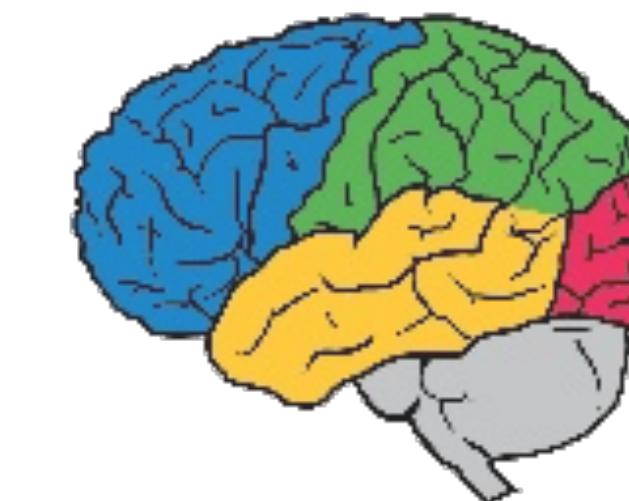




Meta-Learning: from Few-Shot Learning to Rapid Reinforcement Learning

Chelsea Finn

Sergey Levine

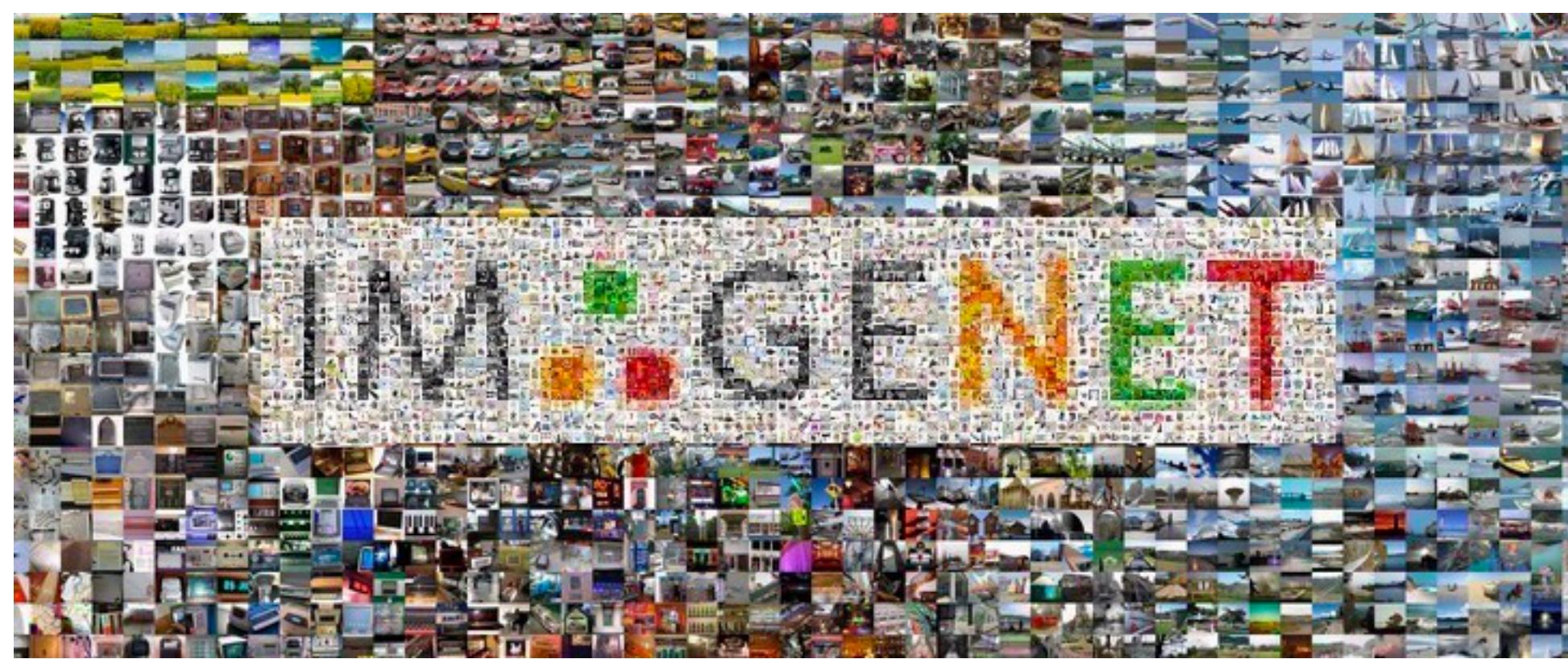


BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Large, diverse data
(+ large models)



Broad generalization



Russakovsky et al. '14

GPT-2

Radford et al. '19

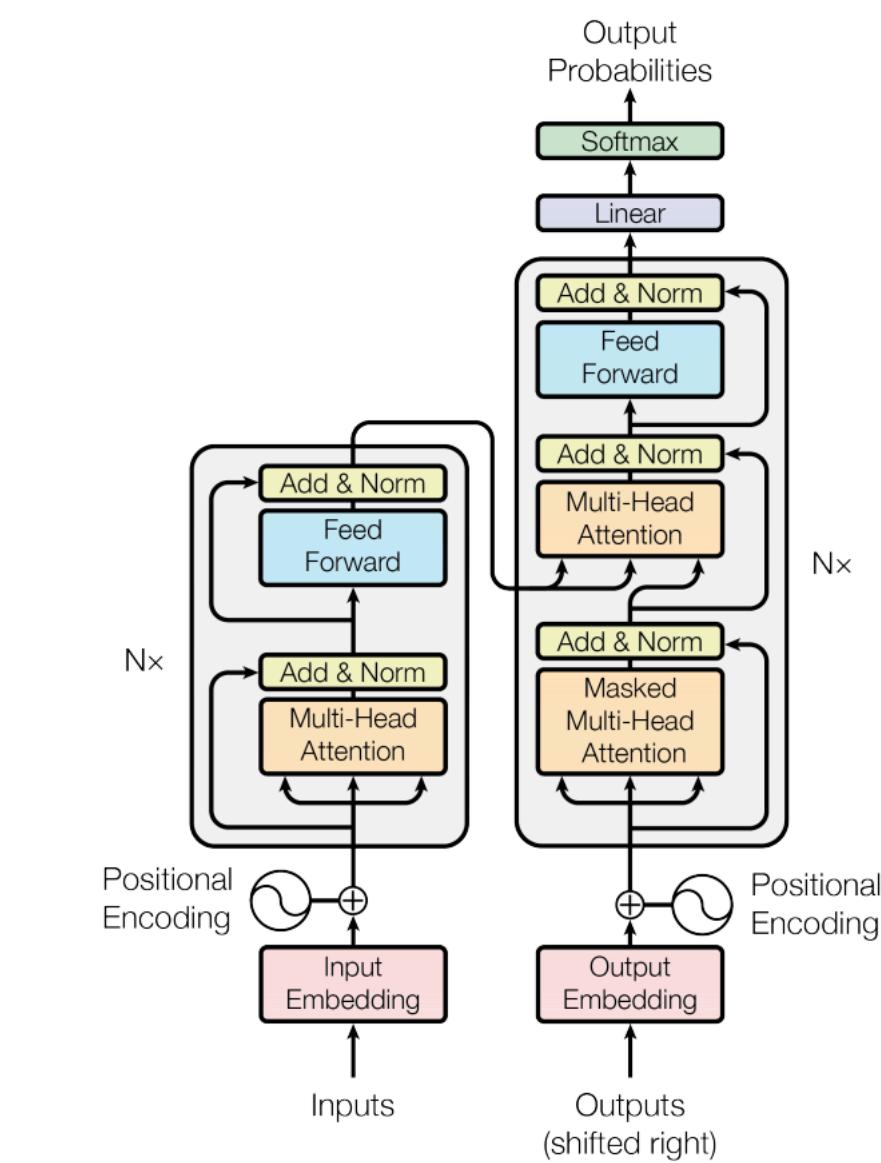


Figure 1: The Transformer - model architecture.

Vaswani et al. '18

Under the paradigm of **supervised learning**.

What if you don't have a large dataset?

medical imaging

robotics

personalized education,

translation for rare languages

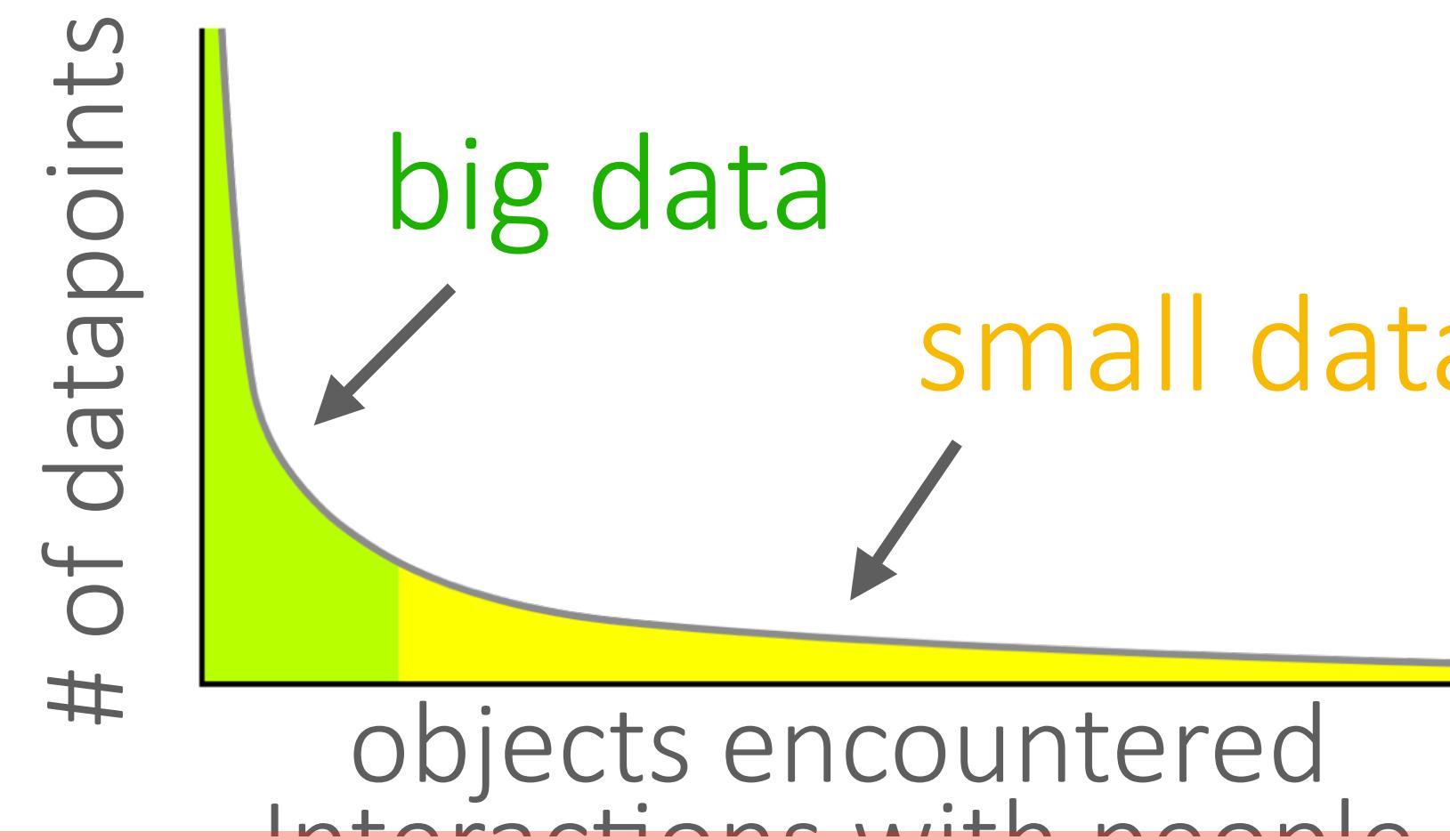
recommendations

What if you want a general-purpose AI system in the real world?

Need to continuously adapt and learn on the job.

Learning each thing from scratch won't cut it.

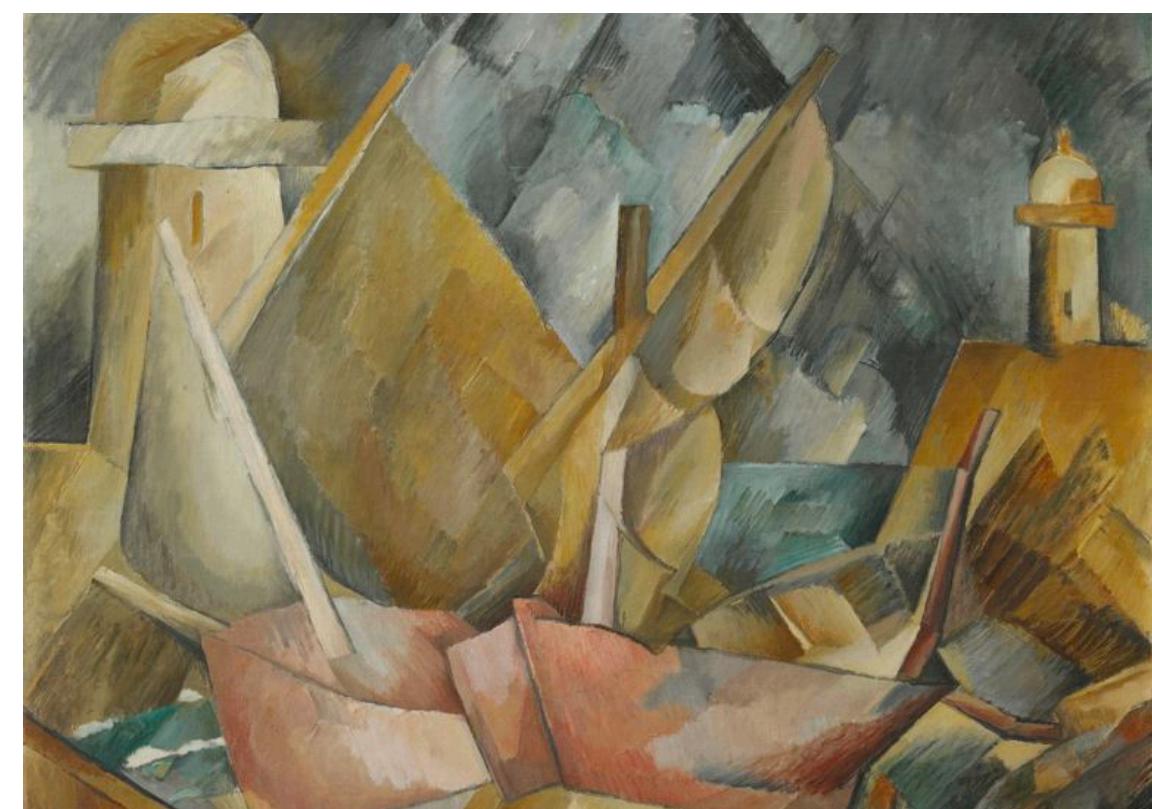
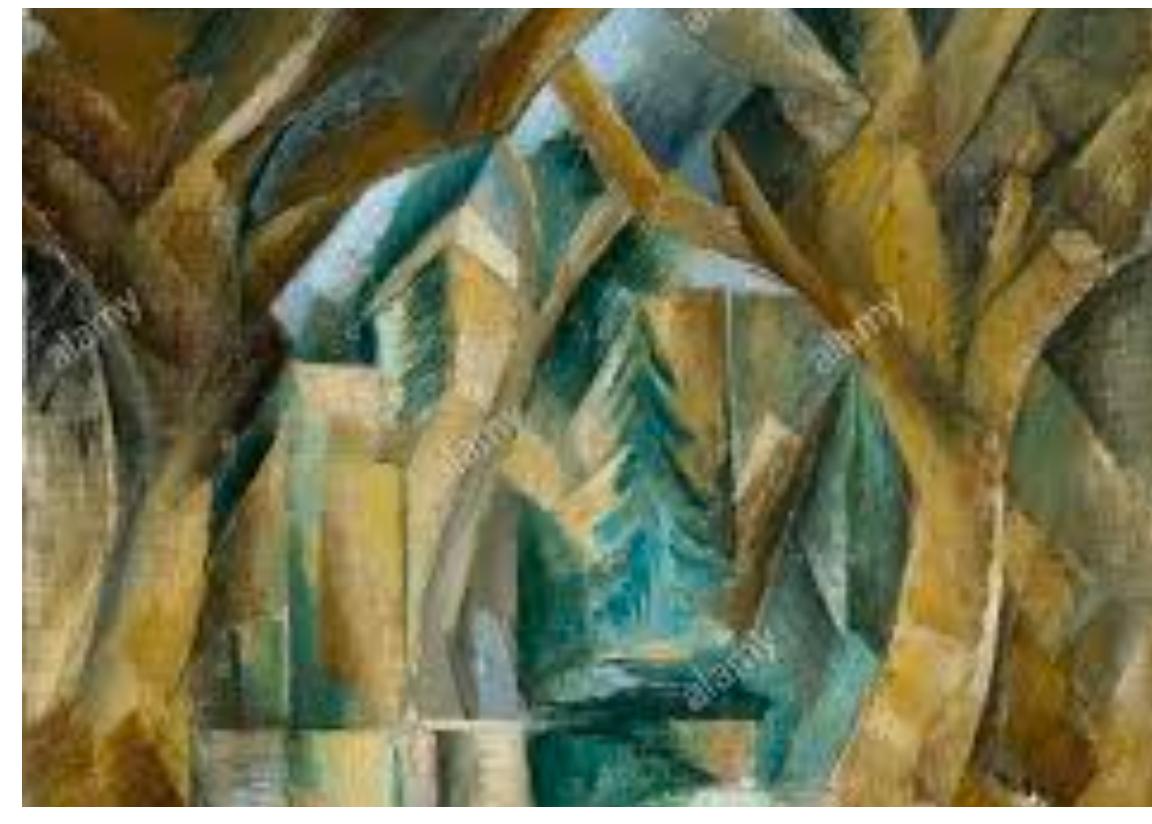
What if your data has a long tail?



These settings break the supervised learning paradigm.

training data

Braque



Cezanne



test datapoint



By Braque or Cezanne?

How did you accomplish this?

Through previous experience.

How might you get a machine to accomplish this task?

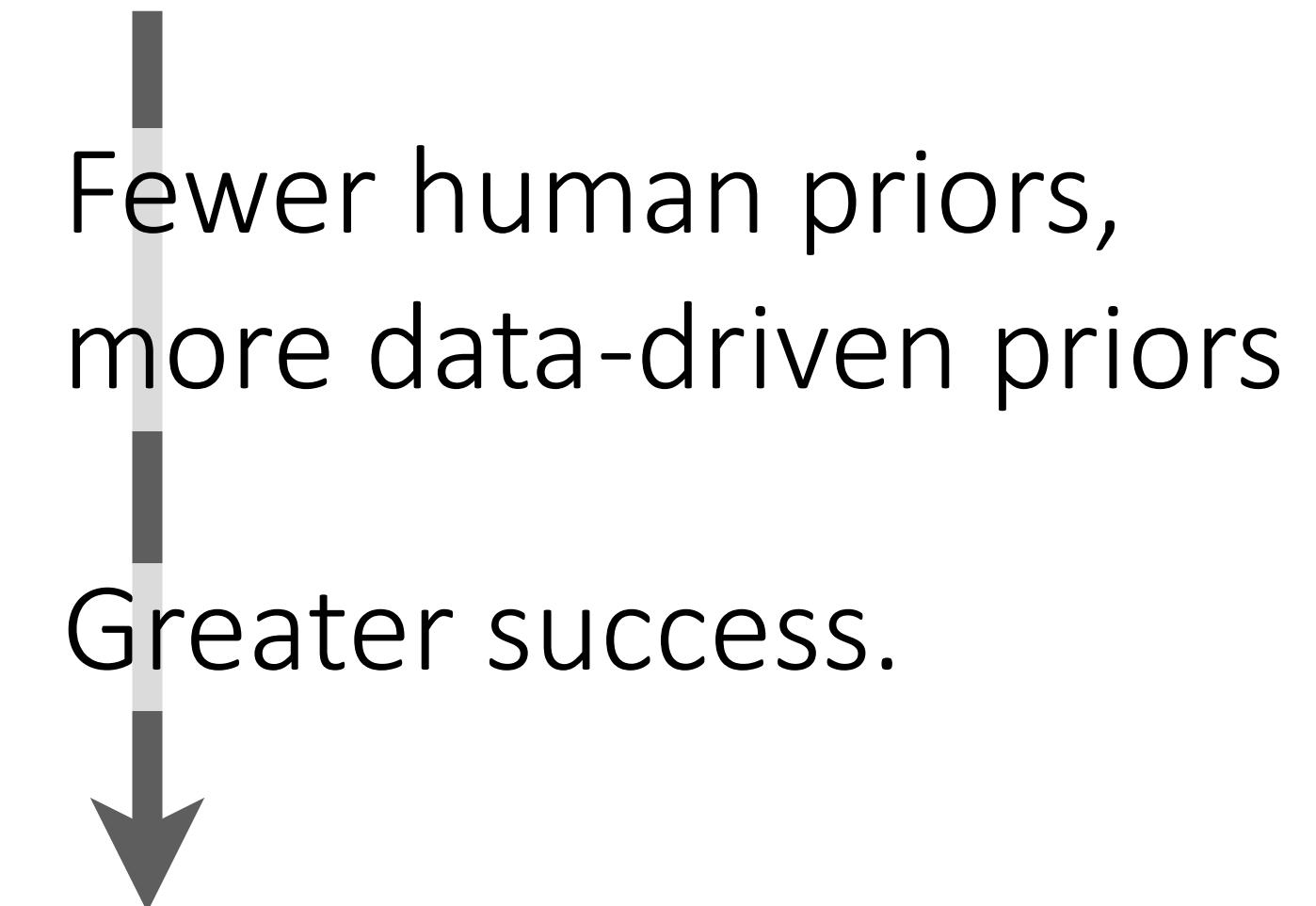
Modeling image formation

Geometry

SIFT features, HOG features + SVM

Fine-tuning from ImageNet features
Domain adaptation from other painters

???



Can we explicitly **learn priors from previous experience**
that lead to efficient downstream learning?

Outline

- **Problem statement**
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

Two ways to view meta-learning

Mechanistic view

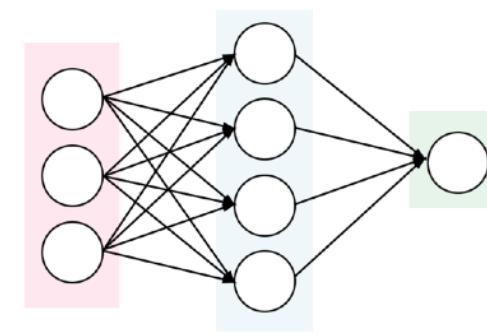
- Deep neural network model that can read in an entire dataset and make predictions for new datapoints
- Training this network uses a meta-dataset, which itself consists of many datasets, each for a different task
- This view makes it easier to implement meta-learning algorithms

Probabilistic view

- Extract prior information from a set of (meta-training) tasks that allows efficient learning of new tasks
- Learning a new task uses this prior and (small) training set to infer most likely posterior parameters
- This view makes it easier to understand meta-learning algorithms

Problem definitions

supervised learning:



$$\arg \max_{\phi} \log p(\phi | \mathcal{D})$$

model parameters training data

$$\mathcal{D} = \{(x_1, y_1), \dots, (x_k, y_k)\}$$

input (e.g., image) label

$$= \arg \max_{\phi} \log p(\mathcal{D} | \phi) + \log p(\phi)$$

data likelihood regularizer (e.g., weight decay)

$$= \arg \max_{\phi} \sum_i \log p(y_i | x_i, \phi) + \log p(\phi)$$

What is wrong with this?

- The most powerful models typically require large amounts of labeled data
- Labeled data for some tasks may be very limited

Problem definitions

supervised learning:

$$\arg \max_{\phi} \log p(\phi | \mathcal{D})$$

$$\mathcal{D} = \{(x_1, y_1), \dots, (x_k, y_k)\}$$

can we incorporate *additional* data?

$$\arg \max_{\phi} \log p(\phi | \mathcal{D}, \mathcal{D}_{\text{meta-train}})$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$



The meta-learning problem

meta-learning:

$$\arg \max_{\phi} \log p(\phi | \mathcal{D}, \mathcal{D}_{\text{meta-train}})$$

$$\mathcal{D} = \{(x_1, y_1), \dots, (x_k, y_k)\}$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$

what if we don't want to keep $\mathcal{D}_{\text{meta-train}}$ around forever?

learn *meta-parameters* θ : $p(\theta | \mathcal{D}_{\text{meta-train}})$

whatever we need to know about $\mathcal{D}_{\text{meta-train}}$ to solve new tasks

$$\begin{aligned} \log p(\phi | \mathcal{D}, \mathcal{D}_{\text{meta-train}}) &= \log \int_{\Theta} p(\phi | \mathcal{D}, \theta) p(\theta | \mathcal{D}_{\text{meta-train}}) d\theta \\ &\approx \log p(\phi | \mathcal{D}, \theta^*) + \log p(\theta^* | \mathcal{D}_{\text{meta-train}}) \end{aligned}$$

$$\arg \max_{\phi} \log p(\phi | \mathcal{D}, \mathcal{D}_{\text{meta-train}}) \approx \arg \max_{\phi} \log p(\phi | \mathcal{D}, \theta^*)$$

this is the meta-learning problem

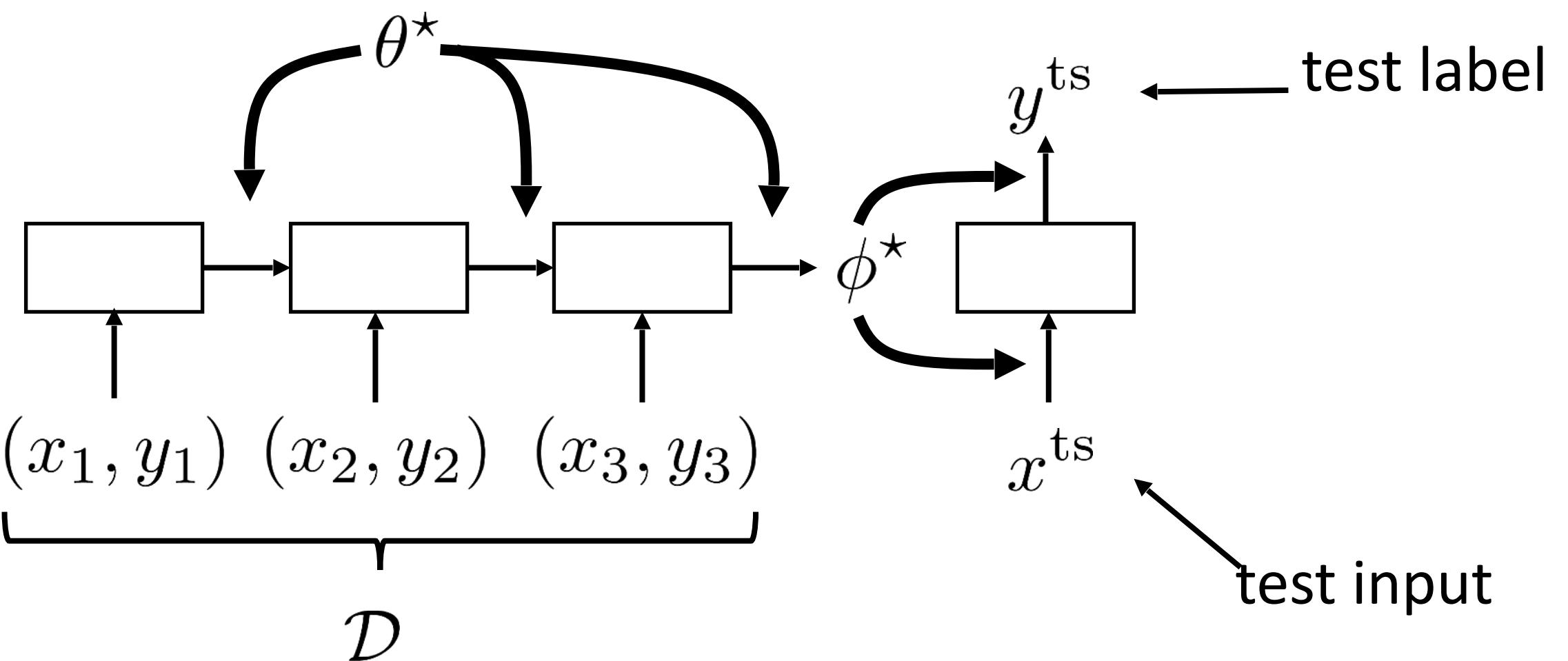
assume $\phi \perp\!\!\!\perp \mathcal{D}_{\text{meta-train}} | \theta$

$$\theta^* = \arg \max_{\theta} \log p(\theta | \mathcal{D}_{\text{meta-train}})$$

A Quick Example

meta-learning: $\theta^* = \arg \max_{\theta} \log p(\theta | \mathcal{D}_{\text{meta-train}})$

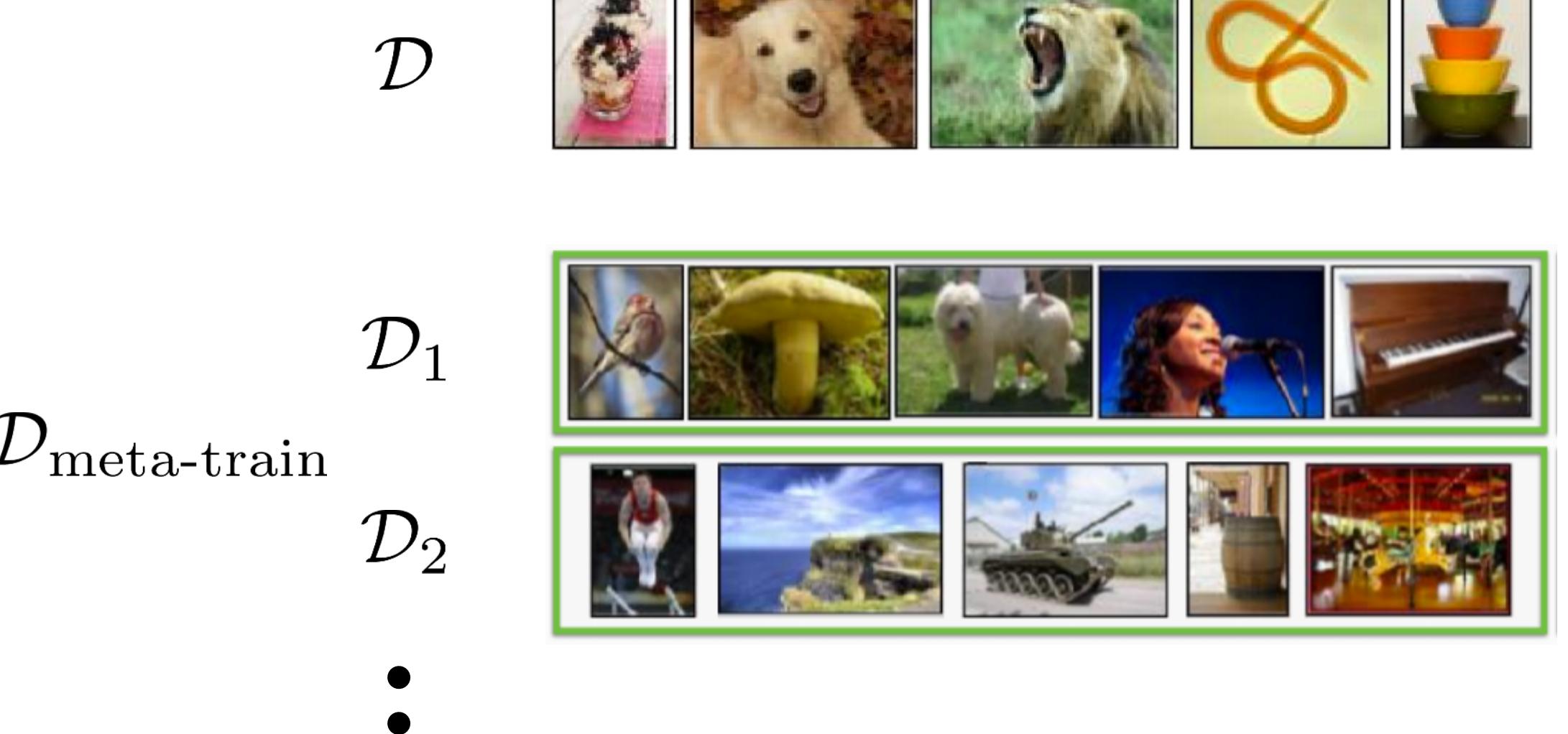
adaptation: $\phi^* = \arg \max_{\phi} \log p(\phi | \mathcal{D}, \theta^*)$



$$\mathcal{D} = \{(x_1, y_1), \dots, (x_k, y_k)\}$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$



How do we train this thing?

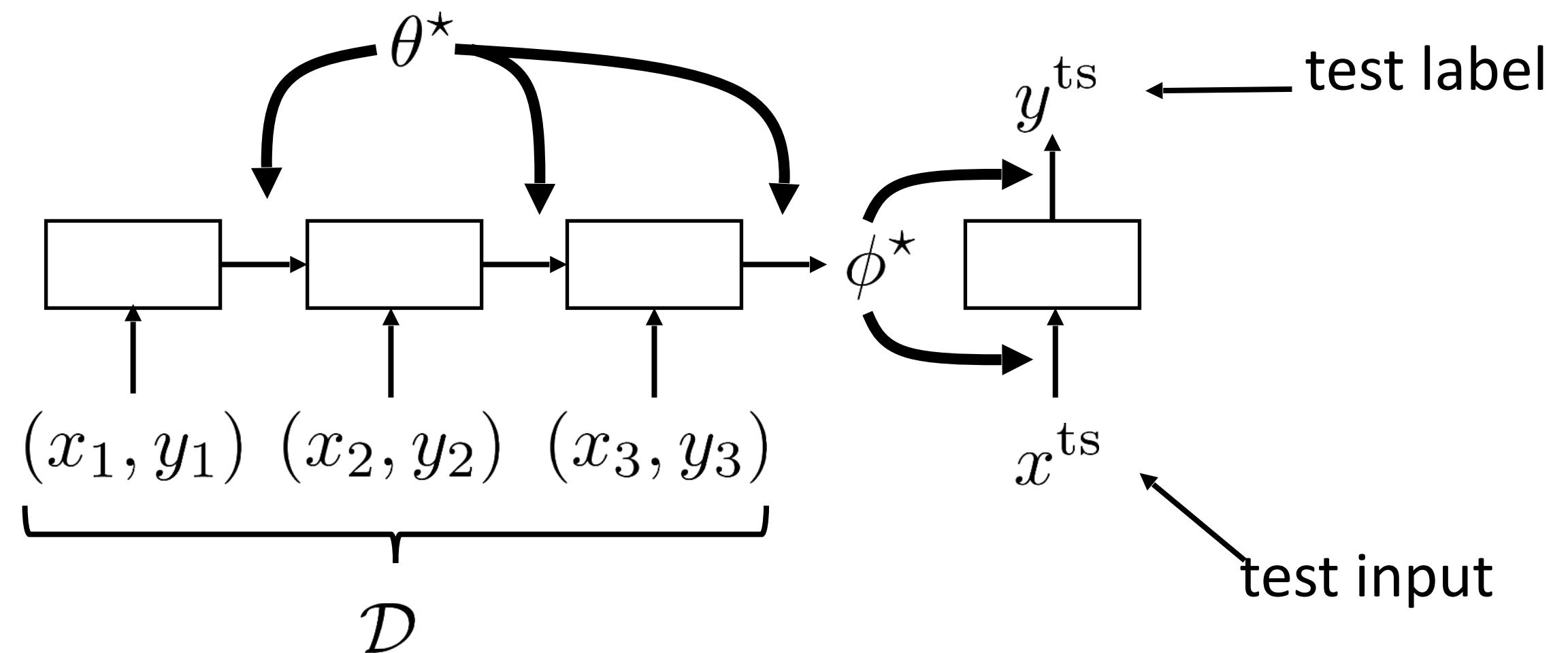
meta-learning: $\theta^* = \arg \max_{\theta} \log p(\theta | \mathcal{D}_{\text{meta-train}})$

adaptation: $\phi^* = \arg \max_{\phi} \log p(\phi | \mathcal{D}, \theta^*)$

$$\mathcal{D} = \{(x_1, y_1), \dots, (x_k, y_k)\}$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$



Key idea:

“our training procedure is based on a simple machine learning principle: test and train conditions must match”

Vinyals et al., Matching Networks for One-Shot Learning

How do we train this thing?

meta-learning: $\theta^* = \arg \max_{\theta} \log p(\theta | \mathcal{D}_{\text{meta-train}})$

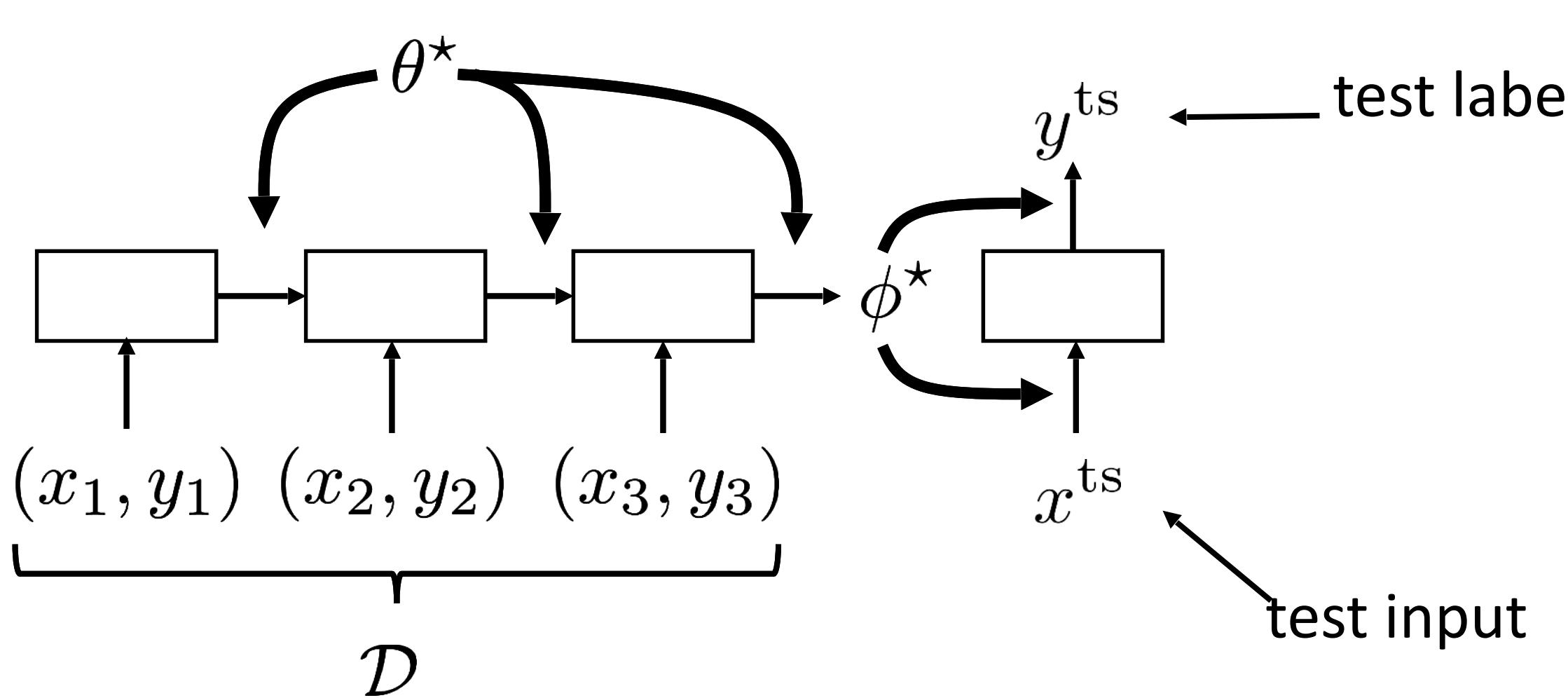
$$\mathcal{D} = \{(x_1, y_1), \dots, (x_k, y_k)\}$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$$

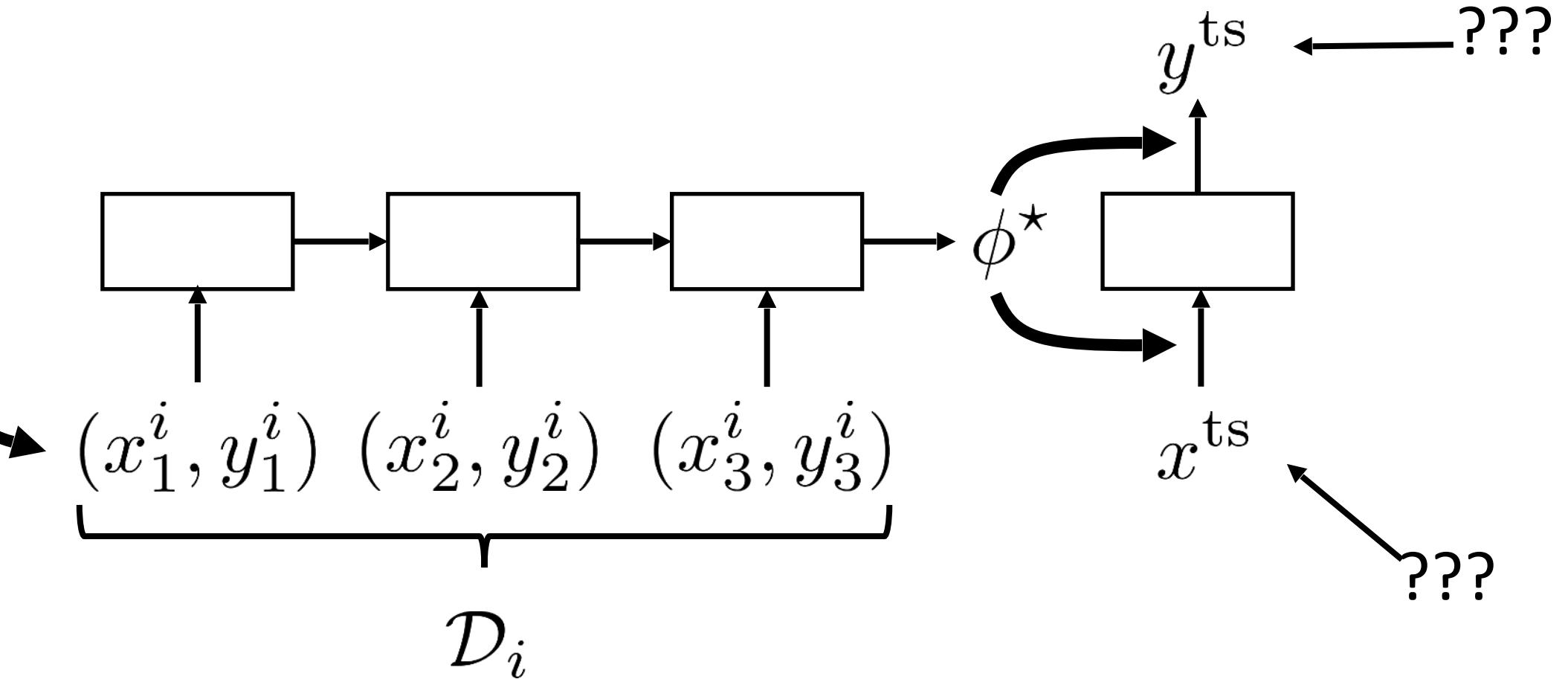
adaptation: $\phi^* = \arg \max_{\phi} \log p(\phi | \mathcal{D}, \theta^*)$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$

(meta) test-time



(meta) training-time

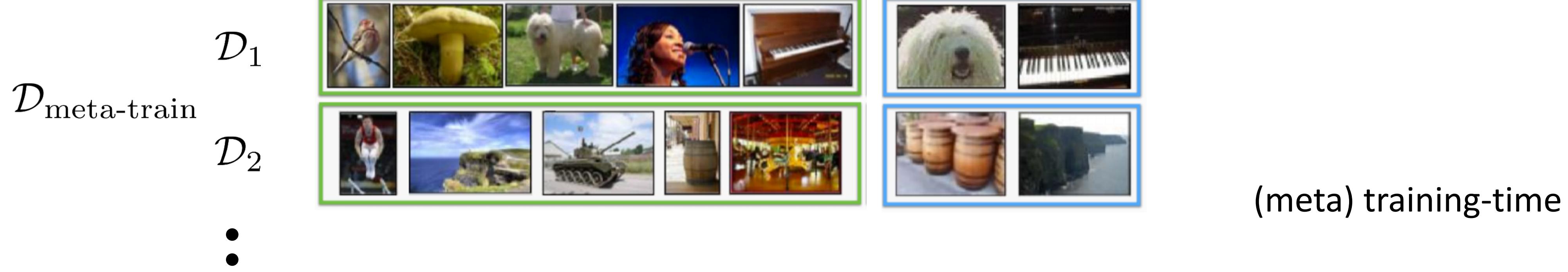


Key idea:

"our training procedure is based on a simple machine learning principle: test and train conditions must match"

Vinyals et al., Matching Networks for One-Shot Learning

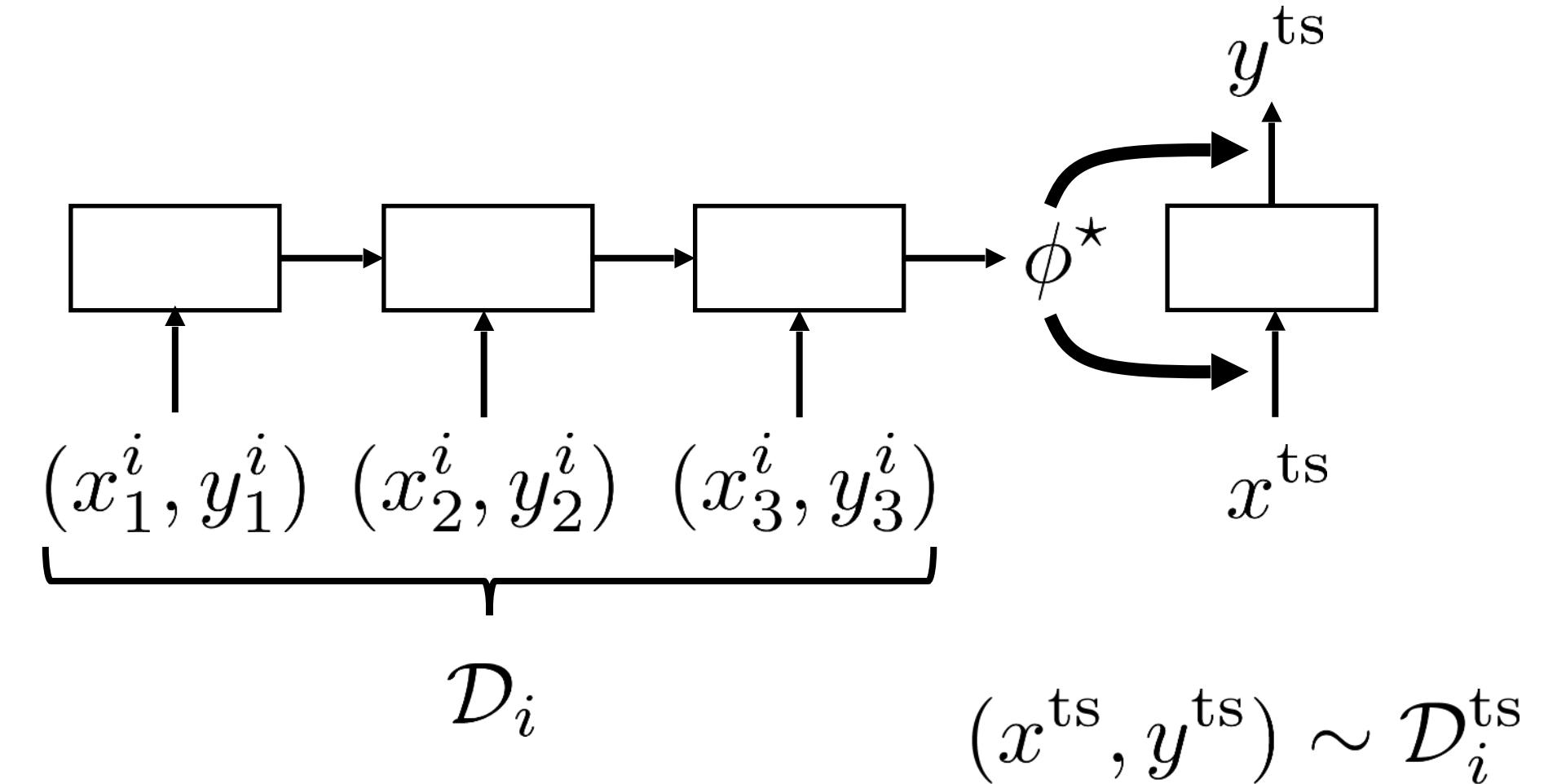
Reserve a test set for each task!



