WHEN LABELS ARE SCARCE: AN OSCILLATORY STATE SPACE MODEL FOR VIBRATION DIAGNOSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine fault diagnosis from vibration requires learning from scarce labelled fault recordings while meeting the computational constraints of edge devices for local inference. We introduce DualRes, a compact oscillatory state-space model that combines two complementary spectral views of vibration, capturing rapid changes and fine frequency structure. Time-aligned views are processed by selective oscillatory memory, which learns how long to retain temporal patterns. The encoder contains 39,528 parameters. We evaluate supervised learning across six bearing datasets and a gearbox benchmark, with an additional gearbox pilot. Recording-level splits and explicit accounting of labelled duration distinguish data efficiency from repeated exposure to correlated samples. On the main gearbox benchmark, DualRes achieves state-of-the-art performance among the nine evaluated methods at six of seven label budgets. With about six labelled seconds per class, it improves macro-F1 by 16.1 percentage points over the next strongest comparator. On the same benchmark, DualRes achieves a 1.44-fold recording-level speedup and a 24.8-fold reduction in checkpoint storage relative to a selective state-space baseline under matched hardware and runtime conditions. However, performance on bearing datasets varies across tasks. These findings support oscillatory memory as a compact approach to vibration diagnosis under limited labelled exposure.

1 INTRODUCTION

Vibration sensors collect abundant operational data, but labelled recordings of representative faults remain scarce. Moreover, windows extracted from the same recording come from the same component and share its mounting configuration and operating history. A useful classifier must therefore learn from limited labelled exposure while remaining effective across changes in components, mounting configurations and operating conditions. At the same time, deployment on resource-constrained edge hardware imposes limits on model storage, memory and inference latency.

Evaluation design is therefore central to this problem. Randomly assigning windows from one recording to training and evaluation sets can place near-identical signals in both. Moreover, even when recordings are kept separate, both sets may still contain measurements from the same physical bearing. Vieira et al. (2026) document how partitioning and bearing diversity affect evaluation. Accordingly, we explicitly report what is held out and the unique labelled duration, distinguishing transfer across operating conditions from generalization to unseen components.

Together, these modelling and evaluation challenges raise two questions: which compact representation captures diagnostic vibration structure, and how should its data efficiency be evaluated? Model size alone answers neither question. A compact classifier must still distinguish faults under changing conditions, while its execution cost depends on the signal-processing and inference pipeline as well as its parameters.

The representation design draws on the structure of mechanical vibration, which combines local excitations, resonant responses and temporal modulation (Randall & Antoni, 2011). Short analysis windows capture rapid changes, whereas longer windows distinguish closely spaced frequencies. To combine these complementary views, we align their frame centers and process the resulting sequence with selectively damped oscillatory memory. We call this model DualRes, reflecting its two aligned spectral resolutions. Figure 7 in the appendix summarizes the design and compares its performance with the evaluated methods across labelled-data budgets.

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.

Our contributions are threefold. **Architecture:** a center-aligned multi-resolution classifier combines selective oscillatory memory and normalized state updates in a 39,528-parameter encoder. **Evaluation:** recording-level splits and explicit duration accounting distinguish labelled exposure, recording coverage and operating-condition shift. **Empirical analysis:** comparisons, learning curves, component ablations and matched desktop measurements characterize the accuracy–cost trade-off across bearing and gearbox tasks.

2 Related Work

Vibration and evaluation. Classical bearing analysis uses recurring excitations and resonances (Randall & Antoni, 2011); public benchmarks enable data-driven evaluation (Lessmeier et al., 2016; Vieira et al., 2026). DualRes learns from the magnitudes of a fixed short-time Fourier transform (STFT), without fault-order templates or an assumption of speed invariance. Unlike auxiliary-task meta-learning (Snell et al., 2017; Finn et al., 2017), our supervised setting uses only the stated labelled data for training.

Sequence representations. Structured and complex-valued recurrent models, including S4, S4D, S5 and the linear recurrent unit (LRU), establish compact alternatives to attention (Gu et al., 2022a;b; Smith et al., 2023; Orvieto et al., 2023); coRNN develops coupled oscillatory recurrence (Rusch & Mishra, 2021). Mamba introduces selective updates, and Mamba-2 relates state-space models (SSMs) to structured attention (Gu & Dao, 2024; Dao & Gu, 2024). For time-series classification, MambaSL adapts a single Mamba layer, while TSCMamba combines wavelet and temporal views (Jung & Kim, 2026; Ahamed & Cheng, 2025). Our contribution is the specific combination of aligned spectral resolutions and normalized selective oscillatory updates, rather than complex states or time–frequency fusion alone.

Alongside these state-space approaches, attention models provide different ways to organize a sequence: PatchTST operates on patches, iTransformer on variates, and Crossformer on cross-dimension structure (Nie et al., 2023; Liu et al., 2024; Zhang & Yan, 2023); Medformer and ModernTCN provide multi-granularity attention and convolutional alternatives (Wang et al., 2024; Luo & Wang, 2024). ROCKET and MiniRocket classify sequences using convolutional feature transforms (Dempster et al., 2020; 2021). The comparison spans these distinct model families, following classification benchmarking practice (Middlehurst et al., 2024). Appendix D documents their classification adaptations.

Selection and deployment. Repeated evaluation can bias model selection even without within-run leakage (Cawley & Talbot, 2010). Deployment introduces a separate concern: resource-constrained inference requires distinguishing parameter counts, activation memory and measured execution cost (Lin et al., 2020; Cai et al., 2020; Banbury et al., 2021); we therefore measure complete inference pipelines under matched conditions.

3 Multi-Resolution Oscillatory State Space Model

DualRes separates local spectral analysis from temporal memory (Figure 1). It uses vibration and the acquisition sampling rate, with no operating-condition inputs. Appendix A provides full parameterizations, initialization and implementation details.

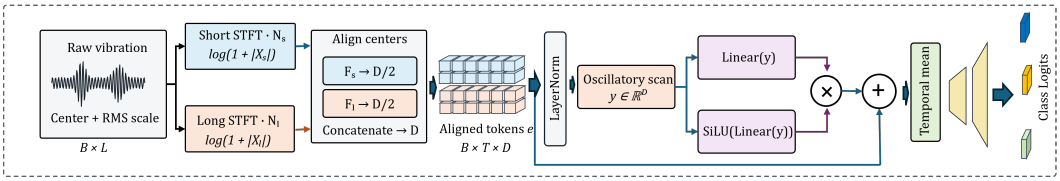

Figure 1: Overall architecture: two center-aligned magnitude STFTs feed learned projections, an oscillatory selective scan, a gated residual readout and temporal mean pooling. The classifier is trained directly on labelled vibration.

3.1 ALIGNED SPECTRAL REPRESENTATION

The frontend represents the same signal at two resolutions: one localizes short-lived changes, while the other distinguishes more closely spaced frequencies. Aligning their frame centers lets the memory process both views as one observation at each time step. The vibration signal is resampled to $f_s = 64\,\text{kHz}$ and divided into windows $x \in \mathbb{R}^L$, each containing $L = 32{,}768$ samples ($0.512\,\text{s}$). Each window is mean-centered and normalized by its root-mean-square (RMS) amplitude. Normalization removes absolute amplitude while retaining within-window variation. Let $X^{(N)}_{j,:}$ be its one-sided, unnormalized STFT with periodic Hann window, no padding and hop $H = 128$. The short and long analysis windows contain $N_s = 256$ and $N_l = 1024$ samples, respectively, corresponding to 4 and 16 ms. Their frequency-bin spacings are 250 and 62.5 Hz (Harris, 1978).

To align the two resolutions, we crop $\kappa = (N_l - N_s)/(2H) = 3$ frames from each end of the short sequence:

$$(j + \kappa)H + N_s/2 = jH + N_l/2 = \tau_j, \qquad 0 \le j < T. \tag{1}$$

This gives $T = 249$ aligned pairs at token rate $f_z = f_s/H = 500\,\text{Hz}$. Define $s_j = \log(1+|X^{(N_s)}_{j+\kappa,:}|)$ and $l_j = \log(1 + |X^{(N_l)}_{j,:}|)$. Affine projections form

$$e_j = [W_s s_j + a_s \parallel W_l l_j + a_l] \in \mathbb{R}^D, \qquad D = 64, \tag{2}$$

with $D/2$ channels per branch. All frequency bins are retained, but magnitudes discard STFT phase. Figure 2 illustrates the alignment.

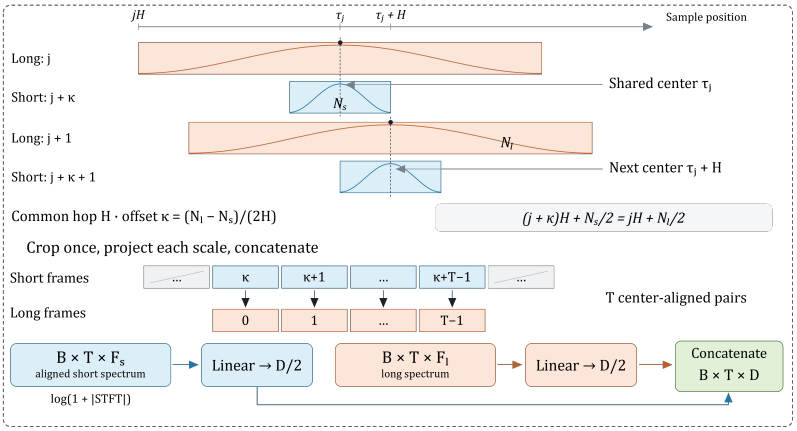

Figure 2: Dual-resolution alignment. Short and long Hann windows have different supports but paired frames share the center τ_j. Cropping the short sequence by $\kappa = (N_l - N_s)/(2H)$ gives T aligned frames. Independent projections map the F_s and F_l magnitude bins into $D/2$ channels before concatenation; B denotes batch size.

