Large-Scale Meta-Learning with Continual Trajectory Shifting

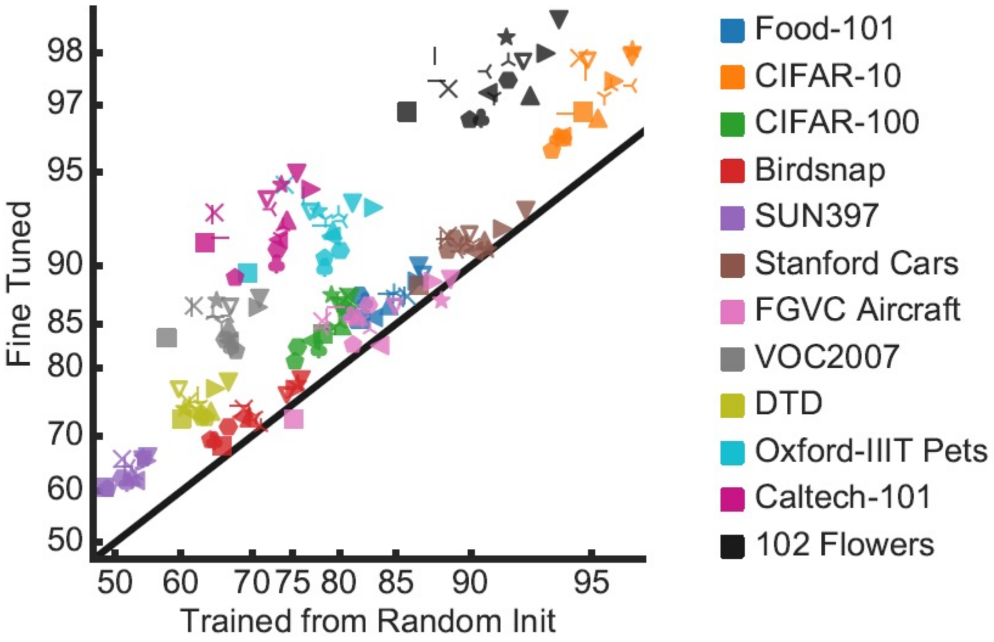
JaeWoong Shin*, Hae Beom Lee*, Boqing Gong, Sung Ju Hwang

(*: Equal contribution)

ICML 2021

Beyond Few-shot Learning

- Meta-learning is effective for solving few-shot learning.
- What if **many-shot**? We already know that knowledge transfer is effective for many-shot dataset as well (e.g. ImageNet finetuning).

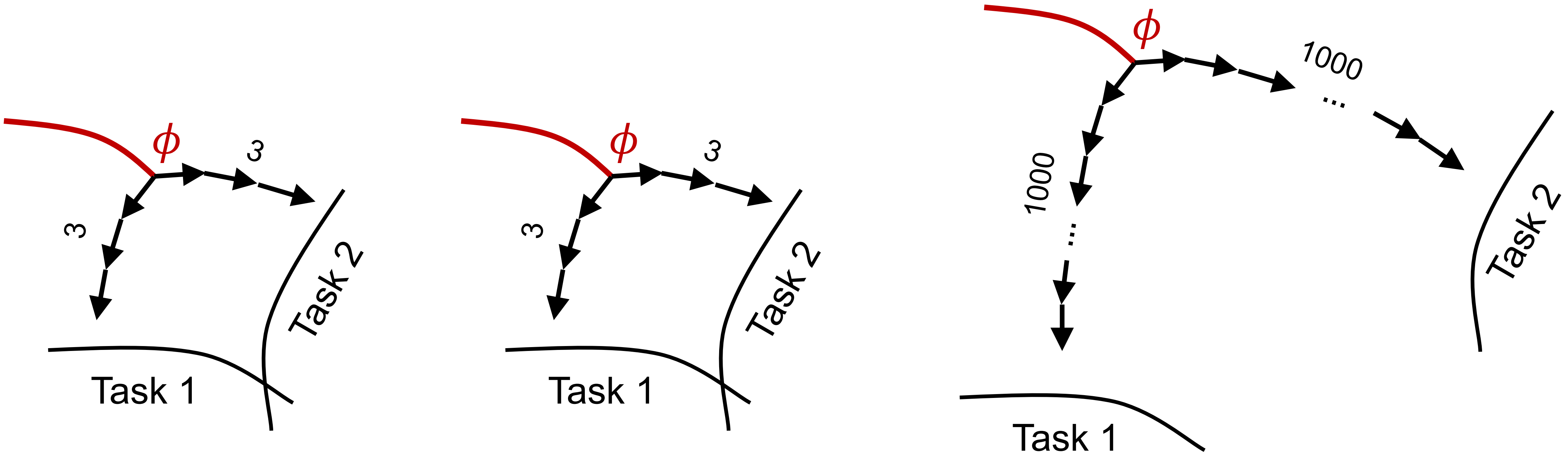


Kornblith et al., Do Better ImageNet Models Transfer Better?, CVPR 2019

Large-Scale Meta-Learning

Large-scale meta-learning: **many-shot** and **heterogeneous** task distribution.

→ Requires **long horizon** for inner-optimizations.



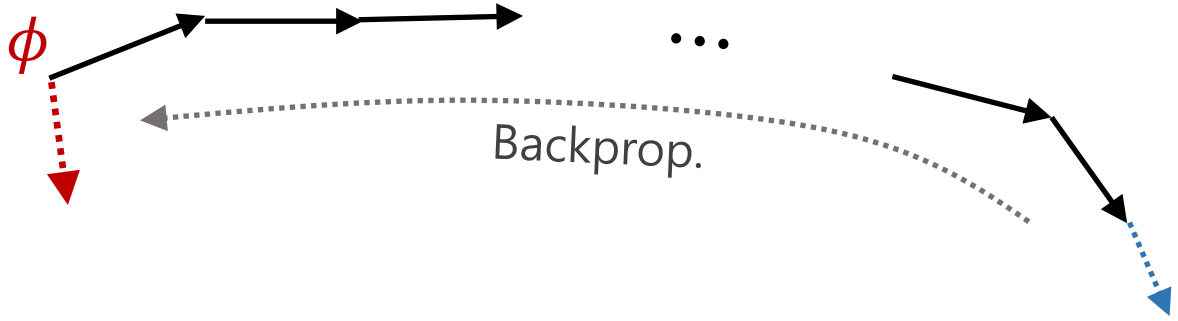
Few-shot

Many-shot
Homogeneous

Many-shot
Heterogeneous

Large-Scale Meta-Learning

High computational cost of backpropagating through long inner process.

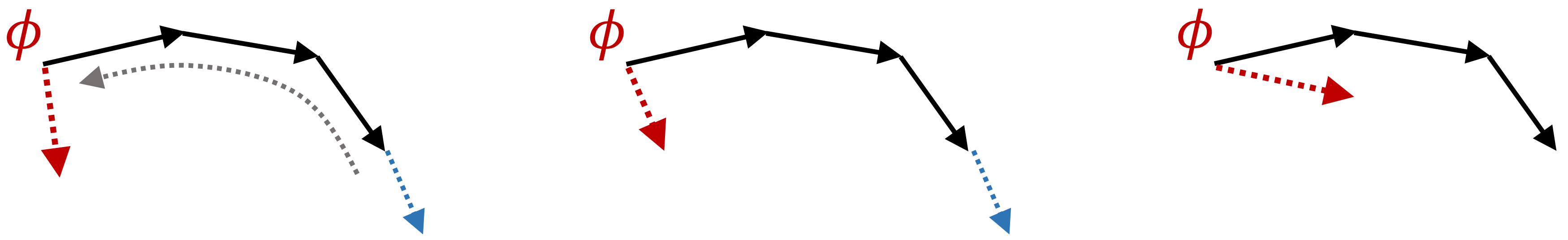


Backpropagation through learning process –
Reverse Mode Differentiation (RMD)

Franceschi et al., Forward and Reverse Gradient-Based Hyperparameter Optimization, ICML 2017

First-order Approximations

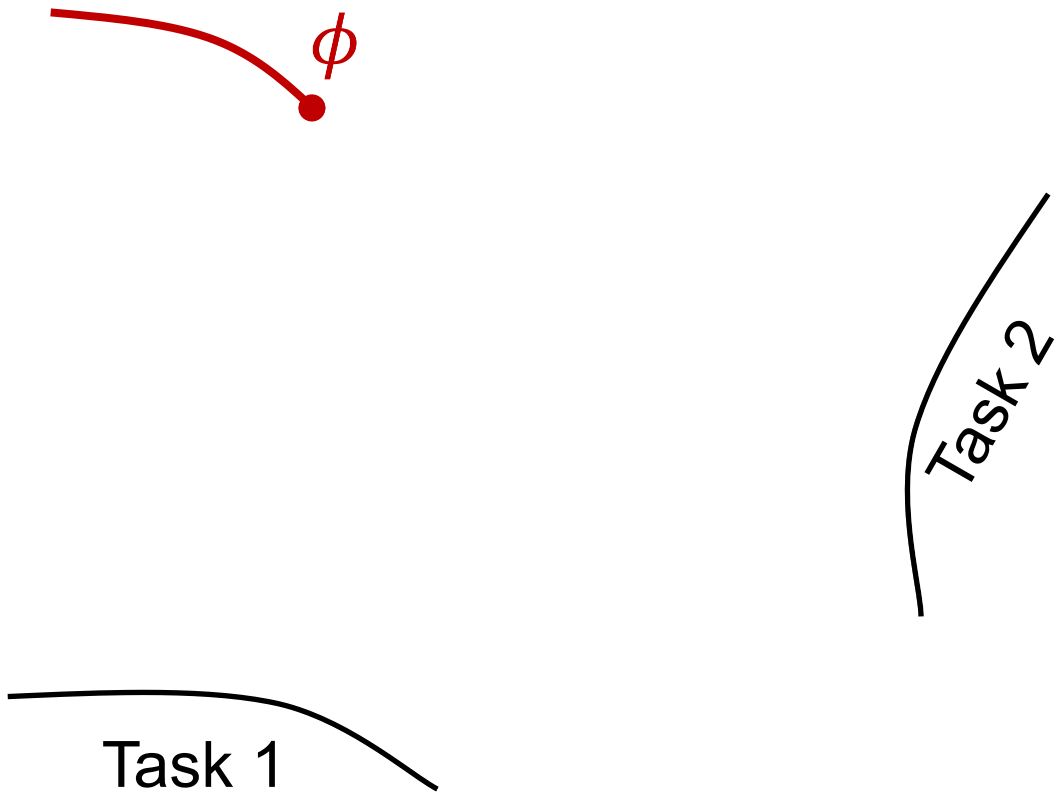
First-order approximation can be used to reduce the computational cost.



