

Distributionally robust minimization in meta-learning for system identification

Matteo Rufolo, Dario Piga, Marco Forgone

Abstract—Meta learning aims at learning how to solve tasks, and thus it allows to estimate models that can be quickly adapted to new scenarios. This work explores distributionally robust minimization in meta learning for system identification. Standard meta learning approaches optimize the expected loss, overlooking task variability. We use an alternative approach, adopting a distributionally robust optimization paradigm that prioritizes high-loss tasks, enhancing performance in worst-case scenarios. Evaluated on a meta model trained on a class of synthetic dynamical systems and tested in both in-distribution and out-of-distribution settings, the proposed approach allows to reduce failures in safety-critical applications.

I. INTRODUCTION

System Identification (SYSID) aims to develop mathematical models of dynamical systems based on observed input-output trajectories. Classic SYSID models each system independently, using only its own input-output data and ignoring knowledge from similar systems. This often leads to suboptimal results when data is scarce. A promising alternative is to integrate *meta-learning*, a framework introduced in the 1980s [1] and recently revitalized for its ability to enable rapid adaptation across related tasks. By training a *meta-learner* on a distribution of similar systems, the model can generalize efficiently to unseen dynamics with minimal additional data [2]. This approach offers a paradigm shift in SYSID, enabling more data-efficient and generalizable modeling.

Within the SYSID framework, a meta-learning approach treats the identification of different dynamical systems as a series of related tasks, with the aim of generalizing across multiple systems and efficiently adapt to newly encountered ones [3]. One particular meta-learning paradigm used in SYSID is in-context learning, see [4]. In this approach, a large model is pre-trained on a broad set of related datasets, allowing it to process a *context* sequence consisting of input-output data from a system, along with a *query* input sequence, and subsequently predict the corresponding query output without requiring task-specific re-training. Pre-training is performed offline in a supervised setting over *synthetic datasets*, derived from random input signals applied to a diverse set of randomly sampled systems within a given class. In contrast, inference follows a zero-shot learning paradigm, wherein the pre-trained model directly provides predictions for a new system without requiring additional fine-tuning. Unlike traditional SYSID methods, this approach trains a meta-model to generalize across related systems,

removing the need for system-specific identification. This enables deployment without requiring prior knowledge of the underlying dynamics. The meta model functions as a *learned* SYSID algorithm that identifies system-specific models from the context and utilizes this knowledge to generate output predictions for the query input.

An important characteristic of existing meta-learning approaches is their focus on minimizing the expected value of the loss over a collection of tasks. This approach implicitly assumes that all tasks are equally important, disregarding differences in task complexity or loss magnitude. While effective in many cases, this uniform weighting can be problematic in safety-critical applications.

Example 1: Consider an autonomous vehicle trained to follow urban trajectories using data from diverse weather conditions. Standard training assigns equal weight to all scenarios, following the expected risk minimization principle with a uniform task weighting strategy. While effective for common conditions, it may lead to poor performance in rare but critical cases, such as snowstorms, where obscured lane markings and reduced traction pose significant challenges.

To address this issue, recent research has proposed alternative strategies that enhance the robustness of meta-learning frameworks. Some approaches incorporate adversarial learning techniques [5]. Others modify the task distribution by prioritizing tasks with higher loss during training, thereby focusing on the most challenging and potentially *problematic* instances [6], [7].

In this paper, we adapt the distributionally robust meta-learning formulation of [7], based on optimization of a tail loss [8], to the in-context SYSID framework. This formulation aligns well with meta-learning, where task variability and distributional shifts demand robustness for reliable generalization. This adaptation relies on the premise that tasks vary in difficulty, with more complex ones offering greater training informative content. Our goal is to exploit these tasks more effectively, following an active learning-inspired strategy [9]. Specifically, after mathematically formulating the robust meta-learning problem for SYSID, we employ a heuristic, adapted from [10], to solve the distributionally robust optimization problem. We then analyze its convergence properties, leveraging theoretical insights from the original formulation, but applying them to our in-context framework and trying to provide a more practical interpretation. With respect to [7], we refine the convergence assumptions to suit the in-context SYSID setting considered in this paper, providing a practical interpretation of these assumptions in the given context.

The rest of the paper is structured as follows. Section II introduces the problem setting, presenting the probabilistic meta-modeling framework from [11] and discussing the limitations of the conventional meta-learning minimization formulation. Section III focuses on distributionally robust minimization for meta-learning in SYSID. This includes a heuristic for robust optimization, along with new theoretical results analyzing the convergence properties of the proposed approach. Section IV presents numerical examples that highlight the improved robustness of the proposed approach compared to the standard method, in both in-distribution (ID) and out-of-distribution (OOD) scenarios. To provide a more comprehensive assessment, an additional analysis is included at the end of Section IV, investigating the role of probabilistic estimation within the distributionally robust framework. Specifically, the proposed method is compared to a non-probabilistic variant trained using the Root Mean Squared Error (RMSE) loss, following [4], which does not incorporate uncertainty information.

II. IN-CONTEXT SYSTEM IDENTIFICATION

In this paper, following the footsteps of [4], [11], an in-context learner \mathcal{M}_ϕ referred to as a *meta model* is employed for probabilistic modeling of a *class* of dynamical systems, taking advantage of the in-context capabilities of Transformer architectures [12]. The meta model \mathcal{M}_ϕ is trained to provide probabilistic output estimates over a prediction horizon, on a potentially *infinite stream* of datasets $\{D^{(i)} = (u_{1:n}^{(i)}, y_{1:n}^{(i)}), i = 1, 2, \dots, \infty\}$, realizations of a random variable with distribution $p(\mathcal{D})$. Each dataset $D^{(i)}$ is obtained by applying a random input sequence $u_{1:n}^{(i)}$, with $u_k^{(i)} \in \mathbb{R}^{n_u}$, to a dynamical system $S^{(i)}$ randomly sampled within the class, resulting in the corresponding output sequence $y_{1:n}^{(i)}$, with $y_k^{(i)} \in \mathbb{R}^{n_y}$.¹ Formally, the datasets $D^{(i)}$ can be seen as independent realizations of a random variable with distribution $p(D)$, induced by the random generation of systems and inputs, whose analytical form is unknown. Since all datasets are drawn from the same distribution $p(D)$, assumed to be well-behaved², it is intuitive that their shared structure enables few-shot adaptation, allowing knowledge to be transferred effectively across different realizations.

