LEARNING COMPOSABLE FORECASTING OPERATORS VIA BLOCK INFILLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon forecasting is usually formulated either as a fixed mapping to a target window or as a recursive process that advances through its own predictions. We ask whether forecasting can instead be organized around a reusable local prediction problem. We introduce Augure, which learns a shared masked block-infilling operator and composes it across future chunks. The same operator is used at every chunk, predicts each chunk jointly, and can be applied additional times at inference without changing its parameters or output head. Across standard probabilistic forecasting benchmarks, this inductive bias is competitive with direct, recurrent, autoregressive, and diffusion-based alternatives. Controlled ablations show that the local training problem matters: masked infilling, exposure to varying prefix lengths, and the absence of explicit rollout-depth specialization each affect the quality of the resulting operator. A checkpoint trained at $H{=}96$ can be reused up to $H{=}720$, with little native-relative cost on some datasets and larger degradation on others. These results suggest that masked block infilling provides a useful compositional inductive bias while also exposing a central limitation of repeated reuse: performance can degrade as model-generated prefixes accumulate.

1 INTRODUCTION

Forecasting models encode a particular representation of the future. Direct predictors map an observed window to a fixed target, while recursive predictors repeatedly consume their own outputs. This choice determines the local prediction problem, how computation is reused across horizons, and how errors enter later predictions.

We study a different inductive bias: the same *local block-infilling problem* is solved throughout the future. The horizon is partitioned into chunks of patches; during training, one future chunk is stochastically masked and infilled from the preceding context and any visible target patches. At inference, the same operator receives a fully masked chunk, predicts it jointly, commits the result to the prefix, and repeats. Forecast length is therefore controlled by the number of operator applications rather than a horizon-specific output head.

This formulation raises two questions. First, can masked block infilling serve as the forecasting mechanism itself rather than only as a representation-learning objective? Second, what properties of the training formulation make the resulting local rule reusable when it is applied at chunk positions and prefix lengths beyond those represented during training? The second question is especially important because architectural composability does not imply compositional stability: every committed prediction changes the context on which subsequent chunks depend.

Infilling changes the local prediction problem in a way that is useful for composition. In a direct forecaster, output positions have fixed roles within a prescribed horizon, allowing the model to specialize to particular forecast offsets. In Augure, the missing patches vary across training examples, so no position is permanently assigned to be observed or predicted. The shared operator must instead infer local temporal structure from whatever target evidence remains visible together with the causal prefix. We hypothesize that this encourages a prediction rule tied more strongly to local temporal relationships than to a particular absolute forecast position. Such a rule is naturally reusable across future chunks: at inference, complete masking removes the local target evidence and turns the same infilling problem into forecasting from the prefix alone.

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

We introduce Augure[1], a probabilistic encoder-decoder forecaster built around this principle. Each chunk contains B patches of P timesteps. One decoder is shared across chunks, models the current chunk bidirectionally, and conditions causally on the observed lookback and completed chunks. Training samples one target chunk and predicts only its independently masked patches. At inference, the chunk is fully masked, predicted in one pass, and its median forecast is re-encoded into the prefix before the next application.

The resulting operator is local in output size but compositional in horizon. With default $B{=}6$ and $P{=}8$, a model trained at $H{=}96$ is exposed to only two chunk positions, yet the same parameters can be executed for fifteen chunks at $H{=}720$. We use this setting as a controlled test of operator reuse rather than as a claim of horizon invariance. Reuse is nearly cost-free relative to native training on ETTh2, while ETTm1 and Weather show substantially larger penalties. This variation motivates mechanistic ablations that isolate the roles of masked infilling, target-chunk sampling, and explicit chunk-depth conditioning.

Across standard long-horizon forecasting benchmarks, Augure is competitive with direct, recurrent, autoregressive, and diffusion baselines. The mechanistic study further shows that changing the local training problem changes the behavior of the learned operator. On ETTm1, replacing masked infilling with direct block prediction worsens probabilistic accuracy at most horizons, restricting training to a fixed chunk increasingly degrades longer-horizon performance, and explicitly conditioning the decoder on rollout depth performs substantially worse. The corresponding ETTh2 and Weather controls are included in the extended ablation results in Sec. B.3.

Our contributions are threefold. First, we formulate probabilistic forecasting around masked block infilling used directly as the prediction rule, rather than as an auxiliary pretraining objective. Second, we obtain variable-horizon generation by repeatedly applying one shared local operator, making forecast length a composition choice at inference. Third, we provide a mechanistic study of this inductive bias, separating the effects of infilling, chunk sampling, and explicit depth conditioning, and we characterize the limits of reusing a short-horizon checkpoint at longer horizons.

2 Related Work

The design of a forecasting model is closely tied to the local prediction problem it is trained to solve. Direct multi-step models map a context window to a complete target horizon, recursive models learn a transition that is repeatedly rolled forward, and masked models learn to recover missing content from surrounding context. Augure combines elements of the latter two: it uses masked infilling as the local forecasting operation and composes that operation causally across future blocks.

2.1 Direct and Recursive Forecasting

Direct long-horizon forecasters such as DLinear Zeng et al. (2023), PatchTST Nie et al. (2023), iTransformer Liu et al. (2024a), N-BEATS Oreshkin et al. (2019), and N-HiTS Challu et al. (2023) are commonly optimized for a specified prediction length. Large pretrained forecasters such as TimesFM Das et al. (2024) and Moirai Woo et al. (2024) broaden horizon coverage through training across many tasks and sequence lengths. Recursive forecasters instead reuse a local transition. DeepAR Salinas et al. (2020) predicts sequentially, while Timer Liu et al. (2024b) adopts next-token prediction for time series. In these models, generated values enter subsequent context, producing the familiar train-test discrepancy addressed by techniques such as scheduled sampling Bengio et al. (2015).

Augure also reuses a local rule, but the repeated unit is a complete future chunk. All positions inside a chunk are predicted jointly, while causality is imposed only across completed chunks. The resulting local prediction problem is therefore block-level infilling rather than one-step transition prediction.

[1] *Augure* is the French word for *augury*: the practice of forecasting future events from present signs.

2.2 Blockwise Sequence Generation and Forecasting

Blockwise decoding first developed as a general sequence-generation strategy, particularly for language generation. Semi-autoregressive decoding Wang et al. (2018) and blockwise parallel decoding Stern et al. (2018) generate several output tokens per iteration to reduce sequential decoding depth. Block-Recurrent Transformers Hutchins et al. (2022) pursue a related idea at the representation level by carrying recurrent state across blocks in long sequences. These works provide the sequence-modeling lineage for reusing computation at a granularity larger than a single token.

In time-series forecasting, block- or segment-level recurrence appears in models such as SegRNN Lin et al. (2025), which treats segments as the recurrent unit. Augure is closer to this forecasting setting in granularity, but assigns a different task to each future block: the current chunk is represented as a stochastically masked target whose missing patches are infilled with bidirectional modeling inside the chunk, while previously completed chunks provide causal conditioning. Thus the block is both the prediction unit and the unit over which the same infilling operator is composed.

Diffusion forecasters provide a separate form of repeated computation within time-series modeling. TimeGrad Rasul et al. (2021) and NsDiff Ye et al. (2025) iteratively denoise samples from a forecast distribution. Augure instead uses one decoder pass per chunk; repeated computation extends the forecast by another block rather than refining the same output.

2.3 Masked Infilling as a Forecasting Primitive

Masked modeling has been widely used to learn representations by recovering missing content. BERT Devlin et al. (2019) and masked autoencoders He et al. (2022) established this paradigm in language and vision, while Ti-MAE Li et al. (2023), TimeMAE Cheng et al. (2026), and masked pretraining in PatchTST Nie et al. (2023) adapt masked reconstruction to time series. These works use masking primarily as a representation-learning or pretraining mechanism, whereas Augure trains the same masked-infilling mechanism that becomes the forecasting operator when the target chunk is fully masked at inference.

