S^{3}M: SPECTRAL STATE SPACE MODEL FOR STRUCTURED MULTIVARIATE TIME SERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Multivariate streams exhibit temporal memory and structured interactions among variables. Standard selective state space models capture the former but rely on channel-separable transitions, whereas graph neural networks capture the latter without a recurrent state-space interpretation. We propose the spectral state space model (S^{3}M) for structured multivariate time series, a graph-structured state space model in which the latent state is a graph signal and the linear state transport is a learned mixture of graph-spectral filters. A dynamic observation graph captures context-dependent relations, a compact B-spline filter bank provides frequency-selective transport, and a top-s router selects a sparse spectral subspace at every time step. Availability-aware pooling prevents unobserved entries from receiving readout weight. We establish exact graph-Fourier diagonalization of the linear transport, characterize when it produces cross-variable mixing in measurement coordinates, prove optimality of the implemented top-s simplex rule, and bound the state of the gated, layer-normalized recurrence. Evaluated on the large-scale benchmark PhysioNet 2019, our method outperforms the state-of-the-art (SOTA) baselines on the standard primary metric and shows effectiveness. Controlled interventions quantify the contributions of spectral filtering, recurrent memory, graph coupling, and availability-aware readout.

1 INTRODUCTION

Multivariate time series are governed by two coupled structures: persistence along time and interactions among variables. In intensive care, for example, a change in one laboratory value changes the meaning of another measurement, while the reliability of that relation depends on observation density and clinical regime. A model that treats variables independently can recover such interactions only after the recurrent computation, so cross-variable evidence cannot be propagated within the state transition itself. This motivates a recurrent transition that can mix variables within the state update rather than only after it.

Selective state space models (SSMs) offer efficient long-range memory through input-dependent decay (Gu et al., 2022; Gu and Dao, 2023). Their usual diagonal state transition is fast and stable, but it is channel-separable, so cross-variable evidence cannot be propagated during recurrence. Spectral graph models instead represent cross-variable structure through the Laplacian eigenbasis (Defferrard et al., 2016; Kipf and Welling, 2017). Their filters have a direct frequency interpretation, but they are typically used as external encoders or predictors rather than as part of the recurrent state transition. Existing graph–SSM combinations generally append a graph block or a temporal transform to an SSM (Zhang et al., 2022). As a result, spectral propagation and state evolution remain only loosely coupled, and the transition itself is not graph-spectral. Together, these limitations expose a research gap: temporal memory, graph-based cross-variable mixing, and irregular observation handling are usually realized by separate mechanisms, and no recurrent state transition is itself graph-spectral, input-selectively sparse, and availability-aware.

To address this research gap, we propose spectral state space model (S^{3}M) for structured multivariate time series, a graph-structured state space model whose linear transport is a learned mixture of graph-spectral filters. A dynamic observation graph captures context-dependent relations, a compact B-spline filter bank provides frequency-selective transport, and a top-s router selects a sparse spectral subspace at every time step. Availability-aware pooling prevents unobserved entries from receiving

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.

readout weight. We establish graph-Fourier equivalence for the linear transport, characterize when it induces cross-variable mixing, prove top-s support optimality, and bound the implemented gated, layer-normalized state. S^{3}M makes the relationship between spectral propagation and state evolution the design primitive. We construct an observation-conditioned graph, represent the hidden state as a graph signal, and let a bank of learnable spectral filters parameterize the linear transport inside a gated recurrent update. The router uses the current block input and graph spectrum to choose a small set of filters; it does not use that block's preceding recurrent state. The resulting model couples memory and cross-variable propagation while retaining an explicit spectral interpretation. We evaluate on PhysioNet 2019 (P19), a large-scale benchmark with 40,336 patient trajectories, 2,904,192 hourly windows, 98,742,528 variable-hour positions, and 89.8612% missingness. Our method outperforms the state-of-the-art (SOTA) baselines on AUROC and shows effectiveness. Main contributions are as follows.

- We introduce a graph-signal latent state and a spectral transition for selective SSMs, addressing the lack of cross-variable propagation inside channel-separable recurrence and making graph-frequency propagation intrinsic to the state update.
- We develop a dynamic observation graph, a learnable B-spline filter bank, and top-s input-dependent spectral routing with an explicit sparse-simplex objective, addressing the gap between external graph modules and recurrent spectral state evolution.
- We establish graph-Fourier diagonalization of the linear transport, prove top-s optimality when the support is selected jointly with its weights, characterize when the realized graph-spectral operator is non-diagonal in measurement coordinates, and prove a uniform bound for the implemented state update.

2 Related Work

State space sequence models. Structured SSMs (Gu et al., 2022; Gu and Dao, 2023) obtain linear-time sequence processing by restricting or structuring the transition . Their standard channel-separable form does not encode a relation graph among variables. Consequently, cross-variable evidence cannot be propagated inside the recurrence, and graph structure must be injected post hoc.

Irregular multivariate time series. Attention-based models such as mTAND (Shukla and Marlin, 2021) handle irregular observation times, and graph-guided models such as Raindrop (Zhang et al., 2022) incorporate cross-variable structure. However, their temporal and graph mechanisms remain separate from a graph-spectral recurrent transition. Thus, irregularity is addressed outside the graph-spectral recurrence; S^{3}M makes such a transition the state-update operator itself.

Spectral and graph time-series models. Localized spectral filtering and its polynomial approximations are well established . Recent graph and frequency-guided models (Defferrard et al., 2016; Kipf and Welling, 2017) typically use these operators as an encoder or forecasting block . However, in these designs, graph-spectral operators are usually external to recurrence, and their frequency bands are not selected online. In contrast to these modular designs, S^{3}M places a learned graph-spectral operator inside the recurrent state update and routes its frequency bands online, thereby coupling spectral propagation with state evolution.

The three relevant model families address complementary aspects of the problem: SSMs organize temporal memory, irregular-time models encode observation times and masks, and graph models mix information across variables. However, these capabilities are typically realized by separate mechanisms, and the recurrent transition itself is rarely graph-spectral, input-selectively sparse, and availability-aware. S^{3}M targets this research gap by coupling memory, cross-variable propagation, and spectral routing in a single state update. The resulting division of roles is summarized in Table 1.

Table 1: Positioning of S^{3}M against representative recent families.

