RELIABLE VIRTUAL SENSING: A MULTI-DOMAIN BENCHMARK FOR ROBUSTNESS UNDER SENSOR FAILURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Virtual sensing, the estimation of hard-to-measure quantities from available sensor measurements, is a critical enabler for control and monitoring in cyber-physical systems. However, when sensors fail, learning-based predictors can produce physically implausible estimates that propagate to system-level failures. We argue that real-world deployment demands robustness and introduce MuViS-C, the first multi-domain benchmark of robustness against common sensor failures in learning-based virtual sensing. Building on an existing nominal-performance benchmark and established corruption taxonomies, it covers ten sensor failure modes, from subtle drifts to catastrophic signal dropouts, at multiple severities. These are paired with complementary robustness measures capturing average error under corruption, relative degradation, and worst-case fragility. Across nine datasets from six domains, we benchmark six architectures spanning gradient-boosted trees and the major inductive biases for sequence modeling: convolution, recurrence, attention, and MLP-mixing. On the attention-based architecture, we further probe three robustification strategies. We find that (i) every model degrades substantially under corruption, becoming worse than a naïve predictor on at least one corruption setting, (ii) gradient-boosted tree ensembles achieve strong robustness, and (iii) dedicated robustification closes the gap between the attention-based architecture and the most robust models, though each strategy hurts nominal performance. The benchmark's multi-domain design proves essential, as model rankings shift across datasets, and no single domain captures the full robustness picture. MuViS-C is open-source and extensible to new datasets, failure modes, measures, and models.

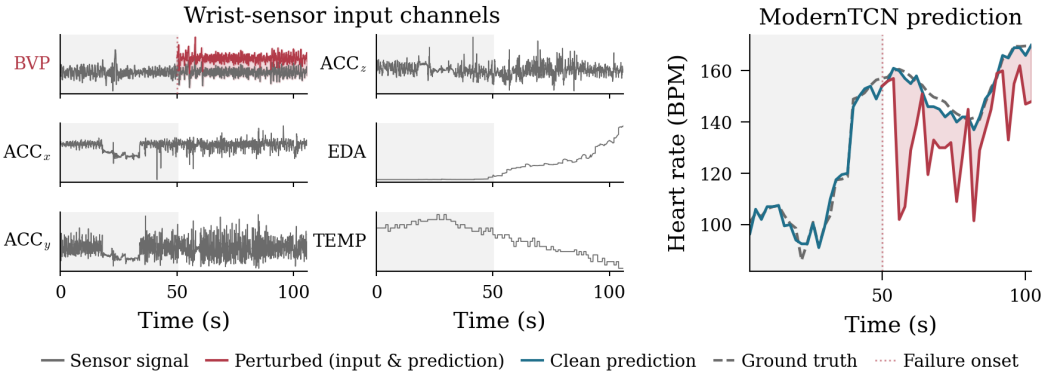


Figure 1: **Motivating Example.** An offset on the Blood Volume Pulse channel corrupts ModernTCN heart-rate estimates, causing substantial errors despite only one of six sensors being affected.

1 INTRODUCTION

Virtual sensing plays a central role in modern cyber-physical systems (CPS), enabling the estimation of variables from available sensors when direct measurement is technically or economically infeasible

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.

(Martin et al., 2021). With applications ranging from air quality monitoring and vehicle dynamics to industrial processes and health monitoring, virtual sensors provide a cost-effective and often safety-critical alternative to physical instrumentation (Ma et al., 2024; Zetterberg, Oskar and Tevell, Axel, 2023; Mondal et al., 2024; Reiss et al., 2019b). These models can be physics-based, data-driven, or hybrid, depending on system knowledge and data availability. Recent efforts have standardized the evaluation of learning-based models under nominal operating conditions (Brandt et al., 2026). However, these models still depend on physical sensors as inputs and therefore inherit their most critical vulnerability: sensors fail. Thermal drift, occlusion, electrical noise, and calibration errors can corrupt measurements in real-world deployments. When such failures occur, they induce distribution shifts that learning-based models are rarely trained to handle. The result can be catastrophic errors that propagate to system-level failures (Brandt et al., 2025).

Robustness to such deployment-time hazards has been identified as a core problem in ML safety (Hendrycks et al., 2022), with standardized corruption benchmarks in computer vision (Hendrycks & Dietterich, 2018) setting the methodological precedent. Virtual sensing, by contrast, has no comparable framework. In practice, sensor failures are rarely as simple as additive Gaussian noise or complete signal dropout. They are often subtle, structured, and temporally correlated: corrupted measurements may remain within valid physical ranges while conveying systematically misleading information. While physical sensor fault archetypes are well-catalogued (Balaban et al., 2009; Jesus et al., 2017) and their impact on learned models has been probed in adjacent tasks (Windmann et al., 2025; Dix et al., 2023), virtual sensing itself lacks a multi-domain robustness benchmark. To bridge this gap, we introduce **MuViS-C**, the first standardized multi-domain benchmark of robustness against common sensor failures in learning-based virtual sensing, extending the MuViS benchmark (Brandt et al., 2026) with a systematic robustness evaluation. Our contributions are:

(1) The MuViS-C benchmark. A reproducible evaluation protocol implementing ten sensor failure modes, from subtle drifts to catastrophic signal dropouts, swept across the failure $\times$ channel $\times$ severity grid and aggregated into three robustness measures: mPC and rPC adapted from Michaelis et al. (2020), and s_{cross}, a new measure capturing worst-case fragility. The benchmark is open-source and extensible, supporting custom models, failure modes, measures, and datasets. Code is available at `https://anonymous.4open.science/r/MuViS-C-D536/README.md`.

(2) An empirical study of virtual sensor robustness. Benchmarking six architectures spanning gradient-boosted trees and the major inductive biases for sequence modeling (convolution, recurrence, attention, and MLP-mixing) across nine datasets from six domains, we expose pervasive fragility under structured sensor failures. Model rankings shift substantially across domains, so robustness conclusions drawn from any single domain do not transfer, making multi-domain evaluation necessary for general claims about virtual sensor reliability. On the attention-based architecture, we probe three representative robustification methods spanning input dropout, adversarial training, and purpose-built self-supervised pretraining, establishing reference points and quantifying the trade-off between nominal performance and robustness.

2 Related work

Virtual sensing. Methods for virtual sensing range from mechanistic white-box models, through gray-box hybrids, to fully data-driven black-box approaches that can be deployed without deep domain expertise (Chen et al., 2020). While individual domains like battery state-of-charge estimation, air-quality monitoring, or vehicle dynamics have produced tailored solutions (Ma et al., 2024; Zetterberg, Oskar and Tevell, Axel, 2023; Mondal et al., 2024), cross-domain comparison has been hampered by inconsistent preprocessing, metrics, and data splits. The MuViS benchmark (Brandt et al., 2026) addresses this gap by consolidating datasets from six physical domains into a unified evaluation framework, showing that no single model dominates across domains. However, MuViS evaluates models exclusively under nominal conditions, leaving open how they behave when input sensors degrade.

Sensor failure modeling. Balaban et al. (2009) map physical failure mechanisms across eight common sensor types (thermocouples, RTDs, piezoelectric, piezoresistive, strain gages, Hall effect, magnetostrictive, and LVDTs) to five behavioral fault categories (bias, drift, scaling, noise, and hard faults), showing that these categories are tied to the sensing principle rather than the application domain. Jesus et al. (2017) arrive at a similar taxonomy for wireless sensor networks, identifying

six failure modes (offset, drift, crash, trimming, outliers, and noise) and surveying fusion-based mitigation strategies that exploit spatial, temporal, and value redundancy. Unlike Balaban et al., they define these modes at the sensor interface, characterizing a fault solely by the deviating values a sensor produces and abstracting from its physical cause. The convergence of both taxonomies, reinforced by additional fault-type analyses (Ni et al., 2009) and empirical prevalence studies in deployed sensor networks (Sharma et al., 2010), confirms that the same fault archetypes recur across domains, from aerospace and environmental monitoring to industrial systems, whether viewed through the sensing principle or at the interface. This interface level is precisely where a learning-based virtual sensor observes its inputs, and it also covers faults arising after transduction, in signal conditioning, analog-to-digital conversion, or data transport. Together, these taxonomies supply the physically grounded fault vocabulary that any sensor-robustness benchmark must build on.

Robustness benchmarks. A large body of work on systematic robustness evaluation under realistic input corruptions exists in the computer vision domain. ImageNet-C/P (Hendrycks & Dietterich, 2018) introduced a standardized suite of 15 corruption types at five severity levels and demonstrated large accuracy drops even for state-of-the-art classifiers, catalyzing a body of work on corruption-robust architectures. Analogous benchmarks exist for object detection (Michaelis et al., 2020), segmentation (Kamann & Rother, 2020), and 3D perception (Dong et al., 2023), establishing corruption robustness as a first-class evaluation axis alongside clean accuracy. Recent work extends this perspective to machine learning on industrial time series data, where sensor faults can cause catastrophic degradation in learned virtual sensors (Brandt et al., 2025). Dix et al. (2023) demonstrate similar sensitivity in time series classification, while Windmann et al. (2025) report substantial accuracy losses under injected data-quality issues in forecasting pipelines. Yet virtual sensing lacks what computer vision has long established: a systematic, multi-domain robustness benchmark with physically grounded corruptions.

Robustness improvement methods. A broad landscape of techniques has been proposed to improve model robustness under distributional shift. Adversarial training augments the training set with worst-case perturbations to harden models against adversarial inputs (Szegedy et al., 2014; Goodfellow et al., 2015), with projected gradient descent as a prominent generation method (Madry et al., 2018). Complementary lines of work provide formal stability guarantees (Raghunathan et al., 2018; Cohen et al., 2019) or encourage invariant representations through training on diverse environments (Ganin et al., 2016; Tobin et al., 2017). Representation-learning approaches, from denoising autoencoders (Vincent et al., 2008) to masked autoencoders, learn features that are inherently more tolerant to input corruption. Brandt et al. (2025) extended this idea to sensor data with a self-supervised masking scheme that simulates common sensor faults during pretraining, enabling robust virtual sensing in closed-loop autonomous driving. In the multi-sensor setting, Mena et al. (2024) propose input sensor dropout, randomly masking entire sensor streams during training to improve robustness to missing sensors at inference time.

3 THE MUVIS-C BENCHMARK

We introduce MuViS-C, a multi-domain robustness benchmark for learning-based virtual sensing. It extends the MuViS framework (Brandt et al., 2026) with a systematic robustness evaluation built on physically grounded sensor-fault taxonomies and the corruption-benchmark methodology established in computer vision. We retain its problem formulation: given an input window $\mathbf{X}_i \in \mathbb{R}^{T \times C}$ of C complementary sensor channels over T time steps, a model f estimates a continuous scalar $\hat{y}_i(t_0)$ anchored at a reference time t_0. To assess robustness, we perturb individual channels of $\mathbf{X}_i$ with the sensor failure modes catalogued in Section 3.2 and measure the resulting change in $\hat{y}_i(t_0)$. Following Freiesleben & Grote (2023), we call a model robust if such interventions in its input do not cause changes in its output beyond a given tolerance. Figure 2 gives a high-level overview of the benchmark.

3.1 DATASETS

We evaluate on **nine** datasets spanning six application domains. Each dataset provides N multivariate time-series windows $\mathbf{X} \in \mathbb{R}^{N \times T \times C}$ with scalar targets $\mathbf{y} \in \mathbb{R}^N$. For more details, see Appendix A. From environmental and health sensing, the **Beijing PM10 and PM2.5 Quality** datasets (Zhang et al., 2017) involve estimating particulate matter concentration ($10\,\mu$m and $2.5\,\mu$m) from 9-dimensional meteorological and pollutant time series, while **PPGDalia (HR)** (Reiss et al., 2019a) targets heart rate

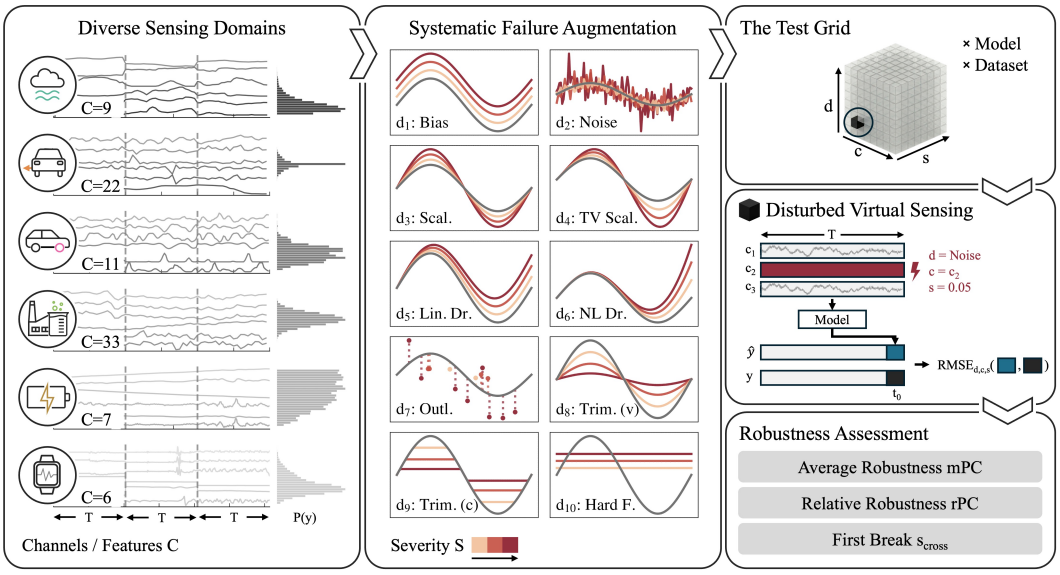


Figure 2: **MuViS-C** evaluates models across diverse sensing domains (left) by applying realistic sensor failures at increasing severity to individual channels (center). At test time, a sliding window (step size 1) feeds a $C \times T$ input tensor to the model, which produces a scalar virtual sensing estimate for t_0. Each test in the failure $\times$ channel $\times$ severity grid (right top) perturbs one channel with a single failure mode at a defined severity (right center). The resulting degraded RMSEs are aggregated into complementary robustness measures (bottom right).

estimation (BPM) from wrist-worn multimodal sensors (BVP, EDA, temperature, and acceleration). In industrial and energy systems, the **Tennessee Eastman Process (TEP)** (Rieth et al., 2017) represents a canonical chemical plant simulation where 33 process variables are used to infer a single chemical concentration, and the **Panasonic 18650PF (Batt.)** dataset (Kollmeyer, 2018) targets battery State-of-Charge (SoC) estimation from current, voltage, and temperature signals across diverse thermal conditions. For automotive and motorsport dynamics, the **REVS Motorsport** dataset (Kegelman et al., 2017) focuses on lateral velocity (v_y) estimation during high-speed racing; we evaluate three separate recording sessions with different car-track-combinations independently **(MMR/T66-13/T66-14)**. Finally, the **Vehicle Dynamics Dataset (Veh.)** (Mori et al., 2025) targets real-time tire temperature (t_{tire}) estimation from 11 channels capturing driver inputs and vehicle states.