Augure keeps the same infilling operator active at inference. During training it predicts only the masked patches of a stochastically masked future chunk; during inference the whole current chunk is masked, so the infilling problem becomes chunk forecasting. MaskGIT Chang et al. (2022) likewise retains masked prediction during generation, but iteratively refines masks within one generation grid. Augure instead advances through the future chunk by chunk, coupling bidirectional local infilling with causal inter-chunk composition.

TiRex Auer et al. (2025) introduces CPM by masking randomly positioned contiguous patch spans throughout the training sequence so that its recurrent state learns to bridge missing intervals. Toto 2.0 Khwaja et al. (2026) adapts CPM to a transformer, again training with random contiguous masked spans and using fully masked forecast positions at inference. Augure shares the broader idea of training prediction under missing patches, but its masking geometry is different: the mask is restricted to a sampled future target chunk, and the resulting infilling operator is then composed across chunks.

2.4 Reusable Forecasting Operators

Parameter reuse across time is fundamental to recurrent models, while recent pretrained forecasters such as TiRex Auer et al. (2025), TimesFM Das et al. (2024), Moirai Woo et al. (2024), and Toto 2.0 Khwaja et al. (2026) demonstrate broad forecasting behavior from shared parameters and large-scale pretraining. Our setting isolates a narrower in-domain question: one local block-infilling operator is trained inside a bounded horizon and then reused at larger composition depths without changing its parameters. The native-versus-reused experiments therefore measure how that learned forecasting rule behaves as the number of operator applications and the length of the generated prefix increase.

3 METHOD

3.1 NOTATION AND SETUP

Let $\mathbf{x} \in \mathbb{R}^L$ denote a univariate lookback window and $\mathbf{y} \in \mathbb{R}^H$ the future target. Augure follows a channel-independent design: each variate is processed separately with the same shared model parameters. For Augure, we apply RevIN independently to each variate before patching. For each window, the RevIN statistics are computed only from the observed lookback $\mathbf{x}$; the same transformation is applied to the training target and inverted on forecasts at inference, with no learnable affine parameters. For readability, the equations below suppress normalization bars and operate in this model space. The normalized sequence is partitioned into non-overlapping patches of P timesteps. Each patch $\mathbf{p}_j \in \mathbb{R}^P$ is mapped by a residual MLP to an embedding $\mathbf{z}_j \in \mathbb{R}^d$.

The lookback contains $N_{\text{ctx}} = \lceil L/P \rceil$ patch tokens. The future contains $N_{\text{hor}} = \lceil H/P \rceil$ patch tokens and is partitioned into $K = \lceil N_{\text{hor}}/B \rceil$ chunks. Each chunk contains up to B contiguous future patches. We denote the values of chunk i by $\mathbf{C}_i \in \mathbb{R}^{B \times P}$. When the requested horizon does not exactly fill the final patch or chunk, padding is used only for batching and the returned forecast is truncated to the first H timesteps. The chunk is the unit predicted by the local forecasting operator.

3.2 OPERATOR VIEW

A direct multi-step forecaster learns a prediction rule for a specified output horizon. Augure instead defines one chunk-level rule whose parameters and output size are shared across future chunks. Let $\mathbf{s}_{k-1}$ denote the committed temporal prefix before chunk k. E_θ encodes that prefix, $\mathcal{D}_\theta$ decodes the current chunk conditioned on it, and H_θ maps decoder states to quantiles. We write their composition as the local forecasting operator

$$\mathcal{G}_\theta\Big(\tilde{\mathbf{C}}, \mathbf{s}\Big) = H_\theta\Big[\mathcal{D}_\theta\Big(\mathrm{Embed}(\tilde{\mathbf{C}}) \mid E_\theta(\mathrm{Embed}(\mathbf{s}))\Big)\Big] \in \mathbb{R}^{B \times P \times Q}. \tag{1}$$

This notation describes the complete encoder-decoder computation rather than introducing an additional network.

At inference, the next chunk is represented by a fully masked block and the median prediction is committed to the prefix:

$$\begin{aligned} \hat{\mathbf{q}}_k &= \mathcal{G}_\theta\big(\mathbf{0}^{B \times P}, \mathbf{s}_{k-1}\big)\,, \\ \hat{\mathbf{c}}_k &= \hat{\mathbf{q}}_k^{(0.5)}, \qquad \mathbf{s}_k = \mathbf{s}_{k-1} \oplus \hat{\mathbf{c}}_k. \end{aligned} \tag{2}$$

The same update is repeated $K = \lceil H/(PB) \rceil$ times, so the requested horizon changes composition depth rather than the local forecasting rule.

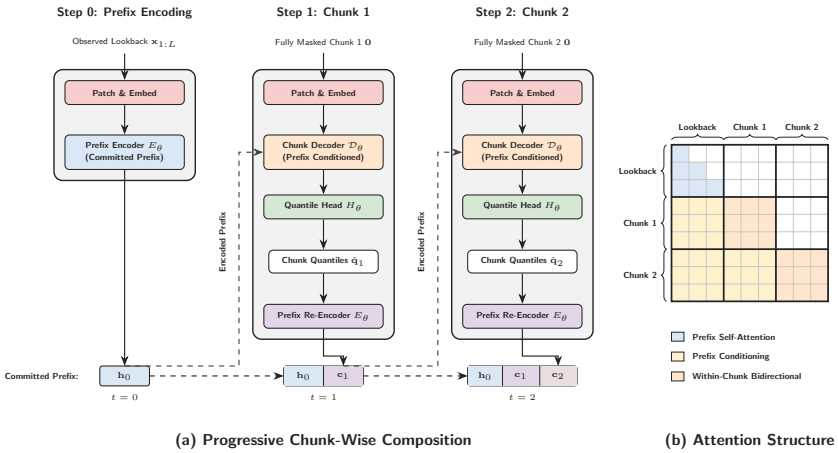
Equation 2 is a value-space description of the recurrence. In the implementation, the observed lookback is encoded once. After a chunk is committed, only that new chunk is encoded while attending to the cached causal prefix, extending the key-value cache used by the next decoder call. The complete prefix is therefore not re-encoded from scratch.

3.3 ARCHITECTURE

Augure uses an encoder-decoder transformer. The encoder represents the observed lookback and previously committed chunks with causal self-attention. The decoder predicts one chunk at a time and uses bidirectional self-attention inside the current chunk, while conditioning only on the causal prefix. Inter-chunk causality therefore follows directly from the prefix construction: no later chunk is available when the current one is predicted. This yields a semi-autoregressive generation pattern Wang et al. (2018); Stern et al. (2018).

At inference, the encoder state is maintained as a causal key-value cache. Each committed chunk extends that cache once, and the resulting prefix conditions the next decoder call. Rotary positional embeddings (RoPE) Su et al. (2024) are used in both encoder and decoder attention, providing positional information without a learned absolute-position lookup table.

A two-layer residual MLP maps decoder states to nine quantile levels ranging from 0.1 to 0.9 for every timestep in the chunk. The 0.5 quantile is committed to the prefix, while all predicted quantiles are retained in the returned probabilistic forecast.



(a) Progressive Chunk-Wise Composition (b) Attention Structure

Figure 1: Overview of Augure. The future is partitioned into chunks of B patches. The encoder represents the observed lookback and any previously completed chunks. The decoder receives the current target chunk as a masked block and predicts all positions in that chunk jointly. Attention is bidirectional within the current chunk and causal across previously completed chunks.

3.4 Masked Chunk Infilling (Training)

Training uses one target chunk per optimization step. A chunk index $i \sim \mathcal{U}\{1, \ldots, K\}$ is sampled once and shared across the minibatch. Chunks before i form a ground-truth prefix, chunk i is the only prediction target, and later chunks are not provided to the prediction rule.

