EPIC-PIML: EPISTEMIC-CONSTRUCTED PHYSICS-INFORMED MACHINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning the dynamics of complex robotic systems via a hybrid physics-informed machine learning (HPIML) approach, which combines known physics with machine learning (ML), has shown promising performance in decision-making and optimal control tasks. However, under environmental uncertainty, HPIML is not necessarily a robust predictive model: the in-distribution region favors the ML term, while the out-of-distribution region favors the physics term. Therefore, adaptively weighing the outputs of these two terms across different environments or regions is important for an accurate and robust learning model. This paper proposes Epistemic-Constructed Physics-Informed Machine Learning (EPIC-PIML), which introduces an uncertainty mechanism that coordinates the outputs of physics and ML to produce a well-adjusted prediction from the HPIML model. Specifically, we attach an Epistemic Neural Network to a pre-trained HPIML model to estimate its epistemic uncertainty in closed form, and this uncertainty then drives the mechanism that determines the relative contribution of the physics and ML terms to the HPIML model's overall prediction. We prove that this mechanism converges to the correct trade-off as data and capacity grow, with a matching guarantee that its uncertainty estimate yields valid prediction sets even out-of-distribution. We further embed EPIC-PIML in a model predictive control framework, enabling closed-loop robotic control. We demonstrate our proposed method across two robotic systems at different scales: a quadrotor and an autonomous racing car. Across these experiments, the proposed model outperforms four baselines in system modeling and control performance.

1 INTRODUCTION

Model-based control in robotic systems, such as model-based reinforcement learning Moerland et al. (2023) or model predictive control Morari & Lee (1999), decides what actions to do next by minimizing a cost, predicted forward in time using a model of the system's dynamics Brunke et al. (2022). A model built purely from known physics is simple, interpretable, and requires no data at all. Still, it is rarely complete: real systems have unmodeled effects, such as drag, friction, or wear, that a nominal physics model omits. A model built purely with machine learning (ML) can, in principle, fit any residual effect. Still, it needs enough data to have actually seen the situation it is asked to predict, and it can fail silently when it has not Nghiem et al. (2023). Hybrid physics-informed machine learning (HPIML) combines the two: a known physics term predicts what physics already explains, and a learned term predicts the residual, the part physics misses Brunton & Kutz (2022); Karniadakis et al. (2021). This combination has been shown to improve prediction accuracy with less training data and to handle model mismatch better than either term alone, across domains ranging from aerial vehicles Chee et al. (2022); Torrente et al. (2021) to autonomous racing Kabzan et al. (2019) to soft robots Gao et al. (2024).

However, HPIML has a blind spot that it applies its learned correction with the same confidence everywhere, whether or not that confidence is warranted Mitchell (1997). In distribution, where the ML term has seen similar data before, its correction is trustworthy and improves the prediction. Out of distribution, where it has not, the same correction can be arbitrarily wrong, yet HPIML applies it just as confidently. What HPIML is missing is a mechanism that evaluates this uncertainty at each prediction and uses it to weigh the physics and ML terms accordingly, rather than always applying the same fixed combination.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We propose Epistemic-Constructed Physics-Informed Machine Learning (EPIC-PIML) to close this research gap. We attach an Epistemic Neural Network (ENN) Osband et al. (2023a) to a pre-trained HPIML model. Using a linear construction, the ENN estimates the model's epistemic uncertainty, the uncertainty that more data would resolve, in closed form, at the cost of a single extra forward pass rather than repeated sampling. This uncertainty then drives an adaptive mechanism that rescales the ML term: it lets the ML correction through when the model is confident, and falls back toward the physics term when it is not. We build the above mechanism from a family of monotone, ML-inspired function approximators, such as min-max networks Daniels & Velikova (2010), so that "more uncertain implies less ML correction" holds by an intuitive construction as the ML term is more uncertain in out-of-distribution region. We further prove that this mechanism converges to the correct trade-off as training data and mechanism capacity grow, and that its uncertainty estimate yields valid prediction sets even before convergence, including for out-of-distribution data.

To validate EPIC-PIML, we evaluate it against two criteria, system modeling accuracy and closed-loop control performance, on two robotic systems at different scales: a quadrotor and an autonomous racing car. We embed EPIC-PIML within a model-based planner, using model predictive path integral control Williams et al. (2017) to convert its uncertainty-aware predictions into control actions, re-evaluating the mechanism at each planning step. Across both systems, EPIC-PIML predicts the system dynamics more accurately than four baselines, such as physics alone, ML alone, physics-regularized ML, and HPIML without an uncertainty mechanism, and the resulting controller also tracks reference trajectories more accurately even under out-of-distribution conditions.

In summary, our contributions are:

- Develop an uncertainty-aware hybrid physics-informed machine learning architecture that attaches an Epistemic Neural Network to a pre-trained HPIML model.
- Propose an adaptive mechanism, built from monotone function approximators, that uses this uncertainty to re-balance the physics and ML terms of HPIML, with theoretical guarantees on both its convergence and the coverage of its uncertainty estimate.
- Provide a closed-loop model predictive controller built on EPIC-PIML, deployed and evaluated on two robotic systems via well-known simulators for each system.

2 Related work

In regression, a Bayesian neural network (BNN) Neal (1996) quantifies uncertainty by treating a network's weights as random variables, and Query-by-Committee (QBC) Burbidge et al. (2007) does so from the disagreement across an ensemble of models. Both are well-established, but they impose a high computational cost during training and inference, since each requires many models or many posterior samples. An Epistemic Neural Network (ENN) Osband et al. (2023a) instead estimates uncertainty from a single lightweight network and can be attached to an already pre-trained model, making it a practical choice for HPIML. ENN has already been applied to physics-informed models for uncertainty in learning partial differential equations Jacob et al. (2026), and to learning Q-functions for decision-making Chenreddy & Delage (2026); Osband et al. (2023b). At planning time, robust MPC Bemporad & Morari (1999) and GP-based MPC Hewing et al. (2020) instead use uncertainty by tightening constraints or adding a cost penalty from a fixed predictive uncertainty, such as a Gaussian process's variance, produced separately from the mean prediction. None of these methods feed the uncertainty estimate back into the predictive model itself to rebalance a hybrid physics-and-ML prediction: our mechanism does so directly and differentiably, needs no explicit chance constraint, and applies to both deterministic and stochastic HPIML backbones, such as a neural-network residual Chee et al. (2022) or a Gaussian-process residual Torrente et al. (2021), since the ENN's uncertainty is estimated independently of the point prediction and can reshape it directly, unlike a Gaussian process, whose mean and variance are coupled outputs of the same model. Nor do any of these study how well such uncertainty holds up over the multi-step, closed-loop predictions a controller actually needs. Separately, conformal prediction builds distribution-free prediction sets with guaranteed coverage from any point predictor Shafer & Vovk (2008); Angelopoulos & Bates (2023); our own coverage guarantee (Theorem 2) shares this goal, but follows directly from the ENN's Gaussian structure rather than a separate calibration step. This gap motivates our uncertainty mechanism: an explicit, monotonic use of ENN's uncertainty to decide, at each prediction, how much to trust the learned correction.

3 PRELIMINARIES

3.1 HYBRID PHYSICS-INFORMED MACHINE LEARNING

We consider a discrete-time dynamics model of a robotic system with the assumption that we can observe the full state of the system

$$\mathbf{x}_{k+1} = f_{\text{phy}}(\mathbf{x}_k, \mathbf{u}_k) + \delta_f(\mathbf{x}_k, \mathbf{u}_k), \tag{1}$$

where $\mathbf{x}_k$ and $\mathbf{u}_k$ are the vectors of states and control inputs at the time step k, respectively. f_{phy} is a known dynamics function, which can be a dynamic model or kinematic model of the system. In contrast, δ_f represents unmodeled dynamics or unknown disturbances. $\mathbf{x} \in \mathbb{R}^{n_x}$ and $\mathbf{u} \in \mathbb{R}^{n_u}$, where n_x and n_u are the dimensions of $\mathbf{x}$ and $\mathbf{u}$. In short, we denote $\mathbf{z}_k := \text{col}(\mathbf{x}_k, \mathbf{u}_k)$, where $\mathbf{z} \in \mathbb{R}^{n_x+n_u}$.

In order to learn the unmodeled term δ_f via a dataset observed from the states and control inputs, $\mathcal{D} = \{[\mathbf{x}_i, \mathbf{u}_i], \mathbf{x}_{i+1}\}_{i=1}^{N}$, a hybrid PIML architecture is provided Gao et al. (2024)

$$\hat{\mu}_{k+1} := f_{\text{hpiml}}(\mathbf{z}_k; \theta_{\text{ml}}) = f_{\text{phy}}(\mathbf{z}_k) + f_{\text{ml}}(\mathbf{z}_k; \theta_{\text{ml}}), \tag{2}$$

where the parameter θ_{ml} of the ML model f_{ml} in (2) is used to learn the dynamics of δ_f by fitting the prediction $\hat{\mu}_{k+1}$ into the dataset $\mathcal{D}$.

Rather than using an ML model to learn the whole dynamics, the architecture in (2) reduces the complexity of the learning task by focusing on the residual given a known dynamics f_{phy} on the collected data domain $\mathcal{D}$. However, this structure always assigns equal weight to the predictions of both f_{phy} and f_{ml}, whether in or out of distribution with the training data set. Hence, this is a drawback for HPIML's generalization capability. Next, we introduce a technique for computing the uncertainty of HPIML.

3.2 EPISTEMIC NEURAL NETWORK

Epistemic Neural Network (ENN) Osband et al. (2023a) generalizes a feed-forward neural network (NN) model by taking an additional *epistemic index* ϵ, drawn from a reference distribution P_ϵ (typically a standard Gaussian), as input alongside the input $\mathbf{z}$ of the predictive model. Where the NN's outputs can only express a *marginal* prediction, ENN's outputs $f_{\text{ml}}(\mathbf{z}, \epsilon; \theta_{\text{ml}})$ vary with ϵ, and integrating this variation over P_ϵ produces a *joint* prediction across multiple inputs by learning the uncertainty which can be resolved with more data (*epistemic uncertainty*).