Method family	Cross-variable state	Spectral transition	Input-dependent routing	Missingness-aware readout
Diagonal SSMs (Gu and Dao, 2023)	No	No	Temporal	Optional
Irregular attention models (Shukla and Marlin, 2021)	Encoder-level	No	Attention	Yes
Graph-guided irregular models (Zhang et al., 2022)	Graph block	Separate	Usually no	Varies
Spectral GNNs (Defferrard et al., 2016)	Graph block	Yes	Usually no	No
S^{3}M (ours)	Recurrent graph signal	Yes	Sparse spectral	Yes

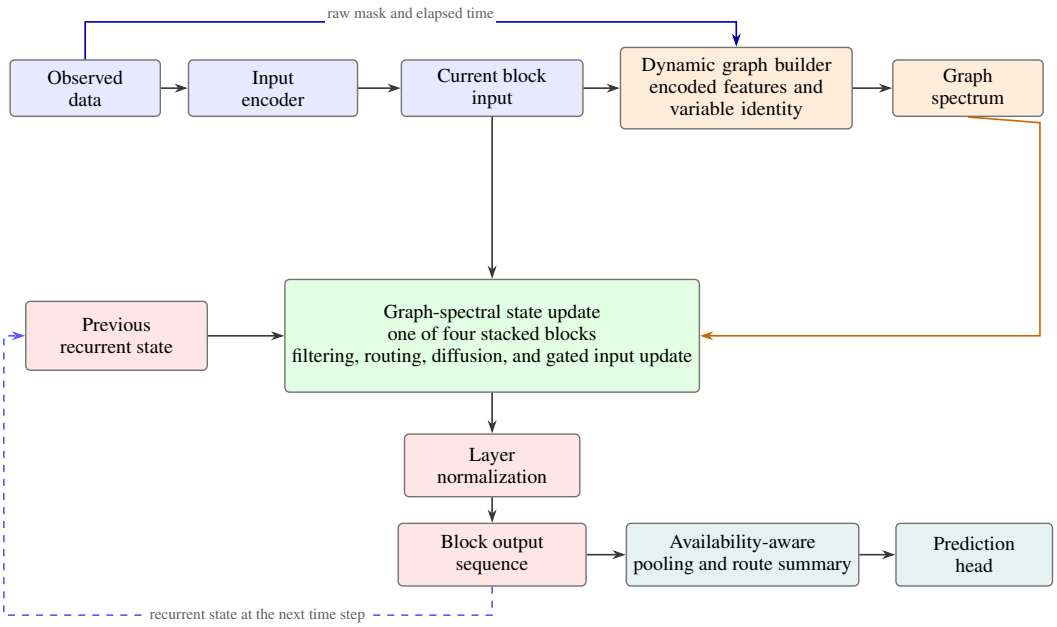

Figure 1: Architecture of S^{3}M. The observed sequence is encoded into the current block input. At each block and time step, the current representation and variable identity, together with the raw observation mask and elapsed time, construct the dynamic graph and its spectrum. The current block input, graph spectrum, and previous recurrent state participate in the graph-spectral state update, which contains the filter bank, sparse router, explicit Laplacian diffusion, current-input projection, and gated nonlinear recurrence. The normalized block output is passed to the next time step; four blocks with separate parameters are stacked. The final block sequence is pooled using observation availability and combined with the routing summary before prediction.

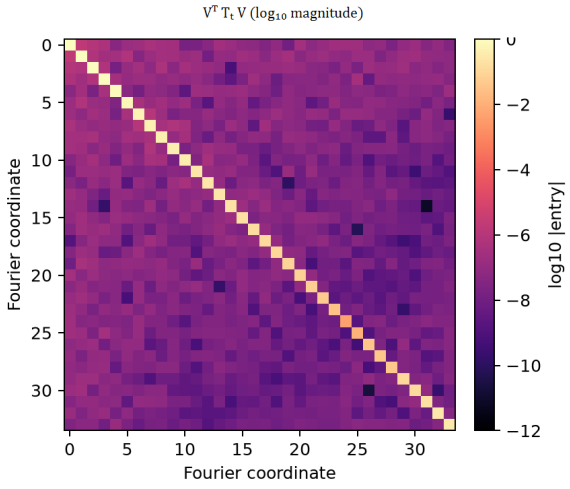

Figure 2: Shared-basis audit for the linear transport, before the gate and layer normalization. The heatmap shows $\log_{10}|(V_t^{(\ell)})^\top \mathcal{T}_t^{(\ell)} V_t^{(\ell)}|$ for one representative P19 transition; the off-diagonal Frobenius ratio averaged 1.59×10^{-6} across the audit. Here $\mathcal{T}_t^{(\ell)}$ is the transport operator, while $A_t^{(\ell)}$ denotes graph affinity.

3 METHOD

For a sequence with D variables and T ordered observations, let $U_t = [x_t, m_t, \Delta_t] \in \mathbb{R}^{D\times 3}$ contain value, availability, and elapsed-time channels. The learned variable identifier s_i is not part of this input tensor: it enters only the graph encoder. We use $L = 4$ blocks and hidden width h. Let $X_t^{(1)}$ be the encoded input and $X_t^{(\ell)} = H_t^{(\ell-1)}$ for $\ell > 1$; each block maps its full input sequence to a recurrent state sequence $H_{1:T}^{(\ell)}$. The model starts each block from $H_0^{(\ell)} = 0$. All symbols below carry a block index when their parameters or graph differ across blocks.

Notation. The variable index is the fixed canonical measurement basis. For block ℓ and step t, $A_t^{(\ell)}$ is the affinity matrix, $W_t^{(\ell)}$ its symmetrized self-looped version, $Q_t^{(\ell)} = \mathrm{diag}(W_t^{(\ell)}\mathbf{1})$ the degree matrix, and $L_t^{(\ell)} = I - (Q_t^{(\ell)})^{-1/2} W_t^{(\ell)} (Q_t^{(\ell)})^{-1/2}$ the normalized Laplacian. We write $L_t^{(\ell)} = V_t^{(\ell)} \Lambda_t^{(\ell)} (V_t^{(\ell)})^\top$ with orthogonal $V_t^{(\ell)}$. A response $g_{\ell,k}(\Lambda_t^{(\ell)})$ maps back to measurement coordinates as $G_{t,k}^{(\ell)} = \mathrm{sg}(V_t^{(\ell)}) g_{\ell,k}(\Lambda_t^{(\ell)})\, \mathrm{sg}(V_t^{(\ell)})^\top$, where sg is the identity in the forward pass and blocks gradients in the backward pass; $\pi_{t,k}^{(\ell)}$ are sparse, nonnegative weights summing to one, and $\bar{G}_t^{(\ell)} = \sum_k \pi_{t,k}^{(\ell)} G_{t,k}^{(\ell)}$. We use $\xi \in [0, 2]$ for a generic normalized-Laplacian eigenvalue, κ_ℓ for the learned graph-diffusion coefficient, and u_ℓ for the learned update-gate scalar. The linear transport $\mathcal{T}_t^{(\ell)} = \bar{G}_t^{(\ell)} - \kappa_\ell L_t^{(\ell)}$ is followed by the input projection, gate, pointwise $\tanh$, and layer normalization in the implemented recurrence.