Within the target chunk, each patch is masked independently with probability m and replaced by zeros before patch embedding. We use the validation-selected default $m = 0.5$. Thus m is a Bernoulli masking probability, not an exact masked-patch fraction; the realized number of masked patches varies across examples. Supervision is applied only to masked positions. No free-running rollout is performed during training. Crucially, enforcing a chunk capacity strictly less than the training horizon ($B \times P < H$) is what forces the model to learn compositional KV-cache passing during training. For our $H = 96$ reuse setting, $B = 6$ and $P = 8$ gives a chunk capacity of 48, exposing the operator to multiple target-chunk positions rather than collapsing training to a single future block.

Masking extremes and inference. Because patches are masked independently, the number masked varies across examples. With the default B=6 and m=0.5, both extremes occur with probability $(0.5)^6 = 1/64$: an example may contain no masked patch, which contributes no supervised positions, or a fully masked chunk, which must be predicted from the causal prefix alone. Most examples lie between these extremes and train Augure as an infiller using both prefix information and visible within-chunk structure, whereas inference systematically uses the fully masked endpoint. The masking ablation in Sec. 5.3 tests this train-to-inference shift. Unlike Contiguous Patch Masking (CPM) in TiRex Auer et al. (2025) and Toto 2.0 Khwaja et al. (2026), which masks randomly located contiguous spans, Augure restricts masking to a sampled future target chunk.

For example b in the minibatch, let $\mathcal{M}_i^{(b)}$ be the masked patch indices and let

$$N_{\mathcal{M}} = P \sum_{b=1}^{N_{\text{batch}}} |\mathcal{M}_i^{(b)}|$$

be the number of supervised scalar positions. The batch loss is

$$\mathcal{L}(\theta) = \frac{1}{Q\, N_{\mathcal{M}}} \sum_{b=1}^{N_{\text{batch}}} \sum_{p \in \mathcal{M}_i^{(b)}} \sum_{t=1}^{P} \sum_{q=1}^{Q} \ell_{\tau_q}\left(y_{i,p,t}^{(b)}, \hat{q}_{i,p,t,q}^{(b)}\right), \tag{3}$$

for $N_{\mathcal{M}} > 0$, where $\ell_\tau(y, \hat{q}) = (\tau - \mathbb{I}[y < \hat{q}])(y - \hat{q})$. Masks are sampled independently across examples and target patches. An example with no masked patch contributes no supervised positions; if the entire minibatch contains none, the implementation returns zero loss.

Training and inference differ in both masking regime and prefix content. Training samples the masking pattern stochastically and uses ground-truth preceding chunks, whereas inference always selects the fully masked case and conditions on model-generated preceding chunks. The first difference concentrates inference on a rare endpoint of the training masking distribution; the second introduces recurrent distribution shift, because errors committed at one chunk can alter the conditioning state used by later chunks.

3.5 Chunk-Wise Inference

Inference predicts the future one chunk at a time. The observed lookback is encoded once. For each chunk $k = 1, \ldots, K$:

1. **Create the target chunk.** Set the next chunk to zero, representing a fully masked target.
2. **Predict the chunk.** Decode all $B \cdot P$ scalar positions jointly while conditioning on the cached prefix, producing $\hat{\mathbf{q}}_k \in \mathbb{R}^{B \times P \times Q}$.
3. **Commit and extend the prefix.** Set $\hat{\mathbf{c}}_k = \hat{\mathbf{q}}_k^{(0.5)}$, encode only this committed chunk against the existing causal cache, and append the resulting keys and values. All quantiles remain part of the returned forecast, but only the median quantile path updates the recurrent prefix.

The same decoder parameters are used at every stage, so increasing the requested horizon changes only the number of applications. Caching avoids re-encoding the full committed prefix, although each new chunk still attends to an increasingly long prefix. The exact attention and cache scaling is given in Appendix A.7.

How earlier errors propagate. A predicted chunk is not copied into the masked values of the next target chunk, but it is encoded into the conditioning prefix. Augure therefore does not eliminate dependence on earlier predictions. Errors can propagate through the cached prefix, and the reuse experiments in Sec. 5 measure how this effect changes as more predicted chunks are committed.

Algorithm 2 gives the complete procedure in the appendix.

4 Experimental Setup

4.1 Datasets

We evaluate on seven standard benchmarks: ETTh1, ETTh2, ETTm1, ETTm2, Exchange Rate, Weather, and Electricity; dataset statistics are listed in Table 5. The full Augure comparison is reported on all seven datasets, with Exchange Rate evaluated only at feasible horizons. Following Nie et al. (2023), the lookback is fixed to L=336 and the evaluated horizons are $H \in \{96, 192, 336, 720\}$. Chronological splits are 60/20/20 for ETT and 70/10/20 for the remaining datasets.

4.2 Baselines and Metrics

We compare with DLinear Zeng et al. (2023), PatchTST Nie et al. (2023), DeepAR Salinas et al. (2020), SegRNN Lin et al. (2025), and NsDiff Ye et al. (2025). DLinear, PatchTST, and SegRNN use 9-level quantile heads with pinball loss; for SegRNN, we preserve the original embedding and recurrent backbone and replace only the deterministic output layer. DeepAR retains its Gaussian likelihood, and NsDiff is evaluated from 100 Monte Carlo samples converted to the same quantiles. All models use identical chronological splits, lookback, and native target horizons. Masked pretraining methods are discussed as methodological context rather than direct end-to-end forecasting baselines.

All MAE, CRPS, and WQL values are computed in the original data units after inverse transformation. Augure and SegRNN use non-affine lookback RevIN; the remaining baselines retain their native scaling conventions. Table 6 lists the exact transform and statistic scope for every model. We provide the code, configurations, processed data, and dataset splits required to reproduce the experiments. Model hyperparameters and training settings are collected in Sec. A.

Table 1: Horizon-native forecasting results after inverse transformation to the original data units. Lower is better. Best values are in **bold** and second-best values are underlined. All models report probabilistic metrics from their respective predictive distributions. **Note:** DeepAR's higher error profile is due to its native `MeanScaler` processing uncentered data in our unified raw-unit evaluation pipeline; see Appendix A.6 for an exact breakdown of baseline scaling behaviors.