3.2 Severity-Parameterized Sensor Failures

Failure catalogue. We define a suite of ten failure modes (see Figure 2, center panel and Table 1), each acting on a single channel of the input window. The modes are taken from a sensor fault taxonomy for virtual sensing (Brandt et al., 2025), which defines fault functions $D(\cdot)$ that map a nominal channel $\mathbf{x} \in \mathbb{R}^T$ to a corrupted channel $\tilde{\mathbf{x}} = D(\mathbf{x})$, with $D(x_t)$ denoting the t-th entry ($t \in \{0, \ldots, T-1\}$). We extend each $D(\cdot)$ into a continuous family $D_{s,k}(\cdot)$ that maps a severity $s \in [0, 1]$ to a perturbation intensity, with $s = 0$ leaving the signal unchanged (or minimally perturbed) and $s = 1$ applying the mode's maximal perturbation, bounded by a scale parameter k.

Design principle. All input data is z-score normalized per channel before evaluation. On this common scale, k acts as a single, dataset-agnostic bound on the maximum perturbation magnitude; we therefore fix k across the benchmark and omit it from the notation hereafter, writing $D_s(\cdot)$. Inspired by the classical 3σ rule used as a naïve threshold for anomaly detection (Chandola et al., 2009), we anchor $k = 3$ so that additive and stochastic perturbations at $s = 1$ reach the magnitude conventionally treated as anomalous. Each evaluation perturbs exactly one input channel at a time while all remaining channels retain their clean values, the standard atomic condition in sensor-fault taxonomies (Balaban et al., 2009; Jesus et al., 2017). This allows per-channel attribution of degradation and enables exhaustive sweeping of the full failure $\times$ channel $\times$ severity grid (Section 3.3).

Table 1: Severity-parameterized sensor failure modes. $p_{\max}$ is the outlier probability at $s = 1$, and α the soft-clip damping factor.

Failure mode	**Definition** $D_s(x_t)$	**Type**
Bias	$D_s(x_t) = x_t + s\,k$	Additive (const.)
Noise	$D_s(x_t) = x_t + \varepsilon_t,\ \varepsilon_t \sim \mathcal{N}(0, (sk)^2)$	Additive (stochastic)
Scaling	$D_s(x_t) = x_t\big(1 + s(k-1)\big)$	Multipl. (const.)
Time-varying scaling	$D_s(x_t) = x_t\big(1 + s(k-1)\frac{t}{T-1}\big)$	Multipl. (time-var.)
Linear drift	$D_s(x_t) = x_t + sk\frac{t}{T-1}$	Drift (linear)
Non-linear drift	$D_s(x_t) = x_t + sk\big(\frac{t}{T-1}\big)^2$	Drift (quadratic)
Outliers	$D_s(x_t) = x_t + m_t\,\delta_t, \quad m_t \sim \text{Ber}(sp_{\max}),\ \delta_t \sim U(-k, k)$	Sporadic spikes
Trimming (var.)	$h = (1-s)k, \quad D_s(x_t) = \begin{cases} -h + \alpha(x_t + h) & x_t < -h \\ h + \alpha(x_t - h) & x_t > h \\ x_t & \lvert x_t\rvert \le h \end{cases}$	Saturation (soft)
Trimming (const.)	$h = (1-s)k,\ D_s(x_t) = \text{clip}(x_t, -h, h)$	Saturation (hard)
Hard fault	$D_s(x_t) = s\,k$	Stuck-at

Defaults: $k = 3,\ p_{\max} = 0.3,\ \alpha = 0.4$.

3.3 Evaluation protocol

For every (dataset, model) pair, failure mode $d \in \mathcal{D}$, and affected input channel $c \in \mathcal{C}$, we sweep severity s over the grid $\mathcal{S} = \{0.05, 0.10, \ldots, 1.0\}$ of $|\mathcal{S}| = 20$ equidistant steps of size $\Delta s = 0.05$, recording the corrupted $\text{RMSE}_{d,c,s}$ at each grid point. All RMSE values are bootstrap means over 200 resamples of the test set. Two additional reference points anchor our evaluation: $\text{RMSE}_{\text{clean}}$, the model's nominal RMSE on the unperturbed test set; and $\text{RMSE}_{\text{naive}}$, the RMSE of a severity-invariant baseline that always predicts the training-set mean and depends only on the dataset. We measure robustness with three metrics: the mean Performance under Corruption (mPC) and the relative Performance under Corruption (rPC) from Michaelis et al. (2020), and the Baseline Crossing Severity (s_{cross}) we propose to capture worst-case fragility. All three are defined below.

Mean performance under corruption (mPC). mPC is the average $\text{RMSE}_{d,c,s}$ over the corruption grid:

$$\text{mPC} = \frac{1}{|\mathcal{D}|\,|\mathcal{C}|\,|\mathcal{S}|} \sum_{d \in \mathcal{D}} \sum_{c \in \mathcal{C}} \sum_{s \in \mathcal{S}} \text{RMSE}_{d,c,s}\,. \tag{1}$$

It instantiates the original mPC with RMSE as the performance metric, and additionally averages over the affected channel c. mPC is reported in the original units of the target variable and is therefore not directly comparable across datasets.

Relative performance under corruption (rPC). mPC is influenced by a model's nominal performance: a stronger model enters the severity sweep at a lower mPC floor. To measure *degradation* independently of where a model starts, we normalize mPC by $\text{RMSE}_{\text{clean}}$:

$$\text{rPC} = \frac{\text{mPC}}{\text{RMSE}_{\text{clean}}}\,. \tag{2}$$

$\text{rPC} = 1$ indicates no degradation on average, and values above 1 signal performance loss. This metric plays the same role as the Relative mCE (Hendrycks & Dietterich, 2018), isolating degradation from nominal accuracy and is the reciprocal of the robustness score of Windmann et al. (2025) (see Appendix F). A complementary normalization of mPC against the naïve mean predictor, the normalized Performance under Corruption (nPC), is defined and reported in Appendix E.

Baseline crossing severity (s_{cross}). Both mPC and rPC summarize *average-case* behavior. To capture *worst-case* fragility we ask: at what severity does the model first become no better than the naïve mean predictor? s_{cross} is the earliest severity at which *any* $\text{RMSE}_{d,c,s}$ is no better than $\text{RMSE}_{\text{naive}}$:

$$s_{\text{cross}} = \min_{d \in \mathcal{D},\, c \in \mathcal{C}} \ \min\big\{ s \in \mathcal{S} \;\big|\; \text{RMSE}_{d,c,s} \ge \text{RMSE}_{\text{naive}} \big\}\,. \tag{3}$$

If no crossing is observed within the measured range, s_{cross} is undefined and denoted by "–" in tables. Unlike mPC and rPC, which average over the (d, c, s) grid, s_{cross} is a minimum, since averaging would mask the single worst-case combination it is designed to detect.

Cross-dataset aggregation. We summarize mPC and rPC across datasets in two complementary ways. Following Fleming & Wallace (1986), we use the geometric mean; rPC enters it directly as a dimensionless ratio, whereas mPC is first normalized per dataset by the best model (1.0 = best). Since rPC averages over the affected channel c and datasets differ in their number of channels, rPC magnitudes are not directly comparable across datasets. We therefore treat the geometric mean as indicative and report mean ranks following Demšar (2006) as our primary aggregate; statistical testing details are deferred to Appendix J. For s_{cross} we report only the mean rank, since no crossing has no finite value; it is encoded as a sentinel above 1, so such entries tie for the best rank.

Complementarity. The three metrics are designed to be read together: mPC measures absolute error under corruption, rPC isolates degradation from nominal performance, and s_{cross} captures worst-case fragility. A model with low mPC but high rPC is accurate yet fragile; low rPC but high mPC indicates robustness without strong nominal performance; low rPC but low s_{cross} reveals smooth average degradation masking an early collapse on the weakest combination. Reporting all three prevents any single metric from hiding the tradeoff between nominal performance and robustness. Additional diagnostics in Appendix O reveal whether a model's degradation concentrates on individual channels.

3.4 Standardized interface

Listing 1 illustrates the core evaluation workflow. Users instantiate a `Testbed` with a dataset, register one or more models, and optionally extend the default failure suite or add custom metrics. A single call to `run()` sweeps the full evaluation grid and returns a `Results` object for inspection and export. A detailed description of the programming interface is provided in Appendix P.

```python
import muvis_c as rob

def my_predict_fn(X: np.ndarray) -> np.ndarray: ...  # (N,T,C) -> (N,)

testbed = rob.Testbed(dataset=rob.PPGDalia)
testbed.add_model("MyModel", predict_fn=my_predict_fn)
results = testbed.run()
results.summary()
```

Listing 1: Evaluating a custom virtual sensing model with MuViS-C.

4 Experiments and results

With MuViS-C in place, we conduct a two-part empirical study. The first part characterizes how state-of-the-art machine learning models behave under sensor failure. We evaluate six baseline architectures encompassing gradient-boosted tree ensembles and representative neural inductive biases. **XGBoost** (Chen & Guestrin, 2016) and **Catboost** (Prokhorenkova et al., 2019) are trained on flattened inputs $\mathbb{R}^{N\times(T\cdot C)}$. For sequential processing, **xLSTM-Mixer** (Kraus et al., 2025) is a recurrent architecture that stacks scalar-memory sLSTM blocks with exponential gating to jointly mix temporal and cross-variate information by reconciling original and reversed sequence views. **TST** (Zerveas et al., 2021) applies a standard Transformer encoder where each token corresponds to a full time-step vector, enabling multi-head self-attention across the temporal dimension. **PatchTSMixer** (Ekambaram et al., 2023) segments time series into patches and applies lightweight MLP-Mixer blocks to mix information across patches, channels, and hidden features, capturing both temporal patterns and cross-variate correlations. **ModernTCN** (Luo & Wang, 2024) modernizes temporal convolutional networks by adopting a Transformer-block-like layout that decouples temporal and feature mixing. All sequential models operate on inputs $\mathbf{X} \in \mathbb{R}^{N\times T\times C}$.

The baselines reveal where current models break; the second part of our study asks whether established robustification techniques can mitigate these failures, and at what cost. We evaluate three

representative strategies: input dropout (**ISensD** Mena et al. (2024)), purpose-built self-supervised pretraining (**F2F** Brandt et al. (2025)), and adversarial training (**PGD** Madry et al. (2018)). ISensD randomly masks entire sensor channels during training to improve robustness to missing sensors at inference. PGD frames robustness as a min-max optimization, generating adversarial perturbations via multi-step Projected Gradient Descent during training. F2F is a self-supervised pretraining scheme in which an encoder is trained to reconstruct channels corrupted by realistic sensor failure modes; a virtual sensor is then trained on the frozen robust embedding. All three are applied on top of the TST backbone: F2F's published implementation is built on a TST encoder, so fixing TST as the shared backbone enables a fair head-to-head comparison. TST is also mid-pack on nominal performance (Appendix D) and among the most fragile baselines on rPC (Table 3), leaving clear headroom for robustification gains without ceiling effects. Full training, tuning, and compute details are reported in Appendix B.

Tables 2 and 3 report mPC and rPC, respectively; Table 4 reports s_{cross}. Omnibus tests reach significance for mPC and rPC (Friedman, both $p < 0.001$) but not for s_{cross} (Friedman, $p = 0.187$). A naïve-normalized variant (nPC) is reported in Appendix E for interpretability. Full statistical details, including all p-values, critical distances, and post-hoc cluster memberships, are given in Appendix J.

Table 2: mPC per model and dataset. Bold = best, underline = second-best per column. GM is the geometric mean of best-normalized values (1.0 = best); arrows show GM change vs. TST.

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14	Rank	GM
XGBoost	**97.12**	66.32	0.05	12.38	**0.05**	**4.44**	0.17	0.12	**0.11**	**2.3**	1.202
CatBoost	98.81	67.51	0.04	11.46	**0.05**	5.00	**0.16**	**0.11**	**0.11**	2.6	**1.162**
M-TCN	98.73	69.00	0.08	**5.27**	0.06	11.39	0.22	0.14	0.15	5.6	1.403
P-TSMixer	107.16	74.35	0.09	10.25	**0.05**	7.92	0.34	0.21	0.23	7.2	1.705
xLSTM-M	98.33	67.26	0.05	9.65	**0.05**	4.47	0.18	0.12	0.13	3.0	1.209
TST	106.23	73.51	0.06	12.52	**0.05**	8.17	0.21	0.14	0.16	6.0	1.476
TST-PGD	101.39	**65.98**	0.07	15.36	**0.05**	4.57	0.25	0.15	0.16	5.6	1.430 ↓-0.046
TST-ISensD	108.56	74.41	0.11	15.33	**0.05**	5.66	0.24	0.17	0.19	7.7	1.618 ↑+0.142
TST-F2F	118.44	68.75	**0.03**	23.53	0.07	4.96	0.18	0.12	0.12	5.1	1.318 ↓-0.158

Table 3: rPC per model and dataset. Bold = best, underline = second-best per column. GM is the geometric mean across datasets; arrows indicate GM change relative to TST.