MAML FOMAML Reptile

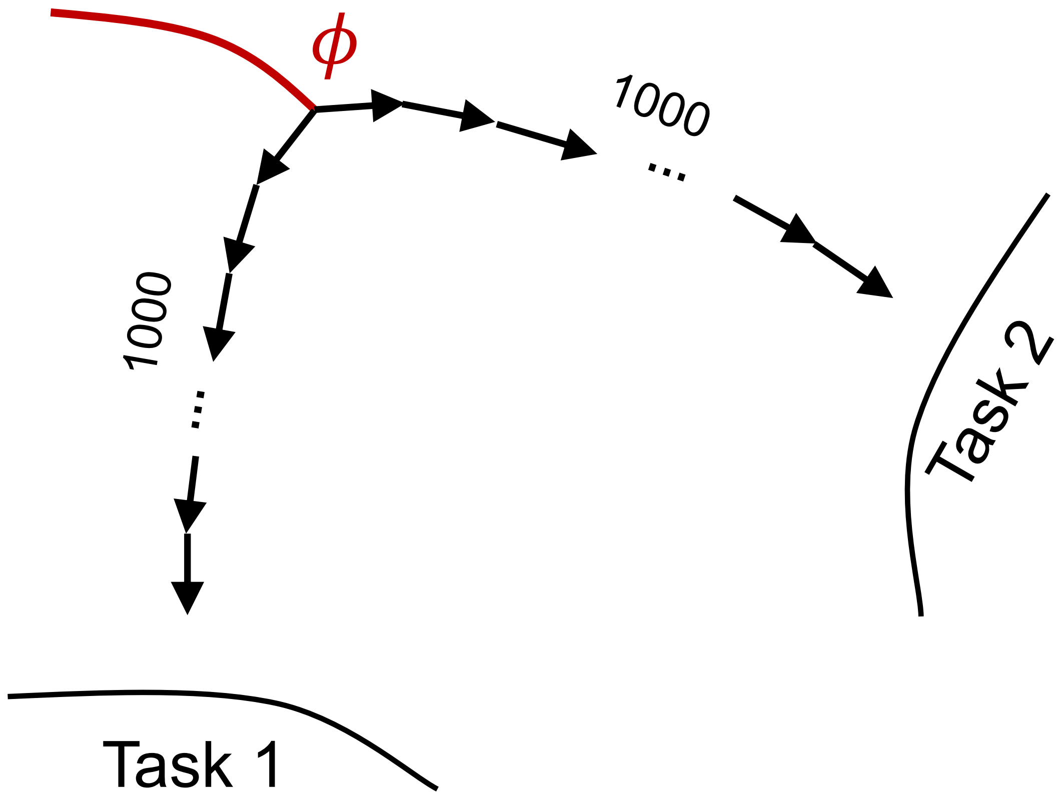
Too Slow Meta-update

However, even Reptile is inefficient for long-horizon case.
→ ex) 1000 inner-gradient steps per each meta-update.



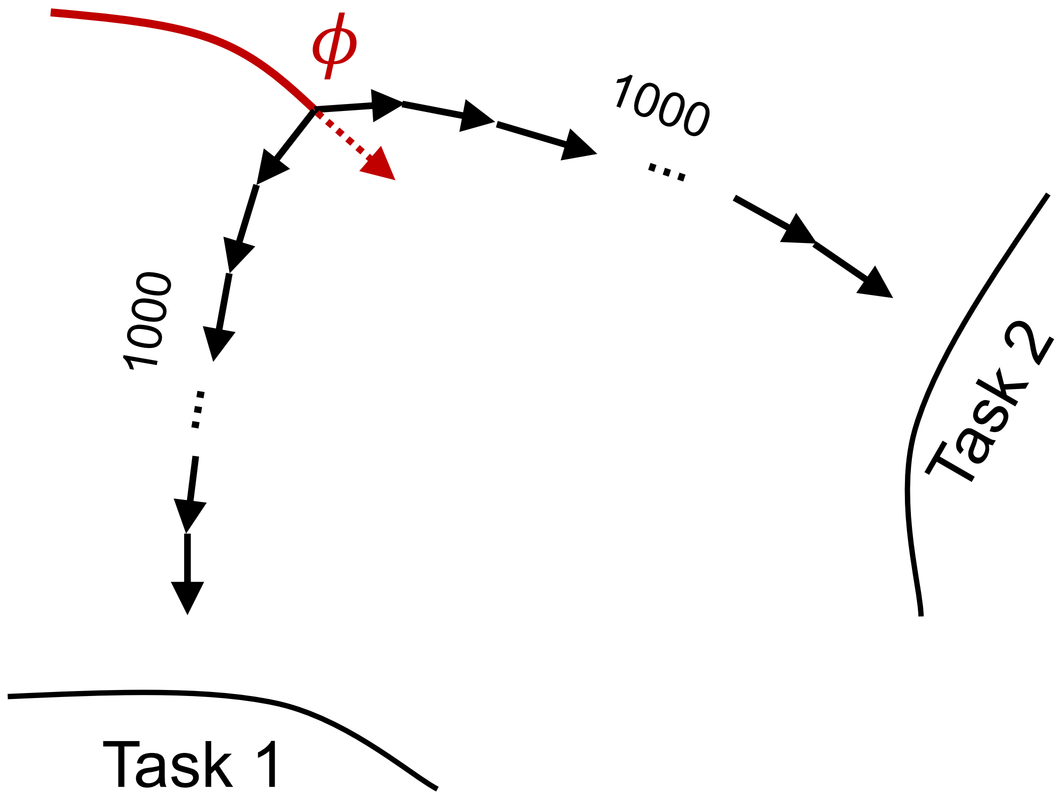
Too Slow Meta-update

However, even Reptile is inefficient for long-horizon case.
→ ex) 1000 inner-gradient steps per each meta-update.



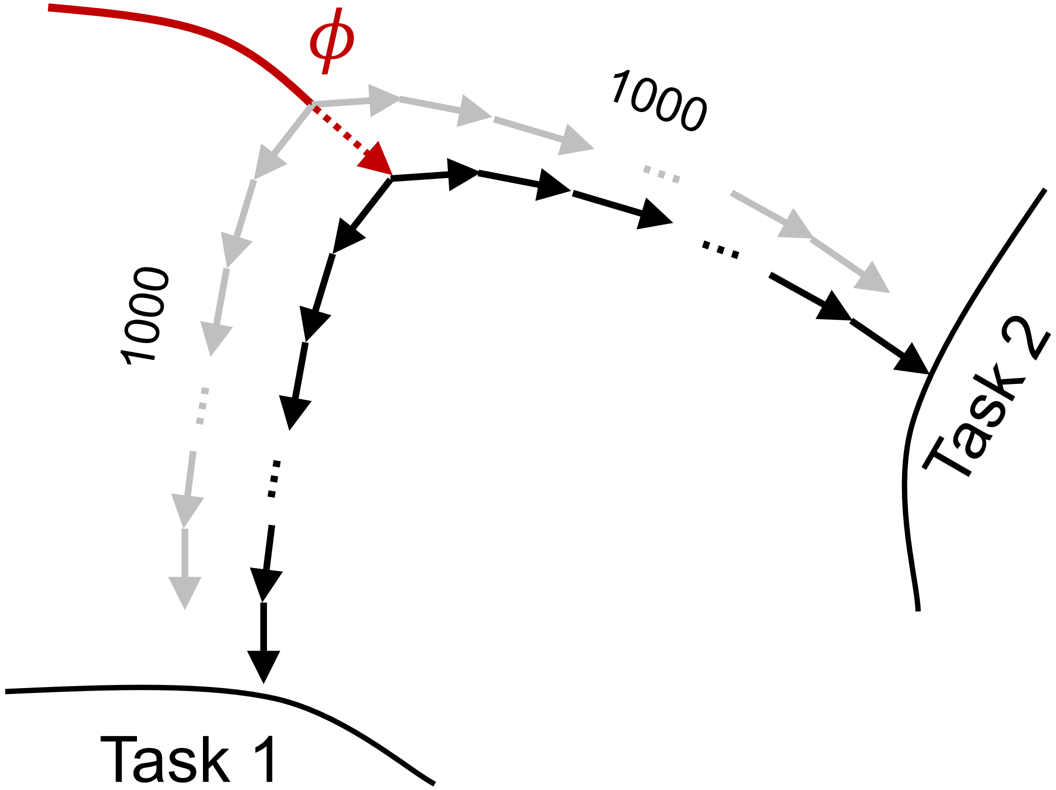
Too Slow Meta-update

However, even Reptile is inefficient for long-horizon case.
→ ex) 1000 inner-gradient steps per each meta-update.



