Quickly solving new tasks

with meta-learning and without

December 5, 2022

Séb Arnold
smr.arnold@gmail.com

Ph.D. Defense Committee: Fei Sha, Maja Matarić, Salman Avestimehr, Stefanos Nikolaidis, Jesse Thomason

How to quickly solve new tasks?

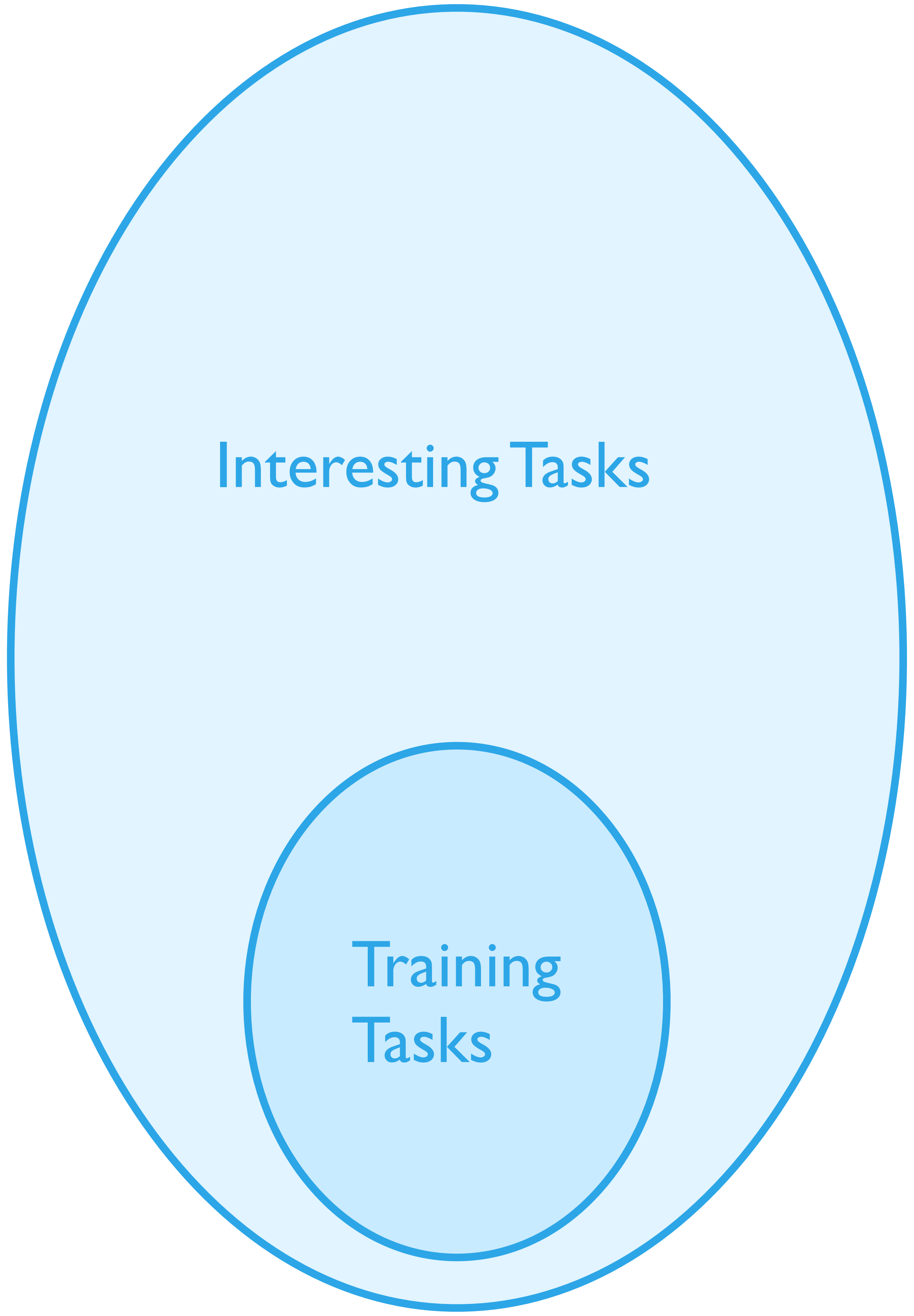
- How do we get *personalized* models for
  - Education?
  - Agriculture?
  - Code?
  - Skiing?

Artwork by DALL-E 2

How to quickly solve new tasks?

- How do we get *personalized* models for
  - Education?
  - Agriculture?
  - Code?
  - Skiing?

- Fundamental problem
  - Always more interesting tasks than training tasks.

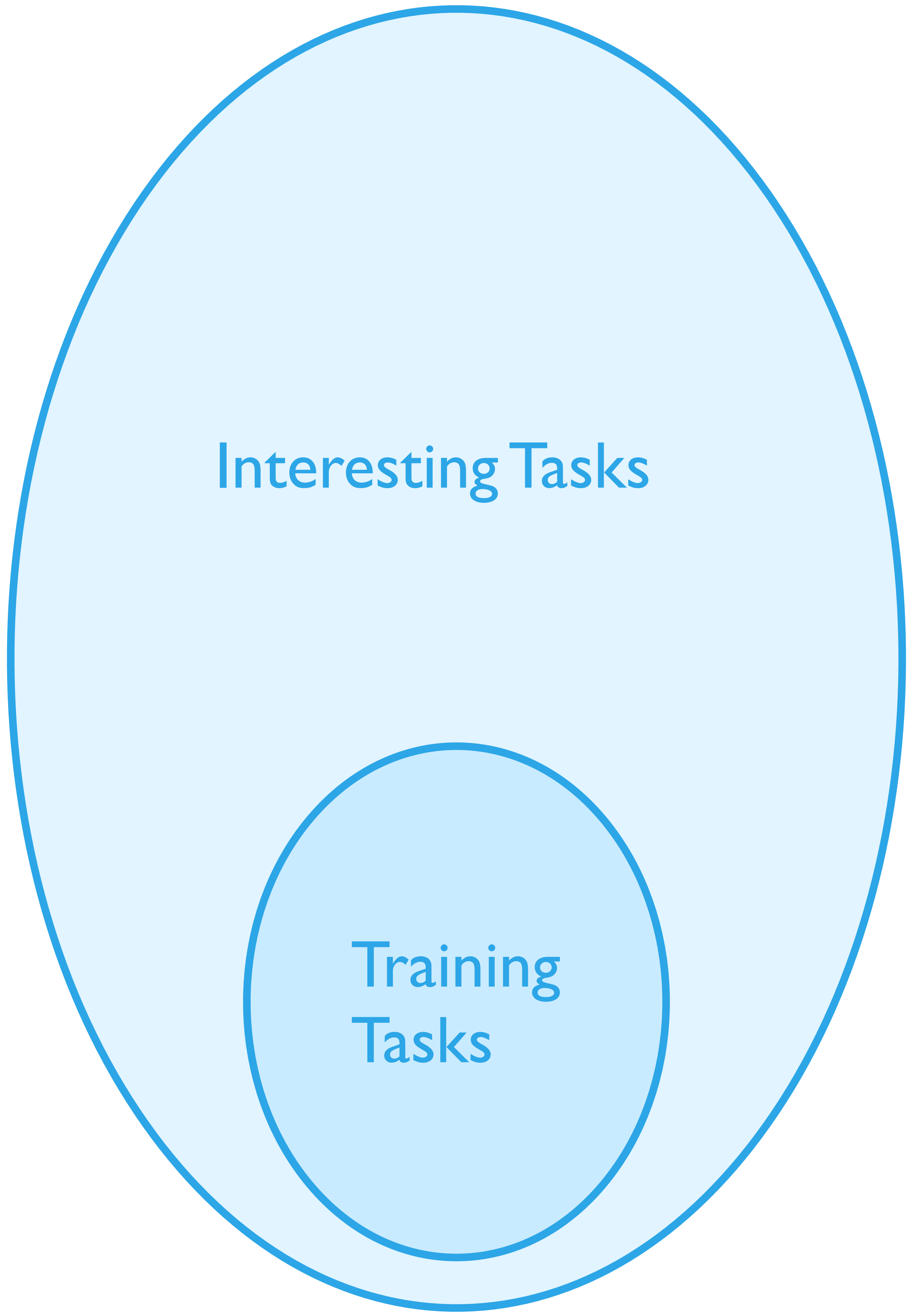


How to quickly solve new tasks?

- **How do we get *personalized* models for**
  - Education?
  - Agriculture?
  - Code?
  - Skiing?

- **Fundamental problem**
  - Always more interesting tasks than training tasks.

- **Need: models that adapt quickly**
  - How to adapt?
  - Inner workings of fast adaptation?



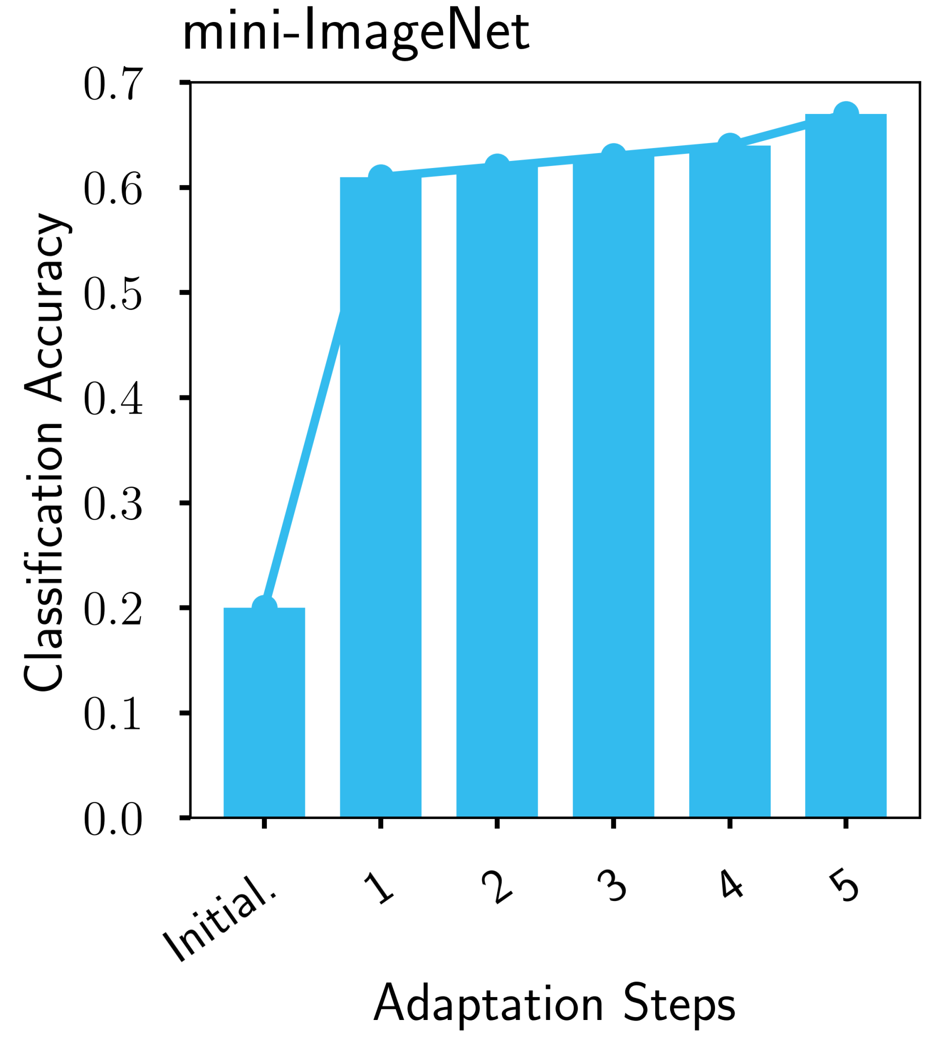
Outline

- **Part I — Meta-Learning to Adapt Fast**
  - When meta-learning fails… [*AISTATS'19*]
  - …and when it succeeds. [*ArXiv'21*]
- **Part II — Fast Adaptation without Meta-Learning**
  - Fast finetuning with rewards and more. [*In submission*]
- **Part III — Meta-Learning with Many Tasks**
  - Picking the right tasks. [*NeurIPS'21*]
  - Optimizing with many tasks. [*NeurIPS'19*]

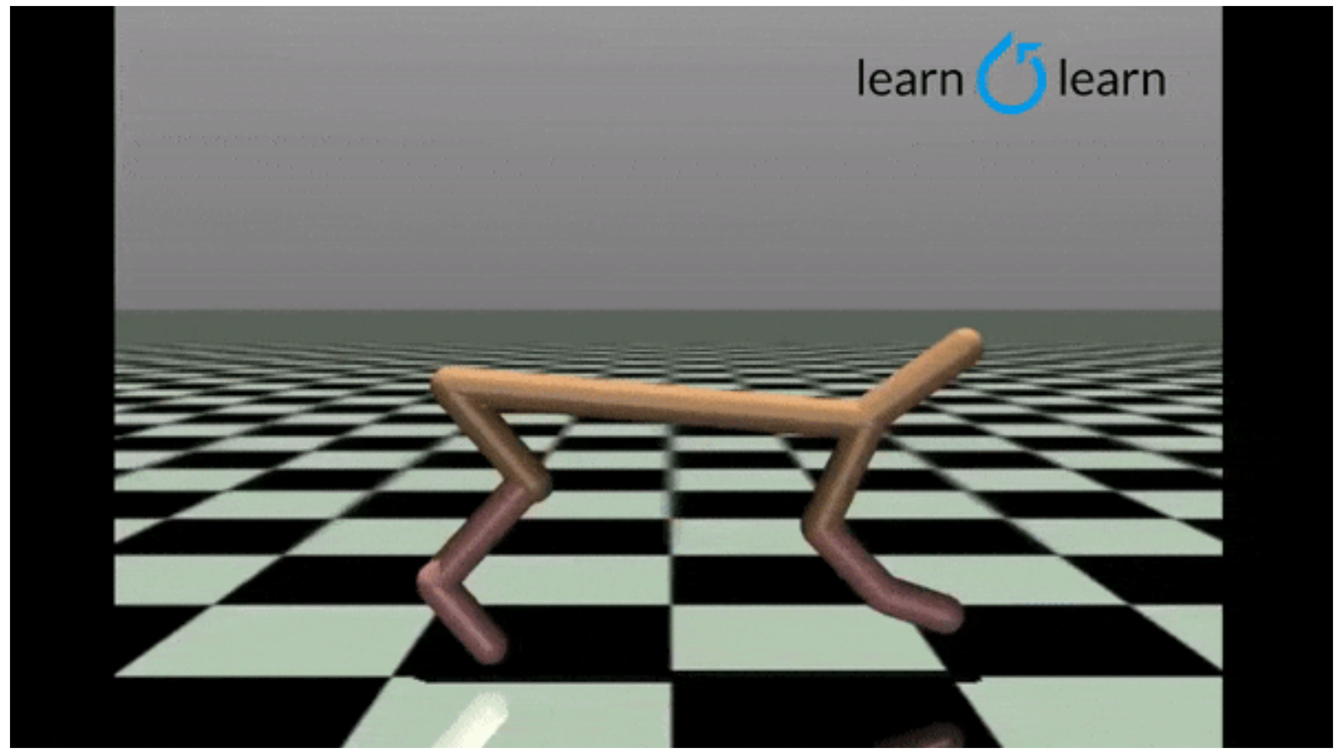
Why meta-learning?

- Definition
  - « Learn how to learn » from data.
- Core assumptions
  - Designing inductive biases is hard.
  - Learning them from data is easier.
- Success stories
  - Few-shot image classification.
  - Meta-reinforcement learning.
  - Prompting large language models.

Image Classification



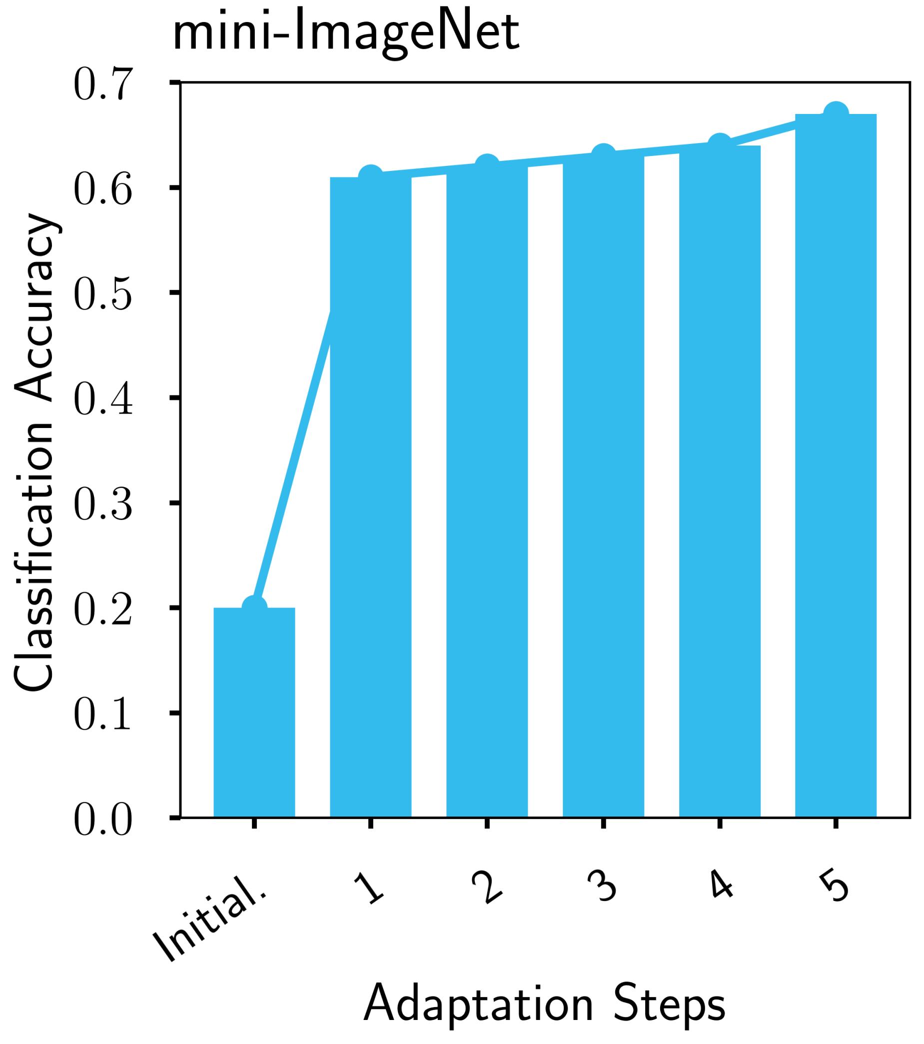
Robotics Control



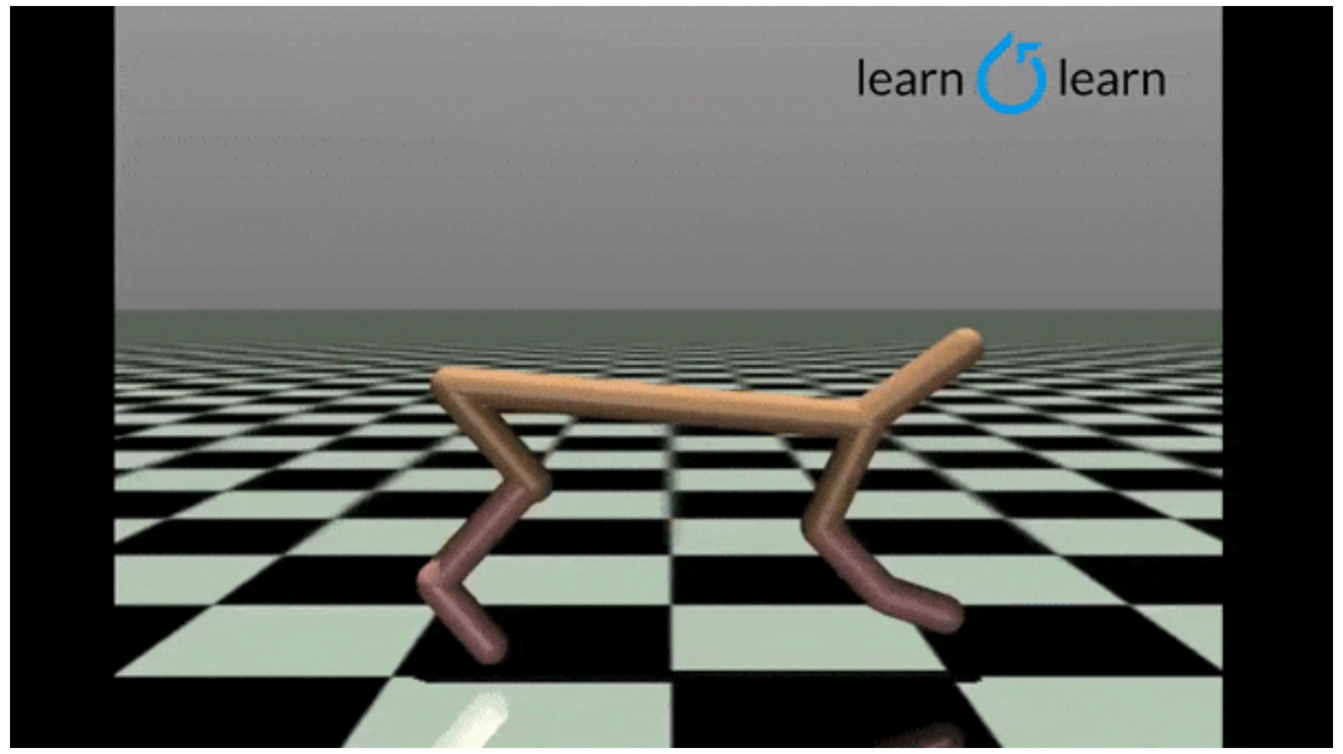
Why meta-learning?

- Definition
  - « Learn how to learn » from data.
- Core assumptions
  - Designing inductive biases is hard.
  - Learning them from data is easier.
- Success stories
  - Few-shot image classification.
  - Meta-reinforcement learning.
  - Prompting large language models.

Image Classification

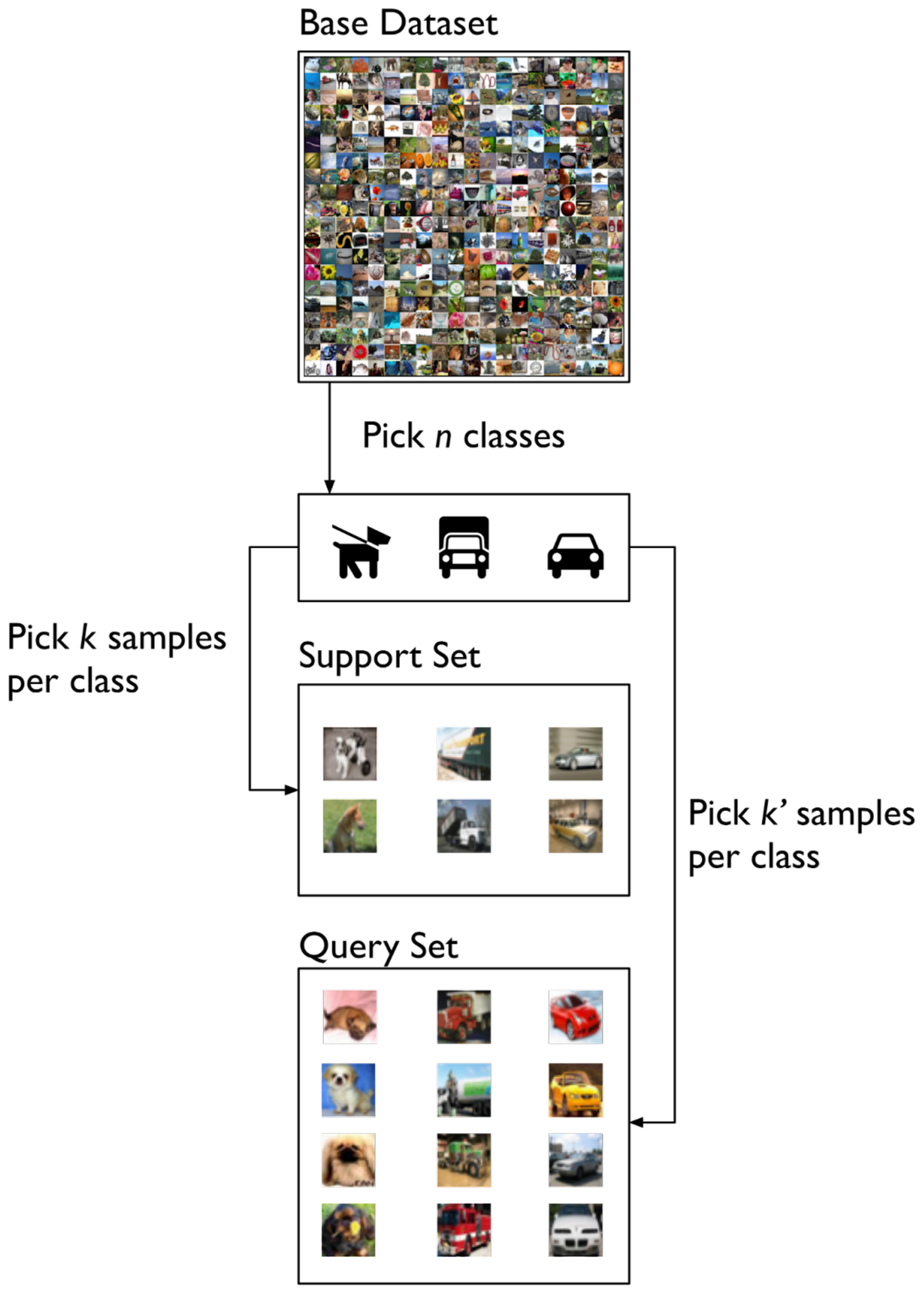


Robotics Control



Few-shot learning in a nutshell

- **Motivation**
  - Learn on a large set of train tasks.
  - Quickly solve unseen **test task with limited data**.
- **Few-shot image classification**
  - Support set for **quickly** solving the task.
  - Query set to evaluate quality of solution.
- **Other flavors**
  - Few-shot RL
  - Few-shot NLP
  - Few-shot [X]



Fei-Fei et al., « One-Shot Learning of Object Categories », PAML 2006

Model-Agnostic Meta-Learning (MAML)

- **Intuition**
  - Find **initial parameters** that adapt quickly to any task.
  - Compatible with any task-objective $\mathcal{L}_\tau$.

- **Learning objective**

$$\min_\theta \; \mathbb{E}_\tau \left[\mathcal{L}_\tau(\theta') \right]$$

$$\text{s.t.} \;\; \theta' = \theta - \alpha \nabla_\theta \mathcal{L}_\tau(\theta)$$

- **Meta-training**
  - Sample a task τ, compute $\nabla_\theta \mathcal{L}_\tau(\theta')$, and optimize θ with SGD.

Finn et al., « Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks », ICML 2017

Model-Agnostic Meta-Learning (MAML)

- **Intuition**
  - Find initial parameters that adapt quickly to any task.
  - Compatible with any task-objective $\mathcal{L}_\mathcal{T}$.

- **Learning objective**

$$\min_\theta \; \mathbb{E}_\mathcal{T} \left[\mathcal{L}_\mathcal{T}(\theta') \right]$$

$$\text{s.t. } \; \theta' = \theta - \alpha \nabla_\theta \mathcal{L}_\mathcal{T}(\theta)$$

Learned initialization

- **Meta-training**
  - Sample a task $\mathcal{T}$, compute $\nabla_\theta \mathcal{L}_\mathcal{T}(\theta')$, and optimize θ with SGD.

Finn et al., « Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks », ICML 2017

Model-Agnostic Meta-Learning (MAML)

- **Intuition**
  - Find initial parameters that adapt quickly to any task.
  - Compatible with any task-objective $\mathcal{L}_\tau$.

- **Learning objective**

$$\min_\theta \; \mathbb{E}_\tau \left[\mathcal{L}_\tau(\theta')\right]$$

$$\text{s.t.} \;\; \theta' = \theta - \alpha \nabla_\theta \mathcal{L}_\tau(\theta)$$

Learned initialization

- **Meta-training**
  - Sample a task τ, compute $\nabla_\theta \mathcal{L}_\tau(\theta')$, and optimize θ with SGD.

Finn et al., « Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks », ICML 2017

Model-Agnostic Meta-Learning (MAML)

- **Intuition**
  - Find initial parameters that adapt quickly to any task.
  - Compatible with any task-objective $\mathcal{L}_\tau$.

- **Learning objective**

Adapted parameters

$$\min_\theta \; \mathbb{E}_\tau \left[\mathcal{L}_\tau(\theta')\right]$$

$$\text{s.t.} \;\; \theta' = \theta - \alpha \nabla_\theta \mathcal{L}_\tau(\theta)$$

Learned initialization

- **Meta-training**
  - Sample a task τ, compute $\nabla_\theta \mathcal{L}_\tau(\theta')$, and optimize θ with SGD.

Finn et al., « Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks », ICML 2017

Model-Agnostic Meta-Learning (MAML)

- **Intuition**
  - Find initial parameters that adapt quickly to any task.
  - Compatible with any task-objective $\mathcal{L}_\tau$.

- **Learning objective**

Adapted parameters

$$\min_\theta \ \mathbb{E}_\tau \left[\mathcal{L}_\tau(\theta')\right]$$

$$\text{s.t.} \ \ \theta' = \theta - \alpha \nabla_\theta \mathcal{L}_\tau(\theta)$$

Learned initialization

- **Meta-training**
  - Sample a task τ, compute $\nabla_\theta \mathcal{L}_\tau(\theta')$, and optimize θ with SGD.

Model-Agnostic Meta-Learning (MAML)

- Intuition
  - Find initial parameters that adapt quickly to any task.
  - Compatible with any task-objective $\mathcal{L}_\tau$.

- Learning objective

Adapted parameters

$$\min_\theta \ \mathbb{E}_\tau \left[\mathcal{L}_\tau(\theta')\right]$$

$$\text{s.t.} \ \ \theta' = \theta - \alpha \nabla_\theta \mathcal{L}_\tau(\theta)$$

Learned initialization

- Meta-training
  - Sample a task τ, compute $\nabla_\theta \mathcal{L}_\tau(\theta')$, and optimize θ with SGD.

Model-Agnostic Meta-Learning (MAML)

- **Intuition**
  - Find **initial parameters** that adapt quickly to any task.
  - Compatible with any task-objective $\mathcal{L}_\tau$.

- **Learning objective**

Adapted parameters

$$\min_\theta \; \mathbb{E}_\tau \left[\mathcal{L}_\tau(\theta') \right]$$

$$\text{s.t.} \;\; \theta' = \theta - \alpha \nabla_\theta \mathcal{L}_\tau(\theta)$$

Learned initialization

- **Meta-training**
  - Sample a task τ, compute $\nabla_\theta \mathcal{L}_\tau(\theta')$, and optimize θ with SGD.

Finn et al., « Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks », ICML 2017

Meta-Learning to Adapt Fast

Part I

Q1: What is meta-learned with MAML?

- Model setups
  - Shallow: $\hat{y} = cx$
  - Deep: $\hat{y} = abx$
  - Both encode a linear function.

