AN IRREGULAR PATCH IS A LOCAL FUNCTION: RETHINKING PATCH REPRESENTATION FOR IRREGULAR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Patch-based forecasters for irregular multivariate time series have mainly improved *where* patches are placed; we study *what* a patch should represent. For a geometry-weighted mean, a locally linear signal incurs a level error equal to its local slope times the weighted offset of observations from the patch center. We therefore represent each patch as a weighted local-linear estimate of its level and change. The closed-form estimator is exact for locally linear signals and retains second-order bias under non-degenerate asymmetric sampling, where the weighted mean becomes first-order. It uses only timestamps and response-independent support weights, so it applies to fixed hard windows and adaptive soft supports. Replacing the patch interior lowers MSE and MAE in all eight dataset–metric pairs for both a fixed-grid PatchTST-style backbone and APN; the fixed-grid comparison shares hard weights, whereas APN variants share the support parameterization but are trained separately. Our full model, Adaptive Kernel Patching (AKP), adds sample-conditioned supports, support-mass-gated aggregation, and cross-variable mixing. Against 17 baselines, each tuned with 100 Optuna trials and evaluated over five final runs, AKP has the best mean on five of eight pairs; on USHCN it improves MSE/MAE by $4.0\%/17.4\%$ over the strongest competitor.

1 INTRODUCTION

Irregular multivariate series occur in clinical, climate, and sensor data, where observations are uneven in time and misaligned across variables (Silva et al., 2012; Johnson et al., 2016; Menne et al., 2009; Kaluža et al., 2010; Shukla & Marlin, 2020). Patching compresses these observations into tokens for standard backbones (Nie et al., 2023; Zhang et al., 2024). Existing irregular patchers focus on *where* a patch is formed, through fixed, hierarchical, or adaptive supports (Zhang et al., 2024; Luo et al., 2025; Liu et al., 2026); we ask *what* the patch should represent.

Patch interiors belong to different estimator classes. PatchTST projects raw windows without within-window timestamps; APN uses a timestamp-dependent weighted mean; and the TTCN reader of t-PatchGNN/TFMixer learns value-dependent filters (Nie et al., 2023; Liu et al., 2026; Zhang et al., 2024; Qiu et al., 2026). Our analysis targets APN's geometry-weighted local-constant class and its hard-window analogue. In Figure 1b, the same linear trend is observed on the left or on the right half of a patch, and pooling returns two different values. For a locally linear signal, this error equals the local slope times the weighted sampling offset (Proposition 1); under non-degenerate asymmetric sampling, the smooth-signal bias is first rather than second order. Thus a weighted mean entangles dynamics with sampling geometry.

Local-linear regression removes this leading design-bias term (Fan, 1992; 2018). We use its closed-form weighted estimate of level and width-scaled change as the patch token (Figure 1c). It depends only on timestamps and support weights, while separate time and mass channels preserve sampling information.

Building on this representation, *Adaptive Kernel Patching* (AKP) fits local-linear patches on sample-conditioned soft supports, pools them using support mass, and mixes variables before query-time decoding.

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

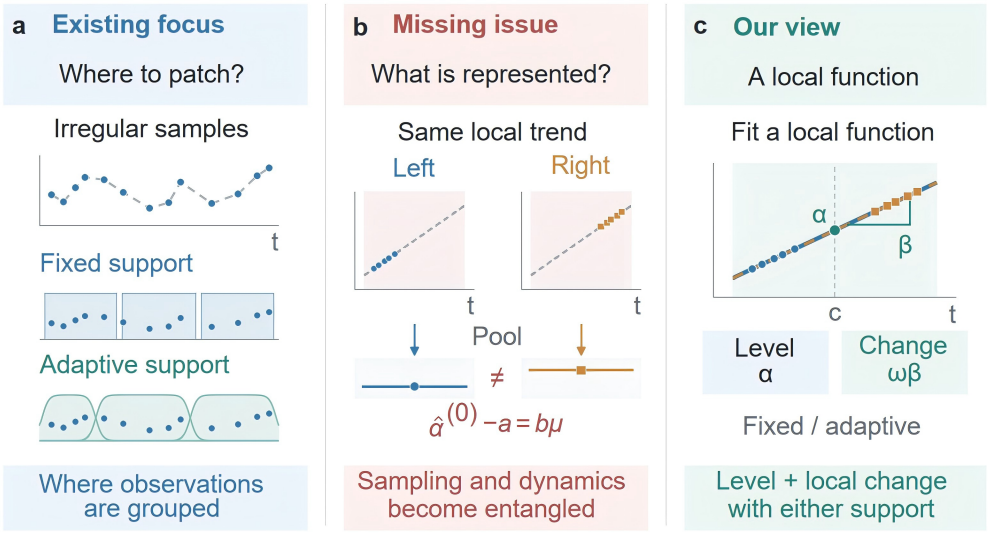


Figure 1: **From patch support to patch representation.** (a) Prior work decides where observations are grouped. (b) Under irregular sampling, pooling mixes the local trend with where the samples fall. (c) We fit a local function and keep its level α and change $\omega\beta$, on fixed or adaptive supports.

Across four benchmarks and 17 baselines, each with 100 Optuna trials and five final runs, AKP has the best mean on five of eight metric columns. The interior replacement improves both a fixed grid and APN on every dataset, and synthetic bias rates match the predicted first-/second-order behavior. Our contributions are as follows.

- We identify patch representation as a design axis and characterize the sampling-offset bias of local-constant patch summaries.
- We propose closed-form local-linear patching, which estimates level and change from timestamps and response-independent support weights.
- We present AKP and validate the representation on two backbones, four benchmarks, and controlled synthetic and corruption studies.

2 Related Work

Irregular multivariate time series. Irregular-series forecasters use decay or imputation, continuous-time dynamics, set and attention encoders, graph structures, or spectral mixing to relate timestamped observations (Che et al., 2018; Chen et al., 2018; Kidger et al., 2020; Biloš et al., 2021; Schirmer et al., 2022; Horn et al., 2020; Shukla & Marlin, 2021; Zhang et al., 2022; Yalavarthi et al., 2024; Li et al., 2025; Qiu et al., 2026). These methods differ in their temporal and cross-variable backbones; our work targets the patch token consumed by such backbones.

Patching for irregular series. Patching compresses a long series into local tokens (Nie et al., 2023; Zhang & Yan, 2023). For irregular data, t-PatchGNN uses fixed-duration time-aware patches, Hi-Patch builds multiscale patch graphs, and APN learns sample-dependent soft boundaries (Zhang et al., 2024; Luo et al., 2025; Liu et al., 2026). This line of work has mainly advanced the support: where a patch begins and ends and how softly it weights observations. The within-support summary is typically a weighted mean or a generic learned encoder. We leave the support mechanism open and instead replace the weighted-mean interior with a local function estimate.

Local polynomial regression. Kernel-weighted means of the Nadaraya–Watson type (Nadaraya, 1964; Watson, 1964) underlie irregular-series encoders: IPN uses an RBF-weighted mean and its kernel mass as an intensity channel, while mTAN forms learned kernel averages through time attention (Shukla & Marlin, 2019; 2021). Classical local-polynomial regression shows that local-constant

fits carry a design bias when observations are asymmetric around the target, whereas local-linear fitting removes the leading term and adapts to the design (Cleveland, 1979; Fan, 1992; 1993; 2018). We bring this estimator into patch tokenization and retain kernel mass as a support descriptor, not calibrated uncertainty.

3 METHOD

Problem setup. For one sample, let $\mathbf{X} \in \mathbb{R}^{l \times n_v}$ and $\mathbf{M} \in \{0,1\}^{l \times n_v}$ denote the historical values and the observation mask of n_v variables at timestamps $\mathbf{t} \in \mathbb{R}^l$. Given future query timestamps $t^{\mathrm{f}}_1, \ldots, t^{\mathrm{f}}_{l_f}$, the task is to predict the values $y^{(n)}_j$ of every variable n at every query t^{f}_j; only observed targets are evaluated. We form n_p patches per variable, and indices i, p, and n refer to observations, patches, and variables. A patching model makes two choices for each patch. Its *support* is a set of weights $w^{(n)}_{p,i} \geq 0$ that decides how much observation i belongs to patch p; its *representation* is what the model computes from the weighted observations. Section 3.1 addresses the representation. Sections 3.2–3.4 describe how AKP builds the support, aggregates patches, and forecasts (Figure 2).

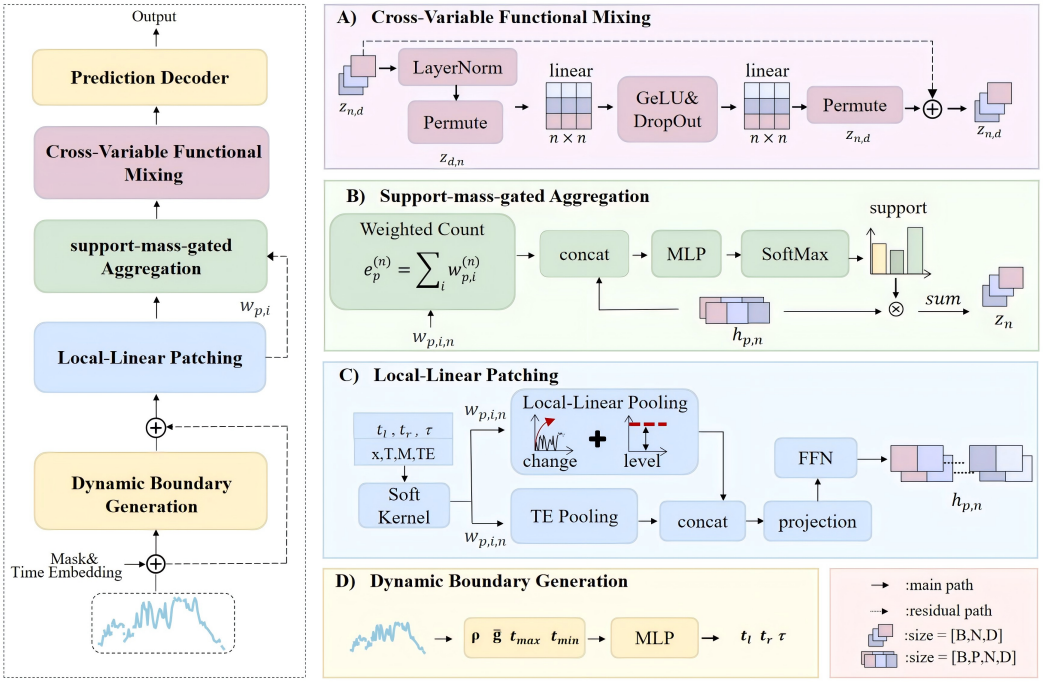


Figure 2: **Overview of AKP.** Dynamic boundary generation (D) places soft supports; local-linear patching (C) estimates each patch's level and change on its support; support-mass-gated aggregation (B) pools the patches of a variable using their support mass; cross-variable mixing (A) precedes a timestamp-conditioned decoder.

3.1 LOCAL-LINEAR PATCH REPRESENTATION

Local-constant patches entangle dynamics and sampling. Consider variable n and patch p with reference time $c^{(n)}_p$, the center of its support. The geometry-weighted patch summary considered here is the weighted mean $\hat{\alpha}^{(0,n)}_p = \sum_i w^{(n)}_{p,i} x^{(n)}_i \big/ \sum_i w^{(n)}_{p,i}$, a local-constant estimate of the signal level at $c^{(n)}_p$. Write $\pi^{(n)}_{p,i} = w^{(n)}_{p,i} / \sum_j w^{(n)}_{p,j}$ for the normalized weights and $\mu^{(n)}_p = \sum_i \pi^{(n)}_{p,i} (t_i - c^{(n)}_p)$ for the weighted offset of the observations from the center.

Proposition 1 (Sampling–dynamics entanglement). *If the signal in patch p is locally linear, $x^{(n)}_i = a^{(n)}_p + b^{(n)}_p (t_i - c^{(n)}_p)$, then*

$$\hat{\alpha}^{(0,n)}_p - a^{(n)}_p = b^{(n)}_p \, \mu^{(n)}_p. \tag{1}$$

If the total weight is positive and the weighted variance of the timestamps is nonzero, weighted local-linear regression recovers $a_p^{(n)}$ and $b_p^{(n)}$ exactly and uniquely.

Equation (1) is an algebraic identity conditional on a realized set of weights. Its interpretation as design bias, and the controlled replacement studied below, additionally require the weights to be determined by sampling geometry—timestamps, masks, and support parameters—rather than by the observed responses. If a weighting network reads x_i, it implements a content-adaptive selector as well as a smoother; replacing it changes the weighting hypothesis class, not only the local polynomial order.

The error in Eq. (1) is the product of a property of the signal, the slope $b_p^{(n)}$, and a property of the sampling, the offset $\mu_p^{(n)}$. It vanishes only when the signal is flat or the observations are balanced around the center, and balance is exactly what irregular sampling does not provide. The same structure governs smooth signals: Proposition 2 (Appendix A) shows that under asymmetric sampling the local-constant bias is first-order in patch width, whereas the local-linear bias remains second-order.

