TIMESWARM: SOFT WEIGHTED ASSIGNMENT-BASED RELATIONAL MODELING OF CHANNEL-PATCHES FOR MULTIVARIATE TIME SERIES CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multivariate time series classification (MTSC) requires jointly modeling discriminative temporal patterns and dependencies across variables or channels. Existing methods typically process channels independently, represent interactions only at the channel level, or route an entire multivariate patch as a single unit, thereby overlooking heterogeneous local dynamics across channels and temporal regions. We address this limitation by coupling local representation specialization with cross-channel relational learning at the channel-patch level. To this end, we propose TimeSWARM, a mixture-of-experts framework for dynamic channel-patch specialization and relational modeling. Each channel is partitioned into local patches, and an input-conditioned router combines temporal and spectral descriptors to produce expert assignments for each channel-patch. Sparsified routing weights are used to aggregate multi-scale temporal experts, enabling specialization across channels, temporal regions, and samples. The resulting dense routing distributions are reused as soft latent-group assignments for cross-channel interaction, so that the same routing mechanism determines both how each local pattern is represented and which channels it communicates with. Extensive experiments on the UEA multivariate time series archive show that TimeSWARM achieves the best average accuracy and average rank among the compared methods, together with the highest number of dataset-level top-1 results.

1 INTRODUCTION

Multivariate time series classification (MTSC) aims to learn discriminative patterns from multiple temporally evolving variables or channels and underpins applications such as human activity recognition, physiological signal analysis, industrial monitoring, Internet of Things (IoT) sensing, and audio understanding (Bagnall et al., 2018; Ismail Fawaz et al., 2019). In contrast to univariate classification, MTSC considers not only temporal dynamics within individual channels, but also dependencies among channels. An effective MTSC model should therefore determine both *which local temporal patterns are informative* and *how information should be exchanged across channels*.

Existing methods address these two requirements through different modeling paradigms, as illustrated in Figure 1. Channel-independent approaches (Nie et al., 2023; Zhou et al., 2023) process channels independently which avoids potentially harmful interference between weakly related channels. In contrast, channel-dependent approaches jointly model multiple channels through feature correlation, attention, or shared latent representations (Wang et al., 2024; Zhang & Yan, 2023; Liu et al., 2024; Li et al., 2025b; Wang et al., 2025). Other methods introduce structured channel grouping to restrict cross-channel interaction to selected subsets of channels (Ying et al., 2025). Meanwhile, shapelet- and periodic-patching approaches (Le et al., 2024; Mu et al., 2025) demonstrate the importance of localized temporal structures for discriminative MTSC representations.

Recent studies have further recognized that different channels may play heterogeneous roles in multivariate systems and have introduced structured or role-aware mechanisms for selective cross-channel interaction (Ying et al., 2025; Hu et al., 2026). However, such structural assignments are

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.

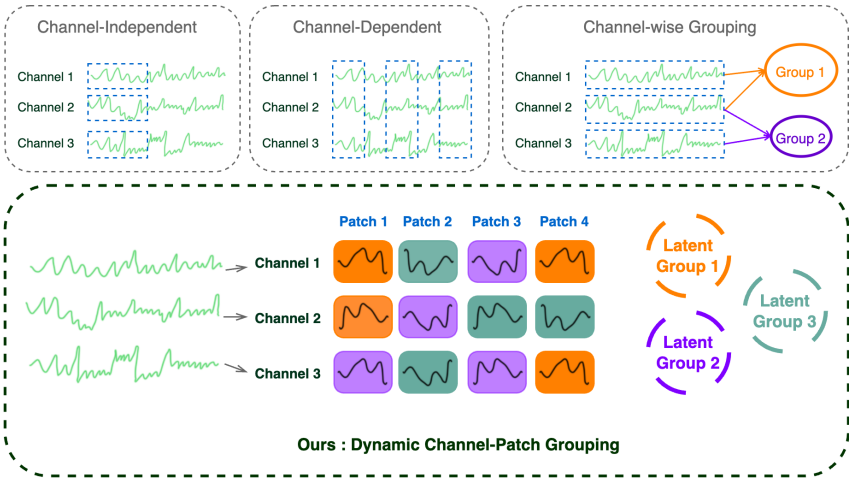


Figure 1: Comparison of representative multivariate time series modeling paradigms. Channel-independent methods process channels separately, channel-dependent methods globally correlate channels, and channel-level grouping methods assign relationships at the granularity of entire channels. In contrast, TimeSWARM first maps individual channel-patches to their corresponding expert or latent-group embeddings through a mixture-of-experts (MoE) module, and then mixes these embeddings across channels using the cross-channel, cross-expert probability structure induced by the input-dependent router softmax. This design allows each channel-patch's expert preference and latent-group affiliation to vary across channels, temporal regions, and input samples.

commonly defined at the granularity of entire channels. In practice, the role and relevance of a channel may vary across local temporal regions, motivating a finer channel-patch-level representation. Mixture-of-experts (MoE) architectures provide a natural mechanism for adapting computation to heterogeneous local patterns (Shazeer et al., 2017). Recent MTSC approaches have explored expert routing over temporally localized representations or fine-grained time series tokens, while other methods explicitly learn patch-specific channel dependencies (Wen et al., 2025; Tao et al., 2026; Guo et al., 2026; Li et al., 2026). However, these approaches typically use expert assignments for representation specialization or construct relational structures through a separate similarity, graph, or dependency-learning mechanism. Consequently, they do not use a shared channel-patch-level assignment to jointly adapt how a local pattern is represented and with which channels it interacts as its temporal dynamics change.

In this context, we propose TimeSWARM (**S**oft **W**eighted **A**ssignment-based **R**elational **M**odeling), which routes each channel–patch to specialized multi-scale temporal experts through MoE routing. The resulting dense routing vector describes the patch's relative preferences over the learned temporal experts and is interpreted as a soft latent-group assignment. Channel–patches with similar routing profiles exhibit similar preferences over these learned transformations and are therefore treated by the model as belonging to similar latent specialization groups. This grouping is model-induced and does not assume a predefined semantic relationship between the underlying channels. Rather than introducing an additional graph learner or relation-specific router, TimeSWARM reuses these latent assignments to mediate sample and patch-specific cross-channel interaction. A sparsified version of the routing distribution is used for selective expert aggregation to generate channel-patch MoE embedding, whereas the original dense distribution across channels and experts is retained for relational modeling. This yields a compact channel–latent-group–channel interaction mechanism in the same assignment space used for expert specialization, avoiding a separately constructed channel graph or clustering module. The resulting channel-patch representations are further contextualized across successive temporal patches to capture inter-patch temporal dependencies before classification. The key contributions of this work are summarized as follows:

- **Channel-patch conditional expert specialization.** We introduce a conditional MoE representation mechanism that independently routes each channel-patch using local temporal–spectral characteristics. Independently parameterized multi-scale temporal experts provide

candidate transformations, allowing the representation function to adapt across channels, temporal regions, and input samples.

- **Routing-coupled relational modeling.** We further reuse the dense expert-preference distributions learned by the channel-patch MoE as soft latent-group assignments for cross-channel interaction. This couples relational refinement to the same specialization space used for local representation learning, enabling sample and patch-specific information exchange without introducing a separate graph-construction or relation-specific grouping mechanism, before the refined channel-patch representations are passed to the inter-patch contextualization module.
- **Comprehensive evaluation and analysis.** We evaluate TimeSWARM on the complete UEA multivariate time series classification archive against state-of-the-art MTSC methods. Ablations and routing analyses examine the complementary contributions of channel-patch expert specialization, routing-coupled cross-channel interaction, and inter-patch temporal contextualization.

2 Related Work

Multivariate time series classification (MTSC) requires capturing both discriminative temporal patterns and dependencies among input channels. Convolutional approaches remain effective for extracting local temporal structure, while recent methods increasingly employ multi-scale modeling to capture patterns with different temporal extents (Dempster et al., 2021; Le et al., 2024; Donghao & Xue, 2024; Mu et al., 2025). In parallel, channel-aware architectures seek to model interactions among heterogeneous input channels rather than processing channels independently or combining them uniformly. Representative approaches include Crossformer (Zhang & Yan, 2023) and SGN (Ying et al., 2025), with the latter combining structured k-means-based channel grouping with multi-scale temporal modeling. These developments highlight the importance of jointly capturing heterogeneous temporal patterns and selective cross-channel interactions.

Mixture-of-experts (MoE) provides a complementary mechanism for handling such heterogeneity by conditionally assigning inputs to specialized computational functions (Shazeer et al., 2017). MoE has subsequently been adopted in time-series classification at different routing granularities, including sample-level and temporally localized representations (Wen et al., 2025; Tao et al., 2026). Such conditional specialization is particularly appealing for MTSC because different local patterns may benefit from different feature transformations.

Despite these advances, channel-specific temporal specialization and cross-channel relational modeling are generally designed as separate mechanisms in MTSC. Channel-grouping approaches determine cross-channel relations independently of the local representation transformations applied to individual temporal patterns, whereas MoE-based MTSC methods primarily use routing for conditional expert specialization. Consequently, local specialization and relational interaction remain largely decoupled. In contrast, TimeSWARM performs conditional specialization at the channel-patch level and reuses the resulting dense expert-assignment distribution as a soft latent grouping representation for cross-channel interaction. This couples local temporal specialization with cross-channel relational modeling without requiring a separate graph-construction or clustering mechanism.