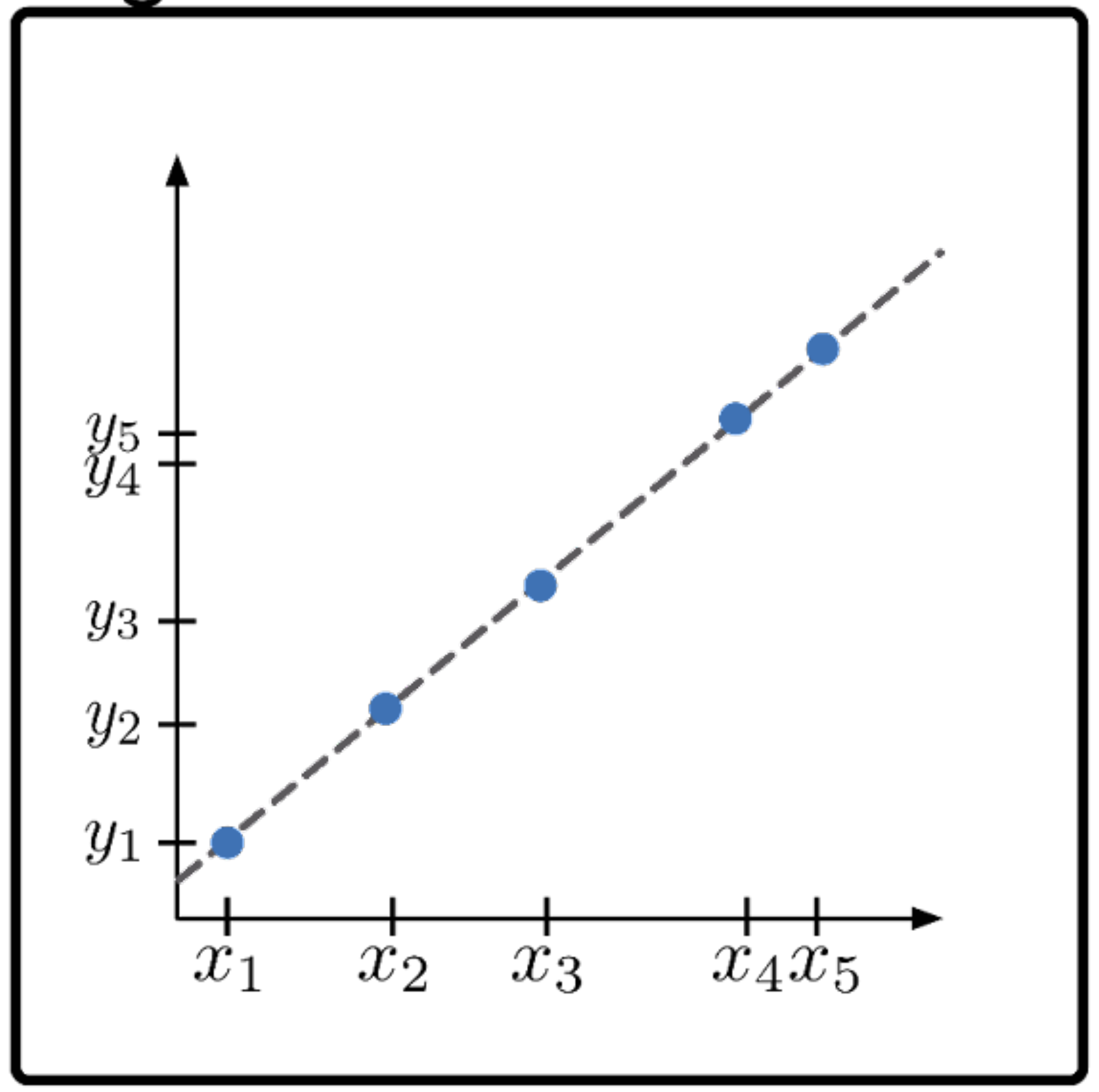
- Task setups
  - Linear regression:

$$\mathcal{L}_{\mathcal{T}} = \frac{1}{2}(\hat{y} - y_{\mathcal{T}})^2$$

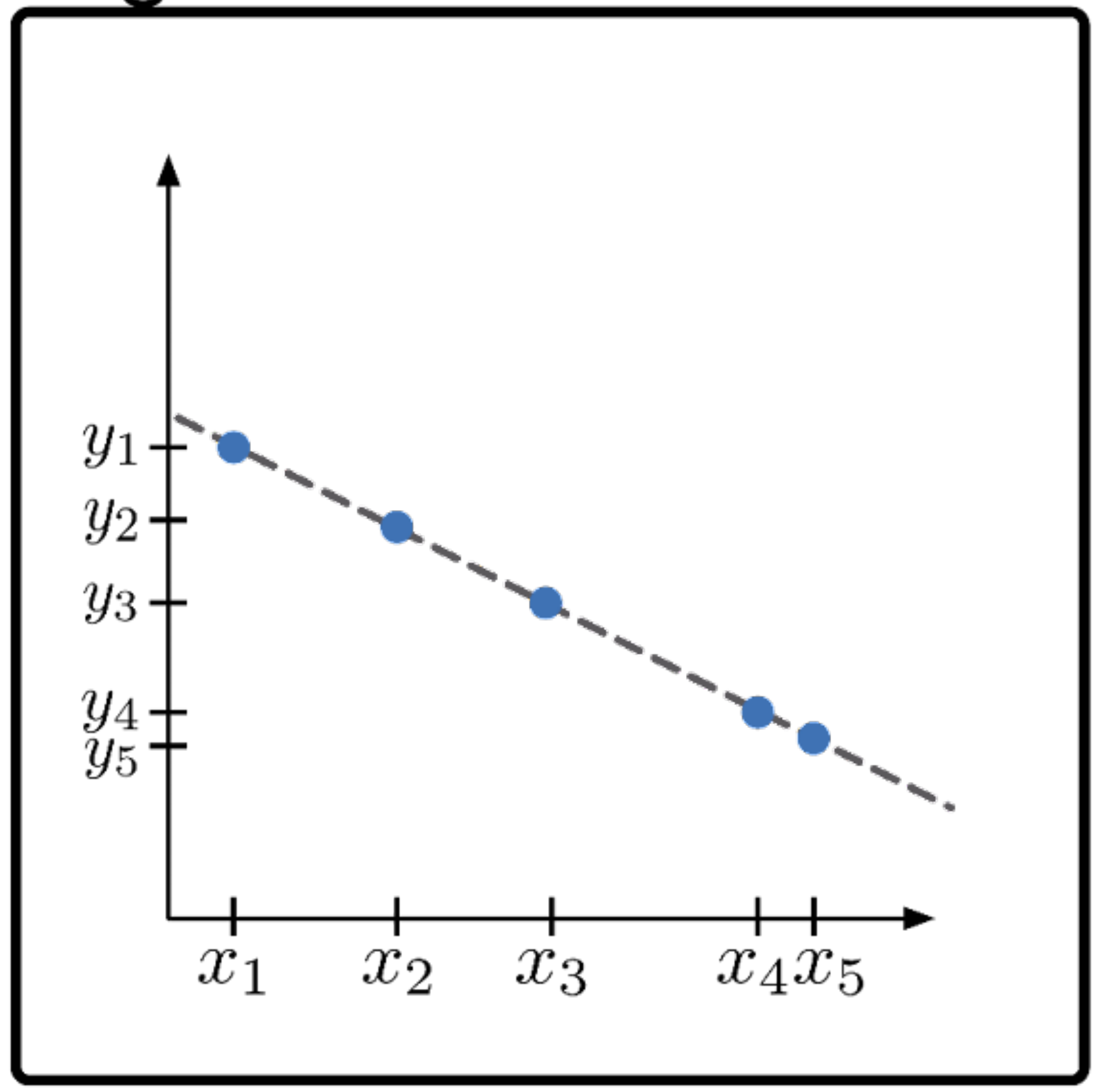
where: $x, y_{\mathcal{T}}, a, b, c \in \mathbb{R}$
with:

- 10k $(x, y_{\mathcal{T}})$ samples,
- 1000 tasks $\mathcal{T}$,
- x's are **constant across tasks**.

Regression — Task 1



Regression — Task 2



… and for tasks 3 to 1,000.

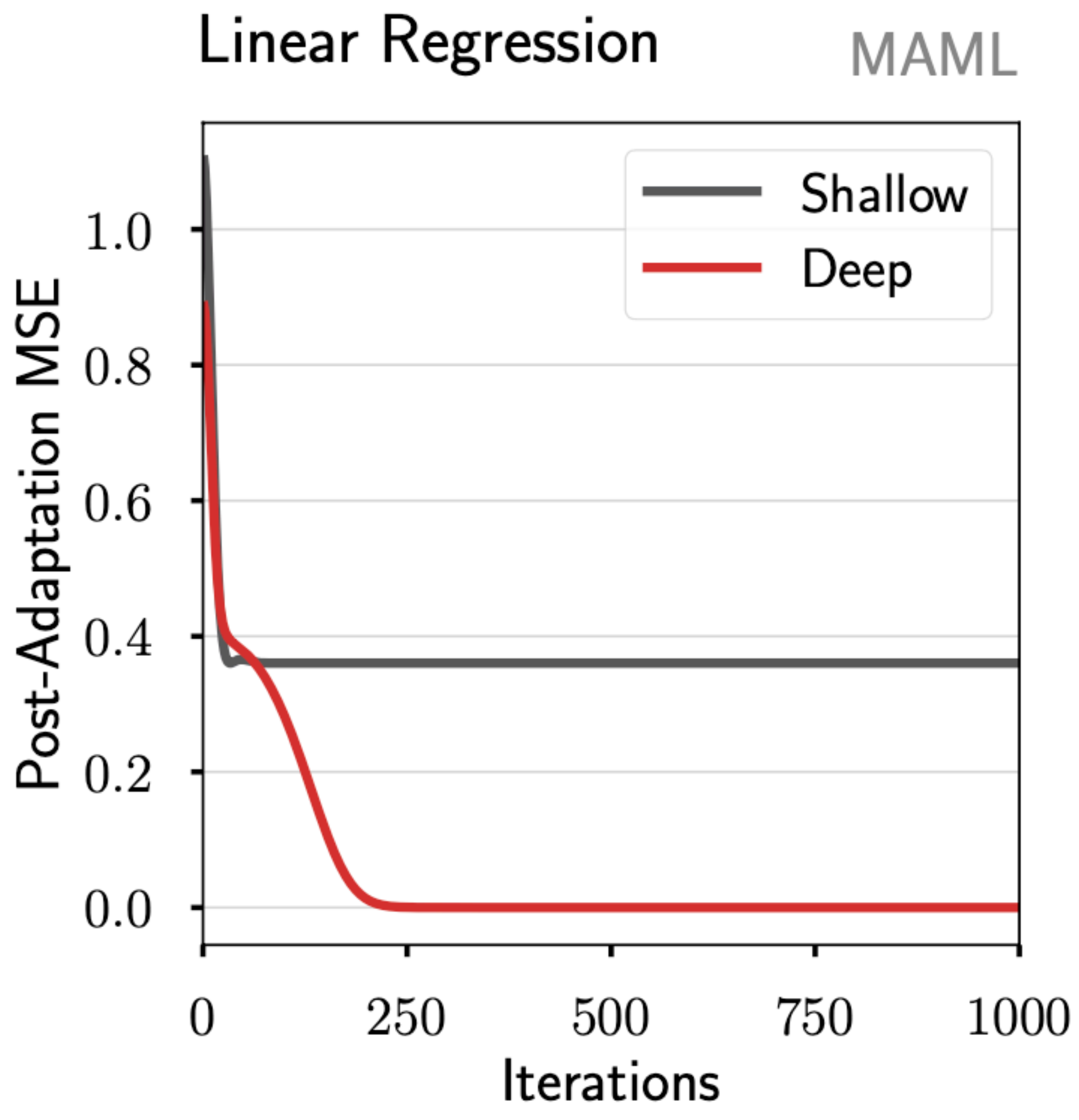
MAML fails on linear regression?

- **Model setups**
  - Shallow: $\hat{y} = cx$
  - Deep: $\hat{y} = abx$
  - Both encode a linear function.
- **Task setups**
  - Linear regression:

$$\mathcal{L}_{\tau} = \frac{1}{2}(\hat{y} - y_{\tau})^2$$

where: $x, y_{\tau}, a, b, c \in \mathbb{R}$
with:

- 10k (x, y_{τ}) samples,
- 1000 tasks τ,
- x's are **constant across tasks**.



MAML fails on linear regression?

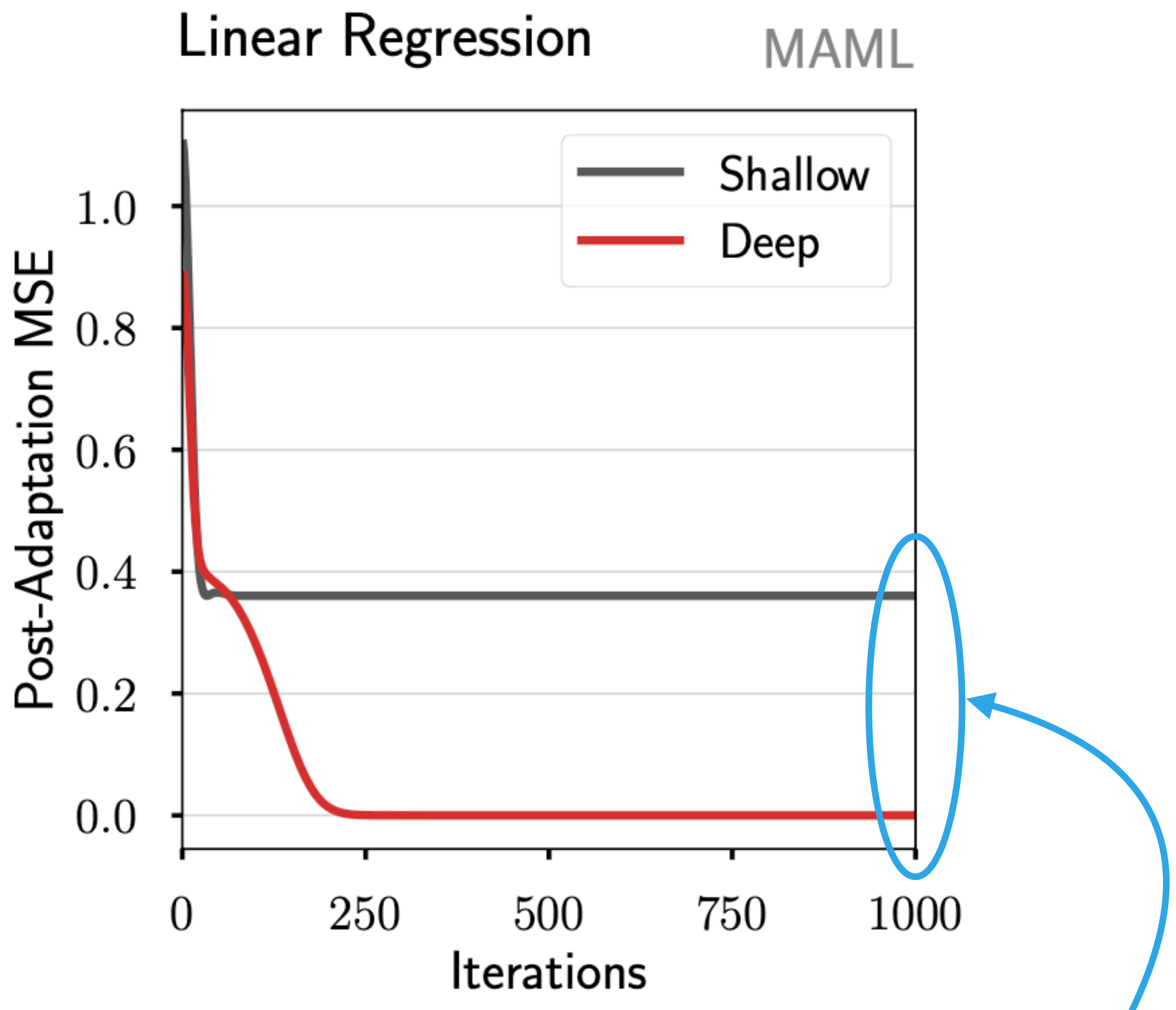
- **Model setups**
  - Shallow: $\hat{y} = cx$
  - Deep: $\hat{y} = abx$
  - Both encode a linear function.

- **Task setups**
  - Linear regression:

$$\mathcal{L}_\tau = \frac{1}{2}(\hat{y} - y_\tau)^2$$

where: $x, y_\tau, a, b, c \in \mathbb{R}$
with:

- 10k (x, y_τ) samples,
- 1000 tasks τ,
- x's are **constant across tasks**.



Shallow fails; Deep succeeds.

MAML fails on linear regression?

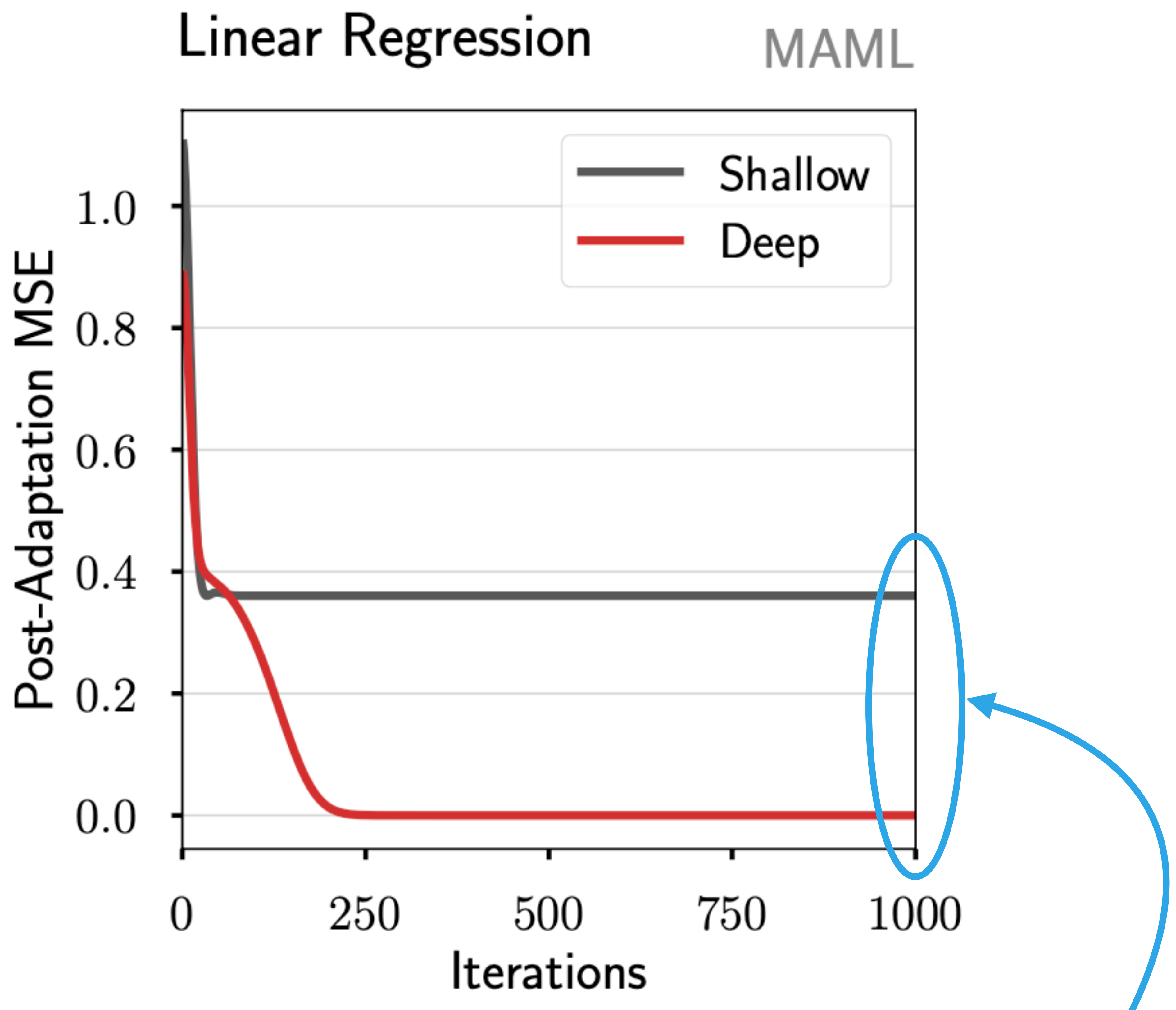
- **Model setups**
  - Shallow: $\hat{y} = cx$
  - Deep: $\hat{y} = abx$
  - Both encode a linear function.

- **Task setups**
  - Linear regression:

$$\mathcal{L}_\tau = \frac{1}{2}(\hat{y} - y_\tau)^2$$

where: $x, y_\tau, a, b, c \in \mathbb{R}$
with:

- 10k (x, y_τ) samples,
- 1000 tasks τ,
- x's are **constant across tasks**.



Shallow fails; Deep succeeds.

Depth enables meta-learning

- Shallow: $$\frac{\partial}{\partial c}\frac{1}{2}(cx - y)^2 = (cx - y)x$$
- Deep: $$\frac{\partial}{\partial b}\frac{1}{2}(abx - y)^2 = (abx - y)ax$$

Depth enables meta-learning

- Shallow: $$\frac{\partial}{\partial c}\frac{1}{2}(cx - y)^2 = (cx - y)x$$

Same error terms

- Deep: $$\frac{\partial}{\partial b}\frac{1}{2}(abx - y)^2 = (abx - y)ax$$

Depth enables meta-learning

- Shallow: $$\frac{\partial}{\partial c}\frac{1}{2}(cx - y)^2 = (cx - y)x$$
- Deep: $$\frac{\partial}{\partial b}\frac{1}{2}(abx - y)^2 = (abx - y)ax$$

Same error terms

Extra degree of freedom (learnable!)

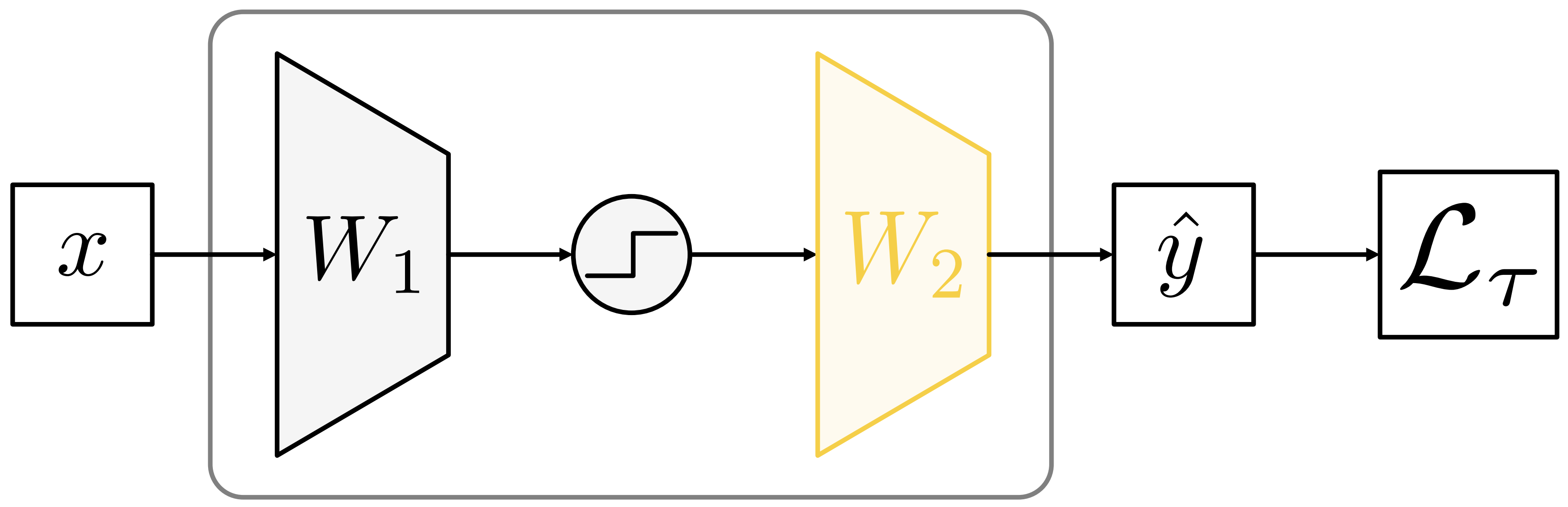
Depth enables meta-learning

- Derivative of linear models
  - Shallow: $\frac{\partial}{\partial c}\frac{1}{2}(cx-y)^2 = (cx-y)x$
  - Deep: $\frac{\partial}{\partial b}\frac{1}{2}(abx-y)^2 = (abx-y)ax$

Same error terms

Extra degree of freedom (learnable!)

- Derivatives of non-linear models
  - Forward pass: $\hat{y} = W_2\,\sigma(W_1 x)$
  - Backward pass:



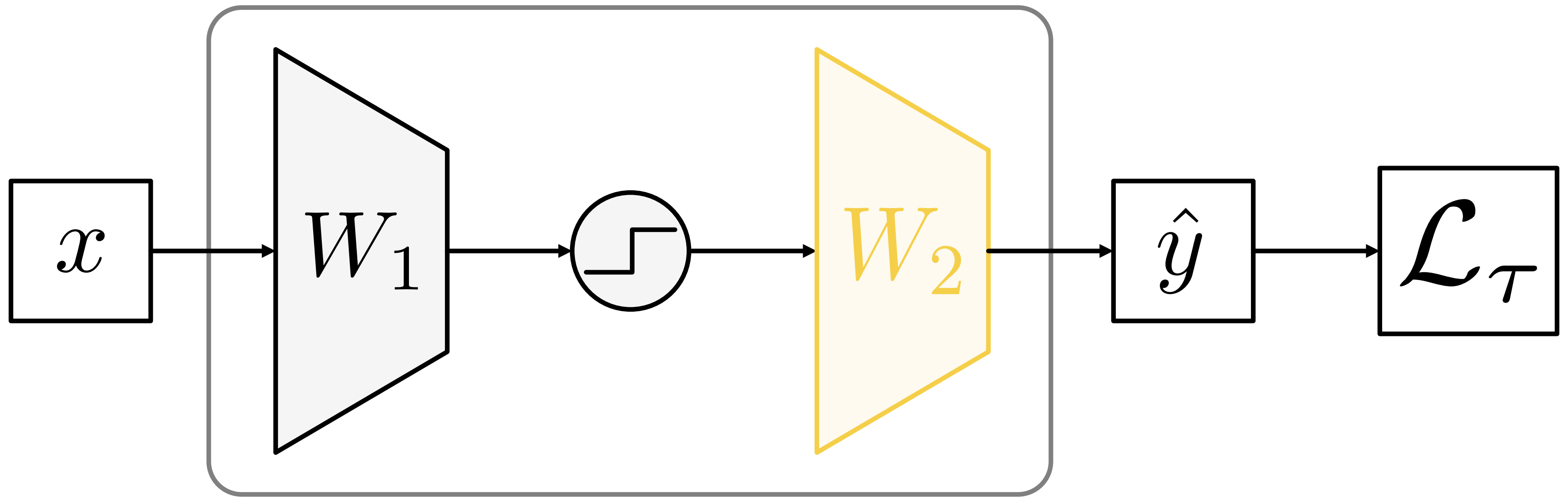
Depth enables meta-learning

- Derivative of linear models
  - Shallow: $\frac{\partial}{\partial c}\frac{1}{2}(cx-y)^2 = (cx-y)x$
  - Deep: $\frac{\partial}{\partial b}\frac{1}{2}(abx-y)^2 = (abx-y)ax$

Same error terms

Extra degree of freedom (learnable!)

- Derivatives of non-linear models
  - Forward pass: $\hat{y} = W_2\,\sigma(W_1 x)$
  - Backward pass: $$\frac{\partial \mathcal{L}_\mathcal{T}}{\partial W_1} = \frac{\partial \mathcal{L}_\mathcal{T}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{z_1}{W_1} = \frac{\partial \mathcal{L}_\mathcal{T}}{\partial \hat{y}} \cdot W_2 \cdot \frac{z_1}{W_1}$$



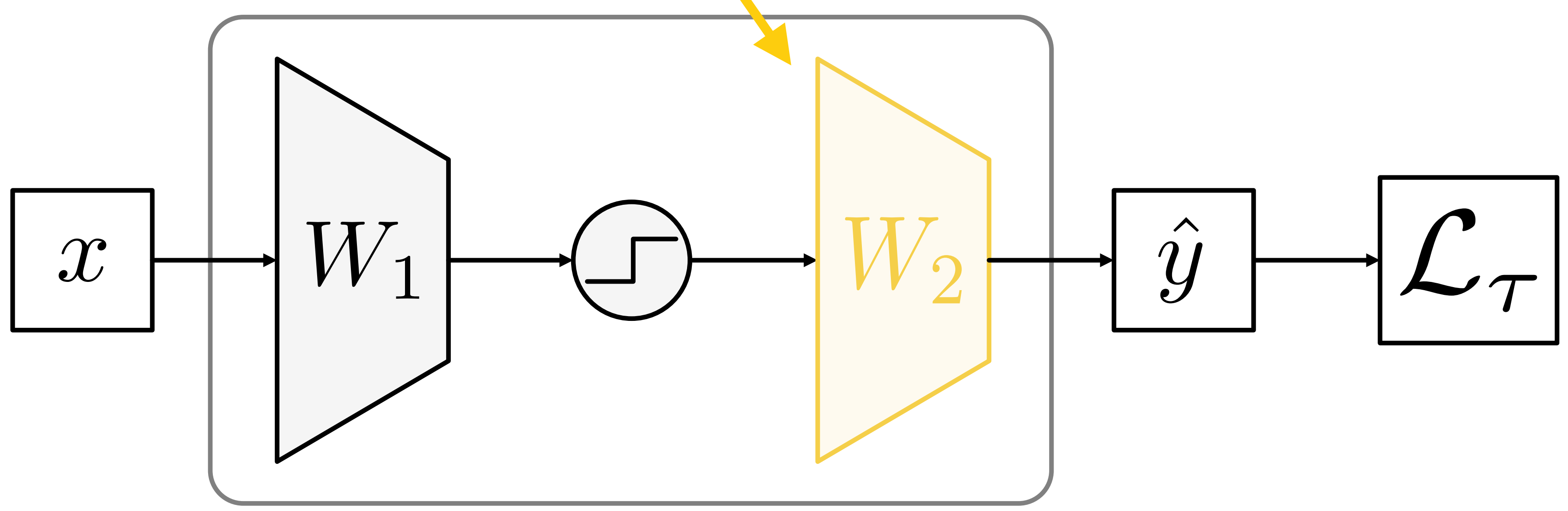
Depth enables meta-learning

- Derivative of linear models
  - Shallow: $\frac{\partial}{\partial c}\frac{1}{2}(cx-y)^2 = (cx-y)x$
  - Deep: $\frac{\partial}{\partial b}\frac{1}{2}(abx-y)^2 = (abx-y)ax$

Same error terms

Extra degree of freedom (learnable!)

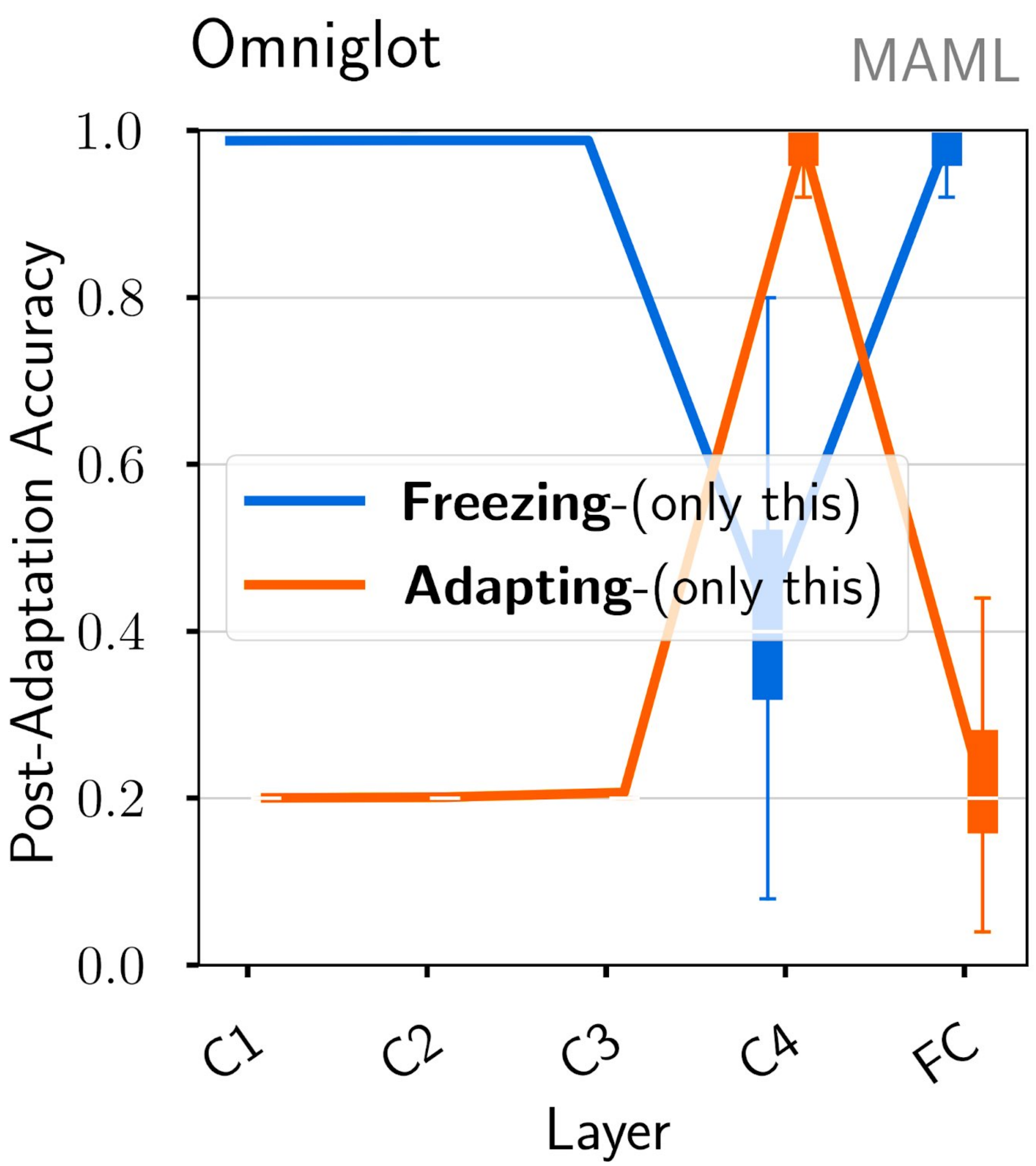
- Derivatives of non-linear models
  - Forward pass: $\hat{y} = W_2\,\sigma(W_1 x)$
  - Backward pass: $$\frac{\partial \mathcal{L}_\tau}{\partial W_1} = \frac{\partial \mathcal{L}_\tau}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{z_1}{W_1} = \frac{\partial \mathcal{L}_\tau}{\partial \hat{y}} \cdot W_2 \cdot \frac{z_1}{W_1}$$



The structure of MAML solutions

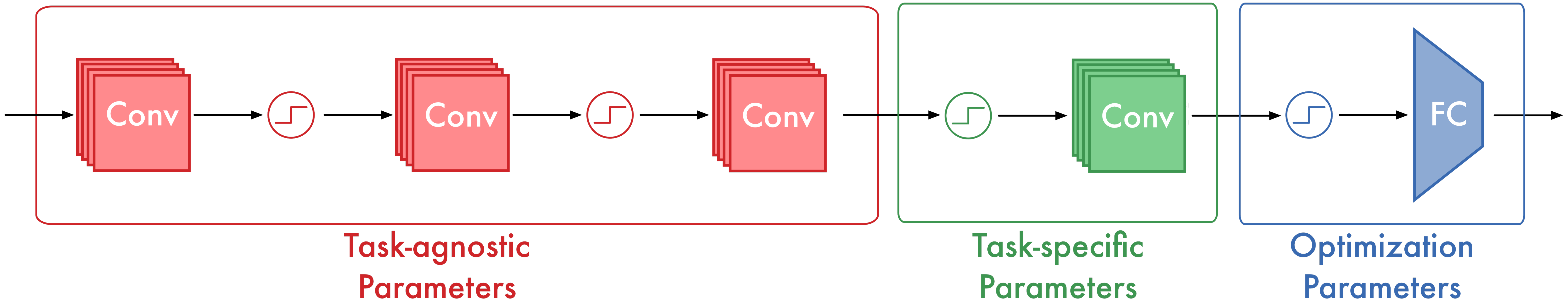
Does our theory hold in practice?