Dataset	H	DLinear			PatchTST			DeepAR			SegRNN			NsDiff			Augure (ours)		
		MAE	CRPS	WQL	MAE	CRPS	WQL	MAE	CRPS	WQL	MAE	CRPS	WQL	MAE	CRPS	WQL	MAE	CRPS	WQL
ETTh1	96	1.834	1.541	0.311	2.686	2.202	0.444	4.963	3.896	0.785	**1.716**	**1.371**	**0.276**	1.953	1.582	0.319	1.833	1.509	0.304
	192	1.966	1.641	0.330	2.075	1.668	0.335	4.981	3.922	0.788	**1.841**	**1.470**	**0.295**	2.034	1.660	0.333	1.978	1.651	0.332
	336	2.036	1.691	0.338	2.279	1.854	0.371	5.002	3.953	0.791	**2.005**	**1.588**	**0.318**	2.156	1.743	0.349	2.058	1.721	0.344
	720	2.261	1.866	0.370	2.991	2.532	0.502	5.050	4.023	0.797	2.380	1.973	0.391	2.348	1.913	0.379	**2.147**	**1.789**	**0.355**
ETTh2	96	3.113	2.586	0.148	3.086	2.622	0.150	17.455	13.281	0.760	**2.334**	**1.857**	**0.106**	2.949	2.378	0.136	2.466	2.044	0.117
	192	2.956	2.512	0.144	2.872	2.609	0.149	17.481	13.342	0.763	**2.523**	**2.031**	**0.116**	2.871	2.331	0.133	2.557	2.137	0.122
	336	3.344	2.595	0.149	3.166	2.742	0.157	17.461	13.365	0.765	2.721	**2.175**	**0.125**	2.965	2.393	0.137	**2.713**	2.374	0.136
	720	4.300	3.467	0.199	3.959	2.935	0.169	17.389	13.376	0.769	3.103	**2.495**	**0.144**	3.342	2.726	0.157	**3.005**	2.599	0.150
ETTm1	96	1.528	1.320	0.267	1.626	1.325	0.269	4.941	3.859	0.782	1.518	**1.231**	**0.249**	1.737	1.443	0.293	**1.514**	1.262	0.256
	192	**1.652**	1.463	0.297	1.983	1.496	0.304	4.934	3.858	0.783	1.690	**1.356**	**0.275**	1.856	1.543	0.313	1.691	1.415	0.287
	336	1.847	1.558	0.316	2.061	1.603	0.326	4.929	3.860	0.784	1.937	1.528	0.310	1.978	1.649	0.335	**1.802**	**1.516**	**0.308**
	720	**2.015**	1.752	0.355	2.756	2.354	0.478	4.935	3.872	0.786	2.131	**1.681**	**0.341**	2.098	1.753	0.356	2.082	1.780	0.361
ETTm2	96	3.130	2.530	0.148	5.978	5.407	0.315	17.143	12.955	0.755	**1.865**	**1.501**	**0.088**	2.299	1.894	0.110	1.993	1.669	0.097
	192	2.724	2.336	0.136	2.501	2.091	0.122	17.146	12.975	0.757	**2.173**	**1.744**	**0.102**	2.563	2.129	0.124	2.252	1.941	0.113
	336	3.334	2.601	0.152	6.275	5.578	0.325	17.151	13.000	0.758	**2.408**	**1.917**	**0.112**	2.870	2.408	0.140	2.522	2.252	0.131
	720	3.578	3.115	0.181	6.401	5.739	0.334	17.183	13.066	0.761	**2.714**	**2.155**	**0.125**	3.324	2.756	0.160	2.771	2.498	0.145
Electricity	96	278.737	233.278	0.087	316.167	255.246	0.095	506.276	409.248	0.181	300.268	246.343	0.092	312.371	357.658	0.134	**200.170**	**162.456**	**0.061**
	192	311.734	259.673	0.097	318.714	259.117	0.097	559.144	457.794	0.168	320.267	257.560	0.096	313.384	350.095	0.131	**216.187**	**180.112**	**0.067**
	336	329.452	272.800	0.102	336.588	273.495	0.102	521.843	413.337	0.183	331.212	260.271	0.097	317.390	356.202	0.133	**243.696**	**205.840**	**0.077**
	720	366.159	307.151	0.142	406.507	328.202	0.137	639.178	526.468	0.184	374.617	295.408	0.109	329.095	355.770	0.131	**293.042**	**258.086**	**0.096**
Exchange Rate	96	0.036	0.028	0.028	0.044	0.031	0.031	0.733	0.544	0.544	0.028	0.023	0.023	0.031	0.026	0.026	**0.017**	**0.014**	**0.014**
	192	0.038	0.039	0.039	0.043	0.033	0.033	0.733	0.543	0.543	0.034	0.028	0.028	0.036	0.031	0.031	**0.027**	**0.024**	**0.024**
	336	0.057	**0.039**	**0.039**	0.048	0.041	0.041	0.731	0.539	0.539	0.053	0.049	0.049	0.048	0.043	0.043	**0.042**	**0.039**	**0.039**
Weather	96	15.761	15.792	0.095	15.184	12.556	0.075	167.468	128.539	0.770	11.953	9.619	0.058	18.735	17.499	0.105	**9.816**	**8.197**	**0.049**
	192	16.771	16.046	0.096	16.198	12.809	0.077	167.556	128.617	0.770	13.767	11.032	0.066	17.558	16.511	0.099	**10.592**	**9.020**	**0.054**
	336	17.372	16.544	0.099	15.539	12.905	0.077	167.603	128.710	0.771	14.250	11.507	0.069	19.065	17.766	0.106	**10.900**	**9.358**	**0.056**
	720	17.808	17.416	0.104	16.932	13.822	0.083	167.629	128.793	0.771	15.788	12.667	0.076	16.091	14.489	0.087	**12.165**	**10.835**	**0.065**

5 RESULTS

5.1 MAIN FORECASTING RESULTS

Table 1 compares horizon-native models, each trained for the horizon at which it is evaluated. Augure uses B=6, P=8, and the validation-selected masking probability m=0.5. Native-horizon performance is dataset dependent. SegRNN is particularly strong on the ETT benchmarks, especially ETTm2, while Augure obtains the lowest MAE, CRPS, and WQL at every horizon on Weather and Electricity. At H=720, Augure has the lowest MAE on ETTh1 and ETTh2 and the second-lowest MAE on ETTm1 and ETTm2; SegRNN retains the lowest CRPS on ETTh2, ETTm1, and ETTm2. The comparison therefore supports competitiveness rather than uniform dominance, with Augure improving its relative position at some longer ETT horizons. The block-infilling objective is therefore competitive as a native-horizon forecaster before any reuse beyond its training horizon.

5.2 REUSING A SHORT-HORIZON OPERATOR AT LONGER HORIZONS

We next keep the H_0=96 checkpoint fixed and increase only the number of chunk applications. With B=6 and P=8, training at H_0=96 exposes two chunk positions, while evaluation at H=720 requires fifteen applications of the same local rule. For each target horizon H, we compare two models evaluated on the *same* H-step forecasting task. Let $\theta_{\text{reuse}} = \theta_{H_0}$ denote the checkpoint trained at the reference horizon H_0=96 and then reused at horizon H, and let $\theta_{\text{native}}(H) = \theta_H$ denote a separate checkpoint trained directly for that target horizon. For an error metric $\mathcal{E}$ such as MAE or CRPS, we define

$$\Delta_{\mathcal{E}}(H) = \frac{\mathcal{E}(\theta_{\text{reuse}}; H) - \mathcal{E}(\theta_{\text{native}}(H); H)}{\mathcal{E}(\theta_{\text{native}}(H); H)}. \tag{4}$$

Thus $\Delta_{\mathcal{E}}(H) > 0$ means that reusing the H_0=96 checkpoint incurs additional error relative to training natively at H, while $\Delta_{\mathcal{E}}(H) < 0$ means that the reused checkpoint performs better. The quantity measures the cost of *operator reuse at a fixed evaluation horizon*; it does not measure how the forecasting error grows from H_0 to H. This comparison separates the difficulty of forecasting farther into the future from the additional cost of asking a short-horizon operator to run at a greater composition depth. At H=720, reuse is essentially cost-free on ETTh2 (-1.1% MAE, -0.3% CRPS) and remains within a single-digit percentage gap on ETTm2. ETTm1 and Weather behave differently: the corresponding gaps reach $+21.8\%/+27.0\%$ and $+17.6\%/+17.4\%$ in MAE/CRPS.

The divergence is therefore not a monotonic consequence of increasing K alone; it depends on how the generated prefix interacts with the dynamics of the series. This contrast motivates the mechanistic analysis below rather than a stronger claim of horizon invariance. The corresponding native and reused Augure errors across horizons are reported in Sec. B.2.

5.3 Ablation Studies

Mechanistic ablations. We first test the components most directly tied to the proposed inductive bias. The *direct block* control removes masked infilling and predicts the complete next chunk directly from the prefix while preserving the block structure and model scale. The *fixed chunk* control keeps infilling but replaces random target-chunk sampling with a single configured target index i_{fixed} for all training steps. The *scalar depth* control keeps random chunk sampling and defines $r_i = i/K_{\text{train}}$. A learned affine projection is added to every target-chunk token, $\mathbf{z}_{i,p}^{\text{depth}} = \mathbf{z}_{i,p} + \mathbf{W}_r r_i + \mathbf{b}_r$. At inference the same formula is evaluated without clamping, so rollout positions beyond the training range use $r_i > 1$.

