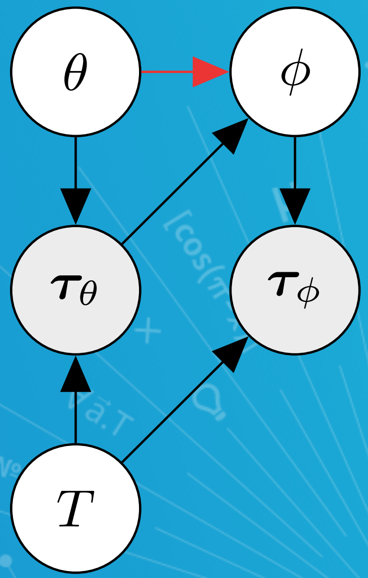
Carnegie Mellon University
School of Computer Science

Probabilistic Graphical Models

Elements of Meta-Learning

Maruan Al-Shedivat

Lecture 27, April 27, 2020



Reading: see class homepage

Outline

- **Part 1:** Intro to Meta-Learning
  - Motivation and some examples
  - General formulation and probabilistic view
  - Gradient-based and other types of meta-learning
  - Neural processes and relation of meta-learning to GPs
- **Part 2:** Elements of Meta-RL
  - What is meta-RL and why does it make sense?
  - On-policy and off-policy meta-RL
  - Continuous adaptation

Goals for the lecture:
Introduction & overview of the key methods and developments.

[Good starting point for you to start reading and understanding papers!]

Introduction to Meta-Learning

- Motivation and some examples
- General formulation and probabilistic view
- Gradient-based and other types of meta-learning
- Neural processes and relation of meta-learning to GPs

> Much of machine learning can be characterized as the search for a solution that, once found, no longer need be changed.
>
> [...] Machine learning has been more concerned with the results of learning than the ongoing process of learning.

— Rich Sutton, Anna Koop, David Silver (2007)

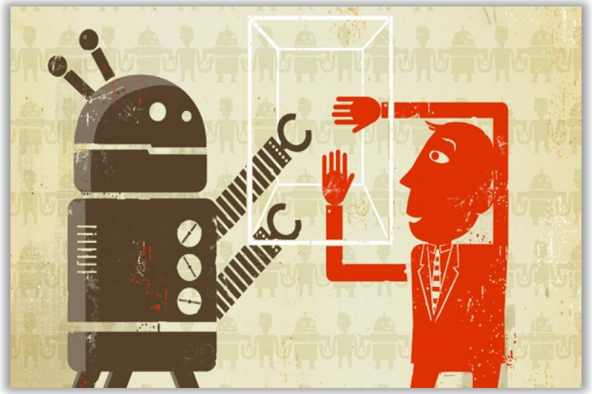
When is standard machine learning not enough?

Standard ML finally works for well-defined, stationary tasks

But how about…

Complex dynamic world?

Heterogeneous data from people?



Interactive robotic systems?

What is meta-learning?

- Standard learning: Given a distribution over examples (single task), learn a function that minimizes the loss

$$\hat{\phi} = \arg\min_{\phi} \mathbb{E}_{z\sim\mathcal{D}} \left[l(f_{\phi}(z)) \right]$$

- Learning-to-learn: Given a distribution over tasks, output an adaptation rule that can be used at test time to generalize from a task description

distribution over tasks/datasets

adaptation rule takes a task description as input and outputs a model

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}_{T\sim\mathcal{P}} \left\{ \mathcal{L}_T[g_{\theta}(T)] \right\}, \quad \text{where}$$

$$\mathcal{L}_T[g_{\theta}(T)] := \mathbb{E}_{z\sim\mathcal{D}_T} \left[l(f_{\phi}(z)) \right], \; \phi := g_{\theta}(T)$$

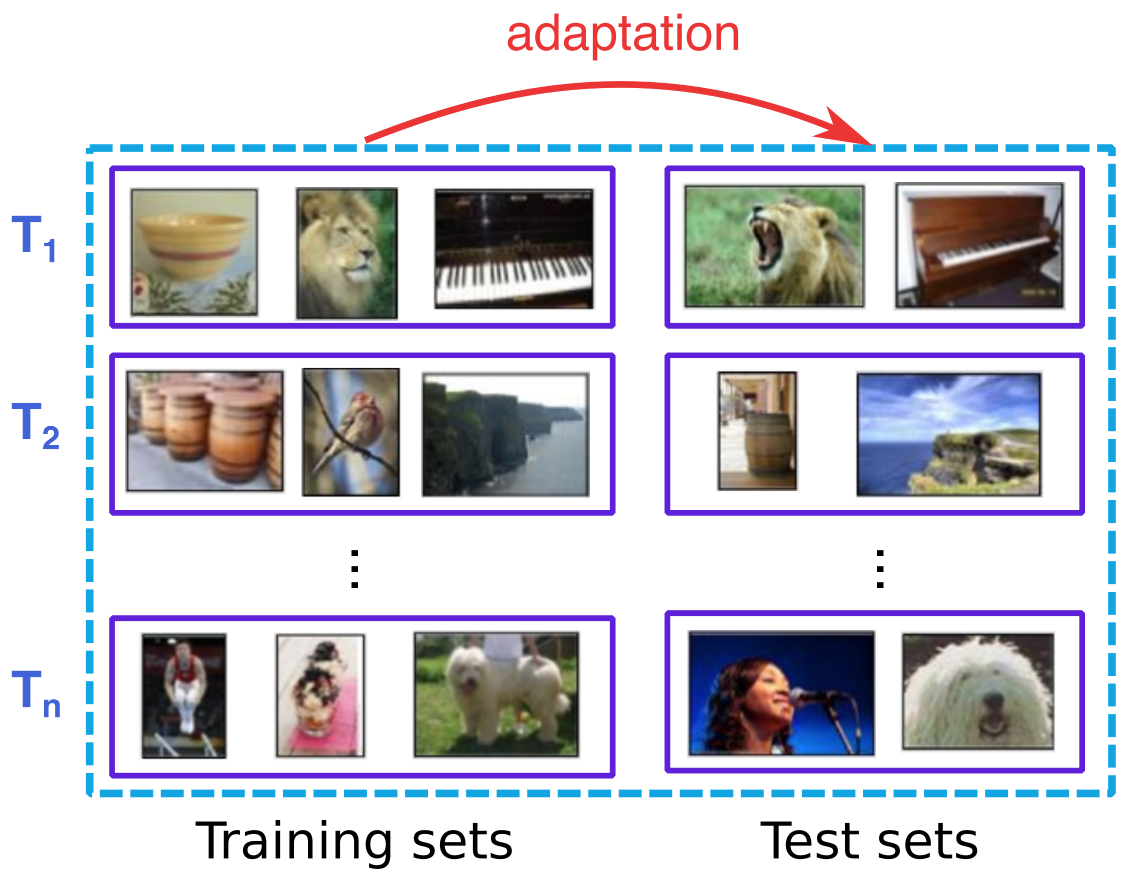
distribution over examples for task T

A Toy Example: Few-shot Image Classification

T_1

A Toy Example: Few-shot Image Classification

A Toy Example: Few-shot Image Classification

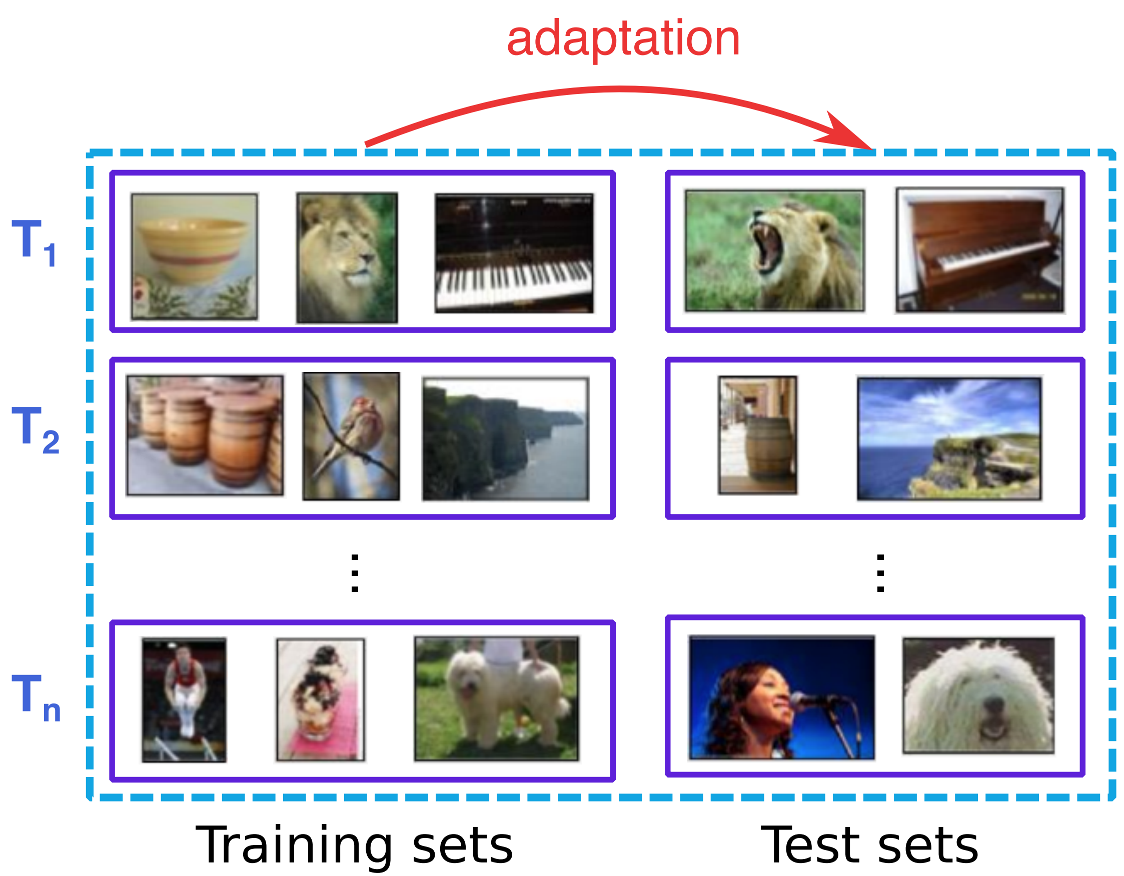


A Toy Example: Few-shot Image Classification

distribution over tasks/datasets

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{T \sim \mathcal{P}} \left[\mathcal{L}_T \left[g_\theta(T) \right] \right], \text{ where}$$

$$\mathcal{L}_T \left[g_\theta(T) \right] := \mathbb{E}_{z \sim \mathcal{D}_T} \left[L \left(f_\phi(z) \right) \right], \; \phi := g_\theta(T)$$



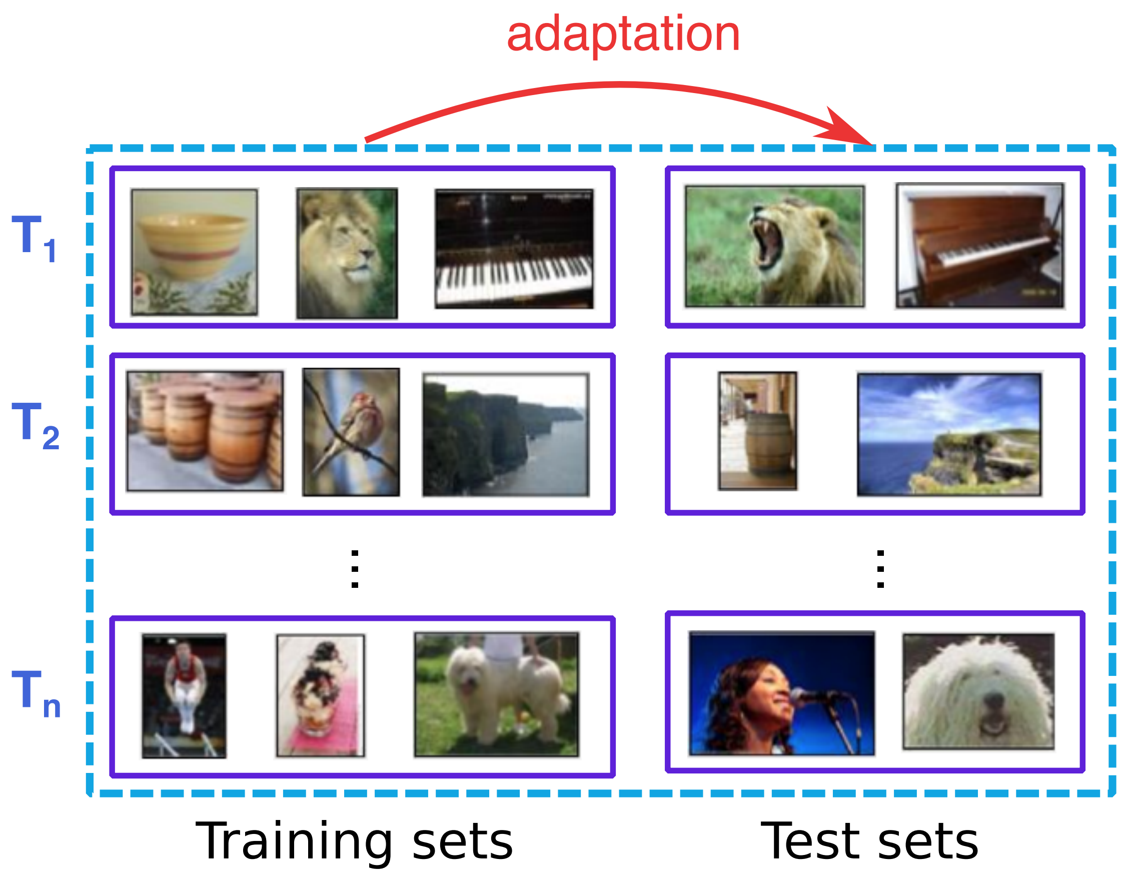
A Toy Example: Few-shot Image Classification

distribution over tasks/datasets

adaptation rule takes a task description and outputs a model

$$\hat{\theta} = \underset{\theta}{\text{argmin}}\, \mathbb{E}_{T \sim \mathcal{P}} \left[\mathcal{L}_T \left[g_\theta(T) \right] \right], \text{ where}$$

$$\mathcal{L}_T \left[g_\theta(T) \right] := \mathbb{E}_{z \sim \mathcal{D}_T} \left[L \left(f_\phi(z) \right) \right], \, \phi := g_\theta(T)$$