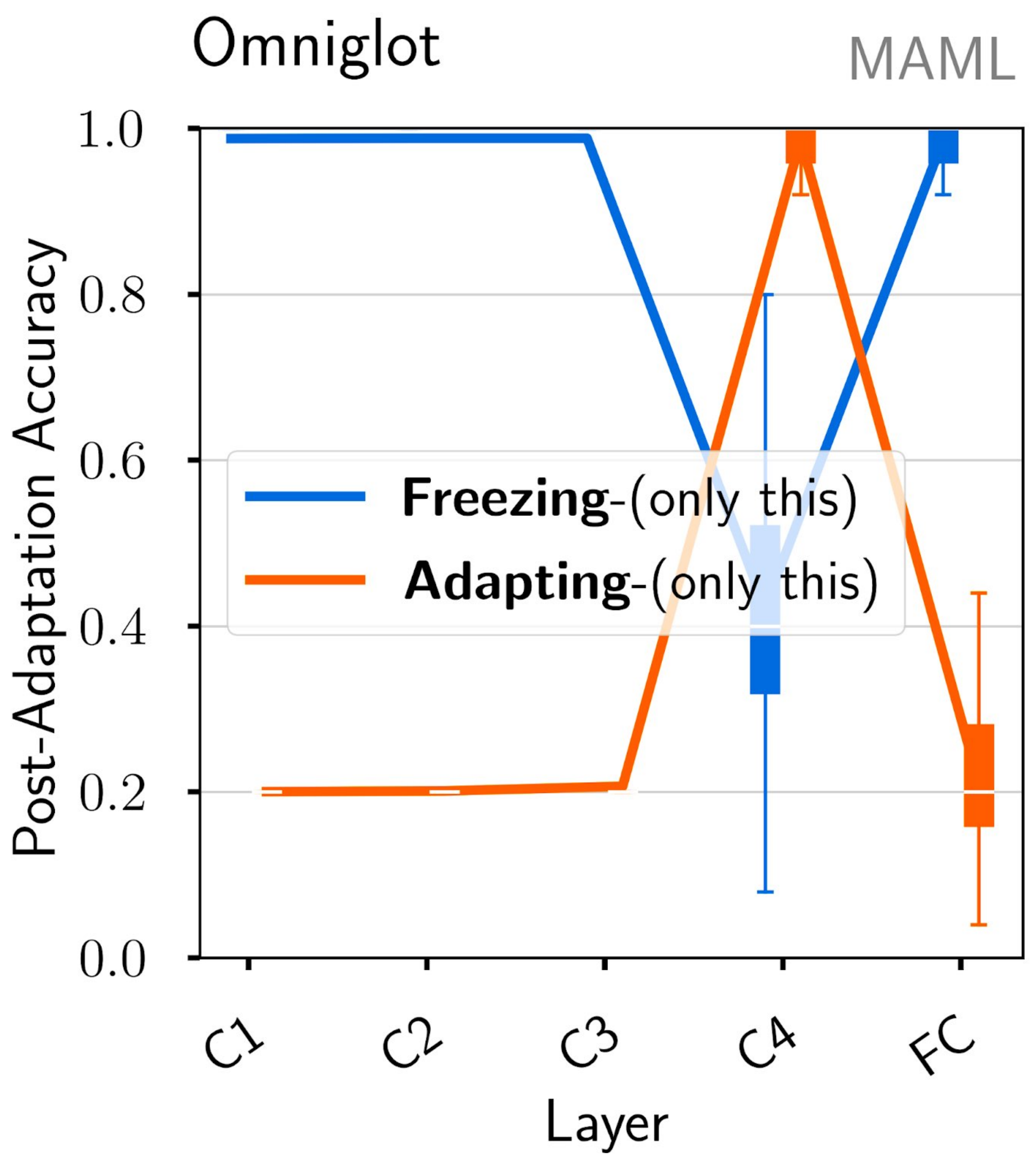
- **Experimental setup**
  - Omniglot[1]: classify 1600+ characters.
  - Meta-train a CNN until convergence.
- **Layerwise analysis**
  - Freezing-X: only X is frozen (rest is adapted).
  - Adapting-X: only X is adapted (rest is frozen).



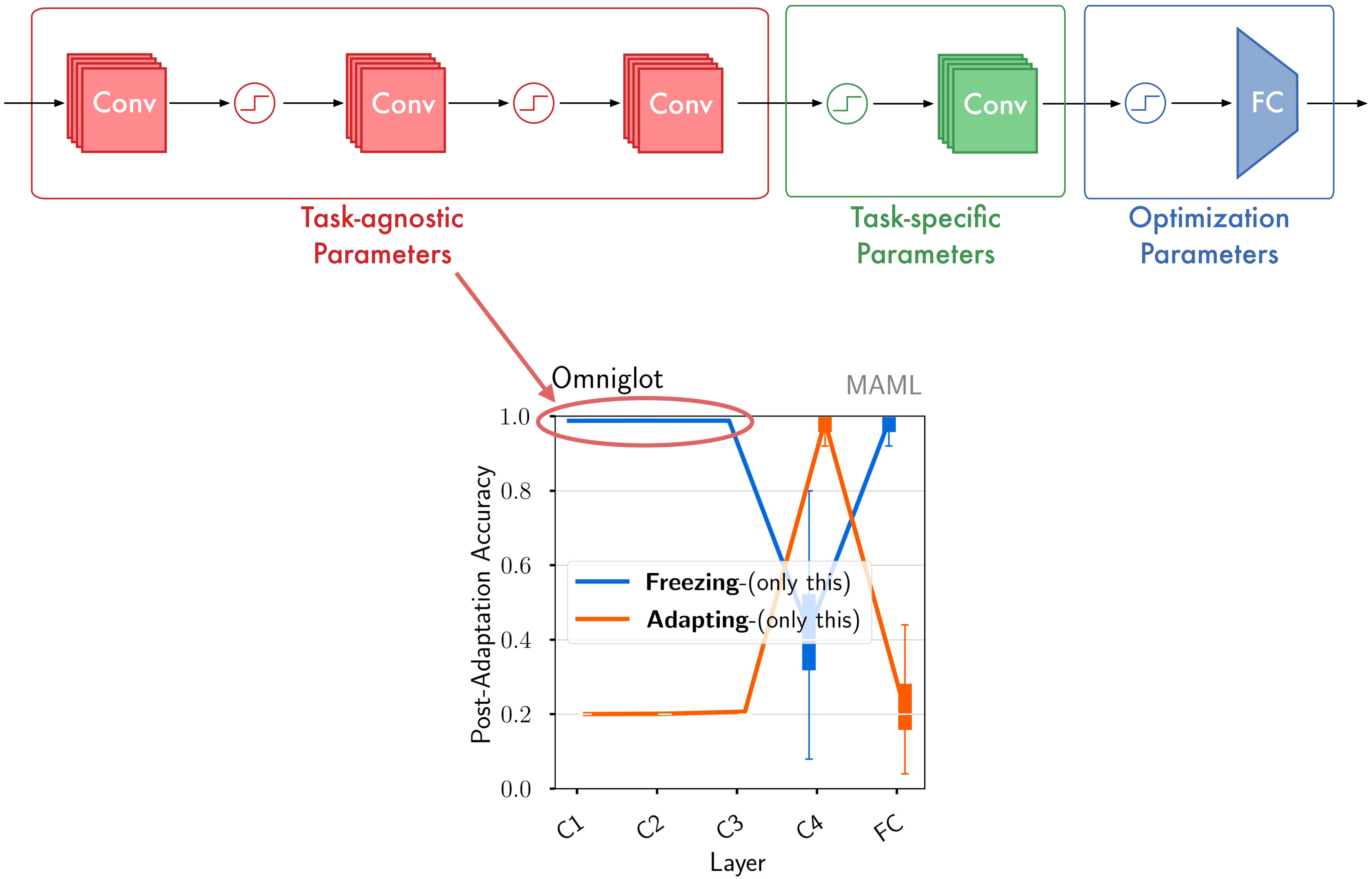
1: Lake et al., « Human-level concept learning through probabilistic program induction », Science 2015

The structure of MAML solutions

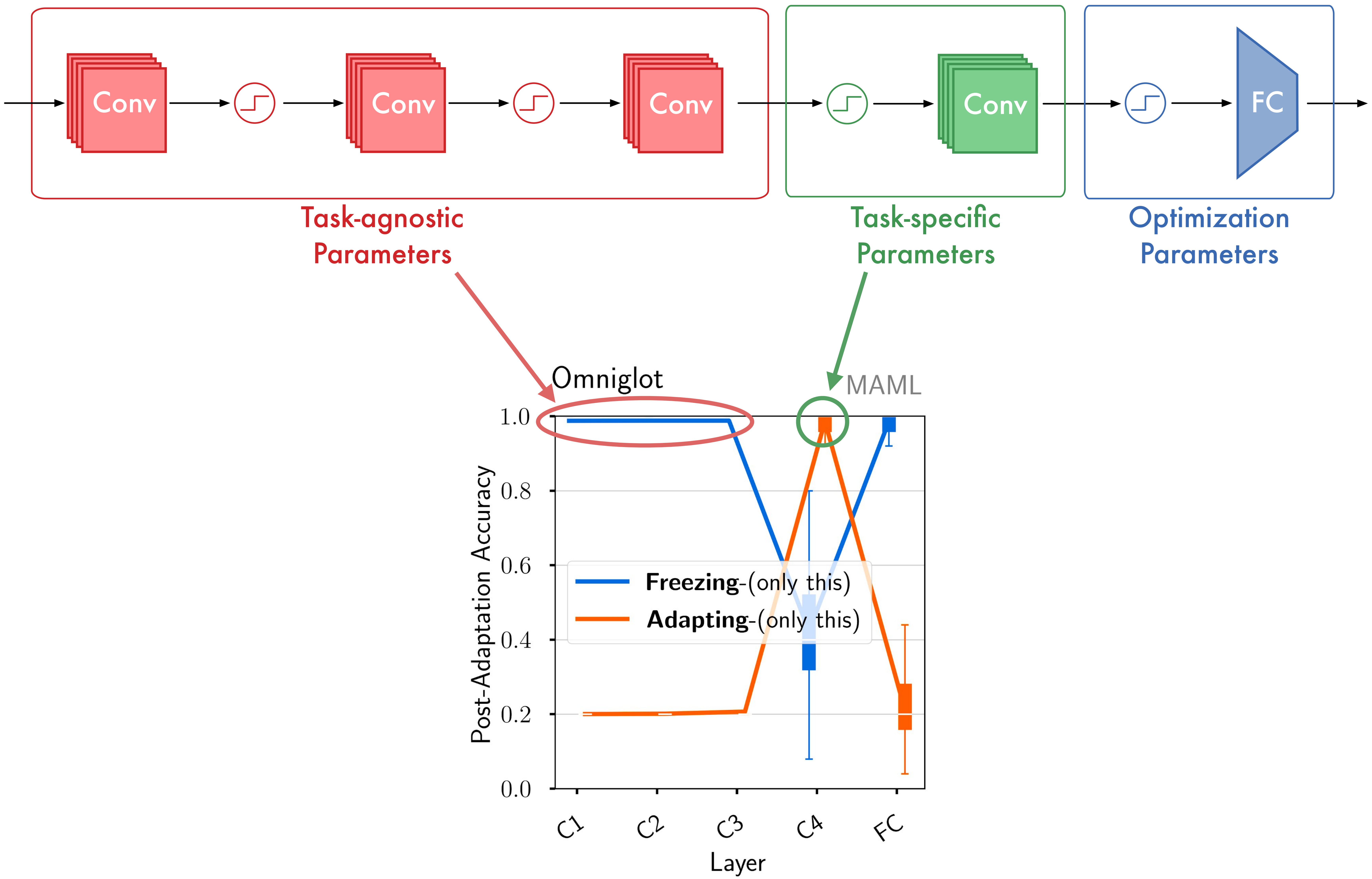




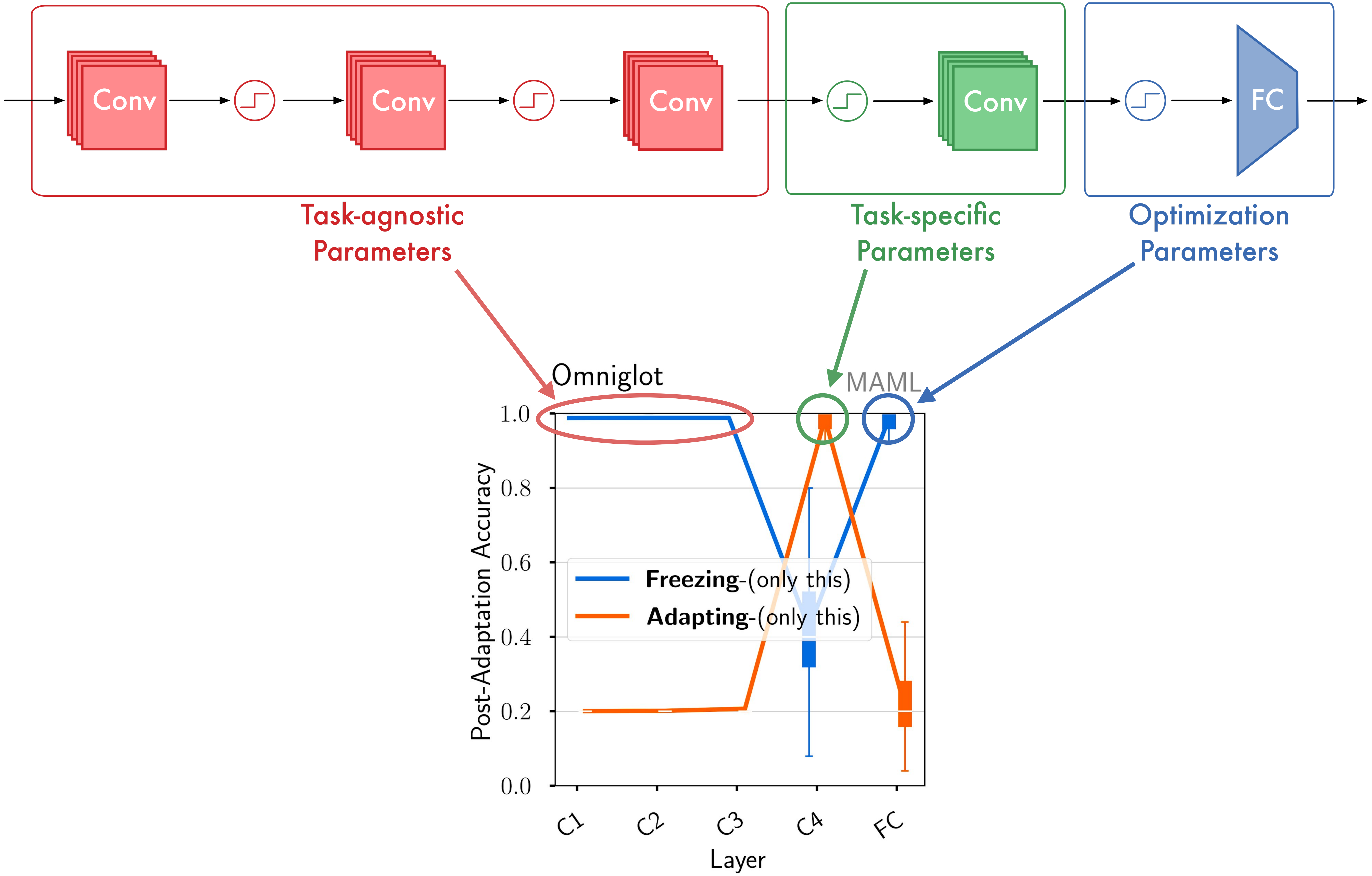
The structure of MAML solutions



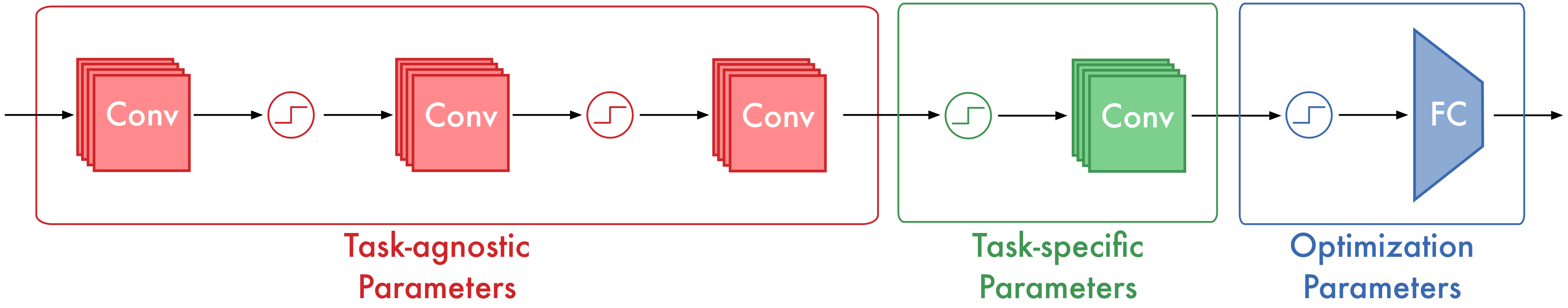
The structure of MAML solutions



The structure of MAML solutions



The structure of MAML solutions



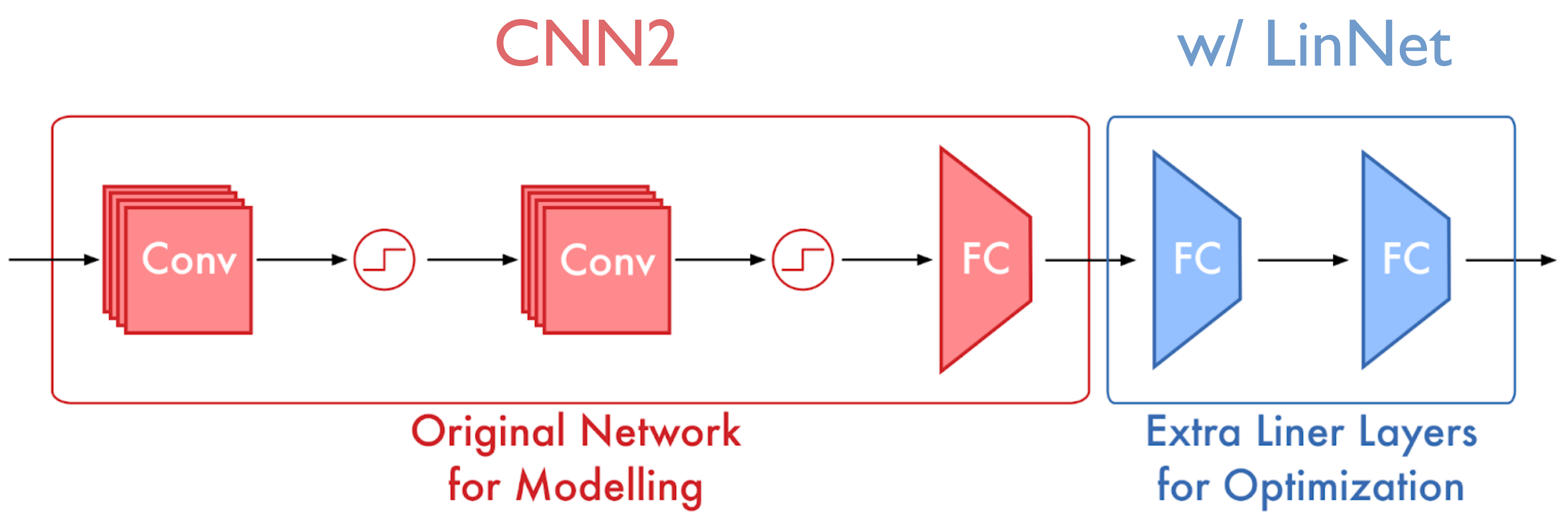
Similar story holds across settings.

- Datasets
  - Omniglot
  - mini-ImageNet
  - CIFAR-FS
- Architectures
  - CNN4
  - CNN6
  - ResNet9
- Algorithms
  - MAML
  - Reptile[1]

1: *Nichol et al., « On First-Order Meta-Learning Algorithms », ArXiv 2018*

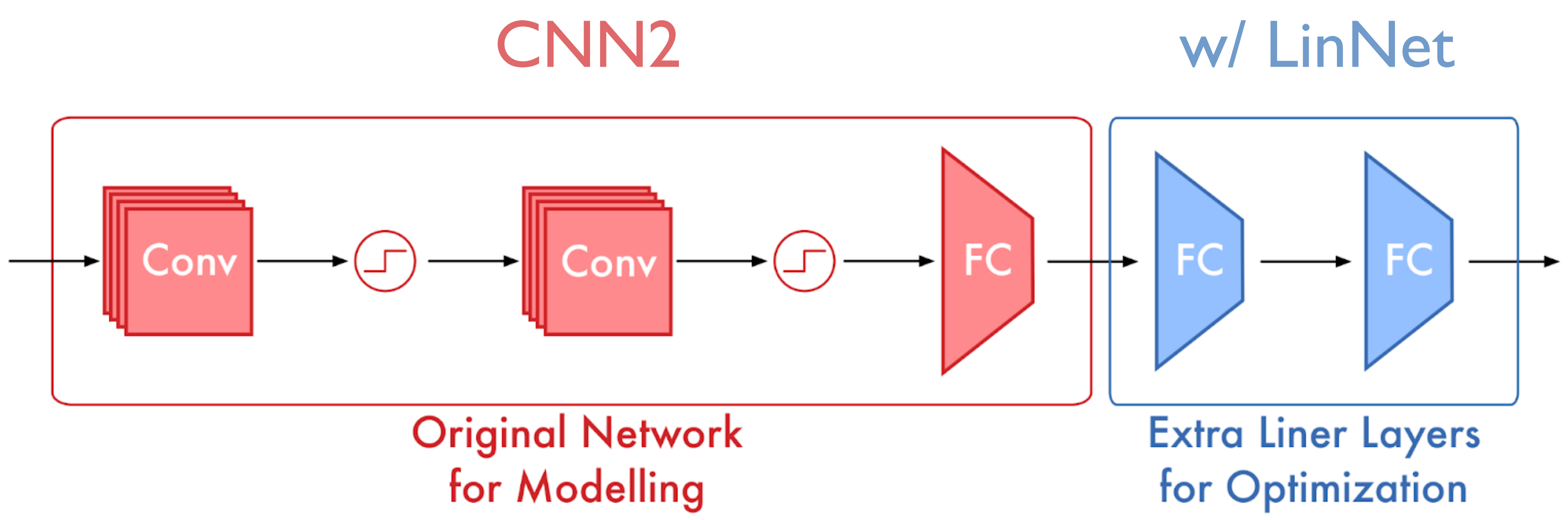
Can linear layers improve meta-learning?

- CNN4
  - 4x Conv + 1x FC
- CNN2
  - 2x Conv + 1x FC
- CNN2 w/ LinNet
  - 2x Conv + 3x FC
  - No activations on last 2x FC!



Can linear layers improve meta-learning?

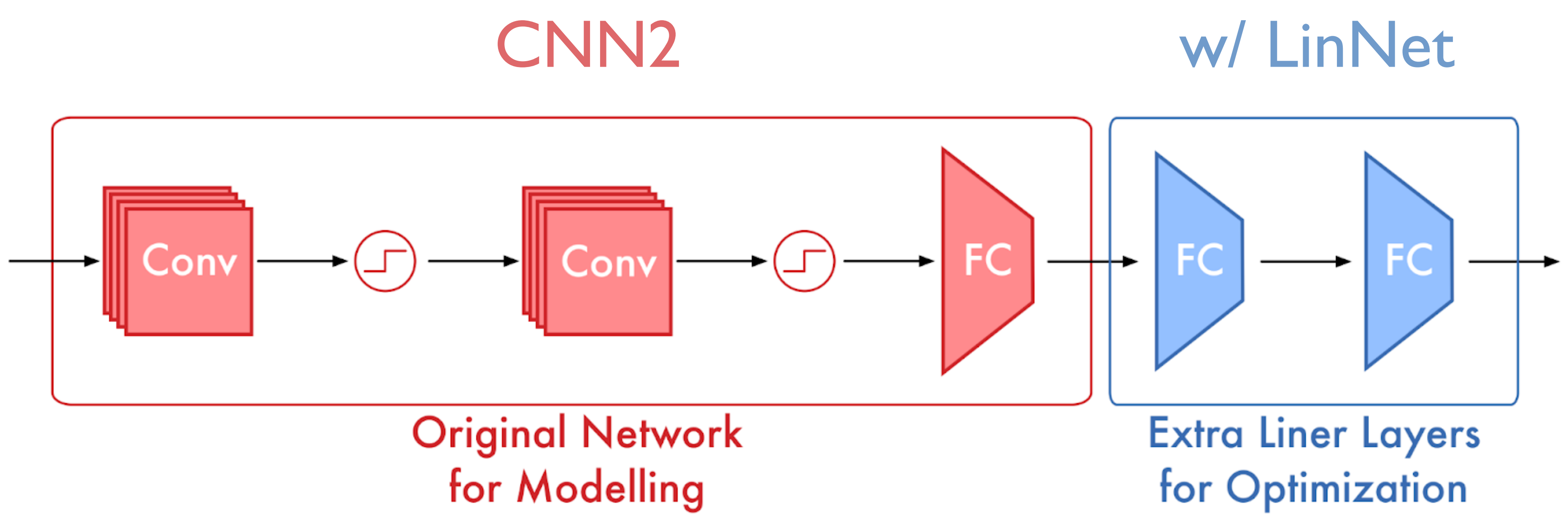
- CNN4
  - 4x Conv + 1x FC
- CNN2
  - 2x Conv + 1x FC
- CNN2 w/ LinNet
  - 2x Conv + 3x FC
  - No activations on last 2x FC!



	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet
CIFAR-FS	70.9%	62.2%	**66.1%**
mini-ImageNet	64.1%	52.6%	**60.5%**
Omniglot	98.5%	66.8%	**88.1%**

Can linear layers improve meta-learning?

- CNN4
  - 4x Conv + 1x FC
- CNN2
  - 2x Conv + 1x FC
- CNN2 w/ LinNet
  - 2x Conv + 3x FC
  - No activations on last 2x FC!

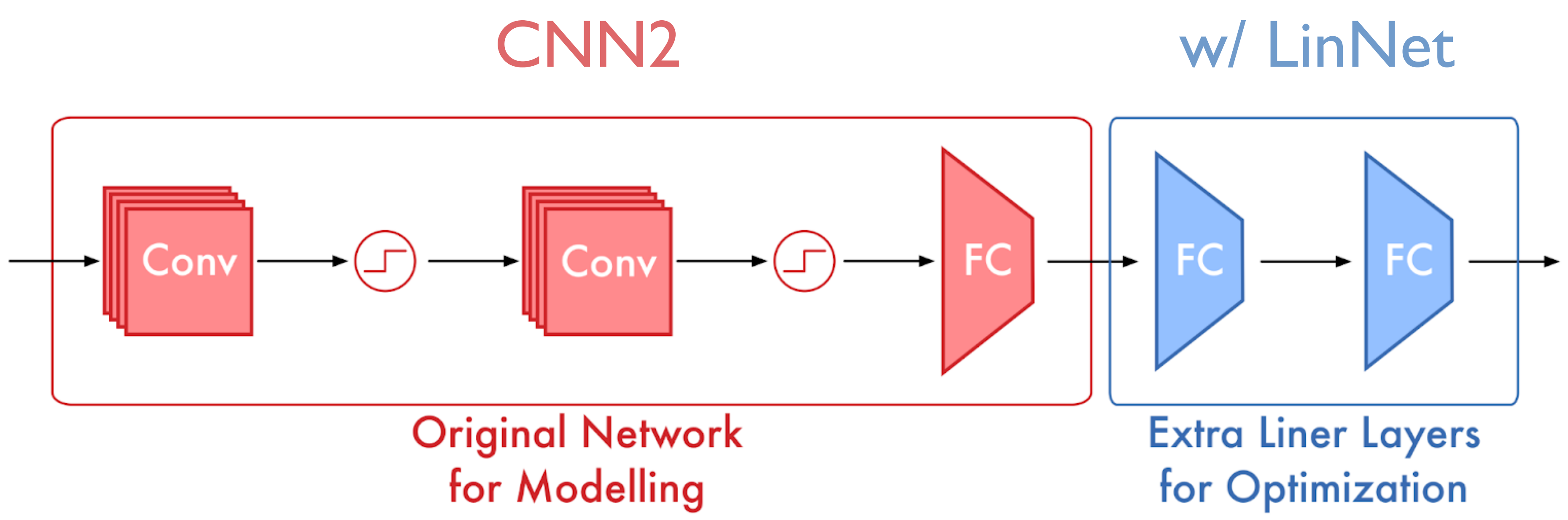


	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet
CIFAR-FS	70.9%	62.2%	**66.1%**
mini-ImageNet	64.1%	52.6%	**60.5%**
Omniglot	98.5%	66.8%	**88.1%**

30%+ degradation
due to shallow

Can linear layers improve meta-learning?

- CNN4
  - 4x Conv + 1x FC
- CNN2
  - 2x Conv + 1x FC
- CNN2 w/ LinNet
  - 2x Conv + 3x FC
  - No activations on last 2x FC!



	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet
CIFAR-FS	70.9%	62.2%	**66.1%**
mini-ImageNet	64.1%	52.6%	**60.5%**
Omniglot	98.5%	66.8%	**88.1%**

30%+ degradation due to shallow

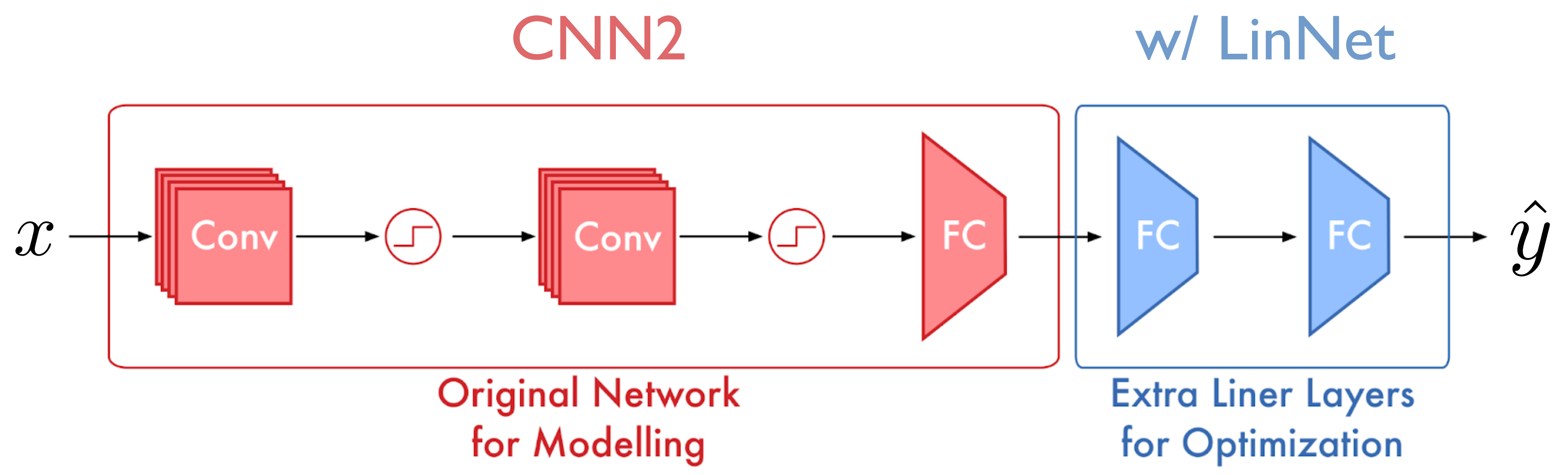
Recover 20%+ with linear layers

Meta-Optimizer for *faster* adaptation

- **How to bridge remaining 10%?**
  - Linear layers ➞ linear gradient transformation.
  - Non-linear layers ➞ cannot be collapsed ➞ bloat.