Design. We therefore estimate the local function directly by weighted least squares,

$$\left(\hat{\alpha}_p^{(n)}, \hat{\beta}_p^{(n)}\right) = \arg\min_{\alpha,\beta} \sum_{i=1}^{l} w_{p,i}^{(n)} \left[x_i^{(n)} - \alpha - \beta\,(t_i - c_p^{(n)})\right]^2. \tag{2}$$

With $u_i = t_i - c_p^{(n)}$, weighted moments $S_k = \sum_i w_{p,i}^{(n)} u_i^k$ and $T_k = \sum_i w_{p,i}^{(n)} u_i^k x_i^{(n)}$, and $D = S_0 S_2 - S_1^2$, the solution is $\hat{\alpha}_p^{(n)} = (S_2 T_0 - S_1 T_1)/D$ and $\hat{\beta}_p^{(n)} = (S_0 T_1 - S_1 T_0)/D$, computed in one pass over the observations. This is the mathematical estimator analyzed below. The implementation uses the following explicit numerical extension for nearly empty or rank-deficient patches:

$$\widetilde{S}_0 = \max(S_0, \epsilon_D), \qquad \widetilde{D} = \max(\widetilde{S}_0 S_2 - S_1^2, \epsilon_D), \tag{3}$$

and evaluates $\hat{\beta} = (\widetilde{S}_0 T_1 - S_1 T_0)/\widetilde{D}$ and $\hat{\alpha} = (T_0 - \hat{\beta} S_1)/\widetilde{S}_0$, with $\epsilon_D = 10^{-6}$. This is a deterministic clamp, not the solution of Eq. (2) and not a scale-aware ridge estimator. We therefore treat clamped pairs as a stabilized implementation branch and invoke the exactness and bias-order results only for non-degenerate pairs; Appendix E.2 reports how often each branch is used. To remove the direct dependence of the slope on patch duration, we scale it by the patch width $\omega_p^{(n)}$ and use the value representation

$$\mathbf{z}_p^{(n)} = \left[\hat{\alpha}_p^{(n)},\ \omega_p^{(n)} \hat{\beta}_p^{(n)}\right]^{\mathsf{T}}, \tag{4}$$

so that the second entry is the change of the fitted line across the patch. This width-scaled slope is a value channel, not a measurement of the support's effective radius; the latter depends on the full weight profile and is reported separately in Appendix E.2. The same weights pool a learnable encoding $\phi(t_i)$ of the observation timestamps (Shukla & Marlin, 2021; Kazemi et al., 2019) into $\bar{\boldsymbol{\phi}}_p^{(n)} = \sum_i \pi_{p,i}^{(n)} \phi(t_i)$. The concatenation $[\mathbf{z}_p^{(n)}; \bar{\boldsymbol{\phi}}_p^{(n)}]$ is mapped to the patch token $\mathbf{h}_p^{(n)} \in \mathbb{R}^d$ by a linear projection and a residual feed-forward block (Figure 2C).

Advantages. For a non-degenerate local support, the estimator removes the sampling term of Eq. (1) exactly for locally linear signals and reduces its order for smooth ones, at the cost of five weighted sums per patch, $O(n_v n_p l)$ operations per sample, and no iterative fitting. It depends only on timestamps and support weights, so it applies unchanged to hard fixed windows and learned time-only soft windows. Timing information is kept rather than discarded: where the observations lie is carried by $\bar{\boldsymbol{\phi}}_p^{(n)}$ and by the support mass of Section 3.3, in channels separate from the level and change of the signal.

3.2 Dynamic Boundary Generation

Variables in an irregular series differ in observation rate and observed span, so a single grid of patches fits some variables poorly. AKP conditions the supports on each variable's sampling pattern (Figure 2D). For each variable we compute five statistics: the observation ratio, the mean gap between observations, the observed span, and the first and last observation times. A small MLP maps these statistics to residual adjustments of learnable per-patch positions, widths, and temperatures, which

are initialized so that patch p covers $[pS, (p+1)S]$ for patch spacing S. The result is a left boundary $\ell_p^{(n)}$, a right boundary $r_p^{(n)}$, and a temperature $\tau_p^{(n)}$, which define the soft support

$$w_{p,i}^{(n)} = m_i^{(n)} \, \sigma\Big(\frac{r_p^{(n)} - t_i}{\tau_p^{(n)}}\Big) \, \sigma\Big(\frac{t_i - \ell_p^{(n)}}{\tau_p^{(n)}}\Big), \tag{5}$$

with center $c_p^{(n)} = (\ell_p^{(n)} + r_p^{(n)})/2$ and width $\omega_p^{(n)} = r_p^{(n)} - \ell_p^{(n)}$. The two sigmoids form differentiable boundaries and the mask $m_i^{(n)}$ removes unobserved entries. Eq. (5) decides *where* the local function is estimated, and Eq. (2) decides *what* is estimated there; the two are learned jointly but can be replaced independently. The parameterization is detailed in Appendix C, with annotated core code in Appendix G.

3.3 Support-Mass-Gated Aggregation

Patches of equal duration can rest on very different numbers of observations, and an estimate from a nearly empty patch deserves less weight than one from a well-populated patch. We summarize the support of each estimate by the mass that its window collects,

$$e_p^{(n)} = \sum_i w_{p,i}^{(n)}. \tag{6}$$

Because $w_{p,i}^{(n)} \in [0,1]$, $e_p^{(n)}$ is a differentiable observation count: an observation well inside the support counts fully, one on a soft boundary counts partially, and for hard windows $e_p^{(n)}$ equals the number of observations in the patch. It plays the role of the intensity channel of Shukla & Marlin (2019), here attached to each patch estimate; we use it as a descriptor of support rather than as a calibrated uncertainty. A gate reads the patch token together with its support mass and pools the patches of each variable (Figure 2B), in the manner of attention-based pooling (Ilse et al., 2018):

$$g_p^{(n)} = \mathrm{MLP}_{\mathrm{gate}}\left([\mathbf{h}_p^{(n)}; \log(e_p^{(n)} + \epsilon_e)]\right), \qquad \gamma_p^{(n)} = \mathrm{softmax}_p\left(g_p^{(n)}\right), \qquad \mathbf{h}^{(n)} = \sum_{p=1}^{n_p} \gamma_p^{(n)} \mathbf{h}_p^{(n)}. \tag{7}$$

where $\epsilon_e = 10^{-9}$ only protects the logarithm and is distinct from the regression stabilizer ϵ_D.

3.4 Cross-Variable Mixing and Forecasting

The variable representations are stacked into $\mathbf{H} \in \mathbb{R}^{n_v \times d}$ and passed through a residual MLP that mixes along the variable axis (Figure 2A), in the spirit of token mixing in MLP-Mixer and TSMixer (Tolstikhin et al., 2021; Chen et al., 2023). Its last linear layer is zero-initialized, so the module starts as the identity mapping and learns cross-variable interactions from there. For each query t_j^{f}, a learnable encoding of t_j^{f} is concatenated with the mixed representation of variable n and decoded by a two-layer MLP into $\hat{y}_j^{(n)}$. With $m_{j,\mathrm{f}}^{(n)} \in \{0,1\}$ indicating observed targets, AKP is trained end-to-end with the masked Huber loss (Huber, 1964)

$$\mathcal{L} = \frac{\sum_{n=1}^{n_v} \sum_{j=1}^{l_f} m_{j,\mathrm{f}}^{(n)} \, \ell_\delta\big(\hat{y}_j^{(n)} - y_j^{(n)}\big)}{\sum_{n=1}^{n_v} \sum_{j=1}^{l_f} m_{j,\mathrm{f}}^{(n)}}, \qquad \delta = 0.1. \tag{8}$$

4 Experiments

4.1 Setup

Datasets. We use four standard benchmarks for irregular multivariate forecasting with a common preprocessing pipeline: PhysioNet 2012 (P12; 36 post-processed clinical variables, observations through 35.5 h, then the next three target timestamps) (Silva et al., 2012), MIMIC-III (MIMIC; 96 variables, observations through 35.75 h, then the next three target timestamps) (Johnson et al., 2016), HumanActivity (HA; 12 sensor coordinates, 3,000 ms history and 300 ms horizon) (Kaluža et al., 2010), and USHCN (5 climate variables, the first three years and the next three measurements) (Menne et al., 2009). Appendix B gives the processed instance counts, exact splits, and metrics.

Baselines. We compare with 17 methods in five groups: recurrent and continuous-time models (GRU-D (Che et al., 2018), NeuralFlow (Biloš et al., 2021), CRU (Schirmer et al., 2022)); set- and attention-based models (SeFT (Horn et al., 2020), mTAN (Shukla & Marlin, 2021), Warpformer (Zhang et al., 2023), PrimeNet (Chowdhury et al., 2023)); graph-based models (Raindrop (Zhang et al., 2022), GraFITi (Yalavarthi et al., 2024), GNeuralFlow (Mercatali et al., 2024), HyperIMTS (Li et al., 2025)); patch-based and mixing models (t-PatchGNN (Zhang et al., 2024), APN (Liu et al., 2026), TFMixer (Qiu et al., 2026)); and regular-time forecasters (PatchTST (Nie et al., 2023), iTransformer (Liu et al., 2024), TimeMixer (Wang et al., 2024)).

Protocol. Each method is tuned separately on each dataset with a 100-trial Optuna search (Akiba et al., 2019); search spaces are method-specific, while splits, preprocessing, and validation MSE selection are shared. The selected configuration is retrained from scratch for five independent runs (seeds 2024–2028), and we report the mean MSE and MAE on observed future targets. The main comparison and plug-in study use this five-run protocol; architectural variants, sensitivity analyses, and controlled diagnostics marked in the text use three runs (2024–2026). Their full-model references can therefore differ slightly, e.g., USHCN MSE is 0.1520 in Table 1 versus 0.1504 in Table 3. Search dimensions, stopping details, and standard deviations are deferred to Appendices D and F.

4.2 Main Results

Table 1: Forecasting error on four benchmarks (mean of five runs; standard deviations in Appendix D). Lower is better; **bold** and underlined mark the best and second-best result in each column.

Method	HumanActivity		USHCN		PhysioNet		MIMIC	
	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓
Recurrent and continuous-time								
GRU-D (2018)	0.0889	0.2093	0.2097	0.3045	0.3419	0.3992	0.4753	0.4542
NeuralFlow (2021)	0.1024	0.2253	0.1986	0.3109	0.3806	0.4285	0.6021	0.5279
CRU (2022)	0.1344	0.2479	0.2168	0.3180	0.6128	0.5741	0.5717	0.5065
Set and attention-based								
SeFT (2020)	1.3650	0.9706	0.3358	0.4104	0.7773	0.6757	0.9224	0.6624
mTAN (2021)	0.0709	0.1839	0.1915	0.3257	0.3801	0.4281	0.7725	0.6124
Warpformer (2023)	0.0416	0.1158	0.1888	0.2939	0.3069	0.3661	0.4081	0.3896
PrimeNet (2023)	4.2500	1.7015	0.4924	0.4947	0.7952	0.6858	0.9071	0.6613
Graph-based								
Raindrop (2022)	0.0814	0.1963	0.2406	0.3187	0.3515	0.4083	0.4939	0.4554
GraFITi (2024)	0.0410	**0.1120**	0.1683	0.2668	0.3057	0.3652	0.3684	0.3695
GNeuralFlow (2024)	0.5588	0.5106	0.2205	0.3286	0.8207	0.6759	0.8800	0.6369
HyperIMTS (2025)	0.0414	0.1147	0.1979	0.2829	**0.3005**	0.3598	**0.3567**	0.3687
Patch-based and mixing								
t-PatchGNN (2024)	0.0433	0.1226	0.1666	0.2718	0.3091	0.3650	0.4343	0.4009
APN (2026)	0.0431	0.1203	0.1953	0.2934	0.3141	0.3674	0.5308	0.4550
TFMixer (2026)	0.0429	0.1164	0.1583	0.2604	0.3203	0.3666	0.4094	0.3992
Regular-time								
PatchTST (2023)	0.0731	0.1747	0.1815	0.2736	0.3402	0.3969	0.5811	0.4612
iTransformer (2024)	0.0811	0.1862	0.2767	0.3414	0.3851	0.4324	0.5571	0.4732
TimeMixer (2024)	0.0564	0.1497	0.1829	0.2823	0.3188	0.3711	0.4692	0.4317
AKP (ours)	**0.0408**	0.1128	**0.1520**	**0.2152**	0.3076	**0.3513**	0.4033	**0.3680**

AKP has the best five-run mean on five of the eight dataset–metric pairs in Table 1 and the second-best mean on a sixth. Its largest margin is on USHCN, where it reaches an MSE of 0.1520 and an MAE of 0.2152, 4.0% and 17.4% below TFMixer, the strongest competitor on that dataset. It also has the lowest mean MSE on HumanActivity (0.0408 against 0.0410 for GraFITi) and the lowest MAE on PhysioNet (0.3513 against 0.3598 for HyperIMTS, −2.4%) and MIMIC (0.3680 against 0.3687). Against the two irregular patching baselines, t-PatchGNN and APN, AKP is lower on all eight dataset–metric pairs. The main table is comparative context; the more diagnostic evidence for our contribution is the controlled interior replacement in Table 2; its fixed-grid rows share hard weights, whereas its APN rows share the support mechanism but are separately trained. On USHCN, it also

trains faster per sample than TFMixer and HyperIMTS (Appendix E.8), and across the five seeds its standard deviation is at most 0.0062 on every dataset–metric pair (Appendix D).