3.2 SELECTIVE OSCILLATORY MEMORY

State rotation represents oscillatory dynamics between frames, while input-dependent damping controls memory retention. The input write is weighted by one minus the retention coefficient, coupling new information to state decay. Hereafter, t indexes the aligned spectral frames. For each fused token e_t, we first compute $v_t = \text{LayerNorm}(e_t)$. Learned projections produce the input u_t and parameterize the positive damping increments $\delta_{t,d}$ and bounded complex write/read vectors $b_{t,k}, c_{t,k}$. Each channel d maintains $K = 8$ complex states, initialized at zero:

$$q_{t,d,k} = \alpha_{t,d} e^{-i\theta_k} q_{t-1,d,k} + (1 - \alpha_{t,d}) b_{t,k} u_{t,d}, \qquad \alpha_{t,d} = e^{-\delta_{t,d}}. \tag{3}$$

Here $\theta_k = 2\pi f_k/f_z$, with learned frequencies $f_k \in (0, f_z/2)$ shared across channels. These describe dynamics across spectral frames, not raw carrier frequencies. The read/write vectors are shared across channels; states and damping are channel-specific. Figure 3 illustrates the mechanism.

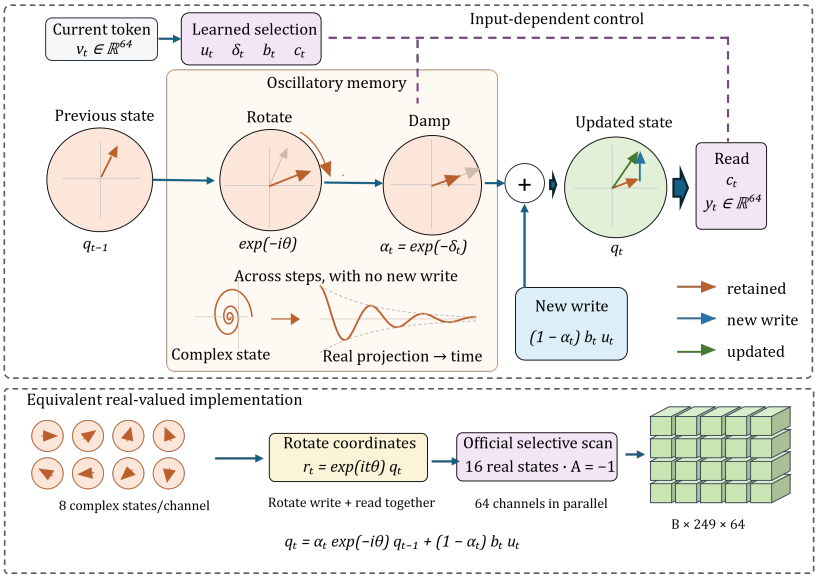

Figure 3: Oscillatory memory rotates and damps the previous state, adds a normalized write, and reads the real projection. The lower panel shows its equivalent real-valued scan implementation; the illustrated spiral is schematic.

As damping vanishes, the normalized write also diminishes. Bounded writes yield bounded states by induction (Appendix A).

To obtain an output from the memory state, we compute the real readout and gated residual:

$$y_{t,d} = \sum_{k=1}^{K} \mathrm{Re}(\overline{c_{t,k}} q_{t,d,k}),$$
$$o_t = e_t + (W_m y_t + a_m) \odot \mathrm{SiLU}(W_g y_t + a_g). \tag{4}$$

The gate transforms the memory readout; the residual preserves fused input tokens. For efficient implementation, the official real selective-scan kernel (Gu & Dao, 2024; 2026) implements Eq. 3 via rotating coordinates, without an additional standard Mamba block.

3.3 Learning and Cost

For classification, temporal mean pooling gives $\bar{o} = T^{-1} \sum_t o_t$, from which the classification head computes logits $\ell = W_h \,\mathrm{Dropout}_{0.1}(\bar{o}) + a_h$. All parameters are trained jointly from labelled support $\mathcal{S}$ using

$$\mathcal{L} = -|\mathcal{S}|^{-1} \sum_{(x,y)\in\mathcal{S}} \log \mathrm{softmax}(\ell(x))_y. \tag{5}$$

Training also applies positive multiplicative damping jitter ($\sigma = 0.10$), shared across time within each window/channel; jitter and dropout are disabled at inference. The encoder has 39,528 parameters and the head adds $65C$ for C classes. The scan costs $O(TDK)$; fixed-resolution frontend and pointwise operations also scale linearly in sequence length. Measured latency and memory are reported separately from these analytical costs.

4 Benchmark Design and Evaluation Protocol

We evaluate how diagnostic performance changes with the amount and distribution of labelled vibration. Each training set contains only the recordings and windows specified by its label budget. Evaluation recordings are kept separate before any windows are extracted. Unless stated otherwise, experiments use seed 41; three-seed studies additionally use seeds 42 and 43.

Study roadmap. We compare nine architectures under common label budgets, then repeat the top 3 methods across three seeds. We vary how many recordings provide the same total labelled duration, and we also evaluate the selected training recipe on a second gearbox dataset. Component ablations and matched desktop profiling assess architectural contributions and execution costs. Appendix E details replication coverage.

Tasks and split units. We use six public bearing datasets to test changes in component identity, operating conditions, geometry and speed profiles: Paderborn, Case Western Reserve University (CWRU), KAIST, UORED-VAFCLS (abbreviated UORED below), HUST and Ottawa (Lessmeier et al., 2016; Case Western Reserve University, 2026; Jung et al., 2023; Sehri & Dumond, 2023; Nguyen Duc Thuan & Hoang Si Hong, 2023; Huang & Baddour, 2019). The eight-class MCC5-THU gearbox benchmark (MCC5 below) varies motion mode and fault severity (Chen et al., 2024). A separate three-model pilot uses the labelled release of the 2009 Prognostics and Health Management gearbox challenge (PHM2009) for classification of 14 gearbox configurations (PHM Society, 2009). Paderborn/UORED hold out bearing identities; CWRU/KAIST hold out conditions; HUST holds out geometries; Ottawa holds out speed profiles. MCC5 has two directions: low-severity torque circulation to speed circulation (M1), and high-severity speed circulation to torque circulation (M2). The test sets include the available fault-severity levels, so motion and severity both vary. The dataset inventory, manifests and detailed split rules are in Appendices B and C.

Label budgets. For MCC5, each labelled window spans 0.512 s, and windows do not overlap. We evaluate:

- **Increasing recording count.** Use 1, 2, 3, 4 or 6 recordings per class, with four windows from each recording. Labelled duration increases from 2.048 to 12.288 s/class.
- **Changing allocation at fixed duration.** Compare 12 windows from one recording, four windows from each of three recordings, and two windows from each of six recordings. Every configuration provides 12 windows and 6.144 s/class.
- **Increasing duration within a fixed recording pool.** Keep the same six recordings per class and use four, eight or twelve windows from each. The corresponding durations are 12.288, 24.576 and 36.864 s/class.

Our primary setting uses four windows from each of three recordings per class (6.144 s/class). PHM2009 uses the same support budget and training duration; Appendix B describes its repeat-separated folds. For the bearing datasets, three overlapping labelled windows per class span 1.024 s of unique signal.

Training and evaluation. Methods use the same labelled training samples and vibration channels within each comparison. All models are trained directly on the labelled training data without pretraining. Neural-network models are trained using Adam for 300 updates on MCC5 and 100 updates on the bearing datasets. Classical methods use their respective training procedures. Appendix C.3 gives optimizer settings and gradient accumulation details. Because each optimizer update covers the complete labelled training set, more training data requires more computation even when the update count is unchanged. We evaluate the final training checkpoint rather than selecting it using test performance; repeated development on these datasets still limits claims of independent confirmation (Cawley & Talbot, 2010). The primary metric is macro-F1, the unweighted mean of per-class F1 scores; accuracy is secondary. Both use recording-level predictions and are averaged equally across folds. Neural models average window probabilities; MiniRocket averages decision scores.

Comparisons and costs. Appendix D records classification adaptations of the released implementations. Efficiency reporting separates parameters, storage, training time, memory and inference latency; Appendix I.2 defines the workloads and hardware used for these measurements.

5 Results and Analysis

5.1 Predictive performance under limited labels

Bearing diagnosis. At 1.024 labelled seconds/class, we compare DualRes, MambaSL and MiniRocket, selected for the multi-seed study based on their performance in the initial benchmark. DualRes exceeds MambaSL's mean macro-F1 on Paderborn, KAIST and UORED, ties on Ottawa, and trails on CWRU and HUST (Table 1). The largest positive difference is 7.71 percentage points on KAIST. On UORED, DualRes gains 6.24 percentage points over MambaSL, although results vary substantially across seeds. HUST remains difficult: MiniRocket performs best under its geometry shift. Figure 9 gives the single-seed comparison across all nine methods.

Table 1: Bearing macro-F1 and accuracy (%). Mean ± sample SD over three seeds, each averaged over both folds, at 1.024 unique labelled seconds/class and 100 supervised updates.

	DualRes		MambaSL		MiniRocket	
Dataset	F1	Accuracy	F1	Accuracy	F1	Accuracy
Paderborn	**48.92 ± 1.87**	**49.96 ± 1.13**	45.69 ± 2.44	47.21 ± 2.91	33.50 ± 0.13	42.08 ± 0.26
CWRU	68.04 ± 4.06	**64.29 ± 3.57**	**70.04 ± 3.03**	61.90 ± 7.43	66.37 ± 0.00	60.71 ± 0.00
KAIST	**45.00 ± 2.61**	51.11 ± 1.92	37.29 ± 3.05	40.00 ± 3.33	43.37 ± 1.59	**54.44 ± 1.92**
UORED	**64.51 ± 15.73**	**64.58 ± 15.73**	58.27 ± 3.71	58.33 ± 3.61	36.33 ± 0.00	43.75 ± 0.00
HUST	14.16 ± 2.49	20.63 ± 1.37	20.63 ± 5.71	28.17 ± 3.64	**22.76 ± 0.77**	**29.37 ± 1.37**
Ottawa	**99.44 ± 0.97**	**99.44 ± 0.96**	**99.44 ± 0.97**	**99.44 ± 0.96**	98.32 ± 0.00	98.33 ± 0.00

Gearbox diagnosis across motion modes. DualRes leads the nine-method MCC5 comparison in macro-F1 at six of seven label budgets; MiniRocket leads at 2.048 seconds/class (Table 2). In the three-seed comparison of DualRes, MambaSL and MiniRocket, DualRes exceeds the stronger comparator by 16.1 points at the primary budget and 11.9 at the largest. Table 2 also reports accuracy; Appendix E reports results for individual folds and across seeds.