We follow the *epinet* construction of Osband et al. (2023a): the epistemic index is injected additively on top of the base f_{hpiml} model in (2) through two neural networks, a learnable net and a prior net, giving the predictions

$$\hat{\mathbf{x}}_{k+1} = \underbrace{f_{\text{hpiml}}(\mathbf{z}_k; \theta_{\text{ml}})}_{\text{base model}} + \underbrace{\sigma^L(\texttt{sg}[\tilde{\mathbf{x}}_{k+1}], \epsilon; \theta_L)}_{\text{learnable}} + \underbrace{\sigma^P(\texttt{sg}[\tilde{\mathbf{x}}_{k+1}], \epsilon; \theta_P)}_{\text{prior net, fixed}}, \ \epsilon \sim \mathcal{N}(0, I_{D_z}) \tag{3a}$$

$$\tilde{\mathbf{x}}_{k+1} = [\mathbf{z}_k, \texttt{last_layer}(f_{\text{ml}}(\mathbf{z}_k; \theta_{\text{ml}}))] \tag{3b}$$

where D_z is the dimension of the epistemic index ϵ, $\texttt{sg}[\cdot]$ denotes the stop-gradient operator, so the learnable and prior nets see the reference features $\tilde{\mathbf{x}}_{k+1}$ as a constant rather than backpropagating through θ_{ml}. There are two networks here: a learnable net σ^L (with parameter θ_L) and a fixed prior net σ^P (with parameter θ_P), where σ^P has no trainable parameters and is typically randomly initialized Osband et al. (2018). Hence, it provides a persistent baseline for variation across ϵ that exists before any training occurs. With the prior net fixed, σ^L is instead trained to calibrate around that baseline variation rather than erase it, typically via a perturbed loss function (Section 4). One more advantage of these two neural networks is that they can be trained separately from the base model.

The sum of σ^L and σ^P in (3a) is called `epinet`. For one-step ahead prediction, this sum has zero mean, $\mathbb{E}_{\epsilon\sim P_\epsilon}[\sigma^L + \sigma^P] = \mathbf{0}$, which lets us compute the mean and the covariance matrix of the HPIML model via this `epinet` term

$$\hat{\mu}_{k+1} = \mathbb{E}_\epsilon[f_{\text{hpiml}}(\mathbf{z}_k; \theta_{\text{ml}})], \qquad (= f_{\text{hpiml}}(\mathbf{z}_k; \theta_{\text{ml}}), \text{ similar to } \hat{\mu}_{k+1} \text{ in (2)}) \tag{4a}$$

$$\Sigma_{k+1} = \mathbb{E}_\epsilon\big[[\sigma^L(\tilde{\mathbf{x}}_{k+1}, \epsilon; \theta_L) + \sigma^P(\tilde{\mathbf{x}}_{k+1}, \epsilon; \theta_P)][\sigma^L(\tilde{\mathbf{x}}_{k+1}, \epsilon; \theta_L) + \sigma^P(\tilde{\mathbf{x}}_{k+1}, \epsilon; \theta_P)]^\top\big]. \tag{4b}$$

For multi-input multi-output systems, the full covariance $\Sigma_{k+1} \in \mathbb{R}^{n_x \times n_x}$ also captures cross-output correlations. Also, the mean prediction of (4a) is exactly the prediction of (2), since $\mathbb{E}_\epsilon[\cdot]$ here is an exact expectation rather than an empirical average over sampled ϵ. Different from applications of ENN in learning and control Jacob et al. (2026); Chenreddy & Delage (2026), we leverage the covariance matrix Σ_{k+1}, which represents the epistemic uncertainty of HPIML, to propose a mechanism to adapt the effect of f_{phy} and f_{ml} for the predictive model (3a).

4 Proposed Method

4.1 Epistemic-Constructed Physics-Informed Machine Learning

The general (nonlinear-in-ϵ) Epistemic Neural Network (ENN) introduced in Section 3 carries no theoretical guarantee for the prior and learnable networks in that generality Osband et al. (2023a), but its linear-in-ϵ specialization is proven to converge in distribution to the exact Bayesian linear-regression posterior as the dimension of ϵ grows. We therefore adopt the linear-in-ϵ specialization directly: σ^L, σ^P no longer take ϵ as an input to a nonlinear map, but instead become ϵ-free coefficient matrices that ϵ multiplies

$$\hat{\mathbf{x}}_{k+1} = f_{\text{hpiml}}(\mathbf{z}_k; \theta_{\text{ml}}) + \left[\sigma^L(\texttt{sg}[\tilde{\mathbf{x}}_{k+1}]; \theta_L) + \sigma^P(\texttt{sg}[\tilde{\mathbf{x}}_{k+1}]; \theta_P)\right] \epsilon, \tag{5a}$$

$$\tilde{\mathbf{x}}_{k+1} = [\mathbf{z}_k, \texttt{last_layer}(f_{\text{ml}}(\mathbf{z}_k; \theta_{\text{ml}}))]. \tag{5b}$$

For $\tilde{\mathbf{x}}_{k+1}$, we can also substitute $\mathbf{z}_k$ by $f_{\text{phy}}(\mathbf{z}_k)$ to have physics information for the learnable and prior networks, then $\tilde{\mathbf{x}}_{k+1} = [f_{\text{phy}}(\mathbf{z}_k), \texttt{last_layer}(f_{\text{ml}}(\mathbf{z}_k; \theta_{\text{ml}}))]$. With the architecture given by (5), we can define the closed-form covariance matrix of HPIML that requires no sampling over ϵ

$$H_{k+1} = \sigma^L(\texttt{sg}[\tilde{\mathbf{x}}_{k+1}]; \theta_L) + \sigma^P(\texttt{sg}[\tilde{\mathbf{x}}_{k+1}]; \theta_P), \tag{6}$$

for the deterministic, ϵ-free matrix multiplying ϵ in (5a), and using $\mathbb{E}_\epsilon[\epsilon] = \mathbf{0}$ and $\mathbb{E}_\epsilon[\epsilon\epsilon^\top] = I_{D_z}$, we have $\mathbb{E}_\epsilon[\left(\sigma^L + \sigma^P\right)\epsilon] = H_{k+1}\mathbb{E}_\epsilon[\epsilon] = \mathbf{0}$ (validating the covariance identification in (4b)) and

$$\Sigma_{k+1} = \mathbb{E}_\epsilon\left[H_{k+1}\epsilon\epsilon^\top H_{k+1}^\top\right] = H_{k+1}H_{k+1}^\top. \tag{7}$$

Computing Σ_{k+1} therefore costs a single forward pass and a matrix product, not Monte Carlo estimation over ϵ, which is essential for evaluating it at every rollout step of the planner in Section 5. Note that Σ_{k+1} depends only on the architecture above, not on the uncertainty mechanism introduced next.

4.2 Uncertainty Mechanism of EPIC-PIML

Adaptive mechanism via model uncertainty (EPIC). In order to improve the accuracy of the predictive model, we introduce an adaptive mechanism $\alpha(.)$ to the HPIML framework, which receives the covariance matrix Σ as input

$$\hat{\mu}_{k+1}(\alpha) = f_{\text{phy}}(\mathbf{z}_k) + \alpha(\Sigma_{k+1}; \theta_\vartheta) \odot f_{\text{ml}}(\mathbf{z}_k; \theta_{\text{ml}}), \tag{8}$$

where $\alpha(\Sigma_{k+1}; \theta_\vartheta) \in [0, 1]^{n_x}$ is applied element-wise as $[\alpha(\Sigma_{k+1}; \theta_\vartheta)]_i = \alpha_i(\Sigma_{k+1,ii}; \theta_{\vartheta,i})$, where $\Sigma_{k+1,ii}$ is the i-th diagonal entry of Σ_{k+1}, $i = 1, \ldots, n_x$, and $\odot$ denotes element-wise multiplication. Here, by focusing on the uncertainty of each state, we consider only the i-th output's own uncertainty, i.e., the value of $\Sigma_{k+1,ii}$ in the covariance matrix Σ_{k+1}, when predicting the i-th output of $\hat{\mu}_{k+1}$. This also reduces the computational burden in inference and training by relaxing the correlation between n_x states. Because the n_x inputs $\Sigma_{k+1,ii}$, $i = 1, \ldots, n_x$, are independent of each other, we use n_x independent mechanism functions $\alpha(.) = [\alpha_1(.), \ldots, \alpha_{n_x}(.)]$, one for each state, each with its own parameters $\theta_\vartheta = [\theta_{\vartheta,1}, \ldots, \theta_{\vartheta,n_x}]$, for faster convergence in training. Independent parameters mean that no output needs to share its scale with any other: $\theta_{\vartheta,i}$ absorbs whatever units and magnitude $\Sigma_{k+1,ii}$ has for output i, so no cross-output normalization is required. Figure 1 illustrates the complete architecture of EPIC-PIML and its inference is presented in Algorithm 1 at Appendix B.

Since Σ represents the uncertainty of f_{ml}, we want the predictive model to prioritize f_{phy} and reduce the weight of f_{ml} as this uncertainty increases. Each $\alpha_i(\cdot)$ is therefore monotonically decreasing in $\Sigma_{k+1,ii}$: when epistemic uncertainty for output i is high, $\alpha_i \to 0$ and the prediction falls back to

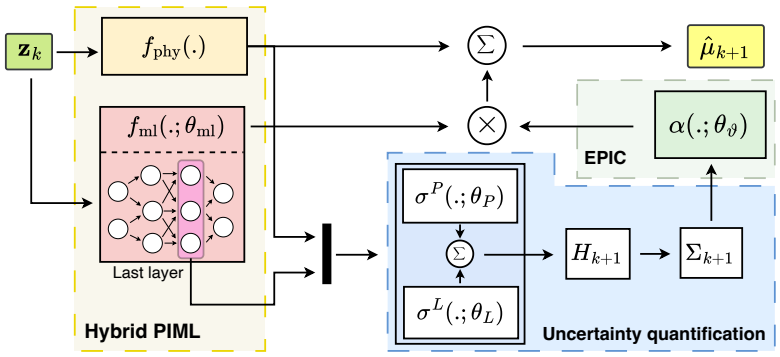


Figure 1: EPIC-PIML architecture with input $\mathbf{z}_k$ and output $\hat{\mu}_{k+1}$.

the physics term; when it is low, $\alpha_i \to 1$ and the ML correction is fully applied. To construct $\alpha(.)$, we may adopt monotonicity-informed machine learning techniques such as min-max NN Daniels & Velikova (2010), monotonic Taylor NN Nguyen et al. (2025), or lattice NN You et al. (2017), since the mechanism is bounded between 0 and 1 and these families let us enforce that bound by construction. This paper adopts the min-max NN as the higher-capacity family, since it is a common and simple method within the monotonicity-informed NN group.

Training θ_ϑ well requires an unseen data set to quantify uncertainty on, which raises a separate problem: there is not enough data to train a monotonicity-informed NN for the mechanism $\alpha(.)$. We address this by proposing two architectures for the mechanism: a min-max NN for sufficient data and a Kalman gate for insufficient data (Appendix A).

