From Faults to Features: Pretraining to Learn Robust Representations against Sensor Failures

Jens U. Brandt[1*] **Noah C. Puetz**[1] **Marcus Greiff**[2] **Thomas Lew**[2] **John Subosits**[2]
Marc Hilbert[3] **Thomas Bartz-Beielstein**[1]
[1]TH Köln [2]Toyota Research Institute
[3]Toyota Gazoo Racing Europe

Abstract

Machine learning models play a key role in safety-critical applications, such as autonomous vehicles and advanced driver assistance systems, where their robustness during inference is essential to ensure reliable operation. Sensor faults, however, can corrupt input signals, potentially leading to severe model failures that compromise reliability. In this context, pretraining emerges as a powerful approach for learning expressive representations applicable to various downstream tasks. Among existing techniques, masking represents a promising direction for learning representations that are robust to corrupted input data. In this work, we extend this concept by specifically targeting robustness to sensor outages during pretraining. We propose a self-supervised masking scheme that simulates common sensor failures and explicitly trains the model to recover the original signal. We demonstrate that the resulting representations significantly improve the robustness of predictions to seen and unseen sensor failures on a vehicle dynamics dataset, maintaining strong downstream performance under both nominal and various fault conditions. As a practical application, we deploy the method on a modified Lexus LC 500 and show that the pretrained model successfully operates as a substitute for a physical sensor in a closed-loop control system. In this autonomous racing application, a supervised baseline trained without sensor failures may cause the vehicle to leave the track. In contrast, a model trained using the proposed masking scheme enables reliable racing performance in the presence of sensor failures.

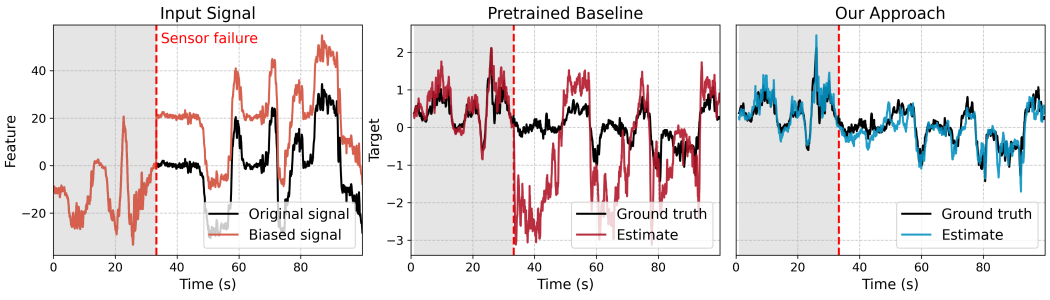

Figure 1: Although traditional pretraining with masking produces representations suitable for downstream tasks, these representations can lack robustness to common sensor failures like signal offsets. In contrast, our proposed pretraining scheme yields representations that are robust to sensor faults.

*Contact: jens_uwe.brandt@th-koeln.de

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

1 Introduction

Sensor failures pose a significant challenge to the reliable deployment of machine learning models in safety-critical domains, such as autonomous driving and robotics [Durlik et al., 2024, Ji et al., 2022]. Failures can stem from hardware malfunctions, challenging environmental conditions, or malicious attacks [Schober et al., 2022, Balaban et al., 2009]. To improve reliability in such circumstances, any model ingesting raw sensor data needs to be robust to distribution shifts [Braiek and Khomh, 2024, Freiesleben and Grote, 2023].

Self-supervised learning (SSL) offers a promising solution to this problem, enabling the extraction of meaningful representations from unlabeled data. The learned representations can be used for various downstream tasks, such as classification, regression, or anomaly detection [Zhang et al., 2024, Zerveas et al., 2021]. Masked Image Modeling (MIM) is a major family of SSL methods that aims to train generative models capable of predicting an original image from a corrupted or masked version [Balestriero et al., 2023]. In masked modeling for time series applications, common approaches involve replacing randomly selected segments of the input sequence with a constant (e.g., zero) or a learnable value, and subsequently training the model to reconstruct the original unmasked input [Zerveas et al., 2021, Goswami et al., 2024, Dong et al., 2023, Nie et al., 2022]. This masking strategy encourages the model to learn robust representations that effectively capture the underlying patterns and structures in the data, even when missing significant portions of the inputs. This body of work suggests that masked modeling is suitable for learning representations robust to sensor outages. However, while Balaban et al. [2009] and Jesus et al. [2017] outline sensor failure modes that are predominantly characterized by additive noise and constant or time-dependent offsets, these differ markedly from the masks typically used in masked modeling. This suggests that sensor outages remain out-of-distribution even after pretraining, potentially leading to performance degradation.

Motivated by this discrepancy and the need for robust sensing in safety-critical applications, we propose a novel self-supervised masking scheme for robust representation learning that explicitly targets sensor failure robustness by incorporating three common outage scenarios into the reconstruction objective. We further systematically analyze the effectiveness of our method on a dataset from the vehicle dynamics domain and demonstrate it in a real-world application involving autonomous racing.

2 Related work

Early foundational work by Vincent et al. [2010] introduces stacked denoising autoencoders, demonstrating how models pretrained to reconstruct inputs subject to noise using a local denoising criterion can learn robust and invariant features. Building upon this concept, more recent work in the vision domain by He et al. [2022] introduces Masked Autoencoder (MAE), which is trained to reconstruct original inputs from partially masked data, encouraging the model to learn robust representations.

Zerveas et al. [2021] proposes a transformer-based self-supervised framework specifically designed for multivariate time series. Their approach demonstrates improved performance in downstream classification and regression tasks by employing a masked input modeling strategy, where portions of the input are masked (set to zero). Yuan et al. [2024] explores multi-task SSL using large unlabeled accelerometer datasets from the UK Biobank. Their methodology incorporates multiple common time-series augmentation techniques, such as reversing, permuting, and time-warping, alongside a multi-head architecture for pretraining. Their findings highlight the strong generalization capabilities of pretrained deep convolutional neural networks when finetuned on diverse activity recognition benchmarks. Narayanswamy et al. [2024] introduces LSM, a multi-modal representation learning model for wearable sensor data. This study of the LSM explores scaling laws and demonstrates superior few-shot learning performance compared to established benchmarks by combining a vision transformer architecture with various masking strategies.

To further evaluate the reliability of deep learning methods beyond standard benchmarks, Hendrycks and Dietterich [2019] provides a dataset for common image corruptions. Building on this, Hendrycks et al. [2019a,b] show that an additional self-supervised loss significantly enhances model resilience against common input corruptions, adversarial attacks, and label corruptions. They furthermore show improved robustness for the latter two cases using adversarial pretraining following Madry et al. [2018], with a model pretrained on ImageNet using adversarial training and then fine-tuned on CIFAR-10 and CIFAR-100.

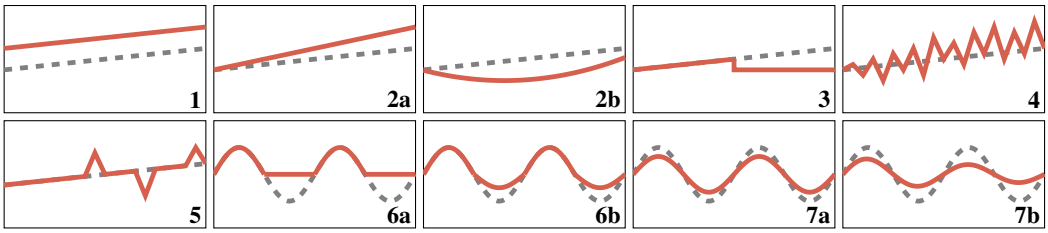

Figure 2: Sensor failure modes. The faulty sensor output $\tilde{x}$ (red) and underlying signal x (gray).

Alternative approaches include data augmentation and domain randomization, which both expose models to varied input conditions during training. Adversarial training represents a specialized form of data augmentation that enhances resilience by training on adversarially generated examples designed to maximize prediction errors [Shorten and Khoshgoftaar, 2019, Tobin et al., 2017]. Imputation strategies address missing or corrupted data by reconstructing absent values to support robust inference [Wang et al., 2024]. While these approaches advance robustness across various domains, real-world sensor failures often exhibit structured patterns, such as persistent biases, drifts, or hard faults, that differ from the random or adversarial corruptions typically considered.

These findings motivate the design of a pretraining scheme that explicitly prepares models for real-world sensor failures. To date, the impact of pretraining on common input corruptions, especially in the domain of multivariate time series data, remains underexplored. In this setting, we aim to train models for accurate predictions under realistic sensor failures and input disturbances (see Figure 1).

3 Sensor failure modes and robustness via reconstruction

3.1 Sensor failure modes

Sensor faults manifest as unexpected deviations in measurements [Balaban et al., 2009]. These deviations reflect distinct failure modes independent of their physical root causes or application domain. A systematic classification of these modes is essential for fault diagnosis and mitigation. Following the taxonomy of Balaban et al. [2009] (1,2,7) and Jesus et al. [2017] (3-6), we define the principal failure modes relative to a nominal sensor model representing normal operation: $x = x_{\text{nom}} + \epsilon_{\text{nom}}$, where x_{nom} is the true signal and ϵ_{nom} is sensor noise. We define various fault functions $D(\cdot)$ that act on x to produce the faulty sensor signal, $\tilde{x}$, under different failures (see Figure 2).

1. **Bias**. An offset from the true measurement: $D(x) = x + \beta$, where β is a constant offset.
2. **Drift**. An offset from the true measurement: $D(x) = x + \beta(t)$ where the time-varying offset $\beta(t)$ is either (a) linear or (b) nonlinear in t.
3. **Hard fault**: The sensor measurement is stuck at a constant value C: $D(x) = C$.
4. **Noise**. Additive zero-mean noise: $D(x) = x + \epsilon$, with $\text{Var}(\epsilon) \gg \text{Var}(\epsilon_{\text{nom}})$.
5. **Outliers**. Occasional deviations from the true measurement at random times: $D(x) = x + \gamma(t)\delta(t)$, where $\gamma(t) \in \{0, 1\}$ indicates outlier presence and the $\delta(t)$ is sampled uniformly over $[l, u]$.
6. **Trimming**. Signal values are restricted to $[l, u]$ and, for values outside these bounds, are either (a) fixed at the boundary or (b) vary proportionally with their deviation from the boundary:
$$D(x) = \begin{cases} l + \alpha(x - l) & \text{if } x < l \\ u + \alpha(x - u) & \text{if } x > u \\ x & \text{otherwise} \end{cases}$$
We consider (a) hard clamping with $\alpha = 0$, and (b) controlled dampening strength with $0 < \alpha < 1$.
7. **Scaling**. A multiplicative error affecting signal amplitude: $D(x) = \alpha(t)$ where (a) the gain α is constant, or (b) $\alpha(t)$ is a function of time t.

These sensor failure modes can occur due to manufacturing errors, wear and tear with long-term usage, incorrect calibration, or mishandling [Balaban et al., 2009], and they can have detrimental or even catastrophic effects on the performance of machine learning models, especially when deployed in real-world robots and autonomous vehicles.

3.2 Robustness via reconstruction

Reconstruction-based SSL involves training models to reconstruct original input data that has been partially masked or corrupted. For language and image data, common tasks include predicting missing words or pixels, image inpainting, and colorization [Balestriero et al., 2023]. In multivariate time series, this typically involves masking intervals of the input with zeros [Zerveas et al., 2021] or learnable masks [Goswami et al., 2024]. Early SSL methods, such as denoising autoencoders [Vincent et al., 2010], corrupt inputs by masking features and calculate the loss based on the full reconstructed input. Grounded in the manifold hypothesis [Chapelle et al., 2006]—that natural high-dimensional data reside near a non-linear low-dimensional manifold—this corruption process pushes data away from the manifold, forcing the neural network to project corrupted inputs back onto it by capturing essential input features [Vincent et al., 2010]. Modern SSL methods modify the reconstruction objective. Approaches like BERT [Devlin et al., 2019] and MAEs [He et al., 2022] compute the reconstruction loss solely on masked input regions. This design choice is deliberate: by not requiring the model to reproduce the already visible parts, it avoids wasting capacity on trivial copying, encouraging the model to focus on inferring the missing content from the context provided by the unmasked parts, emphasizing the learning of global feature dependencies.

We hypothesize that, although calculating the loss only on masked regions relaxes the requirement of exact reconstruction, models still implicitly project inputs onto the data manifold. This occurs because repeatedly training on diverse mask configurations fosters a universal representation that allows reconstructions coherent with the data's true structure. Following the argumentation of Vincent et al. [2010], since the learned representation is robust to input masking by design, utilizing this training paradigm should enhance robustness to similar input perturbations in downstream tasks. Models trained with partial context learn stable patterns and dependencies, becoming resilient to variations or corruptions. Thus, they retain coherent internal representations under noisy or incomplete inputs, potentially benefiting real-world applications. However, the perturbations encountered in real-world scenarios, such as the diverse sensor failure modes described in Section 3.1, can differ substantially from the masks typically employed during pretraining, potentially limiting direct transferability.

4 Method

4.1 A self-supervised pretraining method to learn robust representations

We propose a pretraining scheme that simulates real-world sensor failures by introducing artificial faults to the input data and training the model to reconstruct the original signal (see Figure 3). Following the sensor failure definitions in Section 3.1, we formulate the pretraining objective as a multi-task learning problem with n distinct masking functions $\{D_i\}_{i=1}^{n}$. Formally, consider an input tensor $\mathbf{X} \in \mathbb{R}^{B \times S \times F}$, where B is the batch size, S is the sequence length, and F is the number of channels. We partition each batch $\mathbf{X}$ into n sub-batches $\{\mathbf{X}_i\}_{i=1}^{n}$, where each $\mathbf{X}_i \in \mathbb{R}^{B_{\text{sub}} \times S \times F}$ has size $B_{\text{sub}} = B/n$ and corresponds to one of the n distinct masking functions D_i.

For each sample in sub-batch $\mathbf{X}_i$, we generate a binary mask tensor $\mathbf{M}_i \in \{0, 1\}^{B_{\text{sub}} \times S \times F}$ indicating the indices to be masked. The masking process follows the geometric distribution sampling strategy from Zerveas et al. [2021], controlled by two parameters: the overall masking ratio mr (proportion of total elements to mask per sample) and the average mask block length lm. This strategy generates contiguous masked spans, making reconstruction challenging via pure interpolation and promoting the learning of feature inter-dependencies. An element $M_{b,s,f} = 0$ indicates that the b-th sample within sub-batch i is masked at timestep s and channel f. Once the mask tensor $\mathbf{M}_i$ is generated, the corresponding sensor failure-inspired masking strategy is applied directly to this mask. $\tilde{\mathbf{X}}_i$ denotes the respective masked sub-batch, where the masked regions are corrupted according to the chosen strategy. The implemented masking strategies, which are applied after feature-wise z-score normalization of the input data, are detailed below.

Mean masking: Sets the values at the masked indices to the respective channel mean $C_{b,f} = \mu_f$ resulting in a tensor $\mathbf{C}_i \in \mathbb{R}^{B_{\text{sub}} \times S \times F}$ (see Section 3.1, case 3). Since the input data $\mathbf{X}_i$ undergoes channel-wise z-score normalization before masking, the mean of each channel is zero. Consequently, replacing masked values with μ_f is equivalent to setting them to zero. This simplification allows us to implement mean masking by directly multiplying the normalized input with the binary mask $\mathbf{M}_i$

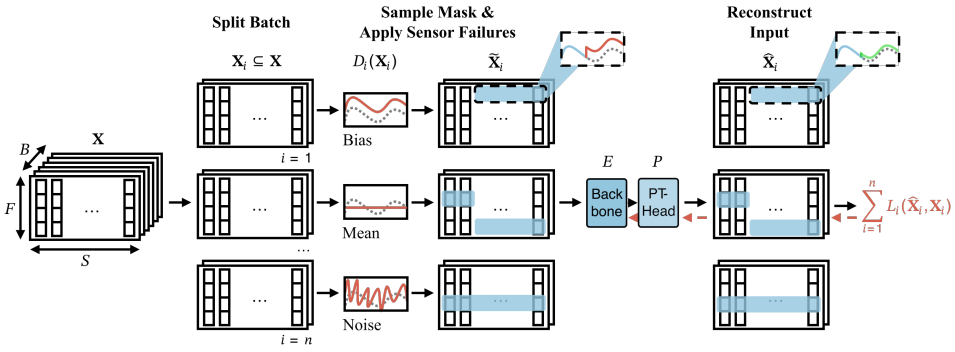

Figure 3: Pretraining scheme: The input $\mathbf{X}$ is split into n sub-batches $\mathbf{X}_i$, each associated with a sensor failure mode D_i. For each sub-batch, a binary mask identifies positions to be corrupted according to D_i. The encoder E maps the corrupted input $\tilde{\mathbf{X}}_i$ to a latent representation, which is passed through the pretraining head P to produce a reconstruction $\hat{\mathbf{X}}_i$. The model is trained to reconstruct the original input by minimizing the sum of n reconstruction losses.

(where $M_{b,s,f} = 0$ indicates a masked position). Formally, the corrupted signal is given by

$$\tilde{\mathbf{X}}_i = \mathbf{X}_i \odot \mathbf{M}_i, \tag{1}$$

where $\odot$ denotes element-wise multiplication. This approach is commonly used in masked modeling for time series data and reflects a standard fallback in technical systems.

