Model Predictive Task Sampling for Efficient and Robust Adaptation

Qi (Cheems) Wang[1*] **Zehao Xiao**[2*] **Yixiu Mao**[1*] **Yun Qu**[1*] **Jiayi Shen**[2] **Yiqin Lv**[1] **Xiangyang Ji**[1†]
[1]Department of Automation, Tsinghua University; [2]Informatics Institute, University of Amsterdam
[†]Correspondence Author: xyji@tsinghua.edu.cn

Abstract

Foundation models have revolutionized general-purpose problem-solving, offering rapid task adaptation through pretraining, meta-training, and finetuning. Recent crucial advances in these paradigms reveal the importance of challenging task prioritized sampling to enhance adaptation robustness under distribution shifts. However, ranking task difficulties over iteration as a preliminary step typically requires exhaustive task evaluation, which is practically unaffordable in computation and data-annotation. This study provides a novel perspective to illuminate the possibility of leveraging the dual importance of adaptation robustness and learning efficiency, particularly in scenarios where task evaluation is risky or costly, such as iterative agent-environment interactions for robotic policy evaluation or computationally intensive inference steps for finetuning foundation models. Firstly, we introduce **M**odel **P**redictive **T**ask **S**ampling (MPTS), a framework that bridges the task space and adaptation risk landscape, providing a theoretical foundation for robust active task sampling. MPTS employs a generative model to characterize the episodic optimization process and predicts task-specific adaptation risk via posterior inference. The resulting risk learner amortizes the costly evaluation of task adaptation performance and provably approximates task difficulty rankings. MPTS seamlessly integrates into *zero-shot*, *few-shot*, and *supervised finetuning* settings. Empirically, we conduct extensive experiments in pattern recognition using foundation models and sequential decision-making. Our results demonstrate that MPTS significantly enhances adaptation robustness for tail or out-of-distribution (OOD) tasks and improves learning efficiency compared to state-of-the-art (SOTA) methods. The code is available at the project site https://github.com/thu-rllab/MPTS.

1 Introduction

Generalization across diverse scenarios remains a central challenge in artificial general intelligence. The rise of generative AI offers a promising solution, driving the development of foundation models [1–3]. Unlike traditional task-specific models, which might fail in new tasks, foundation models enable fast deployment across diverse scenarios without learning from scratch. Their rapid problem-solving stems from widely adopted adaptation learning paradigms, including pretraining, meta-learning, and supervised finetuning (SFT).

These paradigms train machine learners over a task distribution, consolidating past experience into problem-solving priors to handle unseen but related tasks in zero-shot or few-shot settings [1,4]. Each iteration samples a task batch, e.g., from a uniform distribution, and then executes the learning-to-adapt step (see Fig. 1/2b). Large language models (LLMs), for instance, treat episodic corpus datasets as tasks and perform in-context learning for adaptation [5]. Similarly, in obtaining generalist robotic policies, decision-making environments, such as Markov decision processes (MDPs), are randomized for robots to perform policy optimization. These task distributions are typically determined by identifiers; e.g., in Fig. 2a, varying physics parameters configure different MDPs as tasks for domain randomization (DR) [6] and meta reinforcement learning (Meta-RL) [7].

Research Motivations: Distribution shifts [8,9] are prevalent in real-world scenarios, making task adaptation robustness at test time increasingly critical [10,11]. In this context, task sampling strategies play a pivotal role, yet uniform sampling often causes catastrophic failures in risk-sensitive scenarios due to the undersampling of critical tasks. Two real-world applications highlight this case: (i) *Tail tasks.* In developing autonomous-driving systems, traffic accidents are rare in training datasets but disproportionately important for testing robustness [12]. (ii) *OOD tasks.* Robots trained in controlled environments struggle in unstructured real-world settings, e.g., leading to errors in navigation and object manipulation. To improve robustness, challenging task prioritized sampling [12–14] has gained traction, where assessing task difficulty is central to robust optimization.

Preliminary Work. *Equal Contributions.

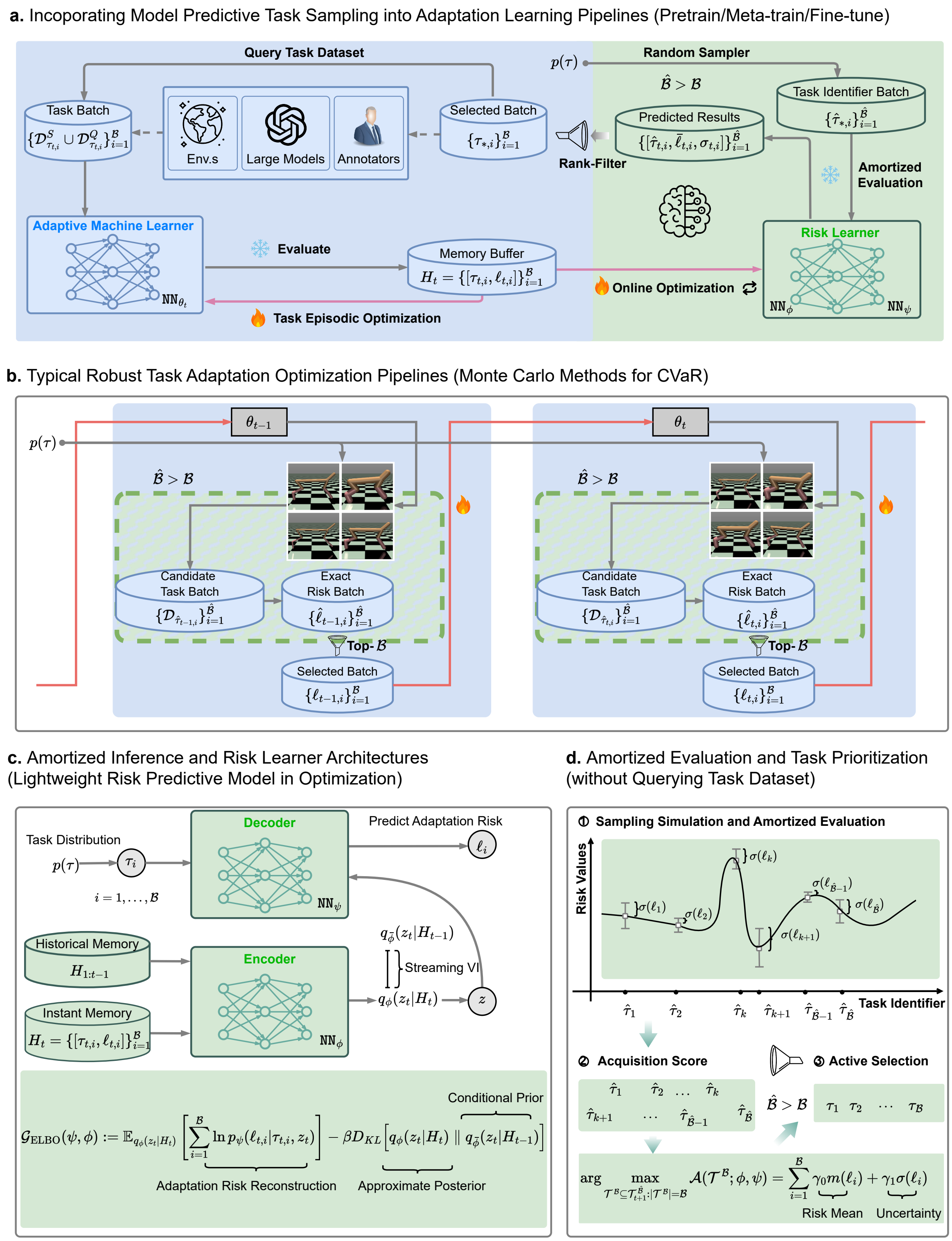

Figure 1: **Framework of MPTS in Adaptation Learning. a.** The left is the standard optimization pipeline in Meta-Learning, DR, or SFT, with the machine learner a foundation model or an RL policy. The right is MPTS using a predict-then-optimize strategy, which integrates posterior inference, adaptation outcome simulation, and challenging task subset selection. [Snow: frozen models; Fire: updated models] **b.** With CVaR$_\alpha$ for DR or Meta-RL as an example, each iteration resamples $\hat{\mathcal{B}}$ MDPs for adaptive policies to interact and evaluate and then select Top-$\mathcal{B}$ worst MDP episodes for optimization. Dashed blue parts are what MPTS amortizes. **c.** The risk learner utilizes the risk history $H_{1:t}$ to train under a streaming VI framework. **d.** The risk learner simulates adaptation outcomes $p(\ell|\boldsymbol{\tau}, H_{1:t}; \boldsymbol{\theta}_t)$ for $\hat{\mathcal{B}}$ candidate identifiers, computes acquisition scores, and selects the Top-$\mathcal{B}$ identifiers for the $(t+1)$-th iteration.

These methods [13–18] evaluate, rank, and prioritize difficult tasks for iterative optimization (see Fig. 1b). However, precisely evaluating tasks—via losses, human annotations, or gradients—incurs high computational costs. For instance, in LLM alignment, task evaluation through SFT requires extensive forward passes, while preference optimization consumes millions of expert annotations [19]. Similarly, in DR and Meta-RL, agents must interact with numerous MDPs to collect post-adaptation episodes and compute returns. These challenges uncover the urgent need for more efficient learning strategies when enhancing robust adaptation, particularly when deploying foundation models or when environment interactions are costly.

Motivated by the above pressing demands, we dive into *robust active task sampling*, a paradigm that has the potential to eliminate unnecessary costs associated with task construction, intensive annotations, or computational overhead during the evaluation of a machine learner's adaptation to tasks. In scenarios involving zero-shot learning, few-shot learning, or SFT, we aim to develop a task sampling strategy that requires *fewer* learning resources but retains *more* deployment benefits such as adaptation robustness in pattern recognition with foundation models and risk-averse sequential decision-making.

Developed Approach: Note that our brain is energy-efficient and simulates the outcome of decision-making in unencountered scenarios from accumulated experience, without actual trials. This capability arises from mechanisms like implicit information gating and active task selection [20–22]. Inspired by this, we propose a model-based optimization approach for adaptive learning, dynamically adjusting task sampling strategies using predicted outcomes as feedback. This work explores the design of risk predictive models, referred to as risk learners, for robust task sampling based on two key insights: (i) Adaptation risk is probably predictable in episodic learning, providing a basis for task difficulty ranking and selection; (ii) Generative modeling of adaptation risk captures risk landscapes with quantified uncertainty, aligning optimization with robustness principles.

To this end, we introduce **M**odel **P**redictive **T**ask **S**ampling (MPTS), a framework for risk-aware task selection. As shown in Fig. 1a, MPTS leverages historical risk to train a lightweight risk learner, which forecasts adaptation risks across the task space to guide the task sampler and optimize the adaptive machine learner. This way amortizes expensive tasks' evaluation for ranking their difficulty to select subset (see Fig. 1b). The risk learner in Fig. 1c adopts a variational autoencoder (VAE) [23] structure, generating adaptation risk estimates via posterior inference [24]. Finally, the acquisition function in Fig. 1d integrates worst-case performance and predictive uncertainty into the rule of subset selection.

MPTS also draws inspiration from active inference [25], which operates through a loop of perception, action, and learning to minimize uncertainty about the planning environment. Here, subset selection from the task batch can be viewed as online planning to derive a robust machine learner. Technically, MPTS specifies or infers identifiers from the task distribution (see examples in Fig. 2a) to establish mappings between identifiers and adaptation risk. It employs streaming variational inference (VI) for risk learner training. Furthermore, by simulating adaptation outcomes in a larger identifier batch for subset selection, MPTS balances exploration (uncertainty minimization) and exploitation (worst-case robustness) across the task space. In primary, our proposed MPTS enjoys several benefits in practice:

1. Adaptation Robustness. The optimization pipeline of MPTS can advance the machine learner's adaptation robustness under severe task distribution shifts, such as tail or OOD task scenarios;
2. Learning Efficiency. Constructing the lightweight risk learner to amortize expensive evaluation, MPTS can diminish computational overhead, avoid unnecessary annotations, and promote efficient exploration in the task space;
3. Framework Versatility. Learning from risk histories, MPTS serves as a plug-play module to rank the task difficulties in optimization and allows seamlessly integration into robust zero-shot or few-shot learning and SFT.

This work evaluates MPTS across few-shot regression, image classification with foundation models, Meta-RL, robotic DR, and prompt-tuning foundation models. Empirical results demonstrate MPTS's outstanding adaptation robustness across diverse scenarios. Compared to SOTA robust adaptation methods, MPTS significantly reduces computational overhead, memory usage, and environment interactions while, in some cases, accelerating learning.

2 Adaptation and Robustness

Notations. We represent a task sample by $\tau \sim p(\tau)$, with $\mathcal{T}$ denoting the task space. Each task τ within the distribution is specified by an identifier, a real-valued vector $\boldsymbol{\tau}$, as illustrated in Fig. 2a. The task-specific risk function $\ell : \mathcal{D}_\tau^S \cup \mathcal{D}_\tau^Q \times \Theta \mapsto \mathbb{R}$ evaluates the adaptation performance of a machine learner $\boldsymbol{\theta}$ on τ. For example, in regression, the support dataset $\mathcal{D}_\tau^S = \{[\boldsymbol{x}_i, \boldsymbol{y}_i]\}_{i=1}^K$ enables rapid adaptation to obtain the model $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\mathcal{D}_\tau^S, \boldsymbol{x})$; while the query dataset $\mathcal{D}_\tau^Q = \{[\boldsymbol{x}_i, \boldsymbol{y}_i]\}_{i=K+1}^{K+N}$ is used for post-adaptation evaluation as risk $\ell = -\frac{1}{N}\sum_{i=1}^N \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\mathcal{D}_\tau^S, \boldsymbol{x}_i)$.

If $|\mathcal{D}_\tau^S| = \varnothing$, ℓ measures zero-shot adaptation; otherwise, it reflects few-shot adaptation risk. In SFT, each sample $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_{\text{SFT}}$ is treated as a task. The episodic task batch history is defined as $\hat{H}_t = \left\{\boldsymbol{\theta}_t, \left(\boldsymbol{\tau}_{t,i}, \mathcal{D}_{\tau_{t,i}}, \ell_{t,i}\right)\right\}_{i=1}^{\mathcal{B}}$, where $\mathcal{B}$ is the task batch size and $\boldsymbol{\theta}_t$ represents the machine learner's parameter in t-th iteration. The tuple set $\left\{\left(\boldsymbol{\tau}_{t,i}, \mathcal{D}_{\tau_{t,i}}, \ell_{t,i}\right)\right\}_{i=1}^{\mathcal{B}}$ includes the sampled

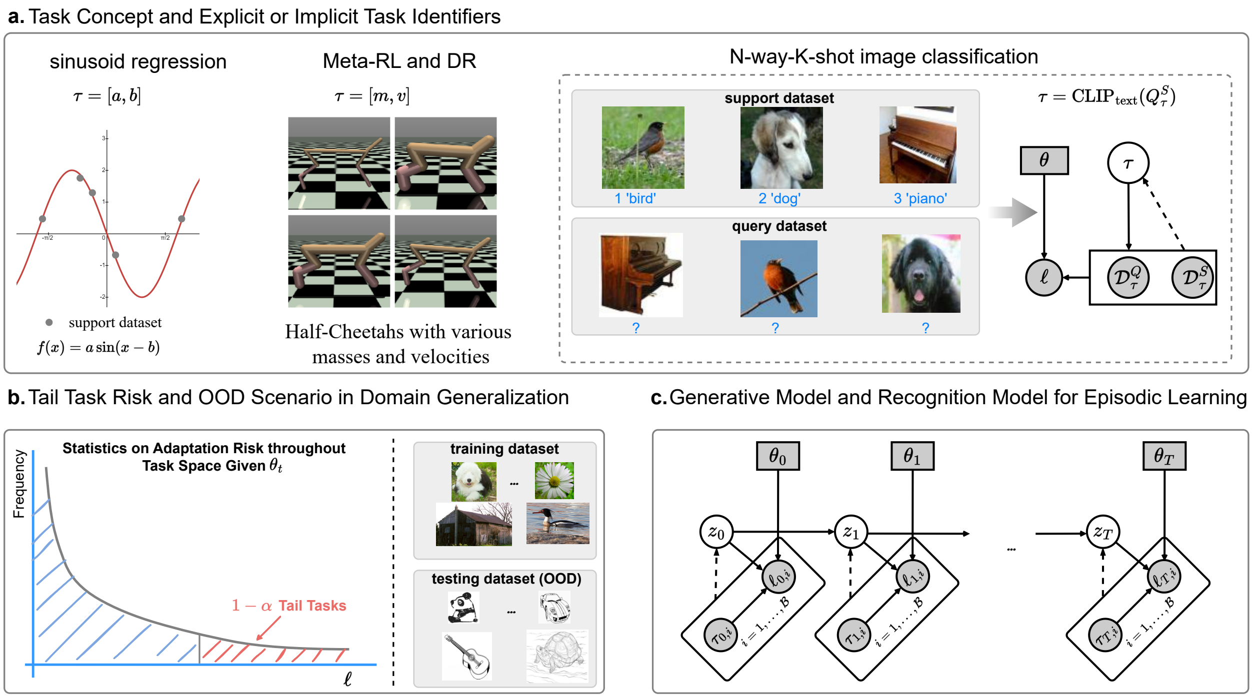

Figure 2: **Fundamental Concepts: Task Identifiers, Episodic Learning and Probabilistic Graphical Models. a.** The task distribution is uniform and defined over meaningful identifiers $\boldsymbol{\tau}$. For example, the amplitude and the phase $[a, b]$ specifies a sinusoid curve to complete with `K-shot` observed data points. Robots like Half-Cheetahs are trained to accomplish different locomotion tasks with varying masses and velocities. Some multimodal pattern recognition tasks' identifiers are implicit but can be described from a reference model, e.g., text encoders in CLIP[1]. **b.** The tail task generalization corresponds to CVaR_α, i.e., the integral of tail task risk values in red. In OOD generalization, this work prompt-tunes CLIP on ImageNet[26] to test on ImageNet-S[27]. **c.** Here, the generative model includes grey units as observed variables and white ones as unobservable. The solid directed lines describe the *generative model*[28]. We use the dash-directed lines to indicate the *recognition model* and approximate inference within autoencoding variational Bayes[23].

task identifier batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$, the support and query dataset $\{\mathcal{D}_{\tau_{t,i}} := \mathcal{D}^S_{\tau_{t,i}} \cup \mathcal{D}^Q_{\tau_{t,i}}\}_{i=1}^{\mathcal{B}}$, and the evaluated adaptation risk $\{\ell_{t,i}\}_{i=1}^{\mathcal{B}}$. For simplicity, the risk history is expressed as $H_t = \{[\boldsymbol{\tau}_{t,i}, \ell_{t,i}]\}_{i=1}^{\mathcal{B}}$, which depends on $\boldsymbol{\theta}_t$.

Adaptation Risk Function. The learning setup optimizes the machine learner within $p(\tau)$. Our analysis is interested in the *risk landscape* in the task space as illustrated in Fig. 1d. Such a perspective emphasizes the interplay between the task identifier $\boldsymbol{\tau}$, the task-specific dataset $\mathcal{D}^S_\tau \cup \mathcal{D}^Q_\tau$ and the adaptation risk function ℓ conditioned on $\boldsymbol{\theta}$. Regarding adaptation performance, we mainly examine zero-shot learning, few-shot learning, and SFT scenarios.

Zero-Shot Adaptation. During training, we evaluate ℓ on the query dataset $\mathcal{D}^Q_\tau$ conditioned on the machine learner $\boldsymbol{\theta}$, i.e., $\ell(\mathcal{D}^Q_\tau; \boldsymbol{\theta})$. With robotic DR[6] as an example, $\ell(\mathcal{D}^Q_\tau; \boldsymbol{\theta})$ denotes the negative return of trajectories collected under the policy $\boldsymbol{\theta}$ in MDP τ. This setup is without support information.

Few-Shot Adaptation. The form of ℓ is specific to meta-learning methods. For instance, MAML[29] implements a bi-level optimization framework. In this case, $\ell(\mathcal{D}^Q_\tau, \mathcal{D}^S_\tau; \boldsymbol{\theta})$ is written as $\ell(\mathcal{D}^Q_\tau; \boldsymbol{\theta}_{\text{meta}} - \alpha\nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}^S_\tau; \boldsymbol{\theta}))$, where $\boldsymbol{\theta}_{\text{meta}}$ denotes the meta initialization, and the inside-bracket term corresponds to finetuning $\boldsymbol{\theta}_{\text{meta}}$ tailored to τ with α the learning rate.

3 Results

This section reports primary findings in robust adaptation and analyzes the effect of MPTS. Prior to elaborating on the experimental setups, we outline the *predict-then-optimize* workflow underpinning MPTS.

Optimization Outcome Prediction with Theoretical Guarantee & MPTS Guided Risk Minimization. First, we characterize the optimization pipeline for a family of robust adaptation methods, i.e., the Monte Carlo strategy for CVaR_α minimization[30]:

$$\cdots \xrightarrow{\text{update}} \boldsymbol{\theta}_{t-1} \xrightarrow{\text{evaluate}} \{[\hat{\boldsymbol{\tau}}_{t-1,i}, \hat{\ell}_{t-1,\mathrm{B}}]\}_{i=1}^{\hat{\mathcal{B}}} \xrightarrow{\text{Top-}\mathcal{B}} H_{t-1} := \{[\boldsymbol{\tau}_{t-1,i}, \ell_{t-1,i}]\}_{i=1}^{\mathcal{B}} \xrightarrow{\text{update}} \boldsymbol{\theta}_t \xrightarrow{\text{evaluate}} \{[\hat{\boldsymbol{\tau}}_{t,i}, \hat{\ell}_{t,i}]\}_{i=1}^{\hat{\mathcal{B}}} \xrightarrow{\text{Top-}\mathcal{B}} \cdots, \quad (1)$$

which picks up the tail tasks to optimize in each iteration. Existing works to prioritize challenging tasks over iterations [12–14,18] take the above steps yet suffer from: (i) learning efficiency issues, such as the need for extensive evaluation of the machine learner across tasks for subset selection, and (ii) restricted batch sizes for evaluating or exploring the task space due to sample or memory constraints. Notably, nearly all of these approaches fail to leverage the optimization outcomes $H_{1:t}$.