A Toy Example: Few-shot Image Classification

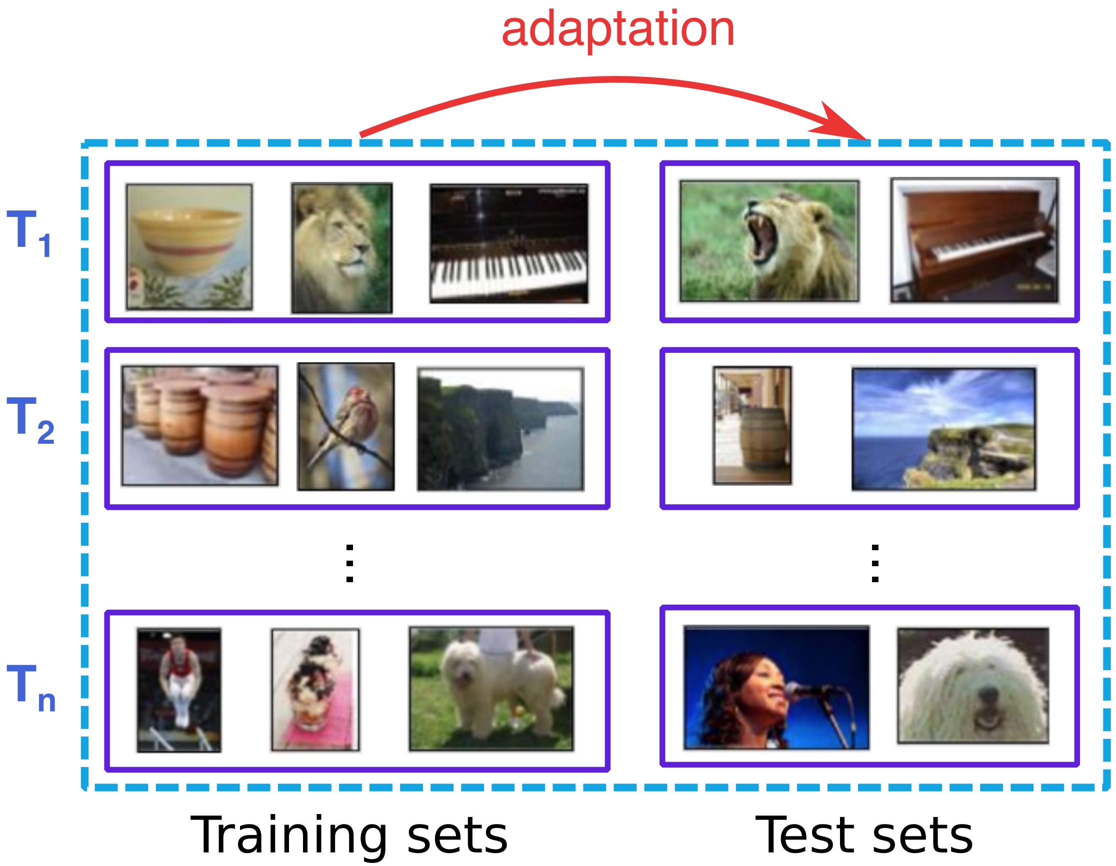
distribution over tasks/datasets

adaptation rule takes a task description and outputs a model

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{T \sim \mathcal{P}} \left[\mathcal{L}_T \left[g_\theta(T) \right] \right], \text{ where}$$

$$\mathcal{L}_T \left[g_\theta(T) \right] := \mathbb{E}_{z \sim \mathcal{D}_T} \left[L \left(f_\phi(z) \right) \right], \; \phi := g_\theta(T)$$

distribution over examples for task T



A Toy Example: Few-shot Image Classification

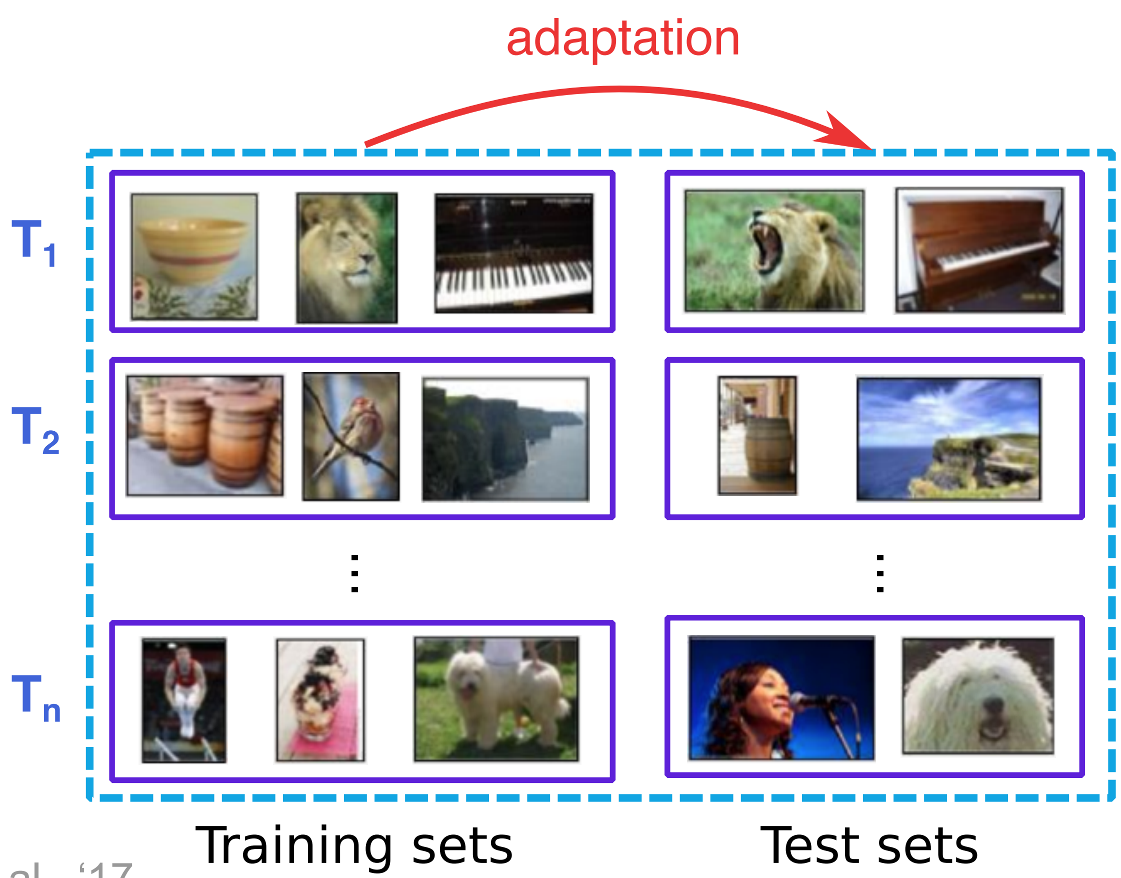
distribution over tasks/datasets

adaptation rule takes a task description and outputs a model

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \, \mathbb{E}_{T \sim \mathcal{P}} \left[\mathcal{L}_T \left[g_\theta(T) \right] \right], \text{ where}$$

$$\mathcal{L}_T \left[g_\theta(T) \right] := \mathbb{E}_{z \sim \mathcal{D}_T} \left[L \left(f_\phi(z) \right) \right], \; \phi := g_\theta(T)$$

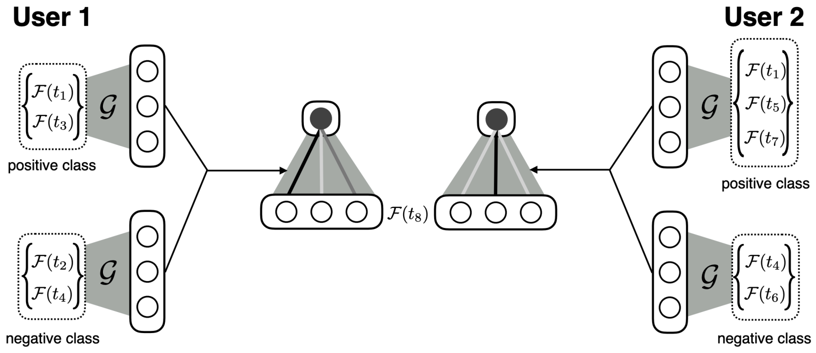
distribution over examples for task T



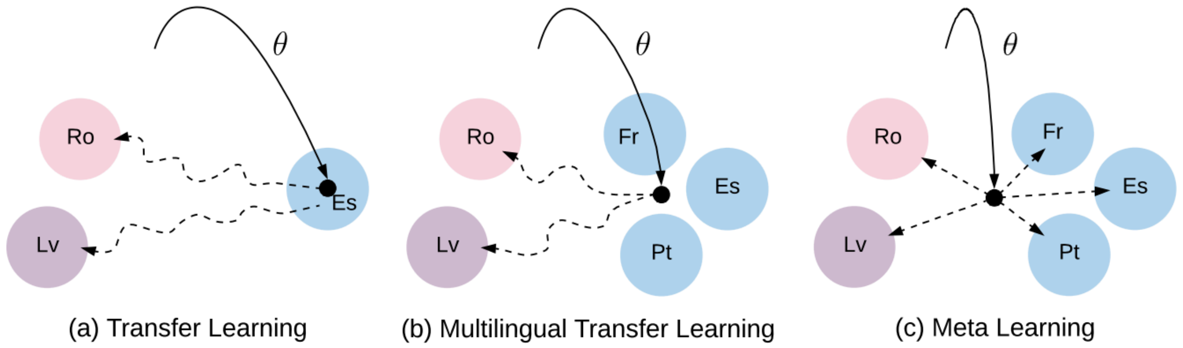
Meta-learning + adaptation methods:

- Recurrent nets (Santoro et al., '16, Duan et al., '17, Wang et al., '17, Mishra et al., '17, …)
- Learned optimizers (Schmidhuber, '87, Bengio et al., '90, Li & Malik, '16, Andrychowitcz et al., '16, …)
- …

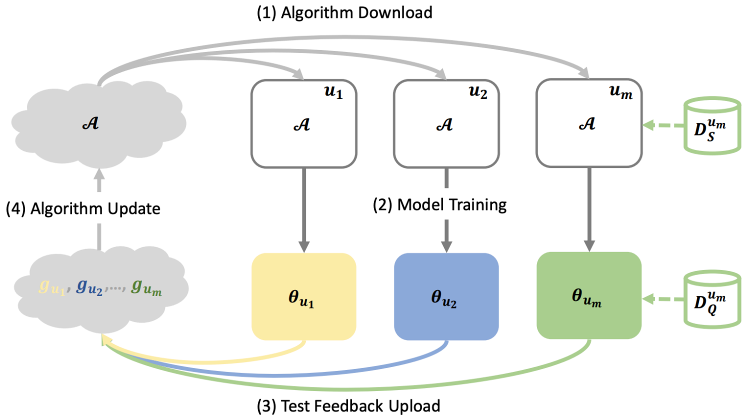
Other (practical) Examples of Few-shot Learning