3 Method

Let $\mathbf{X} \in \mathbb{R}^{L \times C}$ denote a multivariate time series of length L with C channels, and let $y \in \{1, \ldots, K\}$ denote its class label. We partition each channel into W local temporal patches of length P and denote the patch from channel c at temporal location w by $\mathbf{X}_{c,w} \in \mathbb{R}^P$. Our central premise is that local subsequences within multivariate time series data may exhibit heterogeneous temporal characteristics and therefore need not benefit from a uniformly shared representation module. At the same time, local patterns from different channels or variables may exhibit similar or overlapping temporal semantics and can benefit from selective interaction. A central difficulty is that a channel need not have a fixed relational role throughout a sequence. Which other channels are informative may change with the local temporal regime. Moreover, two channel-patches with different raw values may still require similar temporal transformations. This suggests using the

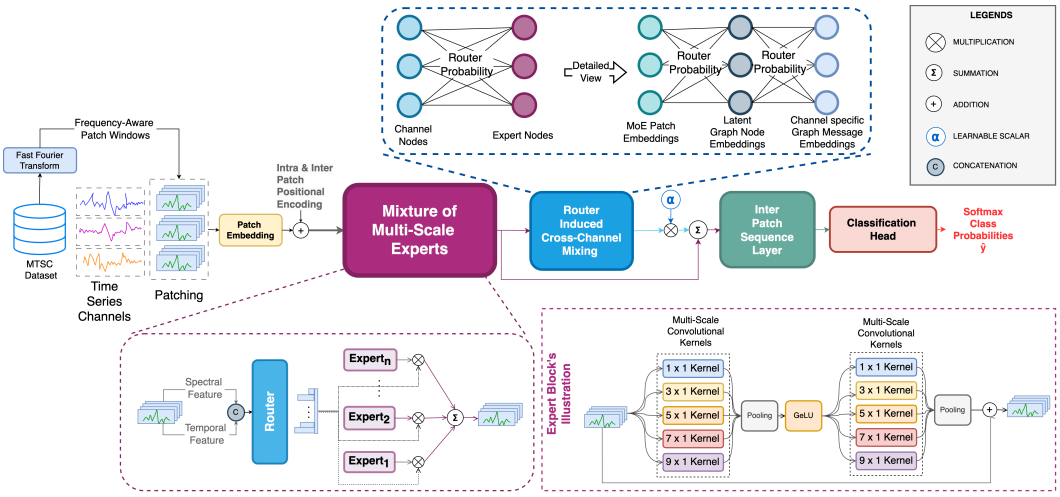


Figure 2: Overview of TimeSWARM. Detailed architectural and implementation configurations are provided in Appendix A.6.

computation preferred by a local pattern as a task-dependent cue to its functional role. We therefore couple *conditional local specialization* and *cross-channel relational modeling* through a shared routing mechanism. As illustrated in Figure 2, every channel-patch is independently analyzed by a mixture of multi-scale temporal experts. A patch-wise router computes the relevance of each expert to the local pattern. Unlike conventional expert routing, we additionally interpret the dense routing distribution as a soft assignment to latent temporal groups and reuse it to induce cross-channel interaction. Thus, the same local semantics determine both *how* a channel-patch is represented and *with which* channel-patches it shares information. The next subsections delineates the major technical parts of the proposed method, additional details about the architectural design can be found in the Appendix A.6.1.

3.1 Patch-Wise Expert Specialization and Latent Group Assignment

Let $\mathbf{H}_{c,w} \in \mathbb{R}^{D\times P}$ denote the embedded representation of channel–patch (c, w). We introduce E independently parameterized temporal experts $\{\mathcal{E}_e\}_{e=1}^{E}$, where each expert employs multi-scale 1D convolutions with different temporal receptive fields. The experts therefore provide multiple candidate transformations for heterogeneous local temporal patterns.

For each channel–patch, a lightweight router extracts a compact descriptor $\psi(\mathbf{H}_{c,w})$ and produces a dense routing distribution

$$\mathbf{R}_{c,w} = \mathrm{softmax}\left(\frac{g_r(\psi(\mathbf{H}_{c,w}))}{\tau}\right) \in \mathbb{R}^{E}, \tag{1}$$

where $g_r(\cdot)$ denotes the router and τ is the routing temperature. Routing is independently conditioned on every sample, channel, and temporal patch; hence, different regions of the same channel may exhibit different expert preferences. Beyond determining expert specialization, we interpret $\mathbf{R}_{c,w}$ as a soft latent group assignment over the E experts. Channel–patches with similar routing profiles exhibit compatible preferences over the learned local-pattern transformations and are therefore treated by the model as belonging to similar latent functional groups. This grouping is model-induced and does not assume a predefined semantic relationship between the underlying channels or variables. This provides a data-dependent grouping mechanism at the channel–patch level, rather than imposing a fixed channel partition across the entire sequence. Viewed this way, $\mathbf{R}_{c,w}$ is more than a dispatch decision. It provides a compact, task-learned profile of which temporal transformations the local pattern calls for. For expert aggregation, we apply the top-1-preserving adaptive sparsification described in Appendix A.6.3. The resulting weights $\widetilde{\mathbf{R}}_{c,w}$ preserve the dominant expert affiliation while allowing additional experts to contribute for less decisive routing distributions.

The MoE representation is then computed as

$$\mathbf{H}_{c,w}^{\text{moe}} = \text{LN}\left(\mathbf{H}_{c,w} + \sum_{e=1}^{E} \widetilde{R}_{c,w,e}\mathcal{E}_e(\mathbf{H}_{c,w})\right). \tag{2}$$

Sparsification is used only for expert aggregation. We retain the original dense routing distribution $\mathbf{R}_{c,w}$ from the final MoE layer. Consequently, $\widetilde{\mathbf{R}}_{c,w}$ encourages a dominant expert specialization for representation learning, whereas $\mathbf{R}_{c,w}$ preserves the complete soft group-affiliation structure used to infer cross-channel relationships. This separation allows expert specialization to remain sparse without discarding the graded compatibility information required for subsequent relational interaction. Further details on patch construction, input embedding, and routing are provided in Appendices A.6.1, A.6.2, and A.6.3.

3.2 Multi-Scale Temporal Experts

Multi-scale convolutional architectures have shown strong effectiveness in time series modeling (Wu et al., 2023; Ismail Fawaz et al., 2020), as these can capture temporal patterns over multiple receptive fields. In this context, we implement each expert using multi-scale one-dimensional convolutions, enabling it to capture local temporal shapes at different scales. This is particularly useful for heterogeneous multivariate time series, where discriminative patterns may occur over substantially different temporal durations. Each MoE block contains E independently parameterized temporal experts. Given $\mathbf{H} \in \mathbb{R}^{D\times P}$, a multi-scale convolutional layers block uses the odd-valued kernel set

$$\mathcal{K} = \{1, 3, 5, \ldots, 2Q-1\}, \tag{3}$$

where Q is the number of convolutional branches. Branch outputs are averaged rather than concatenated:

$$\mathcal{I}(\mathbf{H}) = \frac{1}{Q}\sum_{q=1}^{Q} \text{Conv}_{2q-1}(\mathbf{H}). \tag{4}$$

Each expert contains two multi-scale blocks with a GELU non-linearity:

$$\mathcal{E}_e(\mathbf{H}) = \text{BN}\left[\mathcal{I}_{e,2}\left(\text{GELU}\left(\mathcal{I}_{e,1}(\mathbf{H})\right)\right)\right] + \mathbf{H}. \tag{5}$$

The residual path preserves the original patch representation while each expert learns a distinct multi-scale transformation.To avoid dataset-specific tuning of the number of experts, we determine it automatically from the channel dimensionality as $E = \min(10, \max(3, 2\lfloor \log_2 C \rfloor))$, where C denotes the number of input channels. The logarithmic growth (Dempster et al., 2021) allows the expert pool to increase for higher-dimensional datasets while preventing the number of experts, and consequently the computational cost, from growing linearly with the channel count. The lower bound ensures a minimum level of expert diversity for low-dimensional datasets, whereas the upper bound keeps the expert pool manageable for datasets containing a large number of channels.

3.3 Cross-Channel Interaction

One of the major components of our proposed method is to reuse the routing probabilities in equation 1 as the relational structure for cross-channel interaction. For a fixed sample and patch location w, the dense routing distributions of all C channels are collected into $\mathbf{R}_w \in \mathbb{R}^{C\times E}$, where each row represents the soft expert-group membership of a channel-patch and each column corresponds to a latent temporal group associated with an expert. Since $\mathbf{R}_w$ is recomputed from the input at every patch location, the resulting relational structure is both sample-dependent and temporally local. Let $\mathbf{H}_w$ denote the corresponding MoE representations. We first aggregate channel-patches into the E latent groups:

$$\mathbf{D}_w = \text{diag}\left(\mathbf{R}_w^\top \mathbf{1}\right), \qquad \mathbf{U}_w = \mathbf{D}_w^{-1}\mathbf{R}_w^\top \mathbf{H}_w, \tag{6}$$

where $\mathbf{D}_w$ normalizes the routing mass received by each latent group. The group representations are then propagated back to the channels using the same routing assignments:

$$\mathbf{M}_w = \mathbf{R}_w \mathbf{U}_w. \tag{7}$$

Combining equation 6 and equation 7 gives

$$\mathbf{M}_w = \underbrace{\mathbf{R}_w \mathbf{D}_w^{-1} \mathbf{R}_w^\top}_{\mathbf{A}_w} \mathbf{H}_w, \tag{8}$$

where $\mathbf{A}_w \in \mathbb{R}^{C \times C}$ denotes the router-induced relational weights between channels at patch location w. This formulation provides a *channel → latent group → channel* bipartite factorization of the relational operation. This factorization also constrains the induced channel relations. A_w is symmetric, positive semi-definite, row-stochastic, and has rank at most E. It therefore acts as a low-rank soft averaging operator over latent functional groups, reducing to within-group averaging when the assignments become hard. Rather than learning an additional pairwise channel graph, TimeSWARM reuses the dense MoE assignments as latent group coordinates for cross-channel interaction. Moreover, $\mathbf{A}_w$ need not be explicitly materialized; message passing is performed in the E-dimensional latent group space. The relational message operates on the complete $D \times P$ patch representation rather than on a temporally pooled token. Therefore, the router determines *which* channel-patches communicate, while their position-preserving representations determine *what* information is exchanged. Finally, the relational feature representation is incorporated through a gated residual update

$$\overline{\mathbf{H}}_w = \mathbf{H}_w^{\text{moe}} + \sigma(\gamma)\phi_m(\mathbf{M}_w), \tag{9}$$

where ϕ_m is a shared feature projection and γ is a learnable scalar. The routing probabilities remain differentiable through the relational branch, allowing the assignments to be influenced by both expert specialization and the downstream relational objective. A detailed description of this module can be found in Appendix A.6.4.

3.4 Inter-Patch Contextualization

The preceding modules specialize local temporal patterns within each patch and model cross-channel interactions at a particular patch location. However, processing channel–patches independently does not explicitly capture how local patterns evolve across successive temporal regions. Therefore, we introduce a lightweight inter-patch module that operates along the patch axis while preserving individual channel identities. Let $\overline{\mathbf{H}}_{c,w} \in \mathbb{R}^{D \times P}$ denote the cross-channel refined representation of channel c at patch w. We first summarize its within-patch content as

$$\mathbf{q}_{c,w} = \frac{1}{P} \sum_{p=1}^{P} \overline{\mathbf{H}}_{c,w,:,p} \in \mathbb{R}^D. \tag{10}$$

For each channel independently, the resulting sequence $\mathbf{Q}_c = [\mathbf{q}_{c,1}, \ldots, \mathbf{q}_{c,W}]$ is processed along the patch dimension by a lightweight residual temporal convolutional block,

$$\widehat{\mathbf{Q}}_c = \text{LN}\left(\mathbf{Q}_c + \phi_{\text{inter}}(\mathbf{Q}_c)\right), \tag{11}$$

where $\phi_{\text{inter}}(\cdot)$ denotes the inter-patch convolutional transformation. The resulting contextual representation for each patch is broadcast over its within-patch positions and added back to $\overline{\mathbf{H}}_{c,w}$. In this way, local position-preserving features are retained while each patch is contextualized by neighboring temporal regions of the same channel. Finally, the contextualized patches are mapped back to the original temporal axis using overlap-aware reconstruction, where representations from overlapping windows are averaged. The reconstructed channel representations are concatenated and processed by a lightweight temporal prediction head, followed by global average and maximum pooling and a linear classifier. Thus, TimeSWARM captures complementary structure at three levels: (i) local

Table 1: Comprehensive performance comparison of TimeSWARM with recent state-of-the-art MTSC baselines on all 30 datasets in the UEA archive. A few baseline results are unavailable because of excessive execution times on large datasets or high model complexity.