Let $e(\Delta) = \mathrm{clip}_{[0,20]}(\log(1+\min(|\Delta|, 10^6)))/10$, the elapsed-time feature used by the graph branch. For graph width d_f, temperature τ_g, and self-loop γ, the code constructs

$$
\begin{aligned}
f_{t,i}^{(\ell)} &= \psi_\ell([X_{t,i}^{(\ell)}, s_i, m_{t,i}, e(\Delta_{t,i})]), \qquad a_{t,ij}^{(\ell)} = \begin{cases} 0, & i = j, \\ \exp\left(\mathrm{clip}_{[-\tau_g,\tau_g]} \frac{(f_{t,i}^{(\ell)})^\top f_{t,j}^{(\ell)}}{\sqrt{d_f}}\right), & i \neq j, \end{cases} \\
A_t^{(\ell)} &= [a_{t,ij}^{(\ell)}]_{i,j=1}^{D}, \qquad W_t^{(\ell)} = \tfrac{1}{2}(A_t^{(\ell)} + (A_t^{(\ell)})^\top) + \gamma I, \\
Q_t^{(\ell)} &= \mathrm{diag}(W_t^{(\ell)}\mathbf{1}), \qquad L_t^{(\ell)} = I - (Q_t^{(\ell)})^{-1/2} W_t^{(\ell)} (Q_t^{(\ell)})^{-1/2}.
\end{aligned}
\tag{1}
$$

Here ψ_ℓ is the learned graph-feature MLP, and s_i is a learned variable identifier used only in this branch. The elapsed-time transformation is applied by the graph module; the input encoder receives the third channel $\Delta_{t,i}$ as stored in the preprocessed input. Since the code exponentiates every off-diagonal score, $A_t^{(\ell)}$ is positive off-diagonal before symmetrization. Thus $W_t^{(\ell)}$ is symmetric with positive degrees, and $L_t^{(\ell)}$ is positive semidefinite with eigenvalues in $[0, 2]$.

3.1 SPECTRAL STATE SPACE MODEL

3.1.1 INPUT ENCODING AND SPECTRAL FILTER BANK

Figure 1 gives an overview of S^{3}Marchitecture. The initial input encoder receives the three observed channels and does not receive s_i or a log-transformed elapsed time:

$$
X_t^{(1)} = \mathrm{Dropout}(\mathrm{Linear}_2(\mathrm{GELU}(\mathrm{LayerNorm}(\mathrm{Linear}_1(U_t))))), \qquad X_t^{(1)} \in \mathbb{R}^{D\times h}. \tag{2}
$$

For each block, let $\{B_r\}_{r=1}^R$ be the fixed degree-4 B-spline basis on $[0, 2]$. The kth filter in block ℓ is the bounded response

$$
g_{\ell,k}(\xi) = 2\,\mathrm{sigmoid}\left(\sum_{r=1}^{R} \theta_{\ell,k,r} B_r(\xi)\right), \qquad G_{t,k}^{(\ell)} = \mathrm{sg}(V_t^{(\ell)}) g_{\ell,k}(\Lambda_t^{(\ell)})\, \mathrm{sg}(V_t^{(\ell)})^\top. \tag{3}
$$

The input encoder is a Linear–LayerNorm–GELU–Linear stack followed by dropout, as in the released implementation. Each block has its own graph encoder and filter coefficients; the response is bounded between zero and two. The released configuration sets `detach_eigenvectors=true`: gradients through $V_t^{(\ell)}$ are stopped in the spectral-filter branch, while gradients through the eigenvalues $\Lambda_t^{(\ell)}$ and the separate Laplacian-diffusion term remain enabled. This stop-gradient changes only the backward pass; it leaves the forward operator and the algebraic spectral results below unchanged.

3.1.2 SPARSE SPECTRAL TRANSITION

To keep the transition equations compact, we suppress the block superscript ℓ in Eqs. (4)–(6) where unambiguous; the four blocks retain separate parameters. The router uses the current block input, four graph-spectrum summaries, and normalized sequence position. Define $\tau_t = (t-1)/\max(T-1,1)$ and

$$c_t^{(\ell)} = \left[D^{-1} \sum_i X_{t,i}^{(\ell)};\ \mathrm{mean}(\xi_t^{(\ell)});\ \mathrm{std}(\xi_t^{(\ell)});\ \min_j \xi_{t,j}^{(\ell)};\ \max_j \xi_{t,j}^{(\ell)};\ \tau_t;\ \sin(\pi\tau_t) \right],$$

where $\xi_t^{(\ell)}$ is the vector of eigenvalues of $L_t^{(\ell)}$. A learned MLP maps this context to $q_t \in \mathbb{R}^K$; let $w_t = \mathrm{softmax}(q_t)$ and $S_t = \mathrm{Top}_s(w_t) = \mathrm{Top}_s(q_t)$. The sparse weights restrict and renormalize the dense softmax. At a score tie on the cutoff, any tied top-s support is valid:

$$\begin{aligned} q_t &= \mathrm{MLP}_{\mathrm{router}}(c_t), \quad w_t = \mathrm{softmax}(q_t), \quad S_t = \mathrm{Top}_s(w_t) = \mathrm{Top}_s(q_t), \\ \pi_{t,k} &= \frac{\mathbf{1}\{k \in S_t\} w_{t,k}}{\sum_{j \in S_t} w_{t,j}}, \quad \bar{G}_t = \sum_{k=1}^{K} \pi_{t,k} G_{t,k}. \end{aligned} \tag{4}$$

Thus the router score is not computed from H_{t-1} of the same block: it is computed in parallel for all time steps from X_t and the corresponding graph spectrum before the recurrent scan. The linear transport is

$$\begin{aligned} \mathcal{T}_t &= \bar{G}_t - \kappa_\ell L_t, & P_t &= \mathcal{T}_t H_{t-1} + W_{\mathrm{in}}^{(\ell)} X_t, \\ Z_t &= (1-u_\ell) H_{t-1} + u_\ell \tanh(P_t), & H_t &= \mathrm{LayerNorm}_\ell(Z_t). \end{aligned} \tag{5}$$

Here $\kappa_\ell = 2\,\mathrm{sigmoid}(a_\ell)$ is the learned diffusion coefficient bounded by two, initialized at 0.75, and $u_\ell = \mathrm{sigmoid}(b_\ell)$ is a learned scalar update gate initialized at 0.7. $W_{\mathrm{in}}^{(\ell)}$ is the block's learned linear input projection. The code has no time-varying B_t, output map C_t, or feedthrough map E_t: the normalized state itself is the block output.