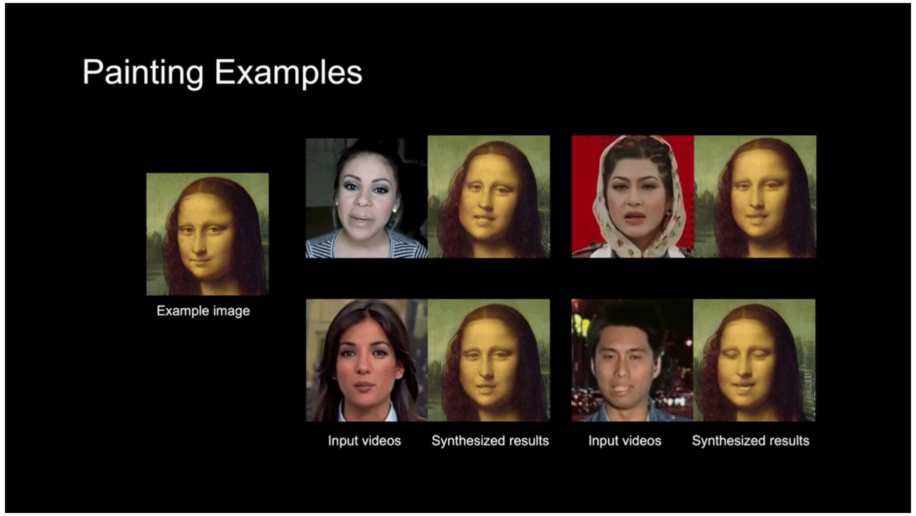
Few-shot learning for cold-start problem in recommendation (Vartak et al., NIPS 2017)



Low-resource translation (Gu*, Wang* et al., EMNLP 2018)

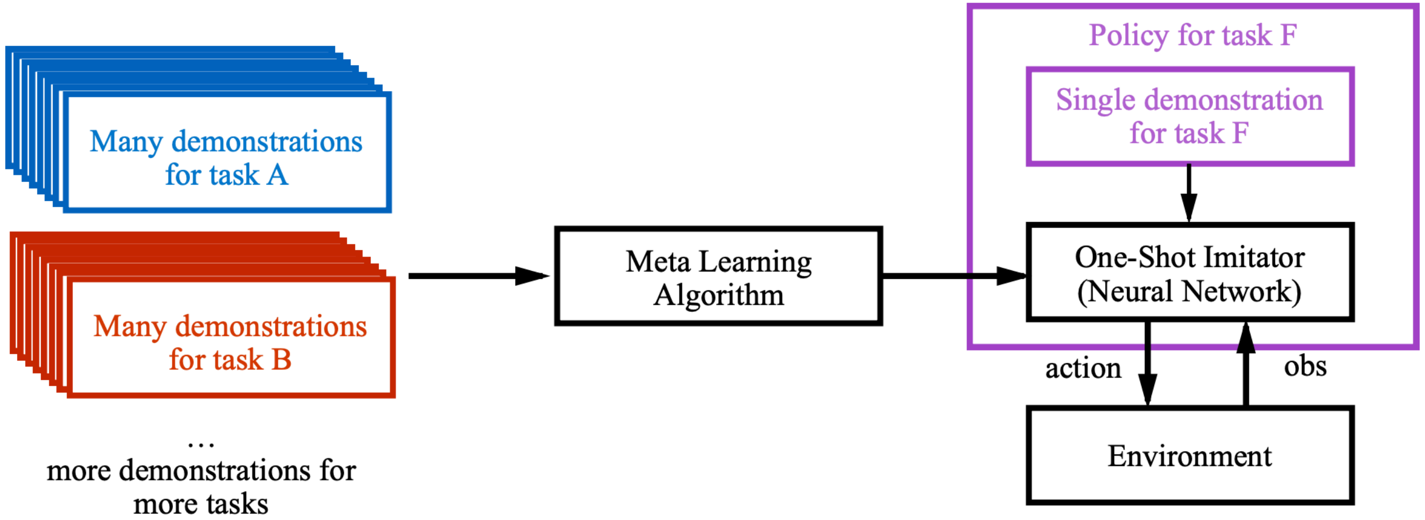


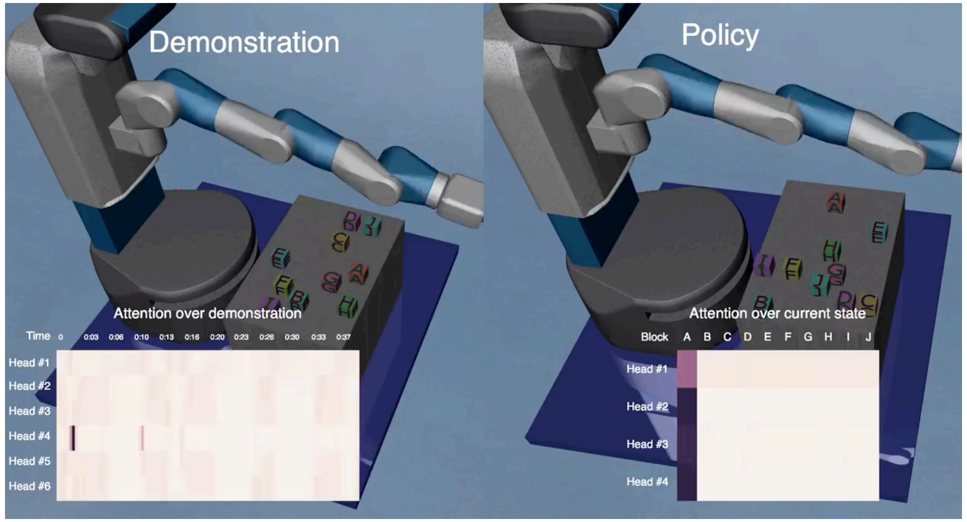
Federated recommender systems (Chen*, Luo* et al., 2018)



Few-shot video-to-video (Wang et al., 2019)

One More Example: One-shot Imitation Learning





Back to Our Few-shot Classification Example

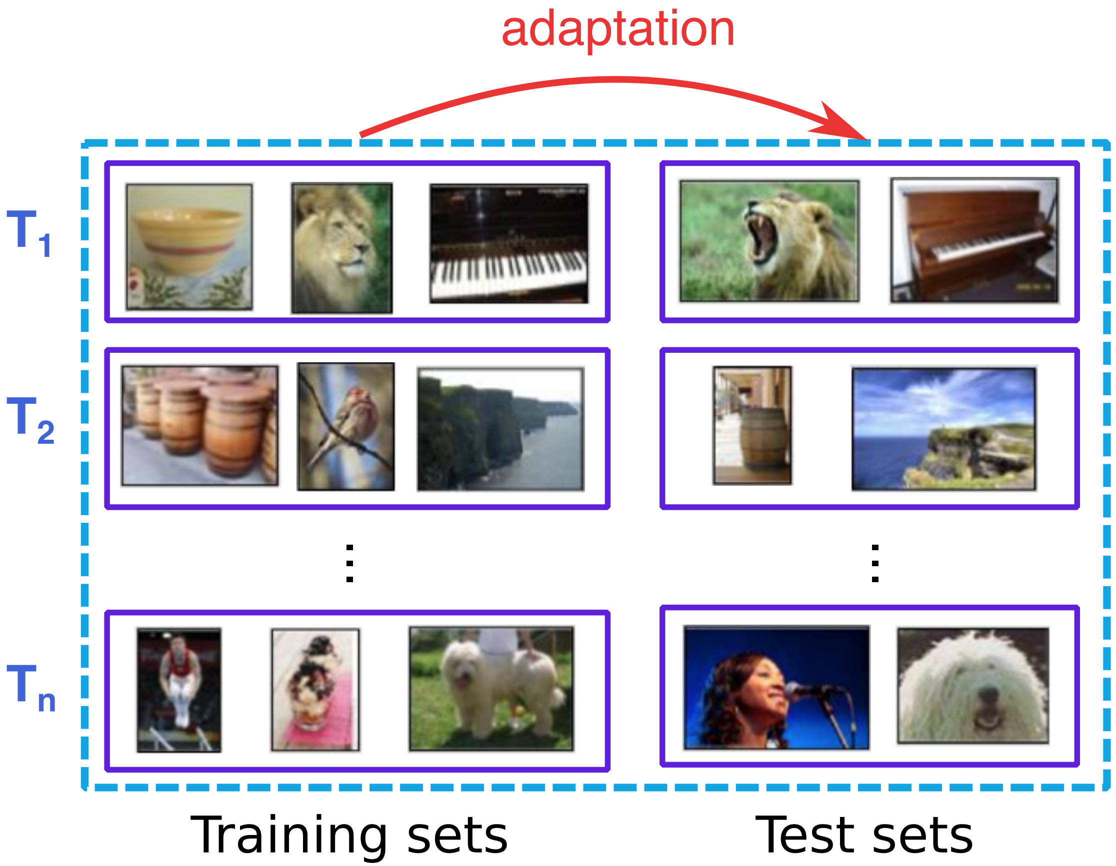
distribution over tasks/datasets

adaptation rule takes a task description and outputs a model

$$\hat{\theta} = \underset{\theta}{\arg\min}\, \mathbb{E}_{T\sim\mathcal{P}}\left[\mathcal{L}_T\left[g_\theta(T)\right]\right], \text{ where}$$

$$\mathcal{L}_T\left[g_\theta(T)\right] := \mathbb{E}_{z\sim\mathcal{D}_T}\left[L\left(f_\phi(z)\right)\right],\ \phi := g_\theta(T)$$

distribution over examples for task T



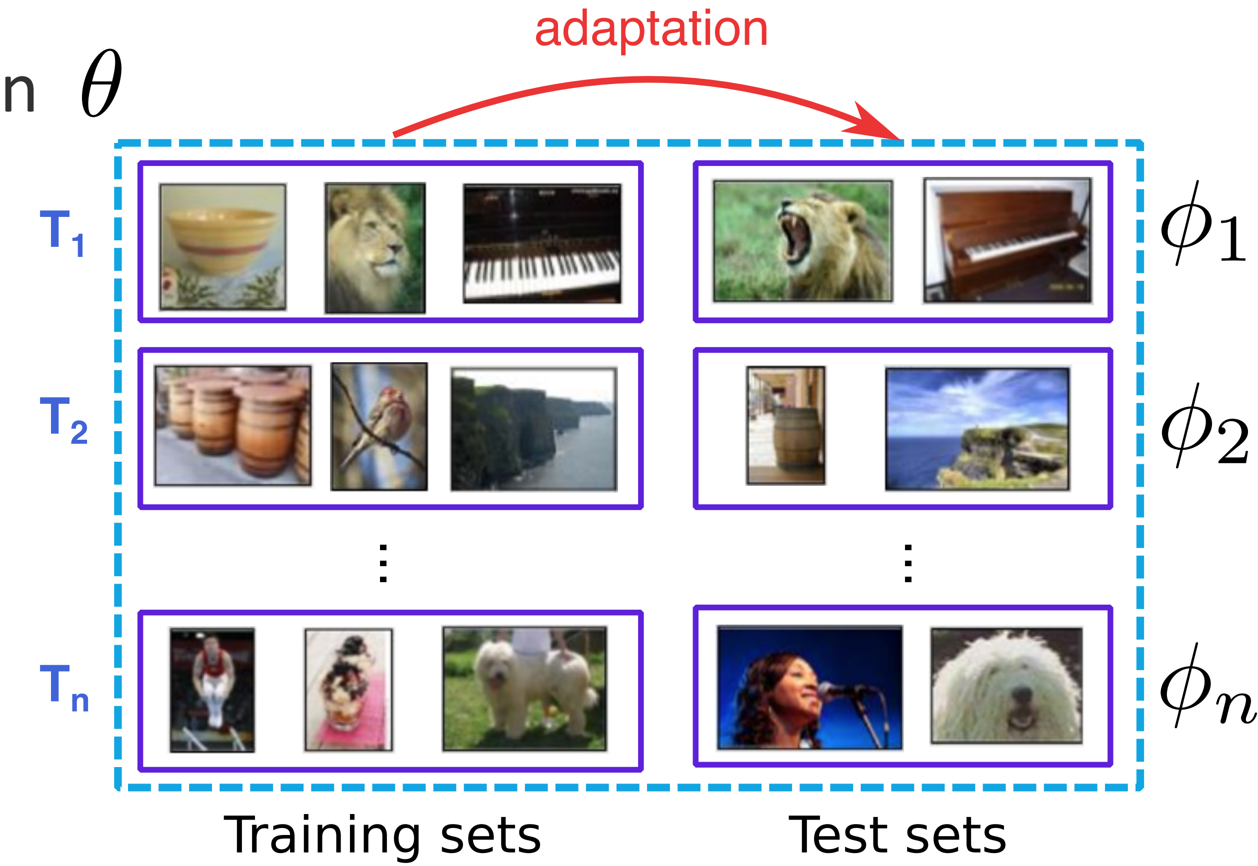
Model-agnostic Meta-learning (MAML)