$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \dots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

$$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \dots, (x_l^i, y_l^i)\}$$



Key idea:

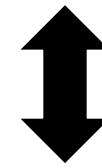
“our training procedure is based on a simple machine learning principle: test and train conditions must match”

Vinyals et al., Matching Networks for One-Shot Learning

The complete meta-learning optimization

meta-learning: $\theta^* = \arg \max_{\theta} \log p(\theta | \mathcal{D}_{\text{meta-train}})$

adaptation: $\phi^* = \arg \max_{\phi} \log p(\phi | \mathcal{D}^{\text{tr}}, \theta^*)$



$$\phi^* = f_{\theta^*}(\mathcal{D}^{\text{tr}})$$

learn θ such that $\phi = f_{\theta}(\mathcal{D}_i^{\text{tr}})$ is good for $\mathcal{D}_i^{\text{ts}}$

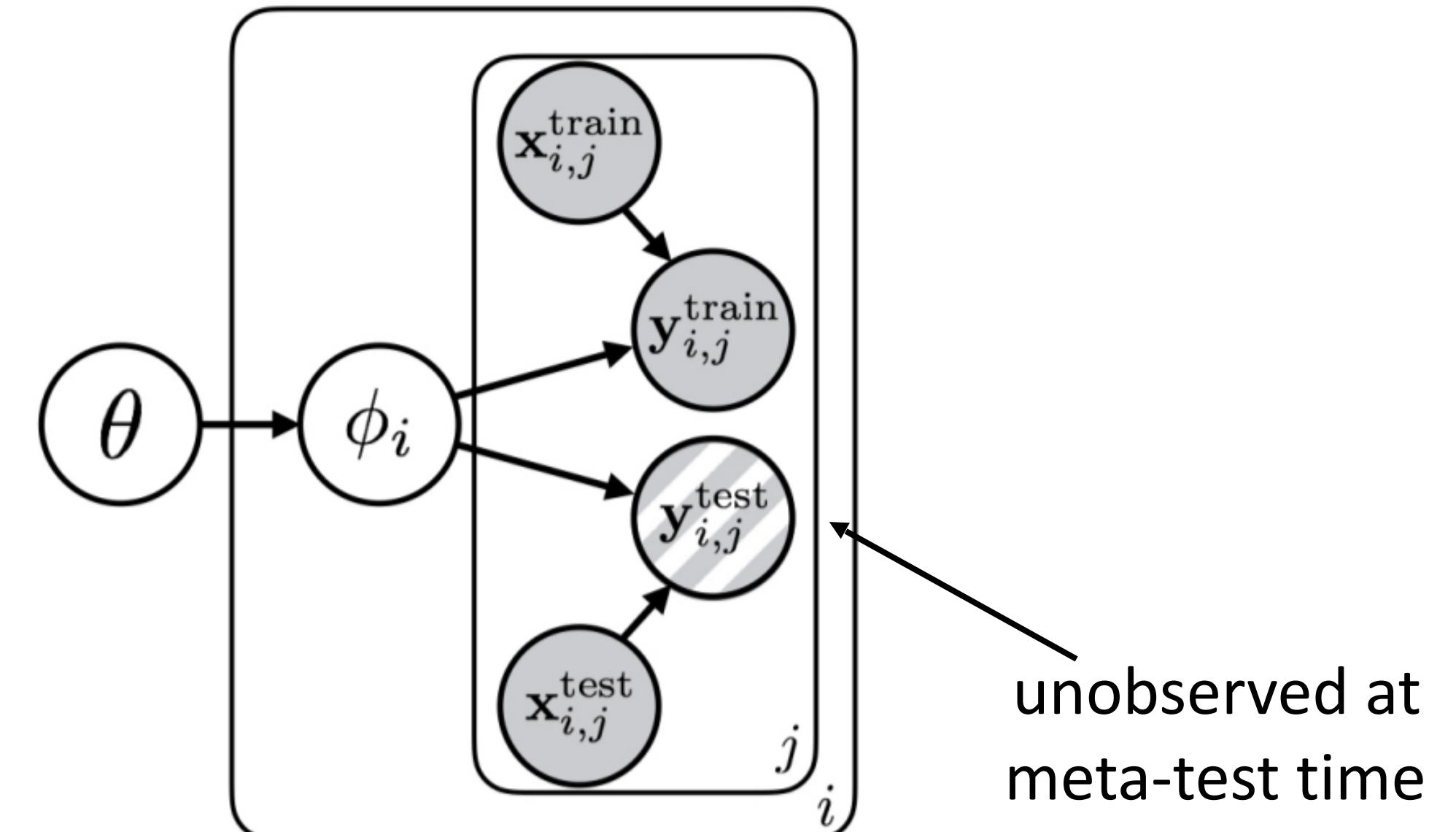
$$\theta^* = \max_{\theta} \sum_{i=1}^n \log p(\phi_i | \mathcal{D}_i^{\text{ts}})$$

where $\phi_i = f_{\theta}(\mathcal{D}_i^{\text{tr}})$

$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \dots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

$$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \dots, (x_l^i, y_l^i)\}$$



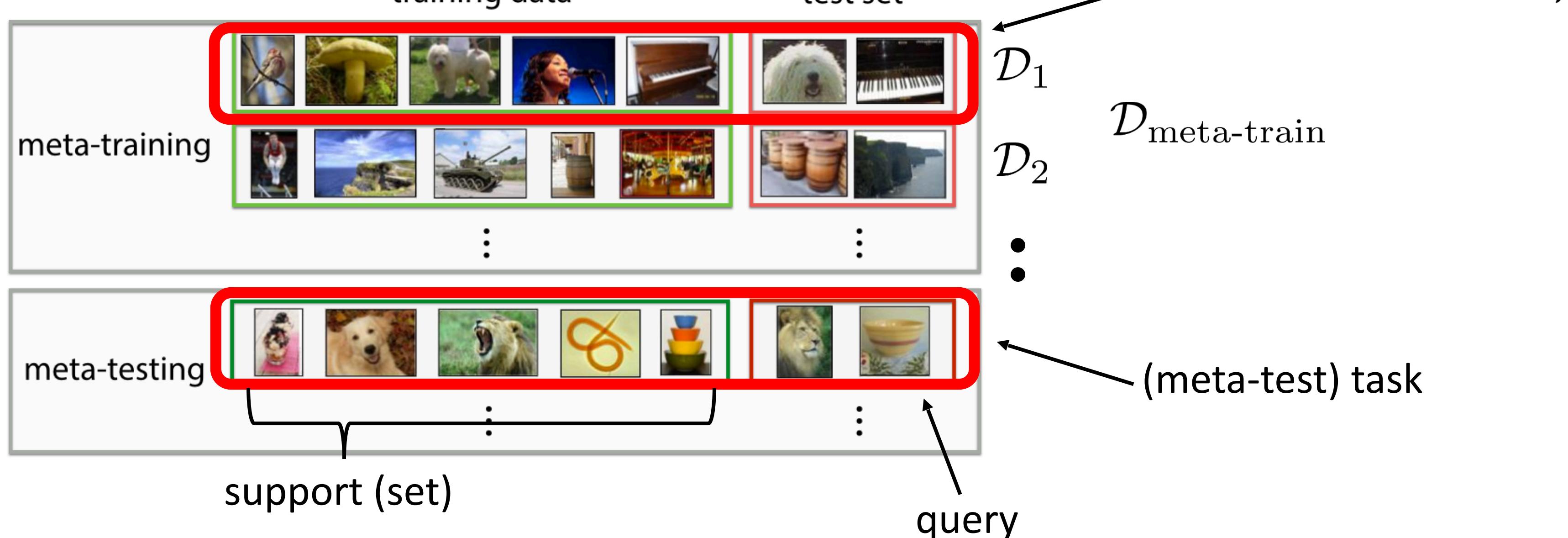
Some meta-learning terminology

learn θ such that $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$ is good for $\mathcal{D}_i^{\text{ts}}$

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n \log p(\phi_i | \mathcal{D}_i^{\text{ts}})$$

where $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$

$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \dots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$



Closely related problem settings

meta-learning:

$$\theta^* = \max_{\theta} \sum_{i=1}^n \log p(\phi_i | \mathcal{D}_i^{\text{ts}})$$

where $\phi_i = f_{\theta}(\mathcal{D}_i^{\text{tr}})$

$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \dots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

$$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \dots, (x_l^i, y_l^i)\}$$

multi-task learning: learn model with parameters θ^* that solves multiple tasks $\theta^* = \arg \max_{\theta} \sum_{i=1}^n \log p(\theta | \mathcal{D}_i)$
can be seen as special case where $\phi_i = \theta$ (i.e., $f_{\theta}(\mathcal{D}_i) = \theta$)

hyperparameter optimization & auto-ML: can be cast as meta-learning

hyperparameter optimization: θ = hyperparameters, ϕ = network weights

architecture search: θ = architecture, ϕ = network weights

very active area of research! but outside the scope of this tutorial

Qs: slido.com/meta

Outline

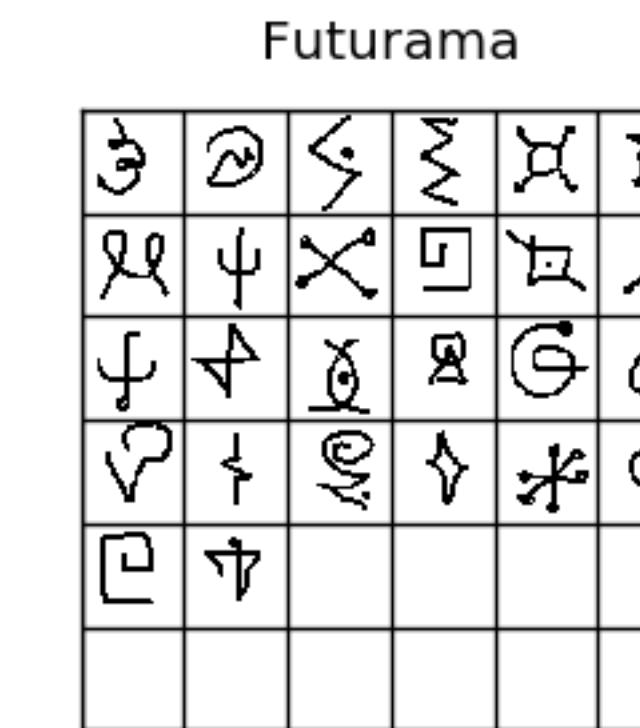
- Problem statement
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

General recipe

How to *evaluate* a meta-learning algorithm

the Omniglot dataset Lake et al. Science 2015

1623 characters from 50 different alphabets



...

many classes, few examples

the “transpose” of MNIST

statistics more reflective
of the real world

20 instances of each character

Proposes both **few-shot discriminative & few-shot generative** problems

Initial few-shot learning approaches w/ Bayesian models, non-parametrics

Fei-Fei et al. '03 Lake et al. '11 Salakhutdinov et al. '12 Lake et al. '13

Other datasets used for few-shot image recognition: Minilmagenet, CIFAR, CUB, CelebA, others

General recipe

How to *evaluate* a meta-learning algorithm

5-way, 1-shot image classification (Minilmagenet)

Given 1 example of 5 classes:



Classify new examples



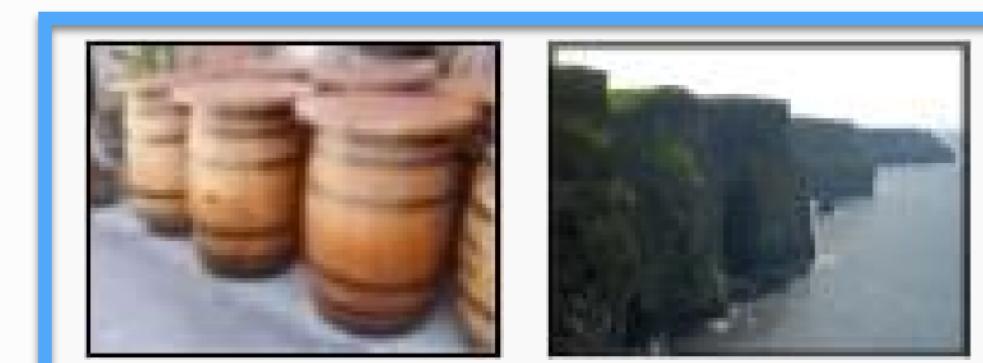
held-out classes

\mathcal{T}_1



meta-training

\mathcal{T}_2



training classes

:

:

any ML
problem

Can replace image classification with: regression, language generation, skill learning,
Q.

General recipe

How to *design* a meta-learning algorithm

1. Choose a form of $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$
2. Choose how to optimize θ w.r.t. max-likelihood objective using $\mathcal{D}_{\text{meta-train}}$

Can we treat $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$ as an **inference** problem?

Neural networks are good at inference.

Outline

- Problem statement
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

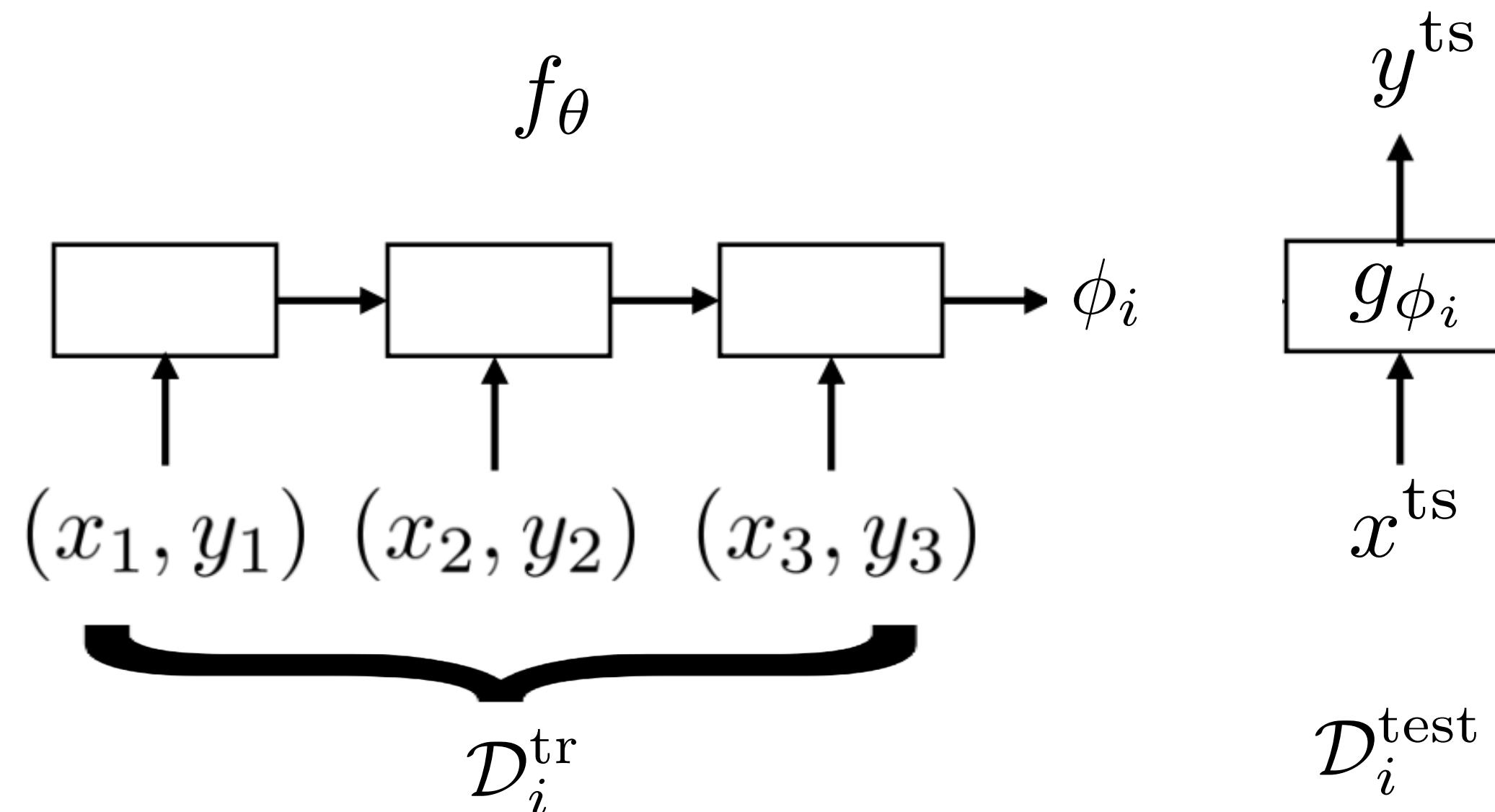
Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$

For now: Use **deterministic** (point estimate) $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$



(Bayes will come back later)



Train with standard supervised learning!

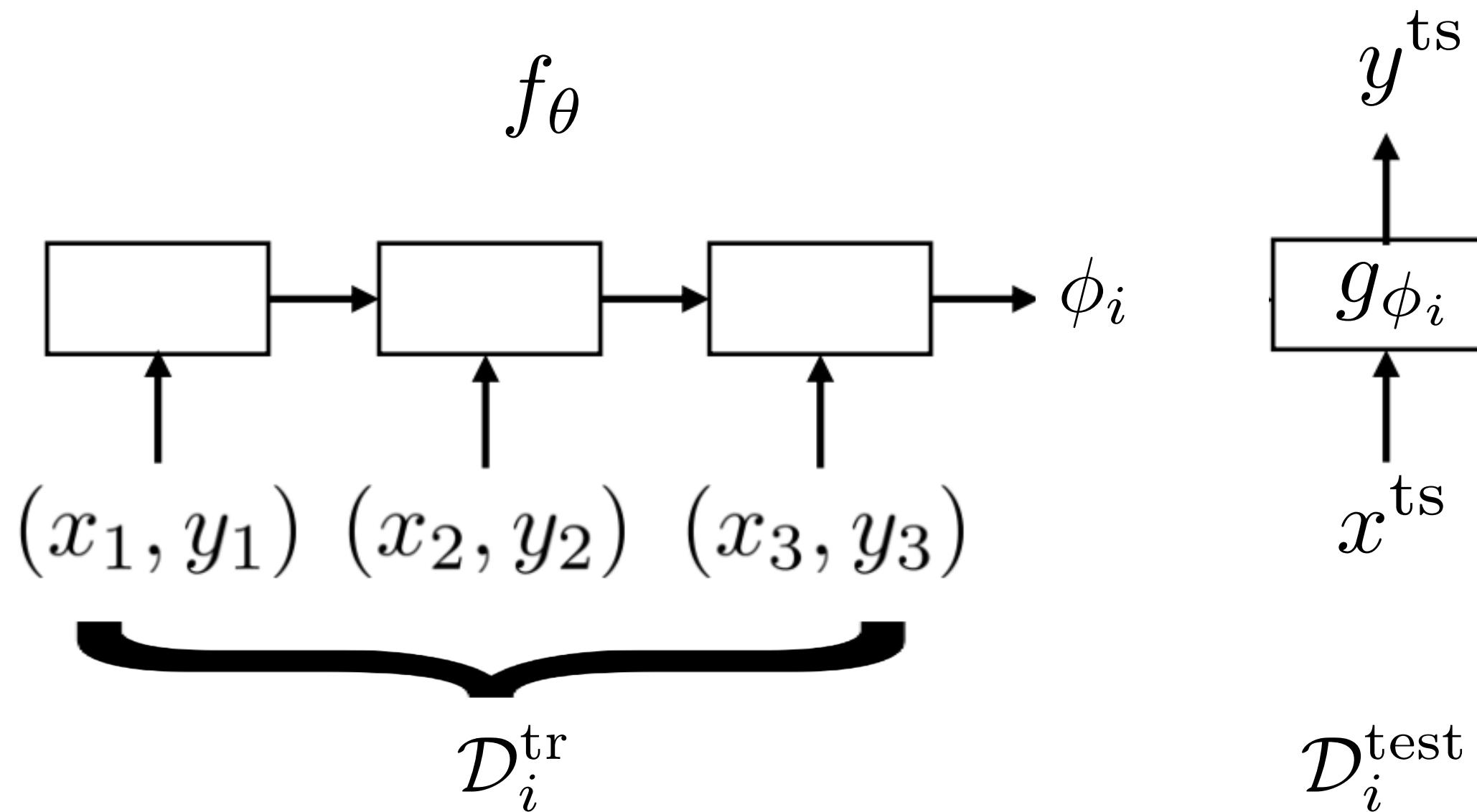
$$\max_{\theta} \sum_{\mathcal{T}_i} \sum_{(x,y) \sim \mathcal{D}_i^{\text{test}}} \log g_{\phi_i}(y|x)$$

$$\mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$$

$$\max_{\theta} \sum_{\mathcal{T}_i} \mathcal{L}(f_\theta(\mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{test}})$$

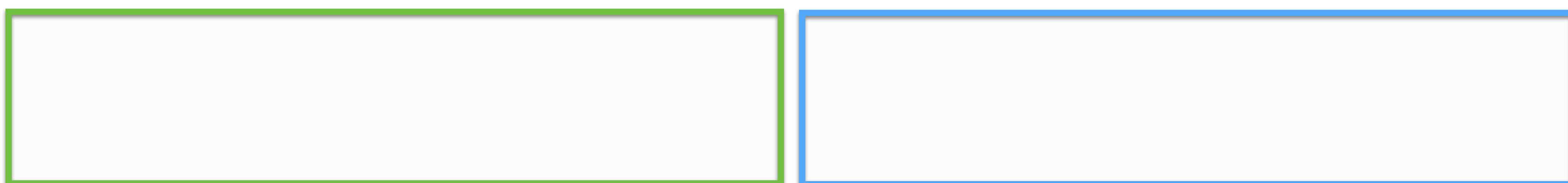
Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$



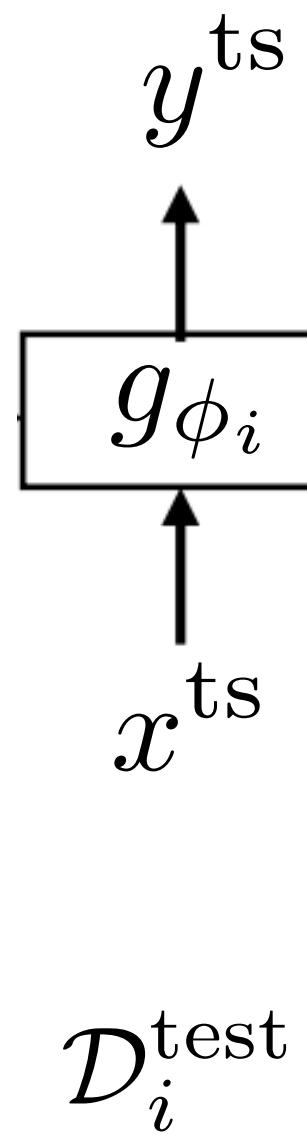
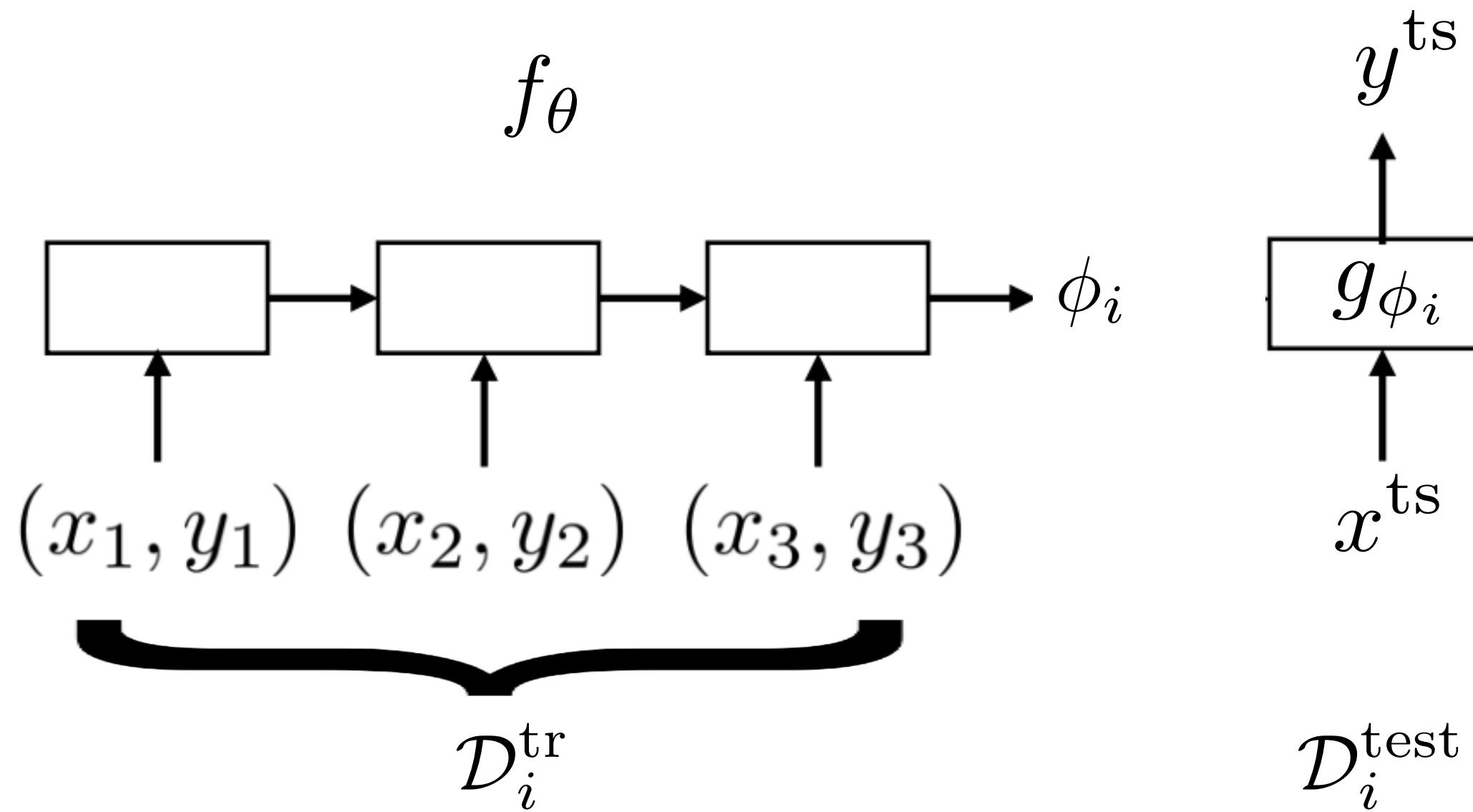
1. Sample task \mathcal{T}_i (or mini batch of tasks)
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from \mathcal{D}_i

\mathcal{D}_i



Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$

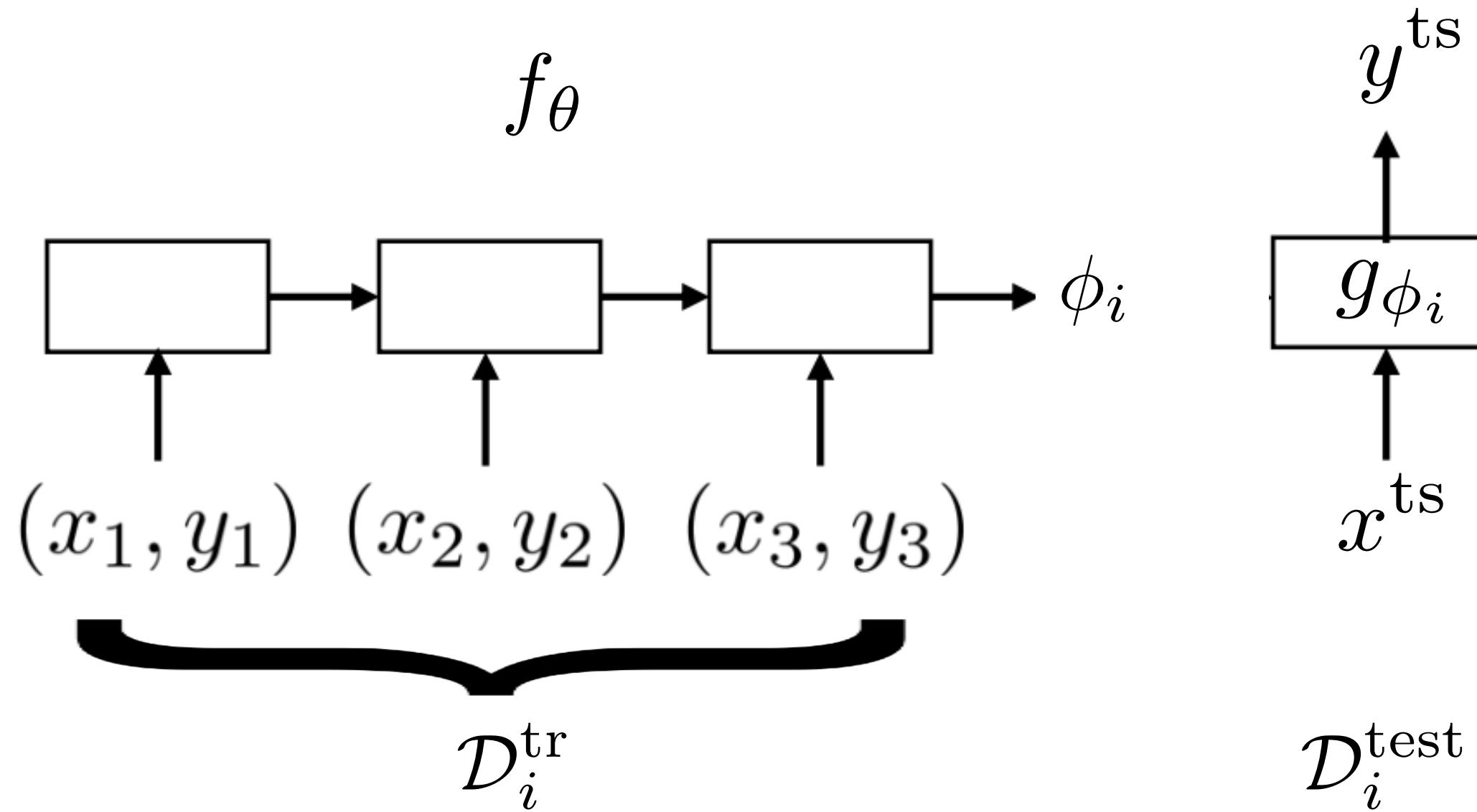


1. Sample task \mathcal{T}_i (or mini batch of tasks)
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from \mathcal{D}_i
3. Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$



Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$



Form of f_θ ?

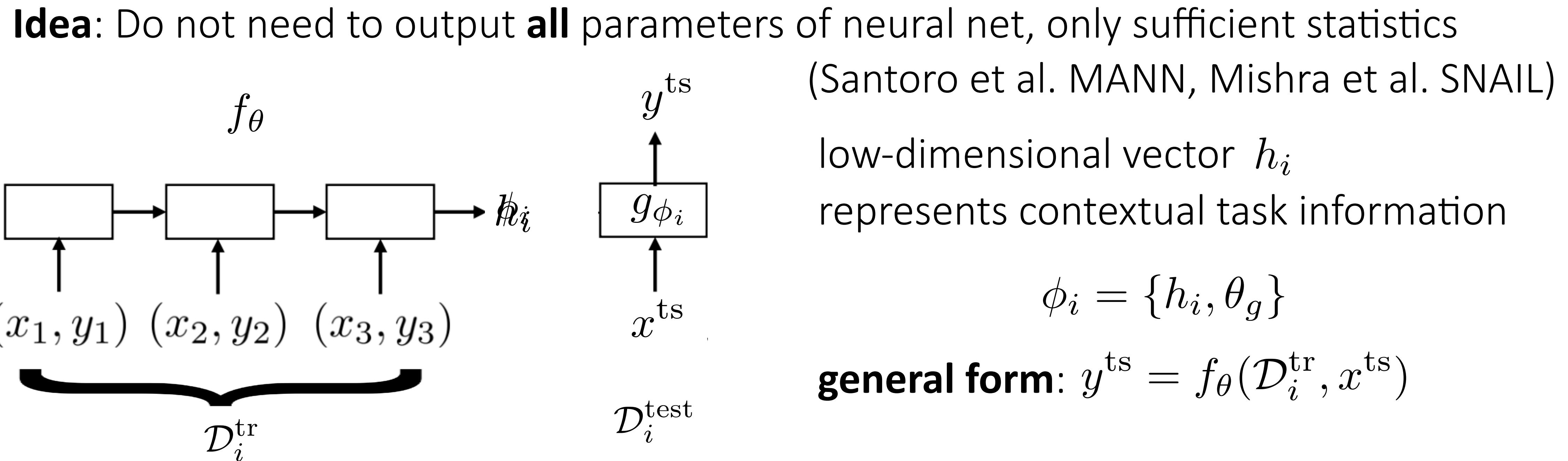
- LSTM
- Neural turing machine (NTM)
- Self-attention
- 1D convolutions
- feedforward + average

Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$

Challenges

Outputting all neural net parameters does not seem **scalable?**



Is there a way to infer **all parameters** in a scalable way?

Qs: slido.com/meta What if we treat it as an **optimization** procedure?

Outline

- Problem statement
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

Optimization-Based Inference

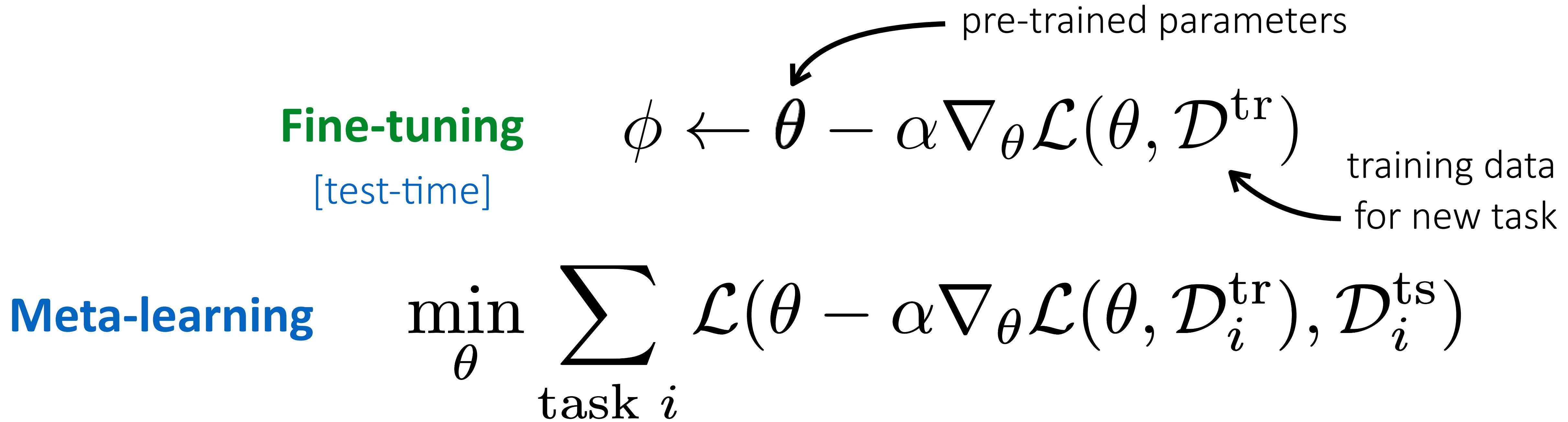
Key idea: Acquire ϕ_i through optimization.

$$\max_{\phi_i} \log p(\mathcal{D}_i^{\text{tr}} | \phi_i) + \log p(\phi_i | \theta)$$

Meta-parameters θ serve as a prior. What form of prior?

One successful form of prior knowledge: **initialization** for **fine-tuning**

Optimization-Based Inference



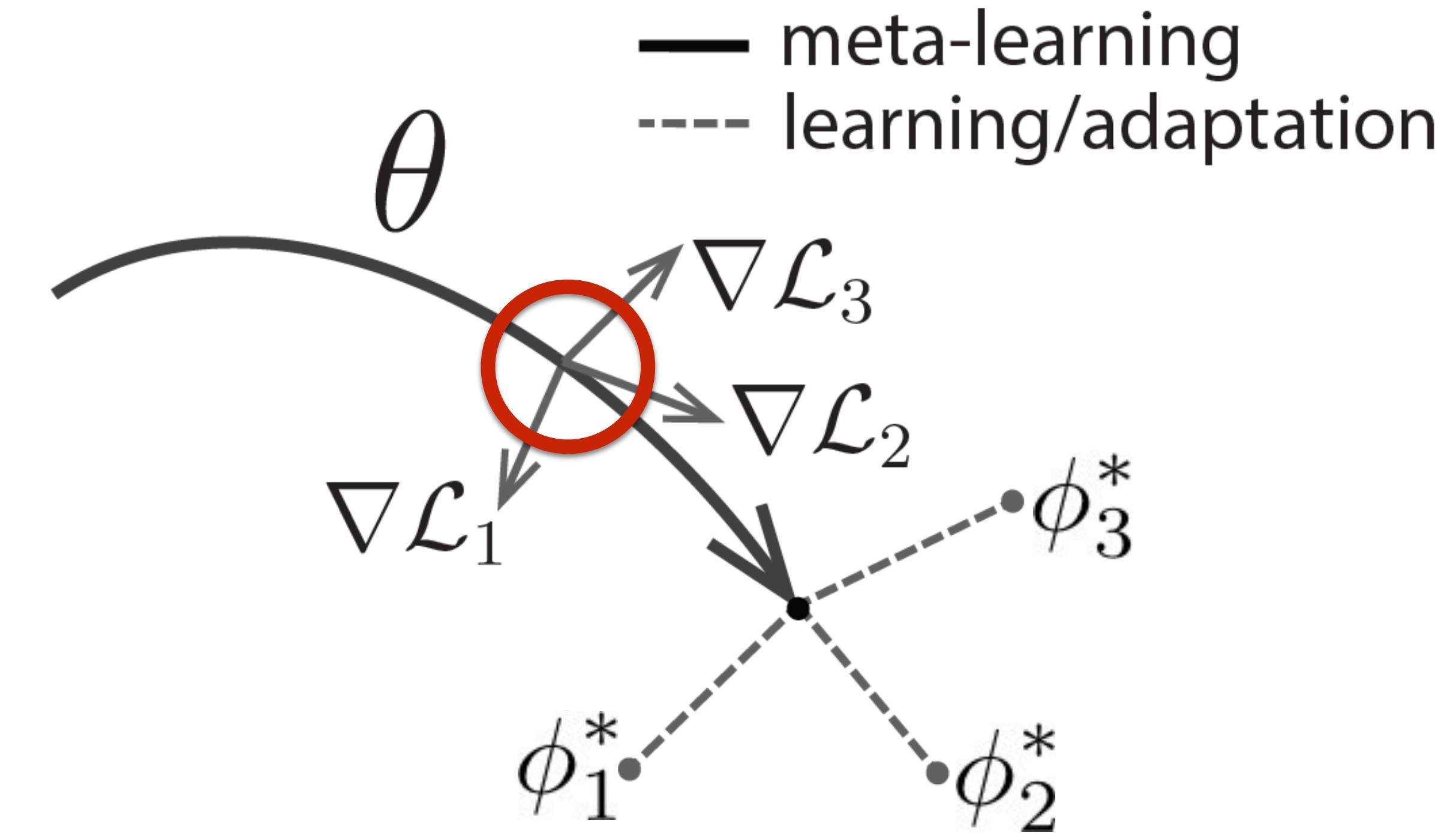
Key idea: Over many tasks, learn parameter vector θ that transfers via fine-tuning

Optimization-Based Inference

$$\min_{\theta} \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

θ parameter vector
being meta-learned

ϕ_i^* optimal parameter
vector for task i



Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

General Algorithm:

~~Amortized approach~~ Optimization-based approach

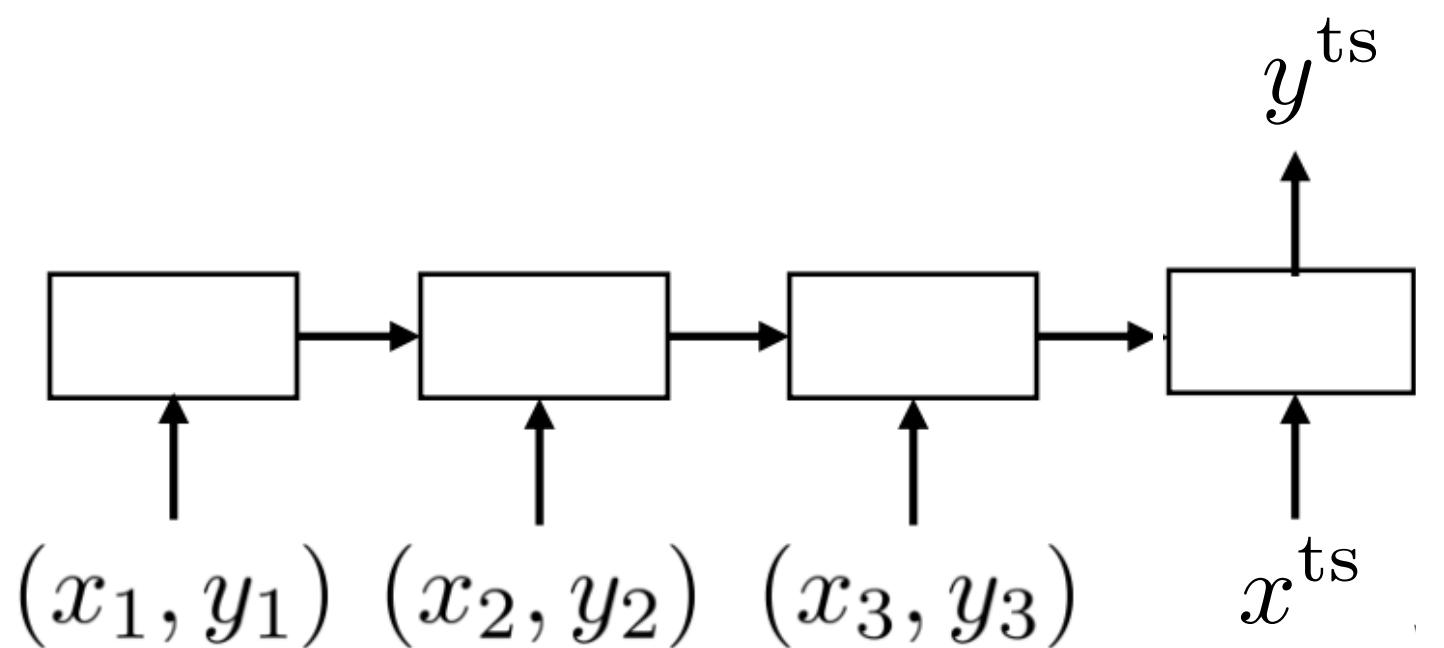
1. Sample task \mathcal{T}_i (*or mini batch of tasks*)
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from \mathcal{D}_i
3. ~~Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\text{tr}})$~~ Optimize $\phi_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$

→ brings up **second-order** derivatives (more on this later)

Optimization vs. Black-Box Adaptation

Black-box adaptation

general form: $y^{\text{ts}} = f_{\theta}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$



Model-agnostic meta-learning

$$\begin{aligned} y^{\text{ts}} &= f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) \\ &= f_{\phi_i}(x^{\text{ts}}) \end{aligned}$$

$$\text{where } \phi_i = \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$$

MAML can be viewed as **computation graph**,
with embedded gradient operator

Note: Can mix & match components of computation graph

Learn initialization but replace gradient update with learned network

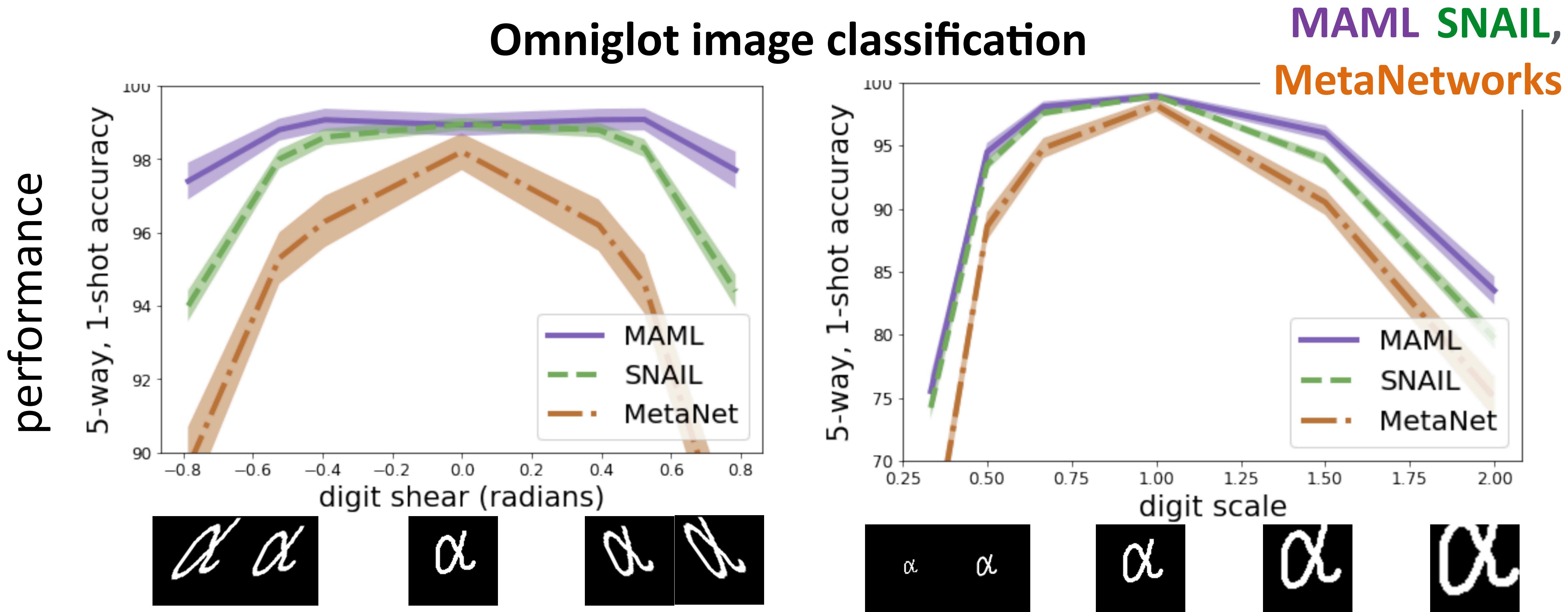
$$\begin{aligned} \text{where } \phi_i &= \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}}) \\ &f(\theta, \mathcal{D}_i^{\text{tr}}, \nabla_{\theta} \mathcal{L}) \end{aligned}$$

Ravi & Larochelle ICLR '17
(actually precedes MAML)

Qs: slido.com This **computation graph view** of meta-learning will come back again!

Optimization vs. Black-Box Adaptation

How well can learning procedures generalize to similar, but extrapolated tasks?



Qs: slido.com/meta

Does this structure come at a cost?

Finn & Levine ICLR '18

Black-box adaptation

$$y^{\text{ts}} = f_{\theta}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$

Optimization-based (MAML)

$$y^{\text{ts}} = f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$

Does this structure come at a cost?

For a sufficiently deep f ,

MAML function can approximate any function of $\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}$

Finn & Levine, ICLR 2018

Assumptions:

- nonzero α
- loss function gradient does not lose information about the label
- datapoints in $\mathcal{D}_i^{\text{tr}}$ are unique

Why is this interesting?

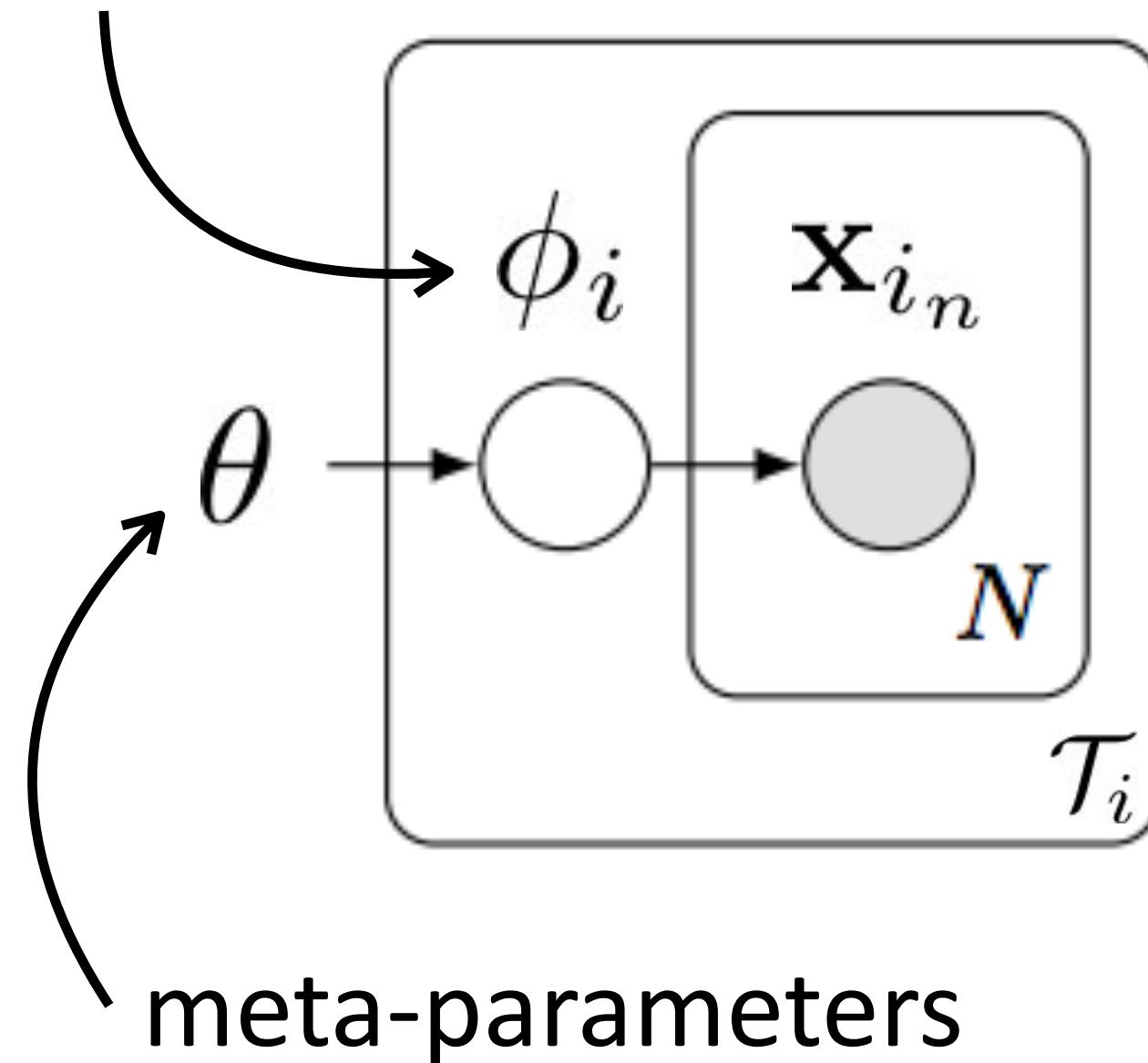
MAML has benefit of inductive bias without losing expressive power.