Following the notation of [11], which extends [4] and frames in-context SYSID in a probabilistic setting, each system $S^{(i)}$ generates two input-output sequences. The first sequence, referred to as the *context* $(u_{1:m}^{(i)}, y_{1:m}^{(i)})$, implicitly informs the meta model about the system's dynamics. The second sequence, denoted as the *query* $(\tilde{u}_{1:n}^{(i)}, \tilde{y}_{1:n}^{(i)})$, represents the data to be predicted. We refer to *task* as a combination of the two previously defined sequences, namely: $\{D^{(i)} = ((u_{1:m}^{(i)}, y_{1:m}^{(i)}), (\tilde{u}_{1:n}^{(i)}, \tilde{y}_{1:n}^{(i)})), i = 1, 2, \dots, \infty\}$.

Throughout the paper, we use the encoder-decoder Transformer architecture proposed in [11] as meta model \mathcal{M}_ϕ .

¹In the following, for notational convenience, we consider the case $n_y = 1$. The extension to the multi-output case $n_y > 1$ is trivial.

²The specific conditions that $p(\mathcal{D})$ must satisfy in order to ensure theoretical guarantees are stated in Proposition 3.2.

The meta model processes three components: the input-output context sequence $(u_{1:m}, y_{1:m})$, the initial portion of the input-output query sequence $(\tilde{u}_{1:c}, \tilde{y}_{1:c})$, used as initial conditions, and the remaining input sequence $\tilde{u}_{c+1:n}$. It then outputs the estimated mean $\mu_{c+1:n}$ and standard deviation $\sigma_{c+1:n}$ of the predicted query output distribution:

$$\mu_{c+1:n}, \sigma_{c+1:n} = \mathcal{M}_\phi(\tilde{u}_{c+1:n}, \tilde{u}_{1:c}, \tilde{y}_{1:c}, u_{1:m}, y_{1:m}). \quad (1)$$

To simplify the notation, we define $X = (\tilde{u}_{c+1:n}, \tilde{u}_{1:c}, \tilde{y}_{1:c}, u_{1:m}, y_{1:m})$ as the set of all input variables processed by the meta model \mathcal{M}_ϕ , and $\theta = (\mu_{c+1:n}, \sigma_{c+1:n})$ as the output parameters returned by \mathcal{M}_ϕ . We have then: $\theta = \mathcal{M}_\phi(X)$.

Both the standard meta learning loss used in [11] and the distributionally robust one introduced in this paper are defined in terms of the Kullback-Leibler (KL) divergence between the unknown query output density $p(\tilde{y}_{c+1:n}|X)$ and the predicted query output distribution. The latter is parameterized as a Gaussian $q_{\theta(\phi, X)}(\tilde{y}_{c+1:n})$, with mean and standard deviation returned by \mathcal{M}_ϕ . Thus, the risk function is defined as:

$$\begin{aligned} \ell(\tilde{y}_{c+1:n}, X; \phi) &= \log \frac{p(\tilde{y}_{c+1:n}|X)}{q_{\theta(\phi, X)}(\tilde{y}_{c+1:n})} \\ &= -\log q_{\theta(\phi, X)}(\tilde{y}_{c+1:n}) + K \\ &= \frac{1}{2} \log(2\pi\sigma_{c+1:n}) + \frac{1}{2} \left(\frac{\tilde{y}_{c+1:n} - \mu_{c+1:n}}{\sigma_{c+1:n}} \right)^2 + K, \end{aligned} \quad (2)$$

where K is a constant independent of ϕ .

A. Standard learning: expected KL loss minimization

In line with standard meta-learning practice, \mathcal{M}_ϕ was trained in [11] by minimizing the *expected value* $J(\phi)$ of the risk function (2) over the dataset distribution $p(D)$, i.e.,

$$\phi^* = \arg \min_{\phi} \underbrace{\mathbb{E}_{p(D)} [-\log q_{\theta(\phi, X)}(\tilde{y}_{c+1:n})]}_{=J(\phi)}, \quad (3)$$

which is the expected KL divergence between $p(\tilde{y}_{c+1:n}|X)$ and $q_{\theta(\phi, X)}(\tilde{y}_{c+1:n})$. In practice, since the functional form of $p(D)$ is unknown and, in any case, evaluating the expectation above is generally intractable, the Monte Carlo average:

$$\tilde{J}(\phi) = \frac{1}{b} \sum_{i=1}^b -\log q_{\theta(\phi, X^{(i)})}(\tilde{y}_{c+1:n}^{(i)}) \quad (4)$$

computed over b datasets sampled from $p(D)$ was used to approximate $J(\phi)$. The loss $\tilde{J}(\phi)$ was then minimized using stochastic minibatch gradient descent, where b datasets are resampled from $p(D)$ at each iteration, with b serving as the minibatch size in the optimization process. However, the uniform task weighting corresponding to the expected KL risk minimization (3) can be problematic in applications where users not only aim to optimize the average performance but also wish to mitigate failures in the worst-case scenarios [8].

III. ROBUST LEARNING: TAIL KL LOSS MINIMIZATION

In order to avoid these failures, in this work we analyze the meta-learning framework from the perspective of the task

distribution $p(D)$, emphasizing the importance of tasks with worse performance. To this aim, we modify the optimization problem (3) by minimizing the expected *tail loss* of the task distribution, defined as the expectation over tasks with higher risk values. This approach aims to mitigate performance disparities across tasks and enhance robustness against the most challenging instances. In order to minimize the expected tail loss over the task distribution, we focus on dataset realizations whose associated risk function (2) exceeds a predefined threshold. To formally define this objective, we begin by introducing the necessary mathematical framework.