HyperIMTS attains the lowest MSE on PhysioNet and MIMIC (0.3005 and 0.3567, against 0.3076 and 0.4033 for AKP), and GraFITi the lowest MAE on HumanActivity (0.1120 against 0.1128). On these two clinical datasets AKP ranks higher in MAE than in MSE. On PhysioNet, part of this gap is consistent with the Huber objective of Eq. (8), which penalizes residuals larger than $\delta = 0.1$ linearly rather than quadratically: the loss sensitivity in Appendix F lowers MSE only at the cost of higher MAE. The same experiment does not support a loss-only explanation on MIMIC-III, where using MSE as the training objective does not improve test MSE.

4.3 Local-Linear Patching as a Plug-In

Proposition 1 concerns the patch interior, not whether a response-independent support is fixed or adaptive. We test this on two backbones with opposite supports: PatchTST-style patching on a fixed grid of hard windows, and APN, which learns soft adaptive supports. On the fixed grid, variant (2) averages the observed values in each window and variant (3) fits the local-linear estimate on the same hard-window weights; variant (1) is the original, mask-blind PatchTST. On APN, variant (5) replaces APN's weighted average (4) with the local-linear estimate under the same support parameterization. Because the two APN variants are tuned and trained separately, their learned boundary values need not be identical; this is an architecture-level interior comparison, not a frozen-support counterfactual. Each variant is tuned separately and trained with the MSE loss. The controlled comparisons are $(2) \to (3)$ and $(4) \to (5)$, which keep the backbone, loss, and downstream decoder family fixed while changing the patch interior. Feature-dimension-matched controls are reported separately in Appendix D.3.

Why these two backbones. The mechanism-isolating comparison requires value-independent weights $w_{p,i} = w_{p,i}(t, m)$. APN satisfies this condition, so its weighted-mean and local-linear interiors test the same support mechanism; PatchTST supplies the matched hard-window analogue. We do not transplant the estimator into t-PatchGNN or TFMixer because their TTCN reader computes value-dependent softmax weights. Replacing that reader would change both estimator order and weighting hypothesis class, so any result would not isolate Proposition 1. Both methods remain main-table baselines.

Table 2: Patch-interior variants. Parentheses give the change relative to the row above within each backbone; row (1) is the original PatchTST. Mean of five runs; each variant is tuned separately. The fixed-grid rows share hard weights, while the APN rows share the support mechanism but not frozen learned boundary values.

Backbone / interior	HumanActivity		USHCN		PhysioNet		MIMIC	
	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓
Fixed grid (PatchTST-style)								
(1) Original, mask-blind	0.0731	0.1747	0.1815	0.2736	0.3402	0.3969	0.5811	0.4612
(2) Local constant	0.0460 (–37.1%)	0.1239 (–29.1%)	0.1961 (+8.0%)	0.2775 (+1.4%)	0.3402 (0.0%)	0.3813 (–3.9%)	0.5780 (–0.5%)	0.4566 (–1.0%)
(3) Local linear	**0.0429** (–6.7%)	**0.1172** (–5.4%)	**0.1679** (–14.4%)	**0.2586** (–6.8%)	**0.3319** (–2.4%)	**0.3737** (–2.0%)	**0.5605** (–3.0%)	**0.4469** (–2.1%)
Adaptive soft supports (APN)								
(4) Local constant	0.0431	0.1203	0.1953	0.2934	0.3141	0.3674	0.5308	0.4550
(5) Local linear	**0.0419** (–2.8%)	**0.1174** (–2.4%)	**0.1861** (–4.7%)	**0.2903** (–1.1%)	**0.3095** (–1.5%)	**0.3654** (–0.5%)	**0.4272** (–19.5%)	**0.4013** (–11.8%)

The feature-dimension-matched controls and their uncertainty are reported in Appendix D.3; they use the same token width as the local-linear arm and are not part of the main ranking.

The local-linear interior improves both backbones on every dataset. Changing only the interior lowers MSE and MAE in all eight backbone–dataset settings of Table 2. MSE falls by 2.4–14.4% on the fixed grid and by 1.5–19.5% on APN. On MIMIC, APN also becomes far more stable: the empirical standard deviation of MSE drops from 0.1972, where one baseline run diverges, to 0.0050. The fixed grid uses hard windows, so its patches are local by construction, whereas APN's supports

are learned; the estimator helps in both. The two changes on the fixed grid are also separable: on USHCN, making PatchTST mask-aware raises MSE by 8.0%, and the local-linear interior then lowers it by 14.4%. Appendix D.4 removes the remaining APN geometry ambiguity on three datasets by freezing the local-constant support parameters before fitting the local-linear interior.

4.4 Fixed-Configuration Architectural Variants

Table 3 reports controlled architectural variants of AKP. Each row replaces one operation while retaining the remaining stages: the local-linear interior becomes a local-constant mean, the dynamic boundary becomes a sample-independent base support, the variable mixer becomes an identity map, or the gate input/aggregation is simplified. All variants reuse the hyperparameters and three seeds of the full model. The first three rows replace the indicated operation in the shared forward path, while the final two are dedicated gate ablations. The table is therefore a fixed-configuration sensitivity analysis rather than a fully retuned causal attribution.

Table 3: Controlled architectural variants of AKP (mean of three runs, shared hyperparameters). Each row replaces one operation. Parentheses give the change relative to the full model; positive values mean higher error. Percentages are computed from the unrounded run means.

Variant	HumanActivity		USHCN		PhysioNet		MIMIC	
	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓
Full AKP	0.0408 (ref.)	0.1125 (ref.)	0.1504 (ref.)	0.2156 (ref.)	0.3077 (ref.)	0.3513 (ref.)	0.4030 (ref.)	0.3677 (ref.)
w/o local-linear patching	0.0425 (+4.1%)	0.1140 (+1.4%)	0.2334 (+55.2%)	0.2340 (+8.5%)	0.3101 (+0.8%)	0.3535 (+0.6%)	0.4108 (+1.9%)	0.3702 (+0.7%)
w/o dynamic boundary	0.0426 (+4.3%)	0.1183 (+5.2%)	0.1520 (+1.1%)	0.2139 (−0.8%)	0.3098 (+0.7%)	0.3540 (+0.8%)	0.4115 (+2.1%)	0.3736 (+1.6%)
w/o cross-variable mixing	0.0414 (+1.4%)	0.1120 (−0.4%)	0.2021 (+34.4%)	0.2611 (+21.1%)	0.3137 (+2.0%)	0.3540 (+0.8%)	0.4386 (+8.8%)	0.3823 (+4.0%)
w/o support mass in gate	0.0413 (+1.2%)	0.1125 (+0.1%)	0.1712 (+13.8%)	0.2223 (+3.1%)	0.3073 (−0.1%)	0.3519 (+0.2%)	0.4133 (+2.6%)	0.3693 (+0.5%)
w/o gate (uniform mean)	0.0410 (+0.5%)	0.1130 (+0.4%)	0.1584 (+5.3%)	0.2231 (+3.5%)	0.3070 (−0.2%)	0.3526 (+0.4%)	0.4033 (+0.1%)	0.3699 (+0.6%)

Local-linear interior. Replacing the local-linear summary by a local-constant one is the only tested operation replacement that raises error in all eight dataset–metric pairs. The largest effect is on USHCN ($+55.2\%$ MSE, $+8.5\%$ MAE), the benchmark with the most asymmetric patches (Section 4.5).

Dynamic boundary operation. Replacing the sample-conditioned residuals with sample-independent, learned base supports raises MSE on all four datasets ($+4.3\%$ on HumanActivity, $+1.1\%$ on USHCN, $+0.7\%$ on PhysioNet, $+2.1\%$ on MIMIC) and MAE on three; on USHCN, MAE is 0.8% lower without sample conditioning.

Cross-variable mixing operation. Replacing the mixing module by an identity map raises MSE on all four datasets, most on USHCN ($+34.4\%$ MSE, $+21.1\%$ MAE) and MIMIC ($+8.8\%$, $+4.0\%$); on HumanActivity, MAE changes by -0.4%. We attribute the MIMIC effect to sparsity: a variable holds about one observation per sample, so related variables supply most of its usable history.

Gated aggregation operation. In mean performance, a learned gate beats a uniform mean in MSE only when it reads the support mass. Reading patch content alone, the gate has higher MSE than a uniform mean on all four datasets (USHCN: 0.1712 against 0.1584); with the support mass, it has lower mean error in seven of eight dataset–metric pairs, all except PhysioNet MSE (0.3077 against 0.3070). However, replacing the full gate by a uniform mean does not clear two standard errors for MSE on any dataset. We therefore treat gating as an aggregation choice rather than as a separately established contribution. Within each (sample, variable) group, the learned gate weights correlate with $\log e_p^{(n)}$ at $\rho = +0.75$ on USHCN and $\rho = -0.58$ on HumanActivity, so the gate reweights patches by their support in a dataset-specific direction (Appendix E.5).

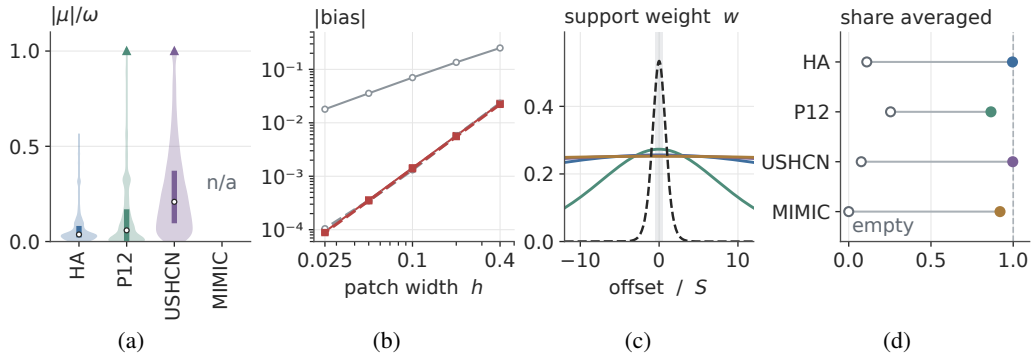


Figure 3: **Why local-linear patching helps.** (a) Test-patch asymmetry. (b) Synthetic level bias versus patch width; gray/red: local constant/linear, dashed: symmetric sampling. (c) Learned supports and a local $\tau = S/2$ window (dashed). (d) Observation share per patch; filled/open: learned $\tau/\tau = S/2$.

4.5 Why Local-Linear Patching Helps

Theory and data. Under asymmetric sampling, synthetic local-constant/local-linear bias has fitted order $0.95/1.99$ (predicted $1/2$); both are second-order under symmetry, and only the local-constant bias grows with sampling skew (Figure 3b; Appendix E.1). Test-timestamp asymmetry $|\mu|/\omega$ has median $0.21/0.06/0.04$ on USHCN/PhysioNet/HumanActivity (Figure 3a); MIMIC is too sparse for the effective-sample-size threshold (Kish, 1965).

The learned supports are wide. The boundary generator mainly shifts patches, by 1.8–12.2 patch spacings S; widths and temperatures stay near their initial values (Appendix E.4). With τ/S between 5.5 and 52.5, the soft supports are nearly flat (Figure 3c), so one patch averages 86–100% of its variable's observations, against 8–26% for a local window with $\tau = S/2$ (Figure 3d; almost none on MIMIC). In this regime the local-constant summaries of a variable collapse toward a single mean, while the local-linear level still changes with the patch center $c_p^{(n)}$. By Eq. (1) the gap is $b\,\mu_p^{(n)}$, which grows as $c_p^{(n)}$ moves away from the observations. We attribute part of the large ablation effect on USHCN to this combination of wide supports and asymmetric sampling; Appendix F tests fixed temperatures. These measurements are also a qualification of the word "local": in the trained AKP model, local-linear refers to the order of the interior fit, while the learned support can be broad and the fitted line can extrapolate to a patch center. The post-selection sweep alone does not establish superiority over a validation-selected fixed temperature; the separate comparison in Appendix F.3 gives dataset-dependent results.

Controlled irregularity. Across five corruption types, local-linear APN is resolved in $13/19$ PhysioNet settings but only $1/20$ USHCN settings (Appendix E.7). The gain is not monotone in gap variability, consistent with Eq. (1): slope and the signed within-patch offset μ, not irregularity alone, determine the correction.

