systems to be manipulated and evaluated at a controllable granularity.
- ❸ We develop evaluation-budgeted construction that converts performance feedback into attributable design evidence through controlled evaluation, attribution, and evidence-conditioned reflection.
- ❹ We build a design memory that accumulates instance-level records and pattern-level regularities across datasets and demonstrate that Nuwa achieves strong one-step and multi-step construction compared with model-centric and agent-based baselines.

2 Related Work

Model-centric forecasting and system configuration. TSF progress has largely been architecture-centric, spanning patch and Transformer models [2, 1, 4], foundation models [8, 9, 10], and LLM-based forecasters [13, 14, 15]. Yet model-level rankings are unstable across datasets, horizons, and training configurations [16, 17]. This has motivated model recommendation and modular analysis. ARIES [24] recommends whole models from dataset relations, while TimeRecipe [25] benchmarks module combinations offline. Nuwa differs by making configuration an online design object: modules, exogenous variables, and hyperparameters are changed through controlled design actions whose effects are attributed during construction.

Agentic time series forecasting. Recent agentic TSF methods add reasoning, tool use, workflow execution, reflection, and memory to forecasting [18]. AlphaCast [19] and MemCast [26] emphasize reasoning or experience-conditioned prediction; CastFlow [20] and Cast-R1 [27] learn multi-step forecasting workflows; TimeSeriesScientist [21] and AutoML-Agent [22] automate analysis or ML pipelines. These systems make forecasting more adaptive, but feedback is usually attached to predictions, reasoning traces, tool trajectories, or whole pipelines. Nuwa instead attaches feedback to explicit design variables, enabling the agent to identify which data, module, or training choice changed performance.

[3] Named after Nüwa, the goddess in Chinese mythology who constructed the world—echoing our framework's goal of constructing forecasting systems.

Exogenous variables and data-centric forecasting. External factors are crucial for many forecasting tasks. TimeXer [4] incorporates exogenous variables into Transformer forecasting, Time-MMD [28] and Fidel-TS [29] broaden multimodal resources, and influence-aware forecasting shows that endogenous-only histories can impose a performance ceiling [23]. Data-centric agents such as DCATS [30] optimize data quality and metadata-driven cleaning. Nuwa treats exogenous variables as a controllable design subspace, not as fixed inputs, and evaluates them jointly with module and hyperparameter choices.

Evaluation-grounded construction. A shared gap is that forecasting feedback is seldom converted into reusable design evidence. AutoML and NAS can search configurations, but usually as black-box optimization; modular benchmarks expose correlations, but are offline; agentic methods provide planning and memory, but mostly for prediction refinement or pipeline automation. Nuwa combines these threads by defining a structured design space, evaluating controllable design actions under fixed conditions, attributing local effects to modular choices, and storing reliable traces in Design Memory for transfer across datasets.

3 Method

3.1 Problem Formulation and Overview

Given a time series dataset $\mathcal{D}$, Nuwa aims to construct a dataset-specific forecasting system rather than select a fixed model architecture. Unlike model-centric forecasting or agentic systems that optimize reasoning traces, tool-use workflows, or whole pipelines [18, 20, 27], Nuwa exposes the controllable variables inside the forecasting system and treats construction as evidence-conditioned design. At each step, the agent conditions on a data view $v_{\mathcal{D}}$, performance diagnostics g_t, and Design Memory $\mathcal{M}_t$; the full context definition is given in Appendix A.2.

We define the structured Design Space as $\mathcal{S} = \mathcal{S}_{\mathrm{M}} \times \mathcal{S}_{\mathrm{X}} \times \mathcal{S}_{\mathrm{H}}$, where a design instance $s = (s_{\mathrm{M}}, s_{\mathrm{X}}, s_{\mathrm{H}})$ specifies modules, exogenous variables, and hyperparameters, and instantiates a forecasting system F_s. The construction objective is

$$R(s; \mathcal{D}, \mathcal{E}) = \mathrm{Eval}(F_s; \mathcal{D}, \mathcal{E}), \qquad s^\star_{\mathcal{D}} \in \arg\min_{s \in \mathcal{S}} R(s; \mathcal{D}, \mathcal{E}). \tag{1}$$

Here R is a forecasting error or a cost-aware score measured under evaluation condition $\mathcal{E}$. Rather than solving this objective as black-box search, Nuwa constructs s through design actions whose local effects are evaluated, attributed, and stored as reusable evidence.

Following evaluatology [31], each comparison specifies the evaluated object (EO), affected object (AO), and evaluation condition (EC). In Nuwa, the EO is the target design variable or action in one subspace, the AO is the forecasting outcome on $\mathcal{D}$, and the EC fixes non-target subspaces and protocol choices. This makes feedback attributable: a measured improvement is tied to what changed and what was held fixed. Appendix A.7 gives the full EC specification.

Figure 1 summarizes the framework. Given budget B, Nuwa proposes candidates in one target subspace, evaluates selected actions under controlled conditions, attributes local effects, and reflects to accept, retry, switch subspace, stop, or continue. One-step construction stops after the first evaluated candidate set; multi-step construction feeds attributed evidence and updated diagnostics into later proposals. The resulting traces are written to Design Memory for cold-start construction and iterative refinement. Additional details on action families, subset selection, attribution, memory schema, and skill interfaces are provided in Appendix A.3–A.11.

3.2 Structured Design Space

The Design Space defines what the agent can change and what can be held fixed during evaluation. This explicit structure is necessary for attributable feedback: an observed improvement is only useful if it can be attached to a known design variable. Nuwa uses three subspaces.

Module Space. The Module Space follows the modular decomposition of CombinationTS [32]. A forecasting model is expressed as a five-stage pipeline:

$$F_s = \mathcal{T}_{\mathrm{out}}^{-1} \circ \mathcal{G} \circ \Phi \circ \mathcal{B} \circ \mathcal{T}_{\mathrm{in}}, \tag{2}$$

Table 1: Module-level decomposition of representative forecasting models and two Nuwa-constructed systems (★). The five existing models are chosen to maximize architectural diversity; Nuwa-built rows are dataset-specific configurations discovered during evaluation-grounded construction. The full decomposition table covering all baselines is in Appendix A.3.2.

System	Input Trans.	Embedding	Encoder	Decoder	Output Trans.
DLinear [1]	trend-seasonal decomp.	None	None	Linear	branch aggregation
PatchTST [2]	norm.	Patch	Transformer	Linear	de-norm.
iTransformer [3]	norm.	Variate	Transformer	Linear	de-norm.
TimeMixer [34]	trend-seasonal + multi-scale	Point	Multi-branch MLP	FMM	branch aggregation
TimeFilter [7]	norm.	Patch	GNN + MLP	Linear	de-norm.
Nuwa-built (Weather) ★	RevIN + Trend-Seasonal Decomp.	Patch-wise	Geometric Attention	Linear	branch agg.
Nuwa-built (Solar) ★	RevIN + Multi-Scale Decomp.	Patch-wise	Geometric Attention	Linear	branch agg.

where $\mathcal{T}_{\text{in}}$ is Input Transformation, $\mathcal{B}$ is Embedding, Φ is Encoder, $\mathcal{G}$ is Decoder, and $\mathcal{T}_{\text{out}}^{-1}$ is Output Transformation. Representative design choices at each stage include (but are not limited to): *Input Transformation*—normalization, trend-seasonal decomposition, multi-scale decomposition, cycle-residual decomposition, or covariate alignment; *Embedding*—None, Point-wise, Patch-wise, or Variate-wise [33]; *Encoder*—Transformer, MLP, multiscale mixing, or frequency-domain filtering; *Decoder*—Linear projection, convolutional linear, or future multi-mixing; *Output Transformation*—de-normalization, rescaling, or branch aggregation. The design space is open: any new forecasting method can be decomposed into this five-stage view, and its modules can be incorporated as additional options without changing the construction procedure. This decomposition makes module-level changes addressable: the agent can replace one component while keeping other modules, exogenous variables, and hyperparameters fixed. A fuller module catalog and the module-replacement interface are given in Appendix A.3. Table 1 illustrates how representative forecasting models map into this space.

Exogenous Space. The Exogenous Space contains candidate external variables that may influence the target series. Unlike methods that assume a fixed set of covariates, Nuwa treats exogenous variables as open-world design variables sourced from real-world data. The agent operates on $\mathcal{S}_{\text{X}}$ through typed operations: *Discover*—identify candidate variables from external sources using domain knowledge and LLM priors; *Align*—resolve temporal mismatches, lag relations, and frequency differences; *Add*—include a candidate in the forecasting system; *Filter*—screen a variable group for redundancy or collinearity; *Remove*—drop a variable after controlled evaluation shows no benefit. Each exogenous candidate must specify its source, forecast-time availability, temporal alignment rule, expected lag relation, leakage risk, and whether it is evaluated alone or as part of a bounded variable group. The full operation definitions are given in Appendix A.3.

Hyperparameter Space. The Hyperparameter Space contains model and training configurations that can change both model ranking and training stability. Nuwa treats them as first-class design variables organized into bounded groups: *Architecture*—hidden dimension, number of layers, attention heads, patch length; *Optimization*—learning rate, scheduler, batch size, dropout rate; *Protocol*—look-back length, training epochs, early-stopping patience. Nuwa evaluates bounded hyperparameter actions under fixed module and exogenous-variable settings, preferring small coupled groups (e.g., learning rate and scheduler together) over unconstrained grid search to preserve attribution.

3.3 Evaluation-Budgeted Construction

Nuwa maintains an incumbent configuration s_t and constructs alternatives under an evaluation budget B through four stages: *Propose*, *Evaluate*, *Attribute*, and *Reflect & Decide*. The LLM proposes typed actions from data characteristics, validation diagnostics, and Design Memory; deterministic procedures check feasibility, rank candidates, compute evidence quality, and enforce acceptance rules. Each round produces both a forecasting configuration and evidence specifying what changed, what was held fixed, and its measured effect. The incumbent remains the reference throughout a round, so all candidates in that round are compared against the same starting system.

Typed proposal and budgeted selection. At step t, the context is $c_t = (v_{\mathcal{D}}, g_t, s_t, \mathcal{M}_t, e_{<t})$, where $v_{\mathcal{D}}$ uses training data and g_t summarizes available training and validation diagnostics. For a selected subspace $k_t \in \{\mathrm{M}, \mathrm{X}, \mathrm{H}\}$, the LLM proposes

$$\mathcal{C}_t^{k_t} = \mathrm{Propose}_{k_t}(c_t), \qquad a = (k_t, \delta) \in \mathcal{C}_t^{k_t}. \tag{3}$$

Each action declares a configuration change, the non-target settings to preserve, and the diagnostic or retrieved evidence motivating it. Compatibility and availability checks remove infeasible actions and leakage-prone exogenous inputs before ranking. Among admissible proposals, Nuwa selects a budgeted subset:

$$\mathcal{U}_t^{k_t} \subseteq \mathcal{C}_t^{k_t}, \qquad |\mathcal{U}_t^{k_t}| \le b_t, \qquad \sum_t b_t \le B. \tag{4}$$

Selection uses a deterministic acquisition policy inspired by sequential model-based optimization and adaptive allocation [35, 36, 37]:

$$\alpha_t(a) = \hat{G}_t(a \mid c_t, \mathcal{M}_t) + \lambda U_t(a \mid \mathcal{M}_t) - \gamma C_t(a) - \eta L_t(a). \tag{5}$$

Here $\hat{G}_t$ estimates improvement from comparable memory records, U_t favors uncertainty or novelty, C_t accounts for evaluation cost, and L_t captures implementation risk among admissible actions. The score allocates evaluations; measured outcomes determine acceptance. The execution policy fixes scoring parameters and tie-breaking, with redundancy-aware selection for exogenous groups (Appendix A.6).