Probabilistic Interpretation of Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Meta-parameters θ serve as a prior. One form of prior knowledge: **initialization** for **fine-tuning**

task-specific parameters



$$\begin{aligned} & \max_{\theta} \log \prod p(\mathcal{D}_i | \theta) \\ &= \log \prod_i \int p(\mathcal{D}_i | \phi_i) p(\phi_i | \theta) d\phi_i \quad (\text{empirical Bayes}) \\ &\approx \log \prod_i p(\mathcal{D}_i | \hat{\phi}_i) p(\hat{\phi}_i | \theta) \end{aligned}$$

MAP estimate

How to compute MAP estimate?

Gradient descent with early stopping = MAP inference under **Gaussian prior** with mean at initial parameters [Santos '96]

(exact in linear case, approximate in nonlinear case)

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Meta-parameters θ serve as a prior. One form of prior knowledge: **initialization** for **fine-tuning**

Gradient-descent + early stopping (MAML): implicit Gaussian prior $\phi \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$

Other forms of priors?

Gradient-descent with explicit Gaussian prior $\phi \leftarrow \min_{\phi'} \mathcal{L}(\phi', \mathcal{D}^{\text{tr}}) + \frac{\lambda}{2} \|\theta - \phi'\|^2$

Rajeswaran et al. implicit MAML '19

Bayesian linear regression on learned features Harrison et al. ALPaCA '18

Closed-form or **convex optimization** on learned features
ridge regression, logistic regression **support vector machine**

Bertinetto et al. R2-D2 '19

Lee et al. MetaOptNet '19

Qs: slido.com/meta Current **SOTA** on few-shot image classification

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Challenges

How to choose architecture that is effective for inner gradient-step?

Idea: Progressive neural architecture search + MAML

(Kim et al. Auto-Meta)

- finds highly non-standard architecture (deep & narrow)
- different from architectures that work well for standard supervised learning

Minilmagenet, 5-way 5-shot MAML, basic architecture: 63.11%

MAML + AutoMeta: **74.65%**

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Challenges

Second-order meta-optimization can exhibit instabilities.

Idea: [Crudely] approximate $\frac{d\phi_i}{d\theta}$ as identity
(Finn et al. first-order MAML, Nichol et al. Reptile)

Idea: Automatically learn inner vector learning rate, tune outer learning rate
(Li et al. Meta-SGD, Behl et al. AlphaMAML)

Idea: Optimize only a subset of the parameters in the inner loop
(Zhou et al. DEML, Zintgraf et al. CAVIA)

Idea: Decouple inner learning rate, BN statistics per-step (Antoniou et al. MAML++)

Idea: Introduce context variables for increased expressive power.
(Finn et al. bias transformation, Zintgraf et al. CAVIA)

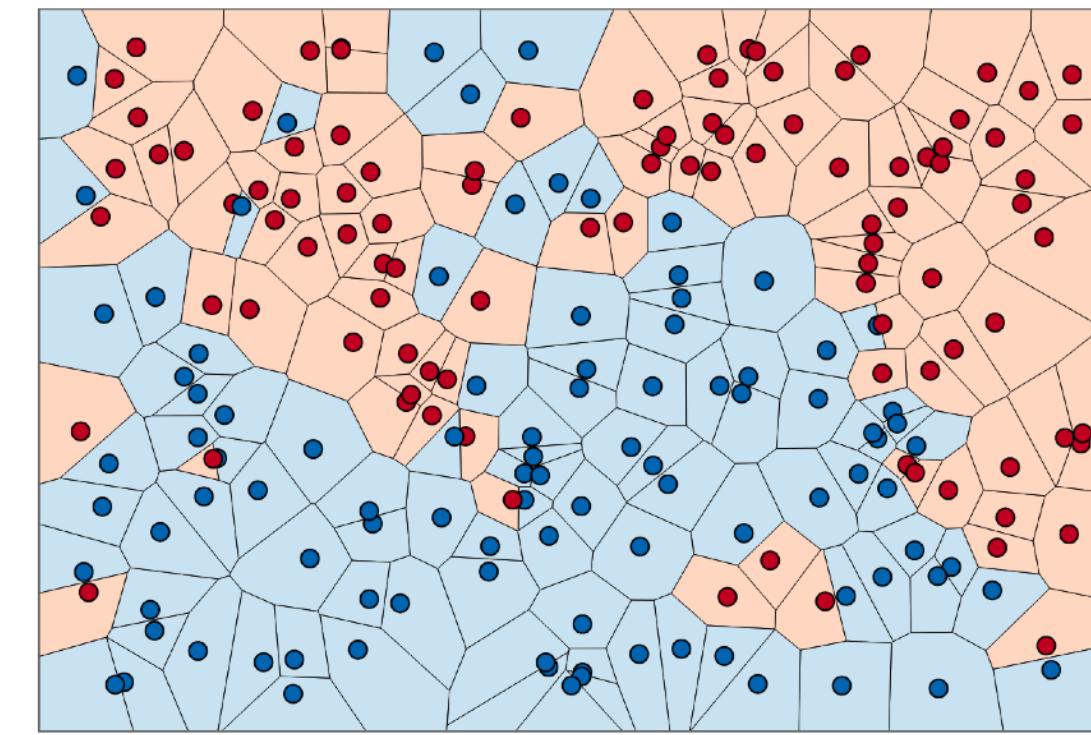
Qs: slido.com/presentation/optimization-based-inference **Takeaway:** a range of simple tricks that can help optimization significantly

Outline

- Problem statement
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
- Non-parametric methods
- Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-reinforcement learning
- Challenges & frontiers

So far: Learning parametric models.

In low data regimes, **non-parametric** methods are simple, work well.



During **meta-test time**: few-shot learning \leftrightarrow low data regime

During **meta-training**: still want to be parametric

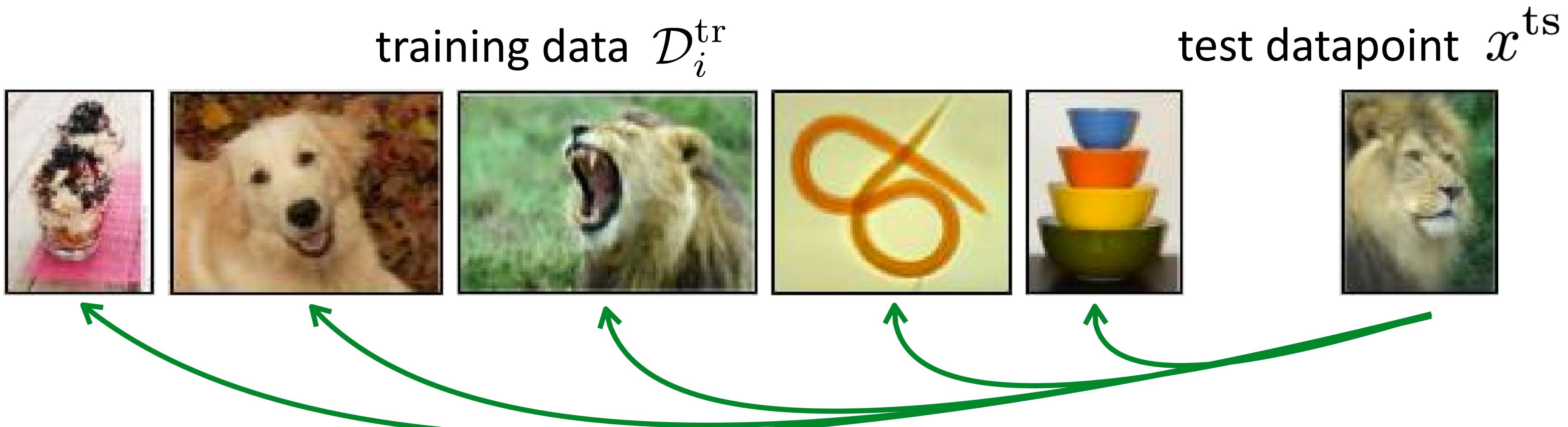
Can we use **parametric meta-learners** that produce effective **non-parametric learners**?

Note: some of these methods precede parametric approaches

Qs: slido.com/meta

Non-parametric methods

Key Idea: Use non-parametric learner.



Compare test image with training images

In what space do you compare? With what distance metric?

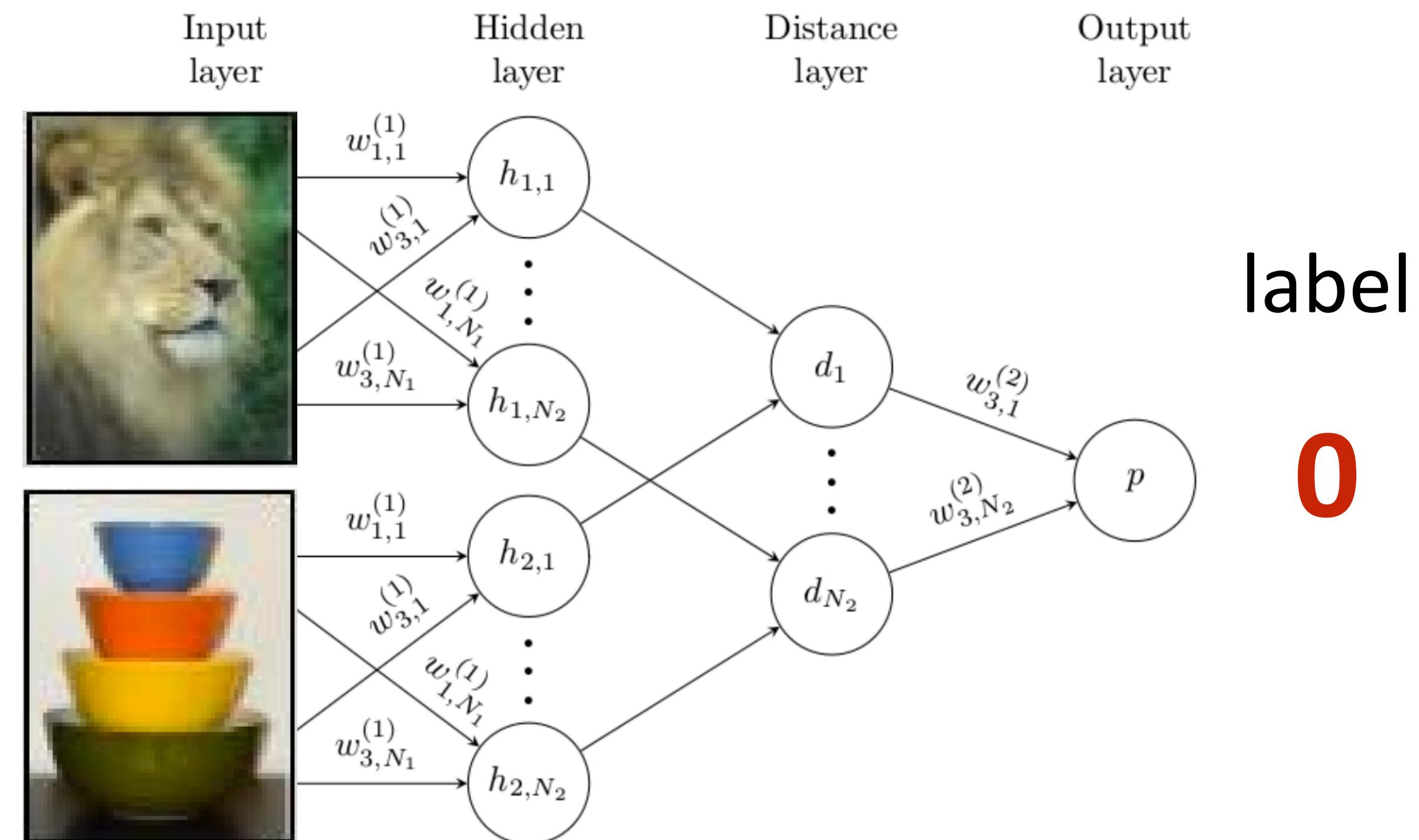
~~pixel space, ℓ_2 -distance?~~

Learn to compare using data!

Non-parametric methods

Key Idea: Use non-parametric learner.

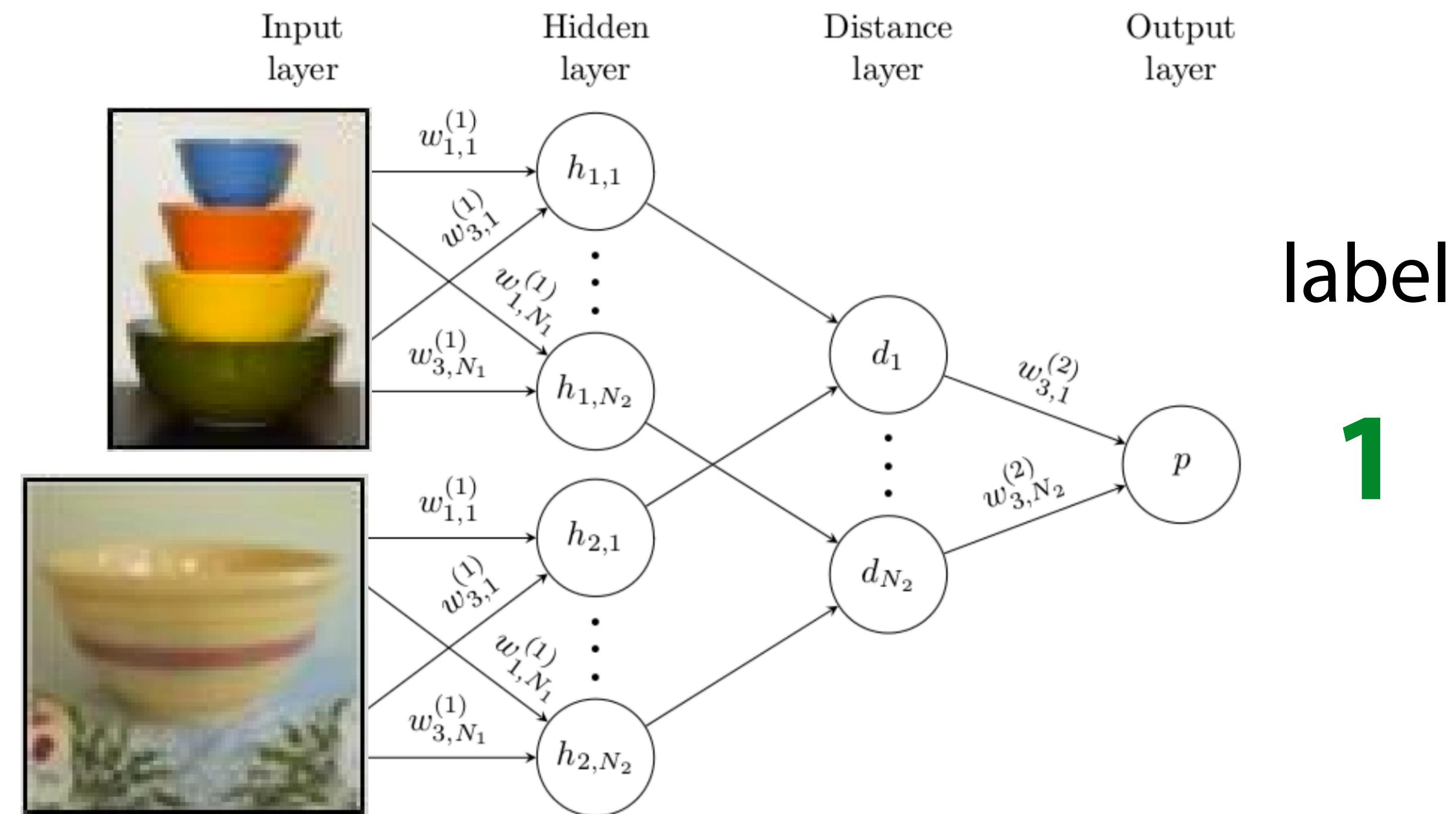
train Siamese network to predict whether or not two images are the same class



Non-parametric methods

Key Idea: Use non-parametric learner.

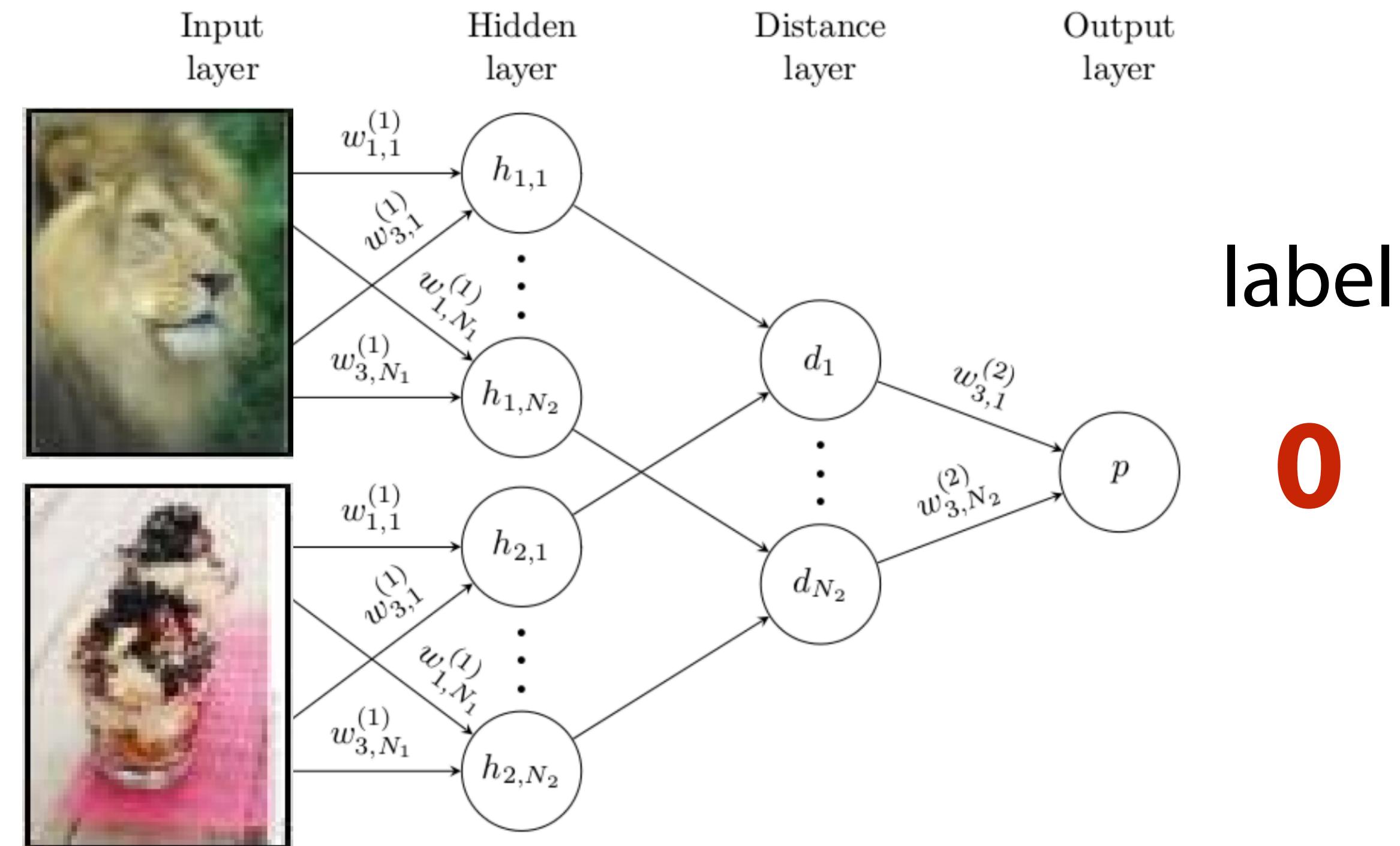
train Siamese network to predict whether or not two images are the same class



Non-parametric methods

Key Idea: Use non-parametric learner.

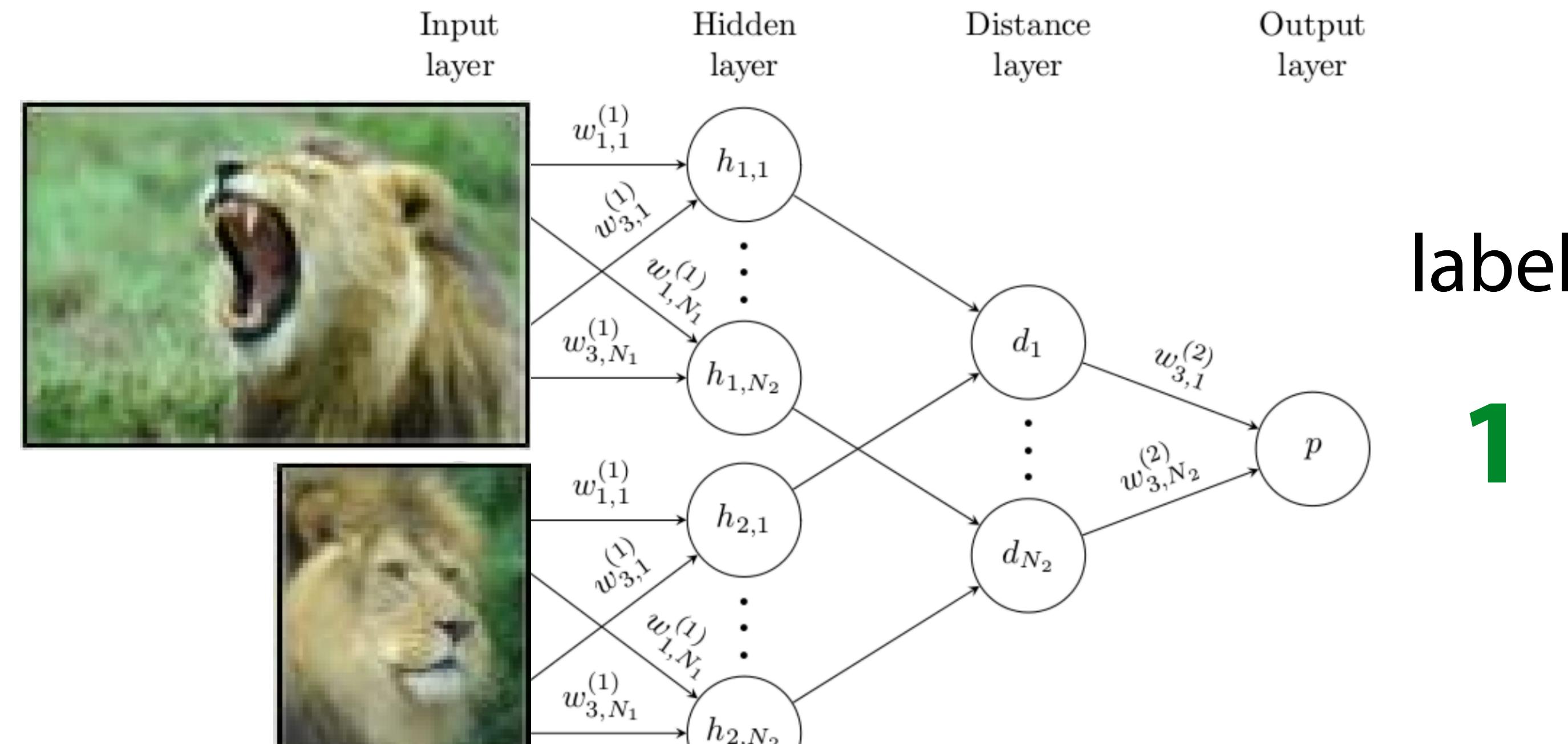
train Siamese network to predict whether or not two images are the same class



Non-parametric methods

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class



Meta-test time: compare image \mathbf{x}_{test} to each image in $\mathcal{D}_j^{\text{tr}}$

Meta-training: 2-way classification

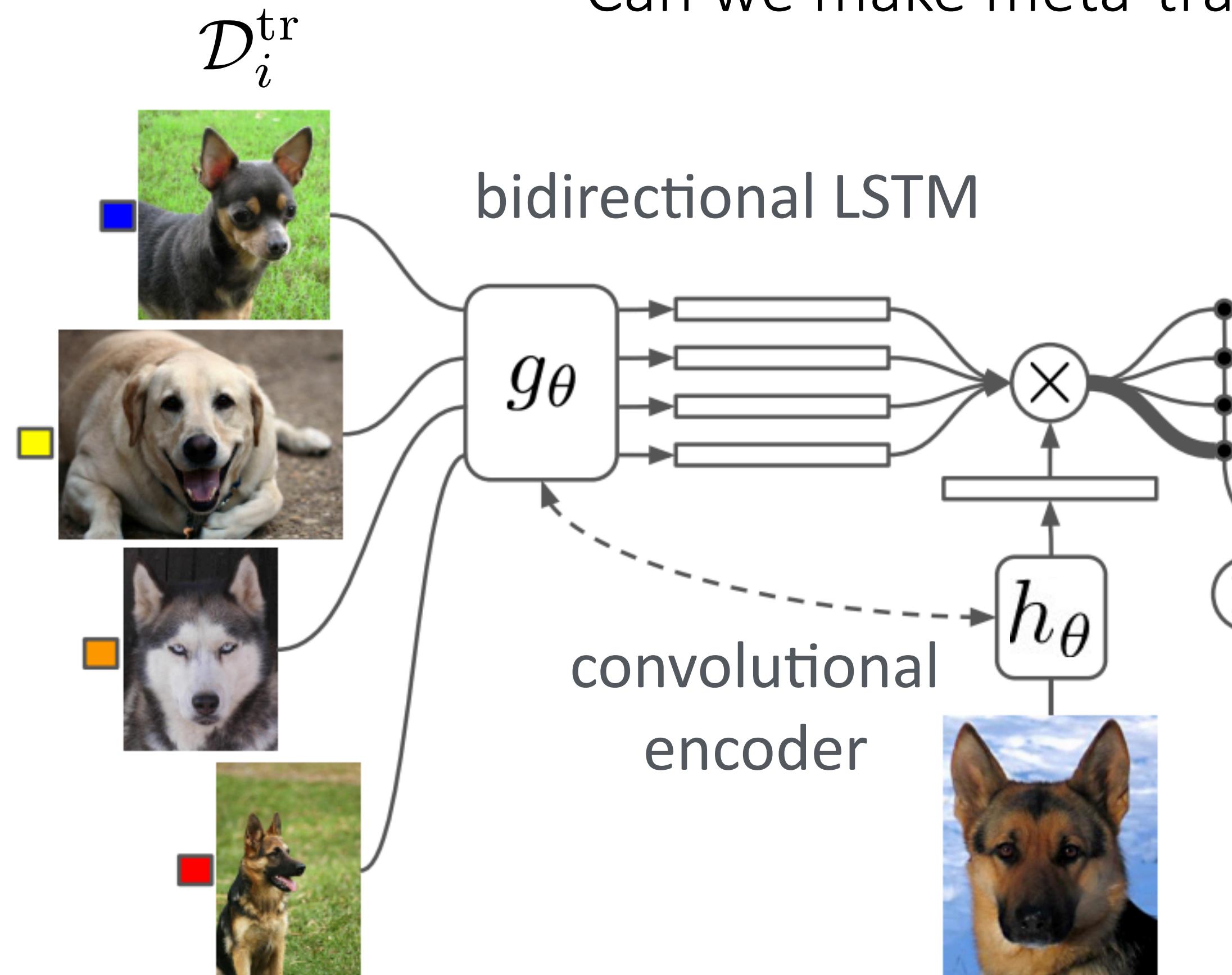
Can we **match** meta-train & meta-test?

Qs: slido.co ... , ...

Non-parametric methods

Key Idea: Use non-parametric learner.

Can we make meta-train & meta-test match?



Weighed nearest neighbors in learned embedding space

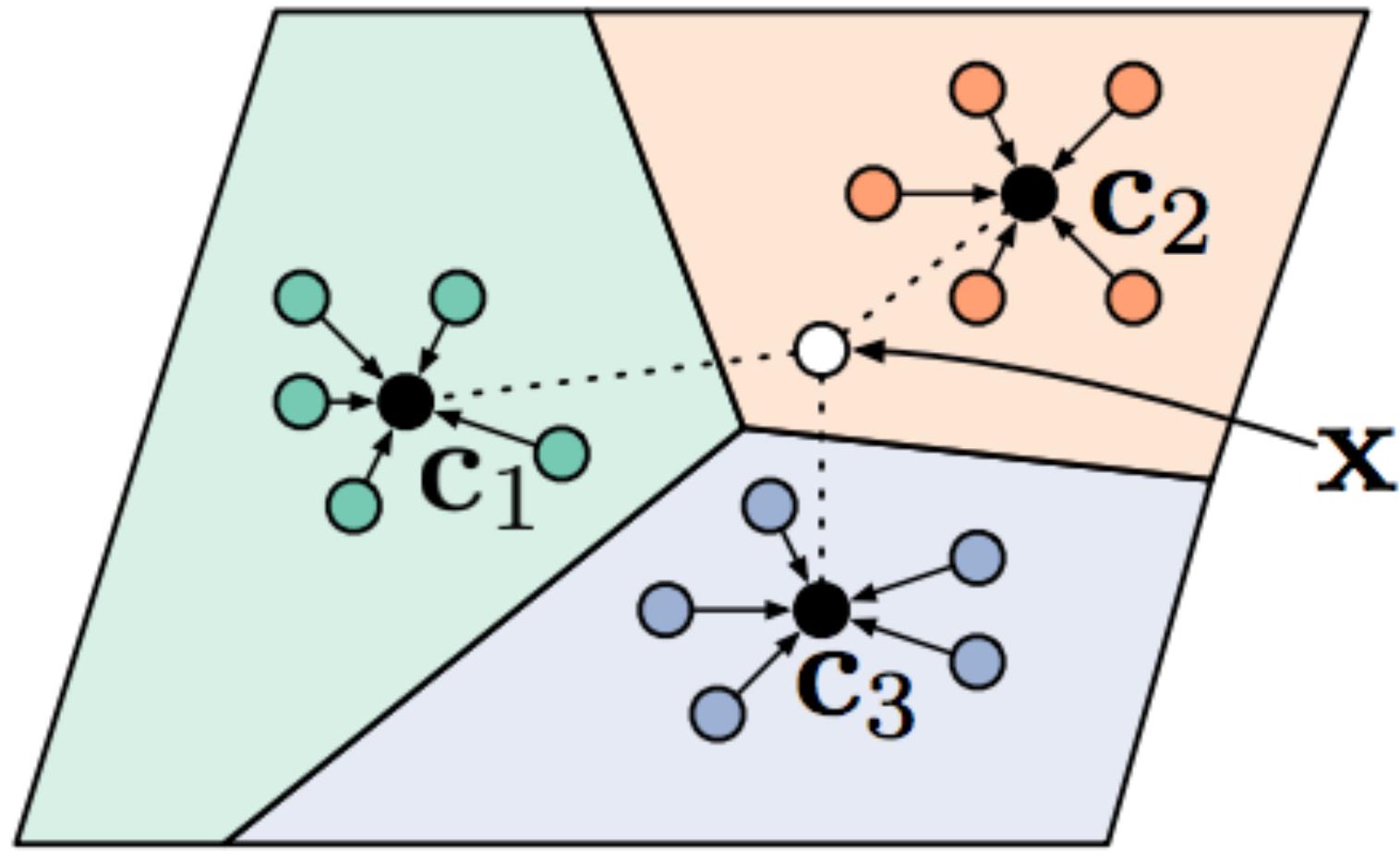
$$\hat{y} = \sum_{i=1}^k a(\hat{x}, x_i) y_i$$

What if >1 shot?

Can we aggregate class information to create a prototypical embedding?

Non-parametric methods

Key Idea: Use non-parametric learner.



(a) Few-shot

$$\mathbf{c}_k = \frac{1}{|\mathcal{D}_i^{\text{tr}}|} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} f_\theta(x)$$
$$p_\theta(y = k|x) = \frac{\exp(-d(f_\theta(x), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\theta(x), \mathbf{c}_{k'})))}$$

d: Euclidean, or cosine distance

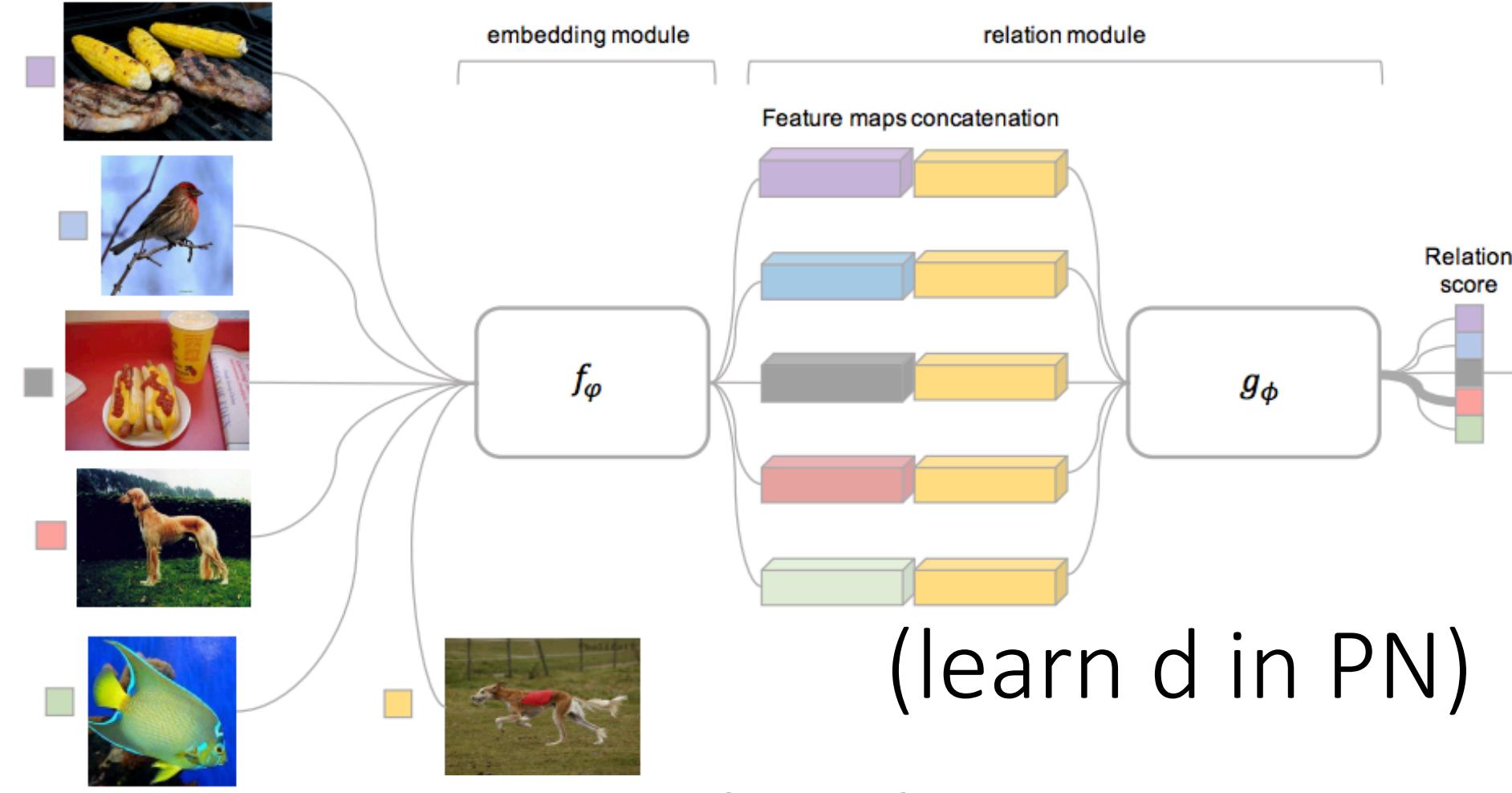
Non-parametric methods

So far: Siamese networks, matching networks, prototypical networks
Embed, then nearest neighbors.

Challenge

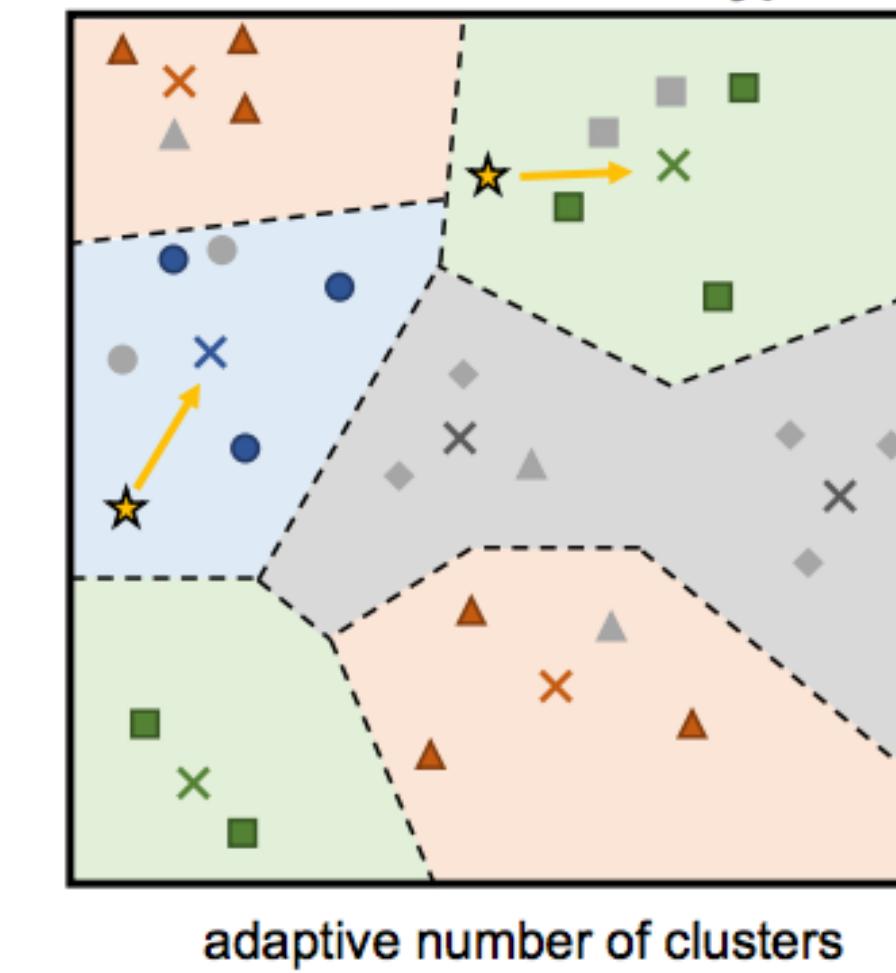
What if you need to reason about more complex relationships between datapoints?

Idea: Learn non-linear relation module on embeddings



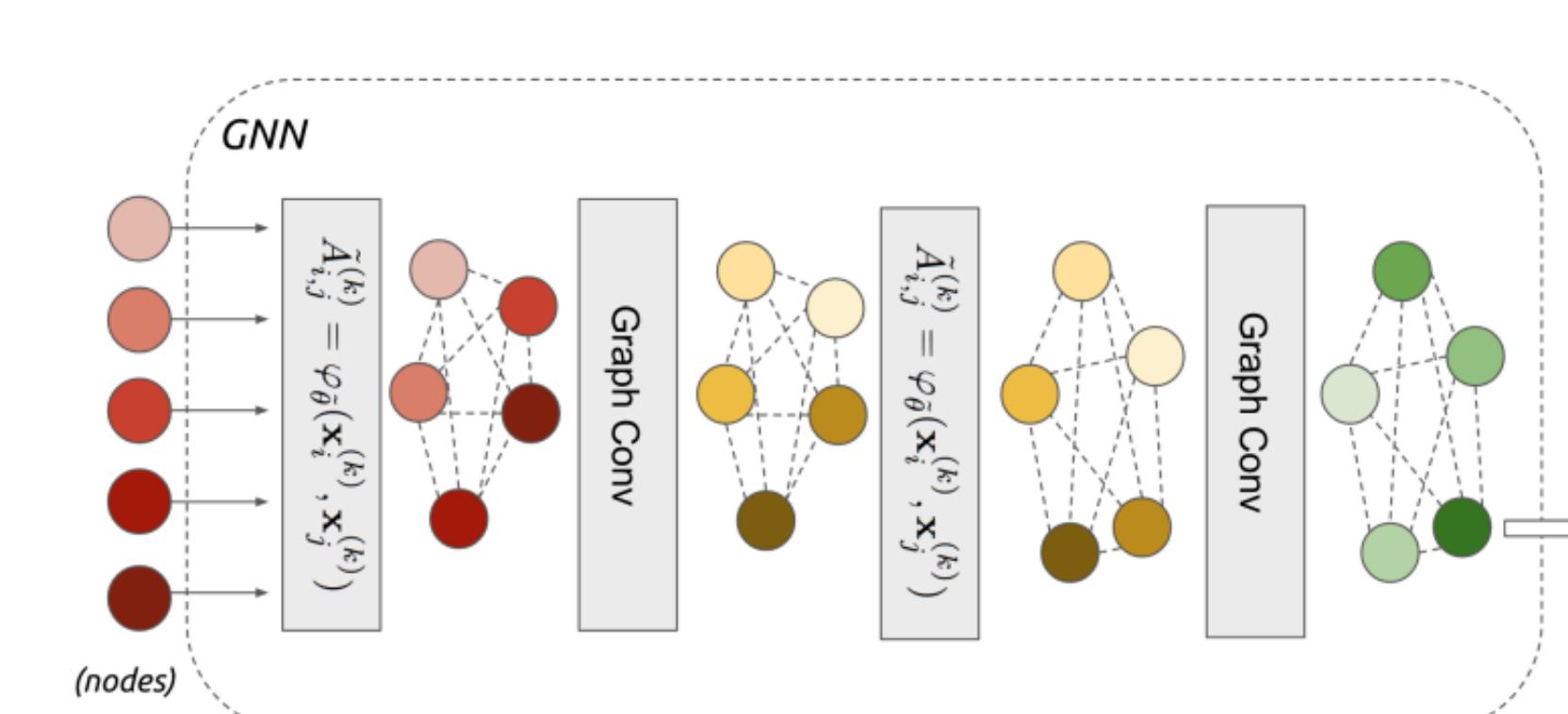
Qs: slic Sung et al. Relation Net

Idea: Learn infinite mixture of prototypes.



Allen et al. IMP, ICML '19

Idea: Perform message passing on embeddings



Garcia & Bruna, GNN

Amortized vs. Optimization vs. Non-Parametric

Computation graph perspective

Black-box amortized

$$y^{\text{ts}} = f_{\theta}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$

Optimization-based

$$\begin{aligned} y^{\text{ts}} &= f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) \\ &= f_{\phi_i}(x^{\text{ts}}) \end{aligned}$$

where $\phi_i = \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$

Non-parametric

$$\begin{aligned} y^{\text{ts}} &= f_{\text{PN}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) \\ &= \text{softmax}(-d(f_{\theta}(x), c_k)) \end{aligned}$$

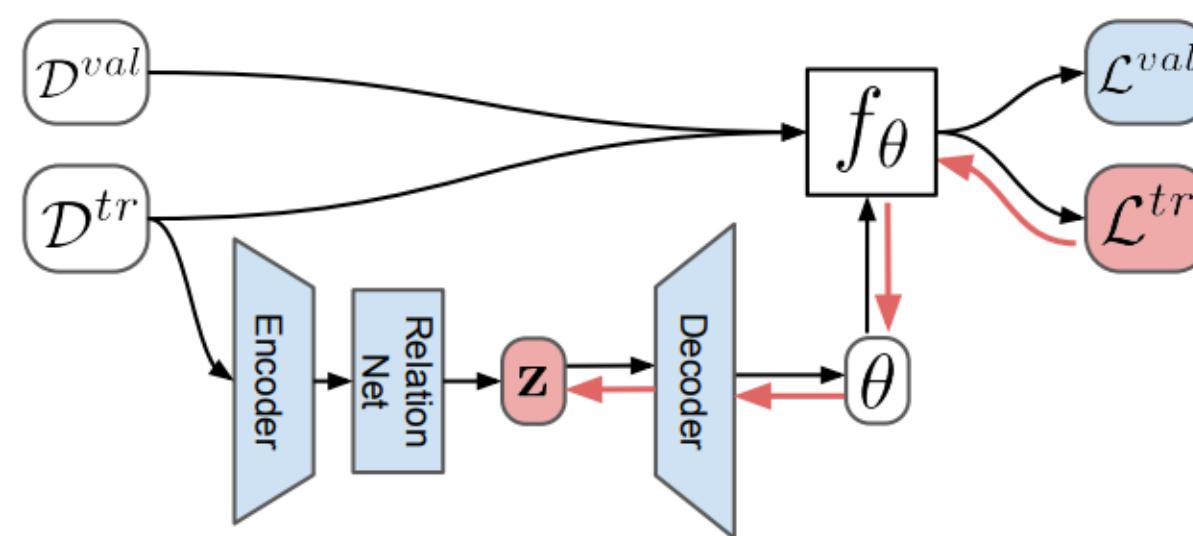
where $c_k = \frac{1}{|\mathcal{D}_i^{\text{tr}}|} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} f_{\theta}(x)$

Note: (again) Can mix & match components of computation graph

Both condition on data & run gradient descent.

Jiang et al. CAML '19

Gradient descent on relation net embedding.



Rusu et al. LEO '19

MAML, but initialize last layer as ProtoNet during meta-training

Triantafillou et al. Proto-MAML '19

Intermediate Takeaways

Black-box amortized

- + easy to combine with **variety of learning problems** (e.g. SL, RL)
- **challenging optimization** (no inductive bias at the initialization)
- often **data-inefficient**
- **model & architecture** intertwined

Optimization-based

- + handles **varying & large K** well
- + **structure lends well to out-of-distribution tasks**
- **second-order optimization**

Non-parametric

- + **simple**
- + entirely **feedforward**
- + **computationally fast & easy to optimize**
- **harder to generalize to varying K**
- hard to scale to **very large K**
- so far, **limited to classification**

Generally, well-tuned versions of each perform **comparably** on existing few-shot benchmarks!

Outline

- Problem statement
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

I can't believe it's not Bayesian

Recall parametric approaches: Use deterministic $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$ (i.e. a point estimate)

+



-

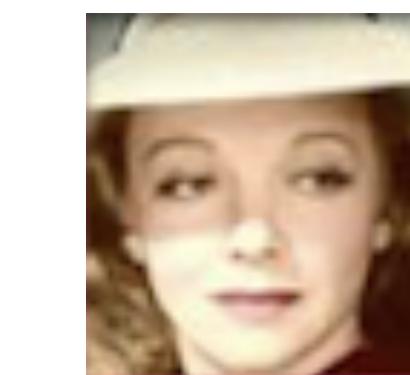


✓ Smiling,
✓ Wearing Hat,
✓ Young

ta



✓ Smiling,
✓ Wearing Hat,
✗ Young



✗ Smiling,
✓ Wearing Hat,
✓ Young

Important for:

Why/when is this a problem?

Few-shot learning problems may be **ambiguous**.
(even with prior)

Can we learn to **generate hypotheses**
about the **underlying** function?