We denote the support of $p(D)$ by \mathbb{D} . From (2), for a given parameter vector ϕ , we define the meta-learning operator \mathcal{L}_ϕ that maps each dataset realization to its respective risk:

$$\mathcal{L}_\phi : \mathbb{D} \rightarrow \mathbb{R}; \quad D^{(i)} \mapsto \ell(\tilde{y}_{c+1:n}^{(i)}, X^{(i)}; \phi). \quad (5)$$

Note that \mathcal{L}_ϕ is a deterministic operator. Since the sequences $\tilde{y}_{c+1:n}^{(i)}$ and $X^{(i)}$ uniquely define the dataset realization $D^{(i)}$, in the rest of the paper the dependence of ℓ on $\tilde{y}_{c+1:n}^{(i)}$ and $X^{(i)}$ will be written in the compact form: $\ell(\tilde{y}_{c+1:n}^{(i)}, X^{(i)}; \phi) = \ell(D^{(i)}; \phi)$.

Using the operator (5), the risk function can be viewed as a random variable with the corresponding *probability density function* (PDF) $p_\ell(l)$ and cumulative distribution function (CDF) $F_\ell(l) = P\{\ell(D; \phi) \leq l, D \in \mathbb{D}, l \in \mathbb{R}\}$. For simplicity, we abbreviate $\ell(D; \phi)$ as ℓ .

From the definition of the CDF $F_\ell(l)$, we define a quantile known as the Value-at-Risk (VaR) [8] for the random variable ℓ , at any probability level $\alpha \in (0, 1]$ (confidence interval), given the density $p(D)$ and the meta model parameters ϕ :

$$VaR_\alpha[\mathcal{L}_\phi] = \inf_{l \in \mathbb{R}} \{l \mid F_\ell(l) \geq \alpha\}. \quad (6)$$

Let us focus on the domain of the random variable ℓ with $\ell(D; \phi) \geq VaR_\alpha[\mathcal{L}_\phi]$, namely: $\mathbb{D}_\alpha := \{D \in \mathbb{D} \mid \ell(D; \phi) \geq VaR_\alpha[\mathcal{L}_\phi]\}$. We can then define the conditional expectation over \mathbb{D}_α , known as the Conditional Value-at-Risk (CVaR) [8], as:

$$CVaR_\alpha[\mathcal{L}_\phi] = (1 - \alpha)^{-1} \int_{\mathbb{D}_\alpha} \ell(D; \phi) p(D) dD. \quad (7)$$

The primary objective of the robust learning problem addressed in this work is to minimize $CVaR_\alpha[\mathcal{L}_\phi]$ in (7), which focuses on the tail of the risk function distribution, i.e., dataset realizations $D^{(i)}$ where $\ell(D^{(i)}; \phi)$ exceeds $VaR_\alpha[\mathcal{L}_\phi]$. To this aim, we introduce a new probability distribution $p_\alpha(D; \phi)$ over the restricted dataset support \mathbb{D}_α :

$$p_\alpha(D; \phi) := \frac{p(D)}{\mathcal{Z}_\alpha(\phi)}, \quad \text{with } D \in \mathbb{D}_\alpha, \quad (8)$$

where $\mathcal{Z}_\alpha(\phi)$ is the normalization factor over the updated support, i.e. $\mathcal{Z}_\alpha(\phi) = \int_{\mathbb{D}_\alpha} p(D) dD$. The probability distribution $p_\alpha(D; \phi)$ in (8) thus characterizes dataset realizations whose risk function ℓ is larger than the $VaR_\alpha[\mathcal{L}_\phi]$ in (6).

By substituting the definition of the risk function (2) into (7) and from the definition of $p_\alpha(D; \phi)$ in (8), minimization of the $CVaR_\alpha[\mathcal{L}_\phi]$ is equivalent to solving the

optimization problem:

$$\phi^* = \arg \min_{\phi} \underbrace{\mathbb{E}_{p_\alpha(D; \phi)} [-\log q_{\theta(\phi, X)}(\tilde{y}_{c+1:n})]}_{=J_\alpha(\phi)}. \quad (9)$$

However, since neither $p(D)$ nor $p_\alpha(D; \phi)$ is known in closed form, the expectation over $p_\alpha(D; \phi)$ in (9) cannot be computed analytically. Moreover, as there is no straightforward way to sample from $p_\alpha(D; \phi)$, directly approximating the expectation using a Monte Carlo average is not feasible. To address this challenge, inspired by [7], we adopt the quantile estimation strategy described in Algorithm 1. The main idea of the quantile estimation strategy is to sample from $p(D)$ a batch of b tasks (step 2), and then to approximate the loss J_α by only considering the $\lfloor (1 - \alpha)b \rfloor$ tasks with the highest risk (steps 4-5), where $\lfloor \cdot \rfloor$ denotes the floor operator. This subset of tasks, denoted by \mathcal{B}_α , is used to define the loss approximation \tilde{J}_α . The gradient of \tilde{J}_α is then employed to update the meta-model parameters ϕ via standard gradient descent (steps 6-7). Properties of the quantile estimation

Algorithm 1 Gradient-based robust Optimization with Quantile Estimation

Require: Initial meta-model parameters $\phi^{(0)}$, batch size b , confidence interval $\alpha \in (0, 1]$, maximum iterations max_iter , learning rate $\eta > 0$