Dataset	M.FCN	TapNet	W.MUSE	Shape Net	SVPT	Shape former	MPTS Net	SGN	**Ours**
ArticularyWordRecognition	97.3	98.7	99	98.7	99.3	99	97.7	99	99.3
AtrialFibrillation	26.7	33.3	33.3	40	40	53.3	53.3	66.7	66.7
BasicMotions	95	100	100	100	100	100	100	100	100
CharacterTrajectories	98.5	99.7	99	98	99	99.2		99.6	99.6
Cricket	91.7	95.8	100	98.6	100	94.4	94.4	100	100
DuckDuckGeese	67.5	57.5	57.5	72.5	70	64	68	64	76
EigenWorms	50.4	48.9	89	87.8	92.5	92.5		85.5	80.9
Epilepsy	76.1	97.1	100	98.7	98.6	98.6	97.1	97.8	99.3
ERing	13.3	13.3	43	13.3	93.7	87.4	94.4	95.9	96.3
EthanolConcentration	37.3	32.3	13.3	31.2	33.1	41.1	43.3	44.5	39.9
FaceDetection	54.5	55.6	54.5	60.2	51.2	65.8	69.8	70.3	69.5
FingerMovements	58	53	49	58.9	60	55	64	64	65
HandMovementDirection	36.5	37.8	36.5	33.8	39.2	41.9	63.5	75.7	66.2
Handwriting	28.6	35.7	60.5	45.1	43.3	30.2	34.4	50.4	61.5
Heartbeat	66.3	75.1	72.7	75.6	79	81.5	75.6	77.1	82.4
InsectWingbeat	16.7	20.8		25	18.4	31.4		68	74.1
JapaneseVowels	97.6	96.5	97.3	98.4	97.8	99.2	98.6	98.9	98.9
Libras	85.6	85	87.8	85.6	88.3	95.5	87.2	83.9	95
LSST	37.3	56.8	59	59	66.6	63.8	60.4	63.7	49.2
MotorImagery	51	59	50	61	65	67	65	65	63
NATOPS	88.9	93.9	87	88.3	90.6	96.1	94.4	98.3	96.7
PEMS-SF	69.9	75.1		75.1	86.7	92.5	94.2	88.4	97.1
PenDigits	97.8	98	94.8	97.7	98.3	99.1	98.9	99.1	99.1
PhonemeSpectra	11	17.5	19	29.8	17.6	29.3	14.4	23.1	31.4
RacketSports	80.3	86.8	93.4	88.2	84.2	88.8	87.5	93.4	94.1
SelfRegulationSCP1	87.4	65.2	71	78.2	88.4	91.8	92.8	93.9	92.8
SelfRegulationSCP2	47.2	55	46	57.8	60	56.1	57.2	60.6	62.2
SpokenArabicDigits	99	98.3	98.2	97.5	98.6	99.7	99.5	99.7	100
StandWalkJump	6.7	40	33.3	53.3	46.7	66.7	53.3	53.3	73.3
UWaveGestureLibrary	89.1	89.4	91.6	90.6	94.1	90	88.1	92.2	93.1
Avg. Acc (UEA-30)	62.1	65.7	69.1	69.9	73.3	75.7	75.8	79.1	**80.8**
Avg. Acc (UEA-10)	67.7	67.8	67.2	71.0	73.2	74.8	75.4	77.6	**79.7**
Avg. Acc (UEA-25)	60.0	64.3	64.7	67.6	71.8	74.0	74.0	76.8	**78.8**
Avg. Rank (UEA-30)	7.6	6.5	5.5	5.3	4.2	3.5	3.9	2.7	**2.0**
Avg. Rank (UEA-10)	7.2	7.1	6.3	5.8	4.9	3.9	4.3	2.3	**1.8**
Avg. Rank (UEA-25)	7.8	6.6	5.6	5.2	4.1	3.7	4.4	2.7	**1.9**
Win	30	27	25	27	23	22	24	16	-
Draw	0	1	2	1	3	2	2	6	-
Lose	0	2	3	2	4	6	4	8	-
Top-1 (UEA-30)	0	2	3	1	6	6	1	9	**17**
Top-1 (UEA-10)	0	0	0	0	1	1	0	3	**5**
Top-1 (UEA-25)	0	1	3	1	5	4	1	9	**15**

temporal specialization within patches, (ii) latent-group-mediated interaction across channels, and (iii) temporal contextualization across successive patches.

4 EXPERIMENTS

We evaluate our proposed method on the UEA Multivariate Time Series Classification Archive (Bagnall et al., 2018), which contains 30 datasets spanning diverse application domains and data characteristics. Following recent MTSC studies (Ying et al., 2025; Mu et al., 2025; Zhou et al., 2023), we adopt the standard TSLib-based evaluation protocol introduced with TimesNet (Wu et al., 2023). In particular, we majorly follow the data pre-processing steps and experimental design settings of SGN (Ying et al., 2025), a closely related k-means clustering based channel-grouping method, to maintain consistency with previously reported baseline MTSC results. We compare TimeSWARM against a broad set of recent state-of-the-art MTSC baselines, including SGN (Ying et al., 2025), MPTSNet (Mu et al., 2025), TimeMixer++ (Wang et al., 2025), ShapeNet (Li et al., 2021), SVPT (Zuo et al., 2023), and MiniROCKET (Dempster et al., 2021). The selected baselines cover a broad range of MTSC modeling paradigms, including convolutional feature extraction,

Transformer-based temporal modeling, MLP-based, shape-aware representation learning, structured channel interaction, and randomized convolutional transforms. All models are trained and evaluated on a High Performance Computing (HPC) cluster equipped with NVIDIA A100 GPUs, each with 40GB of VRAM. Detailed dataset statistics, implementation settings, and baseline descriptions are provided in Appendices A.1, A.2, and A.3, respectively.

4.1 Main Results

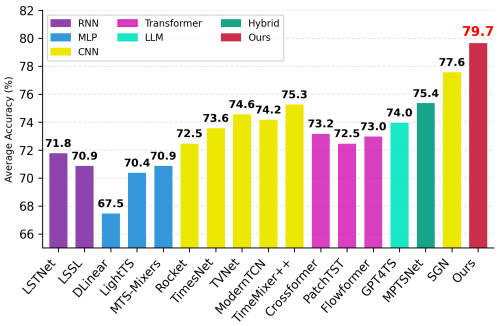


Figure 3: Average classification accuracy across the 10 core UEA MTSC benchmarks.

Figure 3 delineates the average accuracy of our proposed method and other state-of-the-art baselines on UEA-10, a commonly used 10-dataset subset of the full UEA-30 archive (see Appendix A.1 for dataset details). As illustrated, our method achieves the highest average accuracy of 79.7%, outperforming the strongest competing method, SGN, by 2.1% and the recent hybrid baseline MPTSNet by 4.3%. Notably, our method also surpasses strong CNN-based models such as TimeMixer++ (75.3%) and TVNet (74.6%), transformer-based models such as Crossformer (73.2%), PatchTST (72.5%), and Flowformer (73.0%), as well as other state-of-the-art RNN, MLP, and LLM-based baselines. These results show that the advantage of our method on the core UEA-10 datasets is not restricted to comparison against a specific architectural family, but is instead consistently maintained across a diverse range of recent MTSC approaches.

Furthermore, Table 1 provides a comprehensive comparison of TimeSWARM with recent MTSC baselines on the full 30-dataset UEA archive. TimeSWARM achieves the highest average accuracy of 80.8%, compared with 79.1% for SGN, the strongest competing baseline, and 75.8% and 75.7% for MPTSNet and ShapeFormer, respectively. It also obtains the best average rank of 2.0 on UEA-30, compared with 2.7 for SGN, 3.5 for ShapeFormer, and 3.9 for MPTSNet. Moreover, TimeSWARM records 17 dataset-level top-1 results on UEA-30, compared with 9 for SGN and 6 for ShapeFormer, and achieves 5 top-1 results on the commonly evaluated UEA-10 subset. On the UEA-25 subset, TimeSWARM maintains the highest average accuracy (78.8%) and best average rank (1.9), indicating that the aggregate performance advantage is preserved when restricting the comparison to datasets with broadly available baseline results. Baseline results are taken from the comprehensive comparison reported by Ying et al. (2025), while the results of TimeSWARM are obtained from our experiments following the same benchmark protocol. Overall, TimeSWARM achieves the highest average accuracy, best average rank, and the largest number of dataset-level top-1 results on UEA-30, UEA-10 and UEA-25 evaluations. Pairwise Wilcoxon signed-rank tests Bagnall et al. (2018); Le et al. (2024) further show statistically significant improvements over seven of the eight competing state-of-the-art baselines from Table 1, including MPTSNet and ShapeFormer.

Table 2: Ablation study of the major components of TimeSWARM. Δ denotes the performance degradation relative to the full model.

Variant	UEA-30	Δ	UEA-10	Δ
Proposed Method	**80.8**	–	**79.7**	–
w/o Cross-Channel Interaction	77.1	↓ 3.7	77.7	↓ 2.0
w/o Channel-Patch Expert Specialization	74.7	↓ 6.1	74.6	↓ 5.1
w/o Inter-Patch Contextualization	75.8	↓ 5.0	76.2	↓ 3.5

4.2 Ablation Study

We evaluate the contribution of the major components of TimeSWARM on both the complete UEA-30 benchmark and the UEA-10 subset. As shown in Table 2, removing any of the three major components decreases average classification accuracy on both benchmarks, supporting their complementary contributions. The largest degradation occurs when removing channel–patch expert

specialization, reducing average accuracy from 80.8% to 74.7% on UEA-30 and from 79.7% to 74.6% on UEA-10. These results highlights the importance of conditional expert specialization for representing heterogeneous local temporal patterns. Removing the routing-induced cross-channel interaction reduces average accuracy by 3.7 points on UEA-30 and 2.0 points on UEA-10. In this variant, local expert specialization is retained, while the relational refinement induced by the dense MoE routing assignments is removed. The resulting degradation shows that these assignments provide useful relational structure beyond their role in expert aggregation. Removing the inter-patch contextualization module also causes substantial drops of 5.0 and 3.5 points, respectively, showing the benefit of contextualizing successive local patches within each channel. Overall, channel–patch expert specialization serves as the primary local representation mechanism, while its dense routing assignments are additionally reused as a compact latent-group representation for cross-channel interaction. This reuse avoids introducing a separate graph-construction or relation-specific assignment module. Together, these results support the complementary roles of channel–patch expert specialization, latent-group-mediated cross-channel interaction, and inter-patch temporal contextualization in TimeSWARM.