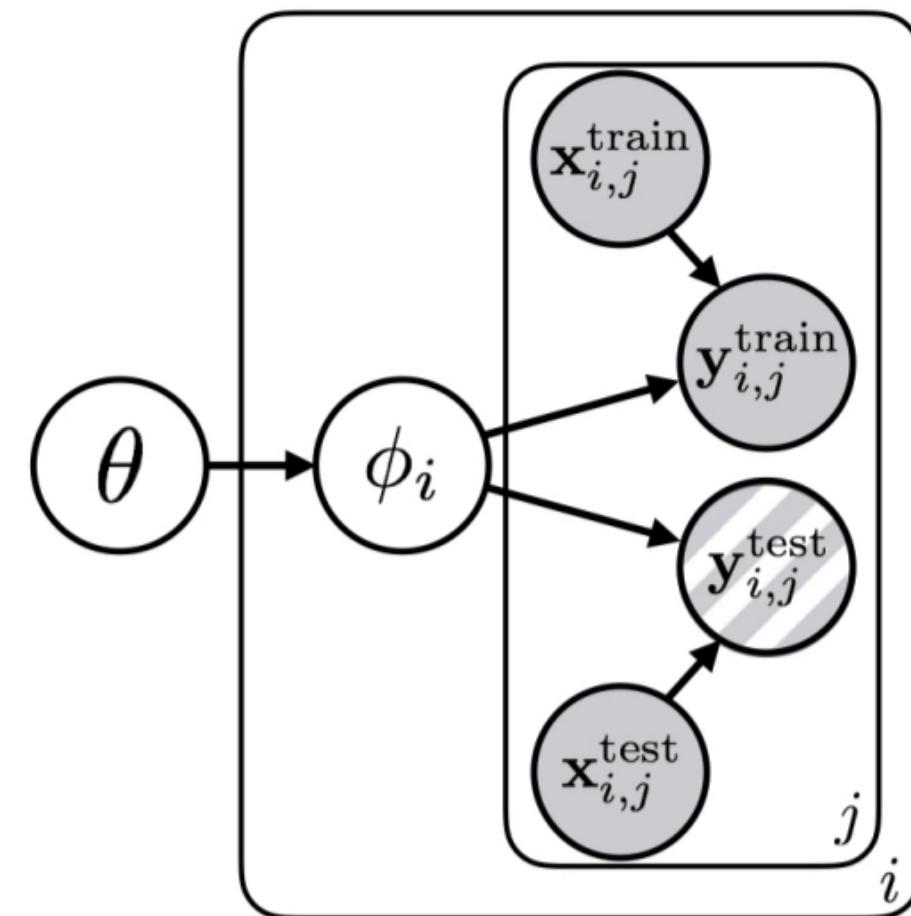
i.e. sample from $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$

- **safety-critical** few-shot learning
(e.g. medical imaging)
- learning to **actively learn**
- learning to **explore** in meta-RL

Active learning w/ meta-learning: Woodward & Finn '16,
Konyushkova et al. '17, Bachman et al. '17

Qs: slide ta

Meta-learning with ambiguity



$$\theta \sim p(\theta)$$

$$\phi_i \sim p(\phi_i | \theta)$$

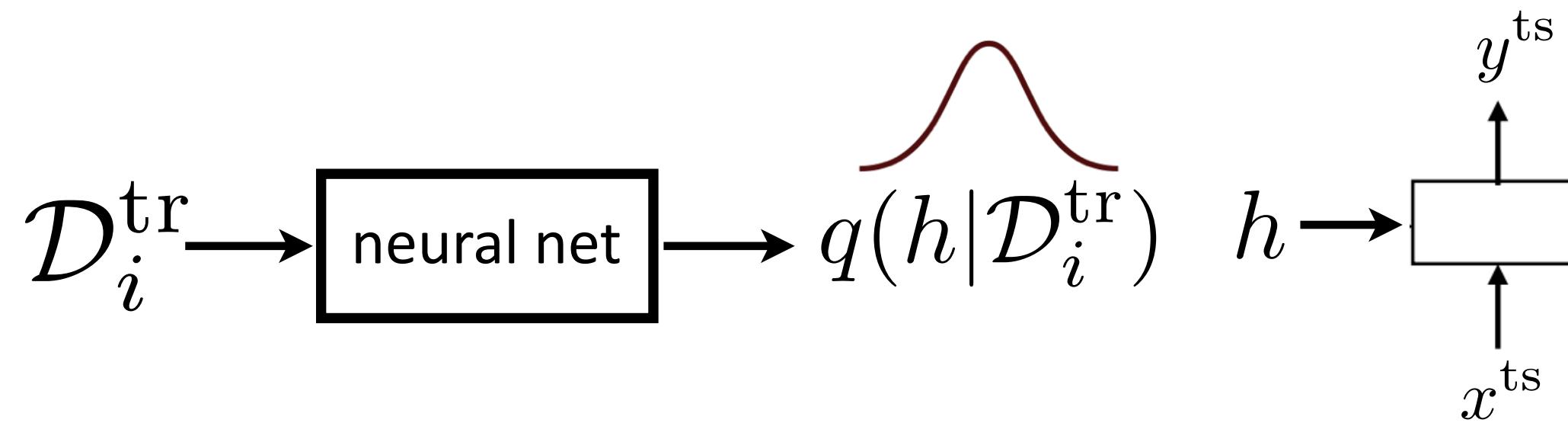
Goal: sample $\phi_i \sim p(\phi_i | x_i^{\text{train}}, y_i^{\text{train}}, x_i^{\text{test}})$

$$\log p(y_i^{\text{train}} | x_i^{\text{train}}, \phi_i)$$

$$\log p(y_i^{\text{test}} | x_i^{\text{test}}, \phi_i)$$

Black-Box Amortized Inference

Amortized Variational Inference

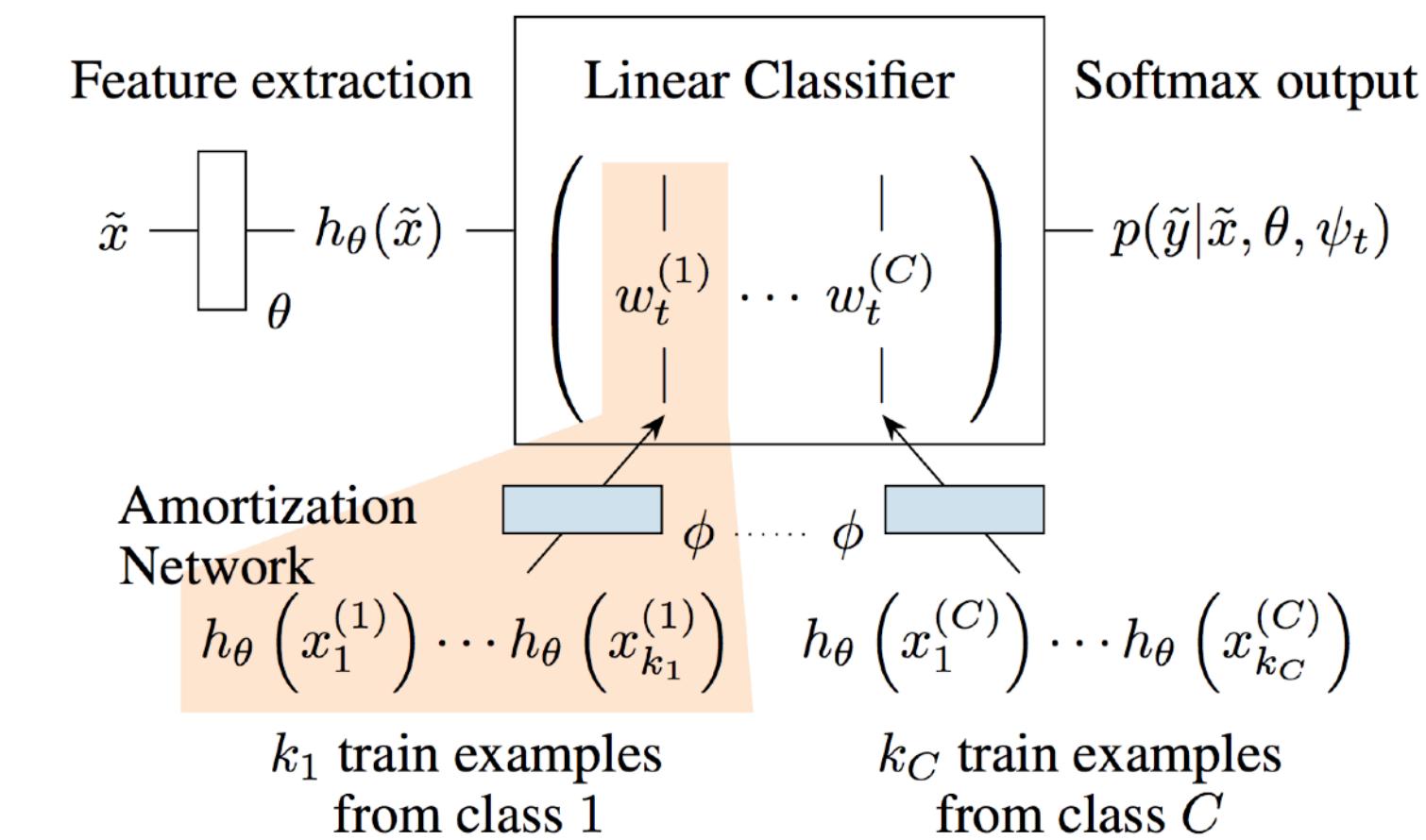


Simple idea: NN produces Gaussian distribution over h_i .

Train with amortized variational inference.

(Kingma & Welling VAE '13)

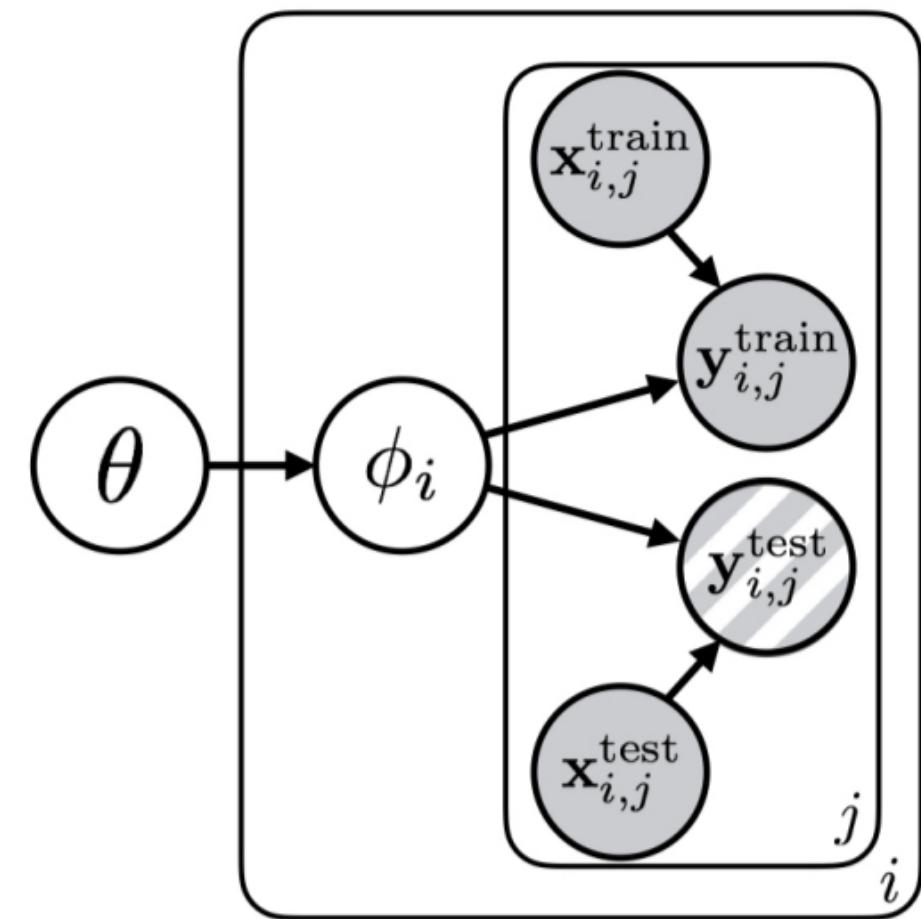
Output distribution over weights of last layer



Gordon et al. VERSA '19

Qs: slido.com/me What about Bayesian **optimization-based** meta-learning?

Meta-learning with ambiguity



$$\theta \sim p(\theta)$$

$$\phi_i \sim p(\phi_i | \theta)$$

Goal: sample $\phi_i \sim p(\phi_i | x_i^{\text{train}}, y_i^{\text{train}}, x_i^{\text{test}})$

$$\log p(y_i^{\text{train}} | x_i^{\text{train}}, \phi_i)$$

$$\log p(y_i^{\text{test}} | x_i^{\text{test}}, \phi_i)$$

~~x_i^{test}~~

What about Bayesian **optimization-based** meta-learning?

Model $p(\phi_i | \theta)$ as Gaussian

Same amortized variational inference for training.

(Ravi & Beatson '19)

Amortized Bayesian Meta-Learning

Stein Variational Gradient (BMAML)

Gradient-based inference on last layer only.

Use SVGD to avoid Gaussian modeling assumption.

Ensemble of MAMLs (EMAML)

(Kim et al. Bayesian MAML '18)

Can we model **non-Gaussian posterior over all parameters**?

Qs: slido.com/meta

Sampling parameter vectors

$$\theta \sim p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$$

$$\log p(y_i^{\text{train}} | x_i^{\text{train}}, \phi_i)$$

$$\phi_i \sim p(\phi_i | \theta)$$

$$\log p(y_i^{\text{test}} | x_i^{\text{test}}, \phi_i)$$

Goal: sample $\phi_i \sim p(\phi_i | x_i^{\text{train}}, y_i^{\text{train}})$

$$p(\phi_i | x_i^{\text{train}}, y_i^{\text{train}}) \propto \int p(\theta) p(\phi_i | \theta) p(y_i^{\text{train}} | x_i^{\text{train}}, \phi_i) d\theta$$

⇒ this is completely intractable! ☐

what if we knew $p(\phi_i | \theta, x_i^{\text{train}}, y_i^{\text{train}})$?

⇒ now sampling is easy! just use ancestral sampling!

key idea: $p(\phi_i | \theta, x_i^{\text{train}}, y_i^{\text{train}}) \approx \delta(\hat{\phi}_i)$

this is **extremely** crude

but **extremely** convenient!

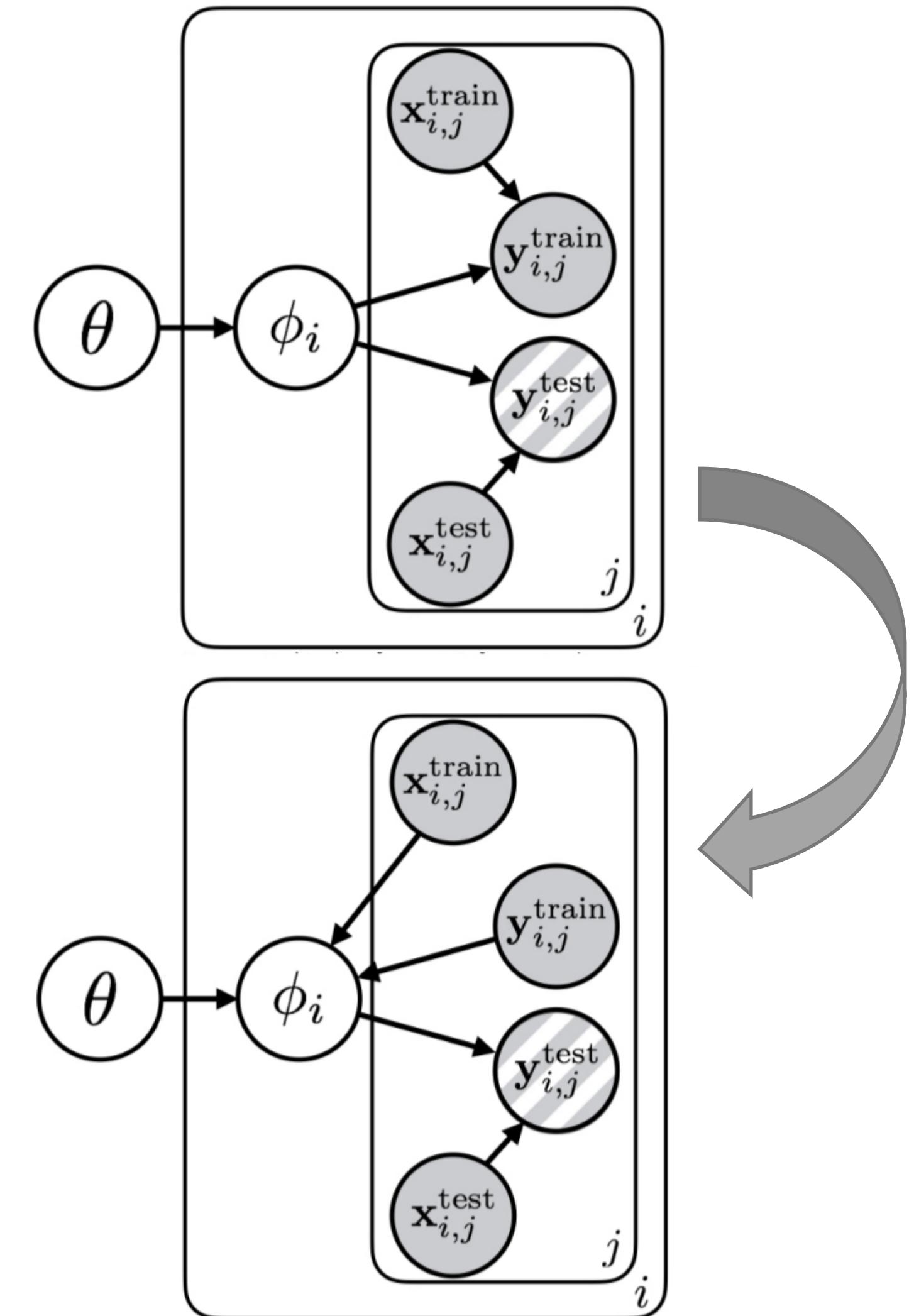
$$\hat{\phi}_i \approx \theta + \alpha \nabla_\theta \log p(y_i^{\text{train}} | x_i^{\text{train}}, \theta)$$

(Santos '92, Grant et al. ICLR '18)

Qs: slido.com/meta

Training is harder. We use **amortized variational inference**.

Finn*, Xu* et al. Probabilistic MAML '18

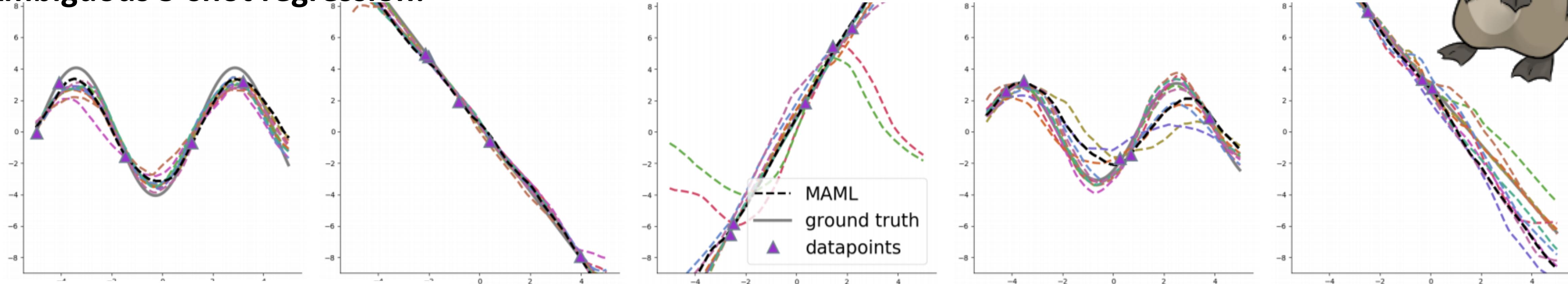


PLATIPUS

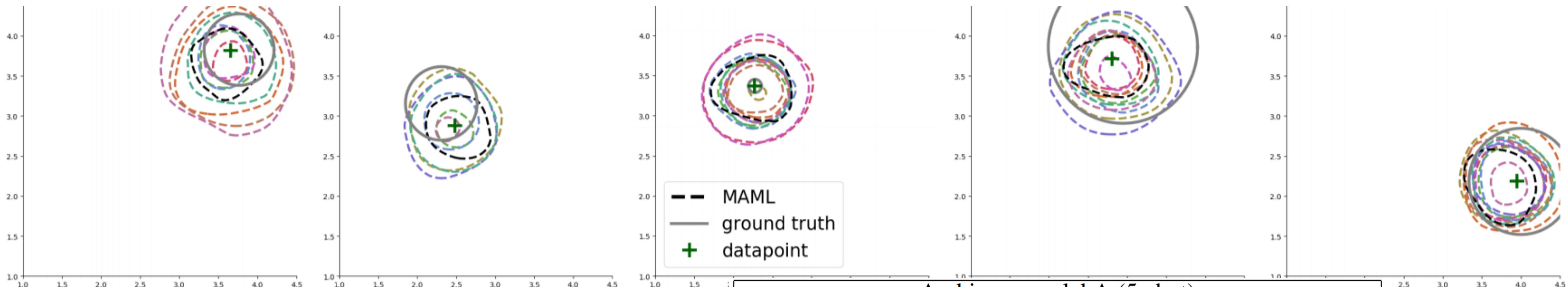
Probabilistic LATent model for Incorporating Priors and Uncertainty in few-Shot learning



Ambiguous 5-shot regression:



Ambiguous 1-shot classification:



**Better models ambiguous few-shot
image classification problems:**

Ambiguous celebA (5-shot)		
	Accuracy	Coverage (max=3)
MAML	$69.26 \pm 2.18\%$	1.00 ± 0.0
MAML + noise	$54.73 \pm 0.8\%$	2.60 ± 0.12
PLATIPUS (ours)	$69.97 \pm 1.32\%$	2.62 ± 0.11

Qs: slido.com/meta

Finn*, Xu* et al. Probabilistic MAML '18

Deep Bayesian Meta-Learning: Further Reading

Edwards & Storkey, Towards a Neural Statistician. 2017

Black-box approaches:

Gordon et al., VERSA 2019

Garnelo et al. Conditional Neural Processes 2018

Optimization-based approaches:

Kim et al., Bayesian MAML. 2018

Xu et al., Probabilistic MAML. 2018

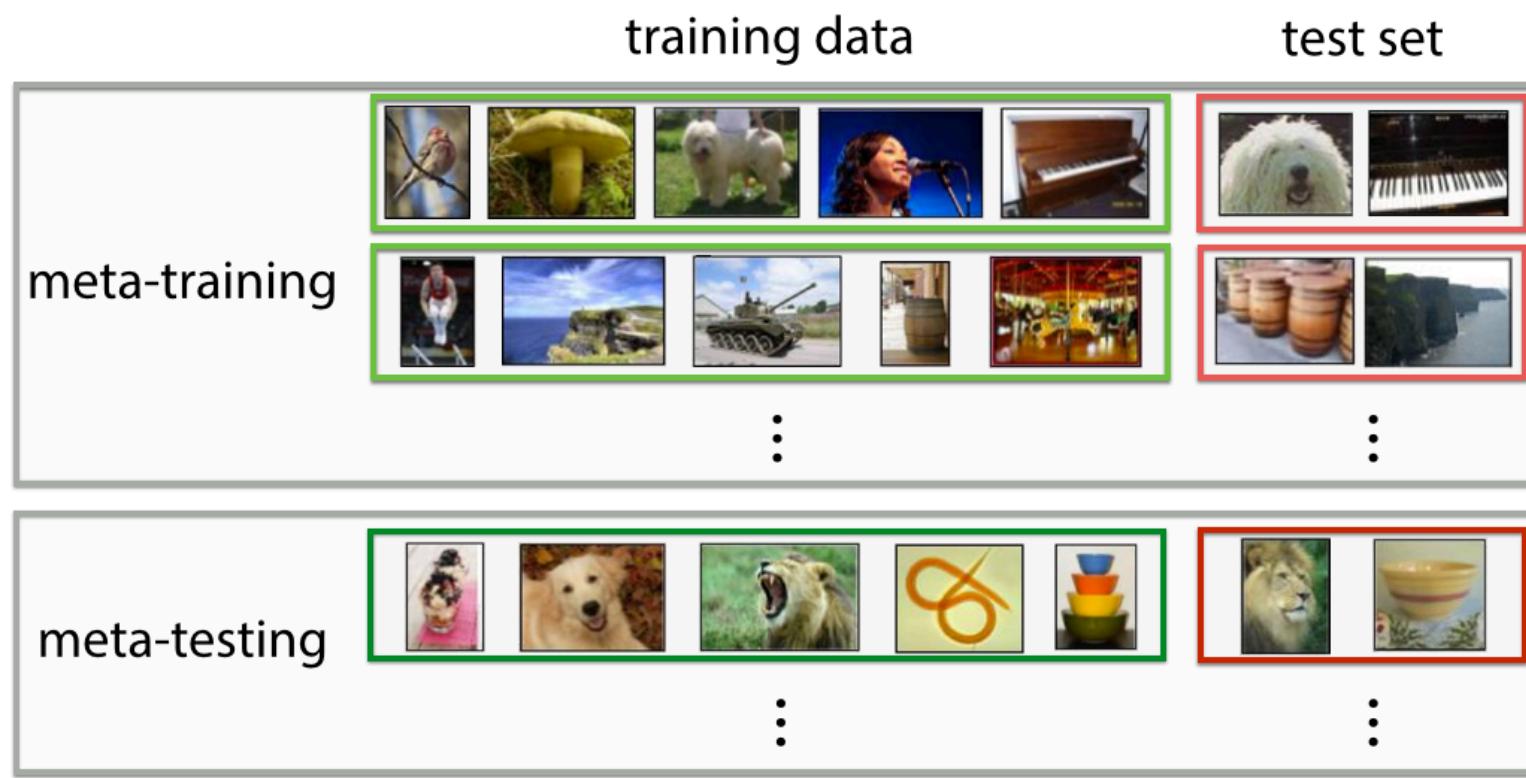
Ravi & Beatson., Amortized Bayesian Meta-Learning 2019

Outline

- Problem statement
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

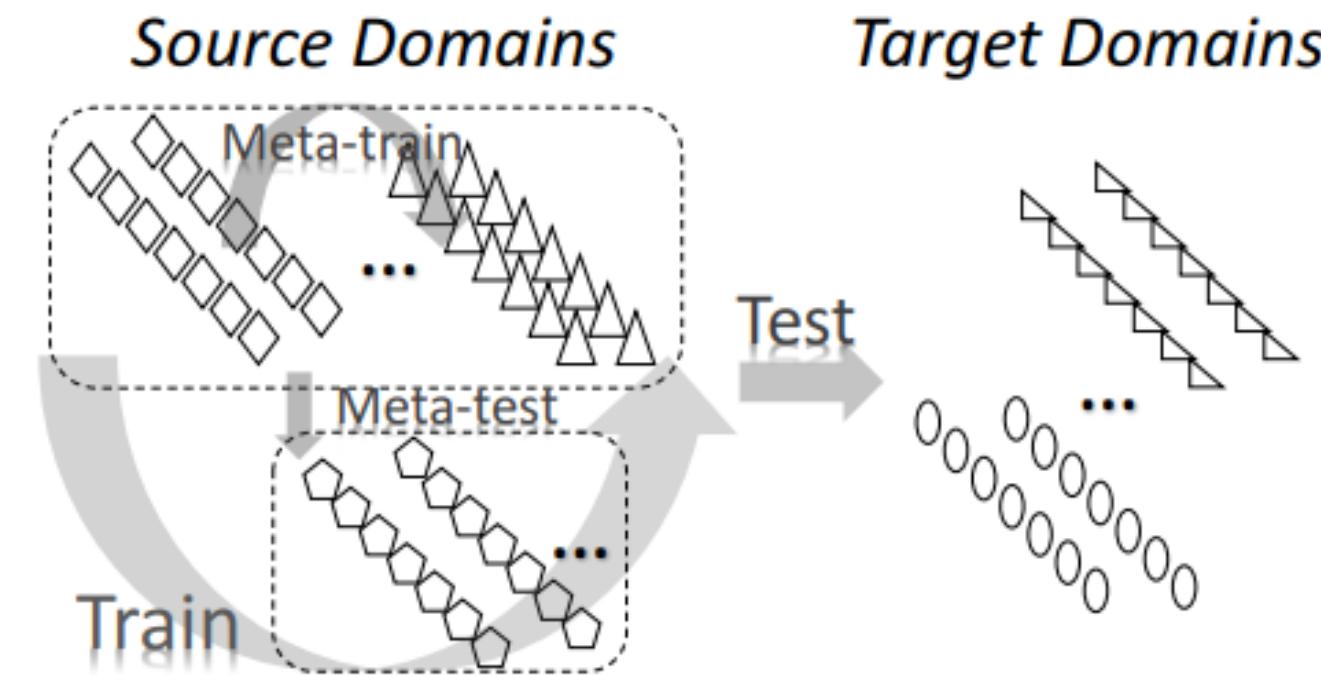
Applications in computer vision

few-shot image recognition



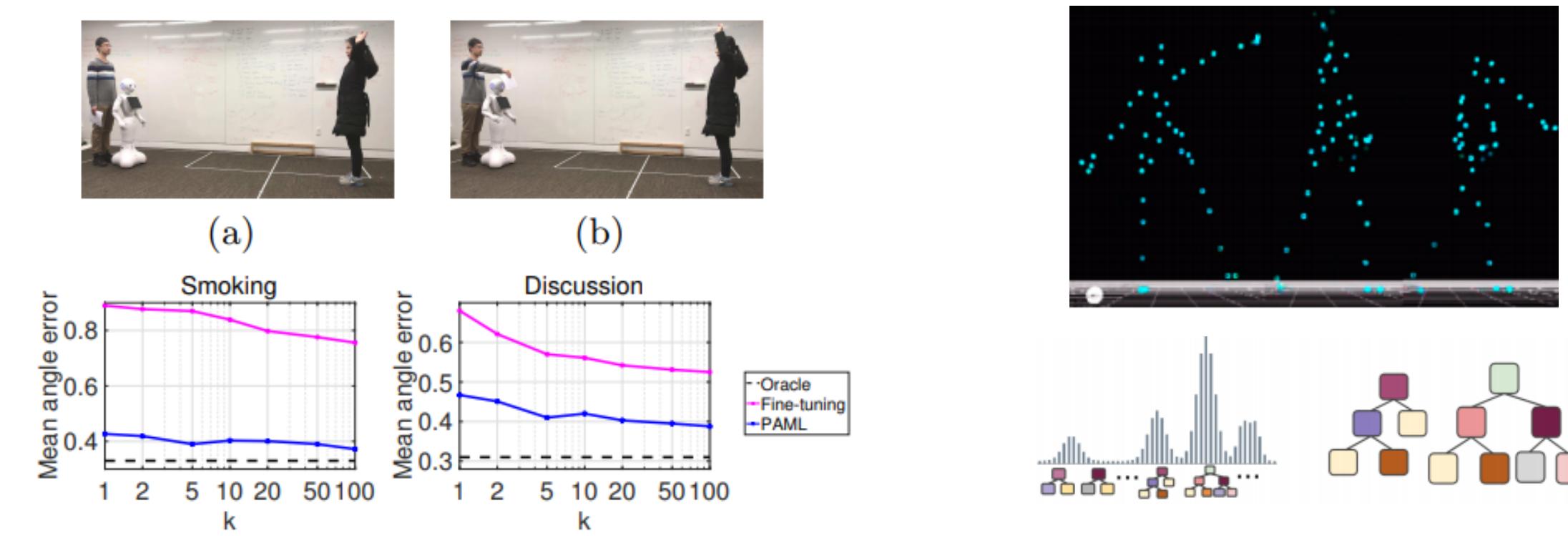
see, e.g.: Vinyals et al. **Matching Networks for One Shot Learning**, and many many others

domain adaptation



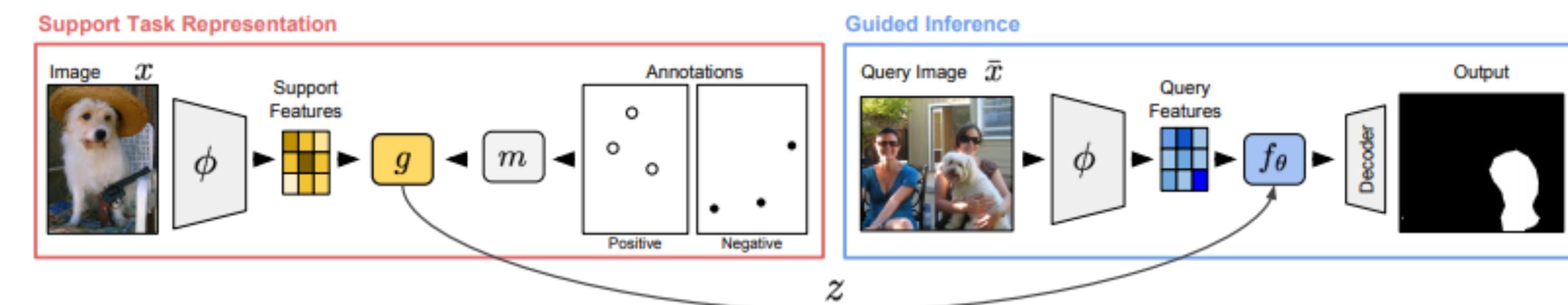
see, e.g.: Li, Yang, Song, Hospedales. **Learning to Generalize: QS: Meta-Learning for Domain Adaptation**.

human motion and pose prediction



see, e.g.: Gui et al. **Few-Shot Human Motion Prediction via Meta-Learning**.
Alet et al. **Modular Meta-Learning**.

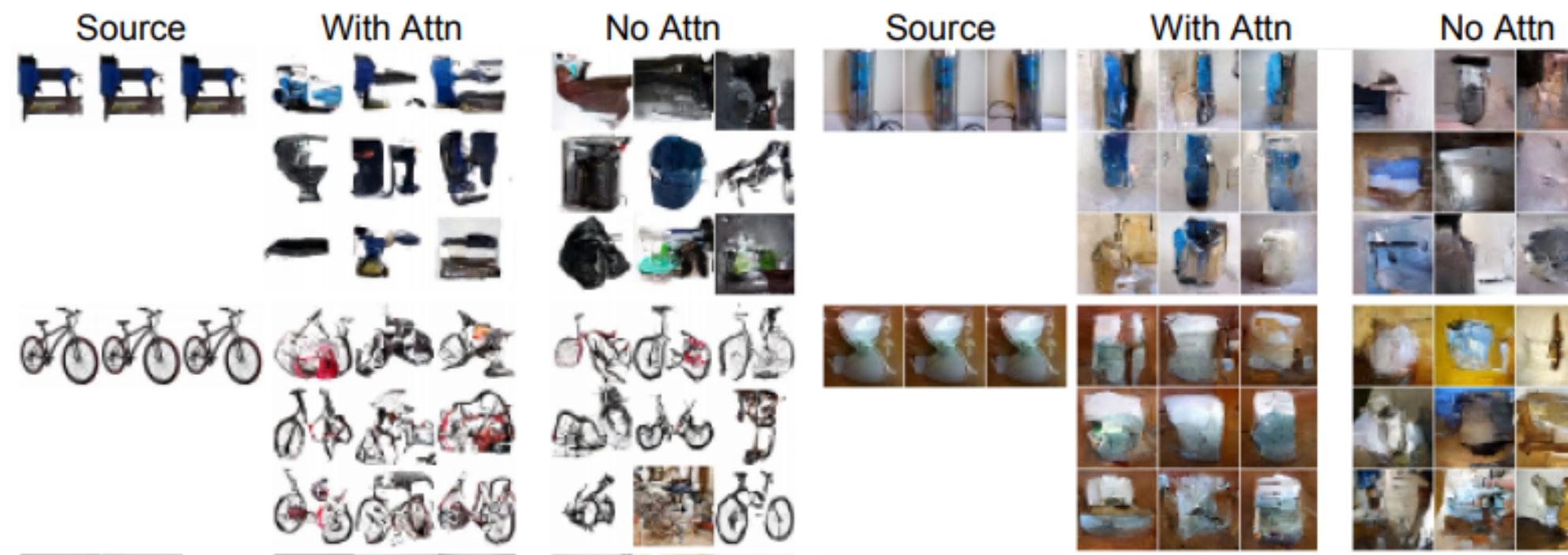
few-shot segmentation



see, e.g.: Shaban, Bansal, Liu, Essa, Boots. **One-Shot Learning for Semantic Segmentation**.
Rakelly, Shelhamer, Darrell, Efros, Levine. **Few-Shot Segmentation Propagation with Guided Networks**.
Dong, Xing. **Few-Shot Semantic Segmentation with Prototype Learning**.

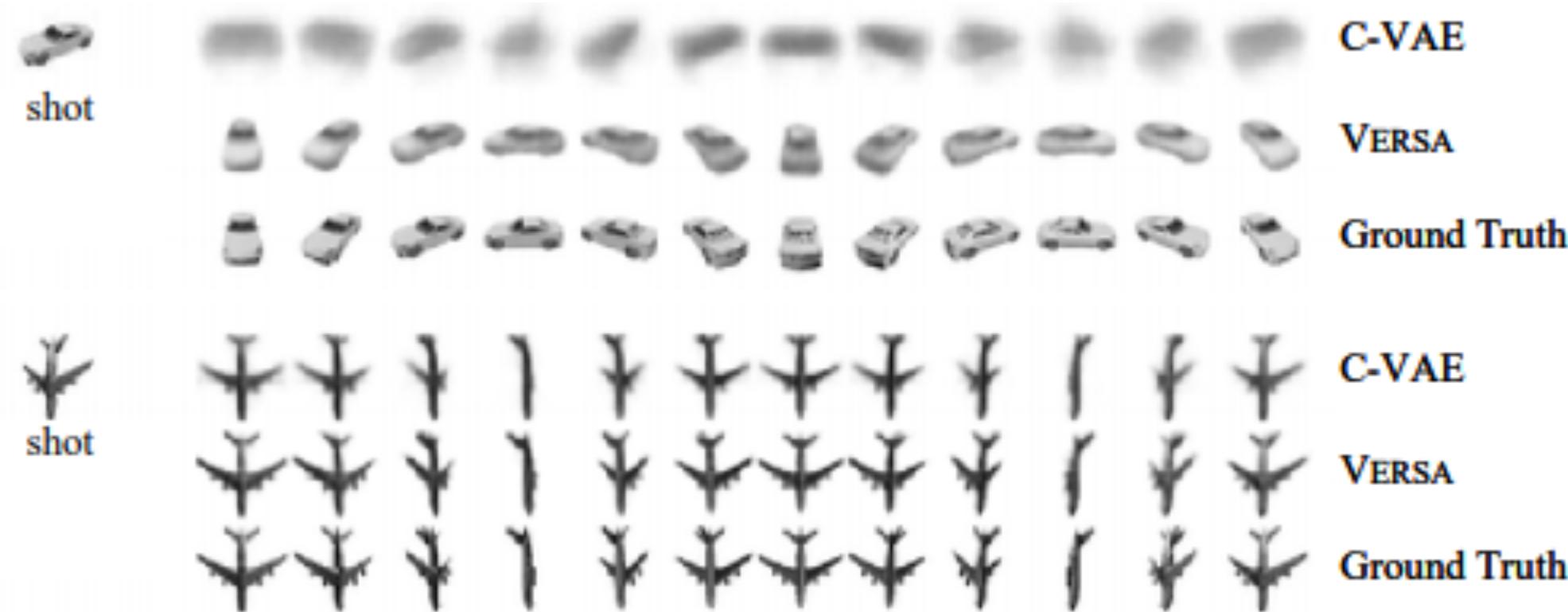
Applications in image & video generation

few-shot image generation



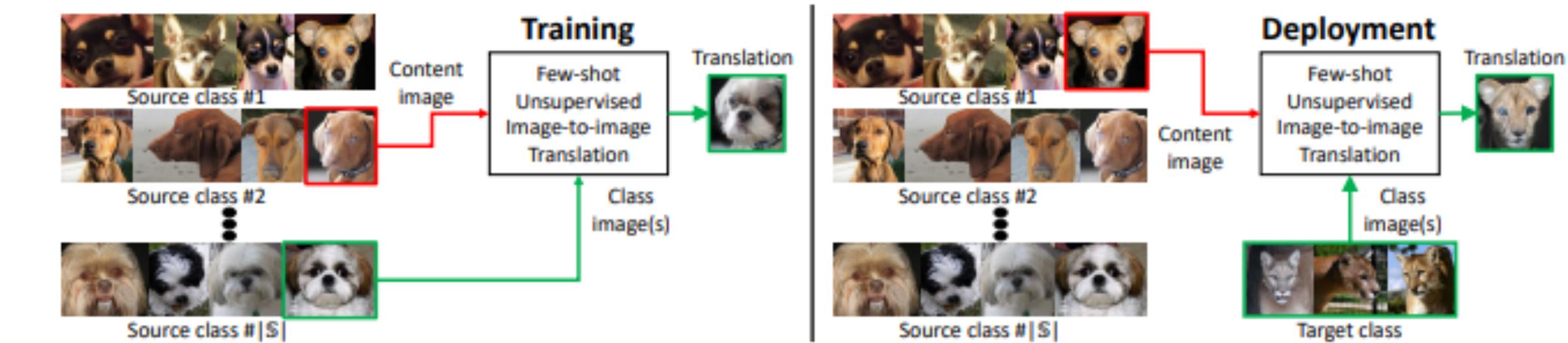
see, e.g.: Reed, Chen, Paine, van den Oord, Eslami, Rezende, Vinyals, de Freitas. **Few-Shot Autoregressive Density Estimation**. and many many others.

generation of novel viewpoints



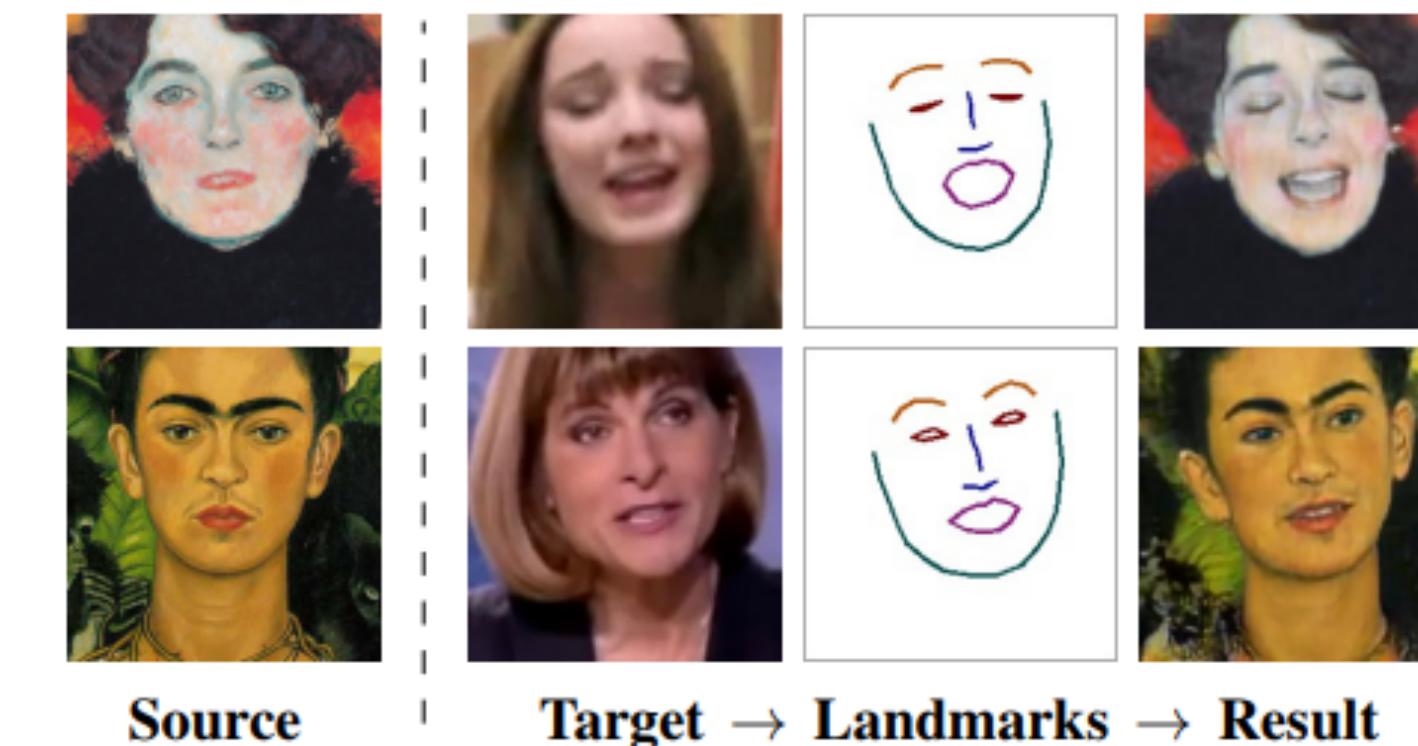
see, e.g.: Gordon, Bronskill, Bauer, Nowozin, Turner. **VERSAs: Versatile and Efficient Few-Shot Learning**.
QS: <https://arxiv.org/abs/1806.04351>

few-shot image-to-image translation



see, e.g.: Liu, Huang, Mallya, Karras, Aila, Lehtinen, Kautz. **Few-Shot Unsupervised Image-to-Image Translation**.

generating talking heads from images

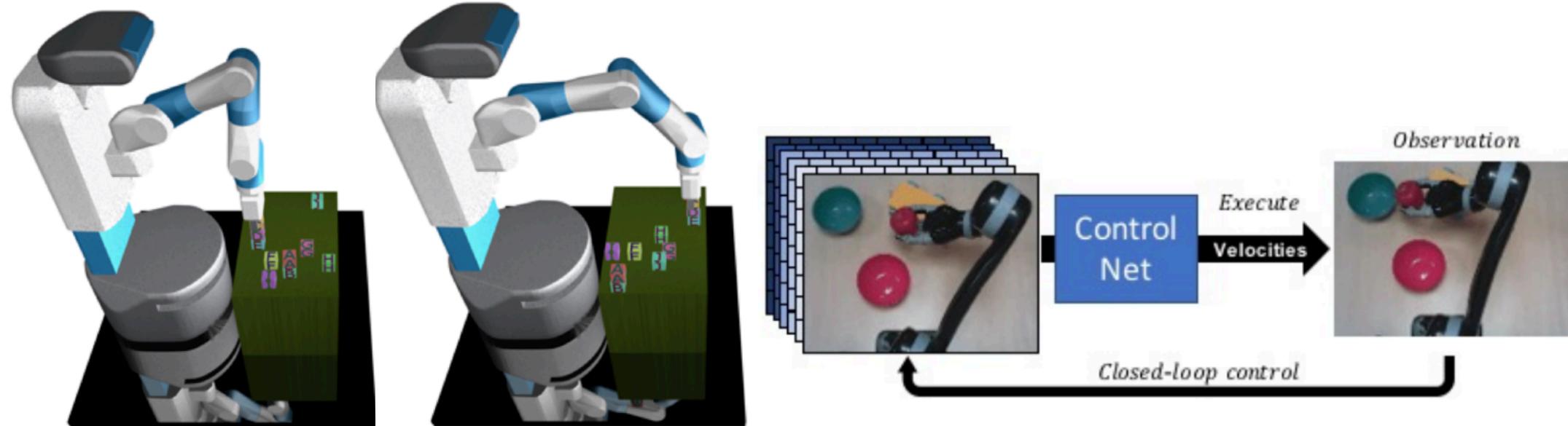


see, e.g.: Zakharov, Shysheya, Burkov, Lempitsky. **Few-Shot Adversarial Learning of Realistic Neural Talking Head Models**

One-Shot *Imitation* Learning

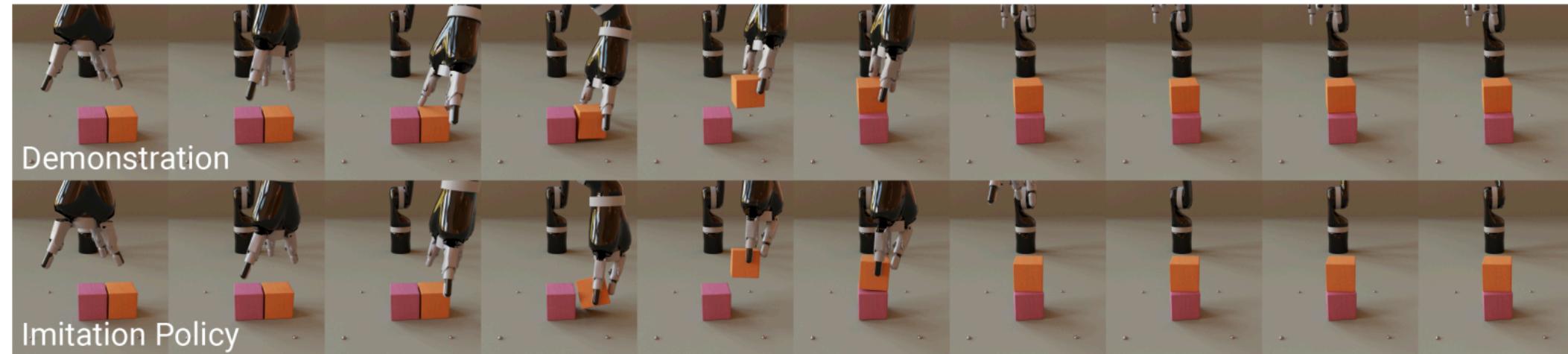
Goal: Given one demonstration of a new task, learn a policy
meta-learning with supervised imitation learning

