



e l l i s

INSTITUTE
FINLAND



ICL^T · AALTO
INTELLIGENT CONVERGENCE LAB
Learning for Credible Intelligence

A? Aalto University
School of Science

Learning from the Past Without Being Trapped by It

Towards AI that continues to learn and adapt

Qi Chen

Principal Investigator · ELLIS Institute Finland
Assistant Professor · Aalto CS

Espoo · 2026-10

Introduction

reusing knowledge while staying able to learn

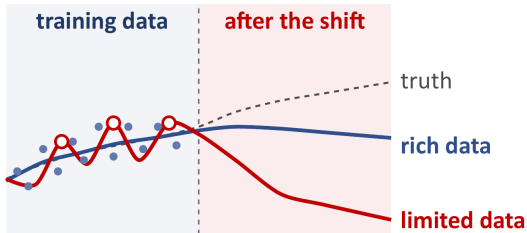
Central questions:

How does past knowledge help us learn?

When can it make adaptation harder?

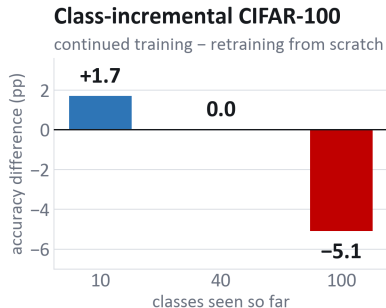
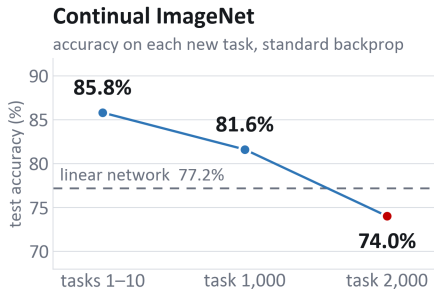
Past knowledge helps us learn from fewer examples

- ◇ A few examples may fit **many different explanations**
- ◇ Relevant past knowledge helps us **choose what generalizes**
- ◇ We reuse it through **priors, pretrained models, and experience from earlier tasks**



⇒ **We need less new data when we can reuse what we already know.**

Continual training can lead to loss of plasticity



◇ Same pattern in **RL**: as an ant robot's friction keeps changing, standard PPO's reward collapses

⇒ **More training does not always mean a greater capacity to learn.**

Dohare, Hernandez-Garcia, Lan, Rahman, Mahmood, Sutton. Loss of plasticity in deep continual learning. Nature 632, 2024

Fundamentals

why the past can trap you

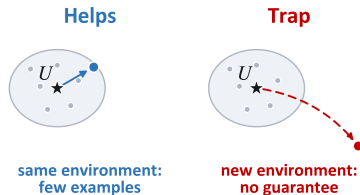
Central questions:

When does prior knowledge help, and when does it hold you back?

Can more data fix it?

Meta-learning captures shared task structure

- ◇ A meta-learner extracts **shared knowledge** U from n earlier tasks, m examples each
- ◇ The analysis considers new tasks from the **same task environment**



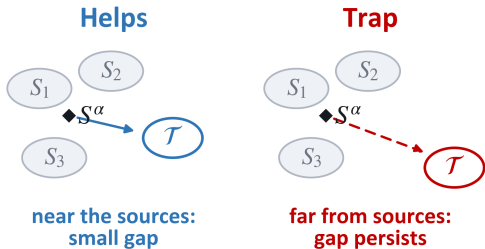
$$|\text{gen}_{\text{meta}}| \leq \sqrt{\frac{2\sigma^2}{nm} I(U, W_{1:n}; S_{1:n})} \leq \underbrace{\sqrt{\frac{2\sigma^2}{nm} I(U; S_{1:n})}}_{\text{environment level}} + \underbrace{\sqrt{\frac{2\sigma^2}{nm} \sum_{i=1}^n I(W_i; S_i | U)}}_{\text{task level}}$$

⇒ **Helps when tasks share one environment and each has few examples.**

Transfer depends on the gap between domains

- ◇ Combine source domains into a **weighted mixture** S^α
- ◇ The target-risk bound includes **source risk and a distribution gap**
- ◇ More source samples alone do not remove this gap for a fixed representation and mixture

$$R_{\mathcal{T}}(u, v) \leq R_{S^\alpha}(u, v) + \mathbf{W}_1(\tilde{\mathcal{J}}_u, \tilde{S}_u^\alpha)$$



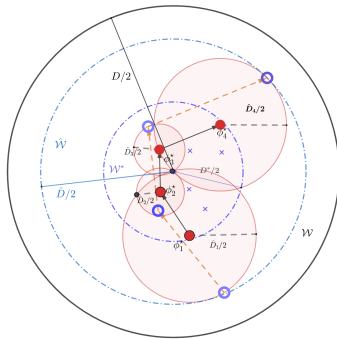
⇒ **Transfer requires relevant sources and a suitable representation.**