Controlled evaluation. Applying an action gives $s'_{t,j} = \mathrm{Apply}(s_t, a_{t,j})$. Before execution, the resolved configuration difference is checked against the declared action and evaluation condition $\mathcal{E}_{t,j} = (s_{\neg k_t}, p_t)$. The protocol p_t fixes split boundaries, horizon, metric, preprocessing, seed policy, and non-target training settings. If lookback or training budget is itself the target HP, that field is the declared intervention; the remaining conditions stay fixed. Nuwa compares validation outcomes

$$R_t = R_{\mathrm{val}}(s_t; \mathcal{D}, \mathcal{E}_{t,j}), \qquad R'_{t,j} = R_{\mathrm{val}}(s'_{t,j}; \mathcal{D}, \mathcal{E}_{t,j}). \tag{6}$$

Reference results can be reused when configuration, data version, split, seed, and training fidelity match exactly. Validation feedback determines checkpoint selection, construction decisions, and stopping; the test split is evaluated after the final configuration is frozen. Full EC requirements are given in Appendix A.7.

Local attribution and evidence quality. The evaluated object is the declared action, and the affected object is the forecasting outcome under the fixed EC. Nuwa records its local effect as

$$\tau_{t,j}(a_{t,j}; \mathcal{D}, \mathcal{E}_{t,j}) = R'_{t,j} - R_t. \tag{7}$$

Negative values indicate improvement for loss metrics. The effect applies to the stated configuration and conditions [31], extending module-level controlled benchmarking [25, 32] to online construction. A coupled action receives a joint effect; changing both embedding and width, for example, does not isolate the embedding's contribution.

Each comparison also yields $q_{t,j} = (n, \bar{\tau}, \sigma, \mathrm{iso}, \mathrm{ec}, \mathrm{leak}, \mathrm{valid})$. Configuration differences determine action isolation; protocol metadata determine EC agreement; availability metadata determine leakage status; execution logs determine validity. Repeated, seed-paired comparisons provide the count n, mean effect $\bar{\tau}$, and effect dispersion σ; a single comparison retains an unestimated dispersion. These fields are computed from records and govern whether evidence supports acceptance and pattern aggregation (Appendix A.8).

Evidence-gated reflection. Reflection uses measured effects, quality tags, and diagnostics to update the incumbent and plan further work. Let $\mathcal{A}_t = \{j : \mathrm{Eligible}(q_{t,j}) \wedge \bar{\tau}_{t,j} < -\epsilon\}$ contain candidates satisfying isolation, EC, availability, execution, and reliability requirements, together with the improvement margin ϵ. Writing $\bar{R}'_{t,j}$ for the candidate's mean validation loss over the paired evaluations, the update is

$$s_{t+1} = \begin{cases} s'_{t,j^\star}, & j^\star = \arg\min_{j \in \mathcal{A}_t} \bar{R}'_{t,j}, \quad \mathcal{A}_t \neq \varnothing, \\ s_t, & \mathcal{A}_t = \varnothing. \end{cases} \tag{8}$$

Thus selection is among eligible improvements, and an unreliable low-loss trial does not displace a reliable candidate. The policy fixes reliability criteria and the improvement margin before evaluating the proposals. Promising but inconclusive comparisons can receive additional verification; persistent diagnostics can redirect proposals to another subspace. For example, residual autocorrelation can motivate a lookback or decomposition action, while training instability can motivate optimization HPs. These signals guide the next comparison; attribution remains attached to the action actually evaluated. Switching subspaces preserves the incumbent until another candidate passes Eq. 8.

Feedback, memory, and termination. Every round records the evaluated actions, conditions, effects, quality tags, and rationale ρ_t, including rejected trials. If construction continues, the next context combines these records with diagnostics of the retained incumbent and the updated memory; later proposals can revise the hypotheses used in earlier rounds. One-step construction stops after the first evaluated candidate set, whereas multi-step construction allocates further budget after observing feedback. Construction stops when the budget is exhausted or the policy's stopping condition is met. The budget unit is specified per experiment; candidate counts are reported separately from reference fits, seed repetitions, failed runs, cache hits, and LLM cost. The full procedure and decision rules appear in Appendices A.4 and A.9; Appendix A.11 specifies the eight interfaces implementing this division of responsibilities.

3.4 Design Memory

Design Memory turns controlled comparisons into reusable construction evidence.[4] We write $\mathcal{M}_t = (\mathcal{Z}_t, \mathcal{P}_t)$, where $\mathcal{Z}_t$ stores instance-level records and $\mathcal{P}_t$ aggregates conditional design preferences. Retrieval uses the target data view and current diagnostics to identify relevant actions and the conditions under which they were effective.

Instance-level records. Each comparison writes

$$z_{t,j} = (v_{\mathcal{D}}, g_t, s_t \to s'_{t,j}, a_{t,j}, \mathcal{E}_{t,j}, R_t, R'_{t,j}, \tau_{t,j}, q_{t,j}, \rho_t), \tag{9}$$

which binds the changed action to its reference, measured outcome, fixed conditions, and evidence quality. Reliable improvements and reliable negative results both inform retrieval: the former support candidate actions, while the latter narrow their applicable conditions. Failed or incomplete comparisons retain their status for diagnosis and retry decisions, without contributing an estimated improvement to pattern aggregation. The full schema is in Appendix A.10.

Pattern-level regularities. Patterns summarize comparable, quality-qualified records into conditional preferences over modules, variables, and HPs. Their support and reliability are updated as new comparisons confirm or contradict a preference; contradictory evidence can narrow a pattern's scope rather than erase the underlying observations. Patterns influence proposals and expected-gain estimates, while acceptance remains governed by target validation evidence and Eq. 8. This connects experience-conditioned agentic forecasting [26] with reusable design-evaluation evidence.

Memory isolation. For transfer evaluation on a target dataset, dataset-masked memory removes all instance records originating from that dataset before retrieval. Pattern summaries, support counts, reliability estimates, and contradiction flags are then recomputed from the retained records. This keeps both memory levels consistent with the same source evidence while allowing newly acquired target validation feedback to guide subsequent construction steps.

4 Experiments

We evaluate Nuwa's three design subspaces and Design Memory through five research questions: **(RQ1) Module Space construction:** Can Nuwa construct forecasting systems that compete with fixed models and existing forecasting agents? (cf. Table 2) **(RQ2) Hyperparameter Space construction:** Can Nuwa find competitive hyperparameter configurations with fewer candidate evaluations than exhaustive search? (cf. Table 3) **(RQ3) Exogenous Space construction:** Can Nuwa retrieve useful exogenous variables and improve forecasting by filtering harmful or redundant candidates? (cf.

[4] We have uploaded Nuwa's Design Memory to the DojoDuck platform at https://dojoduck.co/.

Table 2: Module Space construction results (CastFlow-style grouped metrics [20]). **Bold**: best; underline: second best (ties included). Lower is better.

	Short-term Forecasting										Long-term Forecasting									
Dataset	BE		DE		NP		FR		PJM		ETTh		ETTm		WP		Solar		MOPEX	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Prophet	992.90	16.94	320.63	13.25	57.79	5.00	1035.70	14.42	52.26	5.53	46.70	4.59	21.18	3.11	8071.24	56.06	64.81	6.44	12.40	2.50
ARIMA	869.14	17.04	372.91	11.82	55.51	4.45	964.00	13.22	33.02	4.00	10.88	2.43	2.71	1.17	2106.64	28.77	80.48	4.60	9.19	1.85
XGBoost [41]	1344.39	16.44	227.01	10.21	28.31	3.24	1044.82	10.14	30.36	3.86	13.37	2.96	3.41	1.37	2384.63	37.99	33.85	3.50	6.91	1.68
LightGBM [40]	1168.62	15.71	231.44	10.21	30.92	3.38	933.84	9.96	29.63	3.88	12.22	2.75	2.91	1.26	2290.27	34.81	29.50	2.80	9.63	1.84
Autoformer [39]	891.02	16.62	331.37	13.33	46.11	4.86	934.55	13.37	77.59	6.43	11.60	2.64	4.23	1.57	2568.13	37.01	39.14	4.34	6.17	1.96
DLinear [1]	658.66	12.84	239.98	12.84	32.22	3.84	811.62	10.45	42.16	4.65	8.51	2.24	2.63	1.14	1932.59	29.33	19.06	2.74	5.26	1.33
PatchTST [2]	627.27	11.44	208.93	10.04	24.64	3.27	797.42	9.13	31.88	4.17	8.40	2.17	2.62	1.11	2270.10	32.47	19.44	2.78	5.36	1.44
iTransformer [3]	606.71	11.25	230.05	10.23	27.10	3.44	940.51	11.10	35.14	4.37	8.31	2.14	3.14	1.28	2064.60	29.90	17.96	2.67	4.88	1.34
TimeXer [4]	703.00	10.70	252.05	10.26	27.31	3.44	834.86	9.38	25.88	3.65	8.77	2.20	2.56	1.11	2170.25	31.92	20.12	2.92	4.83	1.28
ConvTimeNet [38]	621.73	10.98	231.26	10.80	28.13	3.62	882.85	8.97	28.71	3.79	8.62	2.18	2.71	1.16	2081.37	28.52	23.22	3.10	5.07	1.29
Chronos [8]	625.88	9.63	223.24	10.00	24.39	3.21	812.40	8.24	**25.70**	**3.63**	9.40	2.25	2.75	1.18	2269.43	29.63	79.28	4.57	5.29	1.31
TimesFM [42]	784.98	10.92	219.57	9.98	25.11	3.20	870.17	8.84	33.99	4.14	9.20	2.26	2.78	1.15	2228.19	29.84	45.91	3.37	5.58	1.35
Sundial [10]	651.50	11.00	264.72	10.71	28.68	3.36	942.46	8.54	30.72	3.99	9.44	2.32	3.21	1.24	2168.81	31.22	100.40	6.61	5.15	1.42
Time-LLM [13]	729.68	11.88	253.97	10.85	33.05	3.71	936.46	11.51	35.84	4.32	8.04	2.08	2.57	1.13	2833.24	38.05	17.25	2.68	5.77	1.44
PromptCast [43]	824.27	13.38	293.24	10.93	43.16	3.93	1105.16	14.42	47.62	5.58	9.46	2.82	4.25	1.60	2626.89	36.20	18.21	2.84	6.70	1.59
TokenCast [44]	654.38	11.48	232.21	10.84	31.49	3.59	963.09	11.82	32.37	3.92	8.28	2.17	2.50	1.10	2521.39	34.29	18.13	2.74	5.52	1.39
S^2IP-LLM [45]	701.24	11.89	313.82	11.15	29.67	3.42	872.55	10.93	35.67	4.16	9.68	2.39	3.01	1.29	2497.38	33.13	18.46	2.96	6.29	1.58
TimeReasoner [46]	691.28	11.62	272.19	10.77	31.27	3.62	899.47	11.01	34.38	4.22	9.14	2.18	4.04	1.57	2228.24	31.57	20.91	2.98	5.92	1.46
Time-R1 [47]	635.23	10.23	210.28	9.89	26.28	3.35	842.19	10.92	27.35	3.80	8.02	2.04	2.80	1.11	2054.88	30.13	17.42	2.67	5.83	1.35
TimeSeriesScientist [21]	906.68	14.94	245.40	10.39	46.31	4.16	904.65	12.71	45.40	5.39	9.09	2.24	5.28	1.61	2551.19	33.59	25.82	3.14	5.60	1.42
CastMind [19]	653.98	10.71	248.98	10.23	24.16	3.19	804.89	9.10	26.53	3.74	8.47	2.21	2.48	1.16	2131.02	30.22	18.89	2.80	5.19	1.43
CastFlow [20]	**546.87**	**9.60**	200.47	9.50	23.92	3.11	707.96	8.16	27.45	3.74	8.00	2.03	**2.36**	**1.04**	1719.99	27.40	16.90	2.50	3.60	1.19
NUWA	603.41	10.03	**190.01**	**9.41**	**22.81**	**2.75**	**683.35**	**8.10**	**25.70**	**3.63**	**7.90**	**2.01**	2.44	1.08	**1703.10**	**26.71**	**16.65**	**2.46**	**3.46**	**1.04**

Table 4) **(RQ4) Design Memory ablation and scaling:** How do memory components and memory coverage affect construction quality? (cf. Figure 2) **(RQ5) Pattern-level knowledge generation:** What design patterns can Nuwa distill from evaluation traces across the three subspaces? (cf. Appendix B.4, Table 11)

Experimental protocols are detailed in Appendices B.1–B.2.