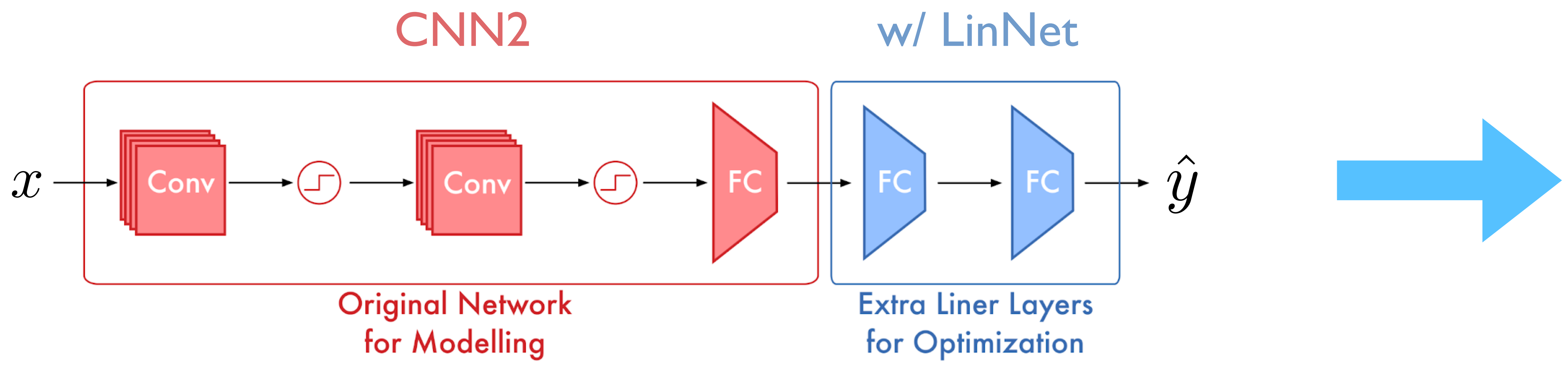
Meta-Optimizer for *faster* adaptation

- **How to bridge remaining 10%?**
  - Linear layers ➞ linear gradient transformation.
  - Non-linear layers ➞ cannot be collapsed ➞ bloat.



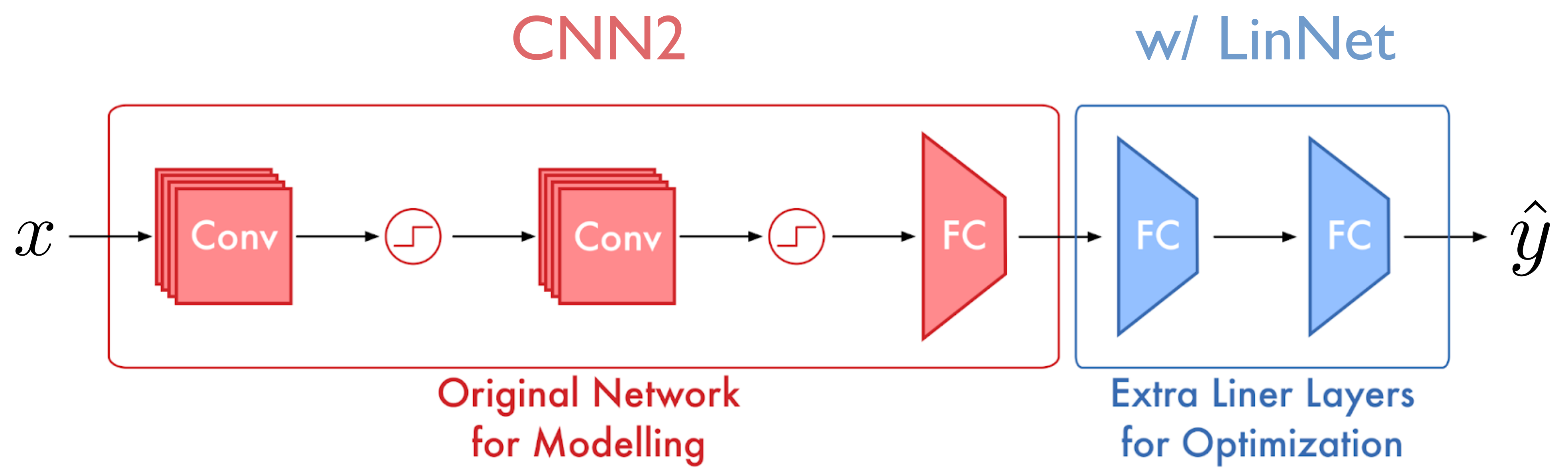
Meta-Optimizer for *faster* adaptation

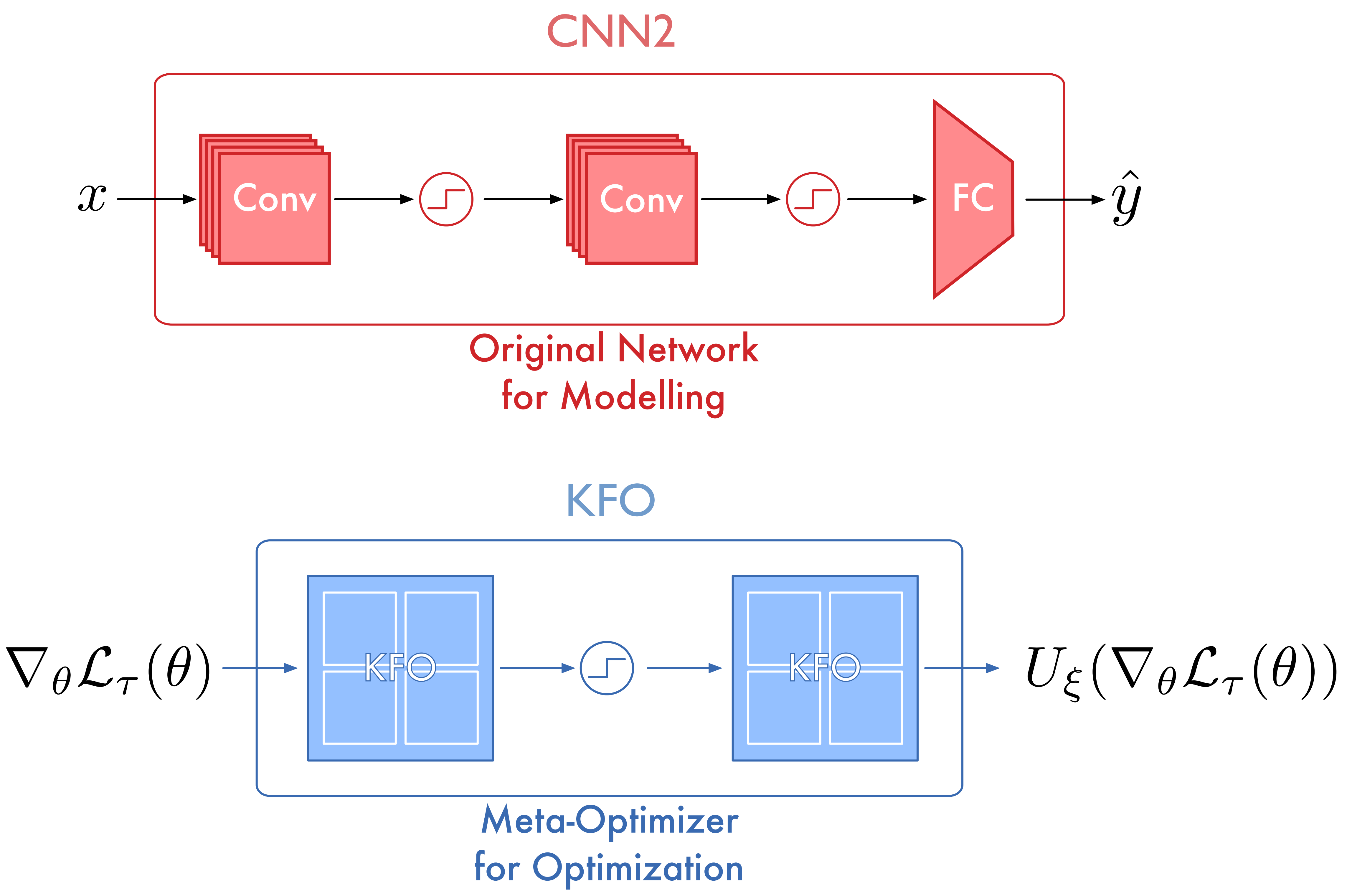
- How to bridge remaining 10%?
  - Linear layers ➞ linear gradient transformation.
  - Non-linear layers ➞ cannot be collapsed ➞ bloat.



Meta-Optimizer for *faster* adaptation

- How to bridge remaining 10%?
  - Linear layers ➞ linear gradient transformation.
  - Non-linear layers ➞ cannot be collapsed ➞ bloat.





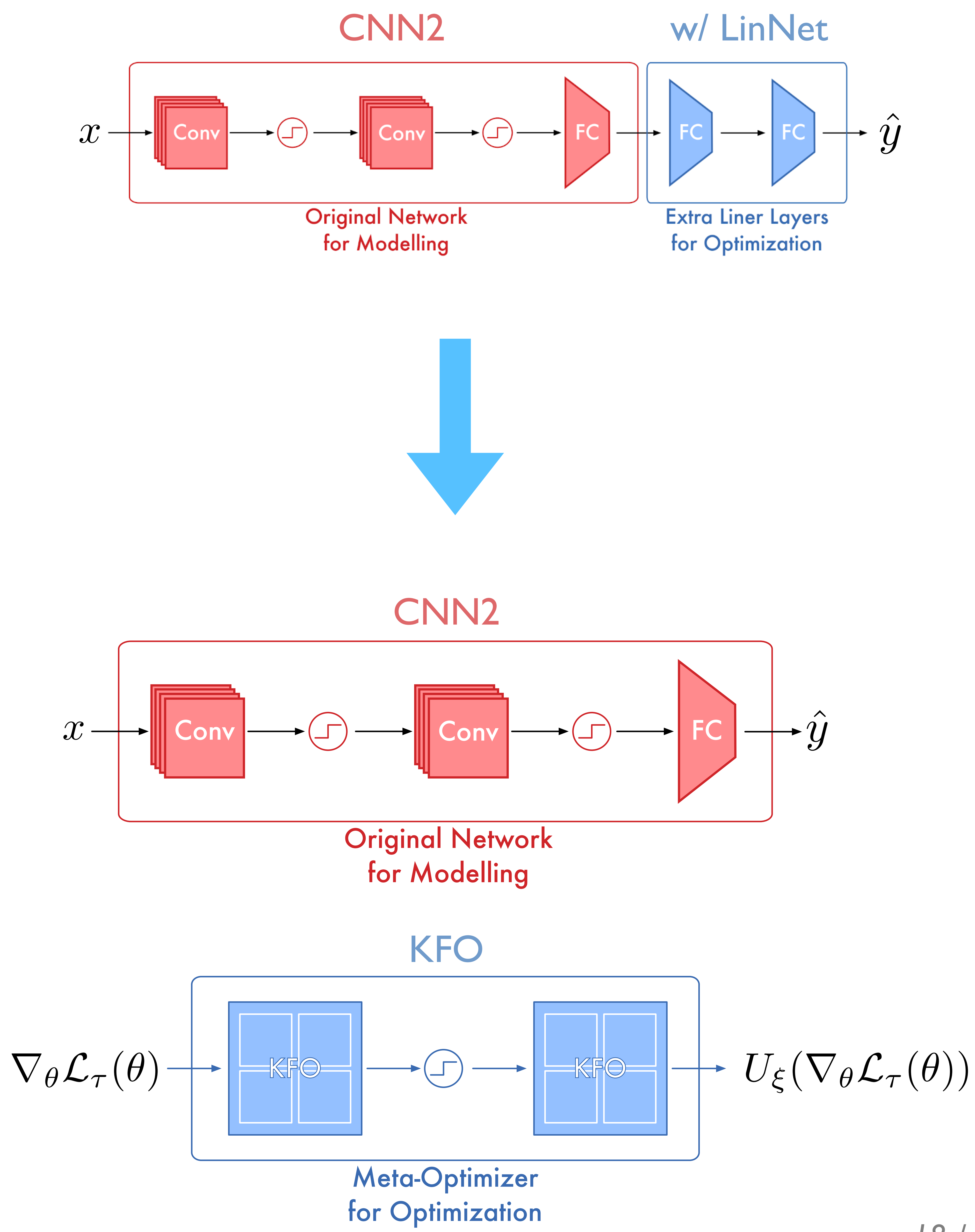
Meta-optimizer for *faster* adaptation

- How to bridge remaining 10%?
  - Linear layers ➞ linear gradient transformation.
  - Non-linear layers ➞ cannot be collapsed ➞ bloat.
- Meta-optimizers

$$\min_{\theta,\xi} \; \mathbb{E}_{\tau}[\mathcal{L}_{\tau}(\theta')]$$

$$\text{s.t.} \;\; \theta' = \theta - U_{\xi}(\nabla_{\theta}\mathcal{L}_{\tau}(\theta))$$

Neural network U_{ξ}
Parameterized by ξ



Meta-KFO further bridges the shallow-deep gap

	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet	MAML w/ MSGD	MAML w/ MC	MAML w/ T-Nets	MAML w/ KFO
CIFAR-FS	70.9%	62.2%	66.1%	62.8%	68.4%	66.4%	**69.6%**
mini-ImageNet	64.1%	52.6%	**60.5%**	59.9%	58.9%	58.5%	59.1%
Omniglot	98.5%	66.8%	88.1%	74.1%	94.6%	92.3%	**96.6%**

Meta-KFO further bridges the shallow-deep gap

Prior results

	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet	MAML w/ MSGD	MAML w/ MC	MAML w/ T-Nets	MAML w/ KFO
CIFAR-FS	70.9%	62.2%	66.1%	62.8%	68.4%	66.4%	**69.6%**
mini-ImageNet	64.1%	52.6%	**60.5%**	59.9%	58.9%	58.5%	59.1%
Omniglot	98.5%	66.8%	88.1%	74.1%	94.6%	92.3%	**96.6%**

Meta-KFO further bridges the shallow-deep gap

		Prior results		Baselines w/ optim. parameters			
	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet	MAML w/ MSGD	MAML w/ MC	MAML w/ T-Nets	MAML w/ KFO
CIFAR-FS	70.9%	62.2%	66.1%	62.8%	68.4%	66.4%	**69.6%**
mini-ImageNet	64.1%	52.6%	**60.5%**	59.9%	58.9%	58.5%	59.1%
Omniglot	98.5%	66.8%	88.1%	74.1%	94.6%	92.3%	**96.6%**

Meta-KFO further bridges the shallow-deep gap

		Prior results		Baselines w/ optim. parameters			KFO results
	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet	MAML w/ MSGD	MAML w/ MC	MAML w/ T-Nets	MAML w/ KFO
CIFAR-FS	70.9%	62.2%	66.1%	62.8%	68.4%	66.4%	**69.6%**
mini-ImageNet	64.1%	52.6%	**60.5%**	59.9%	58.9%	58.5%	59.1%
Omniglot	98.5%	66.8%	88.1%	74.1%	94.6%	92.3%	**96.6%**

Meta-KFO further bridges the shallow-deep gap

		Prior results		Baselines w/ optim. parameters			KFO results
	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet	MAML w/ MSGD	MAML w/ MC	MAML w/ T-Nets	MAML w/ KFO
CIFAR-FS	70.9%	62.2%	66.1%	62.8%	68.4%	66.4%	**69.6%**
mini-ImageNet	64.1%	52.6%	**60.5%**	59.9%	58.9%	58.5%	59.1%
Omniglot	98.5%	66.8%	88.1%	74.1%	94.6%	92.3%	**96.6%**

Recover 30% w/ KFO

Meta-KFO further bridges the shallow-deep gap

Prior results | Baselines w/ optim. parameters | KFO results

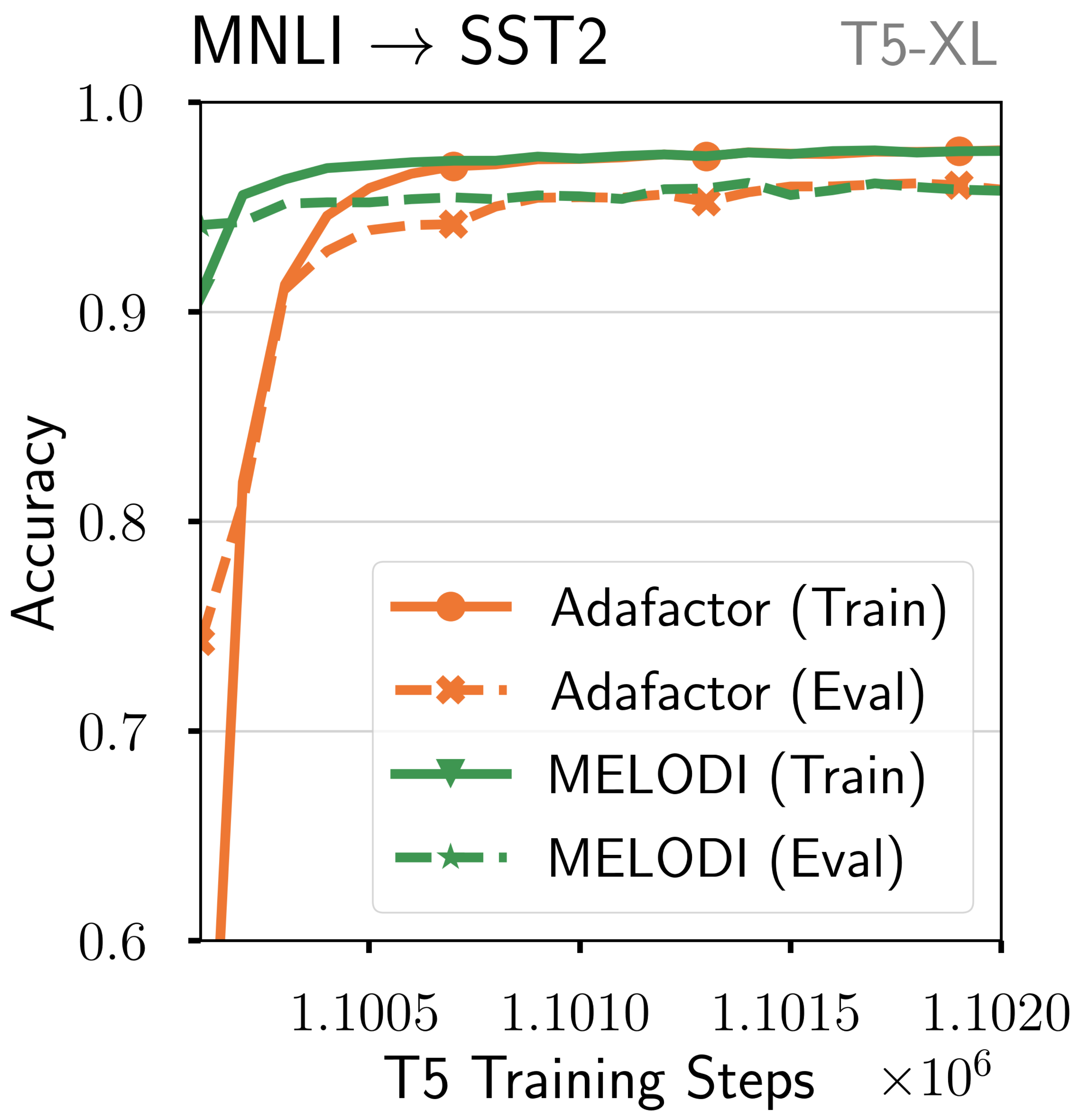
	MAML (CNN4)	MAML (CNN2)	MAML w/ LinNet	MAML w/ MSGD	MAML w/ MC	MAML w/ T-Nets	MAML w/ KFO
CIFAR-FS	70.9%	62.2%	66.1%	62.8%	68.4%	66.4%	**69.6%**
mini-ImageNet	64.1%	52.6%	**60.5%**	59.9%	58.9%	58.5%	59.1%
Omniglot	98.5%	66.8%	88.1%	74.1%	94.6%	92.3%	**96.6%**

Recover 30% w/ KFO

Only 2% degradation & no bloat!

Preview: scaling meta-optimizers to large language models

- How to scale-up meta-optimizers to modern LLMs?
- Insight: NLP is « text-in, text-out »
  - Language model: inductive bias for language generation.
  - Meta-optimizer: inductive bias for fast-adaptation.
- Setup
  - Collect prompt-tuning parameter trajectories.
  - Train-time: meta-optimizer learns to fast-forward trajectories.
  - Test-time: use meta-optimizer to finetune on unseen task.
- Results
  - MELODI is 8x faster than Adafactor on (unseen) SST2 tasks.



Work in progress. With: Ice Pasupat, Vincent Zhao, Kelvin Guu, and Fei Sha.

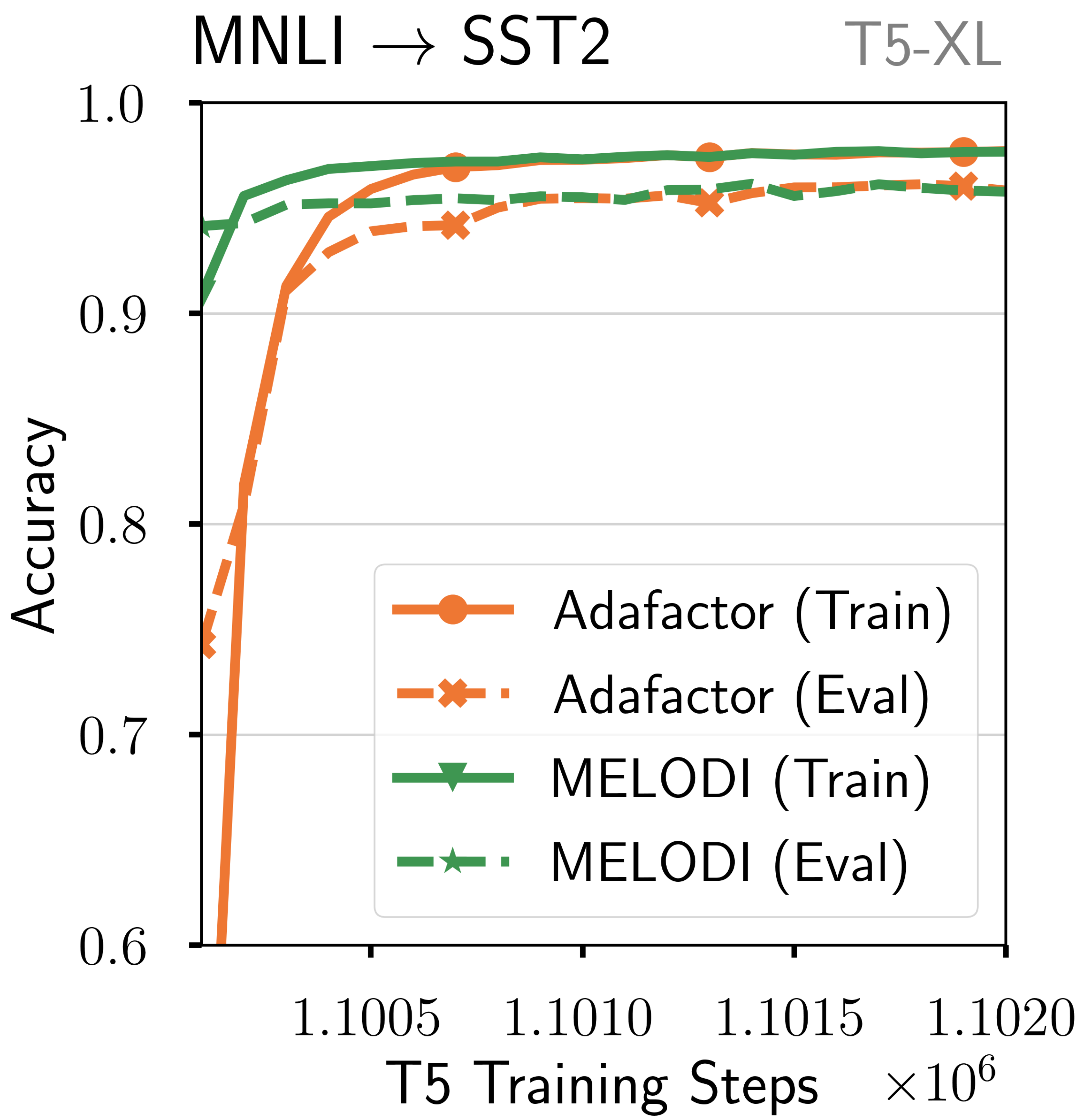
Preview: scaling meta-optimizers to large language models

- How to scale-up meta-optimizers to modern LLMs?

- Insight: NLP is « text-in, text-out »
  - Language model: inductive bias for language generation.
  - Meta-optimizer: inductive bias for fast-adaptation.

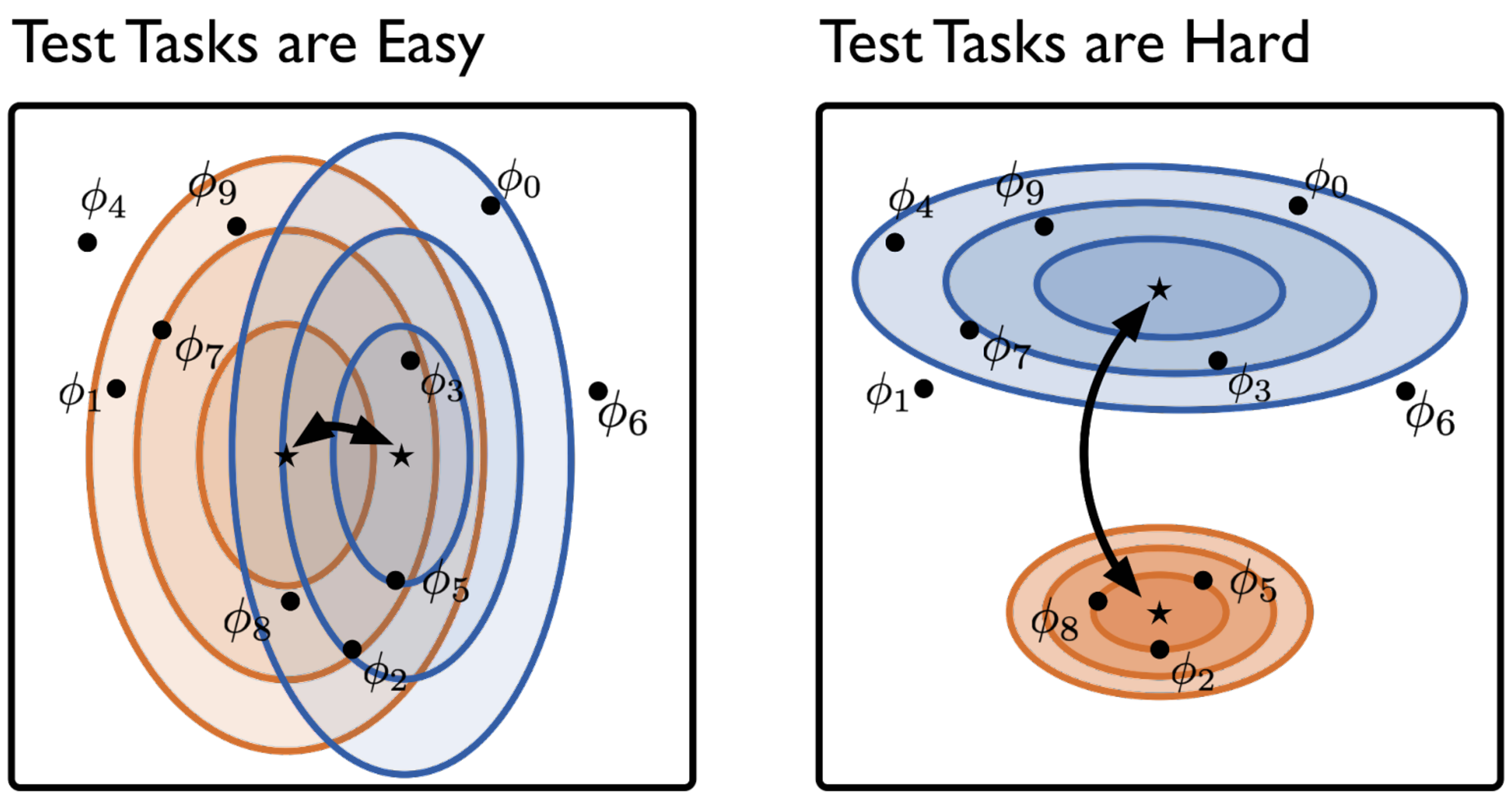
- Setup
  - Collect prompt-tuning parameter trajectories.
  - Train-time: meta-optimizer learns to fast-forward trajectories.
  - Test-time: use meta-optimizer to finetune on unseen task.

- Results
  - MELODI is 8x faster than Adafactor on (unseen) SST2 tasks.



Q2: When is adaptation required?

- **Motivation**
  - Recent papers suggests **we don't need adaptation**.
  - Much simpler recipe:
    - Pretrained representations.
    - Nearest-centroid classification.
- **Core question**
  - When do we need to adapt representations?

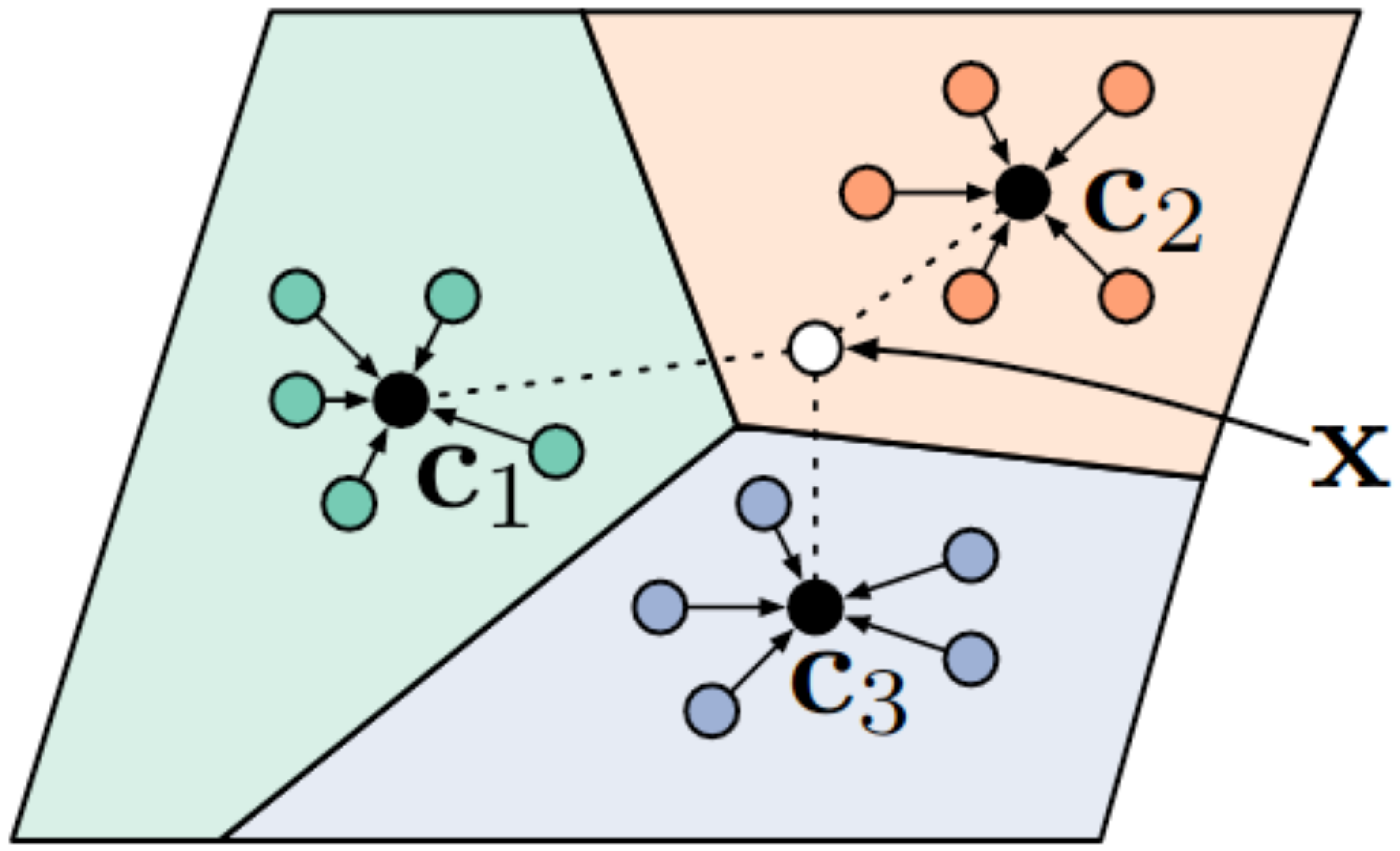


Prototypical Networks (ProtoNet)

- **Method**
  - « Nearest-centroid classification in learned embedding space. »
  - Pseudo-code:
    1. Embed all support sample x as $\phi(x)$.
    2. Compute mean embedding for each class.
    3. Classify query sample x' as nearest centroid.