Let us predict what to optimize from the cumulated risk episodes. MPTS differs from prior works and reuses $H_{1:t}$ to train the risk learner. Coupling the identifier $\boldsymbol{\tau}$ and adaptation risk $\ell(\mathcal{D}_{\tau}^{Q}, \mathcal{D}_{\tau}^{S}; \boldsymbol{\theta})$ forms a streaming database to online learn. In **Methods** part, Theorem 1 provides a provable basis for ranking tasks from predicted outcomes, suggesting stable ranking relation of task difficulties under perturbations in $\boldsymbol{\theta}$, e.g., a gradient update with a small learning rate. Thus, the candidate tasks $\mathcal{T}_{t+1}^{\hat{\mathcal{B}}}$ at $\boldsymbol{\theta}_t$ probabilistically preserve their relative difficulty rank at $\boldsymbol{\theta}_{t+1}$. Moreover, learning stochastic adaptation risk provides a probabilistic risk landscape over iterations.

MPTS surrogates CVaR optimization with efficiency and exploration benefits. Learning $p(\ell|\boldsymbol{\tau}, H_{1:t}; \boldsymbol{\theta}_t)$ enables efficient evaluation across infinite tasks with minimal computation, expanding the pseudo batch size $\hat{\mathcal{B}}$ for subset selection and fostering exploration. For clarity, we treat MPTS as a risk minimization framework under specific acquisition criteria. As shown in Fig. 1 and Fig. 2c, its core workflow involves training the risk learner $p(\ell|\boldsymbol{\tau}, H_{1:t}; \boldsymbol{\theta}_t)$, evaluating task-specific adaptation risk via posterior inference, and screening task subsets using the upper confidence bound (UCB) principle [31] for $(t+1)$-th optimization. These operations are formalized in Eq. (2), where the Monte Carlo estimate of the risk learner yields the mean $m(\ell)$ and standard deviation $\sigma(\ell)$ of task adaptation risk, while the acquisition function $\mathcal{A}(\cdot)$ quantifies total subset risk.

$$\text{Approximate Optimization Outcome after Adaptation}: \quad \max_{\boldsymbol{\psi}\in\boldsymbol{\Psi}} \mathcal{L}_{\text{ML}}(\boldsymbol{\psi}) := \ln p_{\boldsymbol{\psi}}(H_t|H_{1:t-1}) \tag{2a}$$

$$\text{Amortized Evaluation with Risk Learners}: \quad p_{\boldsymbol{\psi}}(\ell|\boldsymbol{\tau}_i, H_{1:t}; \boldsymbol{\theta}_t) \xrightarrow{\text{Monte Carlo Estimates}} \{m(\ell_i), \sigma(\ell_i)\}_{i=1}^{\hat{\mathcal{B}}} \tag{2b}$$

$$\text{Active Subset Selection under the UCB Principle}: \quad \mathcal{T}_{t+1}^{\mathcal{B}} = \arg\max_{\mathcal{T}^{\mathcal{B}}\subseteq\mathcal{T}_{t+1}^{\hat{\mathcal{B}}}:|\mathcal{T}^{\mathcal{B}}|=\mathcal{B}} \mathcal{A}(\mathcal{T}^{\mathcal{B}}; \boldsymbol{\phi}, \boldsymbol{\psi}) \tag{2c}$$

Approximating optimization outcome relies on streaming VI [32,33], with the risk learner a lightweight model. Selecting a portion of challenging tasks to optimize, MPTS can be viewed as a surrogate of $\text{CVaR}_{1-\mathcal{B}/\hat{\mathcal{B}}}$ minimization while circumventing extra computations, annotations, or environment interactions. This design not only enhances learning efficiency but also aligns with the overarching goals of robust adaptation. Repeating the boxed steps of MPTS until convergence brings a robust adaptive machine learner, and the implementation details are attached in **Methods**.

Adaptation Learning Benchmark. The experimental design considers the benchmark typicality and the practical challenges. Downstream tasks span across pattern recognition and sequential decision-making, with certain experiments involving multimodal foundation models. These experiments mainly examine few-shot adaptation and include (1) `K-shot` sinusoid regression [29], (2) `N-way K-shot` image classification [34] with CLIP models and (3) Meta-RL [29]. Additionally, MPTS validates scenarios like (4) robotic DR [35] for zero-shot adaptation and (5) SFT CLIP models towards image classification.

Backbones and Task Robust Baselines. This study primarily compares MPTS with risk minimization principles and focuses on robustness improvement. While these methods—including MPTS—are agnostic to zero-shot, few-shot learning, or finetuning techniques, we adopt SOTA backbones for experiments. For sinusoid regression and Meta-RL, we use MAML [29]. As CLIP [1] has strong zero-shot performance, we extend it with MaPLe [36] for `N-way K-shot` image classification. For robotic DR in Ergo-Reacher and Lunar-Lander [35], we employ TD3 [37] due to its stability. In SFT, we again use MaPLe for prompt-tuning in image classification.

Baselines include Expected/Empirical Risk Minimization (ERM) [38], Distributionally Robust Risk Minimization (DRM) [14,16,17,39,40], and Group Distributionally Robust Risk Minimization (GDRM) [41–43]. Accordingly, adaptation robustness is evaluated via CVaR_α across validation/testing tasks with $\alpha = \{0.9, 0.7, 0.5\}$, including some OOD results. We also compare computational cost, memory usage, and sample efficiency. For fairness, all baselines share the same task batch $\mathcal{B}$ in optimization, excluding pruned easier tasks. ERM and GDRM use batch size $\mathcal{B}$, while DRM samples $\hat{\mathcal{B}} = 2\mathcal{B}$, filtering half for stable optimization. See Supplementary Notes F/G for details.

3.1 Demonstration of the MPTS's role in `K-shot` sinusoid regression

In `K-shot` sinusoid regression [29], the function family $\{f(x) = a_i \sin(x - b_i) | (a_i, b_i) \in [0.1, 5.0] \times [0.0, \pi]\}$ is specified by the identifier $\boldsymbol{\tau} = [a, b]$. This serves as a toy case to illustrate MPTS and the role of the risk learner.

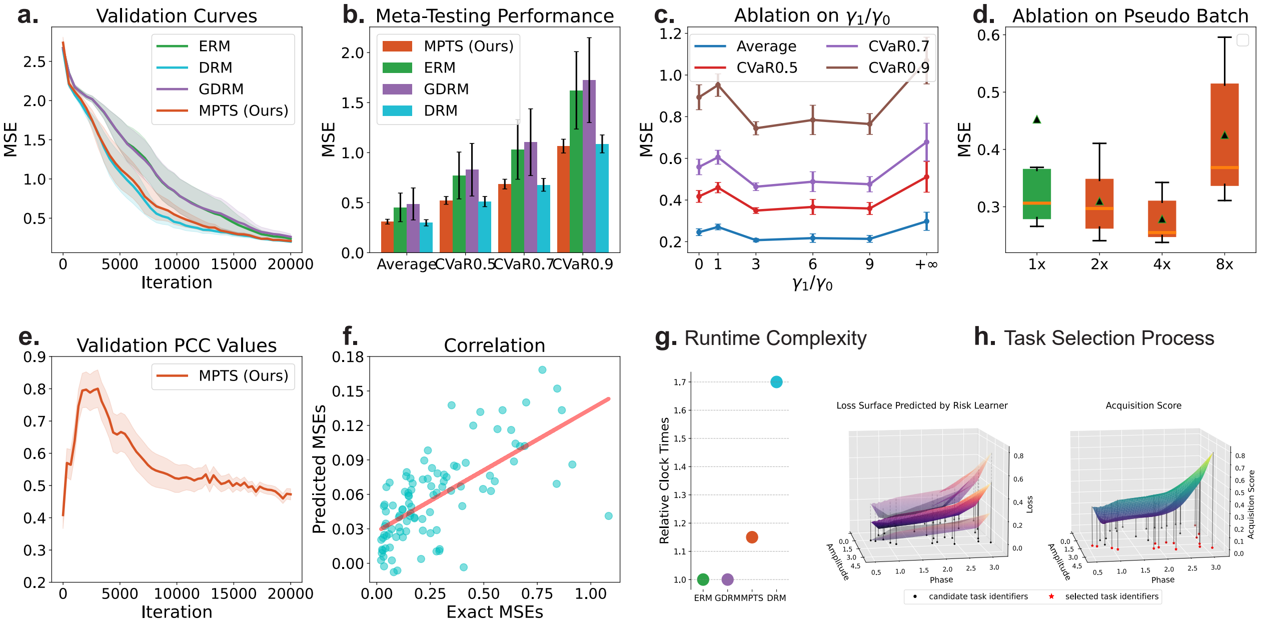

Figure 3: `K-shot` **Sinusoid Regression Results (7 Runs). a.** Shown are curves of averaged MSEs on the validation task set during meta-training for all methods . **b.** The meta-trained machine learners are tested on a fixed task set, reporting the average MSEs and CVaR values. **c.** Displayed are meta-testing results with MPTS machine learners trained by various γ_1/γ_0 ratios. **d.** Displayed are meta-testing results with MPTS machine learners trained in various pseudo batch sizes, i.e., $\hat{\mathcal{B}} = \{1\mathcal{B}, 2\mathcal{B}, 4\mathcal{B}, 8\mathcal{B}\}$. **e.** The PCC values are tracked during meta-training. **f.** At a specific iteration, the statistical correlation between predicted and exact adaptation risk values of the task batch is visualized with overall $\rho_{\bar{\ell},\ell} = 0.669$. **g.** The required relative run-time is computed for all methods during meta-training with ERM as the anchor. **h.** At some meta-training time-step, we visualize the subset selection from the pseudo batch under the risk learner.

The risk learner allows for roughly scoring the task difficulty over iterations. In MPTS, for the screened subset at $(t+1)$-th iteration, we track the predicted risk values $\{\bar{\ell}_{t+1,i} :\approx \mathbb{E}_{q_{\phi}(\boldsymbol{z}_t|H_t)}[p_{\psi}(\ell|\boldsymbol{\tau}_{t+1,i}, H_{1:t})]\}_{i=1}^{\mathcal{B}}$ and corresponding exact evaluations $\{\ell_{t+1,i}\}_{i=1}^{\mathcal{B}}$ from $\boldsymbol{\theta}_{t+1}$ to compute the Pearson correlation coefficient (PCC) $\rho_{\bar{\ell},\ell} := \frac{\sum_{i=1}^{\mathcal{B}}(\bar{\ell}_{t+1,i} - \text{Mean}[\{\bar{\ell}_{t+1,\cdot}\}])(\ell_{t+1,i} - \text{Mean}[\{\ell_{t+1,\cdot}\}])}{\sqrt{\sum_{i=1}^{\mathcal{B}}(\bar{\ell}_{t+1,i} - \text{Mean}[\{\bar{\ell}_{t+1,\cdot}\}])^2}\sqrt{\sum_{i=1}^{\mathcal{B}}(\ell_{t+1,i} - \text{Mean}[\{\ell_{t+1,\cdot}\}])^2}}$. For continuous risk values, PCC reasonably quantifies the effect of ranking in a batch. The risk learner amortizes the exact evaluation $\ell(\mathcal{D}_{\tau}^{Q}, \mathcal{D}_{\tau}^{S}; \boldsymbol{\theta}_t)$ $\forall \boldsymbol{\tau} \in \mathcal{T}$ and $\boldsymbol{\theta}_t \in \boldsymbol{\Theta}$ using risk histories, indirectly reflecting adaptation difficulty. Only if the risk learner approximately ranks tasks, MPTS can trust amortized evaluations for worst subset selection.

As shown in Fig. 3e, $\rho_{\bar{\ell},\ell}$ remains between 0.4–0.8 across iterations, validating the reliability of the risk learner in predicting adaptation outcomes. However, PCC declines over time—a trend also observed across experiments—likely due to $\boldsymbol{\theta}_t$ model convergence. This reduces task diversity, negatively affecting the risk learner's training after local task space overoptimization. Fig. 3f shows the statistical correlation between predicted and exact adaptation risk at a specific iteration. Scattered points demonstrate strong overall alignment, despite varying value scales between iterations. Notably, difficult tasks with high MSEs are well identified and clustered around or above the correlation slope along the x-axis.

MPTS accelerates the learning process and improves comprehensive adaptation performance under active sampling. In Fig. 3a, MPTS converges faster, completing optimization in 15K iterations, compared to 20K for ERM and GDRM, due to its uncertainty-guided worst-case acquisition. DRM processes $2\mathcal{B}$ tasks to filter half per iteration, raising 0.7× computational overhead over ERM (Fig. 3g). In contrast, MPTS incurs only 0.14× runtime increase, a marginal overhead. To illustrate active task sampling, Fig. 3h visualizes predicted risk values over the task space. Selected tasks favor regions with high deviations, clustering in high-risk areas.

In meta-testing, Fig. 3b shows MPTS and DRM achieve the lowest average and CVaR_{α} mean-square errors (MSEs), with their advantage over GDRM and ERM increasing at higher confidence levels α. Prior work[40] confirms DRM's efficiency sacrifice for robustness, relying on intensive task evaluation. Using MAML, gradient-based inner-loop adaptation further increases overhead, whereas MPTS bypasses it via probabilistic prediction, reducing computational cost.

The appropriate hyper-parameter setup secures performance and efficiency. We first analyze the acquisition function $\mathcal{A}(\mathcal{T}^{\mathcal{B}}; \phi, \psi)$ by varying trade-off parameters $\{\gamma_0, \gamma_1\}$ in Fig. 1d/Eq. (17). Meta-testing machine learners trained with $\frac{\gamma_1}{\gamma_0} = \{1.0, 3.0, 6.0, 9.0\}$, $\gamma_0 = 0.0$ and $\gamma_1 = 0.0$ (Fig. 3c) shows that higher uncertainty weights lower average MSEs. However, removing worst-case considerations ($\gamma_0 = 0.0$) weakens performance. We further examine the impact of pseudo batch size $\hat{\mathcal{B}}$ in Fig. 3d. Increasing $\hat{\mathcal{B}}$ reduces average MSEs, but excessively large values (e.g., $\hat{\mathcal{B}} = 8 \times \mathcal{B}$) degrade performance. This occurs because an enlarged identifier set under worst-case selection might over-optimize local task regions, hindering global generalization. Thus, MPTS configuration follows two principles: (i) $\hat{\mathcal{B}}$ should be moderate to encourage exploration while preventing excessive local optimization. (ii) Since adaptation robustness is the priority, we consistently set $\gamma_0 \in \mathbb{R}^+$ as the default in all experiments.

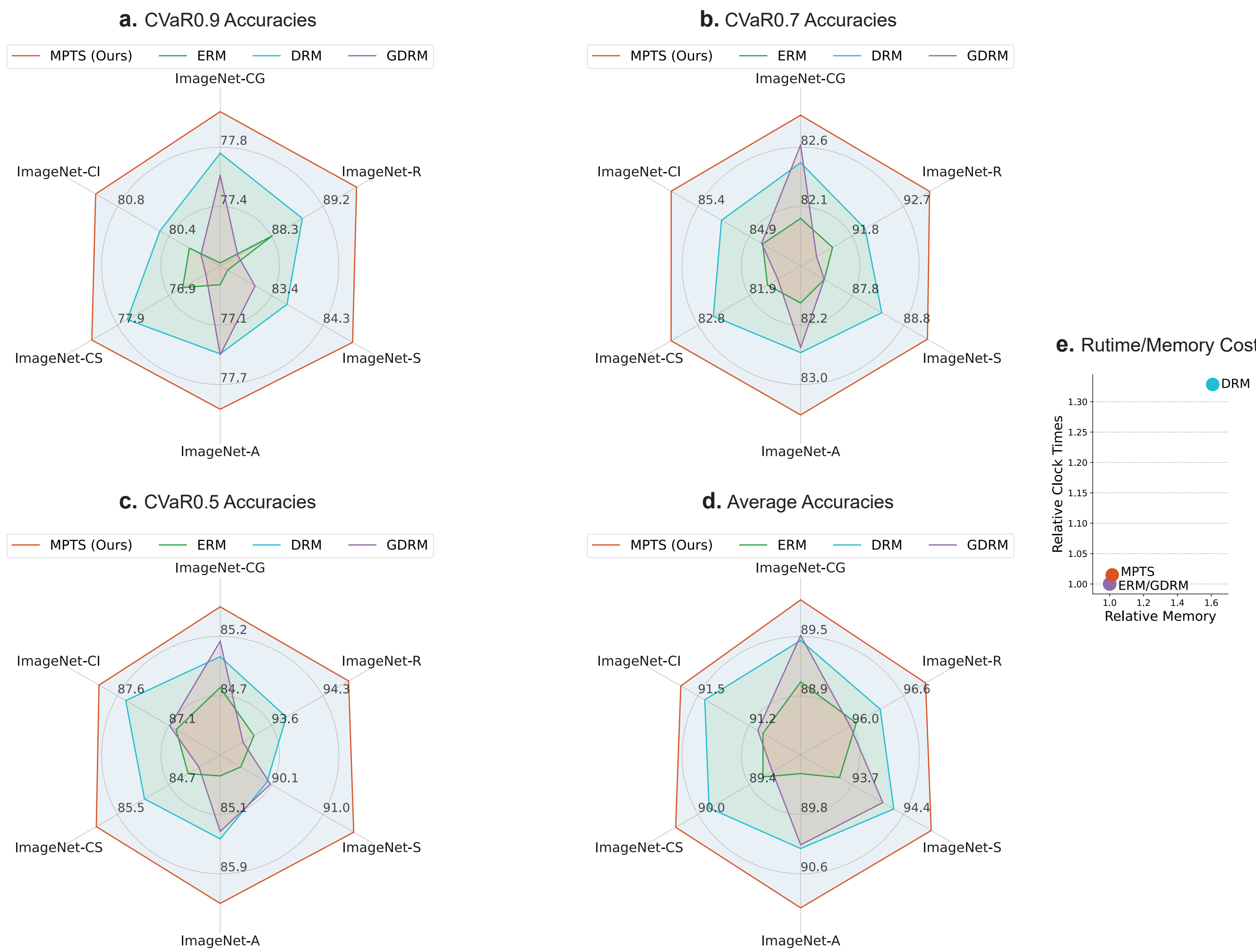

Figure 4: `5-way 1-shot` **Meta-testing Classification Results (3 Runs in Average). a-d.** Shown are testing $\text{CVaR}_{0.9}$, $\text{CVaR}_{0.7}$, $\text{CVaR}_{0.5}$ and average accuracies with meta-trained machine learners on different datasets. **e.** With experiments on ImageNet-A as an example, we report the memory cost and clock time relative to ERM during meta-training for all methods.

3.2 Few-Shot adaptation benefits from MPTS in robustness and learning efficiency

Result analysis in `N-way K-shot` image classification. We perform `5-way 1-shot` image classification using MaPLe, with six meta-training datasets from ImageNet-CG[44], ImageNet-CI[44], ImageNet-CS[44], ImageNet-A[45], ImageNet-S[27] and ImageNet-R[46]. Fig. 4e compares computational time and memory usage across methods during meta-training. The overhead from optimizing risk learners in MPTS is negligible, whereas DRM incurs 1.3× computational time and 1.6× memory usage relative to ERM.

In meta-testing, MPTS achieves the highest average accuracy across all six datasets in Fig. 4a-d and Supplementary Notes Table 4. Robustness evaluation illustrates comprehensive accuracy increases in $\text{CVaR}_{0.5}$, $\text{CVaR}_{0.7}$ and $\text{CVaR}_{0.9}$ for both MPTS and DRM. Performance trends remain consistent across datasets, with all robust methods outperforming ERM. Among them, MPTS and DRM lead in all metrics, though DRM exchanges more computational resources for CVaR_α accuracies. Overall, this benchmark result witnesses the comprehensive merits of prioritizing challenging tasks.

Result analysis in Meta-RL. We first analyze meta-training results in Fig. 5a-b. MPTS achieves the highest $\text{CVaR}_{0.9}$ validation returns on most benchmarks. DRM sacrifices average returns on HalfCheetahMassVel, HalfCheetahVel, and Walker2dVel, whereas MPTS maintains average performance comparable to ERM on HalfCheetahMassVel and Walker2dVel. GDRM behaves intermediate performance, while DRM balances average and $\text{CVaR}_{0.9}$ returns, excelling on ReacherPos. Fig. 5c witnesses the risk learner's strong task difficulty discrimination capability, measured by $\rho_{\bar{\ell},\ell}$. In Fig. 5d, DRM consumes 1.5x runtime on Walker2dVel due to extra environment interactions, while MPTS avoids this inefficiency.

Meta-testing results in Fig. 5e-f highlight MPTS's robustness, with return gains increasing at higher α values. In extreme cases ($\text{CVaR}_{0.9}$), MPTS surpasses ERM by over 20% on all benchmarks. Average performance varies: Walker2dMassVel and Walker2dVel show minor differences, while HalfCheetahMassVel favors MPTS with slightly higher variance. HalfCheetah marginally benefits GDRM and ERM, whereas ReacherPos favors MPTS and DRM with reduced variance. Overall, MPTS is close to ERM in average performance while offering superior adaptation robustness and computational efficiency compared to DRM. Methods often trade off worst-case and average performance in Meta-RL, as implied in work [17].

3.3 MPTS retains multi-faced advantages beyond robustness in zero-shot continuous control

MPTS dominates the overall performance in DR training. In Fig. 6a Ergo-Reacher, distinguished from Meta-RL conclusion, MPTS and DRM improve both average and CVaR_α performance. This likely stems from MPTS's broader task exploration via larger $\hat{\mathcal{B}}$. Meanwhile, $\rho_{\bar{\ell},\ell}$ fluctuates near 0.4 throughout training. In Fig. 6b Lunar-Lander, MPTS maintains the leading trend in average and CVaR_α returns. In contrast, DRM and GDRM not only underperform in average returns but also achieve the lowest $\text{CVaR}_{0.9}$ values, failing in robust optimization. Simple worst-case selection or reweighting tends to degrade performance when unsolvable tasks are frequently sampled. MPTS mitigates this by balancing worst-case and uncertainty-guided selection, preventing over-optimization on a finite set of difficult MDPs. Here, $\rho_{\bar{\ell},\ell}$ peaks above 0.6 before stabilizing near 0.3, consistent with prior findings that task selection converges, reducing task difficulty discrimination. MPTS's runtime in Lunar-Lander is comparable to ERM and GDRM in Fig. 6e. In Lunar-Lander, the identifier $\boldsymbol{\tau} \in \mathbb{R}^+$ represents the main engine strength. Fig. 6f shows task sampling frequency, where MPTS favors lower-engine-strength MDPs while still exploring all engine-strengths early in training.

MPTS facilitates policy adaptation in the worst-case and OOD MDPs. For DR-trained policies, Fig. 6c-d confirm MPTS and DRM's superior CVaR_α returns in Ergo-Reacher, while ERM exhibits a minor dip in average returns. In Lunar-Lander, MPTS attains the highest CVaR_α returns, remaining stable even as α increases—outperforming ERM by over 20%. Additionally, MPTS and ERM yield top average returns with minimal variance. For OOD generalization, we shift $\boldsymbol{\tau}$'s range from training interval $\boldsymbol{\tau} \in [4.0, 20.0]$ to testing interval $\boldsymbol{\tau} \in [1.0, 4.0) \cup (20.0, 23.0]$. All methods struggle in hard OOD tasks (Fig. 6g left), but MPTS dominates in difficult cases, demonstrating strong adaptation. DRM exhibits high variability and weak generalization, even for easier tasks (Fig. 6g right).