Datasets and Baselines. We evaluate on five short-term EPF markets (BE, DE, NP, FR, PJM) and five long-term benchmarks (ETTh, ETTm, Weather, Solar, MOPEX), using the task-specific protocols in Appendix B.1 [20]. Baselines span statistical models (Prophet, ARIMA), deep learning architectures (DLinear [1], PatchTST [2], iTransformer [3], TimeXer [4], ConvTimeNet [38], Autoformer [39], LightGBM [40], XGBoost [41]), and foundation/agentic systems (Chronos [8], TimesFM [42], Sundial [10], Time-LLM [13], PromptCast [43], TokenCast [44], S^2IP-LLM [45], TimeReasoner [46], Time-R1 [47], TimeSeriesScientist [21], CastMind [19], CastFlow [20]).

Implementation. The original experimental campaign used NVIDIA V100 GPUs (4000 GPU-hours, 14,953 total tasks) and Claude Sonnet 4.6 (`claude-sonnet-4-6`, temperature 0.1) for agentic decisions. We report MSE and MAE following the CastFlow grouped-metrics protocol [20]; module options follow CombinationTS [32] with additional encoder types. Training uses up to 100 epochs with early stopping patience 7; full protocol details are in Appendix B.1.

Selection and reporting. In construction experiments, training data determine preprocessing and data views; validation MSE determines checkpoints, candidate acceptance, and stopping. The selected configuration is frozen before final test reporting. Within each comparison, the horizon, split, metric scale, and non-target training conditions are held fixed; lookback and training settings vary only when included in the declared search space. Dataset-masked memory removes target-origin records and recomputes their derived patterns.

RQ1: Module Space Construction. Table 2 shows that Nuwa-constructed modular forecasting systems are competitive with or surpass both fixed forecasting models and existing agentic forecasting systems under the CastFlow-style evaluation protocol (Appendix B.1), achieving the best or tied-best MSE on four of five EPF markets and four of five long-term datasets; Nuwa's advantage is strongest on datasets where the optimal module configuration differs from any single baseline identity, confirming that evaluation-grounded construction can explore a broader design space than selecting among fixed models, and extended multivariate long-term results under the same protocol are reported in Appendix B.3.

RQ2: Hyperparameter Space Construction. Table 3 shows that Nuwa obtains competitive hyperparameter configurations at substantially lower evaluation cost than exhaustive search: multi-

Table 3: Hyperparameter Space construction under fixed module configurations. Cost is the number of evaluated candidates. Nuwa one-step and multi-step settings are marked by ♠ and ◆, respectively. **Bold**: best; underline: second best (among non-exhaustive methods). Lower is better.

Model	Method	ETTh1			Weather			Solar		
		MSE	MAE	Cost	MSE	MAE	Cost	MSE	MAE	Cost
PatchTST	No Search	0.3936	0.4042	1	0.1780	0.2195	1	0.2379	0.2780	1
	Nuwa (one-step) ♠	0.3863	0.4074	1	0.1612	0.2103	1	0.2347	0.2765	1
	Nuwa (multi-step) ◆	**0.3764**	**0.3983**	8	**0.1552**	**0.2041**	8	**0.2019**	**0.2493**	8
	Exhaustive Grid	0.3764	0.3983	48	0.1552	0.2041	48	0.2019	0.2493	48
SimpleTM	No Search	0.3801	0.4021	1	0.1666	0.2108	1	0.2174	0.2720	1
	Nuwa (one-step) ♠	0.3801	0.4021	1	0.1488	0.2010	1	0.2031	0.2555	1
	Nuwa (multi-step) ◆	**0.3784**	**0.3985**	3	**0.1450**	**0.1951**	6	**0.1973**	**0.2539**	11
	Exhaustive Grid	0.3731	0.3943	48	0.1450	0.1951	48	0.1972	0.2542	48

step Nuwa matches the exhaustive-grid result (48 candidates, search space defined in Appendix B.1) on Weather and Solar using only 8 evaluations for PatchTST, and matches or nearly matches the grid optimum for SimpleTM with 3–11 evaluations, demonstrating that Nuwa does not enumerate the Cartesian product but spends evaluation budget on a small set of candidates selected from data diagnostics and memory.

RQ3: Exogenous Space Construction. Table 4 shows that Nuwa's retrieval-and-screening pipeline consistently discovers useful exogenous variables while rejecting harmful or redundant ones: **Nuwa (Filter)** achieves the best or tied-best MSE on all five EPF markets and all three non-EPF datasets, while **Nuwa (NoFilter)** is uniformly worse than the filtered setting, confirming that unfiltered open-world retrieval introduces noise that degrades forecasting quality; on EPF DE and EPF FR, expert-selected variables approach the filtered quality, but Nuwa's automated screening matches or exceeds them without domain expertise.

Table 4: Exogenous Space construction results. Each cell lists MSE above MAE. Lower is better.

Dataset	Nuwa (Filter)	Nuwa (NoFilter)	Timestamp	Expert Exog.
EPF BE	**0.401** **0.257**	0.425 0.271	0.420 0.263	0.416 0.263
EPF DE	**0.525** 0.470	0.608 0.498	0.585 0.484	0.546 **0.452**
EPF FR	**0.420** 0.229	0.432 0.236	0.436 **0.225**	0.460 0.244
EPF NP	**0.264** **0.290**	0.298 0.318	0.309 0.314	0.267 0.291
EPF PJM	**0.110** **0.209**	0.127 0.230	0.129 0.227	**0.110** 0.210
Traffic	**0.139** **0.222**	– –	0.147 0.226	– –
Solar1h	**0.283** **0.323**	– –	0.309 0.334	– –
Exchange	**0.118** **0.257**	– –	0.120 0.258	– –

RQ4: Design Memory Ablation and Scaling. We compare No Memory and Full Memory with random retention of 25% or 50% of records, module masking that removes evidence about the current model family, and dataset masking that removes target-origin records. Masks are applied before pattern aggregation, so retrieved records and derived patterns use the same retained evidence pool (Appendix B.1). Figure 2 shows that full Design Memory obtains the best overall average rank (1.50, versus 5.67 without memory), although it is not uniformly best on every individual metric: this is expected because Design Memory stores transferable evidence rather than a deterministic lookup table of best scores, and some masked settings can match or even improve individual metrics when the remaining evidence transfers well to the target dataset; the aggregate trend confirms that removing memory or masking relevant evidence generally degrades the quality of proposed configurations while full memory provides the most reliable construction prior.

RQ5: Pattern-Level Knowledge Generation. Nuwa distills pattern-level knowledge from accumulated evaluation traces across the three design subspaces (full tables with evidence counts are in Appendix B.4): in the Module and Hyperparameter Space, patch embedding with no encoder is strongly preferred for low-dimensional, high-trend data, while Geometric Attention dominates on high-correlation, high-dimensional series, and $\text{seq_len} = 336$ with $\text{lr} = 0.001$ are broadly

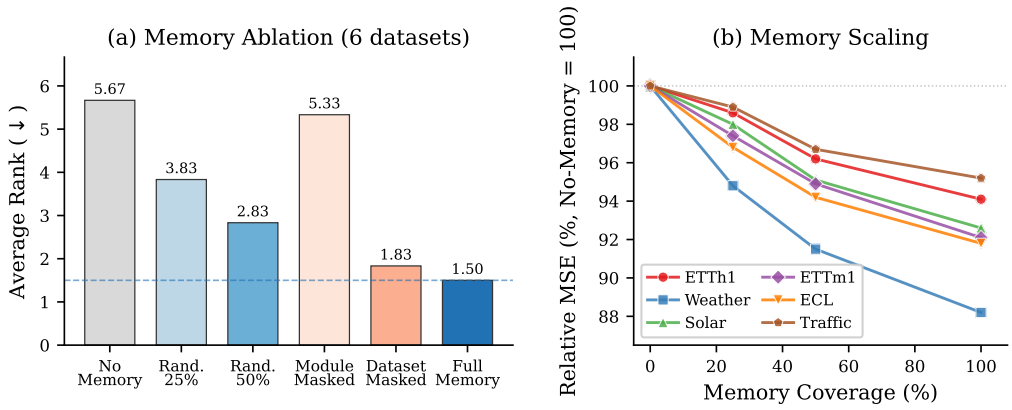

Figure 2: Design Memory ablation and scaling across six datasets. (a) Average rank under different memory settings; lower is better. (b) Per-dataset relative MSE as memory coverage increases from 0% to 100%; No-Memory baseline is normalized to 100.

reliable defaults; in the Exogenous Space, filtering retrieved candidates to 1–2 variables and prioritizing market-internal forecasts yield the strongest signals, while unfiltered open-world retrieval is consistently identified as a harm pattern.

5 Limitations and Future Work

Attribution reliability. Nuwa attributes performance changes to design actions by fixing the non-target subspaces and evaluation protocol. This turns performance feedback into reusable design evidence, but the reliability of the local effect $\tau_{t,j}$ depends on the quality of the evaluation condition. Inconsistent splits, preprocessing, seed policy, training budget, or exogenous-variable availability can mix the target action with unrelated variation, while leakage or timestamp misalignment may create misleading gains. Nuwa records evidence quality $q_{t,j}$ and treats incomplete comparisons as low-confidence evidence. Future work should strengthen automatic evaluation-condition auditing, uncertainty-aware attribution, and adaptive verification.

Memory transfer. Design Memory improves construction by reusing instance-level records and pattern-level regularities, but its benefit depends on memory coverage and the similarity between previous tasks and the current dataset. Out-of-domain retrieval can weaken proposal quality, and overly broad pattern transfer may bias candidate selection. The current memory stores data views, diagnostics, evaluation conditions, evidence quality, and contradiction flags, but it is still mainly symbolic design evidence. Future work can extend memory to reusable model states for foundation-model-based forecasting systems, including checkpoints, adapters, prompt states, calibration heads, and dataset-specific fine-tuning traces. This would connect evaluation-grounded construction with continual learning, allowing Nuwa to accumulate both declarative design knowledge and executable model experience.