3.1.3 AVAILABILITY-AWARE PREDICTION

Let $r_{t,i}$ be the learned linear pooling score from the final block's state. For the binary availability mask $m_{t,i} \in \{0,1\}$, the code masks unavailable entries with a large negative score before applying a softmax over all time-variable positions. Writing the resulting normalized weights as $\alpha_{t,i}$, the final representation is

$$\begin{aligned} \widetilde{r}_{t,i} &= m_{t,i} r_{t,i} - (1-m_{t,i}) 10^4, \quad \alpha_{t,i} = \mathrm{softmax}(\widetilde{r})_{t,i}, \quad z = \sum_{t,i} \alpha_{t,i} H_{t,i}^{(L)}, \\ \bar{\pi} &= \frac{1}{LT} \sum_{\ell=1}^{L} \sum_{t=1}^{T} \pi_t, \qquad \widehat{y} = \mathrm{head}([z, \bar{\pi}]). \end{aligned} \tag{6}$$

The same availability mask used in the input also controls readout, and routing summaries average the sparse weights across all blocks and time steps. The implementation does not introduce a separate zero-vector fallback for an all-missing sequence.

3.1.4 TRAINING OBJECTIVE

For a minibatch of B patient logits a_b and labels $y_b \in \{0,1\}$, the model is trained with positive-weighted binary cross-entropy, a temporal smoothness penalty on router weights, and an L_2 penalty on filter coefficients:

$$\begin{aligned} \mathcal{L}_{\mathrm{WBCE}} &= -\frac{1}{B} \sum_{b=1}^{B} \left[\omega_+ y_b \log \sigma(a_b) + (1-y_b) \log(1-\sigma(a_b)) \right], \\ \mathcal{L}_{\mathrm{smooth}} &= \frac{1}{LBK(T-1)} \sum_{\ell=1}^{L} \sum_{b=1}^{B} \sum_{t=2}^{T} \sum_{k=1}^{K} \left(\pi_{b,t,k}^{(\ell)} - \pi_{b,t-1,k}^{(\ell)} \right)^2, \\ \mathcal{L}_{\mathrm{filter}} &= \frac{1}{LKR} \sum_{\ell=1}^{L} \sum_{k=1}^{K} \sum_{r=1}^{R} \theta_{\ell,k,r}^2, \quad \mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{WBCE}} + 10^{-2} \mathcal{L}_{\mathrm{smooth}} + 10^{-4} \mathcal{L}_{\mathrm{filter}}. \end{aligned} \tag{7}$$

Here σ is the sigmoid and $\omega_+ = N_-/\max(N_+, 1)$, where N_+ and N_- are positive and negative patient counts from the training split only. Each penalty is averaged over the same axes as in the implementation: examples for the task loss; adjacent times, filters, and blocks for router smoothness; and filter coefficients and blocks for L_2 regularization.

3.2 Theoretical Analysis

Definition 1 (Graph-spectral linear transport)**.** *For block ℓ and time t, define $d_{t,j}^{(\ell)} = \sum_{k=1}^{K} \pi_{t,k}^{(\ell)} g_{\ell,k}(\xi_{t,j}^{(\ell)}) - \kappa_\ell \xi_{t,j}^{(\ell)}$ and $\mathcal{T}_t^{(\ell)} = \bar{G}_t^{(\ell)} - \kappa_\ell L_t^{(\ell)}$. This is the linear transport component of the implemented block, before input projection, update gate,* tanh*, and layer normalization. It is distinct from the affinity matrix $A_t^{(\ell)}$.*

Lemma 1 (Spectral coordinates at a fixed step)**.** *Let $L_t^{(\ell)} = V_t^{(\ell)} \Lambda_t^{(\ell)} (V_t^{(\ell)})^\top$ and write $\mathcal{T}_t^{(\ell)} = V_t^{(\ell)} \operatorname{diag}(d_t^{(\ell)}) (V_t^{(\ell)})^\top$. For a state vector h_{t-1}, define current-basis coordinates $\widetilde{h}_{t-1|t} = (V_t^{(\ell)})^\top h_{t-1}$ and $\widetilde{h}_{t|t} = (V_t^{(\ell)})^\top h_t$. Then the homogeneous transport $h_t = \mathcal{T}_t^{(\ell)} h_{t-1}$ is equivalent to $\widetilde{h}_{t|t} = \operatorname{diag}(d_t^{(\ell)}) \widetilde{h}_{t-1|t}$. If the graph is fixed over time, the linear transport acts coordinatewise in one graph Fourier basis. This statement applies only to that linear component: the implemented input projection, pointwise* tanh*, and per-variable layer normalization make the full state update nonlinear and do not preserve independent scalar graph-frequency recurrences.*

Proof. By definition, $\mathcal{T}_t^{(\ell)} = V_t^{(\ell)} \operatorname{diag}(d_t^{(\ell)}) (V_t^{(\ell)})^\top$. Multiplication on the left by $(V_t^{(\ell)})^\top$ and orthogonality give $(V_t^{(\ell)})^\top h_t = \operatorname{diag}(d_t^{(\ell)}) (V_t^{(\ell)})^\top h_{t-1}$; multiplying by $V_t^{(\ell)}$ recovers the variable-coordinate update. When the graph changes, $V_t^{(\ell)\top} h_{t-1}$ need not equal $V_{t-1}^{(\ell)\top} h_{t-1}$, which is why the basis is indexed by the receiving step. Since $d_{t,j}^{(\ell)}$ is a scalar function of $\xi_{t,j}^{(\ell)}$, equal eigenvalues receive equal responses, including within repeated-eigenvalue eigenspaces. □

Thus the code's graph-compatible linear transport becomes scalar multiplication in the graph Fourier coordinates chosen at each receiving step; the subsequent nonlinear operations remain part of the full block.

Implementation of the shared-basis assumption. For each block and step, both $\bar{G}_t^{(\ell)}$ and $L_t^{(\ell)}$ are functions of the same symmetric Laplacian, so the linear transport is exactly diagonalized by $V_t^{(\ell)}$ in algebra. We audited the validation-selected P19 checkpoint by measuring $(V_t^{(\ell)})^\top \mathcal{T}_t^{(\ell)} V_t^{(\ell)}$ before the pointwise gate and layer normalization. Across the audited batch, the mean off-diagonal Frobenius ratio was 1.59×10^{-6}, the median was 1.56×10^{-6}, and the maximum was 2.76×10^{-6} (maximum absolute off-diagonal entry 4.14×10^{-6}), consistent with floating-point and eigendecomposition reconstruction error. Figure 2 shows a representative matrix from this audit.