Black-box amortized inference



Duan et al. One-Shot
Imitation Learning '17

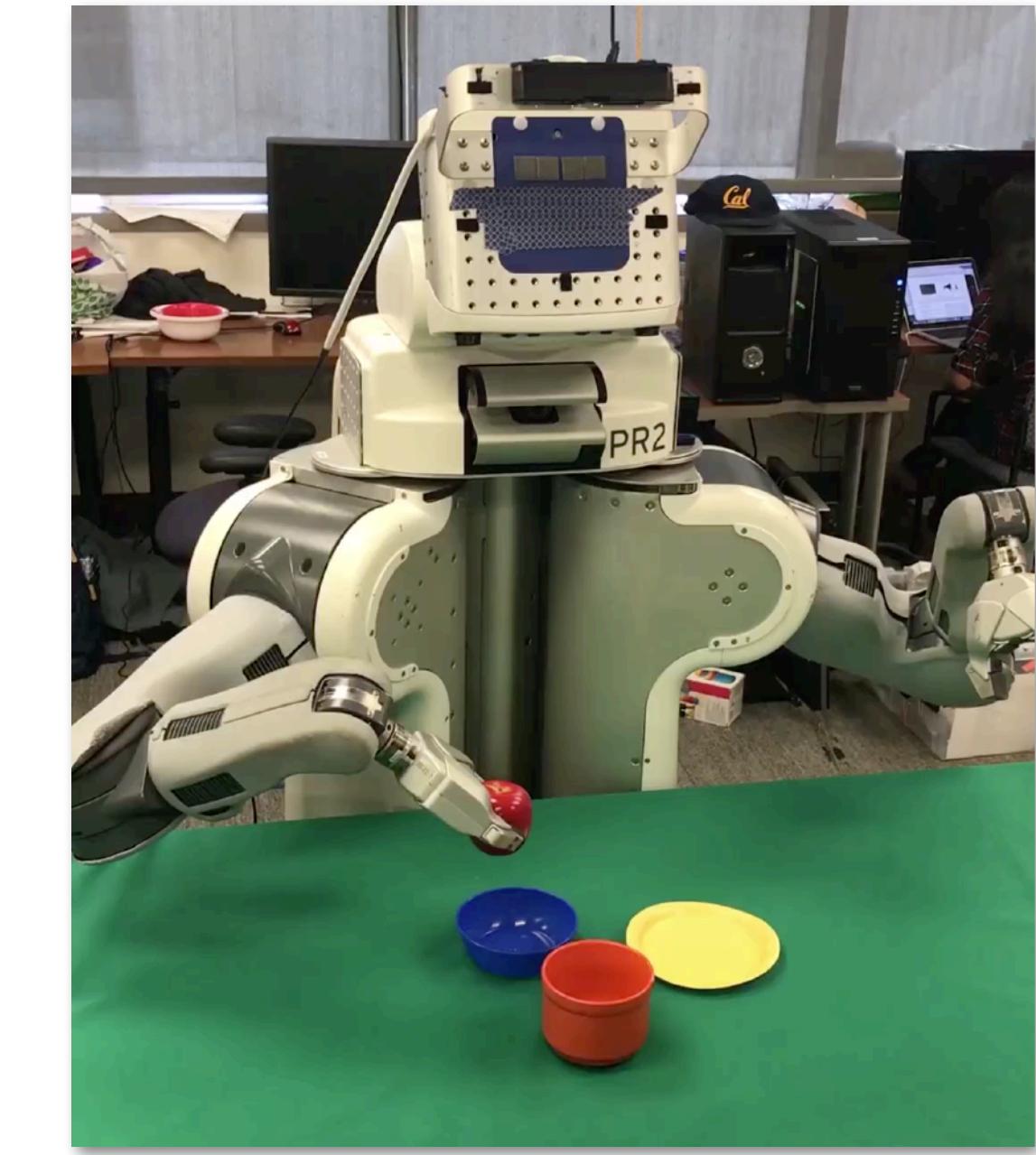
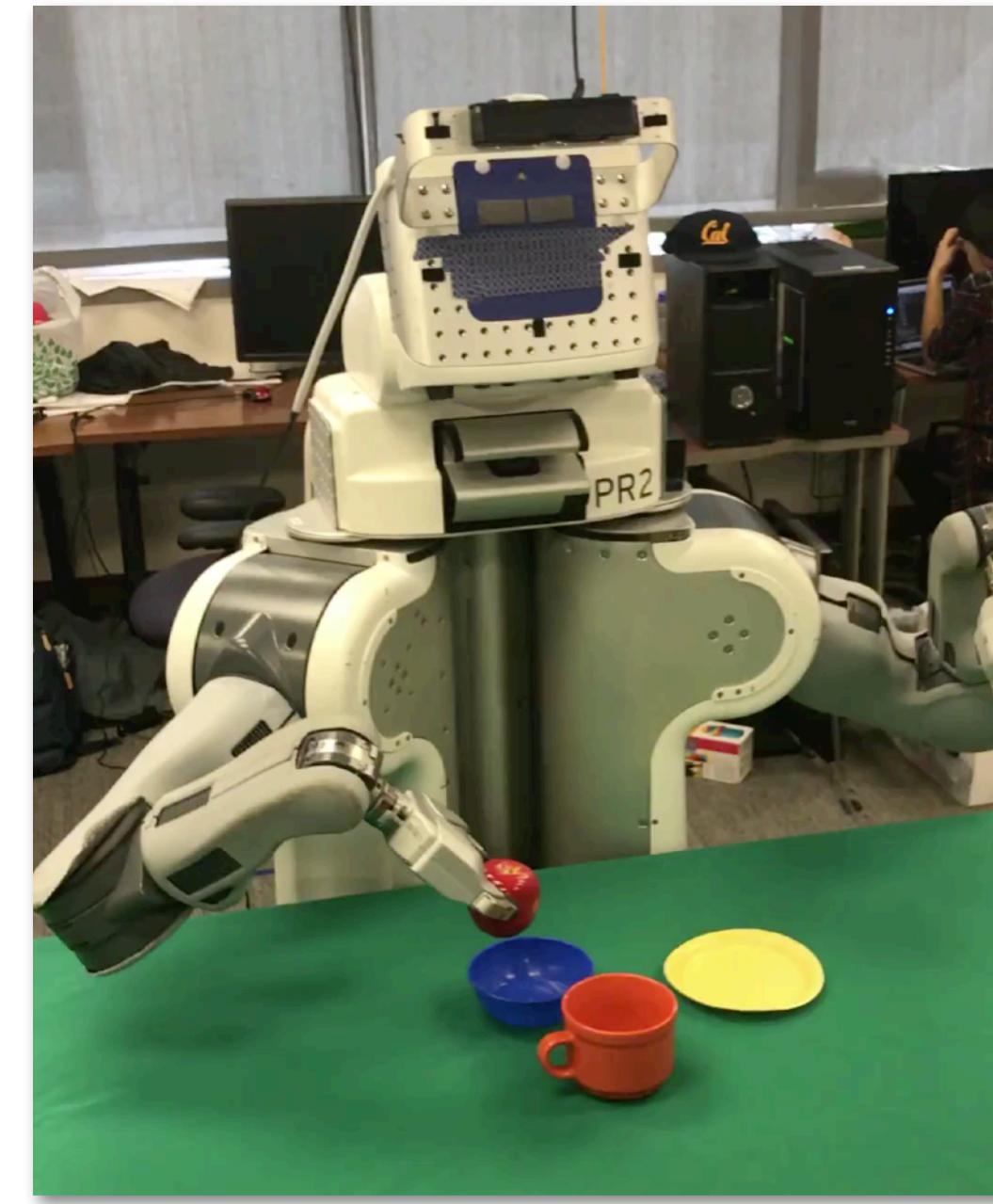
James et al. Task-
Embedded Control '18



Le Paine et al. One-Shot High Fidelity Imitation '19

Optimization-based inference

Finn*, Yu* et al. Meta Imitation Learning '17
input demo
(via teleoperation)



Also: **One-shot inverse RL** (Xu et al. MandRIL '18, Gleave & Habryka '18), **One-shot hierarchical imitation** (Yu et al. '18)

Learning to Learn from *Weak* Supervision

input human demo



resulting policy



Learning to Learn from *Weak* Supervision

meta-training

$$\min_{\theta} \sum_{\text{task } i} \mathcal{L}_{\text{test}}^i (\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{train}}^i (\theta))$$

fully supervised

meta-test

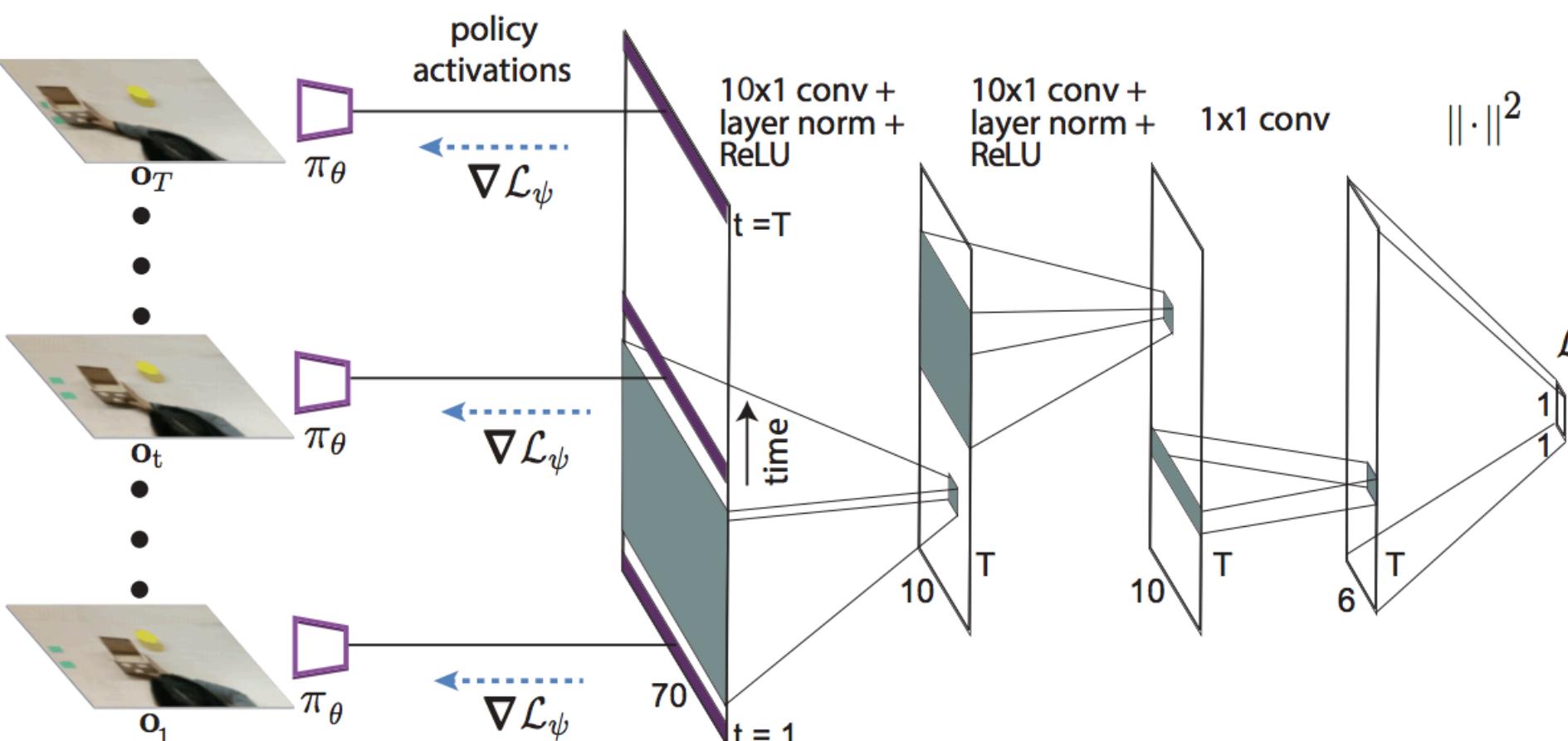
$$\theta' \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta)$$

weakly supervised

What if the weakly supervised loss is unavailable?

$$\min_{\theta, \psi} \sum_{\text{task } i} \mathcal{L}_{\text{test}}^i (\theta - \alpha \nabla_{\theta} \mathcal{L}_{\psi}^i (\theta))$$

\mathcal{L}_{ψ}



Yu*, Finn*, Xie, Dasari, Zhang,
Abbeel, Levine RSS '18

Grant, Finn, Peterson, Abbott, Levine,
Darrell, Griffiths NIPS CIAI Workshop '17

Meta-Learning for *Language*

Adapting to *new programs*

Meta Program Induction

Learn new program from a few I/O examples.

Devlin, Bunel* et al. NeurIPS '17*

Program Synthesis

Question:

How many CFL teams are from York College?

SQL:

```
SELECT COUNT CFL Team FROM  
CFLDraft WHERE College = "York"
```

Result:

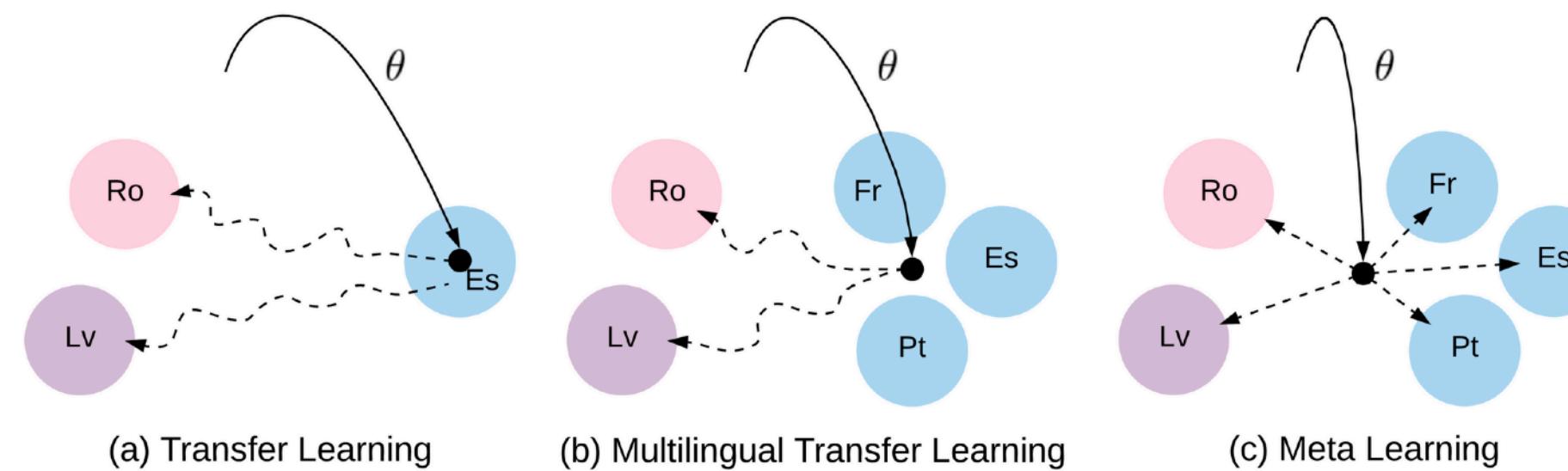
2

Construct pseudo-tasks with relevance function

Huang et al. NAACL '18

Adapting to *new languages*

Low-Resource Neural Machine Translation



Learn to translate new language pair w/o a lot of paired data?

Gu et al. EMNLP '18

Learning *new words*

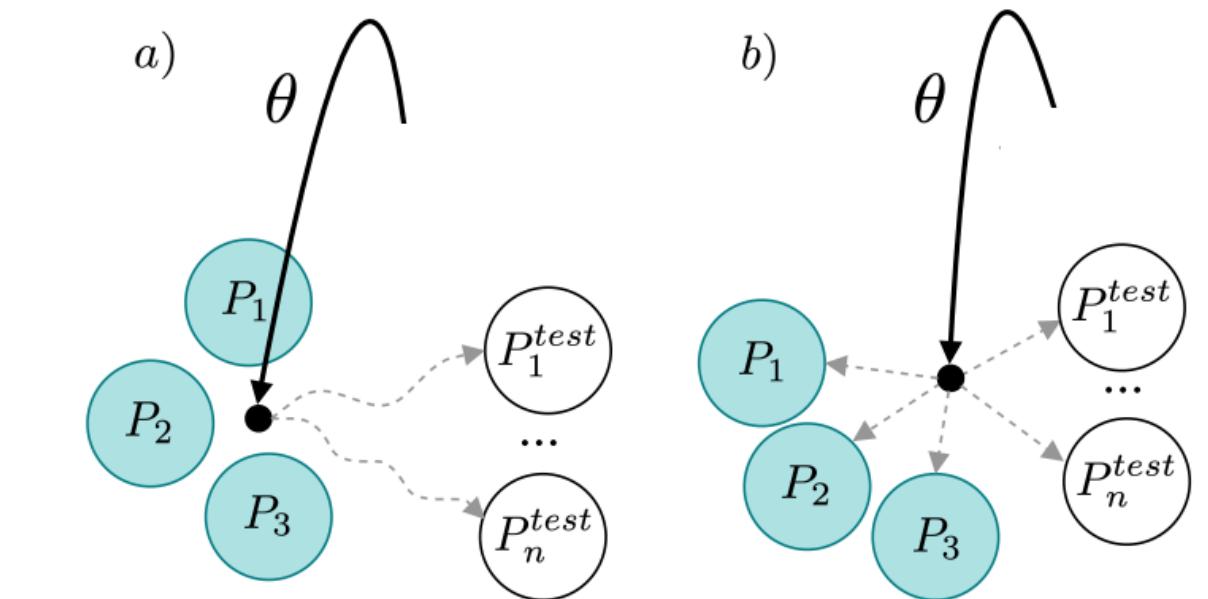
One-Shot Language Modeling

Learn how to use a new word from one example usage.

Vinyals et al. Matching Networks, '16

Adapting to *new personas*

Personalizing Dialogue Agents



Adapt dialogue to a persona with a few examples

Lin, Madotto* et al. ACL '19*

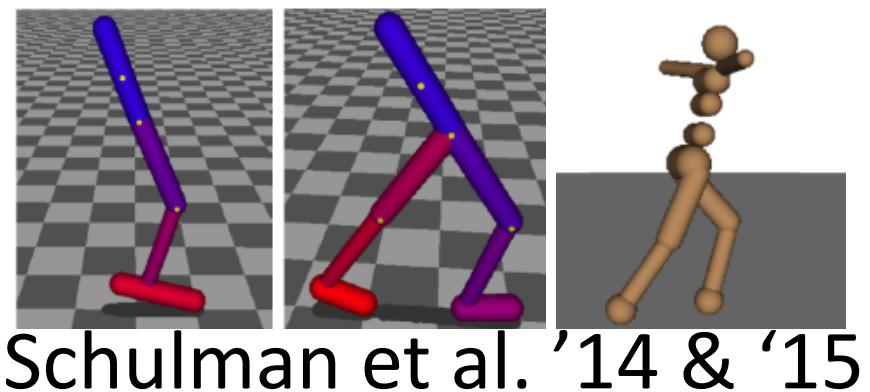
Outline

- Problem statement
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

Why should we care about meta-RL?



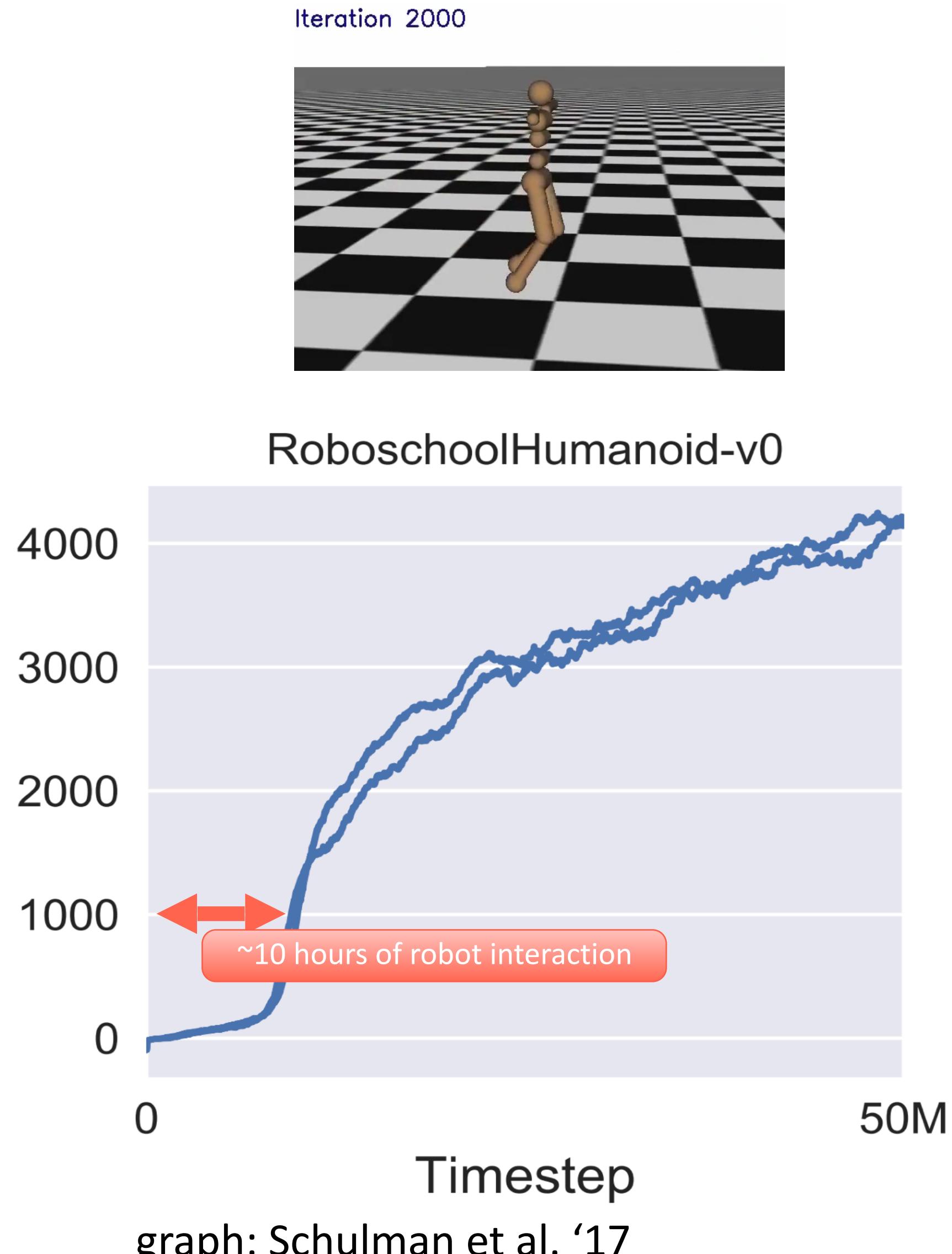
Mnih et al. '13



Schulman et al. '14 & '15



Qs: slido.com/meta



people can learn new skills **extremely** quickly
how?
we never learn from scratch!

Can we **meta-learn** reinforcement learning “algorithms” that are much more efficient?

The reinforcement learning problem

Markov decision process

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\}$$

\mathcal{S} – state space

states $s \in \mathcal{S}$ (discrete or continuous)

\mathcal{A} – action space

actions $a \in \mathcal{A}$ (discrete or continuous)

\mathcal{P} – transition function, i.e. $p(s_{t+1}|a_t, s_t) = \mathcal{P}(s_t, a_t, s_{t+1})$

r – reward function

$$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$

$r(s_t, a_t)$ – reward

$\pi_\theta(a|s)$ – policy with params θ

$$\theta^* = \arg \max_{\theta} E_{\pi_\theta} \left[\sum_{t=0}^T r(s_t, a_t) \right]$$

expectation under π_θ and \mathcal{P}



Andrey Markov



Richard Bellman

The reinforcement learning problem

$$\theta^* = \arg \max_{\theta} E_{\pi_{\theta}} \left[\sum_{t=0}^T r(s_t, a_t) \right]$$

$$\theta^* = \arg \max_{\theta} E_{\pi_{\theta}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \leftarrow \text{infinite horizon, discounted return}$$

$$\theta^* = \arg \max_{\theta} E_{\pi_{\theta}(s, a)} [r(s_t, a_t)]$$

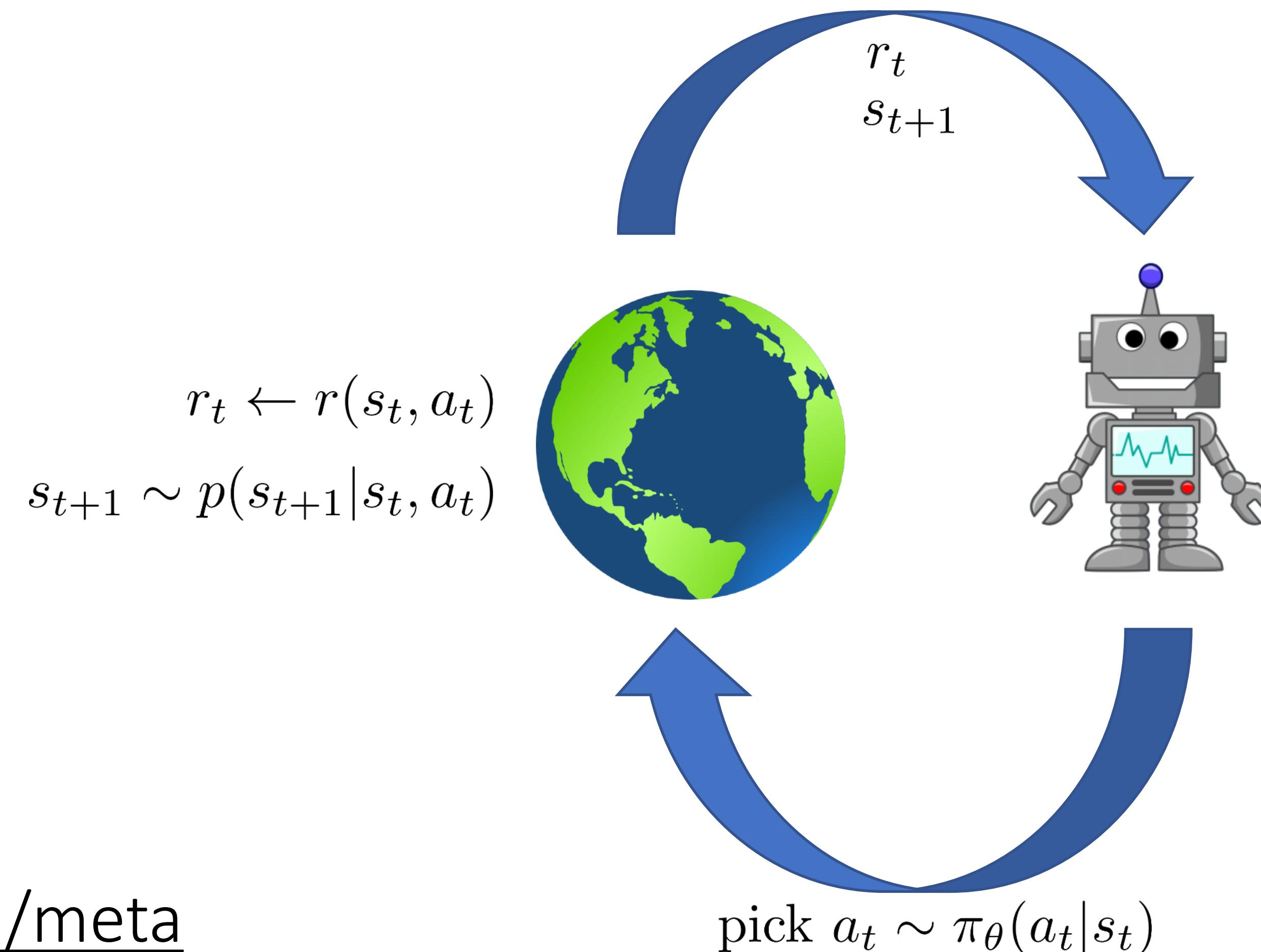
↑
stationary distribution

$$p_{\theta}(\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_T, \mathbf{a}_T) = \underbrace{p(\mathbf{s}_1)}_{\pi_{\theta}(\tau)} \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\theta^* = \arg \max_{\theta} E_{\pi_{\theta}(\tau)} [R(\tau)]$$

Every RL algorithm in a nutshell

$$\theta^* = \arg \max_{\theta} E_{\pi_{\theta}(\tau)} [R(\tau)]$$



store (s_t, a_t, s_{t+1}, r_t) in buffer \mathcal{B}
improve π_{θ} ...
...directly, via policy gradients
...via value function or Q-function
...implicitly, via model $\hat{p}(s_{t+1} | s_t, a_t)$

Meta-learning so far...

learn θ such that $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$ is good for $\mathcal{D}_i^{\text{ts}}$

Probabilistic view:

$$\theta^\star = \arg \max_{\theta} \sum_{i=1}^n \log p(\phi_i | \mathcal{D}_i^{\text{ts}})$$

where $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$

Deterministic view:

$$\theta^\star = \arg \min_{\theta} \sum_{i=1}^n \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{ts}})$$

where $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$

$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \dots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

$$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \dots, (x_l^i, y_l^i)\}$$

The meta reinforcement learning problem

“Generic” learning (deterministic view):

$$\begin{aligned}\theta^* &= \arg \min_{\theta} \mathcal{L}(\theta, \mathcal{D}^{\text{tr}}) \\ &= f_{\text{learn}}(\mathcal{D}^{\text{tr}})\end{aligned}$$

Reinforcement learning:

$$\begin{aligned}\theta^* &= \arg \max_{\theta} E_{\pi_{\theta}(\tau)}[R(\tau)] \\ &= f_{\text{RL}}(\mathcal{M}) \quad \mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\}\end{aligned}$$

MDP

“Generic” meta-learning (deterministic view):

$$\begin{aligned}\theta^* &= \arg \min_{\theta} \sum_{i=1}^n \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{ts}}) \\ \text{where } \phi_i &= f_{\theta}(\mathcal{D}_i^{\text{tr}})\end{aligned}$$

Meta-reinforcement learning:

$$\begin{aligned}\theta^* &= \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)] \\ \text{where } \phi_i &= f_{\theta}(\mathcal{M}_i)\end{aligned}$$

MDP for task i

The meta reinforcement learning problem

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

where $\phi_i = f_{\theta}(\mathcal{M}_i)$

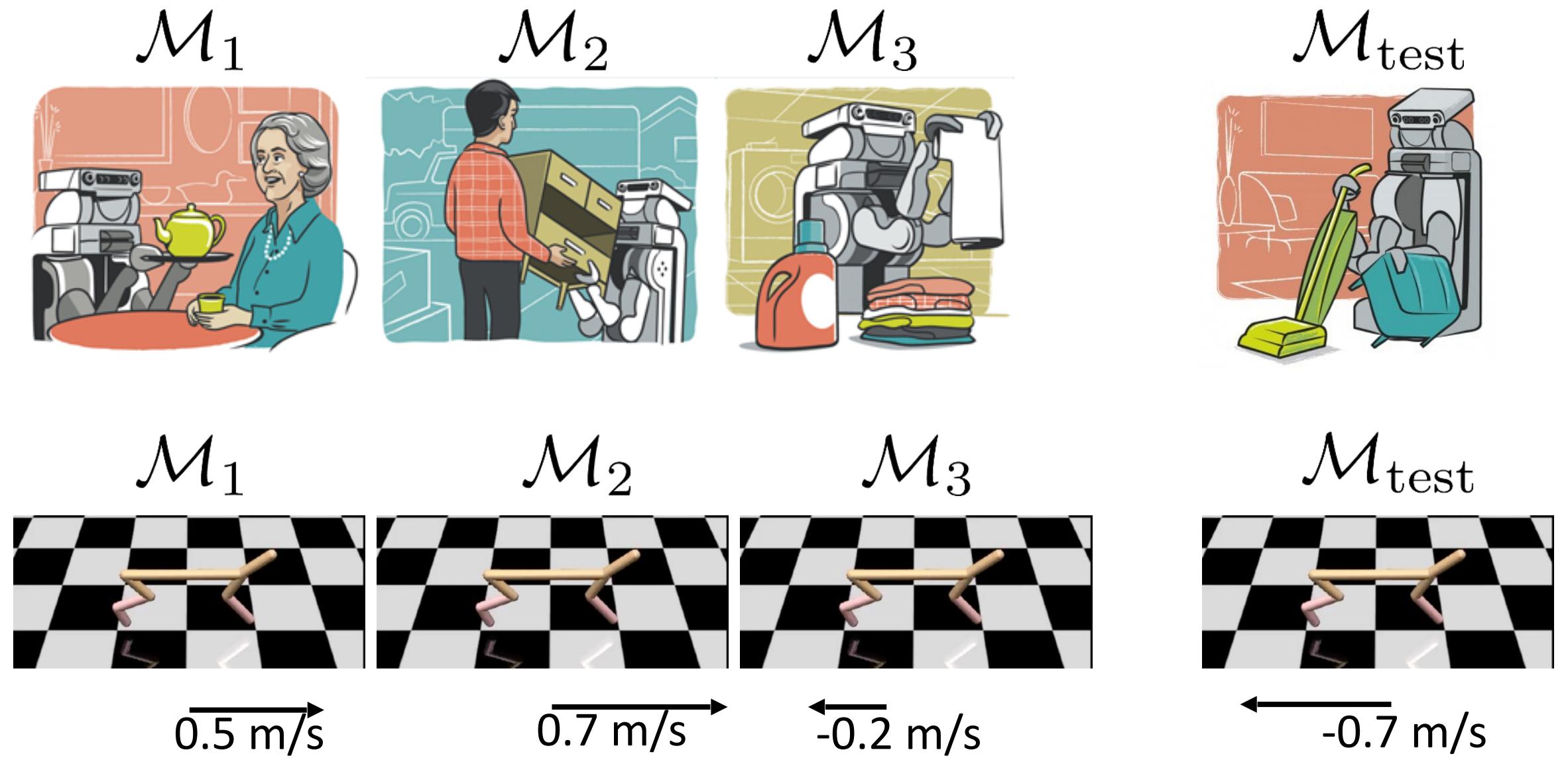
assumption: $\mathcal{M}_i \sim p(\mathcal{M})$

meta test-time:

sample $\mathcal{M}_{\text{test}} \sim p(\mathcal{M})$, get $\phi_i = f_{\theta}(\mathcal{M}_{\text{test}})$

$\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$
meta-training MDPs

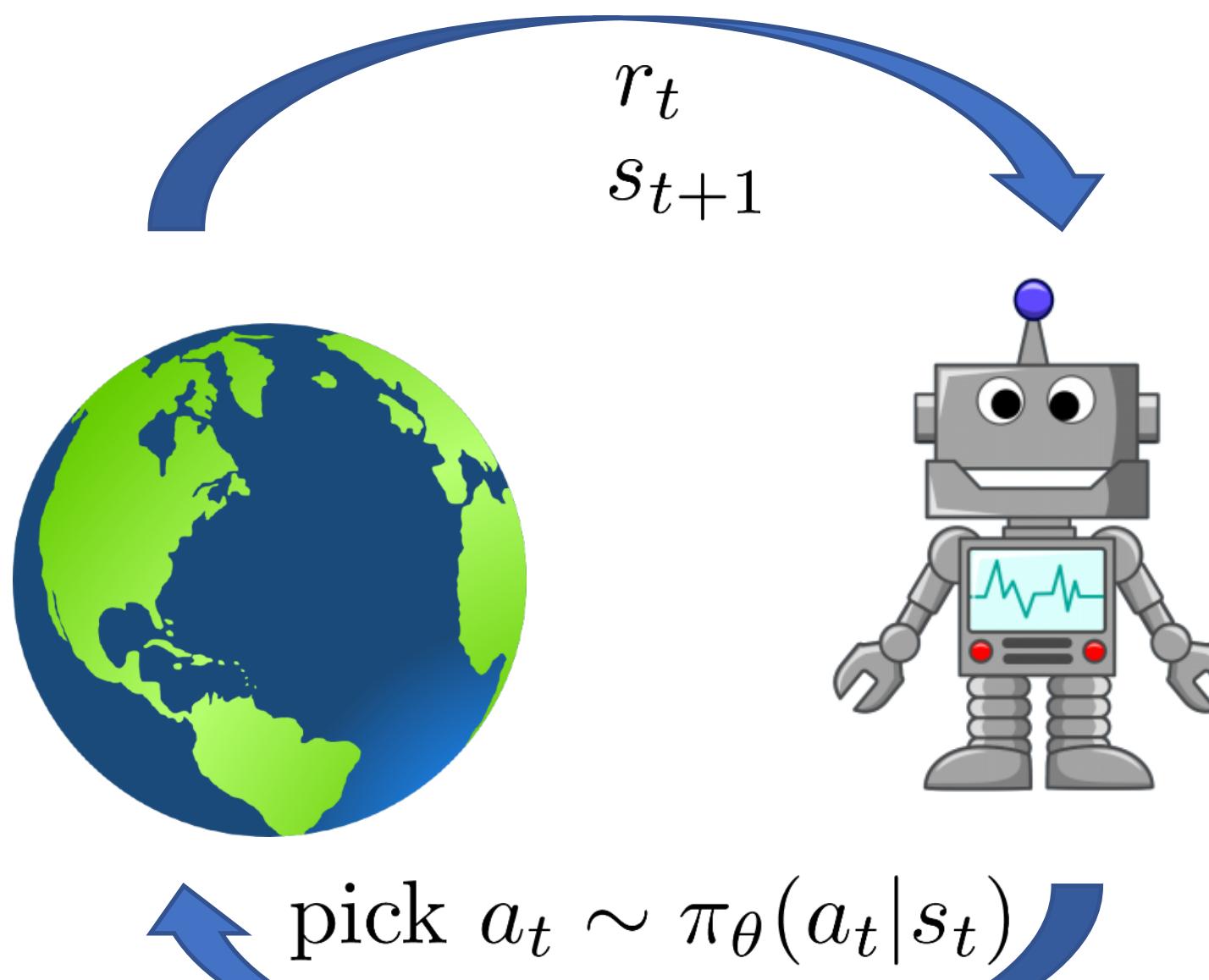
Some examples:



Meta-RL with recurrent policies

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

where $\phi_i = f_{\theta}(\mathcal{M}_i)$



use (s_t, a_t, s_{t+1}, r_t) to improve π_{θ}

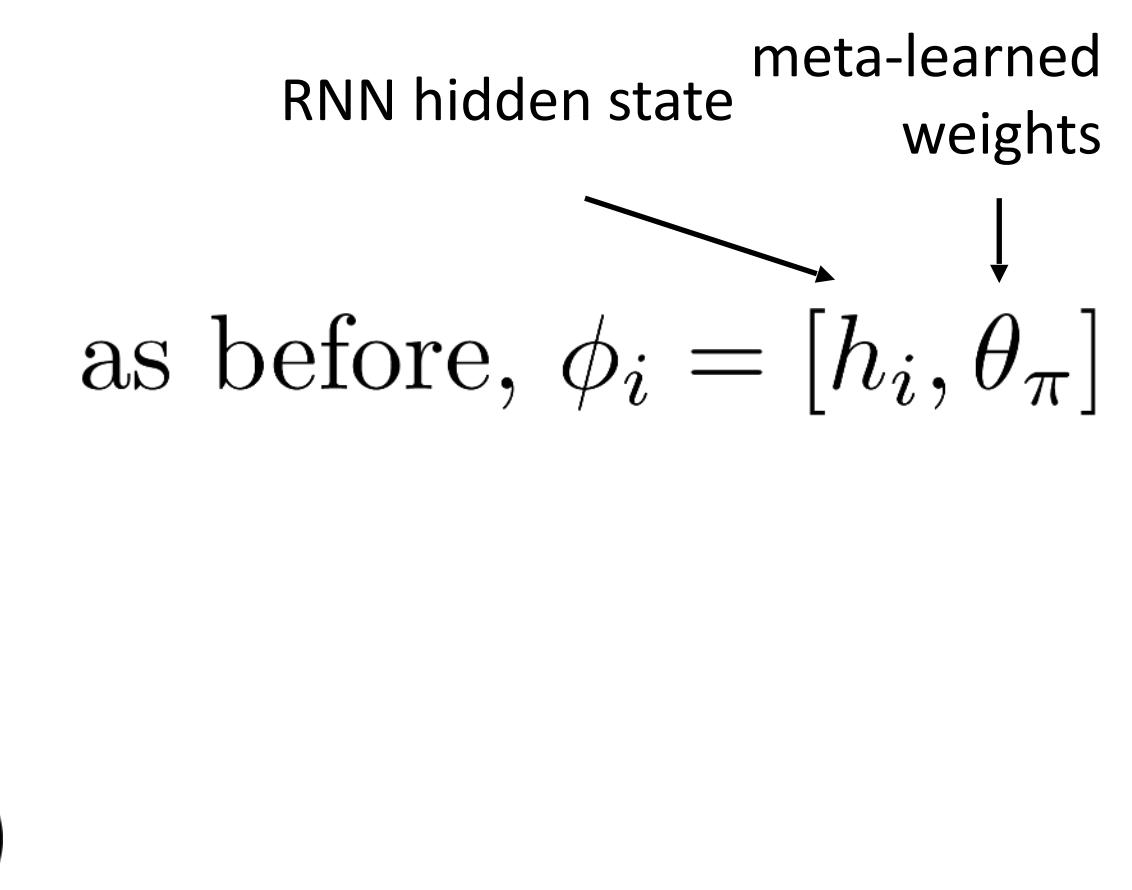
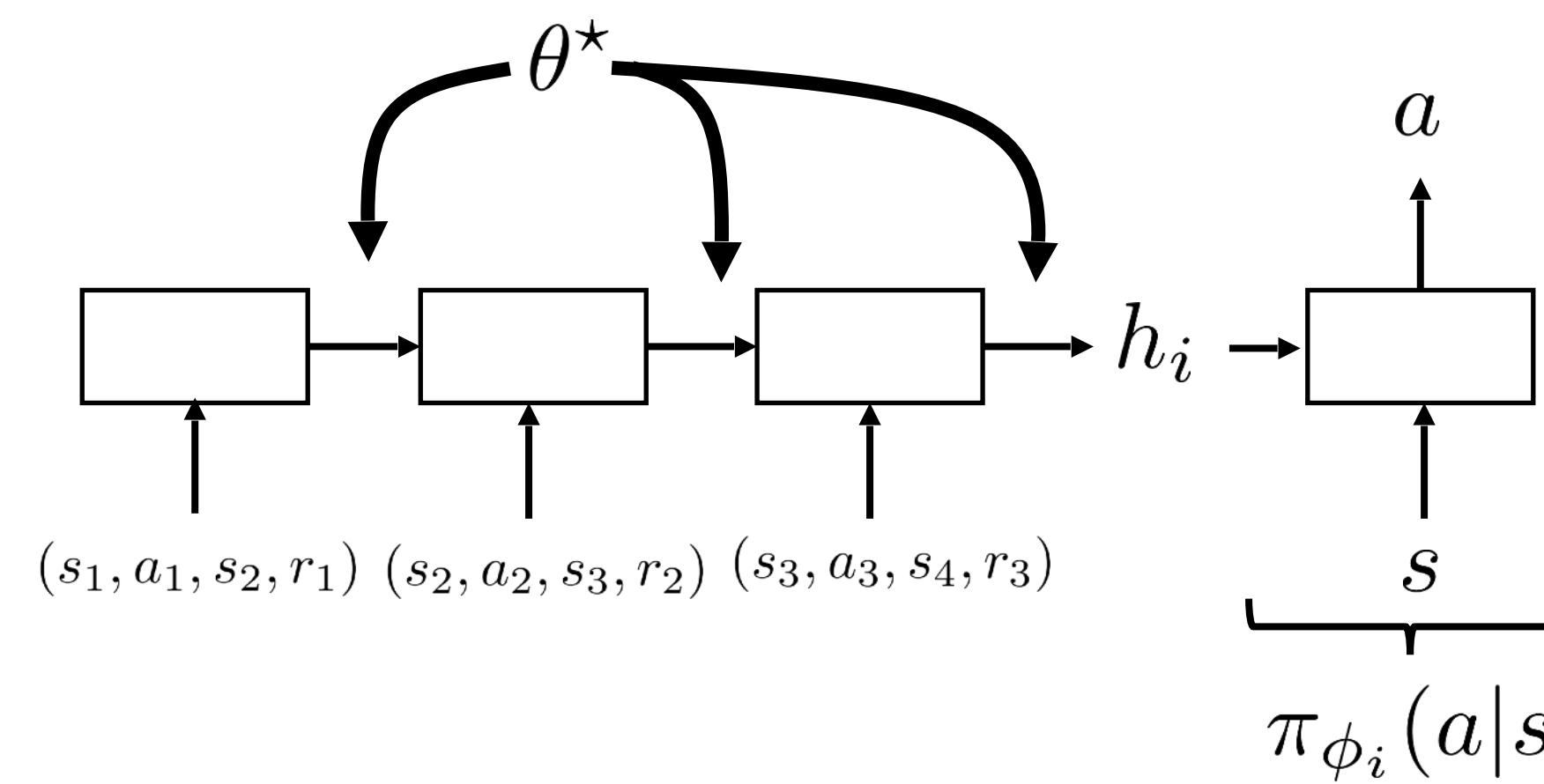
main question: how to implement $f_{\theta}(\mathcal{M}_i)$?

what should $f_{\theta}(\mathcal{M}_i)$ do?

1. improve policy with experience from \mathcal{M}_i

$$\{(s_1, a_1, s_2, r_1), \dots, (s_T, a_T, s_{T+1}, r_T)\}$$

2. (new in RL): choose how to interact, i.e. choose a_t
meta-RL must also *choose* how to *explore*!



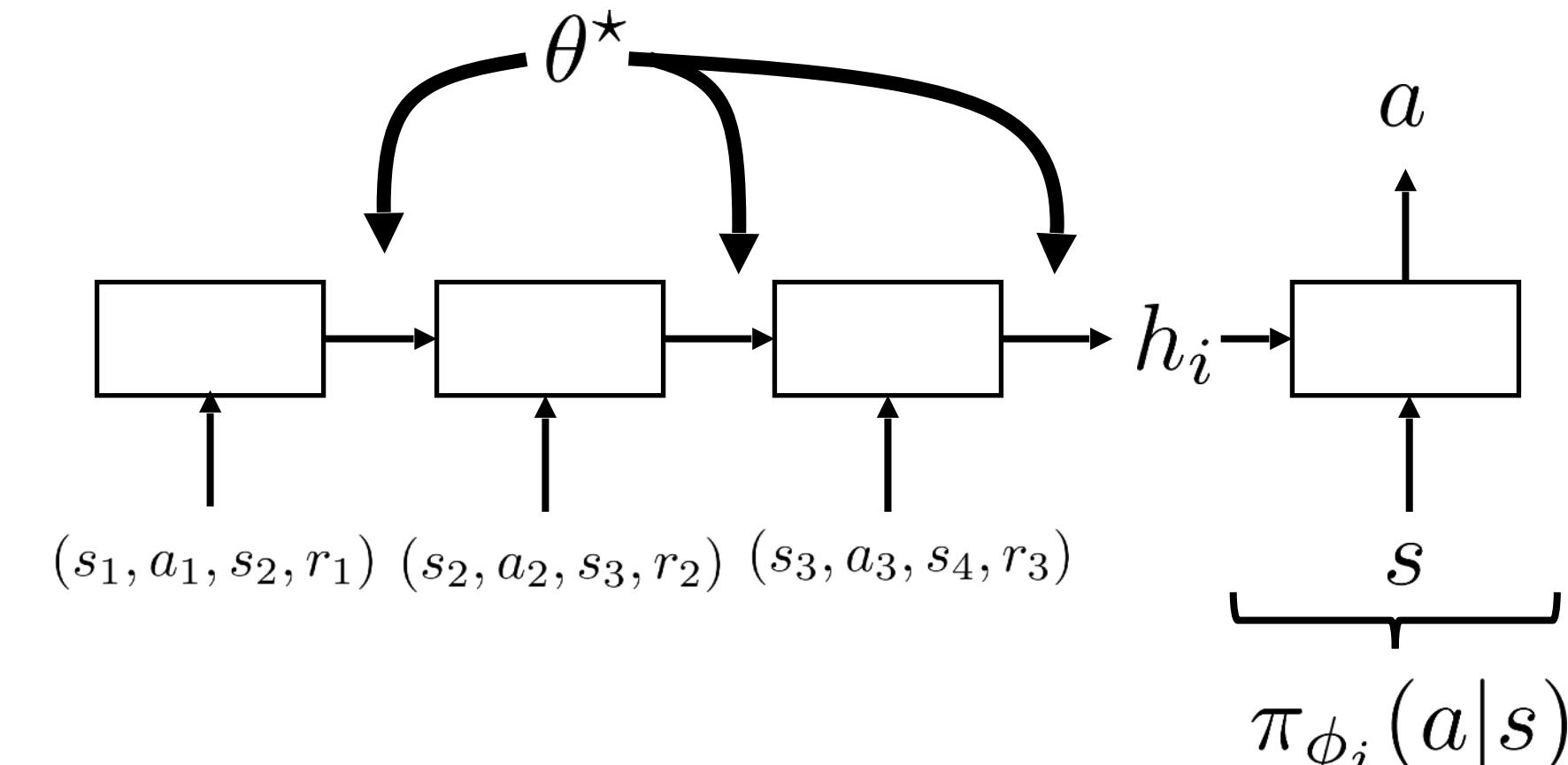
Meta-RL with recurrent policies

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

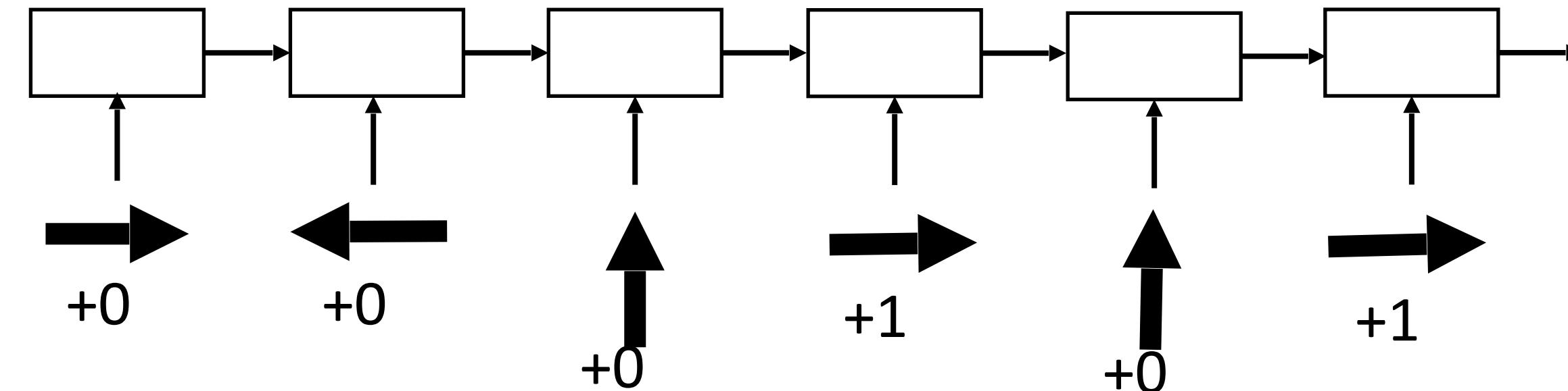
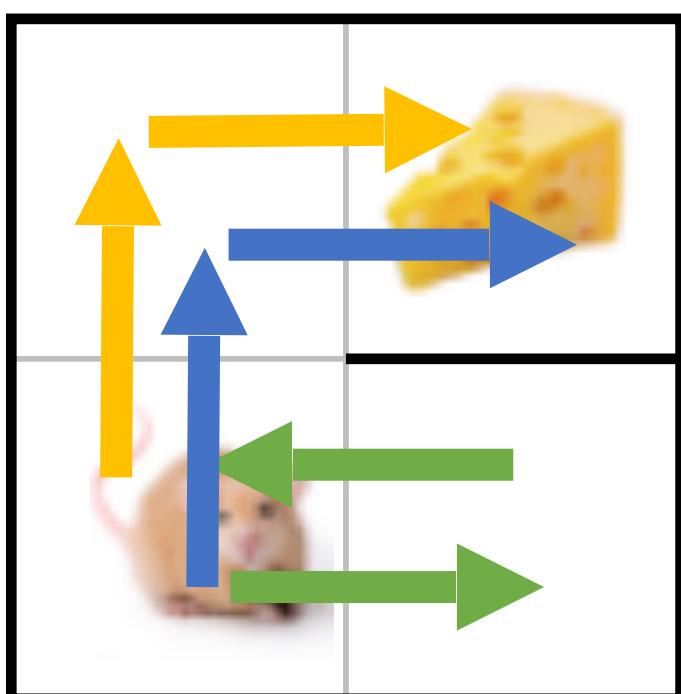
where $\phi_i = f_{\theta}(\mathcal{M}_i)$

so... we just train an RNN policy?