4.3 Training and Inference of EPIC-PIML

The base model can be pre-trained or trained jointly with the learnable network. For generality, we adopt the pre-trained setting and partition the dataset into two disjoint subsets: $\mathcal{D} = \mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{tune}}$, with $\mathcal{D}_{\text{base}} \cap \mathcal{D}_{\text{tune}} = \emptyset$. The parameters of (8) are trained sequentially. Since f_{phy} is fixed and known, only θ_{ml} is fit, by minimizing the mean-squared prediction error with L_2 regularization on $\mathcal{D}_{\text{base}}$

$$\mathcal{L}_{\text{base}}(\theta_{\text{ml}}|\mathcal{D}_{\text{base}}) = \frac{1}{|\mathcal{D}_{\text{base}}|} \sum_{i=1}^{|\mathcal{D}_{\text{base}}|} \left\|\mathbf{x}_{i+1} - f_{\text{phy}}(\mathbf{z}_i) - f_{\text{ml}}(\mathbf{z}_i;\theta_{\text{ml}})\right\|_2^2 + \gamma\|\theta_{\text{ml}}\|_2^2. \tag{9}$$

With θ_{ml} fixed, the learnable network θ_L is trained via a perturbed loss. For each sample i, a reference perturbation matrix $\mathbf{C}_i \in \mathbb{R}^{n_x \times D_z}$ is drawn i.i.d. from $\mathcal{N}(0,1)$, which introduces diversity between epistemic indices ϵ, allowing `epinet` to capture residual uncertainty of the base model

$$\mathcal{L}_{\text{epi}}(\theta_L|\mathcal{D}_{\text{base}}) = \frac{1}{|\mathcal{D}_{\text{base}}|} \sum_{i=1}^{|\mathcal{D}_{\text{base}}|} \mathbb{E}_\epsilon\left[\|\mathbf{x}_{i+1} - \hat{\mathbf{x}}_{i+1}(\alpha=\mathbf{1}) + \lambda_1 \mathbf{C}_i\epsilon\|_2^2\right] + \lambda_2\|\theta_L\|_2^2, \tag{10}$$

where $\hat{\mathbf{x}}_{i+1}(\alpha=\mathbf{1})$ is the prediction of the full model with the mechanism set to one, and θ_P is fixed and randomly initialized on the unit sphere as in Osband et al. (2018).

With θ_{ml} and θ_L both fixed, the mechanism parameters θ_ϑ are learned on the combined dataset $\mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{tune}}$. This mixes two regimes: on $\mathcal{D}_{\text{base}}$, which the base model has already fit, the `epinet` typically reports low epistemic uncertainty, teaching the mechanism what a trustworthy ML correction looks like. On $\mathcal{D}_{\text{tune}}$, which the base model has never observed, the `epinet` produces honest held-out uncertainty estimates regardless of whether the sample is in-distribution or out-of-distribution, teaching the mechanism to suppress when that uncertainty is genuinely high

$$\mathcal{L}_{\text{tune}}(\theta_\vartheta|\mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{tune}}) = \frac{1}{|\mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{tune}}|} \sum_{i=1}^{|\mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{tune}}|} \left\|\hat{\mu}_{i+1}(\alpha) - \mathbf{x}_{i+1}\right\|_2^2 + \beta\|\theta_\vartheta\|_2^2 \tag{11}$$

where $\gamma, \lambda_1, \lambda_2, \beta$ are regularization hyperparameters.

Once θ_L and θ_ϑ are calibrated, we refit θ_{ml} once more by minimizing $\mathcal{L}_{\text{base}}$ (9), now over the combined dataset $\mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{tune}}$; $\theta_L, \theta_P, \theta_\vartheta$ remain fixed at their already-calibrated values.

4.4 THEORETICAL GUARANTEES

We establish two properties that together justify EPIC-PIML: (1) as mechanism capacity and the combined dataset $\mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{tune}}$ grow, the tuning loss converges to zero under a realizability assumption on the true disturbance (and, by Osband et al. (2023a)'s existing ENN convergence result, so does the training loss $\mathcal{L}_{\text{epi}}$), so the model recovers the true disturbance δ_f wherever $\mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{tune}}$ has support (Theorem 1); and (2) even where that support has not yet been reached, including out-of-distribution, the uncertainty estimate Σ_{k+1} yields a valid prediction set despite the train-inference mismatch introduced by the mechanism (Theorem 2).

Assumption 1 (Realizable Mechanism)**.** *$\delta_f : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ is continuous on the (compact) support of P. For each output i, there exists a continuous, monotone non-increasing $\alpha_i^\star : \mathbb{R}_{\geq 0} \to [0,1]$ such that*

$$\alpha_i^\star(\Sigma_{k+1,ii}) \cdot [f_{ml}(\mathbf{z}_k)]_i = [\delta_f(\mathbf{z}_k)]_i \quad \text{for } P\text{-a.e. } \mathbf{z}_k. \tag{12}$$

Theorem 1 (Convergence of EPIC-PIML)**.** *Under Assumption 1, let each $\alpha_i(\cdot;\theta_{\vartheta,i})$ range over the min-max family* (19) *with L, M pieces, and let $\hat{\theta}_\vartheta$ minimize* (11) *on $\mathcal{D}_{base} \cup \mathcal{D}_{tune} \sim P^{N_\vartheta}$ i.i.d., where $N_\vartheta \doteq N_{train} + N_{tune}$. Then for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\mathcal{L}_P(\hat{\theta}_\vartheta) \;\leq\; \varepsilon_{approx}(L,M) + \varepsilon_{gen}(L,M,N_\vartheta,\delta), \tag{13}$$

where $\mathcal{L}_P(\theta_\vartheta) = \mathbb{E}_P[\|\hat{\mu}(\alpha) - \mathbf{x}\|_2^2]$, $\varepsilon_{approx}(L,M) \to 0$ as $L, M \to \infty$ by universal approximation of (19) *over the compact range of $\Sigma_{k+1,ii}$ and $\varepsilon_{gen}(L,M,N_\vartheta,\delta) \to 0$ as $N_\vartheta \to \infty$ provided L, M grow at a suitably controlled rate relative to N_ϑ. Consequently $\mathcal{L}_P(\hat{\theta}_\vartheta) \to 0$: the mechanism-adjusted prediction $\hat{\mu}(\alpha)$ converges to $\mathbf{x}_{k+1} = f_{phy} + \delta_f$ in mean-squared error.*

Proof. See Appendix C.1. □

Because Σ_{k+1} is used directly inside the closed-loop controller of Section 5 to make decisions under uncertainty, it must remain a valid uncertainty quantification even where the mechanism has not converged, including out-of-distribution; otherwise the controller could be planning against a miscalibrated estimate exactly where reliability matters most. The `epinet` is trained with $\alpha = \mathbf{1}$ but deployed with $\alpha \neq \mathbf{1}$, so the prediction center shifts by a mechanism-induced bias Δ_{k+1}. We verify that Σ_{k+1} still yields a valid prediction set under this mismatch, and bound the growth of the coverage threshold.

Theorem 2 (Uncertainty Coverage)**.** *Under the EPIC-PIML model* (8) *and the Gaussian prior interpretation of the linear ENN, define the mechanism-induced bias*

$$\Delta_{k+1}(\alpha) = (\mathbf{1} - \alpha(\Sigma_{k+1};\theta_\vartheta)) \odot f_{ml}(\mathbf{z}_k;\theta_{ml}) \tag{14}$$

and the non-centrality $\lambda_{k+1} = \Delta_{k+1}^\top \Sigma_{k+1}^{-1} \Delta_{k+1}$. Then the prediction set

$$\mathcal{C}_{1-\delta}(\mathbf{z}_k) = \left\{\mathbf{x} : (\mathbf{x} - \hat{\mu}_{k+1}(\alpha))^\top \Sigma_{k+1}^{-1} (\mathbf{x} - \hat{\mu}_{k+1}(\alpha)) \leq \chi^2_{n_x,\,1-\delta}(\lambda_{k+1})\right\} \tag{15}$$

satisfies $P(\mathbf{x}_{k+1} \in \mathcal{C}_{1-\delta}(\mathbf{z}_k)) \geq 1-\delta$, where $\chi^2_{n_x,1-\delta}(\lambda)$ is the $(1-\delta)$ quantile of the non-central chi-squared distribution with n_x degrees of freedom and non-centrality λ. Moreover, suppose the ML term is bounded, i.e. there exists $B_f < \infty$ with $\|f_{ml}(\mathbf{z}_k)\|_2 \leq B_f$ for all $\mathbf{z}_k$, and the epistemic covariance never fully collapses, i.e. there exists $\sigma_{min} > 0$ with

$$\Sigma_{k+1} \succeq \sigma_{min}^2 I_{n_x} \quad \text{for all } \mathbf{z}_k. \tag{16}$$

Then $\lambda_{k+1} \leq B^2 := B_f^2/\sigma_{min}^2$ uniformly over $\mathbf{z}_k$ and over the realized mechanism $\alpha \in [0,1]^{n_x}$, and the coverage threshold is bounded by the fixed quantile $\chi^2_{n_x,1-\delta}(B^2)$.

Proof. See Appendix C.2. □

5 EPIC-PIML-BASED MODEL PREDICTIVE CONTROL

We adopt a data-driven model predictive control (MPC) approach Torrente et al. (2021); Chee et al. (2022), where the learning model serves as the predictive model in the MPC framework. At decision step k, EPIC-PIML-based MPC (EPIC-MPC) selects optimal control inputs $\mathbf{u}_k$ by planning T_H steps ahead through the model (8) with its uncertainty mechanism, re-evaluating α online at every predicted state.

Solving this optimization with gradient-based methods, however, becomes computationally expensive for complex data-driven MPC problems. We therefore adopt Model Predictive Path Integral (MPPI) planning Williams et al. (2017), a gradient-free method. Candidate action sequences $(\mathbf{u}_k, \ldots, \mathbf{u}_{k+T_H-1})$ are drawn from a time-dependent Gaussian $\mathbf{u}_{k+h} \sim \mathcal{N}(\mathbf{m}_h, \mathbf{s}_h^2 I)$, rolled out through (8), and scored by the objective function

$$\Phi(\mathbf{u}_{k:k+T_H-1}) = \sum_{h=0}^{T_H-1} \gamma_{\text{mppi}}^h [R(\hat{\mathbf{x}}_{k+h}, \mathbf{u}_{k+h}) - c_{\text{risk}}\text{trace}(\Sigma_{k+h})] + R^{T_H}(\hat{\mathbf{x}}_{k+T_H}), \tag{17}$$

where $\hat{\mathbf{x}}_k = \mathbf{x}_k$ is the current state, $\hat{\mathbf{x}}_{k+h} = \hat{\mu}_{k+h}(\alpha)$ is the mechanism-adjusted rollout state (8), γ_{mppi} is the discount factor, $R(\cdot)$ is the stage reward function, and $R^{T_H}(\cdot)$ is the terminal cost of the MPPI objective. The term $c_{\text{risk}}\,\text{trace}(\Sigma_{k+h})$ penalizes the epistemic uncertainty (7) along the rollout with weight $c_{\text{risk}} \geq 0$, so the planner avoids trajectories through regions where the model is uncertain, and $c_{\text{risk}} = 0$ recovers the plain reward objective.