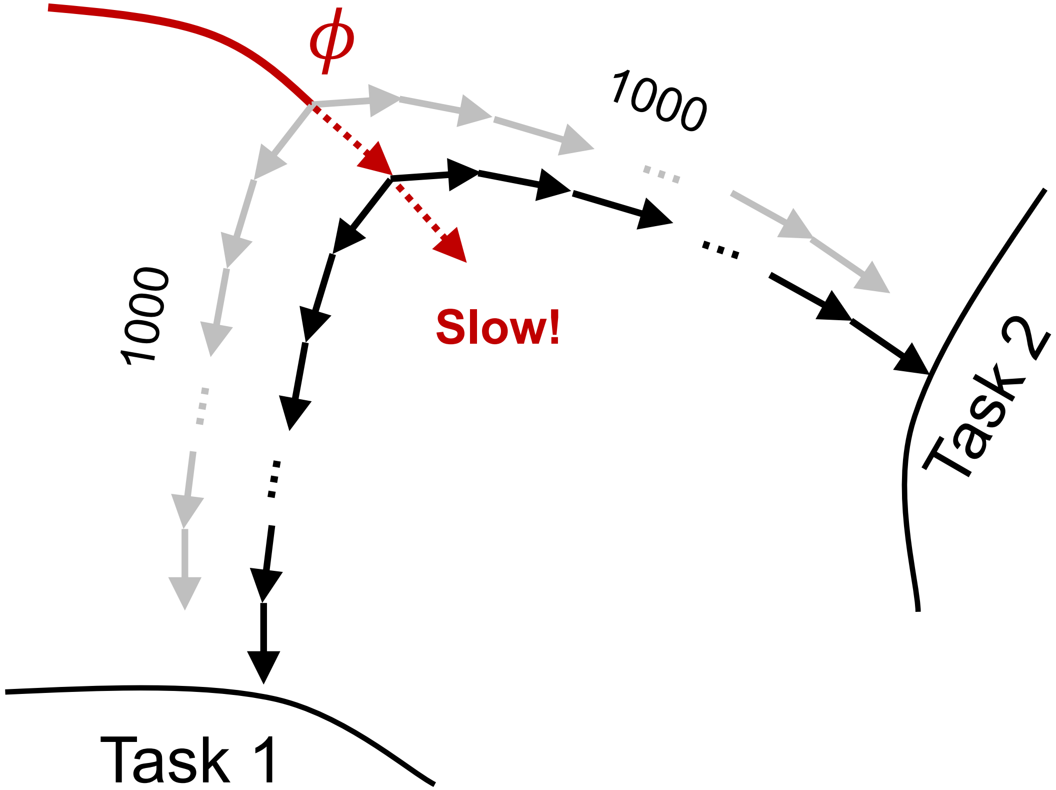
Too Slow Meta-update

However, even Reptile is inefficient for long-horizon case.
→ ex) 1000 inner-gradient steps per each meta-update.



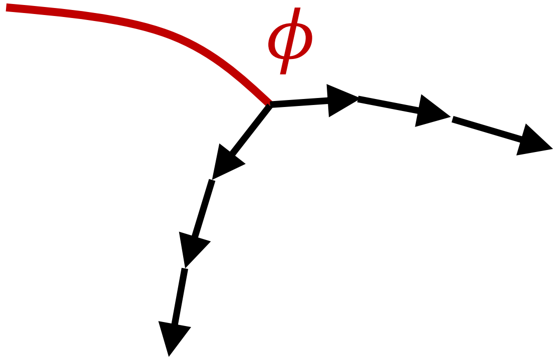
Too Slow Meta-update

However, even Reptile is inefficient for long-horizon case.
→ ex) 1000 inner-gradient steps per each meta-update.



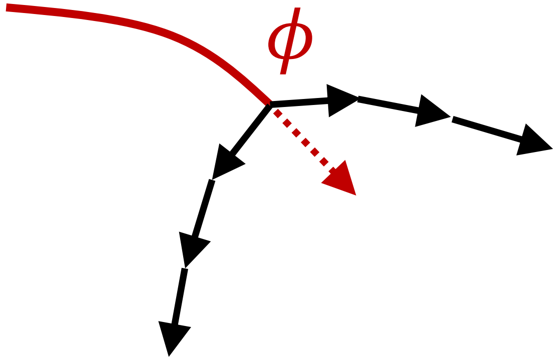
Idea – Continual Trajectory Shifting

What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?



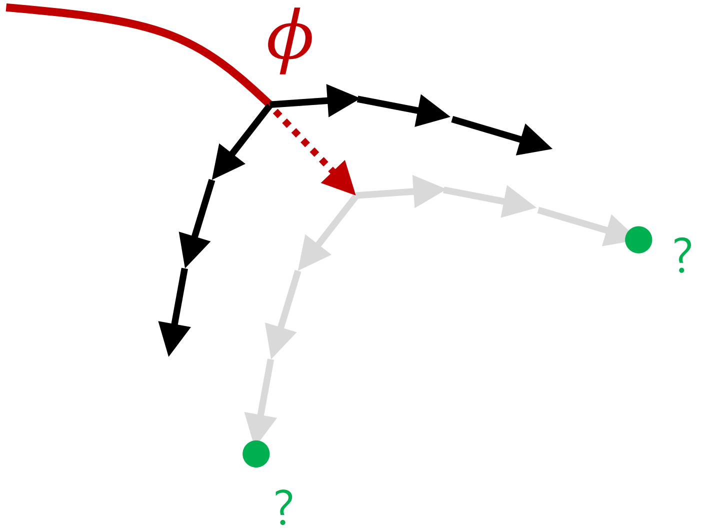
Idea – Continual Trajectory Shifting

What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?



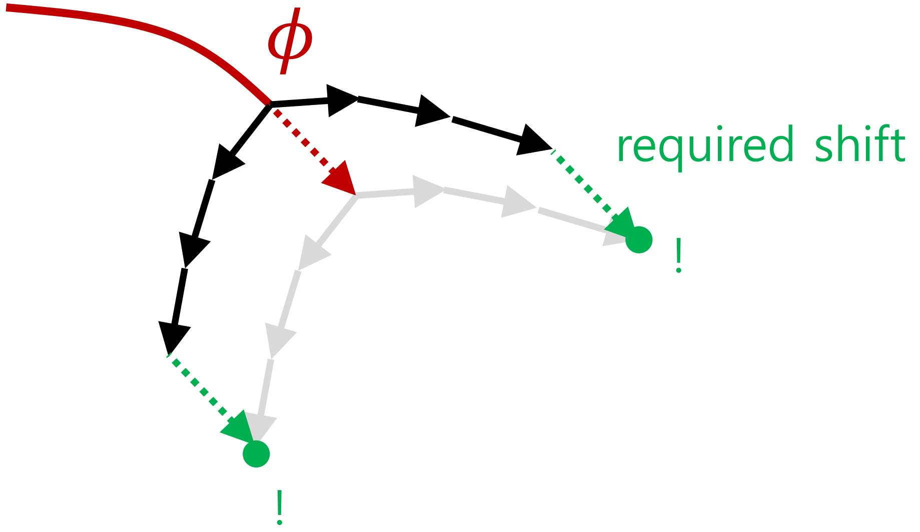
Idea – Continual Trajectory Shifting

What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?



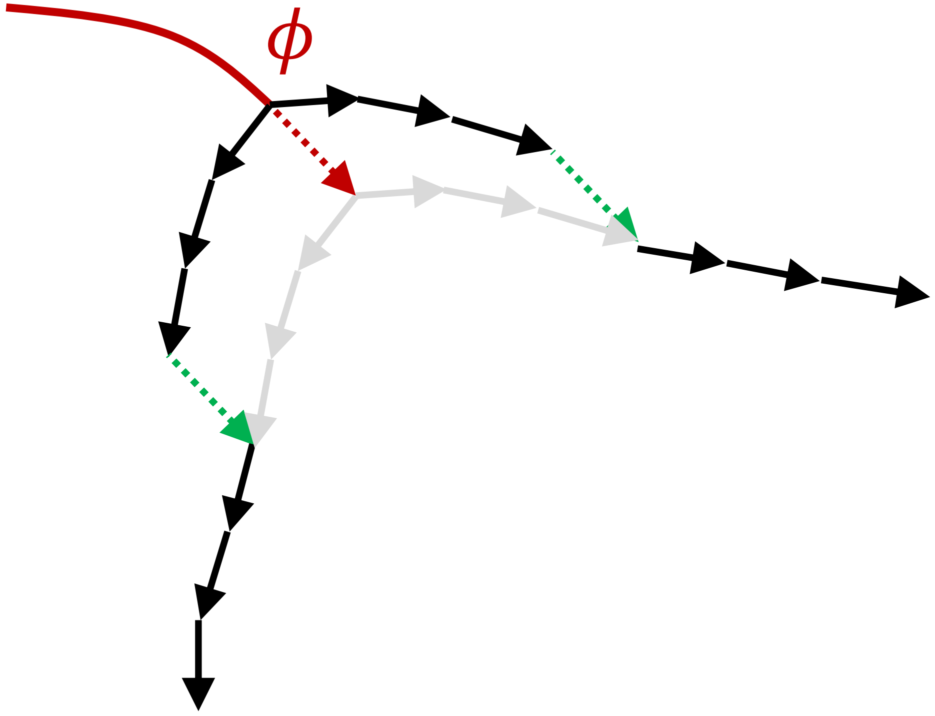
Idea – Continual Trajectory Shifting

What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?