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14	Rank	GM
XGBoost	1.060	1.087	**2.159**	1.301	1.007	1.187	1.395	1.211	1.425	3.4	1.281
CatBoost	1.073	1.100	3.669	1.270	1.007	1.302	1.517	1.263	1.353	4.1	1.385
M-TCN	1.086	1.149	9.200	2.187	1.059	4.837	1.853	1.854	1.896	8.1	2.112
P-TSMixer	1.043	1.108	2.613	**1.099**	**1.006**	2.043	1.247	1.509	1.559	3.8	1.395
xLSTM-M	**1.042**	**1.061**	5.550	1.309	1.007	1.178	1.777	1.645	1.634	4.7	1.529
TST	1.091	1.155	3.761	1.817	1.010	2.157	1.897	1.858	1.948	7.9	1.716
TST-PGD	1.070	1.066	2.833	1.175	**1.006**	1.324	**1.182**	**1.108**	1.147	**2.7**	**1.253** ↓-0.463
TST-ISensD	1.134	1.105	4.129	1.272	1.011	1.217	1.525	1.608	1.736	6.2	1.482 ↓-0.234
TST-F2F	1.095	1.102	2.208	1.521	1.231	**1.159**	1.202	1.126	**1.113**	4.1	1.272 ↓-0.444

4.1 Gradient-boosted trees as a strong robustness baseline

Gradient-boosted trees hold up well under corruption without any dedicated defense. On mPC (Table 2), XGBoost achieves the best mean rank (2.3) with CatBoost a close second (2.6), both significantly outranking P-TSMixer and TST-ISensD. On rPC, XGBoost again leads all non-robustified models, trailing only the adversarially robustified TST-PGD (Table 3). XGBoost is in fact the only model in the benchmark that holds an exclusive top-cluster position on both statistics (Appendix J). No neural architecture reaches the joint top position, even with dedicated robustification. This

suggests that the inductive biases of gradient-boosted trees confer an implicit corruption resilience that neural architectures in our benchmark do not replicate, setting a strong baseline to beat.

4.2 TRADE-OFF BETWEEN NOMINAL PERFORMANCE AND ROBUSTNESS

Figure 3 plots each model's clean-data error against its rPC, both as geometric means across datasets. The clean-data axis is normalized per dataset against the best model, so 1.0 marks the best achievable nominal performance and the optimal corner sits at the bottom-left where a model is both accurate and degrades gracefully. M-TCN, the strongest model nominally (Appendix D), is the least robust, with an rPC (Table 3) of 2.112 and the worst mean rank (8.1), significantly outranked by TST-PGD, XGBoost, and P-TSMixer. P-TSMixer shows the inverse pattern: among non-robustified baselines, it ranks among the weakest on clean-data error yet sits in the top non-significant cluster on rPC (Figure 12), degrading gracefully from a poor nominal baseline. Among non-robustified models, xLSTM-M comes closest to the tree ensembles on both axes. It is also the only one to stay below the naïve-predictor threshold on PM10 and HR; all other non-robustified models cross on every dataset (Table 4).

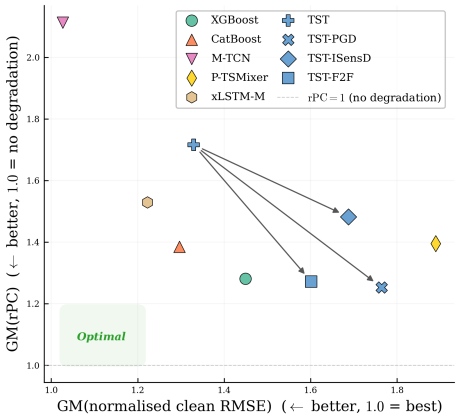


Figure 3: Nominal performance vs. robustness. GM(rPC) aggregated across datasets (y) vs. GM of normalized clean RMSE (x, normalized following mPC aggregation). Arrows indicate the effect of robustification.

A genuine trade-off emerges in robustification training. The arrows in Figure 3 trace each robustified TST variant's position relative to vanilla TST, and all three point toward lower rPC but higher clean-data error: every robustification method we evaluated trades nominal performance for robustness. Statistically, vanilla TST suffers a significant rPC deficit relative to TST-PGD, XGBoost, and P-TSMixer, and all three robustification strategies close it. Every robustified variant joins the leading non-significant rPC cluster, and TST-PGD significantly outranks vanilla TST itself, though none establishes a significant advantage over the strongest non-robustified baselines (Figure 12).

Table 4: Baseline crossing severity s_{cross}. "–" denotes never crosses within the measured severity range (best possible). Bold = best, underline = second-best per column.

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14	Rank
XGBoost	0.500	0.400	0.100	0.500	0.250	0.050	0.150	0.100	0.150	5.6
CatBoost	0.550	0.350	0.100	0.950	0.300	0.100	0.100	0.150	0.100	4.9
M-TCN	0.450	0.600	0.450	0.450	0.050	0.050	0.050	0.100	0.050	5.9
P-TSMixer	0.650	0.450	0.400	0.050	0.300	0.050	0.050	0.050	0.050	6.2
xLSTM-M	–	0.550	0.600	–	0.300	0.100	0.150	0.150	0.050	**3.5**
TST	0.250	0.200	0.450	0.500	0.500	0.050	0.050	0.050	0.050	6.2
TST-PGD	0.050	0.400	0.400	0.450	**0.800**	0.050	0.400	0.600	0.450	4.4
TST-ISensD	0.300	0.350	0.200	0.300	0.350	**0.200**	0.400	0.400	0.350	4.4
TST-F2F	0.200	**0.950**	–	0.050	0.050	0.100	–	–	–	3.7

4.3 BENCHMARK DIVERSITY ACROSS DATASETS

The nine datasets span a wide difficulty spectrum, validating the multi-domain design. On rPC (Table 3), TEP appears easiest, with six of nine models staying within 1% of clean performance, yet the channel-level diagnostics reveal localized fragility behind this average (Appendix O). Battery, in contrast, is universally challenging, with every model at least doubling its clean error. Crucially, model rankings shift across datasets: M-TCN achieves the best mPC (Table 2) on heart rate but the

worst on Vehicle Dynamics; TST-F2F dominates Batt. and REVS on s_{cross} yet collapses at the lowest severity on HR and TEP (Table 4). No single dataset captures the full picture, and a benchmark restricted to any one domain would miss failure patterns that are prominent in others.

4.4 Additional analyses

Per-failure-mode heterogeneity. Disaggregating rPC by failure mode (Appendix G, Table 11) reveals that corruption difficulty varies substantially: the geometric-mean rPC across models ranges from 1.073 for varying trimming to 1.922 for hard faults, with hard faults producing roughly an order of magnitude more degradation than the easiest mode. All models agree on the easiest mode (varying trimming) and all but TST-ISensD and TST-F2F agree on the hardest (hard fault); for those two, noise is the worst case. Although models broadly agree at the extremes of difficulty, the best model varies mode by mode, and the per-mode winner is always either XGBoost or one of the robustified TST variants: TST-F2F leads on bias, hard faults, and linear and nonlinear drift; TST-ISensD leads on both trimming variants; TST-PGD wins on both scaling variants; and XGBoost leads on outliers and noise. No model ranks best on more than four of the ten failure modes, reinforcing the value of multi-failure evaluation.

Robustness under detectable sensor faults. When a reliable failure detector is available and the operator or a system can replace a faulty channel with its training-set mean (hard fault at $s = 0$), TST-ISensD achieves a geometric-mean rPC of 1.034 and a perfect mean rank of 1.0 across all nine datasets, indicating virtually no degradation and significantly outranking CatBoost, xLSTM-M, TST, and M-TCN (Appendix H). This result follows directly from ISensD's training procedure, which randomly zeroes out input channels and thus explicitly prepares the model for this intervention. The finding suggests that pairing a simple input-dropout augmentation with a failure detection mechanism is a practical strategy for maintaining prediction quality under sensor faults.

5 Limitations and future work

First, our ten-mode failure catalogue, grounded in established sensor-fault taxonomies, covers the dominant archetypes recurring across domains but cannot be exhaustive; MuViS-C's injection framework is designed to absorb new modes as they are characterized, and we plan to broaden the catalogue as the community surfaces additional failure patterns from field deployments. Second, we evaluate three representative robustification methods on a shared TST backbone to enable a fair head-to-head comparison with F2F; broader families of defenses, including techniques from the vision-robustness literature, and the question of whether our findings transfer across backbones remain open, and we plan to address both in future work using MuViS-C as the shared testbed. Third, MuViS-C evaluates virtual sensors in open loop, measuring estimation error rather than its downstream consequences. This keeps the benchmark model-agnostic and applicable to recorded datasets, but in deployed systems estimates feed into controllers or monitoring logic, where the impact of a given error depends on the downstream task. Coupling MuViS-C with closed-loop simulation environments to capture these system-level effects is an important direction for future work.

6 Conclusion

MuViS-C provides a testbed for evaluating learning-based virtual sensor reliability under realistic sensor failures. Its multi-domain design proves essential: datasets span a wide difficulty spectrum, model rankings shift across domains, and no single dataset captures the full picture. We find that (i) every model evaluated degrades substantially under corruption, becoming worse than a naïve predictor on at least one failure-mode/channel combination; (ii) gradient-boosted tree ensembles achieve strong robustness without any explicit defense, setting a challenging baseline for neural methods; (iii) dedicated robustification training can elevate a fragile neural baseline into the leading robustness cluster, though every method incurs a clean-data error penalty; and (iv) when reliable failure detection and input intervention are available, a simple input-dropout augmentation (ISensD) reduces average degradation to near zero. We release MuViS-C as an open-source, extensible platform and welcome contributions of new datasets, failure modes, measures, and models.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 2623–2631, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330701. URL https://dl.acm.org/doi/10.1145/3292500.3330701.

Edward Balaban, Abhinav Saxena, Prasun Bansal, Kai F. Goebel, and Simon Curran. Modeling, Detection, and Disambiguation of Sensor Faults for Aerospace Applications. *IEEE Sensors Journal*, 9(12):1907–1917, December 2009. ISSN 1558-1748. doi: 10.1109/JSEN.2009.2030284. URL https://ieeexplore.ieee.org/abstract/document/5297818. Conference Name: IEEE Sensors Journal.

Jens U. Brandt, Noah C. Pütz, Marcus Greiff, Thomas Jonathan Lew, John Subosits, Marc Hilbert, and Thomas Bartz-Beielstein. From Faults to Features: Pretraining to Learn Robust Representations against Sensor Failures. October 2025. URL https://openreview.net/forum?id=9aElHWiZ72.

Jens U. Brandt, Noah C. Puetz, Jobel Jose George, Niharika Vinay Kumar, Elena Raponi, Marc Hilbert, Thomas Bäck, and Thomas Bartz-Beielstein. MuViS: Multimodal Virtual Sensing Benchmark, March 2026. URL http://arxiv.org/abs/2603.24602. arXiv:2603.24602 [eess].

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL https://dl.acm.org/doi/10.1145/1541880.1541882.

Jiayao Chen, Weihua Gui, Ning Chen, Jiayang Dai, Chunhua Yang, and Xu Li. A Dynamic Grey-Box Model and its Application in the Sintering Process of Ternary Cathode Material. *IFAC-PapersOnLine*, 53(2):11866–11871, 2020. ISSN 24058963. doi: 10.1016/j.ifacol.2020.12.700. URL https://linkinghub.elsevier.com/retrieve/pii/S2405896320310144.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1310–1320. PMLR, May 2019. URL https://proceedings.mlr.press/v97/cohen19c.html.

Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006. ISSN 1533-7928. URL http://jmlr.org/papers/v7/demsar06a.html.

Marcel Dix, Gianluca Manca, Kenneth Chigozie Okafor, Reuben Borrison, Konstantin Kirchheim, Divyasheel Sharma, Kr Chandrika, Deepti Maduskar, and Frank Ortmeier. Measuring the Robustness of ML Models Against Data Quality Issues in Industrial Time Series Data. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pp. 1–8, July 2023. doi: 10.1109/INDIN51400.2023.10218129. URL https://ieeexplore.ieee.org/document/10218129/. ISSN: 2378-363X.

Yinpeng Dong, Caixin Kang, Jinlai Zhang, Zijian Zhu, Yikai Wang, Xiao Yang, Hang Su, Xingxing Wei, and Jun Zhu. Benchmarking Robustness of 3D Object Detection to Common Corruptions in Autonomous Driving. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1022–1032, Vancouver, BC, Canada, June 2023. IEEE. ISBN 979-8-3503-0129-8. doi: 10.1109/CVPR52729.2023.00105. URL https://ieeexplore.ieee.org/document/10205484/.

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 459–469, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 979-8-4007-0103-0. doi: 10.1145/3580305.3599533. URL https://doi.org/10.1145/3580305.3599533.

Philip J. Fleming and John J. Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Commun. ACM*, 29(3):218–221, 1986. ISSN 0001-0782. doi: 10.1145/5666.5673. URL https://dl.acm.org/doi/10.1145/5666.5673.

Timo Freiesleben and Thomas Grote. Beyond generalization: a theory of robustness in machine learning. *Synthese*, 202(4):109, September 2023. ISSN 1573-0964. doi: 10.1007/s11229-023-04334-9. URL https://doi.org/10.1007/s11229-023-04334-9.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, January 2016. ISSN 1532-4435.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, March 2015. URL http://arxiv.org/abs/1412.6572. arXiv:1412.6572 [stat].

Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. September 2018. URL https://openreview.net/forum?id=HJz6tiCqYm.

Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved Problems in ML Safety, June 2022. URL http://arxiv.org/abs/2109.13916. arXiv:2109.13916 [cs].

Steffen Herbold. Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48):2173, April 2020. ISSN 2475-9066. doi: 10.21105/joss.02173. URL https://joss.theoj.org/papers/10.21105/joss.02173.

Gonçalo Jesus, António Casimiro, and Anabela Oliveira. A Survey on Data Quality for Dependable Monitoring in Wireless Sensor Networks. *Sensors*, 17(9):2010, September 2017. ISSN 1424-8220. doi: 10.3390/s17092010. URL https://www.mdpi.com/1424-8220/17/9/2010. Number: 9.

Christoph Kamann and Carsten Rother. Benchmarking the Robustness of Semantic Segmentation Models. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8825–8835, June 2020. doi: 10.1109/CVPR42600.2020.00885. URL http://arxiv.org/abs/1908.05005. arXiv:1908.05005 [cs].