Update the prior as the task environment changes

- ❖ In a changing environment, a single fixed prior may become less useful
- ❖ Adapt both the prior ϕ_t and **how strongly each task relies on it**
- ❖ The bound balances adaptation against **mismatch with the prior**

$$R_{\text{excess}} \leq \kappa_t \left(a\beta_t + \frac{b\|\phi_t - w_t\|^2 + \epsilon_t + \epsilon_0}{\beta_t} + \Delta_t \right)$$

- Across tasks, the bound also depends on **how far the best meta-parameters move**



⇒ Adaptation requires tracking change as well as retaining knowledge.

Chen, Shui, Han, Marchand. On the Stability–Plasticity Dilemma in Continual Meta-Learning: Theory and Algorithm. NeurIPS 2023

Large Models

broad knowledge still needs task-specific adaptation

Central questions:

Why does a pretrained model still need to adapt?

Where can we adapt it, and at what cost?

Why a large model still needs continual learning

Tasks and evidence keep changing, and broad knowledge is not always precise enough for a new task.

☐ Update the weights

- ◇ Learns the new task, but can degrade what the model already does

☐ Keep the weights, add an adapter

- ◇ How to add it, how much to add, and how to infer with it afterwards

☐ Change nothing, even if the model knows everything

- ◇ It must still infer **effectively** (find what is relevant) and **precisely** (use the right piece)

⇒ The question is not whether to adapt, but what to change and how to use it.

Adapt weights, generation path, or search process

Our recent work adapts large models to new tasks in three ways:

1. **Post-training:** adapt the **weights** to a target task, and measure what else is lost (RL vs. SFT)
2. **Training-free:** keep a flow model frozen and adapt its **generation path** to make an edit
3. **Search process:** keep the LLM frozen and adapt the **acquisition function**
 - 3.1 MCTS decides where to search
 - 3.2 Cluster selection decides which LLM groups to keep

From weights, to the generation path, to the acquisition function

⇒ **Continual learning can adapt around a model, not only update it.**

During Training

learning a new task while retaining existing capabilities

Central questions:

When do we need to update the model weights?

Does on-policy learning help preserve previously acquired capabilities?

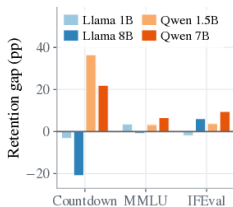
If so, does this come at the cost of preventing the learning of new tasks?

Does on-policy RL preserve existing capabilities better?

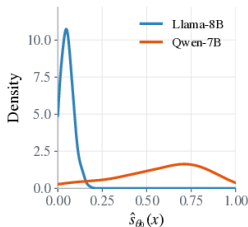
- ◇ **SFT**: learn from supplied demonstrations
- ◇ **On-policy RL**: learn from rewards on the model's own sampled responses
- ◇ A common explanation for better RL retention is **on-policy data**
- ◇ We evaluate 4 models $\times$ 3 target tasks, with retention measured on 6 held-out tasks

Does the retention advantage hold at comparable target-task gains?

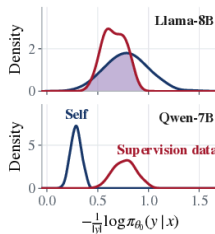
RL retention depends on the model and task



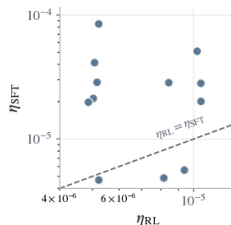
(a) RL v.s. SFT



(b) Correct-response probability



(c) Self vs. supervision data



(d) Learning rate at matched gain

- ◇ (a) On-policy RL often retains more than SFT, but not always
- ◇ (b) Weak model: limited reward signal; RL learns little at small learning rates
- ◇ (c, d) SFT usually needs a large learning rate for new data, and forgets more
- ◇ (c, d) Compatible data: a small one suffices, learns well, forgets less

⇒ The gap depends on the initial model and the learning rate needed.