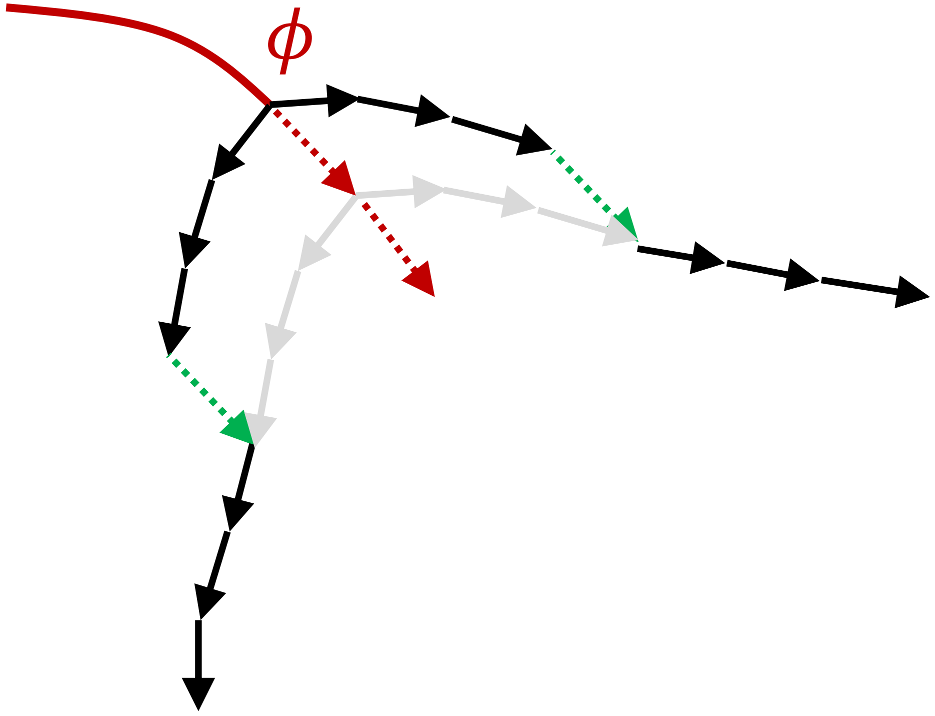
Idea – Continual Trajectory Shifting

What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?



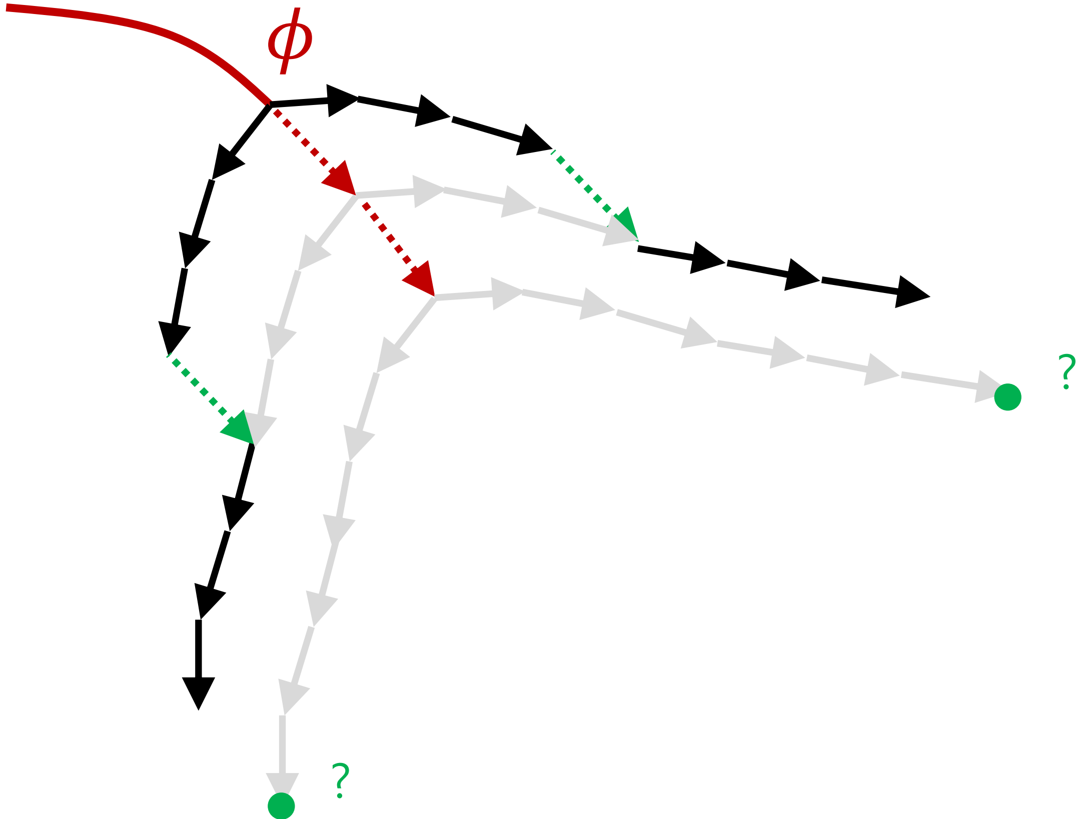
Idea – Continual Trajectory Shifting

What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?



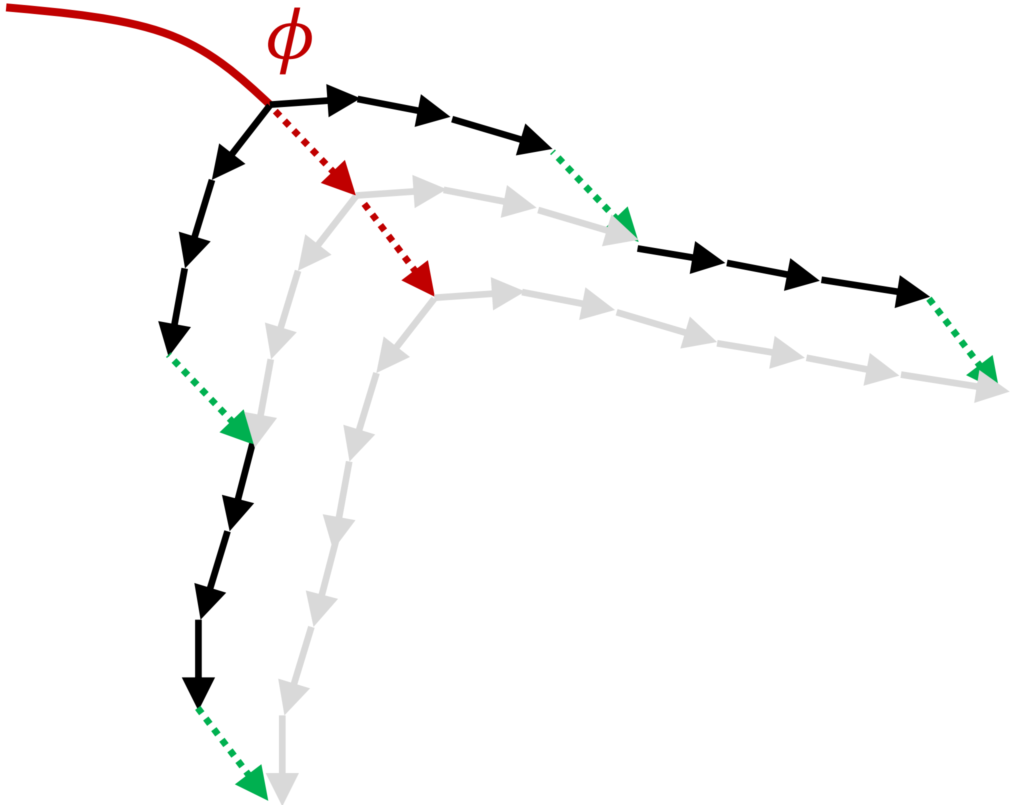
Idea – Continual Trajectory Shifting

What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?



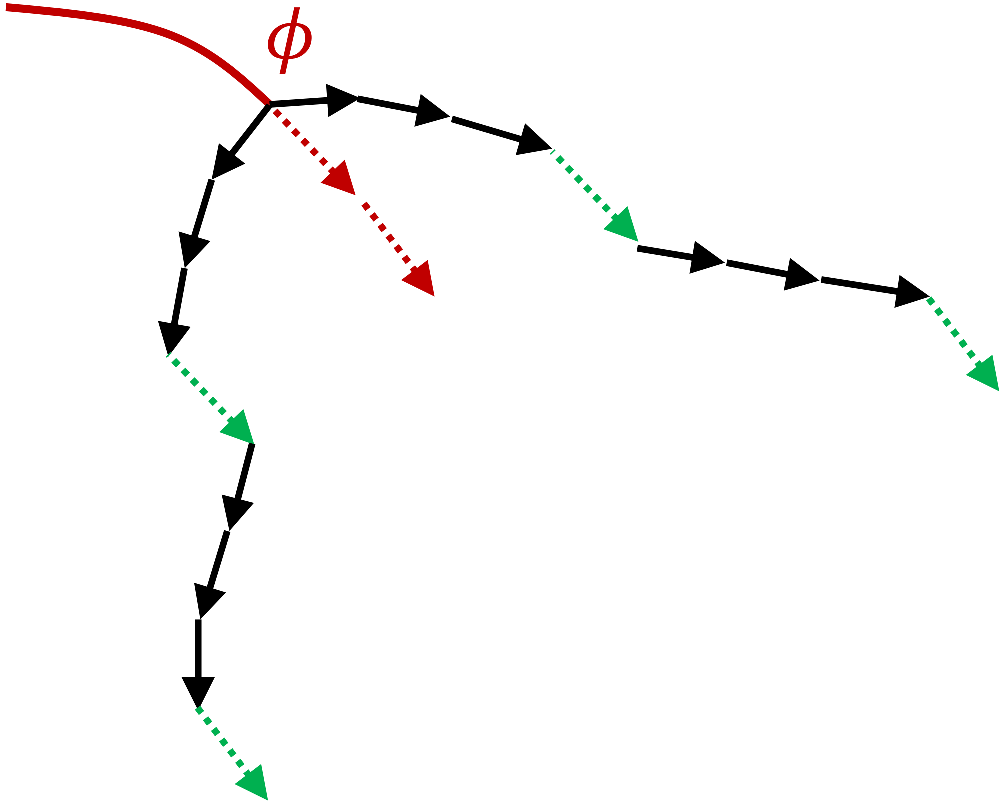
Idea – Continual Trajectory Shifting

What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?