Given the top-K scoring trajectories with weights $\Omega_i = \exp(\tau_{\text{mppi}}\Phi_i)$, where τ_{mppi} is the temperature factor, MPPI updates the sampling distribution for iteration j as

$$\mathbf{m}_h^{(j)} = \frac{\sum_{i=1}^K \Omega_i \mathbf{u}_{k+h,i}}{\sum_{i=1}^K \Omega_i}, \quad \mathbf{s}_h^{(j)} = \left(\frac{\sum_{i=1}^K \Omega_i (\mathbf{u}_{k+h,i} - \mathbf{m}_h^{(j)})^2}{\sum_{i=1}^K \Omega_i} \right)^{1/2} \tag{18}$$

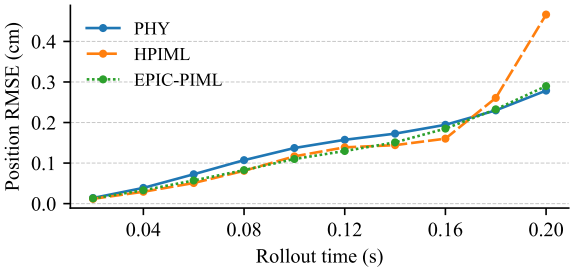
for $h = 0, \ldots, T_H - 1$. After J iterations, EPIC-MPC executes $\mathbf{u}_k = \mathbf{m}_0^{(J)}$ in receding-horizon fashion. Algorithm 2 in the Appendix B summarizes EPIC-MPC at deployment: at each step it plans through the model via receding-horizon MPPI.

6 EXPERIMENTAL RESULTS

For a comprehensive investigation, we compare EPIC-PIML with four baselines: PHY, whose predictive model is f_{phy}; ML, whose predictive model is $f_{\text{ml}}(.)$, learning the entire robot dynamics from data; PIML, whose ML model has a physics (f_{phy}) regularization added to its training loss; and HPIML, whose predictive model is the hybrid $f_{\text{hpiml}}(.)$. Also, all five models are trained with the same amount of training data $\mathcal{D}$, and are constructed with the same architecture (hyperparameters) for a fair comparison. For system identification, we evaluate the five models on the test dataset $\mathcal{D}_{\text{test}}$ using position root mean square error (RMSE). In terms of control, we evaluate reference-position tracking performance for both the quadrotor and the autonomous racing car. We show more detailed results for both the quadrotor and the autonomous car in Appendix D.

6.1 3D QUADROTOR ON `safe-control-gym`

For this scenario, we consider a quadrotor with 12 states and 4 control inputs on `safe-control-gym` Yuan et al. (2022), where a nominal physics-based dynamic model is used to construct f_{phy}. Further details of the experimental setup are provided in Appendix D.1. Figure 2 presents the system modeling accuracy across all five models. At the first prediction step, HPIML and EPIC-PIML achieve highly accurate results, with a one-step RMSE of $0.006\,\text{cm}$, lower than that of the PHY model. However, beyond 10 rollout steps, HPIML's position RMSE grows rapidly, while PHY and EPIC-PIML remain robust: their mean RMSE after 20 steps stays below $0.5\,\text{cm}$, compared to $1.2\,\text{cm}$ for HPIML. The ML and PIML models exhibit much higher error, exceeding $90\,\text{cm}$ after 20 rollout steps, since a pure ML model requires a large amount of training data to learn complex dynamics, whereas only 16,478 training samples are available here for the quadrotor. This further confirms that the HPIML and EPIC-PIML approaches work well under a limited data



Model	1 step	10 steps	20 steps
PHY	0.007	0.128	**0.473**
ML	3.003	24.77	97.76
PIML	3.023	24.78	94.10
HPIML	**0.006**	0.133	1.203
EPIC-PIML	**0.006**	**0.117**	**0.483**

Figure 2: Open-loop rollout position RMSE (cm) on $\mathcal{D}_{\text{test}}$ (↓) ($\mathcal{D}_{\text{test}}$ collected from a different track) with step time $\Delta t = 0.02\,\text{s}$. Left: rollout RMSE. Right: mean rollout RMSE at 1, 10, and 20 steps.

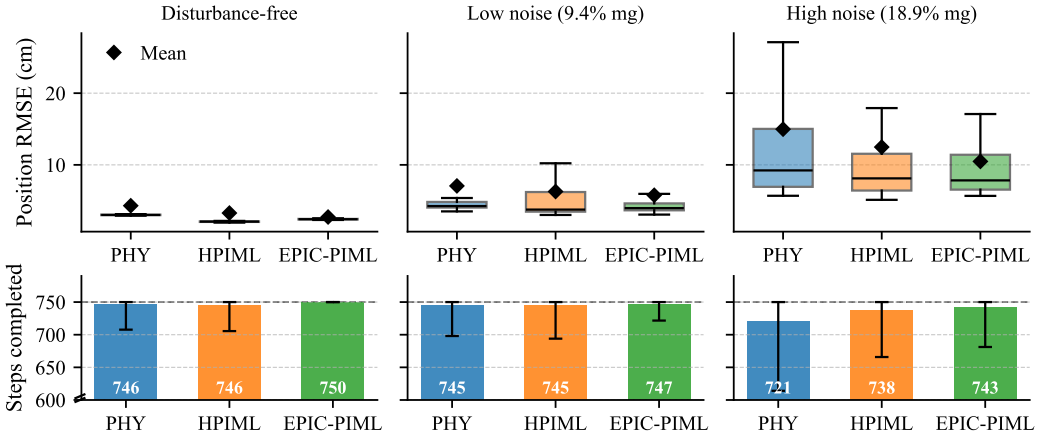


Figure 3: Quadrotor tracking performance across 100 episodes with different initial positions under noise (the reference track matches training). Top: box plots of position RMSE (cm) (↓). Bottom: the mean number of steps completed (↑), with a maximum of 750. (mg: the product of mass and gravitational acceleration, expressing the disturbance force as a fraction of quadrotor weight.)

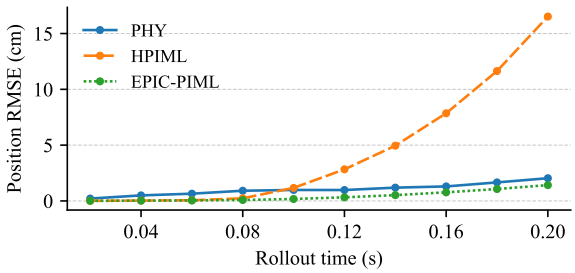
collection budget, a common constraint in robotics practice since real-world experiments are costly. Appendix D.1 illustrates the 3D quadrotor trajectories of five models.

Figure 3 presents the closed-loop reference-tracking performance of PHY, HPIML, and EPIC-PIML under the MPPI framework across three noise conditions. Each episode requires the quadrotor to complete three cycles within 15 s, starting from randomized initial positions distinct from the training data $\mathcal{D}$. EPIC-PIML consistently achieves the smallest mean position RMSE across all three noise levels. Its interquartile spread is also the smallest among the three models, showing that EPIC-PIML remains robust across different noise levels and OOD initial states. Since 750 steps correspond to completed track, EPIC-PIML also completes the track most reliably under OOD conditions, achieving the highest mean number of steps completed with the smallest standard deviation.

6.2 Autonomous Racing Car on `F1TENTH-gym`

We consider an F1TENTH car (scale 1/10 relative to a full-scale car) with 7 states and 2 control inputs on `F1TENTH-gym` O'Kelly et al. (2020). Although it has fewer states than the quadrotor, the nominal physics f_{phy} we use here is a kinematic model. Hence, we have less physics information available than the quadrotor's dynamic model. Because PHY is only a kinematic model, the low first-step RMSE of HPIML and EPIC-PIML, $0.009\,\text{cm}$ (Figure 4), demonstrates their ML terms' ability to learn the car's unmodeled longitudinal and lateral forces. However, after rolling out 10 or 20 steps, HPIML becomes worse than both PHY and EPIC-PIML, reaching a mean RMSE of $89\,\text{cm}$ at 20 rollout steps, on par with the pure black-box ML and PIML models.

The control objective is for the car to race along the track's centerline at a constant speed. Figure 5 illustrates the control performance of PHY, HPIML, and EPIC-PIML. At speed $4\,\text{m}\,\text{s}^{-1}$, where the training data is collected the most, HPIML and EPIC-PIML outperform the PHY model, both with lateral offset below $2\,\text{cm}$ (a smaller lateral offset means the car stays closer to the center line) and with faster lap times. At speeds 5 and $6\,\text{m}\,\text{s}^{-1}$, HPIML crashes into the track boundary, while PHY



Model	1 step	10 steps	20 steps
PHY	0.212	2.036	12.987
HPIML	**0.009**	16.519	89.229
EPIC-PIML	**0.009**	**1.419**	**7.956**
ML	4.064	42.097	88.471
PIML	3.199	30.075	64.710

Figure 4: Open-loop rollout position RMSE (cm) on $\mathcal{D}_{\text{test}}$ (↓) with step time $\Delta t = 0.02\,\text{s}$. Left: rollout RMSE. Right: mean rollout RMSE at 1, 10, and 20 steps.

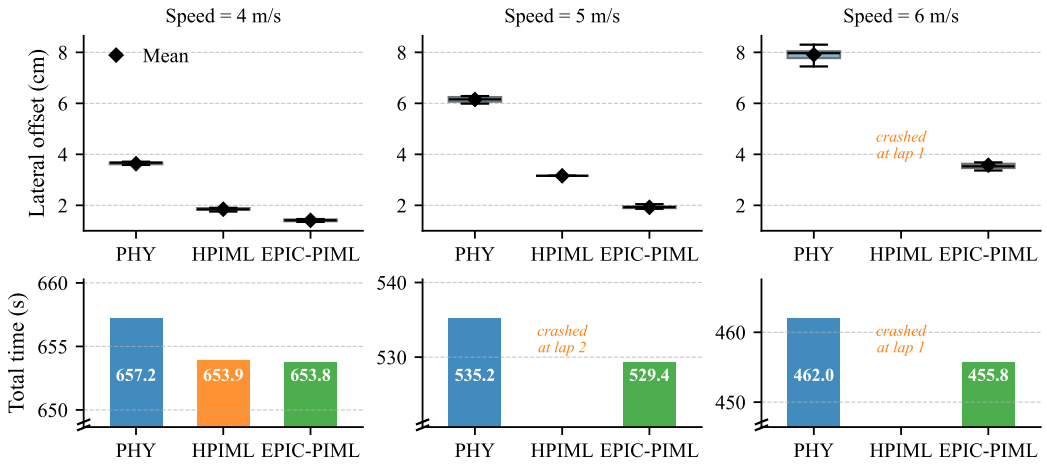


Figure 5: F1TENTH tracking performance at target speeds of 4, 5, and $6\,\text{m}\,\text{s}^{-1}$. Top: box plots of lateral-offset RMSE (cm; ↓). Bottom: total time to complete 10 laps (s; ↓).

and EPIC-PIML still complete all 10 laps. Additionally, EPIC-PIML always achieves a lower lateral offset than PHY and completes 10 laps faster, by about 8 s at speed $6\,\text{m}\,\text{s}^{-1}$. More details and setup of this scenario are presented in Appendix D.2.