4.3 Visualization Analysis of Routing Behavior

Figure 4 visualizes the patch-wise routing behavior of TimeSWARM for two test instances from the *HandMovementDirection* dataset. The routing patterns vary across both channels and temporal patches, and also differ between the two samples, confirming that expert assignment is channel–patch specific and input-dependent. While some experts dominate broader temporal regions, others are preferred only by particular local patterns. These dominant-assignment maps illustrate the router's specialization behavior, whereas the relational module retains the complete dense routing profiles as soft latent-group assignments that are subsequently reused for cross-channel interaction. Additional visualization analyses are provided in Appendix A.5.

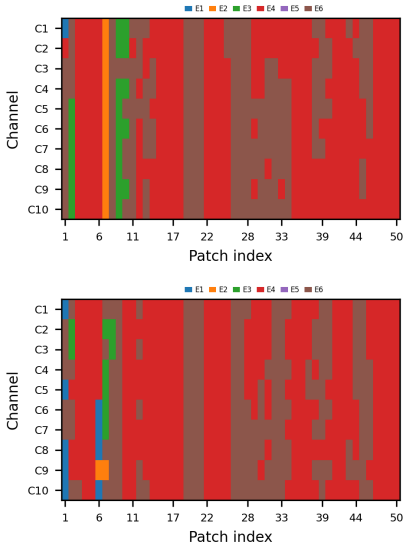


Figure 4: Visualization of dominant expert assignments for two test samples from the *HandMovementDirection* dataset. Different colors represent different dominant experts.

5 Conclusion

We introduced TimeSWARM, a multivariate time series classification framework that performs conditional temporal specialization at the channel–patch level and reuses the resulting routing assignments for cross-channel relational modeling. Each channel–patch is conditionally processed by multi-scale temporal experts, while the corresponding dense expert-preference distribution provides a soft latent group assignment through which information can be exchanged across channels. This enables sample, channel and patch-specific relational interaction without requiring a separately learned channel graph or clustering mechanism. Inter-patch contextualization further captures dependencies across successive local temporal regions. Experiments on the UEA archive and component-wise ablations demonstrate the complementary contributions of channel–patch expert specialization, latent-group-mediated cross-channel interaction, and inter-patch temporal contextualization. More broadly, this suggests a view of MoE routing beyond expert dispatch: routing assignments can provide a compact, input-dependent latent coordinate system through which local representation specialization can be reused to structure relational interaction.

Reproducibility Statement

We follow the standard dataset splits and evaluation protocol used in previous state-of-the-art MTSC studies (Wu et al., 2023; Ying et al., 2025; Mu et al., 2025). Full implementation and training details, including random seeds, are provided in the appendix. Source code and configura-

tions have been released to facilitate reproducibility. Source code repository link (anonymous): https://anonymous.4open.science/r/MTSC-ECE5/

AI USE STATEMENT

Generative AI tools were used solely to assist with language editing, writing refinement, and initial drafting of selected parts of the manuscript. All AI-assisted text was subsequently reviewed, revised, and customized by the authors, who take full responsibility for the final content. Generative AI was not used to generate any figures, or produce the core scientific ideas and contributions of this work. The research design, methodological development, experimental analysis, and conclusions are the authors' original work.

REFERENCES

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, pp. 3–4, 2013.

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels: A. dempster et al. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 248–257, 2021.

Luo Donghao and Wang Xue. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022.

Yunhe Guo, Zhengyang Zhou, Kedong Yan, and Yang Wang. Tvsrnet: A unified framework for multivariate time-series classification via triple-view sparsified routing. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 573–589. Springer, 2026.

Shiyan Hu, Tengxue Zhang, Jianxin Jin, Xiangfei Qiu, Bin Yang, and Chenjuan Guo. Teamwork: Multivariate time series anomaly detection via asymmetric role-aware channel modeling. In *Forty-third International Conference on Machine Learning*, 2026.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review: H. ismail fawaz et al. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data mining and knowledge discovery*, 34(6):1936–1962, 2020.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural networks*, 116:237–245, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The Twelfth International Conference on Learning Representations*, 2015.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Xuan-May Le, Ling Luo, Uwe Aickelin, and Minh-Tuan Tran. Shapeformer: Shapelet transformer for multivariate time series classification. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1484–1494, 2024.

Chenghan Li, Mingchen Li, and Ruisheng Diao. Tvnet: a novel time series analysis method based on dynamic convolution and 3d-variation. In *International Conference on Learning Representations*, volume 2025, pp. 35851–35889, 2025a.

Dongyuan Li, Shun Zheng, Chang Xu, Jiang Bian, and Renhe Jiang. Routing channel-patch dependencies in time series forecasting with graph spectral decomposition. In *The Fourteenth International Conference on Learning Representations*, 2026.

Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace Lai-Hung Wong. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8375–8383, 2021.

Zhe Li, Xuanxuan Li, Zhongwen Rao, Lujia Pan, and Zenglin Xu. Mts-mixers: Multivariate time series forecasting via factorized temporal and channel mixing. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2025b.

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.

Yang Mu, Muhammad Shahzad, and Xiao Xiang Zhu. Mptsnet: Integrating multiscale periodic local patterns and global dependencies for multivariate time series classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 19572–19580, 2025.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Patrick Schäfer and Ulf Leser. Multivariate time series classification with weasel+muse, 2018. URL `https://arxiv.org/abs/1711.11343`.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *The Twelfth International Conference on Learning Representations*, 2017.

Xie Tao, Tan Zexi, Xiao Haoyi, Li Mengke, Zhang Yiqun, Lu Yang, Yang Cuie, and Cheung Yiu-Ming. Anchormoe: Interpretable time series classification via anchor-routed moe. In *Proceedings of the 32nd ACM SIGKDD conference on knowledge discovery and data mining (KDD)*, 2026.

Shiyu Wang, Jiawei Li, Xiaoming Shi, Zhou Ye, Baichuan Mo, Wenze Lin, Shengtong Ju, Zhixuan Chu, and Ming Jin. Timemixer++: A general time series pattern machine for universal predictive analysis. In *International Conference on Learning Representations*, volume 2025, pp. 16980–17016, 2025.

Yihe Wang, Nan Huang, Taida Li, Yujun Yan, and Xiang Zhang. Medformer: A multi-granularity patching transformer for medical time-series classification. *Advances in Neural Information Processing Systems*, 37:36314–36341, 2024.

Yunshi Wen, Tengfei Ma, Ronny Luss, Debarun Bhattacharjya, Achille Fokoue, and Anak Agung Julius. Shedding light on time series classification using interpretability gated networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Flowformer: Linearizing transformers with conservation flows. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 24226–24242. PMLR, 17–23 Jul 2022.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Zenan Ying, Zhi Zheng, huijun hou, Tong Xu, Qi Liu, Jinke Wang, and Wei Chen. Sgn: Shifted window-based hierarchical variable grouping for multivariate time series classification. In *Advances in Neural Information Processing Systems*, pp. 119820–119851, 01 2025. doi: 10.52202/085713-3609.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26317. URL https://doi.org/10.1609/aaai.v37i9.26317.

Tianping Zhang, Yizhuo Zhang, Wei Cao, Jiang Bian, Xiaohan Yi, Shun Zheng, and Jian Li. Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. *arXiv preprint arXiv:2207.01186*, 2022.

Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 6845–6852, 2020.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

Rundong Zuo, Guozhong Li, Byron Choi, Sourav S Bhowmick, Daphne Ngar-yin Mah, and Grace LH Wong. Svp-t: A shape-level variable-position transformer for multivariate time series classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11497–11505, 2023.

Table 3: A comprehensive summary 30 datasets from UEA multivariate time series data archive

Dataset	Train Samples	Test Samples	Channels	Length	Classes	Domain
ArticularyWordRecognition	275	300	9	144	25	Motion
AtrialFibrillation	15	15	2	640	3	Physiological
BasicMotions	40	40	6	100	4	Human Activity
CharacterTrajectories	1422	1436	3	182	20	Motion
Cricket	108	72	6	1197	12	Human Activity
DuckDuckGeese	60	40	1345	270	5	Audio
EigenWorms	128	131	6	17984	5	Motion
Epilepsy	137	138	3	206	4	Human Activity
ERing	30	30	4	65	6	Human Activity
EthanolConcentration	261	263	3	1751	4	Industrial
FaceDetection	5890	3524	144	62	2	Physiological
FingerMovements	316	100	28	50	2	Physiological
HandMovementDirection	320	147	10	400	4	Physiological
Handwriting	150	850	3	152	26	Human Activity
Heartbeat	204	205	61	405	2	Audio
InsectWingbeat	30000	20000	200	78	10	Audio
JapaneseVowels	270	370	12	29	9	Audio
Libras	180	180	2	45	15	Human Activity
LSST	2459	2466	6	36	14	Industrial
MotorImagery	278	100	64	3000	2	Physiological
NATOPS	180	180	24	51	6	Human Activity
PEMS-SF	267	173	963	144	7	Traffic
PenDigits	7494	3498	2	8	10	Motion
PhonemeSpectra	3315	3353	11	217	39	Audio
RacketSports	151	152	6	30	4	Human Activity
SelfRegulationSCP1	268	293	6	896	2	Physiological
SelfRegulationSCP2	200	180	7	1152	2	Physiological
SpokenArabicDigits	6599	2199	13	93	10	Audio
StandWalkJump	12	15	4	2500	3	Physiological
UWaveGestureLibrary	120	320	3	315	8	Human Activity

A Appendix

A.1 Data Description

Table 3 summarizes the 30 datasets in the UEA Multivariate Time Series Classification Archive (Bagnall et al., 2018) used in our experiments. The archive exhibits substantial diversity in both dataset scale and time series characteristics. The number of training samples ranges from only 12 in *StandWalkJump* to 30,000 in *InsectWingbeat*, while the number of test samples varies from 15 to 20,000. The dimensionality also differs considerably across datasets, ranging from 2 channels to 1,345 channels, and sequence lengths range from 8 time steps in *PenDigits* to 17,984 time steps in *EigenWorms*. The classification tasks range from binary problems to 39-class recognition in *PhonemeSpectra*. The datasets additionally span a broad range of application domains, including human activity recognition, physiological sensing, motion analysis, audio and speech processing, industrial sensing, and traffic monitoring.