John C. Kegelman, Lene K. Harbott, and J. Christian Gerdes. Insights into vehicle trajectories at the handling limits: analysing open data from race car drivers. *Vehicle System Dynamics*, 55(2):191–207, February 2017. ISSN 0042-3114. doi: 10.1080/00423114.2016.1249893. URL https://doi.org/10.1080/00423114.2016.1249893. _eprint: https://doi.org/10.1080/00423114.2016.1249893.

Phillip Kollmeyer. Panasonic 18650PF Li-ion Battery Data. 1, June 2018. doi: 10.17632/wykht8y7tg.1. URL https://data.mendeley.com/datasets/wykht8y7tg/1.

Maurice Kraus, Felix Divo, Devendra Singh Dhami, and Kristian Kersting. xLSTM-Mixer: Multivariate Time Series Forecasting by Mixing via Scalar Memories. October 2025. URL https://openreview.net/forum?id=JlVn0XRpy0.

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

Fangyuan Ma, Cheng Ji, Jingde Wang, Wei Sun, and Ahmet Palazoglu. Soft Sensor Modeling Method Considering Higher-Order Moments of Prediction Residuals. *Processes*, 12(4), March 2024. ISSN 2227-9717. doi: 10.3390/pr12040676. URL https://www.mdpi.com/2227-9717/12/4/676.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. February 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Dominik Martin, Niklas Kühl, and Gerhard Satzger. Virtual Sensors. *Business & Information Systems Engineering*, 63(3):315–323, June 2021. ISSN 1867-0202. doi: 10.1007/s12599-021-00689-w. URL https://doi.org/10.1007/s12599-021-00689-w.

Francisco Mena, Diego Arenas, and Andreas Dengel. Increasing the Robustness of Model Predictions to Missing Sensors in Earth Observation, September 2024. URL http://arxiv.org/abs/2407.15512. arXiv:2407.15512 [cs].

Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S. Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming, March 2020. URL http://arxiv.org/abs/1907.07484. arXiv:1907.07484 [cs].

Pratik Mondal, Divyakumar Bhavsar, Kanupriya Mittal, and Mayank Mittal. Estimating State-of-Charge in Lithium-Ion Batteries Through Deep Learning Techniques: A Comparative Evaluation. *IEEE Access*, PP:1–1, January 2024. doi: 10.1109/ACCESS.2024.3408220.

Daiki Mori, Rajan K. Aggarwal, Nicholas D. Broadbent, Takao Kobayashi, and J. Christian Gerdes. Vehicle Dynamics Dataset for Highly Dynamic Automated Driving. 2025. doi: 10.25740/hh613qz0317. URL https://purl.stanford.edu/hh613qz0317.

Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava. Sensor network data fault types. *ACM Trans. Sen. Netw.*, 5(3), June 2009. ISSN 1550-4859. doi: 10.1145/1525856.1525863. URL https://doi.org/10.1145/1525856.1525863.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features, January 2019. URL http://arxiv.org/abs/1706.09516. arXiv:1706.09516 [cs].

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified Defenses against Adversarial Examples. February 2018. URL https://openreview.net/forum?id=Bys4ob-Rb.

Attila Reiss, Ina Indlekofer, and Philip Schmidt. PPG-DaLiA, 2019a. Published: UCI Machine Learning Repository.

Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven. Deep PPG: Large-Scale Heart Rate Estimation with Convolutional Neural Networks. *Sensors*, 19(14), July 2019b. ISSN 1424-8220. doi: 10.3390/s19143079. URL https://www.mdpi.com/1424-8220/19/14/3079.

Cory A. Rieth, Ben D. Amsel, Randy Tran, and Maia B. Cook. Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation, July 2017. URL https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/6C3JR1.

Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Transactions on Sensor Networks*, 6(3):1–39, June 2010. ISSN 1550-4859, 1550-4867. doi: 10.1145/1754414.1754419. URL https://dl.acm.org/doi/10.1145/1754414.1754419.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, February 2014. URL http://arxiv.org/abs/1312.6199. arXiv:1312.6199 [cs].

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. IEEE Press, 2017. doi: 10.1109/IROS.2017.8202133. URL https://doi.org/10.1109/IROS.2017.8202133.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pp. 1096–1103, New York, NY, USA, July 2008. Association for Computing Machinery. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390294. URL https://dl.acm.org/doi/10.1145/1390156.1390294.

Alexander Windmann, Henrik Steude, Daniel Boschmann, and Oliver Niggemann. Quantifying Robustness: A Benchmarking Framework for Deep Learning Forecasting in Cyber-Physical Systems, 2025. URL https://arxiv.org/abs/2504.03494. Version Number: 3.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A Transformer-based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 2114–2124, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467401. URL https://doi.org/10.1145/3447548.3467401.

Zetterberg, Oskar and Tevell, Axel. Creating a Virtual Tyre Temperature Sensor, 2023. ISSN: 1404-6342 Series: Master's Theses in Mathematical Sciences.

Shuyi Zhang, Bin Guo, Anlan Dong, Jing He, Ziping Xu, and Song Xi Chen. Cautionary tales on air-quality improvement in Beijing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205):20170457, September 2017. ISSN 1364-5021. doi: 10.1098/rspa.2017.0457. URL https://doi.org/10.1098/rspa.2017.0457.

REPRODUCIBILITY STATEMENT

MuViS-C is released as an open-source package, together with all code needed to reproduce the main results in this paper, at `https://anonymous.4open.science/r/MuViS-C-D536/README.md`. The ten failure modes are defined in Table 1, including their default parameters. The severity grid, bootstrap procedure, and robustness measures are specified in Section 3.3, and the statistical testing protocol is in Appendix J. All reported metrics are computed directly by the released `Testbed`. All nine datasets are publicly available and use the unmodified MuViS splits and preprocessing (Appendix A). Appendix B describes the baseline adaptations, training settings, robustification methods, and hyperparameter search spaces. The final configurations for all checkpoints ship with the code. All runs use a fixed seed (42). Full reproduction takes about 155 GPU hours on a single NVIDIA H100, though runs parallelize across (model, dataset) pairs. Appendix P documents the interface for evaluating new models, datasets, failure modes, and metrics.

LARGE LANGUAGE MODEL USAGE

In this work, we used generative AI tools as a general-purpose writing and coding assistant. We additionally used generative AI tools for providing feedback on research methodology and experiments, implementing methods, and supporting qualitative and thematic data analysis. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, proposing or refining hypotheses, designing research methodology or experiments, assisting with translation, cleaning and reformatting datasets, or interpreting results, and generating synthetic datasets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, creation of artifacts, drafting parts of the research paper, summarizing and analyzing existing literature, brainstorming, sourcing and searching for information, editing the research paper to improve readability, and identifying relevant literature. We have reviewed all AI-assisted work: we manually polished all AI-generated text, and we verified and tested all AI-generated code for correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

A DATA AND PREPROCESSING

The dataset selection, train/test splits, and preprocessing pipeline (cleaning, resampling, channel selection, and sliding-window construction) are taken unchanged from the MuViS benchmark (Brandt et al., 2026); we refer the reader to that paper for the full design rationale and summarize the resulting dataset statistics in Table 5.

Table 5: Benchmark datasets. C: features, T: sequence length, N: number of samples.

Dataset	**Domain**	C	T	N_{train}	N_{test}
BeijingPM10Quality	Air quality	9	24	11,918	5,048
BeijingPM25Quality	Air quality	9	24	11,918	5,048
Panasonic18650PFData	Battery SoC	7	120	199,827	158,126
PPGDalia	Wearable / Bio	6	512	51,757	12,940
REVS/2013 Monterey	Motorsport	22	20	120,777	21,357
REVS/2013 Targa 66	Motorsport	22	20	33,520	11,109
REVS/2014 Targa 66	Motorsport	22	20	46,739	9,036
TennesseeEastman	Chemical	33	20	240,500	470,500
VehicleDynamics	Automotive	11	50	1,384	280

B IMPLEMENTATION DETAILS

Each of the nine model classes was trained on each of the nine benchmark datasets, yielding 81 (model $\times$ dataset) checkpoints used throughout the evaluation. All hyperparameters are reported in the accompanying repository.

B.1 BASELINES

All neural networks consume tensors in the (N, T, C) layout and emit a single scalar per window. Apart from TST, which targets representation learning for regression and classification, the architectures were originally proposed for multivariate forecasting and required minor adjustments for the scalar virtual-sensing setting.

TST. Used as-is with the built-in linear regression head producing a single output.

ModernTCN. Instantiated in regression mode with a single output unit; RevIN is disabled. Inputs are transposed to (N, C, T) to match the original implementation.

xLSTM-Mixer. The forecast horizon is set to zero, and predictions are obtained through a regression pathway (mirroring the model's classification adaptation in the released code) that flattens the post-mixer tokens through a single linear head.

PatchTSMixer. The Hugging Face implementation of `PatchTSMixerForRegression` is used with `num_targets=1`, `mode="common_channel"`, and `head_aggregation="use_last"`.

B.2 OPTIMIZATION

All neural networks are trained for 100 epochs with Adam, batch size 256, and MSE loss. Inputs are z-score normalized per feature using statistics estimated on the 90% training portion of an additional 90/10 train/val split. Checkpoints are selected by best validation loss, and test metrics are reported on the predefined temporal test split. All runs use seed 42.

B.3 ROBUSTIFICATION METHODS

F2F (self-supervised pretraining + fine-tuning). The encoder is pretrained for 600 epochs with a multi-task masked-reconstruction objective. For each batch, samples are split into three disjoint subsets, and the positions selected by a geometric binary mask are corrupted by (i) zeroing (mean replacement on z-scored data), (ii) additive uniform bias, or (iii) additive Gaussian noise; the three task losses are summed with equal weights. Pretraining uses AdamW with separated decay/no-decay parameter groups, a cosine schedule with linear warmup, gradient clipping (after warmup), and zero weight decay. A two-layer MLP projection head maps the flattened encoder output $(T \cdot d_{\text{model}})$ back to the reconstruction target. After pretraining, the projection head is discarded, the regression head is reinitialized, the backbone is frozen, and only the head is fine-tuned for 100 epochs with the same supervised setup as the TST baseline.

PGD (adversarial training). Each batch is split into a clean portion and an adversarial portion, where the adversarial fraction ρ is tuned per dataset. Adversaries are generated with PGD under an ℓ_∞ budget for a fixed number of iterations, initialized from a random point inside the ϵ-ball. All other optimization settings match the TST baseline.

ISensD (input sensor dropout). For each training sample, a non-empty subset of input channels is sampled uniformly from the $2^C - 1$ possibilities and the remaining channels are zeroed before the forward pass. The model is therefore trained over the full distribution of sensor-availability patterns rather than a fixed configuration. Optimization is otherwise identical to the TST baseline.

B.4 COMPUTE RESOURCES

All experiments were implemented in Python 3.13.7 and executed on a single NVIDIA H100 (80 GB) GPU with CUDA 13.2. To accelerate evaluation, compatible models were compiled with `torch.compile` and inference was run under `torch.autocast` (`bfloat16`). Full reproduction of the paper, covering training for all 81 (model $\times$ dataset) configurations followed by the complete severity sweep, takes approximately 155 hours of GPU time on a single device, though training and evaluation are embarrassingly parallel across (model, dataset) configurations and can

be spread across multiple GPUs for substantial wall-clock speedups. Hyperparameter tuning and preliminary experiments leading up to the final results required considerably more compute.

B.5 Hyperparameter selection

B.5.1 Baseline architectures

Hyperparameters for XGBoost and CatBoost were adopted directly from the MuViS benchmark (Brandt et al., 2026). For the remaining baselines (TST, ModernTCN, PatchTSMixer, xLSTMMixer), we followed a comparable protocol: Optuna (Akiba et al., 2019) with multivariate TPE sampling, 100 trials per architecture and dataset, minimizing validation RMSE on a 10 % subsample of the training data. Where available, the first trial was initialized with author-recommended defaults. Table 6 lists the search spaces.

Table 6: Search spaces for baseline architectures. Brackets denote continuous ranges; braces denote categorical choices.

Model	Parameter	Range / Choices
TST	d_{model}	{64, 128, 256}
	n_{heads}	{4, 8, 16}
	num_layers	[1, 4]
	dim_feedforward	{128, 256, 512}
	dropout	[0.0, 0.2]
	learning rate	$[10^{-5}, 10^{-3}]$ (log)
ModernTCN	d_{model}	{32, 64, 128}
	num_blocks	[1, 4]
	large_kernel	{7, 13, 25, 31, 51, 71}
	small_kernel[†]	{3, 5, 7, 13, 25}
	ffn_ratio	{1, 2, 4}
	dropout / head_dropout	[0.0, 0.3]
	learning rate	$[10^{-5}, 10^{-2}]$ (log)
PatchTSMixer	patch_length[‡]	{2, 4, 5, 8, 10, 16, 20, 32, 64}
	d_{model}	{32, 64, 128}
	num_layers	[2, 8]
	expansion_factor	{2, 4}
	dropout / head_dropout	[0.0, 0.3]
	learning rate	$[10^{-5}, 10^{-2}]$ (log)
xLSTMMixer	embedding_dim	{128, 256, 512}
	num_heads	{4, 8, 16}
	num_blocks	[1, 4]
	num_mem_tokens	{0, 2, 4, 8}
	conv1d_kernel_size[§]	{0, 2, 4, 8, 16, 32}
	dropout	[0.0, 0.3]
	learning rate	$[10^{-5}, 10^{-2}]$ (log)

† Constrained to $\leq$ large_kernel.
‡ Constrained to values that evenly divide the sequence length and yield ≥ 4 patches.
§ Constrained to $\leq$ sequence length.

B.5.2 Robustness methods

All three robustness variants use the TST architecture with the per-dataset hyperparameters found above; only the method-specific parameters differ.

ISensD. Input-Sensor Dropout is parameter-free: during training, each input channel is independently zeroed out with a fixed probability. No additional tuning is required beyond the base TST configuration.

F2F. We adopt the optimizer settings and masking parameters from the original publication (Brandt et al., 2025), adjusting only the average mask length to match the sequence length of each dataset. No further search is performed.

PGD. Adversarial training with Projected Gradient Descent introduces four parameters: the perturbation budget ϵ, the PGD step size α, the number of inner steps K, and the fraction of samples in each batch that are adversarially perturbed, r_{adv}. Because these parameters control the strength of the adversarial perturbation rather than the model architecture, tuning them on clean validation loss alone risks selecting configurations that merely regularize the model without genuinely improving robustness. We therefore optimize a *robustness-aware objective* on the validation set.