Bias masking: Introduces a constant offset to the masked regions (see Section 3.1, case 1). Specifically, for each b-th sample and f-th feature, a bias value $C_{b,f}$ is drawn independently from a uniform distribution $C_{b,f} \sim \mathcal{U}(l\sigma_{[f]}, u\sigma_{[f]})$, where l is the lower bound of the uniform distribution and u is the upper bound multiplied with the channel-wise standard deviation $\sigma_{[f]}$. This sampled bias $C_{b,f}$ is then replicated across all S time steps to form the bias tensor $\mathbf{C}_i \in \mathbb{R}^{B_{\text{sub}} \times S \times F}$. Normalization eliminates the need for separate bounds in each channel. We let

$$\tilde{\mathbf{X}}_i = \mathbf{X}_i + \mathbf{C}_i \odot (\mathbf{1} - \mathbf{M}_i), \tag{2}$$

where $\mathbf{1}$ is a tensor of ones matching $\mathbf{M}_i$'s dimensions, and the element-wise subtraction inverts $\mathbf{M}_i$. Each channel has a single sampled offset, and $\mathbf{M}_i$ ensures that these offsets are applied only to the masked positions. We assume that for short observed sequences, the bias error serves as an effective proxy for drift, outliers, trimming, and scaling behaviors. For domains with lower sampling rates or longer sequences, including additional fault types in pretraining could be beneficial.

Noise masking: Adds random Gaussian noise to the masked regions (see Section 3.1, case 4). Specifically, a zero-mean noise tensor $\mathbf{C}_i \in \mathbb{R}^{B_{\text{sub}} \times S \times F}$ is drawn independently for each element $C_{b,s,f} \sim \mathcal{N}(0, r\sigma^2_{[f]})$, where r is a scaling factor for the channel-wise variance. The noise is applied as in Equation (2).

Loss calculation: Once the sub-batches $\{\tilde{\mathbf{X}}_i\}_{i=1}^n$ with their respective masking corruptions are prepared, they are sequentially processed by the model. The reconstruction loss (Mean Squared Error over masked elements) is defined as

$$\mathcal{L} = \sum_{i=1}^{n} \left[\frac{1}{|\mathcal{M}_i|} \sum_{(b,s,f) \in \mathcal{M}_i} \left([X_i]_{b,s,f} - [\hat{X}_i]_{b,s,f} \right)^2 \right], \tag{3}$$

where $\mathcal{M}_i = \{(b, s, f) \mid [M_i]_{b,s,f} = 0\}$ is the set of masked indices in $\mathbf{M}_i$, $|\mathcal{M}_i|$ denotes its cardinality, and $[\hat{X}_i]_{b,s,f}$ is the reconstruction of the original value $[X_i]_{b,s,f}$ at the index (b, s, f), given the masked input $[\tilde{\mathbf{X}}_i]_b$. The implementation follows the masked reconstruction paradigm from time series, vision, and language modeling [Zerveas et al., 2021, He et al., 2022, Devlin et al., 2019].

Architecture: Our encoder-backbone is based on the transformer architecture described by Zerveas et al. [2021]. The encoder E processes the corrupted input $\tilde{\mathbf{X}}_i$:

$$\mathbf{z}_i = E(\tilde{\mathbf{X}}_i, \theta_E), \tag{4}$$

where $\mathbf{z}_i \in \mathbb{R}^{B_{\text{sub}} \times S \times d_{\text{model}}}$ is the latent representation that captures temporal dependencies and contextual information from the input, and θ_E are the learnable parameters. d_{model} is the dimension of the initial input projection. The sequence dimension of the latent representation is concatenated:

$$\mathbf{z}_i^{\text{cat}} = \text{Concat}(\mathbf{z}_i) \in \mathbb{R}^{B_{\text{sub}} \times (S \cdot d_{\text{model}})}. \tag{5}$$

This concatenated representation is then processed by the pretraining head $P(\cdot)$, a multi-layer perceptron with parameters θ_P, which performs the reconstruction by mapping the latent representation back into the input space

$$\hat{\mathbf{X}}_\mathbf{i} = P(\mathbf{z}_i^{\text{cat}}, \theta_P). \tag{6}$$

4.2 Finetuning procedure for downstream use

For finetuning on a supervised task, we replace the pretraining head P with a finetuning head F (again a multi-layer-perceptron). The encoder is then frozen; its parameters, $\theta_E^\star$, are not updated during the finetuning procedure. Only the parameters θ_F of the finetuning head are trained. For a finetuning batch $\mathbf{X} \in \mathbb{R}^{B_{FT} \times S \times F}$ where B_{FT} is the finetuning batch size, and associated labels y, the frozen encoder E computes the latent representation

$$\mathbf{z} = E(\mathbf{X}, \theta_E^\star), \tag{7}$$

which is concatenated as in Equation (5) and mapped to the target space

$$\hat{y} = F(\mathbf{z}^{\text{cat}}, \theta_F). \tag{8}$$

The finetuning objective is then

$$\mathcal{L}_{\text{FT}} = \ell\Big(F\big(\text{Concat}(E(\mathbf{X}, \theta_E^\star)), \theta_F\big),\, y\Big), \tag{9}$$

where $\ell(\cdot, \cdot)$ denotes a mean squared error loss function. This procedure leverages the robust representations learned during pretraining while adapting the finetuning head to the downstream task.

5 Sensor failure robustness results

We evaluate our method on the problem of vehicle sideslip angle (VSA) estimation, as accurate VSA estimation is necessary for vehicle stability and control [Liu et al., 2020]. Modern advanced driver-assistance systems, such as Electronic Stability Control, utilize the VSA to adjust brake forces at each wheel, thereby mitigating oversteer and understeer and enhancing overall performance and safety [Liu et al., 2020, Chindamo et al., 2018, Bonfitto et al., 2020]. However, systems that directly measure the VSA can be prohibitively expensive for consumer vehicles, necessitating alternative estimation methods. Virtual sensing, which leverages machine learning models to estimate the VSA from other sensors, such as wheel speed sensors in the car, offers a cost-effective solution. Yet virtual sensing is prone to sensor failures, motivating the development of robust models. VSA estimation involves heterogeneous sensor inputs (vehicle control signals, wheel speeds, and accelerations) with diverse statistical properties. Since similar modalities are used in robotics and aviation, the study of VSA estimation may have broader implications for robust estimation beyond the domain of driving.

The VSA is the angle between the vehicle's direction of travel and the heading angle of the vehicle [Liu et al., 2020], expressed as a function of the vehicle's longitudinal (v_x) and lateral (v_y) velocities:

$$\beta = \arctan\left(\frac{v_y}{v_x}\right). \tag{10}$$

In practice, the VSA can be estimated directly or computed from the estimated lateral velocities in combination with the typically more accessible longitudinal velocities. Both approaches are common in virtual sensing, depending on the available sensor data and model design [Liu et al., 2020]. In the following, we first assess the effectiveness of our approach on a public dataset augmented with simulated sensor failures and then validate it in a real-world autonomous racing scenario.

5.1 Simulation of sensor failures

The REVS Program Vehicle Database [Kegelman et al., 2016a,b] comprises measurements collected from expert drivers operating under real-world racing conditions at two race tracks, providing data representative of extreme vehicle dynamics and operating scenarios. We evaluate the proposed method under several simulated sensor outages on two test sets (one from the same track and another from an unseen track) and compare its performance to a classical masking scheme and a non-pretrained baseline, respectively. Details on the dataset, features, and splitting are given in Section A.

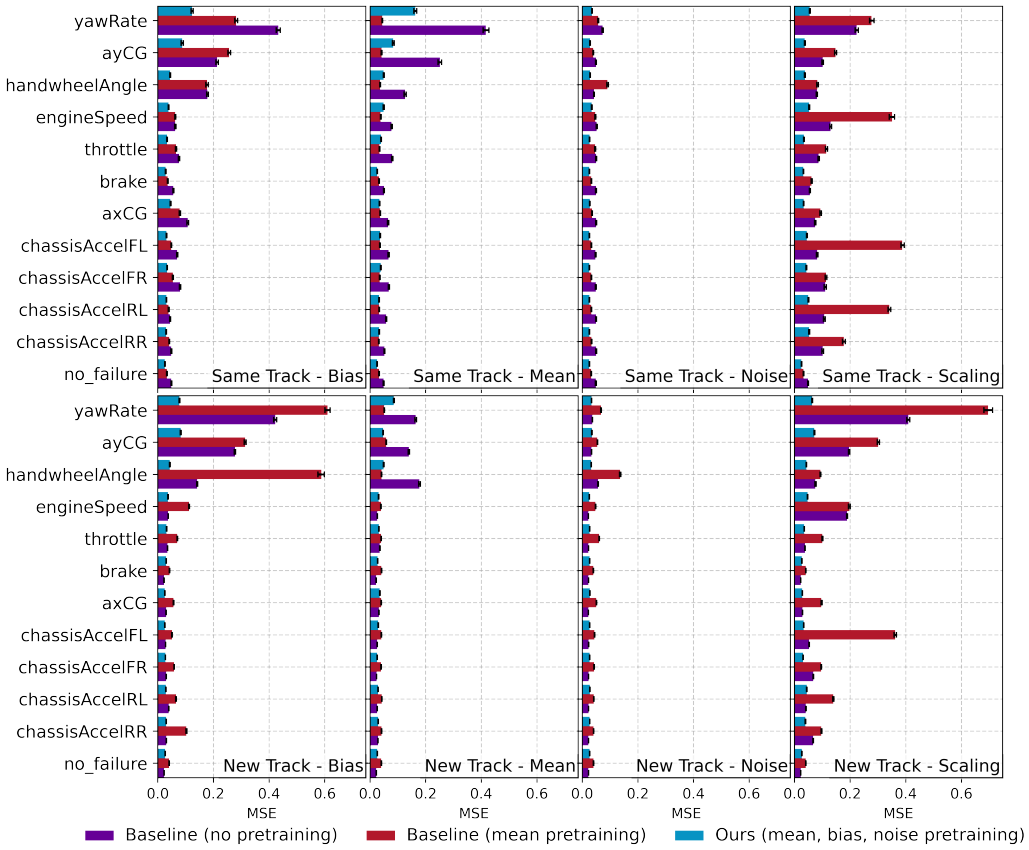

Figure 4: Lateral velocity MSE under individual sensor failures on seen and unseen tracks for the non-pretrained baseline (purple), mean-only pretrained baseline (red), and our multi-mask approach (blue). Bootstrapped means (200 resamples) with 95% confidence intervals on the test set.

We pretrain the model using 11 sensor signals with a fixed sequence length of 20 and a 20 Hz sampling rate. To align the pretraining task close to actual sensor failures, we set the average mask length lm to 20—masking an entire channel on average—and use a masking ratio mr of 0.1 to retain sufficient information in the remaining channels for reconstruction. We set the parameters of the noise mask to $\mu_f = 0$ and $r = 1.0$ and the parameters of the bias mask to $l = -3$ and $u = 3$ (see Section 4.1). Further implementation details can be found in Section B. As a downstream task, we fine-tune an estimator that learns to estimate the lateral velocity.

We evaluate the robustness using the three pretraining failure modes (bias: one-standard-deviation offset; noise: zero-mean, standard deviation of the respective feature times 0.4; hard faults: channel mean replacement), an unseen scaling failure ($\alpha = 2$) to test generalization to novel sensor faults, and nominal conditions (see Section 3.1). Each error is applied separately to every channel. We report the lateral velocity MSE (Figure 4) with: model pretrained on all three corruptions (Mean, Bias, and Noise); a non-pretrained baseline; and a mean-only pretrained baseline (traditional zero masking).

Nominal conditions: Without sensor corruptions, our method improves downstream performance on the seen track and matches the performance of the non-pretrained baseline on the unseen track.

Seen failures: Under noise corruptions, pretraining yields subtle performance gains. For sensor outages simulated by mean-value replacement, the model pretrained solely on mean masking exhibits the best robustness. However, its robustness against bias errors on an unseen track degrades below the non-pretrained baseline. Our approach achieves the strongest robustness to bias failures as evidenced by reduced error magnitude and variance. On an unseen track, the performance of our approach is largely consistent with those on the seen track.

Unseen failure: The strong estimation performance under the unseen scaling error suggests that the model is robust to sensor faults that were not used during pretraining, indicating that the learned representations generalize to out-of-distribution failure modes. Evaluations on all remaining failure modes presented in Section 3.1 and detailed in Section J show similar results, with our approach consistently improving robustness.

Ablations: Varying the masking combinations confirms that incorporating bias masking is critical for robustness to both bias and scaling errors, with our approach achieving a good trade-off (Section C). When subjecting models to increasingly severe sensor failures, our model remains robust across a wide range of bias, noise, and scaling strengths, consistently outperforming the baselines (Section D). We also note improved performance over baselines with simultaneous sensor failures, as detailed in Section I. Repeating training across ten random seeds demonstrates that our scheme is consistent, whereas the performance of a mean-only baseline varies markedly on unseen failure types (Section E). Finally, we observe that larger models tend to enhance robustness across failure modes (Section F). Additional experiments indicate that, compared to a state-of-the-art baseline (SimMTM), our approach provides enhanced robustness under sensor failure conditions (Section K). Tested in a different domain, our method also improves reliability in environmental sensing tasks (Section M).

5.2 Enhanced robustness for closed-loop control

To show the benefits of the proposed pretraining procedure, we evaluate it in a closed-loop autonomous racing scenario using the controller described in [Lew et al., 2024]. This experiment shows how estimation errors propagate and impact the coupled controller-vehicle system in a real-world application. A modified 2019 Lexus LC 500 is driven on a skidpad, with our model estimating the VSA from other available sensors (for an overview of the available sensors in this experiment and implementation details, see Section N). The estimated VSA is then used for feedback control. While racing autonomously along the planned trajectory shown in Figure 5, we inject artificial sensor outages and record the system response. The experiment is repeated with a baseline model directly trained on the VSA estimation task without initial pretraining.

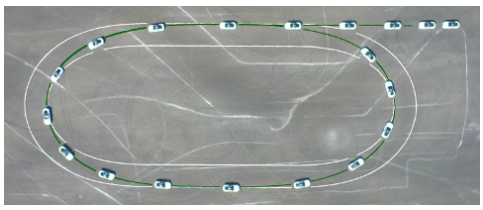

Figure 5: Nominal trajectory of the car (green) and virtual track limits (white).

Open-loop tests indicate that VSA estimation is most sensitive to potential failures in the wheel speed, steering angle, yaw rate, and longitudinal velocity sensors. Thus, we compare our approach against the baseline by corrupting one wheel-speed sensor at a time by applying heavy noise, bias, and hard faults (each channel set to its mean). Additional closed-loop experiments and details are in Section O.

The first attempt with the baseline model fails, and the safety driver must intervene after a few seconds of autonomous driving. The second attempt is shown in Figure 6. The pretrained model maintains a good VSA estimate throughout, resulting in a stable trajectory (see top-right subplot) even under challenging conditions. The baseline's VSA estimates under noise show occasional deviations with increased variance. Bias failure leads to significant offset errors (see bottom-left subplot), whereas mean input errors have minimal effect, suggesting robustness to such disturbances. Notably, the baseline's estimation leads to unstable trajectories (see top-left subplot) that cause a violation of the virtual track boundaries. Additional experiments with outages at the other sensors show similar behavior for the pretrained model (see Section O). One notable exception arises under longitudinal-velocity sensor outages: here the pretrained model also produces unstable trajectories, but the baseline's parallel open-loop estimations still degrade to a greater extent. Our approach demonstrates significant improvements in robustness across a vast majority of tested failure cases, highlighting its potential for reliable deployment in real-world applications.

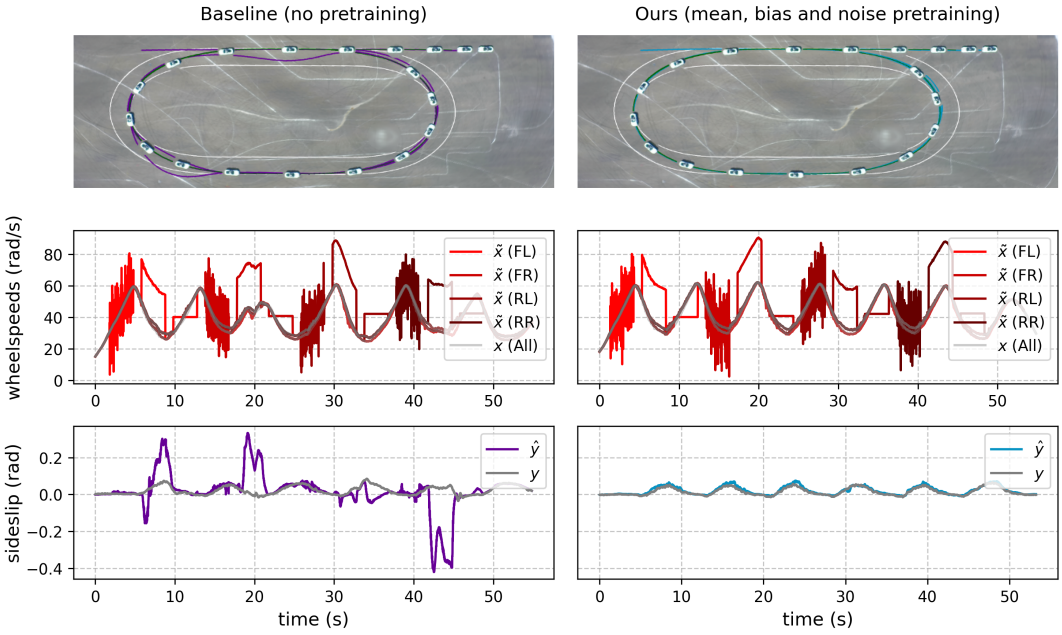

Figure 6: Robust virtual sensing in closed-loop control. Top: full driven trajectory (car image illustrative, indicating driving direction). Middle: perturbed ($\tilde{x}$) vs. true (x) wheelspeed measurements. Bottom: sideslip estimate ($\hat{y}$) under corruptions vs. true VSA (y).