A separate gearbox pilot. To assess the training recipe on another gearbox dataset, we evaluate DualRes on PHM2009 without changing its architecture or training settings. It exceeds MiniRocket, the strongest comparator, by 7.14 macro-F1 percentage points and leads in both folds (Table 3).

5.2 How should a limited label budget be allocated?

DualRes improves from 22.29% to 51.12% mean macro-F1 as coverage grows from one to six recordings/class (Figure 4a). However, this changes duration, diversity and computation together. To examine duration separately, we keep the same six recordings per class. Doubling duration gives little improvement, while mean macro-F1 reaches 54.74% at the largest budget. More exposure is useful, but its effect is not monotonic.

To examine recording diversity at a fixed labelled duration, we compare how the same twelve windows are distributed across recordings. DualRes and MambaSL perform best using four windows from each of three recordings, rather than taking all twelve windows from one recording or two windows from each of six recordings (Figure 4c).

5.3 What do the components contribute?

Ablations remove one spectral branch, oscillator rotation or damping jitter while retaining the rest of the architecture and training settings (Table 4). Removing rotation reduces macro-F1 by 8.73 percentage points on MCC5 and 27.18 points on UORED, but slightly improves performance on Paderborn. The benefit of combining resolutions also varies across datasets: the full frontend improves over either single-resolution control on CWRU, UORED and HUST, whereas the long-only control is better on KAIST and MCC5. Damping jitter likewise provides no consistent advantage across datasets. Appendix H details the controls and their effects across datasets.

Table 2: MCC5 macro-F1 (F1) and accuracy (Acc.), in percent: two-fold means. Column headers give unique labelled seconds/class. Neural methods use 300 optimizer updates; MiniRocket uses a ridge classifier trained on the same labelled data.

Method	Metric	2.048	4.096	6.144	8.192	12.288	24.576	36.864
DualRes	F1	19.61	**35.46**	**48.55**	**45.96**	**55.22**	**48.04**	**55.63**
	Acc.	**24.17**	**42.92**	**50.00**	**48.33**	**59.17**	**53.33**	**58.33**
MambaSL	F1	11.88	24.14	24.00	25.31	27.15	26.54	27.92
	Acc.	19.17	27.50	29.17	31.25	32.92	32.50	33.33
MiniRocket	F1	**22.48**	28.24	28.35	33.25	37.97	38.57	42.61
	Acc.	22.92	30.42	30.42	35.42	39.17	39.17	44.17
TSCMamba	F1	14.50	16.76	7.52	14.15	16.13	10.47	13.16
	Acc.	17.50	19.17	13.75	16.67	17.50	12.50	15.83
Medformer	F1	4.42	6.54	12.83	14.94	18.31	25.83	24.25
	Acc.	11.67	12.50	17.50	21.67	24.17	29.58	28.75
ModernTCN	F1	3.92	6.50	6.25	8.04	10.62	17.43	20.55
	Acc.	10.83	17.50	14.58	19.17	18.75	19.17	21.67
PatchTST	F1	5.28	8.54	10.45	12.88	16.76	16.60	26.02
	Acc.	10.42	18.75	19.17	21.25	25.42	25.83	30.00
iTransformer	F1	14.79	18.13	13.23	15.11	10.76	12.73	10.50
	Acc.	15.83	18.75	14.58	17.08	11.25	15.83	12.92
Crossformer	F1	10.34	10.15	11.75	12.27	13.15	20.83	17.94
	Acc.	20.83	19.58	17.08	18.33	19.17	25.83	22.50

First five columns: 1/2/3/4/6 recordings $\times$ 4 windows. Last two: 6 recordings $\times$ 8/12 windows.

Table 3: PHM2009 configuration recognition (%). Support: 3×4 disjoint windows/class; evaluation: 140 recordings/fold. Neural models use 300 updates; MiniRocket uses a ridge classifier trained on the same labelled data.

Method	P1 F1	P2 F1	Mean F1	Mean accuracy
DualRes	**69.34**	**52.94**	**61.14**	**60.36**
MambaSL	37.68	34.78	36.23	39.64
MiniRocket	57.41	50.60	54.00	52.14

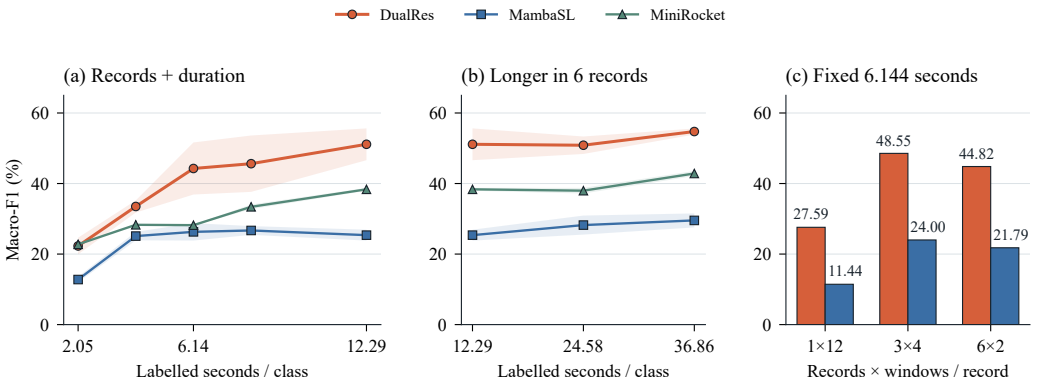

Figure 4: Three different data-efficiency questions on MCC5. (a) Increase recordings and duration jointly. (b) Increase duration in the same six-record pool. (c) Redistribute the same 12 windows across recordings. Panels (a,b): markers show three-seed means and shaded bands show one sample SD; Panel (c): single-run bars. Every value averages both folds.

Table 4: Component ablations: recording macro-F1 (%), both folds. Labelled duration: 1.024 s/class for bearing datasets and 6.144 s/class for MCC5. All interventions are retained.

Dataset	Full	Short only	Long only	No rotation	No jitter
Paderborn	51.07	47.46	51.85	**52.42**	50.69
CWRU	**71.28**	60.65	64.14	68.04	**71.28**
KAIST	47.12	34.22	**55.67**	43.84	47.12
UORED	**81.23**	54.05	53.61	54.05	**81.23**
HUST	**16.65**	5.51	14.33	11.30	16.16
Ottawa	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
MCC5	48.55	42.84	**51.72**	39.82	50.08

5.4 How does training loss change under the same label budget?

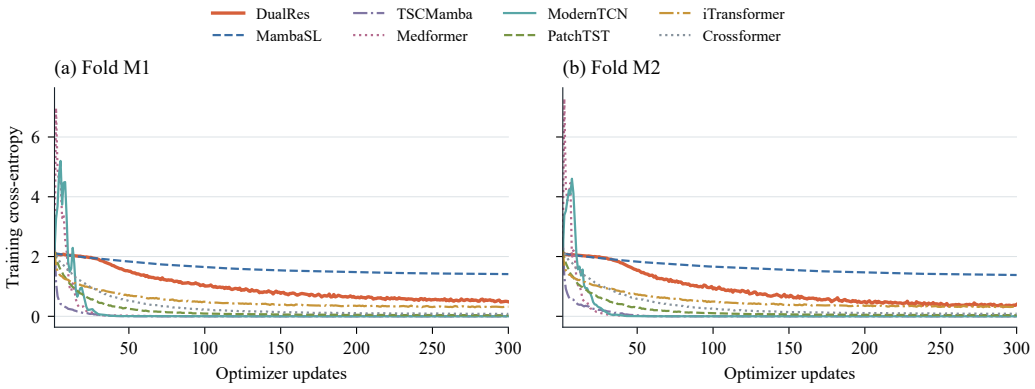

Figure 5: Training cross-entropy for eight neural methods, without smoothing, at 12.288 labelled seconds/class (six recordings, four windows each). Table 2 reports held-out performance.

Several comparators reach near-zero training loss before DualRes yet attain lower macro-F1 on held-out data (Figure 5; Table 2). Lower training loss therefore does not necessarily imply better generalization. Appendix G.2 documents selection of training duration using source-domain validation data.

5.5 Accuracy, storage and measured execution cost

DualRes has 40,048 parameters including the gearbox classification head, versus MambaSL's 15,592. Its checkpoint nevertheless requires 24.8-fold less storage (0.164 versus 4.065 MiB), since checkpoint size also depends on saved buffers.

Turning to inference speed, on MCC5, DualRes takes 1.75 ms/window versus 2.00 ms for MambaSL, and 196.65 versus 283.85 ms per recording, including preprocessing (1.44-fold speedup). Some attention models are faster but have lower macro-F1 (Figure 6); Appendix I.2 gives measurements for all seven datasets.

Table 5 summarizes costs across all seven datasets.

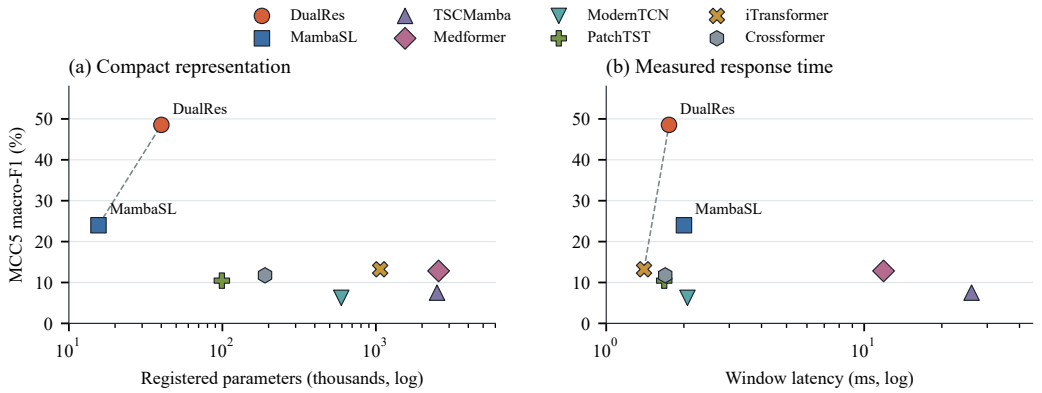

Figure 6: MCC5 primary setting: macro-F1 versus (a) parameter count and (b) time to process one window under the same measurement conditions. Dashed segments join the nondominated observed neural models. Latency includes model-specific preprocessing and transfers on one RTX A4000, FP32 with TF32 off. MiniRocket's CPU costs are reported separately.