Robustness-aware tuning objective. The baseline architectures are tuned on clean validation RMSE, since the goal of the first part of our study is to characterize the robustness of state-of-the-art models tuned for nominal performance. For the robustification methods, an analogous clean-data objective would be self-defeating: the tuner would simply minimize the robustness hyperparameters that enlarge perturbations during training (e.g., the PGD budget ϵ), recovering near-vanilla training. We therefore tune these methods against a robustness-aware proxy that jointly rewards low clean error and stability under a small set of corruptions. For each trial we train on a 10 % subsample of the training data and evaluate a lightweight robustness proxy R on the validation set. R is the geometric mean of the per-corruption relative performance ratios $u_{\mathrm{rel}} = \mathrm{RMSE}_{\mathrm{clean}} \,/\, \mathrm{RMSE}_{\mathrm{corr}}$ computed over a minimal failure subset: bias and noise at severities $\{0.33, 0.66, 1.0\}$ plus the hard-fault at severity 0 (stuck-at-mean), applied to a single randomly selected feature (fixed per study). The tuning objective is $\mathrm{RMSE}_{\mathrm{clean}} \,/\, R$, which jointly rewards low clean error and high robustness. We run 50 Optuna trials with multivariate TPE sampling; the first trial is seeded with author-recommended defaults. Table 7 lists the search space.

Table 7: Search space for PGD adversarial training parameters.

Parameter	Choices
ϵ (perturbation budget)	{0.05, 0.1, 0.2, 0.3}
α (step size)	{0.01, 0.05, 0.1}
K (inner steps)	{7, 20}
r_{adv} (adversarial batch ratio)	{0.25, 0.5}

Design rationale. The tuning failure subset, bias, noise, and hard-fault at severity 0, deliberately mirrors the corruption types used by F2F during training, ensuring a comparable level of corruption exposure across methods. Crucially, this subset is a strict subset of the full benchmark evaluation grid: it covers only three of the failure modes and a small number of severities, and perturbs a single feature rather than all channels. This limits the overlap between the tuning signal and the test-time evaluation, reducing the risk of overfitting to the benchmark while still providing a meaningful robustness gradient for the optimizer. We recommend this minimal-subset protocol for future robustness tuning in order to balance informativeness with test-set integrity.

C Examples

To give the reader an intuitive sense of how the severity-parameterized failure modes of Section 3.2 act on real data, Figure 4 visualizes each of the ten failure modes applied at two severity levels to a single input channel from the PPG-DaLiA and Vehicle Dynamics datasets.

D Nominal performance

Table 8 reports clean-data RMSE for every (model, dataset) pair. Because per-dataset RMSE spans four orders of magnitude, we aggregate with the best-normalized geometric mean (GM). ModernTCN leads on six of nine datasets ($\mathrm{GM} = 1.027$, mean rank 2.2), and all three robustness-aware TST variants pay a clean-data penalty relative to the vanilla backbone ($+0.27$ to $+0.44$ in GM).

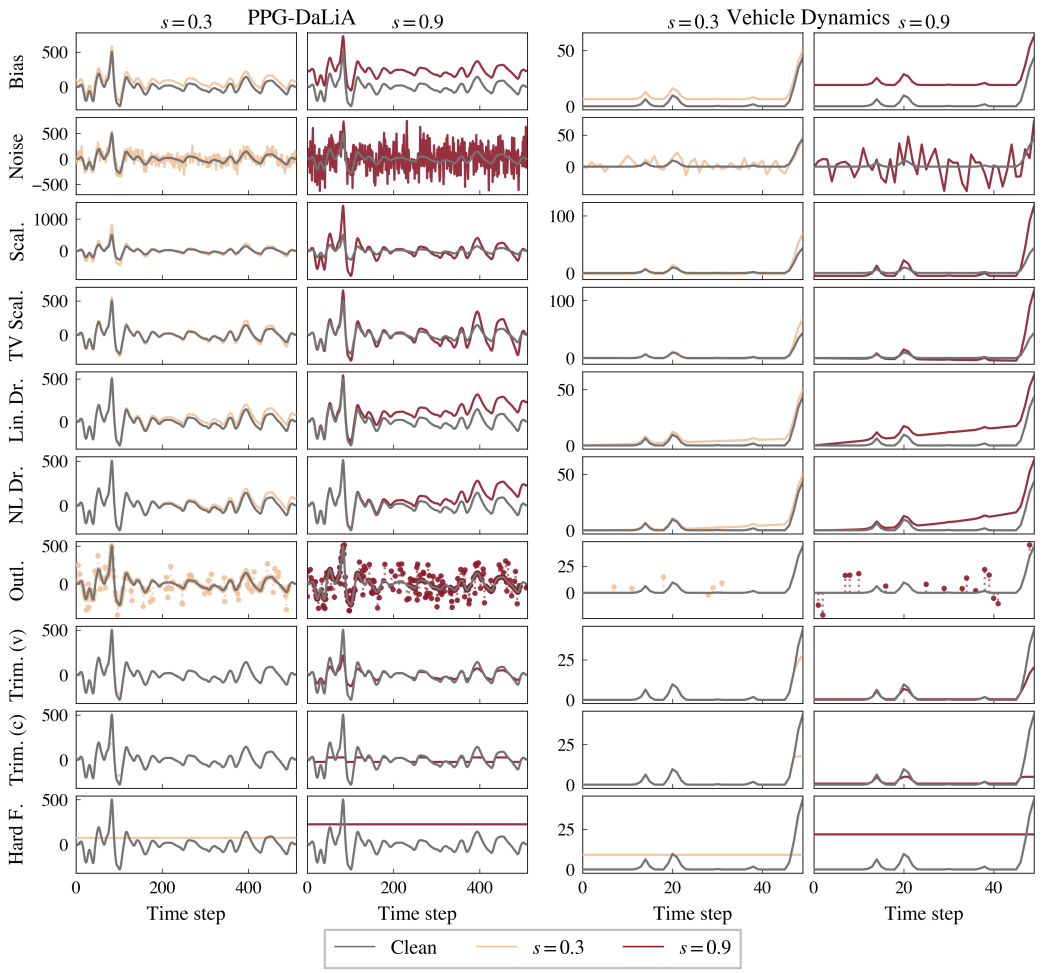


Figure 4: Illustration of the ten sensor failure modes applied to a representative input window from PPG-DaLiA (left) and Vehicle Dynamics (right). Each failure is shown at two severity levels alongside the clean signal, highlighting the qualitative differences between additive, multiplicative, drift-based, saturation-based, and stuck-at corruptions.

E Normalized performance under corruption (nPC)

Dividing mPC by the RMSE of the naïve mean predictor, a severity-invariant baseline that always predicts the training-set mean, yields:

$$\mathrm{nPC} = \frac{\mathrm{mPC}}{\mathrm{RMSE}_{\mathrm{naive}}}. \tag{4}$$

A value below 1 indicates that the model adds value over this trivial reference; a value of 1 marks the point at which the model is, on average, no better than the naïve predictor. The omnibus test is not significant (see Appendix J), so results should be read descriptively.

F Metric discussion

Relationship between rPC and nPC. The two normalized metrics are linked by a model-specific scaling factor:

$$\mathrm{rPC} = \mathrm{nPC} \times \frac{\mathrm{RMSE}_{\mathrm{naive}}}{\mathrm{RMSE}_{\mathrm{clean}}}. \tag{5}$$

Table 8: Nominal RMSE with 95% percentile bootstrap confidence intervals (test-set resampling, $n_{\text{boot}} = 200$). Each cell stacks the upper CI (top, grey), the bootstrap mean (centre, bold if best), and the lower CI (bottom, grey).

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14	Rank	GM
	98.097	67.091	0.025	9.727	0.051	3.974	0.122	0.101	0.078		
XGBoost	91.610	61.040	0.025	9.511	**0.051**	3.741	0.120	0.098	**0.077**	3.7	1.449
	85.757	56.036	0.025	9.300	0.051	3.541	0.117	0.095	0.076		
	98.480	67.400	0.011	9.193	0.051	4.047	0.110	0.091	0.081		
CatBoost	92.052	61.351	0.011	9.022	**0.051**	3.839	0.109	0.087	0.080	3.2	1.297
	86.119	56.008	0.011	8.811	0.051	3.652	0.107	0.085	0.079		
	96.674	65.837	0.009	2.484	0.055	2.518	0.118	0.076	0.080		
M-TCN	**90.942**	**60.060**	**0.009**	**2.411**	0.055	**2.356**	0.116	**0.074**	0.079	**2.2**	**1.027**
	85.371	55.037	0.009	2.344	0.055	2.220	0.115	0.072	0.078		
	109.455	72.728	0.033	9.587	0.054	4.076	0.276	0.146	0.150		
P-TSMixer	102.740	67.104	0.033	9.321	0.054	3.876	0.269	0.142	0.148	7.7	1.889
	97.036	61.907	0.033	9.024	0.054	3.716	0.264	0.140	0.146		
	100.817	70.090	0.009	7.632	0.053	4.126	0.103	0.077	0.082		
xLSTM-M	94.353	63.377	**0.009**	7.371	0.052	3.790	**0.101**	0.075	0.081	3.3	<u>1.222</u>
	88.400	57.810	0.009	7.166	0.052	3.479	0.100	0.073	0.080		
	103.926	70.092	0.017	7.035	0.054	4.023	0.113	0.080	0.083		
TST	97.340	63.627	0.017	6.889	0.054	3.789	0.112	0.078	0.082	4.4	1.329
	91.163	58.515	0.017	6.748	0.054	3.561	0.110	0.076	0.081		
	102.189	67.419	0.024	13.264	0.055	3.669	0.211	0.140	0.141		
TST-PGD	94.721	61.901	0.024	13.069	0.054	3.455	0.207	0.136	0.138	6.2	1.764 ↑+0.435
	88.588	57.215	0.024	12.852	0.054	3.246	0.204	0.132	0.136		
	102.909	72.737	0.026	12.238	0.054	4.887	0.159	0.110	0.109		
TST-ISensD	95.732	67.344	0.026	12.052	0.054	4.654	0.156	0.107	0.108	7.3	1.687 ↑+0.358
	89.714	62.294	0.025	11.852	0.054	4.401	0.153	0.105	0.106		
	117.895	68.616	0.012	15.820	0.061	4.545	0.151	0.110	0.110		
TST-F2F	108.129	62.415	0.012	15.471	0.061	4.283	0.148	0.107	0.108	6.9	1.601 ↑+0.272
	100.705	58.105	0.012	15.151	0.060	4.042	0.144	0.104	0.106		

Table 9: nPC per model and dataset. Bold = best, <u>underline</u> = second-best per column. GM is the geometric mean across datasets; arrows show GM change vs. TST.

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14	Rank	GM
XGBoost	**0.712**	<u>0.587</u>	0.227	0.543	<u>0.768</u>	**0.537**	<u>0.259</u>	<u>0.324</u>	<u>0.267</u>	**2.3**	<u>0.428</u>
CatBoost	0.724	0.597	<u>0.172</u>	0.503	**0.767**	0.604	**0.256**	**0.303**	**0.262**	2.6	**0.413**
M-TCN	0.724	0.611	0.345	**0.231**	0.876	1.378	0.334	0.374	0.362	5.6	0.499
P-TSMixer	0.785	0.658	0.358	0.449	0.814	0.958	0.522	0.589	0.559	7.2	0.607
xLSTM-M	<u>0.721</u>	0.595	0.215	<u>0.423</u>	0.789	<u>0.540</u>	0.280	0.339	0.319	3.0	0.430
TST	0.779	0.651	0.270	0.549	0.810	0.988	0.330	0.395	0.387	6.0	0.525
TST-PGD	0.743	**0.584**	0.285	0.674	0.819	0.553	0.381	0.414	0.384	5.6	0.509 ↓-0.016
TST-ISensD	0.796	0.659	0.443	0.673	0.819	0.685	0.370	0.474	0.452	7.7	0.576 ↑+0.051
TST-F2F	0.868	0.608	**0.113**	1.032	1.115	0.600	0.276	0.331	0.290	5.1	0.469 ↓-0.056

The ratio $\mathrm{RMSE}_{\text{naive}}/\mathrm{RMSE}_{\text{clean}}$ is itself interpretable: it measures how much better than naïve the model is on clean data, and thus quantifies the headroom available for degradation before the model becomes useless.

Relationship to mCE. The normalized Performance under Corruption (nPC) is analogous to the mean Corruption Error (mCE) of ImageNet-C (Hendrycks & Dietterich, 2018), with one deliberate departure in the choice of reference. In the original ImageNet-C protocol, each corruption type's summed error is normalized by AlexNet's error on that same corruption, equalizing the difficulty of different corruption types in the denominator. nPC instead uses the naïve mean predictor, a severity-invariant baseline, as a single, shared reference. Because this denominator is constant across models, severities, and failure modes, differences in nPC directly reflect the intrinsic difficulty of each failure mode and dataset, which we consider informative for practitioners assessing deployment risk. This choice also means that nPC preserves difficulty differences across failure modes as a finding of interest, rather than normalizing them away.

Relationship between rPC, Relative mCE, and the Robustness Score. The relative Performance under Corruption (rPC) was introduced by Michaelis et al. (2020) to isolate degradation from nominal accuracy by normalizing with the model's own clean-data performance.Hendrycks & Dietterich (2018) achieve the same goal with the Relative mCE, which *subtracts* the clean error rate before normalizing by AlexNet. Windmann et al. (2025) define a closely related quantity, the relative performance ratio $u_{\mathrm{rel}} = \mathrm{RMSE}_{\mathrm{clean}}/\mathrm{RMSE}_{d,c,s}$, which is simply the reciprocal of the per-combination degradation ratio. We use the rPC that shares the "lower-is-better" direction of mPC, making tables and rankings immediately readable without direction-switching.

G Per-failure-mode evaluation

The main evaluation aggregates over all failure modes, features, and severities into a single mPC and rPC per model and dataset. Here we report both metrics *per failure mode*, retaining the average over features and severities but disaggregating across failure modes to reveal which corruption types drive the overall scores.