5 Conclusion

We separate where an irregular patch is placed from what it represents. A closed-form weighted local-linear estimate removes the slope–sampling-offset term of a local-constant summary. The replacement improves fixed-grid and APN backbones on all four benchmarks; AKP has the best five-run mean on five of eight metric columns against 17 baselines. Patch support and patch representation are therefore distinct design axes.

Limitations and future work. The first-order interior cannot model abrupt curvature. In the trained model, learned supports are often broad, so "local-linear" describes the estimator order rather than a guaranteed local radius. The validation-selected fixed-temperature control covers only HumanActivity and USHCN and does not constrain effective radius. Sparse P12/MIMIC pairs also trigger numerical floors, outside the exactness claim. Future work should constrain support radius and add scale-aware sparse-data fallbacks.

Ethics Statement

This work uses four public benchmarks. PhysioNet 2012 and MIMIC-III contain de-identified clinical records, and MIMIC-III was accessed through credentialed access under its data use agreement; USHCN contains climate records and HumanActivity anonymized motion-sensor recordings. Forecasts on clinical data are research results and are not validated for clinical decision making.

Reproducibility Statement

Section 3 and Appendices C and G specify the model and its core implementation, and Appendix A contains complete proofs. Appendix B gives the datasets, splits, and metrics; Section 4.1 and Appendix F give the search protocol, seeds, and hyperparameters. We will release code and configuration files upon publication.

LLM Usage Statement

Large language models were used to assist with language editing, literature search, and LaTeX formatting. All methods, experiments, and results were designed, run, and verified by the authors, who take full responsibility for the content.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural odes. *Advances in neural information processing systems*, 34:21325–21337, 2021.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023.

Ranak Roy Chowdhury, Jiacheng Li, Xiyuan Zhang, Dezhi Hong, Rajesh K Gupta, and Jingbo Shang. Primenet: Pre-training for irregular multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7184–7192, 2023.

William S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, 1979.

Jianqing Fan. Design-adaptive nonparametric regression. *Journal of the American Statistical Association*, 87(420):998–1004, 1992.

Jianqing Fan. Local linear regression smoothers and their minimax efficiencies. *The Annals of Statistics*, 21(1):196–216, 1993.

Jianqing Fan. *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*. Routledge, 2018.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 2020.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35 (1):73–101, 1964.

Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *International conference on machine learning*, pp. 2127–2136. Pmlr, 2018.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):160035, 2016.

Boštjan Kaluža, Violeta Mirchevska, Erik Dovgan, Mitja Luštrek, and Matjaž Gams. An agent-based approach to care in independent living. In *International joint conference on ambient intelligence*, pp. 177–186. Springer, 2010.

Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33:6696–6707, 2020.

Leslie Kish. Sampling organizations and groups of unequal sizes. *American sociological review*, pp. 564–572, 1965.

Boyuan Li, Yicheng Luo, Zhen Liu, Junhao Zheng, Jianming Lv, and Qianli Ma. Hyperimts: Hypergraph neural network for irregular multivariate time series forecasting. *arXiv preprint arXiv:2505.17431*, 2025.

Xvyuan Liu, Xiangfei Qiu, Xingjian Wu, Zhengyu Li, Chenjuan Guo, Jilin Hu, and Bin Yang. Rethinking irregular time series forecasting: A simple yet effective baseline. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

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, 2024.

Yicheng Luo, Bowen Zhang, Zhen Liu, and Qianli Ma. Hi-Patch: Hierarchical patch GNN for irregular multivariate time series. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Matthew J Menne, Claude N Williams Jr, and Russell S Vose. The us historical climatology network monthly temperature data, version 2. *Bulletin of the American Meteorological Society*, 90(7): 993–1008, 2009.

Giangiacomo Mercatali, Andre Freitas, and Jie Chen. Graph neural flows for unveiling systemic interactions among irregularly sampled time series. In *Advances in Neural Information Processing Systems*, volume 37, pp. 57183–57206, 2024.

Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. 2023.

Xiangfei Qiu, Kangjia Yan, Xvyuan Liu, Xingjian Wu, and Jilin Hu. Bridging time and frequency: A joint modeling framework for irregular multivariate time series forecasting. In *ICML*, 2026.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *International conference on machine learning*, pp. 19388–19405. PMLR, 2022.

Satya Narayan Shukla and Benjamin Marlin. Interpolation-prediction networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2019.

Satya Narayan Shukla and Benjamin M. Marlin. A survey on principles, models and methods for learning from irregularly sampled time series. *arXiv preprint arXiv:2012.00168*, 2020.

Satya Narayan Shukla and Benjamin M Marlin. Multi-time attention networks for irregularly sampled time series. In ... *International Conference on Learning Representations*, volume 2021, pp. 14897, 2021.

Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 computing in cardiology*, pp. 245–248. IEEE, 2012.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 38626–38652, 2024.

Geoffrey S Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

Vijaya Krishna Yalavarthi, Kiran Madhusudhanan, Randolf Scholz, Nourhan Ahmed, Johannes Burchert, Shayan Jawed, Stefan Born, and Lars Schmidt-Thieme. Grafiti: Graphs for forecasting irregularly sampled time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16255–16263, 2024.

Jiawen Zhang, Shun Zheng, Wei Cao, Jiang Bian, and Jia Li. Warpformer: A multi-scale modeling approach for irregular clinical time series. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3273–3285, 2023.

Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. Irregular multivariate time series forecasting: A transformable patching graph neural networks approach. In *ICML*, pp. 60179–60196, 2024.

Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik. Graph-guided network for irregularly sampled multivariate time series. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Kwm8I7dU-l5.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023.

A PROOFS

We drop the superscript (n) and the subscript p when no confusion arises, write $u_i = t_i - c$ for the offset of observation i from the patch center, and use the normalized weights $\pi_i = w_i / \sum_j w_j$.

A.1 PROOF OF PROPOSITION 1

For the local-constant representation,

$$\hat{\alpha}^{(0)} = \sum_i \pi_i x_i = \sum_i \pi_i \, (a + b\, u_i) = a + b \sum_i \pi_i u_i = a + b\, \mu,$$

which is Eq. (1). For the local-linear estimator, the pair (a, b) attains zero weighted residual in Eq. (2), since $x_i = a + b\, u_i$ for every i. The normal matrix of Eq. (2) is $\mathbf{G} = \sum_i w_i \, [1, u_i]^\mathsf{T} [1, u_i]$, with determinant

$$\det(\mathbf{G}) = \Big(\sum_i w_i \Big) \Big(\sum_i w_i u_i^2 \Big) - \Big(\sum_i w_i u_i \Big)^2 = \Big(\sum_i w_i \Big)^2 \mathrm{Var}_\pi(u).$$

If $\sum_i w_i > 0$ and $\mathrm{Var}_\pi(u) > 0$, then $\mathbf{G}$ is nonsingular and the weighted least-squares solution is unique. Because (a, b) already attains zero residual, the unique solution is $\hat{\alpha} = a$ and $\hat{\beta} = b$. □

A.2 BIAS ORDERS FOR SMOOTH SIGNALS

The next result explains the rates in Figure 3b. Let $\sigma^2 = \mathrm{Var}_\pi(u) = \sum_i \pi_i u_i^2 - \mu^2$.

Proposition 2 (Bias orders). *Let the signal $x(t)$ be twice differentiable on the patch with $|x''| \le M$, and let all observations with positive weight satisfy $|u_i| \le h$. Then*

(i) the local-constant estimate satisfies $\hat{\alpha}^{(0)} - x(c) = x'(c)\, \mu + R_0$ with $|R_0| \le M h^2 / 2$;

(ii) if $\sigma \ge \kappa h$ for some $\kappa > 0$, the local-linear estimate satisfies $|\hat{\alpha} - x(c)| \le M h^2 / \kappa^2$.

Consequently, if the sampling is asymmetric at scale h, that is, $|\mu| \ge \eta h$ for a fixed $\eta > 0$, and $x'(c) \ne 0$, the local-constant bias is of order h, whereas the local-linear bias is of order h^2.

Proof. Taylor's theorem gives $x_i = x(c) + x'(c)\, u_i + r_i$ with $|r_i| \le M u_i^2 / 2 \le M h^2 / 2$. (i) Averaging with weights π_i gives $\hat{\alpha}^{(0)} = x(c) + x'(c)\, \mu + \sum_i \pi_i r_i$, and $|\sum_i \pi_i r_i| \le M h^2 / 2$. (ii) The local-linear intercept is a linear smoother, $\hat{\alpha} = \sum_i \ell_i x_i$ with equivalent weights $\ell_i = \pi_i \, (S_2 - \mu\, u_i) / \sigma^2$, where $S_2 = \sum_i \pi_i u_i^2$. These weights reproduce constants and lines: $\sum_i \ell_i = (S_2 - \mu^2)/\sigma^2 = 1$ and $\sum_i \ell_i u_i = (S_2 \mu - \mu S_2)/\sigma^2 = 0$. Hence $\hat{\alpha} - x(c) = \sum_i \ell_i r_i$ and $|\hat{\alpha} - x(c)| \le (M h^2 / 2) \sum_i |\ell_i|$. Since $S_2 \le h^2$ and $|\mu| \le h$, $\sum_i |\ell_i| \le (S_2 + |\mu| h)/\sigma^2 \le 2 h^2 / \sigma^2 \le 2/\kappa^2$, which proves (ii). The final statement follows from (i), since $|\hat{\alpha}^{(0)} - x(c)| \ge |x'(c)|\, \eta h - M h^2 / 2$. □

The condition $\sigma \ge \kappa h$ requires that the observations are not collapsed onto a single point relative to the patch width; it is the quantitative form of the nonzero-variance condition in Proposition 1.

B DATASETS AND EVALUATION

Table 4 summarizes the processed benchmarks used by every method. P12, MIMIC-III, and USHCN first hold out 10% of instances for testing and then 10% of the remaining instances for validation, giving approximately 81/9/10% train/validation/test splits. HumanActivity applies the same two-stage split to the 25 raw trajectories before windowing; unequal trajectory lengths yield 949/193/218 forecast windows after chunking. The counts below are loader instances; an instance with no observed future value contributes no term to the evaluation set. Given the history window, a model predicts every observed value in the forecast window at its timestamp. With $\mathcal{Q}$ the set of observed future targets,

$$\mathrm{MSE} = \frac{1}{|\mathcal{Q}|} \sum_{(n,j) \in \mathcal{Q}} \big(\hat{y}_j^{(n)} - y_j^{(n)} \big)^2, \qquad \mathrm{MAE} = \frac{1}{|\mathcal{Q}|} \sum_{(n,j) \in \mathcal{Q}} \big| \hat{y}_j^{(n)} - y_j^{(n)} \big|.$$

Table 4: Processed benchmark statistics used in our experiments. Split counts are train/validation/test loader instances.

Dataset	Variables	Split counts	History → targets	Domain
PhysioNet (Silva et al., 2012)	36	9,703/1,079/ 1,199	35.5 h → next 3	ICU records
MIMIC-III (Johnson et al., 2016)	96	17,212/1,913/ 2,125	35.75 h → next 3	ICU records
HumanActivity (Kaluža et al., 2010)	12	949/193/218	3,000 ms → 300 ms	Body-worn sensors
USHCN (Menne et al., 2009)	5	901/101/112	first 3 years → next 3	Climate stations

Within-patch asymmetry. Figure 3a reports $|\mu_p^{(n)}|/\omega_p^{(n)}$ for every patch of each test split, with weights from Eq. (5) on a uniform patch grid. A patch is included when its effective sample size $(\sum_i w_{p,i})^2/\sum_i w_{p,i}^2$ is at least three (Kish, 1965).

C Implementation Details

Dynamic boundary generation. For each variable, the five statistics of Section 3.2 are fed to a two-layer MLP with hidden width 64 that outputs, for every patch, residual adjustments to a learnable base position, log-width, and temperature. The residuals are scaled by 0.01 before they are added to the per-variable, per-patch base parameters. The base parameters are initialized so that patch p covers $[pS, (p+1)S]$ with temperature $\ln 2$; Appendix E.4 reports how far training moves each quantity.

Patch tokens. The local-linear estimate uses the closed form of Section 3.1 with the floors $S_0, D \geq \epsilon_D = 10^{-6}$ described there. The slope is scaled by $\max(\omega_p^{(n)}, 10^{-3})$ as the numerical version of Eq. (4). The pooled time encoding and the value representation are concatenated, linearly projected to width d, and refined by a residual feed-forward block. The feed-forward block has widths $d \to 2d \to d$ with GELU and LayerNorm. A sinusoidal positional encoding over the patch index and dropout are applied before aggregation.