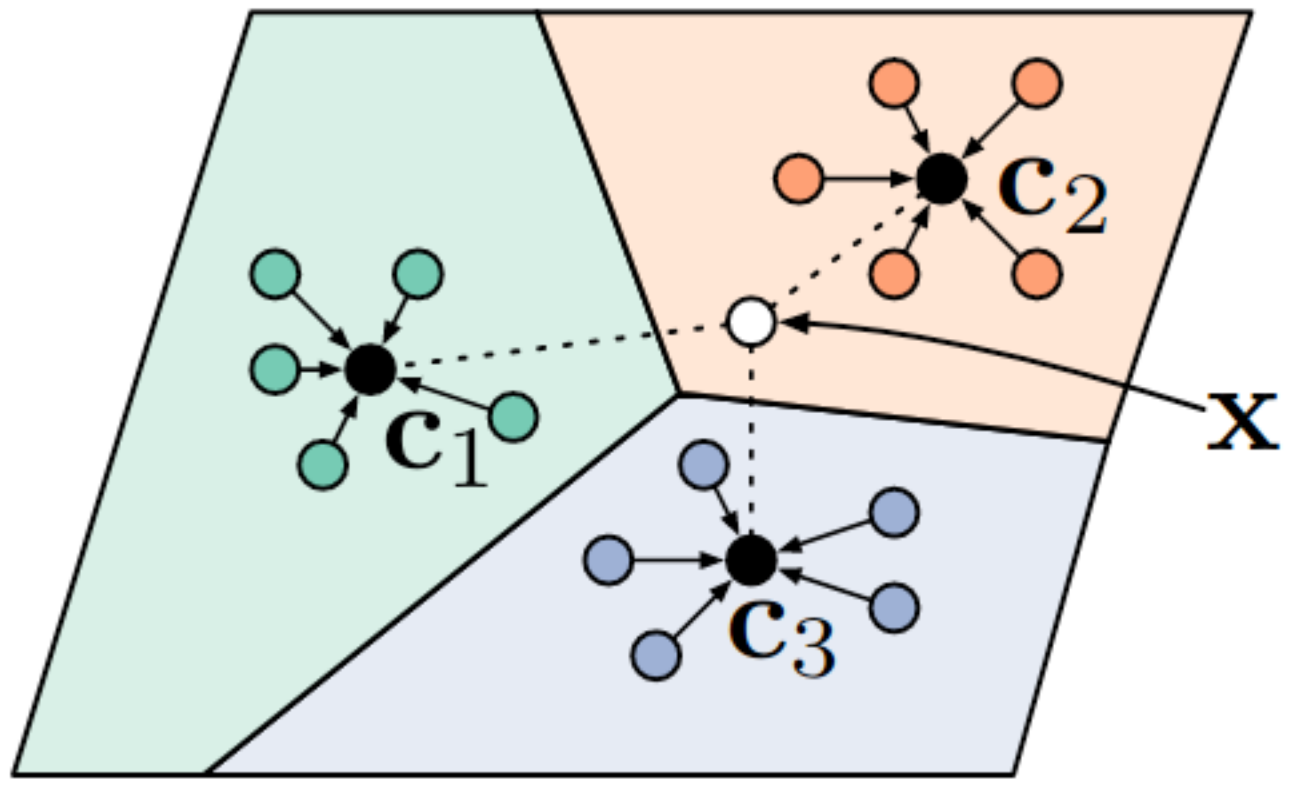
$$p(y = c_i \mid \mathbf{x} = x') = \mathrm{softmax}_{c_j}(\phi(x')^\top c_i))$$

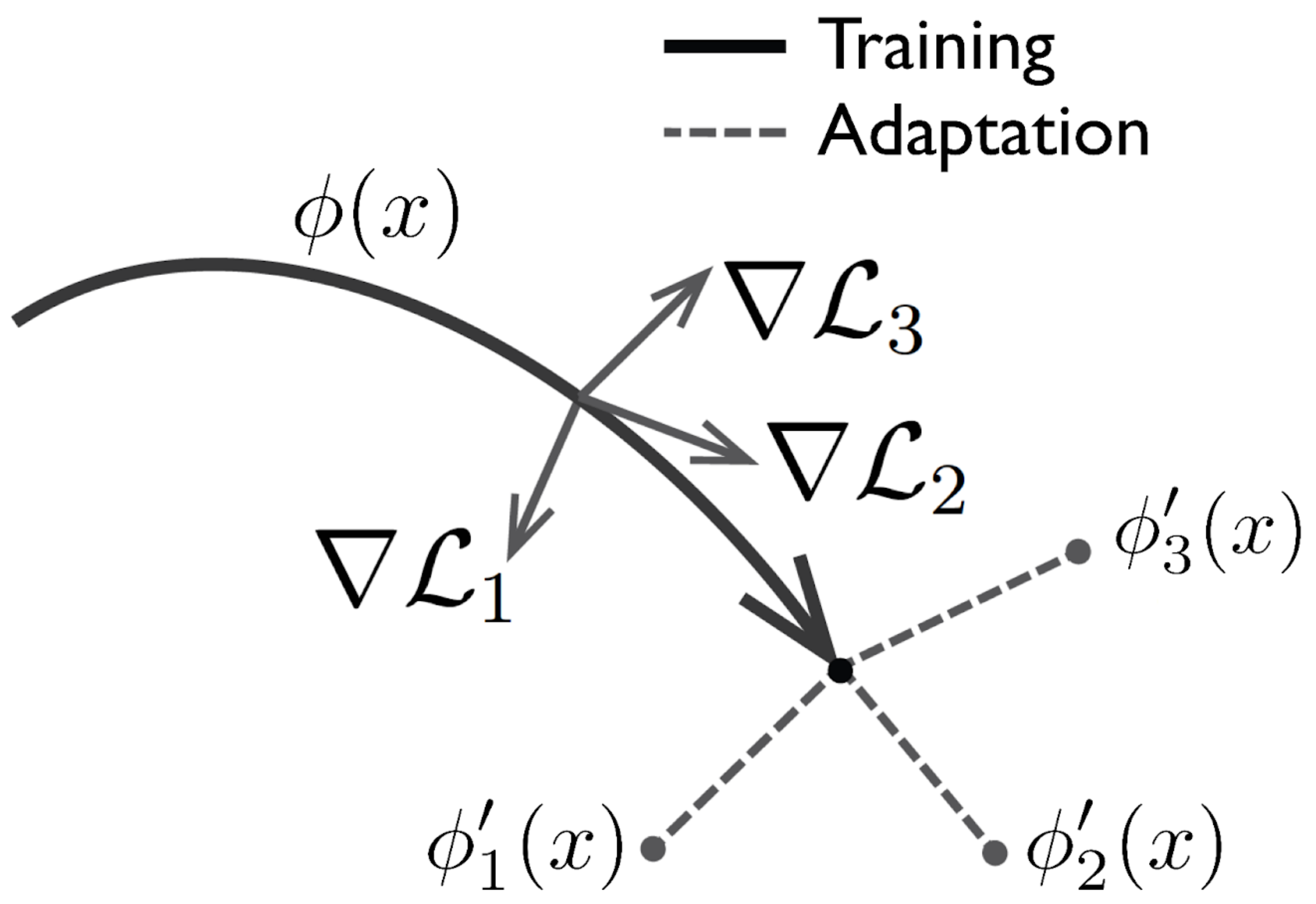
$$\text{where} \quad c_i = \frac{1}{k} \sum_{x \in c_i} \phi(x)$$



- **Extensions**
  - MetaOptNet, SimpleShot, DeepEMD, Proto-NCA, …

Contrasting ProtoNet and MAML





- Transfer Algorithms
  - Task-agnostic data representations.
  - Example: ProtoNet.
- Adaptation Algorithms
  - Task-specific data representations.
  - Example: MAML.

Transfer *v.s.* Adaptation

- Empirical study
  - Architecture: 4-layer CNN.
  - Dataset: CIFAR-FS & mini-ImageNet.
  - Metrics: accuracy & confidence intervals.
- Results
  - ProtoNet outperforms MAML.
  - 2-9% absolute improvement seems significant.
- Why is transfer so effective?

CIFAR-FS

	5-ways 1-shot	5-ways 5-shots
ProtoNet	**57.9%** ±0.8	**76.7%** ±0.6
MAML	53.8% ±1.8	67.6% ±1.0

mini-ImageNet

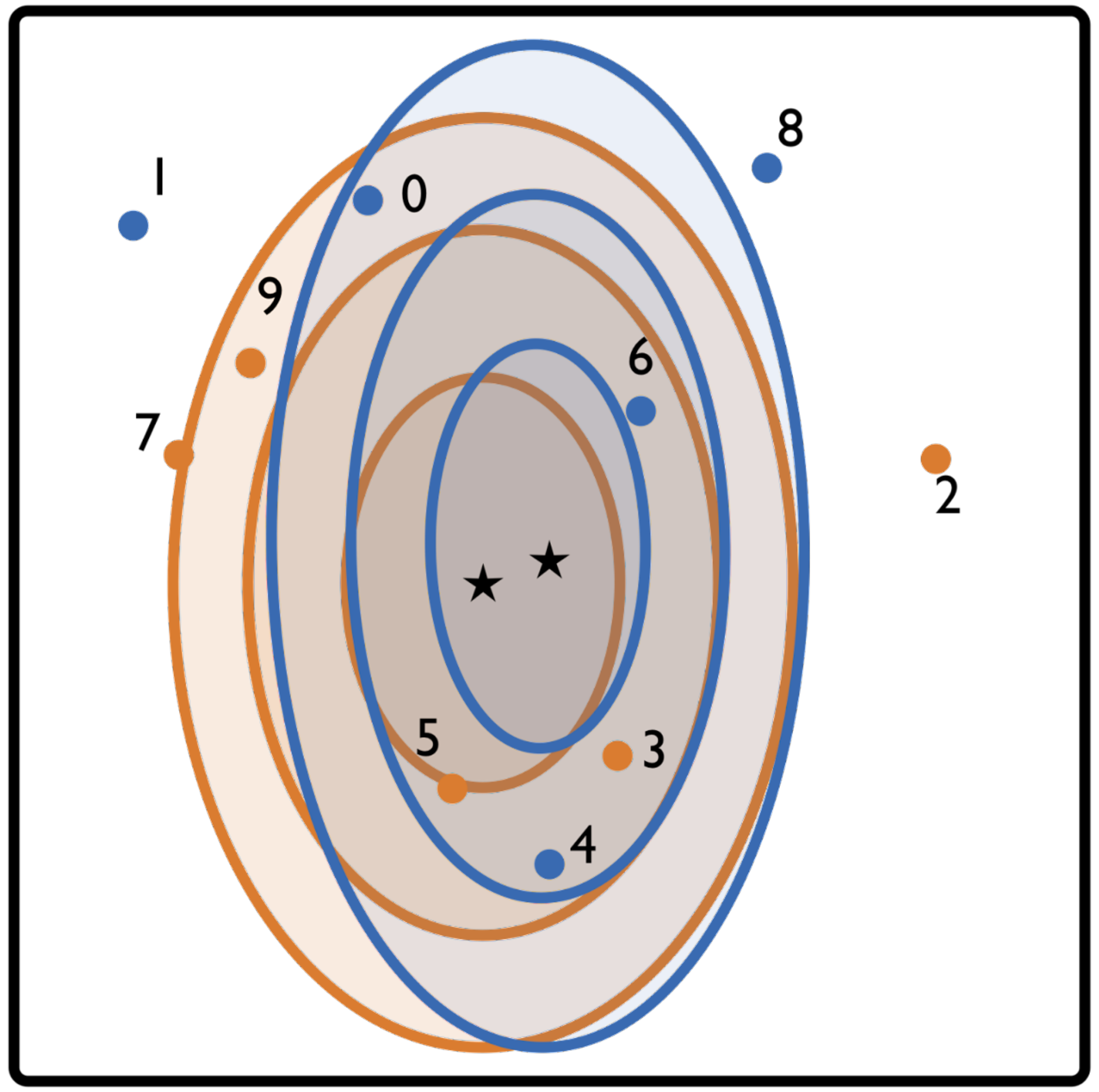
	5-ways 1-shot	5-ways 5-shots
ProtoNet	**42.9%** ±0.6	**65.9%** ±0.6
MAML	40.9% ±1.5	58.9% ±0.9

Results from *Bertinetto et al., « Meta-Learning with Differentiable Closed-Form Solvers », ICLR 2019*

A few caveats in the comparisons

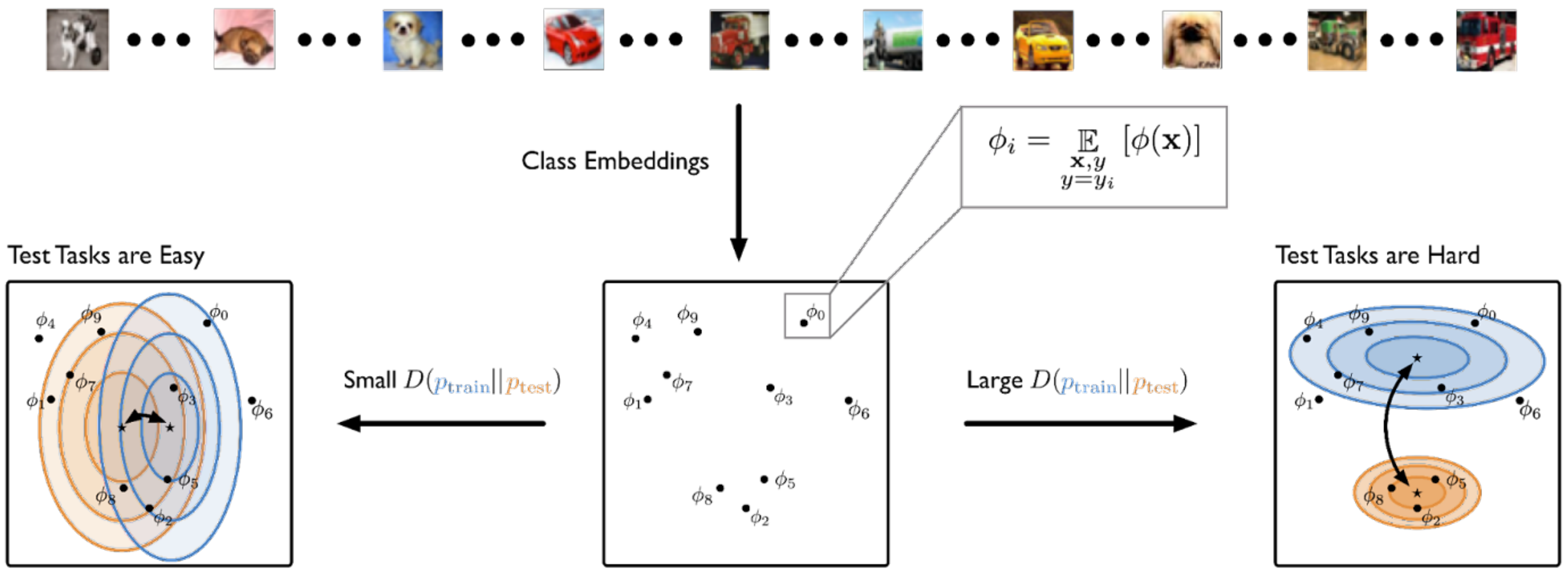
- **Where did our train and test tasks come from?**
  - Classes randomly assigned to train or test splits.
- **Underlying assumptions?**
  - No distribution shift between train and test classes.
  - Train and test classes semantically closely related.
- **More thorough studies with alternative splits**
  - Can we use semantic information?
    - Expensive → What if no semantics?
  - Can we train on dataset *X* and test on *Y*?
    - Is transfer easy or hard? How to tell?

Class Distributions



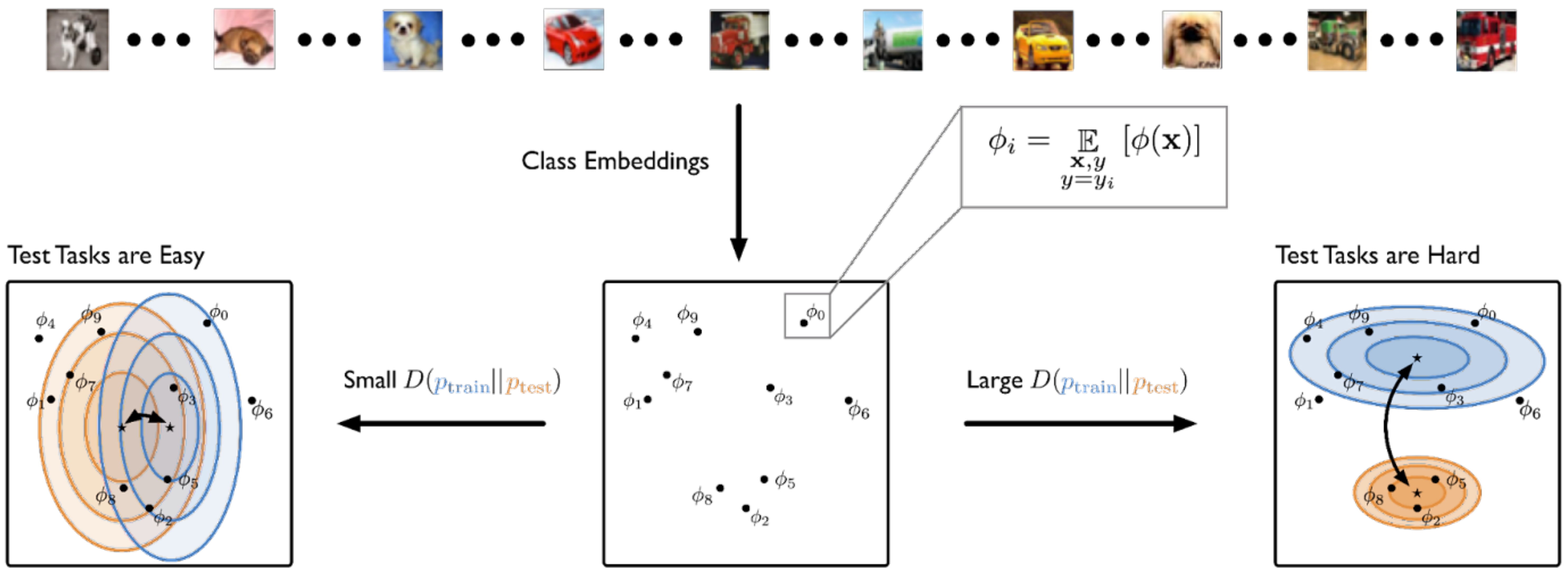
Train Classes
Test Classes

Automatic Taskset Generation (ATG)



- Overview
  - Embed classes with trained feature extractor.
  - Cluster, subject to **divergence constraint**.
  - Assign according to clusters' likelihood.

Automatic Taskset Generation (ATG)

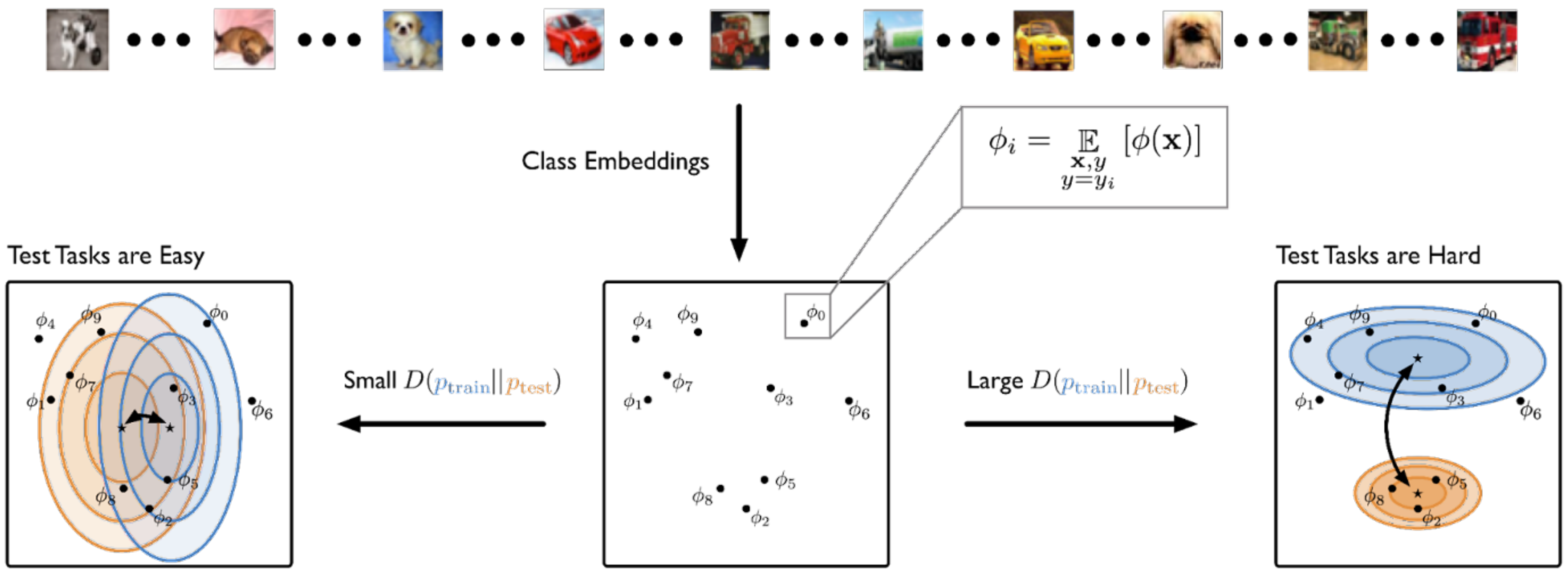


- Overview
  - Embed classes with trained feature extractor.
  - Cluster, subject to divergence constraint.
  - Assign according to clusters' likelihood.

$$\max_{p_{\text{train}}, p_{\text{test}}} \sum_i \log(p_{\text{train}}(c_i) + p_{\text{test}}(c_i))$$

$$\text{s.t.} \quad D(p_{\text{train}}||p_{\text{test}}) = R$$

Automatic Taskset Generation (ATG)



- **Overview**
  - Embed classes with trained feature extractor.
  - Cluster, subject to **divergence constraint**.
  - Assign according to clusters' likelihood.

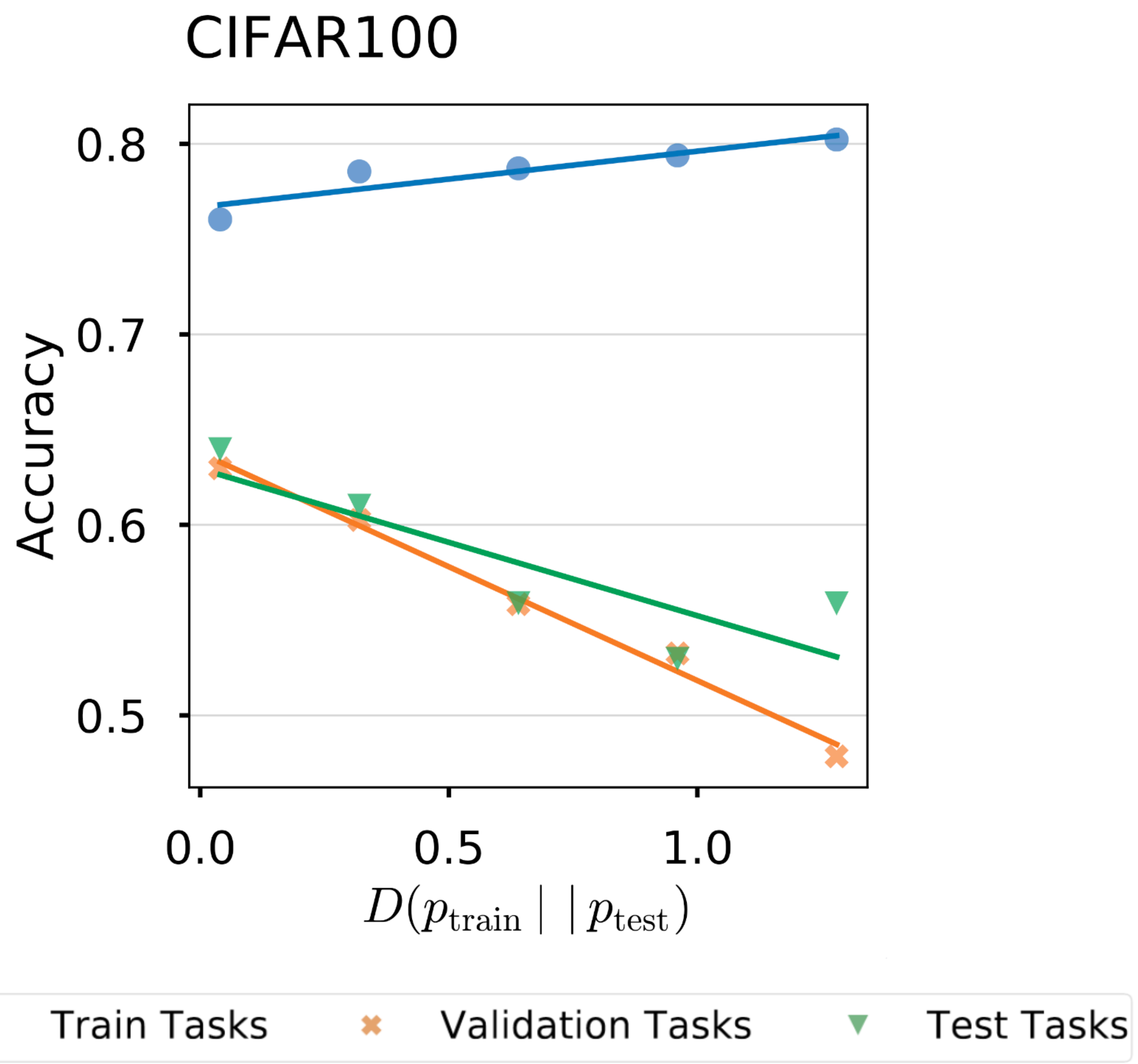
$$\max_{p_{\text{train}}, p_{\text{test}}} \sum_i \log(p_{\text{train}}(c_i) + p_{\text{test}}(c_i))$$
$$\text{s.t.} \quad D(p_{\text{train}}||p_{\text{test}}) = R$$

- **Desiderata**
  - Automatic ✓
  - No human knowledge ✓
  - Control over train-test task difficulty ✓

Making train and test tasks more different

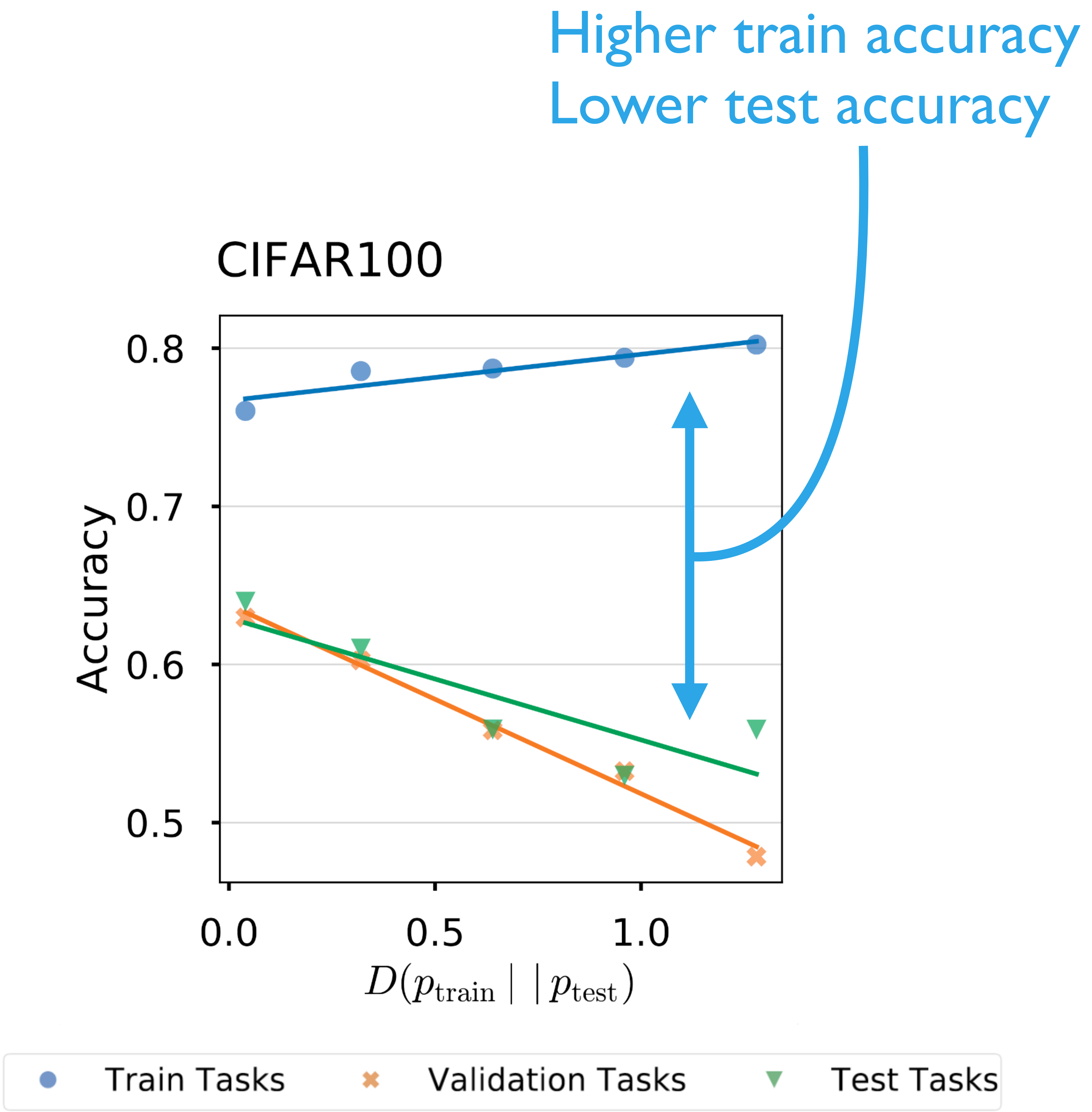
- **Setup**
  - Generate tasksets for various R.
  - Measure classification accuracy.
    - Re-trained feature extractor $\phi(x)$.
    - ProtoNet (nearest centroid).

- **Results**
  - Train-to-test **task difficulty increases**.
  - True **across datasets**.
    - Incl. EMNIST & Labelled Faces in the Wild (LFW10) **with no semantics**!