Lemma 2 (Shared spectral basis of the transport)**.** *With the definitions in eqs. (3) and (4), the linear transport in block ℓ satisfies $\mathcal{T}_t^{(\ell)} = V_t^{(\ell)} \operatorname{diag}(d_{t,1}^{(\ell)}, \ldots, d_{t,D}^{(\ell)}) (V_t^{(\ell)})^\top$, $\mathcal{T}_t^{(\ell)} L_t^{(\ell)} = L_t^{(\ell)} \mathcal{T}_t^{(\ell)}$, and $\left\| \mathcal{T}_t^{(\ell)} \right\|_2 = \max_j |d_{t,j}^{(\ell)}|$.*

Proof. Write $g_{\ell,k}(\Lambda_t^{(\ell)}) = \operatorname{diag}(g_{\ell,k}(\xi_{t,1}^{(\ell)}), \ldots, g_{\ell,k}(\xi_{t,D}^{(\ell)}))$. By linearity, $\bar{G}_t^{(\ell)} = V_t^{(\ell)} \operatorname{diag}(\sum_k \pi_{t,k}^{(\ell)} g_{\ell,k}(\xi_{t,j}^{(\ell)}))_j (V_t^{(\ell)})^\top$. Subtracting $\kappa_\ell L_t^{(\ell)}$ gives the displayed diagonal entries $d_{t,j}^{(\ell)}$. Both matrices are diagonal in the same orthogonal basis, so they commute; symmetry then gives the stated spectral norm. □

Lemma 3 (Support optimality with variable support)**.** *Let $u_t \in \mathbb{R}^K$, $1 \le s \le K$, and $\eta > 0$. Define $f_t(\pi) := -u_t^\top \pi + \eta \sum_k \pi_k \log \pi_k$ and consider $\min_{\pi \in \Delta_{K-1},\, \|\pi\|_0 \le s} f_t(\pi)$, where $\Delta_{K-1} := \{\pi \ge 0 : \mathbf{1}^\top \pi = 1\}$ and $0 \log 0 := 0$. Every minimizer has support equal to a set of s largest scores, up to ties. On any fixed support S, the unique minimizer is the restricted softmax $\pi_k^{(S)} = \exp(u_{t,k}/\eta) / \sum_{j \in S} \exp(u_{t,j}/\eta)$ for $k \in S$, and $\pi_k^{(S)} = 0$ otherwise.*

Proof. Fix a nonempty support S with $|S| \le s$ and impose $\pi_k = 0$ for $k \notin S$. On the relative interior of this face the Lagrangian is $\mathcal{L}(\pi, \nu) = -\sum_{k\in S} u_{t,k}\pi_k + \eta \sum_{k\in S} \pi_k \log \pi_k + \nu(\sum_{k\in S} \pi_k - 1)$. Stationarity gives $\pi_k = c \exp(u_{t,k}/\eta)$, and normalization gives the restricted softmax. Strict convexity of entropy makes it unique on this face, with optimized value $f_t^\star(S) := \min_{\mathrm{supp}(\pi)\subseteq S} f_t(\pi) = -\eta \log(\sum_{k\in S} e^{u_{t,k}/\eta})$. Let $Z(S) = \sum_{k\in S} e^{u_{t,k}/\eta}$. If $i \notin S$, $j \in S$, and $u_{t,i} > u_{t,j}$, the exchange $S' = (S\backslash\{j\})\cup\{i\}$ satisfies $Z(S') - Z(S) = e^{u_{t,i}/\eta} - e^{u_{t,j}/\eta} > 0$, so $-\eta \log Z(S') < -\eta \log Z(S)$. Repeating exchanges terminates because there are finitely many supports and yields a support containing the s largest scores. If scores tie at the boundary, replacing one tied index leaves $Z(S)$ unchanged, so all tied choices are minimizers. Finally, if $|S| < s$ and $s < K$, adding any index strictly increases $Z(S)$ and strictly lowers the objective; therefore every minimizer has s active entries (and when $s = K$, the ordinary softmax has all K entries positive). This proves top-s optimality even though support selection and weight optimization are performed jointly. □

Taking the optimization scores to be $q_t^{(\ell)}$ and $\eta = 1$ recovers the router in eq. (4): softmax preserves score ordering, and restricting then renormalizing the dense softmax gives the restricted softmax on the selected support. Therefore the code's top-s rule solves the stated sparse-simplex objective, including support selection; at a boundary tie, every tied top-s support is an optimizer.

Lemma 4 (When a graph-basis diagonal operator is coordinate diagonal)**.** *Let $V \in \mathbb{R}^{D\times D}$ be orthogonal. The following are equivalent: (i) $V\,\mathrm{diag}(d)V^\top$ is diagonal in the canonical variable basis for every $d \in \mathbb{R}^D$; (ii) V is a signed permutation matrix. Consequently, if V is not a signed permutation, there exists a diagonal response d for which $V\,\mathrm{diag}(d)V^\top$ has a nonzero off-diagonal entry.*

Proof. For each coordinate k, choose $d = e_k$, the kth standard basis vector. Then $V\,\mathrm{diag}(e_k)V^\top = v_k v_k^\top$, where v_k is the kth column of V. A rank-one matrix $v_k v_k^\top$ is diagonal only if v_k has at most one nonzero coordinate: otherwise an off-diagonal product $(v_k)_i(v_k)_j$ is nonzero for some $i \ne j$. Orthogonality forces each column to have exactly one entry of magnitude one, and distinct columns must use distinct coordinates; hence V is a signed permutation. The converse is immediate because conjugating a diagonal matrix by a signed permutation only permutes and signs its diagonal entries. Taking the contrapositive proves the final statement. □

For a fixed response d, the off-diagonal entries are $\left[V\,\mathrm{diag}(d)V^\top\right]_{ij} = \sum_{r=1}^D d_r v_{ir} v_{jr}$ for $i \ne j$. A scalar graph response is constant within each repeated-eigenvalue eigenspace, so rotations within such an eigenspace do not change the operator. A nonzero off-diagonal entry in the canonical variable basis cannot be represented by any diagonal channel-separable transition in that same basis.