Decomposing each update's effect on retention

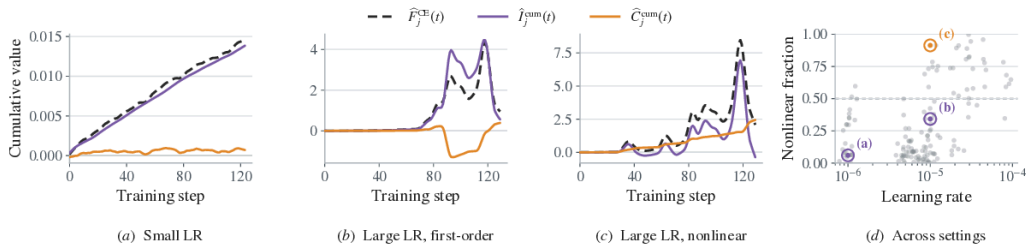
Training updates the weights by $\Delta\theta_t = \theta_{t+1} - \theta_t$; on held-out task j :

$$\hat{\mathcal{L}}_j^{\text{CE}}(\theta_{t+1}) - \hat{\mathcal{L}}_j^{\text{CE}}(\theta_t) = \hat{I}_{j,t} + \hat{C}_{j,t}$$

- ◇ **Gradient-update alignment:** $\hat{I}_{j,t} = \nabla_{\theta} \hat{\mathcal{L}}_j^{\text{CE}}(\theta_t)^{\top} \Delta\theta_t$
- ◇ **Curvature contribution:** $\hat{C}_{j,t} = \int_0^1 (1 - \tau) \Delta\theta_t^{\top} \nabla_{\theta}^2 \hat{\mathcal{L}}_j^{\text{CE}}(\theta_t + \tau \Delta\theta_t) \Delta\theta_t d\tau$
- ◇ **Known-answer loss:** $\hat{\mathcal{L}}_j^{\text{CE}}(\theta) = -\frac{1}{n_j} \sum_i \log \pi_{\theta}(y_i^{(j)} | x_i^{(j)})$ (base model correct)
- ◇ **Step size:** $\Delta\theta_t \propto \eta$, so $\hat{I}_{j,t} = O(\eta)$ and $\hat{C}_{j,t} = O(\eta^2)$

⇒ **Large learning rates make curvature matter.**

Larger learning rates make the nonlinear part matter



- ◇ (a) Small learning rate: the first-order part explains almost all of the change
- ◇ (b–d) Large learning rate: the nonlinear part can grow, depending on the task

⇒ **Larger steps add nonlinear forgetting.**

RL Does Not Always Forget Less. Fig. 3. Under review

16

At Inference

the weights stay fixed; the generation path adapts

Central questions:

How can we follow an edit instruction while preserving the rest of the image?

Edit images by steering a frozen flow model

- ◇ Keep the pretrained FLUX model **fixed**
- ◇ Adjust the latent during sampling using an **edit reward** and a **fidelity constraint**

$$dx_t = v_{\theta}(x_t, t, c^{\text{edit}}) dt + \sum_k u_k \delta(t - t_k)$$

- ◇ Editing and preservation can demand **conflicting updates** to the same latent
- ◇ Require $r_{\text{fid}} \geq \epsilon$ to limit deviation from the source image

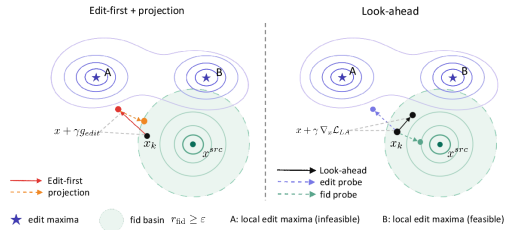
⇒ **Adapt the generation path while keeping model parameters fixed.**

Look ahead to reduce edit–preservation conflict

Evaluate rewards after trial steps in the editing and preservation directions.

$$\mathcal{L}_{\text{LA}}(x) = r_{\text{edit}}(x + \alpha \nabla r_{\text{edit}}) + \lambda r_{\text{fid}}(x + \beta \nabla r_{\text{fid}})$$

- Compared with greedy edit-then-project: **fewer constraint violations** at similar edit strength