3.4 MPTS also reserves the potential of robust SFT

In SFT, each labeled example in the dataset can be viewed as a task. We apply prompt tuning to adapt pretrained models using SFT datasets. Following MaPLe, we execute prompt tuning on ImageNet [26] and conduct standard evaluation. To assess post-SFT robustness, we test on four OOD datasets—ImageNet-A [45], ImageNet-S [27], ImageNet-R [46], and ImageNet-V [47] for capturing diverse domain shifts.

Fig. 7a-d shows MPTS consistently outperforms baselines in average and CVaRCVaRCVaR accuracies on ID and OOD datasets. MPTS achieves 0.82–3.11% higher $\text{CVaR}_{0.9}$, $\text{CVaR}_{0.7}$ and $\text{CVaR}_{0.5}$ scores over ERM (Supplementary Notes Table 5), with greater OOD advantages than on ImageNet. On 4/5 datasets, DRM ranks second to MPTS in CVaR_α but matches ERM in average accuracy. GDRM's performance varies with α, showing only marginal gains over ERM. Still, DRM sacrifices efficiency for robustness in Fig. 7e. While MPTS shares DRM's optimization goal, its risk predictive module and larger-batch simulation enable better task exploration at minimal computational cost, yielding a more robust machine learner.

4 Discussion

Rapid adaptation to novel scenarios is a cornerstone of artificial general intelligence. However, challenges such as safety, limited annotations, and computational constraints necessitate robust and efficient adaptation mechanisms. This study explores learn-to-adapt optimization via generative modeling and introduces MPTS, a versatile framework for robust active task sampling.

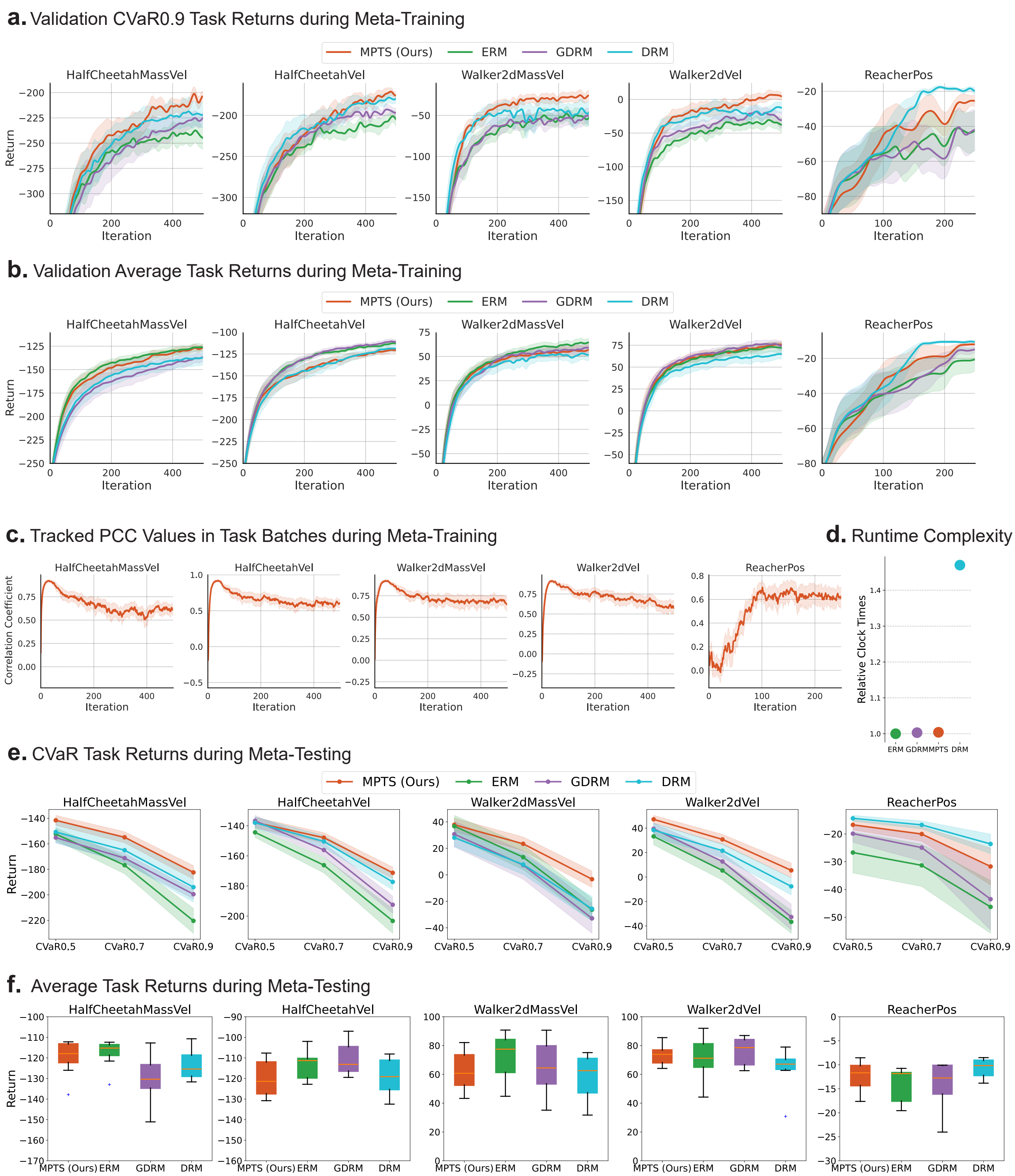

Figure 5: **Meta-RL Results on Five Mujoco Environments (7 Runs). a.** The cumulative returns with standard error of means (SEMs) belonging to $\text{CVaR}_{0.9}$ validation MDPs are displayed during meta-training. **b.** We compute the average cumulative returns with SEMs on validation MDPs during meta-training. **c.** Tracked are the risk learner's PCC values with SEMs over training iterations. **d.** The relative clock time quantifies the computational complexity for all methods on Walker2dVel, where ERM's runtime works as the anchor. **e.** We report CVaR_α returns of meta-testing MDPs. **f.** The box-plot reports results averaged over meta-testing MDPs.

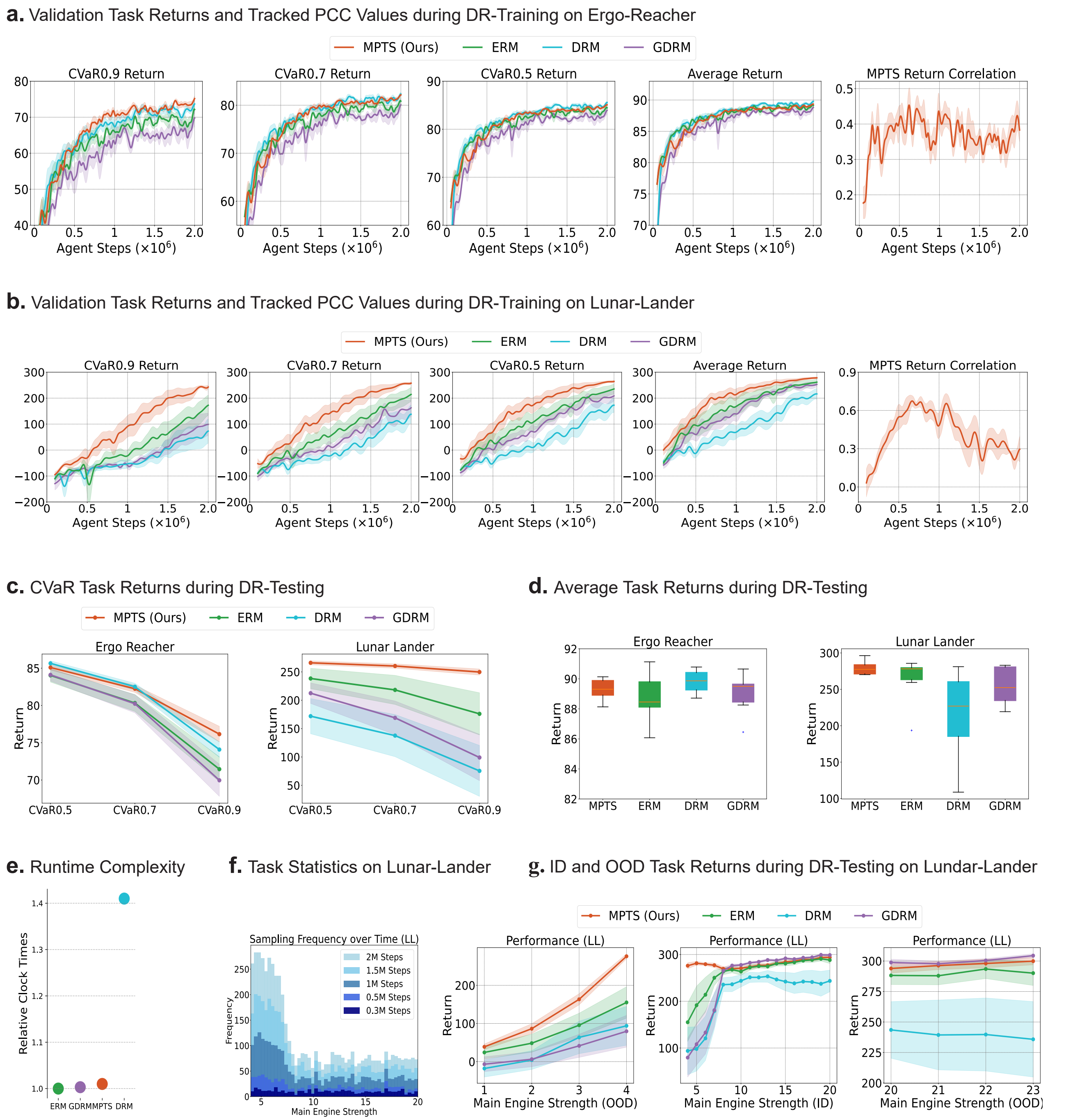

Figure 6: **DR Results on Ergo-Reacher and Lunar-Lander (7 Runs). a.** In Ergo-Reacher, the $\mathrm{CVaR}_{0.9}$, $\mathrm{CVaR}_{0.7}$, $\mathrm{CVaR}_{0.5}$ and average cumulative returns on validation MDPs are reported together with the risk learner's PCC curve during DR training. **b.** In Lunar-Lander, the cumulative returns on validation MDPs are illustrated together with the risk learner's PCC curve during DR training. **c.** We test the DR-trained policies on the fixed MDP set and report the CVaR_{α} cumulative returns. **d.** The returns averaged over DR-testing MDPs are illustrated. **e.** The required runtime is computed for all methods on Lunar-Lander. **f.** In Lunar-Lander, shown are frequencies of sampled identifiers using MPTS during DR training. **g.** In Lunar-Lander, we test the trained policies in both in-distribution (ID) domains and out-of-distribution (OOD) domains to report each task's average returns.

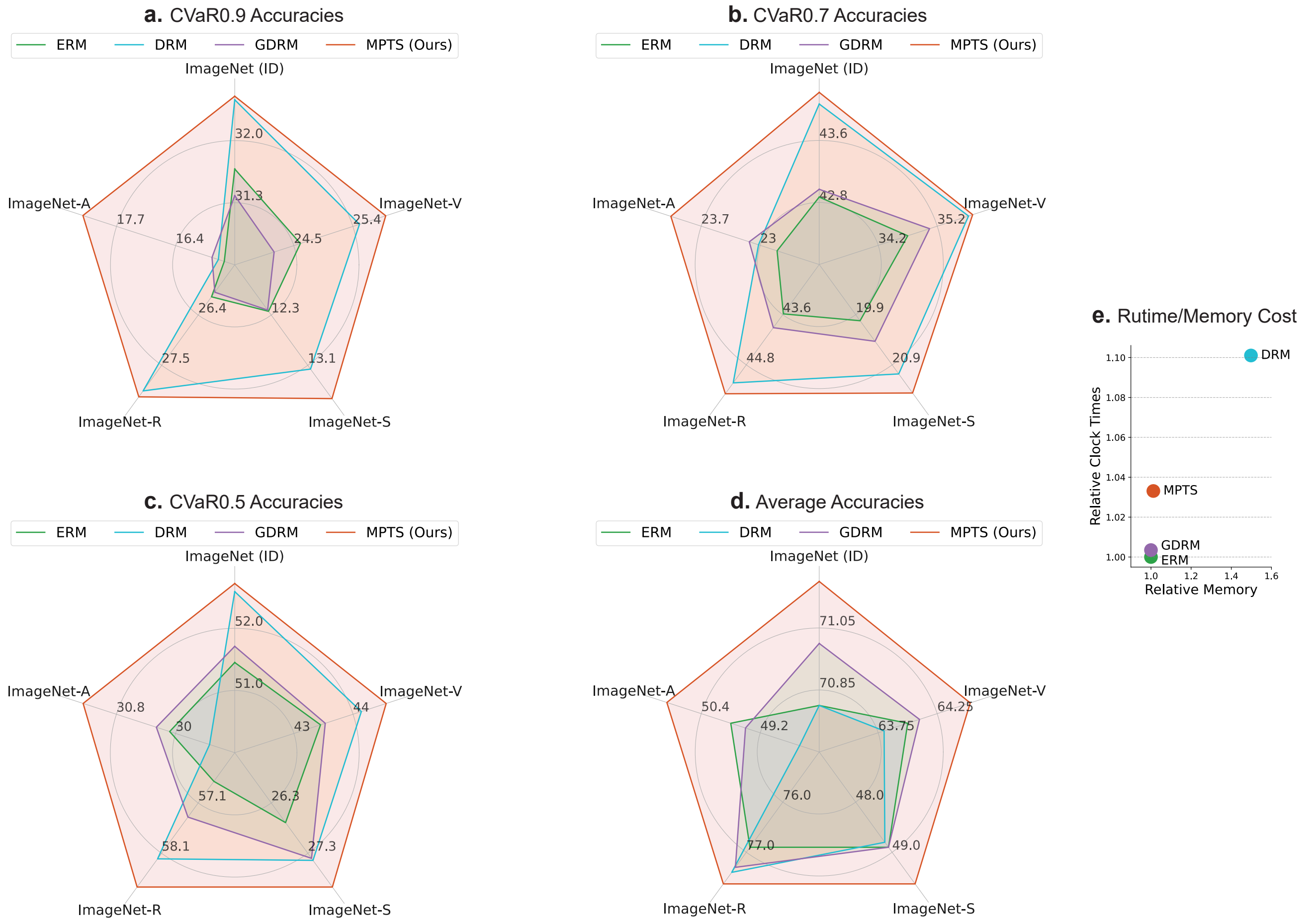

Figure 7: **Testing Classification Results after Prompt-Tuning on ImageNet (3 Runs in Average). a-d.** Shown are testing $\text{CVaR}_{0.9}$, $\text{CVaR}_{0.7}$, $\text{CVaR}_{0.5}$ and average accuracies with the prompt-tuned machine learner on ID and OOD datasets. **e.** During prompt-tuning ImageNet, we report the memory cost and clock time relative to ERM for all methods.

Experiments demonstrate the feasibility of predicting optimization outcomes for active task selection. Meanwhile, MPTS enhances adaptation robustness across diverse scenarios in an efficient manner. These results highlight MPTS's potential to scale CVaR_α principles for foundation model development and large-scale decision-making, without additional learning resources.

5 Methods

In alignment with the realistic necessities, this work focuses on robust adaptation while securing learning efficiency, such as circumventing partial expensive evaluation. Such a purpose facilitates the birth of MPTS. As previously mentioned, the framework is agnostic to adaptation learning methods; hence, we leave out zero-shot learning, few-shot learning, and SFT details.

In Fig. 1a, several roles are involved in the optimization: (1) the **adaptive machine learner**, e.g., foundation models or generalist policies, learns to adapt given some optimizers; (2) the **risk learner** as a critic evaluates and forecasts the task-specific adaptation risk; (3) the **task sampler** as an actor works for screening the task subset for next iteration. These components participate in episodic learning until convergence.

Technically, this work recasts task episodic learning to sequence generation and presents MPTS as the task sampling strategy to balance exploration and exploitation. At first, we introduce the foundation of risk predictive models for ranking task difficulty. To reconcile theory and practice, we introduce a tractable optimization approach to enable functional posterior inference towards adaptation risk. Then, we devise the acquisition function informed by the captured risk landscapes. Finally, an understanding concerning the optimization pipeline is attached to conclude the **Methods** part.

5.1 Theoretical Feasibility of Constructing Risk Predictive Models

We begin by introducing Assumptions 1/2/3, which characterize the smoothness and boundedness conditions essential to the optimization framework. Specifically, under a fixed machine learner $\boldsymbol{\theta}$, it is reasonable to expect that similar tasks, represented by $\boldsymbol{\tau}$, will exhibit sufficiently close adaptation risk values. CVaR_α in Definition 1 is commonly used for measuring the expected risk in the worst-case scenarios, i.e., $1-\alpha$ proportional tail cases, with α a specific confidence level.

Definition 1 (Conditional Value-at-Risk (CVaR)[30]) *Given the machine learner parameter $\boldsymbol{\theta}$, we denote the task specific random variable by $\ell_i := \ell(\mathcal{D}^Q_{\tau_i}, \mathcal{D}^S_{\tau_i}; \boldsymbol{\theta})$. Throughout the task space $\mathcal{T}$, let the cumulative risk distribution and the quantile of risk values respectively be $F(\ell)$ and $\ell^\alpha = \min_\ell\{\ell | F(\ell) \geq \alpha\}$. Then the CVaR at α-robustness level can be estimated as:*

$$\mathit{CVaR}_\alpha[\ell(\mathcal{T};\boldsymbol{\theta})] = \int \ell dF^\alpha(\ell;\boldsymbol{\theta}), \tag{3}$$

where we define the normalized cumulative distribution of task risk values by:

$$F^\alpha(\ell;\boldsymbol{\theta}) = \begin{cases} 0, & l < \ell^\alpha \\ \frac{F(\ell;\boldsymbol{\theta})-\alpha}{1-\alpha}, & l \geq \ell^\alpha. \end{cases} \tag{4}$$

And this induce the tail risk task distribution denoted by $p_\alpha(\tau;\boldsymbol{\theta})$.

Assumption 1 (Lipschitz Continuity) *We assume the adaptation risk function $\ell(\cdot;\boldsymbol{\theta})$ reserves the Lipschitz continuity w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$, i.e.,*

$$|\ell(\mathcal{D}^Q_\tau, \mathcal{D}^S_\tau; \boldsymbol{\theta}) - \ell(\mathcal{D}^Q_\tau, \mathcal{D}^S_\tau; \boldsymbol{\theta}')| \leq \beta_1 ||\boldsymbol{\theta} - \boldsymbol{\theta}'|| \quad \mathit{and} \quad |\ell(\mathcal{D}^Q_\tau, \mathcal{D}^S_\tau; \boldsymbol{\theta}) - \ell(\mathcal{D}^Q_{\tau'}, \mathcal{D}^S_{\tau'}; \boldsymbol{\theta})| \leq \beta_2 ||\boldsymbol{\tau} - \boldsymbol{\tau}'||, \tag{5}$$

where $\forall\{\boldsymbol{\theta},\boldsymbol{\theta}'\} \in \boldsymbol{\Theta}$ and $\forall\{\boldsymbol{\tau},\boldsymbol{\tau}'\} \in \mathcal{T}$ with Lipschitz constants β_1 and β_2.

Assumption 2 (Bounded Sample Gradient) *We assume the norm of the adaptation risk function's gradient $\nabla\ell(\cdot;\boldsymbol{\theta}_t)$ is bounded:*

$$\sup_{\tau\in\mathcal{T}} ||\nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}^Q_\tau, \mathcal{D}^S_\tau; \boldsymbol{\theta}_t)||_2 < G_t, \tag{6}$$

where G_t is a positive constant.

Assumption 3 (Sub-Gaussian Stochastic Gradient) *The stochastic gradient $\tilde{\boldsymbol{g}} := \boldsymbol{g} + \boldsymbol{\epsilon}$ for the machine learner's adaptation at t-th iteration is σ-sub-Gaussian, which means:*

$$\mathbb{E}\left[\exp\left(\eta\boldsymbol{v}^T\boldsymbol{\epsilon}\right)\right] \leq \exp\left(\frac{\eta^2\sigma^2||\boldsymbol{v}||_2^2}{2}\right) \quad \forall\eta\in\mathbb{R} \text{ and } \boldsymbol{v}\in\mathbb{R}^d, \tag{7}$$

where $\mathbb{E}[\tilde{\boldsymbol{g}}] = \boldsymbol{g}$, $\mathbb{E}[||\tilde{\boldsymbol{g}} - \boldsymbol{g}||_2^2] \leq \sigma^2$ and $\sigma \in \mathbb{R}^+$.

Under the aforementioned assumptions, we derive Theorem 1. Specifically, we define a random variable as the sign of the adaptation risk difference and analyze its evolution following gradient updates across a population. Our theoretical analysis demonstrates that, under a sufficiently small learning rate for the machine learner update, a significant proportion of these sign variables remain largely unchanged in a probabilistic sense. This result establishes a rigorous foundation for evaluating relative task difficulty on $\boldsymbol{\theta}_{t+1}$ based on posterior inference outcomes derived from $\boldsymbol{\theta}_t$ and further guides amortizing the sample average Monte Carlo of CVaR_α optimization objective (see Fig. 1a-b).