Limitations: Although EPIC-PIML achieves better accuracy in both system modeling and control across both systems, it requires more computational time than the other baselines. This cost is most clear in the control setting, where EPIC-PIML must be evaluated N_{pop} times per MPPI iteration, across J iterations, at every control step k. For instance, for the autonomous car, EPIC-PIML's average per-step compute time is 119.4 ms, compared to 33.2 ms for HPIML, measured on an AMD Ryzen 9 9950X (16-core) CPU with an NVIDIA RTX 5090 GPU.

7 Conclusion

EPIC-PIML shows that hybrid physics–learning models can benefit from adapting the learned residual based on epistemic uncertainty, particularly when predictions extend beyond the training distribution. The experiments further indicate that one-step accuracy alone can be misleading for model-based control, since small local errors may accumulate substantially over long-horizon rollouts. By retaining the physics model as a reliable fallback while modulating the learned correction, EPIC-PIML improves both predictive robustness and closed-loop control across the two robotic systems considered. This robustness comes at the cost of additional online computation, motivating future work on more efficient, uncertainty-aware architectures and real-time deployment on physical robotic platforms. In future work, we plan to move this cost offline by training a reinforcement learning policy on EPIC-PIML data, with the goal of real-time deployment on real-world robotic systems.

A.I. DISCLOSURE

In this work, we used generative AI tools (e.g., Claude, ChatGPT) to revise passages and correct grammar. All research ideas, methods, experiments, analyses, and conclusions are the authors' own, and the authors reviewed and take full responsibility for all content of this paper.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we have provided comprehensive details across the paper and supplementary materials. The formal proofs for all theoretical claims and explicit listings of our mathematical assumptions are detailed in Appendix C. The architecture of our proposed model is documented in Section 4, and specific hyperparameter configurations and experimental environments are documented in Section 6 and Appendix D.

To facilitate validation, the complete source code, preprocessing scripts, and execution workflows are included in the supplementary materials as an anonymized downloadable zip file, containing three sub-archives: `EPIC-PIML-main.zip`, an overview of our code; `EPIC-PIML-3d-quadrotor.zip`, the code for the quadrotor experiments; and `EPIC-PIML-autonomous-car.zip`, the code for the F1TENTH car experiments. We provide trained model checkpoints directly, together with data collection scripts and instructions in each zip file's README, so that the training and evaluation datasets can be regenerated from the public simulation environments.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, UK, 1999.

Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In A. Garulli and A. Tesi (eds.), *Robustness in identification and control*, pp. 207–226, London, 1999. Springer London.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022.

Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, Cambridge, UK, 2nd edition, 2022.

Robert Burbidge, Jem J. Rowland, and Ross D. King. Active learning for regression based on query by committee. In Hujun Yin, Peter Tino, Emilio Corchado, Will Byrne, and Xin Yao (eds.), *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, pp. 209–218, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

Kong Yao Chee, Tom Z. Jiahao, and M. Ani Hsieh. KNODE-MPC: A knowledge-based data-driven predictive control framework for aerial robots. *IEEE Robotics and Automation Letters*, 7(2): 2819–2826, 2022.

Abhilash Reddy Chenreddy and Erick Delage. Epistemic robust offline reinforcement learning, 2026. URL `https://arxiv.org/abs/2604.07072`.

Hennie Daniels and Marina Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

Junpeng Gao, Mike Y. Michelis, Andrew Spielberg, and Robert K. Katzschmann. Sim-to-real of soft robots with learned residual physics. *IEEE Robotics and Automation Letters*, 9(10):8523–8530, 2024.

Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2020.

Bruno Jacob, Ashish S Nair, Amanda A Howard, Jan Drgona, and Panos Stinis. E-PINNs: epistemic physics-informed neural networks. *Machine Learning for Computational Science and Engineering*, 2(2):39, 2026.

Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099, Seoul, South Korea, 2015.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, USA, 1997.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023.

Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, New York, NY, USA, 1996.

Truong X. Nghiem, Ján Drgoňa, Colin Jones, Zoltan Nagy, Roland Schwan, Biswadip Dey, Ankush Chakrabarty, Stefano Di Cairano, Joel A. Paulson, Andrea Carron, Melanie N. Zeilinger, Wenceslao Shaw Cortez, and Draguna L. Vrabie. Physics-informed machine learning for modeling and control of dynamical systems. In *2023 American Control Conference (ACC)*, pp. 3735–3750, San Diego, CA, USA, 2023.

Nam T. Nguyen, Juan C. Tique, and Truong X. Nghiem. Physics-constrained Taylor neural networks for learning and control of dynamical systems. In *2025 American Control Conference (ACC)*, pp. 1821–1826, Denver, CO, USA, 2025.

Matthew O'Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. F1TENTH: An open-source evaluation environment for continuous control and reinforcement learning. In Hugo Jair Escalante and Raia Hadsell (eds.), *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, volume 123 of *Proceedings of Machine Learning Research*, pp. 77–89, Vancouver, BC, Canada, 2020. PMLR.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31, Montreal, Canada, 2018. Curran Associates, Inc.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Epistemic neural networks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 2795–2823, New Orleans, LA, USA, 2023a. Curran Associates, Inc.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Approximate Thompson sampling via epistemic neural networks. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI '23, Pittsburgh, PA, USA, 2023b. JMLR.org.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(12):371–421, 2008.

Guillem Torrente, Elia Kaufmann, Philipp Föhn, and Davide Scaramuzza. Data-driven MPC for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776, 2021.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017.

Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya R. Gupta. Deep lattice networks and partial monotonic functions. In *Advances in Neural Information Processing Systems*, volume 30, pp. 2981–2989, Long Beach, CA, USA, 2017. Curran Associates, Inc.

Zhaocong Yuan, Adam W. Hall, Siqi Zhou, Lukas Brunke, Melissa Greeff, Jacopo Panerati, and Angela P. Schoellig. Safe-control-gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics. *IEEE Robotics and Automation Letters*, 7(4):11142–11149, 2022.

A EPIC MECHANISM

Monotone min-max networks with sufficient data. We define the min-max NN mechanism Daniels & Velikova (2010) as follows:

$$g_i(t;\theta_{\vartheta,i}) = \min_{l=1,\dots,L} \max_{m=1,\dots,M} \left(w_{i,lm}\, t + b_{i,lm}\right), \quad w_{i,lm} \le 0, \tag{19}$$

evaluated at $t = \Sigma_{k+1,ii}$, with $\theta_{\vartheta,i} = \{w_{i,lm}, b_{i,lm}\}_{l,m}$ and $\alpha_i(t;\theta_{\vartheta,i}) = \mathrm{clip}\big(g_i(t;\theta_{\vartheta,i}), 0, 1\big)$. We use this family for concreteness; any other monotone-by-construction parametric family (e.g. an isotonic or weight-constrained network) would serve the same role in what follows. Each linear piece $w_{i,lm}t + b_{i,lm}$ is non-increasing in t since $w_{i,lm} \le 0$, and a min/max of non-increasing functions is non-increasing, so (19) is monotone decreasing for *any* choice of parameters: monotonicity is guaranteed by construction.

Mechanism form with insufficient data. Viewing the physics term as a prior on the residual $[\mathbf{r}]_i = [\mathbf{x}_{k+1} - f_{\text{phy}}]_i$ with scale τ_i^2, and the ML term as a noisy measurement of $[\mathbf{r}]_i$ with epistemic variance $\Sigma_{k+1,ii}$, Gaussian conjugacy gives the posterior-mean (Kalman) weight $\tau_i^2/(\tau_i^2 + \Sigma_{k+1,ii}) = 1/(1 + \Sigma_{k+1,ii}/\tau_i^2)$. We parameterize this exactly

$$\alpha_i(\Sigma_{k+1,ii};\theta_{\vartheta,i}) = \frac{1}{1 + (\Sigma_{k+1,ii}/c_{\vartheta,i})^{p_{\vartheta,i}}}, \tag{20}$$

with $\theta_{\vartheta,i} = (c_{\vartheta,i}, p_{\vartheta,i})$: at $p_{\vartheta,i} = 1$, $c_{\vartheta,i} = \tau_i^2$ ($c_{\vartheta,i} > 0,\ p_{\vartheta,i} \ge 1$) recovers the Kalman weight, while $p_{\vartheta,i}$ otherwise sharpens the cutoff. Equation (20) is monotone decreasing by construction and satisfies $\alpha_i \to 0$ as $\Sigma_{k+1,ii} \to \infty$, guaranteeing fallback to the physics model in the deep out-of-distribution regime.