Proposition 1 (Non-diagonal transport on the learned graph)**.** *Fix a block and time step. The implemented graph has $L_{ij} < 0$ for every $i \ne j$, because its exponentiated off-diagonal affinities and self-looped degrees are strictly positive. If the realized spectral response $d_{t,j}^{(\ell)}$ is nonconstant across the graph eigenvalues, then $\mathcal{T}_t^{(\ell)}$ has a nonzero off-diagonal entry in the canonical variable basis and cannot be represented by a diagonal channel-separable transition in that basis.*

Proof. The transport is a scalar function of $L_t^{(\ell)}$, so $\mathcal{T}_t^{(\ell)} L_t^{(\ell)} = L_t^{(\ell)} \mathcal{T}_t^{(\ell)}$. Nonconstant $d_{t,j}^{(\ell)}$ implies that $\mathcal{T}_t^{(\ell)}$ is not a scalar multiple of the identity. Suppose instead that it were diagonal in measurement coordinates, say $C = \mathrm{diag}(c_1, \ldots, c_D)$. The (i, j) entry of $CL - LC$ is $(c_i - c_j)L_{ij}$. Since every off-diagonal L_{ij} is nonzero, commutation forces $c_i = c_j$ for every pair, so $C = cI$, a contradiction. Therefore the transport is non-diagonal in measurement coordinates. This argument uses the graph basis and its realized scalar response; it does not claim that the constrained $\mathrm{S}^3\mathrm{M}$ parameterization contains every diagonal transition or every member of an unrestricted spectral family. □

Theorem 1 (Uniform bound for the implemented state)**.** *Consider one block with hidden width h, D variables, finite parameters, and the update in eq. (5). Let $\gamma_{\ell,j}$ and $\beta_{\ell,j}$ be the affine scale and shift of its feature-wise layer normalization, and set $\Gamma_\ell = \max_j |\gamma_{\ell,j}|$ and $B_\ell = \max_j |\beta_{\ell,j}|$. For every finite input sequence and every step $t \ge 1$, $\max_{i,j} |H_{t,i,j}^{(\ell)}| \le \sqrt{h}\,\Gamma_\ell + B_\ell$ and $\left\|H_t^{(\ell)}\right\|_F \le \sqrt{Dh}\,(\sqrt{h}\,\Gamma_\ell + B_\ell)$. The bound is independent of sequence length, graph spectrum, and input magnitude.*

Proof. For a node's pre-normalization vector $z \in \mathbb{R}^h$, write $\bar{z} = h^{-1} \sum_j z_j$, $v = z - \bar{z}\mathbf{1}$, and $\sigma^2 = h^{-1} \sum_j v_j^2$. The zero-sum condition and Cauchy–Schwarz give $|v_j| \le \sqrt{h}\,\sigma$ for each coordinate. The implemented layer normalization is $\gamma_{\ell,j} v_j / \sqrt{\sigma^2 + \epsilon} + \beta_{\ell,j}$ with $\epsilon > 0$, hence its absolute value is at most $\sqrt{h}\,\Gamma_\ell + B_\ell$ (when $\sigma = 0$, the normalized term is zero). This applies to the update-gated $\tanh$ proposal at every step, regardless of how that proposal was formed. Summing the squared coordinate bounds over Dh entries yields the Frobenius bound. The initial state is zero and all operations are finite for finite inputs and parameters, so the result holds at every recurrent step. □

Table 2: P19 native-missingness comparison on the fixed full-dataset `neighbor_v1` split (70/15/15 split, split seed 7). AUROC is the standard primary metric. Values are mean ± sample standard deviation over training seeds 7, 8, and 9 for each method.

Method	AUROC	AUPRC	Accuracy	Macro-F1
S^3M (ours)	**0.9239**±0.0048	0.6669±0.0189	0.8431±0.0305	0.6791±0.0308
mTAND (Shukla and Marlin, 2021)	0.8377±0.0015	0.3278±0.0199	0.7583±0.0399	0.5836±0.0257
DGM2 (Wu et al., 2021)	0.8486±0.0196	0.5087±0.0310	0.8384±0.0212	0.6515±0.0240
iTimER (Liu et al., 2026a)	0.8371±0.0027	0.3260±0.0048	0.7465±0.0064	0.5751±0.0022
TimeCHEAT (Liu et al., 2025)	0.8810±0.0351	0.6092±0.0799	**0.9154**±0.0152	**0.7492**±0.0143
Pyraformer (Liu et al., 2022)	0.9009±0.0032	0.6300±0.0140	0.8658±0.0038	0.6879±0.0040
DLinear (Zeng et al., 2023)	0.8797±0.0185	0.5828±0.0347	0.8503±0.0164	0.6688±0.0228
TimesNet (Wu et al., 2023)	0.8804±0.0053	0.5647±0.0247	0.8172±0.0246	0.6417±0.0214
iTransformer (Liu et al., 2024)	0.8979±0.0015	0.6171±0.0197	0.8867±0.0213	0.7145±0.0247
NCDE (Kidger et al., 2020)	0.9084±0.0051	0.5945±0.0198	0.8778±0.0042	0.7025±0.0042
Koopa (Liu et al., 2023)	0.9060±0.0006	0.6218±0.0022	0.8660±0.0190	0.6918±0.0196
FreTS (Yi et al., 2023)	0.8126±0.0148	0.4485±0.1670	0.8174±0.0804	0.6292±0.0679
TiDE (Das et al., 2023)	0.9132±0.0024	**0.6784**±0.0142	0.8989±0.0096	0.7337±0.0130
FITS (Xu et al., 2024)	0.9004±0.0006	0.6038±0.0075	0.8394±0.0227	0.6622±0.0202
TimeMixer (Wang et al., 2024a)	0.8829±0.0288	0.6200±0.0311	0.8545±0.0422	0.6771±0.0478
RLinear (Li et al., 2024)	0.9048±0.0037	0.6046±0.0057	0.8984±0.0047	0.7311±0.0052

Table 3: Published P19 AUROC context under each cited source protocol.

Method	Venue	Source AUROC (%)	Source protocol
Raindrop (Zhang et al., 2022)	ICLR 2022	87.0	Fixed 80:10:10 split, five runs
TimeCHEAT (Liu et al., 2025)	AAAI 2025	89.5	8:1:1 fixed split
iTimER (Liu et al., 2026a)	AAAI 2026	87.1	8:1:1 fixed split
SPECTRA (Qin et al., 2026)	AAAI 2026	90.0	Five-fold CV with 8:1:1 splits
NCDE (Kidger et al., 2020)	NeurIPS 2020	88.0±0.6	observational-intensity (OI) setting, five runs
S^3M (ours)	this work	**92.39**±0.48	Fixed 70:15:15 split ; three runs

Table 4: Ablation study of the S^3M under training seed 7. We perform the ablation only for seed 7 due to computational cost: a full-dataset run takes approximately 60.3 wall-clock hours (about 2.5 days) on an NVIDIA RTX A6000 (48GB) GPU, and running the full model plus the six ablated variants requires 7 such runs (about 422 GPU-hours, i.e., about 17.6 days).