Table 2 reports the mechanistic results on ETTm1. The full model is best at every horizon on both metrics; direct block is consistently second-best from $H{=}192$ onward, while fixed chunk is competitive only at $H{=}96$. Performance of fixed chunk degrades as the horizon grows, reaching MAE/CRPS of $2.703/2.421$ at $H{=}720$ versus $2.082/1.780$ for Augure. Scalar depth conditioning is strongly detrimental on ETTm1 and Weather, but its effect is dataset dependent: on ETTh2 it remains competitive and slightly improves CRPS at $H{=}720$. Thus explicit rollout depth is not consistently beneficial, rather than universally harmful. The same controls on ETTh2 and Weather are included in Sec. B.3.

Among the mechanistic controls, Direct Block isolates the contribution of infilling while preserving chunk geometry and model capacity. Its consistently weaker native-horizon performance on ETTm1 indicates that the infilling objective contributes beyond blockwise generation alone; the remaining controls probe the complementary roles of chunk sampling and rollout-depth conditioning.

Table 2: Mechanistic ablations on ETTm1, reported in original data units. Each variant is trained natively at the displayed horizon. Lower is better. For each metric and horizon, the best-performing model is shown in **bold** and the second-best distinct model is underlined.

Variant	$H{=}96$		$H{=}192$		$H{=}336$		$H{=}720$	
	MAE	CRPS	MAE	CRPS	MAE	CRPS	MAE	CRPS
Augure (full)	**1.514**	**1.262**	**1.691**	**1.415**	**1.802**	**1.516**	**2.082**	**1.780**
Direct block (no infilling)	<u>1.631</u>	1.425	<u>1.765</u>	<u>1.569</u>	<u>1.895</u>	<u>1.691</u>	<u>2.134</u>	<u>1.935</u>
Fixed chunk (restricted sampling)	1.669	<u>1.406</u>	2.154	1.912	2.220	1.985	2.703	2.421
Scalar depth	2.424	2.163	2.258	1.962	2.667	2.412	2.765	2.474

The trade-off of compositional stability. The native-horizon and reuse ablations probe complementary properties of the same operator family. Native training generally favors the full block-infilling formulation: it is strongest throughout ETTm1 and Weather and remains highly competitive on ETTh2 and ETTm2. Under deep reuse, however, the ablations separate differently. At $H{=}720$, the full model retains the lowest reused MAE/CRPS on ETTh2 and ETTm2, while the Direct Block ablation degrades more slowly on ETTm1 and Weather; Scalar Depth becomes markedly unstable on Weather, and Fixed Chunk exhibits a different dependence on its native reference. Taken together, these behaviors suggest a fidelity–stability trade-off within the design space: the richer infilling objective learns a strong local forecasting rule, but that rule may also be more sensitive to artifacts accumulated in a model-generated prefix.

The native-relative percentages are therefore diagnostic of each ablation's own reuse behavior, not a ranking across variants. For example, Fixed Chunk reaches 2.522 MAE on ETTm1 at $H{=}720$ with a -7.2% reuse gap because its native $H{=}720$ reference is weaker. Cross-variant comparison at a common depth should instead use the absolute reused errors.

Chunk size and masking probability. We compare $m \in \{0.3, 0.5, 0.7\}$ with the preliminary schedule $m \sim \mathcal{U}(0,1)$ while varying B. Smaller m leaves more local target evidence available;

larger m more closely resembles fully masked inference but removes within-chunk structure. On validation, $B{=}6, m{=}0.5$ is the most consistently strong configuration across horizons and is used throughout the main experiments. The complete validation sweep and the post-selection test sweep are reported in Secs. B.1 and B.5.

5.4 Discussion

The experiments separate *applicability* from *stability*. Parameter sharing makes the learned operator executable at larger composition depths, but every committed prediction also changes the prefix seen by subsequent chunks. This explains why reuse can be almost cost-free on ETTh2 yet substantially worse on ETTm1 or Weather at the same target horizon: the number of applications alone does not determine whether the evolving conditioning state remains useful.

SegRNN leads many short- and medium-horizon ETT settings, whereas Augure obtains the lowest MAE on ETTh1 and ETTh2 at $H{=}720$; we report this crossover without assigning a recurrent-memory mechanism, which would require dedicated diagnostics.

Viewed together, the mechanistic controls separate local forecasting quality from robustness under repeated self-conditioning. The full model provides the strongest native behavior across most settings, but the reuse trajectories show that infilling, restricted chunk sampling, and explicit depth conditioning induce different sensitivities as the generated prefix grows. In particular, some ablations degrade more slowly on ETTm1 and Weather, whereas Scalar Depth becomes highly unstable on Weather; on ETTh2 and ETTm2 the full model remains strongest under deep reuse. Compositional stability is therefore a distinct property of the learned operator family rather than a consequence of native accuracy alone. Because models are trained separately per dataset, we do not study cross-domain or cross-frequency transfer here.

6 Limitations and Future Work

Limitations. Augure remains sequential across chunks and trains with ground-truth prefixes while inference conditions on generated ones, so distribution shift can compound with rollout depth. Only the median trajectory updates the prefix, leaving forecast uncertainty unpropagated, and the channel-independent design does not model explicit cross-variate covariance. Reuse is evaluated to $H{=}720$ ($7.5\times$ the reference horizon); deeper rollouts further increase cache length, unseen RoPE positions, and generated-prefix shift.

Future directions. Partially generated-prefix training, scheduled replacement, or rollout-consistency objectives could reduce this gap. Controlled prefix perturbations across the mechanistic variants could directly measure how each local objective responds to prefix artifacts, while state-stable or contractive updates may preserve native forecasting fidelity under deeper composition. Uncertainty-aware updates, cross-variate modeling, and transfer across datasets or frequencies are additional directions.

7 Conclusion

We introduced Augure, a probabilistic forecaster that learns a masked block-infilling rule and composes it chunk-wise. It is competitive natively, and an $H{=}96$ checkpoint can be reused through $H{=}720$ unchanged. The mechanistic ablations reveal that objectives which improve the local forecasting rule can induce different behavior under repeated self-conditioning, exposing a fidelity–stability trade-off and making compositional stability a first-class property of reusable forecasting operators.

Reproducibility. The paper and supplement provide the splits, configurations, procedures, evaluation protocol, ablations, code, and processed data required for reproduction.

REFERENCES

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 6989–6997, 2023.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11305–11315. IEEE, 2022.

Mingyue Cheng, Xiaoyu Tao, Zhiding Liu, Qi Liu, Hao Zhang, Rujiao Zhang, and Enhong Chen. TimeMAE: Self-supervised representations of time series with decoupled masked autoencoders. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, pp. 498–508, 2026.

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*, pp. 10148–10167, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15979–15988. IEEE, 2022.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. *Advances in neural information processing systems*, 35:33248–33261, 2022.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era, 2026. URL https://arxiv.org/abs/2605.20119.

Zhe Li, Zhongwen Rao, Lujia Pan, Pengyun Wang, and Zenglin Xu. Ti-mae: Self-supervised masked time series autoencoders. *arXiv preprint arXiv:2301.08871*, 2023.

Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. SegRNN: Segment recurrent neural network for long-term time series forecasting. *IEEE Internet of Things Journal*, 2025.

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 *International conference on learning representations*, volume 2024, pp. 11116–11140, 2024a.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: generative pre-trained transformers are large time series models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32369–32399, 2024b.

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.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3): 1181–1191, 2020.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 479–488, 2018.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*, 2024.

Weiwei Ye, Zhuopeng Xu, and Ning Gui. Non-stationary diffusion for probabilistic time series forecasting. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 72112–72130. PMLR, 13–19 Jul 2025. URL `https://proceedings.mlr.press/v267/ye25i.html`.

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

AI USAGE DISCLOSURE

Generative AI tools were used to assist with manuscript drafting, restructuring, language editing, implementation support, and debugging or auditing experimental code. The authors independently verified the technical content, experimental results, implementations, and references. Grammarly was used for grammatical and typographical corrections.