Aggregation, mixing, and decoding. The gate MLP of Eq. (7) takes $[\mathbf{h}_p^{(n)}; \log(e_p^{(n)} + \epsilon_e)]$, maps $(d+1) \to d/2 \to 1$ with ReLU and dropout, and outputs one scalar per patch. LayerNorm follows the weighted sum. The cross-variable module applies LayerNorm, a linear map across variables, GELU, dropout, and a second, zero-initialized linear map across variables, and adds the result to its input. The decoder concatenates the mixed representation of a variable with a learnable encoding of each query time and applies an $(d + d_t) \to 2d \to 1$ MLP with ReLU and dropout, where d_t is the selected time-encoding dimension.

Plug-in variants (Table 2). Variant (1) is PatchTST applied to the irregular series, as in the baseline of Table 1. Variant (2) splits the history into the same fixed, non-overlapping windows and summarizes each window by the mean of its observed values; variant (3) fits Eq. (2) with hard 0/1 weights on the same windows. Variant (4) is APN, and variant (5) replaces the time-aware weighted average of APN by Eq. (2) under the same support parameterization. The APN boundary values are learned separately for the two variants; they are not frozen to be identical. All five variants are trained with the MSE loss and tuned with the protocol of Section 4.1. As explained in Section 4.3, the matched comparisons stop at these two geometry-weighted interiors; replacing a content-adaptive TTCN reader would also remove its learned value-dependent filter class.

Baselines. Baselines are run from their public implementations inside a common data pipeline, so every method sees the same splits, normalization, and forecasting queries. Each method–dataset pair is tuned with 100 Optuna trials over a method-appropriate search space, and its validation-selected configuration is retrained with the same five final seeds.

D Full Results with Standard Deviations

Table 5 repeats Table 1 with the standard deviation over five runs below each mean.

Table 5: Forecasting error, mean (top) and standard deviation (bottom) over five runs. **Bold** and underlined: best and second best.

Method	HumanActivity		USHCN		PhysioNet		MIMIC	
	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓
Recurrent and continuous-time								
GRU-D (2018)	0.0889 0.0201	0.2093 0.0285	0.2097 0.0493	0.3045 0.0305	0.3419 0.0029	0.3992 0.0011	0.4753 0.0119	0.4542 0.0054
NeuralFlow (2021)	0.1024 0.0040	0.2253 0.0056	0.1986 0.0065	0.3109 0.0058	0.3806 0.0024	0.4285 0.0006	0.6021 0.0080	0.5279 0.0034
CRU (2022)	0.1344 0.0078	0.2479 0.0110	0.2168 0.0162	0.3180 0.0248	0.6128 0.0048	0.5741 0.0025	0.5717 0.0054	0.5065 0.0031
Set and attention-based								
SeFT (2020)	1.3650 0.0049	0.9706 0.0026	0.3358 0.0092	0.4104 0.0228	0.7773 0.0013	0.6757 0.0007	0.9224 0.0004	0.6624 0.0004
mTAN (2021)	0.0709 0.0036	0.1839 0.0075	0.1915 0.0139	0.3257 0.0086	0.3801 0.0021	0.4281 0.0019	0.7725 0.0429	0.6124 0.0193
Warpformer (2023)	0.0416 0.0002	0.1158 0.0004	0.1888 0.0598	0.2939 0.0591	0.3069 0.0011	0.3661 0.0016	0.4081 0.0036	0.3896 0.0049
PrimeNet (2023)	4.2500 0.0018	1.7015 0.0003	0.4924 0.0007	0.4947 0.0005	0.7952 0.0000	0.6858 0.0000	0.9071 0.0001	0.6613 0.0001
Graph-based								
Raindrop (2022)	0.0814 0.0019	0.1963 0.0024	0.2406 0.0381	0.3187 0.0193	0.3515 0.0055	0.4083 0.0046	0.4939 0.0101	0.4554 0.0077
GraFITi (2024)	<u>0.0410</u> 0.0004	**0.1120** 0.0007	0.1683 0.0135	0.2668 0.0178	<u>0.3057</u> 0.0017	0.3652 0.0020	<u>0.3684</u> 0.0046	0.3695 0.0031
GNeuralFlow (2024)	0.5588 0.3238	0.5106 0.1216	0.2205 0.0421	0.3286 0.0412	0.8207 0.0310	0.6759 0.0100	0.8800 0.0297	0.6369 0.0104
HyperIMTS (2025)	0.0414 0.0002	0.1147 0.0006	0.1979 0.0358	0.2829 0.0189	**0.3005** 0.0015	<u>0.3598</u> 0.0022	**0.3567** 0.0046	<u>0.3687</u> 0.0040
Patch-based and mixing								
t-PatchGNN (2024)	0.0433 0.0011	0.1226 0.0043	0.1666 0.0103	0.2718 0.0122	0.3091 0.0014	0.3650 0.0011	0.4343 0.0051	0.4009 0.0025
APN (2026)	0.0431 0.0005	0.1203 0.0020	0.1953 0.0310	0.2934 0.0190	0.3141 0.0007	0.3674 0.0019	0.5308 0.1972	0.4550 0.1035
TFMixer (2026)	0.0429 0.0003	0.1164 0.0002	<u>0.1583</u> 0.0057	<u>0.2604</u> 0.0088	0.3203 0.0027	0.3666 0.0022	0.4094 0.0082	0.3992 0.0030
Regular-time								
PatchTST (2023)	0.0731 0.0016	0.1747 0.0016	0.1815 0.0120	0.2736 0.0112	0.3402 0.0047	0.3969 0.0046	0.5811 0.0043	0.4612 0.0069
iTransformer (2024)	0.0811 0.0020	0.1862 0.0025	0.2767 0.0551	0.3414 0.0219	0.3851 0.0019	0.4324 0.0022	0.5571 0.0145	0.4732 0.0018
TimeMixer (2024)	0.0564 0.0021	0.1497 0.0048	0.1829 0.0102	0.2823 0.0103	0.3188 0.0017	0.3711 0.0010	0.4692 0.0035	0.4317 0.0028
AKP (ours)	**0.0408** 0.0001	<u>0.1128</u> 0.0002	**0.1520** 0.0062	**0.2152** 0.0046	0.3076 0.0010	**0.3513** 0.0005	0.4033 0.0015	**0.3680** 0.0013

D.1 Uncertainty-Aware Main-Table Reading

The boldface in Table 1 describes the ordering of five-run means, not a claim that every ordering is statistically separable. Table 6 compares AKP with the strongest alternative by mean in each column. The reported difference is Δ = competitor − AKP, so positive values favor AKP; 2SE is the two-standard-error heuristic computed from the empirical standard deviations of the two independent five-run groups. This appendix keeps the main table unchanged while separating large, reproducible gaps from near-ties.

Table 6: Uncertainty-aware reading of the eight main-table columns. "Resolved" means $|\Delta| > 2\text{SE}$; the sign of Δ indicates which method has the lower mean error.

Dataset	Metric	Strongest alternative	Δ	2SE	Resolved
HumanActivity	MSE	GraFITi	$+0.0002$	0.0004	No
HumanActivity	MAE	GraFITi	-0.0008	0.0007	Yes
USHCN	MSE	TFMixer	$+0.0063$	0.0075	No
USHCN	MAE	TFMixer	$+0.0452$	0.0089	Yes
PhysioNet	MSE	HyperIMTS	-0.0071	0.0016	Yes
PhysioNet	MAE	HyperIMTS	$+0.0085$	0.0020	Yes
MIMIC-III	MSE	HyperIMTS	-0.0466	0.0043	Yes
MIMIC-III	MAE	HyperIMTS	$+0.0007$	0.0038	No

The table clarifies why the paper does not use the aggregate ranking as its primary causal argument. The strongest evidence for the proposed representation is the controlled PatchTST/APN interior replacement in Table 2: the support mechanism, decoder family, and training loss are held fixed while the geometry-weighted local constant is replaced by the local-linear estimator.

D.2 Uncertainty of Plug-In Comparisons

Table 7 reports the five-run spread for the plug-in study itself. These are independent final retrainings of separately tuned variants, so the standard deviations describe run-to-run stability rather than paired uncertainty under a frozen support. The APN MIMIC local-constant arm includes one divergent run; the local-linear arm removes that instability, but the large mean gap should not be read as a universal effect size.

Table 7: MSE uncertainty for the plug-in variants: five-run mean $\pm$ empirical standard deviation across the five runs.

Variant	HumanActivity	USHCN	PhysioNet	MIMIC-III
PatchTST (stock)	0.0731 ± 0.0016	0.1815 ± 0.0120	0.3402 ± 0.0047	0.5811 ± 0.0043
PatchTST LC	0.0460 ± 0.0004	0.1961 ± 0.0207	0.3402 ± 0.0011	0.5780 ± 0.0034
PatchTST LL	0.0429 ± 0.0002	0.1679 ± 0.0045	0.3319 ± 0.0010	0.5605 ± 0.0071
APN LC	0.0431 ± 0.0005	0.1953 ± 0.0310	0.3141 ± 0.0007	0.5308 ± 0.1972
APN LL	0.0419 ± 0.0004	0.1861 ± 0.0309	0.3095 ± 0.0012	0.4272 ± 0.0050

Table 8: MAE uncertainty for the plug-in variants: five-run mean $\pm$ empirical standard deviation across the five runs.

Variant	HumanActivity	USHCN	PhysioNet	MIMIC-III
PatchTST (stock)	0.1747 ± 0.0016	0.2736 ± 0.0112	0.3969 ± 0.0046	0.4612 ± 0.0069
PatchTST LC	0.1239 ± 0.0003	0.2775 ± 0.0130	0.3813 ± 0.0007	0.4566 ± 0.0020
PatchTST LL	0.1172 ± 0.0003	0.2586 ± 0.0049	0.3737 ± 0.0007	0.4469 ± 0.0009
APN LC	0.1203 ± 0.0020	0.2934 ± 0.0190	0.3674 ± 0.0019	0.4550 ± 0.1035
APN LL	0.1174 ± 0.0014	0.2903 ± 0.0301	0.3654 ± 0.0028	0.4013 ± 0.0032

Using the same two-standard-error heuristic as Table 6, the fixed-grid LL–LC MSE gap is resolved on all four datasets. The APN gap is resolved on HumanActivity and PhysioNet, not USHCN, while MIMIC is dominated by the unstable LC reference. We therefore present the plug-in study as consistent evidence for the representation, not as a claim that every small ranking difference is distinguishable.

D.3 Feature-Dimension-Matched Plug-In Controls

The local-linear value representation has two value channels, $[\hat{\alpha}, \omega\hat{\beta}]$, whereas the ordinary local-constant summary has one. The main plug-in table therefore establishes the practical comparison under each variant's native input projection, but it does not by itself rule out a feature-count explanation. We therefore add a matched control: the local-constant mean is retained and an MLP produces one additional value feature, so the input width and downstream token width match the

local-linear arm. All entries below are five-run means with standard deviations for MSE; the control is not a new estimator claim, only a capacity check. Within each backbone, this comparison uses the hyperparameters selected for the local-linear arm and holds them fixed across the local-constant, local-linear, and matched-control variants. These rows are therefore not additional independently tuned entries for Table 2.

Table 9: MSE for the feature-dimension-matched control. "LC" is the local-constant interior, "LL" is the local-linear interior, and "LC + feature" keeps the local-constant mean while adding one learned value feature. Each cell shows the five-run mean above the empirical standard deviation.

Dataset	APN LC	APN LL	APN LC + feature	PatchTST LC	PatchTST LL	PatchTST LC + feature
HumanActivity	0.0422 0.0001	0.0419 0.0004	0.0423 0.0002	0.0458 0.0002	0.0429 0.0002	0.0459 0.0002
USHCN	0.1928 0.0254	0.1861 0.0309	0.1790 0.0184	0.1929 0.0103	0.1650 0.0042	0.1889 0.0156
PhysioNet	0.3137 0.0006	0.3095 0.0012	0.3141 0.0010	0.3410 0.0020	0.3317 0.0008	0.3438 0.0022
MIMIC-III	0.4309 0.0033	0.4272 0.0050	0.4294 0.0037	0.5675 0.0054	0.5605 0.0071	0.5638 0.0026

Table 10: MAE for the same feature-dimension-matched control. Each cell shows the five-run mean above the empirical standard deviation.

Dataset	APN LC	APN LL	APN LC + feature	PatchTST LC	PatchTST LL	PatchTST LC + feature
HumanActivity	0.1164 0.0008	0.1174 0.0014	0.1166 0.0010	0.1218 0.0003	0.1172 0.0003	0.1220 0.0002
USHCN	0.2990 0.0259	0.2903 0.0301	0.2759 0.0275	0.2732 0.0114	0.2559 0.0054	0.2712 0.0091
PhysioNet	0.3672 0.0011	0.3654 0.0028	0.3685 0.0033	0.3804 0.0016	0.3736 0.0008	0.3825 0.0021
MIMIC-III	0.4023 0.0015	0.4013 0.0032	0.4032 0.0030	0.4516 0.0006	0.4469 0.0009	0.4510 0.0009

The extra-feature control does not reproduce the local-linear improvement. In MSE, the local-linear arm is lower in seven of eight backbone–dataset settings; the exception is APN on USHCN, where the result is also variable across runs. We therefore read the comparison as evidence against a simple dimensionality explanation rather than as a universal significance claim.