6 Conclusion

We presented Nuwa, an evaluation-grounded agentic framework for constructing time series forecasting systems. Nuwa makes the forecasting system itself the object of construction by defining a structured design space over modules, exogenous variables, and hyperparameters. Through Evaluation-Budgeted Construction, it proposes candidate actions, evaluates them under controlled conditions, attributes local effects, and stores the resulting traces in Design Memory as reusable design-level evidence. This formulation turns performance feedback into knowledge about which modules, variables, and training configurations work under which data conditions. By connecting agentic proposal, controlled evaluation, attribution, and memory, Nuwa provides a path toward forecasting agents that not only predict future values, but also learn how to build better forecasting systems over time.

Acknowledgments and Disclosure of Funding

Funding. This work was supported by the Innovation Funding of ICT, CAS under Grant No. E661030001.

Competing interests. This work used the DojoDuck platform. Fanda Fan and Xiang Ao are core developers of DojoDuck.

References

[1] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022. URL https://arxiv.org/abs/2205.13504.

[2] 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*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

[3] 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*, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

[4] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=INAeUQ04lT.

[5] Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. Duet: Dual clustering enhanced multivariate time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '25, page 1185–1196, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712456. doi: 10.1145/3690624.3709325. URL https://doi.org/10.1145/3690624.3709325.

[6] Hui Chen, Viet Luong, Lopamudra Mukherjee, and Vikas Singh. SimpleTM: A simple baseline for multivariate time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oANkBaVci5.

[7] Yifan Hu, Guibin Zhang, Peiyuan Liu, Disen Lan, Naiqi Li, Dawei Cheng, Tao Dai, Shu-Tao Xia, and Shirui Pan. Timefilter: Patch-specific spatial-temporal graph filtration for time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=490VcNtjh7.

[8] Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR. Expert Certification.

[9] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=e1wDDFmlVu.

[10] Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=LO7ciRpjI5.

[11] Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

[12] Yunshi Wen, Wesley M. Gifford, Chandra Reddy, Lam M. Nguyen, Jayant Kalagnanam, and Anak Agung Julius. Revisiting the generic transformer: Deconstructing a strong baseline for time series foundation models, 2026. URL https://arxiv.org/abs/2602.06909.

[13] 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. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Unb5CVPtae.

[14] Chenxi Liu, Qianxiong Xu, Hao Miao, Sun Yang, Lingzheng Zhang, Cheng Long, Ziyue Li, and Rui Zhao. Timecma: Towards llm-empowered multivariate time series forecasting via cross-modality alignment, 2025. URL https://arxiv.org/abs/2406.01638.

[15] Kuiye Ding, Fanda Fan, Yao Wang, Ruijie Jian, Xiaorui Wang, Luqi Gong, Yishan Jiang, Chunjie Luo, and Jianfeng Zhan. Dualsg: A dual-stream explicit semantic-guided multivariate time series forecasting framework. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, page 508–517, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400720352. doi: 10.1145/3746027.3755458. URL https://doi.org/10.1145/3746027.3755458.

[16] Lorenzo Brigato, Rafael Morand, Knut Joar Strømmen, Maria Panagiotou, Markus Schmidt, and Stavroula Mougiakakou. There are no champions in supervised long-term time series forecasting. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=y01JuBpTBB.

[17] Qinwei Ma, Jingzhe Shi, Jiahao Qiu, and Zaiwen Yang. Position: The inevitable end of one-architecture-fits-all-domains in time series forecasting, 2026. URL https://arxiv.org/abs/2602.01736.

[18] Mingyue Cheng, Xiaoyu Tao, Qi Liu, Ze Guo, and Enhong Chen. Position: Beyond model-centric prediction–agentic time series forecasting. *arXiv preprint arXiv:2602.01776*, 2026.

[19] Xiaohan Zhang, Tian Gao, Mingyue Cheng, Bokai Pan, Ze Guo, Yaguo Liu, Xiaoyu Tao, and Qi Liu. Alphacast: A human wisdom-llm intelligence co-reasoning framework for interactive time series forecasting, 2026. URL https://arxiv.org/abs/2511.08947.

[20] Bokai Pan, Mingyue Cheng, Zhiding Liu, Shuo Yu, Xiaoyu Tao, Yuchong Wu, Qi Liu, Defu Lian, and Enhong Chen. Castflow: Learning role-specialized agentic workflows for time series forecasting, 2026. URL https://arxiv.org/abs/2604.27840.

[21] Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. Timeseriesscientist: A general-purpose ai agent for time series analysis, 2025. URL https://arxiv.org/abs/2510.01538.

[22] Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. AutoML-agent: A multi-agent LLM framework for full-pipeline autoML. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=p1UBWkOvZm.

[23] Zhijian Xu, Hao Wang, and Qiang Xu. Influence-aware forecasting: Breaking the self-stimulation barrier in time series, 2026. URL https://openreview.net/forum?id=0cbUKCyBsH.

[24] Fei Wang, Yujie Li, Zezhi Shao, Chengqing Yu, Yisong Fu, Zhulin An, Yongjun Xu, and Xueqi Cheng. Aries: Relation assessment and model recommendation for deep time series forecasting, 2025. URL https://arxiv.org/abs/2509.06060.

[25] Zhiyuan Zhao, Juntong Ni, Shangqing Xu, Haoxin Liu, Wei Jin, and B. Aditya Prakash. Timerecipe: A time-series forecasting recipe via benchmarking module level effectiveness. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=CsoR8ztR0C.

[26] Xiaoyu Tao, Mingyue Cheng, Ze Guo, Shuo Yu, Yaguo Liu, Qi Liu, and Shijin Wang. Memcast: Memory-driven time series forecasting with experience-conditioned reasoning, 2026. URL https://arxiv.org/abs/2602.03164.

[27] Xiaoyu Tao, Mingyue Cheng, Chuang Jiang, Tian Gao, Huanjian Zhang, and Yaguo Liu. Cast-r1: Learning tool-augmented sequential decision policies for time series forecasting, 2026. URL https://arxiv.org/abs/2602.13802.

[28] Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Kamarthi, Aditya B. Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, and B. Aditya Prakash. Time-mmd: Multi-domain multimodal dataset for time series analysis, 2025. URL https://arxiv.org/abs/2406.08627.

[29] Zhijian Xu, Wanxu Cai, Xilin Dai, Zhaorong Deng, and Qiang Xu. Fidel-ts: A high-fidelity multimodal benchmark for time series forecasting, 2026. URL https://arxiv.org/abs/2509.24789.

[30] Chin-Chia Michael Yeh, Vivian Lai, Uday Singh Saini, Xiran Fan, Yujie Fan, Junpeng Wang, Xin Dai, and Yan Zheng. Empowering time series forecasting with llm-agents, 2025. URL https://arxiv.org/abs/2508.04231.

[31] Jianfeng Zhan, Lei Wang, Wanling Gao, Chenxi Wang, Hongxiao Li, Fanda Fan, and Guoxin Kang. *Evaluatology: The Science of Uncovering the Effects*. BenchCouncil Press, Hong Kong, version 1.0 (first edition, 2025) edition, 2025. ISBN 978-988-71596-8-1.

[32] Xiaorui Wang, Fanda Fan, Chenxi Wang, Yuxuan Yang, Rui Tang, Kuoyu Gao, Simiao Pang, Yuanfeng Shang, Zhipeng Liu, Wanling Gao, Lei Wang, and Jianfeng Zhan. Combinationts: A modular framework for understanding time-series forecasting models, 2026. URL https://arxiv.org/abs/2605.01231.

[33] Yu Chen, Nathalia Céspedes, and Payam Barnaghi. A closer look at transformers for time series forecasting: Understanding why they work and where they struggle. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=kHEVCfES4Q.

[34] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7oLshfEIC2.

[35] Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[36] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[37] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256, May 2002. ISSN 0885-6125. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

[38] Mingyue Cheng, Jiqian Yang, Tingyue Pan, Qi Liu, Zhi Li, and Shijin Wang. Convtimenet: A deep hierarchical fully convolutional model for multivariate time series analysis. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, page 171–180, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713316. doi: 10.1145/3701716.3715214. URL https://doi.org/10.1145/3701716.3715214.

[39] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=J4gRj6d5Qm.

[40] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[41] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. ACM, August 2016. doi: 10.1145/2939672.2939785. URL http://dx.doi.org/10.1145/2939672.2939785.

[42] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[43] Hao Xue and Flora D. Salim. Promptcast: A new prompt-based learning paradigm for time series forecasting, 2023. URL https://arxiv.org/abs/2210.08964.

[44] Xiaoyu Tao, Shilong Zhang, Mingyue Cheng, Qi Liu, Daoyu Wang, Bokai Pan, Tingyue Pan, Changqing Zhang, and Shijin Wang. From values to tokens: An LLM-driven framework for context-aware time series forecasting via symbolic discretization, 2026. URL https://openreview.net/forum?id=JiplNsLiPv.

[45] Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. S2ip-llm: semantic space informed prompt learning with llm for time series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[46] Mingyue Cheng, Jiahao Wang, Daoyu Wang, Xiaoyu Tao, Qi Liu, and Enhong Chen. Can slow-thinking llms reason over time? empirical studies in time series forecasting, 2025. URL https://arxiv.org/abs/2505.24511.

[47] Yucong Luo, Yitong Zhou, Mingyue Cheng, Jiahao Wang, and Daoyu Wang. Time series forecasting as reasoning: A slow-thinking approach with reinforced LLMs, 2026. URL https://openreview.net/forum?id=uwoA5iyTC6.

[48] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

[49] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=wbpxTuXgm0.

[50] Haixu Wu, Teng Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *ArXiv*, abs/2210.02186, 2022. URL https://api.semanticscholar.org/CorpusID:252715491.

[51] Peiwang Tang and Weitai Zhang. Unlocking the power of patch: patch-based mlp for long-term time series forecasting. In *Proceedings of the Thirty-Ninth AAAI Conference*

on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i12.33378. URL https://doi.org/10.1609/aaai.v39i12.33378.

A Additional Method Details

A.1 Summary of Notation

Table 5 summarizes the symbols used across the method and appendix. The table is grouped by role so that the construction context, design actions, evaluation evidence, memory, and selection rule can be read together.

Table 5: Summary of main notation used in Nuwa.

Category	Symbol	Meaning
Data context	$\mathcal{D}$, $v_{\mathcal{D}}$	Time series dataset and its data view, including frequency, horizon, variables, missingness, trend, periodicity, volatility, metadata, and covariates.
Data context	g_t, c_t	Step-level diagnostics and agent context $c_t = (v_{\mathcal{D}}, g_t, s_t, \mathcal{M}_t, e_{<t})$.
Design space	$\mathcal{S}$	Structured design space $\mathcal{S}_{\mathrm{M}} \times \mathcal{S}_{\mathrm{X}} \times \mathcal{S}_{\mathrm{H}}$.
Design space	s_t, F_s	Configuration at step t and the forecasting system instantiated by configuration s.
Action	k_t, $a_{t,j}$	Target subspace $k_t \in \{\mathrm{M}, \mathrm{X}, \mathrm{H}\}$ and candidate action $a_{t,j} = (k_t, \delta_{t,j})$.
Action	$\mathcal{C}_t^{k_t}$, $\mathcal{U}_t^{k_t}$	Proposed candidate set and selected budgeted subset in the target subspace.
Evaluation	$\mathcal{E}_{t,j}$, p_t	Evaluation condition and protocol, including fixed non-target subspaces, split, metric, seed policy, and budget.
Evaluation	R_t, $R'_{t,j}$, $\tau_{t,j}$, $q_{t,j}$	Reference outcome, candidate outcome, local effect, and evidence quality.
Memory	$\mathcal{M}_t$, $\mathcal{Z}_t$, $\mathcal{P}_t$	Design Memory, instance-level records, and pattern-level regularities.
Memory	$z_{t,j}$, $e_{<t}$	Stored evidence record and prior attributed traces used by later proposals.
Selection	α_t, $\hat{G}_t$, U_t, C_t, L_t	Acquisition score, expected gain, uncertainty/novelty, evaluation cost, and risk penalty.