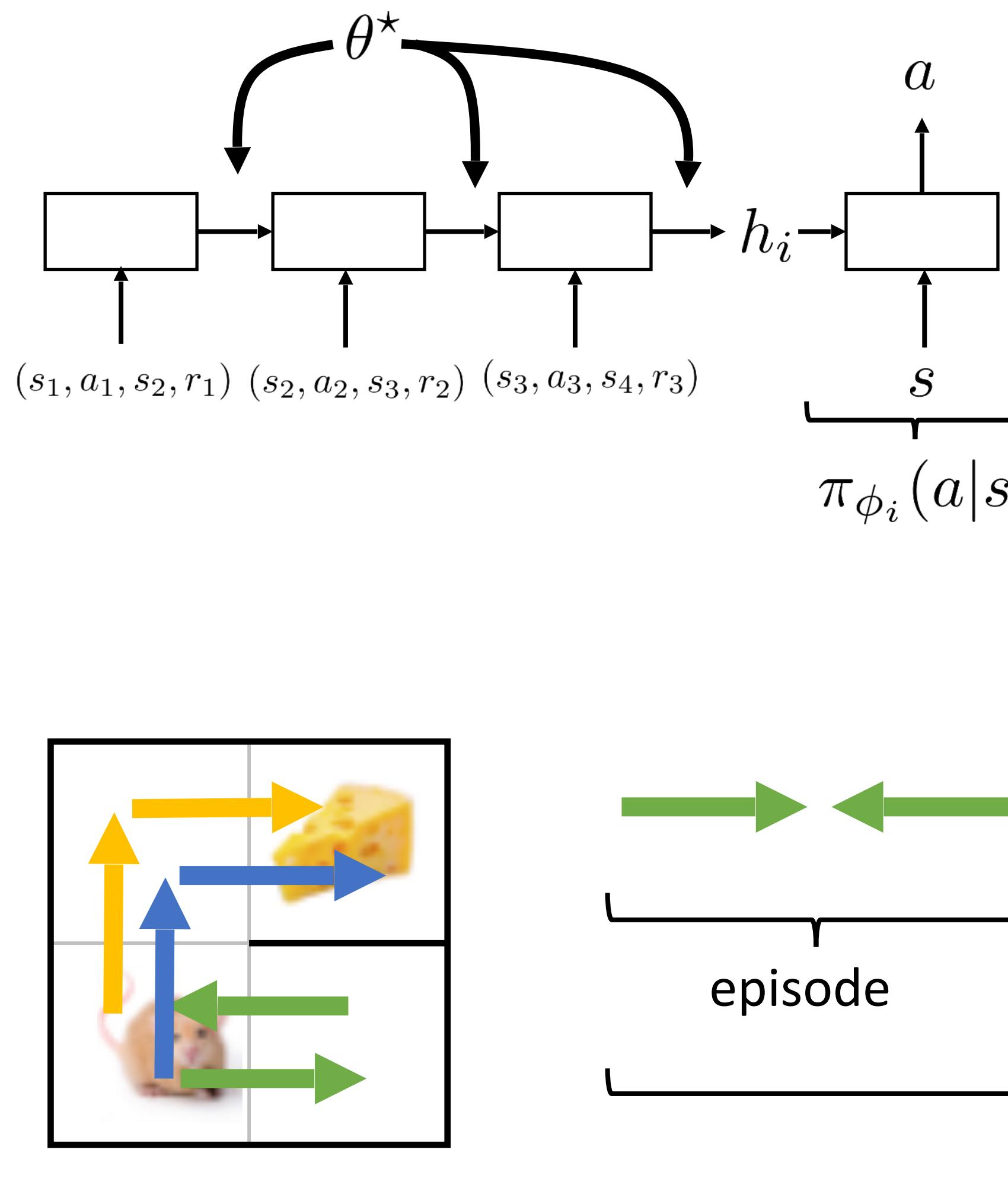
yes!



crucially, RNN hidden state is **not reset between episodes!**



Why recurrent policies learn to explore



1. improve policy with experience from \mathcal{M}_i
 $\{(s_1, a_1, s_2, r_1), \dots, (s_T, a_T, s_{T+1}, r_T)\}$
2. (new in RL): choose how to interact, i.e. choose a_t
meta-RL must also *choose* how to *explore*!

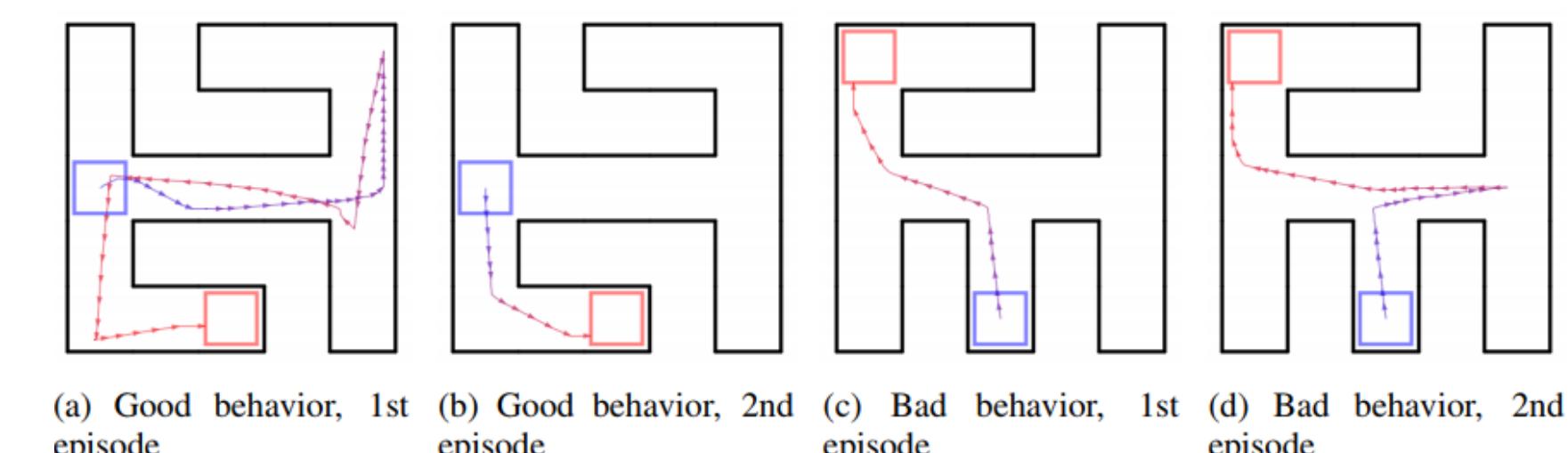
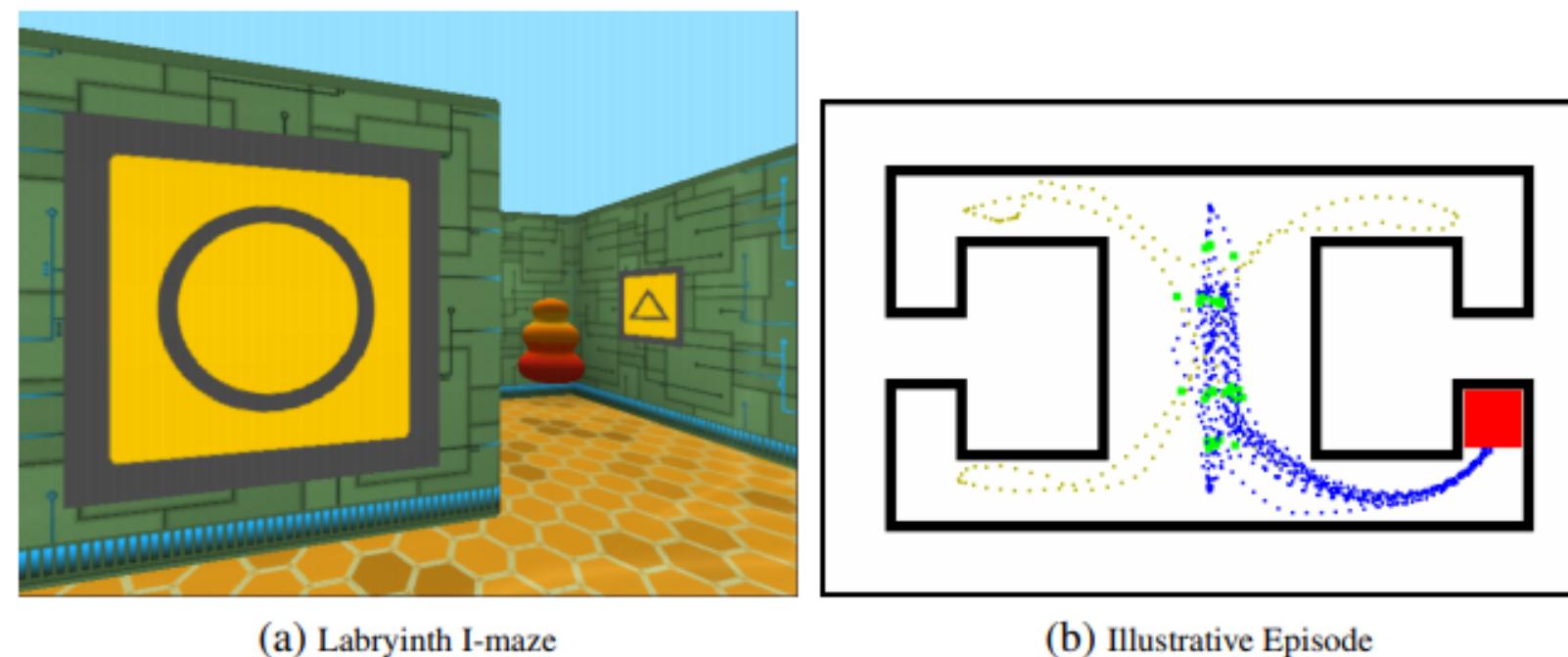
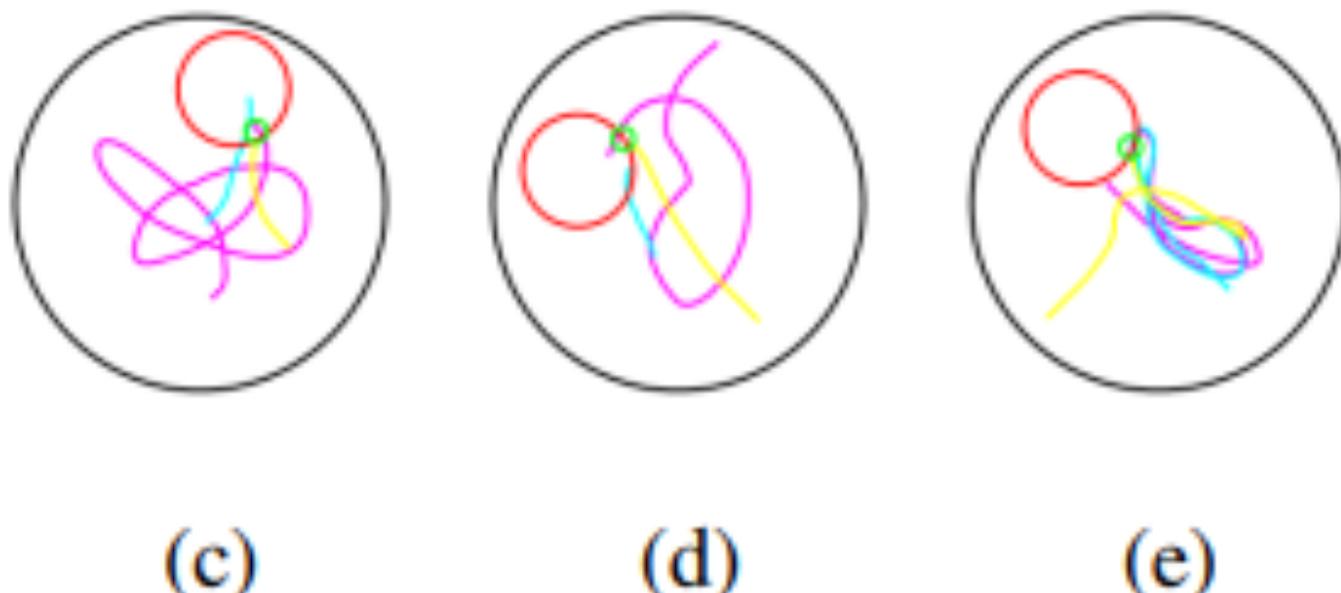
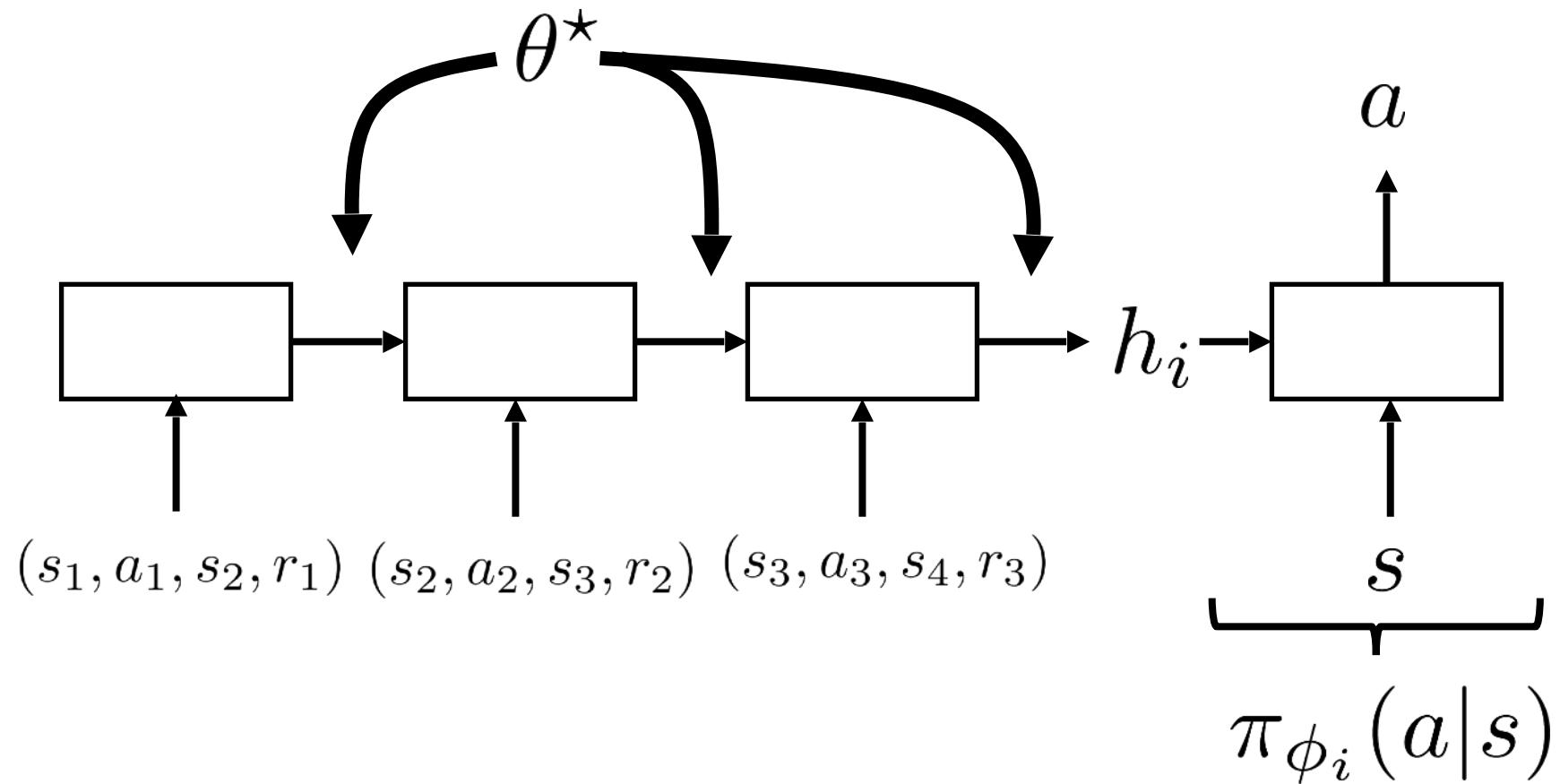
$$\theta^* = \arg \max_{\theta} E_{\pi_{\theta}} \left[\sum_{t=0}^T r(s_t, a_t) \right]$$

optimizing total reward over the entire **meta**-episode with RNN
policy **automatically** learns to explore!

Meta-RL with recurrent policies

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

where $\phi_i = f_{\theta}(\mathcal{M}_i)$

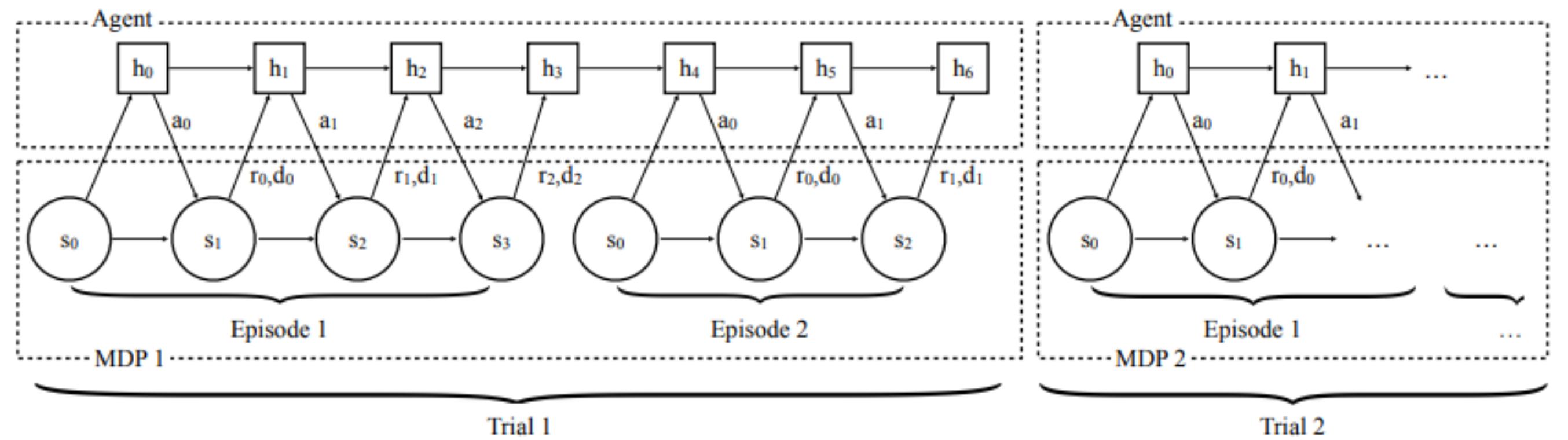


Heess, Hunt, Lillicrap, Silver. **Memory-based control with recurrent neural networks.** 2015.

Wang, Kurth-Nelson, Tirumala, Soyer, Leibo, Munos, Blundell, Kumaran, Botvinick. **Learning to Reinforcement Learning.** 2016.

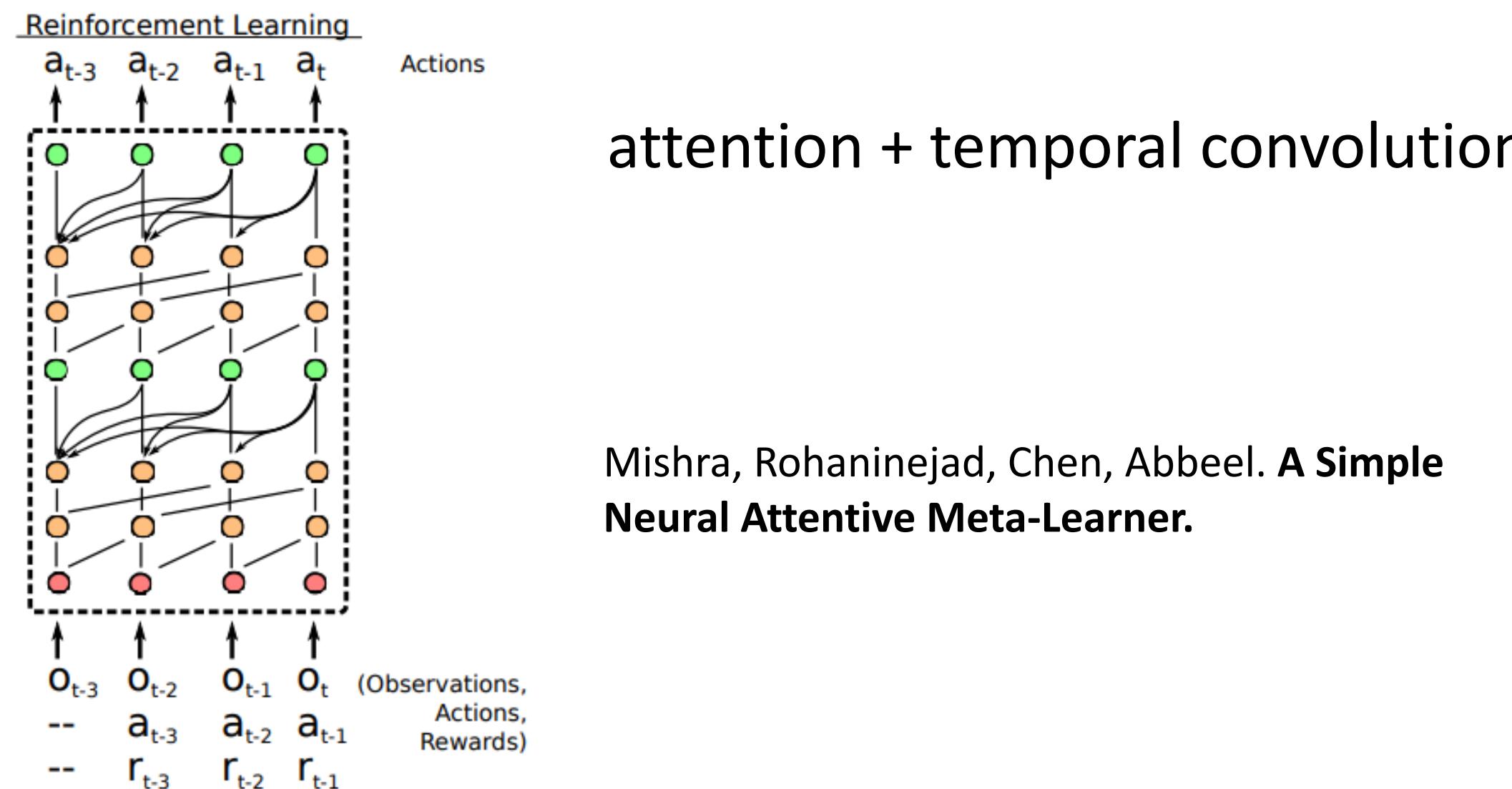
Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel. **RL2: Fast Reinforcement Learning via Slow Reinforcement Learning.** 2016.

Architectures for meta-RL

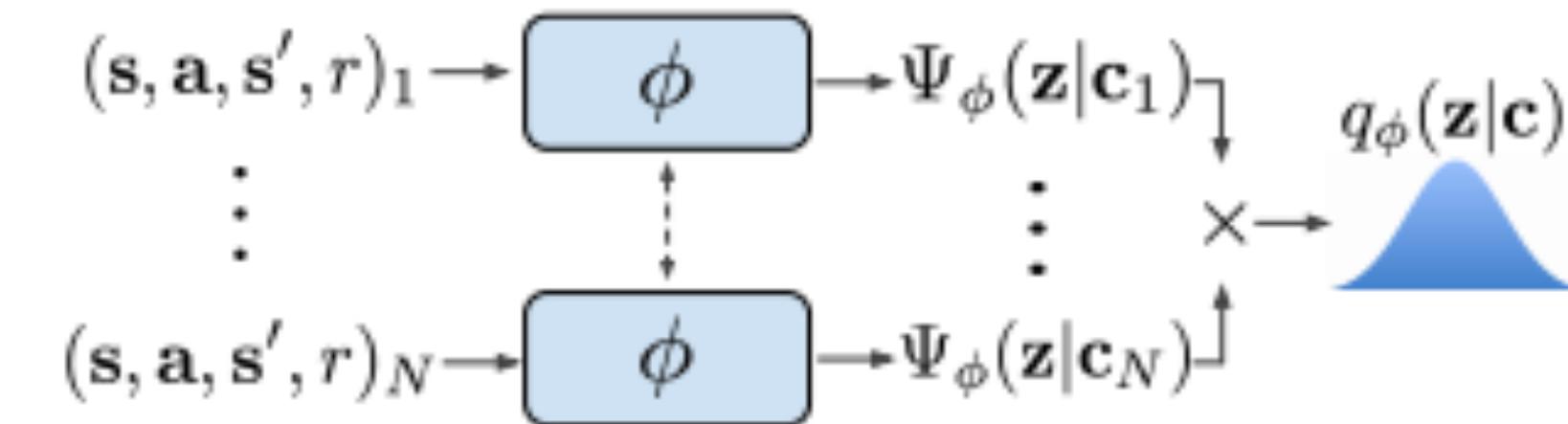


standard RNN (LSTM) architecture

Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel. **RL2: Fast Reinforcement Learning via Slow Reinforcement Learning.** 2016.



Mishra, Rohaninejad, Chen, Abbeel. **A Simple Neural Attentive Meta-Learner.**



parallel permutation-invariant context encoder

Rakelly*, Zhou*, Quillen, Finn, Levine. **Efficient Off-Policy Meta-Reinforcement learning via Probabilistic Context Variables.**

Meta-RL as an optimization problem

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

1. improve policy with experience from \mathcal{M}_i
 $\{(s_1, a_1, s_2, r_1), \dots, (s_T, a_T, s_{T+1}, r_T)\}$

where $\phi_i = f_{\theta}(\mathcal{M}_i)$

what if $f_{\theta}(\mathcal{M}_i)$ is *itself* an RL algorithm?

$$f_{\theta}(\mathcal{M}_i) = \theta + \underbrace{\alpha \nabla_{\theta} J_i(\theta)}$$

requires interacting with \mathcal{M}_i

to estimate $\nabla_{\theta} E_{\pi_{\theta}}[R(\tau)]$

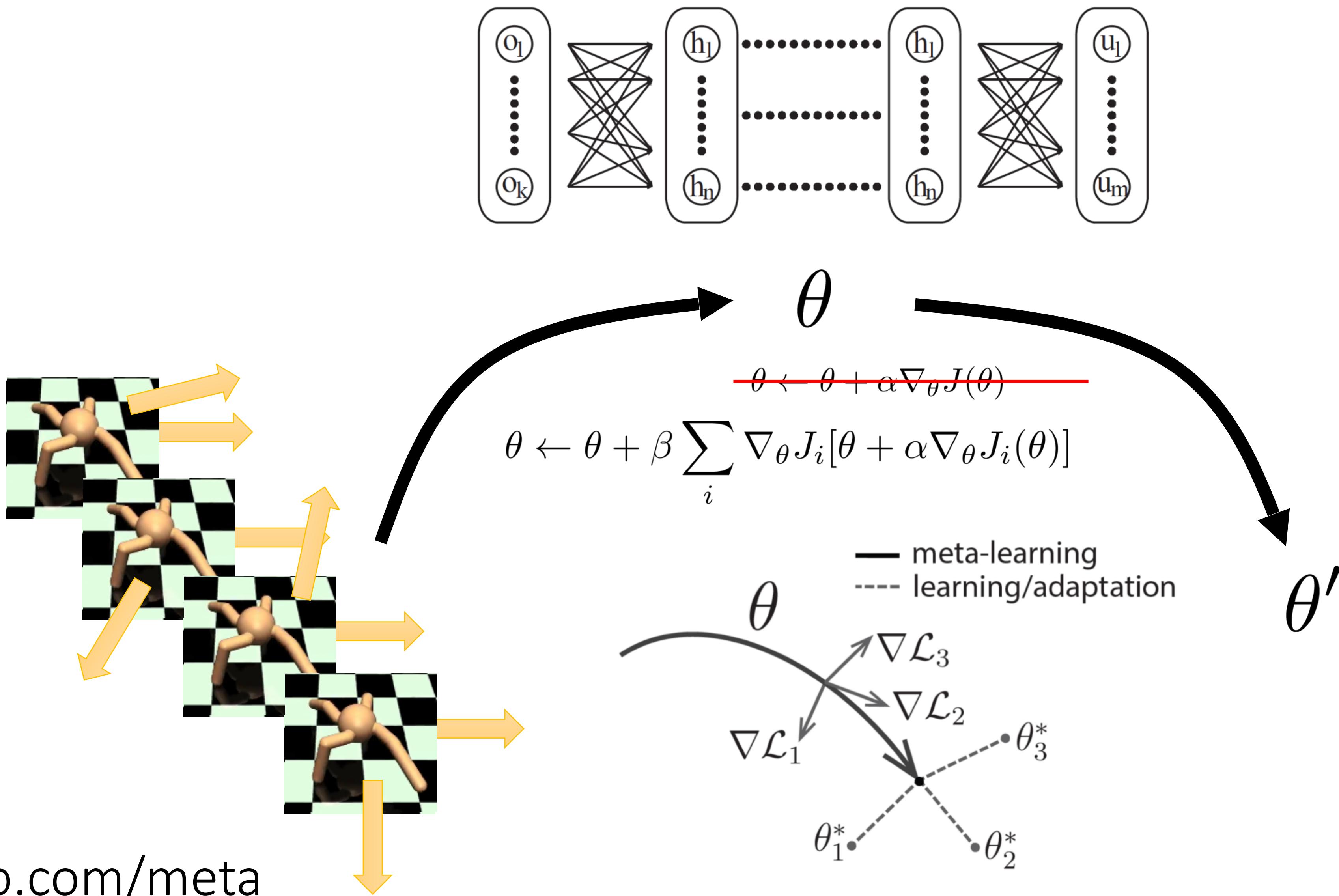
standard RL:

$$\theta^* = \arg \max_{\theta} \underbrace{E_{\pi_{\theta}(\tau)}[R(\tau)]}_{J(\theta)}$$

$$\theta^{k+1} \leftarrow \theta_k + \alpha \nabla_{\theta^k} J(\theta^k)$$

this is model-agnostic meta-learning (MAML) for RL!

MAML for RL in pictures

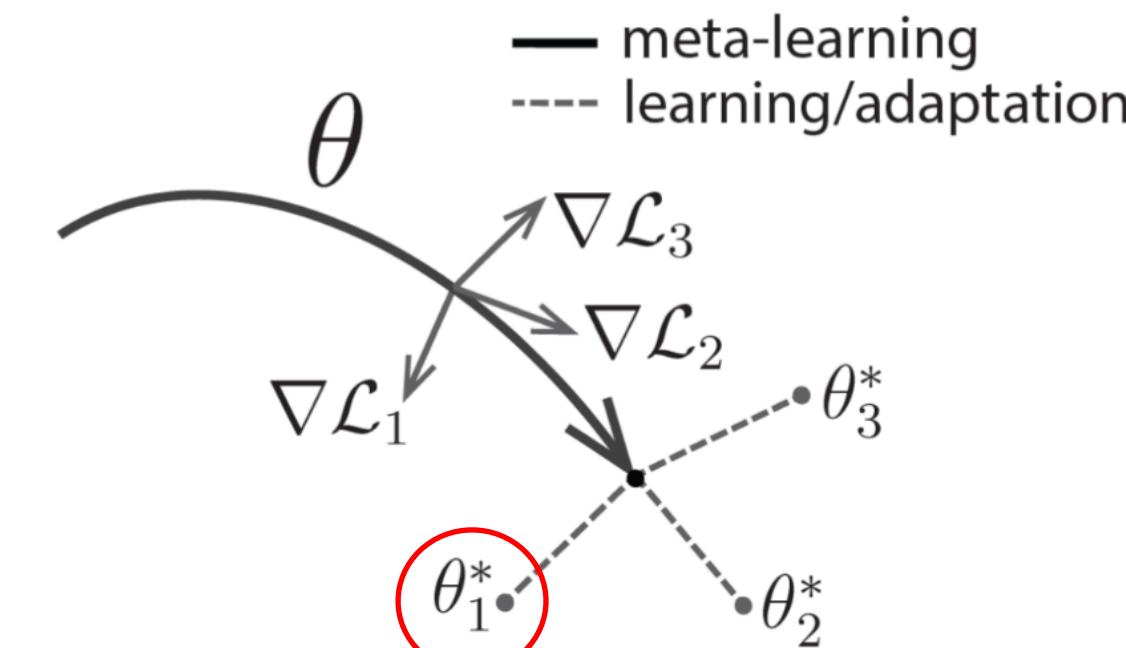
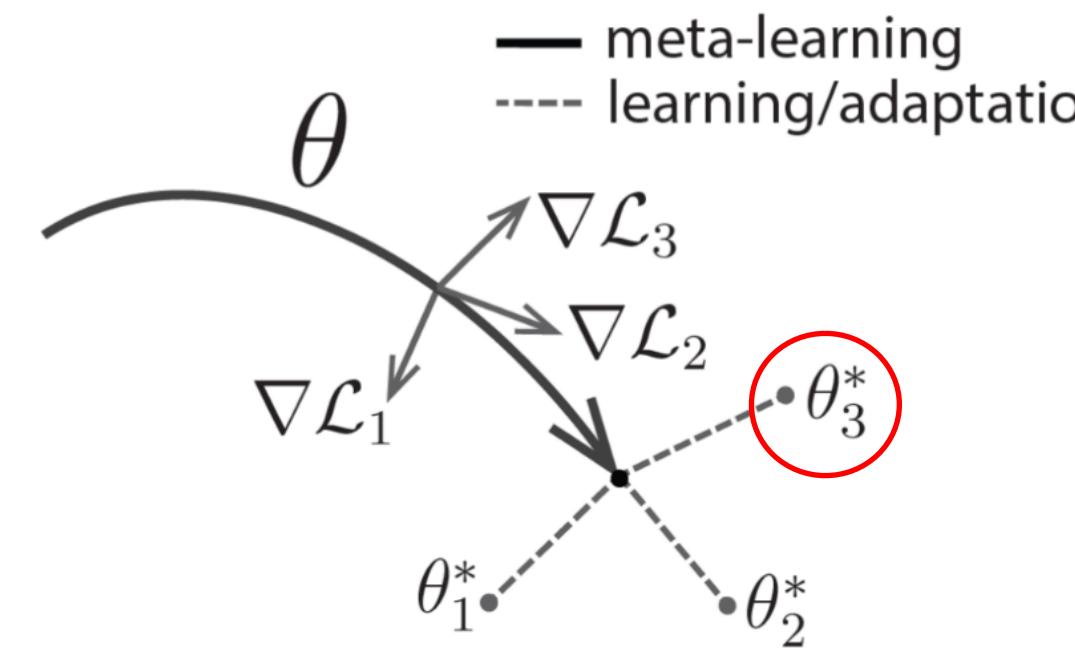
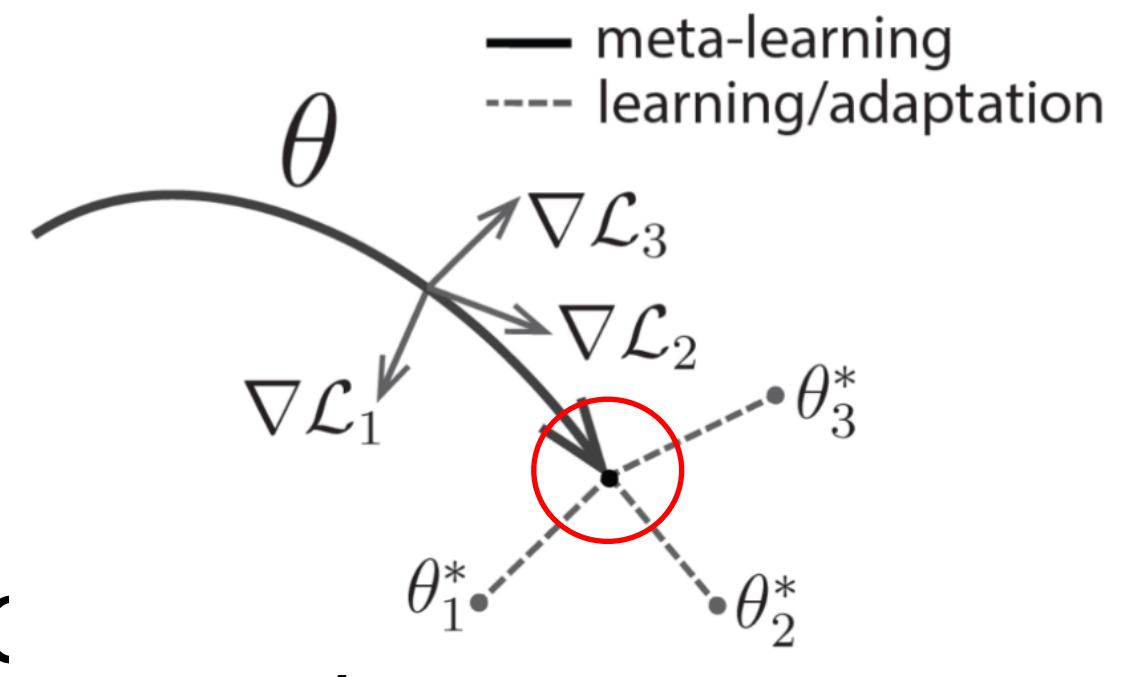
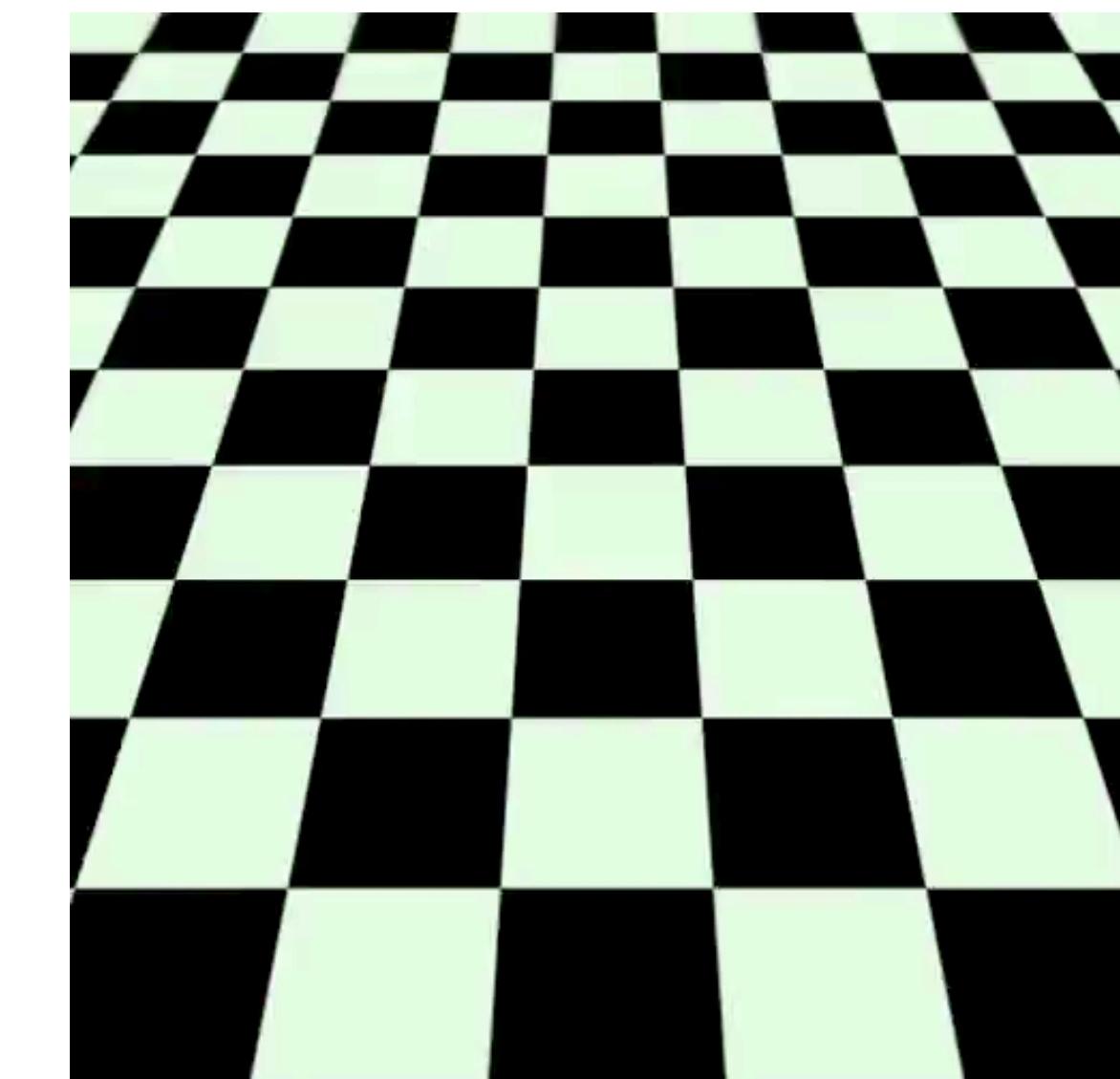
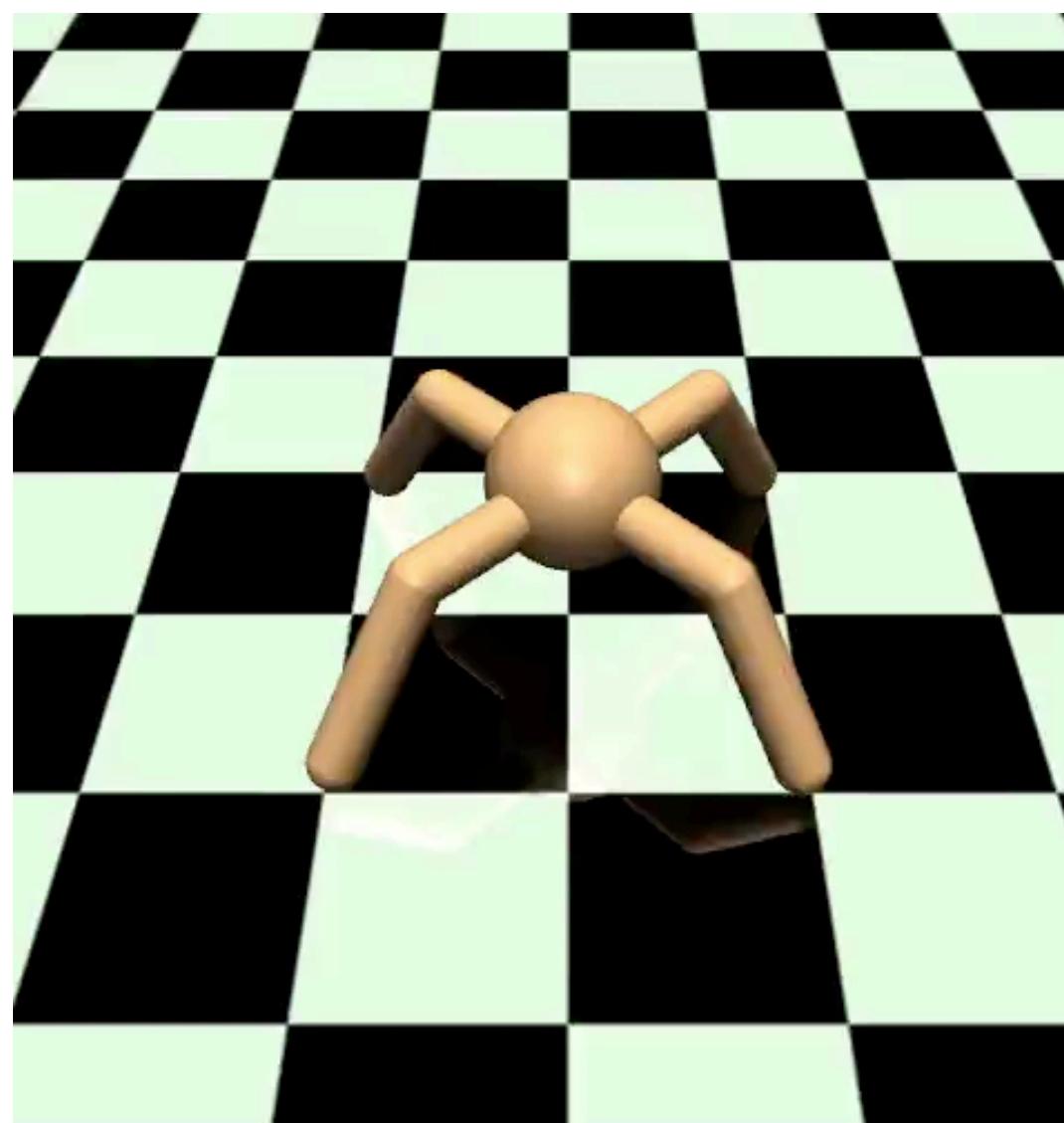
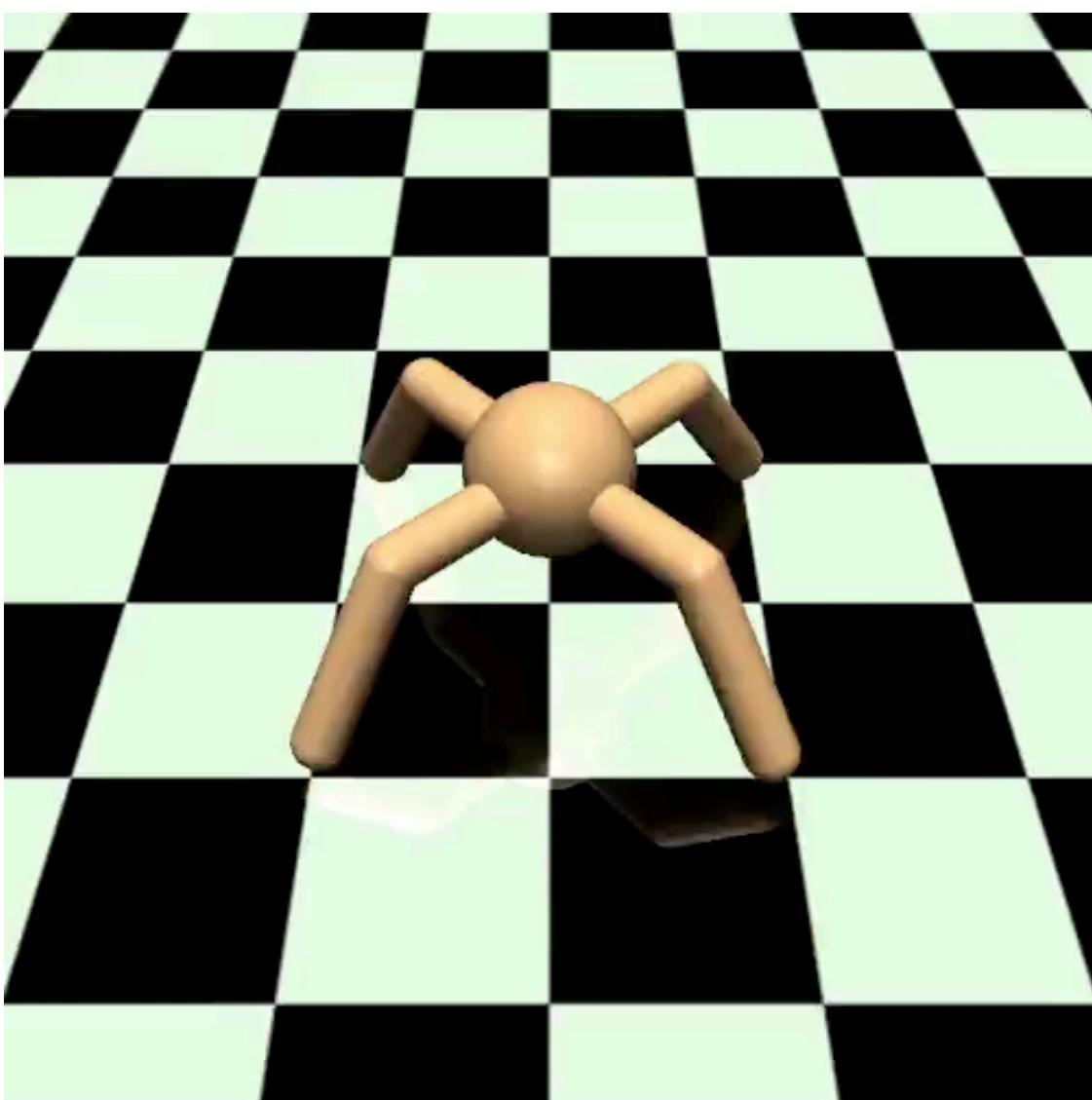


MAML for RL in videos

after MAML training

after 1 gradient step
(forward reward)

after 1 gradient step
(backward reward)



Qs: [slide 1](#)

More on MAML/gradient-based meta-learning for RL

Better MAML meta-policy gradient estimators:

- Foerster, Farquhar, Al-Shedivat, Rocktaschel, Xing, Whiteson. **DiCE: The Infinitely Differentiable Monte Carlo Estimator.**
- Rothfuss, Lee, Clavera, Asfour, Abbeel. **ProMP: Proximal Meta-Policy Search.**

Improving exploration:

- Gupta, Mendonca, Liu, Abbeel, Levine. **Meta-Reinforcement Learning of Structured Exploration Strategies.**
- Stadie*, Yang*, Houthooft, Chen, Duan, Wu, Abbeel, Sutskever. **Some Considerations on Learning to Explore via Meta-Reinforcement Learning.**

Hybrid algorithms (not necessarily gradient-based):

- Houthooft, Chen, Isola, Stadie, Wolski, Ho, Abbeel. **Evolved Policy Gradients.**
- Fernando, Sygnowski, Osindero, Wang, Schaul, Teplyashin, Sprechmann, Pirtzel, Rusu. **Meta-Learning by the Baldwin Effect.**

Meta-RL as... partially observed RL?