6 Limitations and future work

The results demonstrate that the proposed masked pretraining significantly improves robustness against common sensor failures for driving applications. However, while the model extrapolates beyond the masks used during pretraining (see Section 5.1), studying more complex mode configurations (e.g., coupled errors) and assessing its performance in additional domains would broaden the applicability of the proposed method. The experiments show that certain sensor failures, such as errors in the longitudinal velocity, are more detrimental to closed-loop performance than others. A limitation of this work is that all failure modes across all channels are weighted equally, but we speculate that this can be addressed by re-weighting of the loss. Finally, as the method employs multiple mask-based tasks, interference between the gradient updates could lead to poor convergence during training. Leveraging advanced multi-task learning techniques, such as [Yu et al., 2020], may mitigate this and improve performance, and will be explored in future work.

7 Conclusion

In this work, we introduce a novel self-supervised masking scheme designed to strengthen model resilience by directly incorporating common sensor failure modes into the learning objective. By training models to reconstruct signals corrupted by realistic fault simulations, we foster representations significantly more robust to sensor outages than those learned via traditional masking. We demonstrate the effectiveness of this approach through extensive validation on a vehicle dynamics dataset and, through successful deployment as a virtual sensor in closed-loop control of a real autonomous vehicle operating at its limits. The results demonstrate significantly improved robustness against various sensor failures, including generalization to a fault type unseen during pretraining. We believe that by directly addressing this critical vulnerability, our work contributes to the development of resilient deep learning systems, which are essential for robust deployment in safety-critical applications.

Acknowledgements

This work is funded by the European Commission Key Digital Technologies Joint Undertaking - Research and Innovation (HORIZONKDT-JU-2023-2-RIA), under grant agreement No 101139996, the ShapeFuture project - "Shaping the Future of EU Electronic Components and Systems for Automotive Applications".

References

Ienkaran Arasaratnam and Simon Haykin. Cubature kalman filters. *IEEE Transactions on automatic control*, 54(6):1254–1269, 2009.

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.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A Cookbook of Self-Supervised Learning, June 2023. URL http://arxiv.org/abs/2304.12210. arXiv:2304.12210 [cs].

Karl Berntorp, Marcus Greiff, and Stefano Di Cairano. Bayesian sensor fusion of gnss and camera with outlier adaptation for vehicle positioning. In *2022 25th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2022.

Angelo Bonfitto, Stefano Feraco, Andrea Tonoli, and Nicola Amati. Combined regression and classification artificial neural networks for sideslip angle estimation and road condition identification. *Vehicle System Dynamics*, 58(11):1766–1787, November 2020. ISSN 0042-3114. doi: 10.1080/00423114.2019.1645860. URL https://doi.org/10.1080/00423114.2019.1645860. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00423114.2019.1645860.

Houssem Ben Braiek and Foutse Khomh. Machine Learning Robustness: A Primer, May 2024. URL http://arxiv.org/abs/2404.00897. arXiv:2404.00897 [cs].

Olivier Chapelle, Alexander Zien, and Bernhard Schölkopf, editors. *Semi-supervised learning*. Adaptive computation and machine learning series. MIT Press, Cambridge, Massachusetts, 2006. ISBN 978-0-262-03358-9 978-0-262-25589-9.

Daniel Chindamo, Basilio Lenzo, and Marco Gadola. On the Vehicle Sideslip Angle Estimation: A Literature Review of Methods, Models, and Innovations. *Applied Sciences*, 8(3):355, March 2018. ISSN 2076-3417. doi: 10.3390/app8030355. URL https://www.mdpi.com/2076-3417/8/3/355. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. SimMTM: A Simple Pre-Training Framework for Masked Time-Series Modeling. November 2023. URL https://openreview.net/forum?id=ginTcBUnL8&noteId=krCRnerqCT.

Irmina Durlik, Tymoteusz Miller, Ewelina Kostecka, Zenon Zwierzewicz, and Adrianna Łobodzińska. Cybersecurity in Autonomous Vehicles—Are We Ready for the Challenge? *Electronics*, 13 (13):2654, January 2024. ISSN 2079-9292. doi: 10.3390/electronics13132654. URL https://www.mdpi.com/2079-9292/13/13/2654.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research*, 11(19):625–660, 2010. ISSN 1533-7928. URL http://jmlr.org/papers/v11/erhan10a.html.

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.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A Family of Open Time-series Foundation Models. June 2024. URL https://openreview.net/forum?id=FVvf69a5rx&referrer=%5Bthe%20profile%20of%20Yifu%20Cai%5D(%2Fprofile%3Fid%3D~Yifu_Cai1).

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, New Orleans, LA, USA, June 2022. IEEE. ISBN 978-1-6654-6946-3. doi: 10.1109/CVPR52688.2022.01553. URL https://ieeexplore.ieee.org/document/9879206/.

Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, March 2019. URL http://arxiv.org/abs/1903.12261. arXiv:1903.12261 [cs].

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using Pre-Training Can Improve Model Robustness and Uncertainty. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2712–2721. PMLR, May 2019a. URL https://proceedings.mlr.press/v97/hendrycks19a.html. ISSN: 2640-3498.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019b. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/a2b15837edac15df90721968986f7f8e-Abstract.html.

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 Publisher: Multidisciplinary Digital Publishing Institute.

Tianchen Ji, Arun Narenthiran Sivakumar, Girish Chowdhary, and Katherine Driggs-Campbell. Proactive Anomaly Detection for Robot Navigation With Multi-Sensor Fusion. *IEEE Robotics and Automation Letters*, 7(2):4975–4982, April 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3153989. URL https://ieeexplore.ieee.org/document/9720937/.

John C. Kegelman, Lene K. Harbott, Jackie Liao, and J. Christian Gerdes. 2013 Monterey Motorsports Reunion, 2016a. URL https://purl.stanford.edu/tt103jr6546.

John C. Kegelman, Lene K. Harbott, Jackie Liao, and J. Christian Gerdes. 2013 Targa Sixty-Six, 2016b. URL https://purl.stanford.edu/yf219gg2055.

Thomas Lew, Marcus Greiff, Franck Djeumou, Makoto Suminaka, Michael Thompson, and John Subosits. Risk-Averse Model Predictive Control for Racing in Adverse Conditions, October 2024. URL http://arxiv.org/abs/2410.17183. arXiv:2410.17183 [cs].

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A System for Massively Parallel Hyperparameter Tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, March 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/a06f20b349c6cf09a6b171c71b88bbfc-Abstract.html.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A Research Platform for Distributed Model Selection and Training, July 2018. URL http://arxiv.org/abs/1807.05118. arXiv:1807.05118 [cs].

Jizheng Liu, Zhenpo Wang, Lei Zhang, and Paul Walker. Sideslip angle estimation of ground vehicles: a comparative study. *IET Control Theory & Applications*, 14(20):3490–3505, 2020. ISSN 1751-8652. doi: 10.1049/iet-cta.2020.0516. URL https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-cta.2020.0516. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-cta.2020.0516.

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.

Girish Narayanswamy, Xin Liu, Kumar Ayush, Yuzhe Yang, Xuhai Xu, Shun Liao, Jake Garrison, Shyam A. Tailor, Jacob Sunshine, Yun Liu, Tim Althoff, Shrikanth Narayanan, Pushmeet Kohli, Jiening Zhan, Mark Malhotra, Shwetak Patel, Samy Abdel-Ghaffar, and Daniel McDuff. Scaling Wearable Foundation Models. October 2024. URL https://openreview.net/forum?id=yb4QE6b22f.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. September 2022. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

Sebastian A. Schober, Yosra Bahri, Cecilia Carbonelli, and Robert Wille. Neural Network Robustness Analysis Using Sensor Simulations for a Graphene-Based Semiconductor Gas Sensor. *Chemosensors*, 10(5):152, April 2022. ISSN 2227-9040. doi: 10.3390/chemosensors10050152. URL https://www.mdpi.com/2227-9040/10/5/152.

Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60, July 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0197-0. URL https://doi.org/10.1186/s40537-019-0197-0.

Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I. Webb. Monash University, UEA, UCR Time Series Extrinsic Regression Archive, October 2020. URL http://arxiv.org/abs/2006.10996. arXiv:2006.10996 [cs, stat].

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, March 2017. URL http://arxiv.org/abs/1703.06907. arXiv:1703.06907 [cs].

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010. ISSN 1532-4435.

Jun Wang, Wenjie Du, Wei Cao, Keli Zhang, Wenjia Wang, Yuxuan Liang, and Qingsong Wen. Deep Learning for Multivariate Time Series Imputation: A Survey. *CoRR*, January 2024. URL https://openreview.net/forum?id=RhIkAjwkG3&referrer=%5Bthe%20profile%20of%20Wenjie%20Du%5D(%2Fprofile%3Fid%3D~Wenjie_Du1).

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient Surgery for Multi-Task Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html.

Hang Yuan, Shing Chan, Andrew P. Creagh, Catherine Tong, Aidan Acquah, David A. Clifton, and Aiden Doherty. Self-supervised learning for human activity recognition using 700,000 person-days of wearable data. *npj Digital Medicine*, 7(1):1–10, April 2024. ISSN 2398-6352. doi: 10.1038/s41746-024-01062-3. URL https://www.nature.com/articles/s41746-024-01062-3. Publisher: Nature Publishing Group.

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, pages 2114–2124, New York, NY, USA, August 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467401. URL https://dl.acm.org/doi/10.1145/3447548.3467401.

Kexin Zhang, Qingsong Wen, Chaoli Zhang, Rongyao Cai, Ming Jin, Yong Liu, James Zhang, Yuxuan Liang, Guansong Pang, Dongjin Song, and Shirui Pan. Self-Supervised Learning for Time Series Analysis: Taxonomy, Progress, and Prospects, April 2024. URL http://arxiv.org/abs/2306.10125. arXiv:2306.10125 [cs].

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. doi: 10.1098/rspa.2017.0457. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2017.0457. Publisher: Royal Society.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting, March 2021. URL http://arxiv.org/abs/2012.07436. arXiv:2012.07436 [cs].

A Data and preprocessing

The dataset, recorded in 2013 and available at `https://cars.stanford.edu/datadriven/revs-program-vehicle-dynamics-database`, contains various vehicle dynamics measurements. The models were trained on a subset of the 2013 Monterey Motorsports Reunion dataset and tested on a holdout CSV from the same dataset, as well as the 2013 Targa Sixty Six dataset, which is used to test the performance of the models on an unseen track. All measurements were taken on a 1963 Corvette Grand Sport. Two CSV files exhibiting anomalous events (engine issue and vehicle spinning) were excluded. We down-sampled the data to 20 Hz.

Table 1: CSV files used for training and testing

Train Data	**Test - Same Track**	**Test - New Track**	**Excluded**
20130810_01_01_01	20130811_02_01_01	20130222_01_01_03	20130817_01_01_02
20130810_02_01_01		20130222_01_02_03	20130815_01_01_02
20130811_01_01_01		20130222_02_01_03	
20130816_01_01_02		20130223_01_01_03	
20130817_02_01_02		20130223_01_02_03	

The collected signals include parameters related to vehicle motion, engine performance, control inputs, and positional data. Table 2 lists all channels we used for training our models.

Table 2: Input channels and target for downstream task

Feature	**Description**
vyCG (target)	Lateral velocity (m/s)
engineSpeed	Engine speed (RPM)
handwheelAngle	Handwheel angle (deg)
throttle	Throttle position ($\%$)
brake	Brake pedal force ($\%$)
axCG, ayCG	Longitudinal and lateral accelerations (m/s^2)
yawRate	Yaw rate (deg/s)
chassisAccel (FL, FR, RL, RR)	Vertical chassis accelerations (m/s^2)

All rows where the longitudinal velocity vxCG was below 1 m/s were removed, as data in this range exhibited outliers in the sideslip angle, likely resulting from its computation based on longitudinal and lateral velocities. Given Equation (10), small measurement noise in vyCG can lead to large errors in the sideslip angle when vxCG is low. Input data were standardized using z-score normalization, and 10% of the training set was used for validation during training.

B Implementation details

All models were trained using the same architecture and hyperparameters, except for the regularization parameters of the non-pretrained baseline model. The pretrained models (Mean, Bias, and Noise, and only Mean masking) were pretrained for 600 epochs and then finetuned for 50 epochs. The non-pretrained baseline was trained for 50 epochs. The models were implemented in PyTorch and trained on a single NVIDIA H100 GPU. The code is available at `https://github.com/JBrandt97/FaultsToFeatures`.

Pretraining

Model architecture: The proposed model is based on a transformer architecture composed of a backbone and a projection head. The transformer backbone consists of 4 encoder layers, each with an embedding dimension (d_{model}) of 512, 16 attention heads, and a feedforward network dimension of 2048. GELU activations, batch normalization, and residual connections are employed within each transformer block. Dropout is not applied. The projection head transforms the flattened transformer encoder output (sequence length of 20 time steps $\times$ 512 dimensions = 10,240 dimensions) into an

output dimension of 220 (20 time steps $\times$ 11 features) through an intermediate hidden layer of size 512. Similar to the backbone, dropout is set to 0.

Optimization procedure: The model was trained for 600 epochs using an AdamW-based optimizer with custom hyperparameters. The learning rate schedule (cosine) consisted of an initial learning rate of $3.6040 \cdot 10^{-5}$, increased linearly to a peak learning rate of $6.6039 \cdot 10^{-4}$, and then decreased to a minimum learning rate of $6.3811 \cdot 10^{-5}$. Gradient clipping was applied with a maximum norm of 1.0. The optimization parameters β_1 and β_2 were set to 0.96075 and 0.96127, respectively. Weight decay was not used, and a linear warm-up strategy scaled the learning rate from an initial division factor of 20 (total training steps divided by 20) to the peak learning rate. Training was conducted with a batch size of 4096.

Finetuning

Regression head: The pretraining projection head was exchanged with the finetuning regression head. The regression head flattens the latent space and consists of two intermediate layers with 256 and 64 neurons, respectively, and a final output layer with 1 neuron. The activation function is ReLU, and dropout is set to 0.32 and 0.14.

Optimization procedure: The finetuning head was trained for 50 epochs using an AdamW-based optimizer. The learning rate schedule (cosine) involved an initial learning rate of $8.6013 \cdot 10^{-5}$, linearly increasing to a peak learning rate of $4.233 \cdot 10^{-4}$, and subsequently decreasing to a minimum learning rate of $1.987 \cdot 10^{-5}$. Gradient clipping was applied with a maximum norm of 1.0. The Adam optimizer parameters β_1 and β_2 were set to 0.81596 and 0.96962, respectively. A weight decay of 0.0584 was used. A linear warm-up strategy scaled the learning rate from an initial division factor of 10 (total training steps divided by 10) up to the peak learning rate. The encoder backbone remained frozen during this phase.

Non-pretrained baseline model

This baseline utilizes the same transformer backbone and regression head architecture employed during the fine-tuning stage of the pretrained models. However, it was trained directly for 50 epochs without any pretraining phase, and the encoder backbone was not frozen. While the non-pretrained baseline model used the same learning rate schedule and AdamW optimizer parameters (β_1, β_2) as the finetuning stage of the pretrained models, its regularization parameters—specifically weight decay, backbone dropout, and gradient clipping norm—were tuned independently. This separate tuning was necessary as pretraining can act as a form of regularization [Erhan et al., 2010], which the model lacks. The dropout rate in the backbone was set to 0.04, and the weight decay was set to 0.001. Gradient clipping was applied with a maximum norm of 3.0.

Hyperparameter selection

The hyperparameters for pretraining, finetuning, and the non-pretrained baseline model were determined through a combination of automated hyperparameter optimization and informed selection based on established practices in related literature. For the automated tuning part, we utilized Ray Tune [Liaw et al., 2018] with an Asynchronous Successive Halving Algorithm (ASHA) scheduler [Li et al., 2020]. Architectural choices and certain training configurations also drew inspiration from successful transformer-based time series models such as Informer [Zhou et al., 2021] and the work by Zerveas et al. [2021].

Experiments compute resources

All pretraining and fine-tuning experiments were conducted on a single NVIDIA H100 GPU. For the specific experiments presented in Section 5.1, each complete training run (pretraining followed by finetuning) required approximately 3 hours. Model evaluations, including the simulation of sensor failures and performance assessment for both baseline models and our approach across all test sets, were performed on an Apple MacBook Pro equipped with an M3 Max processor and 128 GB of unified memory. A full evaluation cycle covering all sensor failure modes took approximately 20 minutes (Figure 4). The total computational effort for the entire project was substantially greater, including extensive hyperparameter tuning and architectural refinements conducted over several months, as individual model training runs never utilized more than a single H100 GPU.

C Different masking combinations

To further understand the contribution of each mask to the overall robustness and downstream performance, we conducted an ablation study where models were pretrained using different combinations of the mean, bias, and noise masking objectives introduced in Section 4.1. Specifically, we trained separate models using each masking type (Mean only, Bias only, Noise only), all pairwise combinations (Mean+Bias, Mean+Noise, Bias+Noise), and the full combination (Mean+Bias+Noise) presented in the main paper. Each pretrained model was subsequently finetuned on the lateral velocity estimation task. We then evaluated these models following the same protocol as in Section 5.1, assessing their robustness against the four sensor failure modes (mean, bias, noise, scaling) across all input channels on both the seen and unseen test tracks, as well as their baseline performance without induced failures. The results of this comparative analysis are presented in Figure 7.

Each data point (color/shape) represents the model's MSE under a specific sensor failure mode, illustrating robustness and performance variability across failures; the black line denotes performance without failures. Models lacking bias masking exhibit increased variability in bias and scaling errors, while those with bias masking demonstrate enhanced robustness. The three-masking approach achieves the second-best downstream performance without errors and shows decreased variability and good robustness for the seen and unseen tracks.