A.2 Notation and Context Construction

Nuwa separates task context from design evidence. For dataset $\mathcal{D}$, the data view $v_{\mathcal{D}}$ contains static or slowly changing meta-information:

$$v_{\mathcal{D}} = (f, H, N, \mathrm{miss}, \mathrm{trend}, \mathrm{period}, \mathrm{vol}, \mathrm{meta}, \mathrm{cov}),$$

where f is frequency, H is forecasting horizon, N is the number of variables, and the remaining fields summarize missingness, trend, periodicity, volatility, metadata, and available covariates. At construction step t, the performance diagnostic g_t summarizes observed behavior of the current reference configuration, including aggregate metrics, horizon-wise errors, variable-wise errors, residual autocorrelation or spectral patterns, training stability, runtime, and failure logs. The agent context is

$$c_t = (v_{\mathcal{D}}, g_t, s_t, \mathcal{M}_t, e_{<t}),$$

where $s_t = (s_{\mathrm{M}}, s_{\mathrm{X}}, s_{\mathrm{H}})$, $\mathcal{M}_t = (\mathcal{Z}_t, \mathcal{P}_t)$, and $e_{<t}$ contains prior attributed traces.

A.3 Design Space Actions

Nuwa constrains the agent to typed actions in the structured design space. At step t, an action is written as $a = (k, \delta)$, where $k \in \{\mathrm{M}, \mathrm{X}, \mathrm{H}\}$ selects the target subspace and δ specifies the bounded modification. Table 6 lists the action families admitted in each subspace and the conditions that must remain fixed to support attribution.

Module actions may change one component or a small coupled group in the CombinationTS decomposition. Exogenous actions must declare source, forecast-time availability, timestamp alignment, lag relation, leakage risk, and redundancy group. Hyperparameter actions are constrained to bounded groups, such as learning-rate and scheduler changes or depth-width changes, to preserve attribution.

Table 6: Typed action families and their required fixed conditions.

Subspace	Agent action	Design object	Held fixed for attribution
$\mathcal{S}_{\mathrm{X}}$ Exogenous	propose, align, add, remove, group, filter	Forecast-time covariates, lags, transforms, redundancy groups	$\mathcal{S}_{\mathrm{M}}$, $\mathcal{S}_{\mathrm{H}}$, p_t
$\mathcal{S}_{\mathrm{M}}$ Module	replace or compose a bounded module group	Input transformation, embedding, encoder, decoder, output transformation	$\mathcal{S}_{\mathrm{X}}$, $\mathcal{S}_{\mathrm{H}}$, p_t
$\mathcal{S}_{\mathrm{H}}$ Hyperparameter	adjust bounded HP groups	Lookback, learning rate, scheduler, depth, width, dropout, batch, budget	$\mathcal{S}_{\mathrm{M}}$, $\mathcal{S}_{\mathrm{X}}$, p_t

A.3.1 Extended Module Catalog

Table 7 lists representative modules available at each pipeline stage. The catalog is open: any new forecasting method can be decomposed into the five-stage view $\mathcal{T}_{\text{in}} \to \mathcal{B} \to \Phi \to \mathcal{G} \to \mathcal{T}_{\text{out}}^{-1}$, and its components can be registered as additional options without changing the construction procedure or skill interfaces. All techniques appearing in Table 1 of the main text are included as concrete instantiations.

Table 7: Extended module catalog. Each row is a pipeline stage; columns list available modules and their brief descriptions. The catalog is extensible—new modules can be added by registering an implementation that satisfies the stage's tensor interface.

Stage	Module	Description
	RevIN / Instance Norm	Per-instance normalization [48]
	Trend-Seasonal Decomp.	Moving-average based trend-seasonal split
	Multi-Scale Decomp.	Hierarchical down-sampling at multiple resolutions
	Cycle-Residual Decomp.	Period-based cycle extraction
	Covariate Alignment	Exogenous timestamp alignment and concatenation
$\mathcal{B}$	None	No embedding; raw values passed directly
	Point-wise	Per-timestep linear projection
	Patch-wise	Non-overlapping or strided patch tokenization [2]
	Variate-wise	Per-variate token embedding [3]
	None	Identity pass-through; no encoder applied
	Transformer (vanilla)	Multi-head self-attention encoder
	Transformer (auto-corr.)	Auto-correlation attention [39]
	Time-Feature MLP	Alternating time-mixing and feature-mixing MLPs [49]
	Multi-branch MLP	Multi-scale past-decomposable mixing [34]
	2D-Conv (Inception)	TimesNet-style 2D convolution on period-folded tensors [50]
	GNN + MLP	Graph-enhanced frequency filtering [7]
	Geometric Attention	Attention with geometry-aware temporal or variate relations for structured multivariate dependencies
	Frequency-domain filter	Learnable spectral filter in frequency space
$\mathcal{G}$	Linear projection	Single linear layer mapping to horizon
	Convolutional linear	Conv1D followed by linear projection
	Future multi-mixing (FMM)	Future-decomposable mixing [34]
	Auto-Correlation decoder	Cross-correlation based decoding [39]
	De-normalization	Reverse RevIN or instance norm
	Rescaling	Learned or fixed scale restoration
	Branch aggregation	Summation or gating of decomposed branches

A.3.2 Full Module Decomposition Table

Table 8 gives the complete module-level decomposition of all forecasting models and Nuwa-constructed systems referenced in the experiments. This extends the compact view in Table 1 of the main text. Rows marked with ★ denote dataset-specific configurations composed by Nuwa that do not match any single existing model identity.

A.3.3 Module-Replacement Interface

Nuwa replaces modules through a typed tensor interface that guarantees composability. Each pipeline stage defines an input tensor shape and an output tensor shape; any module registered at that stage

Table 8: Complete module-level decomposition following the CombinationTS [32] and TimeRecipe [25] component view. Nuwa-constructed systems (★) are dataset-specific compositions discovered during evaluation-grounded construction.

System	Input Trans.	Embedding	Encoder	Decoder	Output Trans.
DLinear [1]	trend-seasonal decomp.	None	None	Linear	branch aggregation
PatchTST [2]	norm.	Patch	Transformer	Linear	de-norm.
iTransformer [3]	norm.	Variate	Transformer	Linear	de-norm.
TimeXer [4]	norm. + exog. alignment	Patch + Variate	Transformer	Linear	de-norm.
TimesNet [50]	norm.	Point	2D-Conv (Inception)	Linear	de-norm.
TSMixer [49]	norm.	None	Time-Feature MLP	Linear	de-norm.
TimeMixer [34]	trend-seasonal + multi-scale	Point	Multi-branch MLP	FMM	branch aggregation
TimeFilter [7]	norm.	Patch	GNN + MLP	Linear	de-norm.
Autoformer [39]	trend-seasonal decomp.	Point	Transformer	Auto-Correlation	de-norm.
Nuwa-built (ETTh1) ★	RevIN + Multi-Scale Decomp.	Patch-wise	None	Linear	branch aggregation
Nuwa-built (ETTh2) ★	RevIN + Multi-Scale Decomp.	Patch-wise	None	Linear	branch aggregation
Nuwa-built (ETTm1)	RevIN	Variate-wise	Geometric Attention	Linear	de-norm.
Nuwa-built (ETTm2) ★	RevIN	Variate-wise	Time-Feature MLP	Linear	de-norm.
Nuwa-built (Traffic)	RevIN	Variate-wise	Transformer	Linear	de-norm.
Nuwa-built (Weather) ★	RevIN + Trend-Seasonal Decomp.	Patch-wise	Geometric Attention	Linear	branch aggregation
Nuwa-built (ECL)	RevIN	Variate-wise	Geometric Attention	Linear	de-norm.
Nuwa-built (Solar) ★	RevIN + Multi-Scale Decomp.	Patch-wise	Geometric Attention	Linear	branch aggregation

must satisfy both. Formally, let $\mathbf{X}^{(\ell)} \in \mathbb{R}^{B\times T_\ell \times D_\ell}$ denote the tensor entering stage ℓ, where B is batch size, T_ℓ is temporal length, and D_ℓ is feature dimension. A module m at stage ℓ implements

$$m : \mathbb{R}^{B\times T_\ell \times D_\ell} \to \mathbb{R}^{B\times T_{\ell+1} \times D_{\ell+1}},$$

with $T_{\ell+1}$ and $D_{\ell+1}$ determined by the stage contract (e.g., the encoder must preserve $T_\ell = T_{\ell+1}$ while the decoder maps to horizon H). The replacement procedure is:

1. **Validate compatibility**: check that the candidate module's input and output shapes match adjacent stages under the current configuration $(T_\ell, D_\ell, T_{\ell+1}, D_{\ell+1})$.
2. **Transfer shared parameters**: if the candidate shares substructure with the replaced module (e.g., same attention heads but different mixing), copy compatible weights; otherwise initialize from scratch.
3. **Instantiate and register**: insert the candidate into the pipeline graph, fix all non-target modules, and verify a forward pass produces the expected output shape.
4. **Declare fixed conditions**: record which modules, exogenous variables, and hyperparameters are held fixed so the subsequent evaluation is a valid controlled comparison.

When a coupled module group is replaced (e.g., input transformation and its inverse output transformation), both modules are swapped atomically and the change is attributed as a single bounded action.

A.3.4 Exogenous Operation Definitions

The five typed operations on $\mathcal{S}_X$ are defined below. Each operation specifies when it is invoked, what it produces, and what invariants it must preserve.

1. **Discover.** Identify candidate exogenous variables from external sources. The agent uses domain knowledge, LLM priors, dataset metadata, and residual diagnostics to hypothesize variables that may explain unexplained variance in the target series. Sources include public APIs, knowledge bases, time-aligned databases, and user-provided data catalogs. Each discovered variable is tagged with source provenance, expected temporal relation, and preliminary leakage assessment.
2. **Align.** Resolve temporal mismatches between a candidate variable and the target series. This includes frequency conversion (e.g., daily to hourly via interpolation or forward-fill), lag specification (e.g., weather forecast leads actual temperature by 6 hours), timezone alignment, and missing-value imputation at the target frequency. The alignment rule must be deterministic and reproducible to support controlled evaluation.

3. **Add.** Include an aligned candidate variable in the forecasting system's input. The variable is concatenated to the covariate tensor under the current input transformation and embedding. Adding a variable changes only $\mathcal{S}_{\mathrm{X}}$ while holding $\mathcal{S}_{\mathrm{M}}$ and $\mathcal{S}_{\mathrm{H}}$ fixed. The agent must declare whether the variable is available at the forecast origin (forecast-time availability) to avoid information leakage.
4. **Filter.** Screen a bounded variable group for redundancy, collinearity, or low marginal contribution. Given a set of added variables, Filter computes pairwise correlation, variance inflation, or conditional contribution under the current evaluation condition. Variables that fail the screen are marked for potential removal but are not dropped until a controlled comparison confirms no benefit.
5. **Remove.** Drop a previously added variable after controlled evaluation shows it provides no marginal improvement or introduces instability. Removal is itself a controlled action: the agent evaluates the system with and without the target variable under fixed modules and hyperparameters. If the with-variable configuration is not reliably better, the variable is removed and the evidence is stored for future retrieval.