D.4 Frozen-Support Counterfactual

To test the support-geometry objection directly, we start from each seed of the APN local-constant reference and retain only the learned parameters defining the support locations, widths, and temperatures. These support parameters are then frozen while the local-linear projection, decoder, and all remaining parameters are initialized and trained from scratch with the same hyperparameters and five seeds. The maximum absolute change in every frozen support parameter is 0 for every dataset and seed. Thus this is a counterfactual with identical learned supports, rather than two separately optimized support geometries. We restrict this additional experiment to HumanActivity, P12, and USHCN; MIMIC-III remains untested in this analysis. The shared hyperparameters are those selected for the local-linear APN arm; therefore the local-constant reference below is not the independently tuned APN row in Table 2.

Table 11: Frozen-support counterfactual using the learned supports from the APN local-constant reference. Entries are five-run mean $\pm$ empirical standard deviation. "Resolved" means the local-constant versus frozen local-linear mean gap exceeds 2SE.

Dataset	LC MSE	Frozen LL MSE	LC MAE	Frozen LL MAE
HumanActivity	0.0422 ± 0.0001	0.0418 ± 0.0006	0.1164 ± 0.0008	0.1153 ± 0.0008
P12	0.3137 ± 0.0006	0.3139 ± 0.0006	0.3672 ± 0.0011	0.3690 ± 0.0021
USHCN	0.1928 ± 0.0254	0.1632 ± 0.0117	0.2990 ± 0.0259	0.2664 ± 0.0126

Resolved MSE/MAE gaps: HumanActivity no/yes; P12 no/no; USHCN yes/yes.

The frozen result supports the mechanism most clearly on USHCN, where the local-linear interior lowers MSE by 15.4% under exactly the same learned supports. The HumanActivity reduction is small and not MSE-resolved, while P12 is essentially unchanged. This is the appropriate interpretation of the counterfactual: it supports a dataset-dependent interior correction rather than a claim of a uniform gain after support geometry is held fixed.

E Additional Analyses

E.1 Entanglement Measurements

Figure 4 repeats panels (a) and (b) of Figure 3 and adds the sampling-skew study (panel c). At fixed patch width, the local-constant bias grows with the skew, whereas the local-linear and local-quadratic biases stay flat: the skew changes μ, which Eq. (1) multiplies by the slope, and the local-linear fit estimates the slope instead of folding it into the level.

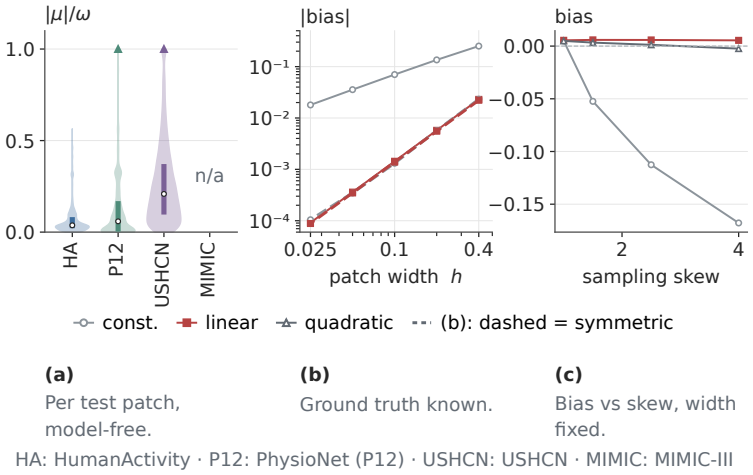


Figure 4: **Entanglement measurements.** (a) Within-patch asymmetry $|\mu|/\omega$ on each test split. (b) Level bias versus patch width on synthetic signals; dashed: symmetric sampling. (c) Level bias versus sampling skew at fixed width.

E.2 Estimator Validity on Learned Supports

The exactness proposition applies when a patch has positive mass and nonzero weighted timestamp variance. To quantify how often the trained numerical operator leaves that regime, we replay the selected checkpoints on the test data and recompute the raw moments before clamping. HumanActivity and USHCN use their complete test splits; P12 and MIMIC-III use 300 test samples because their test sets are larger. Table 12 reports the 5th, 50th, and 95th percentiles for support mass, normalized timestamp variance, and effective RMS radius. Clamp rates include all patch–variable pairs, including pairs with no observed history.

Table 12: Checkpoint-derived validity statistics for the learned supports. Mass is S_0; normalized variance is $(S_0S_2-S_1^2)/(S_0^2\omega^2)$; effective radius is $\sqrt{S_2/S_0}/\omega$. Entries are 5th/50th/95th percentiles.

Dataset	Samples	Mass	Var./ω^2	Radius/ω	S_0 clamp	D clamp
HumanActivity	218	0.295/5.455/6.741	9.98/59.74/87.61	7.13/9.90/14.13	0.0%	0.2%
P12	300	0.016/0.487/7.091	0.00/2.07/15.38	1.03/3.78/10.57	30.6%	45.9%
USHCN	112	8.817/12.662/15.867	71.55/97.86/125.64	9.37/14.57/26.97	0.0%	0.0%
MIMIC-III	300	0.003/0.255/1.184	0.00/26.55/547.70	3.49/19.01/74.21	62.2%	77.0%

The table makes the scope of the theory explicit. P12 and especially MIMIC-III contain many nearly empty or rank-deficient patch–variable pairs, so the implemented token is a stabilized extension

of the ideal weighted least-squares operator there; the exactness and bias-order propositions are not invoked for those clamped pairs. Conversely, the large effective radii on all four datasets show that the trained model often uses a broad linear summary rather than a strictly local smoother. The synthetic local-rate result should therefore be read as a mechanism result, while the forecasting gains are empirical evidence for using the same estimator beyond the strict local regime.

E.3 Noise and Sample-Count Sensitivity

To separate design-bias reduction from variance amplification, we extend the operator-level synthetic study with Gaussian observation noise. We use the asymmetric design ($h = 0.2$, skew $p = 3$), curvature 2, and 50,000 Monte Carlo replicates per cell. Table 13 reports RMSE as local-constant/local-linear, thereby combining bias and variance in the reported quantity. Increasing noise or reducing the observation count can make the local-linear estimate variance-dominated, especially at $n = 4$, even though its design bias remains smaller.

Table 13: Noise-aware operator RMSE under asymmetric sampling. Each entry is local-constant/local-linear RMSE; $h = 0.2$, skew $p = 3$, curvature 2, and 50,000 replicates per cell.

Noise σ	$n = 4$	$n = 16$	$n = 64$
0.00	0.1713/0.0120	0.1441/0.0065	0.1365/0.0063
0.02	0.1717/0.0932	0.1438/0.0101	0.1364/0.0072
0.05	0.1736/0.2723	0.1445/0.0203	0.1365/0.0108
0.10	0.1794/0.5049	0.1463/0.0389	0.1372/0.0184

Curvature changes the deterministic bias gap without changing the conclusion about variance. At $n = 16$ and $\sigma = 0.05$, the local-constant/local-linear RMSE pairs are $0.0668/0.0192$, $0.1445/0.0203$, and $0.2223/0.0232$ for curvature 0, 2, and 4, respectively. Thus the proposed representation is most useful when the support contains enough timestamp variation; in sparse, high-noise patches the paper relies on the numerical floor and the downstream decoder rather than claiming a uniformly lower estimator variance.

E.4 What Dynamic Boundary Generation Learns

Figure 5 measures the trained boundary generator against its initialization, which places patch p on $[pS, (p+1)S]$ with width S and temperature $\ln 2$. Of the three quantities it controls, only the patch position moves far from its initialization (panel a): the median shift is 1.8–12.2 spacings, while the width stays within 0.02–$0.18\,S$ of S and the temperature within 0.02–$0.51 \ln 2$ of $\ln 2$. With the temperature near $\ln 2$, the ratio τ/S lies between 5.5 and 52.5, so the soft supports are nearly flat over the history (panel b), and one patch summary averages 86–100% of its variable's observations (panel c). Recomputing the same boundaries with $\tau = S/2$ would average 8–26% on HumanActivity, PhysioNet, and USHCN, and almost nothing on MIMIC, where a variable holds fewer than two observations per sample.

E.5 Aggregation Diagnostics

We correlate the gate weight $\gamma_p^{(n)}$ with $\log e_p^{(n)}$ within each (sample, variable) group, the axis over which the softmax of Eq. (7) is taken; pooling across groups would mostly reflect that different variables have different support. The gate upweights better-supported patches on USHCN ($\rho = +0.75$) and downweights them on HumanActivity ($\rho = -0.58$). Retraining without $\log e_p^{(n)}$ in the gate input leaves both correlations essentially unchanged ($\Delta = 0.00$ and 0.01), because the patch token is built from the same weights and already carries how much support it had; where the full gate's correlation is near zero, removing the statistic shifts it by about 0.25. The statistic is therefore partly redundant with $\mathbf{h}_p^{(n)}$, rather than an independently validated confidence estimator. Table 14 reports the two gate-operation variants of Table 3 with standard deviations over three runs.

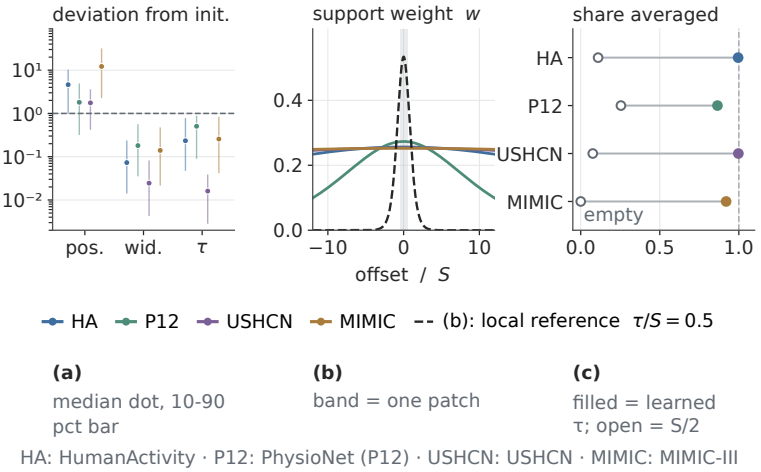


Figure 5: **What the boundary generator moves.** (a) Deviation of position, width, and temperature from initialization. (b) Learned support weights around one patch; dashed: a local window with $\tau/S = 0.5$. (c) Share of a variable's observations averaged by one patch, for the learned τ (filled) and $\tau = S/2$ (open).

Table 14: Gate ablations, mean (top) and standard deviation (bottom) over three runs. **Bold**: best in each column.

Variant	HumanActivity		USHCN		PhysioNet		MIMIC	
	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓	MSE↓	MAE↓
Full AKP	**0.0408** 0.0004	**0.1125** 0.0002	**0.1504** 0.0085	**0.2156** 0.0066	0.3077 0.0012	**0.3513** 0.0005	**0.4030** 0.0009	**0.3677** 0.0006
gate without support mass	0.0413 0.0002	**0.1125** 0.0006	0.1712 0.0130	0.2223 0.0078	0.3073 0.0009	0.3519 0.0006	0.4133 0.0089	0.3693 0.0004
uniform patch mean	0.0410 0.0001	0.1130 0.0006	0.1584 0.0115	0.2231 0.0046	**0.3070** 0.0003	0.3526 0.0005	0.4033 0.0013	0.3699 0.0003

E.6 Cross-Study Effect Sizes

Figure 6 puts operation replacements and one-at-a-time hyperparameter interventions on the same test-MSE scale. This is the most direct summary of which empirical effects support the model design: replacing the local-linear interior raises mean MSE on all four datasets and clears the two-standard-error threshold on three, while cross-variable mixing clears it on all four and sample-conditioned boundaries on three. By contrast, replacing the gate with a uniform mean does not clear the threshold on any dataset, so we do not elevate that auxiliary choice to a separate contribution.

E.7 Controlled Irregularity

We corrupt the sampling of the train, validation, and test histories with five modes and, at every setting, refit the APN plug-in backbone from scratch with a local-constant and with a local-linear patch interior, three fits per arm. Irregularity is measured by gap_{cv}, the coefficient of variation of inter-observation gaps. On PhysioNet, whose native series is close to regular ($\mathrm{gap}_{cv} = 0.14$), the local-linear interior is better by more than two standard errors at 13 of 19 corrupted settings; no uncorrupted P12 control was run. On USHCN, which is already irregular at baseline ($\mathrm{gap}_{cv} = 0.94$), the differences are mostly within two standard errors (1 of 20 conditions, including the uncorrupted control, resolved). The advantage does not grow monotonically with gap_{cv} on either dataset (Spearman $\rho = -0.22$ on PhysioNet and $+0.20$ on USHCN). This null monotonic result is a boundary on the claim, not evidence against the derived mechanism: gap_{cv} measures global gap variability, whereas Eq. (1) depends on signed within-patch asymmetry and local slope.