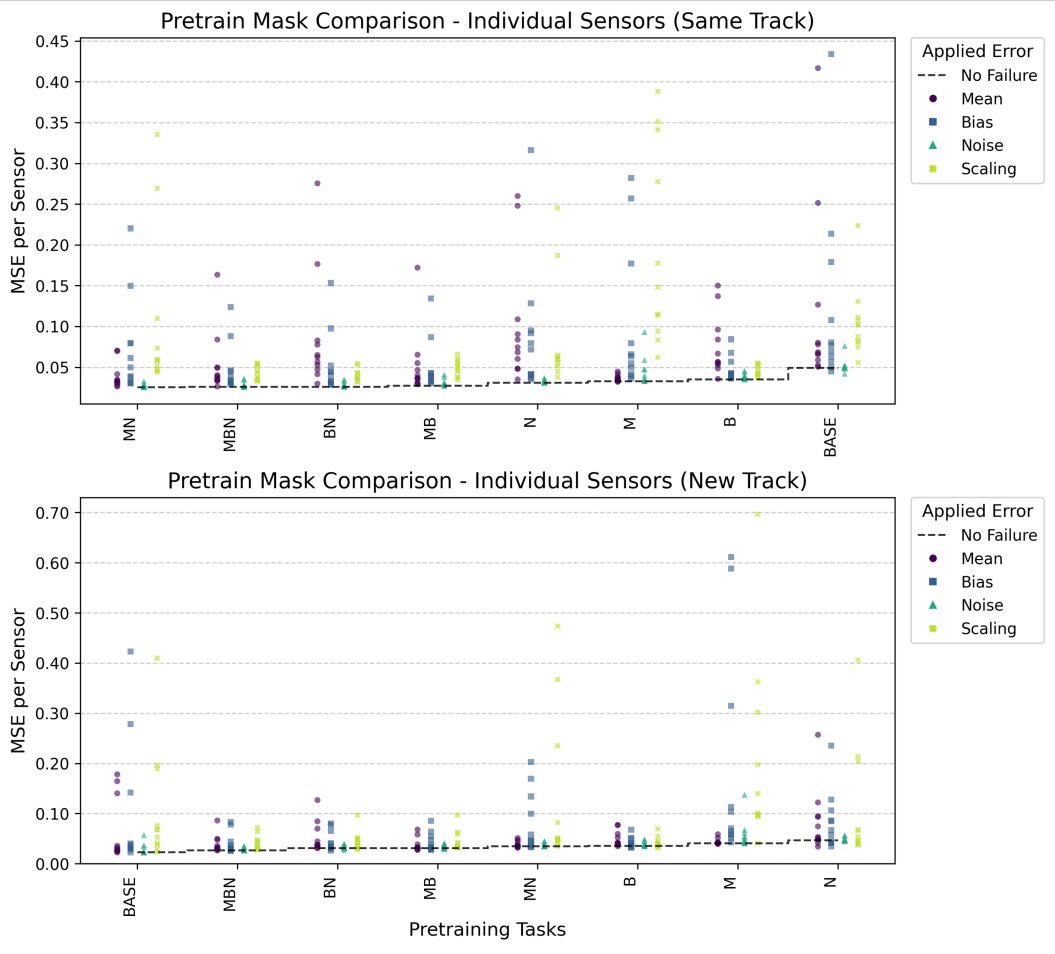

Figure 7: Robustness evaluation for all different masking combinations.

D Robustness to increasing sensor failure strength

This section explores the impact of varying sensor failure strengths on lateral velocity estimation. We tested the models with different severity for bias ($\sigma_{[f]} \in [0.5, 2.5]$), noise ($\sqrt{r} \in [0.2, 1.0]$), and scaling ($\alpha \in [1.5, 3.5]$) failures. The results show a clear distinction: baseline performance degrades notably with stronger failures, while our pretrained model remains robust. The results also highlight significant variation in impact across input features. For the non-pretrained baseline, the lateral velocity estimation shows high sensitivity to failures in some input features, while failures in others have minimal impact, implying the latter may contribute less to the overall estimation.

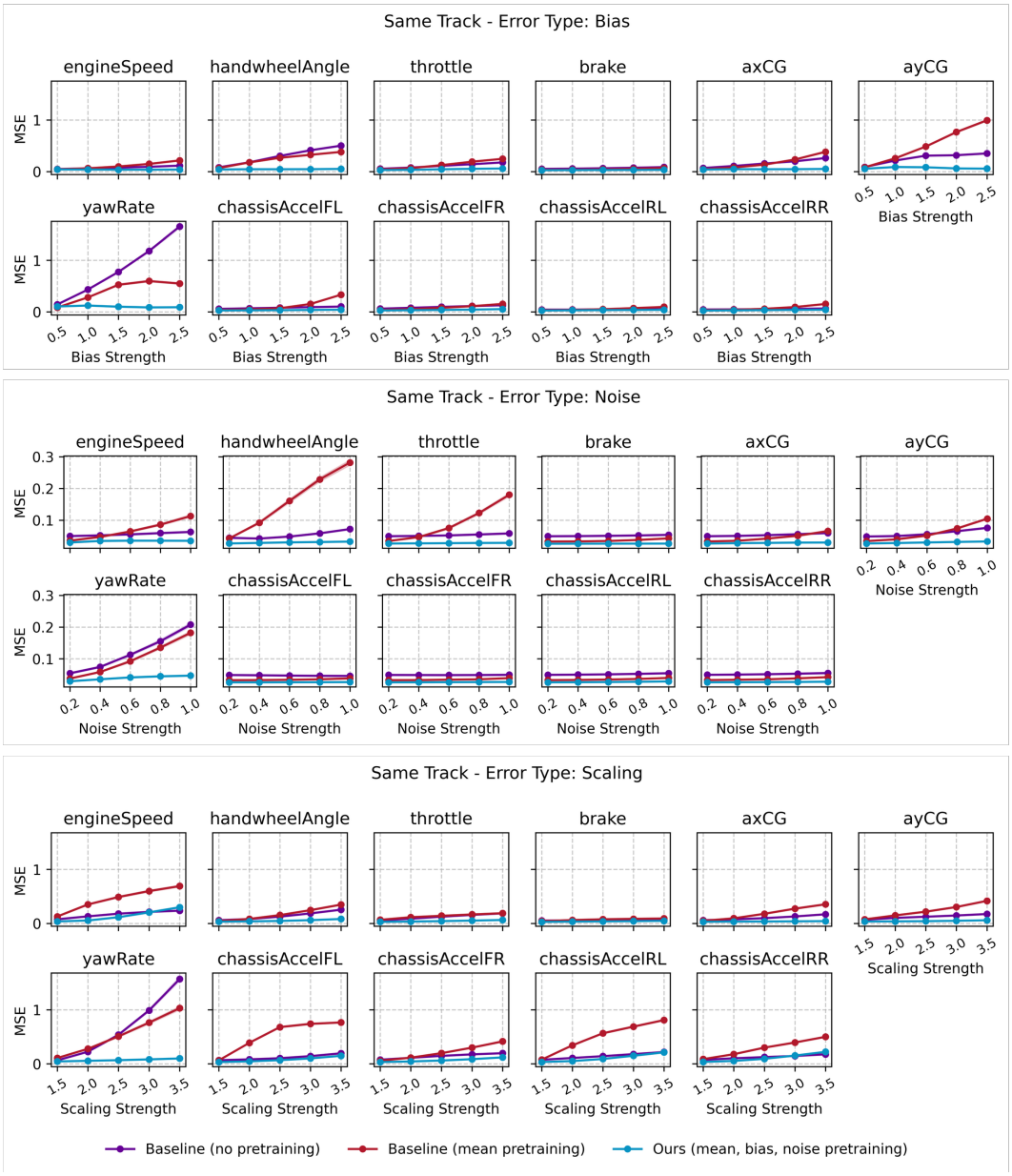

Figure 8: MSE on lateral velocity estimation under sensor failures with varying strengths. This experiment was done on data from a track included in the training data.

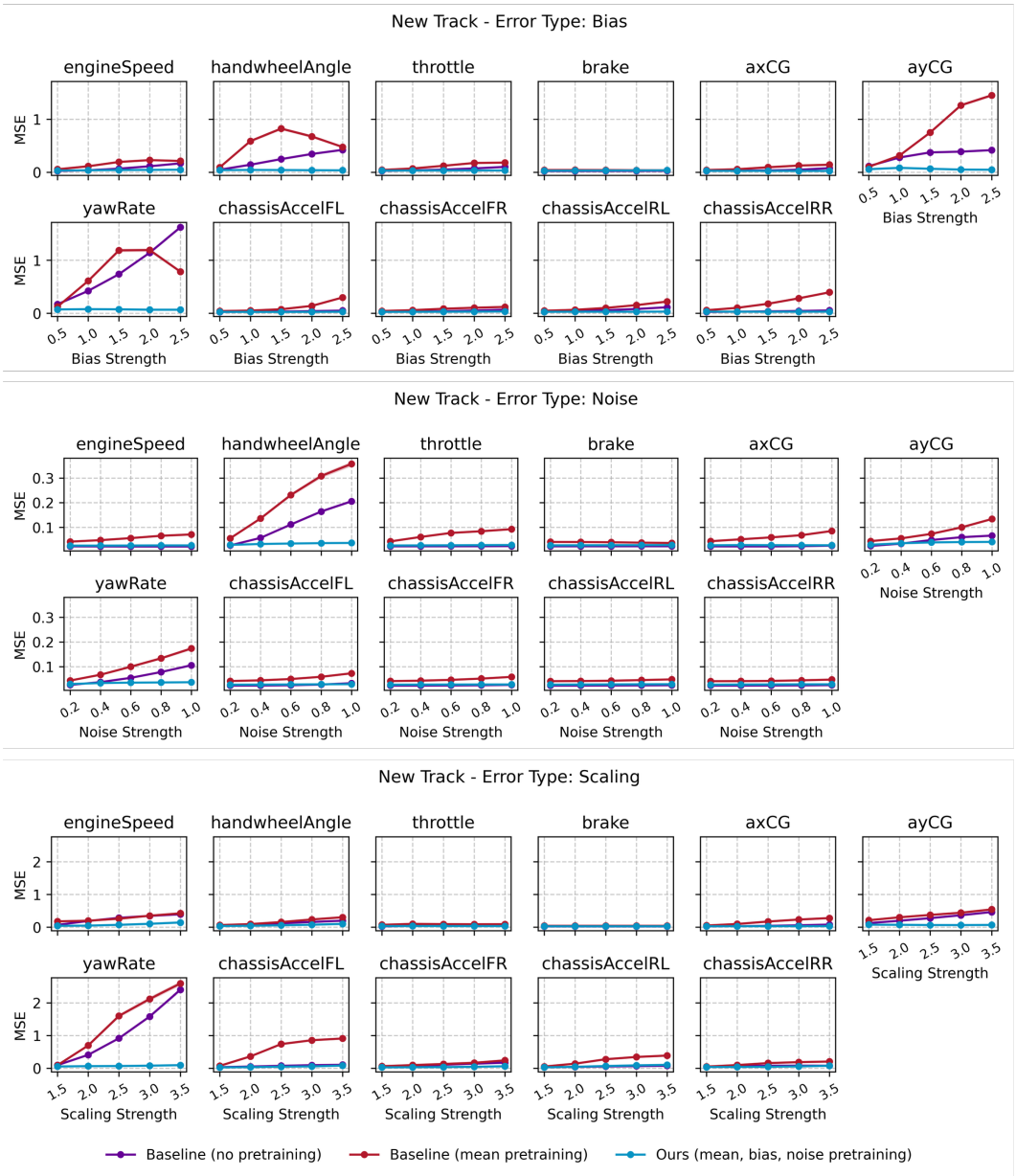

Figure 9: MSE on lateral velocity estimation under sensor failures with varying strengths. This experiment was done on data from a new track not seen during training.

E Pretraining and finetuning over multiple seeds

To assess the influence of random initialization on model performance and robustness, we repeated the pretraining and finetuning process for the non-pretrained baseline model, the model pretrained solely with mean masking, and our proposed model (pretrained with mean, bias, and noise masking) using 10 random seeds. Each resulting model was evaluated following the same robustness procedure described in Section 5.1. The results are shown in Figure 10, where columns distinguish between evaluations on the test set from the same track used during training (left) and an unseen track (right). The rows correspond to the four different sensor failure modes applied during evaluation (mean, bias, noise, scaling). Within each subplot, individual points represent the MSE for a single model trained with a specific seed, color-coded according to the training scheme.

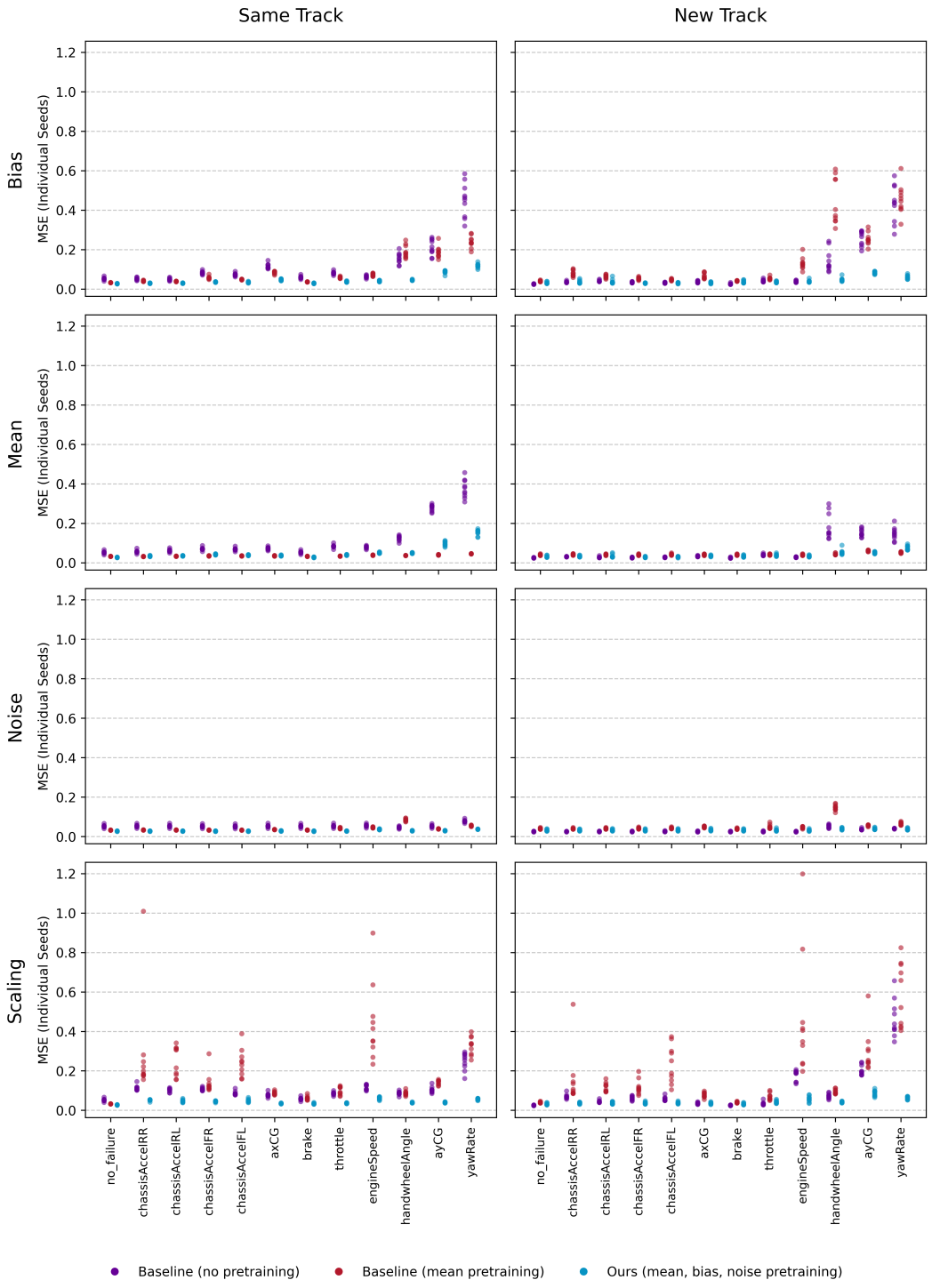

Figure 10: Robustness evaluation across 10 random seeds for the non-pretrained baseline, mean-only pretrained baseline, and our approach. Columns distinguish between test sets (left: same track as training, right: unseen track). Rows represent the four sensor failure modes applied during evaluation (mean, bias, noise, scaling). Each point shows the MSE for a single seed, color-coded by model type.

The results indicate that robustness against noise failures is largely unaffected by the random seed for all three model types. Our proposed approach (Pretrained Mean+Bias+Noise) demonstrates consistently strong robustness across all failure modes and seeds, exhibiting minimal performance

variation. While the model pretrained only on mean masking also shows consistent robustness for mean and noise failures, its performance under bias and scaling failures varies significantly depending on the seed. This suggests that achieving robustness against failure modes not explicitly seen during pretraining (like bias and scaling for the Mean-only model) can be highly sensitive to initialization; a specific seed might yield a model robust to a particular unseen failure, while another might not. In contrast, our multi-masking approach improves generalization and is more robust to the unseen scaling failure across different seeds.

F Effect of model size on robustness

To investigate the influence of model capacity on robustness, we pre-trained models with varying sizes. Specifically, we adjusted the transformer encoder's embedding dimension d_{model} through the sequence [32, 64, 128, 256, 512], setting the feedforward dimension to $4 \times d_{\text{model}}$ accordingly.