Computation. For a given failure mode $d \in \mathcal{D}$, we restrict the summation in Equation equation 1 to that mode and define

$$\mathrm{mPC}_d = \frac{1}{|\mathcal{C}|\,|\mathcal{S}|} \sum_{c \in \mathcal{C}} \sum_{s \in \mathcal{S}} \mathrm{RMSE}_{d,c,s}\,, \tag{6}$$

the mean RMSE under failure mode d, averaged over all affected features c and severities s. The per-failure-mode relative metric follows Equation equation 2 directly:

$$\mathrm{rPC}_d = \frac{\mathrm{mPC}_d}{\mathrm{RMSE}_{\mathrm{clean}}}\,. \tag{7}$$

Cross-dataset aggregation. We aggregate across datasets using the same protocol as in the main paper (Section 3.3). rPC_d is a dimensionless ratio and enters the geometric mean directly. mPC_d carries dataset-specific units; for each dataset–failure-mode pair we first normalize by the best model (1.0 = best), then take the geometric mean across datasets.

Failure-mode difficulty. To rank failure modes by difficulty we compute, for each failure mode d, the geometric mean across all models, of rPC_d for the relative view and of the normalized mPC_d for the absolute view. A higher value indicates that the failure mode causes larger degradation on average; the rPC-based ranking is independent of nominal accuracy, while the mPC-based ranking additionally reflects absolute error magnitude. Tables 10 and 11 sort rows by the respective difficulty score in ascending order (easiest to hardest).

Analysis. The per-failure-mode breakdown reveals substantial heterogeneity in corruption difficulty. The easiest failure modes increase rPC only marginally above 1, whereas the hardest modes raise it considerably, motivating the finer-grained view provided here as a complement to the aggregate scores in the main paper. Model rankings also shift across failure modes: a model that leads under one corruption type may be among the most affected under another, highlighting the practical value of per-failure-mode reporting for practitioners selecting models for deployment environments with known sensor characteristics. Full per-failure-mode tables are provided in Tables 10 and 11.

Table 10: Per-failure-mode normalized mPC. Rows sorted by difficulty (top = easiest). Bold = best model per failure mode; green/red cells = each model's best/worst failure mode; shaded columns are robust TST variants. GM = geometric mean across models.

Failure	XGBoost	CatBoost	M-TCN	P-TSMixer	xLSTM-M	TST	TST-PGD	TST-ISensD	TST-F2F	GM
Trim. (c)	1.248	1.173	**1.076**	1.666	1.140	1.250	1.511	1.360	1.366	1.299
Trim. (v)	1.301	1.200	**1.035**	1.691	1.153	1.244	1.574	1.471	1.469	1.333
Outl.	1.227	1.128	**1.057**	1.664	1.114	1.418	1.656	1.833	1.437	1.368
TV Scal.	1.216	**1.148**	1.303	1.670	1.224	1.367	1.460	1.635	1.424	1.372
Scal.	1.174	**1.133**	1.584	1.684	1.176	1.481	1.428	1.670	1.500	1.411
Noise	1.149	1.100	1.518	1.599	**1.033**	1.545	1.503	2.016	1.553	1.417
Hard F.	1.221	1.232	1.713	1.830	1.284	1.731	1.372	1.500	**1.202**	1.436
NL Dr.	1.349	1.290	1.331	1.830	1.444	1.550	1.496	1.691	**1.285**	1.464
Lin. Dr.	1.319	1.287	1.438	1.849	1.414	1.592	1.455	1.685	**1.258**	1.467
Bias	1.273	1.270	1.736	1.864	1.343	1.717	1.430	1.677	**1.241**	1.489

Table 11: Per-failure-mode rPC. Rows sorted by difficulty (top = easiest). Bold = best model per failure mode; green/red cells = each model's best/worst failure mode; shaded columns are robust TST variants. GM = geometric mean of rPC across models.

Failure	XGBoost	CatBoost	M-TCN	P-TSMixer	xLSTM-M	TST	TST-PGD	TST-ISensD	TST-F2F	GM
Trim. (v)	1.047	1.080	1.177	1.044	1.100	1.092	1.041	**1.017**	1.070	1.073
Trim. (c)	1.089	1.144	1.327	1.116	1.180	1.190	1.084	**1.020**	1.079	1.134
Outl.	**1.067**	1.096	1.298	1.110	1.148	1.344	1.183	1.368	1.131	1.189
TV Scal.	1.143	1.206	1.729	1.204	1.364	1.400	**1.127**	1.319	1.211	1.289
NL Dr.	1.338	1.430	1.863	1.392	1.698	1.675	1.219	1.440	**1.153**	1.451
Scal.	1.216	1.311	2.315	1.337	1.443	1.671	**1.215**	1.484	1.405	1.460
Noise	**1.255**	1.342	2.339	1.339	1.336	1.839	1.348	1.890	1.534	1.547
Lin. Dr.	1.396	1.522	2.148	1.501	1.774	1.836	1.265	1.531	**1.205**	1.552
Bias	1.548	1.725	2.978	1.738	1.935	2.275	1.428	1.750	**1.365**	1.809
Hard F.	1.634	1.843	3.238	1.879	2.038	2.526	1.509	1.725	**1.456**	1.922

H MEAN IMPUTATION AS A FAILURE MITIGATION STRATEGY

The main benchmark evaluates robustness across a wide range of failure modes, severities, and affected channels. In this section we analyze a practically relevant special case: a deployment scenario in which (i) a reliable failure detection mechanism is available, and (ii) the operator can intervene on the input signal. When a sensor is detected as faulty, the simplest corrective action is to replace the failing channel with its training-set mean, which, under z-score normalization, amounts to setting it to zero. This corresponds to the *hard-fault* failure mode at severity $s = 0$ in our benchmark.

Metrics. Since both the failure mode and severity are fixed, the mPC and rPC definitions (Eqs. equation 1–equation 2) reduce to an average over affected channels only:

$$\mathrm{mPC}_{\mathrm{hf},0} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathrm{RMSE}_{\mathrm{hf},\,c,\,0}\,, \tag{8}$$

$$\mathrm{rPC}_{\mathrm{hf},0} = \frac{\mathrm{mPC}_{\mathrm{hf},0}}{\mathrm{RMSE}_{\mathrm{clean}}}\,. \tag{9}$$

Cross-dataset aggregation follows the protocol in Section 3.3: $\mathrm{rPC}_{\mathrm{hf},0}$ enters the geometric mean directly; $\mathrm{mPC}_{\mathrm{hf},0}$ is first normalized per dataset by the best model before computing the geometric mean.

Results. Tables 12 and 13 report mPC and rPC for all models under this scenario. TST-ISensD achieves the best geometric-mean rPC (1.034) and mPC (1.072) as well as the best mean rank on both metrics, and on rPC significantly outranks CatBoost, xLSTM-M, TST, and M-TCN (Appendix J).

On rPC, TST-ISensD attains near-perfect scores close to 1.0 on every dataset, indicating virtually no degradation when a channel is replaced by its mean, a direct consequence of its training procedure, which randomly drops input channels by setting them to zero. The model has explicitly learned to produce accurate predictions when any single channel is absent. Among the remaining models, TST-F2F (GM rPC 1.208) and TST-PGD (1.275) also improve substantially over the base TST (1.718), though neither approaches ISensD's consistency.

Practical implication. These results suggest that when reliable failure detection and input intervention are available, ISensD, a simple dropout-based augmentation applied to input channels during training, provides an effective and easy-to-implement strategy. It transforms a scenario that degrades most models into one where prediction quality is on average nearly unaffected, without requiring architectural changes or adversarial training.

Table 12: mPC under mean imputation (hard-fault at severity s=0); Bold/underline: best/second-best per column. GM is the geometric mean of best-normalized values (1.0 = best); arrows show GM change vs. TST.

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14	Rank	GM
XGBoost	98.61	68.43	0.07	13.58	**0.05**	4.82	0.18	0.12	**0.10**	3.9	1.209
CatBoost	101.66	68.81	0.06	12.92	**0.05**	4.83	0.17	**0.11**	0.12	3.4	1.195
M-TCN	99.15	71.02	0.07	**8.18**	0.06	11.17	0.19	0.12	0.13	5.6	1.304
P-TSMixer	106.20	73.66	0.10	12.22	**0.05**	7.09	0.34	0.23	0.23	7.2	1.658
xLSTM-M	101.45	71.94	0.06	13.00	**0.05**	4.65	0.18	**0.11**	0.12	3.9	1.208
TST	108.01	76.68	0.06	15.12	**0.05**	9.03	0.20	0.13	0.14	6.8	1.403
TST-PGD	112.96	**67.15**	0.07	16.54	**0.05**	5.37	0.23	0.15	0.15	6.8	1.381 ↓-0.021
TST-ISensD	**97.07**	67.90	**0.03**	13.21	**0.05**	**4.62**	**0.16**	**0.11**	0.11	**2.2**	**1.072** ↓-0.331
TST-F2F	114.68	70.63	**0.03**	17.61	0.07	4.98	**0.16**	0.12	0.11	5.2	1.188 ↓-0.215

Table 13: rPC under mean imputation (hard-fault at severity s=0); Bold/underline: best/second-best per column. GM is the geometric mean across datasets; arrows show GM change vs. TST.

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14	Rank	GM
XGBoost	1.076	1.121	2.855	1.428	1.007	1.289	1.482	1.257	1.319	4.8	1.358
CatBoost	1.104	1.122	5.602	1.432	1.007	1.258	1.596	1.306	1.464	5.7	1.500
M-TCN	1.090	1.182	7.315	3.393	1.043	4.741	1.677	1.601	1.628	8.0	2.068
P-TSMixer	1.034	1.098	2.988	1.311	1.004	1.830	1.249	1.594	1.525	4.6	1.429
xLSTM-M	1.075	1.135	6.658	1.764	1.006	1.228	1.758	1.505	1.539	6.2	1.609
TST	1.110	1.205	3.685	2.195	1.005	2.383	1.749	1.667	1.727	7.7	1.718
TST-PGD	1.193	1.085	2.728	1.265	1.003	1.553	1.098	1.069	1.090	3.6	1.275 ↓-0.443
TST-ISensD	**1.014**	**1.008**	**1.160**	**1.096**	**0.999**	**0.992**	**1.016**	**1.014**	**1.021**	**1.0**	**1.034** ↓-0.684
TST-F2F	1.061	1.132	2.405	1.138	1.126	1.163	1.109	1.080	1.064	3.6	1.208 ↓-0.510

I SEVERITY CURVES

The aggregate scores in the main paper compress an entire severity sweep into a single number per (model, dataset) pair. To give the reader a sense of the underlying behaviour, we show four illustrative slices of the full sweep: per-failure-mode severity curves on two datasets, heartrate estimation (HR) and tire temperature estimation (Veh.), for two model groupings. Figures 5 and 6 compare three non-robustified baselines (XGBoost, xLSTM-M, M-TCN); Figures 7 and 8 compare vanilla TST against its adversarially-trained variant TST-PGD.

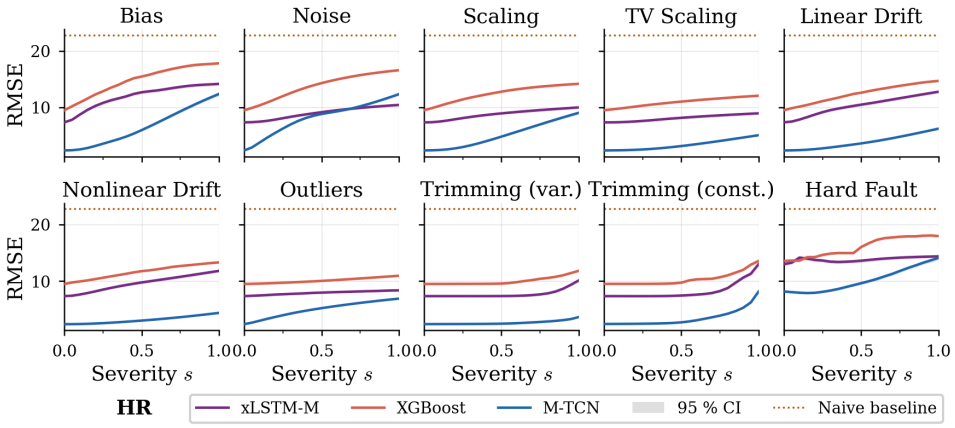


Figure 5: HR severity curves: XGBoost, xLSTM-M, M-TCN.

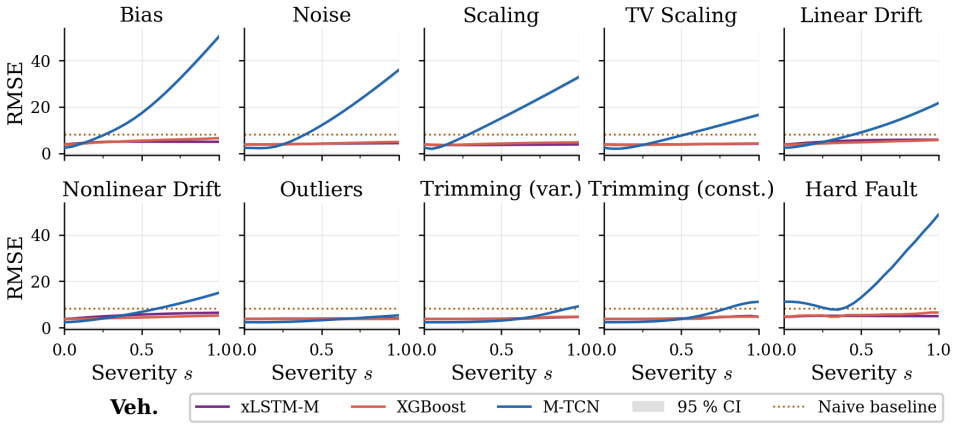


Figure 6: Veh. severity curves: XGBoost, xLSTM-M, M-TCN.

Baseline behaviour across datasets. On HR (Figure 5), M-TCN starts from a substantially lower clean RMSE than xLSTM-M or XGBoost, reflecting its strong nominal performance (Appendix D). Although M-TCN degrades faster in relative terms, its absolute corrupted RMSE remains lowest across most failure modes, which is exactly what mPC rewards: M-TCN is the best mPC model on HR (Table 2). On Vehicle Dynamics (Figure 6), the picture inverts. M-TCN collapses under tire-temperature corruption, degrading far more sharply than the other models and ending well above them at high severity. This is consistent with M-TCN's worst-on-dataset rPC and its weakest mPC ranking on Vehicle Dynamics (Tables 2 and 3).