Theorem 1 (Provably Approximately Invariant Task Difficulties) *Given arbitrary K data points $\{(\boldsymbol{\tau}_i, \ell(\mathcal{D}^Q_{\tau_i}, \mathcal{D}^S_{\tau_i}; \boldsymbol{\theta}_t)\}_{i=1}^K$, the adaptation gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)$ as a σ-sub-Gaussian random variable and $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)$, we denote the relative difficulty via the difference $\Delta_{ij}(\boldsymbol{\theta}_{t+1}) = \ell(\mathcal{D}^Q_{\tau_i}, \mathcal{D}^S_{\tau_i}; \boldsymbol{\theta}_{t+1}) - \ell(\mathcal{D}^Q_{\tau_j}, \mathcal{D}^S_{\tau_j}; \boldsymbol{\theta}_{t+1})$ and $\Delta_{ij}(\boldsymbol{\theta}_t) = \ell(\mathcal{D}^Q_{\tau_i}, \mathcal{D}^S_{\tau_i}; \boldsymbol{\theta}_t) - \ell(\mathcal{D}^Q_{\tau_j}, \mathcal{D}^S_{\tau_j}; \boldsymbol{\theta}_t)$ between t-th and $(t+1)$-th iterations, and the gradient difference as $\boldsymbol{v}_{ij} := \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}^Q_{\tau_i}, \mathcal{D}^S_{\tau_i}; \boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}^Q_{\tau_j}, \mathcal{D}^S_{\tau_j}; \boldsymbol{\theta}_t)$.*

Under Assumption 1/2/3, the set of rank-preserving variable $E_{ij} := \mathbb{1}\left[\mathit{sign}(\Delta_{ij}(\boldsymbol{\theta}_{t+1})) = \mathit{sign}(\Delta_{ij}(\boldsymbol{\theta}_t))\right]$ satisfies the probability inequality:

$$\mathbb{P}(\bigcap_{i<j} E_{ij}) \geq 1 - \xi,$$

when $\eta \leq \frac{\delta_t}{2G_tM_t + \sqrt{8\sigma^2G_t^2\ln\left(\frac{K(K-1)}{2\xi}\right)}}$ with G_t in Assumption 2, $\delta_t := \min_{i\neq j}|\ell(\mathcal{D}^Q_{\tau_i}, \mathcal{D}^S_{\tau_i}; \boldsymbol{\theta}_t) - \ell(\mathcal{D}^Q_{\tau_j}, \mathcal{D}^S_{\tau_j}; \boldsymbol{\theta}_t)| \in \mathbb{R}^+$, the stochastic gradient norm $M_t := ||\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)||_2$.

5.2 Generative Modeling Risk Functions and Posterior Inference

Here, we design the sampling strategy through the lens of risk landscapes and pay more attention to datasets of learning optimization outcome $\{H_t\}_{t=1}^{T}$. To characterize the adaptation risk during batch optimization, we introduce the latent variable $\boldsymbol{z}_t$ to summarize episodic information and present a versatile deep generative model as:

$$p(H_{0:T}, \boldsymbol{z}_{0:T}|\boldsymbol{\theta}_{0:T}) = p(\boldsymbol{z}_0)\prod_{t=0}^{T} p_{\boldsymbol{\psi}}(H_t|\boldsymbol{z}_t;\boldsymbol{\theta}_t)\prod_{t=0}^{T-1} p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t). \tag{8}$$

Within a Bayesian framework, we approximate the underlying function distribution with the latent variable, and the posterior $p(\boldsymbol{z}_t|H_t)$ summarizes the historical risk information and accounts for uncertainty in distributions. The following writes the form of $p(\boldsymbol{z}_t|H_t)$ according to the Bayes rule[48]:

$$p(\boldsymbol{z}_t|H_t) = \frac{p(H_t|\boldsymbol{z}_t)p(\boldsymbol{z}_t|H_{1:t-1})}{\int p(H_t|\boldsymbol{z}_t)p(\boldsymbol{z}_t|H_{1:t-1})d\boldsymbol{z}_t}, \tag{9}$$

where $p(\boldsymbol{z}_t|H_{1:t-1})$ encodes the past evaluation results as the conditional prior. Moreover, $p(H_t|\boldsymbol{z}_t)$ conveys the likelihood of producing observations of the task batch risk values in the t-th iteration. Notably, the exact computation *w.r.t.* the posterior is intractable due to the complicated integral in the denominator.

Generative Process. As illustrated in Fig. 2c, risk values of the task batch ℓ are correlated with the machine learner's parameters $\boldsymbol{\theta}$. In specific, the factorization of the sequential optimization relevant variables arrives at:

$$p_{\boldsymbol{\psi}}(H_t|H_{1:t-1}) = \int p_{\boldsymbol{\psi}}(H_t|\boldsymbol{z}_t)p(\boldsymbol{z}_t|H_{1:t-1})d\boldsymbol{z}_t = \int \Big[\prod_{i=1}^{\mathcal{B}} p_{\boldsymbol{\psi}}(\ell_{t,i}|\boldsymbol{\tau}_{t,i}, \boldsymbol{z}_t;\boldsymbol{\theta}_t)\Big]p(\boldsymbol{z}_t|H_{1:t-1})d\boldsymbol{z}_t, \tag{10}$$

where $\boldsymbol{z}_t$ in the probabilistic graphical model constitutes the distribution over risk functions (For the sake of simplicity, we skip over other variables less relevant to our learning purposes). Here, we assume the conditional independence between task-specific risk values given $\boldsymbol{z}$ and the machine learner's parameter $\boldsymbol{\theta}$ in Eq. (10). And the *primary optimization objective* is to $\max_{\boldsymbol{\psi}\in\boldsymbol{\Psi}} \ln p_{\boldsymbol{\psi}}(H_t|H_{1:t-1})$ for the optimization outcome prediction.

Inference Process. The manner of episodic training, where the task batch and its evaluation arrive sequentially, inspires us to predict adaptation risk values online to actively sample tasks in a batch. However, the exact inference *w.r.t.* $p(\boldsymbol{z}_t|H_t)$ is infeasible as there is no structural information regarding posteriors. In each iteration, the risk function distribution relies on the updated machine learner $\boldsymbol{\theta}$; hence, such *non-stationarity* in the risk function distributions prompts us to involve the streaming VI[32,33] to derive the approximate posterior.

To do so, we handle the streaming task batches and update the posterior in a recursive way:

$$\underbrace{p(\boldsymbol{z}_t|H_t)}_{\textbf{Updated Posterior}} \propto \underbrace{p(H_t|\boldsymbol{z}_t)}_{\textbf{Likelihood}}\underbrace{p(\boldsymbol{z}_t|H_{1:t-1})}_{\textbf{Functional Prior}} \tag{11}$$

where $p(\boldsymbol{z}_t|H_{1:t-1})$ represents the conditional prior using the last time updated posterior as the proxy. The role of the estimated functional posterior is to provide uncertainty-aware prediction and serves the task sampling strategy design, which will be detailed in Section 5.3.

As a result, we can formulate the evidence lower bound (ELBO) as a tractable optimization objective in Eq. (12) from approximate inference.

$$\max_{\boldsymbol{\psi}\in\boldsymbol{\Psi},\boldsymbol{\phi}\in\boldsymbol{\Phi}} \hat{\mathcal{G}}_{\text{ELBO}}(\boldsymbol{\psi},\boldsymbol{\phi}) := \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|H_t)}\left[\sum_{i=1}^{\mathcal{B}} \ln p_{\boldsymbol{\psi}}(\ell_{t,i}|\boldsymbol{\tau}_{t,i},\boldsymbol{z}_t)\right] - D_{KL}\Big[q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|H_t)\parallel p(\boldsymbol{z}_t|H_{1:t-1})\Big] \tag{12}$$

For implementation convenience, we adopt the parameterized Gaussian distribution with diagonal covariance matrices as variational distributions similar to vanilla VAEs[23,49] and neural processes (NPs)[50]. In other words, these distribution parameters are approximated with neural networks, e.g., $q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|H_t) = \mathcal{N}(\boldsymbol{z}_t;\mu_{\boldsymbol{\phi}}(H_t),\Sigma_{\boldsymbol{\phi}}(H_t))$, and the reparameterization trick[23] is used for stochastic gradient estimate.

Definition 2 (Permutation Invariant Function) *With an n-element permutation group $\mathcal{S}_n$, the operator $g\in\mathcal{S}_n$ maps the order set to itself:*

$$g:[1,2,\dots,n]\mapsto[g_1,g_2,\dots,g_n]. \tag{13}$$

Then the function Φ is called permutation invariant if for any set of data points $\boldsymbol{x}_1,\dots,\boldsymbol{x}_n$, the following condition holds:

$$\Phi(g\circ[\boldsymbol{x}_1,\dots,\boldsymbol{x}_n]) = \Phi([\boldsymbol{x}_{g_1},\dots,\boldsymbol{x}_{g_n}]) = \Phi([\boldsymbol{x}_1,\dots,\boldsymbol{x}_n]) \quad \forall g\in\mathcal{S}_n. \tag{14}$$

As for the neural architecture, we employ the DeepSet encoding module[51] to process the set dataset H_t, which corresponds to the permutation invariant function family in Definition 2. Also, in the context of streaming VI, $q_{\phi}(\boldsymbol{z}_t|H_{t-1})$ mostly works as the proxy for the conditional prior as default. Consequently, we can modify the exact ELBO in Eq. (12) and further translate the practical optimization process with the Lagrange multiplier β into:

$$\max_{\boldsymbol{\psi}\in\boldsymbol{\Psi},\boldsymbol{\phi}\in\boldsymbol{\Phi}} \mathbb{E}_{q_{\phi}(\boldsymbol{z}_t|H_t)}\left[\sum_{i=1}^{\mathcal{B}}\ln p_{\psi}(\ell_{t,i}|\boldsymbol{\tau}_{t,i},\boldsymbol{z}_t)\right] \quad \text{s.t.} \quad D_{KL}\Big[q_{\phi}(\boldsymbol{z}_t|H_t) \parallel q_{\bar{\phi}}(\boldsymbol{z}_t|H_{t-1})\Big] \leq \epsilon \Leftrightarrow \tag{15a}$$

$$\max_{\boldsymbol{\psi}\in\boldsymbol{\Psi},\boldsymbol{\phi}\in\boldsymbol{\Phi}} \mathcal{G}_{\text{ELBO}}(\boldsymbol{\psi},\boldsymbol{\phi}) := \mathbb{E}_{q_{\phi}(\boldsymbol{z}_t|H_t)}\left[\sum_{i=1}^{\mathcal{B}}\ln p_{\psi}(\ell_{t,i}|\boldsymbol{\tau}_{t,i},\boldsymbol{z}_t)\right] - \beta D_{KL}\Big[q_{\phi}(\boldsymbol{z}_t|H_t) \parallel q_{\bar{\phi}}(\boldsymbol{z}_t|H_{t-1})\Big], \tag{15b}$$

where $\bar{\phi}$ indicates no gradients computed through ϕ in the term, and $\{\beta \in \mathbb{R}^+, \epsilon \in \mathbb{R}^+\}$ constrains the machine learner's parameter search in next iteration.

5.3 Task Sampling Strategy Design

In robust active task sampling, existing strategies evaluate task batches to rank their difficulties in adaptation and then prioritize challenging subsets for optimization[14,16,17,39,40]. Besides the expensive evaluation cost, these strategies are weak in the efficient exploration of the task space.

As Theorem 1 has established the theoretical foundation of approximately rank task difficulty, this necessitates the development of the risk learner from cumulated risk histories. With the model predictive results as amortized evaluation, specific rules can be incorporated into the acquisition function for active sampling. Meanwhile, it is fascinating for the risk learner to evaluate the machine learner's adaptation to arbitrarily many tasks with minimal computational cost. Hence, we can easily enlarge the pseudo batch size $\hat{\mathcal{B}}$ for more selection candidates and exploit the epistemic uncertainty from the risk learner, encouraging more exploration in the task space.

Evaluating Adaptation Performance through Stochastic Forward Passes. The risk learner and estimated functional posteriors in Eq. (10)/(12) work as tools for the active selection of the task batch. Specifically, the predictive distribution can be depicted as:

$$\begin{aligned} p_{\psi}(\ell|\boldsymbol{\tau},H_{1:t}) &= \int p_{\psi}(\ell|\boldsymbol{\tau},\boldsymbol{z}_t)p(\boldsymbol{z}_t|H_{1:t})d\boldsymbol{z}_t \triangleq \int p_{\psi}(\ell|\boldsymbol{\tau},\boldsymbol{z}_t)q_{\phi}(\boldsymbol{z}_t|H_t)d\boldsymbol{z}_t \\ &\approx \frac{1}{K}\sum_{k=1}^{K}p_{\psi}(\ell|\boldsymbol{\tau},\boldsymbol{z}_t^{(k)}), \text{ with } \boldsymbol{z}_t^{(k)} \sim q_{\phi}(\boldsymbol{z}_t|H_t)\ \forall\boldsymbol{\tau}\sim p(\tau). \end{aligned} \tag{16}$$

The above predictive distribution $p_{\psi}(\ell|\boldsymbol{\tau},H_{1:t})$ benefits from the Bayesian modeling and provides a tractable way to roughly assess difficulties of tasks throughout the whole task space.

Rank-Flitering the Next Task Batch to Episodically Train. After obtaining $p_{\psi}(\ell|\boldsymbol{\tau},H_{1:t})$, we draw up a batch sampling strategy on the basis of its quantified statistics. The criteria resembles the acquisition function in classical Bayesian optimization (BO), which includes a collection of available evaluation principles, such as expected improvement[52], output information theoretical index[53] or UCB[31].

However, it is also necessary to clarify that the search space is on the sequential task batch instead of machine learners' parameters, which differs from the ultimate purpose in BO. Central to our approach is the principle of optimism in the face of uncertainty[54]. We consider the difficult task's prioritization for robustness and the epistemic uncertainty as pivotal elements in developing acquisition functions. The grounds behind this idea are that (i) the subset with the worst performance deserves extra attention in optimization for adaptation robustness, and (ii) task regions with high predictive uncertainty tend to be underexplored in the last few iterations.

As a result, we present the acquisition function built on the UCB principle[31]:

$$\begin{aligned} \mathcal{A}(\mathcal{T}^{\mathcal{B}};\boldsymbol{\phi},\boldsymbol{\psi}) = \sum_{i=1}^{\mathcal{B}}a(\boldsymbol{\tau}_i) = \sum_{i=1}^{\mathcal{B}}\gamma_0 \overbrace{m(\ell_i)}^{\textbf{Risk Mean}} +\gamma_1 \overbrace{\sigma(\ell_i)}^{\textbf{Epistemic Uncertainty}}&, \text{ where } \boldsymbol{\tau}_i \sim p(\tau) \\ \text{with } m(\ell_i) = \mathbb{E}_{q_{\phi}(\boldsymbol{z}_t|H_t)}\Big[p_{\psi}(\ell|\boldsymbol{\tau}_i,\boldsymbol{z}_t)\Big] \text{ and } \sigma(\ell_i) = \mathbb{V}^{\frac{1}{2}}_{q_{\phi}(\boldsymbol{z}_t|H_t)}\Big[p_{\psi}(\ell|\boldsymbol{\tau}_i,\boldsymbol{z}_t)\Big]&, \end{aligned} \tag{17}$$

where $m(\ell_i)$ and $\sigma(\ell_i)$ are, respectively, the adaptation risk mean and standard deviations, which can be estimated from multiple stochastic forward passes $\boldsymbol{z}_t \sim p(\boldsymbol{z}_t|H_{1:t})$ and $\ell \sim p_{\psi}(\ell|\boldsymbol{\tau}_i,\boldsymbol{z}_t)$ using the risk generative model. And $\{\gamma_0,\gamma_1\}$ are hyperparameters to balance considerations.

Then, the Simulate-Rank-Filter operation in Eq. (2)c arrives at the task batch for $(t+1)$-th iteration, i.e., $\mathcal{T}_{t+1}^{\mathcal{B}} = \arg\max_{\mathcal{T}^{\mathcal{B}} \subseteq \mathcal{T}_{t+1}^{\hat{\mathcal{B}}}:|\mathcal{T}^{\mathcal{B}}|=\mathcal{B}} \mathcal{A}(\mathcal{T}^{\mathcal{B}}; \boldsymbol{\phi}, \boldsymbol{\psi})$. This characterizes the step of the active subset selection from $\mathcal{T}_{t+1}^{\hat{\mathcal{B}}}$, the randomly sampled identifier candidate set with $|\mathcal{T}_{t+1}^{\hat{\mathcal{B}}}| = \hat{\mathcal{B}}$. In an implementation, we still perform random sampling from $p(\tau)$ and forecast the task-wise acquisition score $a(\cdot)$ from the risk learner. Candidates in Top-$\mathcal{B}$ acquisition scores are screened to formulate the task batch $\mathcal{T}_{t+1}^{\mathcal{B}}$ for episodic optimization, as illustrated in Fig. 1d. These steps approximately solve Eq. (2)c and obtain $\mathcal{T}_{t+1}^{\mathcal{B}}$ in a heuristic way.

5.4 Sequentially Optimize the Adaptive Machine Learner

Given the screened $\mathcal{T}_{t+1}$, we execute optimization to update the machine learner's parameters. The task-specific adaptation risk in $(t+1)$-th iteration is written as $\ell_{t+1,i}(\boldsymbol{\theta})$ for the selected task τ_i. The developed MPTS is agnostic to any-shot learning methods, and the following includes the standard update rule for zero-shot, few-shot, and SFT scenarios.

Machine Learner Updates in Zero-Shot Adaptation: The zero-shot setup does not require the support dataset to identify the task. Hence, taking the vanilla DR[55] as an instantiation, we can obtain the update rule as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\lambda}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell(\mathcal{D}^{Q}_{\tau_{t+1,i}}; \boldsymbol{\theta}_t), \tag{18}$$

where $\boldsymbol{\theta}$ denotes the zero-shot learning model parameter with λ the learning rate.

Machine Learner Updates in Few-Shot Adaptation: Still, we take the typical optimization-based method MAML[29] as an instantiation and provide the update rule as follows:

$$\ell_{t+1,i}(\boldsymbol{\theta}) = \ell(\mathcal{D}^{Q}_{\tau_{t+1,i}}; \boldsymbol{\theta}_t^{\text{meta}} - \lambda_{1,1} \nabla_{\boldsymbol{\theta}} \ell(\mathcal{D}^{S}_{\tau_{t+1,i}})) \tag{19a}$$

$$\boldsymbol{\theta}_{t+1}^{\text{meta}} = \boldsymbol{\theta}_t^{\text{meta}} - \frac{\lambda_{1,2}}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell_{t+1,i}(\boldsymbol{\theta}), \quad \forall i \in \{1, \dots, \mathcal{B}\} \tag{19b}$$

where $\boldsymbol{\theta}^{\text{meta}}$ denotes the meta initialization, and $\lambda_{1,1}$ and $\lambda_{1,2}$ are, respectively, learning rates in the inner and outer loops.

Machine Learner Updates in SFT: Here, we take finetuning pretrained models to downstream tasks[56] as an instantiation. In this case, each data point $[\boldsymbol{x}, \boldsymbol{y}]$ can be viewed as a task with either its embedding $\boldsymbol{\tau}$ or $\boldsymbol{x}$ as the task identifier. Then the model update rule can be:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\lambda}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell([\boldsymbol{x}_{t+1,i}, \boldsymbol{y}_{t+1,i}]; \boldsymbol{\theta}_t), \tag{20}$$

where $\{[\boldsymbol{x}_{t+1,i}, \boldsymbol{y}_{t+1,i}]\}_{i=1}^{\mathcal{B}}$ denote the sampled task batch for the $(t+1)$-the iteration.

5.5 Overall Algorithm and Interpretation

Implementation Pipelines. Here, we write the general form of MPTS in **Algorithm** 1, where the past risk episodes are reused to train risk learner and serve the active subset selection. We also provide some implementation examples by putting all the ingredients and optimization recipes together in the zero-shot, few-shot, and SFT scenarios. See Supplementary Notes in **Algorithm** 2-7 for details. Since the first iteration in **Algorithm** 2/4/6 does not involve active sampling, due to no latest history, and the task batch follows the standard random sampling setup.

Algorithm 1: Model Predictive Task Sampling

Input : Task distribution $p(\tau)$; Task batch size $\mathcal{B}$; Candidate batch size $\hat{\mathcal{B}}$; Latest updated $\{\boldsymbol{\psi}, \boldsymbol{\phi}\}$; Latest history H_{t-1}; Iteration number K; Learning rate λ_2.
Output : Selected identifier batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$.

```
// Posterior Inference via Stochastic Gradient Variational Bayes
```
for $i = 1$ **to** K **do**
  Perform gradient updates given H_{t-1}:
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \lambda_2 \nabla_{\boldsymbol{\phi}} \mathcal{G}_{\text{ELBO}}(\boldsymbol{\psi}, \boldsymbol{\phi})$ in Eq. (15b);
    $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \lambda_2 \nabla_{\boldsymbol{\psi}} \mathcal{G}_{\text{ELBO}}(\boldsymbol{\psi}, \boldsymbol{\phi})$ in Eq. (15b);
end
```
// Simulating Zero-shot, Few-shot, Adaptation and SFT Results
```
Randomly sample $\{\hat{\boldsymbol{\tau}}_{t,i}\}_{i=1}^{\hat{\mathcal{B}}}$ from $p(\tau)$;
Run amortized evaluation on candidate tasks $\{\delta_i := \gamma_0 m(\ell_i) + \gamma_1 \sigma(\ell_i)\}_{i=1}^{\hat{\mathcal{B}}}$ in Eq. (17);
```
// Active Subset Selection from Predicted Results
```
Rank $\{\delta_i\}_{i=1}^{\hat{\mathcal{B}}}$ and screen Top-$\mathcal{B}$ values;
Return the screened identifier subset $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$.

Connection with Sequential Decision-making and Control. Intuitively, MPTS resembles model predictive control (MPC)[57] when treating task sampling under some criteria as an optimal planning problem. In this case, the episodic learning process specifies an underlying dynamical system for MPTS to predict with only one future time step in the simulation to assess the influence of selecting the task batch, and the feedback as exact adaptation risk information further helps improve the episodic risk prediction system. In addition, through the lens of sequential decision-making, we can interpret the optimization pipeline of MPTS from the actor-critic framework in RL[58]. In detail, the risk learner works as the critic that predicts adaptation performance in the task τ given a fixed machine learner. Accordingly, the actor plays the role of selecting the task batch from the acquisition function and then executing the machine learner's optimization. These two roles are entangled in the MPTS pipeline to achieve robust yet efficient adaptation.

Acknowledgments and Disclosure of Funding

This work is funded by National Natural Science Foundation of China (NSFC) projects with Numbers # 62306326 and # 62495091.

References

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[4] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[7] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[8] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pages 5637–5664. PMLR, 2021.

[9] Anurag Ajay, Abhishek Gupta, Dibya Ghosh, Sergey Levine, and Pulkit Agrawal. Distributionally adaptive meta reinforcement learning. *Advances in Neural Information Processing Systems*, 35:25856–25869, 2022.

[10] Jiuding Sun, Chantal Shaib, and Byron C Wallace. Evaluating the zero-shot robustness of instruction-tuned language models. In *International Conference on Learning Representations*. ICLR, 2024.

[11] Lexin Zhou, Wout Schellaert, Fernando Martínez-Plumed, Yael Moros-Daval, Cèsar Ferri, and José Hernández-Orallo. Larger and more instructable language models become less reliable. *Nature*, pages 1–8, 2024.

[12] Davis Rempe, Jonah Philion, Leonidas J Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17305–17315, 2022.