Idea – Continual Trajectory Shifting

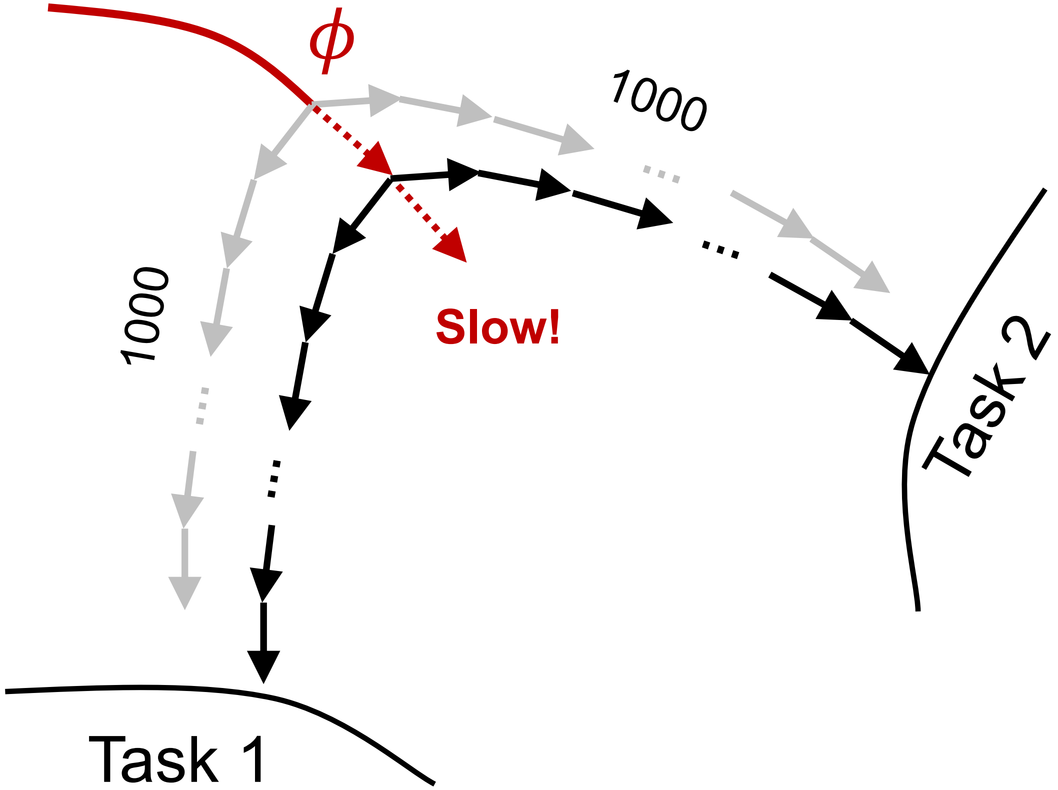
What if we can continuously estimate the **required shift of inner-trajectory** w.r.t. each meta update?

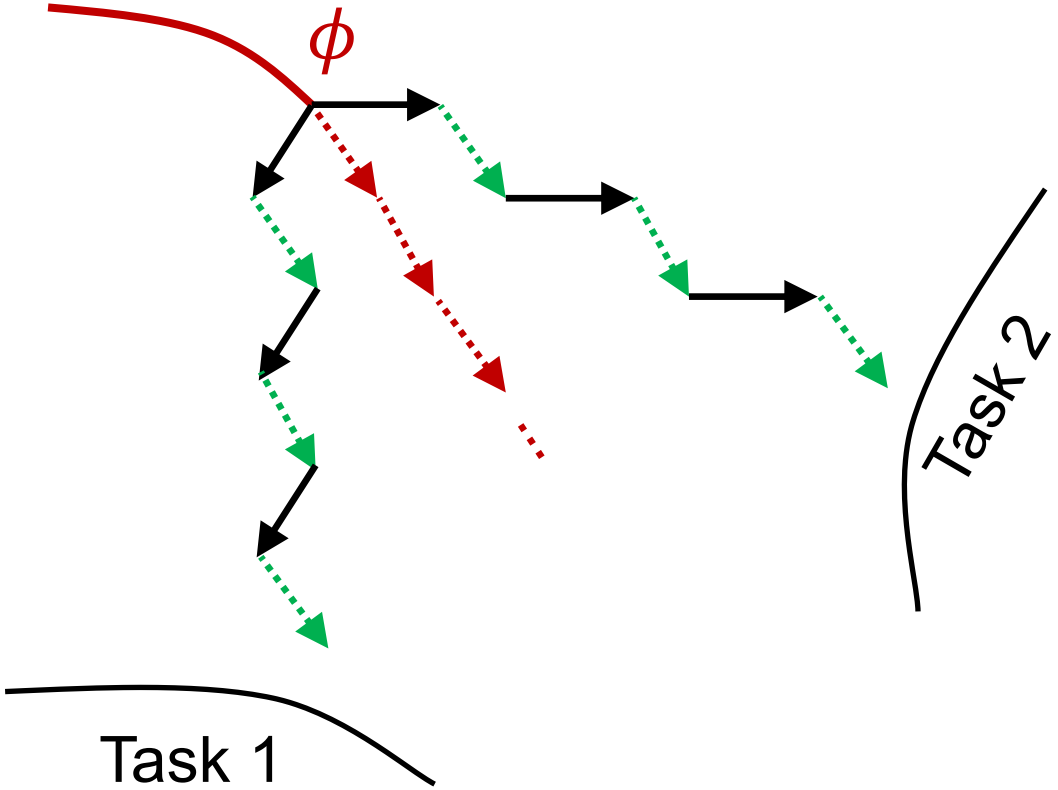


Idea – Continual Trajectory Shifting

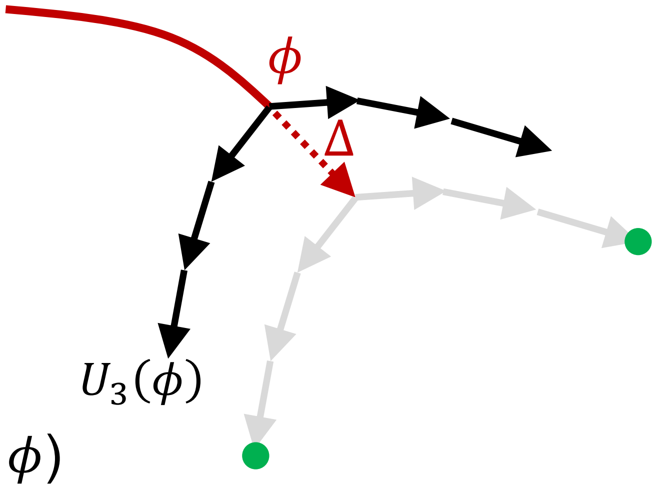
If we perform trajectory shifting for every meta-update…

→ 1000 times more frequent meta-update !!





How to Estimate?



1. First-order Taylor Approximation ($U_k(\phi)$: Update k steps from ϕ)

$$U_k(\phi + \Delta) \approx U_k(\phi) + \frac{\partial U_k(\phi)}{\partial \phi}\Delta$$

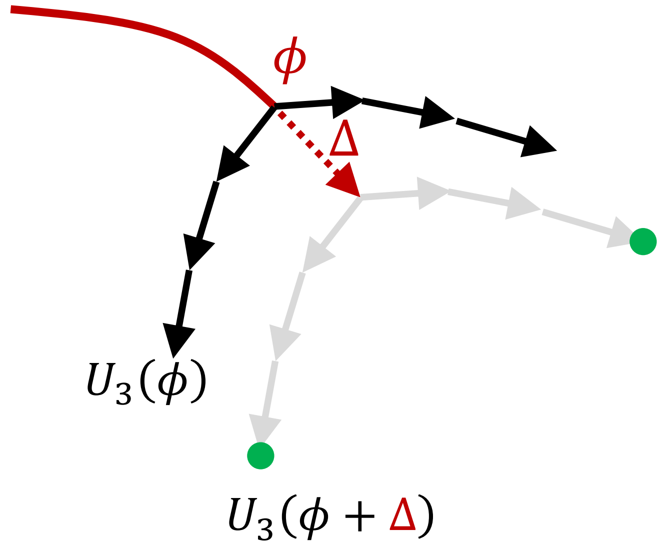
2. Hessian Approximation

$$\frac{\partial U_k(\phi)}{\partial \phi} = \frac{\partial U_k(\phi)}{\partial U_{k-1}(\phi)} \cdots \frac{\partial U_2(\phi)}{\partial U_1(\phi)} \frac{\partial U_1(\phi)}{\partial \phi} = \prod_{i=0}^{k-1}(I - \alpha H_i) \approx I$$