Variant	AUROC	Δ AUROC	AUPRC	Accuracy	Macro-F1
Full S^3M	**0.9187**	**0.00**	**0.6456**	0.8104	0.6472
Identity graph	0.7822	−13.66	0.3714	**0.9124**	0.6779
No spectral filter	0.7696	−14.91	0.3754	0.4996	0.4218
Dense filter selection	0.9176	−0.12	0.6415	0.8015	0.6387
No Laplacian diffusion	0.8645	−5.42	0.4362	0.9099	**0.7064**
No state recurrence	0.6266	−29.21	0.0992	0.0752	0.0706
Unmasked readout	0.5171	−40.16	0.0830	0.0727	0.0678

4 RESULTS

Experimental implementation details are described in the Reproducibility Statement section. Table 2 reports comparison results between our method and SOTA baselines on the full P19 dataset. All rows use the same fixed split (70/15/15, split seed 7, native missingness); external means and sample standard deviations are computed across training seeds 7, 8, and 9. Table 3 records source-reported values under each cited paper's protocol, as well as controlled fixed-split results. These source

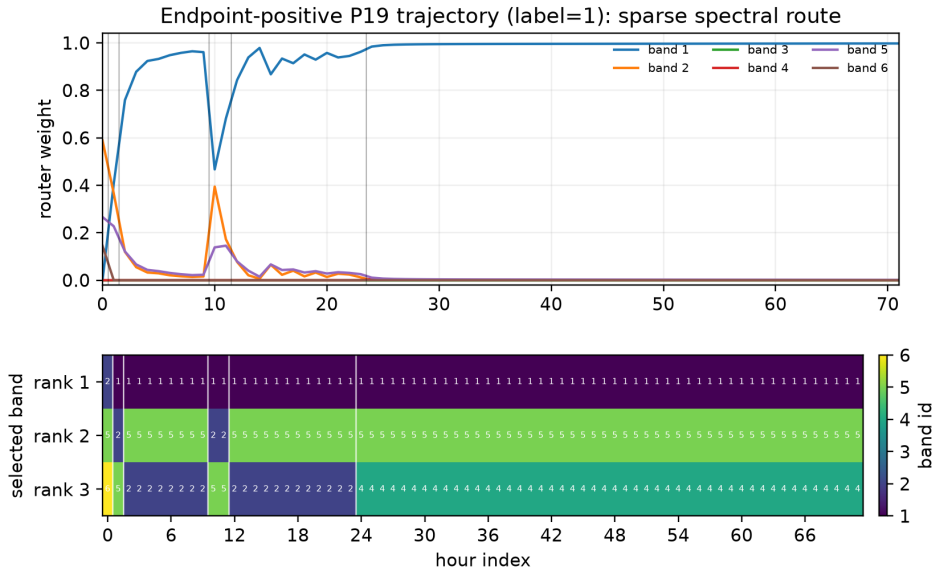

Figure 3: Outcome-stratified mechanism case. Endpoint-positive P19 risk-proxy case (trajectory label 1). Top: router weights for the six learned spectral bands. Bottom: the selected top-three support at each hour; white separators mark the observed support changes.

values provide context and are taken from the cited papers. The comparison results show that our method outperforms the SOTA baselines on AUROC, the standard primary metric for P19. For training seed 7, we apply seven predeclared forward interventions to the identical validation-selected checkpoint and 6,051-example test partition; parameters are frozen, so each variant is a controlled computation-graph intervention reported as a point estimate; the resulting intervention scores are reported in Table 4. Removing the spectral filter, graph coupling, and recurrent memory lowers AUROC by 14.91, 5.42, and 29.21 points; unmasked readout falls to 0.5171 AUROC and 0.0830 AUPRC. Dense and sparse routing perform comparably, with sparse routing slightly higher, so sparsity provides a small performance gain while preserving an inspectable route. The higher accuracy of the identity-graph and no-Laplacian-diffusion controls reflects thresholding under class imbalance. Across all seven interventions, AUROC degrades relative to the full model, confirming that each targeted component contributes positively to performance. P19 provides a trajectory-level endpoint label but no verified hourly onset timestamp. Figure 3 visualizes the corresponding routed weights and selected supports. In an endpoint-positive risk-proxy case, the top-three support changes at hours 1, 2, 10, 12, and 24 (largest change $1/3$), then stabilizes on bands 1, 5, and 4. These graph-Laplacian modes illustrate an inspectable route. Overall, the router settles on a compact set of spectral bands after early support changes, illustrating a sparse and inspectable route.

5 Conclusion

We propose S^{3}M to couple a dynamic observation graph with a sparse graph-spectral recurrent transition, making temporal memory, cross-variable propagation, and spectral routing intrinsic to the state update. On the locked P19 test split, S^{3}M outperorms SOTA baselines on AUROC. Future work includes extending the framework to other structured multivariate domains.

Reproducibility Statement

We evaluate S^{3}M on the large-scale PhysioNet 2019 sepsis (P19) benchmark. The processed corpus contains 40,336 patient trajectories, 2,904,192 hourly windows, 98,742,528 variable-hour positions, and 10,011,306 observed entries, corresponding to an 89.8612% missing rate. We use the full-dataset `neighbor_v1` preprocessing: 34 dynamic variables, three input channels (value, availability,

elapsed time), train-only standardization, and a stratified 70/15/15 train/validation/test split with split seed 7. The means and sample standard deviations are computed across training seeds 7, 8, and 9. No test quantity enters model selection. Each trainning seed run followed a training protocol of up to 50 epochs with early stopping (patience = 6). The best checkpoint was selected based on validation AUROC, and the corresponding test AUROC was evaluated on the locked test partition containing 6,051 examples. Candidate selection was based solely on validation AUROC. The input encoder is Linear–LayerNorm–GELU–Linear followed by dropout. Each of the four blocks uses hidden width $h = 128$, $K = 6$ B-spline filters, top-$s = 3$ routing, a graph-feature width of 64, and its own learned 16-dimensional variable-identifier embedding, which enters only that block's graph encoder. The graph encoder also receives the current block input, availability mask, and elapsed-time feature $e(\Delta)$ defined in the method. Each filter uses a degree-4 B-spline basis with eight functions. The graph temperature is 8.0 and the self-loop is 0.1. The learned diffusion coefficient is initialized at 0.75 and bounded by 2.0; the learned scalar update gate is initialized at 0.7. Seed 7 uses batch size 640, while seeds 8 and 9 use batch size 128. We optimize with AdamW, learning rate 5×10^{-4}, weight decay 10^{-4}, and global gradient-norm clipping at 1.0; the penalty weights are given in Eq. 7. EMA uses decay 0.995; validation checkpoint selection and final evaluation use the EMA weights. All parameter choices were selected using training/validation data only and then frozen for test evaluation. For D variables, hidden width h, graph width d_f, metadata width d_m, K filters, spline basis size R, and sequence length T, a block and time step costs $O(D(h + d_m + 2)d_f + Dd_f^2 + D^2 d_f)$ for graph encoding and pairwise scores, $O(D^3)$ for the symmetric eigendecomposition, $O(KDR)$ for spline-response construction, and $O(D^2 h)$ for each dense state–eigenvector or diffusion multiplication, plus $O(Dh^2)$ for the input projection. Although the routed weight vector has at most s nonzeros, the released implementation constructs all K filter responses and contracts the full $K \times D$ response tensor using a dense `einsum`; response construction therefore remains $O(KDR)$ and response mixing remains $O(KD)$, not $O(sD)$. Top-s sparsity describes the selected weights, but does not reduce these arithmetic costs or the leading $O(D^2 h)$ transforms. In P19, $D = 34$, $T = 72$, $K = 6$, $R = 8$, $s = 3$, and $h = 128$: $D^3 = 39{,}304$, $KDR = 1{,}632$, and $D^2 h = 147{,}968$. Spectral filtering and graph diffusion use three such dense products per step (about $443{,}904$ multiply factors), while the input projection has $Dh^2 = 557{,}056$ factors. The eigensolve is an additional cubic cost whose relative importance grows with D; at $D = 34$, matrix products and learned projections are comparable or larger in these operation counts. The implementation batches graph construction and eigendecomposition over all BT time-variable inputs, so peak graph/eigenvector storage scales as $O(BTD^2)$, in addition to sequence activations. The four-block checkpoint has 195,170 parameters. For seed-7, the original full-dataset run took approximately 60.3 wall-clock hours (about 2.5 days) on a GPU-NVIDIA-RTX A6000 (48GB) machine.