[13] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[14] Qi Wang, Yiqin Lv, Zheng Xie, Jincai Huang, et al. A simple yet effective strategy to robustify the meta learning paradigm. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

[16] Talfan Evans, Shreya Pathak, Hamza Merzic, Jonathan Schwarz, Ryutaro Tanno, and Olivier J Henaff. Bad students make great teachers: Active learning accelerates large-scale visual understanding. *arXiv preprint arXiv:2312.05328*, 2023.

[17] Ido Greenberg, Shie Mannor, Gal Chechik, and Eli Meirom. Train hard, fight easy: Robust meta reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[18] Talfan Evans, Nikhil Parthasarathy, Hamza Merzic, and Olivier J Henaff. Data curation via joint example selection further accelerates multimodal learning. *arXiv preprint arXiv:2406.17711*, 2024.

[19] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[20] Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868, 2018.

[21] Wei-Long Zheng, Zhongxuan Wu, Ali Hummos, Guangyu Robert Yang, and Michael M Halassa. Rapid context inference in a thalamocortical model using recurrent neural networks. *Nature Communications*, 15(1):8275, 2024.

[22] Naomi P Friedman and Trevor W Robbins. The role of prefrontal cortex in cognitive control and executive function. *Neuropsychopharmacology*, 47(1):72–89, 2022.

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[24] Mandt Stephan, Matthew D Hoffman, David M Blei, et al. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

[25] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, Giovanni Pezzulo, et al. Active inference and learning. *Neuroscience & Biobehavioral Reviews*, 68:862–879, 2016.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[27] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] Jakub M Tomczak. *Deep Generative Modeling*. Springer Cham, 2024.

[29] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[30] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[31] P Auer. Finite-time analysis of the multiarmed bandit problem, 2002.

[32] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming variational bayes. *Advances in neural information processing systems*, 26, 2013.

[33] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[34] Muhammad Waleed Gondal, Jochen Gast, Inigo Alonso Ruiz, Richard Droste, Tommaso Macri, Suren Kumar, and Luitpold Staudigl. Domain aligned clip for few-shot classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5721–5730, 2024.

[35] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[36] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19113–19122, 2023.

[37] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[38] Vladimir Naumovich Vapnik, Vlamimir Vapnik, et al. Statistical learning theory. 1998.

[39] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. In *International Conference on Learning Representations*, 2022.

[40] Yiqin Lv, Cheems Wang, Dong Liang, and Zheng Xie. Theoretical investigations and practical enhancements on tail task risk minimization in meta learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=McrzOo0hwr.

[41] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2019.

[42] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] Joey Hejna, Chethan Anand Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. Remix: Optimizing data mixtures for large scale imitation learning. In *8th Annual Conference on Robot Learning*, 2024.

[44] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

[45] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15262–15271, 2021.

[46] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021.

[47] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019.

[48] Stephen M Stigler. Thomas bayes's bayesian inference. *Journal of the Royal Statistical Society: Series A (General)*, 145 (2):250–258, 1982.

[49] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[50] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[51] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[52] J Mockus, V Tiesis, and A Zilinskas. The application of bayesian methods for seeking the extremum, vol. 2. *L Dixon and G Szego. Toward Global Optimization*, 2, 1978.

[53] Binxin Ru, Michael A Osborne, Mark McLeod, and Diego Granziol. Fast information-theoretic bayesian optimisation. In *International Conference on Machine Learning*, pages 4384–4392. PMLR, 2018.

[54] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3 (Nov):397–422, 2002.

[55] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[56] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

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

[58] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[59] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607, 2021.

[60] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

[61] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

[62] Yu Yang, Hao Kang, and Baharan Mirzasoleiman. Towards sustainable learning: Coresets for data-efficient deep learning. In *International Conference on Machine Learning*, pages 39314–39330. PMLR, 2023.

[63] Siddharth Joshi and Baharan Mirzasoleiman. Data-efficient contrastive self-supervised learning: Most beneficial examples for supervised learning contribute the least. In *International conference on machine learning*, pages 15356–15370. PMLR, 2023.

[64] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

[65] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[66] James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for bayesian optimization. *Advances in neural information processing systems*, 31, 2018.

[67] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[68] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[69] Qi Wang, Yanghe Feng, Jincai Huang, Yiqin Lv, Zheng Xie, and Xiaoshan Gao. Large-scale generative simulation artificial intelligence: The next hotspot. *The Innovation*, 4(6), 2023.

[70] Wenjia Xu, Yongqin Xian, Jiuniu Wang, Bernt Schiele, and Zeynep Akata. Attribute prototype network for any-shot learning. *International Journal of Computer Vision*, 130(7):1735–1753, 2022.

[71] Shichen Liu, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Generalized zero-shot learning with deep calibration network. *Advances in neural information processing systems*, 31, 2018.

[72] Aoxue Li, Zhiwu Lu, Jiechao Guan, Tao Xiang, Liwei Wang, and Ji-Rong Wen. Transferrable feature and projection learning with class hierarchy for zero-shot learning. *International Journal of Computer Vision*, 128:2810–2827, 2020.

[73] Rohit Keshari, Richa Singh, and Mayank Vatsa. Generalized zero-shot learning via over-complete distribution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13300–13308, 2020.

[74] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5542–5551, 2018.

[75] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8247–8255, 2019.

[76] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[77] Qi Wang, Marco Federici, and Herke van Hoof. Bridge the inference gaps of neural processes via expectation maximization. In *The Eleventh International Conference on Learning Representations*, 2022.

[78] Muhammad Waleed Gondal, Shruti Joshi, Nasim Rahaman, Stefan Bauer, Manuel Wuthrich, and Bernhard Schölkopf. Function contrastive learning of transferable meta-representations. In *International Conference on Machine Learning*, pages 3755–3765. PMLR, 2021.

[79] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International conference on machine learning*, pages 1704–1713. PMLR, 2018.

[80] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

[81] Momin Abbas, Quan Xiao, Lisha Chen, Pin-Yu Chen, and Tianyi Chen. Sharp-maml: Sharpness-aware model-agnostic meta learning. In *International conference on machine learning*, pages 10–32. PMLR, 2022.

[82] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

[83] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[84] Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. Infinite mixture prototypes for few-shot learning. In *International conference on machine learning*, pages 232–241. PMLR, 2019.

[85] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[86] Marcin Sendera, Marcin Przewięźlikowski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. Hypershot: Few-shot learning by kernel hypernetworks. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2469–2478, 2023.

[87] Yonatan Oren, Shiori Sagawa, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust language modeling. *arXiv preprint arXiv:1909.02060*, 2019.

[88] Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. Task-robust model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, 33:18860–18871, 2020.

[89] Jonathan Gordon, Wessel P Bruinsma, Andrew YK Foong, James Requeima, Yann Dubois, and Richard E Turner. Convolutional conditional neural processes. *arXiv preprint arXiv:1910.13556*, 2019.

[90] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

[91] Andrew Foong, Wessel Bruinsma, Jonathan Gordon, Yann Dubois, James Requeima, and Richard Turner. Meta-learning stationary stochastic process prediction with convolutional neural processes. *Advances in Neural Information Processing Systems*, 33:8284–8295, 2020.

[92] Qi Wang and Herke Van Hoof. Learning expressive meta-representations with mixture of expert neural processes. *Advances in neural information processing systems*, 35:26242–26255, 2022.

[93] Philippe Rigollet and Jan-Christian Hütter. High-dimensional statistics. *arXiv preprint arXiv:2310.19244*, 2023.

[94] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[95] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16816–16825, 2022.

[96] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[97] Erin Catto. Box2d: A 2d physics engine for games, 2007. URL http://box2d.org.

[98] Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*, pages 817–828. PMLR, 2018.

[99] Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 1. 2015.

Contents

1 Introduction **1**

2 Adaptation and Robustness **3**

3 Results **4**

3.1 Demonstration of the MPTS's role in `K-shot` sinusoid regression . . . 5

3.2 Few-Shot adaptation benefits from MPTS in robustness and learning efficiency . . . 7

3.3 MPTS retains multi-faced advantages beyond robustness in zero-shot continuous control . . . 8

3.4 MPTS also reserves the potential of robust SFT . . . 8

4 Discussion **8**

5 Methods **11**

5.1 Theoretical Feasibility of Constructing Risk Predictive Models . . . 12

5.2 Generative Modeling Risk Functions and Posterior Inference . . . 13

5.3 Task Sampling Strategy Design . . . 14

5.4 Sequentially Optimize the Adaptive Machine Learner . . . 15

5.5 Overall Algorithm and Interpretation . . . 15

A Quick Guideline to MPTS **25**

B Research Background **27**

B.1 Adaptation Learning for Cross-Task Generalization . . . 27

B.2 Dataset Curation and Task-Level Robustness . . . 27

B.3 Risk Minimization Principles as Baselines . . . 27

C Task Construction and Identifiers **28**

C.1 Tasks with Explicit Identifiers . . . 29

C.2 Tasks with Implicit Identifiers . . . 29

D Auto-Encoding Adaptation Risk through Streaming VI **30**

D.1 Neural Modules to Parameterize Distributions . . . 30

D.2 Formulation of ELBO & Stochastic Gradient Estimates . . . 30

D.3 Theoretical Guarantee for Task Difficulties' Scoring with Posterior Inference . . . 31

E Prompt-based Few-shot Image Classification **33**

F Backbone Methods & Experimental Details in Any-Shot Learning **34**

F.1 MAML . . . 34

F.2 DR . . . 34

F.3 Multi-Modal Prompt Learning . . . 34

G Experimental Setups & Implementation Details **35**
G.1 Sinusoid Regression 35
G.2 `N-way K-shot` Image Classification 36
G.3 Meta-RL 36
G.4 Robotic DR 36
G.5 Prompt-Tuning Multimodal Foundation Models 37

H Computational Tools & Platforms & Data Availability **37**

I Competing Interests & Author Contributions **38**

Supplementary Notes for Model Predictive Task Sampling

Algorithm 2: MPTS for DR (Zero-Shot Scenarios)

Input : Task distribution $p(\tau)$; Task batch size $\mathcal{B}$; Learning rate λ_1.
Output : Adapted machine learner $\boldsymbol{\theta}$.
Set the initial iteration number $t = 1$;
Randomly initialize machine learner $\boldsymbol{\theta}$;
Randomly initialize risk learner $\{\boldsymbol{\psi}, \boldsymbol{\phi}\}$;
while *not converged* **do**
  Execute **Algorithm** 3 to access the batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$ and induced $\{\mathcal{D}_{\tau_{t,i}}^{Q}\}_{i=1}^{\mathcal{B}}$;
  `// Eval Adaptation Performance`
  Compute the task specific adaptation risk $\{\ell_{t,i} := \ell(\mathcal{D}_{\tau_{t,i}}^{Q}; \boldsymbol{\theta}_t)\}_{i=1}^{\mathcal{B}}$;
  Return $H_t = \{[\boldsymbol{\tau}_{t,i}, \ell_{t,i}]\}_{i=1}^{\mathcal{B}}$ as the Input to **Algorithm** 3;
  `// Update Machine Learner`
  Perform batch gradient updates:
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \frac{\lambda_1}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell_{t,i}$;
  Update the iteration number: $t \leftarrow t + 1$;
end

Algorithm 3: Model Predictive Task Sampling

Input : Task distribution $p(\tau)$; Task batch size $\mathcal{B}$; Candidate batch size $\hat{\mathcal{B}}$; Latest updated $\{\boldsymbol{\psi}, \boldsymbol{\phi}\}$; Latest history H_{t-1}; Iteration number K; Learning rate λ_2.
Output : Task identifier batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$.
`// Functional Posterior Inference`
for $i = 1$ **to** K **do**
  Perform gradient updates given H_{t-1}:
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \lambda_2 \nabla_{\boldsymbol{\phi}} \mathcal{G}_{\text{ELBO}}(\boldsymbol{\psi}, \boldsymbol{\phi})$ in Eq. (15b);
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \lambda_2 \nabla_{\boldsymbol{\psi}} \mathcal{G}_{\text{ELBO}}(\boldsymbol{\psi}, \boldsymbol{\phi})$ in Eq. (15b);
end
`// Simulating Adaptation Results`
Randomly sample $\{\hat{\boldsymbol{\tau}}_{t,i}\}_{i=1}^{\hat{\mathcal{B}}}$ from $p(\tau)$;
Run amortized evaluation on candidate tasks $\{\delta_i := \gamma_0 m(\ell_i) + \gamma_1 \sigma(\ell_i)\}_{i=1}^{\hat{\mathcal{B}}}$ in Eq. (17);
`// Active Subset Selection from Predicted Results`
Rank $\{\delta_i\}_{i=1}^{\hat{\mathcal{B}}}$ and screen Top-$\mathcal{B}$ values;
Return the screened task batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$.

Algorithm 4: MPTS for Model Agnostic Meta Learning (Few-Shot Scenarios)

Input : Task distribution $p(\tau)$; Task batch size $\mathcal{B}$; Learning rates: $\{\lambda_{1,1}, \lambda_{1,2}\}$.
Output : Meta-trained initialization $\boldsymbol{\theta}^{\text{meta}}$.
Set the initial iteration number $t = 1$;
Randomly initialize meta learner $\boldsymbol{\theta}^{\text{meta}}$;
Randomly initialize risk learner $\{\boldsymbol{\psi}, \boldsymbol{\phi}\}$;
while *not converged* **do**
  Execute **Algorithm** 5 to access the batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$ and $\{\mathcal{D}_{\tau_{t,i}}^{S} \cup \mathcal{D}_{\tau_{t,i}}^{Q}\}_{i=1}^{\mathcal{B}}$;
  `// Inner Loop to Fast Adapt`
  for $i = 1$ ***to*** K **do**
    Compute the task-specific gradient: $\nabla_{\boldsymbol{\theta}} \ell(\mathcal{D}_{\tau_{t,i}}^{S}; \boldsymbol{\theta})$;
    Perform gradient updates as fast adaptation:
    $\boldsymbol{\theta}_t^i \leftarrow \boldsymbol{\theta}_t^{\text{meta}} - \lambda_{1,1} \nabla_{\boldsymbol{\theta}} \ell(\mathcal{D}_{\tau_i}^{S}; \boldsymbol{\theta})$;
  end
  `// Outer Loop to Meta-train`
  Evaluate fast adaptation performance $\{\ell_{t,i} := \ell(\mathcal{D}_{\tau_{t,i}}^{Q}; \boldsymbol{\theta}_t^i)\}_{i=1}^{\mathcal{B}}$;
  Return $H_t = \{[\boldsymbol{\tau}_{t,i}, \ell_{t,i}]\}_{i=1}^{\mathcal{B}}$ as the Input to **Algorithm** 5;
  Perform meta initialization updates:
  $\boldsymbol{\theta}_{t+1}^{\text{meta}} \leftarrow \boldsymbol{\theta}_t^{\text{meta}} - \frac{\lambda_{1,2}}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell_{t,i}$;
  Update the iteration number: $t \leftarrow t + 1$;
end

Algorithm 5: Model Predictive Task Sampling

Input : Task distribution $p(\tau)$; Task batch size $\mathcal{B}$; Candidate batch size $\hat{\mathcal{B}}$; Latest updated $\{\boldsymbol{\psi}, \boldsymbol{\phi}\}$; Latest history H_{t-1}; Iteration number K; Learning rate λ_2.
Output : Task identifier batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$.
`// Functional Posterior Inference`
for $i = 1$ **to** K **do**
  Perform gradient updates given H_{t-1}:
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \lambda_2 \nabla_{\boldsymbol{\phi}} \mathcal{G}_{\text{ELBO}}(\boldsymbol{\psi}, \boldsymbol{\phi})$ in Eq. (15b);
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \lambda_2 \nabla_{\boldsymbol{\psi}} \mathcal{G}_{\text{ELBO}}(\boldsymbol{\psi}, \boldsymbol{\phi})$ in Eq. (15b);
end
`// Simulating Adaptation Results`
Randomly sample $\{\hat{\boldsymbol{\tau}}_{t,i}\}_{i=1}^{\hat{\mathcal{B}}}$ from $p(\tau)$;
Run amortized evaluation on candidate tasks $\{\delta_i := \gamma_0 m(\ell_i) + \gamma_1 \sigma(\ell_i)\}_{i=1}^{\hat{\mathcal{B}}}$ in Eq. (17);
`// Active Subset Selection from Predicted Results`
Rank $\{\delta_i\}_{i=1}^{\hat{\mathcal{B}}}$ and screen Top-$\mathcal{B}$ values;
Return the screened task batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$.

Algorithm 6: MPTS for Pretrained Model Finetuning

Input : Task distribution $p(\boldsymbol{x})$; Task batch size $\mathcal{B}$; Learning rate λ_1.
Output : Fine-tuned machine learner $\boldsymbol{\theta}$.
Set the initial iteration number $t = 1$;
Randomly initialize machine learner $\boldsymbol{\theta}$;
Randomly initialize risk learner $\{\psi, \phi\}$;
while *not converged* **do**
  Execute **Algorithm** 7 to access the batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$ and $\{[\boldsymbol{x}_{t,i}, \boldsymbol{y}_{t,i}]\}_{i=1}^{\mathcal{B}}$;
  `// Eval Adaptation Performance`
  Compute the instance-specific adaptation risk $\{\ell_{t,i} := \ell([\boldsymbol{x}_{t,i}, \boldsymbol{y}_{t,i}]; \boldsymbol{\theta}_t)\}_{i=1}^{\mathcal{B}}$;
  Return $H_t = \{[\boldsymbol{\tau}_{t,i}, \ell_{t,i}]\}_{i=1}^{\mathcal{B}}$ as the Input to **Algorithm** 7;
  `// Update Machine Learner`
  Perform batch gradient updates:
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \frac{\lambda_1}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell_{t,i}$;
  Update the iteration number: $t \leftarrow t + 1$;
end

Algorithm 7: Model Predictive Task Sampling

Input : Offline processed $\boldsymbol{\tau}$ dataset; Task batch size $\mathcal{B}$; Candidate batch size $\hat{\mathcal{B}}$; Latest updated $\{\psi, \phi\}$; Latest history H_{t-1}; Iteration number K; Learning rate λ_2.
Output : Task identifier batch $\{\boldsymbol{\tau}_{t,i}\}_{i=1}^{\mathcal{B}}$.
`// Functional Posterior Inference`
for $i = 1$ **to** K **do**
  Perform gradient updates given H_{t-1}:
  $\phi \leftarrow \phi + \lambda_2 \nabla_{\phi} \mathcal{G}_{\text{ELBO}}(\psi, \phi)$ in Eq. (15b);
  $\psi \leftarrow \psi + \lambda_2 \nabla_{\psi} \mathcal{G}_{\text{ELBO}}(\psi, \phi)$ in Eq. (15b);
end
`// Simulating Adaptation Results`
Randomly sample $\{\hat{\boldsymbol{\tau}}_{t,i}\}_{i=1}^{\hat{\mathcal{B}}}$ from $p(\tau)$;
Run amortized evaluation on candidate tasks $\{\delta_i := \gamma_0 m(\ell_i) + \gamma_1 \sigma(\ell_i)\}_{i=1}^{\hat{\mathcal{B}}}$ in Eq. (17);
Rank $\{\delta_i\}_{i=1}^{\hat{\mathcal{B}}}$ and screen Top-$\mathcal{B}$ values;
`// Exact Evaluation or Active Annotations`
Return the screened batch $\{[\boldsymbol{x}_{t,i}, \boldsymbol{y}_{t,i}]\}_{i=1}^{\mathcal{B}}$.

A Quick Guideline to MPTS

Task episodic learning serves as a cornerstone in developing adaptive models by structuring diverse, context-rich learning experiences. One of the pivotal insights underpinning this process is the neural scaling law, which establishes a relationship between task volume, model complexity, and computational resources, offering a principled insight into training foundation models at a certain budget. Recent viewpoints have also shed light on the importance of task quality [15,16,18,59–63], prompting innovative data curation strategies to refine datasets for pretraining, meta-training, and post-training. Evidence suggests that carefully curated data can significantly reduce task sampling complexity, decrease computational demands, and enhance robustness against distributional shifts—sometimes achieving these goals simultaneously. Despite these advancements, a practical operation such as Evaluate-Rank-Filter still faces challenges associated with costly evaluations from intensive task queries, computational overhead, and massive annotations. Addressing these bottlenecks remains essential to fully realize the potential of task episodic learning in robust efficient foundation model training.

Computational Complexity Analysis. The involvement of the risk learner inevitably brings extra computational overhead in optimization. However, the risk learner used in this work is lightweight with the model complexity $\mathcal{O}(|\phi| + |\psi|) << \mathcal{O}(|\boldsymbol{\theta}|)$. We can roughly estimate these extra computations that arise from the predictive model as $\mathcal{O}\big((|\phi| + |\psi|)T_{\text{MPTS}}\big)$ throughout the training phase. Moreover, the computational and task evaluation complexities of different methods are estimated in Table 1. Compared with DRM, MPTS retains more computational and task efficiency when the filtering ratio $\hat{\alpha}$ is high, and the machine learner $\boldsymbol{\theta}$ is largely given similar convergence iteration steps.

Table 1: **Computational Complexities using Different Methods.** Here, we drop out the ranking or reweighting computational complexity as the model complexity of the machine learner considered in this analysis is major, such as the multimodal foundation models. T refers to the required iteration steps until the convergence for separate methods.

	ERM	DRM	GDRM	MPTS (Ours)
computation	$\mathcal{O}(\lvert\boldsymbol{\theta}\rvert T_{\text{ERM}})$	$\mathcal{O}(\frac{1}{1-\hat{\alpha}}\lvert\boldsymbol{\theta}\rvert T_{\text{DRM}})$	$\mathcal{O}(\lvert\boldsymbol{\theta}\rvert T_{\text{GDRM}})$	$\mathcal{O}\big((\lvert\phi\rvert + \lvert\psi\rvert + \lvert\boldsymbol{\theta}\rvert)T_{\text{MPTS}}\big)$
task eval	$\mathcal{O}(\mathcal{B}T_{\text{ERM}})$	$\mathcal{O}(\frac{\mathcal{B}}{1-\hat{\alpha}}T_{\text{DRM}})$	$\mathcal{O}(\mathcal{B}T_{\text{GDRM}})$	$\mathcal{O}(\mathcal{B}T_{\text{MPTS}})$

Choice of Surrogate Models. Among MPTS's core components, the risk learner works to predict the adaptation risk values based on historical information and further serves the calculation of acquisition functions. Importantly, this work investigates the feasibility and effectiveness of risk predictive strategies and does not impose rigid constraints on the form of the risk learner $p(\ell|\boldsymbol{\tau}, H_{1:t})$ too much in modeling. The design of this risk learner $p(\ell|\boldsymbol{\tau}, H_{1:t})$ must meet several criteria: it is tractable in optimization, can process historical risk information, and offers uncertainty in prediction.