A APPENDIX: EXPERIMENTAL DETAILS

A.1 ALGORITHMS

Algorithm 1 summarizes single-chunk infilling training. Algorithm 2 summarizes chunk-by-chunk inference using the same decoder parameters.

Algorithm 1 Chunk-Masked Training (Augure)

Require: Lookback $\mathbf{x} \in \mathbb{R}^L$, horizon $\mathbf{y} \in \mathbb{R}^H$, patch size P, chunk size B, encoder E_θ, decoder $\mathcal{D}_\theta$, quantile head H_θ

Compute RevIN statistics from observed $\mathbf{x}$ only; normalize $\mathbf{x}$ and $\mathbf{y}$ with those statistics $\rightarrow (\bar{\mathbf{x}}, \bar{\mathbf{y}})$
Partition $\bar{\mathbf{y}}$ into $K = \lceil H/(PB) \rceil$ chunks $\bar{\mathbf{C}}_1, \ldots, \bar{\mathbf{C}}_K$
Sample chunk index $i \sim \mathcal{U}\{1, \ldots, K\}$
Set fixed masking probability $m = 0.5$
For each example b, sample target-patch masks independently at rate m to obtain $\mathcal{M}_i^{(b)}$ ▷ chunks $j > i$ are never observed or decoded
Mask $\bar{\mathbf{C}}_i^{(b)}$ on $\mathcal{M}_i^{(b)}$ for each example to obtain $\tilde{\bar{\mathbf{C}}}_i$
$\mathbf{s}_{<i} \leftarrow \bar{\mathbf{x}} \oplus \bar{\mathbf{C}}_1 \oplus \cdots \oplus \bar{\mathbf{C}}_{i-1}$ ▷ committed value-space prefix
$\mathbf{h}_{<i} \leftarrow E_\theta(\mathrm{Embed}(\mathbf{s}_{<i}))$
$\hat{\mathbf{q}} \leftarrow H_\theta\big(\mathcal{D}_\theta(\mathrm{Embed}(\tilde{\bar{\mathbf{C}}}_i) \mid \mathbf{h}_{<i})\big)$
Pool all masked scalar positions across the minibatch and average pinball loss over masked positions and quantiles; return 0 only if no masked position exists in the minibatch
return ℓ

Algorithm 2 Chunk-by-Chunk Inference (Augure)

Require: Lookback $\mathbf{x} \in \mathbb{R}^L$, target horizon H, encoder E_θ, decoder $\mathcal{D}_\theta$, quantile head H_θ, patch size P, chunk size B

Compute RevIN statistics from $\mathbf{x}$ and normalize $\mathbf{x}$
$\mathbf{h}_0 \leftarrow E_\theta(\mathrm{Embed}(\bar{\mathbf{x}}))$ ▷ initialize causal encoder KV cache
for $k = 1, \ldots, K$ **do** ▷ each chunk, $K = \lceil N_{\text{hor}}/B \rceil$
  $\mathbf{C}_k \leftarrow \mathbf{0}^{B \times P}$ ▷ fully masked target chunk
  $\hat{\mathbf{q}}_k \leftarrow H_\theta\big(\mathcal{D}_\theta(\mathrm{Embed}(\mathbf{C}_k) \mid \mathbf{h}_{k-1})\big)$
  $\hat{\mathbf{c}}_k \leftarrow \hat{\mathbf{q}}_k^{(0.5)}$
  $\mathbf{h}_k \leftarrow E_\theta(\mathrm{Embed}(\hat{\mathbf{c}}_k); \mathbf{h}_{k-1})$ ▷ append committed chunk to KV cache
end for
Inverse-RevIN $(\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_K)$ using the stored lookback statistics
return $\mathrm{Trim}_H(\hat{\mathbf{q}}_1 \oplus \cdots \oplus \hat{\mathbf{q}}_K)$

A.2 Model Hyperparameters

Table 3: Augure architecture hyperparameters: the default configuration used for every result in this paper unless stated otherwise (e.g. the chunk-size and masking-schedule sweeps of §5.3).

Parameter	Value
Patch size P	8
Chunk size B	6
Masking probability m	0.5 (fixed)
Hidden dimension d	128
Attention heads	4
Encoder layers	4
Decoder layers	4
Feedforward dimension	512
Positional encoding	RoPE
Quantile levels Q	$\{0.1, 0.2, \ldots, 0.9\}$ (9 levels)
Normalization	RevIN (no affine parameters)
Masked value	zeros
Total parameters	2.5M

A.3 Training Configuration

Table 4: Augure training configuration: the default recipe used to produce every Augure result in this paper.

Parameter	Value
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	10^{-2}
Warmup ratio	0.02
Batch size	64
Gradient accumulation	2
Max epochs	50
Early stopping patience	10 (monitoring CRPS)
Precision	bfloat16
Masking probability m	0.5 (fixed)
Chunk sampling i	$\mathcal{U}\{1, \ldots, K\}$ per optimization step (shared across batch)
Supervision	masked-infilling (pinball) loss on masked positions

A.4 Dataset Statistics

Splits. ETT-family: train/val/test = 60/20/20. Others: 70/10/20. Lookback $L = 336$ fixed across all datasets.

Table 5: The seven benchmark datasets used throughout this paper (Table 1). For Augure, RevIN is applied independently to each variate; lookback L=336 is fixed across all datasets (§4).

Dataset	Variates	Granularity	Timesteps
ETTh1	7	hourly	17,420
ETTh2	7	hourly	17,420
ETTm1	7	15-min	69,680
ETTm2	7	15-min	69,680
Exchange Rate	8	hourly	7,588
Electricity	321	hourly	26,304
Weather	21	10-min	52,696

A.5 Evaluation Metrics

Evaluation pools scalar forecast targets across forecast timesteps and, for multivariate datasets, across variates. Let $\mathcal{I}$ denote the resulting set of N scalar targets, let $\hat{q}_{n,q}$ be the prediction for quantile level τ_q, and let $Q = 9$. The pinball loss is

$$\ell_\tau(y, \hat{q}) = (\tau - \mathbb{I}[y < \hat{q}])(y - \hat{q}). \tag{5}$$

The reported median MAE and quantile-based CRPS approximation are

$$\mathrm{MAE} = \frac{1}{N} \sum_{n \in \mathcal{I}} |y_n - \hat{q}_{n,0.5}|, \qquad \widehat{\mathrm{CRPS}} = \frac{2}{NQ} \sum_{n \in \mathcal{I}} \sum_{q=1}^{Q} \ell_{\tau_q}(y_n, \hat{q}_{n,q}). \tag{6}$$

The implementation reports weighted quantile loss as

$$\mathrm{WQL} = \frac{\widehat{\mathrm{CRPS}}}{\max\left(\frac{1}{N} \sum_{n \in \mathcal{I}} |y_n|,\, 1\right)}. \tag{7}$$

When evaluation is streamed over multiple units, their metric values are aggregated with weights proportional to the number of scalar targets in each unit. All metrics are computed after mapping forecasts back to the original data units.

A.6 Normalization and Scaling

Table 6: Normalization and scaling used by each forecasting model. All transformations are inverted before evaluation in the original data units.

Model	Transform	Statistics and scope
Augure	RevIN (no affine)	Mean and standard deviation computed independently per variate from the current observed lookback.
SegRNN	RevIN (no affine)	Same non-learnable, per-variate lookback normalization used by Augure.
DLinear	GluonTS `StdScaler`	Observed-value mean and standard deviation computed from the current forecast context; the same location and scale are used to map predictions back to the original units.
PatchTST	GluonTS `StdScaler`	Same context-wise observed-value mean/std scaling as DLinear.
DeepAR	GluonTS `MeanScaler`	Scale is the mean absolute observed target magnitude of the current context; no centering is applied.
NsDiff	`torch_timeseries StandardScaler`	Featurewise mean and standard deviation are fitted on the training split and then reused for validation and test data, following the reference NsDiff pipeline.