- Start with a common model initialization θ
- Given a new task T_i, adapt the model using a gradient step:

$$\phi_i = g_\theta(T_i) := \theta - \alpha \nabla_\theta \mathcal{L}_{T_i}(f_\theta)$$

- Meta-training is learning a shared initialization for all tasks:

$$\min_\theta \sum_{T_i \sim \mathcal{P}} \mathcal{L}^{\text{test}}_{T_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}^{\text{train}}_{T_i}(f_\theta)}\right)$$



Finn, Abbeel, Levine, ICML 2017

Model-agnostic Meta-learning (MAML)

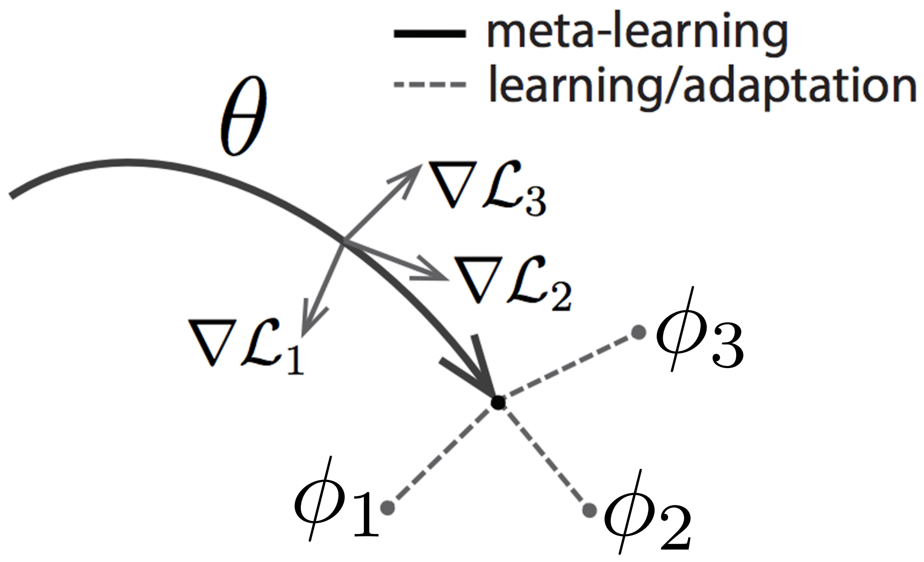
- Start with a common model initialization θ
- Given a new task T_i, adapt the model using a gradient step:

$$\phi_i = g_\theta(T_i) := \theta - \alpha \nabla_\theta \mathcal{L}_{T_i}(f_\theta)$$

- Meta-training is learning a shared initialization for all tasks:

$$\min_\theta \sum_{T_i \sim \mathcal{P}} \mathcal{L}_{T_i}^{\text{test}}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{T_i}^{\text{train}}(f_\theta)}\right)$$

Intuition:



Finn, Abbeel, Levine, ICML 2017

Does MAML Work?

Omniglot (Lake et al., 2011)	5-way Accuracy 1-shot	5-way Accuracy 5-shot	20-way Accuracy 1-shot	20-way Accuracy 5-shot
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	**89.7 ± 1.1%**	**97.5 ± 0.6%**	–	–
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	**98.7 ± 0.4%**	**99.9 ± 0.1%**	**95.8 ± 0.3%**	**98.9 ± 0.2%**

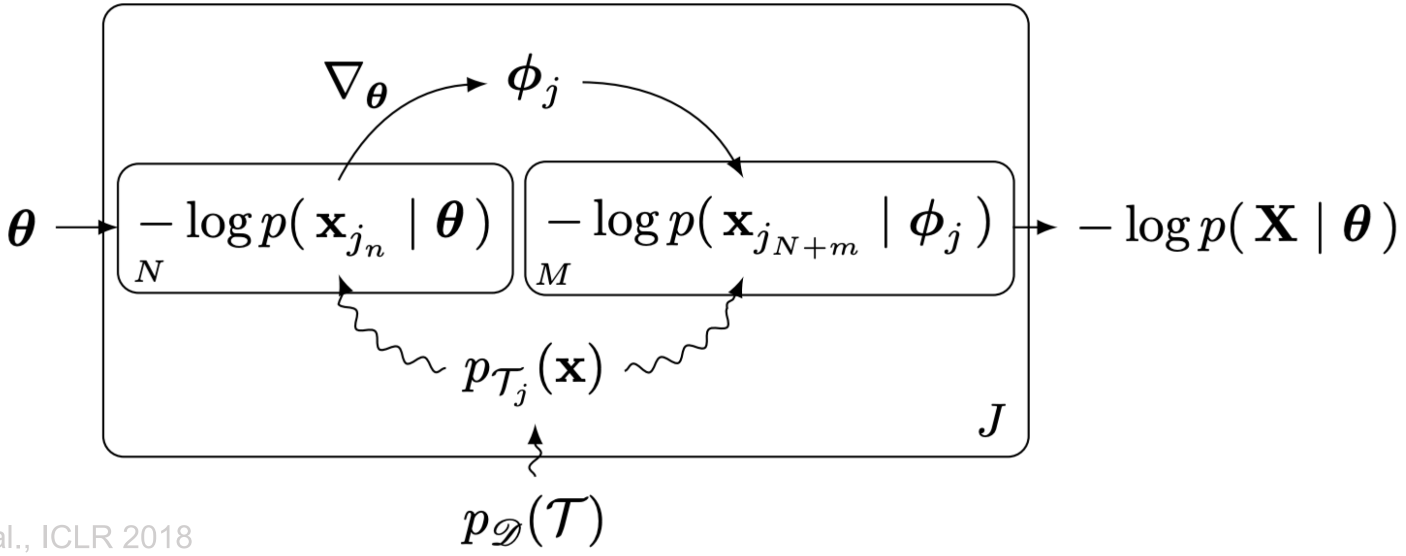
MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy 1-shot	5-way Accuracy 5-shot
fine-tuning baseline	28.86 ± 0.54%	49.79 ± 0.79%
nearest neighbor baseline	41.08 ± 0.70%	51.04 ± 0.65%
matching nets (Vinyals et al., 2016)	43.56 ± 0.84%	55.31 ± 0.73%
meta-learner LSTM (Ravi & Larochelle, 2017)	43.44 ± 0.77%	60.60 ± 0.71%
MAML, first order approx. (ours)	**48.07 ± 1.75%**	**63.15 ± 0.91%**
MAML (ours)	**48.70 ± 1.84%**	**63.11 ± 0.92%**

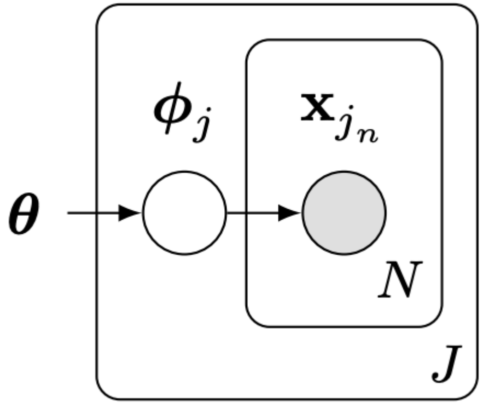
Finn, Abbeel, Levine, ICML 2017

MAML from a Probabilistic Standpoint

- Training points: $\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_N} \sim p_{\mathcal{T}_j}(\mathbf{x})$, testing points: $\mathbf{x}_{j_{N+1}}, \ldots, \mathbf{x}_{j_{N+M}} \sim p_{\mathcal{T}_j}(\mathbf{x})$
- MAML with log-likelihood loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{J}\sum_j \left[\frac{1}{M} \sum_m -\log p\Big(\mathbf{x}_{j_{N+m}} \mid \underbrace{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \frac{1}{N} \sum_n -\log p\left(\mathbf{x}_{j_n} \mid \boldsymbol{\theta}\right)}_{\boldsymbol{\phi}_j}\Big) \right]$$

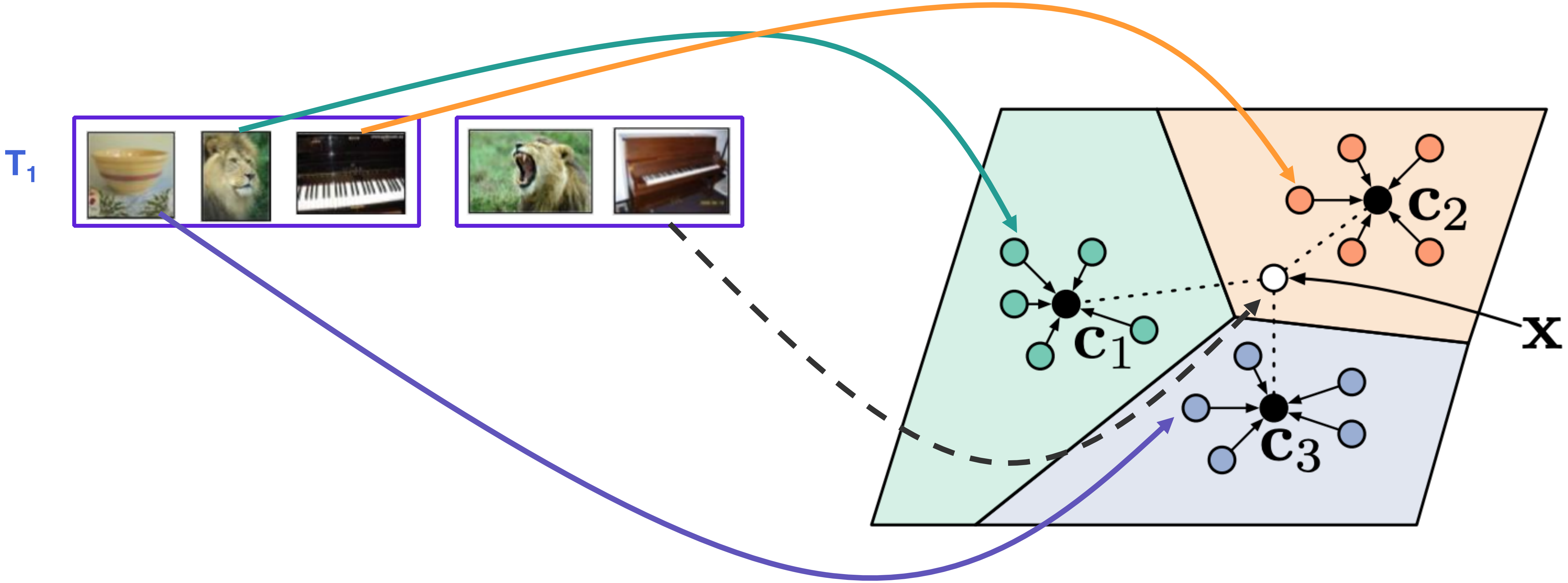




Hierarchical Bayes

Grant et al., ICLR 2018

Prototype-based Meta-learning



Snell, Swersky, Zemel, NIPS 2017

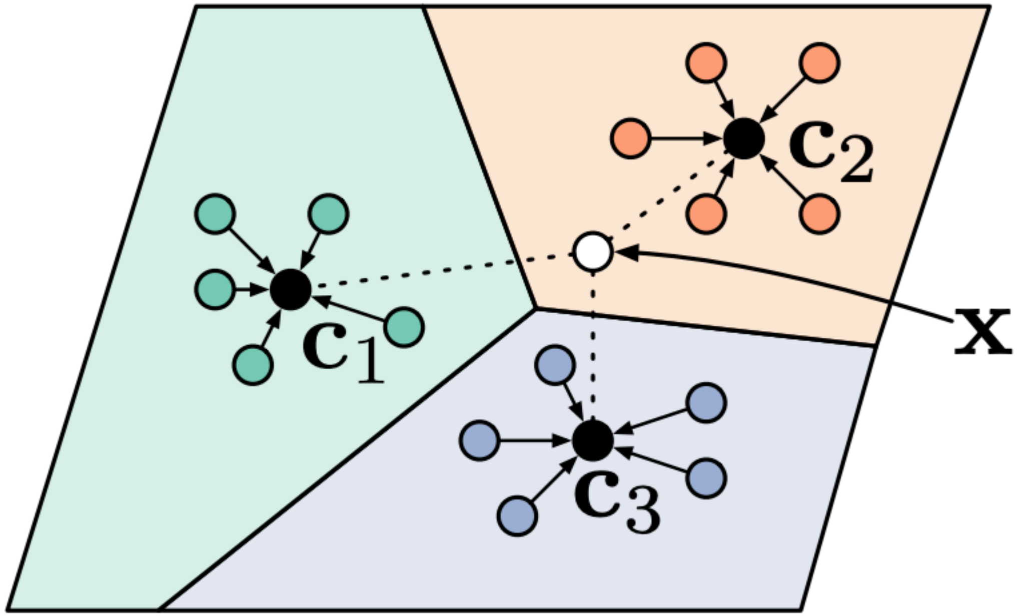
Prototype-based Meta-learning

Prototypes:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$$

Predictive distribution:

$$p_\phi(y = k \mid \mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'}))}$$



Snell, Swersky, Zemel, NIPS 2017

Does Prototype-based Meta-learning Work?