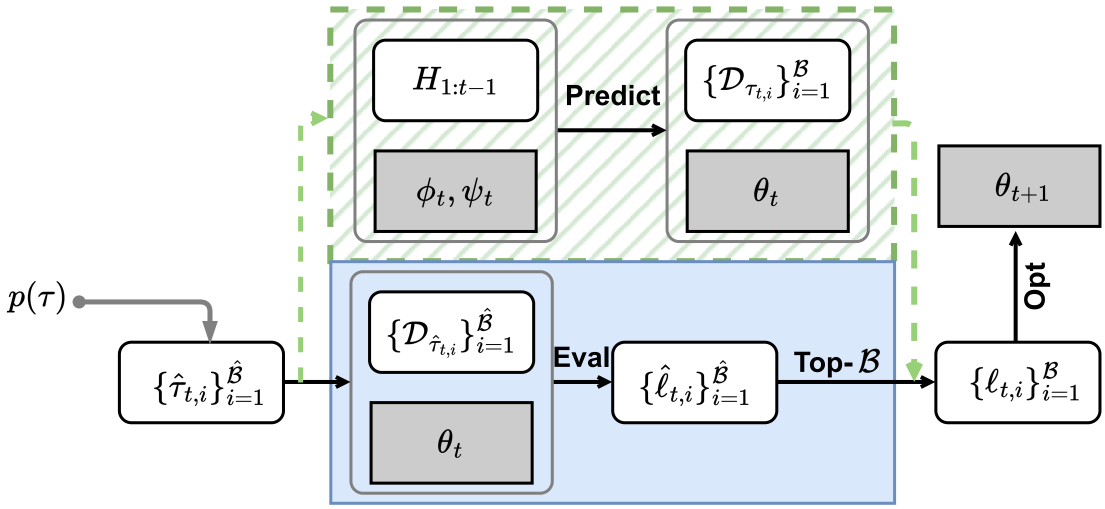

Figure 8: **Risk Predictive Module in MPTS for Active Subset Selection.** MPTS adopts a *predict-then-optimize* strategy and uses a predictive module in green to obtain the preferred task subset. While the traditional method in blue exhausts $\hat{\mathcal{B}}$ tasks in construction and evaluation to filter preferred subset.

A series of candidate probabilistic models exist that probably apply to adaptation risk modeling. One alternative choice can be the Gaussian process[64], which provides an analytical form of the predictive distribution. However, its implementation (i) is less scalable in the case of relatively higher dimensional task identifiers, (ii) holds the cubic runtime complexity in obtaining the predictive covariance matrix, (iii) is sensitive to kernel selection, coupled with limited expressiveness of the Gaussian distribution in learned risk functions. Hence, for simplicity and computational efficiency, we adopt the basic VAE-like model and execute a handful of gradient updates to train the risk learner. We leave more advanced risk learner modeling for future exploration.

Bayesian Optimization for Black-box Functions. This work relates to active sampling and Bayesian optimization. The purpose of BO[65] is to sequentially find a global optimum of a black-box function $f(\boldsymbol{x})$ expensive to evaluate in $\mathcal{S}$, namely $\boldsymbol{x}_* = \arg\max_{\boldsymbol{x}\in\mathcal{S}\subset\mathbb{R}^d} f(\boldsymbol{x})$.

In each iteration $t = 1, \dots, T$, the BO method actively queries $\boldsymbol{x}_t$ to evaluate $f(\boldsymbol{x}_t)$, yeilding an output $\ell_t = f(\boldsymbol{x}_t) + \epsilon$ with a white noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Due to the high cost of function evaluation, the key to BO is constructing a surrogate model to guide the data point to query. The resulting acquisition function[66] works as an active sampling objective to maximize and obtain the candidate $\boldsymbol{x}_t$ based on the previous sequence. BO requires limited function evaluations as observations and exploits the correlations in queried data points. These properties make it more theoretically data efficient than random or grid search in seeking the optimal solution[67]. This work differs from standard BO as task episodic learning is not the optimal parameter search problem.

Specific Pseudo Algorithms in Considered Scenarios. The main paper provides the workflow of MPTS in Algorithm 1. For separate scenarios, we attach detailed pseudo algorithms as follows. These illustrated Algorithms are in the context of supervised learning. Regarding RL scenarios, such as meta RL and DR, there is a slight modification for MPTS. As simply picking up worst-case MDPs restricts the task subspace in optimization[17], we adopt the mixture of the identifier subset from the random sampler and the identifier subset from the MPTS sampler. For example, in meta RL, with the pseudo batch size $\hat{\mathcal{B}} = 1.5\mathcal{B}$, there $1.5\mathcal{B}$ identifier candidates from the random sampler. We retain $0.5\mathcal{B}$ random ones and execute standard MPTS amortized evaluation and acquisition rule to obtain another $0.5\mathcal{B}$ ones from the rest random $\mathcal{B}$ identifiers, formulating the mixed $\mathcal{B}$ task batch for RL training. Such an operation makes RL over the MDP distribution stable in optimization. See the open-source code for more RL details.

B Research Background

B.1 Adaptation Learning for Cross-Task Generalization

Learning from zero-shot or few-shot examples has been identified as a crucial adaptation capability of the machine learner nowadays[68,69]. In SFT, this work treats the individual example as each task to meet MPTS setup. As SFT techniques have been widely discussed in the field[56], we skip this part in the background introduction.

Zero-Shot Adaptation. This assesses the machine learner's generalization capability when directly deploying in unseen scenarios without the help of a support dataset. Such a cross-task generalization is commonly studied in computer vision[70], and the core of the relevant methods is effective semantic representation either from embedding-based methods[71–73] or generative-based methods[74,75]. In the era of the foundation models, the pretraining mechanism between multimodality also sometimes empowers the machine learner, such as CLIP[1], with zero-shot capability. When it comes to sequential decision-making, a commonly seen method is DR[35,55], which places a distribution over environments for the agent to interact.

Few-Shot Adaptation. This examines the machine learner's capability of resolving unseen tasks from some annotated examples as hints. Meta-learning, as the typical learning paradigm, has gained popularity over the past decade. It achieves few-shot adaptation by leveraging past experience and distilling knowledge to unseen but similar scenarios in a few-shot way[76]. In brief, we categorize commonly seen methods into context-based, optimization-based, geometric-based, and others. (i) Formulated in an encoder-decoder structure, the context-based method resembles variational autoencoders and encodes the few-shot information into latent variables or embeddings. Typical ones are neural process families[50,77–79], which aim to constitute exchangeable deep stochastic processes with neural networks. (ii) The optimization-based methods, with their versatile nature and ability to enable cross-task skill transfer, have piqued the interest and engagement of researchers in the field. For example, MAML[29,80–82] reduces meta-learning to a bi-level optimization in the parameter space, and its extensions have been widely investigated in the field. (iii) The deep metric-based methods[83,84] attempt to embed tasks into the latent space and are more suitable for few-shot image classification tasks. Besides, there are other families, such as hyper-networks[85,86], recurrent meta-learning[7], etc.

B.2 Dataset Curation and Task-Level Robustness

Task Curation in Robust Adaptation Learning Pipelines. Recent works[15,16] demonstrate the effectiveness of challenging task prioritization over uniform sampling in improving cross-task generalization and adaptation robustness, particularly when the learning dataset is sufficiently large. Many methods[13–18,39] adopt an Evaluate-Rank-Filter step for iterative model updates, introducing a batch filtering ratio $\hat{\alpha} = 1 - \frac{\mathcal{B}}{\hat{\mathcal{B}}} \in [0, 1)$ to quantify the fraction of discarded tasks in a sample batch. This prioritization of "difficult" tasks aligns with minimizing CVaR_α[30], a robustness metric for tail-case performance. Alternatively, other methods[41–43] focus on constructing uncertainty sets and reweighting tasks within the batch to achieve robust adaptation. Additionally, coreset methods[61–63] aim to select a small subset of tasks that effectively represent the utility of the full dataset, often through gradient approximation in optimization. These approaches address a subproblem of data efficiency, with the acquisition strategy in MPTS serving as an episodic coreset selection mechanism tailored for robustness.

Task Distributional Robustness. The CVaR_α or expected shortfall[30] is a statistical measure to assess the proportional worst-case performance of some models at certain levels. This is widely adopted in risk-averse applications. As implied in Definition 1, CVaR_α describes the expected risk under the normalized $(1 - \alpha)$ proportional tail risk task distribution, and this work specifies the distribution in the task space. Meanwhile, the normalized tail task distribution $p_\alpha(\tau; \boldsymbol{\theta})$ can be viewed as a shifted result from the initial task distribution $p(\tau)$; hence, such a measure provides robustness quantification in the presence of task distribution shifts[14,17,40].

Another indicator to evaluate the machine learner's robustness is the performance in OOD tasks. This refers to the case when the training and the testing task distributions are different. Particularly, in DR and prompt-tuning scenarios, we also use the OOD tasks that never appear in the training task distribution to test the trained policy, and this setup corresponds to domain generalization, a type of substantial distribution shift[8].

B.3 Risk Minimization Principles as Baselines

The risk minimization principles are entangled with task sampling and robust optimization.

Expected/Empirical Risk Minimization (ERM). With the fixed $p(\tau)$, the principle follows the statistical learning theory [38] and minimizes the expectation of adaptation risk over the task space. As a result, we can have:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathbb{E}_{p(\tau)}\Big[\ell(\mathcal{D}_{\tau}^{Q}, \mathcal{D}_{\tau}^{S}; \boldsymbol{\theta})\Big]. \tag{21}$$

It draws batches with a random task sampler to optimize iteratively.

Distributionally Robust Risk Minimization (DRM)[14,16,17,39,40]. We retain the notation of task robust work [14], which terms the tail task risk minimization as DRM. It aims to improve the robustness of adaptation to the tail tasks over iteration. No explicit form exists as the tail task distribution is $\boldsymbol{\theta}$-dependent. The optimization objective is derived as the $\text{CVaR}_{\alpha}(\boldsymbol{\theta})$ [30]:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \text{CVaR}_{\alpha}(\boldsymbol{\theta}) := \mathbb{E}_{p_{\alpha}(\tau;\boldsymbol{\theta})}\Big[\ell(\mathcal{D}_{\tau}^{Q}, \mathcal{D}_{\tau}^{S}; \boldsymbol{\theta})\Big], \tag{22}$$

where we write $p_{\alpha}(\tau;\boldsymbol{\theta})$ to express the $(1-\alpha)$ proportional worst case for easier formulation. In other words, $\mathbb{E}_{p_{\alpha}(\tau;\boldsymbol{\theta})}\Big[\ell(\mathcal{D}_{\tau}^{Q}, \mathcal{D}_{\tau}^{S}; \boldsymbol{\theta})\Big]$ also relates to the task distribution with constraints. Also note that when α approaches 1, the problem degenerates to the worst-case risk minimization.

This work retains the setup in work [14] and picks up the Top-$\mathcal{B}$ in optimization, which corresponds to sample average Monte Carlo of CVaR_{α}. This implies that the actual task batch to evaluate is $\frac{\mathcal{B}}{1-\alpha}$. And for fair comparison with MPTS and light computations, we retain the Monte Carlo estimator for the risk quantile in implementation. To ensure stable training, in all benchmarks, we keep the actual task batch $\hat{\mathcal{B}} = 2\mathcal{B}$ to evaluate and discard the easiest half before the machine learner's optimization.

Group Distributionally Robust Risk Minimization (GDRM)[41]. This can be interpreted as a min-max optimization problem. Such a principle [41] effectively improves robustness in distribution shifts and has shown positive effects on training foundation models [42,43]. It constructs a collection of uncertainty sets over tasks and results in the optimization objective as follows:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sup_{g\in\mathcal{G}} \mathbb{E}_{p_{g}(\tau)}\Big[\ell(\mathcal{D}_{\tau}^{Q}, \mathcal{D}_{\tau}^{S}; \boldsymbol{\theta})\Big], \tag{23}$$

where $\mathcal{G}$ are groups of uncertainty sets, and $p_{g}(\tau)$ indicates the probability measure over the task group. The operation inside Eq. (23) prioritizes the worst group to optimize in a soft way.

GDRM increases the machine learner's robustness by assigning more probability mass to worst cases in a reweighted manner. That means in each iteration with the best selected $p_{\hat{g}}(\tau)$, the optimization problem is reduced to

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathbb{E}_{p_{\hat{g}}(\tau)}\Big[\ell(\mathcal{D}_{\tau}^{Q}, \mathcal{D}_{\tau}^{S}; \boldsymbol{\theta})\Big] = \mathbb{E}_{p(\tau)}\left[\frac{p_{\hat{g}}(\tau)}{p(\tau)}\ell(\mathcal{D}_{\tau}^{Q}, \mathcal{D}_{\tau}^{S}; \boldsymbol{\theta})\right], \tag{24}$$

where we use $\omega(\tau) = \frac{p_{\hat{g}}(\tau)}{p(\tau)}$ to denote the weight.

Given a fixed number of tasks, GDRM will heuristically or dynamically group them into clusters and then perform a reweighting mechanism according to the evaluated risk. In task episodic learning, there is no task grouping operation as the task batch is reset after each iteration. And the default computation of task-specific weights is $\omega(\tau_i) = \frac{\exp(\eta\ell(\mathcal{D}_{\tau_i}^{Q}, \mathcal{D}_{\tau_i}^{S}; \boldsymbol{\theta}))}{\sum_{b=1}^{\mathcal{B}} \exp(\eta\ell(\mathcal{D}_{\tau_b}^{Q}, \mathcal{D}_{\tau_b}^{S}; \boldsymbol{\theta}))}$, where η is the temperature parameter and $\{\boldsymbol{\tau}_b\}_{b=1}^{\mathcal{B}}$ is the identifier of the task batch. The implementation detail can be found in `https://github.com/kohpangwei/group_DRO`.

As revealed in works [14,87,88], the heuristic operation as the Evaluate-Rank-Filter or reweighting mechanism in GDRM is widely adopted for approximate optimization. For example, in task robust meta-learning scenarios, the prerequisite step in DR-MAML [14] is to execute gradient updates in the inner loop for all candidate tasks and then screen the tail task subset to meta-optimize according to the evaluation results.

C Task Construction and Identifiers

Here we refer to the variables that sufficiently configure a task as the task identifier $\boldsymbol{\tau}$. In other literature work, these task identifiers can be viewed as the task representations in a lower dimensional space. To clarify these concepts, we provide more explanations in specific scenarios.

C.1 Tasks with Explicit Identifiers

`K-shot` **Sinusoid Regression.** In this setup[29], meta learners aim at quickly adapting the model to an unseen function $f(x) = a\sin(x-b)$ with the help of K data points randomly sampled from the function. This case treats the amplitude and phase variables (a,b) as the task identifier to configure the task. And the task distribution is induced by the uniform distribution over the task identifier.

Meta Reinforcement Learning. Here, we take the ReacherPos task as an example. The goal of the robot arm is to reach an unobserved target location $[x_1, x_2]$. The end-effector position of the robot arm is initialized randomly, and the step-wise reward corresponds to the feedback to the agent after each move based on its distance to the target location. As the task distribution is specified by a uniform distribution over the target location, $\boldsymbol{\tau} = [x_1, x_2]$ can be viewed as the task identifier. Similarly, we vary physics parameters in simulators to generate diverse MDPs. This constitutes different meta RL benchmarks.

Domain Randomization. DR is a promising paradigm to achieve zero-shot adaptation in unseen scenarios, which is widely adopted in robotics[35] and computer vision[55]. The basic idea is to train the machine learner in a distribution over environments and then directly apply the learned model to new ones.

Table 2: **Benchmarks with Explicit Task Identifiers.** Here, we list the detail information about the task identifier to induce the task distribution.

Benchmarks	Identifier Meaning	Identifier Range
`K-shot` sinusoid regression	amplitude and phase (a,b)	$[0.1, 5.0]\times[0,\pi]$
Meta-RL: HalfCheetahMassVel	mass and velocity (m,v)	$[0.75, 1.25]\times[0, 2.0]$
Meta-RL: HalfCheetahVel	velocity v	$[0, 2.0]$
Meta-RL: ReacherPos	goal location (x_1, x_2)	$[-0.2, 0.2]\times[-0.2, 0.2]$
Meta-RL: Walker2dMassVel	mass and velocity (m,v)	$[0.75, 1.25]\times[0, 2.0]$
Meta-RL: Walker2dVel	velocity v	$[0, 2.0]$
DR: LunarLander	main engine strength s	$[4, 20]$
DR: ErgoReacher	joint damping d and max torque t (×4 joints)	$[0.1, 2.0]\times[2, 20]$

As noted in the main paper, we suppose that the task identifier contains semantics that reflects the difficulty of tasks to resolve and the adaptation risk function is smooth with respect to the identifier. In total, we summarize these bechmarks with explicit task identifiers in Table 2.

C.2 Tasks with Implicit Identifiers

As previously mentioned, we assume the existence of a statistical correlation between task identifiers and adaptation risk values given a specific adaptive machine learner. This implies that the task identifier preserves precise semantics about the task information. These provide the basis for establishing the risk learner from the coupled dataset $\{[\boldsymbol{\tau}_i, \ell_i]\}_{i=1}^{\mathcal{B}}$.

Nevertheless, in several scenarios, it is intractable to access the explicit task identifier. For example, in few-shot image classification, the task information is just the coupled support and query dataset $\mathcal{D}_\tau = \mathcal{D}_\tau^S \cup \mathcal{D}_\tau^Q$. Similarly, in SFT for LLMs, the task can be in the form of the QA pair $\mathcal{D}_\tau = \mathcal{D}_\tau^Q$. There is no explicit representation method, such as $\boldsymbol{\tau}$, for these tasks, which brings difficulty in building up the risk learner. Retaining the prior notation, the episodic task batch can be written as $\hat{H}_t = \left\{\boldsymbol{\theta}_t, \left(\boldsymbol{\tau}_{t,i}, \mathcal{D}_{\tau_{t,i}}, \ell_{t,i}\right)\right\}_{i=1}^{\mathcal{B}}$, where $\boldsymbol{\tau}$ of our interest is unobservable. Some experiments in this work, such as few-shot image classification and SFT, encounter such circumstance.

Task Representation through Identifier Inference. To scale our approach under these circumstances, we propose an alternative candidate schema as the complementary. The probabilistic relationship between variables is depicted in Fig. 2. We consider obtaining the implicit identifier through inference from the task dataset. To do so, we include additional module f_ξ with $\xi \in \Xi$ to embed $\mathcal{D}_\tau^S$ and $\mathcal{D}_\tau^Q$ and further induce a vector $\boldsymbol{\tau} = f_\xi(\mathcal{D}_\tau^S, \mathcal{D}_\tau^Q)$ as the approximate task identifier. These operations imply seeking the appropriate inference module directly influences the risk learner's performance.

Fortunately, there exist pretrained models that enable the task representation to be generalizable to downstream tasks. For example, in the `N-way K-shot` image classification, the task is in the form of support image-label pairs and the query images and the goal is to assign labels to the query images from the support dataset. With the help of CLIP models[1], for a fixed task in the form of $\mathcal{D}_\tau$, we can access a `N` vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ by inputing the set of text-form classes $\{\mathcal{C}_i\}_{i=1}^N$ extracted from the support dataset $\mathcal{D}_\tau^S$, i.e., $\text{CLIP}(\{\mathcal{C}_i\}_{i=1}^N) = [\text{CLIP}_{\text{text}}(\mathcal{C}_1), \dots, \text{CLIP}_{\text{text}}(\mathcal{C}_K)] \coloneqq \boldsymbol{\tau}$. As a result, we can obtain $H_t = \{[\boldsymbol{\tau}_{t,i}, \ell_{t,i}]\}_{i=1}^{\mathcal{B}}$

conditioned on $\boldsymbol{\theta}_t$ for feasible task risk functional prosterior inference. This helps our approach to circumvent the unavailability of exact task identifiers. And it is plausible for the risk predictive model to optimize in learning $p(\ell|\boldsymbol{\tau}, H_{1:t})$. It is worth noting that this case still prefers lightweight models for identifier inference, and the text encoder of CLIP well satisfies this requirement and can be used in the `N-way K-shot` image classification. Details on specific task identifier inference modules can be found in Section E and F.

D Auto-Encoding Adaptation Risk through Streaming VI

Note that the basis of MPTS is to establish the bridge between the task identifier and the adaptation risk value over the course of the machine learner's optimization. In other words, we are seeking a lightweight stochastic risk function in Definition 3 to approximate the posterior $p(\ell|\boldsymbol{\tau}, H_{1:t})$ in the task space.

Definition 3 (Stochastic Risk Function) *Let $\mathfrak{X}$ denote the index set's Cartesian product with the task identifier's dimension $\boldsymbol{\tau} \in \mathbb{N}^d$. For any $k \in \mathbb{N}$ and finite index sequence $\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_k \in \mathfrak{X}$, we write some probability measure over $\mathbb{R}^k$ as $\nu_{(\boldsymbol{\tau}_1,\dots,\boldsymbol{\tau}_k)}$. By introducing the probability space $(\Omega_\tau, \mathcal{F}_{\boldsymbol{\theta}}, \mathcal{P})$ and $\forall \theta \in \Theta$, we can induce a stochastic function $\mathcal{F}_{\boldsymbol{\theta}} : \mathcal{T} \times \Omega_\tau \mapsto \mathbb{R}^k$, so that $\nu_{(\boldsymbol{\tau}_1,\dots,\boldsymbol{\tau}_k)}(C_1 \times \cdots \times C_k) = \mathcal{P}(\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_1) \in C_1, \dots, \mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_k) \in C_k)\ \forall \boldsymbol{\tau}_i \in \mathfrak{X}$ and $C_i \in \mathbb{R}$.*

This section details steps in auto-encoding historical task risk information, parameterizing variational distributions, deriving the approximate optimization objective, and estimating the stochastic gradients of parameters.

D.1 Neural Modules to Parameterize Distributions

Here, we detail the neural modules to parameterize the distributions of interest. For the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|H_t)$ and conditional prior $p(\boldsymbol{z}_t|H_{1:t-1})$, the inputs of the module are a set of task risk pairs. The neural module requires the permutation invariance *w.r.t.* the order of the data points in the set H_t or $H_{1:t-1}$ in Definition 2. Hence, we adopt the DeepSet style neural network[51] to process the collected H_t or $H_{1:t-1}$.