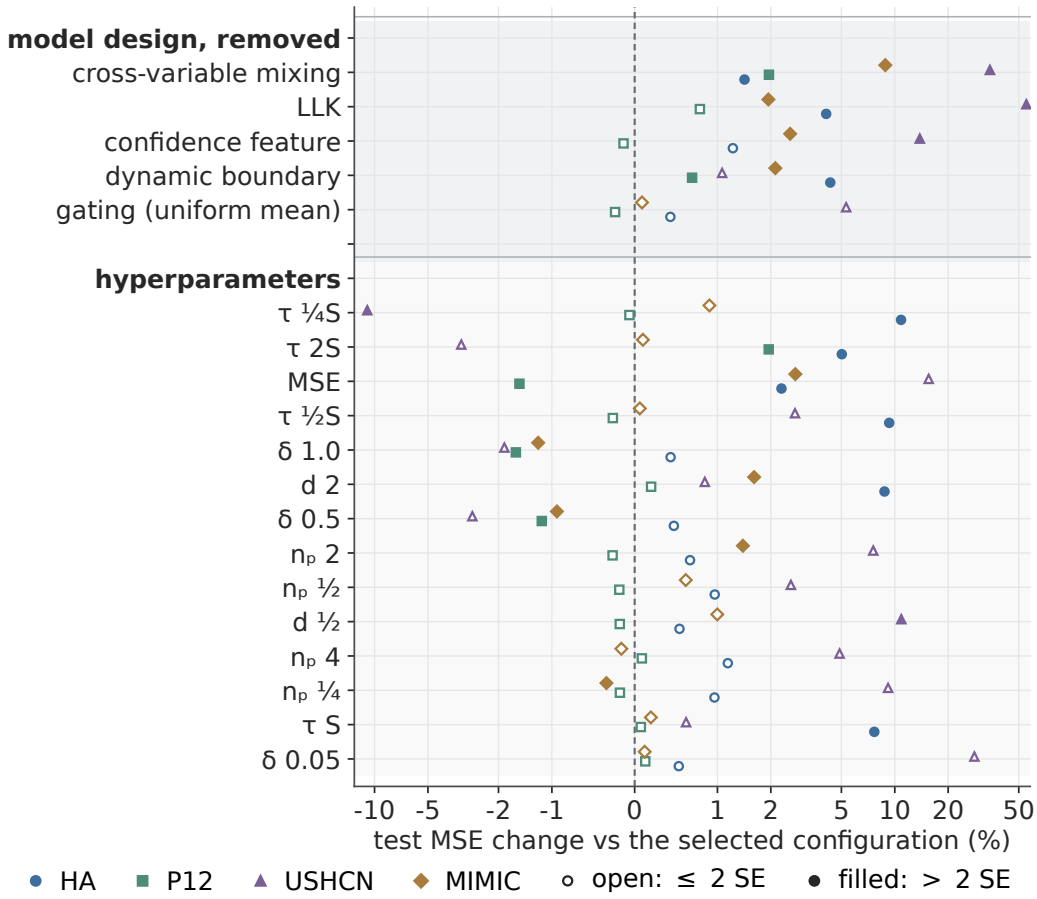


Figure 6: **Architectural operation replacements and hyperparameter moves on one scale.** Each marker is the percentage change in test MSE relative to the same dataset's three-seed full-model reference; positive values are worse. The upper block replaces one operation and the lower block changes one selected hyperparameter at a time. Filled markers exceed two standard errors of the difference between independent three-run means; open markers do not. The symmetric-log axis is linear within $\pm 1\%$. HA, P12, and MIMIC denote HumanActivity, PhysioNet 2012, and MIMIC-III.

E.8 Training Cost

Figure 7 places AKP and four baselines by training time per sample and test MSE on USHCN. AKP trains faster per sample than TFMixer and HyperIMTS while reaching lower MSE than all four.

F Hyperparameters and Sensitivity

F.1 Search Space and Selected Configuration

Table 15 lists the search space used for AKP and the configuration selected on each dataset. Every baseline receives the same 100-trial budget over its own architecture-appropriate search space, as specified in Section 4.1.

The seven searched hyperparameters of Table 15 are selected per dataset by a Tree-structured Parzen Estimator search (Optuna's default sampler) of 100 trials over the ranges given in the table. The architecture definition is the same on all four datasets; only the listed hyperparameter values differ. The search space was fixed before any trial was run. Each trial is trained once under a short protocol (150 epochs with early-stopping patience 5; 300/10 on USHCN) with the Huber objective at $\delta = 0.1$, and is scored by the masked MSE of that run. The selected configuration is then retrained from

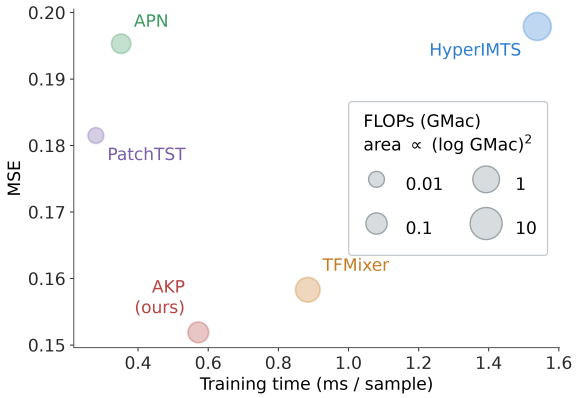


Figure 7: **Accuracy versus training cost on USHCN.** Bubble area grows with FLOPs.

scratch at the protocol every result in this paper uses (500 epochs, patience 10, validation-loss early stopping), with independent seeds. The last row of the table gives the parameter count of the selected model: it is small in absolute terms, and it varies by dataset chiefly through the model width d and the number of patches n_p, which is why a single figure would not describe the four rows.

Table 15: Search space and selected hyperparameters of AKP.

Hyperparameter	Search range	Human Activity	USHCN	PhysioNet	MIMIC
Learning rate	10^{-3}–10^{-2} (log)	5.67×10^{-3}	1.02×10^{-3}	2.29×10^{-3}	2.25×10^{-3}
Batch size	{16, 32, 64}	16	16	16	16
Dropout	0.0–0.3, step 0.05	0.05	0.15	0	0
Model width d	{128, 256, 512}	256	256	128	128
Number of patches n_p	{16, 20, 24, 32, 48, 64, 78, 100}	32	48	16	100
Time-encoding dimension	{8, 12, 16, 32}	12	32	32	32
Gradient clipping	{0.0, 0.5, 1.0}	1	0.5	0	0
Huber δ	fixed	0.1	0.1	0.1	0.1
Regression floor ϵ_D	fixed	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Gate-log floor ϵ_e	fixed	10^{-9}	10^{-9}	10^{-9}	10^{-9}
Max. epochs / patience	fixed	500 / 10	500 / 10	500 / 10	500 / 10
Parameters	–	447 794	465 630	128 986	188 398

F.2 Sensitivity Analysis

Each experiment below varies one hyperparameter of AKP and keeps all others at the selected configuration of Table 15; each setting is trained with seeds 2024–2026 and evaluated with test MSE and MAE on all four datasets. The sweep is a post-selection robustness analysis: test-set improvements are reported but are not used to replace the validation-selected configuration.

The dominant effects are dataset-specific. Pinning τ to any tested fixed value raises HumanActivity MSE by 5.0–10.8%, whereas $\tau = S/4$ lowers USHCN MSE by 11.0%; all four HumanActivity cells and the USHCN $S/4$ cell clear two standard errors. Model width also acts in opposite directions: doubling d raises HumanActivity MSE by 8.8%, while halving it raises USHCN MSE by 10.9%. Varying the number of patches moves HumanActivity, PhysioNet, and MIMIC-III by at most 1.4%; the 2.6–9.2% changes on USHCN do not clear two standard errors because its reference runs are variable. On PhysioNet, Huber $\delta \in \{0.5, 1.0\}$ and MSE lower MSE by 1.1–1.6% but worsen MAE relative to $\delta = 0.1$. Huber $\delta = 0.05$ on USHCN hides one failed seed inside a +28% mean (0.28 test MSE versus about 0.14 for the other two) and does not clear the two-standard-error test. Several

MIMIC-III cells are statistically resolved because its reference variance is small, but all MSE shifts remain within 2.8%. No setting improves on the selected configuration by more than USHCN's $\tau = S/4$ does. Because these rows were specified after the main search and scored on the test split, they remain a post-selection sensitivity analysis. The separate validation-selected baseline for HumanActivity and USHCN is reported in Appendix F.3; MIMIC-III is not included in that selection experiment.

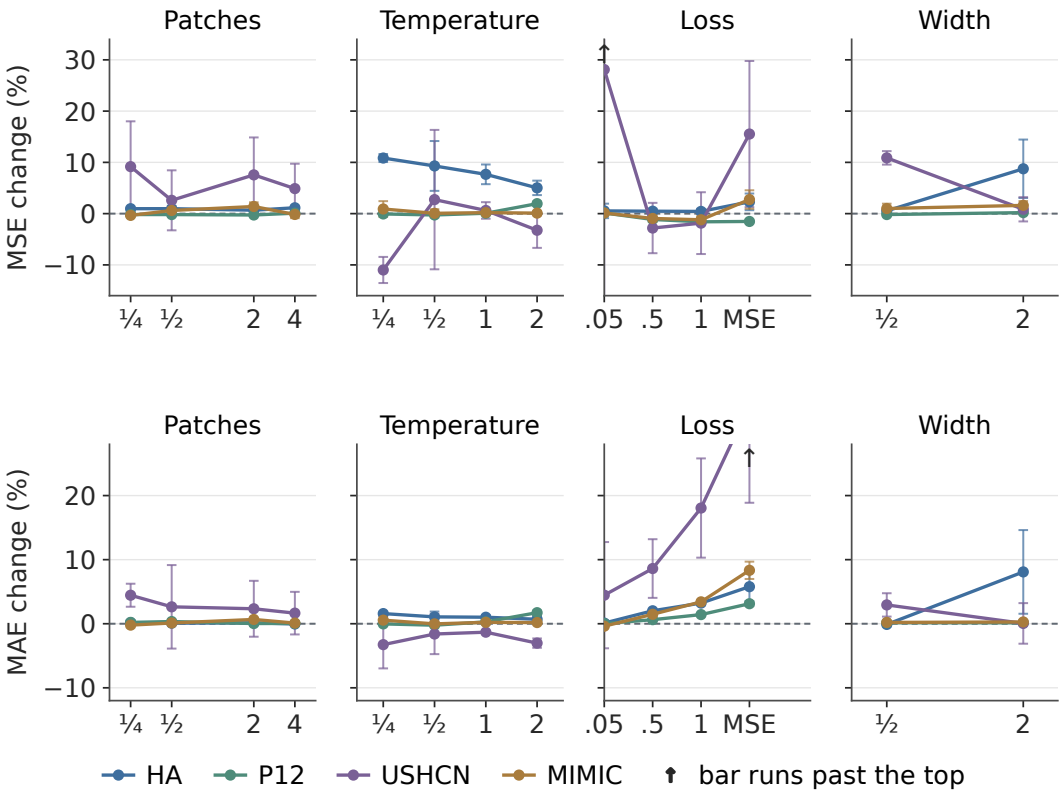


Figure 8: **Sensitivity of AKP to one hyperparameter at a time.** Each point is the mean over three runs, and error bars are the empirical standard deviation across those runs. Values are percentage changes relative to the selected configuration for the same dataset; the dashed horizontal line is therefore 0%. The four panels vary the number of patches, support temperature, training loss, and model width. The selected setting itself is represented by the zero line rather than repeated as a marker. Up arrows mark error bars clipped by the panel limit. HA, P12, and MIMIC denote HumanActivity, PhysioNet 2012, and MIMIC-III.

1. **Number of patches** n_p: $\{n_p^\star/4,\ n_p^\star/2,\ n_p^\star,\ 2n_p^\star,\ 4n_p^\star\}$, where $n_p^\star$ is the selected value.
2. **Support temperature**: the learned τ (default) against fixed $\tau \in \{S/4,\ S/2,\ S,\ 2S\}$ with the generator still adapting positions and widths.
3. **Loss**: Huber $\delta \in \{0.05,\ 0.1,\ 0.5,\ 1.0\}$ and the MSE loss.
4. **Model width** d: $\{d^\star/2,\ d^\star,\ 2d^\star\}$.

All row groups use the same three-seed, full-model reference for a given dataset. The selected configuration of Table 15 attains a test MSE of 0.0408 on HumanActivity, 0.1504 on USHCN, 0.3077 on PhysioNet and 0.4030 on MIMIC-III (Table 3). These are three-seed means, not the five-seed means of Table 1; a sensitivity row states how the corresponding three-seed reference changes when one knob moves.