Table 5: Seven-dataset window costs: median [range] of dataset latency medians and maximum peak allocated CUDA memory. RTX A4000, batch one, FP32. Appendix I.2 reports dataset-specific measurements and MiniRocket's CPU costs.

Method	Window latency: median [range], ms	Maximum allocated MiB
DualRes	1.75 [1.56–2.63]	**11.46**
MambaSL	1.65 [1.53–2.00]	40.94
TSCMamba	22.88 [22.33–26.08]	28.38
Medformer	11.89 [11.14–13.97]	27.45
ModernTCN	2.06 [1.76–3.23]	11.92
PatchTST	1.82 [1.68–2.01]	13.92
iTransformer	**1.33** [1.23–2.22]	13.45
Crossformer	1.77 [1.64–2.26]	14.19

6 Conclusion and Future Work

We introduced DualRes, a compact vibration classifier that combines aligned spectral views with selective oscillatory memory. Its 39,528-parameter encoder captures local spectral structure and its evolution over time. DualRes leads the nine-method gearbox comparison at six of seven label budgets, although its advantages are less consistent across bearing datasets. The accompanying benchmark examines how labelled duration and recording diversity affect diagnostic accuracy. Training and inference measurements complement these results by quantifying the computational resources required.

Nevertheless, the current experiments primarily assess changes in bearings and operating conditions. Generalization to unseen machines therefore warrants further evaluation, as does efficiency on edge devices beyond the desktop measurements reported here.

Building on these findings, future work will investigate adaptive spectral resolution and streaming inference to support continuous monitoring under changing operating conditions and practical deployment constraints.

Reproducibility Statement

Section 3 and Appendix A describe the model, its mathematical formulation and implementation. Section 4 and Appendices B–D specify the datasets, splits, labelled-data budgets, training settings and comparator adaptations; Appendix I.2 details the efficiency measurements. The accompanying anonymous code supplement provides the standalone DualRes implementation, raw-data converters, fixed split/window manifests and runnable training and evaluation scripts for DualRes and the comparators. Appendix J describes the package contents, data requirements and reproduction scope. The code will be made publicly available on GitHub upon acceptance.

AI-Use Statement

Generative AI tools were used to assist with refining the academic style, clarity and final presentation of the manuscript.

References

Md Atik Ahamed and Qiang Cheng. TSCMamba: Mamba meets multi-view learning for time series classification. *Information Fusion*, 120:103079, 2025. doi: 10.1016/j.inffus.2025.103079. URL https://doi.org/10.1016/j.inffus.2025.103079.

Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Jeremy Holleman, Nat Jeffries, Csaba Kiraly, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, Urmish Thakker, Antonio Torrini, Peter Warden, Jay Cordaro, Giuseppe Di Guglielmo, Javier Duarte, Stephen Gibellini, Videet Parekh, Honson Tran, Nhan Tran, Wenxu Niu, and Xuesong Xu. MLPerf Tiny benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/da4fb5c6e93e74d3df8527599fa62642-Abstract-round1.html.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. TinyTL: Reduce memory, not parameters for efficient on-device learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/81f7acabd411274fcf65ce2070ed568a-Abstract.html.

Case Western Reserve University. Bearing data center: Download a data file, 2026. URL https://engineering.case.edu/bearingdatacenter/download-data-file. Accessed September 2026; 48 kHz drive-end release.

Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010. URL https://jmlr.org/papers/v11/cawley10a.html.

Shijin Chen, Zeyi Liu, Xiao He, Dongliang Zou, and Donghua Zhou. Multi-mode fault diagnosis datasets of gearbox under variable working conditions. *Data in Brief*, 54:110453, 2024. doi: 10.1016/j.dib.2024.110453. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC11070670/.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Angus Dempster, François Petitjean, and Geoffrey I. Webb. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 2020. doi: 10.1007/s10618-020-00701-z. URL https://doi.org/10.1007/s10618-020-00701-z.

Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. MINIROCKET: A very fast (almost) deterministic transform for time series classification. In *ACM SIGKDD Conference on Knowledge*

Discovery and Data Mining, 2021. doi: 10.1145/3447548.3467231. URL https://doi.org/10.1145/3447548.3467231.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1126–1135, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu and Tri Dao. Official Mamba implementation, 2026. URL https://github.com/state-spaces/mamba. Software requirements and implementation; accessed September 2026.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/e9a32fade47b906de908431991440f7c-Abstract-Conference.html.

Fredric J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978. URL https://web.mit.edu/xiphmont/Public/windows.pdf.

Huan Huang and Natalie Baddour. Bearing vibration data under time-varying rotational speed conditions. Mendeley Data, version 2, 2019. URL https://data.mendeley.com/datasets/v43hmbwxpm/2.

Wonho Jung, Seong-Hu Kim, SungHyun Yun, Jaewoong Bae, and Yong-Hwa Park. Vibration, acoustic, temperature, and motor current dataset of rotating machine under varying load conditions for fault diagnosis. Mendeley Data, version 6, 2023. URL https://data.mendeley.com/datasets/ztmf3m7h5x/6.

Yoo-Min Jung and Leekyung Kim. MambaSL: Exploring single-layer mamba for time series classification. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YDl4vqQqGP.

Christian Lessmeier, James Kuria Kimotho, Detmar Zimmer, and Walter Sextro. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. In *PHM Society European Conference*, volume 3, 2016. doi: 10.36001/phme.2016.v3i1.1577.

Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. MCUNet: Tiny deep learning on IoT devices. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/86c51678350f656dcc7f490a43946ee5-Abstract.html.

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

Donghao Luo and Xue Wang. ModernTCN: A modern pure convolution structure for general time series analysis. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vpJMJerXHU.

Matthew Middlehurst, Patrick Schäfer, and Anthony Bagnall. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 2024. URL https://doi.org/10.1007/s10618-024-01022-1.

Nguyen Duc Thuan and Hoang Si Hong. HUST bearing: a practical dataset for ball bearing fault diagnosis. *BMC Research Notes*, 16, 2023. doi: 10.1186/s13104-023-06400-4.

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. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26670–26698, 2023. URL https://proceedings.mlr.press/v202/orvieto23a.html.

PHM Society. Public data sets: 2009 phm challenge competition data set, 2009. URL https://phmsociety.org/public-data-sets/. Labelled expanded release; accessed September 2026.

Robert B. Randall and Jérôme Antoni. Rolling element bearing diagnostics—a tutorial. *Mechanical Systems and Signal Processing*, 25:485–520, 2011. doi: 10.1016/j.ymssp.2010.07.017.

T. Konstantin Rusch and Siddhartha Mishra. Coupled oscillatory recurrent neural network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=F3s69XzWOia.

Mert Sehri and Patrick Dumond. University of ottawa rolling-element dataset—vibration and acoustic faults under constant load and speed conditions (UORED-VAFCLS). Mendeley Data, version 5, 2023. URL https://data.mendeley.com/datasets/y2px5tg92h/5.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html.

João Paulo Vieira, Victor Afonso Bauler, Rodrigo Kobashikawa Rosa, and Danilo Silva. Towards a more realistic evaluation of machine learning models for bearing fault diagnosis. *Mechanical Systems and Signal Processing*, 258:114640, 2026. doi: 10.1016/j.ymssp.2026.114640. URL https://doi.org/10.1016/j.ymssp.2026.114640.

Yihe Wang, Nan Huang, Taida Li, Yujun Yan, and Xiang Zhang. Medformer: A multi-granularity patching transformer for medical time-series classification. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/3fe2a777282299ecb4f9e7ebb531f0ab-Abstract-Conference.html.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vSVLM2j9eie.

Supplementary Material

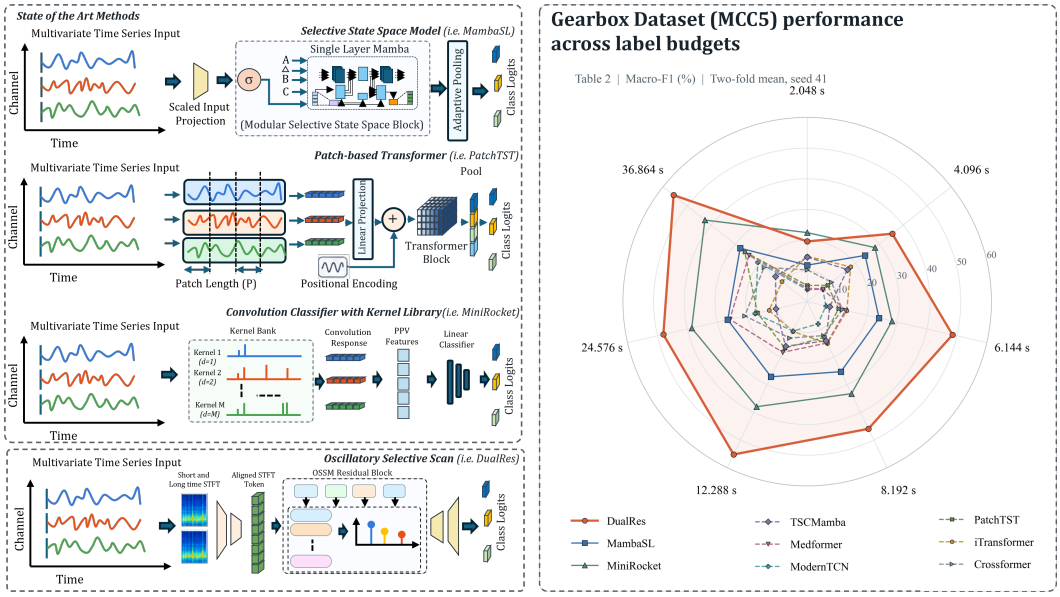

Figure 7: Paper overview. Left: representative sequence classifiers and DualRes's aligned spectral views with oscillatory memory. Right: MCC5 macro-F1 across seven label budgets under the common supervised protocol.