First: a quick primer on **partially observed** Markov decision processes (POMDPs)

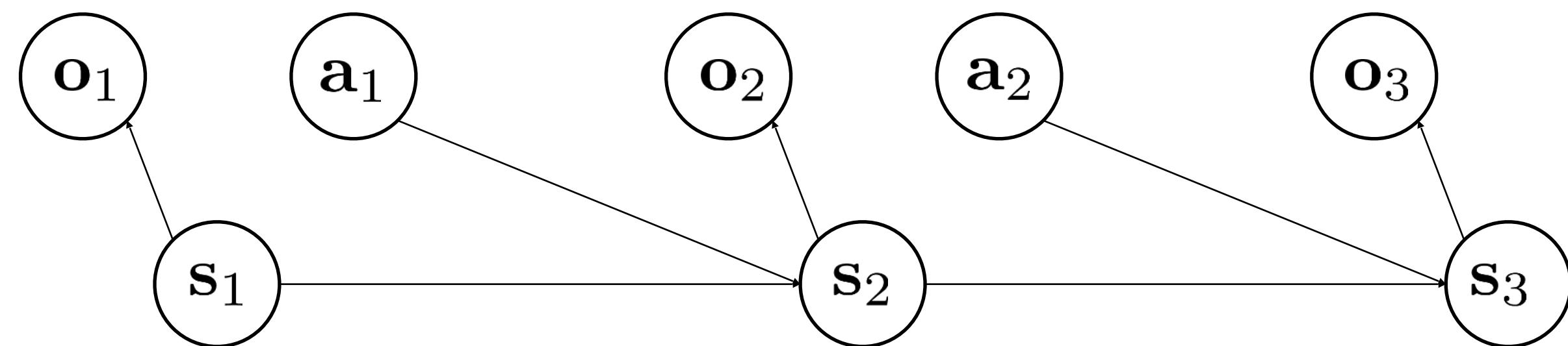
$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{E}, r\}$$

\mathcal{O} – observation space

observations $o \in \mathcal{O}$ (discrete or continuous)

\mathcal{E} – emission probability $p(o_t | s_t)$

policy must act on observations o_t !



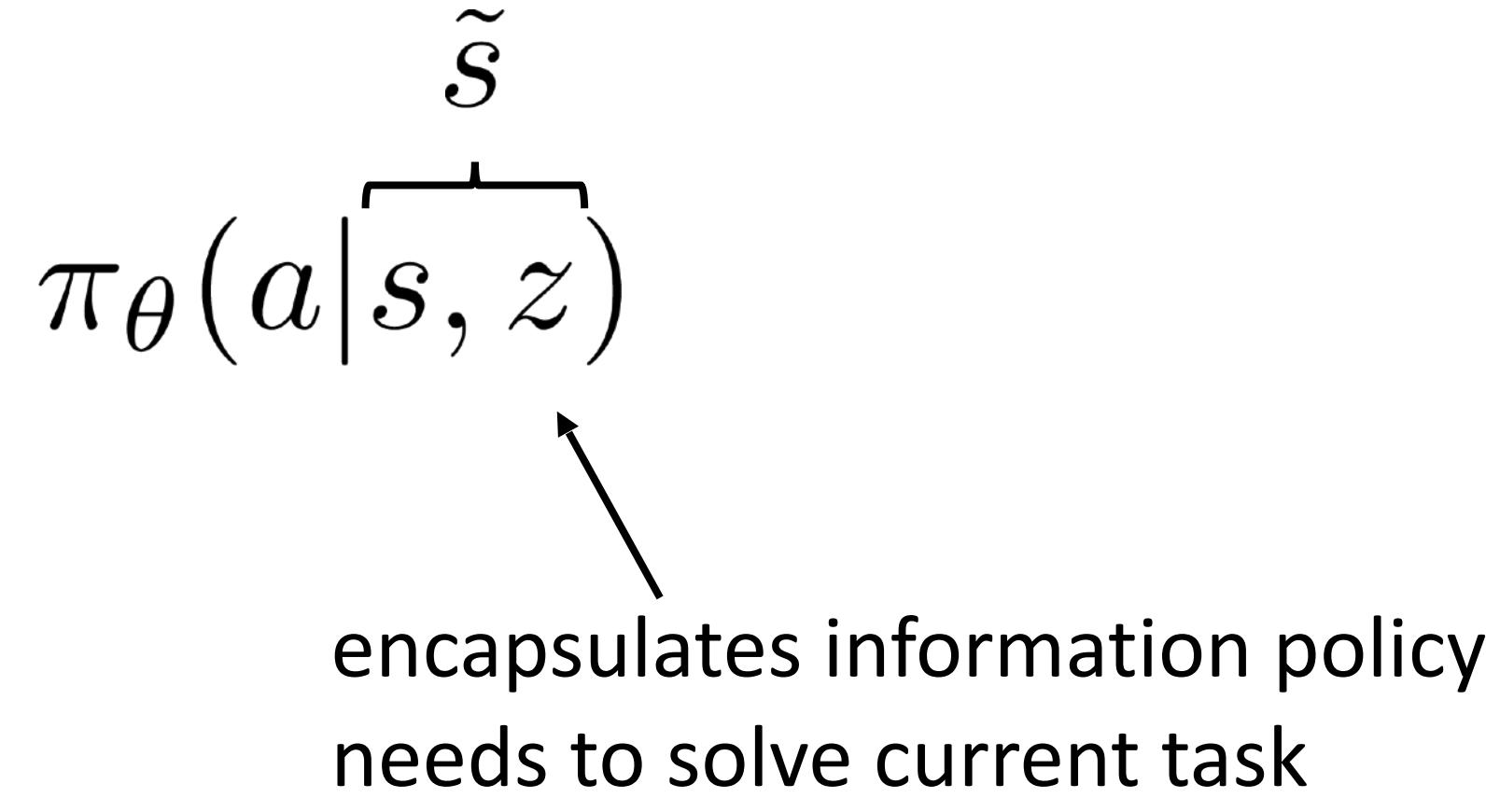
$\pi_\theta(a|o)$

typically requires *either*:

explicit state estimation, i.e. to estimate $p(s_t | o_{1:t})$

policies with memory

Meta-RL as... partially observed RL?



learning a task = inferring z
from *context* $(s_1, a_1, s_2, r_1), (s_2, a_2, s_3, r_2), \dots$

this is just a POMDP!

before: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\}$

now: $\tilde{\mathcal{M}} = \{\tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{O}}, \tilde{\mathcal{P}}, \mathcal{E}, r\}$

$\tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Z}$ $\tilde{s} = (s, z)$

$\tilde{\mathcal{O}} = \mathcal{S}$ $\tilde{o} = s$

key idea: solving the POMDP $\tilde{\mathcal{M}}$ is equivalent to meta-learning!

Meta-RL as... partially observed RL?

$$\pi_\theta(a|s, z)$$

encapsulates information policy
needs to solve current task

learning a task = inferring z

from *context* $(s_1, a_1, s_2, r_1), (s_2, a_2, s_3, r_2), \dots$

exploring via posterior sampling with latent context

- C 1. sample $z \sim \hat{p}(z_t|s_{1:t}, a_{1:t}, r_{1:t})$

some approximate posterior
(e.g., variational)

2. act according to $\pi_\theta(a|s, z)$ to collect more data

act as though z was correct!

this is just a POMDP!

typically requires *either*:

explicit state estimation, i.e. to estimate $p(s_t|o_{1:t})$

policies with memory

need to estimate $p(z_t|s_{1:t}, a_{1:t}, r_{1:t})$

this is not optimal!

why?

but it's pretty good, both in theory and in practice!

Variational inference for meta-RL

policy: $\pi_\theta(a_t|s_t, z_t)$

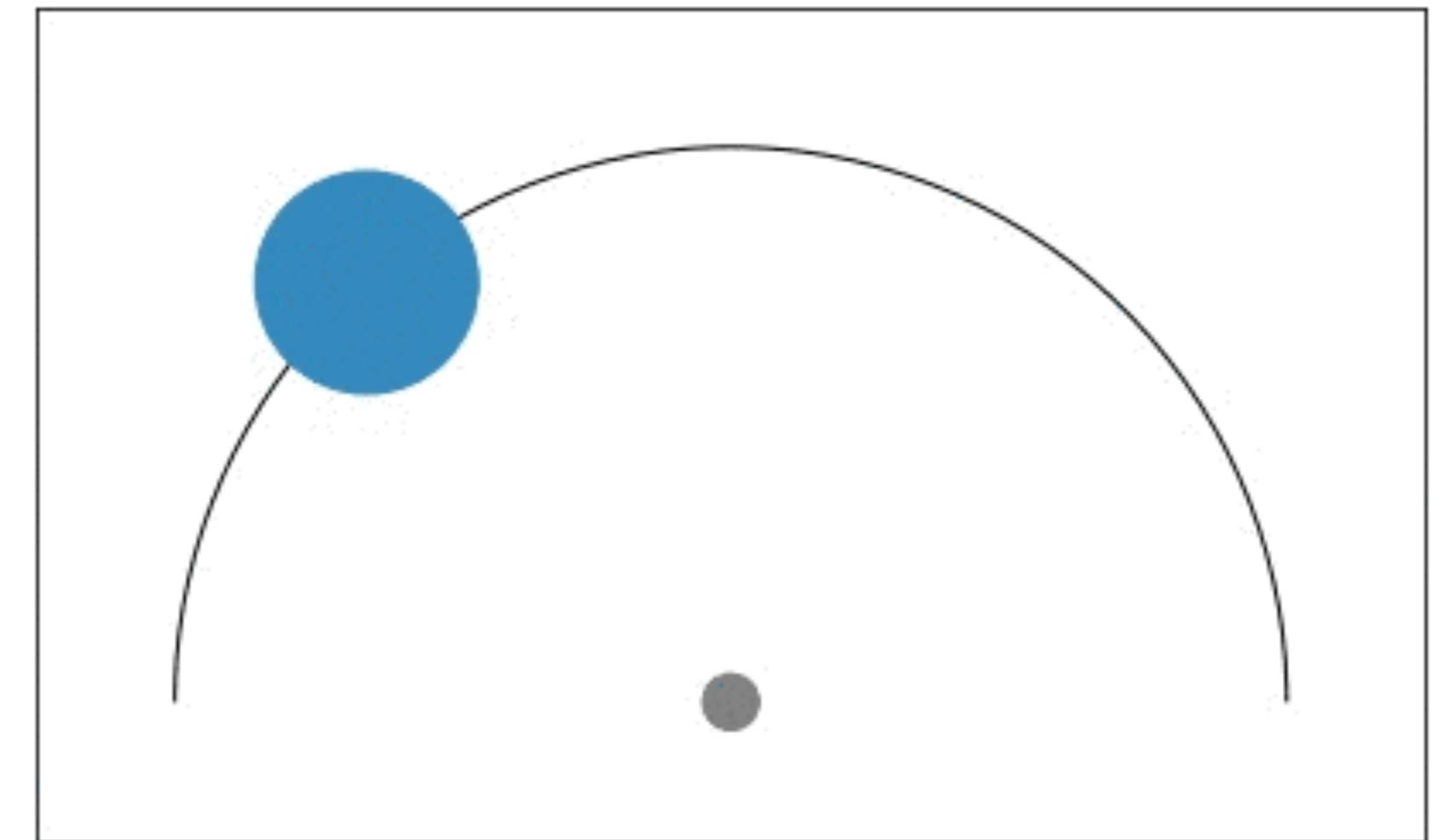
inference network: $q_\phi(z_t|s_1, a_1, r_1, \dots, s_t, a_t, r_t)$

$$(\theta, \phi) = \arg \max_{\theta, \phi} \frac{1}{N} \sum_{i=1}^n E_{z \sim q_\phi, \tau \sim \pi_\theta} [R_i(\tau) - D_{\text{KL}}(q(z|\dots) \| p(z))]$$

maximize post-update reward
(same as standard meta-RL)

stay close to prior

$$z_t \sim q_\phi(z_t|s_1, a_1, r_1, \dots, s_t, a_t, r_t)$$



conceptually *very* similar to RNN meta-RL, but with stochastic z

stochastic z enables exploration via *posterior sampling*

Qs: slido.com/meta

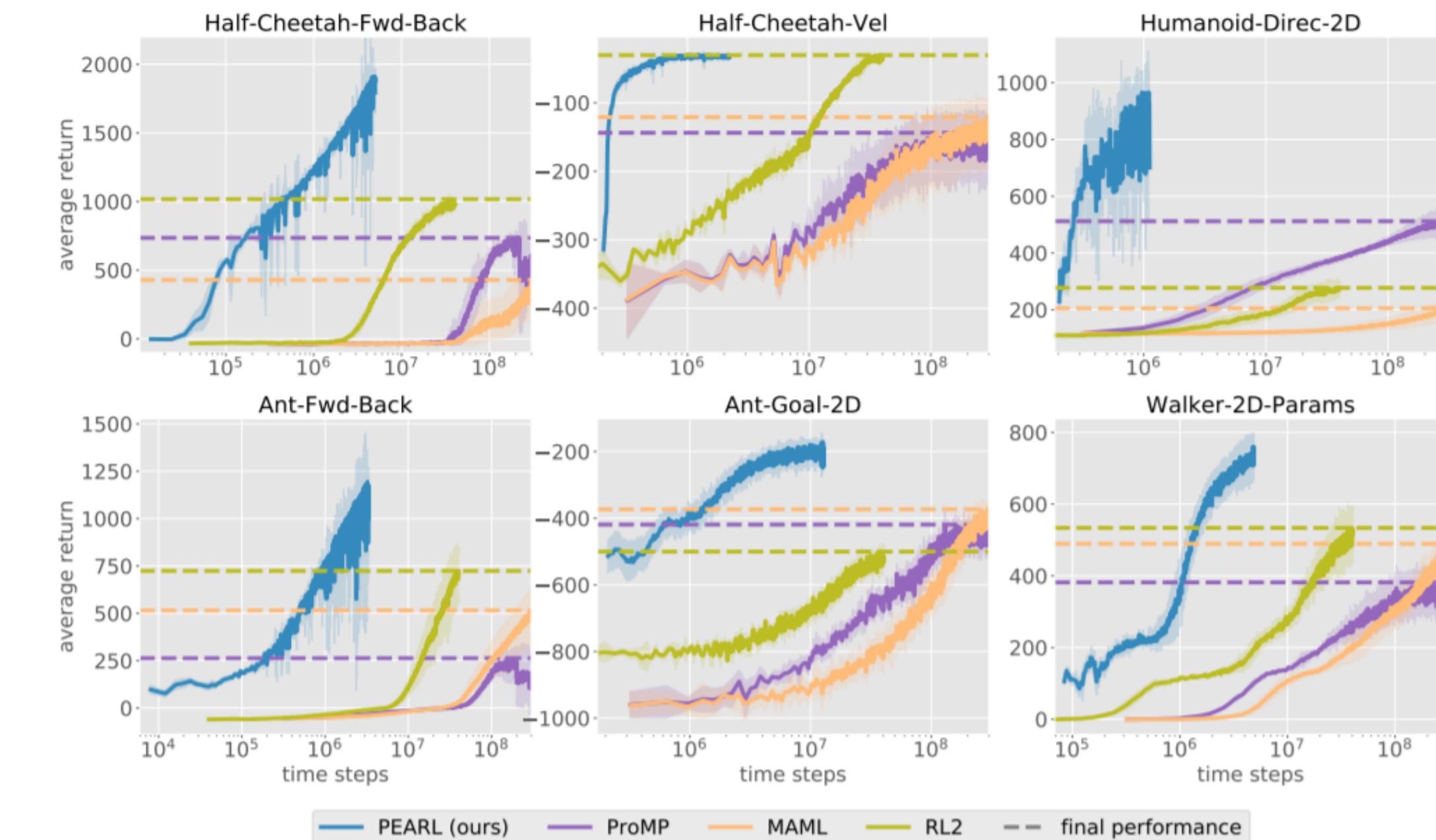
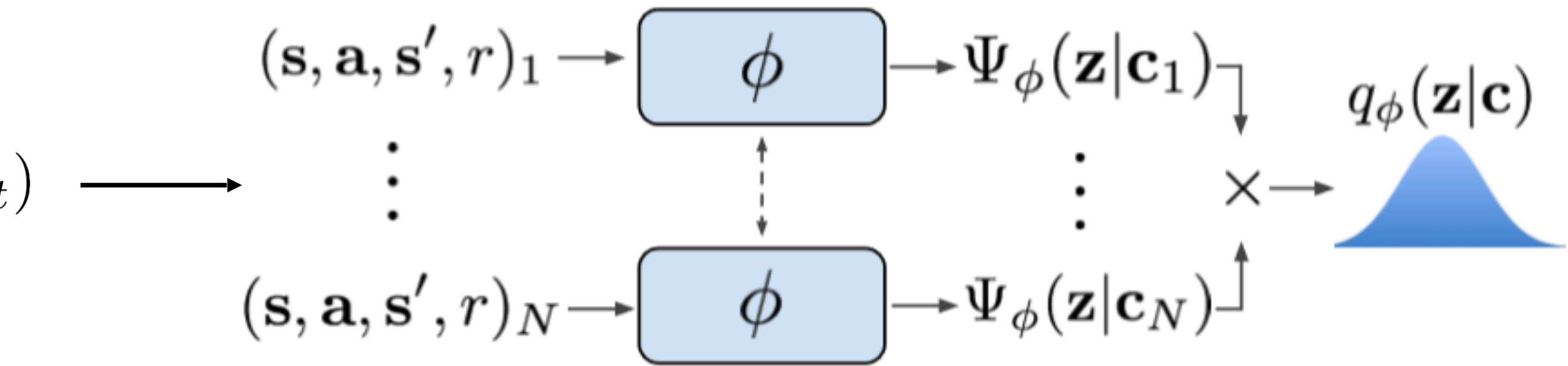
Specific instantiation: PEARL

policy: $\pi_\theta(a_t|s_t, z_t)$

inference network: $q_\phi(z_t|s_1, a_1, r_1, \dots, s_t, a_t, r_t)$

$$(\theta, \phi) = \arg \max_{\theta, \phi} \frac{1}{N} \sum_{i=1}^n E_{z \sim q_\phi, \tau \sim \pi_\theta} [R_i(\tau) - D_{\text{KL}}(q(z|\dots)||p(z))]$$

↑
perform maximization using soft actor-critic (SAC),
state-of-the-art off-policy RL algorithm



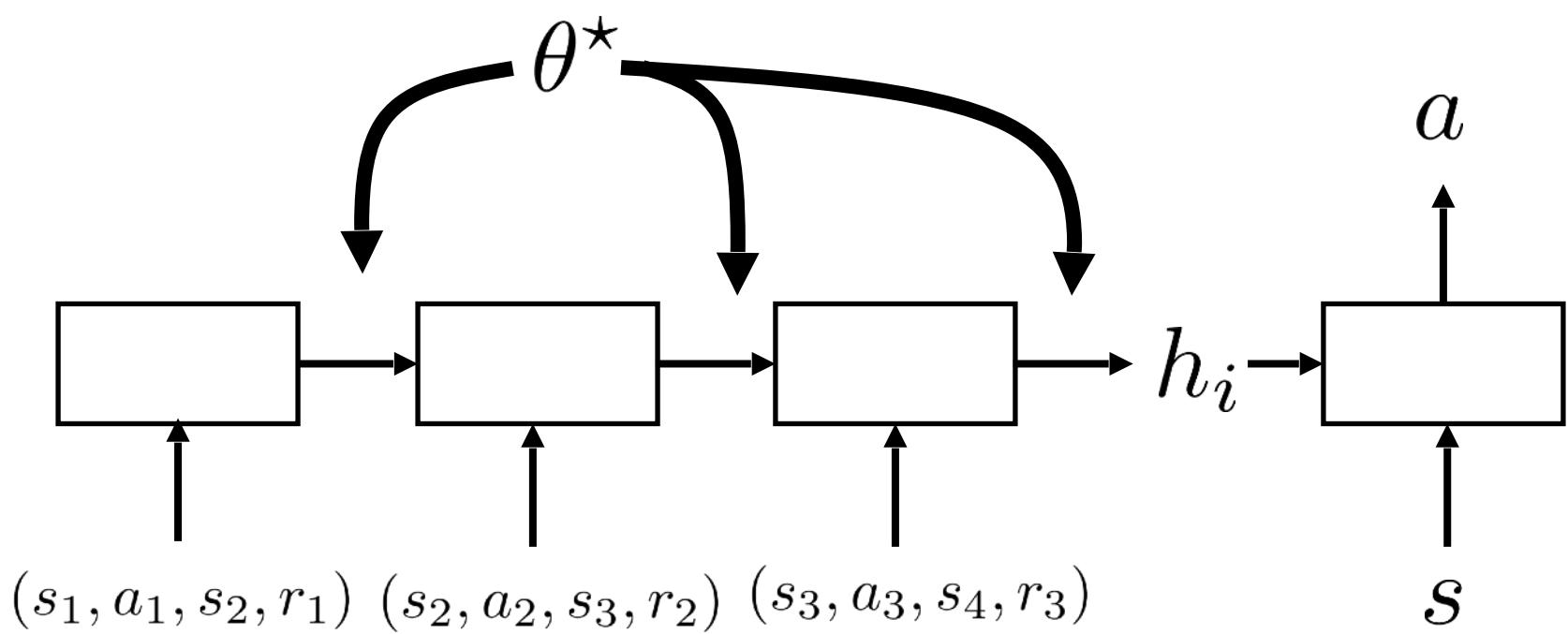
Qs: slido.com/meta

References on meta-RL, inference, and POMDPs

- Rakelly*, Zhou*, Quillen, Finn, Levine. **Efficient Off-Policy Meta-Reinforcement learning via Probabilistic Context Variables.** ICML 2019.
- Zintgraf, Igl, Shiarlis, Mahajan, Hofmann, Whiteson. **Variational Task Embeddings for Fast Adaptation in Deep Reinforcement Learning.**
- Humplik, Galashov, Hasenclever, Ortega, Teh, Heess. **Meta reinforcement learning as task inference.**

The three perspectives on meta-RL

Perspective 1: just RNN it



Perspective 2: bi-level optimization

$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

MAML for RL

Perspective 3: it's an inference problem!

$$\pi_\theta(a|s, z) \quad z_t \sim p(z_t|s_{1:t}, a_{1:t}, r_{1:t})$$

↑
everything needed to solve task

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

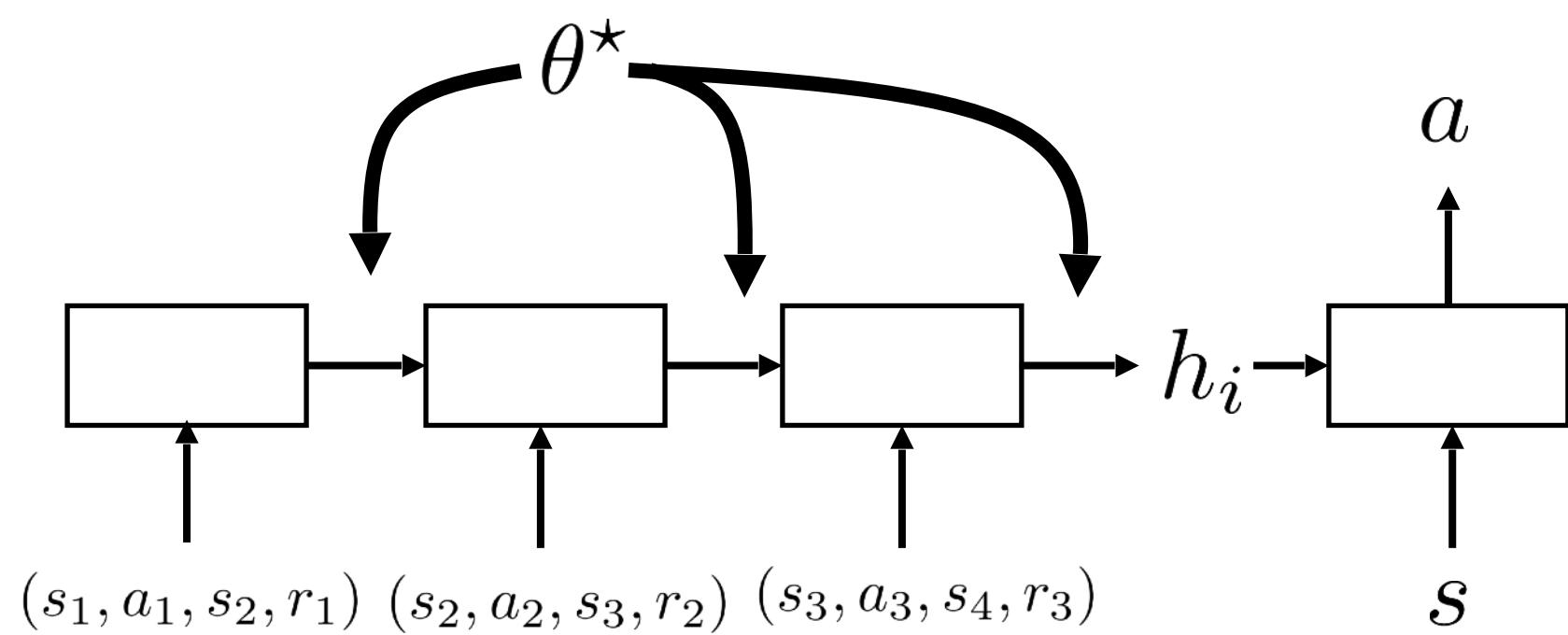
where $\phi_i = f_\theta(\mathcal{M}_i)$

what should $f_\theta(\mathcal{M}_i)$ do?

1. improve policy with experience from \mathcal{M}_i
 $\{(s_1, a_1, s_2, r_1), \dots, (s_T, a_T, s_{T+1}, r_T)\}$
2. (new in RL): choose how to interact, i.e. choose a_t
meta-RL must also *choose* how to *explore*!

The three perspectives on meta-RL

Perspective 1: just RNN it



- + conceptually simple
- + relatively easy to apply
- vulnerable to meta-overfitting
- challenging to optimize in practice

Perspective 2: bi-level optimization

$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

MAML for RL

- + good extrapolation (“consistent”)
- + conceptually elegant
- complex, requires many samples

Perspective 3: it's an inference problem!

$$\pi_\theta(a|s, z) \quad z_t \sim p(z_t|s_{1:t}, a_{1:t}, r_{1:t})$$

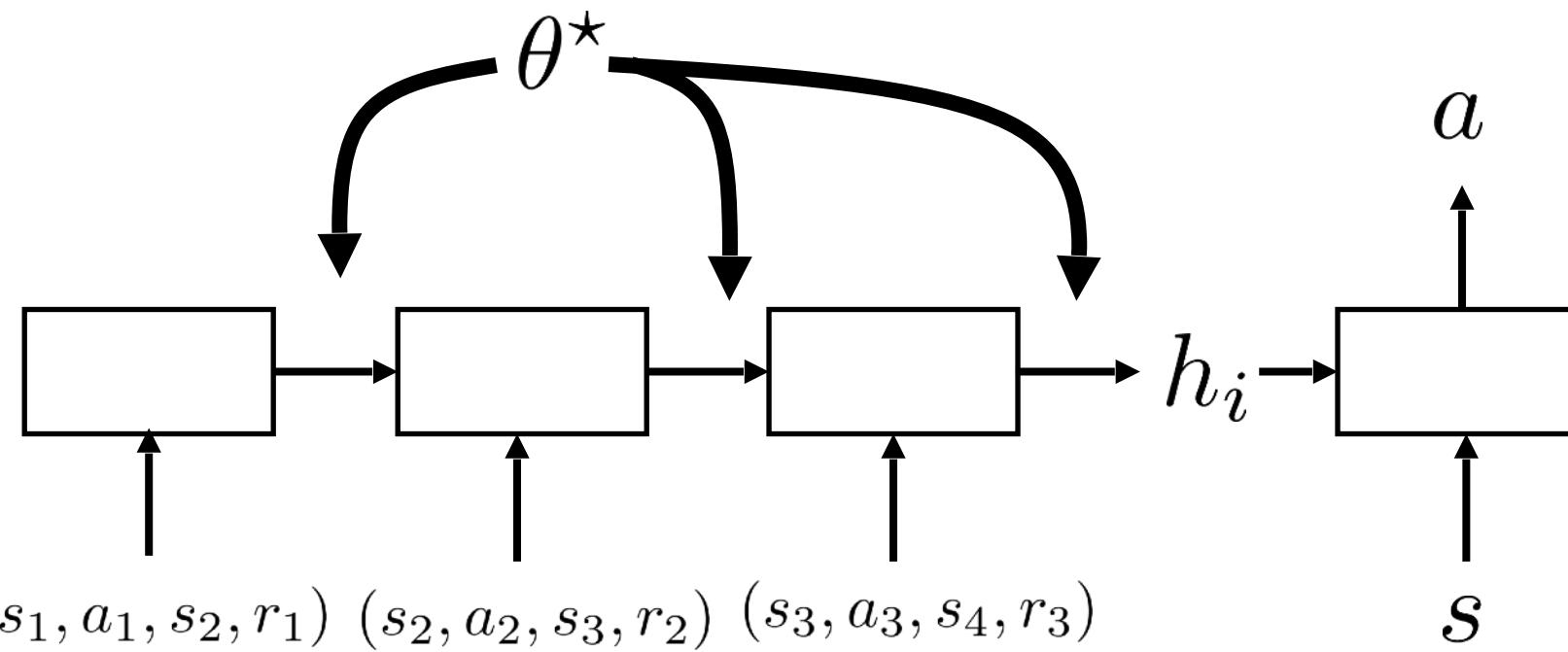
everything needed to solve task

- + simple, effective exploration via posterior sampling
- + elegant reduction to solving a special POMDP
- vulnerable to meta-overfitting
- challenging to optimize in practice

But they're not that different!

just perspective 1,
but with stochastic
hidden variables!
i.e., $\phi = \mathbf{z}$

Perspective 1: just RNN it



Perspective 2: bi-level optimization

$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

MAML for RL

Perspective 3: it's an inference problem!

$$\pi_\theta(a|s, z) \quad z_t \sim p(z_t|s_{1:t}, a_{1:t}, r_{1:t})$$

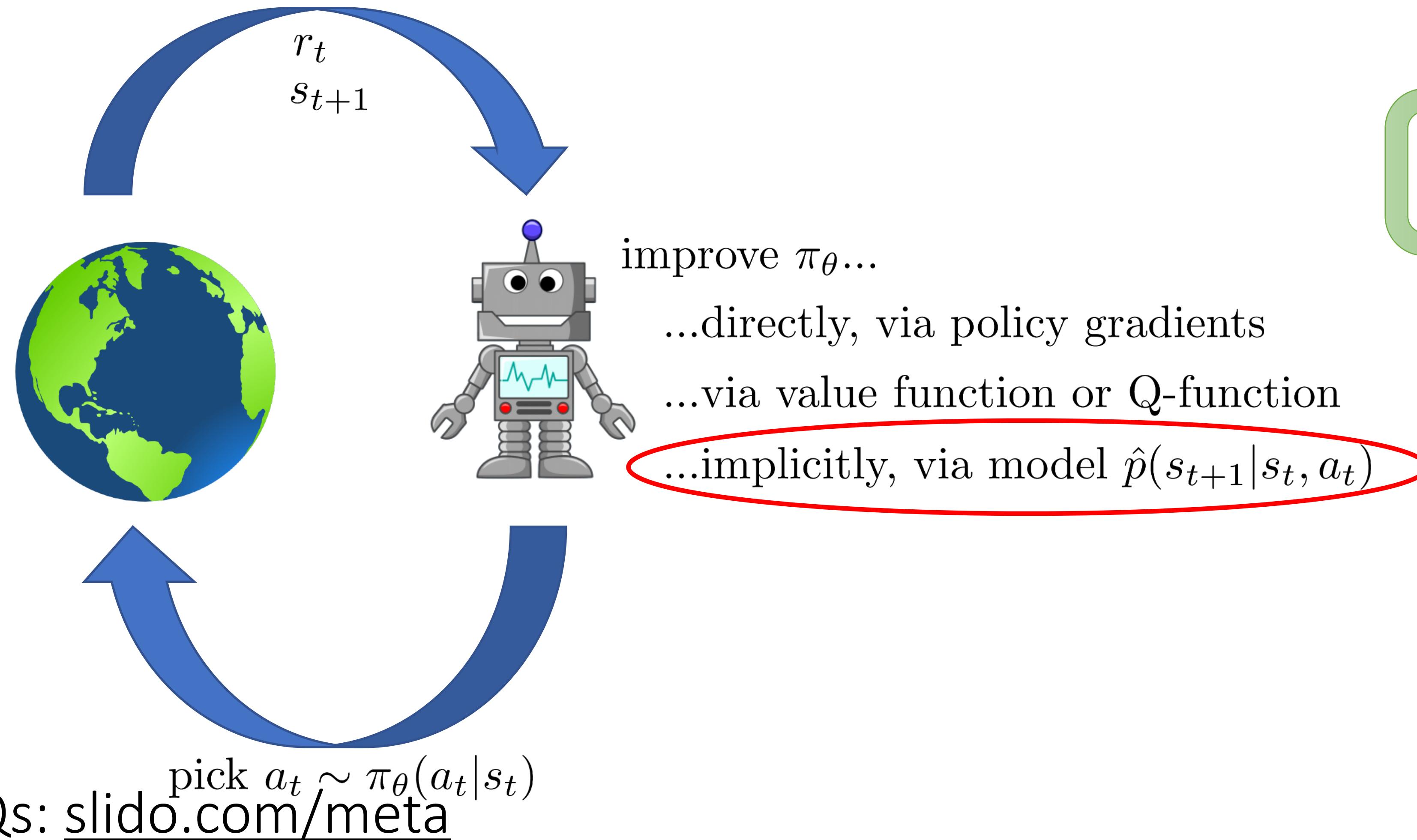
everything needed to solve task

just a particular
architecture choice
for these

Additional Topics in Meta-RL

Model-based meta-RL

$$\theta^* = \arg \max_{\theta} E_{\pi_{\theta}(\tau)} [R(\tau)]$$



short sketch of model-based RL:

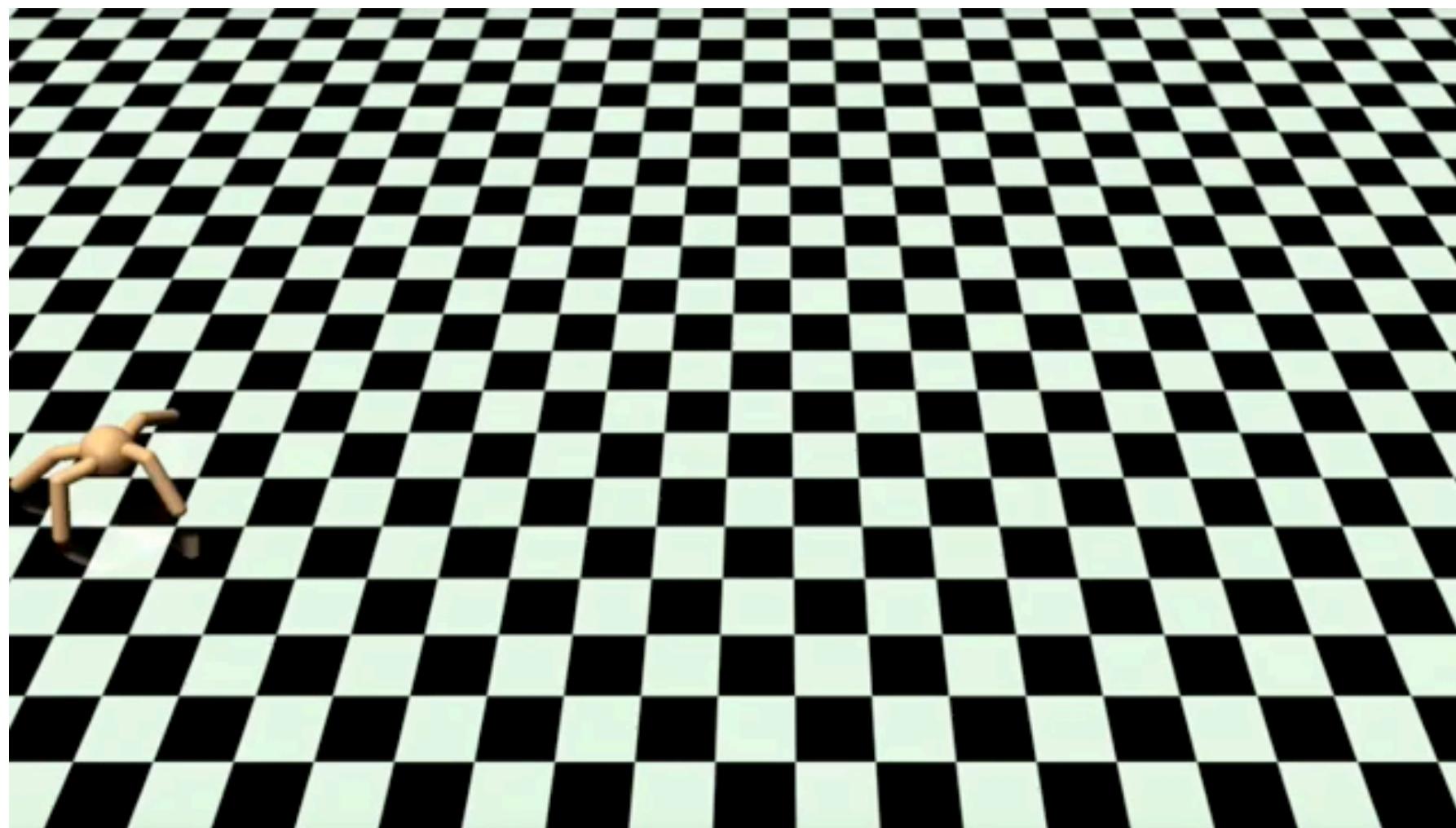
1. collect data \mathcal{B}
2. use \mathcal{B} to get $\hat{p}(s_{t+1}|s_t, a_t)$
3. use $\hat{p}(s_{t+1}|s_t, a_t)$ to plan a

why?

- + requires much less data vs model-free
- + a bit different due to model
- + can adapt extremely quickly!

Model-based meta-RL

example task: ant with broken leg



nice idea, but how much
can we really adapt in just
one (or a few) step(s)?

non-adaptive method:

- 1. collect data $\mathcal{B} = \{s_i, a_i, s'_i\}$
- 2. train $d_\theta(s, a) \rightarrow s'$ on \mathcal{B}
- 3. use d_θ to optimize actions

$$a_t, \dots, a_{t+k} = \arg \max_{a_t, \dots, a_{t+k}} \sum_{\tau=t}^{t+k} r(s_\tau, a_\tau)$$

s.t. $s_{t+1} = d_\theta(s_t, a_t)$

a few episodes

adaptive method:

- 1. take *one* step, get $\{s, a, s'\}$
- 2. $\theta \leftarrow \theta - \alpha \nabla_\theta \|d_\theta(s, a) - s'\|^2$
- 3. use d_θ to optimize a_t, \dots, a_{t+k} , take a_t

Model-based meta-RL

meta-training time

$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \dots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

$$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \dots, (x_l^i, y_l^i)\}$$

$$x \leftarrow (s, a) \quad y \leftarrow s'$$

generate each $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{ts}}$:

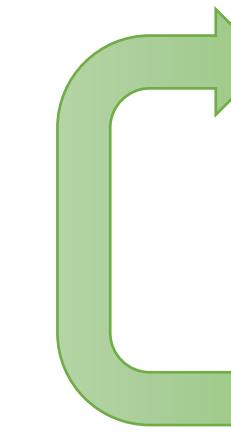
sample subsequence $s_t, a_t, \dots, s_{t+k}, a_{t+k}, s_{t+k+1}$ from past experience

$$\mathcal{D}_i^{\text{tr}} \leftarrow \{(s_t, a_t, s_{t+1}), \dots, (s_{t+k-1}, a_{t+k-1}, s_{t+k})\}$$

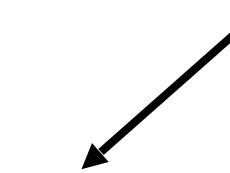
$$\mathcal{D}_i^{\text{ts}} \leftarrow \{(s_{t+k}, a_{t+k}, s_{t+k+1})\}$$

meta-test time

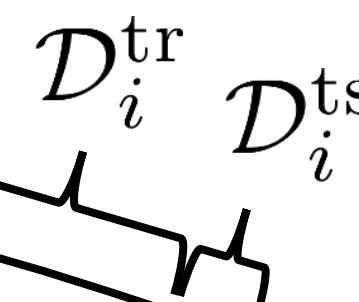
adaptive method:

- 
1. take one step, get $\{s, a, s'\}$
 2. $\theta \leftarrow \theta - \alpha \nabla_\theta \|d_\theta(s, a) - s'\|^2$
 3. use d_θ to optimize a_t, \dots, a_{t+k} , take a_t

assumes past experience has many different dynamics

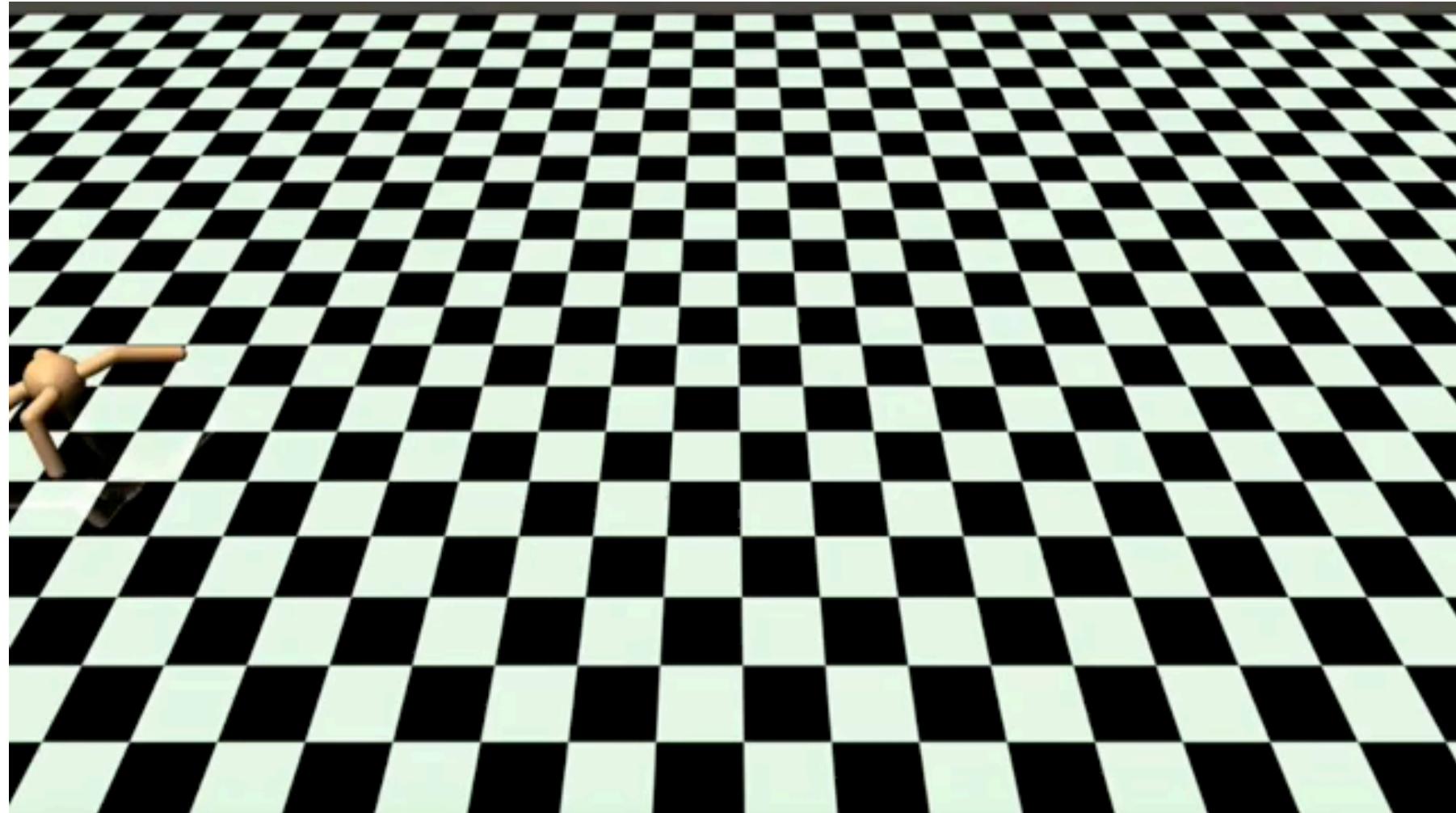


could choose $k = 1$, but $k > 1$
works better (e.g., $k = 5$)



Model-based meta-RL

example task: ant with broken leg



See also:

Saemundsson, Hofmann, Deisenroth. **Meta-Reinforcement Learning with Latent Variable Gaussian Processes.**

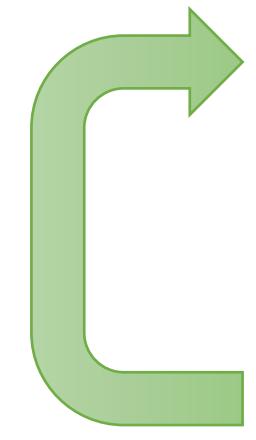
Nagabandi, Finn, Levine. **Deep Online Learning via Meta-Learning: Continual Adaptation for Model-Based RL.**

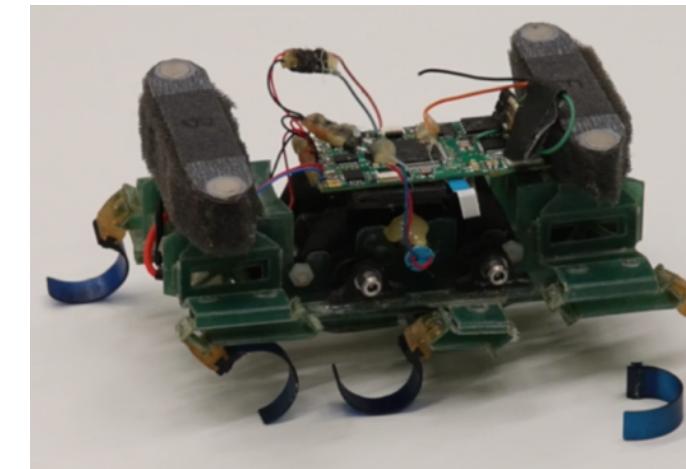
Qs: slido.com/meta

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn.
Learning to Adapt in Dynamic, Real-World Environments
Through Meta-Reinforcement Learning. ICLR 2019.

meta-test time

adaptive method:

- 
1. take *one* step, get $\{s, a, s'\}$
 2. $\theta \leftarrow \theta - \alpha \nabla_{\theta} \|d_{\theta}(s, a) - s'\|^2$
 3. use d_{θ} to optimize a_t, \dots, a_{t+k} , take a_t



model-based RL
(no adaptation)

with MAML

Meta-RL and emergent phenomena

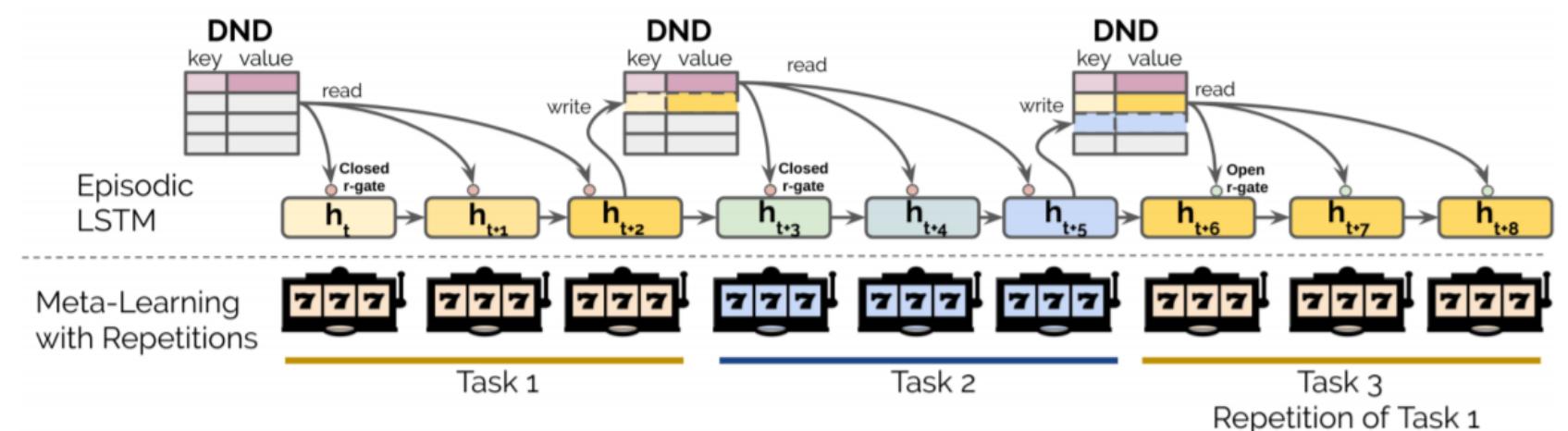
Humans and animals seemingly learn behaviors in a variety of ways:

- Highly efficient but (apparently) model-free RL
- Episodic recall
- Model-based RL
- Causal inference
- etc.