B EPIC-MPC

Algorithm 1: Inference of EPIC-PIML

Require: trained $\theta_{\text{ml}}, \theta_L, \theta_P, \theta_\vartheta$; physics model f_{phy}; state-action $\mathbf{z}_k$
Compute $f_{\text{phy}}(\mathbf{z}_k)$ and $f_{\text{ml}}(\mathbf{z}_k; \theta_{\text{ml}})$
Form reference features $\tilde{\mathbf{x}}_{k+1} = [f_{\text{phy}}(\mathbf{z}_k), \texttt{last_layer}(f_{\text{ml}}(\mathbf{z}_k; \theta_{\text{ml}}))]$
Compute $H_{k+1} = \sigma^L(\texttt{sg}[\tilde{\mathbf{x}}_{k+1}]; \theta_L) + \sigma^P(\texttt{sg}[\tilde{\mathbf{x}}_{k+1}]; \theta_P)$ (6)
Compute $\Sigma_{k+1} = H_{k+1} H_{k+1}^\top$ (7)
Compute the mechanism $\alpha(\Sigma_{k+1}; \theta_\vartheta)$, per output, via (20)
Compute $\hat{\mu}_{k+1}(\alpha) = f_{\text{phy}}(\mathbf{z}_k) + \alpha(\Sigma_{k+1}; \theta_\vartheta) \odot f_{\text{ml}}(\mathbf{z}_k; \theta_{\text{ml}})$ (8)
return $\hat{\mu}_{k+1}(\alpha), \Sigma_{k+1}$

Algorithm 2: EPIC-MPC Deployment

Require: trained $\theta_{\text{ml}}, \theta_L, \theta_\vartheta$ (EPIC-PIML); prior net θ_P; physics model f_{phy}; initial state $\mathbf{x}_0$; horizon T_H, iterations J, population size N_{pop}, elite count K, discount factor γ_{mppi}, temperature τ_{mppi}, risk weight c_{risk}, initial sampling std σ_{init}
for $k = 0, 1, 2, \ldots$ **do**
  Initialize $(\mathbf{m}_h^{(0)}, \mathbf{s}_h^{(0)}) = (\mathbf{0}, \sigma_{\text{init}})$ for $h = 0, \ldots, T_H - 1$
  for $j = 1, \ldots, J$ **do**
    Sample N_{pop} action sequences from $\mathcal{N}(\mathbf{m}_h^{(j-1)}, (\mathbf{s}_h^{(j-1)})^2 I)$
    for each sampled sequence **do**
      Roll out $\hat{\mathbf{x}}_{k+1:k+T_H}$ and collect $\Sigma_{k+1:k+T_H}$ at each step k via Algorithm 1
      Score the sequence with $\Phi(\cdot)$ from (17)
    end for
    Update $(\mathbf{m}_h^{(j)}, \mathbf{s}_h^{(j)})$ from the top-K sequences via (18)
  end for
  Execute $\mathbf{u}_k = \mathbf{m}_0^{(J)}$
  Advance to $\mathbf{x}_{k+1}$
end for

C PROOFS

C.1 PROOF OF THEOREM 1

Proof. By Assumption 1, $\mathcal{L}_P(\alpha^\star) = 0$ for $\alpha^\star = (\alpha_1^\star, \ldots, \alpha_{n_x}^\star)$. Since (19) is a universal approximator of continuous monotone functions on a compact domain Daniels & Velikova (2010), and each $\alpha_i^\star$ is monotone by assumption, for any $\varepsilon_{\text{approx}} > 0$ there exist L, M and $\theta_\vartheta^{(L,M)}$ with $\sup_i \sup_t |\alpha_i(t; \theta_{\vartheta,i}^{(L,M)}) - \alpha_i^\star(t)| \le \varepsilon_{\text{approx}}$ on the compact range of $\Sigma_{k+1,ii}$. Since f_{ml} is fixed and bounded on the support of P, $\mathcal{L}_P(\theta_\vartheta^{(L,M)}) \to \mathcal{L}_P(\alpha^\star) = 0$ as $\varepsilon_{\text{approx}} \to 0$, by continuity of $\mathcal{L}_P$ in α.

For the empirical minimizer $\hat{\theta}_\vartheta$, a uniform convergence bound over the regularized min-max class Anthony & Bartlett (1999) gives $\sup_{\theta_\vartheta} |\mathcal{L}_{\text{tune}}(\theta_\vartheta) - \mathcal{L}_P(\theta_\vartheta)| \le \varepsilon_{\text{gen}}/2$ with probability $\ge 1 - \delta$. Since $\hat{\theta}_\vartheta$ minimizes $\mathcal{L}_{\text{tune}}$ over the same class that contains $\theta_\vartheta^{(L,M)}$,

$$\begin{aligned}\mathcal{L}_P(\hat{\theta}_\vartheta) &\le \mathcal{L}_{\text{tune}}(\hat{\theta}_\vartheta) + \tfrac{\varepsilon_{\text{gen}}}{2} \le \mathcal{L}_{\text{tune}}(\theta_\vartheta^{(L,M)}) + \tfrac{\varepsilon_{\text{gen}}}{2} \\ &\le \mathcal{L}_P(\theta_\vartheta^{(L,M)}) + \varepsilon_{\text{gen}} \le \varepsilon_{\text{approx}}(L, M) + \varepsilon_{\text{gen}}.\end{aligned}$$

□

C.2 PROOF OF THEOREM 2

Proof. Since the `epinet` is trained with $\alpha = \mathbf{1}$ in (10), the linear ENN posterior gives $\hat{\mathbf{x}}_{k+1}|_{\alpha=\mathbf{1}} \sim \mathcal{N}(f_{\text{phy}} + f_{\text{ml}}, \Sigma_{k+1})$ Osband et al. (2023a), i.e. $\mathbf{x}_{k+1} = \hat{\mu}_{k+1}(\mathbf{1}) + H_{k+1}\epsilon$ with $H_{k+1}\epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma_{k+1})$ and H_{k+1} given by (6). By (14), $\hat{\mu}_{k+1}(\mathbf{1}) = \hat{\mu}_{k+1}(\alpha) + \Delta_{k+1}(\alpha)$, so substituting into the line above and rearranging, the residual at the *deployed* mean $\hat{\mu}_{k+1}(\alpha)$ decomposes as

$$\mathbf{x}_{k+1} - \hat{\mu}_{k+1}(\alpha) = \underbrace{H_{k+1}\epsilon}_{\sim \mathcal{N}(\mathbf{0}, \Sigma_{k+1})} + \Delta_{k+1}(\alpha).$$

Applying $\Sigma_{k+1}^{-1/2}$ and letting $\mathbf{d}_{k+1} = \Sigma_{k+1}^{-1/2}\Delta_{k+1}$, the standardized residual $\Sigma_{k+1}^{-1/2}(\mathbf{x}_{k+1} - \hat{\mu}_{k+1}) \sim \mathcal{N}(\mathbf{d}_{k+1}, \mathbf{I})$, so

$$(\mathbf{x}_{k+1} - \hat{\mu}_{k+1}(\alpha))^\top \Sigma_{k+1}^{-1} (\mathbf{x}_{k+1} - \hat{\mu}_{k+1}(\alpha)) \sim \chi^2_{n_x}(\lambda_{k+1}),$$

with $\lambda_{k+1} = \|\mathbf{d}_{k+1}\|_2^2 = \Delta_{k+1}^\top \Sigma_{k+1}^{-1} \Delta_{k+1}$. The coverage bound follows from the definition of the $(1-\delta)$ quantile of $\chi^2_{n_x}(\lambda_{k+1})$. For the uniform bound, (16) gives $\|\Sigma_{k+1}^{-1/2}\|_{\text{op}} \le 1/\sigma_{\min}$. This holds for *any* realized mechanism $\alpha \in [0,1]^{n_x}$, not only the extreme case $\alpha = \mathbf{0}$: since (8) applies α independently per output while $\Sigma_{k+1}^{-1/2}$ need not be diagonal, a mixed α is not bounded by either extreme in general, so

$$\lambda_{k+1} = \|\Sigma_{k+1}^{-1/2}\Delta_{k+1}\|_2^2 \le \|\Sigma_{k+1}^{-1/2}\|_{\text{op}}^2 \, \|\Delta_{k+1}\|_2^2 \le \frac{1}{\sigma_{\min}^2} \, \|(\mathbf{1} - \alpha) \odot f_{\text{ml}}\|_2^2 \le \frac{B_f^2}{\sigma_{\min}^2} = B^2,$$

using $\|(\mathbf{1} - \alpha) \odot f_{\text{ml}}\|_2 \le \|f_{\text{ml}}\|_2$ since each $(1 - \alpha_i) \in [0, 1]$; monotonicity of the quantile in λ gives the stated bound. □

Scope of these guarantees. Together, Theorems 1 and 2 leave a precise scope for what is, and is not, established about Σ_{k+1} itself. The collapse-avoidance property of the prior net (Section 3) explains why Σ_{k+1} carries a nonzero epistemic signal in the first place, not that its magnitude tracks the true prediction error. Theorem 1 establishes that the *mechanism* α converges given Σ_{k+1}, treating Σ_{k+1} as an input rather than validating it. Theorem 2 comes closest to validating Σ_{k+1} directly, but only conditional on the linear-ENN Gaussian-posterior assumption, not as an assumption-free, empirical check that Σ_{k+1} tracks the true error on our data.

D EXPERIMENTS

D.1 QUADROTOR

System configuration. We consider a 3D quadrotor in the `safe-control-gym` environment, controlled at $50\,\mathrm{Hz}$ ($\Delta t = 0.02\,\mathrm{s}$), with mass $m = 0.027\,\mathrm{kg}$, arm length $0.0397\,\mathrm{m}$, and principal moments of inertia $I_{xx} = I_{yy} = 1.4 \times 10^{-5}\,\mathrm{kg\,m^2}$, $I_{zz} = 2.17 \times 10^{-5}\,\mathrm{kg\,m^2}$. The environment exposes the full 12-dimensional state $\mathbf{x} = [x, \dot{x}, y, \dot{y}, z, \dot{z}, \phi, \theta, \psi, p, q, r]$ and injects an unknown aerodynamic drag force $\mathbf{F}_d$ that the nominal physics model does not capture; this is the disturbance f_{ml} is trained to recover. For the closed-loop noise sweep in Figure 3, we additionally inject Gaussian disturbance forces at the vehicle's center of mass, at two levels relative to its hover weight mg: low noise, standard deviation $0.025\,\mathrm{N}$ (9.4% of mg), and high noise, standard deviation $0.05\,\mathrm{N}$ (18.9% of mg), alongside a disturbance-free condition.

Physics model. In this case, we assume that we have a dynamic model (with unmodeled drag force)

$$\begin{aligned}
\dot{\mathbf{p}} &= \mathbf{v}, \\
m\dot{\mathbf{v}} &= R(\boldsymbol{\Theta})\, T\, \mathbf{e}_3 \;-\; mg\, \mathbf{e}_3, \\
\dot{\boldsymbol{\Theta}} &= W(\boldsymbol{\Theta})\, \boldsymbol{\omega}, \\
J\dot{\boldsymbol{\omega}} &= \boldsymbol{\tau} \;-\; \boldsymbol{\omega} \times J\boldsymbol{\omega},
\end{aligned}$$

where $\mathbf{p}, \mathbf{v} \in \mathbb{R}^3$ are the position and velocity in the world frame, $\boldsymbol{\Theta} = [\phi, \theta, \psi]^\top$ are the Euler angles, $\boldsymbol{\omega} \in \mathbb{R}^3$ are the body rates, $R(\boldsymbol{\Theta}) \in SO(3)$ is the body-to-world rotation matrix, $W(\boldsymbol{\Theta})$ is the kinematic map from body rates to Euler-angle rates, $\mathbf{e}_3 = [0, 0, 1]^\top$, $T = \sum_{i=1}^{4} f_i$ is the total rotor thrust, and $\boldsymbol{\tau}$ is the body torque produced by the four rotor thrusts f_i. We take this model, discretized with a 4th-order Runge-Kutta integrator, as f_{phy}. f_{ml} is trained to recover the unknown drag force from data.