A Mathematical and Implementation Details

A.1 Tensor shapes and parameter accounting

Table 6: Per-window tensor shapes; batch axis omitted.

Stage	Shape	Parameters
Raw window	32768	0
Aligned magnitudes	249×129, 249×513	0
Affine spectral projections	$249 \times (32 + 32)$	20,608
LayerNorm	249×64	128
Input and damping projections	two $64 \to 64$ maps	8,320
Complex write/read projections	two $64 \to 16$ maps	2,080
Damping rates and frequencies	$64 + 8$	72
Output mixing and gate	two $64 \to 64$ maps	8,320
Temporal mean	64	0
Classifier	$64 \to C$	$65C$
Encoder total		39,528

Eight complex states correspond to 16 real coordinates/channel. Shared read/write maps account for 2,080 parameters; they are not independent 64×8 projections for every channel.

A.2 Normalization and complete spectral construction

Let $x \in \mathbb{R}^L$ be a univariate vibration window after resampling to $f_s = 64{,}000\,\text{Hz}$. We use $L = 32{,}768$, corresponding to $0.512\,\text{s}$. Each window is first centered and then divided by the RMS

amplitude of the centered signal:

$$\widetilde{x}_n = \frac{x_n - \mu_x}{\max(r_x, 10^{-10})}, \qquad \mu_x = \frac{1}{L}\sum_n x_n, \quad r_x = \sqrt{\frac{1}{L}\sum_n (x_n - \mu_x)^2}. \tag{6}$$

For analysis length $N \in \{256, 1024\}$ and common hop $H = 128$, we compute

$$X_{j,k}^{(N)} = \sum_{m=0}^{N-1} \widetilde{x}_{jH+m} w_N[m] e^{-2\pi i k m/N}, \qquad 0 \le k \le N/2, \tag{7}$$

where $w_N[m] = \frac{1}{2}(1 - \cos(2\pi m/N))$ is a periodic Hann window. Write $N_s = 256$ and $N_l = 1024$ for the short and long analysis lengths, and $F_s = N_s/2 + 1$ and $F_l = N_l/2 + 1$ for their one-sided frequency-bin counts. The diagrams show batch size B explicitly; the equations omit that independent batch dimension.

The short transform yields 255 frames and the long transform 249. To align their centers, define the frame offset $\kappa = (N_l - N_s)/(2H) = 3$ and the shared center $\tau_j = jH + N_l/2$. Retaining short frames $j + \kappa$ gives the shared center in Eq. 1. Here $T = \lfloor (L - N_l)/H \rfloor + 1 = 249$. Exact alignment by an integer-index crop requires integral κ, as satisfied by our settings. The resulting token rate is $f_z = f_s/H = 500\,\text{Hz}$. Figure 2 illustrates this alignment and the subsequent branch projections.

A.3 Coefficient parameterization and initialization

For token t, let $v_t = \mathrm{LayerNorm}(e_t)$. An affine map produces $u_t \in \mathbb{R}^D$. Each channel d maintains $K = 8$ complex states $q_{t,d,k}$, initialized at zero. Frequencies are shared across channels:

$$f_k = \frac{f_z}{2}\,\mathrm{sigmoid}(\rho_k), \qquad \theta_k = 2\pi f_k / f_z, \tag{8}$$

$$\delta_{t,d} = \frac{\exp(\lambda_d + \tanh([W_\gamma v_t + a_\gamma]_d))}{f_z}, \qquad \alpha_{t,d} = \exp(-\delta_{t,d}). \tag{9}$$

At the spectral-frame rate defined in Section 3, initial frequencies are log-spaced from 10 to 200 Hz. Baseline damping rates $\exp(\lambda_d)$ are initialized from half-lives log-spaced between 1 and 64 ms, and the input-dependent damping projection starts at zero.

Separate affine maps from v_t to $2K$ real coordinates parameterize

$$b_{t,k} = \tanh(\beta_{t,k}^{\mathrm{R}}) + i\tanh(\beta_{t,k}^{\mathrm{I}}), \tag{10}$$

$$c_{t,k} = \big(\tanh(\chi_{t,k}^{\mathrm{R}}) + i\tanh(\chi_{t,k}^{\mathrm{I}})\big)/\sqrt{K}. \tag{11}$$

A.4 Bounded-state argument

For bounded writes $|b_{t,k}u_{t,d}| \le M$, the triangle inequality gives

$$|q_{t,d,k}| \le \alpha_{t,d}|q_{t-1,d,k}| + (1 - \alpha_{t,d})M \le \max\{|q_{0,d,k}|, M\}. \tag{12}$$

The second inequality follows by induction. This state bound assumes bounded writes; it does not constrain every learned projection or imply generalization. Equation 3 defines a discrete update rather than an exact discretization of a general forced continuous oscillator.

A.5 Exact damping regularization and cost

During training only, damping increments receive multiplicative jitter:

$$\delta'_{t,d} = \delta_{t,d}\exp(\epsilon_d - \sigma^2/2), \qquad \epsilon_d \sim \mathcal{N}(0, \sigma^2), \quad \sigma = 0.10. \tag{13}$$

Noise is independent across windows and channels but shared across token positions within a window. Its positive multiplier has mean one and preserves positive damping. However, it does not preserve the expected decay coefficient itself. The perturbed δ' defines both decay and normalized writing.

For fixed resolutions, STFT cost is $O(T(256\log 256 + 1024\log 1024))$; projection and mixing costs are $O(T(32(129 + 513) + D^2 + DK))$, and the state scan costs $O(TDK)$. All scale linearly with sequence length at fixed widths and analysis windows. The recurrent state contains DK complex values; measured activation memory is discussed in Appendix I.2.

A.6 Efficient implementation in rotating coordinates

The complex recurrence can be evaluated using a real selective-scan kernel. To do so, we factor the learned rotation out of the state, leaving a real decay and moving the phase factors into the read and write vectors. Write $a_{t,d} = \exp(-\delta_{t,d})$ and let the recurrence be Eq. 3. The substitution $r_{t,d,k} = e^{it\theta_k} q_{t,d,k}$ yields

$$r_t = a_t r_{t-1} + (1 - a_t) e^{it\theta} b_t u_t. \tag{14}$$

Rotating the read vector by the same phase leaves $\mathrm{Re}(\bar{c}_t q_t)$ unchanged. The official real selective scan uses $A = -1$, $\Delta = \delta$ and a write term $\Delta B u$. Scaling its input by

$$g(\delta) = \frac{-\operatorname{expm1}(-\delta)}{\delta}, \qquad \lim_{\delta \to 0} g(\delta) = 1, \tag{15}$$

reproduces the normalized write without a complex CUDA kernel. This is an algebraic representation of the chosen discrete recurrence, not a claim that an unmodified Mamba block implements the same model. For the same recurrence without an input write, $|q_t| = \exp(-\sum_{j \le t} \delta_j)|q_0|$. Asymptotic decay requires the cumulative damping to diverge; positivity alone does not supply a uniform decay rate over every possible input. Neither state boundedness nor asymptotic decay ensures bounded optimization gradients.

A.7 Gradient paths and memory use

Gradients pass through both spectral projections, input-dependent coefficients, the scan, output gate and classifier. The FFT itself has no trained weights. Processing a full window also requires storing intermediate activations, so memory use exceeds the size of the recurrent state alone.

B Datasets and Task Definitions

Table 7 consolidates the retained labels, acquisition rates and evaluation split for each dataset. All tasks use one vibration channel; metadata defines recording roles but is not an input feature. Some experiments use a subset of the classes in the original dataset.

B.1 Bearing task boundaries

Paderborn. The evaluated task excludes combined-fault bearings. Reserved confirmation bearings `K001`, `KA30`, `KI21` and `KB23` are excluded before their signals are accessed. Both artificial and real damage occur in the release; the manifests identify the participating bearings.

CWRU and KAIST. These folds test operating-condition changes, not independent bearing populations. The fixed manifests and signal-channel choices are common to every method.

UORED-VAFCLS. All available stages of one bearing remain in one split. Manufacturer–fault, load–fault and stage-dependent acquisition correlations remain limitations. Although centering removes constant DC offsets, it does not remove manufacturer or gain effects. The scalar speed marker is not a tachometer trace and is not used by the model.

HUST and Ottawa. In HUST, bearing identity changes together with geometry, so the split cannot isolate their effects. Ottawa tests speed-profile changes on the recorded apparatus; near-perfect scores do not establish independent-machine transfer.

B.2 Gearbox recording roles

MCC5-THU. Version 2 contains 240 recordings. We use only `gearbox_vibration_x`. The healthy and missing-teeth classes each contain 12 recordings; every other class contains 36. M1 trains on low-severity torque circulation and evaluates all speed-circulation recordings; M2 trains on high-severity speed circulation and evaluates all torque-circulation recordings. Health and missing teeth have no severity suffix. Each evaluation fold has 120 recordings; motion and severity composition change together.

Table 7: Public datasets and evaluated tasks. Rates refer to the selected native signal, before conversion to 64 kHz. IR/OR/RE denote inner-race, outer-race and rolling-element faults. Each task uses two predefined train–test splits.