The first four models therefore use closely related mean/std normalization, but not at the same statistical scope: Augure and SegRNN compute statistics independently for each forecast window, as do the GluonTS `StdScaler` configurations of DLinear and PatchTST, whereas NsDiff fits one featurewise standardization on the training split. DeepAR differs more substantially because its `MeanScaler` divides by the mean absolute context magnitude without subtracting a location term.

A.7 Implementation

All models are implemented in PyTorch and trained on a single NVIDIA RTX PRO 6000 Blackwell 96 GB with bfloat16 precision. DLinear and PatchTST use the GluonTS implementations with 9-level quantile heads under identical data splits and training budget. SegRNN preserves its original `SaitsEmbedding` and `BackboneSegRNN`, including the backbone's internal last-value centering, and replaces the deterministic output with a 9-quantile linear head optimized by joint pinball loss. Normalization and scaling are reported explicitly in Table 6. NsDiff uses the implementation provided in the public NsDiff repository.[2]

[2] `https://github.com/wwy155/NsDiff`

Incremental inference complexity. Let $n_k = N_{\text{ctx}} + (k-1)B$ be the cached prefix length before chunk k. The lookback is encoded once at cost $O(N_{\text{ctx}}^2)$. Decoding a B-patch chunk and extending the encoder cache each require attention against the existing prefix, giving $O(Bn_k + B^2)$ work per stage. Across K chunks, the leading attention scaling is $O\big(N_{\text{ctx}}^2 + BN_{\text{ctx}}K + B^2K^2\big)$, up to layer and width factors, while KV-cache memory grows as $O(N_{\text{ctx}} + BK)$.

B Additional Results

B.1 Ablation Model Selection (Validation Split)

Table 7: Cross-product ablation over chunk size B and training masking schedule on Weather, validation split. Fixed m denotes a fixed per-patch Bernoulli masking probability; $m \sim \mathcal{U}(0,1)$ denotes the preliminary schedule that samples a masking probability independently for each example before drawing its patch mask. Lower is better; best values are shown in **bold** and second-best distinct values are underlined.

Config	H=96		H=192		H=336		H=720	
	MAE	CRPS	MAE	CRPS	MAE	CRPS	MAE	CRPS
B=3, $m\sim\mathcal{U}(0,1)$	18.726	15.986	**19.147**	16.663	**20.380**	18.448	22.733	20.753
B=3, m=0.3	**17.865**	14.947	19.281	16.719	20.462	18.178	22.357	20.087
B=3, m=0.5	18.273	15.526	19.669	16.968	20.652	18.605	22.664	20.652
B=3, m=0.7	18.262	15.747	19.802	17.500	21.355	19.035	22.737	20.858
B=6, m=0.3	18.171	15.030	20.267	16.965	21.503	18.334	24.298	21.025
B=6, m=0.5 (default)	17.904	**14.759**	19.478	16.363	20.703	**17.661**	**21.501**	**18.937**
B=6, m=0.7	18.201	15.367	19.939	17.126	21.209	18.363	21.832	19.383
B=6, $m\sim\mathcal{U}(0,1)$	18.548	15.460	19.534	16.894	21.097	18.064	22.285	19.809
B=12, m=0.3	19.904	16.256	21.252	17.643	23.798	19.681	25.345	21.131
B=12, m=0.5	19.349	15.828	20.854	17.211	21.822	18.614	22.458	19.012
B=12, m=0.7	18.660	15.940	20.590	17.760	21.310	18.780	22.570	19.970
B=12, $m\sim\mathcal{U}(0,1)$	19.185	15.893	19.353	**16.277**	21.794	18.496	22.015	19.142

Table 7 reports the cross-product ablation on the **validation** split, which is used to select the fixed B=6, m=0.5 masking-probability configuration as Augure's default (§5.3). The $m \sim \mathcal{U}(0,1)$ rows correspond to the earlier schedule in which a masking probability is sampled per example before drawing the patch mask. The test-set sweep is retained only as a post-selection check and plays no role in the design choice.

B.2 Short-Horizon Operator Reuse

Table 8: Native versus reused Augure checkpoints. $\Delta = (\text{reused} - \text{native})/\text{native}$; positive values indicate a reuse penalty. For MAE and CRPS, the lower value is in **bold** and the higher value is underlined.

Dataset	H	MAE			CRPS		
		Native	Reused	Δ	Native	Reused	Δ
ETTh2	192	**2.557**	2.675	+4.6%	**2.137**	2.259	+5.7%
	336	**2.713**	2.817	+3.8%	**2.374**	2.415	+1.7%
	720	3.005	**2.973**	−1.1%	2.599	**2.590**	−0.3%
ETTm1	192	**1.691**	1.724	+2.0%	**1.415**	1.468	+3.7%
	336	**1.802**	1.937	+7.5%	**1.516**	1.686	+11.2%
	720	**2.082**	2.535	+21.8%	**1.780**	2.260	+27.0%
ETTm2	192	**2.252**	2.265	+0.6%	**1.941**	**1.941**	0.0%
	336	2.522	**2.515**	−0.3%	2.252	**2.195**	−2.5%
	720	**2.771**	2.905	+4.8%	**2.498**	2.585	+3.5%
Weather	192	**10.592**	10.982	+3.7%	**9.020**	9.356	+3.7%
	336	**10.900**	12.103	+11.0%	**9.358**	10.495	+12.1%
	720	**12.165**	14.305	+17.6%	**10.835**	12.715	+17.4%

Table 8 reports the full native-versus-reused comparison for Augure. The reused model is always the checkpoint trained at H_0=96 and then applied for additional chunks at the target horizon H. Native values come from a separate Augure checkpoint trained directly at that same H. The reported Δ therefore isolates the cost of reusing the short-horizon operator on an otherwise identical evaluation task.

B.3 Full Mechanistic Ablations

Table 9: Full mechanistic-ablation results (MAE/CRPS) in original data units. Each variant is trained natively at the displayed horizon. Lower is better; best values are in **bold** and second-best distinct values are underlined.

Dataset	Variant	H=96		H=192		H=336		H=720	
		MAE	CRPS	MAE	CRPS	MAE	CRPS	MAE	CRPS
ETTh2	Augure	2.466	**2.044**	**2.557**	**2.137**	**2.713**	2.374	3.005	**2.599**
	Direct block	2.706	2.381	2.862	2.510	2.938	2.600	3.271	2.994
	Fixed chunk	2.648	2.236	2.679	2.334	2.918	2.562	**2.952**	2.657
	Scalar depth	**2.453**	2.059	2.612	2.206	2.789	**2.372**	3.217	2.922
Weather	Augure	**9.816**	**8.197**	**10.592**	**9.020**	**10.900**	**9.358**	**12.165**	**10.835**
	Direct block	10.893	9.582	11.550	10.196	12.454	11.196	12.950	11.875
	Fixed chunk	12.250	10.437	12.578	10.747	14.278	12.765	16.653	15.452
	Scalar depth	18.145	16.164	22.503	19.804	23.092	20.420	34.540	29.001
ETTm2	Augure	**1.993**	**1.669**	**2.252**	**1.941**	2.522	2.252	**2.771**	**2.498**
	Direct block	2.150	1.896	2.336	2.095	2.560	2.318	2.979	2.732
	Fixed chunk	2.017	1.700	2.316	2.043	**2.538**	**2.252**	2.858	2.613
	Scalar depth	2.415	2.080	2.609	2.335	2.706	2.398	2.985	2.609

Table 9 reports the appendix mechanistic results omitted from the main ETTm1 table: ETTh2, ETTm2, and Weather. ETTm2 was evaluated natively at all four horizons after the initial H=96 screening run.