For example, we denote the neural network parameters by $\boldsymbol{\phi} = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_{2,1}, \boldsymbol{\phi}_{2,2}\}$ together with a mean pooling operator $\oplus$, we can have:

$$\boldsymbol{r}_i = h_{\boldsymbol{\phi}_1}(\boldsymbol{\tau}_{k,i}, \ell_{k,i})\ \forall i \in \{1, \dots, \mathcal{B}\}, \quad \bar{\boldsymbol{r}} = \oplus_{i=1}^{\mathcal{B}} \boldsymbol{r}_i, \quad \boldsymbol{\mu}_{\boldsymbol{\phi}} = h_{\boldsymbol{\phi}_{2,1}}(\bar{\boldsymbol{r}}) \text{ and } \boldsymbol{\Sigma}_{\boldsymbol{\phi}} = h_{\boldsymbol{\phi}_{2,2}}(\bar{\boldsymbol{r}}), \tag{25}$$

where the output corresponds to $q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|H_t) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\Sigma}_{\boldsymbol{\phi}})$ (see Fig. 9 for details).

Regarding the task risk functional posterior inference module, this work has a close connection with the NP family[50,77,79,89–92]. Both handle the set data points in probabilistic inference.

D.2 Formulation of ELBO & Stochastic Gradient Estimates

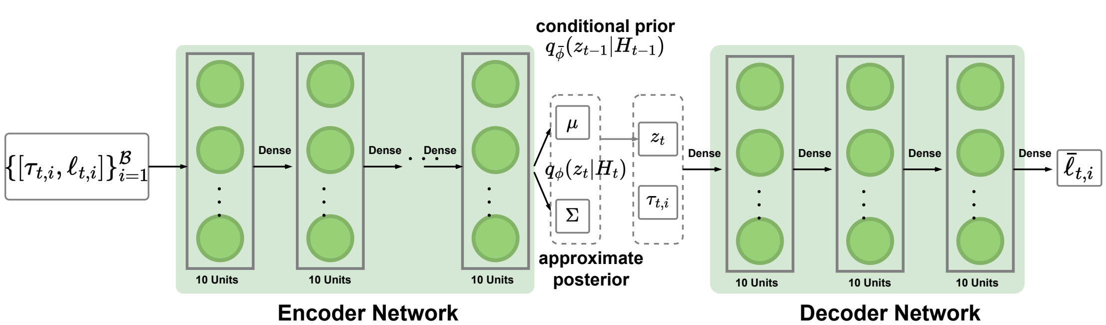

Figure 9: **The Encoder-Decoder Neural Network to Paramterize the Risk Learner.**

Unlike previous risk minimization principles in task episodic learning, ours include an additional risk predictive module, which guides the task batch sampling. Importantly, we use the latent variable to summarize the historical information information and quantify uncertainty in predicting task-specific adaptation risk. The following details the steps.

$$\begin{aligned}
\mathcal{L}_{\text{ML}}(\psi) := \ln p_{\psi}(H_t|H_{1:t-1}) &= \ln\Big[\int p_{\psi}(H_t|z_t)p(z_t|H_{1:t-1})dz_t\Big] && \text{(26a)}\\
&= \ln\Big[\int q_{\phi}(z_t|H_t)\frac{p(z_t|H_{1:t-1})}{q_{\phi}(z_t|H_t)}p_{\psi}(H_t|z_t)dz_t\Big] && \text{(26b)}\\
&\geq \mathbb{E}_{q_{\phi}(z_t|H_t)}\Big[\ln p_{\psi}(H_t|z_t)\Big] - D_{KL}\Big[q_{\phi}(z_t|H_t)\parallel p(z_t|H_{1:t-1})\Big] := \mathcal{G}_{\text{ELBO}}(\psi,\phi) && \text{(26c)}
\end{aligned}$$

Then, we can rewrite the ELBO with the help of reparameterization trick[23] in Eq. (27).

$$\begin{aligned}
\hat{\mathcal{G}}_{\text{ELBO}}(\psi,\phi) &= \mathbb{E}_{q_{\phi}(z_t|H_t)}\Big[\ln p_{\psi}(H_t|z_t)\Big] - D_{KL}\Big[q_{\phi}(z_t|H_t)\parallel p(z_t|H_{1:t-1})\Big] && \text{(27a)}\\
&= \mathbb{E}_{p(\epsilon)}\Big[\ln p_{\psi}(H_t|g_{\phi}(\epsilon,H_t))\Big] - D_{KL}\Big[q_{\phi}(z_t|H_t)\parallel p(z_t|H_{1:t-1})\Big] && \text{(27b)}\\
&\approx \ln p_{\psi}(H_t|g_{\phi}(\epsilon,H_t)) - D_{KL}\Big[q_{\phi}(z_t|H_t)\parallel p(z_t|H_{1:t-1})\Big], \quad \text{with} \quad \epsilon\sim\mathcal{N}(0,I_d) && \text{(27c)}
\end{aligned}$$

Moreover, we estimate the stochastic gradients *w.r.t.* all model parameters based on the reparameterized latent variable distribution.

$$\begin{aligned}
\nabla_{\phi}\mathcal{G}_{\text{ELBO}}(\psi,\phi) &\approx \nabla_{\phi}\ln p_{\psi}(H_t|g_{\phi}(\epsilon,H_t)) - \frac{1}{2}\nabla_{\phi}\Big(\text{Tr}(\hat{\Sigma}^{-1}\Sigma_{\phi}) + (\hat{\mu}-\mu_{\phi})^T\hat{\Sigma}(\hat{\mu}-\mu_{\phi}) - \ln(\det\Sigma_{\phi})\Big) && \text{(28a)}\\
&\text{with} \quad q_{\phi}(z_t|H_t) = \mathcal{N}(\mu_{\phi},\Sigma_{\phi}) \text{ and } p(z_t|H_{1:t-1}) = \mathcal{N}(\hat{\mu},\hat{\Sigma}) && \text{(28b)}\\
\nabla_{\psi}\mathcal{G}_{\text{ELBO}}(\psi,\phi) &\approx \nabla_{\psi}\ln p_{\psi}(H_t|g_{\phi}(\epsilon,H_t)) && \text{(28c)}
\end{aligned}$$

As illustrated in Eq. (28), one stochastic forward pass is required for gradient estimates in the training process. For flexible implementation, we adopt a β-VAE strategy to turn Eq. (27) into

$$\max_{\psi\in\Psi,\phi\in\Phi}\mathcal{G}_{\text{ELBO}}(\psi,\phi) := \mathbb{E}_{q_{\phi}(z_t|H_t)}\left[\sum_{i=1}^{\mathcal{B}}\ln p_{\psi}(\ell_{t,i}|\tau_{t,i},z_t)\right] - \beta D_{KL}\Big[q_{\phi}(z_t|H_t)\parallel q_{\bar{\phi}}(z_t|H_{t-1})\Big] \tag{29}$$

D.3 Theoretical Guarantee for Task Difficulties' Scoring with Posterior Inference

Assumption 1 (Lipschitz Continuity) *We assume the adaptation risk function $\ell(\cdot;\theta)$ reserves the Lipschitz continuity w.r.t. θ and τ, i.e.,*

$$|\ell(\mathcal{D}_{\tau}^{Q},\mathcal{D}_{\tau}^{S};\theta) - \ell(\mathcal{D}_{\tau}^{Q},\mathcal{D}_{\tau}^{S};\theta')| \leq \beta_1\|\theta-\theta'\| \quad \text{and} \quad |\ell(\mathcal{D}_{\tau}^{Q},\mathcal{D}_{\tau}^{S};\theta) - \ell(\mathcal{D}_{\tau'}^{Q},\mathcal{D}_{\tau'}^{S};\theta)| \leq \beta_2\|\tau-\tau'\|, \tag{30}$$

where $\forall\{\theta,\theta'\}\in\Theta$ and $\forall\{\tau,\tau'\}\in\mathcal{T}$ with Lipschitz constants β_1 and β_2.

Assumption 2 (Bounded Sample Gradient) *We assume the norm of the adaptation risk function's gradient $\nabla\ell(\cdot;\theta_t)$ is bounded:*

$$\sup_{\tau\in\mathcal{T}}\|\nabla_{\theta}\ell(\mathcal{D}_{\tau}^{Q},\mathcal{D}_{\tau}^{S};\theta_t)\|_2 < G_t, \tag{31}$$

where G_t is a positive constant.

Assumption 3 (Sub-Gaussian Stochastic Gradient) *The stochastic gradient $\tilde{g} := g + \epsilon$ for the machine learner's adaptation at t-th iteration is σ-sub-Gaussian, which means:*

$$\mathbb{E}\left[\exp\left(\eta v^T\epsilon\right)\right] \leq \exp\left(\frac{\eta^2\sigma^2\|v\|_2^2}{2}\right) \quad \forall\eta\in\mathbb{R} \text{ and } v\in\mathbb{R}^d, \tag{32}$$

where $\mathbb{E}[\tilde{g}] = g$, $\mathbb{E}[\|\tilde{g}-g\|_2^2] \leq \sigma^2$ and $\sigma\in\mathbb{R}^+$.

Given the Assumption 3 and the Chernoff bound[93], we can have the concentration inequality as:

$$\mathbb{P}(\|\tilde{g}-g\|_2 \geq t) \leq 2\exp\left(-\frac{t^2}{2\sigma^2}\right) \quad \forall t\in\mathbb{R}. \tag{33}$$

Theorem 1 (Provably Approximately Invariant Task Difficulties) *Given arbitrary K data points $\{(\boldsymbol{\tau}_i, \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t)\}_{i=1}^K$, the adaptation gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)$ as a σ-sub-Gaussian random variable and $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)$, we denote the relative difficulty via the difference $\Delta_{ij}(\boldsymbol{\theta}_{t+1}) = \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_{t+1}) - \ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_{t+1})$ and $\Delta_{ij}(\boldsymbol{\theta}_t) = \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_t)$ between t-th and $(t+1)$-th iterations, and the gradient difference as $\boldsymbol{v}_{ij} := \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_t)$.*

Under Assumption 1/2/3, the set of rank-preserving variable $E_{ij} := \mathbb{1}[\text{sign}(\Delta_{ij}(\boldsymbol{\theta}_{t+1})) = \text{sign}(\Delta_{ij}(\boldsymbol{\theta}_t))]$ satisfies the probability inequality:

$$\mathbb{P}(\bigcap_{i<j} E_{ij}) \geq 1 - \xi,$$

when $\eta \leq \frac{\delta_t}{2G_t M_t + \sqrt{8\sigma^2 G_t^2 \ln\left(\frac{K(K-1)}{2\xi}\right)}}$ with G_t in Assumption 2, $\delta_t := \min_{i\neq j}|\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_t)| \in \mathbb{R}^+$, the stochastic gradient norm $M_t := ||\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)||_2$.

The purpose of this part is to uncover the mechanism of the risk learner in amortized evaluation of adaptation risk values and scoring the difficulty of tasks. The function of the risk learner relies on Assumptions 1/2/3 and the posterior inference $p(\ell|\tau, H_{1:t}; \boldsymbol{\theta}_t)$ from the historical risk information $H_{1:t}$. The foundation of predicting the outcome of optimization in a rough granularity lies in the Theorem 1, and we detail the proof of such a theorem as below.

①. Any-Shot Adaptation After One-step Gradient Descent.

Here, we consider a set of data points for the risk learner $\{(\boldsymbol{\tau}_i, \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t)\}_{i=1}^K$ under an arbitrary fixed machine learner $\boldsymbol{\theta}_t$, where tasks in the set $\{\tau_i\}_{i=1}^K$ are randomly sampled from $p(\tau)$. Without loss of generality, we can assume that the adaptation risk values satisfy a rank ordering:

$$\ell(\mathcal{D}_{\tau_1}^Q, \mathcal{D}_{\tau_1}^S; \boldsymbol{\theta}_t) \geq \ell(\mathcal{D}_{\tau_2}^Q, \mathcal{D}_{\tau_2}^S; \boldsymbol{\theta}_t) \geq \cdots \geq \ell(\mathcal{D}_{\tau_K}^Q, \mathcal{D}_{\tau_K}^S; \boldsymbol{\theta}_t). \tag{34}$$

The gradient descent as fast adaptation is denoted by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t). \tag{35}$$

After the above operator, we can obtain another set of data points for the updated risk learner $\{(\boldsymbol{\tau}_i, \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_{t+1})\}_{i=1}^K$.

②. Changes of Adaptation Risk Values and Pairwise Ranks.

Based on the Assumption 1, we can perform local approximation over $\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta})$ with the help of first-order Talor expansion *w.r.t.* the $\boldsymbol{\theta}_t$:

$$\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_{t+1}) = \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \eta\nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t)^T\mathcal{L}(\boldsymbol{\theta}_t) + \mathcal{O}(\eta^2||\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)||_2^2) \tag{36a}$$

$$\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_{t+1}) \approx \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t)^T\mathcal{L}(\boldsymbol{\theta}_t) \quad \forall i \in \{1, 2, \ldots, K\}. \tag{36b}$$

One straightforward way to assess the task difficulty is to compare arbitrary paired tasks $\{\boldsymbol{\tau}_i, \boldsymbol{\tau}_j\}$'s adaptation risk values $\{\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t), \ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_t)\}$ with $i < j$. Then, we can estimate the relative difficulty via the difference as:

$$\Delta_{ij}(\boldsymbol{\theta}_{t+1}) \approx \Delta_{ij}(\boldsymbol{\theta}_t) - \eta\left(\nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_t)\right)^T \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t), \tag{37}$$

where we denote the relative difficulty via the difference as $\Delta_{ij}(\boldsymbol{\theta}_{t+1}) = \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_{t+1}) - \ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_{t+1})$ and $\Delta_{ij}(\boldsymbol{\theta}_t) = \ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_t)$ between t-th and $(t+1)$-th iterations. As $\Delta_{ij}(\boldsymbol{\theta}_t)$ is positive, one feasible condition for $\Delta_{ij}(\boldsymbol{\theta}_{t+1}) \in \mathbb{R}^+$ is:

$$\Delta_{ij}(\boldsymbol{\theta}_{t+1}) \approx \Delta_{ij}(\boldsymbol{\theta}_t) - \eta\left(\nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_t)\right)^T \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t) \tag{38a}$$

$$\geq \Delta_{ij}(\boldsymbol{\theta}_t) - 2\eta G_t M > 0, \quad \Rightarrow \eta < \frac{\Delta_{ij}(\boldsymbol{\theta}_t)}{2G_t M_t} \quad \text{with} \quad M_t := ||\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)||_2. \tag{38b}$$

The above implies that when the learning rate η in gradient step is smaller enough, the relative difficulty between the task i and j can be preserved after the machine learner's update with the Assumption 2.

③. Probabilistic Inequality with a Nearly Invariant Ranking Guarantee.

In practice, the stochastic gradient descent is performed, which means the gradient is a random variable $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t) = \boldsymbol{g}_t + \boldsymbol{\epsilon}$ with $\mathbb{E}[\boldsymbol{\epsilon}] = 0, \mathbb{E}[||\boldsymbol{\epsilon}||_2^2] < \sigma^2$ and $\boldsymbol{g}_t = \mathbb{E}[\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)]$. Meanwhile, we denote the gradient difference by $\boldsymbol{v}_{ij} := \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_i}^Q, \mathcal{D}_{\tau_i}^S; \boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau_j}^Q, \mathcal{D}_{\tau_j}^S; \boldsymbol{\theta}_t)$, which leads to:

$$||\boldsymbol{v}_{ij}||_2 \leq 2\sup_{\tau\in\mathcal{T}}||\nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}_{\tau}^Q, \mathcal{D}_{\tau}^S; \boldsymbol{\theta})||_2 < 2G_t, \tag{39}$$

according to the Assumption 2. Another variable is introduced as the minimum separation between arbitrary paired adaptation risk values:

$$\delta_t \coloneqq \min_{i \neq j} |\ell(\mathcal{D}^Q_{\tau_i}, \mathcal{D}^S_{\tau_i}; \boldsymbol{\theta}_t) - \ell(\mathcal{D}^Q_{\tau_j}, \mathcal{D}^S_{\tau_j}; \boldsymbol{\theta}_t)| \in \mathbb{R}^+. \tag{40}$$

Still, to make sure the invariant rank, one necessary condition can be:

$$\Delta_{ij}(\boldsymbol{\theta}_t) - \eta \boldsymbol{v}_{ij}^T \boldsymbol{g}_t \geq 0 \tag{41}$$

And the above inequality reasonably holds when $\eta \boldsymbol{v}_{ij}^T \boldsymbol{g}_t < \delta_t$. Here, we define the random event $E_{ij} \coloneqq \mathbb{1}\left[\text{sign}(\Delta_{ij}(\boldsymbol{\theta}_{t+1})) = \text{sign}(\Delta_{ij}(\boldsymbol{\theta}_t))\right]$ from the task pair together with $E^c_{ij} \coloneqq \mathbb{1}\left[\text{sign}(\Delta_{ij}(\boldsymbol{\theta}_{t+1})) \neq \text{sign}(\Delta_{ij}(\boldsymbol{\theta}_t))\right]$. With the help of σ-sub-Gaussain property in Assumption 3, we can bound the case of the rank flipping as (note some critical conditions that $\boldsymbol{v}_{ij}^T \boldsymbol{g}_t \in \mathbb{R}^+$ and $\eta \boldsymbol{v}_{ij}^T \boldsymbol{g}_t < \delta_t$ as the learning rate η can be typically smaller enough):

$$\mathbb{P}(E^c_{ij}) = \mathbb{P}(\eta \boldsymbol{v}_{ij}^T \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t) \geq \delta_t) \leq \exp\left(-\frac{(\delta_t - \eta \boldsymbol{v}_{ij}^T \boldsymbol{g}_t)^2}{2\eta^2\sigma^2 \|\boldsymbol{v}_{ij}\|_2^2}\right) < \exp\left(-\frac{(\delta_t - 2\eta G_t M_t)^2}{8\eta^2\sigma^2 G_t^2}\right) \tag{42a}$$

$$\mathbb{P}(\bigcup_{i<j} E^c_{ij}) \leq \sum_{i<j} \mathbb{P}(E^c_{ij}) \leq \frac{K(K-1)}{2} \exp\left(-\frac{(\delta_t - 2\eta G_t M_t)^2}{8\eta^2\sigma^2 G_t^2}\right) \tag{42b}$$

$$\mathbb{P}(\bigcap_{i<j} E_{ij}) = 1 - \mathbb{P}(\bigcup_{i<j} E^c_{ij}) \geq 1 - \frac{K(K-1)}{2} \exp\left(-\frac{(\delta_t - 2\eta G_t M_t)^2}{8\eta^2\sigma^2 G_t^2}\right) \geq 1 - \xi. \tag{42c}$$

The condition for the above inequality holds is $\eta \leq \frac{\delta_t}{2G_t M_t + \sqrt{8\sigma^2 G_t^2 \ln\left(\frac{K(K-1)}{2\xi}\right)}}$. With the above **steps ①-③** and corresponding conditions, we complete the proof.

E Prompt-based Few-shot Image Classification

We adopt the standard few-shot image classification setting [29,83], where tasks are constructed using the `N-way K-shot` paradigm for both meta-training and meta-testing. Each task comprises support and query sets. The support set contains `K` examples for each of the `N` classes, while the query set includes `15` examples per class. During meta-training, labels for both support and query data are accessible to the adaptive machine learner. During meta-testing, the query dataset's labels are to be predicted given the labeled support dataset. The class categories of task datasets in the meta-training and meta-testing do not overlap. In experiments, we specifically consider a `5-way 1-shot` image classificationconfiguration. During meta-training, we set the task batch for ERM and DRM as $\mathcal{B} = 4$ (For implementation simplicity, the data loader samples 8 tasks and then randomly keeps half without ranking to optimize). The task batch for DRM is $\mathcal{B} = 8$ before the filtering operation; DRM filters half to optimization. Similarly, that for MPTS is $\mathcal{B} = 8$ and only 4 tasks are screened to optimize.

To enable few-shot learning by prompt-tuning, we integrate the multimodal prompt learning methods MaPLe [36] and prototypical network (ProtoNet) [83]. MaPLe operates on the CLIP model [1], capturing multimodal prompts to refine both visual and textual feature representations with frozen CLIP's parameters. These refined textual features serve as classifiers for predicting refined image features. In parallel, ProtoNet is utilized to derive class-specific visual embeddings from the support set, which assist in distinguishing query samples.

To utilize both MaPLe and ProtoNet, we construct classifiers based on both textual features and visual embeddings. Predictions for query samples are generated by combining the classifiers through a weighted sum. This combination strategy is employed during meta-training to optimize the multimodal prompts. Meanwhile, the CLIP model's parameters remain frozen throughout optimization. During meta-testing, these trained prompts are adopted to create textual and visual classifiers and process query image features. Final predictions for each query sample are made using the same weighted combination approach as in meta-training.

In mathematics, we can characterize the mentioned pipeline as:

$$\text{Textual Classifier from the Textual Features:} \quad \boldsymbol{t}_k = f_{\boldsymbol{\theta}_t}(\boldsymbol{l}_k, \boldsymbol{u}), \tag{43}$$

$$\text{Prototypical Classifier from the Support Image:} \quad \boldsymbol{c}_k = \frac{1}{|S_k|} \sum_{(\boldsymbol{x}_i, y_i) \in S_k} f_{\boldsymbol{\theta}_i}(\boldsymbol{x}_i, \boldsymbol{u}), \tag{44}$$

Classification Likelihood from the Query Dataset:

$$p_{\boldsymbol{\theta}}(y = k | \boldsymbol{x}, \boldsymbol{u}) = \lambda_1 \frac{\exp(-d(f_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{c}_k))}{\sum_{k'} \exp(-d(f_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{c}_{k'}))} + \lambda_2 \frac{\exp(-d(f_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{t}_k))}{\sum_{k'} \exp(-d(f_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{t}_{k'}))}, \tag{45}$$

where $\boldsymbol{\theta}_t$ and $\boldsymbol{\theta}_i$ denote the textual and visual encoders of the CLIP model, respectively. $\boldsymbol{l}_k$ and $\boldsymbol{u}$ respectively denote the textual descriptions of category k and the multimodal prompts. $(\boldsymbol{x}_i, y_i) \in S_k$ denotes the images and labels of category k in the support dataset $\mathcal{D}_\tau^S$. Once the textual classifier $\boldsymbol{t}$ and support visual classifier $\boldsymbol{c}$ are obtained, we predict the query sample $\boldsymbol{x}$ by the classifiers with hyperparameters $\lambda_1 = 0.25$ and $\lambda_2 = 1.0$.