Effect of adversarial training. On HR (Figure 7), TST-PGD starts from a noticeably worse clean RMSE than vanilla TST but degrades more gradually with severity. The clean-data penalty raises TST-PGD's mPC above TST's on this dataset; yet rPC, places TST-PGD second on HR (Table 3), correctly identifying the underlying relative robustness gain that mPC obscures. On Vehicle Dynamics (Figure 8), nominal performance is comparable between the two models, while relative degradation is markedly smaller for TST-PGD across most failure modes. With no clean-data penalty to offset,

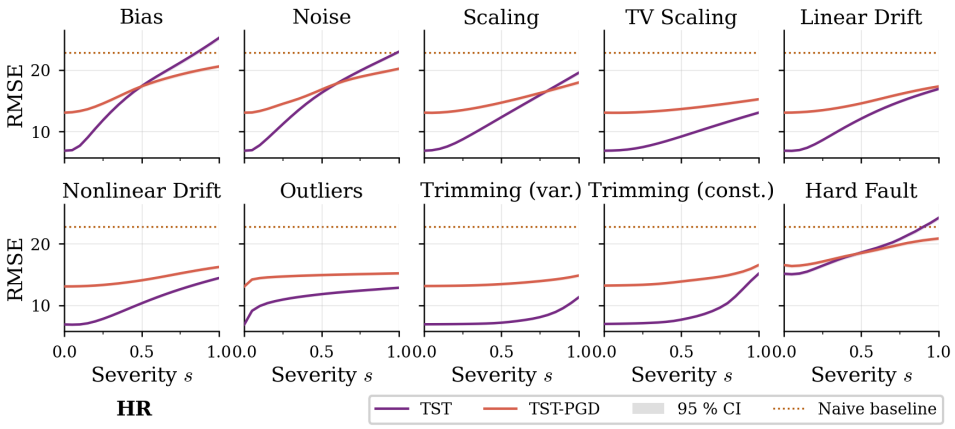


Figure 7: HR severity curves: TST vs. TST-PGD.

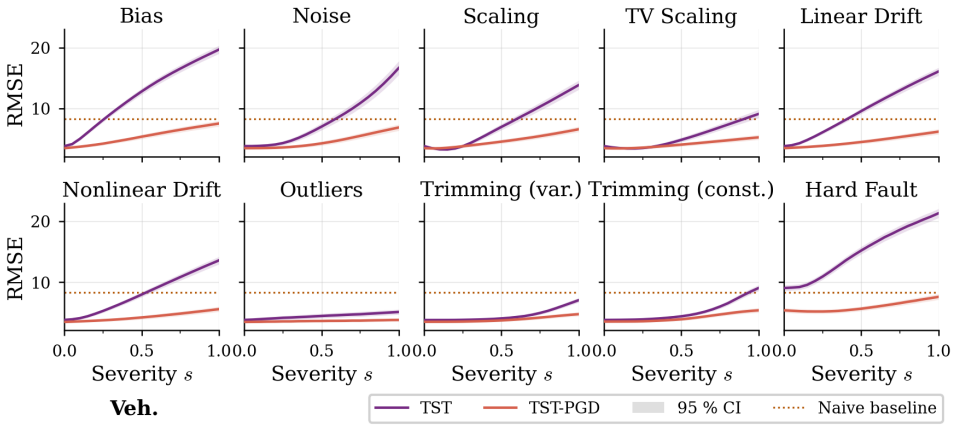


Figure 8: Veh. severity curves: TST vs. TST-PGD.

this robustness gain registers on both mPC and rPC. The two PGD examples together show how rPC and mPC can agree or diverge depending on whether robustification incurs a nominal cost.

J Statistical significance testing

We follow the Demšar (2006) protocol as implemented by `autorank` (Herbold, 2020): Shapiro–Wilk normality tests select between Bartlett's (normal) or Levene's (non-normal) homogeneity test, which determines whether parametric (ANOVA + Tukey HSD) or non-parametric (Friedman + Nemenyi) tests are applied ($\alpha = 0.05$). Results are summarized in Table 14; for metrics with a significant omnibus result, we additionally report critical difference (CD) diagrams that visualize mean ranks and non-significant clusters (connected by horizontal bars).

Table 14: Significance results. N: observations; k: models.

Metric	**Test**	N	k	p	**Post-hoc**
nom	Friedman	9	9	< 0.001	Nemenyi
mPC	Friedman	9	9	< 0.001	Nemenyi
mPC_{hf0}	Friedman	9	9	< 0.001	Nemenyi
rPC	Friedman	9	9	< 0.001	Nemenyi
rPC_{hf0}	Friedman	9	9	< 0.001	Nemenyi
s_{cross}	Friedman	9	9	0.187	—
nPC	ANOVA	9	9	0.069	—

Nominal Performance M-TCN attains the best mean rank, with the tree ensembles, xLSTM-M, plain TST and TST-PGD forming the leading non-significant cluster (Figure 9); the other robustified TST variants and P-TSMixer rank significantly worse.

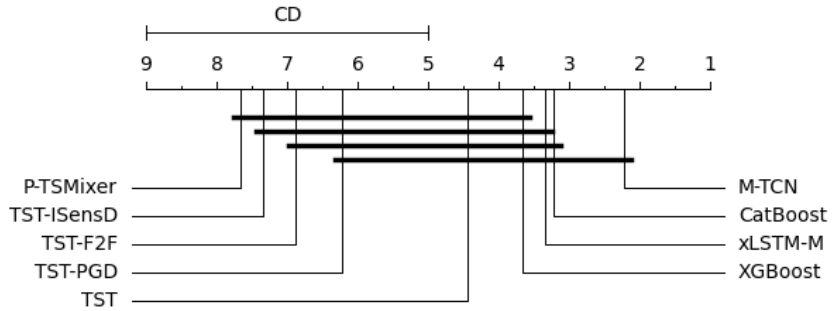


Figure 9: Critical difference diagram for nominal performance (nom). CD $= 4.004$.

mPC XGBoost, CatBoost, and xLSTM-M move to the top, all three significantly outranking P-TSMixer and TST-ISensD (Figure 10). M-TCN, the nominal leader, drops to mid-pack but remains within the leading non-significant cluster.

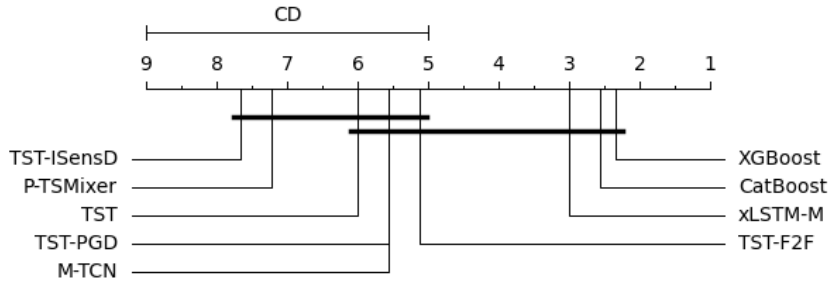


Figure 10: Critical difference diagram for mean performance under corruption (mPC). CD $= 4.004$.

mPC at hard fault with zero severity The ranking shifts substantially relative to mPC: TST-ISensD moves to the top, while plain TST, TST-PGD, and P-TSMixer rank significantly worse than the leader (Figure 11).

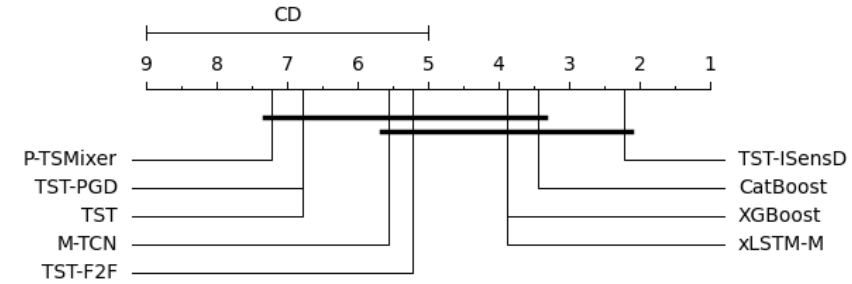


Figure 11: Critical difference diagram for mPC restricted to the hard fault with zero severity. CD $= 4.004$.

rPC TST-PGD, XGBoost, and P-TSMixer lead the ranking; M-TCN ranks last and plain TST is significantly outranked by all three top models (Figure 12). All three robustified TST variants join the top non-significant cluster, meaning robustification eliminates TST's significant deficit.

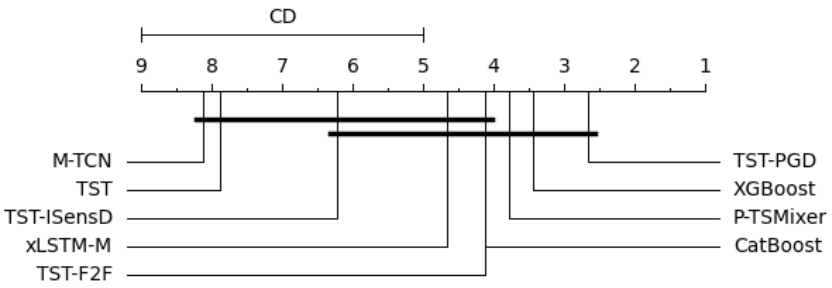


Figure 12: Critical difference diagram for relative performance under corruption (rPC). CD $= 4.004$.

rPC at hard fault with zero severity TST-ISensD ranks best by a large margin and is significantly better than CatBoost, xLSTM-M, plain TST, and M-TCN (Figure 13).

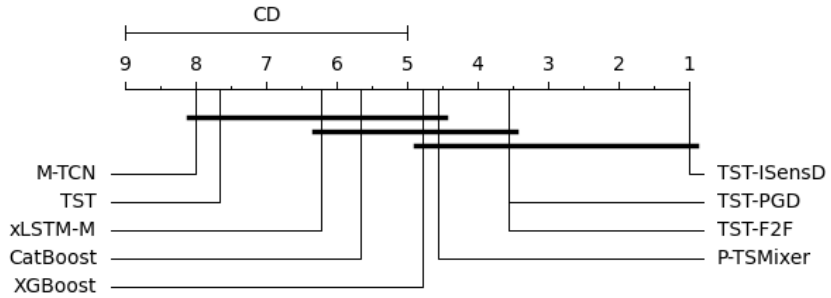


Figure 13: Critical difference diagram for rPC restricted to the hard fault with zero severity. CD $= 4.004$.

Baseline crossing severity s_{cross} Differences are not statistically significant (Friedman, $p = 0.187$); no post-hoc comparisons are performed and no CD diagram is reported.

nPC The omnibus test is not significant (ANOVA, $p = 0.069$); no post-hoc comparisons are performed and no CD diagram is reported. Descriptively, XGBoost, CatBoost, and xLSTM-M rank best, while P-TSMixer and TST-ISensD rank worst, mirroring the aggregate mPC ordering. This is expected since the naive baseline rescaling is constant across models for one dataset, therefore within-dataset ranks are identical to mPC. The discrepancy in significance arises because the normality-test outcomes route nPC to ANOVA and mPC to the Friedman test.

K MULTI-SEED TRAINING

To test whether the robustness ranking depends on a single training run, we retrained all nine models from scratch on all nine datasets under two additional seeds (43 and 44) and compare them with the run reported in the main paper (seed 42). Each seed was evaluated on the full severity sweep (10 failure modes, 20-point severity grid, $k = 3$). See Table 15.

Seed-to-seed variation is small on most datasets: the standard deviation is below 0.03 in 54 of the 81 model–dataset cells, and the gradient-boosted trees vary by at most 0.071. Larger variation is confined to the neural models on Batt. and Veh. (largest for M-TCN on Batt.: 7.087 ± 1.834). The mean ranks on the seed-averaged rPC reproduce the order of the run reported in the main paper: TST-PGD first, followed by XGBoost and P-TSMixer, with TST and M-TCN last.

Table 15: rPC per model and dataset, mean ± standard deviation over the three training seeds (42, 43, 44). Rank is the autorank mean rank on the seed-averaged rPC (lower is better). Bold = best, underline = second-best per column (by the mean).

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14	Rank
XGBoost	1.057±0.003	1.088±0.002	**2.147**±0.011	1.302±0.003	1.007±0.000	1.194±0.008	1.398±0.003	1.211±0.001	1.435±0.009	3.22
CatBoost	1.068±0.005	1.101±0.001	3.598±0.071	1.269±0.000	1.007±0.000	1.303±0.008	1.515±0.003	1.262±0.003	1.345±0.008	4.11
M-TCN	1.091±0.011	1.131±0.016	7.087±1.834	2.281±0.082	1.050±0.031	4.330±0.499	1.946±0.081	1.710±0.243	1.927±0.124	8.22
P-TSMixer	1.044±0.004	1.102±0.006	2.609±0.004	**1.090**±0.009	**1.005**±0.000	1.854±0.291	1.244±0.004	1.505±0.008	1.564±0.025	3.78
xLSTM-M	**1.037**±0.006	1.068±0.006	5.478±0.176	1.329±0.017	1.007±0.001	1.197±0.057	1.711±0.077	1.623±0.027	1.593±0.039	4.89
TST	1.099±0.027	1.167±0.012	4.363±0.563	1.834±0.021	1.010±0.001	2.325±0.150	1.883±0.012	1.891±0.040	1.968±0.058	7.78
TST-PGD	1.061±0.008	**1.065**±0.005	2.946±0.288	1.317±0.124	1.006±0.000	1.350±0.063	**1.171**±0.010	**1.113**±0.005	1.151±0.005	**2.78**
TST-ISensD	1.132±0.030	1.093±0.010	4.507±0.639	1.280±0.018	1.014±0.004	1.373±0.167	1.560±0.031	1.570±0.060	1.675±0.069	6.11
TST-F2F	1.106±0.009	1.100±0.005	2.472±0.261	1.384±0.129	1.256±0.026	**1.179**±0.028	1.218±0.017	1.125±0.006	**1.095**±0.016	4.11

L SIMULTANEOUS SENSOR FAILURES

The main benchmark corrupts one channel at a time. Here $n_{\text{fail}} \in \{1, 2, 3, 4\}$ channels fail simultaneously under the same failure mode and severity. For each dataset and n_{fail} we sample up to 30 channel subsets (all subsets if fewer exist), shared across models, and use a 5-point severity grid $\{0, 0.25, 0.5, 0.75, 1\}$ with $k = 3$. The ranks are calculated based on the rPC evaluation. See Table 16

Table 16: Overall rank on rPC of each model (1 = most robust) with n_{fail} simultaneously failing sensors (seed-44 checkpoints, 5-point severity grid, all nine datasets).