Omniglot

Model	Dist.	Fine Tune	5-way Acc. 1-shot	5-way Acc. 5-shot	20-way Acc. 1-shot	20-way Acc. 5-shot
Matching Networks [32]	Cosine	N	98.1%	98.9%	93.8%	98.5%
Matching Networks [32]	Cosine	Y	97.9%	98.7%	93.5%	98.7%
Neural Statistician [7]	-	N	98.1%	99.5%	93.2%	98.1%
MAML [9]*	-	N	98.7%	**99.9%**	95.8%	**98.9%**
Prototypical Networks (Ours)	Euclid.	N	**98.8%**	99.7%	**96.0%**	**98.9%**

mini-ImageNet

Model	Dist.	Fine Tune	5-way Acc. 1-shot	5-way Acc. 5-shot
Baseline Nearest neighbors*	Cosine	N	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
Matching Networks [32]*	Cosine	N	$43.40 \pm 0.78\%$	$51.09 \pm 0.71\%$
Matching Networks FCE [32]*	Cosine	N	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
Meta-Learner LSTM [24]*	-	N	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
MAML [9]	-	N	$\mathbf{48.70 \pm 1.84\%}$	$63.15 \pm 0.91\%$
Prototypical Networks (Ours)	Euclid.	N	$\mathbf{49.42 \pm 0.78\%}$	$\mathbf{68.20 \pm 0.66\%}$

Snell, Swersky, Zemel, NIPS 2017

"Rapid Learning or Feature Reuse?"

Published as a conference paper at ICLR 2020

RAPID LEARNING OR FEATURE REUSE? TOWARDS UNDERSTANDING THE EFFECTIVENESS OF MAML

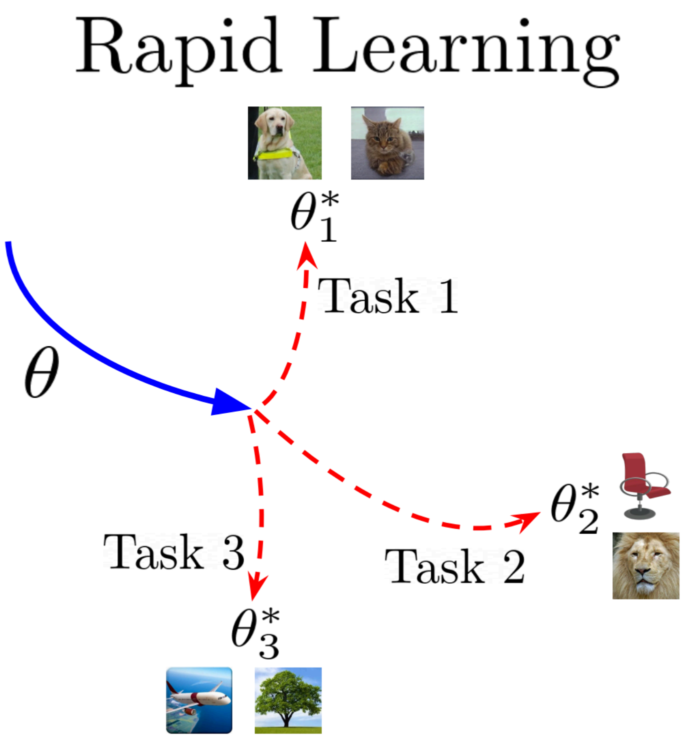
Aniruddh Raghu *
MIT
araghu@mit.edu

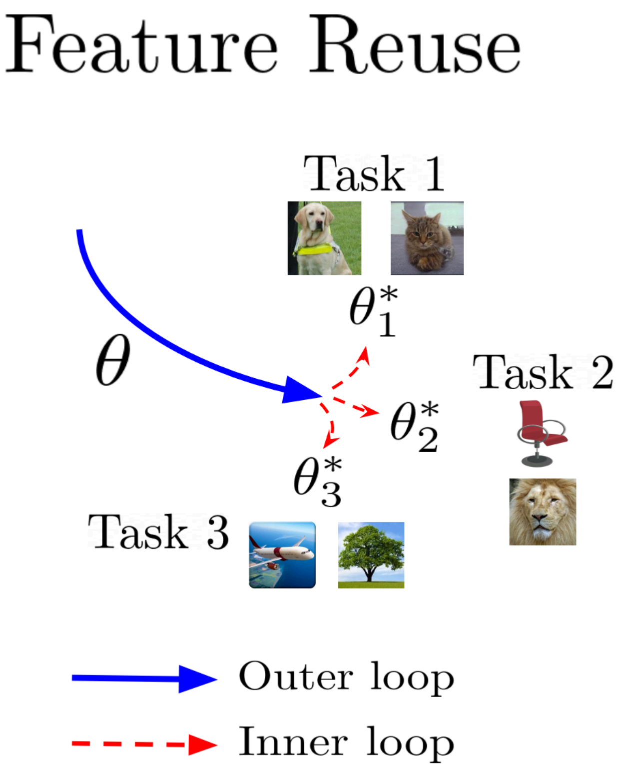
Maithra Raghu *
Cornell University & Google Brain
maithrar@gmail.com

Samy Bengio
Google Brain

Oriol Vinyals
DeepMind

“Rapid Learning or Feature Reuse?”



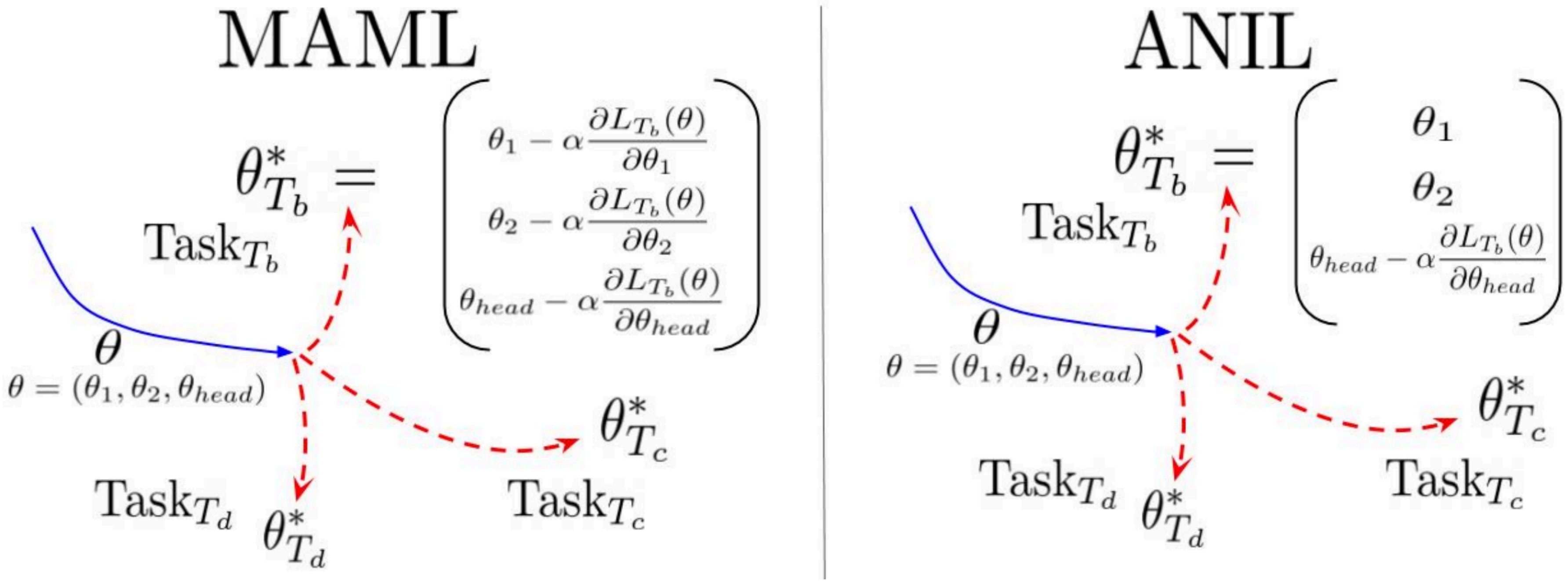


Adaptation is the main contributor to the performance

Good representations is the main contributor to the performance

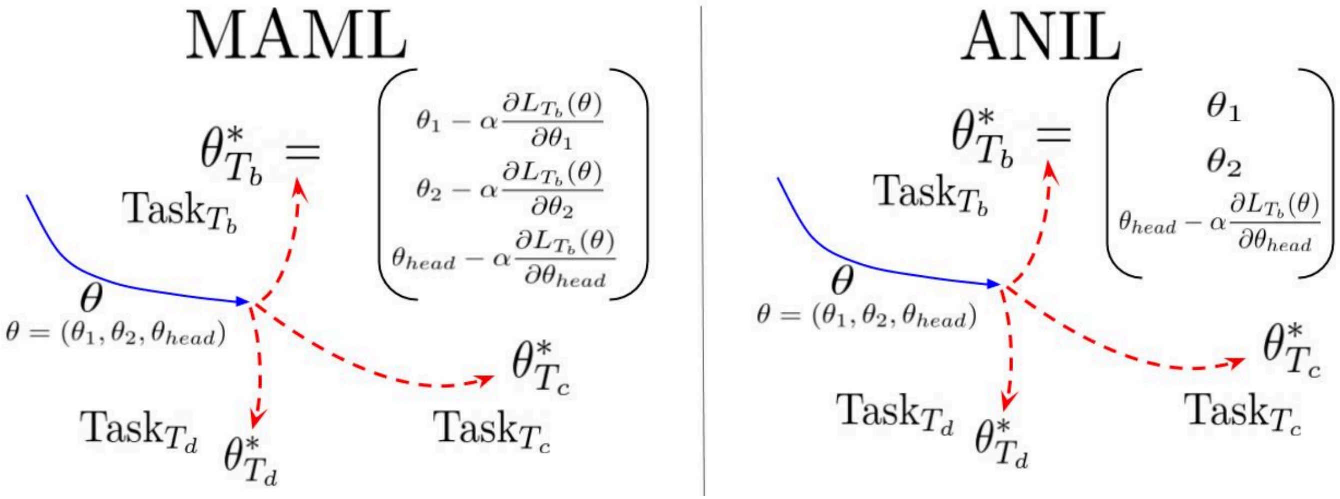
Ragu et al., ICLR 2020

“Rapid Learning or Feature Reuse?”



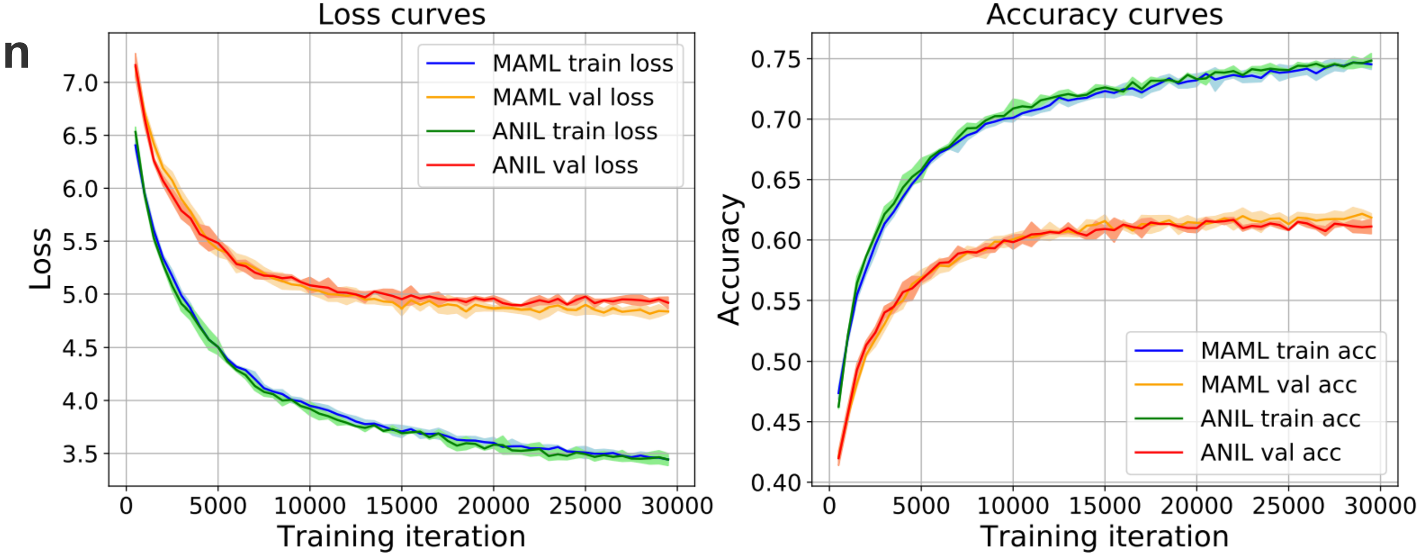
Ragu et al., ICLR 2020

“Rapid Learning or Feature Reuse?”