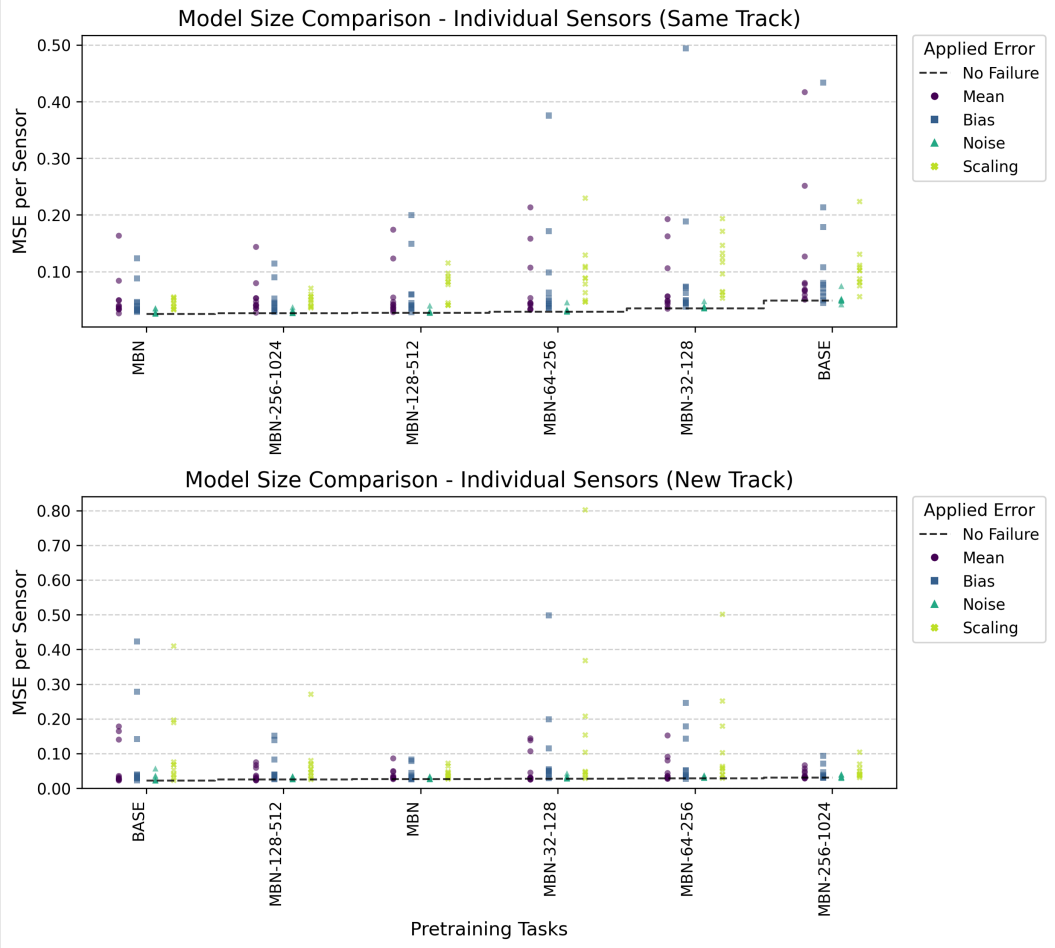

Figure 11: Robustness evaluation for models pretrained with different embedding dimensions (d_{model}). Each point represents the model's MSE under a specific sensor failure mode (color/shape) for a given model size. The upper and lower plots show results on the training and holdout tracks, respectively.

The model (MBN) with $d_{\text{model}} = 512$ corresponds to the architecture used in the main paper. Each pretrained model was finetuned and evaluated using the same robustness protocol described in Section 5.1, assessing performance under mean, bias, noise, and scaling failures across all input channels for both seen and unseen tracks. The results, presented in Figure 11, indicate a general trend where larger models exhibit enhanced robustness and reduced performance variability across

different sensor failure modes and input channels. This suggests that increased model capacity allows for learning more comprehensive representations that better handle diverse input corruptions.

G Effect of pretraining bias bounds

This experiment explores how the range of bias values encountered during pretraining affects robustness to bias and scaling failures of varying strengths during evaluation. We pretrained separate models using the full Mean+Bias+Noise masking strategy, where the bias masking offset (Section 4.1) was sampled uniformly using bounds $l = -k$ and $u = k$, with $k \in \{1, 2, 3, 4\}$. The model with $k = 3$ corresponds to the configuration used in the main paper. These models were then evaluated similarly to the experiments in Section D, specifically focusing on their performance degradation as the severity of applied bias and scaling failures increased. As shown in Figure 12 and Figure 13, models pretrained with narrower bias bounds (e.g., $k = 1$) exhibit a noticeable decline in robustness when the applied bias or scaling errors exceed the magnitude seen during pretraining. This suggests that while pretraining enhances robustness within the experienced distribution of failures, generalization to significantly stronger, out-of-distribution failures remains challenging, highlighting the importance of selecting appropriate pretraining parameters based on expected real-world failure characteristics.

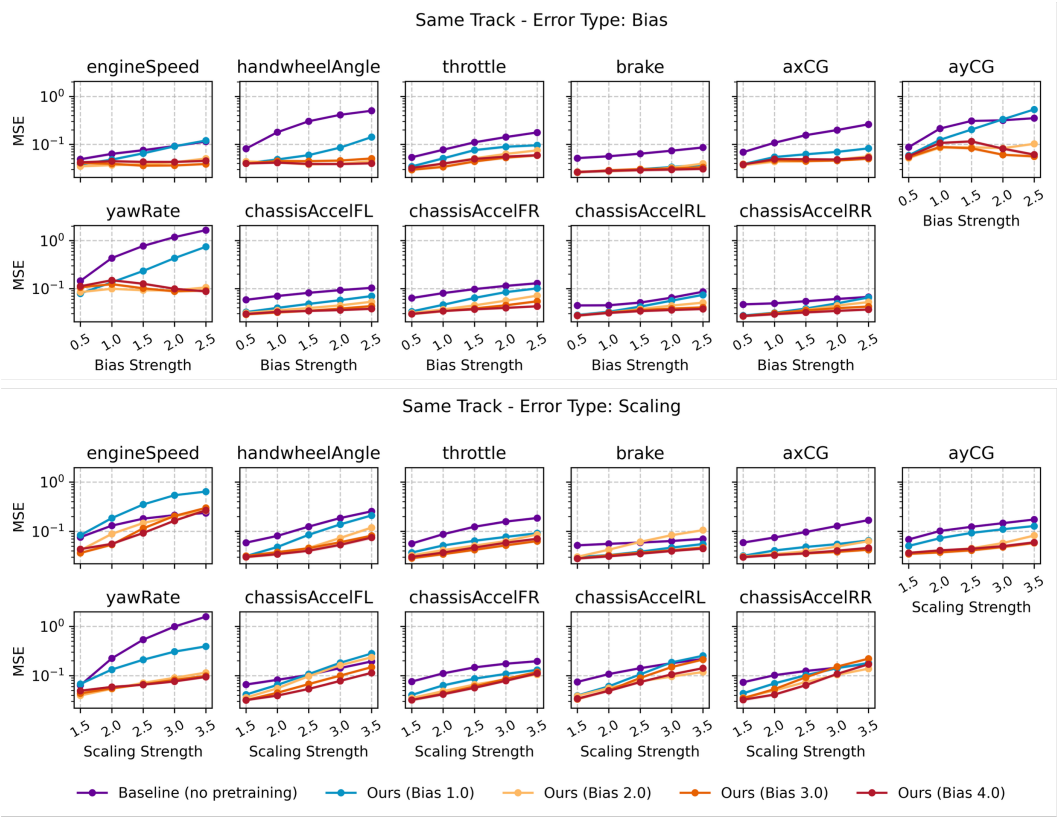

Figure 12: MSE on lateral velocity estimation under bias and scaling failures with varying strengths, evaluated on the same track as training data. Lines represent models pretrained with different bias bounds ($k \in \{1, 2, 3, 4\}$). Log-scaled y-axis.

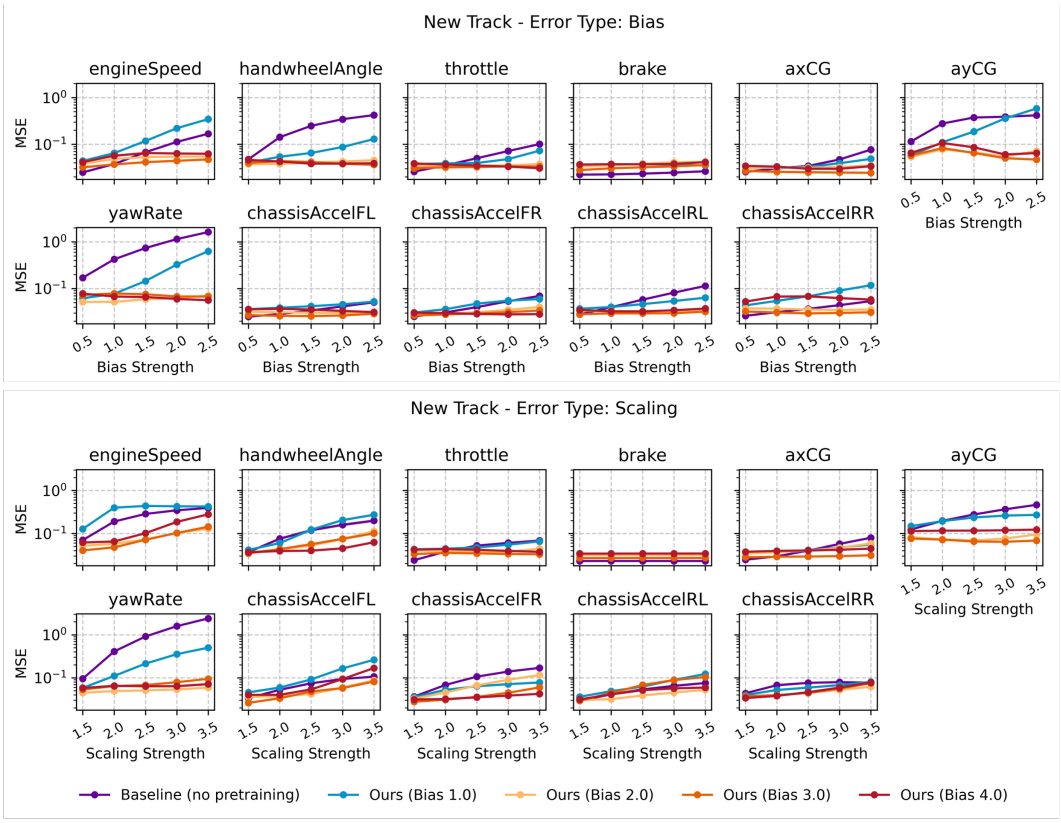

Figure 13: MSE on lateral velocity estimation under bias and scaling failures with varying strengths, evaluated on an unseen track. Lines represent models pretrained with different bias bounds ($k \in \{1, 2, 3, 4\}$). Log-scaled y-axis.

H Effect of pretraining noise ratio

Complementary to the bias bounds experiment, we investigated the impact of the noise level used during pretraining on robustness against noise failures of varying strengths. Models were pretrained using different noise ratios $\sqrt{r} \in \{0.5, 1.0, 1.5, 2.0\}$. The model with $\sqrt{r} = 1.0$ corresponds to our main configuration. We evaluated these models against noise failures with increasing severity (variance scaling $\sqrt{r} \in [0.2, 1.0]$), following the protocol in Section D. The results, depicted in Figure 14, indicate that the magnitude of noise applied during pretraining generally has a limited influence on robustness against noise failures at inference. An exception is the model pretrained with the smallest noise ratio; this model exhibits reduced robustness, particularly when confronted with stronger noise failures during evaluation. This observation aligns with the findings from the bias bounds experiment, where robustness was most pronounced when evaluation conditions closely matched pretraining conditions.

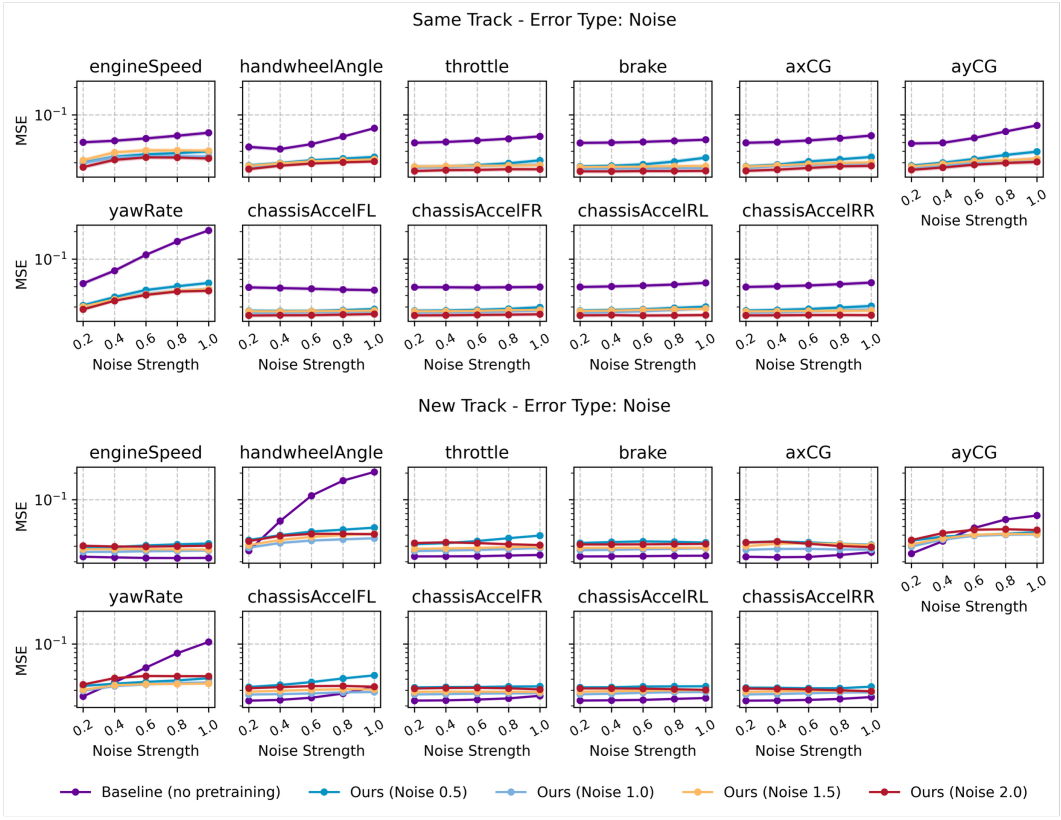

Figure 14: MSE on lateral velocity estimation under noise failures with varying strengths, evaluated on both the same track and an unseen track. Lines represent models pretrained with different noise ratios ($\sqrt{r} \in \{0.5, 1.0, 1.5, 2.0\}$). Log-scaled y-axis.

I Effect of coupled sensor errors

To evaluate robustness under multi-sensor failure scenarios, we systematically vary the number of failing sensors from 1 to 10 out of 11 total signals, where all failing sensors exhibit the same failure mode: bias shifts (multiplicative factor 1.0), additive noise ($\sigma = 0.4$), scaling errors (multiplicative factor 2.0), or complete sensor dropout (mean imputation). For each combination of failure count and error type, we sample 10 random sensor combinations and report average performance. We compare three model configurations: (1) non-pretrained baseline, (2) mean-only pretrained baseline, and (3) our multi-fault pretraining approach (mean, bias, noise). Performance is assessed on both the same track and the new track test data using mean squared error (MSE).

Our approach consistently outperforms the non-pretrained baseline until approximately 5 sensors fail, and surpasses the mean-only pretrained baseline across all tested fault types except complete sensor dropout (mean imputation). As expected, we observe performance degradation for all models as the number of failing sensors increases, with our approach demonstrating improved stability across multiple simultaneous failures. Notably, pretraining exclusively on mean failures can actually worsen robustness to other failure modes, such as scaling, compared to the non-pretrained baseline, highlighting the importance of diverse fault exposure during pretraining. Similar performance patterns are observed in the evaluation on an unseen track.

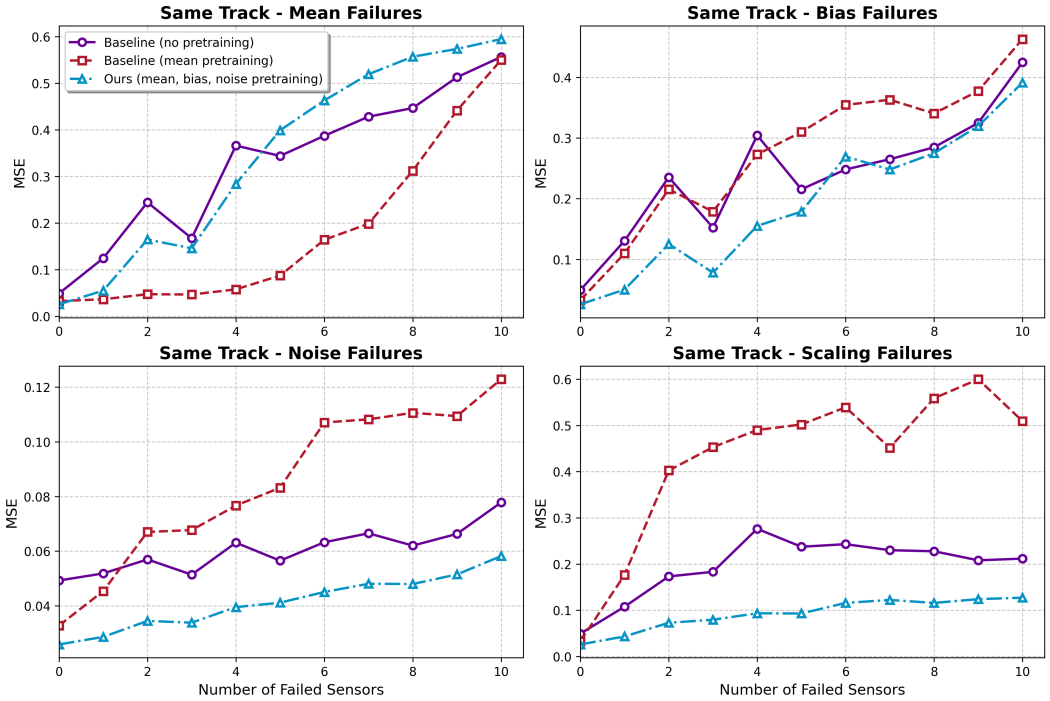

Figure 15: Performance degradation under multiple simultaneous sensor failures on test data on the same track. The x-axis denotes the number of simultaneously failing sensors.