Dataset	Classes	kHz	Retained labels	Evaluation distinction
Paderborn	3	64	Healthy, IR, OR	Held-out bearing identities
CWRU	4	48	Healthy, IR, OR, RE	Held-out operating conditions
KAIST	5	25.6	Healthy, IR, OR, misalignment, unbalance	Held-out load-condition groups
UORED-VAFCLS	3	42	Healthy, IR, OR	Held-out bearing identities; stages grouped by bearing
HUST	7	51.2	Healthy, IR, OR, RE, IR+OR, IR+RE, OR+RE	Held-out bearing geometries
Ottawa	5	200	Healthy, IR, OR, RE, cage fault	Held-out speed-profile groups
MCC5-THU	8	12.8	Healthy; missing teeth; gear wear; gear pitting; tooth crack; tooth break; tooth break + IR; tooth break + OR	Opposite circulation mode; severity composition also changes
PHM2009 mirror	14	$200/3$	Helical configurations H1–H6 and spur configurations S1–S8; case labels, not fourteen fault types	Held-out recording repeat; operating conditions also change

PHM2009. The labelled mirror contains 280 recordings: five speeds, two loads and two repeats per case. P1 uses repeat 1 at 30/35/40 Hz and high load for training and evaluates repeat 2; P2 uses repeat 2 at 50/45/40 Hz and low load for training and evaluates repeat 1. Four disjoint windows are extracted from each training recording, beginning at 1 s and giving 6.144 s/class. Both neural models use 300 optimizer updates. This expanded labelled release differs from the original unlabeled challenge pool and its fault type/location/magnitude output (PHM Society, 2009). We use the first vibration column.

B.3 Acquisition compatibility and provenance

Native rates determine polyphase resampling to 64 kHz (MCC5: $5/1$; PHM2009: $24/25$). However, resampling cannot recover information beyond the original Nyquist limit. Per-window normalization follows Eq. 6 and removes absolute amplitude. Acquisition manifests retain original identities, release paths, versions and available license metadata; raw redistribution remains governed by each release's terms. Fold sizes and training-data accounting are specified below.

C Full Benchmark Specification

C.1 Manifests, split construction and training-data selection

The bearing experiments use fixed recording manifests. The MCC5 scaling experiment uses `ROLES.json`, which enumerates the source record indices at each level and the entire opposite-mode evaluation set. Roles are assigned before windows are extracted. No evaluation record is used for training or for estimating the classical feature-scaling parameters. For M1 the nested source-condition order is 1000 rpm/10 Nm, 3000/20, 2000/10, 1000/20, 3000/10 and 2000/20. For M2 it is 20 Nm/3000 rpm, 10/1000, 10/2000, 20/1000, 10/3000 and 20/2000. The first n entries are retained for every class.

C.2 WINDOW PLACEMENT AND EXPOSURE ACCOUNTING

MCC5 training windows begin at resampled index $15f_s + jL$, for $j = 0, \ldots, W-1$. Adjacent training windows are disjoint. With n recordings per class, W windows per recording, $C = 8$ classes and $L/f_s = 0.512\,\mathrm{s}$, the total number of training windows N_{support}, labelled seconds per class E_{class} and total labelled seconds E_{total} are

$$N_{\mathrm{support}} = CnW, \qquad E_{\mathrm{class}} = 0.512nW, \qquad E_{\mathrm{total}} = CE_{\mathrm{class}}. \tag{16}$$

In contrast to MCC5, the bearing protocol uses 50% overlap *within* a labelled training recording.

Table 8: MCC5 exposure accounting; no overlap correction is needed for these training windows.

Records/class	Windows/record	Windows/class	s/class	Total s
1	4	4	2.048	16.384
2	4	8	4.096	32.768
3	4	12	6.144	49.152
4	4	16	8.192	65.536
6	4	24	12.288	98.304
6	8	48	24.576	196.608
6	12	72	36.864	294.912

For n_w windows its unique duration is $0.512 + 0.256(n_w - 1)$ seconds/class when one training recording is used per class. Thus three windows cover 1.024 unique seconds. Within-recording overlap does not cross split boundaries.

C.3 TRAINING AND CHECKPOINT RULES

Neural models use Adam with initial learning rate 5×10^{-4}, decay 0.99 per update and microbatches of four windows. Update e therefore uses learning rate $5 \times 10^{-4} 0.99^{e-1}$. Let N be the total number of labelled training windows and $\mathcal{L}_b$ the mean loss for a microbatch of b windows. Each microbatch loss is weighted by b/N before accumulating gradients, giving the gradient of the average loss over the complete labelled training set for each optimizer update. However, batch-dependent layers still see microbatch statistics; accumulation is not equivalent to a single full-batch forward pass. Evaluation uses the final checkpoint from each completed training run. Seed repetitions vary model initialization while keeping the development folds and evaluation domains fixed.

C.4 RECORDING-LEVEL SCORING

The aggregation in Section 4 assigns one predicted label per recording; windows are not independent evaluation replicates. With C fixed classes, the primary metric is

$$\mathrm{macroF1} = \frac{1}{C} \sum_{c=1}^{C} \frac{2\mathrm{TP}_c}{2\mathrm{TP}_c + \mathrm{FP}_c + \mathrm{FN}_c}, \tag{17}$$

where TP_c, FP_c and FN_c are the numbers of true positives, false positives and false negatives for class c. Undefined class ratios are set to zero. Accuracy is the fraction of correctly classified evaluation recordings:

$$\mathrm{Accuracy} = \frac{1}{R} \sum_{r=1}^{R} \mathbf{1}[\hat{y}_r = y_r], \tag{18}$$

where R is the number of evaluation recordings, and $\hat{y}_r$ and y_r are the predicted and true labels of recording r. Both metrics are computed separately for each fold and then averaged equally across folds.

Although training examples are balanced across classes, evaluation recordings need not be: MCC5 has six healthy and six missing-teeth recordings per evaluation fold, versus 18 for each other label.

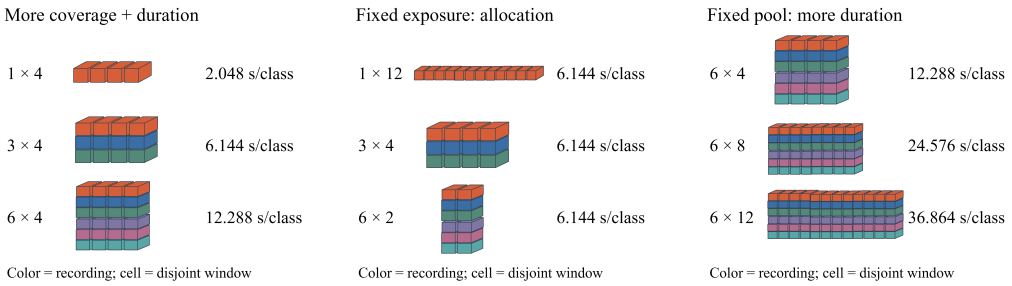

Figure 8: Three support-allocation experiments. Each cell denotes a disjoint 0.512 s window; each row/color within a group is a different recording. Only source recordings supply these windows. The diagrams distinguish adding data from reallocating a fixed duration.

D Comparator Configurations and Adaptations

D.1 Shared recipe and released classification paths

All compared neural models receive the same 32768-sample training windows and label budget under the MCC5 recipe above. The comparison is of released classification architectures under a fixed scarce-label recipe, not a reproduction of each paper's dataset-specific tuning grid. Internal normalization and augmentation follow each documented implementation.

Table 9: MCC5 comparator settings. FFN denotes the feed-forward network width. Full configurations accompany the saved training runs.

Method	Architecture settings
MambaSL	Width 32, expansion 1, convolution 4; selective Δ, B, fixed C; no D skip; gating projection, 360 kernels, 8 heads.
PatchTST	2 layers, width 32, FFN 128, 4 heads; patch 2048, stride 1024; released classification head.
iTransformer	2 layers, width 32, FFN 64, 4 heads; released classification path on a univariate series.
Crossformer	1 configured encoder layer, width 32, FFN 64, 4 heads, factor 3, segment length 1024.
Medformer	6 layers, width 128, FFN 256, 8 heads; patch sizes 128/256/512; inter-granularity attention enabled; released `none,drop0.25` augmentation.
ModernTCN	Stage widths 32/64/128, one block/stage; kernels 9 and 5; patch 32, stride 16; dropout .5, classification dropout .1.
TSCMamba	One Mamba, state 128, expansion 2, convolution 4; projected space 64, image size 64, patch 8; additive fusion and max pooling; dropout .2, ROCKET disabled.
MiniRocket	Requested 10,000 features, actual 9,996; centering/scaling parameters computed from the training data and RidgeClassifierCV.

D.2 Method-specific interpretation

MambaSL uses a fixed released long-sequence configuration across tasks (Jung & Kim, 2026). TSCMamba uses the released scalar Morlet scale `64`, with the resulting continuous wavelet transform (CWT) resized to 64×64. The raw-input tensor is arranged to match the released linear projection. The minimum and maximum used to scale the CWT are calculated from training data only. We use scikit-image 0.24 instead of the released implementation's version 0.22 for NumPy compatibility. MiniRocket's training features are centered and divided by the featurewise centered L_2 norm, with near-zero scaling factors replaced by one. Its grid of ridge regularization strengths has

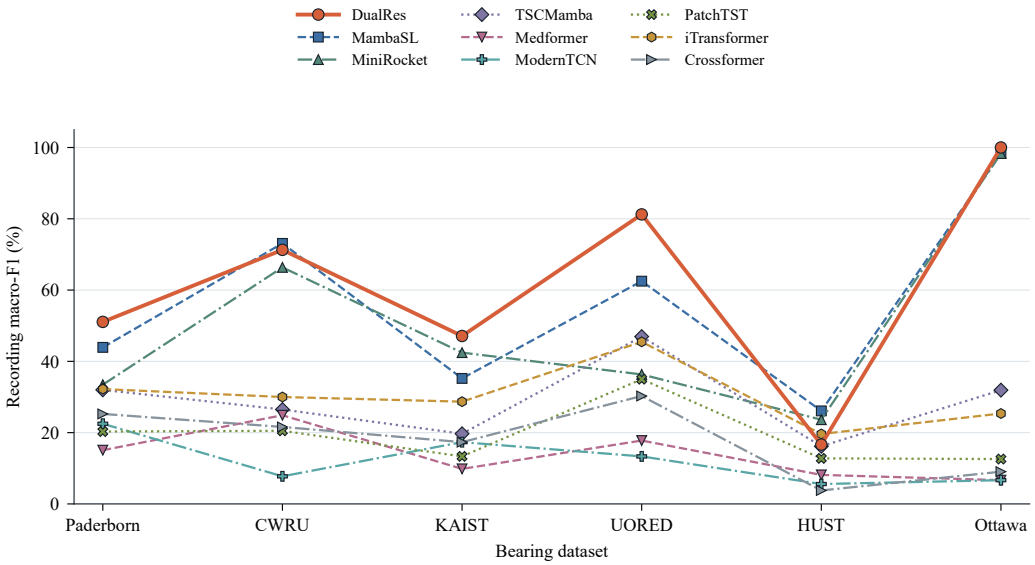

Figure 9: All nine methods on the six bearing datasets at 1.024 unique labelled seconds/class. Markers show recording macro-F1 averaged over both folds. Lines connect results for visual tracking across datasets; they do not represent a continuous variable. Three-seed summaries appear in Table 1.

ten log-spaced values from 10^{-3} to 10^{3}. MiniRocket follows its fixed-transform classifier pipeline (Dempster et al., 2021). Ridge regression does not produce an iterative neural training-loss curve; we report its learned coefficients and storage separately from neural parameter counts.