For the controlled comparisons in Table 2, all baselines use the same full P19 $neighbor_v1$ preprocessing, native missingness, train-only normalization, stratified 70/15/15 split, split seed 7, and patient-level test evaluation. Unless stated otherwise, the classification adapters use AdamW with learning rate 5×10^{-4}, weight decay 10^{-4}, batch size 256, a maximum of 24 epochs, patience 6, gradient-norm clipping at 1.0, and class-weighted cross-entropy. The best checkpoint is selected using validation AUROC only; test labels are never used for model selection or hyperparameter choice. Each controlled baseline is trained with seeds 7, 8, and 9, and the reported value is the mean and sample standard deviation across these runs. mTAND uses the public $enc_mtan_classif$ implementation with hidden size 32, time embedding size 64, 128 reference points, one attention head, learned time embeddings, and frequency 10. DGM2 uses the audited source architecture with latent dimension 10, classifier hidden dimension 20, an Euler ODE solver with 10 integration points, balanced positive/negative minibatches of size 128, Adam with learning rate 10^{-3}, a ReduceLROnPlateau scheduler, a maximum of 20 epochs, and patience 4; its checkpoint is selected by validation AUROC. NCDE uses a hidden dimension of 32 and a piecewise-linear Euler Neural CDE discretization with value, availability, and elapsed-time channels, together with the common AdamW training protocol above. iTimER and TimeCHEAT are evaluated through their official classifier interfaces. iTimER uses 128 reference points, hidden and time-embedding sizes of 128, one attention head, BERT-style pooling, maximum sequence length 512, dropout 0.1, and temperature 0.05. TimeCHEAT uses 128 reference points, eight patches, latent dimension 128, two graph layers, one graph-attention head, a Transformer dimension of 256, three Transformer layers, eight Transformer heads, and dropout 0.1. Both use early-stop (batch size 64, 30 training epochs, and patience 8 for iTimER; batch size 64, 12 training epochs, and patience 4 for TimeCHEAT), AdamW with learning rate 10^{-3} and weight decay 10^{-4}, cosine learning-rate decay, and validation-AUROC checkpoint

selection. The forecasting baselines are converted into patient-level classification adapters while retaining their published temporal operators. Pyraformer uses a 64-dimensional representation and temporal scales $1, 2, 4$; DLinear uses trend/seasonal linear components with a moving-average kernel of 25; TimesNet uses two 3×3 temporal-channel convolutions with 32 channels; iTransformer uses variable tokens, a 64-dimensional representation, two four-head Transformer layers, and dropout 0.1; Koopa uses a 64-dimensional periodic/slow-dynamics decomposition and retains the lowest one-sixth of temporal frequencies; FreTS uses a 48-dimensional frequency-domain representation with separate real and imaginary linear mappings; TiDE uses a dense encoder with widths $256, 128, 64$ and a 64-dimensional residual branch; FITS uses a 64-dimensional frequency-domain gain and bias parameterization; TimeMixer uses 64-dimensional multiscale mixing at scales $1, 2, 4$; and RLinear applies reversible normalization followed by a 64-dimensional linear classifier. These adapters use the common AdamW, class-weighted loss, early-stopping, and validation-only checkpoint-selection protocol described above. All parameter choices were selected using training/validation data only and then frozen for test evaluation.

All codes (including our method and baselines) are included in the Supplementary Material (iclr2027code.zip).

AI USE STATEMENT

We used generative AI only for grammar checking and language polishing.

REFERENCES

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. arXiv:2312.00752, 2023.

A. Gu, K. Goel, and C. Re. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2022.

T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. In *NeurIPS*, 2020.

A. Das et al. Long-term forecasting with TiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023.

S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. Zheng, Y. Zhang, and H. Yu. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *ICLR*, 2022.

Y. Liu et al. Koopa: Learning non-stationary time series dynamics with Koopman predictors. In *NeurIPS*, 2023.

Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. iTransformer: Inverted transformers are effective for time series forecasting. In *ICLR*, 2024.

J. Liu, M. Cao, and S. Chen. TimeCHEAT: A channel harmony strategy for irregularly sampled multivariate time series analysis. In *AAAI*, 2025.

J. Liu, M. Cao, and S. Chen. Beyond observations: Reconstruction error-guided irregularly sampled time series representation learning. In *AAAI*, 2026.

X. Qin, M. Liu, W. Wang, S. Li, T. Li, X. Liu, and X. Cheng. Beyond missing data imputation: Information-theoretic coupling of missingness and class imbalance for optimal irregular time series classification. In *AAAI*, 2026.

S. N. Shukla and B. M. Marlin. Multi-time attention networks for irregularly sampled time series. In *ICLR*, 2021.

W. Wang et al. TimeMixer: Decomposable multiscale mixing for time series forecasting. In *ICLR*, 2024.

Y. Wu et al. Dynamic Gaussian mixture based deep generative model for robust forecasting on sparse multivariate time series. In *AAAI*, 2021.

H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *ICLR*, 2023.

M. Xu et al. FITS: Modeling time series with 10k parameters. In *ICLR*, 2024.

K. Yi et al. Frequency-domain MLPs are more effective learners for time series forecasting. In *NeurIPS*, 2023.

X. Zhang, X. Zeman, T. Tsiligkaridis, and M. Zitnik. Graph-guided network for irregularly sampled multivariate time series. In *ICLR*, 2022.

A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? In *AAAI*, 2023.

Z. Li et al. RLinear: A simple yet effective baseline for time series forecasting. In *AAAI*, 2024.