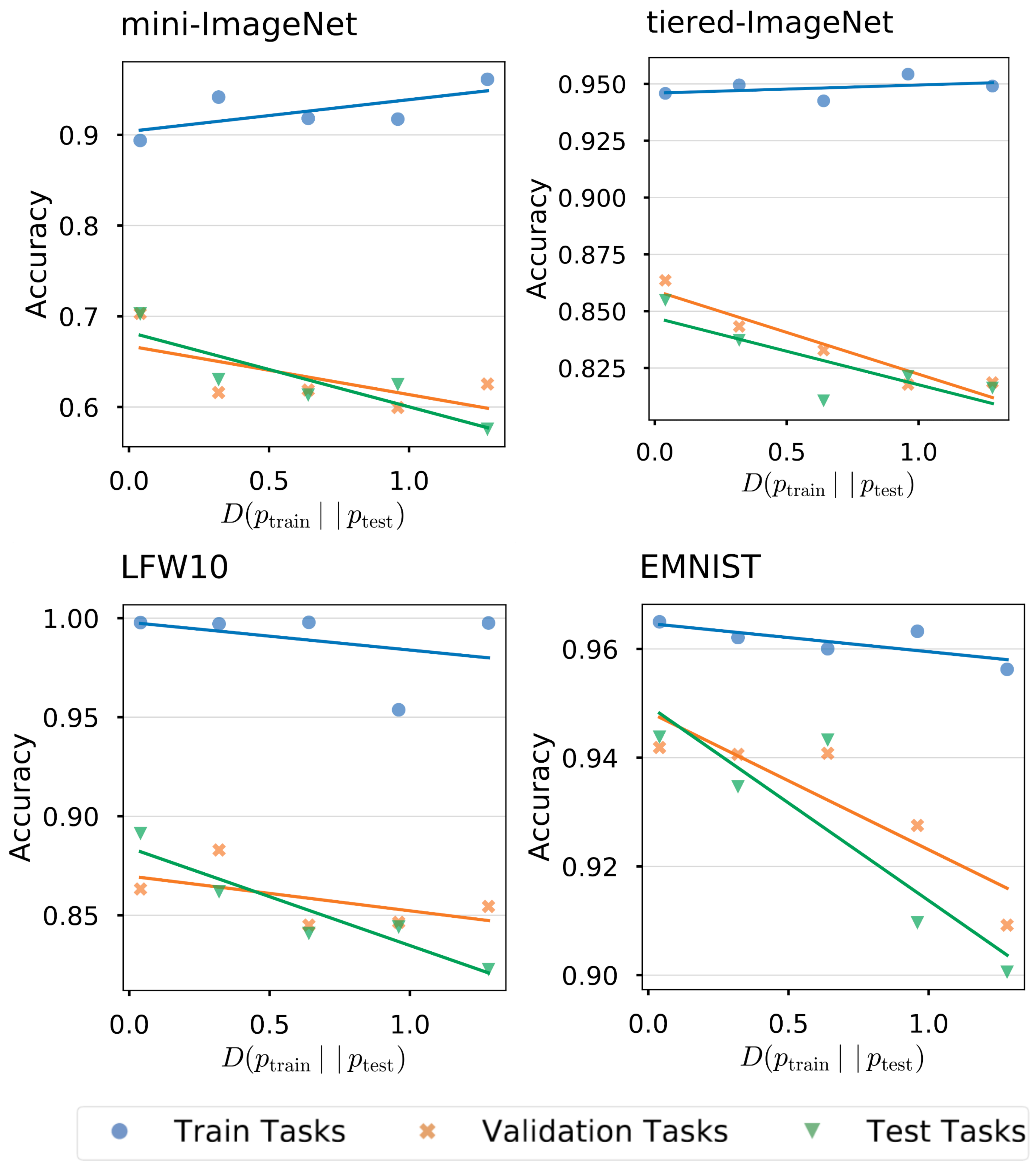
Making train and test tasks more different

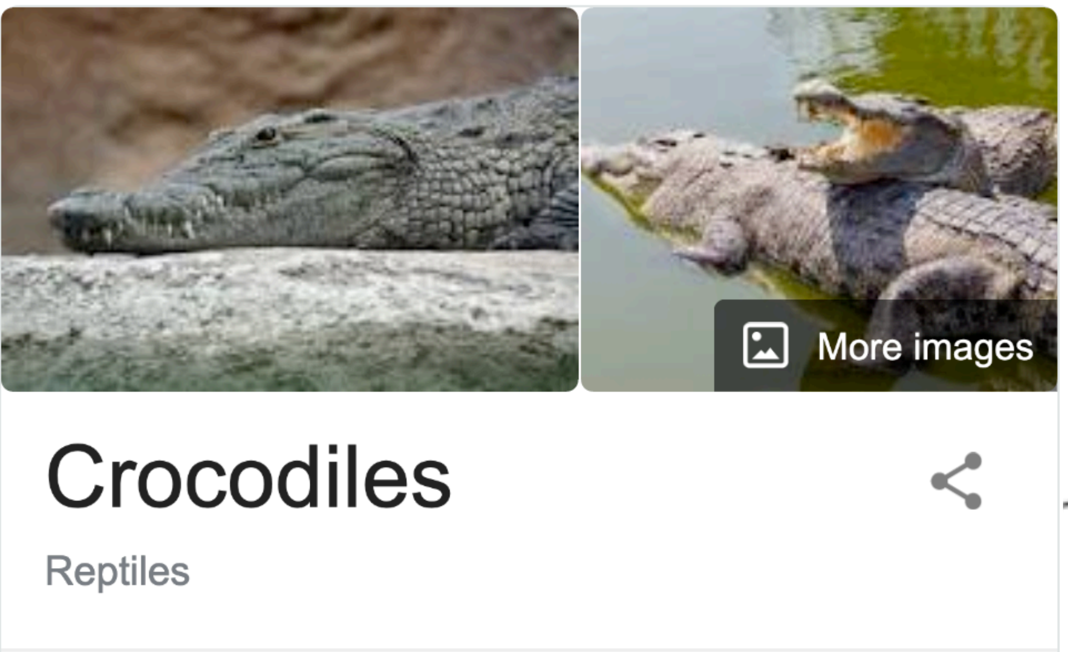
- **Setup**
  - Generate tasksets for various R.
  - Measure classification accuracy.
    - Re-trained feature extractor $\phi(x)$.
    - ProtoNet (nearest centroid).
- **Results**
  - Train-to-test **task difficulty increases**.
  - True **across datasets**.
    - Incl. EMNIST & Labelled Faces in the Wild (LFW10) **with no semantics**!

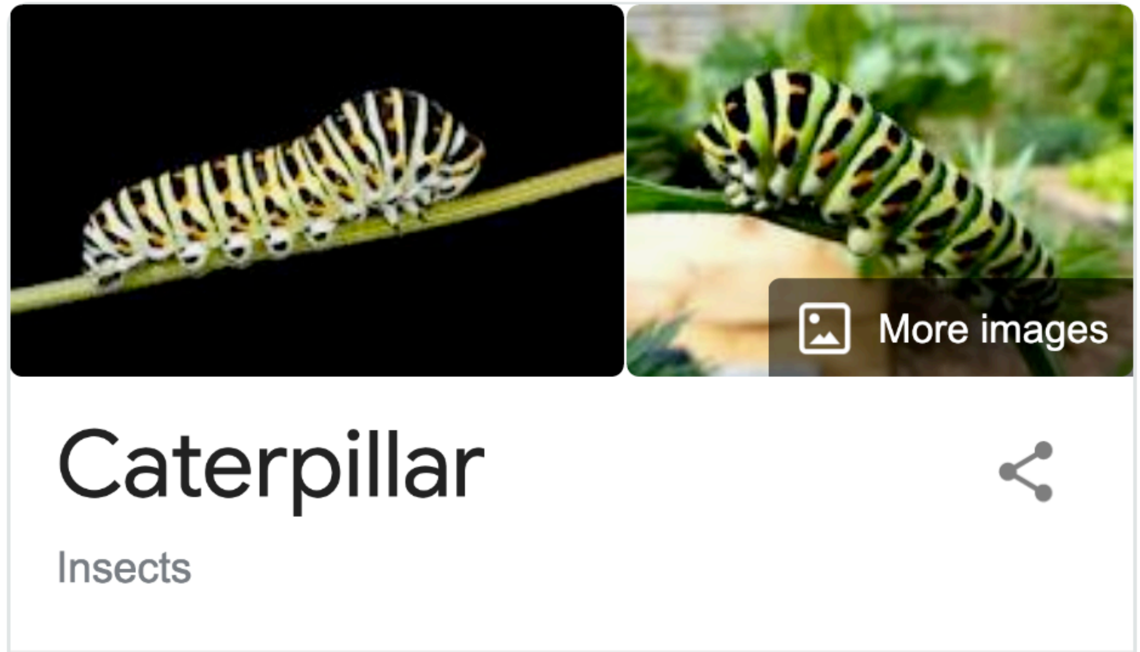


Making train and test tasks more different

- **Setup**
  - Generate tasksets for various R.
  - Measure classification accuracy.
    - Re-trained feature extractor $\phi(x)$.
    - ProtoNet (nearest centroid).
- **Results**
  - Train-to-test **task difficulty increases**.
  - True **across datasets**.
    - Incl. EMNIST & Labelled Faces in the Wild (LFW10) **with no semantics**!




More images
Crocodiles
Reptiles


More images
Caterpillar
Insects


forest
oak_tree
palm_tree
orchid
tiger
willow_tree
spider
poppy
rose
maple_tree
pine_tree
cockroach
dinosaur
aquarium_fish
lobster
ray
trout
bee
butterfly
crab
snake
turtle
caterpillar
crocodile
beetle
lizard


More images
Crocodiles
Reptiles
More images
Caterpillar
Insects
forest
oak_tree
palm_tree
orchid
tiger
willow_tree
spider
poppy
rose
maple_tree
pine_tree
cockroach
dinosaur
aquarium_fish
lobster
ray
trout
bee
butterfly
crab
snake
turtle
caterpillar
crocodile
beetle
lizard

Transfer *v.s.* Adaptation — Redux

- Empirical study
  - ProtoNet *v.s.* MAML, again.
  - Hardest ATG splits for CIFAR-FS and mini-ImageNet.
    - $D(p_{\text{train}} \,||\, p_{\text{test}}) = 0.96$
- Results
  - MAML outperforms ProtoNet by 2-5%.
  - Opposite outcome of original tasksets.
- Intuition
  - Train = test: transfer is enough.
  - Train ≠ test: adaptation is required.

CIFAR-FS – Hard

	5-ways 1-shot	5-ways 5-shots
ProtoNet	35.6% ±0.2	50.5% ±0.3
MAML	**35.9%** ±0.3	**55.7%** ±0.5

mini-ImageNet – Hard

	5-ways 1-shot	5-ways 5-shots
ProtoNet	41.8% ±0.6	60.4% ±0.4
MAML	**44.9%** ±0.8	**62.3%** ±1.0

Take-aways — Part 1

Q1: How to adapt fast?
Freeze task-agnostic parameters; learn optimization parameters.

Q2: When is adaptation required?
When the new tasks are different from train tasks.

Fast Adaptation without Meta-Learning

Part II

New!

Fast Adaptation without Meta-Learning

Part II

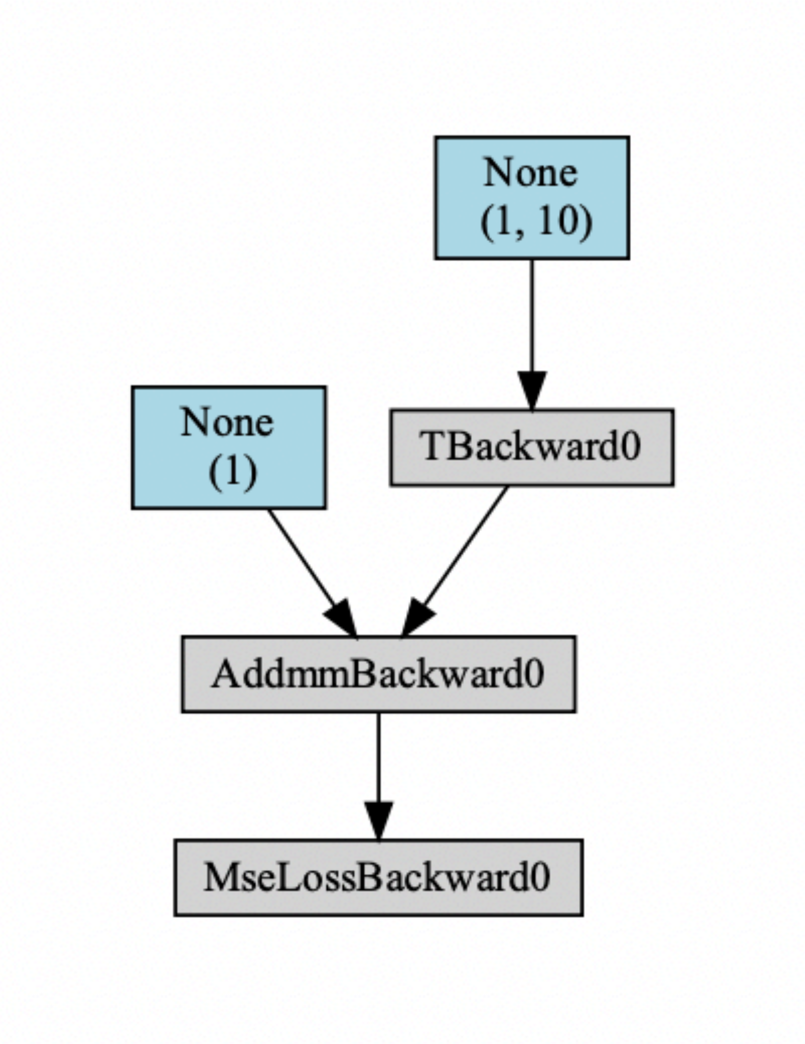
Q3: How to adapt quickly with reinforcement learning?

- Downsides of MAML
  - Expensive pretraining: memory **and** compute.
  - Incompatible with pretrained models.

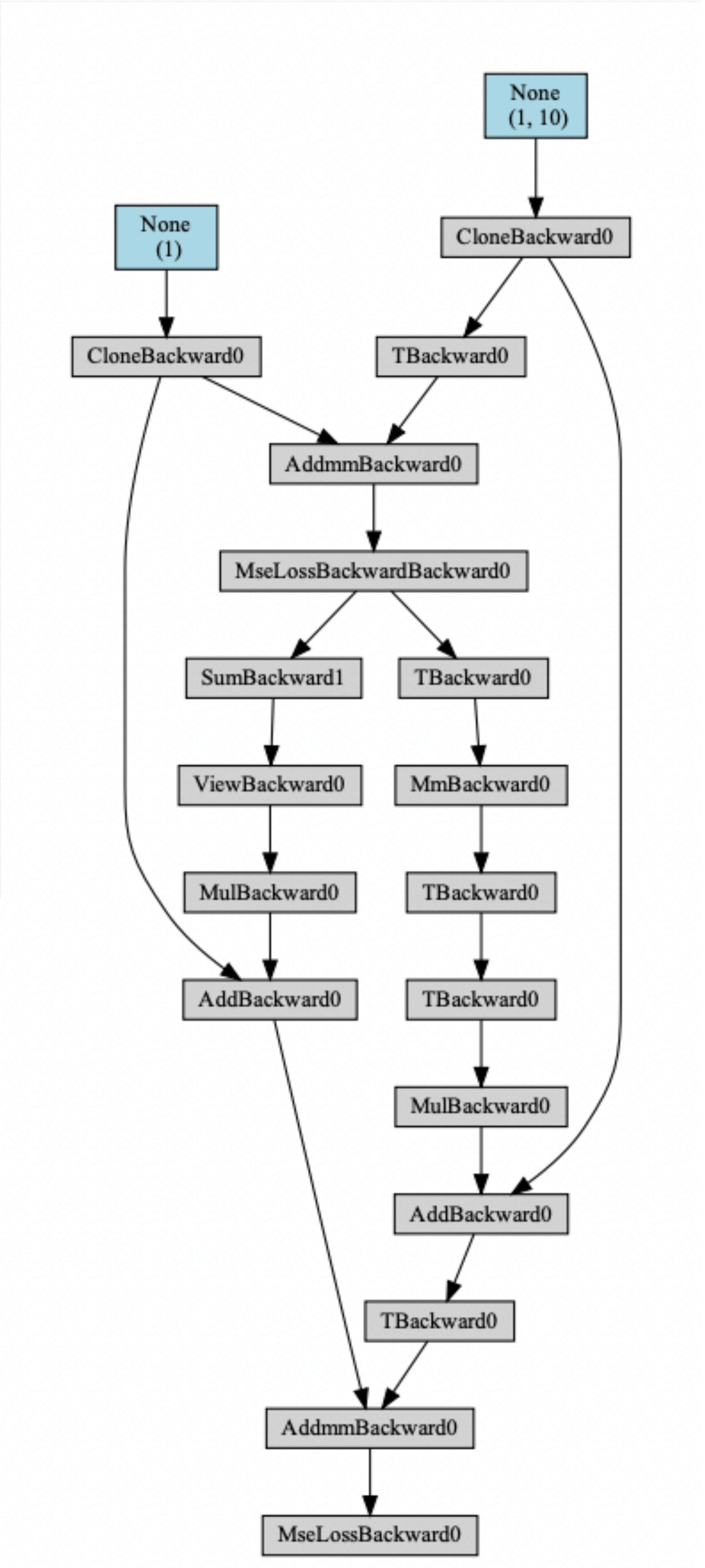
- Ideal scenario
  - Download off-the-shelf pretrained model.
  - Quickly solve new tasks **as if trained with MAML**.

- Core question
  - How to quickly adapt **pretrained representations**?

MSE
(Linear Model)



MAML MSE
(Linear Model)

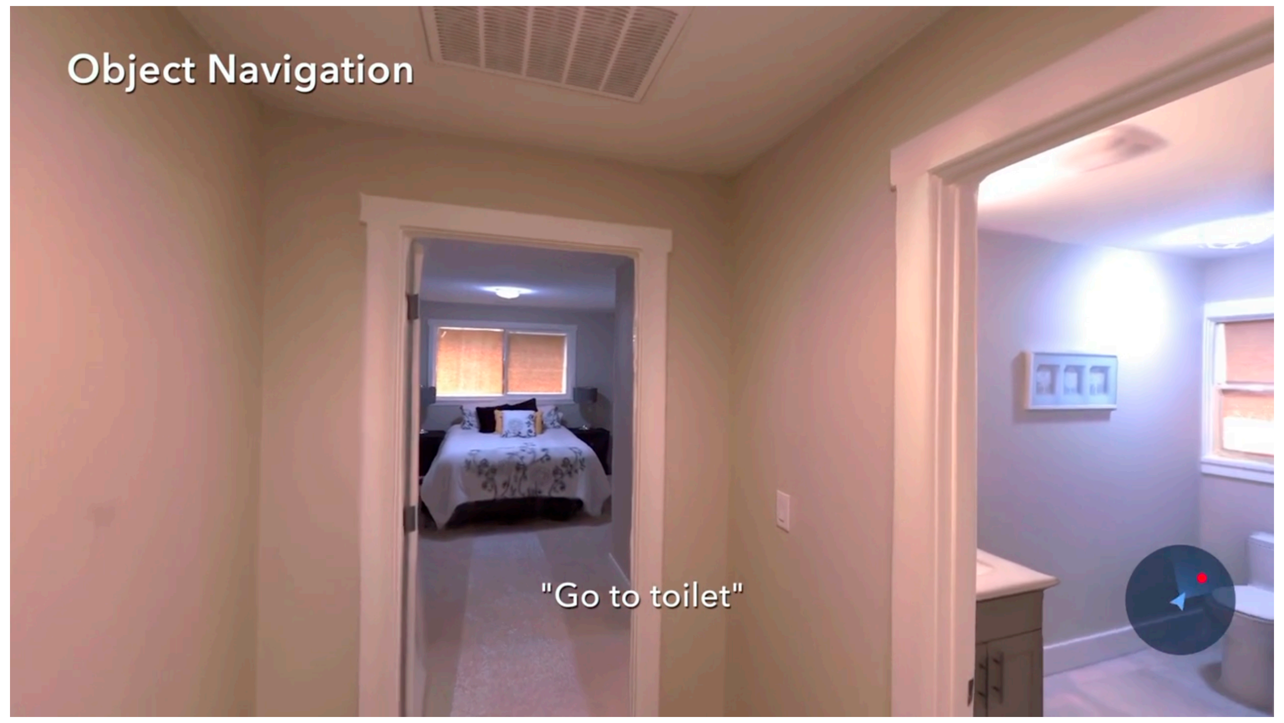


Reference: *Arnold & Sha, « Policy-induced self-supervision improves representation finetuning in visual RL », In submission*

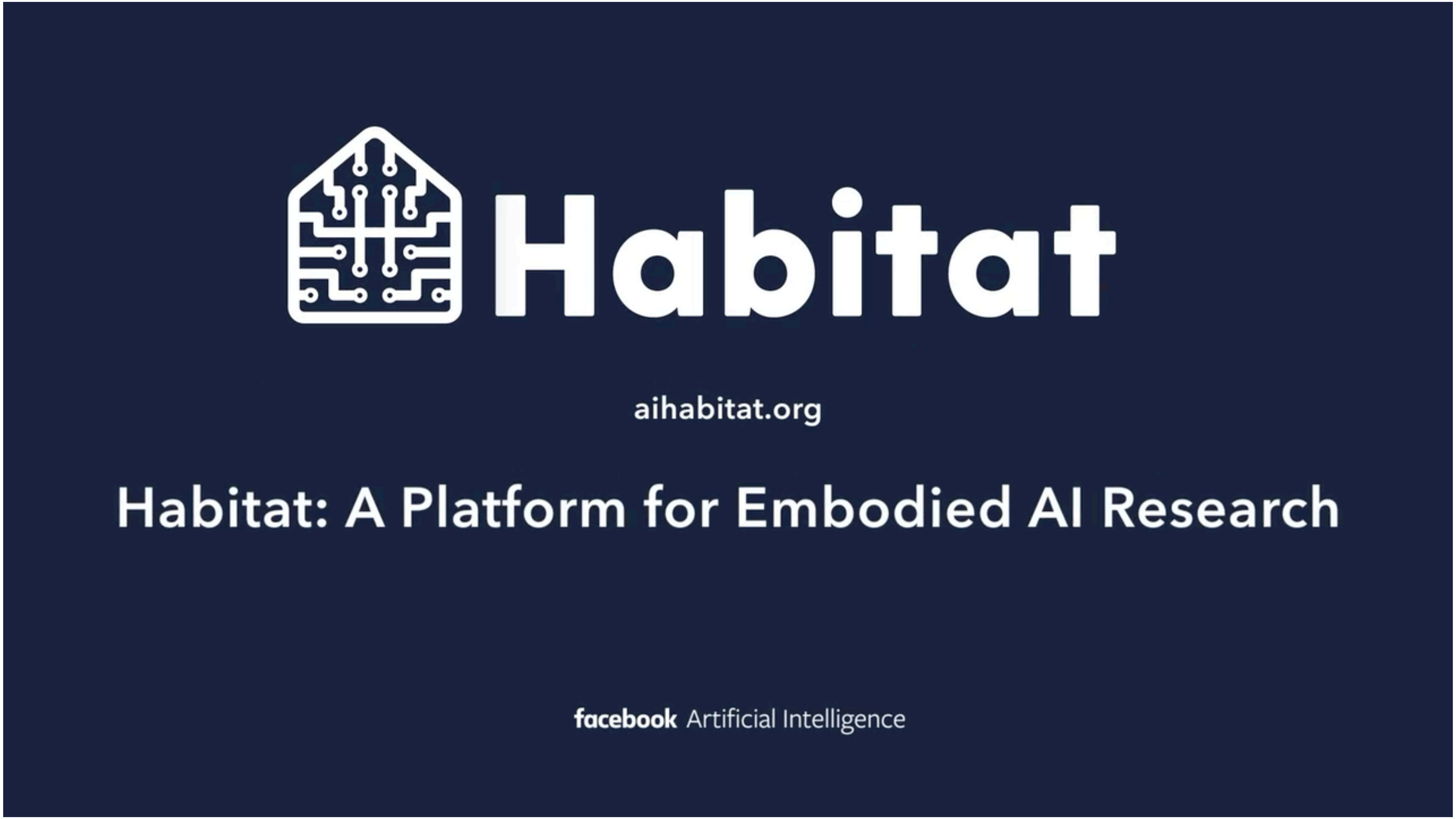
Case study: visual reinforcement learning

- Visual RL
  - Given visual observations, take actions that maximize rewards.
  - Challenge: noisy learning signal.
- Testbed 1: Habitat AI
  - Pretraining: classify ImageNet images.
  - Downstream: robot navigates from camera and GPS observations.

Finetune



Szot et al., « Habitat 2.0: Training Home Assistants to Rearrange their Habitat », NeurIPS 2021


Habitat
aihabitat.org
Habitat: A Platform for Embodied AI Research
facebook Artificial Intelligence

Habitat

aihabitat.org

Habitat: A Platform for Embodied AI Research

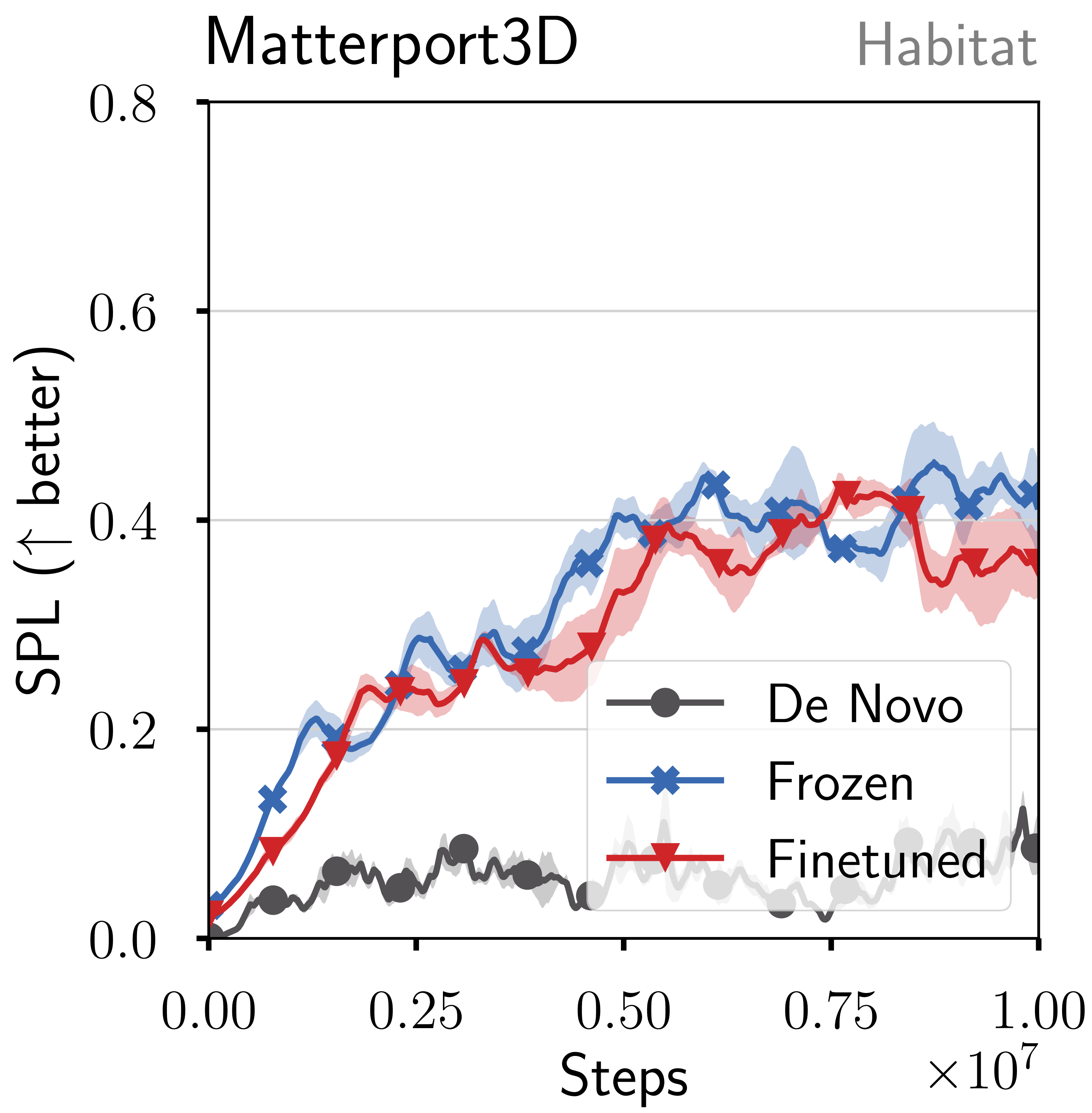
facebook Artificial Intelligence

Running simple baselines

- Downstream learning setup
  - Policy and value heads learned on top of features.
  - Features:

	De Novo	Frozen	Finetuned
Initialization	Random	Pretrained	Pretrained
Finetuned	✓	✗	✓

- Results
  - Finetuned struggles to outperform Frozen.

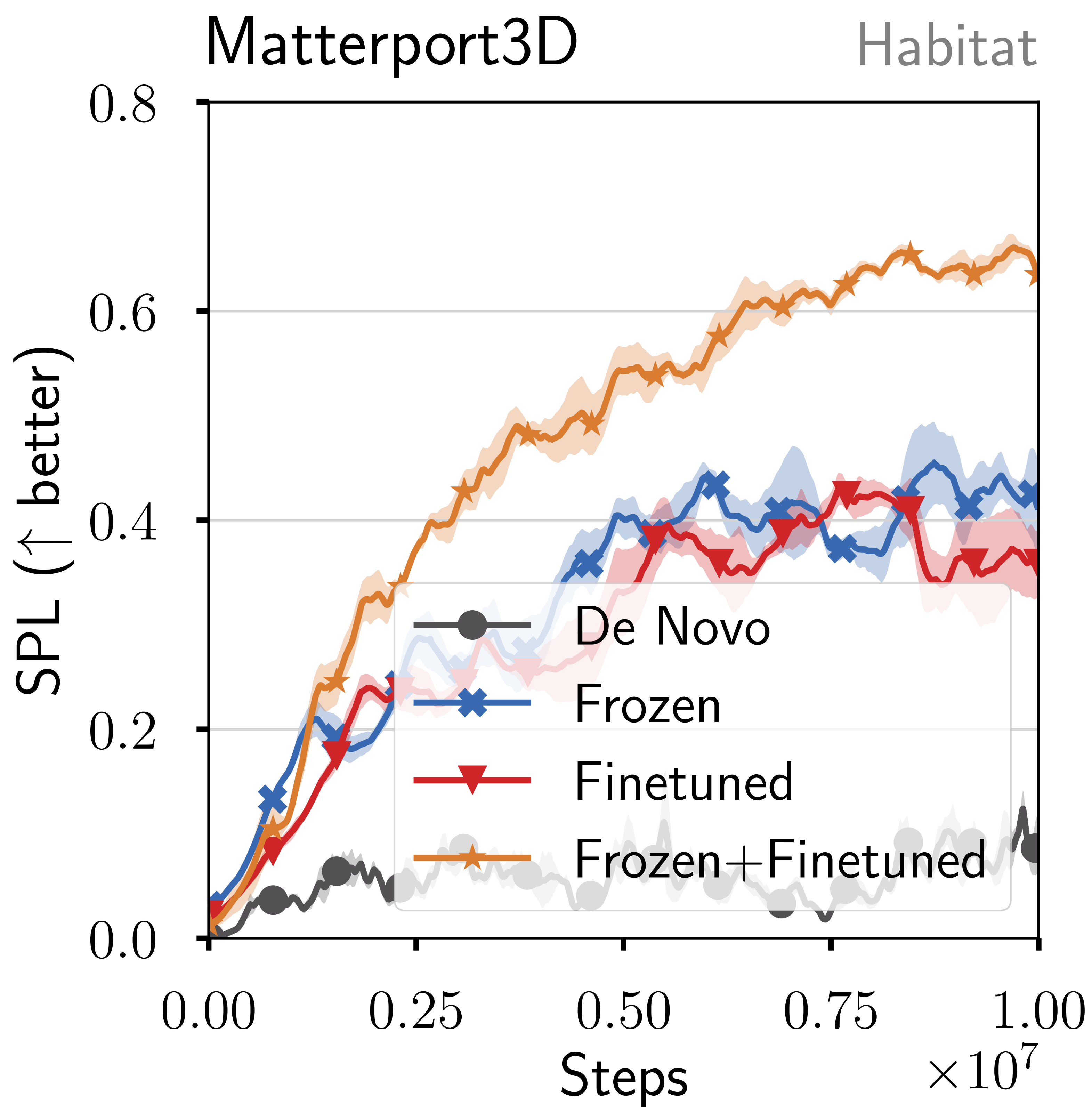


A page from the MAML textbook

- **Can we identify and freeze task-agnostic layers?**
  - Yes, with a little expert data.
  - Idea: measure how well each layer can predict Q-values.

- **Frozen+Finetuned**
  - Initialization: **pretrained**
  - Task-agnostic layers: **frozen**
  - Task-specific layers: **finetuned**

- **Results**
  - Frozen+Finetuned significantly stabilizes finetuning.