According to the broad domain categorization used in Table 3, the benchmark contains 9 human-activity datasets, 8 physiological datasets, 6 audio datasets, 4 motion datasets, 2 industrial datasets, and 1 traffic dataset. These characteristics create markedly different learning regimes. For example, *DuckDuckGeese* contains only 60 training instances but 1,345 channels, representing a high-dimensional, low-sample setting, whereas *InsectWingbeat* contains 30,000 training instances with 200 channels. Similarly, datasets such as *EigenWorms* and *MotorImagery* contain very long sequences, while *PenDigits* consists of extremely short sequences. Evaluating on the complete UEA archive therefore allows us to examine the behavior of TimeSWARM across substantially different sequence lengths, dimensionalities, sample sizes, class cardinalities, and application domains rather than a narrow subset of MTSC settings.

In addition to evaluation on the complete 30-dataset UEA archive, we also report results on a commonly used core 10-dataset subset for focused comparison with recent deep MTSC methods (Ying et al., 2025; Wen et al., 2025; Mu et al., 2025; Zhou et al., 2023). This subset consists of *EthanolConcentration*, *FaceDetection*, *Handwriting*, *Heartbeat*, *JapaneseVowels*, *PEMS-SF*, *SelfRegulationSCP1*, *SelfRegulationSCP2*, *SpokenArabicDigits*, and *UWaveGestureLibrary*. The subset has

become a common evaluation setting in recent MTSC studies, enabling direct comparison across a broad range of deep learning architectures while keeping the experimental scope manageable. We therefore report both core UEA-10 results for comparison with prior work and results over the complete UEA-30 archive for a more comprehensive assessment of generalization. Furthermore, following recent studies (Ying et al., 2025; Mu et al., 2025), we additionally report results on UEA-25 , which excludes *InsectWingbeat*, *EigenWorms*, *SpokenArabicDigits*, *JapaneseVowels*, and *CharacterTrajectories*. Accordingly, Table 1 also reports the average accuracy on UEA-25, enabling direct comparison with these recent state-of-the-art MTSC methods.

A.2 Training Environment and Hyperparameters

We implement and train TimeSWARM using the Time Series Library (TSLib) framework[1], which has also been adopted in several recent MTSC studies (Wu et al., 2023; Ying et al., 2025; Mu et al., 2025). The official test split was never used for gradient-based optimization, loss computation, or parameter updates. To remain consistent with the evaluation protocol adopted by the prior UEA results used for comparison (Wu et al., 2023; Ying et al., 2025; Mu et al., 2025), we report the best test performance across the evaluated configurations. We apply the same reporting procedure across all datasets to ensure comparability with the published benchmark results (Wu et al., 2023; Ying et al., 2025; Mu et al., 2025). Experiments are conducted using Python 3.11.15 and PyTorch 2.8.0 with CUDA 12.8.

The default training configuration is:

- **Batch size:** 16
- **Maximum training epochs:** 100
- **Early stopping patience:** 10 epochs
- **Random seed:** 2021.
- **Optimizer:** Adam (Kingma & Ba, 2015)
- **Learning-rate scheduler:** None

Following common practice in prior MTSC studies (Ying et al., 2025; Mu et al., 2025; Zhou et al., 2023; Wu et al., 2023), a small set of model hyperparameters is selected separately for each dataset through grid search to account for the substantial heterogeneity of the UEA archive in terms of dataset size, sequence length, and channel dimensionality. The search space is:

- **Number of multi-scale convolution kernels:** $\{4, 7\}$ for most datasets; for the shorter-sequence datasets *Libras*, *NATOPS*, *LSST*, and *RacketSports*, a smaller configuration with 3 kernels is additionally considered.
- **Learning rate:** $\{0.01, 0.001\}$
- **Number of MoE layers:** $\{2, 4\}$
- **Patch overlap ratio:** $\{0, 0.5\}$
- **Period candidate:** one of the top five dominant dataset-level periods
- **First-order difference representation:** True or False

A.3 Baselines

We compare TimeSWARM against a broad set of representative and recent state-of-the-art time series models spanning recurrent, convolutional, MLP, Transformer, shapelet, state-space, and hybrid architectures. The baselines appearing in the main UEA-30 comparison and the additional UEA-10 comparison are briefly summarized below. Unless otherwise stated, the reported baseline results are taken from the corresponding published comparisons described in Section 4.1.

- **SGN** models multivariate time series through k-means clustering based variable grouping and shifted-window group interactions to capture structured dependencies across channels (Ying et al., 2025).

[1] https://github.com/thuml/timeseries-Library

- **MPTSNet** integrates multi-scale periodic local-pattern modeling with global dependency learning for multivariate time series classification (Mu et al., 2025).
- **ShapeFormer** combines class-specific shapelet representations with generic Transformer features to capture both discriminative local shapes and global time series characteristics (Le et al., 2024).
- **SVPT** is a shape-level Transformer that represents subsequences as tokens and incorporates their variable and temporal-position information for MTSC (Zuo et al., 2023).
- **ShapeNet** learns discriminative multivariate shapelets by embedding shapelet candidates of different lengths into a unified representation space (Li et al., 2021).
- **TapNet** learns discriminative multivariate time series representations and class prototypes through an attentional prototype-based classification framework (Zhang et al., 2020).
- **TimeMixer++** captures time series patterns across multiple temporal scales and frequency resolutions through multi-scale and multi-resolution mixing (Wang et al., 2025).
- **MLSTM-FCN** combines an LSTM branch with a fully convolutional network to jointly capture recurrent and convolutional representations for multivariate time series classification (Karim et al., 2019).
- **WEASEL+MUSE** is a dictionary-based MTSC method that constructs symbolic representations and discriminative features across multiple variables (Schäfer & Leser, 2018).
- **LSTNet** combines convolutional and recurrent components with skip connections to model both short- and long-range temporal dependencies (Lai et al., 2018).
- **LSSL** employs linear state-space layers to model long-range sequential dependencies through a structured state-space representation (Gu et al., 2022).
- **DLinear** uses simple linear mappings over decomposed time series components, providing a lightweight MLP-based baseline Zeng et al. (2023).
- **LightTS** provides a lightweight time series modeling framework designed to retain competitive predictive performance with reduced model complexity (Zhang et al., 2022).
- **MTS-Mixers** factorizes temporal and channel correlation to model dependencies along the two dimensions using lightweight correlation modules (Li et al., 2025b).
- **ROCKET** transforms time series using a large collection of random convolutional kernels followed by a linear classifier (Dempster et al., 2020).
- **TimesNet** transforms one-dimensional temporal variations into multi-period two-dimensional representations and models intra- and inter-period patterns using convolutional blocks (Wu et al., 2023).
- **TVNet** jointly models temporal and variable dependencies through a temporal–variable representation framework with graph-based interaction modeling (Li et al., 2025a).
- **ModernTCN** advances temporal convolutional networks using large receptive fields and pure convolutional layers based model design for general time series analysis (Donghao & Xue, 2024).
- **Crossformer** incorporates hierarchical encoder-decoder transformer based self-attention mechanisms to explicitly model both temporal and cross-variable dependencies in multivariate time series (Zhang & Yan, 2023).
- **PatchTST** encodes time series as temporal patches tokens leading to a channel-independent Transformer encoder for time series analysis tasks (Nie et al., 2023).
- **Flowformer** introduces flow-based attention to efficiently model long-range dependencies in time series sequences (Wu et al., 2022).
- **GPT4TS** adapts a pretrained LLM (Large Language Model) architecture to time series representations, leveraging pretrained sequence modeling for downstream time series tasks (Zhou et al., 2023).
- **Medformer** is a multi-granularity patching Transformer that captures inter-channel correlations through cross-channel patching and models temporal patterns at multiple scales using intra- and inter-granularity self-attention (Wang et al., 2024).

Table 4: Comparison of of Accuracy and F1 Score of our proposed model vs recent state-of-the-art on UCI-HAR dataset

Model	Accuracy	F1 Score
Crossformer	90.66 ± 1.02	90.68 ± 1.04
iTransformer	93.47 ± 0.15	93.46 ± 0.16
Medformer	90.17 ± 0.52	90.27 ± 0.66
Timesnet	92.71 ± 0.50	92.75 ± 0.51
ModernTCN	92.75 ± 2.03	92.80 ± 2.18
SGN	95.62 ± 0.72	95.64 ± 0.70
Ours	$\mathbf{97.06 \pm 0.11}$	$\mathbf{97.11 \pm 0.11}$

A.4 Additional Experiments on UCI-HAR Dataset

We further evaluate TimeSWARM on the UCI Human Activity Recognition (UCI-HAR) (Anguita et al., 2013) dataset, a widely used benchmark for sensor-based activity classification. The dataset contains smartphone inertial measurements collected from 30 participants performing six daily activities: walking, walking upstairs, walking downstairs, sitting, standing, and lying down. The signals were recorded at 50 Hz using tri-axial accelerometer and gyroscope sensors and segmented into 2.56-second windows, corresponding to 128 time steps, with 50% overlap. Following the processed setting used in SGN (Ying et al., 2025), each sample contains 9 sensor channels and the dataset comprises $10,299$ labeled instances across six activity classes.

For a controlled comparison, we follow the same fixed data partitions and training protocol used in SGN (Ying et al., 2025). The original training partition is further divided to reserve 20% of the training samples for validation, while the test partition remains fixed for final evaluation. We evaluate TimeSWARM over five independent runs using random seeds $\{41, 42, 43, 44, 45\}$ and report the mean and standard deviation of both accuracy and F1 score. To minimize differences caused by training configuration rather than model design, shared hyperparameters such as the learning rate, maximum training epochs, early-stopping patience, and convolutional kernel configuration are matched to the SGN (Ying et al., 2025) experimental setting wherever applicable. Baseline results are taken directly from the comparison reported by SGN (Ying et al., 2025).