E Expanded Predictive Results and Replication

E.1 Bearing tasks and variation across folds

Table 1 summarizes macro-F1 and accuracy across three seeds, whereas Table 10 reports individual-fold macro-F1 from the initial run.

E.2 Gearbox performance across label budgets

Table 2 reports both metrics for all seven label budgets. The per-fold table reports both directions of motion-mode transfer at the primary and six-record settings.

E.3 Three-seed replication coverage

Figure 4(a,b) reports the three-seed means and standard deviations. The MCC5 replication compares DualRes, MiniRocket and MambaSL across all seven budgets, with both folds per seed. Within this nine-method scope, these are the three highest single-run means across the two extended budgets. All three methods are also evaluated across three seeds on each bearing dataset. For HUST, we additionally evaluate iTransformer, ranked third in the broad single-run comparison, across three seeds; its mean macro-F1 is $17.25 \pm 4.21\%$ and accuracy is $21.03 \pm 3.83\%$ (mean $\pm$ sample SD). Other iTransformer bearing results use one seed. The initial training run is shared with the broad comparison at each matching configuration.

E.4 Generalizing the training recipe to another gearbox dataset

All models are trained using the labelled PHM training recordings. Neural models reuse the MCC5 training recipe; MiniRocket recomputes its transform parameters and trains a new ridge classifier

Table 10: Bearing macro-F1 (%), at 1.024 unique seconds/class. Both folds use 100 supervised updates for neural models; MiniRocket trains its ridge classifier. Results from the initial run are reported separately for the two predefined train–test splits.

Dataset	Method	Fold 1	Fold 2	Mean
Paderborn	DualRes	33.24	68.90	**51.07**
	MambaSL	33.56	54.29	43.92
	MiniRocket	26.70	40.21	33.46
CWRU	DualRes	69.17	73.40	71.28
	MambaSL	76.19	70.00	**73.10**
	MiniRocket	68.45	64.29	66.37
KAIST	DualRes	49.00	45.23	**47.12**
	MambaSL	32.43	38.00	35.21
	MiniRocket	57.67	27.23	42.45
UORED	DualRes	88.57	73.89	**81.23**
	MambaSL	75.00	50.00	62.50
	MiniRocket	26.19	46.46	36.33
HUST	DualRes	12.18	21.12	16.65
	MambaSL	22.30	29.91	**26.10**
	MiniRocket	23.25	24.02	23.63
	iTransformer	22.17	17.07	19.62
Ottawa	DualRes	100.00	100.00	**100.00**
	MambaSL	96.64	100.00	98.32
	MiniRocket	96.64	100.00	98.32

Table 11: MCC5 macro-F1 (%) by fold. All nine evaluated methods.

Method	3 rec., M1	3 rec., M2	6 rec., M1	6 rec., M2
DualRes	**42.94**	**54.16**	**46.19**	**64.26**
MambaSL	22.87	25.13	20.12	34.19
MiniRocket	17.71	38.99	23.95	52.00
TSCMamba	8.20	6.83	13.41	18.85
Medformer	9.44	16.21	17.58	19.04
ModernTCN	3.31	9.19	9.38	11.87
PatchTST	8.11	12.79	16.71	16.81
iTransformer	9.36	17.10	8.55	12.97
Crossformer	6.37	17.13	14.27	12.04

on PHM training data using the same procedure. Table 3 reports aggregate scores. To complement these aggregate scores, Figure 10 reports recall for individual gearbox configurations, averaged over folds. Each class has ten evaluation recordings per fold; Table 7 defines H1–H6 and S1–S8.

F Recording Diversity versus Labelled Exposure

F.1 Matched exposure control

The allocations in Figure 4c share both training-set size and update count. Figure 4(c) labels the exact macro-F1 scores. Recording conditions are drawn from the same source pool.

Using three recordings yields higher macro-F1 than using one recording for both methods in both folds, but increasing from three to six recordings does not consistently improve performance.

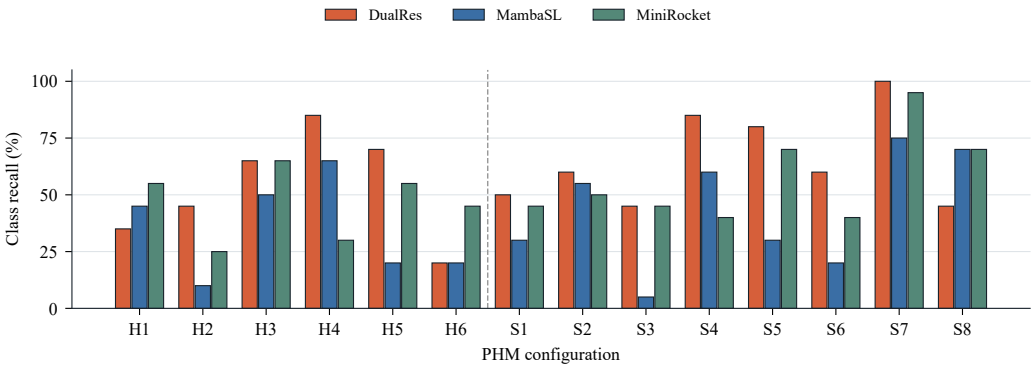

Figure 10: PHM class recall, averaged over both repeat-separated folds. The dashed separator distinguishes helical and spur configurations. This class-level view complements the aggregate scores in Table 3.

G Training Dynamics and Duration Selection

G.1 Optimization across architectures

Figure 5 in the main paper reports the training-loss curves for the compared models. Each training log records the update number, loss averaged over the labelled training set and learning rate. When comparing these curves, differences in stochastic layers should also be considered, even though all eight neural methods use cross-entropy.

G.2 Selection of training duration using source-domain validation data

The separate duration pilot compares DualRes and MambaSL using five source-mode recordings/class for training and holds one operating condition out for validation: M1 holds `2000rpm_20Nm`, M2 holds `20Nm-2000rpm`. Training uses three overlapping windows per recording (5.120 unique seconds/class); this differs from the later non-overlapping benchmark. Checkpoints at 50/100/200/300 updates are assessed only on these source-mode validation files. Opposite-motion recordings are never accessed by this study. We select the duration with the highest mean validation macro-F1 across both models and folds, breaking ties by lower log loss and then fewer updates. This selects 300 updates, which we then use for all MCC5 neural comparators. The training logs record update numbers without per-update wall-clock timestamps; training time is therefore reported separately from these learning curves.

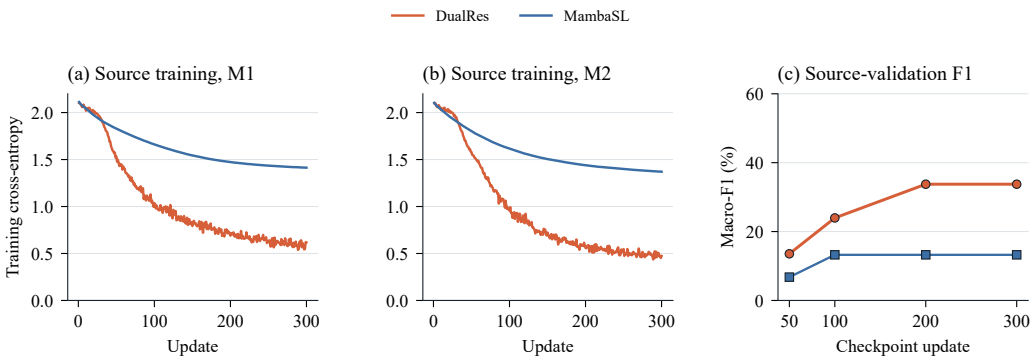

Figure 11: Selecting training duration using source conditions. (a,b) Unsmoothed training losses; (c) source-validation recording macro-F1 averaged over the two folds. The shared 300-update choice uses both methods, not target-fold accuracy.

H Component Ablations

The ablation study comprises 28 control configurations and 56 training runs across the two folds. Each is compared with the full model at the matching support budget. Table 4 in the main paper reports the full model and every control on all seven datasets. These are descriptive one-seed effects; they do not establish uncertainty across initializations.

H.1 Frontend resolution controls

The short-only and long-only variants set the other branch's projected features to zero before LayerNorm. Both retain the 64-channel backbone, 249 frame centers, state capacity and head. The disabled projections remain in the model but are not trained, so the trainable parameter count is lower than the total count. These ablations isolate each spectral branch's contribution; the single-resolution variants are not separately tuned.

H.2 Memory and regularization controls

The no-rotation control fixes oscillator frequencies at zero. The no-jitter control sets jitter strength to zero while retaining the random draw, preserving the sequence of dropout random numbers. Pooling, initialization of retained weights and the remaining recipe stay fixed.

I Desktop Efficiency Measurements

I.1 Cost definitions and training measurements

Registered parameters count all model parameters; the active parameter count includes the scalar parameters in tensors that receive gradients during training. Checkpoint storage includes persistent buffers (such as positional tables) and checkpoint-format overhead, and need not equal four bytes per parameter. GPU allocation and host-memory measurements are defined below.