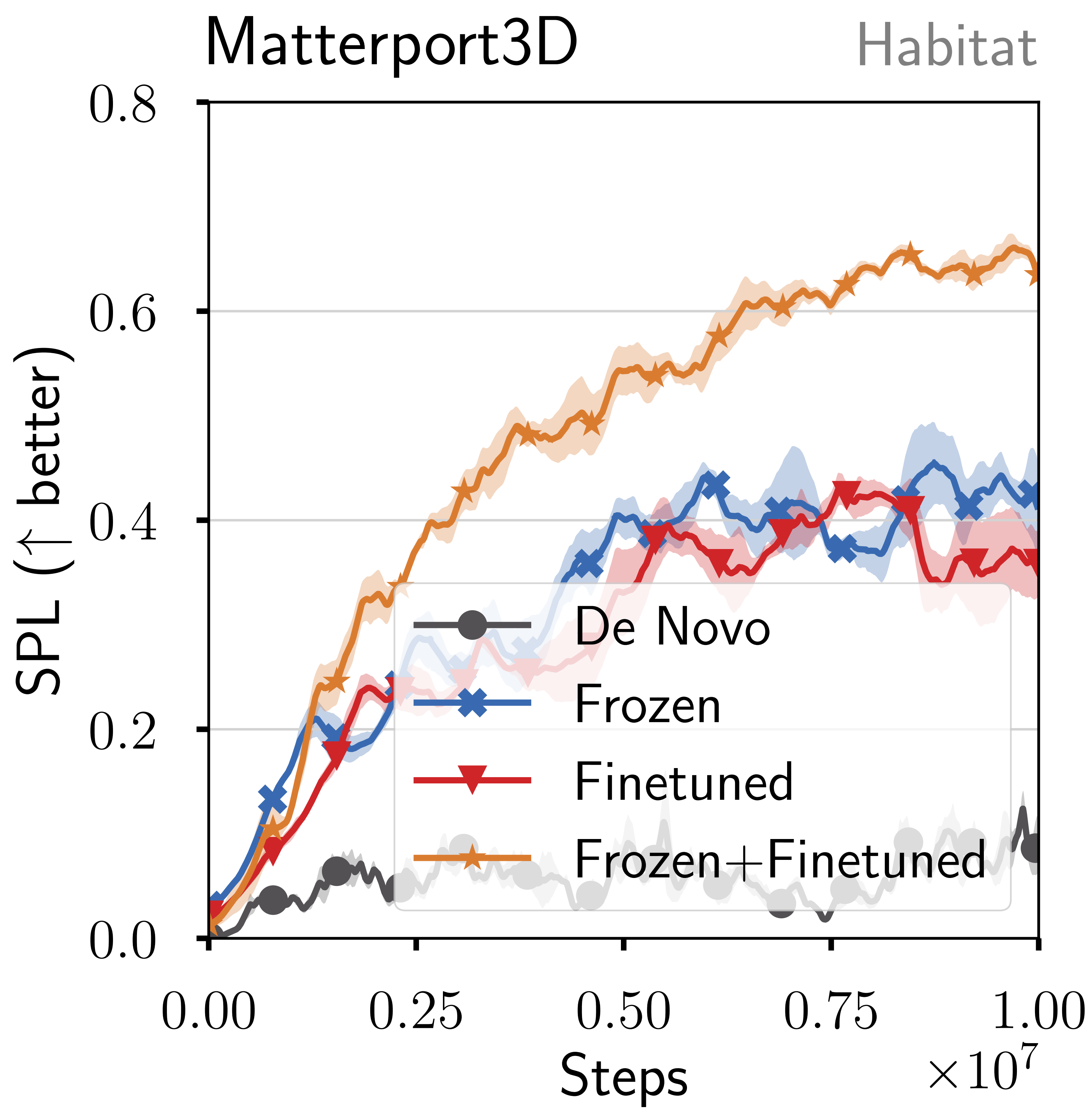
A page from the MAML textbook

- **Can we identify and freeze task-agnostic layers?**
  - Yes, with a little expert data.
  - Idea: measure how well each layer can predict Q-values.

- **Frozen+Finetuned**
  - Initialization: **pretrained**
  - Task-agnostic layers: **frozen**
  - Task-specific layers: **finetuned**

- **Results**
  - Frozen+Finetuned significantly stabilizes finetuning.

- **Can we improve further?**



Fast-adaptation inductive biases for RL with MSR Jump

- **Testbed 2: MSR Jump**
  - Pretraining: white agent **jumps over gray** box (75 pretraining tasks).
  - Downstream: jump over **unseen** box positions (75 evaluation tasks).

Task 1	Task 2	Task 3	Task 4	Task 5
Floor: 15	Floor: 15	Floor: 15	Floor: 15	Floor: 15
Position: 20	Position: 30	Position: 40	Position: 50	Position: 60

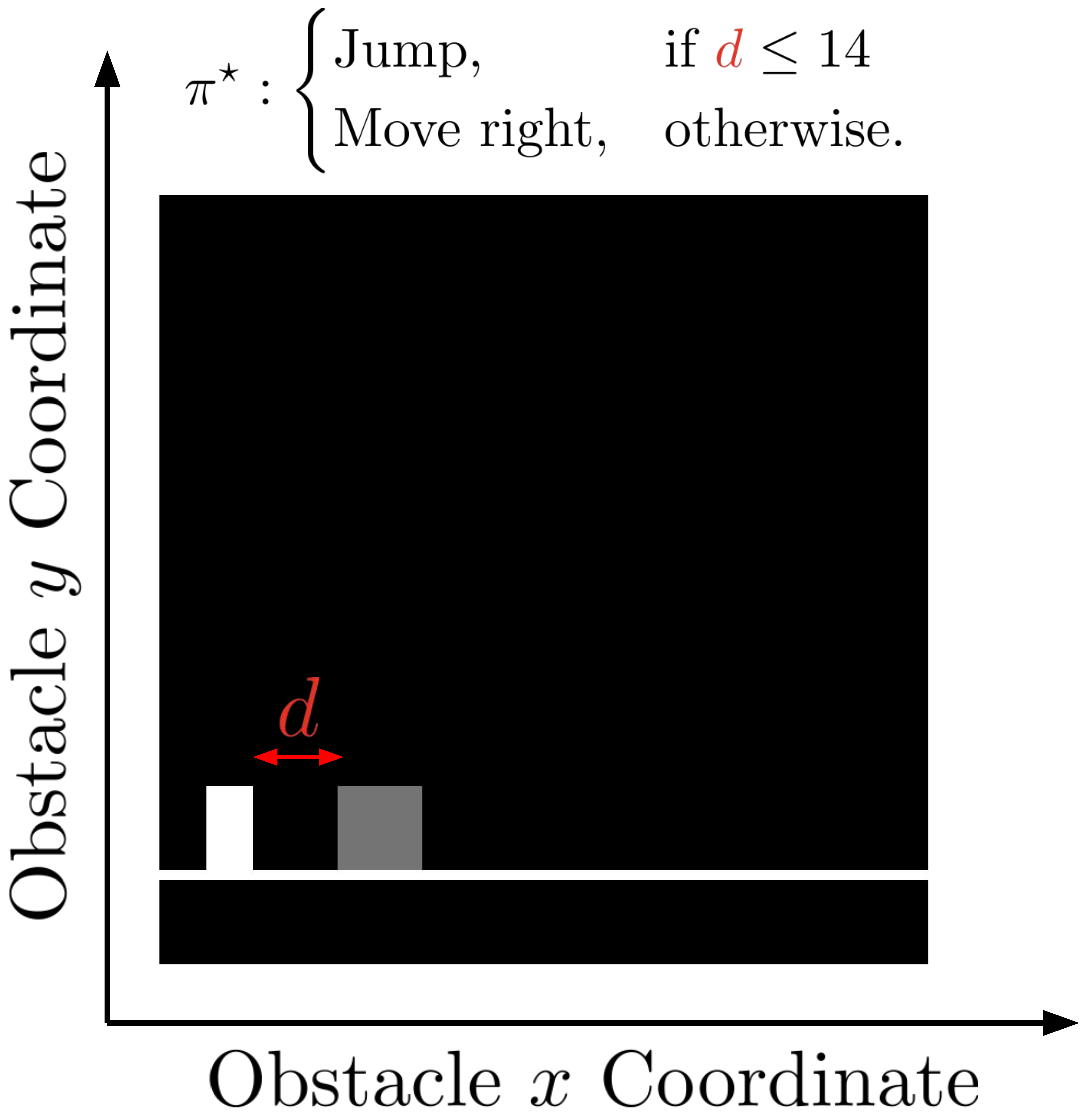
Fast-adaptation inductive biases for RL with MSR Jump

- **Testbed 2: MSR Jump**
  - Pretraining: white agent **jumps over gray** box (75 pretraining tasks).
  - Downstream: jump over **unseen** box positions (75 evaluation tasks).

Task 1	**Task 2**	**Task 3**	**Task 4**	**Task 5**
Floor: 15	Floor: 15	Floor: 15	Floor: 15	Floor: 15
Position: 20	Position: 30	Position: 40	Position: 50	Position: 60

Fast-adaptation inductive biases for RL with MSR Jump

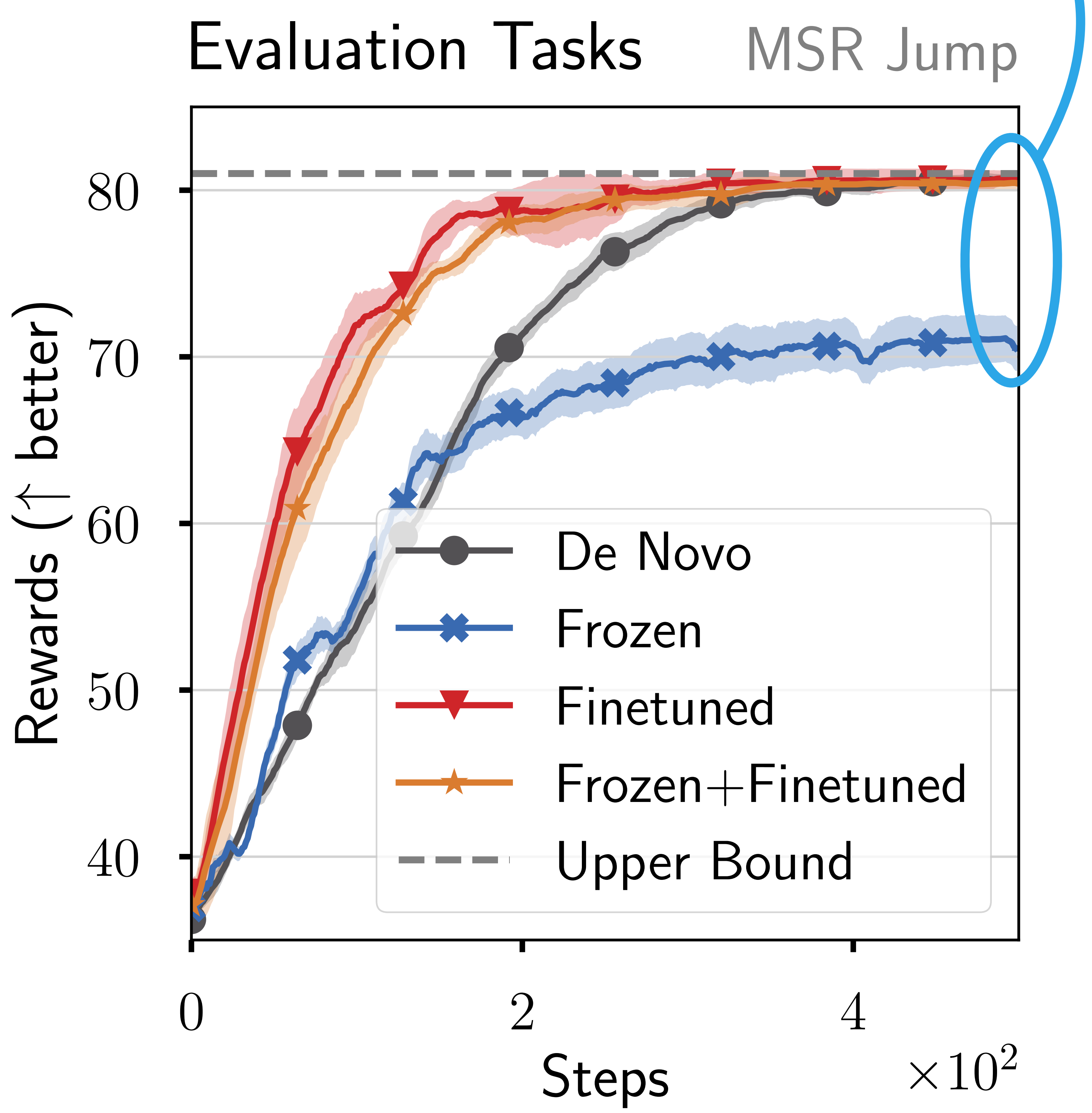
- Testbed 2: MSR Jump
  - Pretraining: white agent **jumps over gray** box (75 pretraining tasks).
  - Downstream: jump over **unseen** box positions (75 evaluation tasks).



A failure mode for transfer in RL

- **Setup**
  - Run our baselines on MSR Jump.
- **Results**
  - Frozen features underperform De Novo finetuning.
- **What is wrong with Frozen features?**

Frozen fails on Jump?

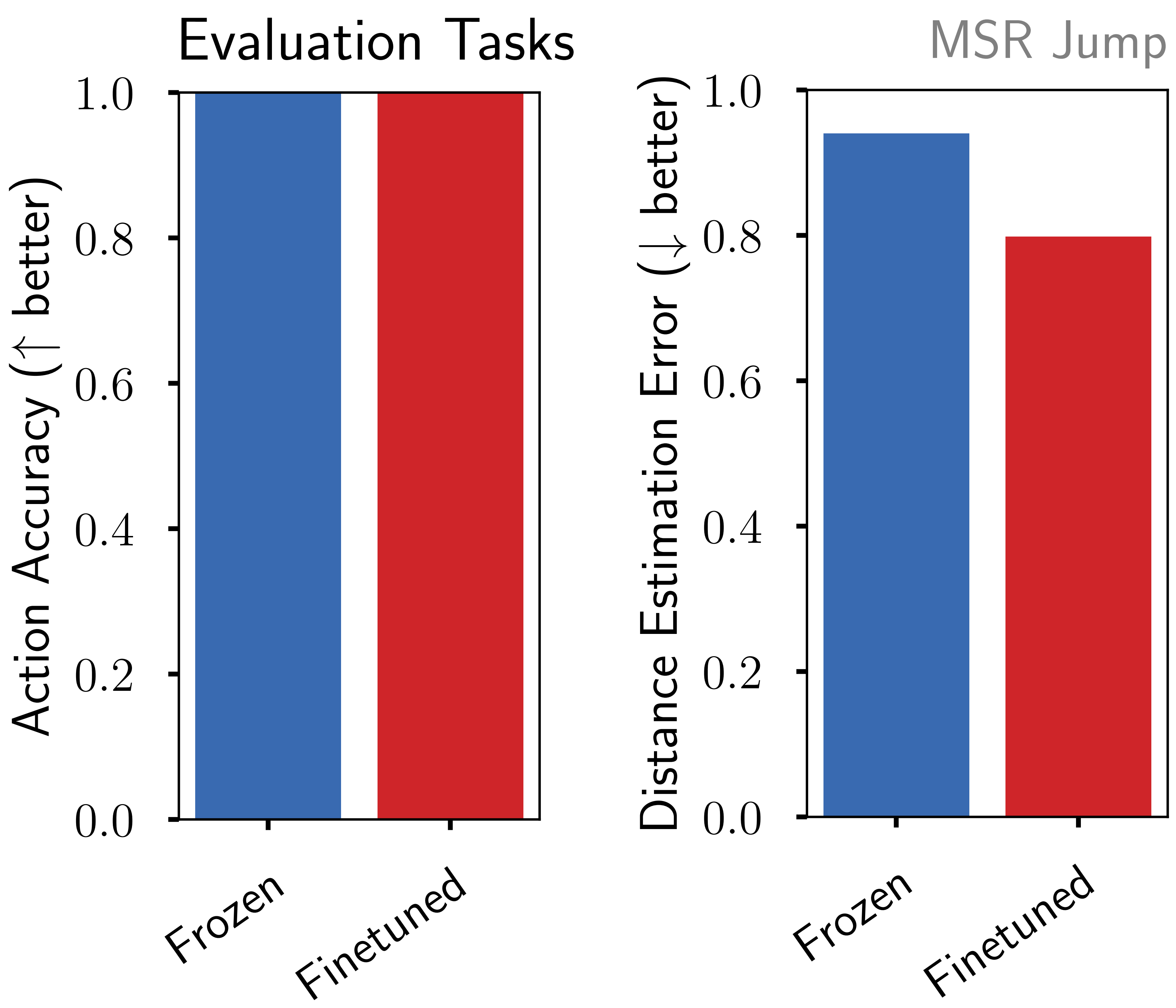


Are Frozen features informative enough?

- **Setup**
  1. Collect optimal trajectories.
  2. Measure quality of Frozen / Finetuned features:
     - Regress optimal actions (Accuracy).
     - Regress distance to box (MSE).

- **Results**
  - Perfect action accuracy (100%).
  - Perfect distance estimation (sub-pixel).

- **Yes, pretrained features are informative enough.**
  - What makes finetuned features so good?

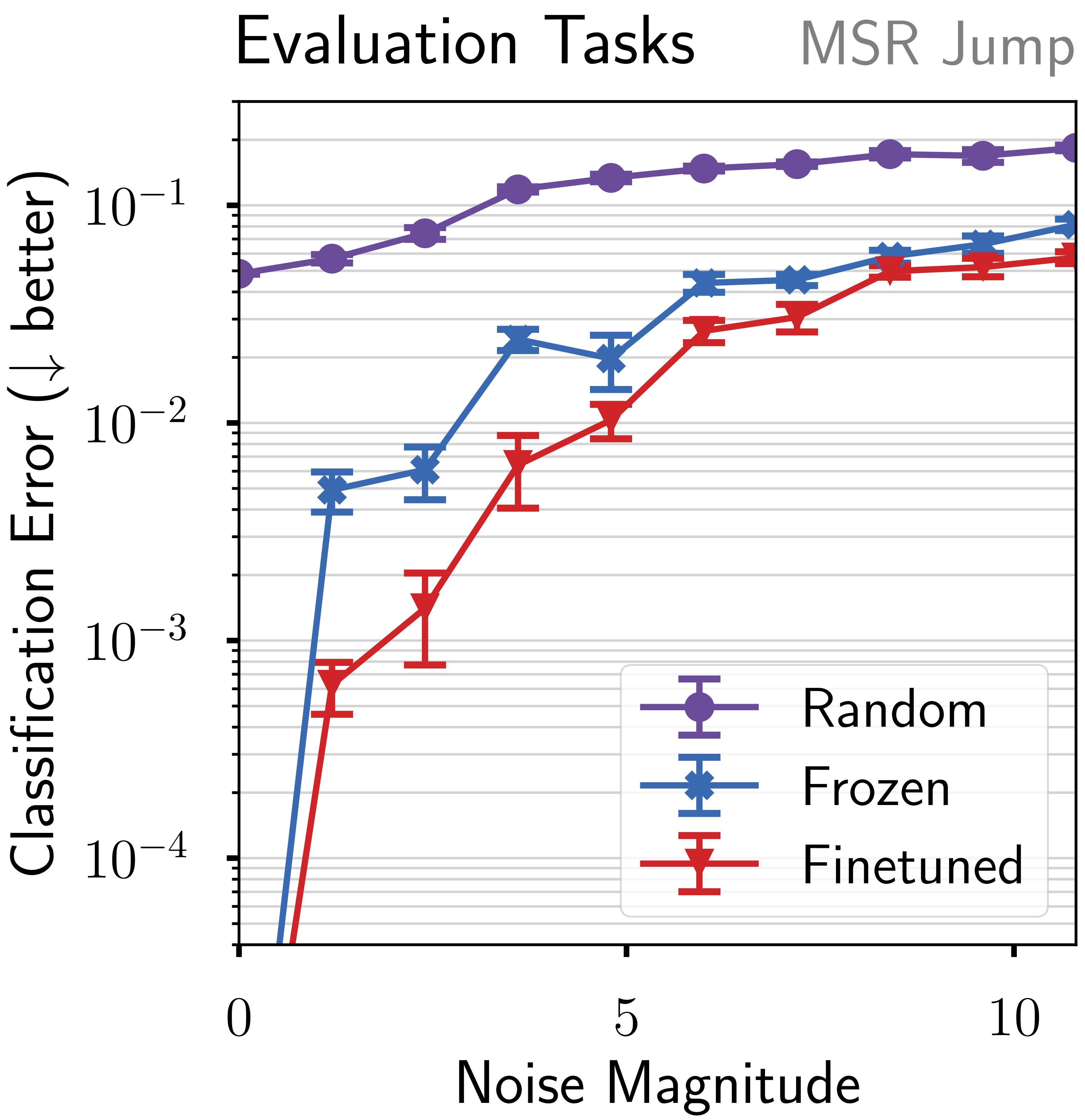


Good features are robust to noise

- **Setup**
  1. Collect optimal trajectories.
  2. Regress optimal actions from noisy features.
  3. Measure classification error.
  4. Repeat for a different noise level.

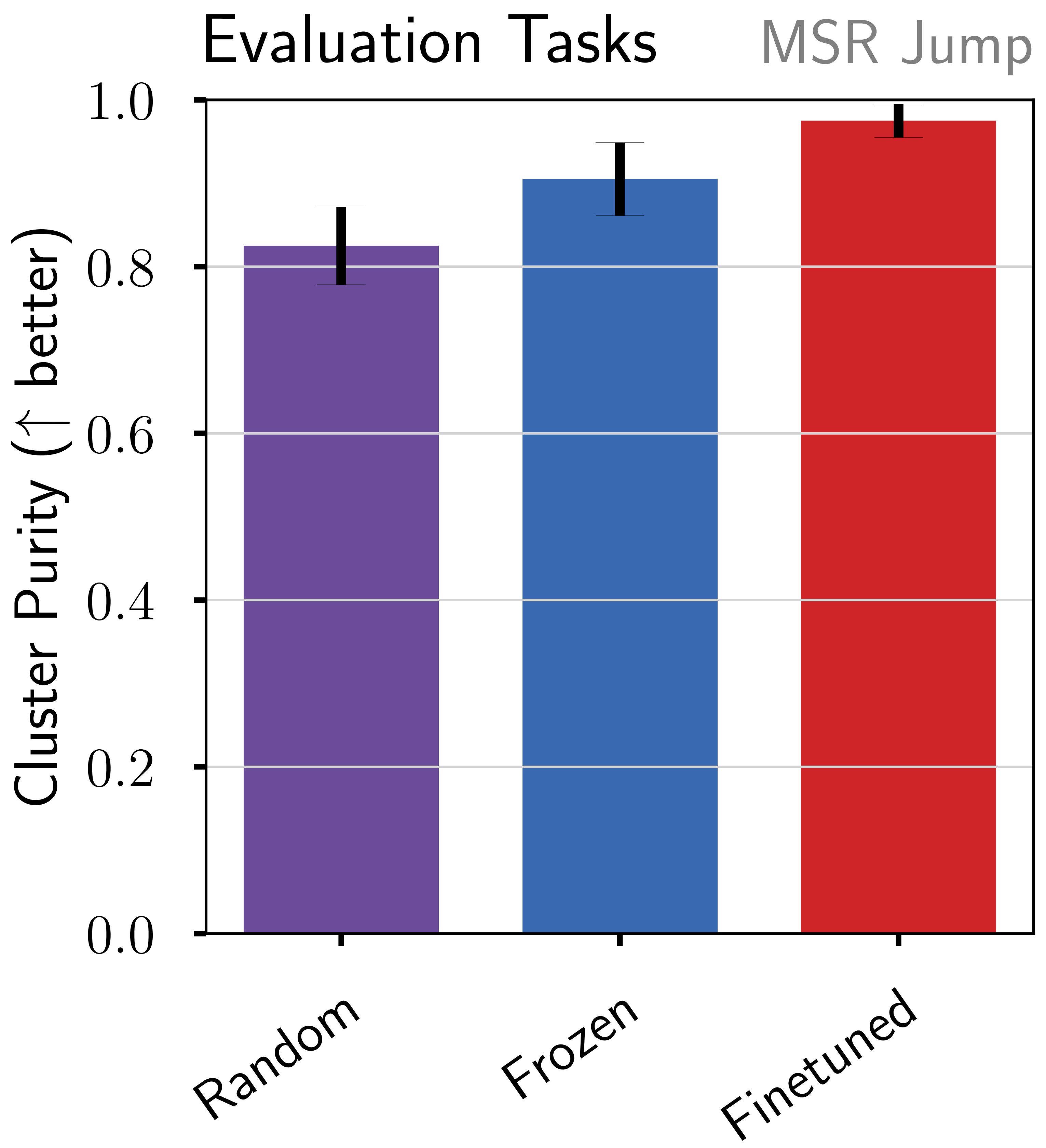
- **Results**
  - Adapted features degrade slower than pretrained ones.

- **Idea 1: enforce noise robustness while finetuning.**

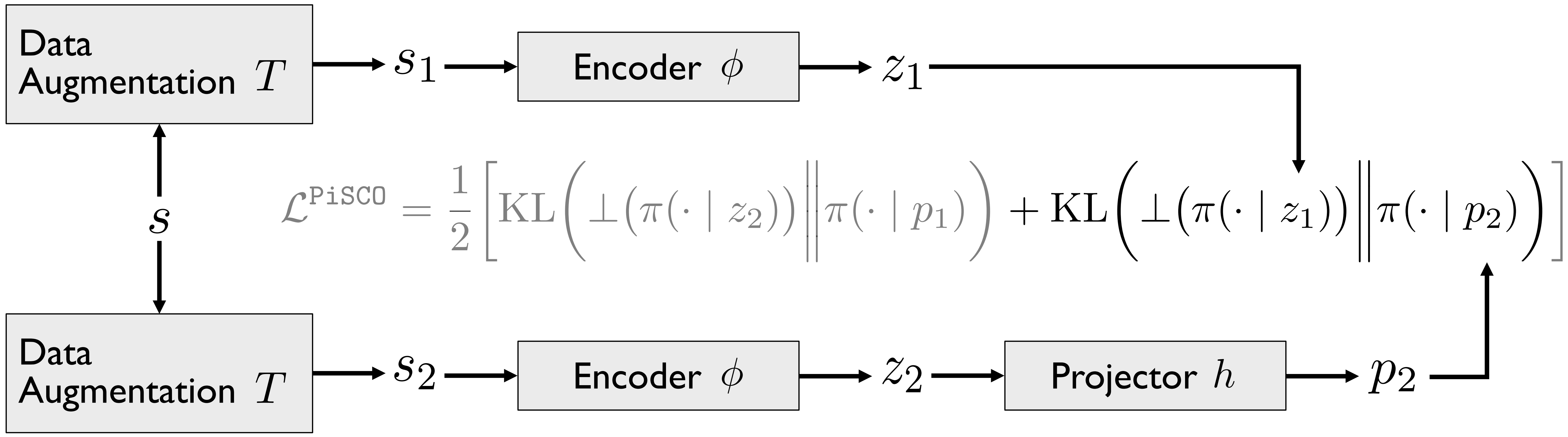


Good features ease decision making

- **Setup**
  - Collect optimal trajectories.
  - Compute features for all observations.
  - Measure cluster purity.
    - Given 5 nearest neighbors, how many induce the same optimal action?
- **Results**
  - Finetuned features yield purer clusters.
- **Idea 2: ensure similar states lead to similar policies.**



Policy-induced self-consistency objective (PiSCO)



Policy-induced self-supervision

$$\mathcal{L}^{\texttt{PiSCO}} = \mathop{\mathbb{E}}_{\substack{s \sim \mathcal{B} \\ s_1, s_2 \sim \mathrm{T}(\cdot|s)}} \left[\frac{1}{2}\left(\mathcal{D}(z_1, p_2) + \mathcal{D}(z_2, p_1)\right)\right]$$

where $\mathcal{D}(z, p) = \mathbf{KL}\left(\perp\left(\pi(\cdot \mid z)\right) \,||\, \pi(\cdot \mid p)\right)$

Recipe: PiSCO with SimSiam

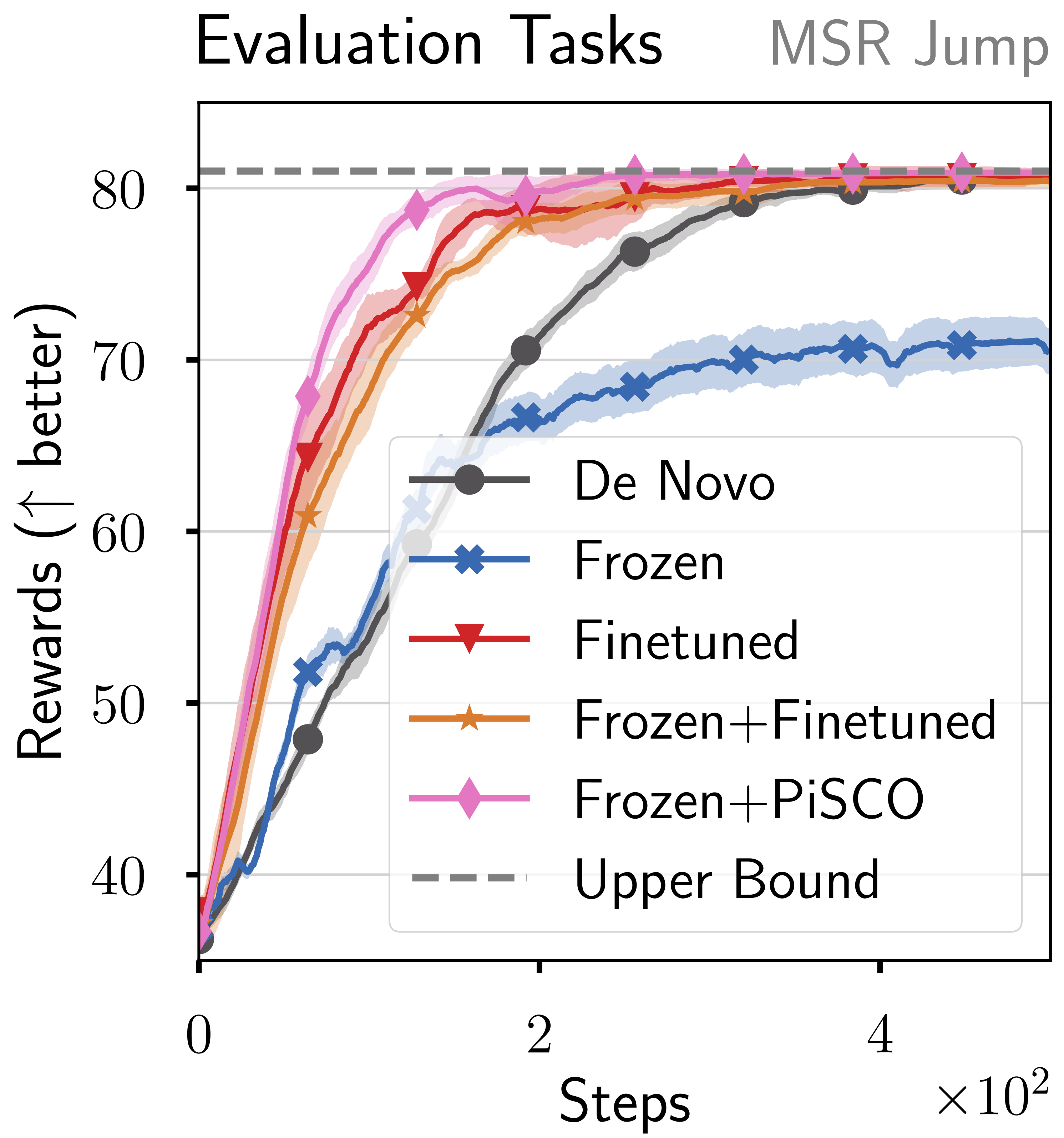
1. Sample state s from replay $\mathcal{B}$.
2. Augment s into s_1, s_2.
3. Compute SimSiam objective with KL of induced policy (not L2-norm).

Chen & He, « Exploring Simple Siamese Representation Learning », CVPR 2021

PiSCO accelerates RL finetuning

- **Frozen+PiSCO**
  - Initialization: **pretrained**.
  - Task-agnostic layers: **frozen**.
  - Task-specific layers: **PiSCO**.

- **Results**
  - PiSCO **accelerates RL finetuning** on Jump and Habitat.