Therefore,

$$U_k(\phi + \Delta) \approx U_k(\phi) + \Delta$$

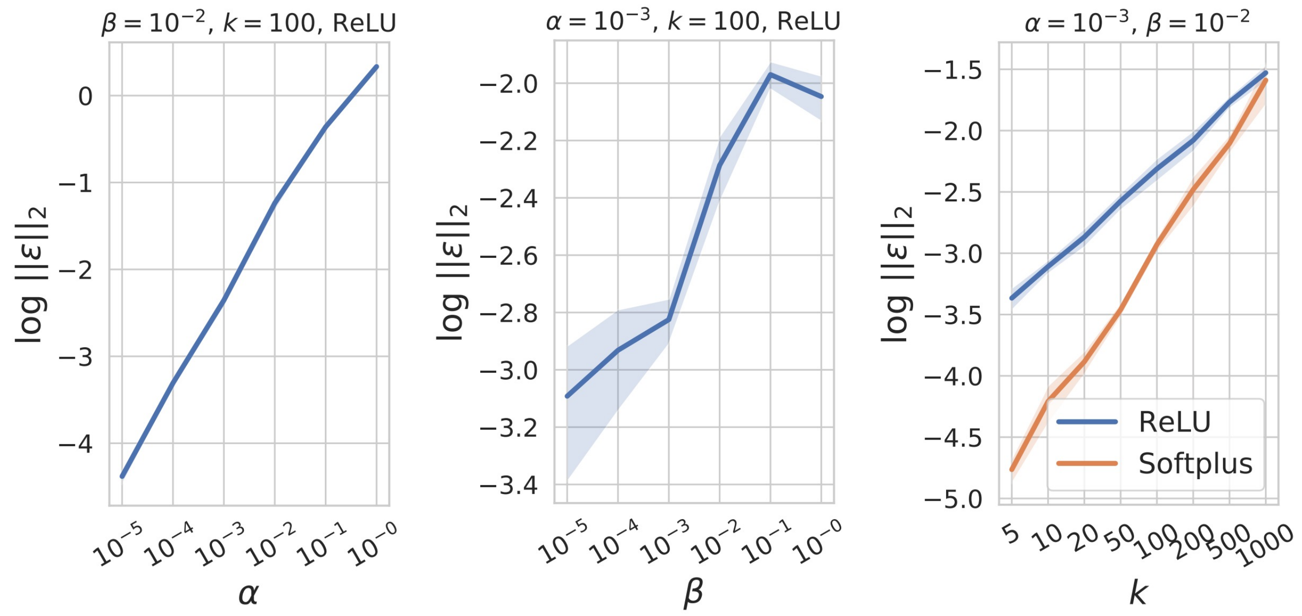
Approximation Error



The approximation errors compound as we keep shifting.

1 shift → $U_1(\phi + \Delta) = U_1(\phi) + \Delta + O(\beta\alpha h + \beta^2)$

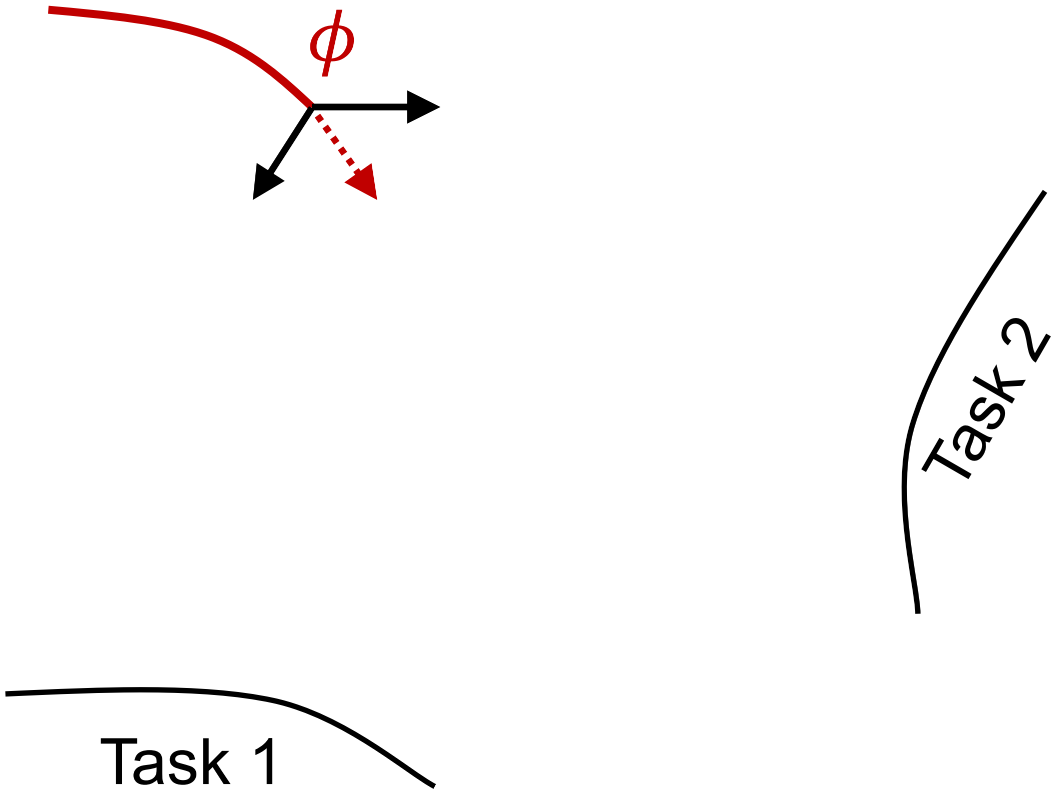
K shift → $U_k(\phi + \Delta_1 + \cdots + \Delta_{k-1})$
$= U_1(\cdots U_1(U_1(\phi) + \Delta_1) \cdots + \Delta_{k-1}) + O(\beta\alpha h \boldsymbol{k^2} + \beta^2 \boldsymbol{k})$



Then, why should it work?

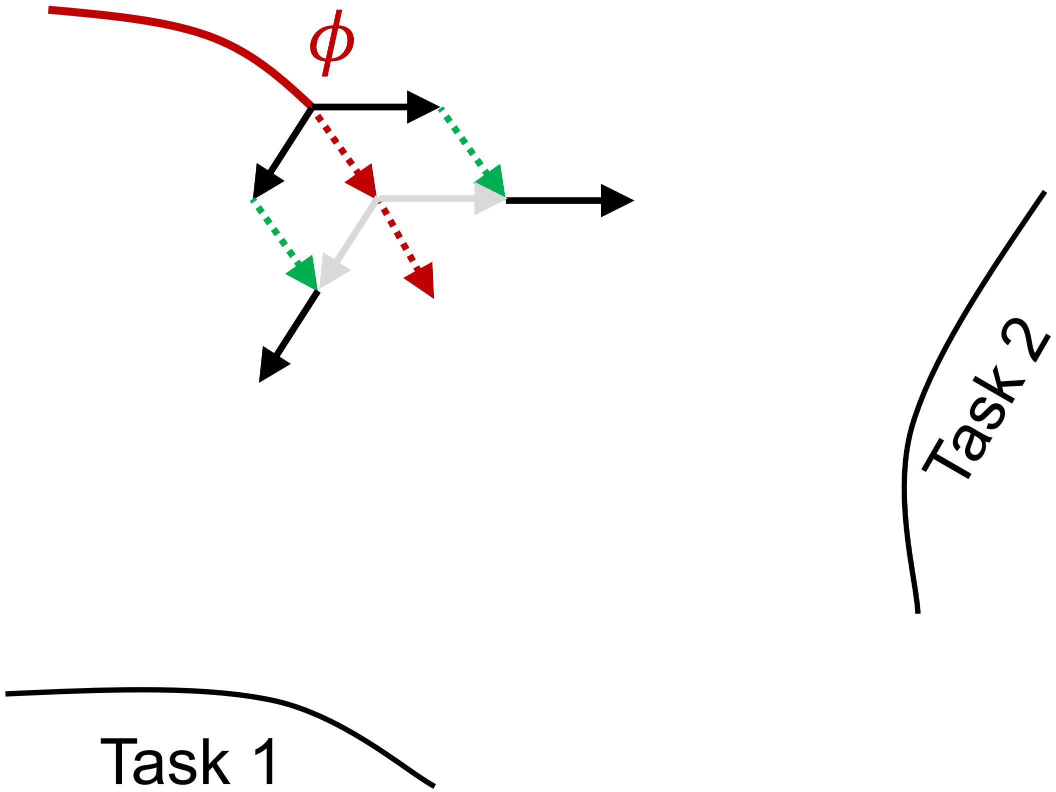
Gradually Increasing k

The horizon size k gradually increases. What does it mean?