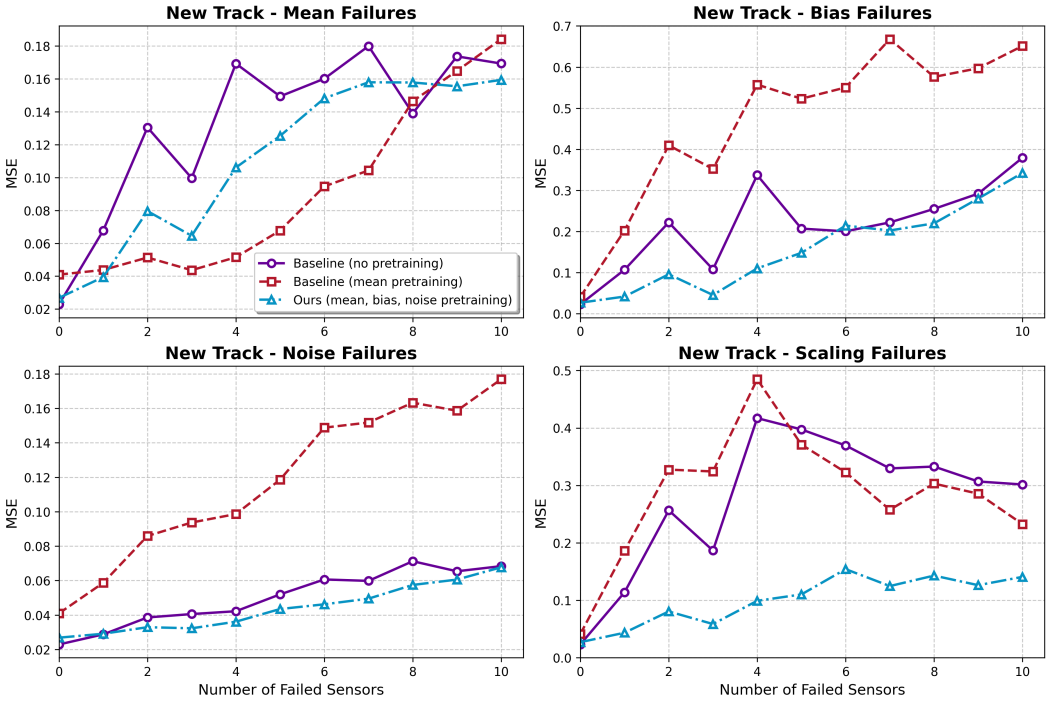

Figure 16: Performance degradation under multiple simultaneous sensor failures on test data on a new track.

J Robustness on further sensor failure modes

To comprehensively evaluate our approach's generalization to the complete taxonomy of sensor failures (see Section 3.1), we conducted additional experiments assessing robustness against drift, outliers, trimming, and varying scaling failures on the lateral velocity estimation task. Following the same experimental protocol and using the same dataset as in Section 5.1, we applied these failure modes individually to each input channel and compared the performance of our pretrained model, the mean-only baseline, and the non-pretrained baseline. The linear and nonlinear drift masks are parameterized such that the maximum drift at the final timestep equals twice the feature standard deviation. Outlier errors are injected with a probability of 0.05 per timestep, with deviations sampled uniformly from $[-3\sigma, 3\sigma]$. Trimming errors restrict values to $[\mu - \sigma, \mu + \sigma]$, with constant trimming clamping at the boundary and dampened trimming using a factor $\alpha = 0.6$. Time-varying scaling errors linearly interpolate the scaling factor from 1.0 to 2.0 over the sequence length. The results demonstrate that our multi-fault pretraining approach maintains improved robustness across these unseen failure modes.

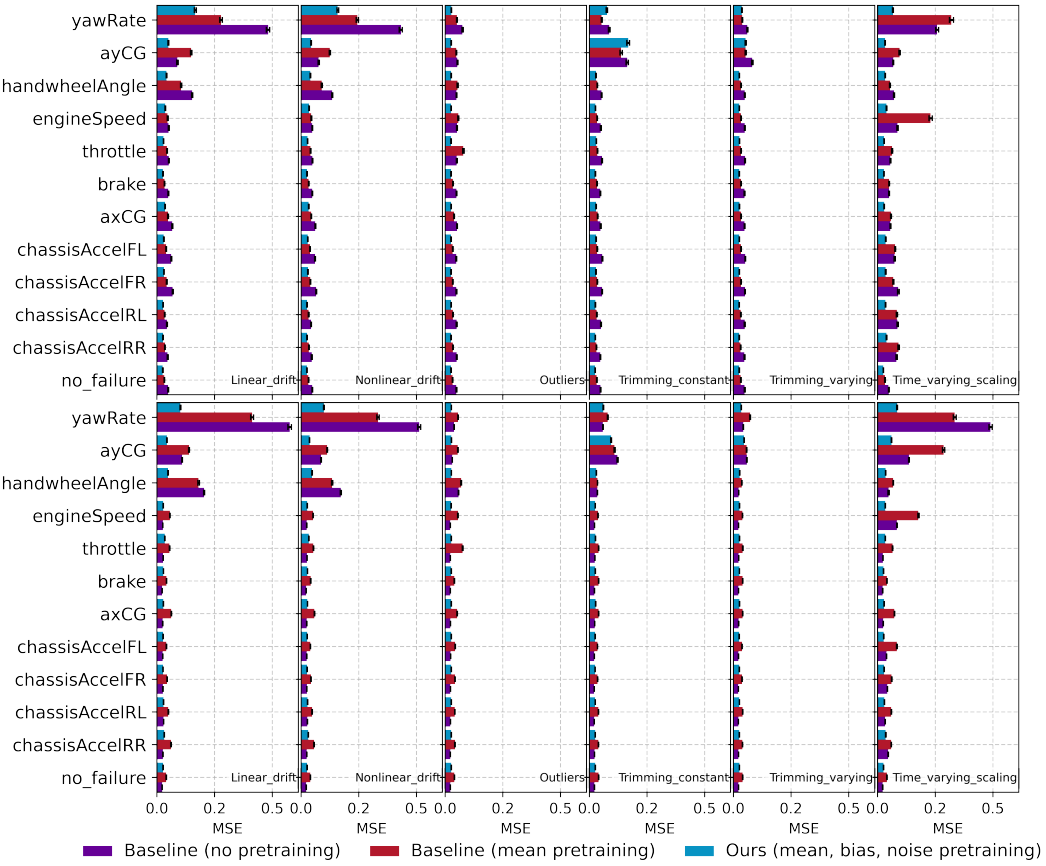

Figure 17: Robustness evaluation of our pretraining approach across additional sensor failure modes on lateral velocity estimation.

K Further Baseline Comparison

Next, we compare our approach against SimMTM [Dong et al., 2023], a recent pretraining method for time series. This comparison demonstrates the specific benefits of robustness-focused pretraining compared to general-purpose time-series representation learning methods.

We adapt SimMTM from its original classification and forecasting setting to our state estimation task, using the recommended hyperparameters from the original paper. Both methods are evaluated on

the REVS dataset following the same experimental protocol described in Section 5.1, applying each failure type individually to all sensor channels and measuring MSE on lateral velocity estimation.

The comparison demonstrates the same performance patterns observed in our baseline comparisons. Our robustness-aware pretraining consistently outperforms SimMTM across all sensor failure scenarios, with particularly pronounced improvements for bias failures. While SimMTM performs very well under nominal conditions (for which it is designed), the performance gap widens substantially when sensors degrade. These results confirm that general-purpose pretraining methods, despite achieving strong nominal performance, fail to learn representations that are robust to common sensor failures.

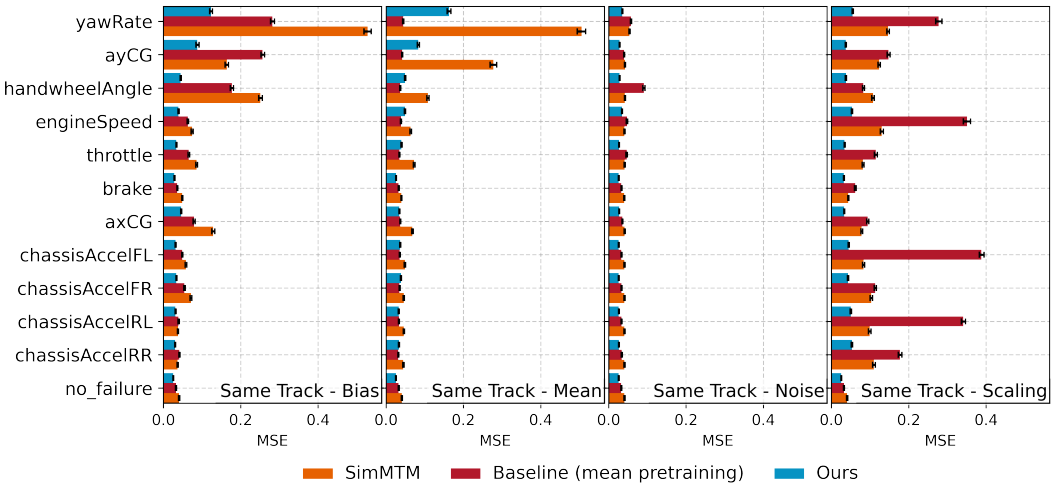

Figure 18: Robustness evaluation of our pretraining approach and SimMTM across different sensor failure types.

L Comparison with Kalman Filtering for VSA estimation

Next, we compare the proposed method with a nonlinear Kalman filter (KF) (see, e.g., Särkkä and Svensson [2023]). The KF estimates a latent state $\{\boldsymbol{x}_k : k = 1, ..., K\}$ given a set of measurements $\{\boldsymbol{y}_k : k = 0, ..., K\}$. It is natural to consider the KF for VSA estimation, as this signal can be modeled in the latent state [Berntorp et al., 2022]. The filter assumes a known estimation model

$$\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0) \quad = \mathrm{N}(\boldsymbol{x}_0; \boldsymbol{m}_0, \boldsymbol{P}_0), \tag{11a}$$
$$\boldsymbol{x}_k \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = \mathrm{N}(\boldsymbol{x}_k; \boldsymbol{f}(\boldsymbol{x}_{k-1}, k-1), \boldsymbol{Q}), \tag{11b}$$
$$\boldsymbol{y}_k \sim p(\boldsymbol{y}_k|\boldsymbol{x}_k) \quad = \mathrm{N}(\boldsymbol{x}_k; \boldsymbol{h}(\boldsymbol{x}_k), \boldsymbol{R}), \tag{11c}$$

where the prior $\{\boldsymbol{m}_0, \boldsymbol{P}_0\}$, the noise covariance $\{\boldsymbol{Q}, \boldsymbol{R}\}$, and the nonlinear functions $\boldsymbol{f}, \boldsymbol{h}$ are known, and N denotes a multivariate Gaussian PDF. Following [Berntorp et al., 2022], we implement a single-track bike model, here defined with the full nonlinear Fiala tire model parametrized as in [Lew et al., 2024]. We define a state space $\boldsymbol{x} = (r, v, \beta, \omega)$ comprising yaw rate r (rad/s), longitudinal velocity v (m/s), side-slip angle β (rad), and average rear wheel speed ω (rad/s). In addition, we define control signals $\boldsymbol{u} = (\delta, \tau)$, with steering angle δ (rad) and engine torque τ (Nm). We describe the time-evolution of the state by a differential equation $\dot{\boldsymbol{x}}(t) = \bar{\boldsymbol{f}}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ as in [Lew et al., 2024], use the shorthand $\boldsymbol{x}_k = \boldsymbol{x}(hk)$, and discretize it by forward Euler at a time-step of h (s), yielding

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + h\bar{\boldsymbol{f}}_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \triangleq \boldsymbol{f}(\boldsymbol{x}_{k-1}, k-1), \tag{12}$$

For the measurement model, we simply include a subset of the measurements used as input to the pretrained transformer model, in particular, the yaw-rate, side-slip angle, and rear-wheel speed, as $\boldsymbol{h}(\boldsymbol{x}_k) = (r_k, v_k, \omega_k)^\top$. We then implement a nonlinear Cubature KF [Arasaratnam and Haykin, 2009] to compute the marginal filtering posterior $p(\boldsymbol{x}_k|\boldsymbol{y}_k)$ and tune the noise levels $\{\boldsymbol{Q}, \boldsymbol{R}\}$ manually as hyperparameters to minimize the root mean square error (RMSE). We refer to [Berntorp et al., 2022] for algorithm details and to the implementation for parameter choices (see Section B). In this setting, the nonlinear KF has access to more information than the pretrained transformer, as

it assumes a parametrized physics-based model along with the control signal trajectory $\boldsymbol{u}(t)$. This makes a direct and truly fair comparison challenging. Nonetheless, we report the MAP estimate when using the same disturbed wheel speeds as realized in the experiment in Sec. 5.2 (see Fig. 6), and compare the side-slip angle reconstruction of the resulting KF to that of the baseline and pre-trained transformer models, with the results reported in Tab. 3 and shown qualitatively in Fig. 19. We note that the nonlinear KF significantly improves over the baseline, despite the failure modes violating the assumptions encoded in the estimation model. However, the model utilizing the proposed pretraining method yields better VSA results, despite not having access to the physics-based vehicle model.

Table 3: Root mean square error (RMSE) and maximum absolute error (MAE) of VSA estimate.

Method	RMSE (rad)	MAE (rad)
Baseline	0.047	0.462
KF	0.018	0.170
Ours (PT)	**0.006**	**0.030**

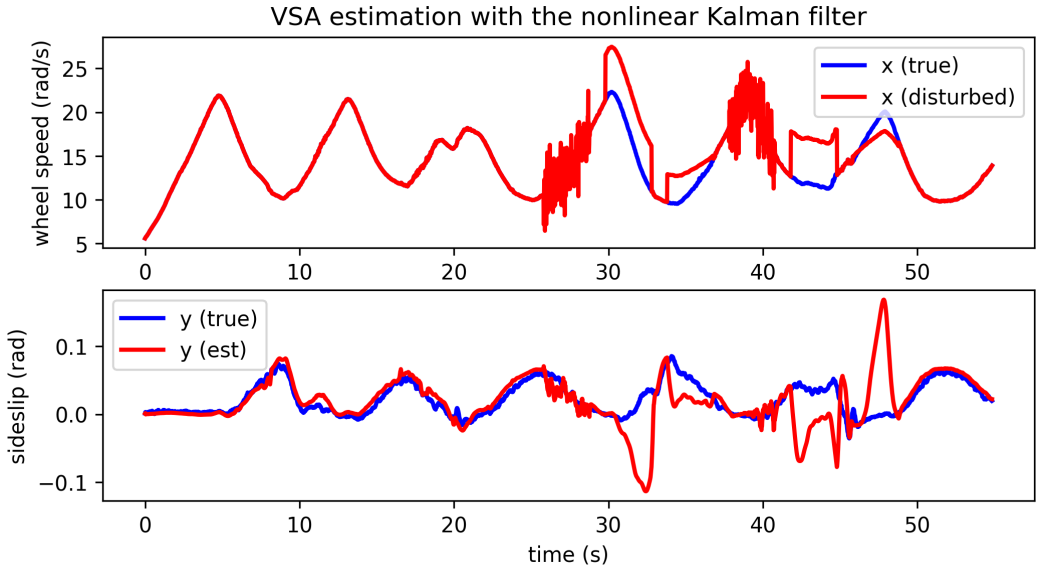

Figure 19: Disturbed average rear wheel speed signals and VSA estimate with the KF (c.f., Fig. 6).

M Robust estimation in environmental sensing

To test the generalization of our approach beyond automotive applications, we evaluate our method on the Beijing PM10 and PM2.5 air quality prediction tasks from the Time Series Extrinsic Regression (TSER) benchmark [Tan et al., 2020, Zhang et al., 2017], where the objective is to predict PM10 and PM2.5 concentrations from nine environmental sensors (SO2, NO2, CO, O3, temperature, pressure, dew point, rainfall, wind speed). We simulate individual sensor failures using the same four error types as in our automotive experiments: bias shifts (two-standard-deviation offset), additive noise ($\sigma = 0.8$), scaling errors (multiplicative factor 2.0), and complete sensor dropout. Each sensor failure is evaluated independently to assess the impact of losing specific environmental measurements on PM10 and PM2.5 prediction accuracy. We compare three model configurations: a baseline without pretraining, a baseline with mean-only pretraining, and our multi-fault pretraining approach (mean, bias, noise). Our approach again improves reliability across failure modes and failing sensors, with particularly pronounced benefits observed for scaling errors in pressure measurements, which cause substantial performance degradation in the baseline models. While the overall performance differences are less pronounced than in the automotive domain, the results demonstrate the cross-domain applicability of our pretraining strategy.