B.4 Mechanistic Controls Under Operator Reuse

Tables 10 and 11 evaluate every $H_0{=}96$ mechanistic checkpoint under additional operator applications. Each cell reports reused error and its percentage difference from the corresponding native-horizon variant. Raw error compares variants at the same extrapolated horizon; the percentage measures native-relative reuse cost.

Table 10: Raw-unit MAE under H=96 checkpoint reuse. Each entry is *reused MAE / percentage difference from the native-horizon variant*; positive percentages indicate a reuse penalty and negative percentages indicate an improvement. Bold and underlined values identify the best and second-best reused MAE at each dataset and horizon; the comparison percentages remain unstyled. Augure denotes the standard Augure checkpoint.

Dataset	Variant	$H{=}96$	$H{=}192$	$H{=}336$	$H{=}720$
ETTh2	Augure	2.466 / –	2.675 / +4.6%	2.817 / +3.8%	**2.973** / −1.1%
	Direct block	2.672 / –	2.853 / +0.7%	3.057 / +4.4%	3.401 / +3.0%
	Fixed chunk	2.597 / –	2.898 / +8.9%	3.080 / +2.1%	3.341 / +11.7%
	Scalar depth	**2.401** / –	**2.606** / +0.6%	**2.727** / −3.6%	3.249 / +0.6%
ETTm1	Augure	1.514 / –	1.724 / +1.9%	1.937 / +7.5%	2.535 / +21.8%
	Direct block	1.594 / –	1.755 / −0.0%	**1.904** / +0.8%	**2.340** / +12.4%
	Fixed chunk	**1.479** / –	1.689 / −6.5%	1.916 / −5.0%	2.522 / −7.2%
	Scalar depth	1.484 / –	**1.683** / +3.8%	2.076 / +14.7%	2.725 / +35.8%
ETTm2	Augure	1.993 / –	2.265 / +0.5%	**2.515** / −0.3%	**2.905** / +4.8%
	Direct block	2.141 / –	2.397 / +2.8%	2.648 / +3.2%	3.034 / +5.5%
	Fixed chunk	2.020 / –	2.350 / −1.0%	2.693 / +4.2%	3.066 / +1.3%
	Scalar depth	**1.932** / –	**2.223** / +0.5%	2.508 / +3.7%	2.972 / +5.4%
Weather	Augure	**9.816** / –	**10.982** / +3.7%	12.103 / +11.0%	14.305 / +17.6%
	Direct block	10.661 / –	11.302 / −0.2%	**12.008** / +1.8%	**13.404** / +6.3%
	Fixed chunk	10.334 / –	11.341 / +2.8%	12.549 / +6.6%	15.087 / −1.7%
	Scalar depth	10.138 / –	11.563 / +5.2%	15.771 / +40.8%	28.934 / +132.7%

Table 11: Raw-unit CRPS under H=96 checkpoint reuse. Each entry is *reused CRPS / percentage difference from the native-horizon variant*; positive percentages indicate a reuse penalty and negative percentages indicate an improvement. Bold and underlined values identify the best and second-best reused CRPS at each dataset and horizon; the comparison percentages remain unstyled.

Dataset	Variant	$H{=}96$	$H{=}192$	$H{=}336$	$H{=}720$
ETTh2	Augure	2.044 / –	2.259 / +5.7%	2.415 / +1.7%	**2.590** / −0.3%
	Direct block	2.356 / –	2.551 / +2.6%	2.775 / +6.8%	3.137 / +3.6%
	Fixed chunk	2.175 / –	2.468 / +7.0%	2.654 / +0.3%	2.908 / +8.9%
	Scalar depth	**2.006** / –	**2.226** / +1.8%	**2.354** / −2.6%	2.743 / −6.6%
ETTm1	Augure	1.262 / –	1.468 / +3.8%	1.686 / +11.2%	2.260 / +27.0%
	Direct block	1.392 / –	1.552 / −0.2%	1.705 / +1.2%	**2.132** / +14.0%
	Fixed chunk	1.234 / –	1.435 / −6.0%	**1.661** / −5.8%	2.246 / −7.7%
	Scalar depth	**1.218** / –	**1.403** / +4.5%	1.786 / +16.9%	2.359 / +32.9%
ETTm2	Augure	1.669 / –	1.941 / −0.0%	**2.195** / −2.6%	**2.585** / +3.5%
	Direct block	1.889 / –	2.155 / +2.9%	2.421 / +4.1%	2.803 / +6.1%
	Fixed chunk	1.691 / –	2.014 / −2.1%	2.354 / +3.4%	2.716 / +0.6%
	Scalar depth	**1.614** / –	**1.913** / +2.0%	2.205 / +5.4%	2.598 / +2.3%
Weather	Augure	**8.197** / –	**9.356** / +3.7%	**10.495** / +12.2%	12.715 / +17.3%
	Direct block	9.365 / –	10.047 / +0.1%	10.817 / +2.1%	**12.191** / +5.3%
	Fixed chunk	8.704 / –	9.730 / +3.6%	10.983 / +7.3%	13.623 / −3.4%
	Scalar depth	8.471 / –	9.779 / +5.6%	13.323 / +38.9%	24.174 / +118.4%

The reuse trajectories distinguish the mechanistic variants beyond their native-horizon behavior. At $H{=}720$, the full model retains the lowest reused MAE/CRPS on ETTh2 and ETTm2, the Direct Block ablation degrades more slowly on ETTm1 and Weather, and Scalar Depth becomes strongly unstable on Weather, reaching $+132.7\%$ MAE and $+118.4\%$ CRPS native-relative reuse cost. These patterns are consistent with a fidelity–stability trade-off across the operator family. Because the percentages are self-relative, absolute reused error remains the appropriate quantity for comparing variants at a fixed composition depth.

B.5 CHUNK-SIZE AND MASKING-PROBABILITY TEST SWEEP

Table 12: Weather test-set results for the chunk-size and masking-schedule sweep, shown only as a post-selection check. The fixed default $B{=}6, m{=}0.5$ uses a fixed per-patch Bernoulli masking probability and is selected from validation; $m \sim \mathcal{U}(0,1)$ is the preliminary variable-probability control. Lower is better; best values are in **bold** and second-best distinct values are underlined.

Config	$H{=}96$		$H{=}192$		$H{=}336$		$H{=}720$	
	MAE	CRPS	MAE	CRPS	MAE	CRPS	MAE	CRPS
$B{=}3$, $m{\sim}\mathcal{U}(0,1)$	9.944	8.586	10.808	9.491	11.479	10.403	12.512	11.464
$B{=}3$, $m{=}0.3$	**9.479**	**8.073**	**10.570**	9.261	11.047	9.895	**12.145**	11.005
$B{=}3$, $m{=}0.5$	10.156	8.725	10.701	9.376	11.536	10.416	12.893	11.861
$B{=}3$, $m{=}0.7$	10.089	8.829	11.010	9.819	11.376	10.188	12.323	11.335
$B{=}6$, $m{=}0.3$	9.842	8.289	10.901	9.226	11.439	9.852	13.409	11.799
$B{=}6$, $m{=}0.5$	9.816	8.197	10.592	**9.020**	**10.900**	**9.358**	12.165	10.835
$B{=}6$, $m{=}0.7$	10.139	8.673	11.167	9.692	11.342	9.934	12.155	10.956
$B{=}6$, $m{\sim}\mathcal{U}(0,1)$	10.196	8.667	11.129	9.737	11.604	10.066	12.212	11.007
$B{=}12$, $m{=}0.3$	10.549	8.711	11.175	9.326	12.332	10.350	13.775	11.710
$B{=}12$, $m{=}0.5$	10.856	9.004	11.195	9.360	11.864	10.311	12.510	10.736
$B{=}12$, $m{=}0.7$	10.536	8.918	10.892	9.370	11.684	10.076	12.164	**10.653**
$B{=}12$, $m{\sim}\mathcal{U}(0,1)$	10.942	9.210	11.266	9.553	11.728	10.025	12.302	10.857