Perhaps each of these is a separate “algorithm” in the brain

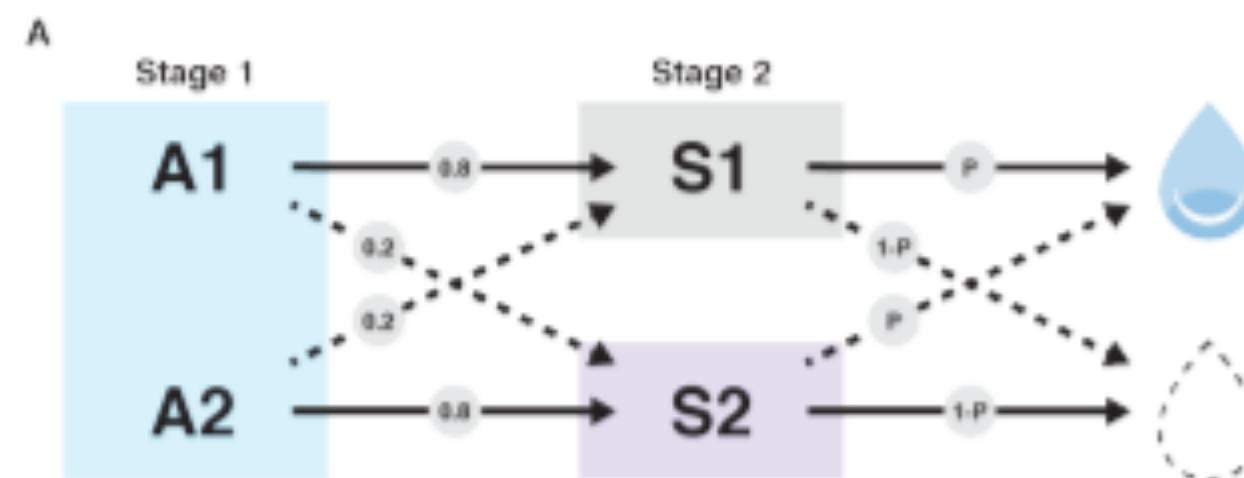
But maybe these are all emergent phenomena resulting from meta-RL?

meta-RL gives rise to
episodic learning



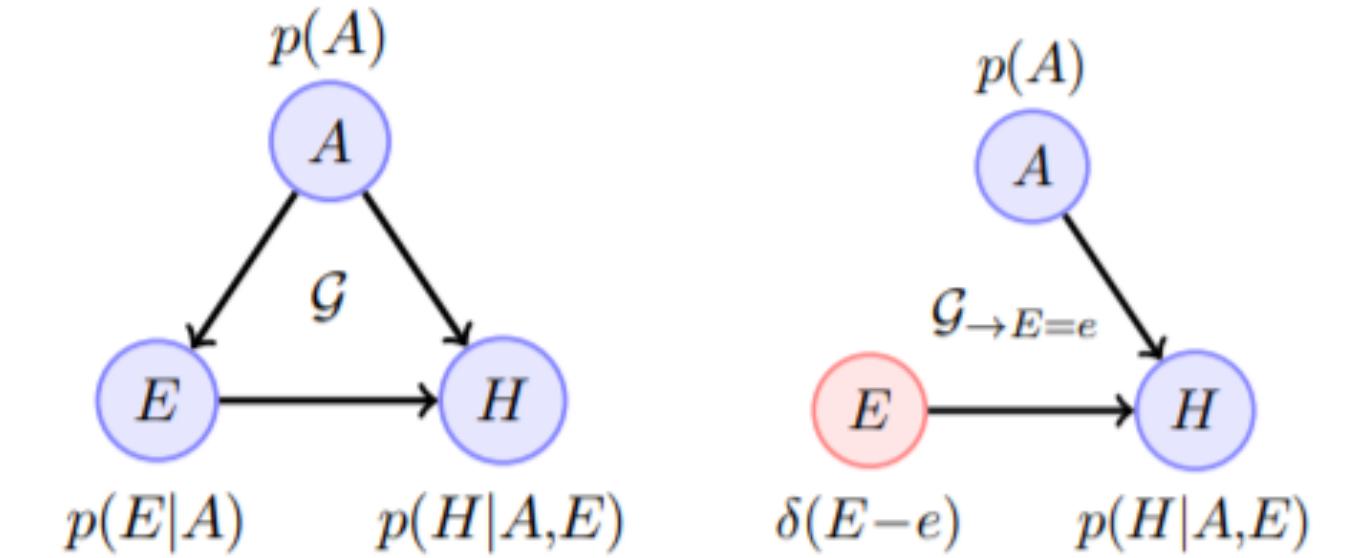
Ritter, Wang, Kurth-Nelson, Jayakumar, Blundell, Pascanu, Botvinick. **Been There, Done That: Meta-Learning with Episodic Recall.**

model-free meta-RL gives rise to
model-based adaptation



Wang, Kurth-Nelson, Kumaran, Tirumala, Soyer, Leibo, Hassabis, Botvinick. **Prefrontal Cortex as a Meta-Reinforcement Learning System.**

meta-RL gives rise to
causal reasoning (!)



Dasgupta, Wang, Chiappa, Mitrovic, Ortega, Raposo, Hughes, Battaglia, Botvinick, Kurth-Nelson. **Causal Reasoning from Meta-Reinforcement Learning.**

Contextual policies and meta-learning

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

where $\phi_i = f_{\theta}(\mathcal{M}_i)$



$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\theta}}[R(\tau)]$$

$$\pi_{\theta}(a_t | s_t, \underbrace{s_1, a_1, r_1, \dots, s_{t-1}, a_{t-1}, r_{t-1}}_{\text{context}})$$

context used to infer whatever we need to solve \mathcal{M}_i
i.e., z_t or ϕ_i (which are really the same thing)

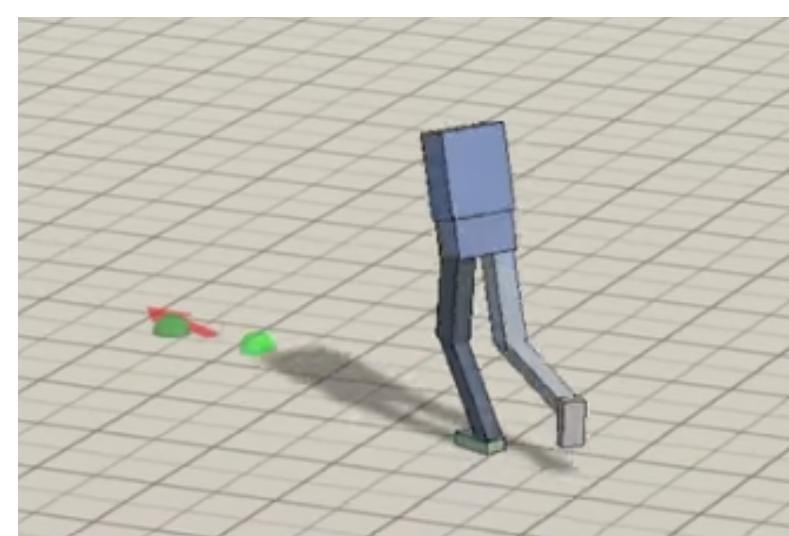
in meta-RL, the *context* is inferred from experience from \mathcal{M}_i

$$\pi_{\theta}(a_t | s_t, \phi_i)$$

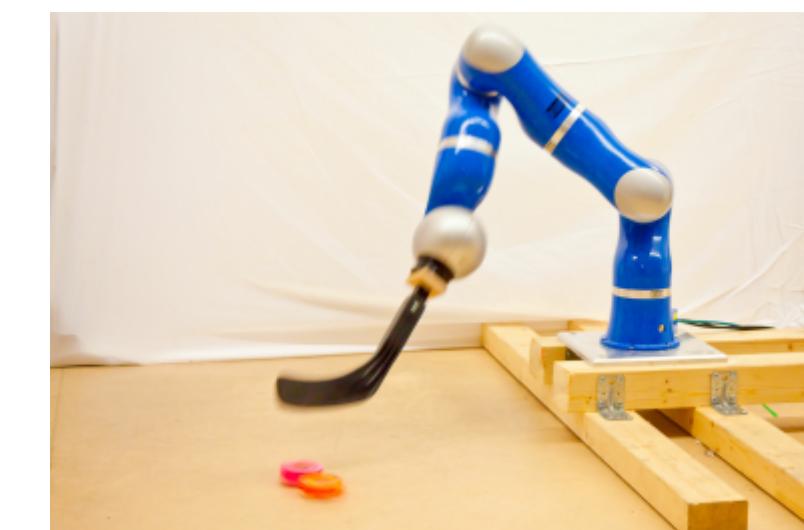
in multi-task RL, the context is typically given



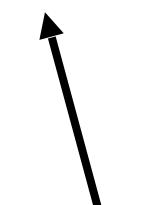
ϕ : stack location



ϕ : walking direction



ϕ : where to hit puck

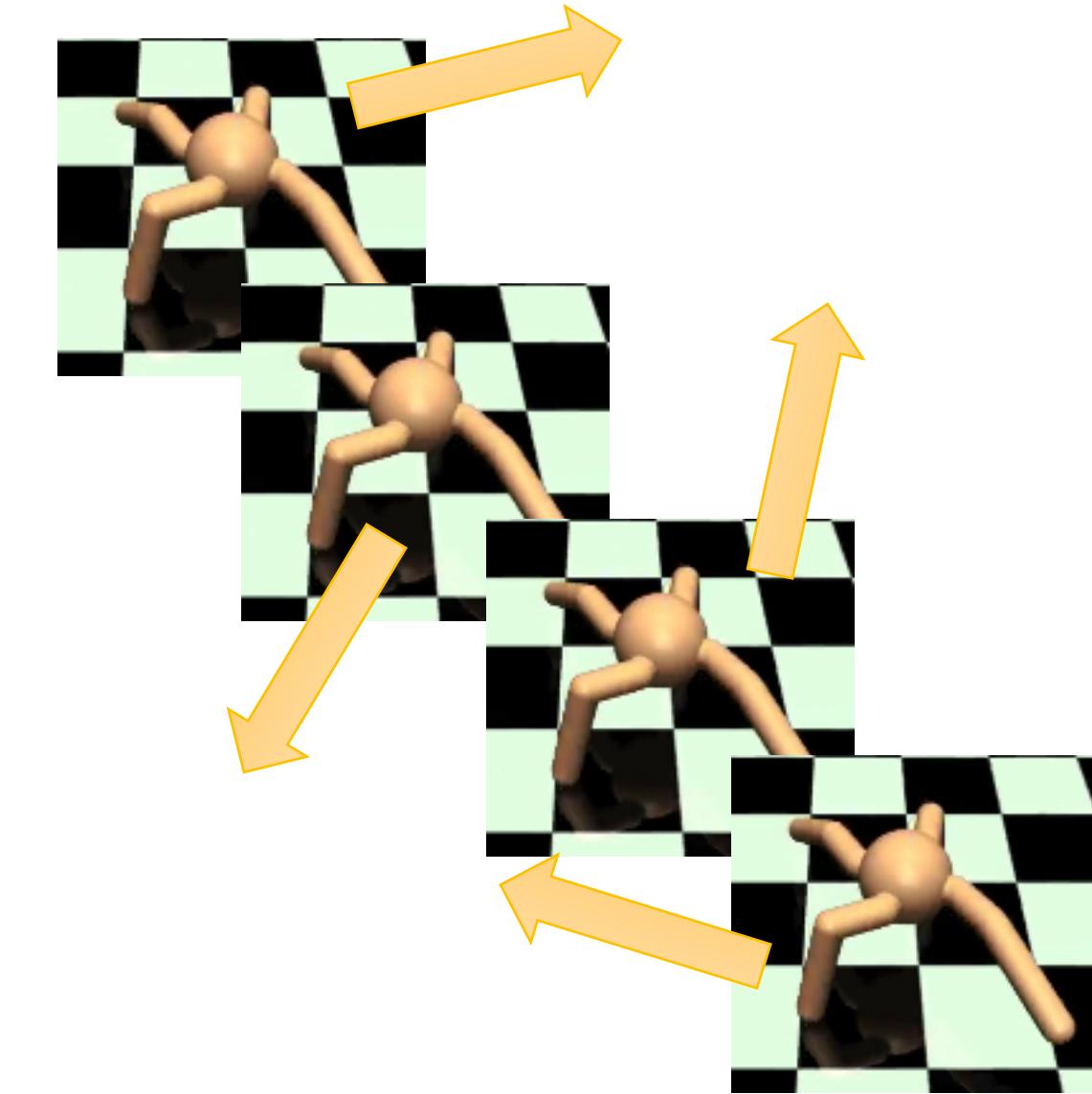
“context”


Outline

- Problem statement
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

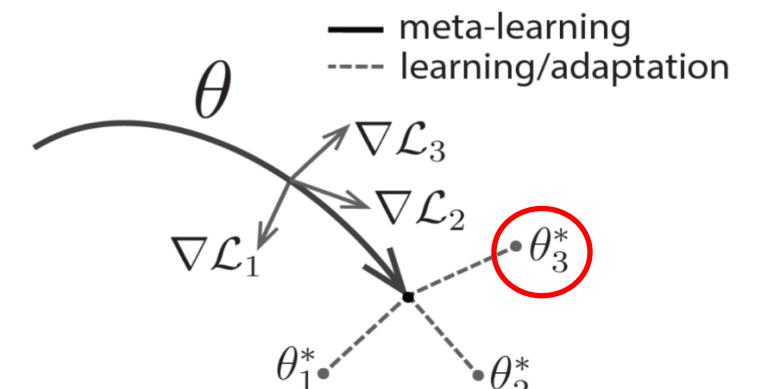
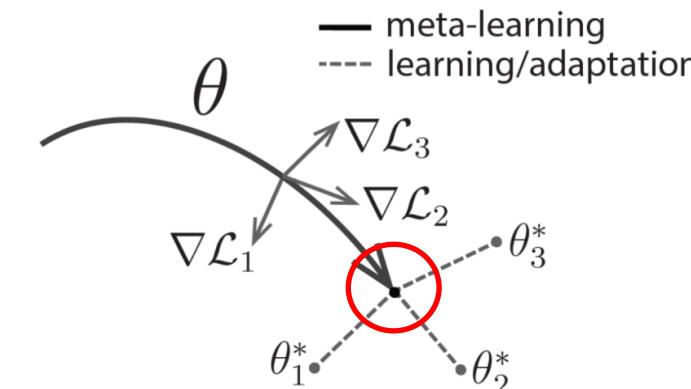
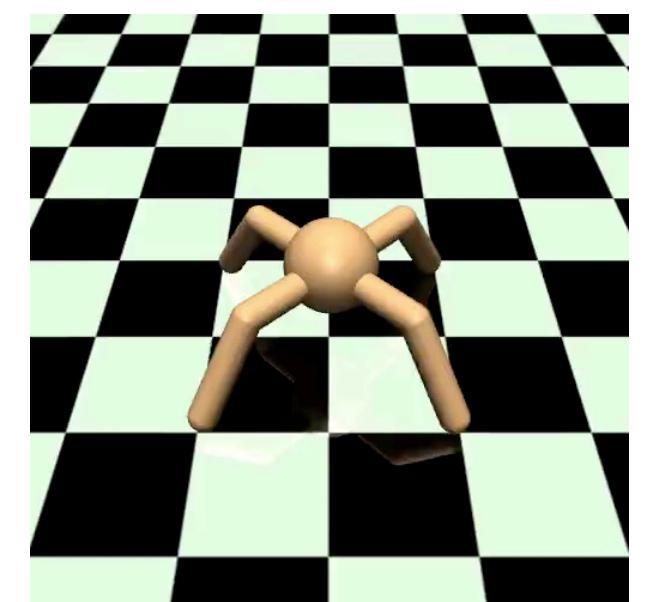
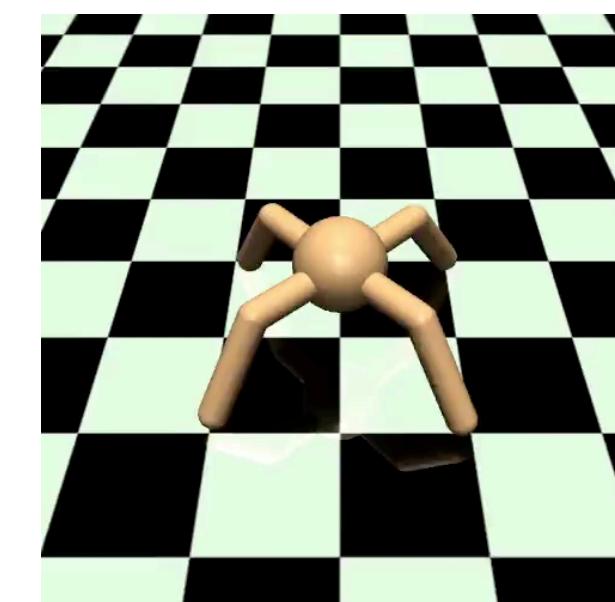
Let's Talk about Meta-Overfitting

- Meta learning requires task distributions
- When there are too few meta-training tasks, we can meta-overfit
- Specifying task distributions is hard!
- What can we do?



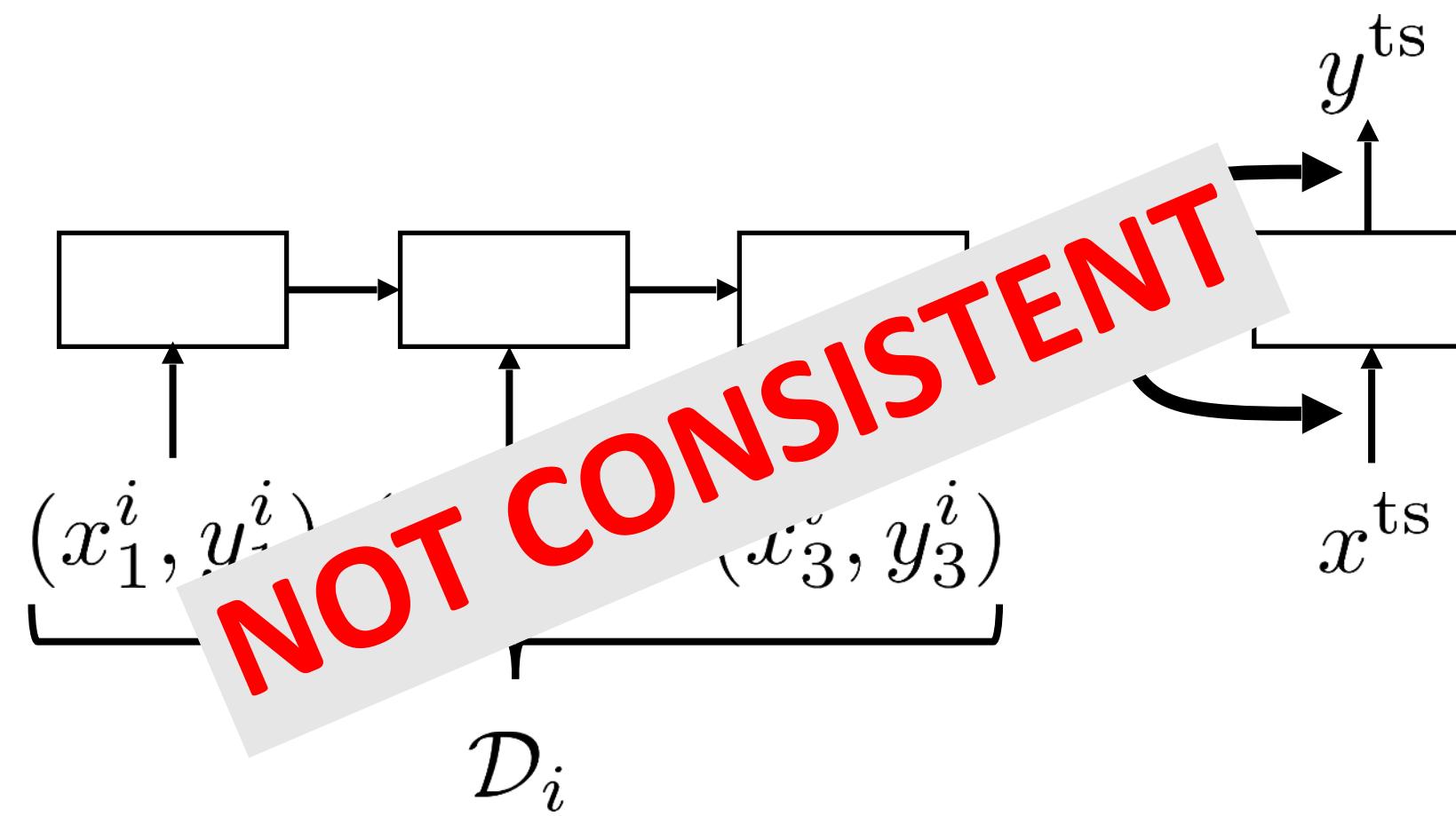
after MAML training

after 1 gradient step



Which algorithms meta-overfit less?

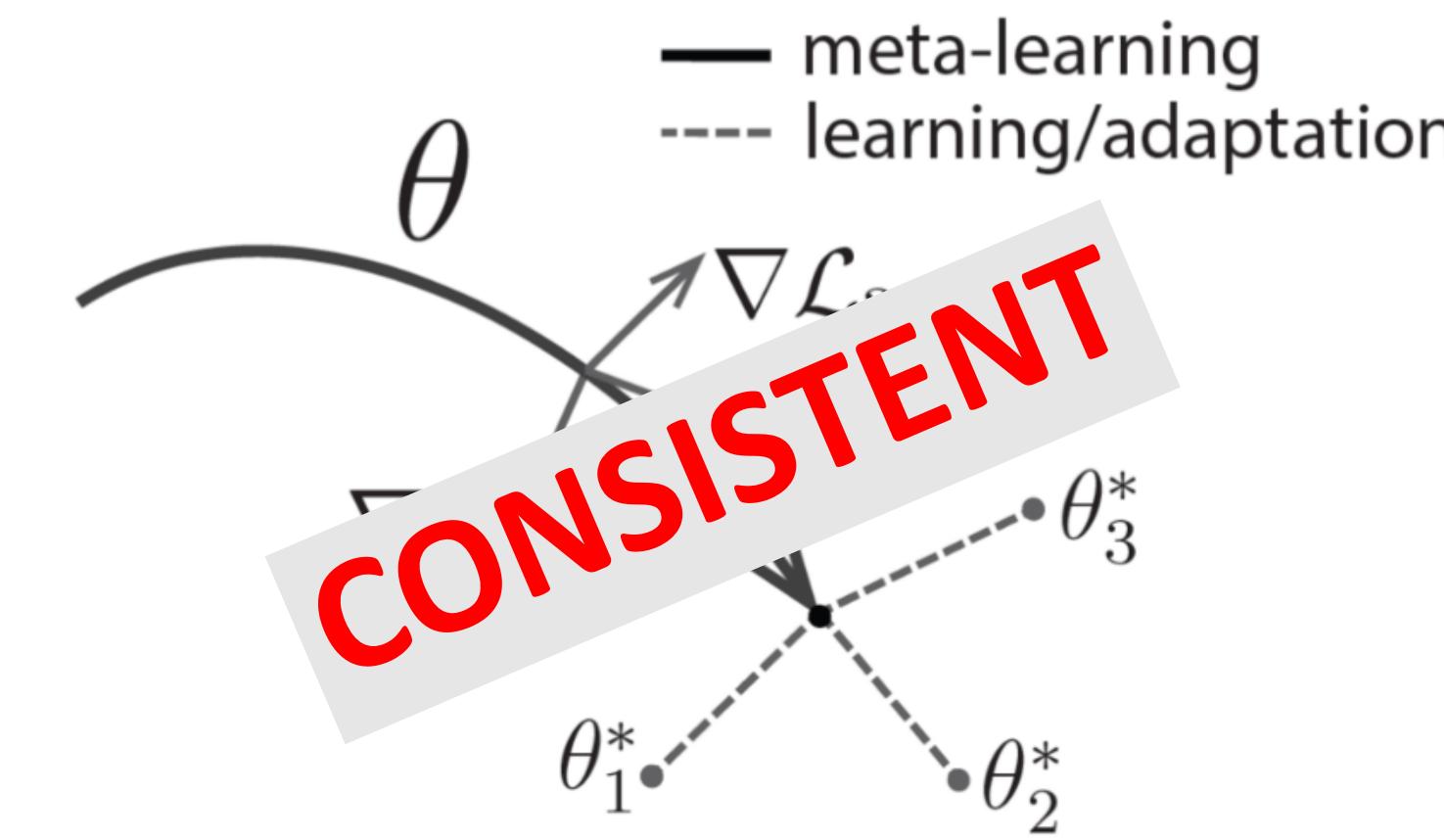
black-box adaptation



+ simple and flexible models

- relies **entirely** on extrapolation of learned adaptation procedure

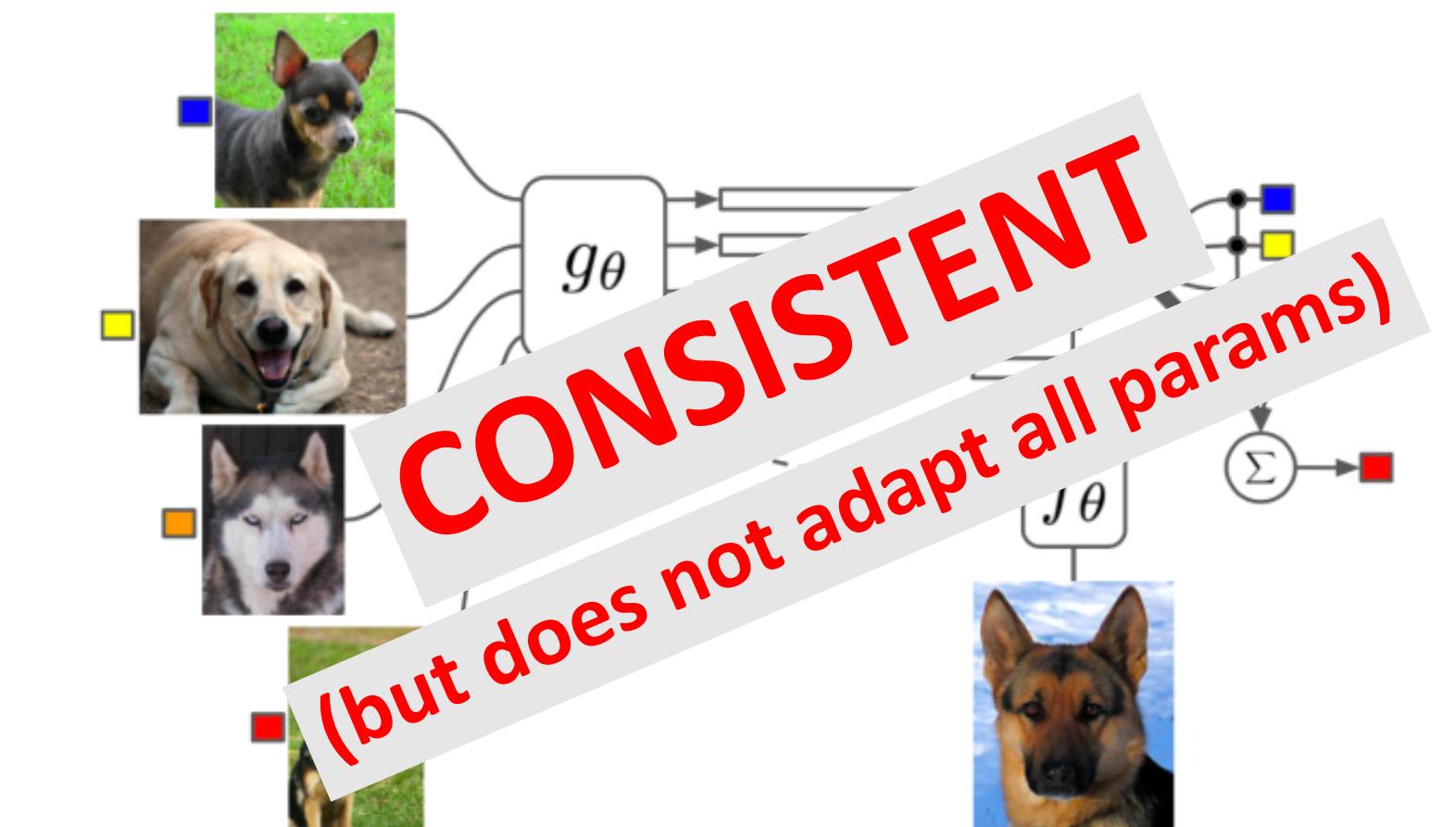
optimization-based



+ at worst just gradient descent

- pure gradient descent is not efficient without benefit of good initialization

non-parametric



+ at worst just nearest neighbor

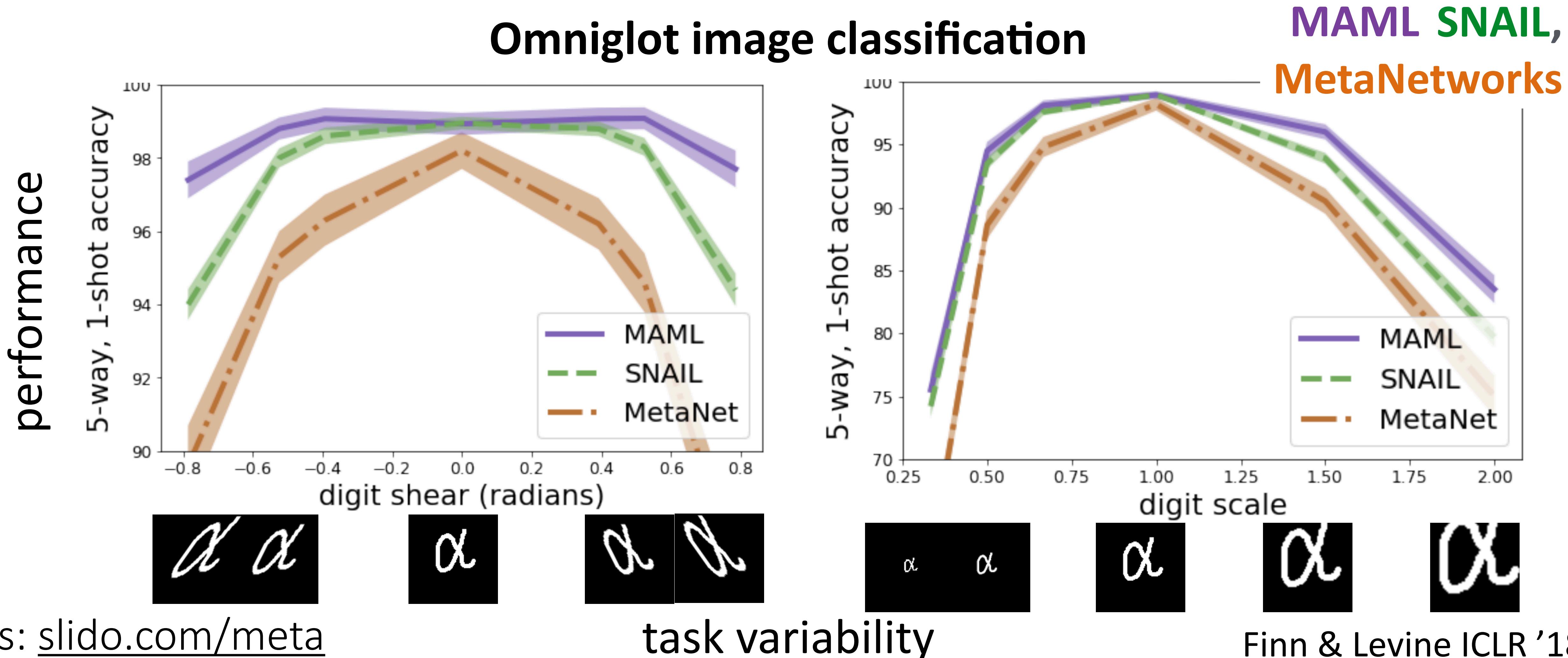
- does not adapt all parameters of metric on new data (might be nearest neighbor in very bad space)

Definition: a consistent meta-learner will converge to a (locally) optimal solution on any new task, regardless of meta-training

Qs: slido.com/meta

Empirical Extrapolation?

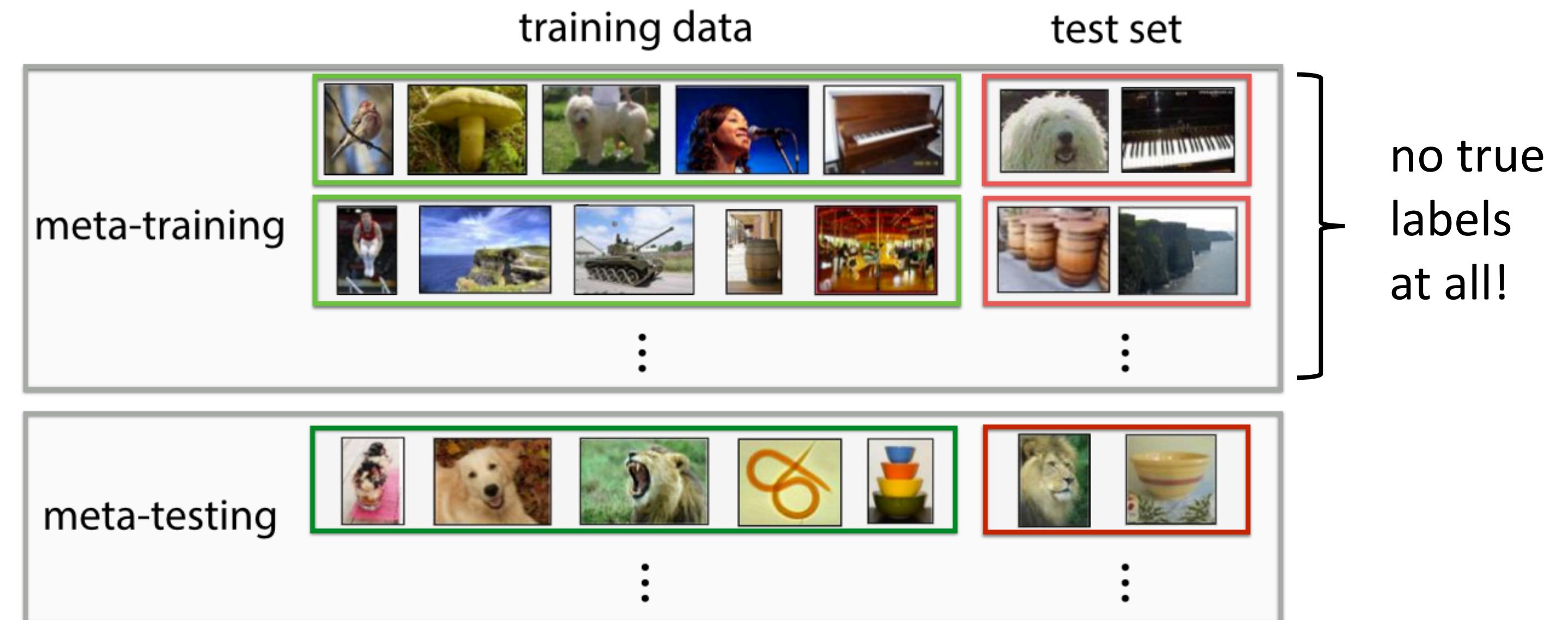
How well can learning procedures generalize to similar, but extrapolated tasks?



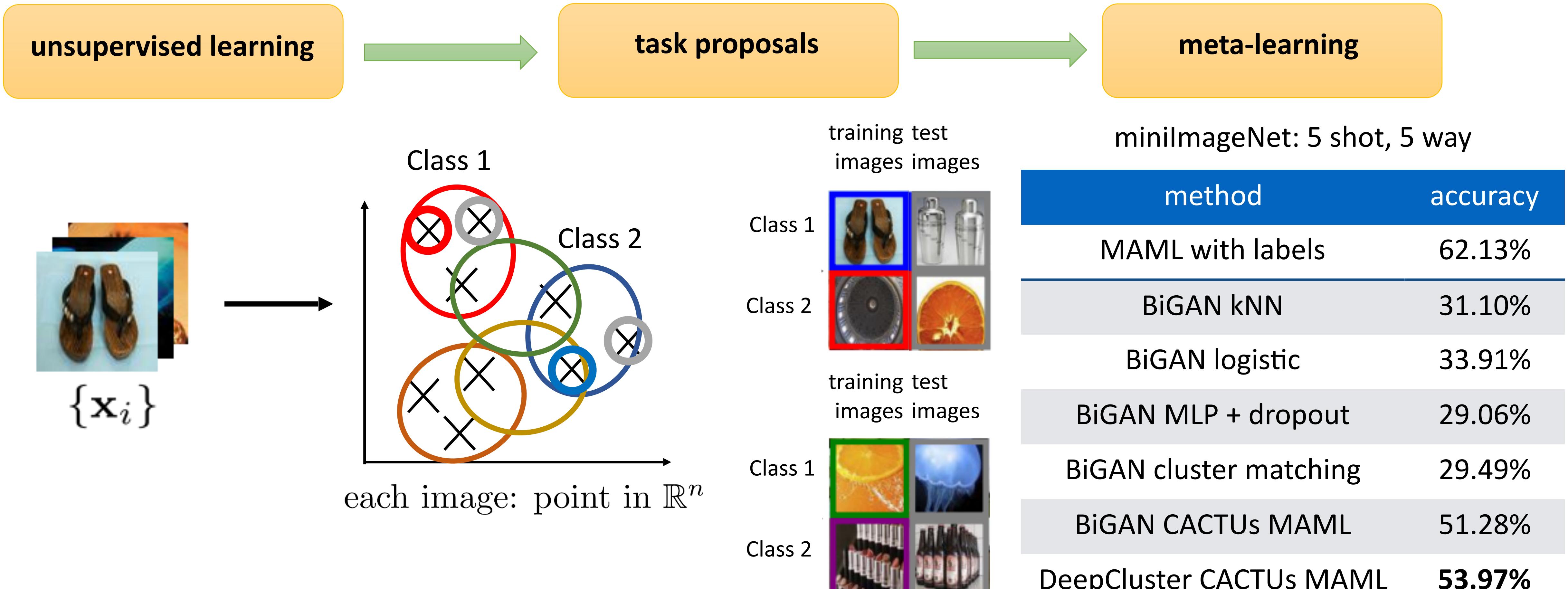
What else can we do?

- When there are too few meta-training tasks, we can meta-overfit
- Specifying task distributions is hard!
- Can we propose new tasks automatically?

Definition: unsupervised meta-learning refers to meta-learning algorithms that learn to solve tasks efficiently, without using hand-specified labels during meta-training



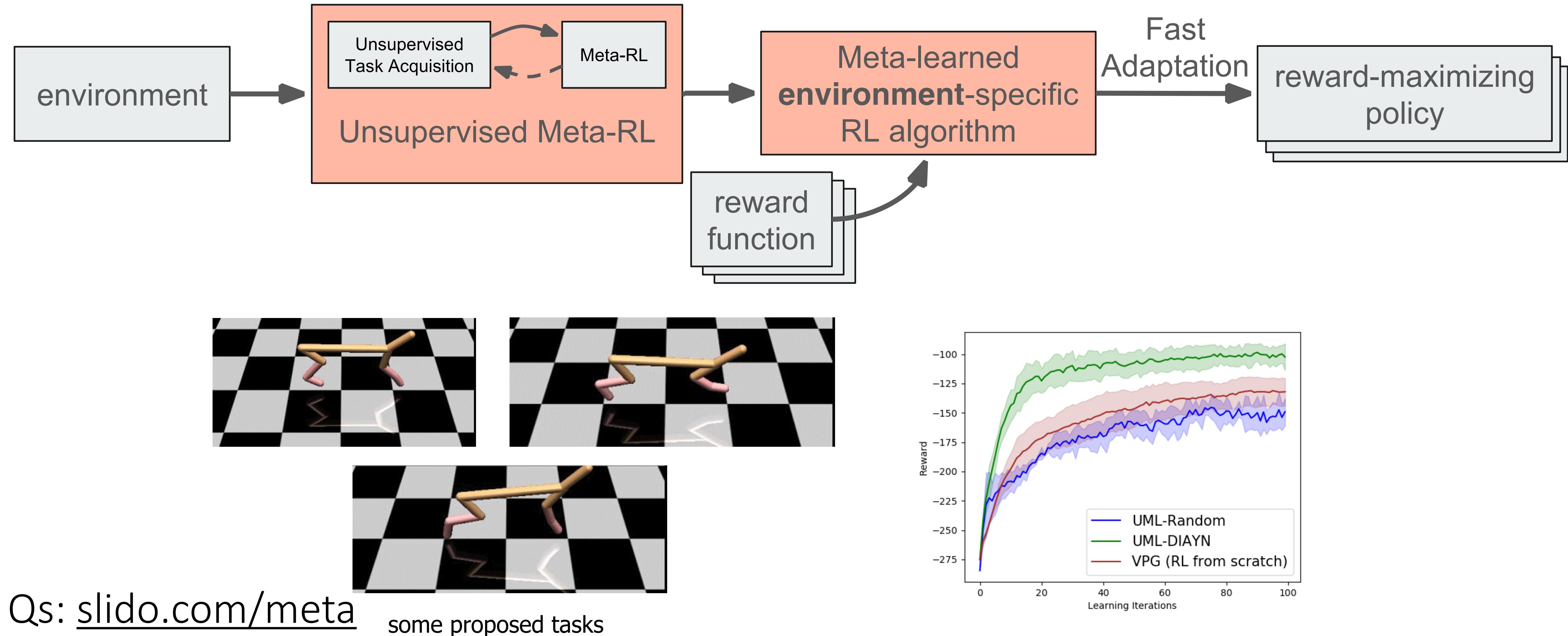
Example: stagewise unsupervised meta-learning



Clustering to Automatically Construct Tasks for Unsupervised Meta-Learning
(CACTUs)

Qs: slido.com/meta

Example: unsupervised meta-reinforcement learning



Gupta, Eysenbach, Finn, Levine. **Unsupervised Meta-Learning for Reinforcement Learning.**

Eysenbach, Gupta, Ibarz, Levine. **Diversity is All You Need.**

More on unsupervised meta-learning

- Unsupervised meta-RL: Gupta, Eysenbach, Finn, Levine. **Unsupervised Meta-Learning for Reinforcement Learning.**
- Unsupervised meta-few-shot classification: Hsu, Levine, Finn. **Unsupervised Learning via Meta-Learning.**
- Unsupervised meta-few-shot classification: Khodadadeh, Boloni, Shah. **Unsupervised Meta-Learning for Few-Shot Image and Video Classification.**
- Using supervised meta-learning to learn unsupervised learning rules: Metz, Maheswaranathan, Cheung, Sohl-Dickstein. **Meta-Learning Update Rules for Unsupervised Representation Learning.**
- Using supervised meta-learning to learn semi-supervised learning rules: Ren, Triantafillou, Ravi, Snell, Swersky, Tenenbaum, Larochelle, Zemel. **Meta-Learning for Semi-Supervised Few-Shot Classification.**

Memorization

Related to meta-overfitting, but subtly different.

Computation graph view: $y^{\text{ts}} = f_{\theta}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$

What will happen if the task data isn't strictly needed to learn the task?

Examples



Meta-training tasks: Cat/dog classifier.

Goal: Learn to quickly recognize a new breed as a cat.

Learn single classifier that doesn't adapt.

Meta-training tasks: Grasping different objects.

Goal: Learn to quickly grasp a new object.

Memorize how to grasp the training objects.

The tasks need to be **mutually exclusive**.

i.e. not possible to learn single function to learn all tasks

What you want the learner to glean from the data must be not present in x .

Challenge: can we learn to **trade off information** from **the data**

vs. **the input** based on amount of data

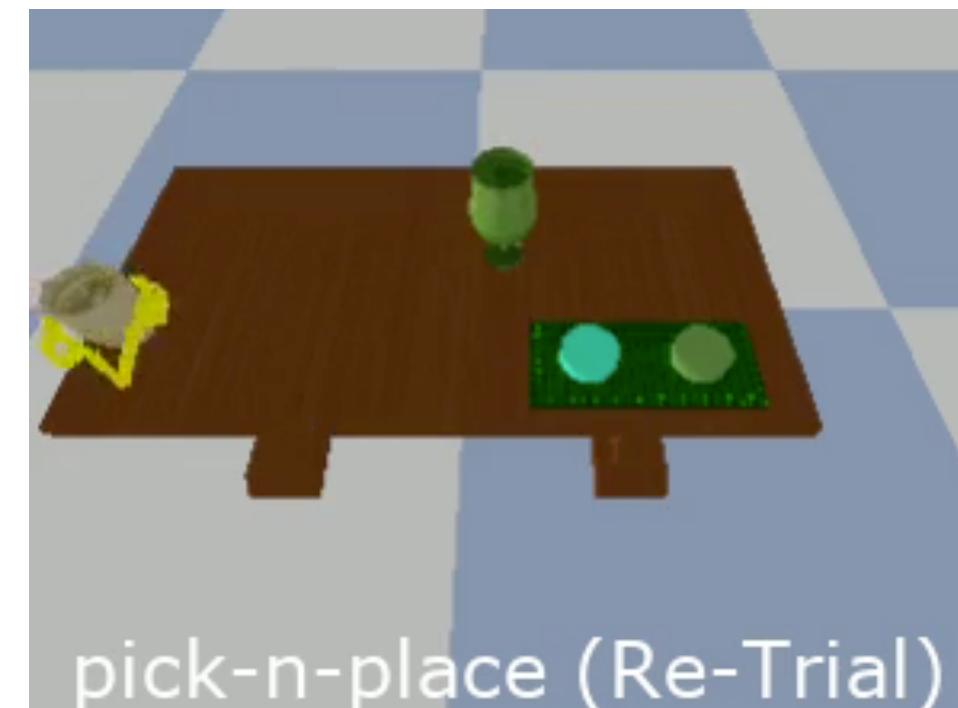
What task information should be in the input vs. data?

So far: The input contains **no information about the task.**

For **broad meta-RL task distributions**, exploration becomes exceedingly challenging.



One option: Provide **demonstration** (to illustrate the task goal) + trials



Zhou et al. Watch-Try-Learn: Meta-Learning Behavior from Demonstrations and Rewards, '19

Other options: language instruction?, goal image?, video tutorial?

The Ultimate Goal

Meta-Learning

Given i.i.d. task distribution,
learn a new task efficiently



learn



More realistically:

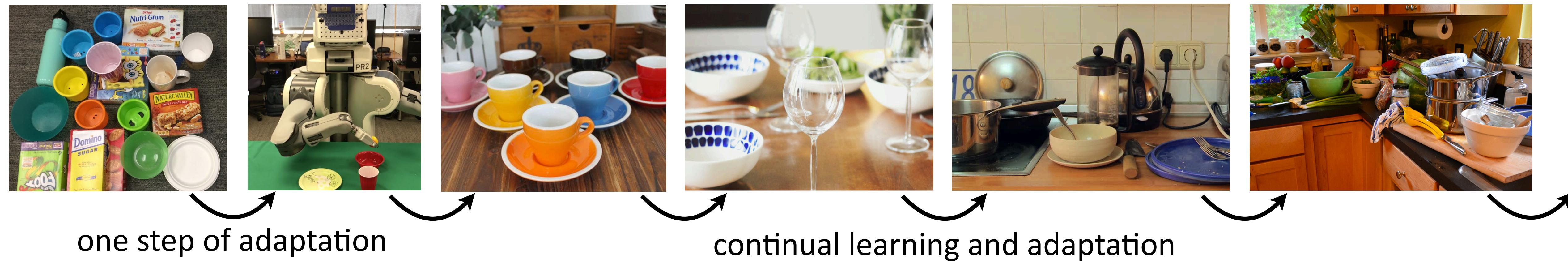
learn learn learn learn learn learn



slow learning

rapid learning

Initial work: Finn*, Rajeswaran* et al.
Online Meta-Learning ICML '19



Outline

- **Problem statement**
- Meta-learning **algorithms**
 - Black-box adaptation
 - Optimization-based inference
 - Non-parametric methods
 - Bayesian meta-learning
- Meta-learning **applications**
 - 5 min break —
- Meta-**reinforcement** learning
- Challenges & frontiers

Thank you!

Questions?