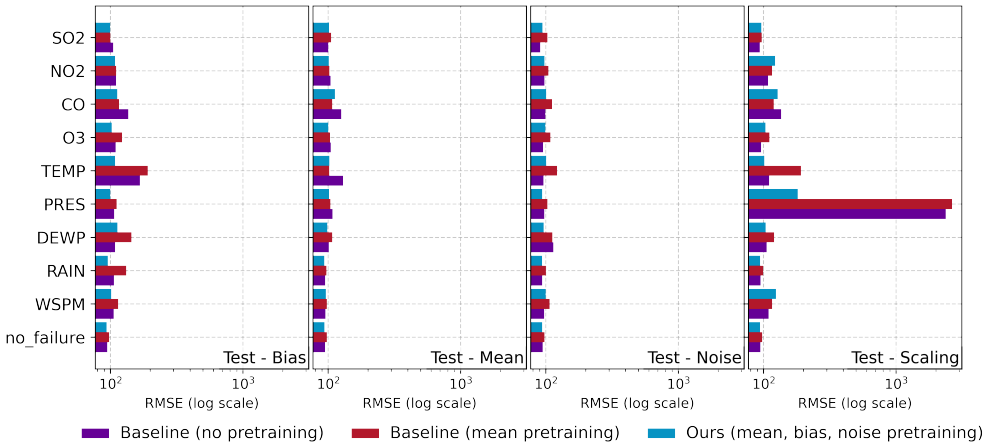

Figure 20: Robustness on Beijing PM10 air quality prediction with single sensor failures.

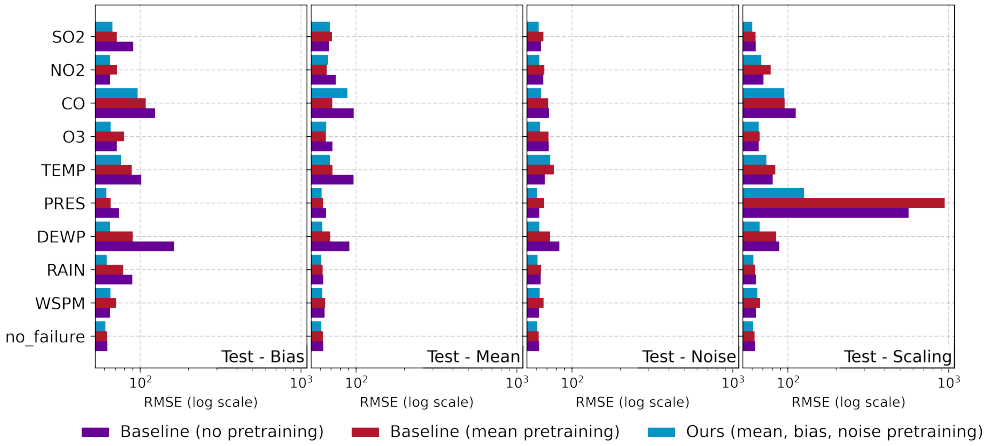

Figure 21: Robustness on Beijing PM2.5 air quality prediction with single sensor failures.

N Closed-loop experiments

The closed-loop experiments conducted on the Lexus LC 500 were implemented utilizing ROS2 Humble. The experimental setup was adapted to mirror the conditions under which the models were originally pretrained and subsequently finetuned. To ensure congruity with the initial training regimen, which was based on the REVS dataset, several modifications were made concerning feature extraction methodologies, input signal frequencies, and overall system integration.

Real-time sensor data from the vehicle served as the principal interface for the estimator. While the REVS dataset was sampled down to a frequency of 20 Hz, the on-vehicle system operated at 62.5 Hz. To preserve temporal alignment with the REVS dataset, the input data stream was down-sampled by selecting every third sample, yielding an effective input frequency of approximately 20.8 Hz. Consequently, the estimator maintained a sequence length of 20, thereby encompassing approximately one second of driving history, a configuration consistent with the previous experiments.

The feature set was tailored to the available onboard signals. The extracted features incorporated yaw rate, longitudinal and lateral accelerations, vertical chassis accelerations at all four corners, individual wheel speeds, steering angle, wheel torques, brake torques, and longitudinal velocity. Table 4 provides an overview of the channels employed for the LC 500 experiments.

We trained a new model using historical data acquired from previous tests with the Lexus LC 500, and the architecture underwent further tuning. In contrast to the REVS dataset approach, the non-pretrained baseline (hereafter referred to as baseline) and pretrained architectures for the Lexus

Table 4: Input channels and target for the LC 500 deployment

Feature	Unit
Sideslip Angle (target)	rad
Yaw Rate	rad s^{-1}
Longitudinal Acceleration	m s^{-2}
Lateral Acceleration	m s^{-2}
Vertical Acceleration	m s^{-2}
Wheel Speed FL	m s^{-1}
Wheel Speed FR	m s^{-1}
Wheel Speed RL	m s^{-1}
Wheel Speed RR	m s^{-1}
Steer Angle FL	rad
Wheel Torque RL	N m
Brake Torque FL	N m
Brake Torque FR	N m
Brake Torque RL	N m
Brake Torque RR	N m
Longitudinal Velocity	m s^{-1}

platform were tuned independently to ensure optimal performance for the closed-loop tests under nominal conditions. The primary distinctions between the Lexus_Pretrained architecture (Lexus PreT), Lexus_Baseline architecture (Lexus Base), and the REVS architecture are delineated below:

Table 5: Comparison of encoder architecture parameters

Parameter	Lexus Base	Lexus PreT	REVS
Feature Dimensionality	15	15	11
Embedding Dimension	128	256	512
Encoder Layers	1	2	4
Attention Heads	4	16	16
Feedforward Network Dimension	128	512	2048
Backbone Output Dimension	300	300	220

Table 6: Comparison of regression head parameters

Parameter	Lexus Base	Lexus PreT	REVS
Input Size	2560	5120	10240
First Estimation Head Dimension	128	256	256
Estimation Head Dropout Rates	(0.4, 0.34)	(0.32, 0.14)	(0.32, 0.14)

The variations between the Lexus model and the REVS model are primarily attributable to the requirement for real-time execution on an Intel Xeon E2278GE CPU @ 3.30 GHz. The difference between the baseline and pretrained models stems from the comparatively simpler nature of the classical end-to-end learning task for the baseline.

N.1 Setup

The architecture of the experimental setup is most effectively elucidated by the diagram presented in Figure 22. The vehicle publishes its state, comprising the features detailed in Table 4. A dedicated "disruptor node" subscribes to both the vehicle state and a predefined disruption schedule. This schedule specifies the timestamp, the sensor to be targeted, the type of error to be injected, and the magnitude of the error (applicable for noise and bias injections). The disruptor node then publishes a "disrupted vehicle state", a replica of the original vehicle state, except for the designated sensors, which are perturbed according to the disruption schedule. The sideslip estimator (which, depending

on the specific experiment, can be the baseline model, the pretrained model, or both) subscribes to this "disrupted vehicle state". It subsequently publishes a "Sideslip Angle Estimation". This estimation, in conjunction with the "disrupted vehicle state", constitutes the "modified vehicle state". This modified vehicle state serves as the control input for the controller, which, in turn, generates the control output directed to the vehicle, thereby closing the loop.

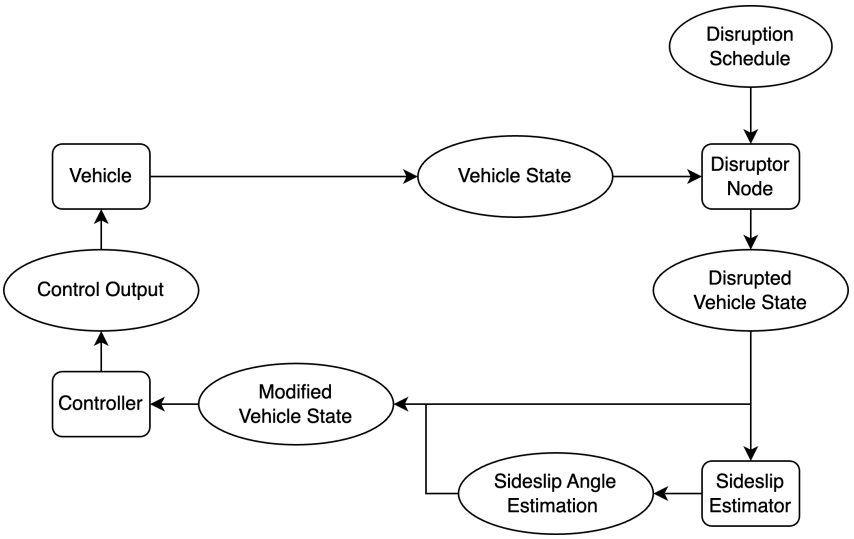

Figure 22: Experimental setup illustrating the closed-loop data flow. The vehicle state is subjected to scheduled disruptions, processed by the sideslip estimator, and the resulting estimation is integrated into the modified vehicle state used by the controller.

To assess the estimator's robustness under authentic closed-loop conditions, controlled fault injection experiments were performed during autonomous driving operations with the LC 500. These experiments were meticulously designed to replicate the perturbation regimes employed with the REVS dataset, while simultaneously demonstrating real-world applicability by directly incorporating the estimated sideslip angle into the vehicle's controller.

Fault schedules were structured sequentially, perturbing a single sensor at a time for five seconds. This perturbation phase was followed by a recovery interval, during which no faults were active. This experimental design facilitated the unambiguous attribution of observed vehicle behavior to specific sensor faults and ensured adequate recovery time between successive fault injections. The injected sensor faults encompassed several distinct perturbation types. Bias errors were introduced by adding one or two standard deviations of the respective sensor signal. Hard faults were simulated by substituting sensor readings with their corresponding channel means. Additionally, Gaussian noise perturbations were applied, characterized by standard deviations of 0.4 and 1.0 times the feature-specific standard deviation.

We additionally tested our method on a race track to evaluate performance across different driving contexts: the skidpad setup being more reflective of regular driving conditions, while the race track introduces racing-like scenarios with increased complexity and variability. The initial results showed comparable robustness across both environments. Ultimately, we decided to focus on the skidpad experiments for the paper due to their superior reproducibility through structured, repeatable maneuvers and clearer communicability of results.

O Sensor failure experiments

For the comparative analysis of the baseline and pretrained models, two distinct experimental scenarios were investigated: Parallel Loop and Separate Loop.

In the Separate Loop scenario, both models functioned as the sideslip estimator in discrete test sessions. This approach permits a robust comparison of the models and an examination of how their respective estimation errors propagate. However, it does not guarantee identical experimental

conditions across tests, primarily due to the controller's reaction to estimation inaccuracies. For instance, if an estimation error causes the vehicle to decelerate, a time-scheduled sensor disruption will occur during a different driving maneuver than if the vehicle had maintained its original speed. This methodology was employed for the Wheel Speeds sensor experiments, the results of which are documented in the main paper (Section 5.2).

In the Parallel Loop scenario, the pretrained model served as the sideslip estimator within the closed-loop system, as previously described. Concurrently, the baseline model operated in an open-loop configuration, processing the same input data but without its output influencing the vehicle controller. The principal advantage of this test scenario lies in the direct comparability of both models using identical input data. Conversely, its limitation is that only the closed-loop model's estimation error can propagate through the system, thereby revealing the real-world consequences for only one of the two models. By employing both testing scenarios, a comprehensive understanding of model performance under sensor failure conditions is achieved. This experimental paradigm was implemented for the wheel speed sensors, the yaw rate sensor, the steering wheel sensor, and the longitudinal velocity sensor.

O.1 Parallel loop skidpad: wheel speed sensor perturbations

The experiments involving the disruption of wheel speed sensors were conducted using a structured perturbation schedule. This schedule consisted of the following sequence for each targeted sensor:

1. 5 seconds of Gaussian noise, with a standard deviation (σ) equal to the standard deviation of the original sensor signal.
2. 1 second recovery interval (no perturbation).
3. 5 seconds of a bias error, where a value equivalent to 2σ of the original signal was added to the sensor readings.
4. 1 second recovery interval.
5. 5 seconds where the sensor signal was replaced by its mean value.
6. 1 second recovery interval.

This perturbation sequence was applied sequentially to each wheel speed sensor: front-left (FL), front-right (FR), rear-left (RL), and rear-right (RR). The 1-second recovery intervals ensured that each 1-second input sequence to the model contained at most one type of disruption, enabling clear analysis of each perturbation's isolated effect. The resulting disruption patterns are shown in Figure 23.

Our pretrained model demonstrated no significant degradation in performance in response to any of the applied disruptions on any wheel. This outcome suggests the successful impartation of robustness through the pretraining process. Conversely, the baseline model experienced dramatic decreases in performance, with the most severe instance observed during the bias error injection on the front-right wheel speed sensor. This event led to an estimation error exceeding 0.30 rad.

It is also noteworthy that a bias error consistently induced a constant-like error pattern in the baseline model's estimation. In contrast, a noise error (most observable with the rear-right wheel speed sensor perturbation) resulted in a noisy error pattern in the estimation. The mean value substitution, however, produced varied error patterns: sometimes a spikey error pattern (e.g., front-left wheel speed), sometimes no discernible real error pattern (e.g., front-right wheel speed), or a pattern similar to that of the bias error (e.g., rear-right wheel speed). These observations confirm that, despite the nearly identical information content across all wheel speed sensors, the baseline model remains susceptible to errors when any single one of them is compromised.

Additionally, a separate experiment was conducted in which a constant bias of 2σ (relative to the standard deviation of the original signal) was applied to the rear-right (RR) wheel speed sensor shown in Figure 24. This sustained perturbation led to only minor estimation errors in our pretrained model (occurring around the 35-second mark in the experiment). In contrast, the baseline model exhibited severe errors. Interestingly, the error magnitude of the baseline model decreased as the absolute value of the wheel speed increased (indicative of the vehicle traveling on a straight path, where the side-slip angle is typically small or negligible). Conversely, as the wheel speed decreased, the estimation error of the baseline model increased.

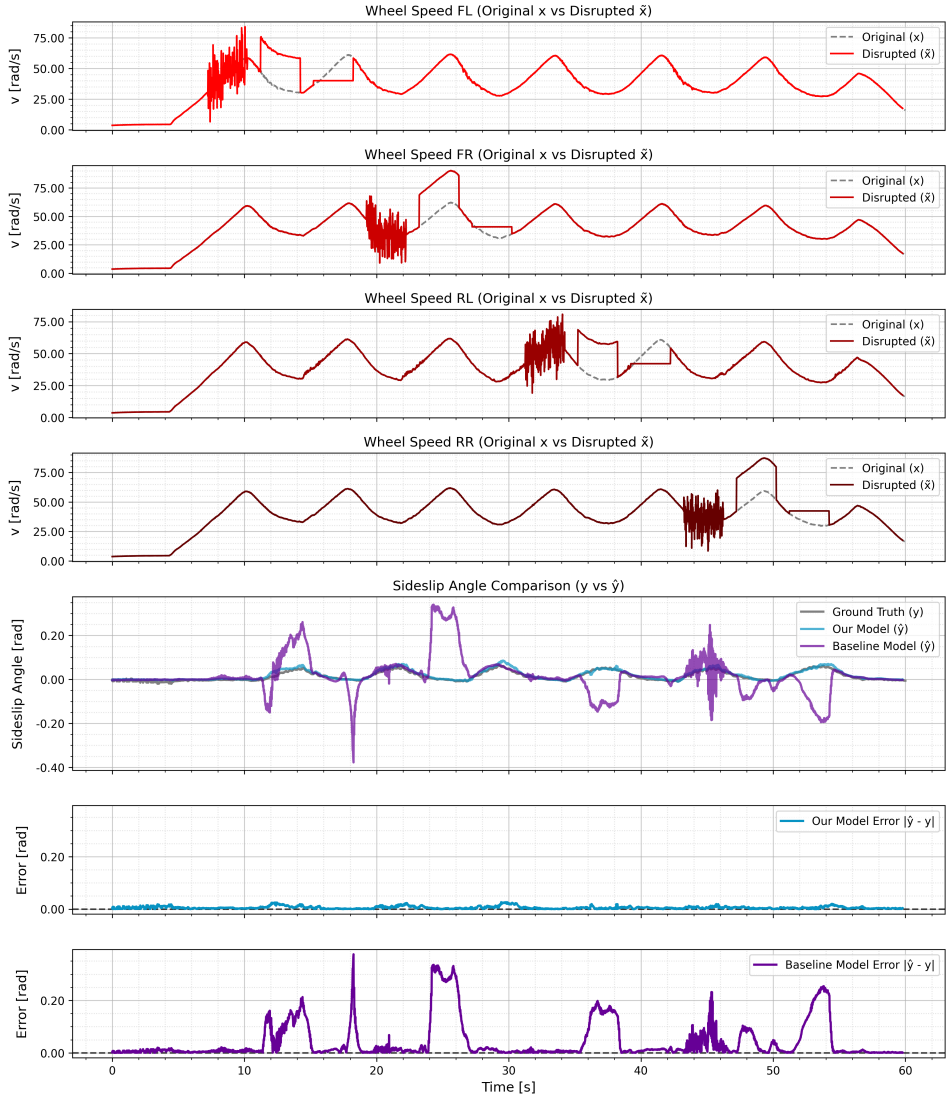

Figure 23: Disruption patterns applied sequentially to individual wheel speed sensors (FL, FR, RL, RR) and the corresponding sideslip angle estimation performance of our model versus the baseline model. The top four plots show the original versus disrupted wheel speed signals. The subsequent plots show the sideslip angle ground truth, our model's estimation, the baseline model's estimation, and their respective errors.