Model	$n_{\text{fail}} = 1$	$n_{\text{fail}} = 2$	$n_{\text{fail}} = 3$	$n_{\text{fail}} = 4$
XGBoost	2	2	2	3
CatBoost	4	4	4	4
M-TCN	9	9	9	9
P-TSMixer	3	3	3	2
xLSTM-M	6	6	6	5
TST	8	8	8	8
TST-PGD	**1**	**1**	**1**	**1**
TST-ISensD	7	7	7	6
TST-F2F	5	5	5	7

Absolute degradation grows with the number of failing sensors, but the ranking is stable: it is identical for one, two and three failing sensors. With four, TST-PGD and M-TCN remain first and last, and only the middle reshuffles: P-TSMixer, xLSTM-M and TST-ISensD move up by one place, XGBoost moves down by one, and TST-F2F drops from fifth to seventh.

M ROBUSTIFICATION BEYOND TST

The main paper applies the robustification methods to TST only. Here the two architecture-agnostic methods, PGD adversarial training and ISensD sensor dropout, are applied to M-TCN, xLSTM-M and P-TSMixer on all nine datasets. The PGD attack parameters (perturbation budget, step size, number of steps and share of adversarial samples per batch) are tuned per architecture and dataset with the same search space and objective as for TST; ISensD uses the same settings as for TST, and all other hyperparameters are those of the backbone's tuned vanilla configuration. F2F is excluded because its pre-training scheme is tied to the TST encoder. All models are evaluated on the full severity sweep. See Table 17 and Table 18.

Table 17: rPC of every backbone without robustification, with PGD and with ISensD.

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14
M-TCN	1.086	1.149	9.200	2.187	1.059	4.837	1.853	1.854	1.896
M-TCN-PGD	1.052	1.103	6.280	1.691	**1.006**	1.178	1.169	1.198	1.117
M-TCN-ISensD	1.233	1.110	10.769	1.613	1.015	1.747	1.521	1.830	1.861
xLSTM-M	1.042	1.061	5.550	1.309	1.007	1.178	1.777	1.645	1.634
xLSTM-M-PGD	1.053	1.059	2.252	1.467	1.007	**1.166**	1.224	1.493	1.242
xLSTM-M-ISensD	1.049	1.052	3.959	1.086	1.007	1.232	1.209	1.278	1.239
P-TSMixer	1.043	1.108	2.613	1.099	**1.006**	2.043	1.247	1.509	1.559
P-TSMixer-PGD	1.042	1.056	2.318	**1.054**	1.008	1.211	1.108	1.131	1.192
P-TSMixer-ISensD	**1.027**	**1.049**	**1.629**	1.068	1.007	1.275	**1.029**	**1.031**	**1.031**
TST	1.091	1.155	3.761	1.817	1.010	2.157	1.897	1.858	1.948
TST-PGD	1.070	1.066	2.833	1.175	**1.006**	1.324	1.182	1.108	1.147
TST-ISensD	1.134	1.105	4.129	1.272	1.011	1.217	1.525	1.608	1.736

Table 18: mPC of every backbone without robustification, with PGD and with ISensD.

Model	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14
M-TCN	98.73	69.00	0.08	**5.27**	0.06	11.39	0.22	0.14	0.15
M-TCN-PGD	**98.17**	67.00	0.07	5.81	**0.05**	5.52	0.23	**0.12**	0.15
M-TCN-ISensD	115.48	70.97	0.17	7.80	0.08	5.36	0.22	0.17	0.20
xLSTM-M	98.33	67.26	**0.05**	9.65	**0.05**	4.47	**0.18**	**0.12**	**0.13**
xLSTM-M-PGD	98.81	67.08	**0.05**	9.44	**0.05**	**4.02**	0.20	**0.12**	**0.13**
xLSTM-M-ISensD	106.17	69.77	**0.05**	12.33	**0.05**	5.23	**0.18**	0.14	0.15
P-TSMixer	107.16	74.35	0.09	10.25	**0.05**	7.92	0.34	0.21	0.23
P-TSMixer-PGD	103.66	72.28	0.08	10.23	**0.05**	5.89	0.33	0.19	0.21
P-TSMixer-ISensD	103.34	73.33	0.14	12.49	0.06	5.71	0.47	0.27	0.48
TST	106.23	73.51	0.06	12.52	**0.05**	8.17	0.21	0.14	0.16
TST-PGD	101.39	**65.98**	0.07	15.36	**0.05**	4.57	0.25	0.15	0.16
TST-ISensD	108.56	74.41	0.11	15.33	**0.05**	5.66	0.24	0.17	0.19

N iTRANSFORMER BASELINE

TST is the only neural baseline older than 2023. We therefore additionally evaluate iTransformer, a more recent transformer that attends across variates rather than across time. We use the implementation from the Time-Series-Library with its classification head as a one-output regression head, tune its hyperparameters per dataset with the protocol used for the other neural baselines, and evaluate it on the full severity sweep. See Table 19.

O DIAGNOSTIC TOOLS

The aggregate metrics rank models but do not show why a model fails. We therefore add four diagnostics that keep the (failure mode, channel, severity) axes the aggregates average out. All four are computed from the existing severity sweep and are available through the `Results` API, individually and as a combined `diagnose()` report:

Table 19: iTransformer under the benchmark protocol.

Metric	PM10	PM2.5	Batt.	HR	TEP	Veh.	MMR	T66-13	T66-14
mPC	109.93	70.71	0.05	10.31	0.05	4.64	0.16	0.11	0.12
rPC	1.091	1.108	4.934	1.187	1.006	1.767	1.309	1.231	1.171
s_{cross}	0.50	0.70	0.55	never	0.40	0.45	0.70	0.90	0.75

- **Attributed s_{cross}** reports the failure mode and channel responsible for the baseline crossing; the definition of s_{cross} is unchanged.
- **rPC$_{\text{worst}}$** takes the maximum instead of the mean over the (failure mode, channel) cells that rPC averages, and names the worst cell.
- **Knee severity** is the severity at which the RMSE curve is steepest, with the failure mode and channel of that curve.
- **Channel concentration** is the share of a model's total excess RMSE (above clean, over all modes and severities) carried by its worst channel; dividing by the uniform share $1/C$.

Table 20: The four diagnostics for two models on TEP (33 channels). Cells read value, failure mode / channel.

Diagnostic	XGBoost	TST-F2F
rPC$_{\text{worst}}$	1.71, hard fault / ch10	15.72, noise / ch15
Channel concentration	87% on ch10	27% on ch15
Attributed s_{cross}	0.25, hard fault / ch10	0.05, hard fault / ch29
Knee severity	0.55, hard fault / ch10	0.55, noise / ch15

For XGBoost on TEP all four diagnostics point to the same channel: 87% of its excess RMSE comes from one channel out of 33, so the exposure is localised and hardening that sensor is indicated. For TST-F2F the exposure is spread across channels (27% on its worst channel) and the baseline crossing occurs on a different channel than its worst cell, so a distributional fix is indicated instead. The diagnostics also separate models with similar aggregate scores: TST-F2F and TST-PGD have nearly the same overall rPC (1.272 and 1.253), but their worst cells lie on different datasets, failure modes and sensors (TST-F2F: 15.72 on TEP under noise, where TST-PGD reaches 1.16; TST-PGD: 12.67 on Batt. under noise, where TST-F2F reaches 6.71). Across datasets (geometric mean), concentration is highest for XGBoost and P-TSMixer (5.1$\times$ the uniform share) and lowest for TST-F2F (3.1$\times$).

P PACKAGE API REFERENCE

This appendix supplements the minimal example in Listing 1 with the details needed to integrate arbitrary models and extend the evaluation suite.

Data format. All input arrays follow an `(N, T, C)` layout: N samples, T time steps, C sensor channels. Labels are one-dimensional arrays of shape `(N,)`. Data is z-score normalized per channel before any failure mode is applied.

Registering a model. Exactly one of `predict_fn` or `model` must be supplied. `predict_fn` is a callable taking a `numpy` array of shape `(N, T, C)` and returning predictions of shape `(N,)`; `model` is a PyTorch `nn.Module` or a scikit-learn estimator with `.predict`:

```
def predict_fn(X: np.ndarray) -> np.ndarray:
    """X: (N, T, C) -> predictions: (N,)"""
```

```
    ...

testbed.add_model("MyFn",  predict_fn=predict_fn)
testbed.add_model("TCN",   model=my_pytorch_module)
testbed.add_model("Ridge", model=sklearn_ridge)
```

Custom datasets. Datasets can be supplied as a benchmark ID (e.g. `rob.PPGDalia`), a directory of `.ts` files, an `(X_train, y_train, X_test, y_test[, name])` tuple, or via `testbed.add_dataset(...)`. The helper `rob.prepare_dataset(df, target, window_size, ...)` converts a time-ordered `pandas` DataFrame into the windowed arrays expected by the testbed.

Custom metrics. Subclass `Metric` and register the instance; `name` becomes a column in the raw DataFrame:

```
from muvis_c import Metric

class MAEMetric(Metric):
    name = "mae"
    def compute(self, y_true, y_pred):   # both shape (N,)
        return float(np.mean(np.abs(y_true - y_pred)))

testbed.add_metric(MAEMetric())
```

Custom failure modes. Subclass `SeverityFailure` and implement `apply`, which returns a corrupted copy of `X` for severity $s \in [0, 1]$ applied to channel `feature_idx`. The base class stores the maximum perturbation scale `self.k`:

```
import torch
from muvis_c import SeverityFailure

class SpikeFailure(SeverityFailure):
    """Single spike at the sequence midpoint."""
    def apply(self, X: torch.Tensor, feature_idx: int,
              severity: float) -> torch.Tensor:
        out = X.clone()
        mid = out.shape[1] // 2
        out[:, mid, feature_idx] += severity * self.k
        return out

testbed.add_failure(SpikeFailure(k=5.0))
```

Testbed configuration. Table 21 lists the constructor arguments.

Results object. `testbed.run()` returns a `Results` instance wrapping a single `pandas` DataFrame (`Results.raw`) with one row per (model, dataset, failure mode, feature, severity) tuple. The DataFrame contains the bootstrap RMSE under corruption (`boot_mean`), the clean reference (`clean_boot_mean`), the naive mean-predictor reference (`baseline_boot_mean`), bootstrap confidence intervals, and one column per registered custom metric. Aggregate quantities are exposed as methods that return `model × dataset` pivot tables (Table 22).

For inspection and export, `Results` additionally exposes `summary()` (formatted score table), `to_csv(output_dir)` (writes `output_dir/per_severity.csv`), and `plot_degradation(...)` (RMSE-vs-severity curves with optional filters on dataset, model, failure, or feature).

Table 21: `Testbed` constructor parameters.

Parameter	Default	Description
`dataset`	`None`	Benchmark ID, path to a `.ts` directory, or `(X_train, y_train, X_test, y_test[, name])` tuple. If `None`, call `add_dataset(...)` before `run()`.
`data_root`	`None`	Root directory for benchmark IDs; falls back to `$MUVIS_C_DATA_DIR` or `~/.muvis_c/data/`.
`failures`	all 10 built-in	Explicit list of `SeverityFailure` instances (replaces defaults). Mutually exclusive with `exclude_failures`.
`exclude_failures`	`None`	Failure-mode classes to remove from the default set, e.g. `[rob.Outliers]`.
`target_features`	`"all"`	Feature indices to corrupt.
`severity_steps`	`20`	Number of severity increments in $[0, 1]$ (yields 21 evaluation points).
`k`	`3.0`	Default maximum perturbation scale passed to built-in failures.
`seed`	`42`	Random seed for reproducibility.
`device`	`"auto"`	`"cuda"`, `"mps"`, `"cpu"`, or auto-detect.
`scaler`	`StandardScaler()`	Per-channel scaler fit on the training set; `None` skips scaling.
`batch_size`	`256`	Batch size used during the sweep.

Table 22: Aggregate metrics on `Results`. RMSE-derived quantities are computed on rows with $s > 0$ unless noted.

Method	Definition	Reading
`nominal_rmse()`	Clean RMSE	Lower is better.
`normalized_performance()`	$\mathrm{RMSE}_{\text{clean}}/\mathrm{RMSE}_{\text{baseline}}$	<1 beats the naive baseline.
`mpc()`	Mean RMSE under corruption (mPC)	Same units as RMSE; lower is better.
`npc()`	$\mathrm{mPC}/\mathrm{RMSE}_{\text{baseline}}$	Scale-free; lower is better.
`rpc()`	$\mathrm{mPC}/\mathrm{RMSE}_{\text{clean}}$	1.0 = no degradation.
`rpc_per_failure()`	rPC per failure mode (averaged over datasets)	Index = failure mode.
`crossing_severity()`	Smallest $s > 0$ at which any (failure, feature) reaches $\mathrm{RMSE}_{\text{baseline}}$	Higher is better; `NaN` = no crossing.
`mpc_hard_fault_zero()`	mPC restricted to hard-fault rows at $s = 0$	Loss of a single channel.
`rpc_hard_fault_zero()`	$\mathrm{mPC}_{\text{hf0}}/\mathrm{RMSE}_{\text{clean}}$	Scale-free counterpart.

Multi-dataset evaluation. Per-dataset `Results` objects are combined with the static `Results.merge`; the combined object exposes all aggregate methods above:

```
all_results = []
for ds_id in rob.BENCHMARK_DATASETS:
    tb = rob.Testbed(dataset=ds_id)
    tb.add_model("MyModel", predict_fn=load_model(ds_id))
    all_results.append(tb.run())

combined = rob.Results.merge(all_results)
combined.summary()
```