A.4 Budgeted Construction Procedure

Nuwa uses the same procedure for one-step and multi-step construction. One-step construction stops after the first evaluated candidate set; multi-step construction reuses attributed evidence and updated diagnostics to form the next context.

Algorithm A.1: Evaluation-Budgeted Construction

Input: dataset $\mathcal{D}$; design space $\mathcal{S}_{\mathrm{M}} \times \mathcal{S}_{\mathrm{X}} \times \mathcal{S}_{\mathrm{H}}$; memory $\mathcal{M}$; evaluation budget B.
Output: constructed configuration $s^{\star}_{\mathcal{D}}$ and attributed evidence records.

1. Build $v_{\mathcal{D}}$ and initialize s_1 from retrieved memory patterns or a default reference configuration. // profile and warm start
2. Evaluate or summarize s_t to obtain diagnostics g_t, and form context $c_t = (v_{\mathcal{D}}, g_t, s_t, \mathcal{M}_t, e_{<t})$. // observe
3. Select a target subspace $k_t \in \{\mathrm{M}, \mathrm{X}, \mathrm{H}\}$ from c_t. // choose controllable object
4. Generate candidates $\mathcal{C}_t^{k_t} = \mathrm{Propose}_{k_t}(c_t)$. // propose
5. Select $\mathcal{U}_t^{k_t} \subseteq \mathcal{C}_t^{k_t}$ under the step budget using Appendix A.6. // budgeted selection
6. For each $a_{t,j} \in \mathcal{U}_t^{k_t}$, instantiate $s'_{t,j} = \mathrm{Apply}(s_t, a_{t,j})$ and check its resolved configuration difference. Keep s_t fixed within the round. // apply action
7. Compare reference and candidate validation outcomes under the subspace-specific EC; reuse a reference result only on an exact configuration and protocol match. // evaluate
8. Compute $\tau_{t,j}$ and $q_{t,j}$; form the eligible improving set $\mathcal{A}_t$ and update the incumbent using Eq. 8. Record the decision rationale and all trial outcomes. // attribute and decide
9. Write instance records, update patterns from quality-qualified comparisons, and choose retry, switch, stop, or continue. If continuing, refresh diagnostics for the retained incumbent and repeat within the remaining budget. // remember and plan

A.5 Subspace-Specific Candidate Proposal

Exogenous-variable candidates. Nuwa proposes an exogenous candidate set $\mathcal{C}_t^{\mathrm{X}}$ from available covariates, external-variable libraries, and retrieved memory records. Each candidate x is represented as

$$x = (\mathrm{name}, \mathrm{source}, \mathrm{availability}, \mathrm{lag}, \mathrm{align}, \mathrm{transform}, \mathrm{risk}, \mathrm{group}).$$

Proposal uses $v_{\mathcal{D}}$ to match domain and temporal metadata, g_t to identify residual patterns or horizon segments that may require external drivers, and $\mathcal{M}_t$ to retrieve variables that helped similar data views. Candidates with unclear forecast-time availability or high leakage risk are not evaluated unless explicitly marked as diagnostic-only.

Module candidates. Nuwa proposes $\mathcal{C}_t^{\mathrm{M}}$ from pattern-level regularities over the modular CombinationTS space. Each candidate specifies the changed component or bounded component group,

unchanged modules, compatibility constraints, and the expected diagnostic it addresses. For example, periodic residuals may trigger decomposition-aware input transformations, cross-variate errors may trigger variate-wise embedding, and long-horizon degradation may trigger decoder or look-back related module changes.

Hyperparameter candidates. Nuwa proposes $\mathcal{C}_t^{\mathrm{H}}$ as bounded hyperparameter groups conditioned on the fixed module and exogenous settings. Candidate groups include look-back length, learning rate and scheduler, hidden width and depth, dropout, batch size, and training budget. When evaluations are expensive, Nuwa can use low-fidelity diagnostics such as short training runs or smaller budgets to remove unstable regions before full evaluation.

A.6 Budgeted Candidate Subset Selection

Nuwa selects a subset of candidates rather than evaluating the entire proposal set. For each candidate action $a \in \mathcal{C}_t^k$, Nuwa computes

$$\alpha_t(a) = \hat{G}_t(a \mid c_t, \mathcal{M}_t) + \lambda U_t(a \mid \mathcal{M}_t) - \gamma C_t(a) - \eta L_t(a),$$

where $\hat{G}_t$ is an expected-gain estimate, U_t is uncertainty or novelty, C_t is evaluation cost, and L_t is implementation risk among admissible actions. Compatibility and forecast-time availability are checked before ranking; a high acquisition score cannot override these requirements. This is an acquisition-style contextual rule related to sequential model-based optimization and adaptive allocation [35, 36, 37].

Let $\mathcal{N}_t(a)$ be retrieved records with similar data views, compatible diagnostics, and matching action type. For loss metrics,

$$\hat{G}_t(a) = -\frac{\sum_{z_i \in \mathcal{N}_t(a)} w_i \tau_i}{\sum_{z_i \in \mathcal{N}_t(a)} w_i + \epsilon_g}, \qquad w_i = \mathrm{sim}(v_{\mathcal{D}}, v_i)\mathrm{sim}(g_t, g_i)\mathrm{rel}(q_i).$$

The uncertainty term can be instantiated as

$$U_t(a) = \frac{1}{\sqrt{|\mathcal{N}_t(a)| + 1}} + \xi\, \mathrm{Novel}(a; \mathcal{M}_t).$$

For exogenous variables, Nuwa applies redundancy-aware subset selection:

$$\mathcal{U}_t^{\mathrm{X}} = \arg \max_{\mathcal{U} \subseteq \mathcal{C}_t^{\mathrm{X}}, |\mathcal{U}| \le b_t} \sum_{x \in \mathcal{U}} \alpha_t(x) - \mu \sum_{x_i, x_j \in \mathcal{U}} \mathrm{Red}(x_i, x_j),$$

where Red measures correlation, semantic overlap, or shared source redundancy. For modular and hyperparameter actions, the same objective is used without the redundancy term unless candidates are known to be coupled.

A.7 Controlled Evaluation Conditions

The evaluation condition $\mathcal{E}_{t,j} = (s_{\neg k_t}, p_t)$ must specify the fixed non-target subspaces and the protocol. The protocol p_t includes data split, forecasting horizon, metric, seed policy, preprocessing rules, training budget, early-stopping rule, and cost measurements. Table 9 makes the EC explicit for each target subspace: the changed object is the EO, while the fixed columns define the EC required for a controlled comparison.

Table 9: Controlled evaluation conditions for each target subspace.

Target	Changed object	Fixed condition
$\mathcal{S}_{\mathrm{X}}$	External variable or bounded variable group	Module, HP, split, metric, seed policy, preprocessing, budget
$\mathcal{S}_{\mathrm{M}}$	One module or bounded module group	Exogenous variables, HP, split, metric, seed policy, preprocessing, budget
$\mathcal{S}_{\mathrm{H}}$	Bounded HP group	Modules, exogenous variables, split, metric, seed policy, preprocessing, non-target training settings

For exogenous variables, the candidate must be available at the forecast origin; otherwise the comparison is marked as leakage risk and excluded from high-confidence attribution. For module actions, a coupled module group is allowed only when the components are functionally inseparable, such as an input transformation and its inverse output transformation. For hyperparameter actions, Nuwa evaluates bounded groups to avoid attributing a large uncontrolled grid jump to a single scalar. When lookback, training epochs, or patience is the target HP, its change is explicitly included in the action; EC equality applies to the remaining protocol fields. Shape-dependent parameters required by a module change are recorded as part of that action. These constraints are the practical checks used before evidence from Table 9 is promoted to high-confidence attribution.

A.8 Attribution and Evidence Quality

At step t, the action $a_{t,j} = (k_t, \delta_{t,j})$ is treated as a controlled intervention:

$$s'_{t,j} = \mathrm{do}_{k_t,\delta_{t,j}}(s_t).$$

The local effect is

$$\tau_{t,j}(a_{t,j}; \mathcal{D}, \mathcal{E}_{t,j}) = R_{\mathrm{val}}(s'_{t,j}; \mathcal{D}, \mathcal{E}_{t,j}) - R_{\mathrm{val}}(s_t; \mathcal{D}, \mathcal{E}_{t,j}).$$

This effect is local to the stated EO, AO, and EC [31]. Nuwa stores

$$q_{t,j} = (n_{t,j}, \bar{\tau}_{t,j}, \sigma_{t,j}, \mathrm{iso}_{t,j}, \mathrm{ec}_{t,j}, \mathrm{leak}_{t,j}, \mathrm{valid}_{t,j}),$$

where $\mathrm{iso}_{t,j}$ indicates whether the action is isolated to the declared target, $\mathrm{ec}_{t,j}$ indicates complete and matching non-target conditions, $\mathrm{leak}_{t,j}$ marks leakage risk, and $\mathrm{valid}_{t,j}$ indicates successful execution. These fields are computed from configuration differences, protocol and availability metadata, run status, and repeated outcomes, rather than assigned by the LLM. For paired seeds $r = 1, \ldots, n$, $\bar{\tau}$ and σ summarize the differences $\tau^{(r)} = R^{(r)}_{\mathrm{val}}(s') - R^{(r)}_{\mathrm{val}}(s)$; dispersion is unestimated when $n = 1$. Only valid records with complete EC, acceptable isolation, no unacceptable leakage, and effects passing the configured reliability threshold are used as high-confidence evidence for pattern mining.

A.9 Reflection and Decision Rules

Reflect & Decide converts attributed evidence into control decisions. For loss metrics, candidate eligibility is defined by

$$\mathrm{Eligible}(q) = \mathrm{iso} \wedge \mathrm{ec} \wedge (\mathrm{leak} = 0) \wedge \mathrm{valid} \wedge (\mathrm{rel}(q) \geq r_{\min}).$$

The admissible improvement set additionally requires $\bar{\tau}_{t,j} < -\epsilon$. Equation 8 selects the lowest mean validation loss within this set, using the policy's fixed tie-breaking rule; an empty set leaves the incumbent unchanged. Reliably non-improving candidates are rejected; promising comparisons with inconclusive effects can be repeated under the same EC. Persistent validation diagnostics can motivate a switch of subspace, while the accepted configuration remains the reference. Diagnostics identify hypotheses for subsequent comparisons; a local interaction can be examined through the four configurations s, $s + a$, $s + b$, and $s + a + b$, keeping all other conditions fixed. The corresponding difference of differences is $I(a, b) = [R_{\mathrm{val}}(s + a + b) - R_{\mathrm{val}}(s + a)] - [R_{\mathrm{val}}(s + b) - R_{\mathrm{val}}(s)]$. Its interpretation is local to these actions and their EC, and any additional evaluations consume the verification budget. Construction stops at budget exhaustion or the policy's stopping condition. In multi-step construction, g_{t+1} describes the retained incumbent, and all trial records are appended to the proposal context with their status.