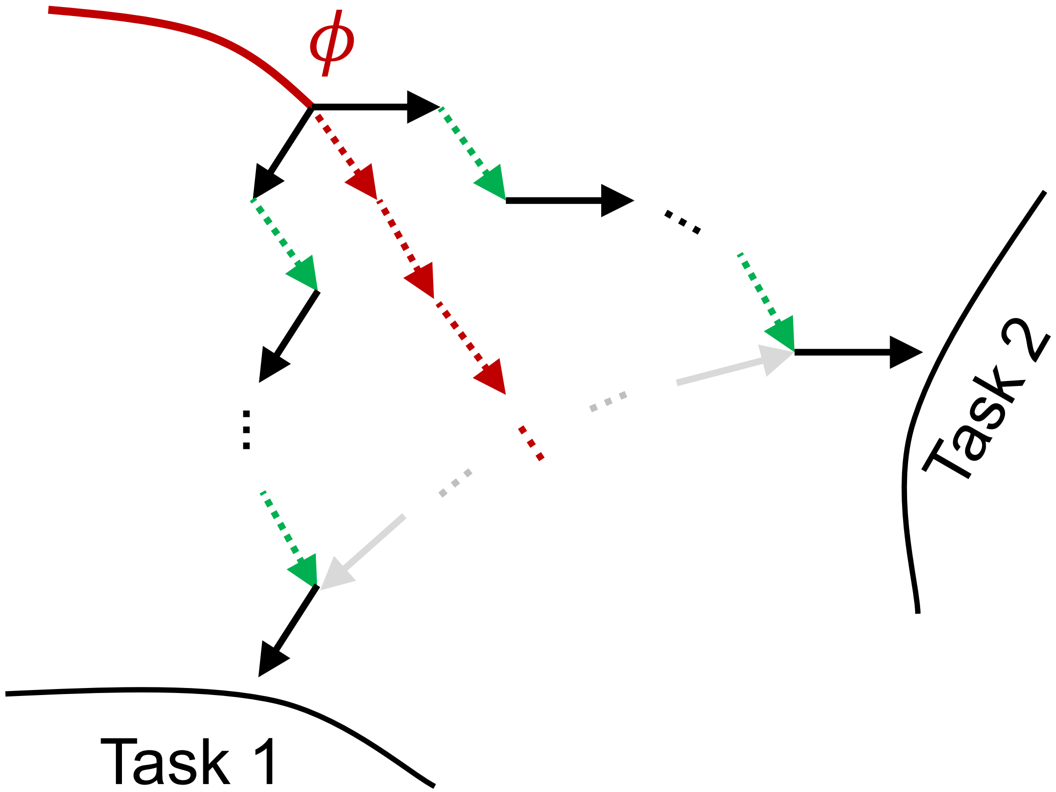
Gradually Increasing k

The horizon size k gradually increases. What does it mean?



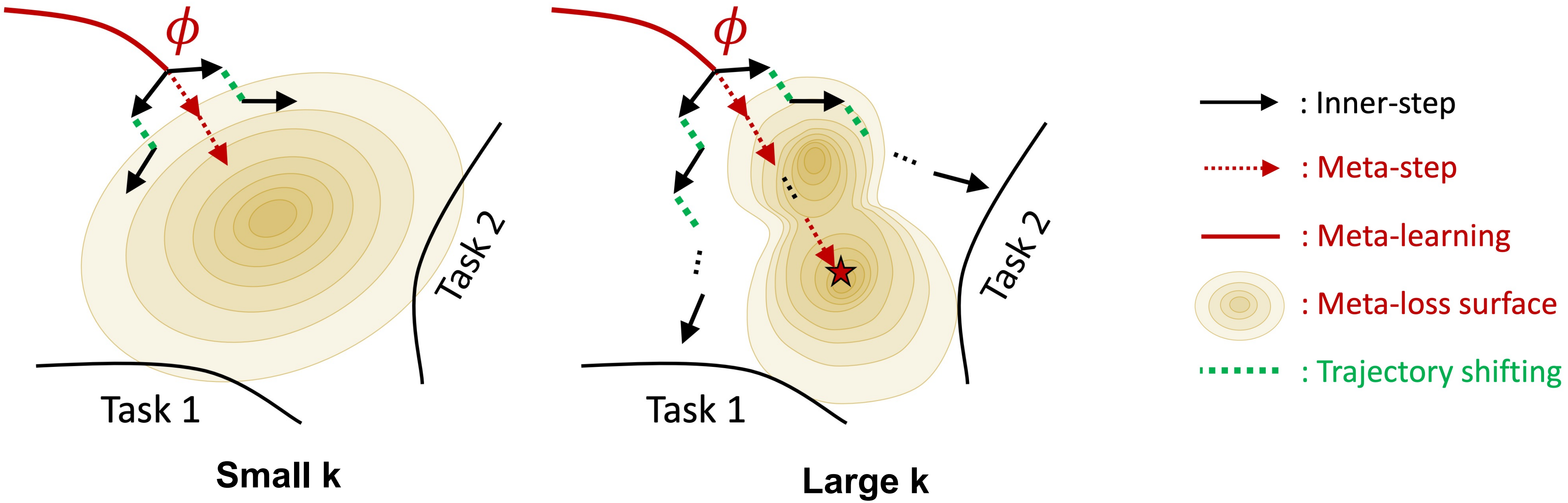
Gradually Increasing k

The horizon size k gradually increases. What does it mean?

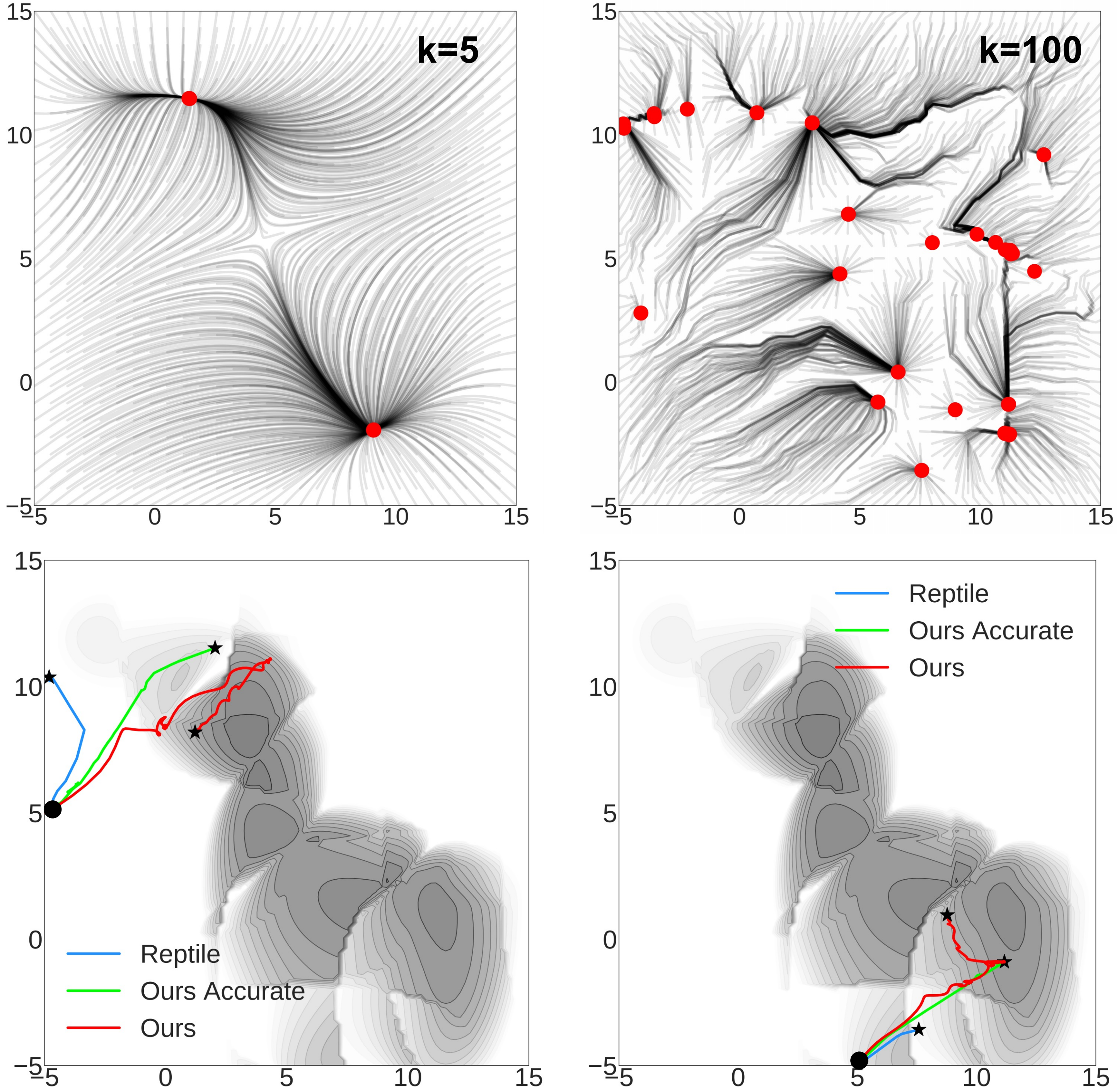


Meta-Level Curriculum Learning

Meta loss surface is smoother for smaller k → regularization effect ！



Synthetic Experiment



Experiments – Image Classification

We experiment with large-scale (**many-shot** and **heterogeneous**) datasets.

Meta-train with seven datasets:

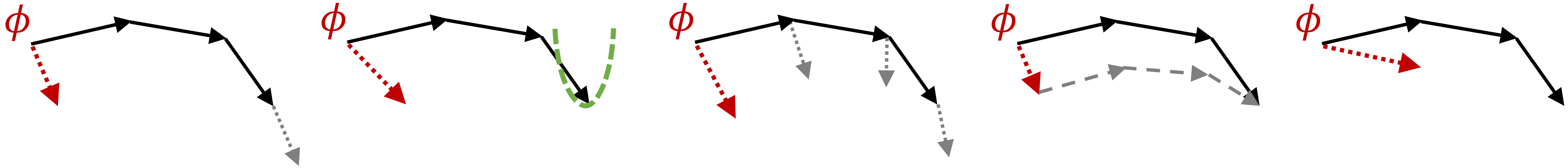
Tiny ImageNet, CIFAR-100, Stanford Dogs, Aircraft, CUB, Fashion-MNIST, and SVHN.

Meta-test with five datasets:

Stanford Cars, QuickDraw, VGG-Flowers, VGG-Pets, and STL10.

Baselines

We compare with the following first-order meta-learning algorithms. Our method (Continual Trajectory Shifting) has been applied to Reptile.