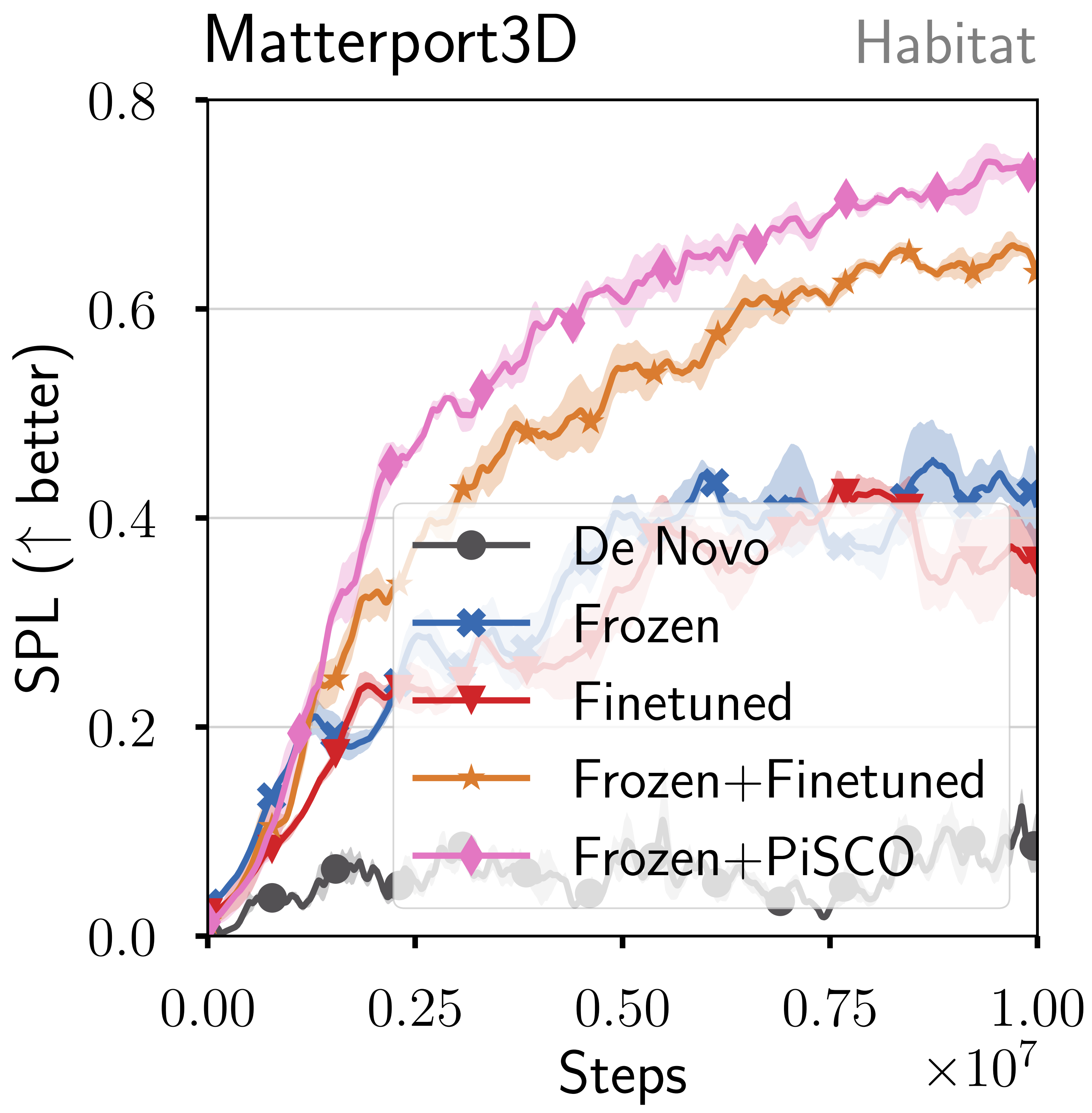


PiSCO accelerates RL finetuning

- **Frozen+PiSCO**
  - Initialization: **pretrained**.
  - Task-agnostic layers: **frozen**.
  - Task-specific layers: **PiSCO**.

- **Results**
  - PiSCO **accelerates RL finetuning** on Jump and Habitat.

- **Take-away**
  - Finetuning works better with **RL-specific inductive biases**.



Take-aways — Part II

Q1: How to adapt fast?
Freeze task-agnostic parameters; learn optimization parameters.

Q2: When is adaptation required?
When the new tasks are different from train tasks.

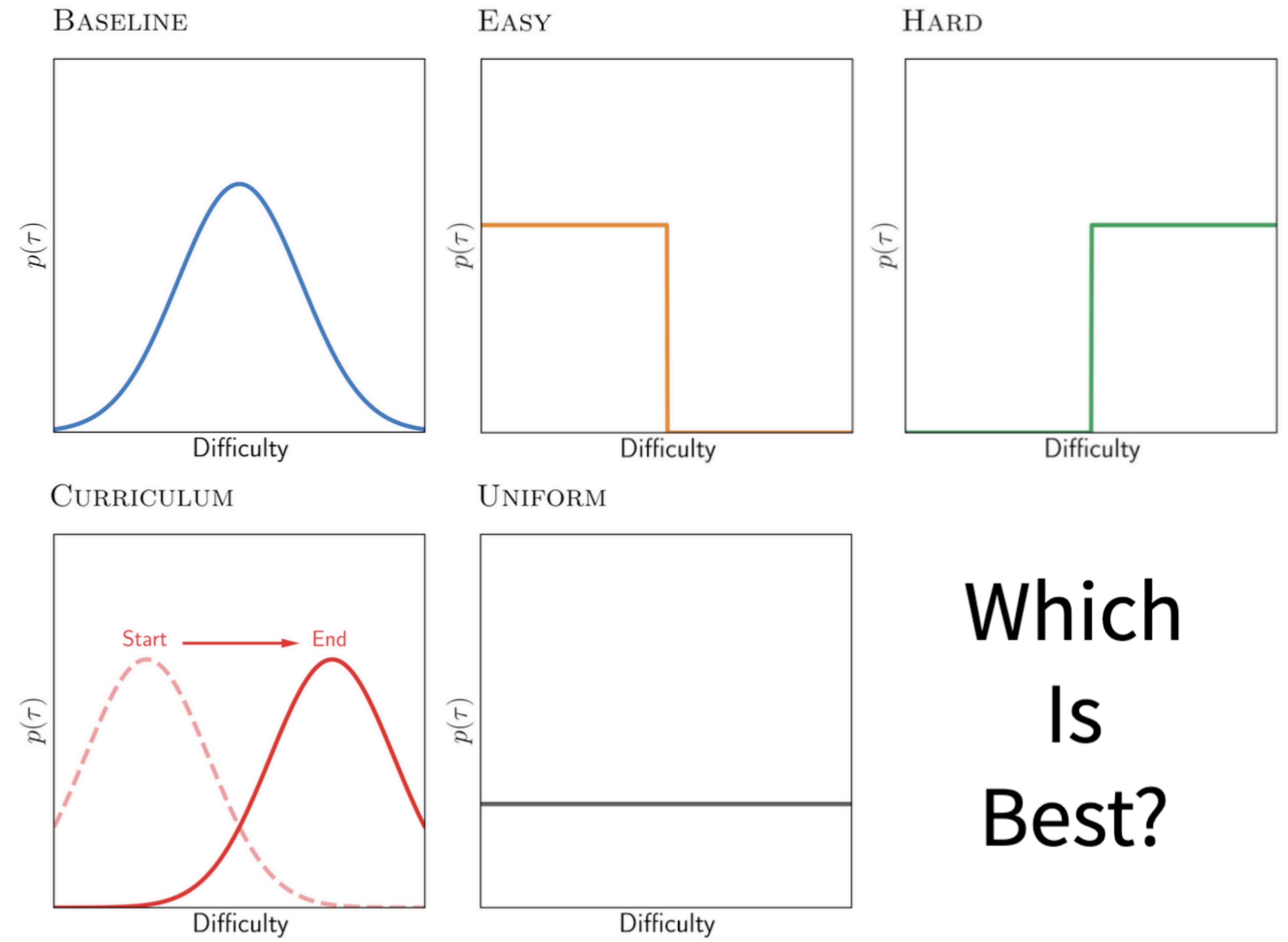
Q3: How to adapt quickly with reinforcement learning?
Freeze task-agnostic parameters; finetune with a policy-induced objective.

Meta-Learning with Many Tasks

Part III

Q4: How to choose training tasks?

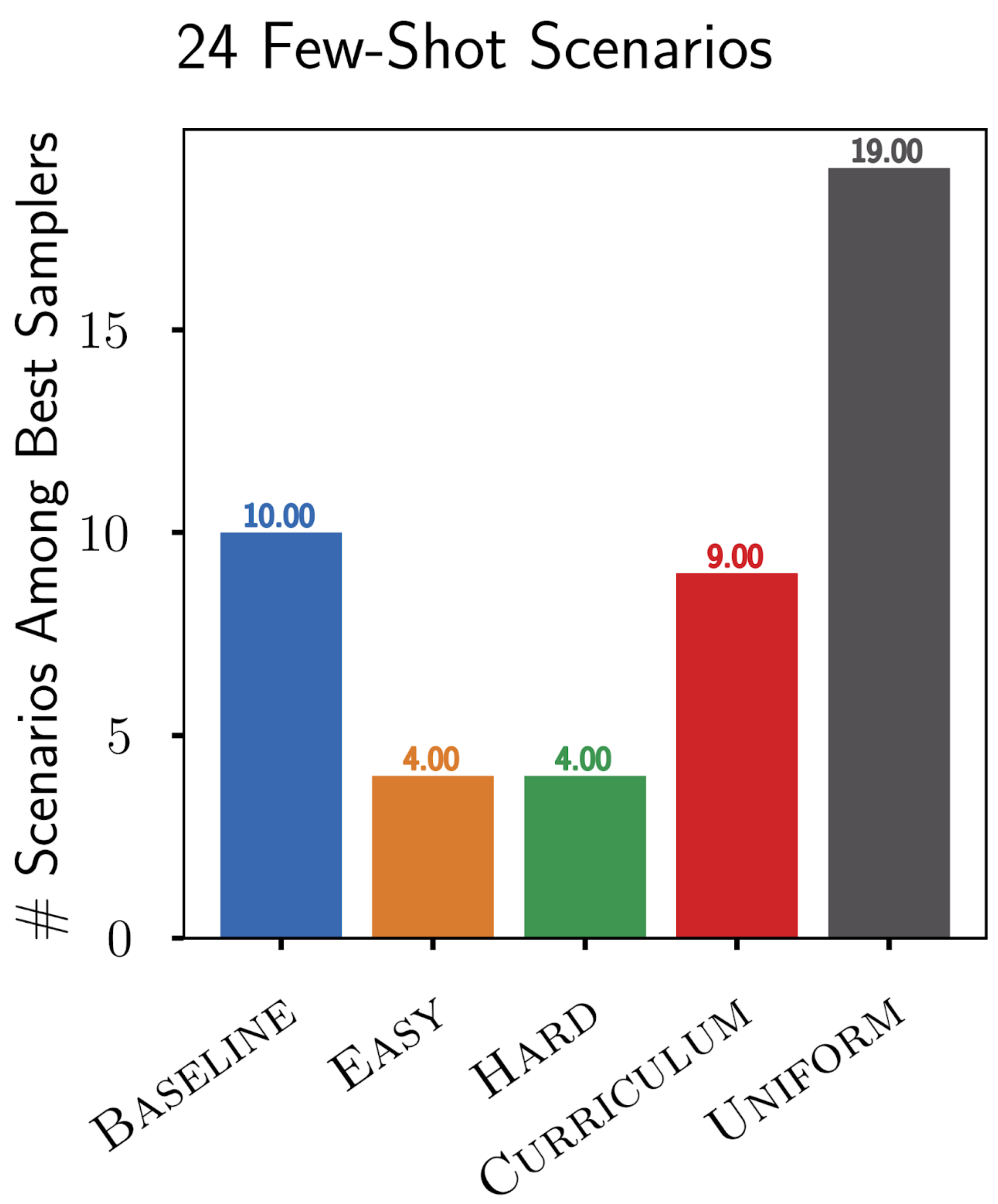
- **Change of assumptions**
  - What if we get to pick which task to train on?
- **Motivation**
  - Streaming *v.s.* **offline** tasksets.
  - Some tasks are more **informative**.
  - « Does sampling **even matter** in few-shot learning? »
- **Core question**
  - How to sample tasks for **best test accuracy**?



Which Is Best?

Sampling matters for episodic training

- **Compare 5 candidate distributions on:**
  - 2 architectures: CNN4, ResNet12.
  - 4 algorithms: MAML, ANIL, ProtoNet (Euclidean & Cosine).
  - 2 datasets: mini-ImageNet, tiered-ImageNet.
  - 2 settings: 5-ways 1-shot & 5-shots.
- **Results**
  - Uniform sampling dominates, Baseline second best.

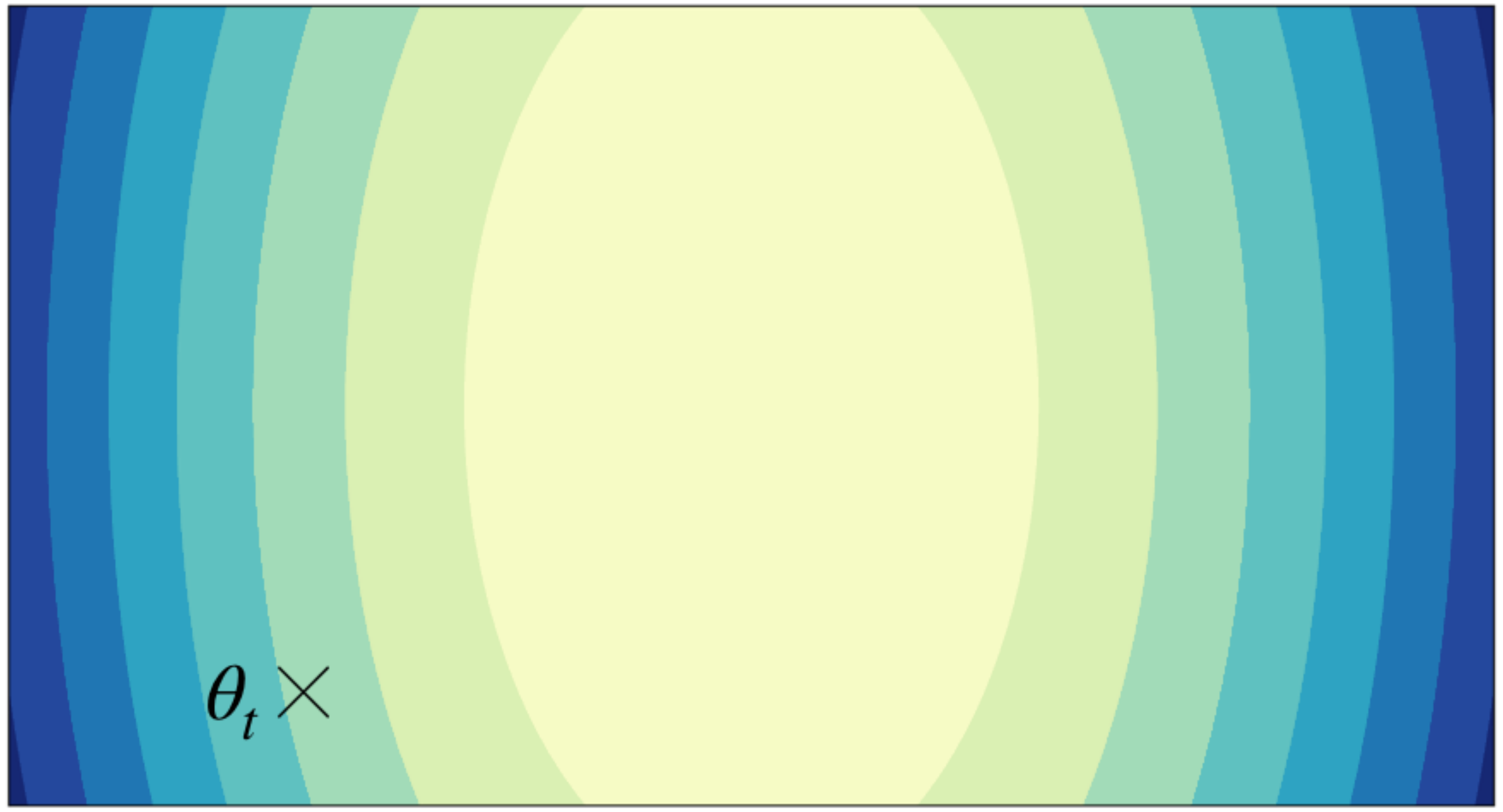


Q5: How to optimize with many tasks?

- **Motivation**
  - mini-ImageNet ➞ 10^{162} different tasks.
  - Joint training over all tasks is intractable.
  - Solution: sample tasks one at a time.

- **Issue: the memoryless SGD**
  - Immediately discards gradients after they are used

- **Core question**
  - How do we reuse information seen in previous tasks?



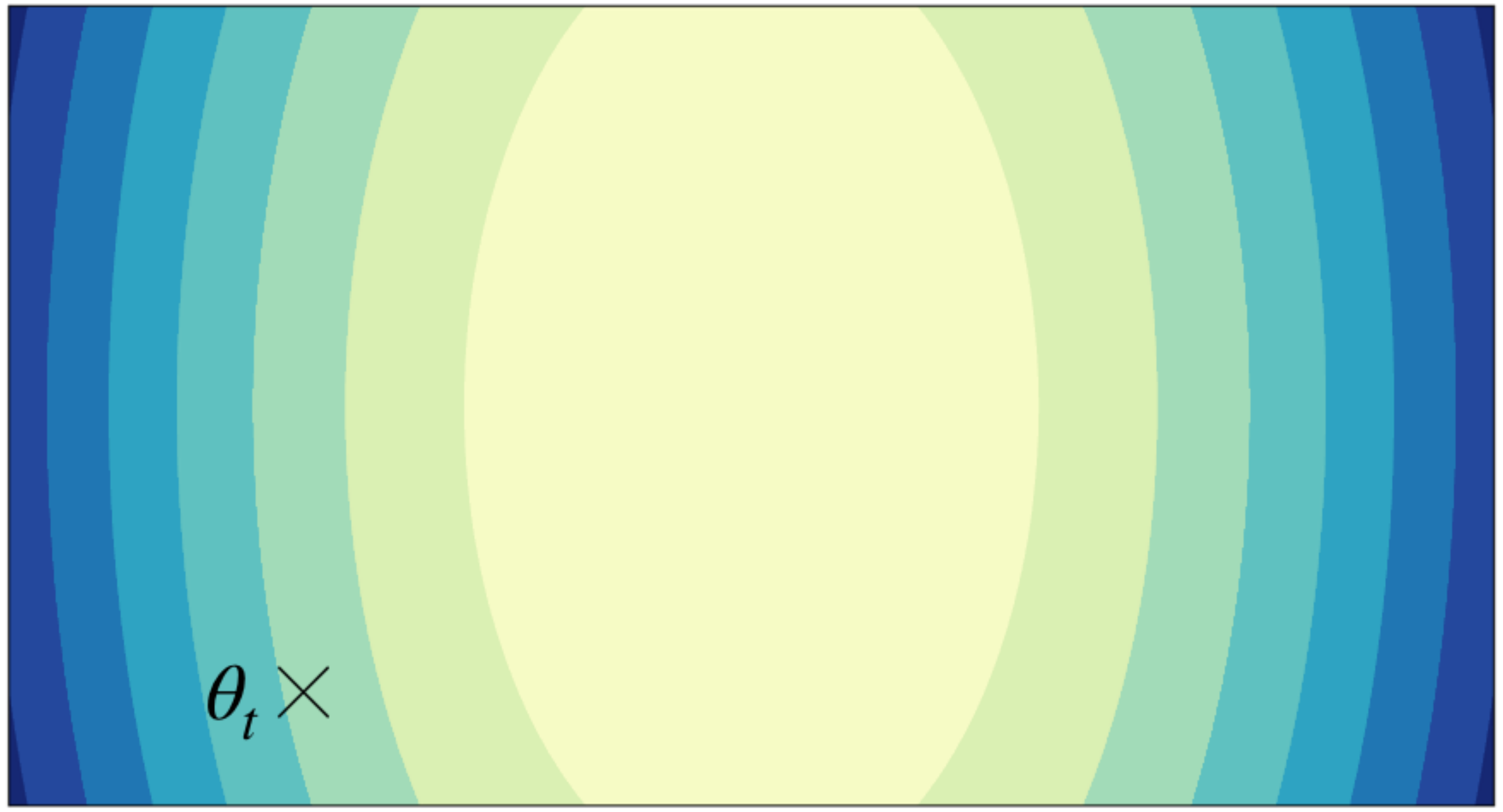
Reference: Arnold *et al.*, *« Reducing the variance in online optimization by transporting past gradients », NeurIPS 2019* (Spotlight)

Q5: How to optimize with many tasks?

- **Motivation**
  - mini-ImageNet → 10^{162} different tasks.
  - Joint training over all tasks is intractable.
  - Solution: sample tasks one at a time.

- **Issue: the memoryless SGD**
  - Immediately discards gradients after they are used

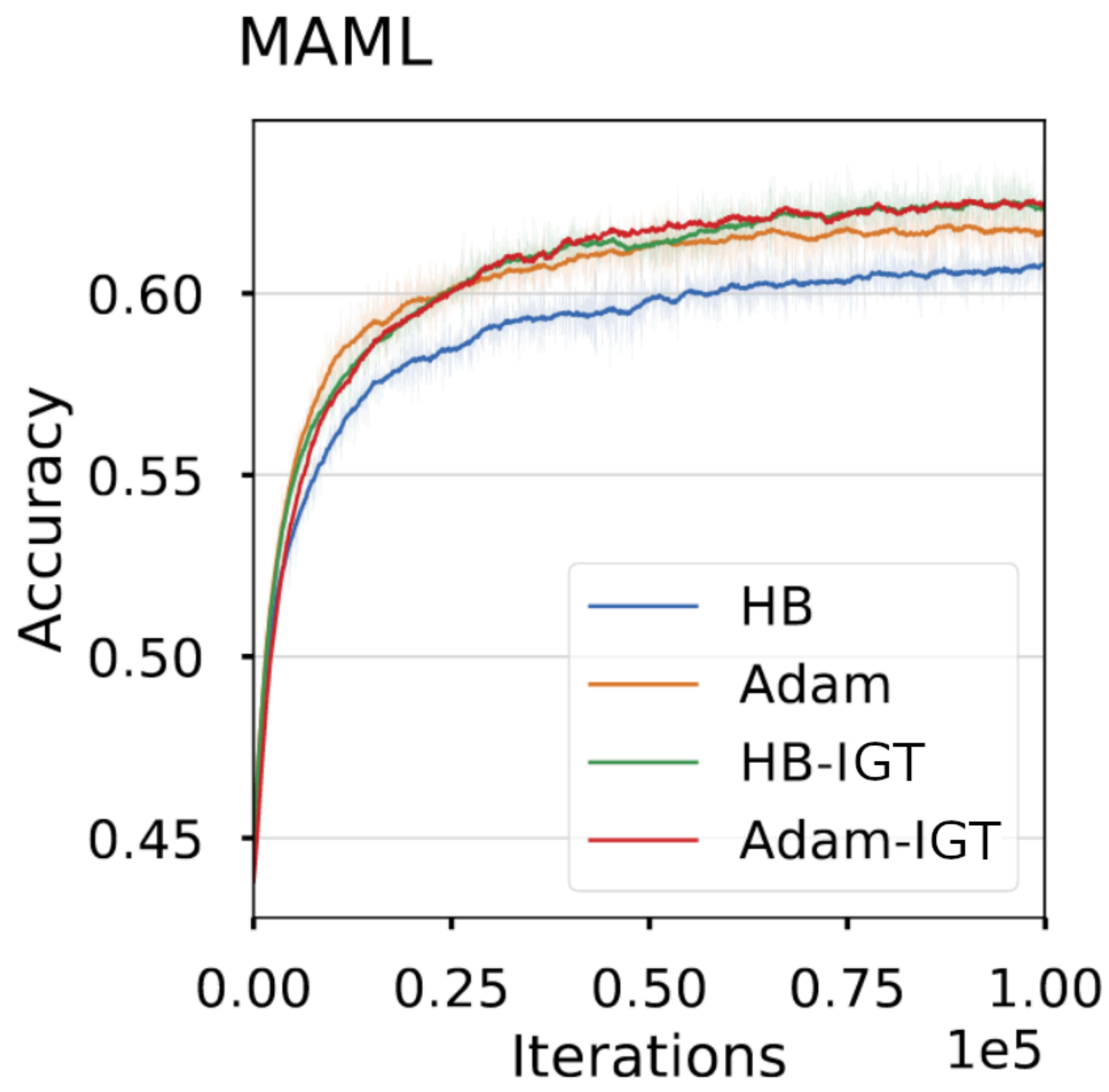
- **Core question**
  - How do we reuse information seen in previous tasks?



Reference: Arnold *et al.*, *« Reducing the variance in online optimization by transporting past gradients », NeurIPS 2019* (Spotlight)

IGT improves meta-learning

- Setup
  - MAML on mini-ImageNet, 5 adaptation steps.
  - 5-ways 5-shots tasks.
  - Only replace the **task-level optimizer**:
    - Fast adaptation with SGD.
    - Meta-learning with IGT.
- Results
  - IGT optimizers improve upon Heavyball and Adam.



Take-aways — Part III

Q1: How to adapt fast?
Freeze task-agnostic parameters; learn optimization parameters.

Q2: When is adaptation required?
When the new tasks are different from train tasks.

Q3: How to adapt quickly with reinforcement learning?
Freeze task-agnostic parameters; finetune with a policy-induced objective.

Q4: How to choose training tasks?
Sample them uniformly over difficulty.

Q5: How to optimize with many tasks?
Retain the information from tasks prior tasks.

Papers in this thesis

Q1: « When MAML Can Adapt Fast and How to Assist When it Cannot »
S. M. R. Arnold, S. Iqbal, and F. Sha. AISTATS, 2021.

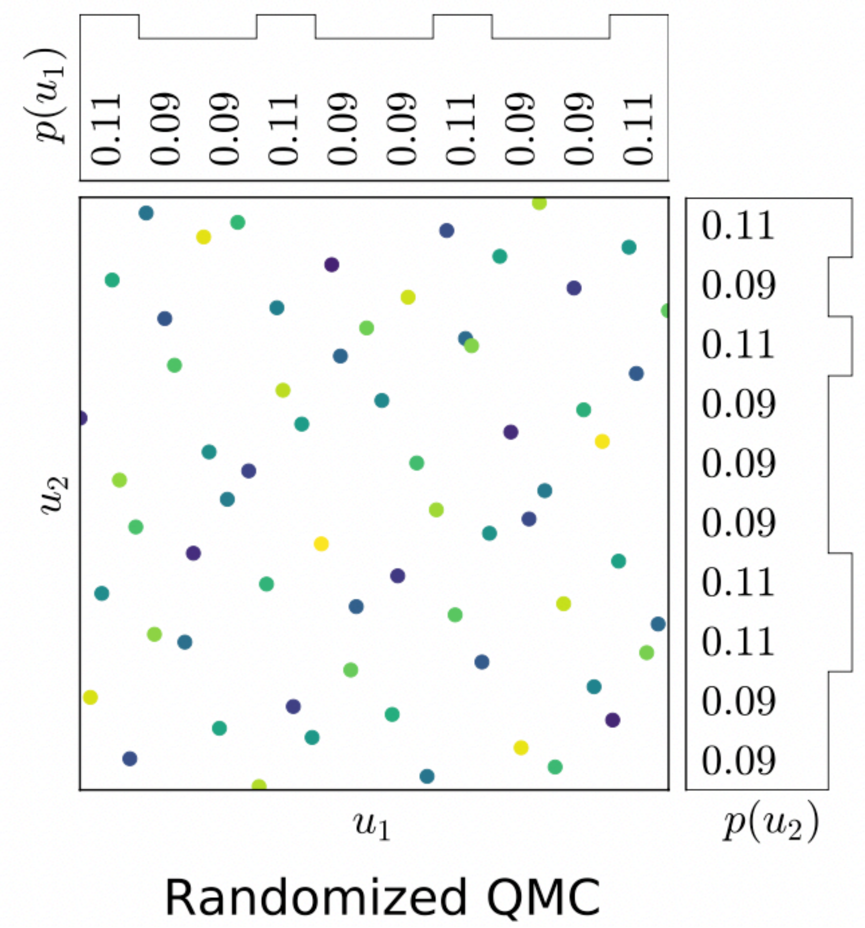
Q2: « Embedding Adaptation is Still Needed for Few-Shot Learning »
S. M. R. Arnold and F. Sha. ArXiv Preprints, 2021.

Q3: « Policy-Induced Self-Supervision Improves Representation Finetuning in Visual RL »
S. M. R. Arnold and F. Sha. In Submission.

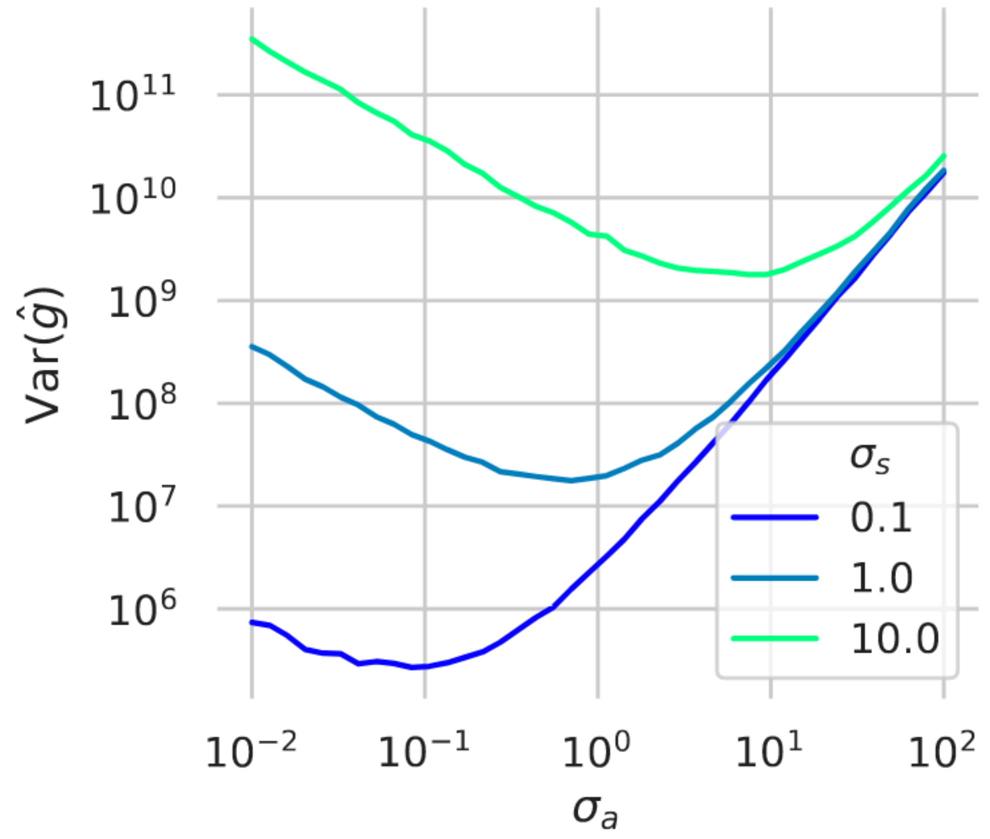
Q4: « Uniform Sampling over Episode Difficulty »
S. M. R. Arnold, G. S. Dhillon*, A. Ravichandran, and S. Soatto. NeurIPS, 2021.*

Q5: « Reducing the Variance in Online Optimization by Transporting Past Gradients »
S. M. R. Arnold, P-A. Manzagol, R. Babanezhad, I. Mitliagkas, N. Le Roux. NeurIPS, 2019.

Papers not in this thesis



- « **Policy Learning and Evaluation with RQMC** »
  - *S. M. R. Arnold, P. L'Ecuyer, L. Chen, Y-F. Chen, and F. Sha.* ***AISTATS, 2022.***
  - Replaces Monte Carlo sampling with Randomized Quasi-Monte Carlo in RL.



- « **Analyzing the Variance of Policy Gradient Estimators for the LQR** »
  - *J. A. Preiss*, S. M. R. Arnold*, C-Y. Wei*, and M. Kloft.* ***NeurIPS 2019.***
  - Derives bounds for the variance of REINFORCE on LQR.

- « **learn2learn: A Library for Meta-Learning Research** »
  - *S. M. R. Arnold, P. Mahajan, D. Datta, I. Bunner, and K. S. Zarkias.* ***ArXiv Preprints, 2020.***
  - Software package, 27 contributors, 2.1k ★ on GitHub.

learn2learn

Outlook

- **A theory for meta-optimization**
  - Scalable meta-optimizers.
  - Meta-overfitting, meta-augmentation, meta-bias, …
  - Emergence of optimization parameters.

- **Defining and measuring task similarity**
  - Data, model, and learning algorithm → establish guidelines for practitioners.
  - For RL tasks?

- **Analyses grounded in real-world tasks**
  - No real-world task today!

« *Only experiments with real Creatures in real worlds can answer the natural doubts about our approach.* »
R. A. Brooks. *Intelligence without representation.* AI'91.