A.10 Design Memory Schema and Pattern Aggregation

Design Memory stores evidence in two layers, $\mathcal{M}_t = (\mathcal{Z}_t, \mathcal{P}_t)$. The instance layer stores controlled comparisons:

$$z_{t,j} = (v_{\mathcal{D}}, g_t, s_t \to s'_{t,j}, a_{t,j}, \mathcal{E}_{t,j}, R_t, R'_{t,j}, \tau_{t,j}, q_{t,j}, \rho_t).$$

The pattern layer summarizes conditional regularities across comparable records. Only quality-qualified comparisons contribute measured effects to pattern aggregation; reliable negative results contribute contradiction evidence. For dataset-masked transfer, remove target-origin instance records

before recomputing all pattern summaries, including support, reliability, and contradiction statistics. For a candidate pattern p, let $\mathcal{Z}(p)$ denote records supporting the same qualitative rule under comparable data views and diagnostics. For loss metrics,

$$\text{score}(p) = \text{rel}(\mathcal{Z}(p)) \left(-\frac{1}{|\mathcal{Z}(p)|} \sum_{z_i \in \mathcal{Z}(p)} \tau_i \right) - \beta \, \text{contradict}(p).$$

Patterns are proposal priors, not immutable rules. When new controlled evidence contradicts a pattern, Nuwa reduces its reliability or narrows its applicable data-view and diagnostic conditions.

A.11 Skill Interfaces

Nuwa uses typed skills as executable contracts between the agent and the construction system. The interfaces follow the same order as the main pipeline: profile context, propose a subspace-specific candidate set, score and select a budgeted subset, evaluate and attribute, reflect and write memory, and plan the next step when construction continues. Following common agent-system practice, each interface defines when it is invoked, which state it may read, which actions it may take, which invariants it must preserve, what structured artifact it must return, how failures are represented, and which downstream interface receives the handoff.

Execution responsibilities and provenance. The LLM proposes actions and interprets feedback; numerical acquisition scoring, compatibility and EC checks, and acceptance gates are deterministic operations under a recorded policy. An LLM's stated confidence is not the evidence-quality descriptor $q_{t,j}$. Each invocation must identify its schema and policy versions, reference and candidate configurations, source-record identifiers, protocol, input/output artifact hashes, and execution status. The policy must specify score normalization and coefficients, tie-breaking, acceptance and reliability thresholds, and stopping rules; these quantities are not inferred from a narrative explanation.

Information and evaluation boundaries. In prospective construction, data-view statistics use the training split, while checkpoint selection, candidate acceptance, and stopping use validation feedback; the test split is reserved for final reporting after selection is frozen. Profiles must identify the feature definition, dependency/backend version, training interval, and data hash, and check timestamp regularity as well as missing values. Dataset masking removes target-origin instance records and recomputes pattern summaries from the remaining evidence. An exact evaluation-cache key includes the resolved configuration, data and code versions, split, seed, and fidelity; a relaxed lookup that changes HPs is a different candidate. Budget records distinguish unique target configurations, exact cache hits, new training runs, seed repeats, failures, and model-call cost.

Data and Performance Profiling Interface

Pipeline stage. Context construction before PROPOSE.
Trigger. A new dataset is introduced, or a new reference configuration s_t has produced fresh evaluation traces.
Readable state. Dataset metadata, training-split observations, available covariates, current configuration s_t, protocol p_t, training logs, and validation traces/residuals.
Allowed actions. Compute data-view descriptors; compute performance diagnostics; generate retrieval keys; flag missing metadata, instability, leakage risk, or protocol incompleteness.
Guardrails. Do not propose design changes; every diagnostic must be tied to a configuration and protocol; distinguish observed covariates from forecast-time-available covariates.
Structured output. `data_view` $v_{\mathcal{D}}$; `diagnostics` g_t; `retrieval_keys`. `risk_flags`; `missing_fields`.
Failure handling. If performance traces are unavailable, return a `partial` data-only context; block operations requiring missing metadata or diagnostics.
Handoff. Pass $c_t = (v_{\mathcal{D}}, g_t, s_t, \mathcal{M}_t, e_{<t})$ to the selected subspace proposal interface.

Exogenous Proposal Interface

Pipeline stage. PROPOSE: generate $\mathcal{C}_t^{\mathrm{X}}$.
Trigger. The planner selects $k_t = \mathrm{X}$, or diagnostics indicate unexplained externally driven errors.
Readable state. c_t, available covariates, external-variable library, data-source metadata, retrieved exogenous records and patterns.
Allowed actions. Propose, align, transform, group, filter, or reject candidate variables; specify with-and-without evaluation conditions.
Candidate schema. `name`; `source`; `availability`; `lag`; `alignment`; `transform`. `leakage_risk`; `redundancy_group`; `expected_effect`; `diagnostic_target`.
Guardrails. A selected variable must be available at the forecast origin; record publication lag and data vintage, not only timestamp alignment. Unknown availability is diagnostic-only and cannot support high-confidence attribution.
Structured output. Candidate set $\mathcal{C}_t^{\mathrm{X}}$, candidate schemas, required fixed conditions, and rejected variables with reasons.
Failure handling. If no safe variable is available, return an empty candidate set with risk flags and suggest switching to $\mathcal{S}_{\mathrm{M}}$ or $\mathcal{S}_{\mathrm{H}}$.
Handoff. Pass $\mathcal{C}_t^{\mathrm{X}}$ to Candidate Scoring and Budgeted Selection.

Module Proposal Interface

Pipeline stage. PROPOSE: generate $\mathcal{C}_t^{\mathrm{M}}$.
Trigger. The planner selects $k_t = \mathrm{M}$, pattern memory recommends a module change, or diagnostics indicate structural error not explained by covariates or HPs.
Readable state. c_t, current module graph, module library, compatibility rules, retrieved module patterns.
Allowed actions. Replace one module; replace a justified bounded module group; bind inverse output transformations to input transformations; reject incompatible modules.
Candidate schema. `target_module`; `delta`; `unchanged_modules`; `compatibility`. `diagnostic_motivation`; `expected_effect`; `implementation_risk`.
Guardrails. Do not change exogenous variables or non-target HPs; verify the resolved-configuration difference. Declare coupled module changes and necessary shape changes. Joint module–HP changes cannot receive isolated module attribution; every candidate must instantiate a valid F_s.
Structured output. Candidate set $\mathcal{C}_t^{\mathrm{M}}$, instantiable configurations, fixed-condition declarations, and rejected module changes with reasons.
Failure handling. If no compatible module action exists, return `blocked` and suggest HP refinement or exogenous exploration.
Handoff. Pass $\mathcal{C}_t^{\mathrm{M}}$ to Candidate Scoring and Budgeted Selection.

Hyperparameter Proposal Interface

Pipeline stage. PROPOSE: generate $\mathcal{C}_t^{\mathrm{H}}$.
Trigger. The planner selects $k_t = \mathrm{H}$, training diagnostics indicate underfitting, overfitting, instability, or inefficient budget use.
Readable state. c_t, current HPs, legal bounds, training curves, resource constraints, retrieved HP records.
Allowed actions. Propose bounded HP groups; assign fidelity level; estimate cost; reject illegal or unsafe settings.
Candidate schema. `hp_group`; `delta`; `bounds`; `fidelity`. `cost`; `stability_risk`; `diagnostic_target`.
Guardrails. Keep modules and exogenous variables fixed; stay inside legal bounds; declare fidelity. If lookback or training budget is the target HP, identify that exception while keeping split boundaries, horizon, metric, and other protocol choices fixed.
Structured output. Candidate set $\mathcal{C}_t^{\mathrm{H}}$, fidelity plan, expected cost, and rejected HP changes

with reasons.
Failure handling. If the budget cannot support evaluation, return low-fidelity candidates or request stop/continue planning.
Handoff. Pass $\mathcal{C}_t^{\mathrm{H}}$ to Candidate Scoring and Budgeted Selection.

Candidate Scoring and Budgeted Selection Interface

Pipeline stage. Between PROPOSE and EVALUATE.
Trigger. A proposal interface returns $\mathcal{C}_t^{k_t}$.
Readable state. c_t, $\mathcal{C}_t^{k_t}$, $\mathcal{Z}_t$, $\mathcal{P}_t$, evaluation budget b_t, cost/risk constraints, redundancy metadata.
Allowed actions. Compute $\hat{G}_t$, U_t, C_t, L_t, and $\alpha_t(a)$; apply redundancy-aware subset selection; reject invalid candidates; reserve budget for verification when required.
Scoring schema. `expected_gain` $\hat{G}_t$; `uncertainty` U_t; `cost` C_t; `risk` L_t. `redundancy`; `score` α_t; `selection_reason`.
Guardrails. Reject prohibited leakage and invalid configurations before ranking; compute scores from the recorded policy and cited evidence, not LLM-assigned confidence. Never select a candidate without a valid EC; expose rejected high-score candidates if risk caused rejection.
Structured output. Ranked candidates, selected subset $\mathcal{U}_t^{k_t}$, rejected candidates with reasons, EC requirements, and budget accounting.
Failure handling. If all scores are invalid or budget is insufficient, return `no_selection` and hand off to Reflection or Multi-Step Planning.
Handoff. Pass $\mathcal{U}_t^{k_t}$ and EC requirements to Evaluation and Attribution.

Evaluation and Attribution Interface

Pipeline stage. EVALUATE and ATTRIBUTE.
Trigger. Candidate Scoring returns a non-empty $\mathcal{U}_t^{k_t}$.
Readable state. Reference s_t, candidate configurations $\{s'_{t,j}\}$, target subspace k_t, protocol p_t, fixed non-target subspaces, random seed policy, resource budget.
Allowed actions. Instantiate candidates, run reference-candidate comparisons, repeat evaluations when requested, compute outcomes, compute local effects, and assess evidence quality.
Guardrails. Enforce EC completeness and fixed non-target conditions; use identical split, metric, preprocessing, seed policy, and non-target training settings. Return validation outcomes to the controller; keep final test reporting separate. Flag failed or partial runs.
Structured output. R_t, $R'_{t,j}$, $\tau_{t,j}$, $q_{t,j}$, runtime, failure logs, protocol violations, and reproducibility metadata.
Failure handling. If execution fails, return failure type, consumed budget, and whether the comparison can be retried without changing EC.
Handoff. Pass candidate evidence to Reflection and Memory.

Reflection and Memory Interface

Pipeline stage. REFLECT & DECIDE.
Trigger. Evaluation and Attribution returns candidate evidence, or Candidate Scoring returns `no_selection`.
Readable state. Candidate evidence, current best configuration, budget state, user or policy constraints, current memory, contradiction flags.
Allowed actions. Accept, reject, retry, switch subspace, stop, continue, write instance records, update pattern reliability, and flag contradictions.
Decision schema. `accepted`; `decision`; `next_subspace`; `rationale` ρ_t. `stop_or_continue`; `memory_updates`.
Guardrails. Apply the recorded acceptance rule deterministically. Do not accept invalid EC or unacceptable leakage, promote single uncertain outcomes to reliable patterns, or infer

residual whiteness from lower MSE. Preserve rejected evidence.
Structured output. Updated s_{t+1}, evidence records e_t, memory writes $z_{t,j}$, updated pattern summaries, and stop/continue decision.
Failure handling. If evidence is inconclusive, choose `retry`, `verify`, or `switch` rather than fabricating attribution.
Handoff. If `continue`, pass updated context to Multi-Step Planning; otherwise return final $s^{\star}_{\mathcal{D}}$.