MiniImageNet-5way-5shot

No visible difference in performance between MAML and ANIL

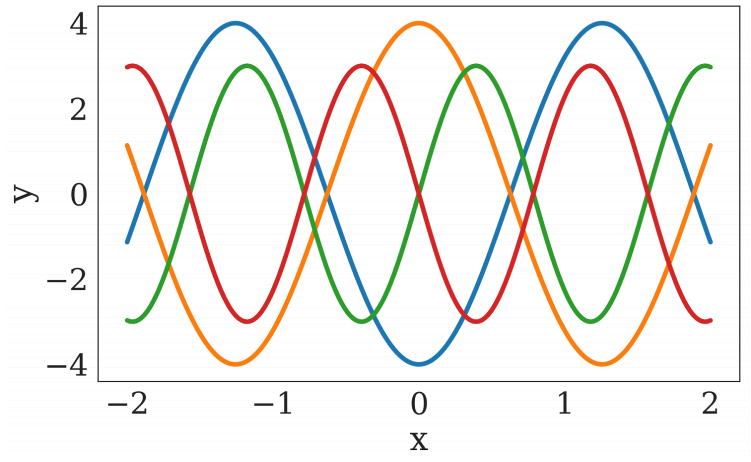


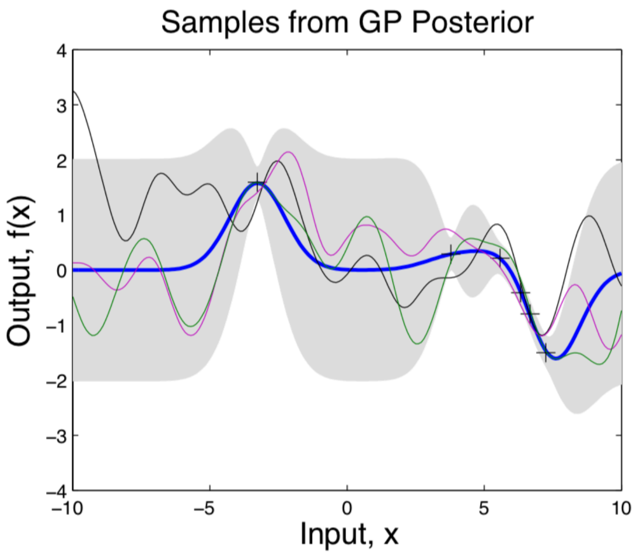
More detailed analysis of the representations learned by MAML vs ANIL at different levels is in the paper

Ragu et al., ICLR 2020

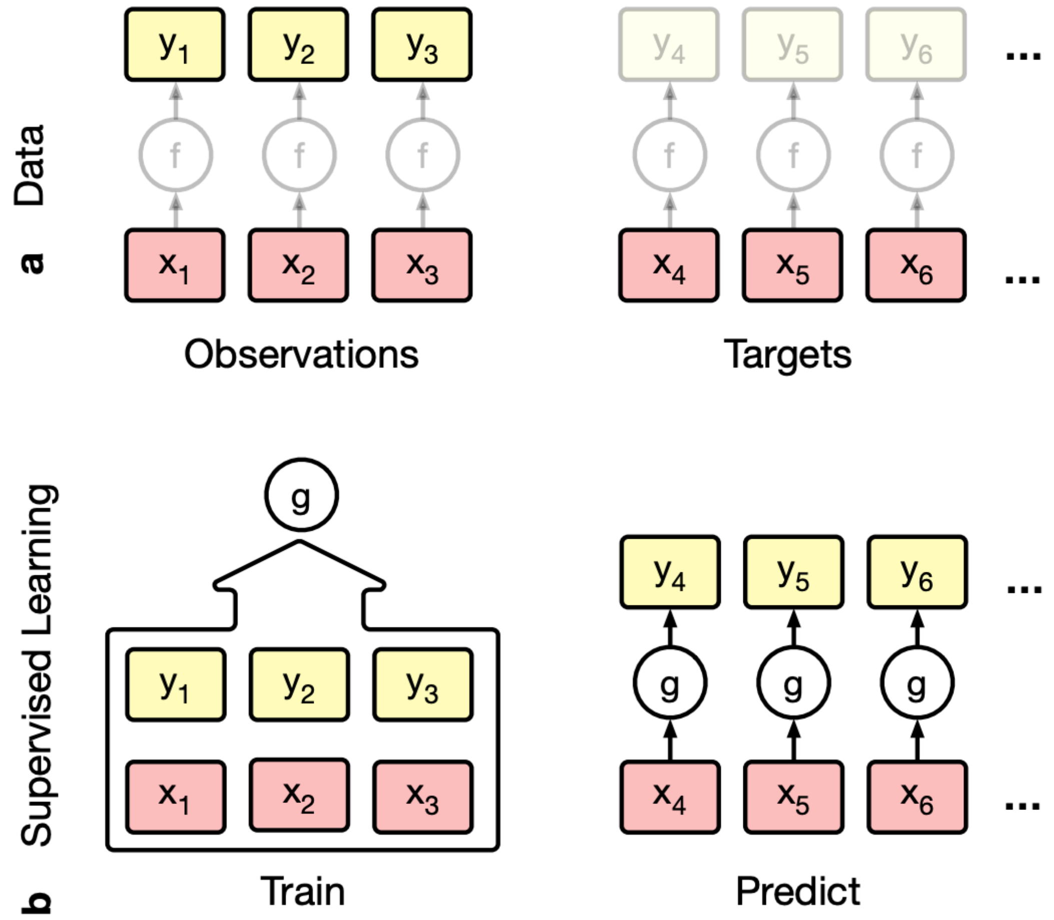
Drawing parallels between meta-learning and GPs

- In few-shot learning:
  - Learn to identify functions that generated the data from just a few examples.
  - The function class and the adaptation rule encapsulate our prior knowledge.
- Recall Gaussian Processes (GPs):
  - Given a few (x, y) pairs, we can compute the predictive mean and variance.
  - Our prior knowledge is encapsulated in the kernel function.

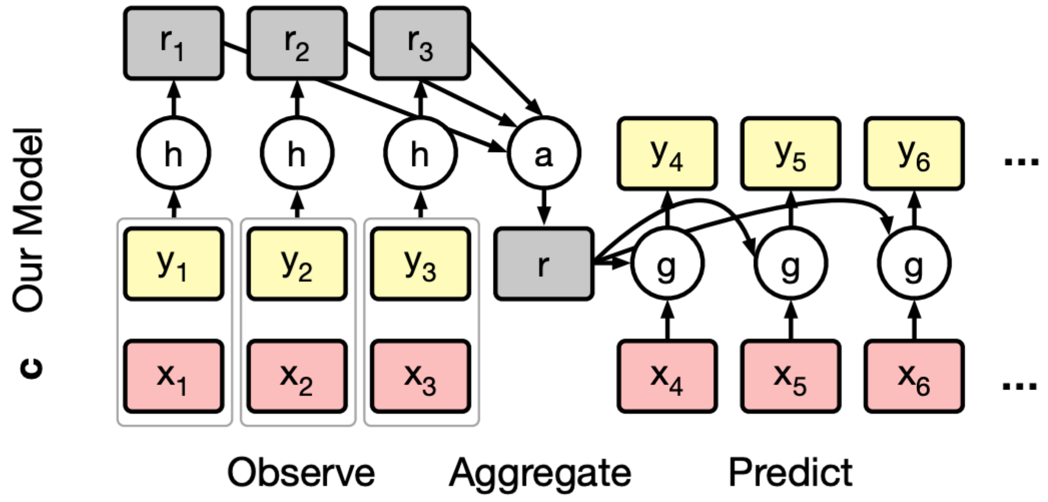




Conditional Neural Processes

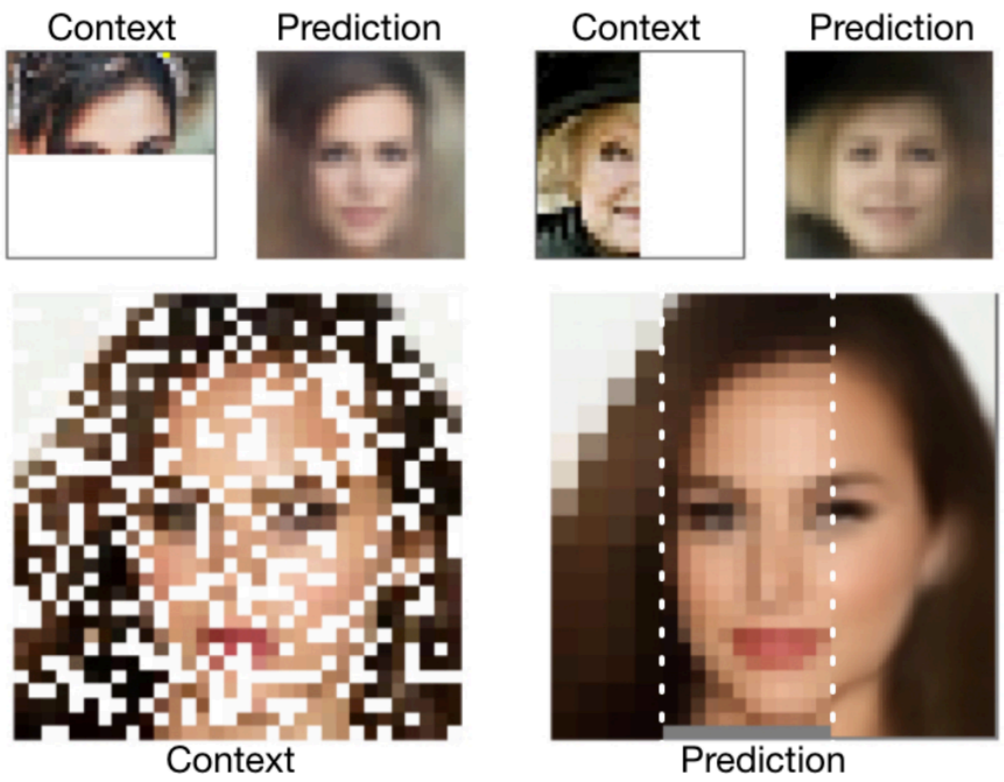


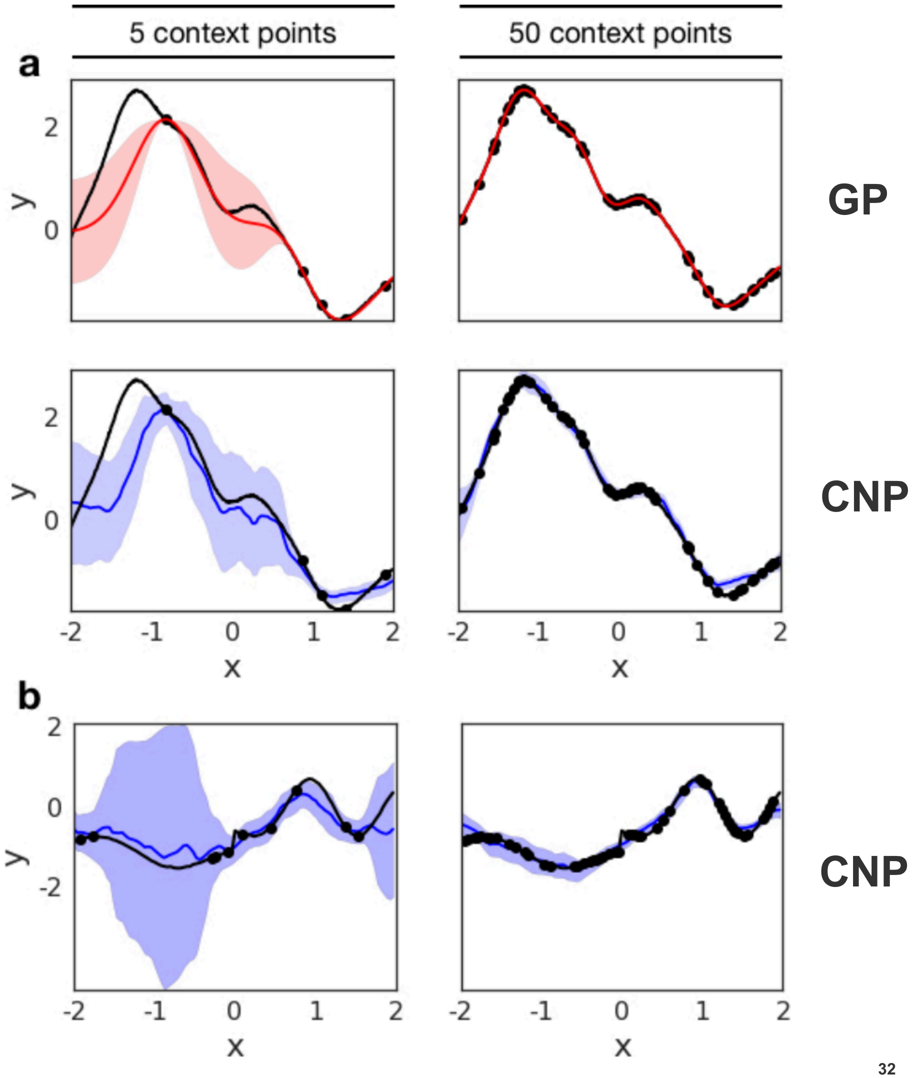
CNP architecture:



Garnelo et al., ICML 2018

Conditional Neural Processes





Garnelo et al., ICML 2018

On software packages for meta-learning

- A lot of research code releases (code is fragile and sometimes broken)
- A few notable libraries that implement a few specific methods:
  - Torchmeta (https://github.com/tristandeleu/pytorch-meta)
  - Learn2learn (https://github.com/learnables/learn2learn)
  - Higher (https://github.com/facebookresearch/higher)
- New!

META
BLOCKS

A Modular Toolbox for Accelerating Meta-Learning Research 🚀

https://github.com/alshedivat/meta-blocks

✓ Library is actively developed
✓ Very modular and FAST
✓ Planned support for many algorithms and meta-RL

Running a tutorial next week!
(drop me an email if interested)

Takeaways

- Many real-world scenarios require building adaptive systems and cannot be solved using "learn-once" standard ML approach.
- Learning-to-learn (or meta-learning) attempts extend ML to rich multitask scenarios—instead of learning a function, learn a learning algorithm.
- Two families of widely popular methods:
  - Gradient-based meta-learning (MAML and such)
  - Prototype-based meta-learning (Protonets, Neural Processes, …)
  - Many hybrids, extensions, improvements (CAIVA, MetaSGD, …)
- Is it about adaptation or learning good representations? Still unclear and depends on the task; having good representations might be enough.
- Meta-learning can be used as a mechanism for causal discovery.
  (See Bengio et al., 2019.)

Elements of Meta-RL

- What is meta-RL and why does it make sense?
- On-policy and off-policy meta-RL
- Continuous adaptation

Recall the definition of learning-to-learn

- Standard learning: Given a distribution over examples (single task), learn a function that minimizes the loss

$$\hat{\phi} = \arg\min_{\phi} \mathbb{E}_{z \sim \mathcal{D}} \left[l(f_\phi(z)) \right]$$

- Learning-to-learn: Given a distribution over tasks, output an adaptation rule that can be used at test time to generalize from a task description

distribution over tasks/datasets

adaptation rule takes a task description as input and outputs a model

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}_{T \sim \mathcal{P}} \left\{ \mathcal{L}_T[g_\theta(T)] \right\}, \quad \text{where}$$

$$\mathcal{L}_T[g_\theta(T)] := \mathbb{E}_{z \sim \mathcal{D}_T} \left[l(f_\phi(z)) \right], \; \phi := g_\theta(T)$$

distribution over examples for task T

Recall the definition of learning-to-learn

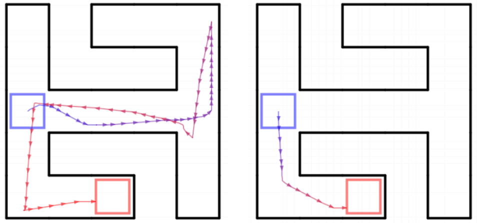
- Standard learning: Given a distribution over examples (single task), learn a function that minimizes the loss

$$\hat{\phi} = \arg\min_{\phi} \mathbb{E}_{z\sim\mathcal{D}} \left[l(f_{\phi}(z)) \right]$$

- Learning-to-learn: Given a distribution over tasks, output an adaptation rule that can be used at test time to generalize from a task description

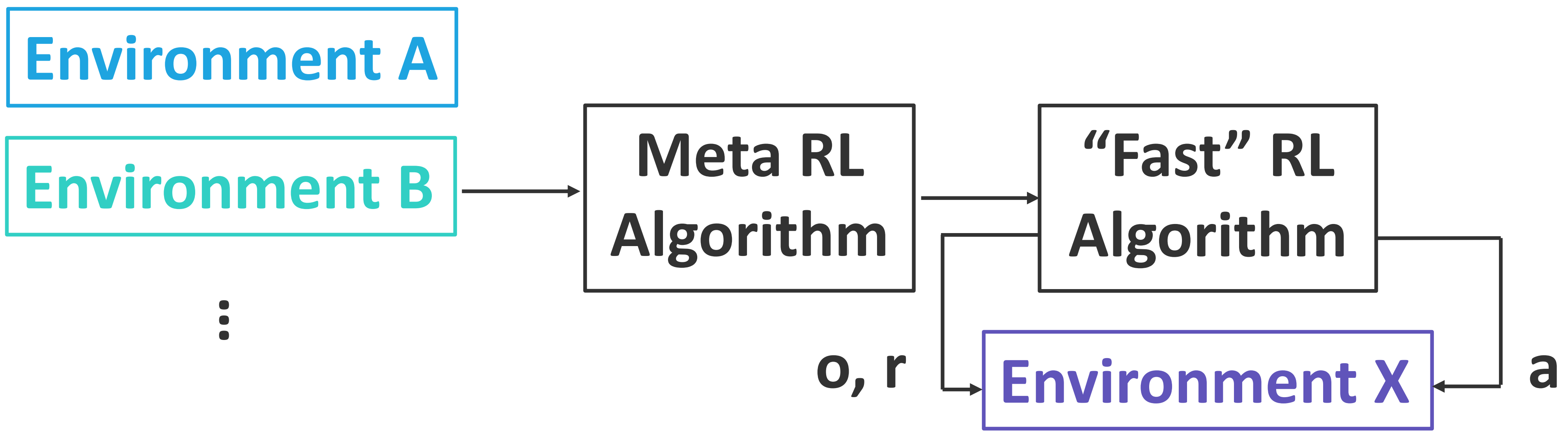
$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}_{T\sim\mathcal{P}} \left\{ \mathcal{L}_T[g_{\theta}(T)] \right\}, \text{ where } \mathcal{L}_T[g_{\theta}(T)] := \mathbb{E}_{z\sim\mathcal{D}_T} \left[l(f_{\phi}(z)) \right], \phi := g_{\theta}(T)$$

- Meta reinforcement learning (RL): Given a distribution over environments, train a policy update rule that can solve new environments given only limited or no initial experience.

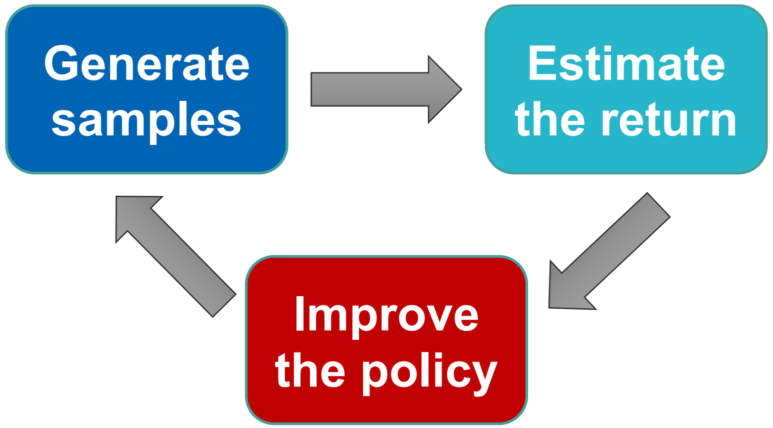


Duan et al., 2016

Meta-learning for RL



On-policy RL: Quick Recap



REINFORCE algorithm:

1. sample $\{\tau_i\}_{i=1}^{N}$ under $\pi_\theta(a_t \mid s_t)$
2. $\hat{J}(\theta) = \sum_i \left(\sum_t \log \pi_\theta(a_{i,t} \mid s_{i,t})\right)\left(\sum_t r(s_{i,t}, a_{i,t})\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta \hat{J}(\theta)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right)\right]$$

On-policy Meta-RL: MAML (again!)

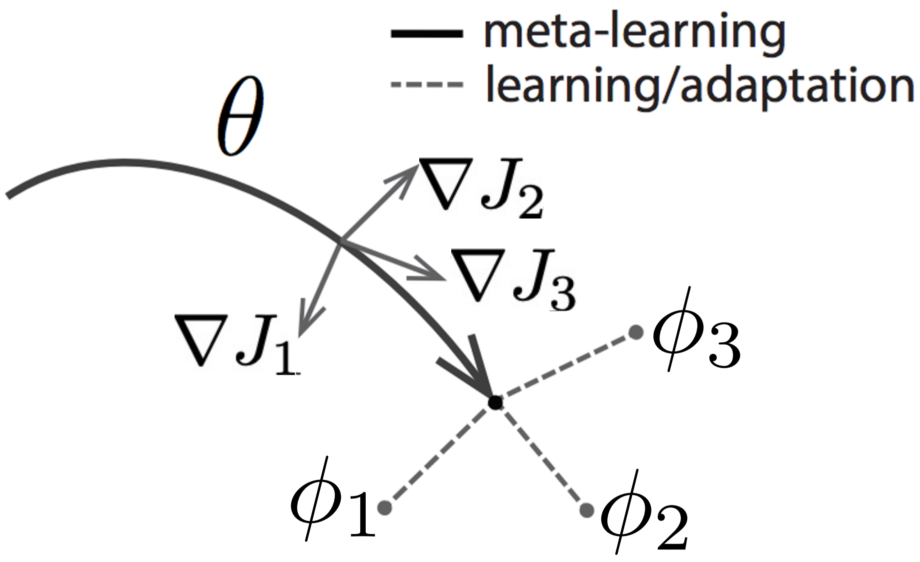
- Start with a common **policy** initialization θ
- Given a new task T_i, collect data using initial **policy**, then adapt using a gradient step:

$$\phi_i = g_\theta(T_i) := \theta - \alpha \nabla_\theta J_{T_i}(\theta)$$

- Meta-training is learning a shared initialization for all tasks:

$$\min_\theta \sum_{T_i \sim \mathcal{P}} J_{T_i}^{\text{test}}\left(\theta - \alpha \nabla_\theta J_{T_i}^{\text{train}}(\theta)\right)$$