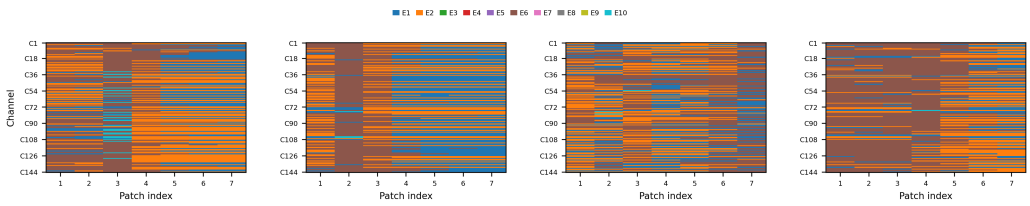
As shown in Table 4, TimeSWARM achieves the strongest performance among the compared approaches, reaching $97.06 \pm 0.11\%$ accuracy and $97.11 \pm 0.11\%$ F1 score. Compared with SGN, the strongest reported baseline, this corresponds to improvements of 1.44% in accuracy and 1.47% in F1 score. The small standard deviations across the five runs also indicate stable performance under different random initializations. These results provide additional evidence that the proposed method generalizes beyond the UEA archive to multichannel inertial-sensor classification.

A.5 Additional Visualization Analysis

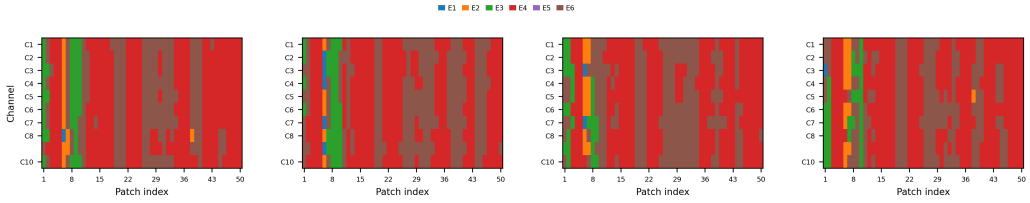
Dominant Expert Assignment Analysis. To qualitatively examine the behavior of the proposed patch-wise router, Figure 5 visualizes the dominant expert assignment across channels and temporal patches for four representative test samples from *FaceDetection*, *HandMovementDirection*, *Heartbeat*, and *NATOPS*. For each channel–patch pair (c, w), we display $\arg\max_e R_{c,w,e}$, i.e., the expert receiving the highest routing probability. These visualizations provide an interpretable summary of the local routing decisions, while the complete dense distribution $\mathbf{R}_{c,w}$ is retained by TimeSWARM for router-induced relational modeling.

Several characteristics of the learned routing behavior can be observed. First, expert assignments vary across both channels and patch locations, confirming that routing is performed at the channel–patch level rather than assigning a fixed expert to an entire physical channel. For example, the same channel can transition between different dominant experts across successive patches, allowing the representation function to adapt as its local temporal characteristics change.

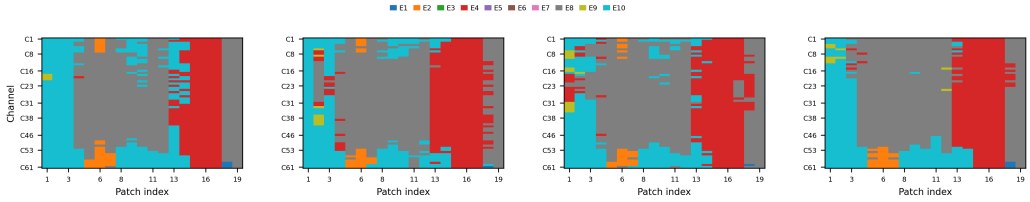
Second, the routing configurations also vary across different samples from the same dataset. This sample-dependent behavior is visible across all four datasets and reflects the input-conditioned nature of the router: channel–patches with different local temporal characteristics need not follow a predetermined routing pattern. At the same time, the maps contain coherent regions in which groups



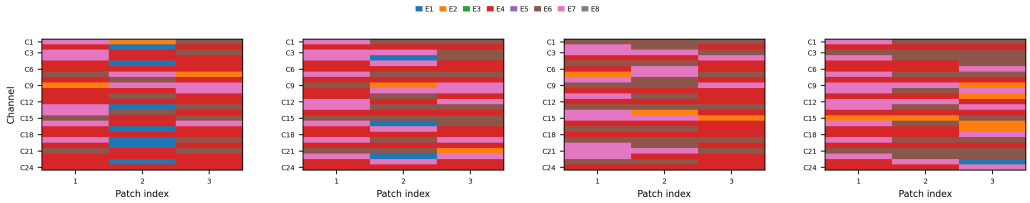
(a) *FaceDetection*



(b) *HandMovementDirection*



(c) *Heartbeat*



(d) *NATOPS*

Figure 5: Channel–patch expert-routing visualization across four representative test samples from four UEA datasets. Each cell indicates the expert receiving the highest routing probability for a particular channel–patch pair. Colors denote expert identities and are dataset specific. These dominant-assignment maps provide a qualitative view of the routing behavior.

of channels or consecutive patches share the same dominant expert, rather than exhibiting purely independent or highly fragmented assignments. This suggests that the router identifies recurring local structures shared across subsets of channels and temporal regions.

The routing patterns further differ substantially across datasets. For *FaceDetection*, which contains a large number of channels, the router forms heterogeneous channel-wise assignments over a relatively small number of patch locations. In *HandMovementDirection*, expert usage changes more extensively along the patch axis, with different experts becoming dominant over different temporal regions. *Heartbeat* exhibits broader channel–patch regions associated with a small subset of dominant experts, whereas *NATOPS*, despite containing only a few patch locations, still shows channel and sample-dependent expert selection. These qualitatively different configurations illustrate that the router does not impose a common grouping pattern across datasets but adapts its assignments to their temporal and multivariate characteristics.

It is important to note that the dominant-expert maps should not be interpreted as the actual cross-channel graph used by TimeSWARM. The relational module retains the complete soft routing distributions and treats them as channel–patch memberships over latent expert groups. Consequently, channels with compatible routing profiles can exchange information through the shared latent groups even when their dominant expert alone does not fully characterize their routing similarity. The visu-

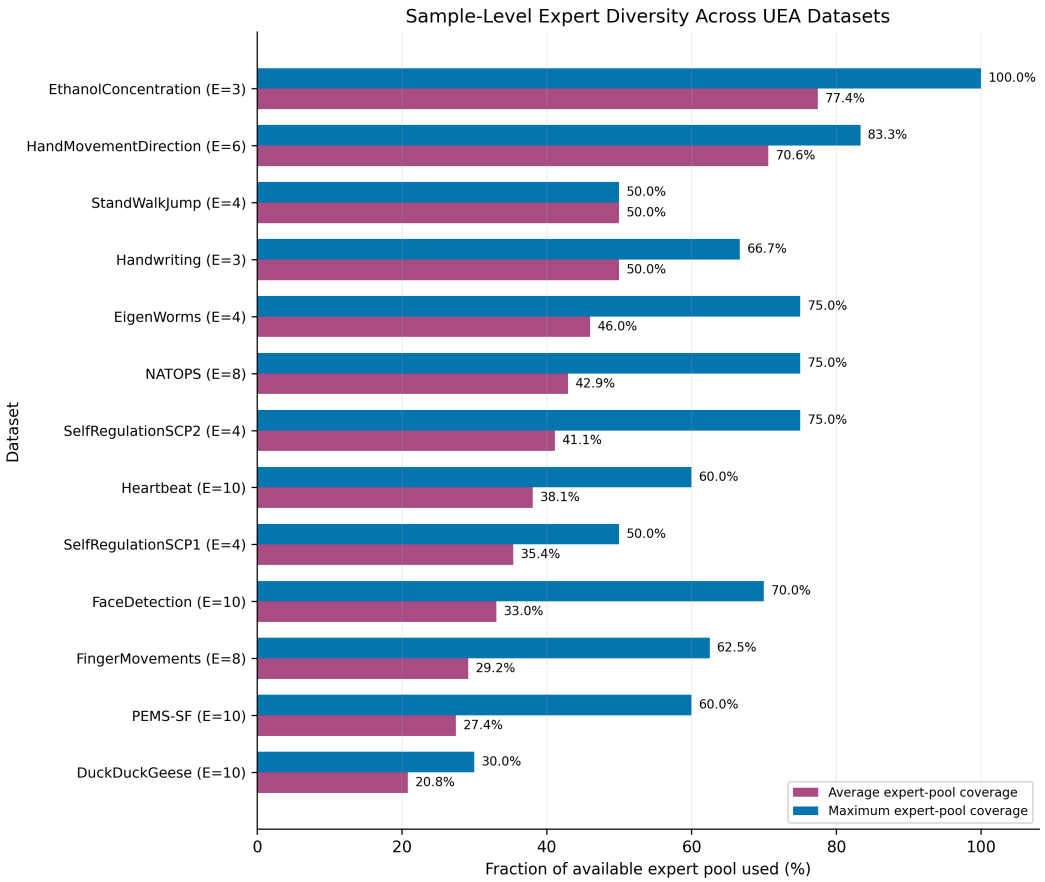
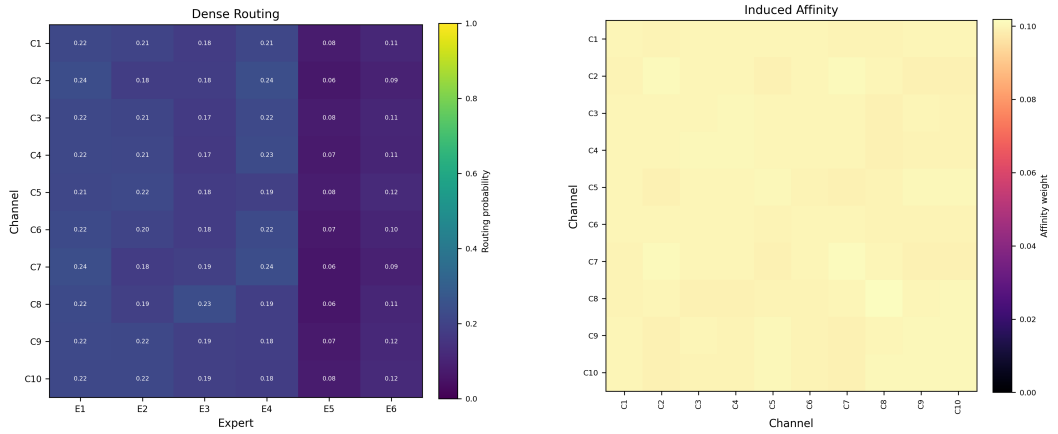


Figure 6: **Sample-level expert utilization diversity across several UEA datasets.** For each dataset, we report the average and maximum fraction of the available expert pool utilized within a sample and MoE layer. Utilization is measured as the number of distinct experts receiving non-zero weight after adaptive sparsification across all channel–patches of a sample, normalized by the dataset-specific expert-pool size E. Although sparsified expert aggregation encourages a dominant local expert affiliation for each channel–patch, different channel–patches within the same sample may favor different specialists. The figure therefore characterizes the diversity of expert utilization across a sample rather than the number of experts associated with an individual routing decision.