Ensure: Optimized meta-model parameters $\hat{\phi}$

- 1: **for** $j = 1$ to max_iter **do**
 - 2: Sample a batch $\mathcal{B} = \{D^{(i)}\}_{i=1}^b, D^{(i)} \sim p(D)$;
 - 3: Compute the risk function $l_i = \ell(D^{(i)}; \phi^{(j)})$ for each $D^{(i)} \in \mathcal{B}$;
 - 4: Select the top $\lfloor (1 - \alpha)b \rfloor$ datasets with the highest risk, forming the subset \mathcal{B}_α ;
 - 5: Approximate the loss J_α using the subset \mathcal{B}_α as \tilde{J}_α ;
 - 6: Compute the gradient $\nabla_\phi \tilde{J}_\alpha$ over the batch \mathcal{B}_α ;
 - 7: Update parameters: $\phi^{(j+1)} \leftarrow \phi^{(j)} - \eta \nabla_\phi \tilde{J}_\alpha$.
 - 8: **end for**
 - 9: **return** Optimized parameters $\hat{\phi}$
-

strategy in Algorithm 1 are reported in the following Lemma from [7]. Before stating the lemma, we define the Monte Carlo approximation of the $VaR_\alpha[\mathcal{L}_\phi]$:

$$\tilde{V}[\mathcal{L}_\phi] := \min_{i=1, \dots, \lfloor (1-\alpha)b \rfloor} \{\ell(D^{(i)}; \phi) \mid D^{(i)} \in \mathcal{B}_\alpha\}. \quad (10)$$

Lemma 3.1: If the following assumptions are satisfied:

- (I) the risk function $\ell(D; \phi)$ is Lipschitz continuous w.r.t. ϕ ,
- (II) the CDF $F_\ell(l)$ is Lipschitz continuous w.r.t. l with Lipschitz constant β_ℓ ,
- (III) the Monte Carlo $VaR_\alpha[\mathcal{L}_\phi]$ approximation error is bounded, i.e. $|\tilde{V}[\mathcal{L}_\phi] - VaR_\alpha[\mathcal{L}_\phi]| \leq \frac{\eta}{\beta_\ell(1-\alpha)^2}$ at every iteration of Algorithm 1,
- (IV) the PDF $p_\alpha(D; \phi)$ is Lipschitz continuous w.r.t. ϕ ,
- (V) the risk function $\ell(D; \phi)$ over the dataset realizations drawn from $p_\alpha(D; \phi)$ is bounded for any ϕ , s.t. $\|\phi\| \leq C < +\infty^2$, with $C \in \mathbb{R}$.

Then Algorithm 1 converges to a local optimum of the robust minimization problem (9), with the improvement guarantee $J_\alpha(\phi^{(j+1)}) \leq J_\alpha(\phi^{(j)})$ \square

Among the assumptions in Lemma 3.1, the strongest are (V) and (III). Assumption (V) is relatively mild, especially given the Lipschitz condition (I). In contrast, Assumption (III) is the most restrictive, and it is a necessary condition as it is possible from the proof in the Appendix H of [7]. Unlike standard Monte Carlo estimation, which typically requires only bounded variance, this condition enforces a direct bound on the estimation error at each iteration. While stronger, it remains reasonable in our context. In particular, the bound becomes looser (i.e., more permissive) for larger α , where the tail quantiles are harder to approximate. For small α , the bound is tighter, but the approximation is easier due to the statistical efficiency of Monte Carlo estimation in this regime. In the following we adapt the assumptions of Lemma 3.1 to make them suited for the Transformer architecture used in [11] (and also adopted in this work) to describe the meta model \mathcal{M}_ϕ .

Proposition 3.2: Suppose that, in addition to Assumptions (I), (III), and (IV) of Lemma 3.1, the following hold:

- (1) there exists $M \in \mathbb{R}$ such that $\sup_{D \in \mathbb{D}} p(D) \leq M$,
- (2) the measure of $\{D \in \mathbb{D} \mid \ell(D; \phi) = L\}$ is bounded for all $L \in \mathbb{R}$
- (3) the standard deviation in the parametrized distribution $q_{\theta(\phi, X)}$ in (9) is nonzero,
- (4) the activation functions used in the Transformer describing the meta model are Lipschitz continuous with respect to the processed input for any ϕ , s.t. $\|\phi\| \leq C < +\infty$, with $C \in \mathbb{R}$,
- (5) every input processed by Layer normalization in the Transformer has nonzero standard deviation for any ϕ s.t. $\|\phi\| \leq C < +\infty$, with $C \in \mathbb{R}$.

Then the statement of Lemma 3.1 holds. \square

Proof: [Proof of Proposition 3.2]

Since Assumptions (I), (III), and (IV) are already shared with Lemma 3.1, it only remains to show that Assumptions (1)–(5) imply Assumptions (II) and (V) of the Lemma.

We now prove that Assumption (II) in Lemma 3.1 holds under the assumptions of Proposition 3.2. Given $L_1, L_2 \in \mathbb{R}^+$, assuming without loss of generality $L_2 \geq L_1$, we have:

$$|F_\ell(L_2; \phi) - F_\ell(L_1; \phi)| \leq \sup_{l \in \mathbb{R}} p_\ell(l)(L_2 - L_1).$$

We need to prove that $\sup_{L \in \mathbb{R}} p_\ell(L)$ is bounded. Since the map \mathcal{L}_ϕ in (5) is deterministic, the only source of randomness in ℓ is the random variable D . Thus, we obtain:

$$\begin{aligned} p_\ell(L) &= \int_{\mathbb{D}} p(D)p(L|D) dD = \int_{\mathbb{D}} p(D)\delta(L - \ell(D, \phi)) dD \\ &\leq M \int_{\mathbb{D}} \delta(L - \ell(D, \phi)) dD \end{aligned} \quad (11)$$

where ϕ is fixed from the definition in (5) and the last inequality follows from Assumption (1). From inequality (11)

³The boundedness restriction on the parameter ϕ was not explicitly stated in [7], but it is actually needed for the Assumption (V).

and Assumption (2), $p_\ell(L)$ is bounded for all $L \in \mathbb{R}$, thus $\sup_{L \in \mathbb{R}} p_\ell(L)$ is bounded and $F_\ell(l)$ is Lipschitz continuous w.r.t. ℓ . Therefore, Assumption (II) of Lemma 3.1 is satisfied.