F.3 Validation-Selected Fixed Temperature

The post-selection sweep above is not a model-selection baseline. We therefore ran a separate validation-based selection on HumanActivity and USHCN. The candidate grid was fixed in advance at $\tau/S \in \{1/4, 1/2, 1, 2\}$; each candidate used the selected AKP hyperparameters and three independent runs, and selection used only the mean best-validation Huber loss. Test results from nonselected candidates were not used in the selection. The selected candidate was then retrained in five independent final runs for the reported test result.

Table 16: Validation-selected fixed-temperature baseline. Selection uses the mean validation loss from three independent runs; test entries use five independent final runs. The corresponding adaptive AKP means are shown below the table.

Dataset	Selected τ/S	Val. loss	Fixed τ MSE	Fixed τ MAE
HumanActivity	2	0.00809	0.0428 ± 0.0005	0.1131 ± 0.0006
USHCN	$1/2$	0.02150	0.1494 ± 0.0147	0.2102 ± 0.0061

Adaptive AKP main-table MSE/MAE: HumanActivity $0.0408/0.1128$; USHCN $0.1520/0.2152$.

The validation-selected fixed temperature is competitive on USHCN and does not replace the main result: its MSE difference from adaptive AKP is within the run-to-run uncertainty, while adaptive AKP has lower mean error on HumanActivity. This result rules out a post-selection explanation based solely on choosing τ after seeing the test set, but it does not establish a universal benefit for adaptive temperatures; MIMIC-III remains untested under this protocol.

F.4 Settings of the Synthetic and Corruption Studies

Synthetic study (Figure 4b,c). The study evaluates the same closed-form order-0 and order-1 patch-interior operators defined in Section 3.1, while fixing the patch geometry by construction. It is therefore an operator-level validation that isolates the estimator from learned boundaries, patch aggregation, and forecasting layers. The ground truth is $f(t) = 1.5t + 2t^2$ on $t \in [0, 1]$ – a linear term plus curvature, so that the linear part is exactly what a local-constant fit cannot represent when its support is asymmetric. No observation noise is injected; the only stochasticity is the sampled design, and the reported quantity is the Monte Carlo estimate of signed design bias. The patch operator is evaluated in double precision. The estimator sees a *single* patch ($N = P = 1$ variable-patch pair) of width h centred at $c = 0.5$, holding n observations at times $t = c - h/2 + h\,u^p$ with $u \sim \mathcal{U}(0, 1)$, weighted by the same logistic kernel the model uses, at $\tau = h/10$. The exponent p is the sampling skew: $p = 1$ places the observations uniformly across the window (symmetric sampling), and $p > 1$ concentrates them toward the left edge, producing the within-patch asymmetry $\mu_p^{(n)}$ of Eq. (1) at a fixed window and a fixed observation count. Each sweep evaluates the same set of arms: the local-constant and local-linear interiors (orders 0 and 1), a quadratic (order 2) arm that brackets the reading from above, and a feature-dimension-matched control that keeps the local-constant mean and lets an MLP emit one extra value feature. The control tests whether an additional learned feature alone explains the separation. Only a subset is drawn: Panel (b) sweeps the patch width over $h \in \{0.4,\ 0.2,\ 0.1,\ 0.05,\ 0.025\}$ at $n = 20$ observations per patch, under symmetric ($p = 1$) and asymmetric ($p = 3$) sampling, and plots the local constant and the local linear for both, four curves in all; each point is the magnitude of the mean *signed* bias over 10^5 replicates, and the reported rates are least-squares slopes of $\log|\text{bias}|$ against $\log h$. Panel (c) fixes $h = 0.2$ and $n = 20$ and sweeps the skew over $p \in \{1.0,\ 1.5,\ 2.5,\ 4.0\}$ at 300 replicates, plotting the *signed* bias of orders 0, 1 and 2 – the range over which the interior orders separate. The order-2 arm is ridge-regularised, so it is read as a bound on the order-1 result rather than as a rate claim of its own.

Controlled-irregularity study. This study takes two real benchmarks with different native irregularity levels – PhysioNet (P12) and USHCN – and corrupts the sampling of the history in a controlled way, sweeping one severity knob at a time. Five corruption modes are used, each with its own severity grid. USHCN additionally includes an uncorrupted 0.0 control:

- **random**: Bernoulli thinning of observations, severity $s \in \{0.2, 0.4, 0.6, 0.8\}$;

- **non-uniform**: the thinning probability is modulated along the series by a sinusoid of frequency 3, giving a nonstationary sampling density, $s \in \{0.2, 0.5, 0.8\}$;
- **clustered**: block-level Bernoulli thinning over blocks of 5% of the history, so surviving observations arrive in runs at the same expected density as *random*, $s \in \{0.3, 0.5, 0.7\}$;
- **long gap**: one contiguous run of length $s\,n$ is removed from a uniformly random position, $s \in \{0.1, 0.25, 0.5\}$;
- **one-sided**: the thinning probability ramps linearly along the series, making the sampling asymmetric rather than merely irregular; $s > 0$ thins the recent end and $s < 0$ the old end, so the severity carries a sign, $s \in \{\pm 0.2, \pm 0.5, \pm 0.8\}$.

The completed campaign contains 19 corrupted conditions on P12 $(4 + 3 + 3 + 3 + 6)$ and those same conditions plus one uncorrupted control on USHCN. All corruptions leave at least 10% (and never fewer than four) of the observations in place, and only the *history* is corrupted: target steps, timestamps, and values are untouched. Each corruption is a deterministic function of the sample index and a fixed seed, so the train, validation, and test splits each receive a fixed corruption and the runs are reproducible. Both arms use the same APN plug-in backbone and differ only in whether the patch interior is local constant or local linear. They are trained under the *same* hyperparameters, epochs and seeds, so that the interior is the only difference between them; those hyperparameters are in turn frozen across all 19 corrupted conditions (and the USHCN control), so the corruption is the only difference between any two runs of the same arm. They are not taken from Table 15 but from a separate, method-appropriate search run once on the *uncorrupted* data with the local-linear interior in place. Both arms then reuse that one setting: re-tuning per condition, or tuning the two interiors separately, would let the hyperparameters absorb part of the effect being measured. The budget is 300 epochs, patience 10 and three seeds per arm, under the MSE objective rather than the Huber objective used elsewhere in this paper, so the comparison directly measures squared-error accuracy. We summarize each corrupted history by gap_{cv}, the coefficient of variation of its inter-arrival gaps, and define the local-linear advantage as $\mathrm{MSE}(\text{local constant}) - \mathrm{MSE}(\text{local linear})$, positive when the local-linear interior wins. Resolution is assessed against twice the standard error of the two independent three-run means.

G ANNOTATED CORE IMPLEMENTATION

This appendix gives a faithful, lightly abridged PyTorch implementation of the operations that distinguish AKP from a conventional forecasting MLP. Tensor reshaping and training-loop boilerplate are omitted, but the learned support parameterization, numerical floors, local-linear moments, patch gate, and zero-initialized variable mixer are retained. The names below follow the mathematical notation of Section 3 and do not depend on an implementation-specific module layout.

G.1 SAMPLE-CONDITIONED SOFT SUPPORTS

Listing 1 implements the support construction of Eq. (5). The five sampling statistics condition residual changes to per-variable base parameters. The factor 0.01 keeps the generator near its uniform initialization at the start of training, while softplus keeps the temperature positive. Base tensors are broadcast over the batch before the function is called.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

EPS_D = 1e-6  # floor for geometric and regression quantities
EPS_E = 1e-9  # floor used inside normalization and logarithms

def build_soft_supports(t, mask, stats, base_shift,
                        base_log_width, base_temperature,
                        boundary_mlp, spacing):
    # t, mask: [B*N, L]; base_*: [B*N, P].
    # stats contains observation ratio, mean gap, span,
    # last time, and first time for each sample-variable pair.
    P = base_shift.size(-1)
    residual = boundary_mlp(stats)              # [B*N, 3P]
    d_shift, d_log_width, d_tau = residual.chunk(3, dim=-1)

    # Uniform-grid initialization plus sample-conditioned residuals.
    grid_left = torch.arange(P, device=t.device) * spacing
    left = grid_left + base_shift + 0.01 * d_shift
    log_width = base_log_width + 0.01 * d_log_width
    tau = F.softplus(base_temperature + 0.01 * d_tau) + EPS_D
    right = left + torch.exp(log_width)

    # Product of two sigmoids gives differentiable left/right edges.
    t_obs = t[:, None, :]                       # [B*N, 1, L]
    weight = torch.sigmoid((right[..., None] - t_obs)
                           / tau[..., None])
    weight = weight * torch.sigmoid((t_obs - left[..., None])
                                    / tau[..., None])
    weight = weight * mask[:, None, :]          # remove missing values
    return left, right, weight
```

Listing 1: Sample-conditioned logistic supports.

G.2 LOCAL-LINEAR PATCH TOKENS

Listing 2 is the implementation counterpart of Eqs. (2)–(4). Using the weights returned by Listing 1, it forms the five sufficient statistics in one pass, solves the 2×2 weighted normal equations, and returns both the patch token and its support mass. The numerical clamp is an explicit fallback for nearly empty or rank-deficient patches; on non-degenerate pairs the computation is the exact closed-form estimator analyzed in Proposition 1.

```python
def local_linear_tokens(t, x, time_code, left, right, weight,
                        project, feed_forward, norm):
    # t, x: [B*N, L]; weight: [B*N, P, L].
    # time_code: [B*N, L, d_t]; left/right: [B*N, P].
    t_obs = t[:, None, :]                       # [B*N, 1, L]
    center = 0.5 * (left + right)
    offset = t_obs - center[..., None]
    mass = weight.sum(dim=-1)                   # support mass used by gate

    # Weighted moments of [1, offset] and the observed values.
    S0 = mass.clamp_min(EPS_D)
    S1 = (weight * offset).sum(dim=-1)
```

```
    S2 = (weight * offset.square()).sum(dim=-1)
    T0 = (weight * x[:, None, :]).sum(dim=-1)
    T1 = (weight * x[:, None, :] * offset).sum(dim=-1)

    # This clamp is the documented numerical fallback, not a ridge fit.
    determinant = (S0 * S2 - S1.square()).clamp_min(EPS_D)
    slope = (S0 * T1 - S1 * T0) / determinant
    level = (T0 - slope * S1) / S0

    # Width scaling makes the second channel a change, not a rate.
    width = (right - left).clamp_min(1e-3)
    value_features = torch.stack([level, width * slope], dim=-1)

    # Timing remains available in a separate channel.
    pooled_time = torch.einsum("bpl,bld->bpd", weight, time_code)
    pooled_time = pooled_time / (mass[..., None] + EPS_E)
    raw_token = torch.cat([value_features, pooled_time], dim=-1)

    token = project(raw_token)
    token = norm(token + feed_forward(token))
    return token, mass
```

Listing 2: Weighted local-linear patch representation.

G.3 Patch Aggregation and Variable Mixing

Listing 3 completes the forward path. The softmax is taken only over patches of the same sample and variable, matching Eq. (7). The variable mixer acts on the pooled $[B, N, d]$ representation; zero-initializing its last layer makes it an exact identity map before learning cross-variable corrections.

```
class CrossVariableMixer(nn.Module):
    def __init__(self, n_variables, hidden_dim, dropout):
        super().__init__()
        self.norm = nn.LayerNorm(hidden_dim)
        self.mlp = nn.Sequential(
            nn.Linear(n_variables, n_variables),
            nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(n_variables, n_variables),
        )
        # Start from independent variables; learn only residual mixing.
        nn.init.zeros_(self.mlp[-1].weight)
        nn.init.zeros_(self.mlp[-1].bias)

    def forward(self, h):                          # h: [B, N, d]
        z = self.norm(h).transpose(1, 2)           # [B, d, N]
        return h + self.mlp(z).transpose(1, 2)


def aggregate_and_decode(tokens, mass, positional_encoding,
                         dropout, gate_mlp, aggregate_norm,
                         variable_mixer, query_code, decoder):
    # tokens: [B, N, P, d]; mass: [B, N, P].
    # query_code: [B, L_f, d_t].
    B, N, P, d = tokens.shape

    # Add patch order before computing content-and-support gate scores.
    h = tokens.reshape(B * N, P, d)
    h = dropout(positional_encoding(h)).reshape(B, N, P, d)
    log_mass = torch.log(mass + EPS_E)[..., None]
    score = gate_mlp(torch.cat([h, log_mass], dim=-1)).squeeze(-1)
    gate = torch.softmax(score, dim=2)             # normalize over P

    pooled = (h * gate[..., None]).sum(dim=2)      # [B, N, d]
    context = variable_mixer(aggregate_norm(pooled))

    # Every variable is decoded at every requested future timestamp.
    L_f = query_code.size(1)
    context = context[:, :, None, :].expand(B, N, L_f, d)
    query = query_code[:, None, :, :].expand(B, N, L_f, -1)
    prediction = decoder(torch.cat([context, query], dim=-1)).squeeze(-1)
    return prediction.transpose(1, 2), gate        # [B, L_f, N]
```

Listing 3: Support-mass-gated aggregation, variable mixing, and decoding.