Data collection. Training trajectories are collected by running a nominal linear MPC controller closed-loop in `safe-control-gym`, tracking a figure-8 reference trajectory, and logging each closed-loop rollout as one episode. Each episode requires the quadrotor to complete exactly 3 cycles in $15\,\mathrm{s}$, resulting in exactly 750 time steps.

Dataset. $\mathcal{D}_{\text{train}}$ consists of 15 episodes. $\mathcal{D}_{\text{tune}}$ consists of 5 further episodes, disjoint from $\mathcal{D}_{\text{train}}$. For system modeling, $\mathcal{D}_{\text{test}}$ consists of 3 episodes collected with a separate, fixed PID controller, not any of the five predictive models, on a figure-8 track scaled $1.75\times$ the training track, with all 12 states randomized at reset, so this test set is neither generated by, nor entangled with, any of the models being evaluated.

Training machine learning models. f_{ml}, the learnable epinet σ^L, and the mechanism $\alpha(\cdot)$ are each a 2-layer MLP with 16 hidden units, trained with Adam, weight decay 10^{-3}, learning rate 10^{-2}, batch size 2048, for 5000 epochs. The epistemic index has dimension $D_z = 16$; the prior net σ^P shares σ^L's architecture but is fixed at initialization, scaled by a factor of 1.5 relative to σ^L. $\mathcal{L}_{\text{epi}}$ is estimated with 4 epistemic-index samples per training step and a perturbation scale of 0.02.

System modeling metrics. Because $\mathcal{D}_{\text{test}}$ has only 3 episodes, we evaluate a single open-loop rollout per episode, starting from that episode's initial state and driven by its recorded control sequence, rather than a sliding window over many overlapping sub-trajectories, and report the mean position RMSE across the 3 episodes at each rollout step (Figure 2). RMSE is restricted to the position dimensions (x, y, z), since the MPC reward weights position $50\times$ heavier than velocity, attitude, or body rates, so position is what the controller actually optimizes for.

Control setup and metrics. Closed-loop episodes are planned with MPPI (Algorithm 2): horizon $T_H = 30$, population $N_{\text{pop}} = 256$, $J = 30$ iterations, elite count $K = 16$, temperature $\tau_{\text{mppi}} = 1.0$, and discount $\gamma_{\text{mppi}} = 0.99$. The risk weight c_{risk} is 0 for all baselines and tuned only for EPIC-PIML. Each noise level (disturbance-free, low, high) is evaluated over 100 closed-loop episodes,

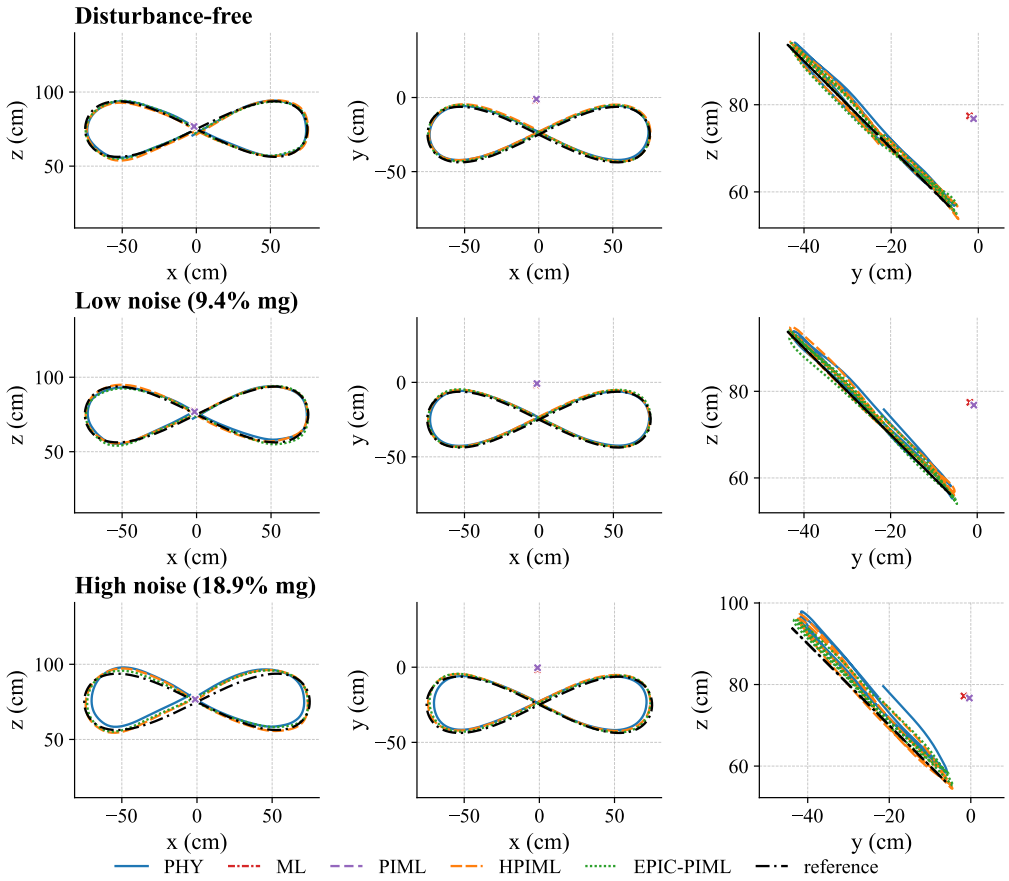


Figure 6: Mean trajectory of all five models over the 100 disturbance-free episodes, steps 250-500 (the second figure-8 cycle).

each starting from a random initial position drawn independently of $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{tune}}$, and $\mathcal{D}_{\text{test}}$; Figure 3 reports the resulting position RMSE and steps completed, where 750 steps means all 3 figure-8 cycles were completed. Figures 6 and 7 show, for all five models, the mean trajectory over these 100 episodes during the second and third figure-8 cycles, respectively.

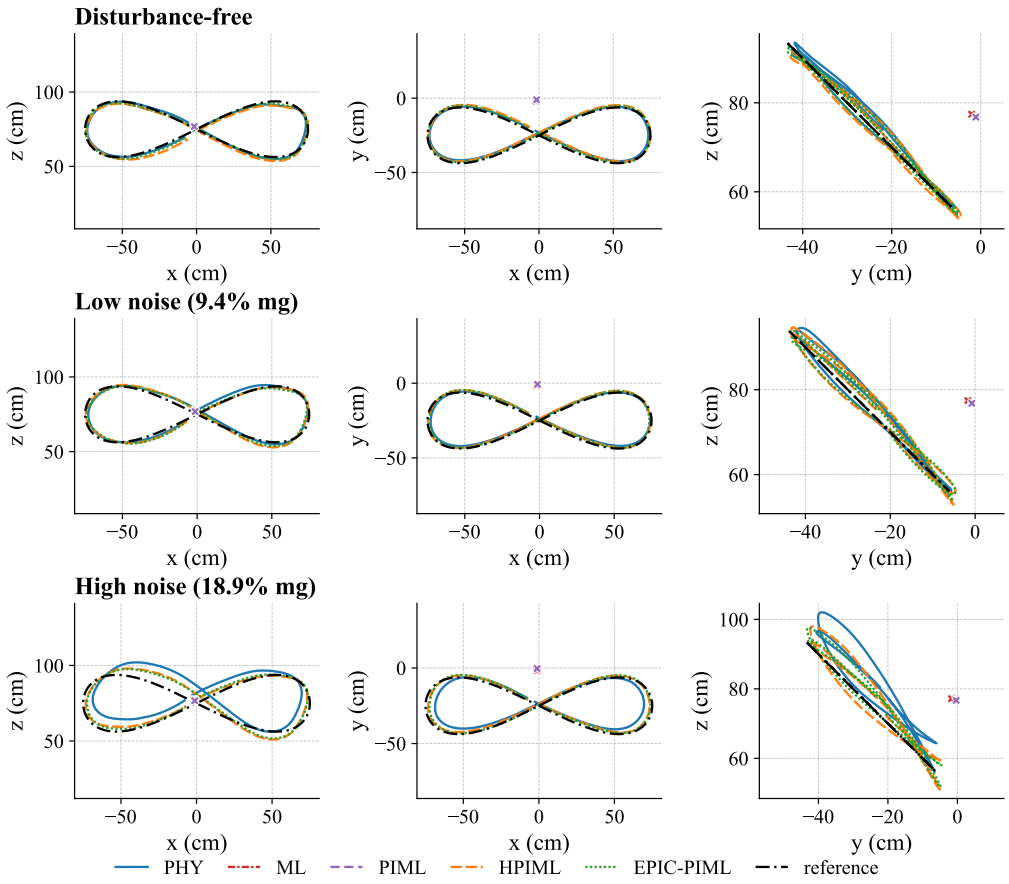


Figure 7: Mean trajectory of all five models over the 100 disturbance-free episodes, steps 500-750 (the third figure-8 cycle).

D.2 AUTONOMOUS RACING CAR

System configuration. We consider a 1/10-scale autonomous racing car from the F1TENTH-gym environment O'Kelly et al. (2020), controlled at $50\,\mathrm{Hz}$ ($\Delta t = 0.02\,\mathrm{s}$), with wheelbase $L = 0.330\,\mathrm{m}$, steering angle bounded to $[-0.419\,\mathrm{rad}, 0.419\,\mathrm{rad}]$, and steering rate bounded to $3.2\,\mathrm{rad\,s^{-1}}$. The state is $\mathbf{x} = [X, Y, \phi, v, \delta, \dot{\psi}, \beta] \in \mathbb{R}^7$, comprising position, heading, speed, steering angle, yaw rate, and side-slip angle, so $n_x = 7$. The control input is $\mathbf{u} = [\delta_{\text{cmd}}, v_{\text{cmd}}] \in \mathbb{R}^2$, the commanded steering angle and commanded speed setpoints, so $n_u = 2$. All experiments are run on the Oschersleben track ($260.7\,\mathrm{m}$ lap length, $2.2\,\mathrm{m}$ nominal width), the only track shipped with F1TENTH-gym's centerline and raceline data.