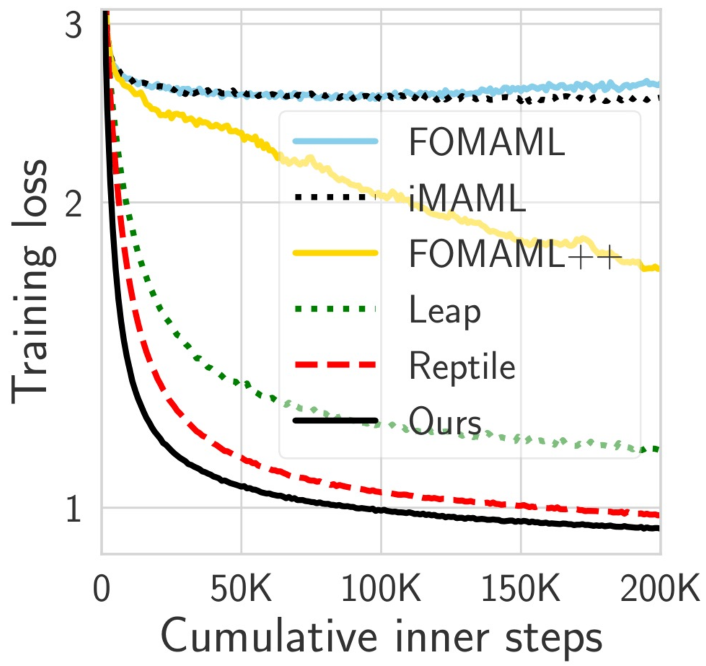
FOMAML[1] iMAML[2] FOMAML++[3] Leap[4] Reptile[5]

[1] Finn et al. 17 Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.
[2] Rajeswaran, et al. 19, Meta-learning with implicit gradients.
[3] Antoniou et al. 19, How to train your MAML.
[4] Flennerhag et al. 18, Transferring Knowledge across Learning Processes.
[5] Nichol et al. 18, On First-Order Meta-Learning Algorithms.

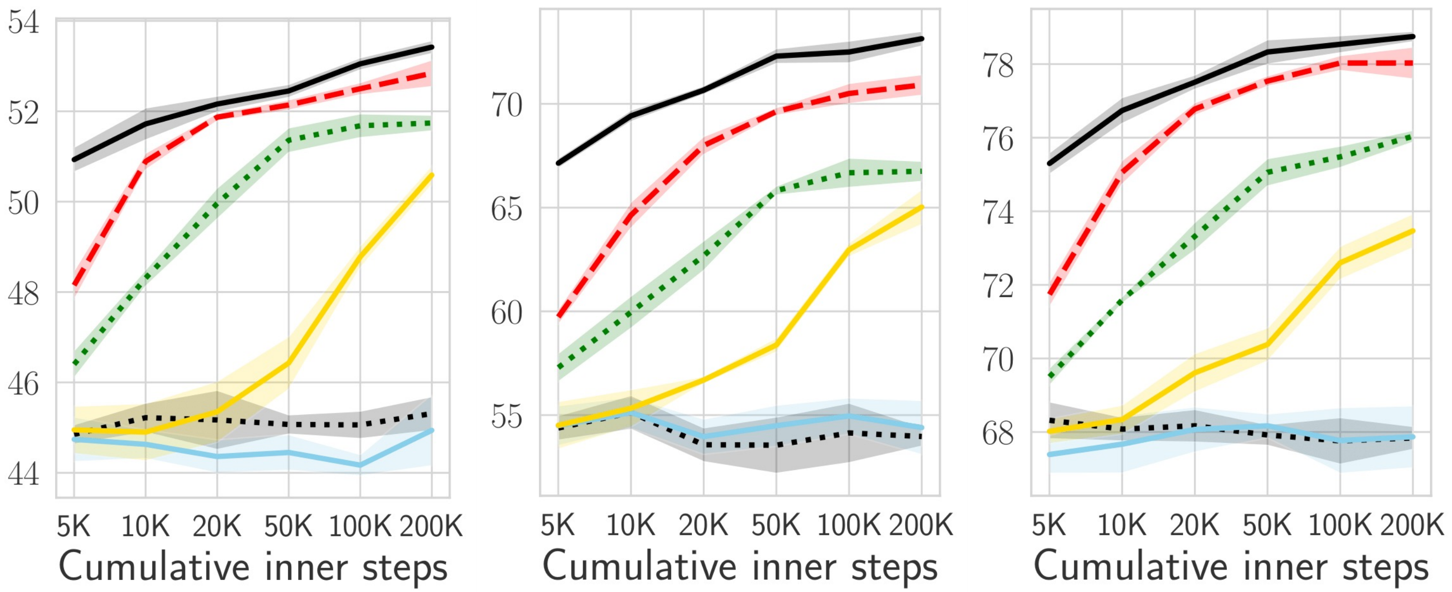
Image Classification Results

Our method outperforms meta-learning baselines, in terms of **meta-convergence** and **test accuracy**.

Meta-convergence



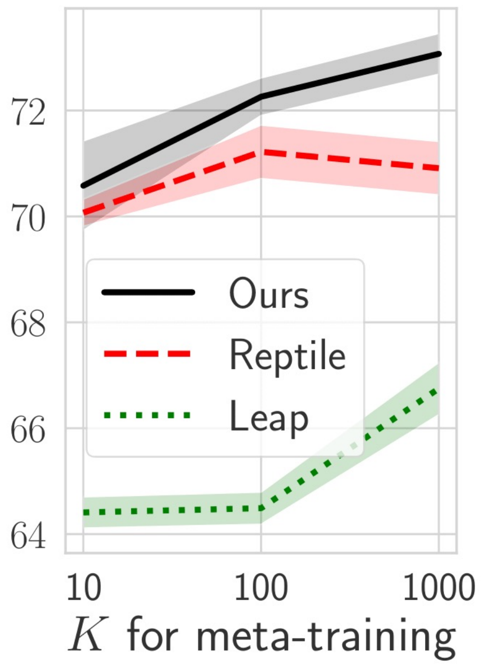
Meta-test time test accuracy



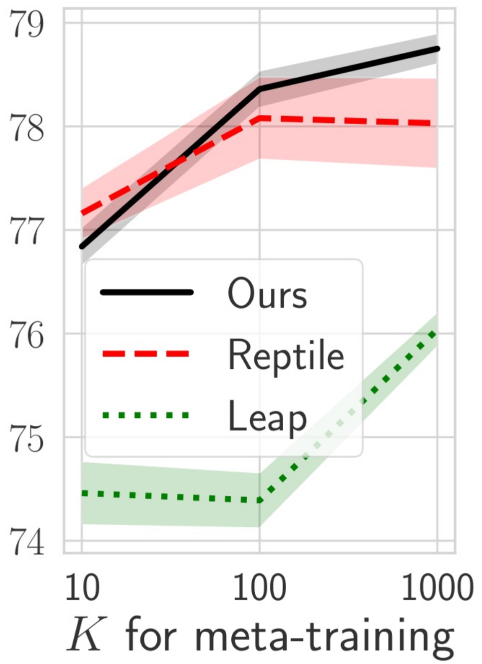
Quickdraw **VGG Pets** **STL10**

Large K is better for many-shot

Longer inner trajectory shows better performance for many-shot learning.



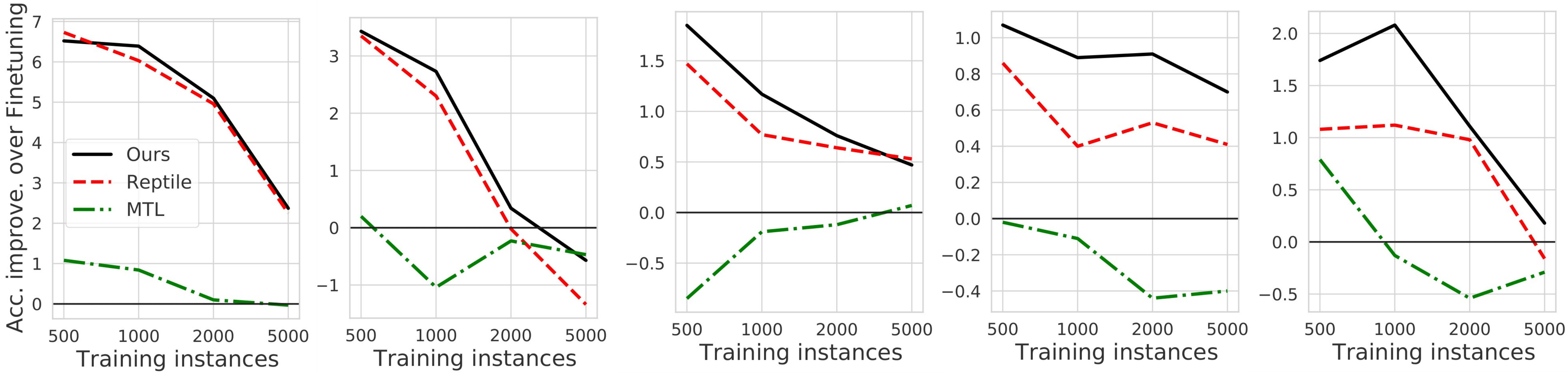
VGG Pets



STL10

Improving on ImageNet Pretraining

Our method **outperforms ImageNet finetuning** under limited data regime.



CIFAR100 SVHN VGG Pets Food CUB

Takeaways

- If the task distribution is many-shot and heterogeneous, we need to increase the length of inner-optimization trajectory.
- In solving the problem, first-order approximations are still inefficient in terms of meta-update frequency.
- We can greatly increase the meta-update frequency by continuously shift the inner-learning trajectories w.r.t each meta-update.