Table 12: MCC5 training and storage costs at 6.144 labelled seconds/class (both folds). Training time and peak allocated training memory were recorded on a shared host; they are not isolated inference benchmarks.

Method	Registered	Active	File MiB	Training s	Train MiB
DualRes	40,048	40,048	**0.164**	**49.8**	41.8
MambaSL	**15,592**	**15,592**	4.065	68.9	251.0
MiniRocket	n/a†	n/a†	0.802	5.2	n/a (CPU)
TSCMamba	2,502,466	2,502,466	9.555	100.0	82.2
Medformer	2,560,904	2,560,904	12.301	444.2	298.6
ModernTCN	595,112	595,112	2.308	65.9	81.2
PatchTST	99,208	99,208	4.392	54.7	38.2
iTransformer	1,066,024	1,066,024	4.079	53.1	51.6
Crossformer	189,576	66,440	4.782	56.0	**37.2**

† MiniRocket has 9,996 fixed-transform features and 79,976 learned linear coefficients; these coefficients are reported separately from neural parameter counts. Storage includes persistent buffers and serialization overhead. GPU allocation is a framework counter, not total device memory. Matched desktop inference is reported separately.

Training time covers only the optimization loop, with CUDA synchronization immediately before and after timing; evaluation is excluded. Separately, evaluation at the extended budgets was resumed from saved final checkpoints after CUDA errors; recovery runtime is not reported as full-workflow training time. Training on a shared host is identified separately from isolated inference measurements.

I.2 MATCHED DESKTOP INFERENCE

We measure all 63 combinations of nine methods and seven datasets. Each measurement uses a fresh process, four CPU/linear-algebra threads and the same first training recording within its dataset. Neural paths use one RTX A4000, FP32, TF32 disabled; MiniRocket runs on CPU and is labelled separately. The saved checkpoint is from the first fold at the primary support budget. No training or other experiment runs concurrently with these measurements.

The window workload measures prediction from a normalized window in CPU memory to scores returned to CPU memory, including transfers and method-specific transforms. It uses 20 warm-ups and 120 timed repetitions. Native-record timing starts from the selected channel in host memory and includes resampling, normalization and all prescribed window predictions; it excludes disk parsing and model loading. Both prepared-record and native-record workloads use two warm-ups, ten repetitions and a common batch cap of four. Bearing evaluation prefixes and the full MCC5 window count follow the original evaluation code. Recording-level timings across datasets therefore correspond to different workloads.

Table 13: Matched desktop inference costs across datasets. Latencies are medians; native-record timing includes preprocessing and all prescribed windows. Neural methods use one RTX A4000; MiniRocket uses four CPU threads. Recording workloads differ across datasets.

Method	Paderborn	CWRU	KAIST	UORED	HUST	Ottawa	MCC5
(a) Window latency (ms)							
DualRes	1.56	1.61	1.60	1.96	1.99	2.63	1.75
MambaSL	1.56	1.92	1.64	**1.53**	1.94	1.65	2.00
MiniRocket (CPU)	55.90	54.52	54.61	54.20	54.03	54.48	60.21
TSCMamba	23.63	22.48	22.88	23.31	22.33	22.42	26.08
Medformer	11.35	11.19	11.14	13.03	13.23	13.97	11.89
ModernTCN	1.76	1.84	3.23	2.15	2.00	2.95	2.06
PatchTST	1.73	1.77	1.89	1.99	2.01	1.82	1.68
iTransformer	**1.33**	**1.23**	**1.37**	2.22	**1.24**	**1.28**	**1.40**
Crossformer	1.64	1.77	2.26	1.84	2.16	1.75	1.69
(b) Native-record latency (ms)							
DualRes	8.22	21.70	**146.51**	**24.44**	**12.34**	41.44	196.65
MambaSL	12.92	29.05	152.33	39.19	14.95	51.25	283.85
MiniRocket (CPU)	255.27	474.24	556.55	581.61	125.75	584.36	4049.40
TSCMamba	329.44	505.36	658.93	762.50	164.65	771.09	5361.47
Medformer	50.61	89.81	226.66	173.33	36.95	153.72	1069.51
ModernTCN	14.27	23.46	155.97	26.18	13.32	40.25	232.70
PatchTST	7.45	26.07	153.04	28.38	14.23	47.61	220.23
iTransformer	**6.05**	**19.53**	150.25	28.26	18.54	**35.40**	**184.22**
Crossformer	10.12	21.35	146.97	30.07	14.46	36.99	210.69
(c) Peak allocated CUDA memory (MiB)							
DualRes	11.46	11.46	11.46	11.46	**11.46**	11.46	**11.46**
MambaSL	40.94	40.94	40.94	40.94	40.94	40.94	40.94
MiniRocket (CPU)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
TSCMamba	28.38	28.38	28.38	28.38	28.38	28.38	28.38
Medformer	26.35	26.57	26.79	26.35	27.23	26.79	27.45
ModernTCN	**10.67**	**10.92**	**11.17**	**10.67**	11.67	**11.17**	11.92
PatchTST	13.90	13.90	13.90	13.90	13.91	13.90	13.92
iTransformer	13.45	13.45	13.45	13.45	13.45	13.45	13.45
Crossformer	14.17	14.18	14.18	14.17	14.19	14.18	14.19

Panel (c) uses the batch-one window workload. Absolute peak allocated CUDA memory includes loaded tensors and temporary allocations, but excludes CUDA context, driver and host memory. N/A: CPU backend, for which CUDA memory is not applicable.

Table 14: Training recordings used for inference measurements and DualRes tail latency. The 95th-percentile (P95) estimates use 120 window and 10 record repetitions. RSS is the sampled absolute process resident memory and is not model-only storage.

Dataset	Native s	Windows	Win. p95 ms	Record p95 ms	RSS MiB
Paderborn	4.00	14	2.10	9.80	853.55
CWRU	10.11	23	2.12	26.01	868.90
KAIST	120.00	23	2.76	153.05	989.84
UORED	10.00	35	2.38	36.10	880.24
HUST	10.00	7	2.86	13.92	871.34
Ottawa	10.00	35	3.16	49.19	911.23
MCC5	60.00	233	2.69	232.98	1012.18

In addition to latency, we measure memory in a separate untimed pass. CUDA allocated and reserved bytes are distinct allocator counters; neither measures the full device or operating-system footprint. Process resident set size (RSS) is sampled every 10 ms and may miss shorter peaks. The exported ledger also contains prepared-record timings, incremental memory, buffers, head/encoder counts, raw timing repetitions, input/checkpoint hashes and prediction agreement checks. Zero incremental RSS does not mean zero memory demand: previously allocated memory can be reused. Tail timings are descriptive estimates from these finite repetition counts.

J Code Supplement and Reproduction

J.1 Package contents

The anonymous supplement `DualRes_OSSM_Anonymous_Code.zip` contains the standalone model in `src/dualres_ossm/model.py`, with training and recording-level evaluation in `src/dualres_ossm/train.py` and `src/dualres_ossm/evaluate.py`. The directory `configs/manifests/` contains 14 fixed manifests: six bearing tasks, seven MCC5 label budgets and the PHM2009 evaluation. Each specifies both folds, recording labels, exact window starts and training settings. Raw-file converters and channel maps are provided for all eight datasets. Portable comparator adapters and a common runner support the nine methods in the main benchmark; `configs/experiments.json` lists the main and replication runs. Historical ablation and efficiency scripts remain under `reference_code/`. Raw datasets, trained checkpoints, experiment logs and manuscript sources are not included in this code archive.

J.2 Installation and prepared inputs

The package README gives installation and execution instructions for Python 3.10 on Linux or WSL with CUDA-enabled PyTorch and the official `mamba_ssm` selective scan. The command `dualres-fetch` obtains the pinned public comparator repositories listed in `configs/upstreams.json`. After downloading the datasets separately, `dualres-prepare` converts the selected vibration channels to the prepared input format. The common `dualres-benchmark` entry point trains and evaluates one method at a time:

```
dualres-benchmark train --method dualres \
  --manifest configs/manifests/mcc5_r3_w4.json \
  --data-root /path/to/prepared-recordings \
  --output outputs/mcc5 --device cuda:0
dualres-benchmark evaluate --method dualres \
  --manifest configs/manifests/mcc5_r3_w4.json \
  --data-root /path/to/prepared-recordings \
  --output outputs/mcc5 --device cuda:0
```

Prepared inputs are one-dimensional float32 NPY recordings resampled to 64 kHz, with recording identity and time origin preserved. Per-window normalization is applied subsequently. `docs/DATA.md` specifies this input format and channel requirements. Conversion preserves the dataset-specific preprocessing used in the original experiments, including resampling boundaries. `docs/`

RUNS.md specifies commands, seeds, dependency versions and aggregation rules. A model must be trained on each fold's training data before evaluation. Neural methods use CUDA; MiniRocket retains its CPU training procedure.

J.3 VERIFICATION AND REPRODUCTION SCOPE

docs/VERIFICATION.md, docs/verification.json and PACKAGE_CHECKS.json document model, manifest, command-line and CPU unit checks. Raw conversion was compared with the original cache on one source recording per dataset. A fresh-package smoke test passed for all nine methods on one CWRU fold: two updates per neural model and one MiniRocket training run, followed by checkpoint reload and one-record evaluation (docs/smoke_checks.json). These are software checks; they do not establish full-campaign reproduction. The full campaign has not been rerun through the package, and the paper reports the original research runs. CHECKSUMS.json records file hashes. The main-comparison runner is independent of the research workspace. By contrast, historical ablation and profiling excerpts are provided for inspection. They depend on research helper modules and experiment-plan files that are not included; changing input or checkpoint paths alone is insufficient to run them. These dependencies are described in reference_code/README.md.

J.4 EXPERIMENTAL SEEDS AND RUNTIME

MCC5 training-data sampling uses permutation seed 1041. Encoder and classifier initializations use the model seeds specified in Section 4. The reported experiments used WSL Ubuntu 22.04, Python 3.10, the official mamba_ssm selective scan and RTX A4000 GPUs. Training settings and recording-level aggregation are documented in docs/PROTOCOL.md.