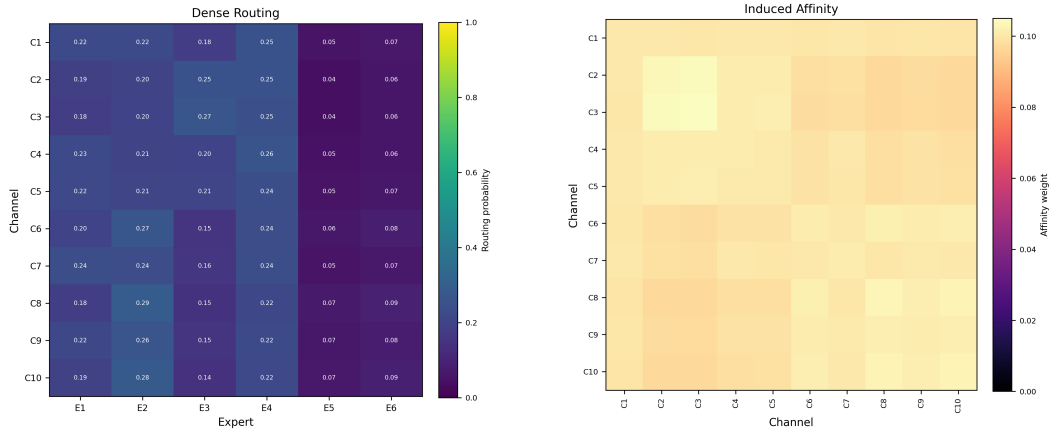
alizations complement the quantitative ablation results by illustrating the underlying routing behavior that jointly supports local expert specialization and routing-induced cross-channel interaction.

Expert-utilization diversity. The sparsified routing mechanism in TimeSWARM encourages a dominant expert affiliation for each channel–patch while allowing additional experts to contribute when the routing distribution is less decisive. We examine whether this local specialization results in repeated use of only a small subset of experts across an entire sample, or whether different channel–patches recruit different specialists. Figure 6 reports the fraction of the available expert pool utilized within each sample and MoE layer, averaged across samples, together with the maximum observed utilization. Since the expert-pool size varies across datasets, the number of utilized experts is normalized by the corresponding number of available experts E. The results reveal substantial dataset-dependent variation in expert utilization. For example, *EthanolConcentration* utilizes on average approximately 77.4% of its available expert pool and reaches all $3/3$ experts for at least one sample–layer pair. Similarly, *HandMovementDirection* utilizes approximately 70.6% of its expert pool on average and reaches $5/6$ experts. For datasets with larger expert pools, *Heartbeat* and *NATOPS* reach $6/10$ and $6/8$ experts, respectively. In contrast, *DuckDuckGeese* exhibits a more concentrated expert-utilization pattern, using approximately 20.8% of its expert pool on average and reaching at most $3/10$ experts. These results show that the sparsified routing mechanism does not



(a) HandMovementDirection, Patch 7: R_w

(b) HandMovementDirection, Patch 7: A_w



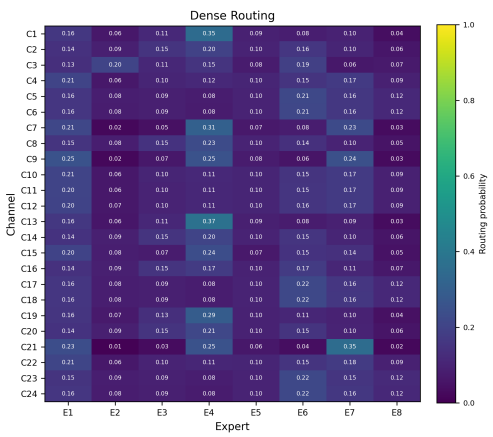
(c) HandMovementDirection, Patch 27: R_w

(d) HandMovementDirection, Patch 27: A_w

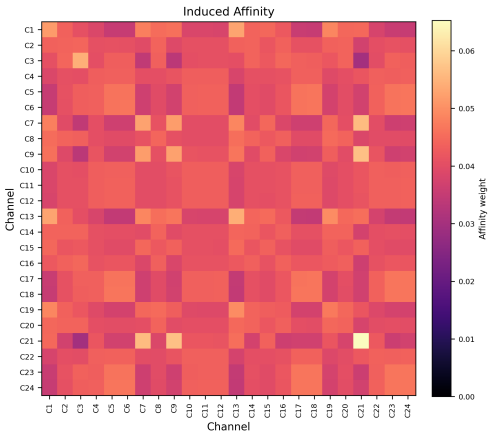
Figure 7: **Dense routing distributions and router-induced relational structures across representative temporal patches.** For each patch, R_w shows the channel-wise expert-preference distributions, while $A_w = R_w D_w^{-1} R_w^\top$ shows the corresponding router-induced channel–channel interaction weights. Changes in the dense routing profiles across patches lead to corresponding changes in the induced relational structure.

impose a fixed expert identity across all channel–patches of a sample. Instead, different local regions can favor different specialists, with the breadth of expert utilization adapting to the characteristics of the dataset and input sample. Importantly, broader utilization is not necessarily preferable in itself; a lower utilization fraction may simply indicate that a smaller subset of the learned specialists is sufficient for a particular dataset.

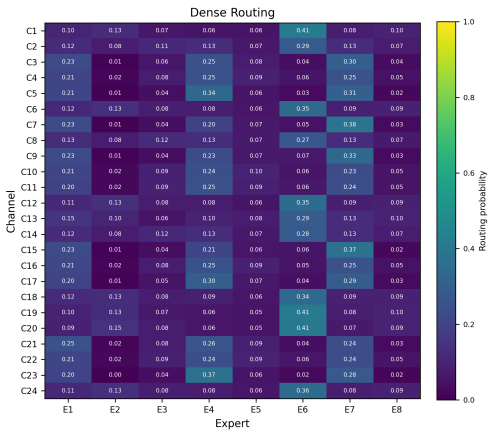
Dense Routing and Router-Induced Relational Structure. The dominant-expert visualizations (Figure 5) provide a compact view of channel–patch specialization by displaying only the highest-probability expert. However, the relational component of TimeSWARM operates on the complete dense routing distribution rather than these discrete dominant assignments. To make this distinction explicit, Figure 7 visualizes the dense routing matrix R_w together with its corresponding router-induced relational matrix $A_w = R_w D_w^{-1} R_w^\top$ for representative temporal patches from *HandMovementDirection* and *NATOPS*. Each row of R_w describes the relative expert preferences of one channel at patch location w. These routing profiles act as soft latent-group coordinates, such that channels with compatible expert-preference distributions exchange information through shared latent groups. The corresponding matrix A_w visualizes the effective channel–channel interaction weights induced by these assignments. Comparing different patches from the same sample shows that changes in the local routing profiles are accompanied by changes in the induced relational structure. For example,



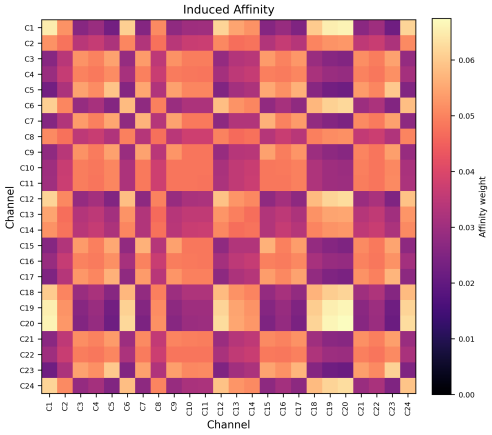
(e) NATOPS, Patch 1: R_w



(f) NATOPS, Patch 1: A_w



(g) NATOPS, Patch 2: R_w



(h) NATOPS, Patch 2: A_w

Figure 7: *Figure 7 continued.*

relatively similar routing profiles produce more homogeneous interaction patterns, whereas greater differentiation among channel-wise expert preferences produces a more structured relational matrix. The NATOPS examples further illustrate that these relations can change between neighboring temporal regions of the same input. These visualizations therefore complement the dominant-expert maps by exposing the dense information that is actually reused by the relational branch. Importantly, A_w represents a model-induced, input- and patch-dependent interaction structure rather than a predefined physical or semantic relationship between channels. Its role here is to demonstrate how the shared routing space provides the relational coordinates through which TimeSWARM couples local expert specialization with cross-channel interaction.

A.6 Additional Method Details of TimeSWARM

This section provides the architectural and implementation details omitted from the main paper for conciseness. These details do not alter the routing-induced latent-group formulation introduced in Section 3; rather, they specify its concrete implementation used in our experiments.

A.6.1 Period-Aware Channel-Patches

Inspired by the FFT-based period identification strategy introduced by Wu et al. (2023) and subsequently adopted in recent MTSC methods (Ying et al., 2025; Mu et al., 2025), we derive a set of dominant periodic candidates from the frequency characteristics of each dataset. Specifically, the top five dominant periods are retained as candidate patch lengths, and the patch length P is selected

from these candidates as a dataset-specific hyperparameter according to the same model-selection protocol used for the remaining hyperparameters. Period extraction is performed once at the dataset level prior to model training; therefore, the model does not estimate a separate period for each individual sample. Given the selected patch length P and an overlap ratio $\rho \in [0, 1)$, the patch stride is defined as

$$s = \max\left(1, \lfloor P(1-\rho) \rfloor\right). \tag{12}$$

The resulting number of patches is

$$W = \begin{cases} 1, & L \le P, \\ \left\lceil \dfrac{L-P}{s} \right\rceil + 1, & L > P. \end{cases} \tag{13}$$

When the final patch extends beyond the sequence boundary, zero padding is applied on the right to ensure complete temporal coverage. We consider two overlap configurations, $\rho \in \{0, 0.5\}$, corresponding to non-overlapping patches and 50% overlapping patches, respectively. The overlap ratio is treated as an additional dataset-specific hyperparameter.

A.6.2 Input Embedding

To complement the raw signal representation, we optionally incorporate its first-order temporal difference to emphasize local variations and transition patterns. Given a channel sequence x_t, the first-order difference is defined as

$$\Delta x_t = x_t - x_{t-1}, \tag{14}$$

with the first position set to zero. The difference is computed on the original sequence before patch extraction and padding, thereby avoiding artificial variations at padded boundaries and preserving continuity across neighboring patches. The usefulness of first-order differences can vary across datasets due to differences in signal characteristics and discriminative temporal patterns. We therefore treat the inclusion of first-order differences as a binary dataset-specific hyperparameter, selected using the same model-selection protocol as the remaining architectural hyperparameters. This allows the model to use the derivative-enhanced representation when local temporal changes are informative, while retaining the raw representation alone when they are not.