Physics model. Unlike the quadrotor's full rigid-body model, here we deliberately test a kinematic model: properly discretized, it matches a dynamic tire model's forecast accuracy at lower computational cost and without the tire model's low-speed singularity Kong et al. (2015), useful for racing's near-zero-speed corners and stop-and-go recovery. A kinematic model, however, has no notion of force or disturbance, so it necessarily leaves more of the true dynamics unmodeled than a dynamic model would. F1TENTH-gym also restricts control to an outer-loop setpoint $\mathbf{u} = [\delta_{\text{cmd}}, v_{\text{cmd}}]$, since an inner-loop controller, not direct wheel or steering torque, drives the real hardware. We fold this rate-limited steering and proportional speed actuator into f_{phy} alongside the kinematic bicycle update, for the first five state dimensions $[X, Y, \phi, v, \delta]$

$$
\begin{aligned}
X_{k+1} &= X_k + \Delta t \cdot v_k \cos\phi_k, \\
Y_{k+1} &= Y_k + \Delta t \cdot v_k \sin\phi_k, \\
\phi_{k+1} &= \phi_k + \Delta t \cdot v_k \tan(\delta_k)/L, \\
\delta_{k+1} &= \delta_k + \Delta t \cdot s_{\max} \cdot \operatorname{sign}(\delta_{\text{cmd}} - \delta_k), \\
v_{k+1} &= v_k + \Delta t \cdot k_p \cdot (v_{\text{cmd}} - v_k),
\end{aligned}
$$

where Δt is the sample time, $s_{\max}$ is the maximum steering rate, k_p is the speed-tracking gain, and L is the wheelbase. f_{phy} leaves the remaining two dimensions, yaw rate $\dot{\psi}$ and side-slip angle β, entirely unmodeled: the kinematic bicycle model has no notion of tire slip, so these pass through f_{phy} unchanged. f_{ml} is therefore trained to recover both the unmodeled tire-slip dynamics of $\dot{\psi}, \beta$ and any residual mismatch on the first five dimensions. k_p is not a single constant: it switches between a stronger gain when starting from rest and a weaker one once moving, and between accelerating and braking, mirroring F1TENTH-gym's own low-level speed controller exactly rather than an abstract proportional law. Concretely $s_{\max} = 3.2\,\mathrm{rad\,s^{-1}}$, and k_p is $10a_{\max}/v_{\max}$ or $10a_{\max}/|v_{\min}|$ once the car is moving (depending on whether v_{cmd} calls for accelerating or braking), or a fifth of those values from a standstill, with $a_{\max} = 9.51\,\mathrm{m\,s^{-2}}$, $v_{\max} = 20\,\mathrm{m\,s^{-1}}$, and $v_{\min} = -5\,\mathrm{m\,s^{-1}}$ the simulator's own actuator limits.

Data collection. Training and evaluation trajectories are collected by running a pure-pursuit path-tracking planner closed-loop in F1TENTH-gym around the track centerline, with each completed lap logged as one episode. The planner's excitation noise is seeded (seed $= 0$), so this dataset is exactly reproducible.

Dataset. $\mathcal{D}_{\text{train}}$ consists of 12 complete laps. $\mathcal{D}_{\text{tune}}$ consists of 4 further laps, disjoint from $\mathcal{D}_{\text{train}}$. $\mathcal{D}_{\text{test}}$ consists of 4 more laps, held out entirely from both training and tuning. Unlike the quadrotor's PID-collected, controller-independent test set, $\mathcal{D}_{\text{test}}$ here is collected by the same pure-pursuit planner as $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{tune}}$, just on held-out laps.

Training machine learning models. f_{ml}, the learnable epinet σ^L, and the mechanism $\alpha(\cdot)$ are each a 2-layer MLP with 16 hidden units, trained with Adam, weight decay 10^{-3}, learning rate 5×10^{-3}, batch size 256, for 5000 epochs. The epistemic index has dimension $D_z = 16$; the prior net σ^P shares σ^L's architecture but is fixed at initialization, scaled by a factor of 1.5 relative to σ^L. $\mathcal{L}_{\text{epi}}$ is estimated with 4 epistemic-index samples per training step and a perturbation scale of 0.02. PIML's physics-consistency loss term is weighted $\lambda_{\text{phys}} = 0.01$ here, compared to $\lambda_{\text{phys}} = 1.0$ for the quadrotor, reflecting the F1TENTH physics model's much larger unmodeled residual: no tire slip at all, versus only unmodeled drag for the quadrotor.

System modeling metrics. Because $\mathcal{D}_{\text{test}}$ has only 4 laps, we evaluate a single open-loop rollout per lap, starting from that lap's initial state and driven by its recorded control sequence, and report the mean position RMSE across the 4 laps at each rollout step and at 1, 10, and 20 steps (Figure 4). RMSE is restricted to the position dimensions (X, Y), the quantity that determines whether the car stays on track and completes a lap; the remaining state dimensions (heading, speed, steering angle, yaw rate, side-slip) are not the primary quantity of interest for this metric.

Control setup and metrics. Closed-loop laps are planned with a CEM-style MPPI variant: horizon $T_H = 20$, population $N_{\text{pop}} = 16384$, $J = 50$ iterations, elite count $K = 256$, softmax temperature $\tau_{\text{mppi}} = 1.0$, and temporally-correlated sampling noise (AR(1), coefficient 0.9) with initial standard deviation $[0.15\,\text{rad}, 2.0\,\text{m}\,\text{s}^{-1}]$ on $[\delta_{\text{cmd}}, v_{\text{cmd}}]$. Unlike the quadrotor's gated MPPI, this controller has no explicit risk-averse cost term: swapping in EPIC-PIML changes only which dynamics model predicts the rollouts used for planning, not the cost function itself. The per-step cost weights position error 10, the gap between the commanded and reference speed 50, steering-command change 20, speed-command change 1, and a soft penalty (weight 100) for crossing the track boundary; the terminal cost weights final position error $5\times$ the per-step position weight. Each (model, speed) pair is evaluated as one 10-lap run on the Oschersleben track, always starting from the same start line, unlike the quadrotor's 100 randomized-initial-condition episodes per noise level, since a fixed racing line has no natural analogue of a randomized starting pose. Figure 5 reports the resulting lateral offset and total time to complete all 10 laps at reference speeds $v \in \{4, 5, 6\}\,\text{m}\,\text{s}^{-1}$, and Figures 8–10 show the corresponding lap-1 trajectories and lateral offsets for all five models at each of those speeds. The main text reports the same trajectory/offset breakdown at $v = 7\,\text{m}\,\text{s}^{-1}$.

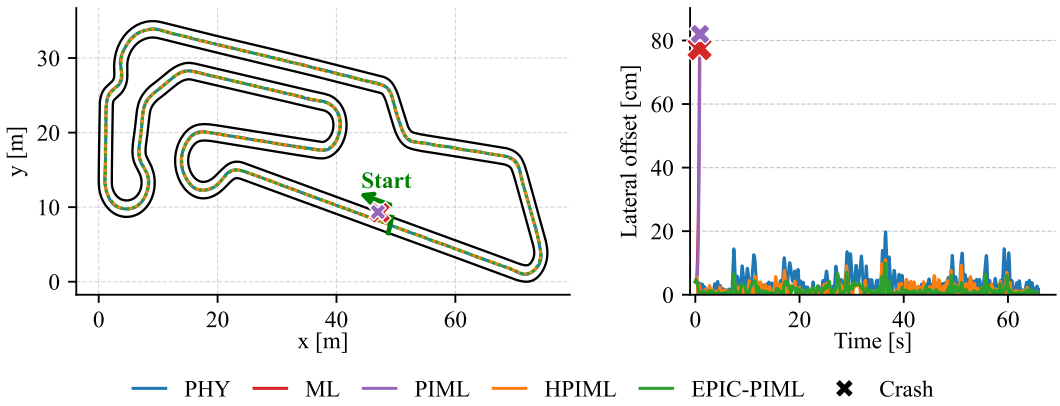


Figure 8: Lap-1 trajectory (left) and lateral offset from the reference path over time (right) for all five models, at constant reference speed $4\,\mathrm{m\,s^{-1}}$ on the Oschersleben track. Crosses mark where a model crashes before completing the lap.

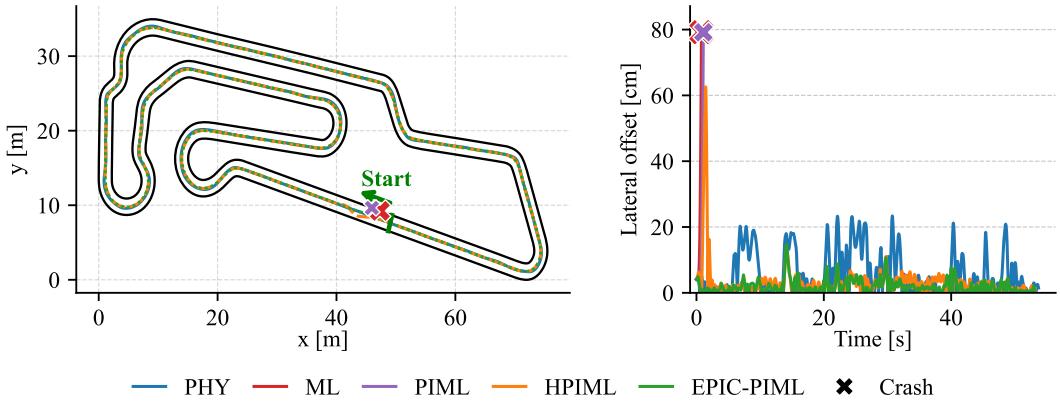


Figure 9: Lap-1 trajectory (left) and lateral offset from the reference path over time (right) for all five models, at constant reference speed $5\,\mathrm{m\,s^{-1}}$ on the Oschersleben track. Crosses mark where a model crashes before completing the lap.

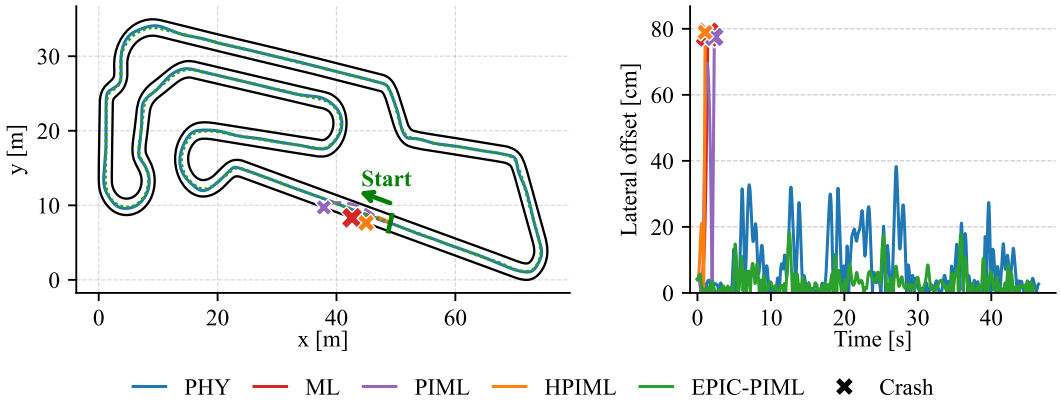


Figure 10: Lap-1 trajectory (left) and lateral offset from the reference path over time (right) for all five models, at constant reference speed $6\,\mathrm{m\,s^{-1}}$ on the Oschersleben track. Crosses mark where a model crashes before completing the lap.