We now prove that, under the assumptions of Proposition 3.2, Assumption (V) is satisfied, which states that the risk function $\ell(D; \phi)$ evaluated over realizations of $p_\alpha(D; \phi)$ is bounded for any ϕ s.t. $\|\phi\| < +\infty$. From the definition of $\ell(D; \phi)$ in (2), Assumption (V) is satisfied if:

$$0 < \sigma_\phi^2(X) < +\infty, \quad |\tilde{y} - \mu_\phi(X)| < +\infty. \quad (12)$$

Using Assumption (3) and since the true output \tilde{y} is bounded, proving that (12) holds only requires that $\mu_\phi(X)$ and $\sigma_\phi(X)$ are bounded for any X . For this, it is sufficient to show that the Transformer architecture is Lipschitz continuous in X . Indeed, if this condition holds, then, since all inputs are bounded, the corresponding output parameters remain bounded for any fixed ϕ s.t. $\|\phi\| < +\infty$. Since the inputs lie within a bounded domain, it is sufficient to verify local Lipschitz continuity w.r.t. X , which holds provided that all Transformer layers satisfy this property. In fact:

- linear layers are inherently Lipschitz continuous;
- positional encodings are trivially Lipschitz continuous;
- RNNs and multilayer perceptrons (MLPs) are Lipschitz continuous, as established in [13];
- the attention mechanism has a local Lipschitz bound proportional to the square root of its input length [14];
- layer normalization is Lipschitz continuous under Assumption (5), as shown in [13].

Using also Assumption (4), the Transformer is Lipschitz continuous w.r.t. its inputs, ensuring that the outputs $\mu_\phi(X)$ and $\sigma_\phi(X)$ remain bounded for any ϕ s.t. $\|\phi\| \leq C < +\infty$.

Summarizing, we have proven that Assumptions in Proposition 3.2 imply Assumptions in Lemma 3.1. Therefore, the statement of Lemma 3.1 (and of Proposition 3.2) holds under the assumptions of Proposition 3.2. \blacksquare

Proposition 3.2 aims to restate the assumptions of Lemma 3.1 in a form more interpretable within a probabilistic in-context learning framework, specifically using a Transformer-based meta-model \mathcal{M}_ϕ . While its assumptions are stricter than those in Lemma 3.1, they offer clearer practical implications. Notably, several of the revised assumptions are readily satisfied in practice. For example, Assumption (3) holds when the predicted standard, provided as output by the meta model, deviation is constrained to be positive via an exponential function [11], and Assumption (4) is met by common deep learning function activations. This refinement enhances practical relevance without weakening the theoretical foundation.

IV. NUMERICAL EXAMPLES

The proposed tail loss optimization strategy is used to train a meta model using synthetic data from a class of dynamical systems with a Wiener-Hammerstein (WH) structure. The meta model adopts the Transformer architecture from [11], comprising approximately 5.5 million parameters. Key hyperparameters include an initial condition length $c = 30$, query length $n - c = 100$, and context length

$m = 400$. All experiments were repeated across different random initializations and dataset realizations, consistently yielding similar results. Training was conducted on an Nvidia RTX 3090 GPU. To ensure reproducibility, the complete codebase is available at <https://github.com/mattrufolo/sysid-robust-transformer>, which also contains an extended version of the paper with additional analyses and results.

A. System Class and Task Distribution

The WH systems used for meta-model training follow a cascade structure G_1-F-G_2 , where G_1 and G_2 are discrete-time Linear Time-Invariant (LTI) systems and F is a static nonlinearity. The LTI blocks have randomly generated orders uniformly sampled from $[1, 10]$, with pole magnitudes in $(0.5, 0.97)$ and phases in $(0, \pi/2)$. The nonlinearity F is modeled as a feedforward NN with one hidden layer of 32 units and hyperbolic tangent activations, with weights sampled from a normal distribution with Kaiming scaling.

Inputs consist of white noise signals drawn from standard Gaussian, and outputs are standardized to zero mean and unit variance. The output are then corrupted with measurement noise, modeled by white Gaussian noise with standard deviation $\sigma_{\text{noise}} = 0.01$.

B. Learning Procedure

For the initial phase of training, the loss function in (4) is minimized using the AdamW optimizer over 100,000 iterations, with a batch size $b = 32$. All hyperparameters follow standard machine learning practice; more details are provided in the accompanying GitHub repository. From this checkpoint, two distinct training procedures are resumed for additional 700,000 iterations: *standard* training (solving (3)) and proposed *robust* training (solving (9)). For standard training, a batch of $b = 32$ tasks is sampled at each iteration. In contrast, for robust training, a batch of $b = 80$ tasks is generated and only the 32 tasks ($\alpha = 0.4$) with the highest loss values are selected for training. Notably, this strategy incurs a longer training time, approximately 29 hours compared to 17.3 hours for standard training. This increase stems mainly from: (i) additional forward passes to evaluate all tasks in each batch before backpropagation, and (ii) the higher cost of dataset generation, performed on the CPU, which accounts for about 36% of the time difference (roughly 4.2 out of 11.7 hours). All the results presented in the rest of the section are obtained in a zero-shot fashion, without any model re-training.

We remark that in [7], training starts directly with the robust procedure. However, we found a two-stage training procedure more efficient. In fact, roughly speaking, only after some iterations the trained meta model can discriminate tasks with high and low loss. To enable a direct comparison with a one-stage approach, we also perform training from scratch using the robust procedure. The one-stage and two-stage approaches achieve comparable predictive performance. However, the two-stage approach is 20% faster than training from scratch (32 vs 40 hours).

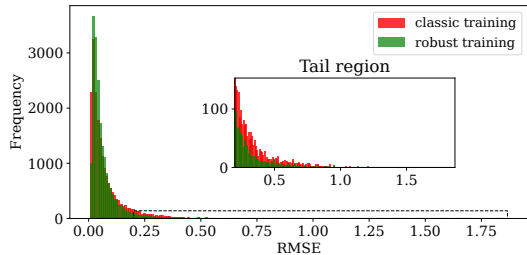


Fig. 1: In-distribution analysis: RMSE histogram over the test batch obtained with standard and robust training.