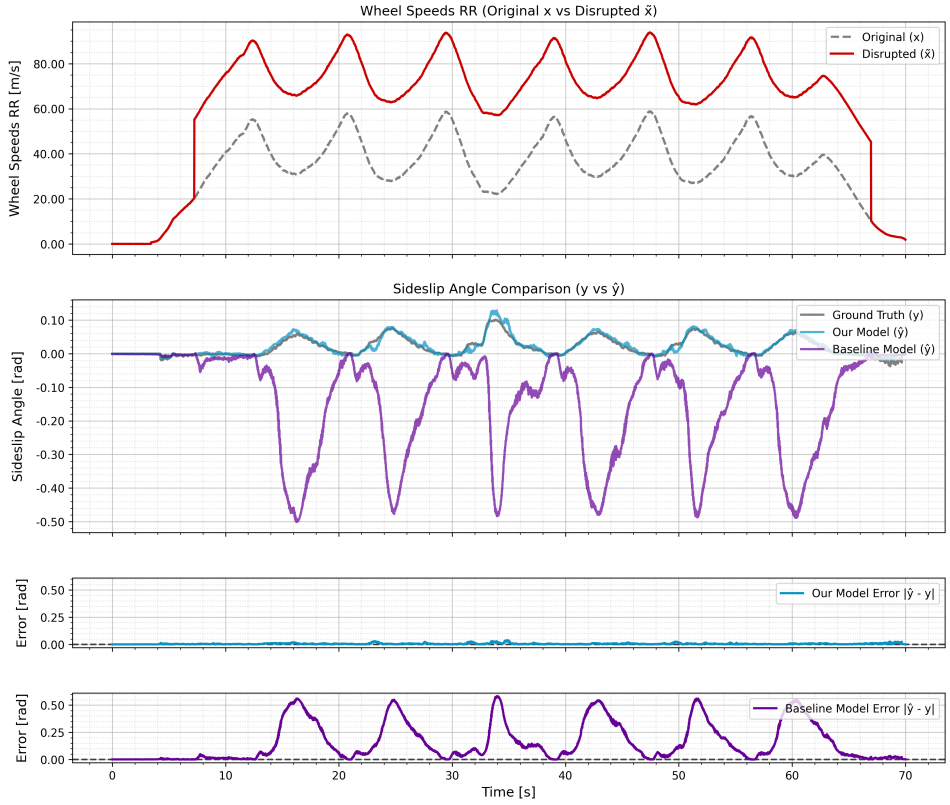

Figure 24: Disruption patterns applied constantly to the rear right wheel speed sensor (RR) and the corresponding sideslip angle estimation performance of our model and the baseline model. The plot shows the original versus the disrupted wheel speed signal. The subsequent plots show the sideslip angle ground truth, our model's estimation, the baseline model's estimation, and their respective errors.

O.2 Parallel loop skidpad: yaw rate sensor perturbations

The robustness of the sideslip angle estimation was further evaluated by introducing simulated perturbations to the yaw rate sensor signal. The perturbation schedule was structured in distinct phases:

1. **Initial phase (0-18 seconds):**
   - 6 seconds of additive Gaussian noise with a standard deviation equal to 0.4σ of the original signal's standard deviation.
   - 1 second recovery interval.
   - 6 seconds where the sensor signal was replaced by its mean value over the recording duration.
   - 1 second recovery interval.
   - 6 seconds of a constant bias error, with an added value equivalent to 1σ of the original signal's standard deviation.
   - 1 second recovery interval.
2. **Incremental noise phase (18 seconds onwards for a defined duration):**
   - The intensity of the additive Gaussian noise was progressively increased second-wise, starting from 0σ and ramping up to a level exceeding 1σ of the original signal's standard deviation.
   - 1 second recovery interval.
3. **Incremental bias phase (Following incremental noise phase for a defined duration):**
   - The magnitude of the constant bias error was incrementally increased, starting from 0.5σ and escalating to 2σ of the original signal's standard deviation.

The impact of these perturbations on the yaw rate signal and the corresponding sideslip angle estimations from both our pretrained model and the baseline model are depicted in Figure 25.

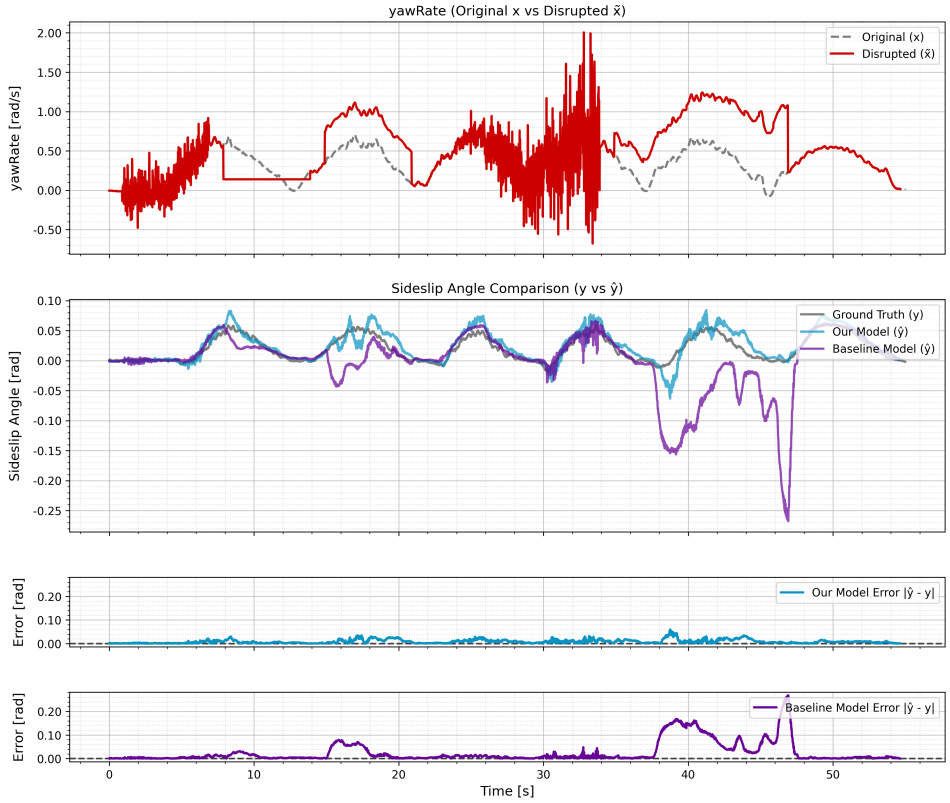

Figure 25: Perturbation effects on yaw rate and sideslip angle estimation during the Parallel Loop Skidpad maneuver. The top panel shows the original versus the disrupted yaw rate signal. The middle panel presents the ground truth sideslip angle alongside the estimations from our model and the baseline model. The two bottom panels display the estimation error for both models.

The yaw rate is a critical input for sideslip estimation, directly reflecting the vehicle's rotational dynamics. As illustrated in Figure 25, our pretrained model consistently maintained a low estimation error across all phases of the perturbation schedule. During the initial phase, the introduction of low-magnitude noise (0.4σ) and mean substitution resulted in negligible deviations in our model's output from the ground truth. Even during the subsequent incremental noise phase, where the perturbation intensity became significantly more pronounced, our model demonstrated remarkable resilience, with estimation errors remaining minimal. Notable deviations in the estimation for our model occurred during the bias perturbations, both the initial 1σ bias and subsequently during the incremental bias phase.

In contrast, the baseline model exhibited significant sensitivity to the yaw rate perturbations. The 1σ bias applied around 15-21 seconds induced a noticeable and sustained error in its sideslip angle estimation. As the noise intensity was incrementally increased (e.g., observe behavior post 25 seconds), the baseline model's error correspondingly escalated, displaying significant noise in its output. The incremental bias phase (e.g., observe behavior towards the latter part of the experiment) led to a diverging estimation error for the baseline model. For instance, at a bias of 2σ, the baseline model's estimation error reached approximately 0.25 rad. This underscores the baseline model's apparent over-reliance on the yaw rate signal and its limited capacity to mitigate the effects of significant sensor corruption, highlighting a stark contrast to the robustness achieved by our pretrained approach.

O.3 Parallel loop skidpad: steering angle sensor perturbations

To assess the impact of corrupted steering input on sideslip angle estimation, a series of simulated perturbations was applied to the steering angle sensor signal. The experiment employed the same multi-stage perturbation schedule as applied for the yaw rate sensor evaluation. Figure 26 illustrates the original versus perturbed steering angle signal alongside the sideslip angle estimations from our pretrained model and the baseline model, as well as their respective estimation errors.

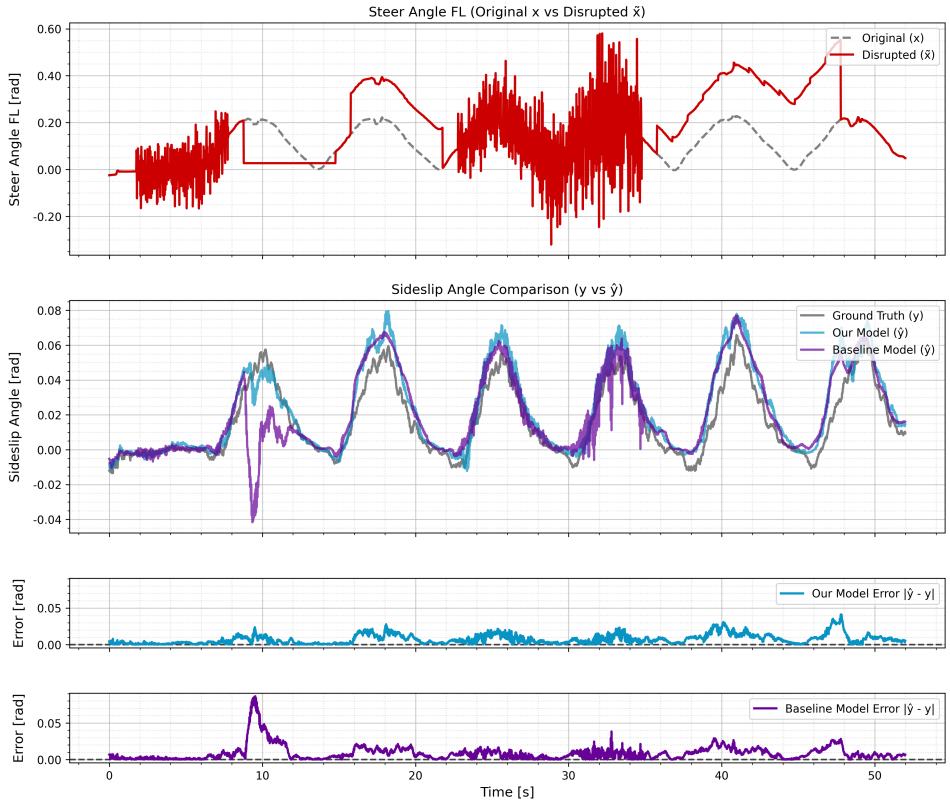

Figure 26: Perturbation effects on steering angle and sideslip angle estimation during the Parallel Loop Skidpad maneuver. The top panel shows the original versus the disrupted steering angle signal. The middle panel presents the ground truth sideslip angle alongside the estimations from our model and the baseline model. The two bottom panels display the estimation error for both models.

The results presented in Figure 26 highlight the differential response of the two models to steering sensor corruption. Our pretrained model demonstrated robust performance throughout the experiment. The initial perturbations (low noise, mean substitution, and 1σ bias) had a minimal effect on its estimation accuracy. More critically, during the incremental noise and incremental bias phases, our model's estimations remained closely aligned with the ground truth, with only minor fluctuations in error.

The baseline model's performance under steering angle perturbations exhibited behavior largely similar to our model during the initial low-noise and bias phases. However, a critical distinction emerged with the introduction of the mean value substitute (around 9-15 seconds), which resulted in a clear and sustained deviation in the baseline model's sideslip estimation. This deviation highlights a specific sensitivity of the baseline model to this type of sensor corruption.

O.4 Parallel loop skidpad: longitudinal velocity sensor perturbations

The final single-sensor perturbation experiment focused on evaluating the impact of corruption in the longitudinal velocity signal. The same structured perturbation schedule utilized for the yaw rate and steering angle sensor experiments was applied. Figure 27 displays the original versus perturbed

longitudinal velocity signal, the ground truth sideslip angle, and the estimations and errors from both our pretrained model and the baseline model.

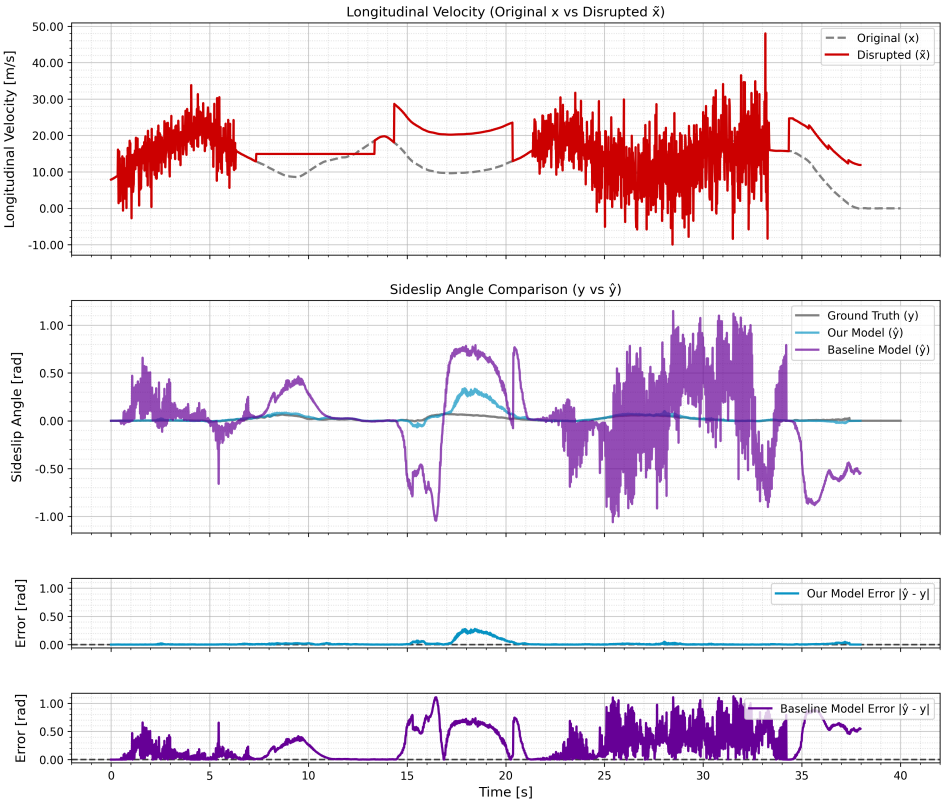

Figure 27: Perturbation effects on longitudinal velocity and sideslip angle estimation during the Parallel Loop Skidpad maneuver. The top panel shows the original versus the disrupted longitudinal velocity signal. The middle panel presents the ground truth sideslip angle alongside the estimations from our model and the baseline model. The two bottom panels display the estimation error for both models.

Under these perturbation conditions, both models experienced significant degradation in performance. The baseline model exhibited near-complete failure in estimating the sideslip angle from the beginning of the first phase, rapidly reaching errors exceeding 0.5 rad. Our model estimate degraded particularly during the 1σ bias perturbation, which led to errors close to 0.3 rad. The safety driver had to intervene to maintain control of the vehicle, necessitating the premature termination of the experiment as soon as the incremental bias phase was initiated, preventing the completion of the full perturbation schedule. Nevertheless, despite the challenging conditions, our model consistently outperformed the baseline model across all applied perturbations before the experiment was stopped. More about this in Section 6.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Section 4.1 for proposed pretraining scheme. See Section 5.1 and Section 5.2 for results on robustness and closed-loop control.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 6 for a discussion of limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results. Results are based on empirical evaluations on public datasets and real-world experiments.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The pretraining scheme is described in detail in Section 4.1. The experimental setup is described in Section 5.1, Section B, Section 5.2 and Section N. The data split and preparation is detailed in Section A. The code and data for the evaluation in Section 5.1 will be made available. While we can't publish the data collected during the closed-loop control experiments, we provide a detailed description of the experimental setup and results in Section 5.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code for the evaluation in Section 5.1 and the appendix will be made available. While we can't publish the data collected during the closed-loop control experiments, we provide a detailed description of the experimental setup and results in Section 5.2 and Section N.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See Section 5.1, Section B, Section 5.2, Section N and Section A

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The experiment in Section 5.1 reports the bootstrapped estimates (200 resamples) showing mse means and their 95% confidence intervals on the test set. Furthermore, in Section D we show the results for different severities of the sensor failures. In Section E we show the evaluation done in Section 5.1 for 10 models trained with different random seeds.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section B and Section N.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research focuses on improving the safety and robustness of deep learning models in critical applications, directly aligning with ethical considerations for AI deployment. We uphold principles of transparency and reproducibility by providing detailed methodology and releasing code, adhering to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The primary positive societal impact—enhancing the safety and reliability of ML systems reliant on potentially faulty sensors—is discussed throughout. Potential negative impacts are implicitly addressed in Section 6, which discusses scenarios where the method might fail (e.g. unevaluated failure modes/domains), potentially leading to unsafe outcomes if deployed without considering these limitations.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This work does not involve the release of a trained model. The research focuses on a methodology for improving robustness using specific vehicle dynamics time series data, which itself does not pose a high risk for misuse. The primary dataset used is publicly available.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The primary dataset used is publicly available and properly credited. Our codebase utilizes parts of the publicly available code from Zerveas et al. [2021], which is also credited. Standard open-source libraries used are governed by their respective permissive licenses.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code implementing the proposed pretraining methodology will be released. Documentation (e.g., README, comments) will be provided alongside the code to facilitate understanding and usage.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This research does not involve crowdsourcing or experiments with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This research does not involve crowdsourcing or experiments with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard component.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.