For each patch $\mathbf{X}_{c,w}$, the input representation is

$$\mathbf{V}_{c,w} = \begin{cases} [\mathbf{X}_{c,w}; \Delta\mathbf{X}_{c,w}], & \text{if first-order differences are enabled,} \\ \mathbf{X}_{c,w}, & \text{otherwise.} \end{cases} \tag{15}$$

A one-dimensional convolution with kernel size 3 projects the input into a D-dimensional latent representation,

$$\mathbf{H}^{(0)}_{c,w} = \phi_{\text{emb}}(\mathbf{V}_{c,w}) \in \mathbb{R}^{D \times P}. \tag{16}$$

We further add sinusoidal positional encodings along two complementary axes: (i) the temporal positions within each patch and (ii) the patch index w along the sequence. The resulting embedded representation is

$$\mathbf{H}^{(0)} \in \mathbb{R}^{B \times C \times W \times D \times P}. \tag{17}$$

A.6.3 Temporal–Spectral Router

The router summarizes each channel–patch using complementary temporal and spectral characteristics. For $\mathbf{H}_{c,w} \in \mathbb{R}^{D \times P}$, we first compute the feature-wise temporal mean

$$\mu_{c,w,d} = \frac{1}{P} \sum_{p=1}^{P} H_{c,w,d,p}. \tag{18}$$

To complement this temporal statistic, we characterize the spectral content of each latent feature using its dominant non-DC Fourier magnitude. Let

$$\mathbf{F}_{c,w,d} = |\mathrm{rFFT}(\mathbf{H}_{c,w,d,:})|_{\text{non-DC}} \tag{19}$$

denote the non-DC spectral magnitudes. We summarize the strength of the dominant spectral component by

$$s_{c,w,d} = \frac{\max_f F_{c,w,d,f}}{\sum_f F_{c,w,d,f} + \epsilon}. \tag{20}$$

This normalized statistic measures how strongly the spectral energy of a latent feature is concentrated around its most prominent component, while remaining insensitive to its absolute magnitude. Together with the temporal mean, it provides a compact temporal–spectral description of the local channel–patch.

The router descriptor is formulated as

$$\psi(\mathbf{H}_{c,w}) = [\boldsymbol{\mu}_{c,w}; \mathbf{s}_{c,w}] \in \mathbb{R}^{2D}. \tag{21}$$

The descriptor is processed by a lightweight MLP consisting of LayerNorm, a hidden projection, GELU activation, and an output projection to E routing logits, with the hidden dimension set to $D/2$. The resulting dense routing probabilities are independently computed for every sample, channel, and patch using equation 1. We use a single dominant spectral statistic per latent feature throughout all experiments, providing a compact router descriptor without substantially increasing the routing overhead. Given the dense routing vector $\mathbf{r}_i \in \mathbb{R}^E$ for channel–patch i, we compute an input-dependent scalar threshold as

$$\tau_i = \sigma\left(\mathbf{W}_2 \,\mathrm{ReLU}\left(\mathbf{W}_1 \mathbf{r}_i + \mathbf{b}_1\right) + b_2\right), \tag{22}$$

where the projection parameters are randomly initialized once and kept fixed during training. Experts satisfying $r_{i,e} > \tau_i$ are retained and their routing probabilities are renormalized; if none exceeds the threshold, the highest-probability expert is retained. This top-1-preserving design encourages each channel–patch to maintain a dominant expert affiliation, strengthening the latent grouping interpretation while still permitting additional experts for less decisive routing distributions. The resulting sparse probabilities are used only for expert aggregation, whereas the original dense routing distribution is preserved for router-induced cross-channel relational modeling.

The architecture further supports stacking multiple MoE blocks, each with independently parameterized experts and a router. The number of MoE blocks is treated as a model hyperparameter, and only the dense routing distribution from the final block is used to induce cross-channel interaction.

A.6.4 Router-Induced Bipartite Relational Refinement

The central relational component of TimeSWARM reuses the dense routing distribution from the final MoE block to determine cross-channel interaction. For a fixed sample and patch location w, we collect the routing probabilities of all C channels into

$$\mathbf{R}_w = \begin{bmatrix} R_{1,w,1} & \cdots & R_{1,w,E} \\ \vdots & \ddots & \vdots \\ R_{C,w,1} & \cdots & R_{C,w,E} \end{bmatrix} \in \mathbb{R}^{C \times E}, \tag{23}$$

where each row corresponds to a channel–patch and each column corresponds to an expert-associated latent group. Thus, channels with similar routing distributions have stronger membership

in the same latent groups. Let $\mathbf{H}_w \in \mathbb{R}^{C\times D\times P}$ denote the corresponding MoE representations. The total routing mass received by group e is

$$m_{w,e} = \sum_{c=1}^{C} R_{c,w,e}. \tag{24}$$

We first aggregate the channel representations into latent group representations

$$\mathbf{U}_{w,e} = \frac{\sum_{c=1}^{C} R_{c,w,e}\mathbf{H}_{c,w}}{m_{w,e}} \in \mathbb{R}^{D\times P}, \tag{25}$$

and subsequently propagate them back to each channel using the same routing assignments,

$$\mathbf{M}_{c,w} = \sum_{e=1}^{E} R_{c,w,e}\mathbf{U}_{w,e}. \tag{26}$$

Accordingly, the router provides a *channel → latent group → channel* bipartite factorization of cross-channel interaction. Equivalently, for a fixed patch the induced channel relation can be written as

$$\mathbf{M}_w = \mathbf{R}_w\mathbf{D}_w^{-1}\mathbf{R}_w^{\top}\mathbf{H}_w, \qquad \mathbf{D}_w = \operatorname{diag}\left(\mathbf{R}_w^{\top}\mathbf{1}\right). \tag{27}$$

In particular, the induced relation between channels c and c' at patch w is $A_{w,c,c'} = \sum_e R_{c,w,e}R_{c',w,e}/m_{w,e}$, making their interaction stronger when they exhibit similar expert-group assignments. Since $\mathbf{R}_w$ is obtained directly from the MoE router, the resulting relational structure is input and patch-dependent without requiring a separate graph learner. The same latent assignments are therefore reused for relational aggregation after serving as expert preference representations. The router determines which channel–patches share information, while the complete $D \times P$ representations determine what information is communicated. The relational message is projected along the latent feature dimension and incorporated through a gated residual update,

$$\overline{\mathbf{H}}_{c,w} = \mathbf{H}_{c,w} + \sigma(\gamma)\,\phi_m(\mathbf{M}_{c,w}), \tag{28}$$

where $\phi_m(\cdot)$ denotes the shared feature projection followed by normalization. The gate is initialized to a small value (0.1 in all experiments), allowing the relational branch to initially act as a lightweight correction to the local MoE representation. A small numerical floor is applied to the group mass in implementation for stability.

A.6.5 Relational Computational Complexity

An explicit patch-wise channel graph requires a $C \times C$ relational matrix. For a position-preserving representation of dimensionality $D\times P$, direct pairwise message propagation incurs a computational complexity of

$$\mathcal{O}(C^2DP) \tag{29}$$

at each patch location.

In contrast, TimeSWARM factorizes cross-channel message passing through E latent groups. Both the channel-to-group aggregation and the group-to-channel propagation require

$$\mathcal{O}(CEDP), \tag{30}$$

yielding an overall relational complexity of

$$\mathcal{O}(CEDP), \tag{31}$$

up to constant factors. Thus, when $E < C$, and particularly when $E \ll C$, the factorized formulation reduces the asymptotic cost of the relational operation relative to explicit pairwise channel propagation. It also avoids materializing a dense $C \times C$ channel-affinity matrix during standard training and inference. The equivalent adjacency

$$\mathbf{A} = \mathbf{R}\mathbf{D}^{-1}\mathbf{R}^{\top} \tag{32}$$

is constructed only when required for analysis or visualization.

We emphasize that this analysis concerns only the router-induced relational module and does not imply an end-to-end computational speedup over existing MTSC architectures. In the current implementation, all temporal experts are evaluated for each channel–patch. Routing sparsification affects expert aggregation rather than expert execution. We emphasize that training- and inference-time efficiency improvements are not the primary focus of this work. The complexity analysis above is intended specifically to characterize the router-induced relational operation and its factorized implementation. In the current implementation, all temporal experts are evaluated for each channel–patch; routing sparsification therefore affects expert aggregation rather than expert execution. Consequently, we do not claim an end-to-end computational speedup over existing MTSC architectures.

A.6.6 PSEUDO-CODE OF TIMESWARM

Algorithm 1 High-level training stage pseudo-code illustration of TimeSWARM

Input: Multivariate time series $\mathbf{X} \in \mathbb{R}^{L \times C}$, period candidate P, overlap ratio ρ, N_{moe} MoE blocks
Output: Class logits $\hat{\mathbf{y}}$

Partition each channel into period-aware patches $\{\mathbf{X}_{c,w}\}$ of length P
Embed patches and add within-patch and patch-position encodings to obtain $\mathbf{H}$
for $\ell = 1$ to N_{moe} **do**
    for all channel–patch pairs (c, w) **do**
        Compute dense routing probabilities $\mathbf{R}_{c,w}^{(\ell)}$ from temporal–spectral patch descriptors
        Obtain sparsified expert weights $\widetilde{\mathbf{R}}_{c,w}^{(\ell)}$
        Process the patch with all multi-scale temporal experts
        Aggregate expert outputs using $\widetilde{\mathbf{R}}_{c,w}^{(\ell)}$ and apply residual normalization
    end for
end for
Retain the *dense* routing matrix $\mathbf{R}$ from the final MoE block
for all patch locations w **do**
    $\mathbf{U}_w \leftarrow \mathbf{D}_w^{-1}\mathbf{R}_w^{\top}\mathbf{H}_w$ ▷ channel → latent groups
    $\mathbf{M}_w \leftarrow \mathbf{R}_w\mathbf{U}_w$ ▷ latent groups → channel
    $\mathbf{H}_w \leftarrow \mathbf{H}_w + \sigma(\gamma)\phi_m(\mathbf{M}_w)$
end for
for all channels c **do**
    Summarize each patch and model dependencies across successive patches using the inter-patch temporal module
    Broadcast the resulting patch context back to the position-preserving patch representations
end for
Reconstruct the temporal representation using overlap-aware patch aggregation
Concatenate channel representations and apply the temporal prediction head
Apply global average and maximum pooling
$\hat{\mathbf{y}} \leftarrow \text{Classifier}(\mathbf{H})$
return $\hat{\mathbf{y}}$