Multi-Step Planning Interface

Pipeline stage. Controller after REFLECT & DECIDE when `continue`.
Trigger. Reflection returns `continue`.
Readable state. Updated context $c_{t+1} = (v_{\mathcal{D}}, g_{t+1}, s_{t+1}, \mathcal{M}_{t+1}, e_{\leq t})$, remaining budget, accepted/rejected evidence, policy constraints, user constraints.
Allowed actions. Select next subspace, proposal mode, step budget, diagnostic target, fixed conditions, and verification requirements.
Planning schema. `next_subspace` k_{t+1}; `mode`; `budget` b_{t+1}.
`fixed_conditions`; `diagnostic_target`; `termination_risk`.
The mode is one of retrieval, exploration, refinement, or verification.
Guardrails. Preserve the accepted incumbent separately from trial configurations; switching subspaces does not itself replace it. Reserve enough budget for controlled evaluation and record later proposals after the feedback on which they depend.
Structured output. Next-step plan, target proposal interface, budget allocation, and explicit reason for continuing.
Failure handling. If remaining budget cannot support a controlled comparison, return `stop` with final configuration and unresolved diagnostics.
Handoff. Route to Exogenous, Module, or Hyperparameter Proposal Interface according to k_{t+1}.

B Additional Experimental Details

B.1 CastFlow-Style Evaluation Protocol

The grouped Module Space comparison in Table 2 follows the CastFlow-style evaluation protocol [20]. Comparability is defined within each forecasting task by its data split, forecasting horizon, metric scale, and training budget. For short-term forecasting, we evaluate on five EPF markets (BE, DE, NP, FR, PJM); for long-term forecasting, we evaluate on ETTh, ETTm, Weather (WP), Solar, and MOPEX. The long-term experiments use prediction horizon $H = 96$. For Module Space construction, Nuwa optimizes across the three design subspaces (Module, Exogenous, Hyperparameter) using multi-step evaluation-grounded construction. Lookback L is part of the experimental configuration: it stays fixed for an isolated module comparison and can vary when explicitly included in the HP search space. The multivariate long-term comparison in Appendix B.3 also uses $H = 96$ and specifies a joint module–HP search.

For Hyperparameter Space experiments, the search space is defined as $e_{\text{layers}} \in \{1, 2\}$, $d_{\text{model}} \in \{64, 128, 256, 512\}$, learning rate $\in \{0.01, 0.001, 0.0001\}$, and $\text{seq_len} \in \{96, 336\}$, yielding $2 \times 4 \times 3 \times 2 = 48$ candidates in total. All hyperparameter evaluations use $\text{pred_len} = 96$ and fixed module configurations. Cost is measured by the number of evaluated candidate configurations.

For Design Memory ablation, memory is instantiated from controlled evaluations over the three design subspaces. Partial memory settings include random subsampling (25% and 50% of records), module-masked memory (removing evidence about the current model family), and dataset-masked memory (removing evidence from the current dataset).

Each mask is applied to instance records before pattern aggregation, so retained patterns use the same evidence pool as retrieval.

B.2 Experimental Protocol and Visualization Details

The main experimental section reports three types of evidence. First, forecasting quality is measured by MSE and MAE under the same data split, prediction horizon, and training budget for all comparable forecasting systems. Second, construction efficiency is measured by the number of evaluated candidate configurations. For Hyperparameter Space experiments, the exhaustive-grid reference evaluates all 48 candidates in the Cartesian product, while Nuwa evaluates only the one-step proposal or the selected multi-step candidates. Third, memory and knowledge experiments report aggregate construction quality through average rank, Hit@K, or reliability-weighted attributed gain, depending on the available evidence.

Selection and metric scope. For prospective construction, preprocessing and data views are fitted on training observations, checkpoint and configuration selection use validation feedback, and final test evaluation follows selection freeze. Metric scale, target variables, forecast horizon, and aggregation are held fixed within each comparison; outcomes from different scales or task definitions are reported separately.

Budget accounting. Candidate-evaluation counts describe the search budget under the stated space and fidelity. Reference fits, seed repetitions, failed executions, and exact cache hits have separate execution counts, while GPU time and LLM calls describe additional resource use. An exact cache hit preserves the complete configuration and evaluation condition; selecting a different HP setting constitutes a different candidate.

Figure 2(a) reports average rank over datasets, with lower values indicating better construction outcomes. Panel (b) reports per-dataset relative MSE as memory coverage increases, normalizing the no-memory baseline to 100.

B.3 Extended Module Space Results

Table 10 reports the multivariate long-term forecasting results under $\mathrm{pred_len} = 96$. Baseline rows are the best configurations found by exhaustive search within the same module and hyperparameter grid that Nuwa explores. The Nuwa row is the multi-step construction result. Best results per column are in bold, including ties at the displayed precision.

Table 10: Extended Module Space results (multivariate, $\mathrm{pred_len} = 96$). Baselines use the best exhaustive-search configuration within Nuwa's design space. Lower is better.

	ETTh1		ETTh2		ETTm1		ETTm2		ECL		Weather		Traffic	
Model	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
PatchTST	0.3764	0.3983	0.2868	0.3467	0.2950	0.3488	0.1715	0.2609	0.1651	0.2590	0.1552	0.2041	0.4514	0.2871
iTransformer	0.3791	0.3982	0.2942	0.3527	0.2957	0.3491	0.1775	0.2661	0.1316	0.2314	0.1548	0.2049	**0.3612**	0.2639
SimpleTM	0.3731	0.3943	0.2925	0.3501	**0.2828**	**0.3397**	0.1678	0.2535	**0.1293**	**0.2296**	0.1450	**0.1951**	0.3664	**0.2608**
PatchMLP [51]	0.3863	0.4055	0.2922	0.3468	0.2909	0.3490	0.1681	0.2578	0.1495	0.2581	0.1462	0.1978	0.4187	0.3180
NUWA	**0.3684**	**0.3922**	**0.2718**	**0.3361**	**0.2828**	**0.3397**	**0.1628**	**0.2525**	**0.1293**	**0.2296**	**0.1445**	0.1973	**0.3612**	0.2639

Nuwa achieves the best or tied-best MSE on all seven datasets, matching the best baseline MSE (SimpleTM on ETTm1 and ECL, iTransformer on Traffic) while discovering novel composed configurations on the remaining datasets. SimpleTM has lower MAE on Weather and Traffic, so the MSE advantage does not extend to every metric.

B.4 Pattern-Level Knowledge Examples

Table 11 reports Top-3 pattern-level knowledge discovered by Design Memory in each subspace. Each row is a conditional regularity that Nuwa derives from attributed evidence records; a pattern is retained only when the evidence count exceeds a reliability threshold and the applicable data-view conditions are consistent across supporting records.

Rows in Table 11 are grouped by subspace so that module, exogenous-variable, and hyperparameter patterns can be inspected separately. The evidence count and applicable data view indicate whether a pattern is broadly reusable or tied to a narrow construction context. Patterns with negative $\bar{\tau}$ are

Table 11: Top pattern-level knowledge discovered by Nuwa. Lower $\bar{\tau}$ indicates larger improvement for loss metrics. Evidence count reports the number of controlled comparisons supporting the pattern.

Subspace	Pattern	Evid.	$\bar{\tau}$	Applicable Data View
Module	Patch embedding + skip encoder outperforms encoder-equipped variants on low-dimensional, high-trend data	14	-0.031	$N \leq 7$, trend> 0.7
Module	Geometric Attention encoder improves over Transformer and MLP on high-correlation, high-dimensional data	18	-0.024	$N \geq 21$, corr> 0.9
Module	Patch embedding ranks 1st overall (Friedman rank 1.13); significantly better than point and none	24	-0.018	General (8 datasets)
Exogenous	Filtering retrieved variables to top-2 by marginal contribution outperforms using all candidates	10	-0.042	EPF markets (5 datasets)
Exogenous	Market-internal forecasts (e.g., day-ahead price) dominate physical measurements as exogenous inputs	8	-0.065	Electricity price, forecast-time available
Exogenous	Unfiltered open-world variables hurt performance relative to timestamp-only baseline on 3/5 markets	10	$+0.019$	EPF, $n_{\text{cand}} > 10$
Hyperparame	seq_len=336 consistently outperforms seq_len=96 under fixed architecture	12	-0.038	Hourly, $H = 96$
Hyperparamete	d_model=128, e_layers=1 matches d_model=512, e_layers=2 at lower compute cost	9	-0.003	$N \leq 21$, GPU≤ 10GB
Hyperparame	lr=0.001 is the most stable learning rate across module configurations	15	-0.011	General (7 datasets)

improvement signals (the proposed action reduces loss relative to the reference), while the single positive entry indicates a harm signal that informs the agent to avoid unfiltered variable addition.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Sections 3 and 4 describe the design space, evaluation-budgeted construction, Design Memory, and the corresponding empirical comparisons. Section 5 discusses attribution reliability and memory transfer.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 5 discusses attribution reliability under the stated evaluation conditions and the dependence of memory transfer on coverage and task similarity.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not include theoretical results such as convergence guarantees, generalization bounds, or complexity proofs. The formal notation in Section 3 Method defines the structured design space, evaluation-budgeted construction procedure, and controlled attribution protocol, which are methodological definitions validated through empirical comparison rather than mathematical theorems.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Sections 3–4 and the appendices describe the design space, construction procedure, skill interfaces, memory updates, and evaluation protocols. We will publicly release the code, complete experimental configurations, execution-policy settings, data preparation instructions, and experiment records at https://github.com/BenchCouncil/Nuwa.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [No]

   Justification: The project repository at https://github.com/BenchCouncil/Nuwa has been initialized with a README; code and reproducibility materials have not yet been uploaded. The main result-to-run mappings and full execution settings remain incomplete.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [No]

   Justification: Section 4 gives the datasets, baselines, metrics, and reported training setup; Appendix B.1 specifies the HP search ranges and horizons, and Appendix B.2 explains metric scope and budget accounting. Exact historical settings for every main result and all numerical policy parameters are not fully specified, so current code defaults are not a substitute for the original configurations.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

Justification: The main result tables do not report seed-level error bars or confidence intervals. The reported point estimates therefore do not establish robustness to training-seed or LLM variability.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: Section 4 reports NVIDIA V100 hardware, approximately 4000 GPU-hours, and 14,953 tasks for the experimental campaign; Appendix B.2 distinguishes candidate-evaluation counts from training runs, seed repetitions, failures, and cache hits. Per-experiment memory and runtime requirements and complete LLM token/API costs are not provided.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research uses publicly available time series datasets and does not involve human subjects, personal data, or dual-use applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The paper presents a methodological framework for time series forecasting system construction without targeting specific applications, so there is no direct societal impact.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper releases a methodological framework rather than high-risk models or sensitive data, so safeguards are not applicable.

    Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All baseline models and datasets are properly cited with their original sources. Dataset licenses and terms of use are provided in the released code repository.

Guidelines:

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

13. **New assets**

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

Answer: [No]

Justification: The project repository at `https://github.com/BenchCouncil/Nuwa` has been initialized, but the code and documentation for the Nuwa framework structure, design space configuration interfaces, and agent-discovered exogenous variables have not yet been uploaded.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The research does not involve crowdsourcing or human subjects.

Guidelines:

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

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

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

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

    Answer: [N/A]

    Justification: The research does not involve human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: Section 4 identifies Claude Sonnet 4.6 (`claude-sonnet-4-6`, temperature 0.1) for the experimental campaign; Section 3.3 distinguishes LLM proposals and reflection from deterministic scoring and acceptance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.