C. In-Distribution Analysis

The ID performance of standard and robust training are evaluated on a test batch of $b = 20000$ tasks sampled from the same task distribution used during training. Table I reports the average root mean square error for the entire test batch and for the 8000 tasks with the highest RMSE values (approximately 40%).

Results show that the robust training procedure not only improves the tail performance (that is exactly the loss minimized during training) by more than 25%, but also enhances the average performance across the entire task distribution by over 20%. Fig. 1 visualizes the RMSE histogram, including a zoomed-in view of the tails, highlighting the improved robustness of the proposed approach.

D. Out-of-Distribution Analysis

To assess the generalization capability of the robust procedure, we evaluate its performance in two OOD scenarios: (i) same WH system class but with different input signals and (ii) a real-world benchmark from www.nonlinearbenchmark.org, where input signal and system dynamics differ from those used in training.

1) Wiener-Hammerstein System with Different Input Classes

We evaluate the performance of the proposed approach on a WH system with two different input signals: (i) a Random Binary Signal (RBS) and (ii) a random-phase multisine. For both scenarios, a test batch of $b = 20000$ tasks is used. Table II presents the average and tail RMSE (8000 tasks) for both cases. The results indicate a consistent improvement in tail performance (by approximately 18% for multisine and 10% for RBS) and in average performance (by 14% for multisine and 4% for RBS). Fig. 2 presents the RMSE histograms for the RBS, highlighting the tail performance. This confirms that robust training enhances performance in both average and tail metrics in OOD scenarios, showing its generalization capability.

TABLE I: RMSE for in-distribution test tasks.

Training	Test RMSE	Test RMSE (tails)
standard	0.088	0.173
robust	0.070	0.129

TABLE II: RMSE for out-of-distribution test tasks.

Input	Training	Test RMSE	Test RMSE (tails)
RBS	standard	0.232	0.604
	robust	0.223	0.547
multisine	standard	0.151	0.292
	robust	0.130	0.240

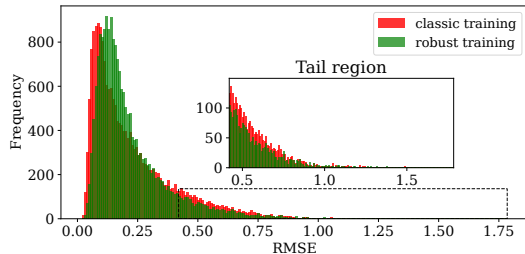


Fig. 2: Out-distribution analysis (RBS): RMSE histogram over the test batch obtained with standard and robust training.

2) Silverbox Benchmark

To further evaluate generalization, we test the trained meta models on the Silverbox SYSID benchmark [15]. The benchmark includes three test sets with real data, generated using different input signals:

- a random-phase multisine;
- an “arrow” signal with linearly increasing amplitude. The arrow signal is used to generate two test inputs:
 - *arrow (no extrapolation)*: the initial portion of the arrow, where the amplitude remains within the amplitude range of the training input;
 - *full arrow*: the complete arrow signal, whose amplitude exceeds that of the training in its last portion.

Since the meta model provides predictions only up to 100 time steps ahead, an *iterative inference* approach is adopted to simulate the whole test datasets of the benchmark. The first inference step utilizes the c initial conditions provided by the benchmark. In subsequent iterations, the final c simulated samples from the previous step are used as initial conditions for the next iteration. This procedure is repeated for all non-overlapping chunks of length 100 within the test dataset.

Additionally, this iterative inference approach is independently applied to all non-overlapping subsequences of length 400 of the Silverbox training dataset, which are processed by the meta model as context. The final RMSE values, averaged over all iterative inference iterations and training dataset subsequences, are reported in Table III. We observe that robust training improves performance for *multisine* (by approximately 10%) and *arrow (no extrapolation)* (by approximately 8%) test inputs. For *full arrow*, performance is nearly identical between the standard and robust procedures. These results confirm that the robust training approach enhances generalization of the meta model.

TABLE III: RMSE for Silverbox benchmark test datasets.

Input	Training	Test RMSE [V]
multisine	standard	0.00928
	robust	0.00837
arrow (no extrapolation)	standard	0.00685
	robust	0.00631
full arrow	standard	0.01088
	robust	0.01079

E. In-Depth Evaluation under equal Training Time and Dataset Budget

To ensure a fair and comprehensive evaluation of the proposed robust approach, two additional comparison scenarios are considered. The results in the previous subsections focused on comparing the *standard* and *robust* training procedures, both after the same number of iterations (700,000). However, two other factors can be employed for comparison: (i) equal training time and (ii) equal number of datasets generated during training.

The second factor is particularly relevant because, in the *robust* approach, a batch of 80 different systems is generated at each iteration, but only 32 (i.e., 40%) are used during backpropagation. In contrast, the *standard* approach generates a batch of 32 systems per iteration, all of which are used. As a result, although both approaches undergo the same number of training iterations in the initial comparison, the total number of generated datasets differs significantly.

To evaluate the implications of this difference, the trained meta-models are tested across all previously introduced scenarios. Table IV reports the resulting RMSE values, both over the entire batch (20,000 systems) and the upper tail (8,000 systems), for WH systems excited by white-noise, multisine, and RBS input signals, as well as for the Silverbox benchmark. The performance of the *robust* model is included as a reference. For clarity, the best-performing RMSE values are shown in **bold**, and the second-best are underlined.

1) *Training Time*: In this scenario, the *standard* training from Section IV-B is extended to match the training time of the *robust* approach. This results in a total of approximately 1.2 million training iterations. The corresponding performance is reported in the second row of Table IV.

From the results in Table IV, we observe that although the *standard* ICL model is trained on approximately 74% more data than in the previously analyzed scenario, the *robust* approach consistently outperforms it across all benchmarks. The performance improvements range from marginal to significant, depending on the specific input scenario. In particular, when comparing this setting to the standard model trained with 700,000 iterations, we note that the increased data volume enhances performance in the in-distribution case and in certain out-of-distribution ones. However, in the most challenging OOD setting—characterized by a RBS input signal—the standard ICL model exhibits a marked performance drop, with an RMSE increase of nearly 50% (approximately 47%).