F Backbone Methods & Experimental Details in Any-Shot Learning

F.1 MAML

In sinusoid regression and Meta-RL, MAML is used as the backbone algorithm. As previously discussed, MAML is widely applied in solving few-shot learning tasks. In mathematics, its optimization objective can be characterized as:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{E}_{p(\tau)} \left[\ell(\mathcal{D}_\tau^Q; \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} \ell(\mathcal{D}_\tau^S; \boldsymbol{\theta})) \right], \tag{46}$$

where the term inside the bracket specifies the adaptation risk $\ell(\mathcal{D}_\tau^Q, \mathcal{D}_\tau^S; \boldsymbol{\theta})$, and $\boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} \ell(\mathcal{D}_\tau^S; \boldsymbol{\theta})$ denotes the gradient update with the learning rate λ as fast adaptation to the task $\boldsymbol{\tau}$. After meta-training, we can access the meta initialization $\boldsymbol{\theta}$ that generalizes across the task space.

When it comes to reinforcement learning scenarios, $\mathcal{D}_\tau$ corresponds to episodic returns collected from MDPs with either the meta policy or the fast adapted policy. To ensure enough coverage of task space, we adopt a mixture strategy of MPTS and random sampling as an empirical regularizer in all RL scenarios, which is similar to work [17].

F.2 DR

Robotic DR refers to the setup that trains the agent in a collection of environments to obtain a generalizable policy. The diversity of environments tends to increase the robustness of policies in deployment. Such a setup does not require few-shot episodes in unseen but similar environments. In mathematics, we can express the optimization objective as:

$$\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{J}(\boldsymbol{\theta}) := \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \mathbb{E}_{p(\tau)} \left[\sum_{t=0}^{H} \gamma^t r_t \right] \tag{47}$$

where $p(\tau)$ defines the distribution over MDPs, and $\{r_t\}_{t=0}^H$ is the episodic stepwise reward after interacting with a specific MDP with H as the horizon. Once finishing the optimization of Eq. (47), we can access the policy $\pi_{\boldsymbol{\theta}}$ as the zero-shot decision-maker in new environments. In this case, the adaptation risk can be in the form $\ell(\mathcal{D}_\tau^Q, \mathcal{D}_\tau^S; \boldsymbol{\theta}) = -\sum_{t=0}^H \gamma^t r_t$.

Remember that MDP distribution $p(\tau)$ is mostly induced by physical parameters, e.g., mass, gravity, friction, etc., or the reward functions. In each training iteration, the machine learner resamples a batch of MDPs and gets the shared policy to interact with them to collect episodes. Consequently, the query dataset contains the episodes collected with no support dataset. Overall, policy optimization follows the standard TD3 algorithm [37] due to its sample efficiency and stability.

F.3 Multi-Modal Prompt Learning

Multi-modal prompt learning is based on the backbone of the prompt tuning method MaPLe [36], which we use on both few-shot and SFT for image classifications.

Few-shot classification. The few-shot prompt learning refers to the common few-shot classification setting [29,83]. We integrate the MaPLe backbone with the ProtoNet [83] to fully utilize the support sets in few-shot learning. As illustrated in Section E, we generate the model prediction using both the textual classifiers from the CLIP textual encoder and the visual classifiers from the support set. By freezing the CLIP model parameters, only the prompts are optimized during meta-training. In mathematics, the optimization objective can be formulated as:

$$\max_{\boldsymbol{u}} \mathbb{E}_{p(\tau)} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_\tau^Q} \left[\log p_{\boldsymbol{\theta}}(y|\boldsymbol{x}, \boldsymbol{u}) \right], \tag{48}$$

where $\boldsymbol{u}$ denotes the learnable multi-modal prompts. $\boldsymbol{\theta} = (\boldsymbol{\theta}_t, \boldsymbol{\theta}_i)$ contains the parameters of the CLIP textual and visual encoders, which are frozen during training. The prediction of each query image $\boldsymbol{x}$ from task τ is calculated by Eq. (45). Loglikelihood maximization is implemented by minimizing the classification cross-entropy loss.

SFT for image classification. The prompt learning setting refers to the 16-shot classification task proposed in work[94,95]. Based on the MaPLe backbone[36], we again freeze the CLIP model parameters and tune multi-modal prompts. The prompts are optimized on a selected ImageNet subset, with 16 samples from each category.

Note that in the SFT setting, we do not have pre-defined `N-way K-shot` tasks, either the splits of support and query sets in each task. Therefore, we replace the "tasks" in the meta-learning setting with training samples. Model predictive task sampling is then achieved through data sampling. In mathematics, the objective can be formulated as:

$$\max_{\boldsymbol{u}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \left[\log p_{\boldsymbol{\theta}}(y|\boldsymbol{x}, \boldsymbol{u})\right], \tag{49}$$

where $\boldsymbol{u}$ and $\boldsymbol{\theta}$ denote the prompts and frozen CLIP parameters as in the few-shot prompt learning setting, respectively. $\boldsymbol{x}$ are training samples from the entire training set $\mathcal{D}$. The prediction of each image $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}, \boldsymbol{u})$ is calculated by $\frac{\exp(-d(f_{\boldsymbol{\theta}_i}(\boldsymbol{x},\boldsymbol{u}),\boldsymbol{t}_k))}{\sum_{k'}\exp(-d(f_{\boldsymbol{\theta}_i}(\boldsymbol{x},\boldsymbol{u}),\boldsymbol{t}_{k'}))}$ with the textual classifiers $\boldsymbol{t}$, obtained similarly to Eq. (43).

G Experimental Setups & Implementation Details

Practical Learning Efficiency and Robustness. Widely recognized in reinforcement learning is the high sample complexity in policy evaluation, which demands massive interactions with environments, while policy optimization over the MDP distribution makes this even more severe. In `N-way K-shot` image classification, we can create `K-shot` classification task from an arbitrary combination of N classes; then the task space complexity $\mathcal{O}\left(C_M^N\right)$ grows with the number of categories M in image datasets. Meanwhile, challenges arise when gradient updates of foundation models consume substantial computational power and memory with a large batch size. Similar circumstance also occurs in robust finetuning foundation models.

Table 3: **A Summary of the Considered Benchmarks.** Here, we list the primary expensive part in task episodic learning for each scenario together with backbone methods. Also note that `N-way K-shot` image classification and SFT requires implicit task identifiers while others can directly access explicit task identifiers as the lower dimensional task representation.

Benchmarks	Adaptation	Backbone	Expensive Part
`K-shot` sinusoid regression	few-shot	MAML	computations
`N-way K-shot` image classification	few-shot	MaPLe	computation/memory
Meta-RL	few-shot	MAML	interactions
Robotic DR	zero-shot	TD3	interactions
SFT	many-shot	MaPLe	computation/memory

Neural architecture of the risk learner. As mentioned in the main paper, the risk learner is in an encoder-decoder structure. For generality sake, we keep the neural architecture same for all benchmarks, including regression, classification and reinforcement learning. The encoder includes an embedding network with 4 hidden layers of size 128 (for Image Classification and Prompt-Tuning) or 10 (for Sinusoid Regression, Meta-RL, and Robotic DR) with the Rectified Linear Unit (ReLU) nonlinear activation units to encode $\{[\boldsymbol{\tau}_{t,i}, \ell_{t,i}]\}$ batch for mean pooling and then maps to $[\boldsymbol{\mu}, \boldsymbol{\Sigma}]$ with an output layer. The decoder is a network with 3 hidden layers with nonlinear activation units to map $[\boldsymbol{z}, \boldsymbol{\tau}]$ to $\ell \in \mathbb{R}$. For further details, please refer to our code.

Visualized Results during Training Phases. Note that the active selection and the random sampler with different batches affect the reflection of the machine learner's performance. Hence, learning curves in sinusoid regression (Fig. 3.a), Meta-RL (Fig. 5.a-b), and DR (Fig. 6.a-b) are actually evaluated in a uniformly sampled validation task dataset for fair comparison. Details on these validation task dataset are attached in the opensourced code.

G.1 Sinusoid Regression

Task setup. For sinusoid regression, we retain the setup in MAML[29], where the few-shot machine learner tries to complete a wave function with the support dataset. In specific, sampling the amplitude a and the phase b configures the wave function, and 10 data points are uniformly sampled from the interval $[-5.0, 5.0]$ coupled with $y = a\sin(x - b)$ to obtain the support dataset. This formulates the $10-$`shot` sinusoid regression task.

Meta training process and neural architectures. The machine learner is a neural network with 2 hidden layers of size 40 with two nonlinear activation units. The task batch for ERM and GDRM is 16, and that for DRM is 32 as default. The temperature parameter in GDRM is $\eta = 0.001$. The learning rates for the inner loop and the outer loop are 0.001. The following is about extra optimization details or setups in MPTS. The task identifier's dimension is 2 with the latent variable is $\boldsymbol{z} \in \mathbb{R}^x$. The batch size of the identifier in training is 32, the Lagrange multiplier is set as 1, and we use the Adam optimizer with the learning rate $3e-4$ to update the risk learner for 20000 step. In sinusoid regression and Meta-RL, we use the standard repository in MAML[29].

G.2 `N-way K-shot` Image Classification

Task Setup. This is a commonly seen benchmark in few-shot learning. It learns a model that can classify images from N distinct classes with support of K labeled examples for each class. The support dataset as reference is in the form $\mathcal{D}_\tau^S = \{\{[\boldsymbol{x}_{i,k}, y_{i,k} = i]\}_{k=1}^K\}_{i=1}^N$. And the query dataset corresponds to the image information for the model to classify. Hence, for a large image dataset with M classes, the complexity of the task space is $\mathcal{O}\Big(C_M^N\Big)$. Here, we include ImageNet-CG[44], ImageNet-CI[44], ImageNet-CS[44], ImageNet-A[45], ImageNet-S[27] and ImageNet-R[46] as the dataset in evaluation.

Meta training process and neural architectures. Explicit $\boldsymbol{\tau}$ are unavailable to specify the task; however, it can be approximately resolved by describing the identifier through a small reference model. Specifically, we leverage CLIP's text encoder to obtain $\boldsymbol{\tau} \approx [\text{CLIP}_{\text{text}}(\mathcal{C}_1), \dots, \text{CLIP}_{\text{text}}(\mathcal{C}_K)]$ with the tokenizations of K class texts $\mathcal{C}_{1:K}$ from $\mathcal{D}_\tau^S$. The machine learner utilizes a prompt learning backbone following MaPLe, with the frozen CLIP model. The task batch for ERM and GDRM is 4, and that for DRM is 8 as default. The temperature parameter in GDRM is 0.001. The learning rate for the outer loop is 0.01. The learning rate for the inner loop follows that in MaPLe[36]. The following is about extra optimization details or setups in MPTS. The task identifier is generated by the frozen CLIP text encoder using the input class names, with a dimensionality of 512. The batch size of the identifier in training is 8, the Lagrange multiplier is set as β, and we use the Adam optimizer with the learning rate 0.01 to update the risk learner.

G.3 Meta-RL

Task Setup. We construct MDP distributions based on Mujoco physics engines[96]. These include HalfCheetahVel, HalfCheetahMassVel, Walker2dVel, Walker2dMassVel, and ReacherPos. The HalfCheetahVel and Walker2dVel tasks involve training the cheetah or walker robot to achieve a target velocity. These tasks define the reward function as the negative absolute difference between the robot's current velocity and the target velocity, supplemented by a control penalty and an alive bonus to facilitate the learning process. The goals and rewards of HalfCheetahMassVel and Walker2dMassVel are the same as those of the corresponding velocity-related tasks, with the additional identifier of varying mass for the cheetah or walker robot. The ReacherPos task tries to move a two-jointed robot arm's end effector close to a target position, and its reward function is defined as the negative L-1 distance between the robot arm's position and the target position, supplemented by a control cost to ensure robustness.

Meta training process and neural architectures. The machine learner is a neural network with 2 hidden layers of size 64 with the Rectified Linear Unit (ReLU) nonlinear activation units. The task batch for ERM and GDRM is 20, and that for DRM is 40 as default. The temperature parameter in GDRM is 0.001. The learning rates for the inner loop and the outer loop are 0.1. The following is about extra optimization details or setups in MPTS. The task identifier is encoded into the latent variable $\boldsymbol{z} \in \mathbb{R}^x$. The batch size of the identifier in training is 30, the Lagrange multiplier is set as $\beta = 0.0001$, and we use the Adam optimizer with the learning rate 0.005 to update the risk learner.

G.4 Robotic DR

Task Setup. We conduct experiments on LunarLander-v2 and ErgoReacher-v0 environments[35]. LunarLander is a 2 degrees of freedom (DoF) environment in which the agent has to land a spacecraft on a designated landing pad without crashing, implemented using Box2D[97]. The reward function of LunarLander awards positive rewards for successful landings, negative rewards for crashes, and additional penalties for fuel consumption and deviation from the landing pad, encouraging efficient and controlled landings. ErgoReacher is a 4 DoF arm environment from[98] in which the arm has to touch a goal with its end effector, implemented in the Bullet Physics Engine[99]. The reward function of ErgoReacher includes the negative distance between the end effector's position and the target, along with other control costs to promote efficient and safe movements. In LunarLander, we randomize the engine strength, while in ErgoReacher, we randomize the joint damping and maximum torque for each of the 4 joints, resulting in a total of 8 parameters. The detailed ranges of the randomized parameters for each environment are provided in Table 2.

DR training process and neural architectures. The machine learner is a neural network with 2 hidden layers of size 10 with the Rectified Linear Unit (ReLU) nonlinear activation units. The task batch for ERM, and GDRM is 10, and that for DRM is 20 as default. The temperature parameter in GDRM is 0.01. The learning rates for actor and critic are 3e-4. The following is about extra optimization details or setups in MPTS. The task identifier is encoded into the latent variable z. The batch size of the identifier in training is 25 for LunarLander and 250 for ErgoReacher. The Lagrange multiplier is set as $\beta = 1.0$, and we use the Adam optimizer with the learning rate 0.005 to update the risk learner.

G.5 Prompt-Tuning Multimodal Foundation Models

Task Setup of Prompt-tuning. We refer the reader to MaPLe's implementation in `https://github.com/muzairkhattak/multimodal-prompt-learning`. For all baselines, we retain the MaPLe's task construction in prompt-tuning.

Prompt-tuning process and neural architectures. The machine learner follows the prompt learning method MaPLe [36] based on the frozen CLIP model (ViT/B-16). The task batch for ERM and GDRM is 4, and that for DRM is 8 as default. The temperature parameter in GDRM is 0.001. The learning rate for the outer loop is 0.005. The learning rate for the inner loop follows that in MaPLe [36]. The following is about extra optimization details or setups in MPTS. As for the neural architecture of the risk learner, the encoder is a neural network with 5 hidden layers with 4 ReLU nonlinear activation units, and the decoder is a neural network with 4 hidden layers with 3 nonlinear activation units. The task identifier's dimension is 512 with the latent embedding from CLIP encoders. The batch size of the identifier in training is 8, the Lagrange multiplier is set as β, and we use the Adam optimizer with the learning rate 0.005 to update the risk learner for 8000 steps. During prompt-tuning, we set the task batch for ERM and DRM as $\mathcal{B} = 4$ (For implementation simplicity, the data loader samples 8 tasks and then randomly keeps half without ranking to optimize). The task batch for DRM is $\mathcal{B} = 8$ before the filtering operation; DRM filters half to optimization. Similarly, that for MPTS is $\mathcal{B} = 8$ and only 4 tasks are screened to optimize.

Table 4: **Testing Classification Results after `5-way 1-shot` Meta-Training on Various Datasets.** We report testing $\text{CVaR}_{0.9}$, $\text{CVaR}_{0.7}$, $\text{CVaR}_{0.5}$ and average accuracies with the meta-trained machine learner on ID and OOD datasets. This table complements the radar part of Fig. 4**a-d**. Best results are in bold, and MPTS's performance gains over ERM are marked in blue.

Dataset	Metrics	ERM	DRM	GDRM	MPTS (Ours)	
ImageNet-CG [44]	$\text{CVaR}_{0.9}$	77.02	77.76	77.61	**78.04**	+1.02
	$\text{CVaR}_{0.7}$	82.00	82.47	82.62	**82.87**	+0.87
	$\text{CVaR}_{0.5}$	84.77	85.03	85.16	**85.45**	+0.68
	Average	89.04	89.46	89.51	**89.87**	+0.83
ImageNet-CI [44]	$\text{CVaR}_{0.9}$	80.24	80.47	80.15	**80.97**	+0.73
	$\text{CVaR}_{0.7}$	84.77	85.17	84.78	**85.66**	+0.98
	$\text{CVaR}_{0.5}$	87.03	87.52	87.09	**87.78**	+0.75
	Average	91.12	91.46	91.15	**91.60**	+0.48
ImageNet-CS [44]	$\text{CVaR}_{0.9}$	76.63	77.71	76.18	**78.4**	+1.77
	$\text{CVaR}_{0.7}$	81.58	82.53	81.40	**83.27**	+1.69
	$\text{CVaR}_{0.5}$	84.40	85.08	84.23	**85.83**	+1.43
	Average	89.24	89.87	89.13	**90.26**	+1.02
ImageNet-A [45]	$\text{CVaR}_{0.9}$	76.69	77.39	77.40	**77.95**	+1.26
	$\text{CVaR}_{0.7}$	81.90	82.57	82.50	**83.41**	+1.51
	$\text{CVaR}_{0.5}$	84.58	85.43	85.33	**86.30**	+1.72
	Average	89.25	90.26	90.21	**91.06**	+1.81
ImageNet-S [27]	$\text{CVaR}_{0.9}$	82.63	83.67	83.11	**84.82**	+2.21
	$\text{CVaR}_{0.7}$	87.26	88.38	87.26	**89.27**	+2.01
	$\text{CVaR}_{0.5}$	89.56	90.02	90.08	**91.54**	+1.98
	Average	93.53	94.27	94.12	**94.78**	+1.25
ImageNet-R [46]	$\text{CVaR}_{0.9}$	88.31	88.84	87.71	**89.79**	+1.48
	$\text{CVaR}_{0.7}$	91.46	92.03	91.18	**93.16**	+1.7
	$\text{CVaR}_{0.5}$	93.36	93.80	93.21	**94.65**	+1.29
	Average	96.05	96.33	95.98	**96.86**	+0.81

H Computational Tools & Platforms & Data Availability

In this research project, we use the Pytorch as the package to implement all methods to run all deep learning epxeriments.

Table 5: **Testing Classification Results after Prompt-Tuning on ImageNet.** We report testing $\mathrm{CVaR}_{0.9}$, $\mathrm{CVaR}_{0.7}$, $\mathrm{CVaR}_{0.5}$ and average accuracies with the prompt-tuned machine learner on ID and OOD datasets. Evaluation on OOD datasets corresponds to the domain generalization setting. This table complements the radar part of Fig. 7**a-d**. Best results are in bold, and MPTS's performance gains over ERM are marked in blue.

Dataset	Metrics	ERM	DRM	GDRM	MPTS (Ours)	
ImageNet [26] (ID)	$\mathrm{CVaR}_{0.9}$	31.68	32.46	31.38	**32.5**	+0.82
	$\mathrm{CVaR}_{0.7}$	42.87	44.07	42.97	**44.22**	+1.35
	$\mathrm{CVaR}_{0.5}$	51.45	52.59	51.71	**52.72**	+1.27
	Average	70.8	70.8	71.0	**71.20**	+0.4
ImageNet-A [45] (OOD)	$\mathrm{CVaR}_{0.9}$	15.33	15.46	15.6	**18.44**	+3.11
	$\mathrm{CVaR}_{0.7}$	22.8	23.02	23.13	**24.06**	+1.22
	$\mathrm{CVaR}_{0.5}$	30.08	29.54	30.26	**31.25**	+1.17
	Average	49.8	48.4	49.5	**51.10**	+1.3
ImageNet-R [46] (OOD)	$\mathrm{CVaR}_{0.9}$	26	28.06	25.9	**28.19**	+2.19
	$\mathrm{CVaR}_{0.7}$	43.58	45.23	43.91	**45.49**	+1.91
	$\mathrm{CVaR}_{0.5}$	56.67	58.21	57.38	**58.77**	+2.1
	Average	76.9	77.4	77.3	**77.63**	+0.73
ImageNet-S [27] (OOD)	$\mathrm{CVaR}_{0.9}$	12.24	13.16	12.22	**13.63**	+1.39
	$\mathrm{CVaR}_{0.7}$	20.02	21.08	20.43	**21.46**	+1.44
	$\mathrm{CVaR}_{0.5}$	26.69	27.44	27.4	**27.97**	+1.28
	Average	48.9	48.8	48.9	**49.63**	+0.73
ImageNet-V [47] (OOD)	$\mathrm{CVaR}_{0.9}$	24.6	25.5	24.2	**25.90**	+1.3
	$\mathrm{CVaR}_{0.7}$	34.7	35.73	35.07	**35.80**	+1.1
	$\mathrm{CVaR}_{0.5}$	43.45	44.14	43.53	**44.56**	+1.11
	Average	64.0	63.8	64.1	**64.53**	+0.53

I Competing Interests & Author Contributions

The author list is Qi Cheems Wang (Q.C.W.), Zehao Xiao (Z.X.), Yixiu Mao (Y.M.), Yun Qu (Y.Q.), Jiayi Shen (J.S.), Yiqin Lv (Y.L.), and Xiangyang Ji (X.J.). The authors declare no competing interests in this work. X.J. launched and sponsored this research on the reliable and efficient adaptation learning project. Z.X. and J.S. from the University of Amsterdam attended this project, and this work was done during their remote visiting the Tsinghua University Intelligent Decision-Making Lab from April 2024 to August 2024. J.S. is now working at Facebook AI Research. The authors confirm their contributions to this work as follows:

Under the supervision of professor X.J., Q.C.W. conceptualized the idea of MPTS, designed the computational framework, formulated the mathematical part, and wrote the draft. Z.X. and J.S. implemented MPTS in sinusoid regression, prompt-based few-shot image classification, and SFT image classification, collected experimental results to visualize, and added implementation details in Supplementary Material. Y.M., Y.Q., and Y.L. implemented MPTS in sinusoid regression, Meta-RL and robotic DR, collected experimental results to visualize, and added implementation details in Supplementary Material. X.J. supervised the progress of MPTS, organized technical discussions, reviewed and revised the original draft. All authors have read the manuscript and approved the public version.

First Author Biography: Qi Wang received Ph.D. degree under supervision of Professor Max Welling and Associate Professor Herke van Hoof in 2022. He is now under supervision of Prof. Xiangyang Ji and works as a research assistant at Tsinghua University. His research focus is on generative modeling and intelligent decision-making. He has published several papers on top-tier conferences such as ICML/NeurIPS/ICLR and was awarded 2023 China Multi-Agent System Outstanding Doctoral Thesis Award.

Correspondence Author Biography: Xiangyang Ji received the B.S. degree in materials science and the M.S. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1999 and 2001, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 2008. He joined Tsinghua University, Beijing, in 2008, where he is currently a Professor with the Department of Automation, School of Information Science and Technology. He has authored more than 100 refereed conference and journal papers. His current research interests include signal processing, computer vision, computational photography, and intelligent decision-making.