⇒ **Look-ahead guidance improves constraint adherence in the experiments.**

Preserve the scene while making the requested edit

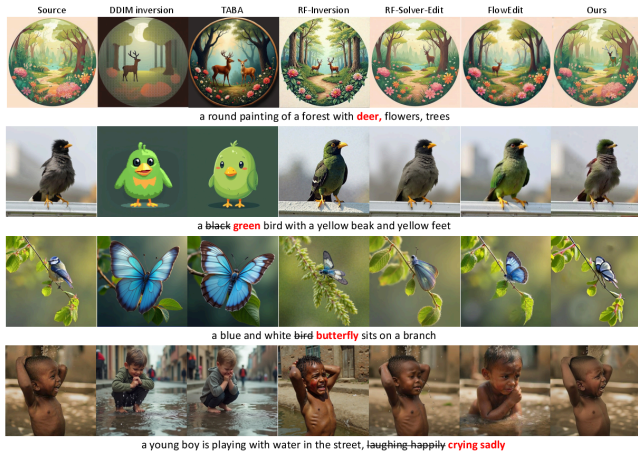


a round painting of a forest with **deer**, flowers, trees

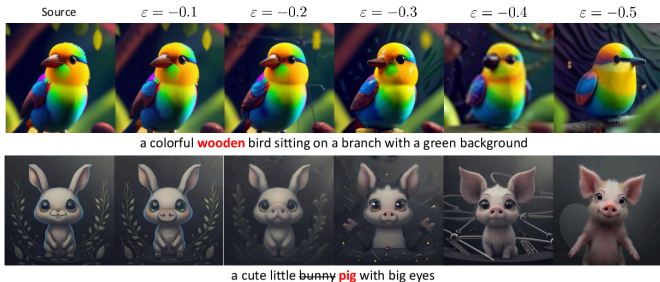
- ◇ In this example, the method adds a deer while keeping much of the forest unchanged
- ◇ On PIE-Bench vs. FlowEdit: PSNR **22.0** → **25.0**; whole-image CLIP **26.1** → **28.6**

⇒ **Explicit fidelity control can improve both preservation and edit alignment.**

The same approach supports different types of edits



Control edit strength with a fidelity threshold



⇒ **Looser fidelity constraints allow stronger edits in these examples.**

- ◇ Cost: about **400 s per image** on one A800; large semantic or style changes can remain difficult
- ◇ Next: lower the cost, and decide which adaptations to consolidate into the model

During Search

the model stays frozen; the search adapts

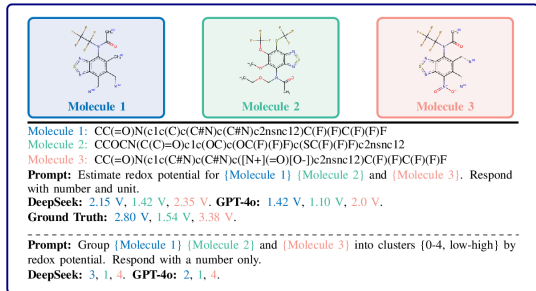
Central questions:

How do we use a foundation model's knowledge without being misled by it?

How can this knowledge guide search with limited observations?

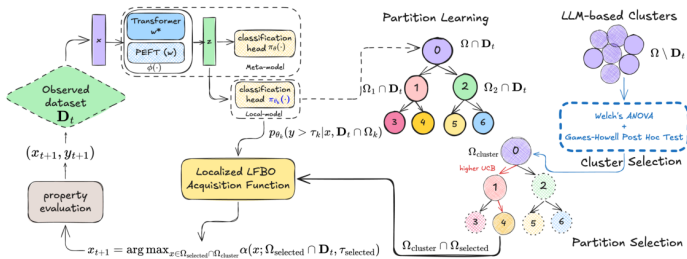
LLM rankings can help when predictions are poor

- Task: molecular discovery as **black-box optimization**, starting with **10 observations**
- On redox prediction, all three tested LLMs have $R^2 < 0$: **worse than predicting the mean**
 - GPT-4o -6.0 · Gemini-2.5-pro -88.0 · Qwen3.5-Plus -0.86
- Their direct groupings achieve **77–91% pairwise ordering accuracy** (random: 50%)



⇒ **Coarse rankings can guide search even when numerical predictions are poor.**

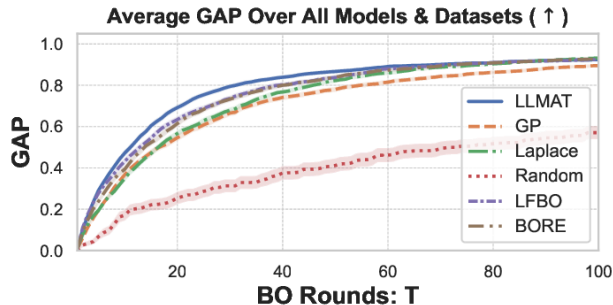
Adapt LLM knowledge for Bayesian optimization



- ❖ **Monte Carlo Tree Search (MCTS)** selects regions
- ❖ One classifier **splits regions and ranks candidates**
- ❖ **Meta-learning** shares an initialization

- ❖ **LLM knowledge** provides initial molecule groups
- ❖ **Observed data** train classifiers and inform pruning

LLMAT finds strong molecules in fewer rounds



- ◇ **GAP**: progress of the best molecule found so far toward the optimum (1 = optimum found)
- ◇ Averaged over all foundation models and datasets, with fixed features
- ◇ LLMAT leads in the **early rounds**, ahead of GP, Laplace, LFBO, BORE and random

⇒ **Model knowledge with local search speeds up early discovery.**

Conclusion

Three principles for learning and adaptation

1. **Check what transfers.** Reuse relevant knowledge and update priors as tasks change.
2. **Choose where to adapt.** Update weights, steer generation, or learn within the search process.
3. **Measure what is retained.** Evaluate new-task gains alongside effects on existing capabilities.

⇒ **Use past knowledge while remaining responsive to new evidence.**