TABLE IV: RMSE over the test batch for different scenarios, and for different methods.

Method	Epochs	Training-time	Case	Test RMSE	Test RMSE (tails)
Standard ICL	1.75M	40h	white-noise	0.065	0.121
			multisine	0.119	0.232
			SilverBox	[0.00820 , 0.00630 ,0.01100]	
			RBS	0.365	0.854
Standard ICL	1.22M	29h	white-noise	0.075	0.141
			multisine	0.132	0.253
			SilverBox	[0.00892,0.00665, <u>0.01094</u>]	
			RBS	<u>0.342</u>	<u>0.810</u>
Robust ICL	0.7M	29h	white-noise	<u>0.070</u>	<u>0.129</u>
			mutlisine	<u>0.130</u>	<u>0.240</u>
			SilverBox	[<u>0.00837</u> , <u>0.00631</u> , 0.01079]	
			RBS	0.223	0.547

We attribute this degradation to *meta-overfitting* [2], [16], wherein the meta-model begins to memorize specific training instances rather than capturing the generalizable structure of the task distribution. This mirrors classical overfitting, but at the meta-learning level. Evidence for this phenomenon includes not only the degradation under RBS excitation but also the modest gains in the multisine and Silverbox benchmarks, whose input characteristics closely resemble the white-noise, dominated training distribution, supporting the hypothesis that the model overfits to this dominant input type.

2) *Dataset Budget*: In this scenario, the *standard* training is continued for 1.75 million iterations, corresponding to the same total number of datasets generated in the *robust* approach. This includes all systems generated during training, even those discarded in the robust setting due to low estimated risk. Specifically, since only 40% of the systems are used for backpropagation in the *robust* method, matching the total number of generated systems requires $1,750,000 * 32 = (700,000 * 32 / 0.4)$ iterations of the *standard* training procedure.

The results for this setting are presented in the first row of Table IV. The observed trends are consistent with the previous comparison: while the *standard* model slightly outperforms the *robust* one in the ID and near-ID cases, its performance deteriorates significantly in the most challenging OOD scenario (RBS), with an RMSE drop exceeding 50% compared to the result in Table II.

To provide a more comprehensive view, Figure 3 shows test RMSE histograms for the *standard* ICL (trained for 1,750,000 iterations) and the *robust* ICL (trained for 700,000 iterations) under white-noise, multisine, and RBS inputs. These results confirm the table-based analysis: while the *standard* model performs slightly better under white-noise and multisine inputs, it suffers substantial meta-overfitting under RBS excitation, as reflected in the heavy tail of the RMSE histogram.

F. Role of Uncertainty

To evaluate the impact of uncertainty estimation in (1), specifically within the robust learning framework, we conducted an additional experiment. In this setup, the robust

TABLE V: Test RMSE for different risk functions used in the robust ICL framework.

Risk	Case	Test RMSE	Test RMSE (tail)
RMSE	White-noise	0.079	0.145
	Multisine	0.143	0.268
	SilverBox	[9.27, 6.85, 10.88]e-3	–
	RBS	0.263	0.679
KL	White-noise	0.070	0.129
	Multisine	0.130	0.240
	SilverBox	[8.37 , 6.31 , 10.79]e-3	–
	RBS	0.223	0.547

training procedure was repeated using the RMSE as the risk function instead of the KL divergence introduced in (2), thereby disregarding the predicted σ values. The results are summarized in Table V, where the best test RMSE values are highlighted in **bold**. Complementary visualizations are provided in Figure 3.

The results demonstrate that incorporating uncertainty estimates through σ significantly enhances the robustness of the learning process. In particular, Figure 4 reports the validation RMSE (computed on held-out tasks sampled every 2,000 iterations) over the full batch and over its “tail” (i.e., the worst-performing tasks). It is evident that the KL-based robust method consistently achieves lower validation errors compared to the RMSE-based counterpart.

While the Transformer-based meta-model functions as a black box, one possible interpretation is that the KL risk does not merely penalize prediction errors, but also encourages the model to calibrate its confidence via the predicted uncertainty σ . This additional signal appears to guide the learning process toward more robust solutions, especially in heterogeneous and challenging task distributions.

These findings are particularly insightful, as they demonstrate that the effectiveness of the proposed robust approach is not solely due to focusing on high-error tasks. Instead, the choice of the risk function plays a pivotal role. The uncertainty-aware KL divergence introduces a second-order signal, confidence in predictions, that enables the meta-model to assess the reliability of its outputs. This in turn

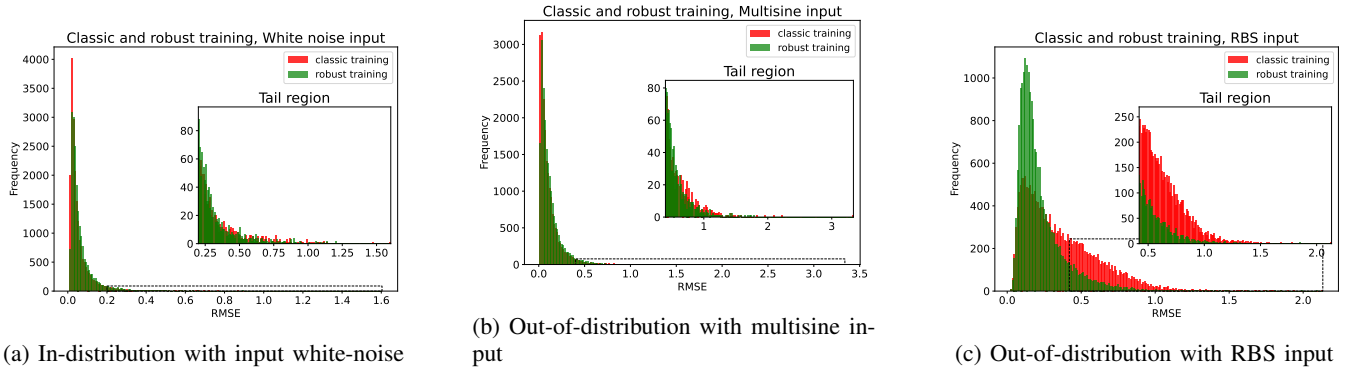


Fig. 3: Histogram analysis: RMSE over the test batch for different input types (in-distribution and out-of-distribution scenarios).

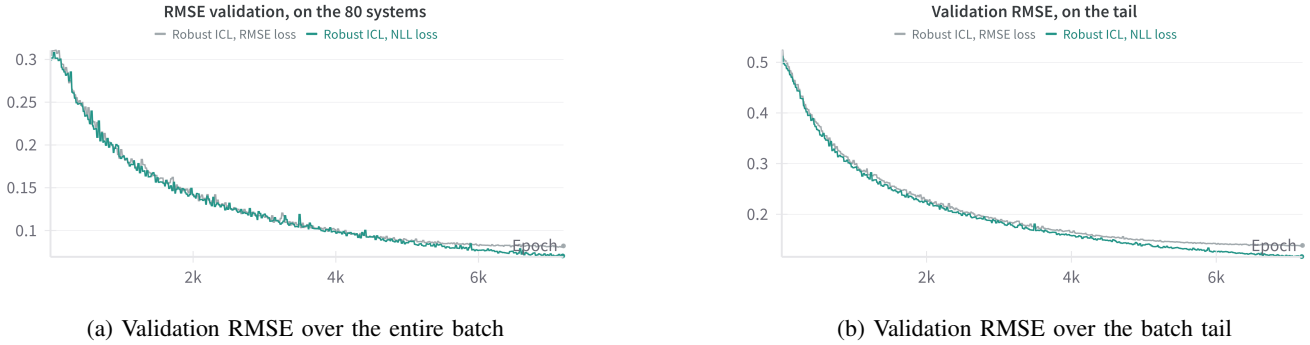


Fig. 4: Comparison of validation losses using robust ICL with RMSE-based and KL-based risk functions.

facilitates more informed risk evaluation and allows the robust optimization paradigm to outperform standard error-based formulations.

V. CONCLUSIONS

This work addresses meta-learning from a distributional robustness standpoint by reformulating the standard expected loss minimization to focus on high-risk tasks. Specifically, the objective is to minimize the expected tail loss over the task distribution, yielding a more robust meta-learning framework. This approach improves both stability and generalization of the resulting meta model. Numerical results show consistent gains not only in tail performance but also across the full task distribution. Robustness is further confirmed in OOD scenarios, where the method achieves reliable improvements.

Current research activities focus on extending the proposed robust learning approach in different directions:

- **Meta-learning with alternative architectures**, such as structured state-space models or diffusion models. This requires appropriately modifying the assumptions in Proposition 3.2 to ensure convergence of the gradient-based robust optimization algorithm;
- **Stochastic optimal control**, with an emphasis on worst-case scenarios to enhance the robustness and safety of the designed controller.

- **Data-efficient training**, to reduce reliance on large synthetic datasets, that often requires prior knowledge of the system, by reusing simulated systems across epochs, while limiting overfitting.
- **Theoretical refinement**, focused on reformulating the assumptions of Lemma 3.1 from a more practical standpoint and deriving a stronger theorem that provides local optimality guarantees under more realistic conditions;
- **Uncertainty analysis**, both quantitatively—by increasing the number of training iterations—and qualitatively, through a theoretical investigation aimed at elucidating the role of uncertainty in guiding robust learning.

REFERENCES

- [1] J. Schmidhuber, “Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook,” Ph.D. dissertation, Technische Universität München, 1987.
- [2] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, “Meta-learning in neural networks: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.
- [3] A. Chakrabarty, G. Wichern, and C. R. Laughman, “Meta-learning of neural state-space models using data from similar systems,” in *World Congress of the International Federation of Automatic Control (IFAC)*, Jul. 2023.
- [4] M. Forgione, F. Pura, and D. Piga, “From system models to class models: An in-context learning paradigm,” *IEEE Control Systems Letters*, vol. 7, pp. 3513–3518, 2023.
- [5] G. Micah, F. Liam, and G. Tom, “Adversarially robust few-shot learning: A meta-learning approach,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

- [6] L. Collins, A. Mokhtari, and S. Shakkottai, "Task-robust model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [7] C. Wang, Y. Lv, Y. Feng, Z. Xie, and J. Huang, "A simple yet effective strategy to robustify the meta learning paradigm," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.
- [8] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *Journal of risk*, 2, 21-42, 2000.
- [9] A. Bemporad, "Active learning for regression by inverse distance weighting," *Information Sciences*, vol. 626, pp. 275-292, 2023.
- [10] H. Rahimian and S. Mehrotra, "Frameworks and results in distributionally robust optimization," *Open Journal of Mathematical Optimization*, vol. 3, 2022.
- [11] M. Rufolo, D. Piga, G. Maroni, and M. Forgone, "Enhanced transformer architecture for in-context learning of dynamical systems," 2025, Accepted for the 2025 European Control Conference (ECC). [Online]. Available: <https://arxiv.org/abs/2410.03291>
- [12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
- [13] X. Qi, J. Wang, Y. Chen, Y. Shi, and L. Zhang, "LipsFormer: Introducing Lipschitz Continuity to Vision Transformers," in *The Eleventh International Conference on Learning Representations*, 2023.
- [14] V. Castin, P. Ablin, and G. Peyré, "How smooth is attention?" in *Forty-first International Conference on Machine Learning*, 2024.
- [15] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in *2013 European Control Conference (ECC)*, 2013.
- [16] L. Kirsch, J. Harrison, J. Sohl-Dickstein, and L. Metz, "General-purpose in-context learning by meta-learning transformers," *arXiv preprint arXiv:2212.04458*, 2022.