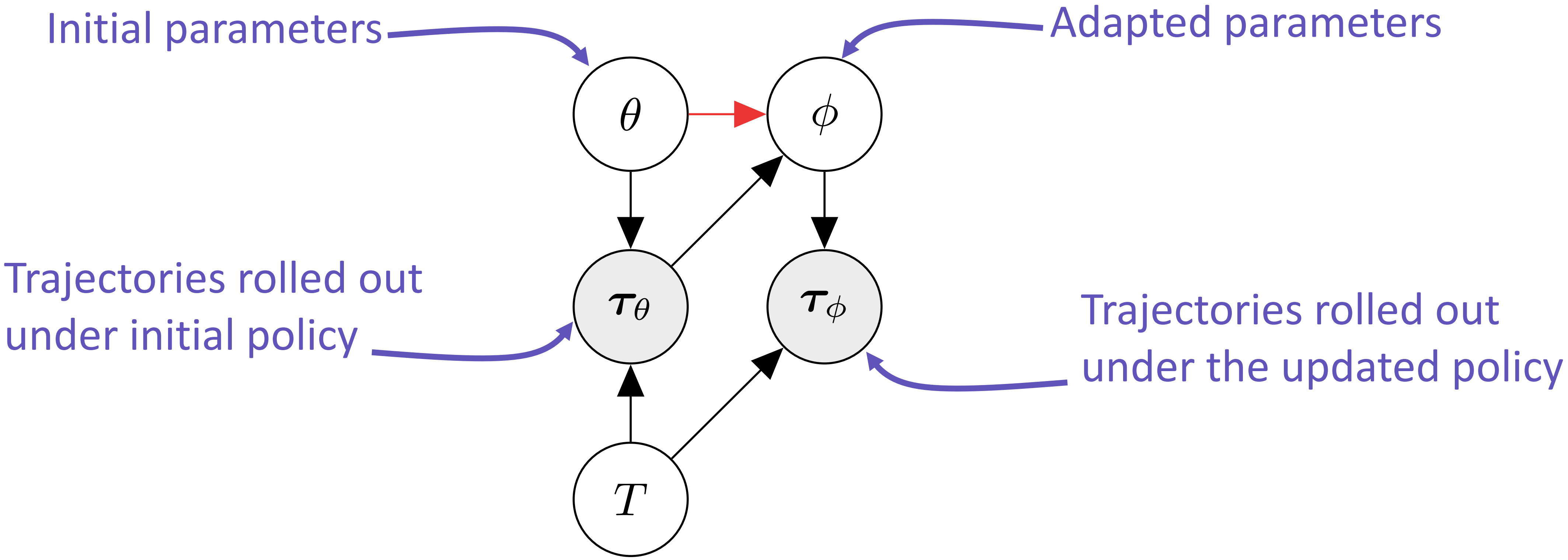
Intuition:



Finn, Abbeel, Levine, ICML 2017

Adaptation as Inference

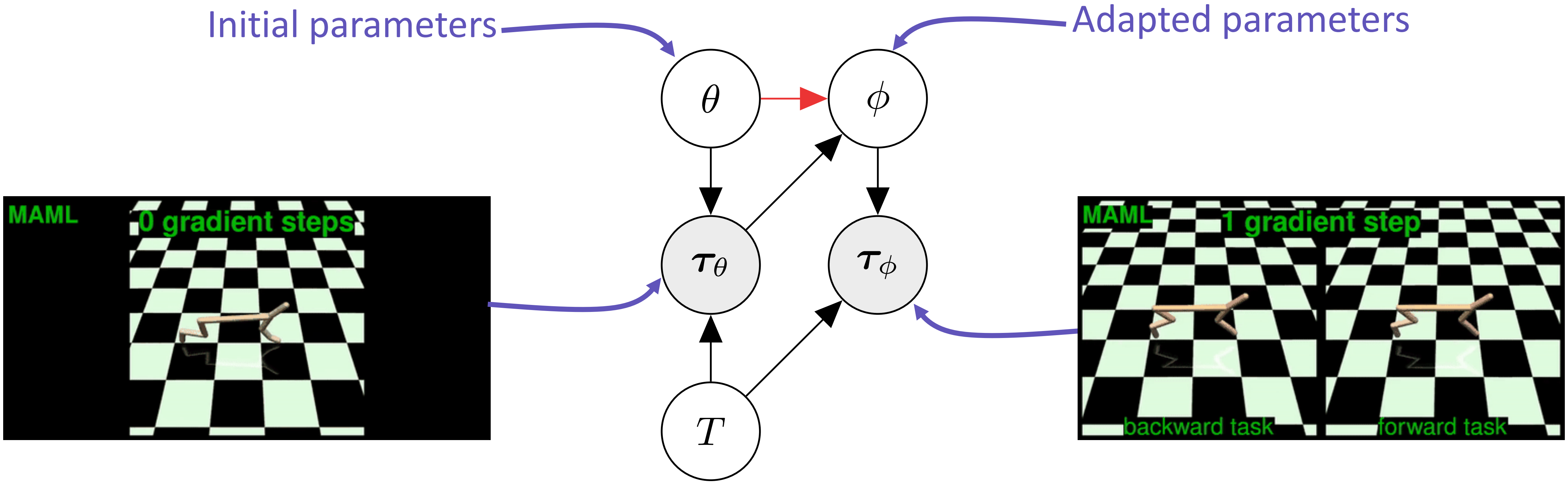
Treat policy parameters, tasks, and all trajectories as **random variables**



meta-learning = learning a prior and adaptation = inference

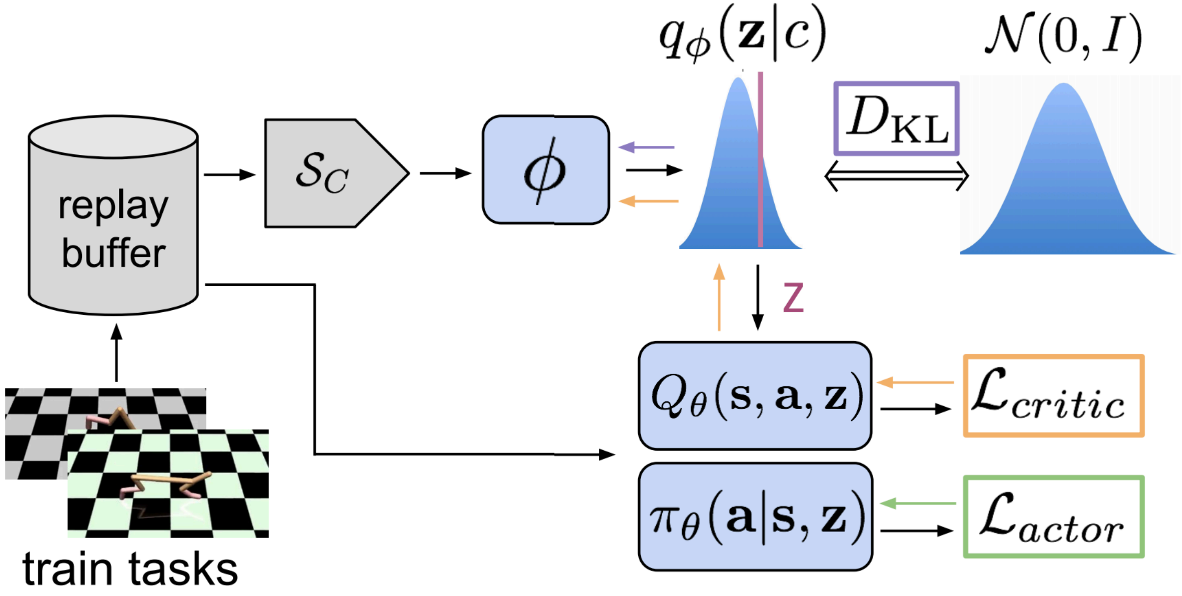
Adaptation as Inference

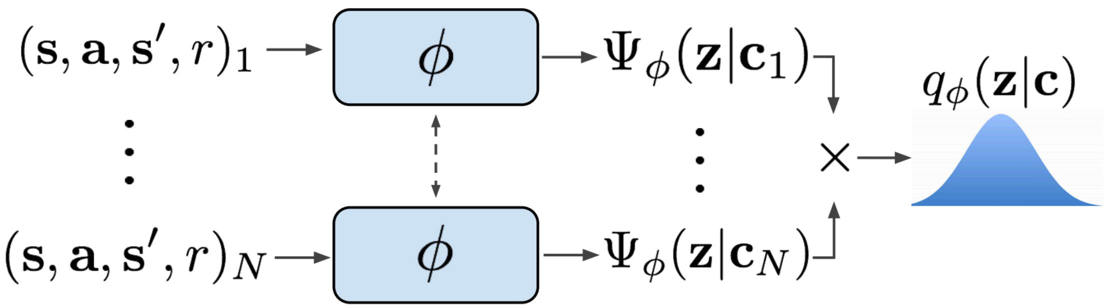
Treat policy parameters, tasks, and all trajectories as **random variables**



meta-learning = learning a prior and adaptation = inference

Off-policy meta-RL: PEARL





$$\mathbb{E}_{\mathcal{T}}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})}[R(\mathcal{T}, \mathbf{z}) + \beta D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})||p(\mathbf{z}))]]$$

$$\mathcal{L}_{critic} = \mathbb{E}_{\substack{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{B} \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}}[Q_\theta(\mathbf{s}, \mathbf{a}, \mathbf{z}) - (r + \bar{V}(\mathbf{s}', \bar{\mathbf{z}}))]^2$$

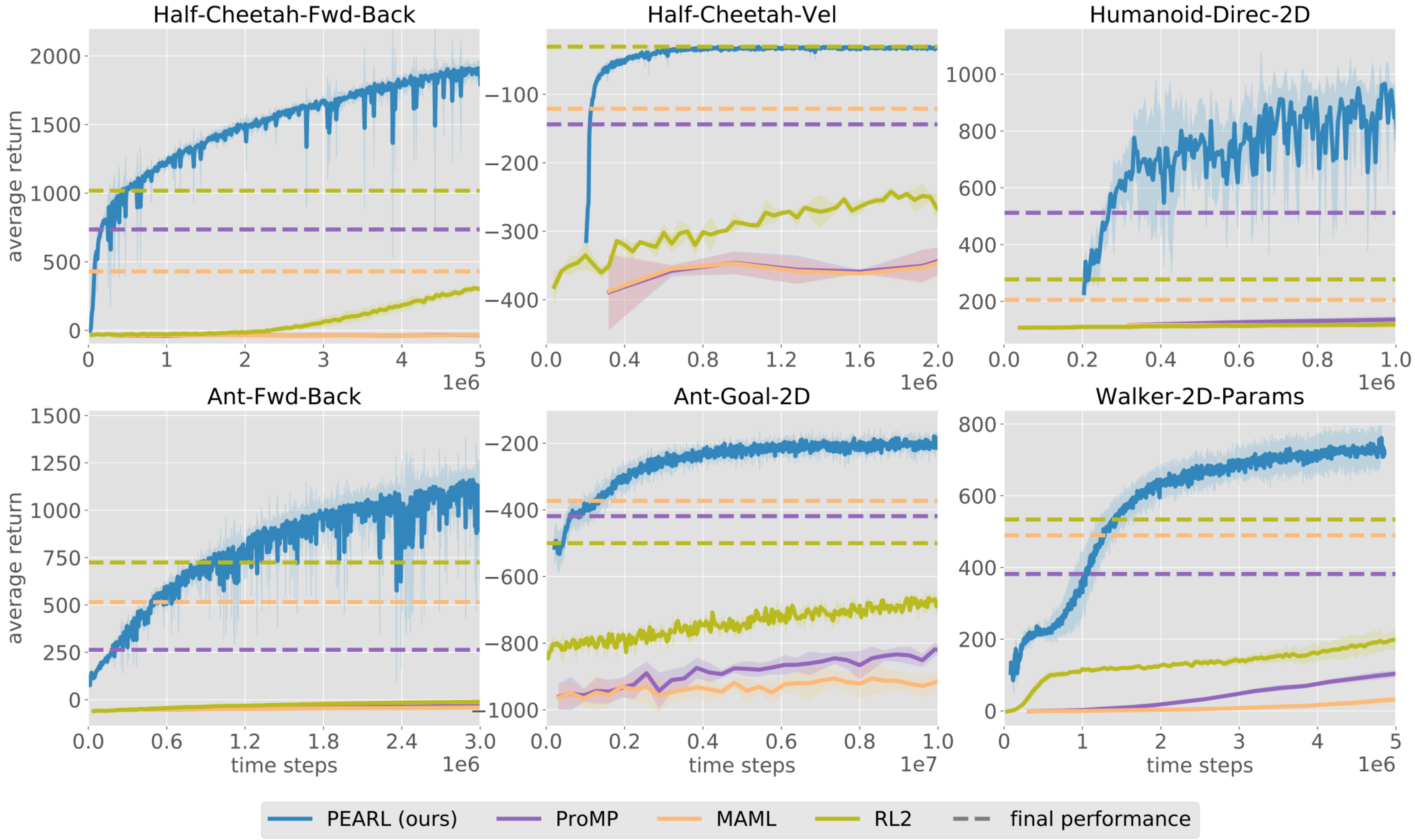
$$\mathcal{L}_{actor} = \mathbb{E}_{\substack{\mathbf{s}\sim\mathcal{B},\mathbf{a}\sim\pi_\theta \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}}\left[D_{\text{KL}}\left(\pi_\theta(\mathbf{a}|\mathbf{s}, \bar{\mathbf{z}}) \middle\| \frac{\exp(Q_\theta(\mathbf{s}, \mathbf{a}, \bar{\mathbf{z}}))}{\mathcal{Z}_\theta(\mathbf{s})}\right)\right]$$

Key points:

- Infer latent representations **z** of each task from the trajectory data.
- The inference network **q** is decoupled from the policy, which enables off-policy learning.
- All objectives involve the inference and policy networks.

Rakelly*, Zhou*, et al., ICML 2019

Off-policy meta-RL: PEARL

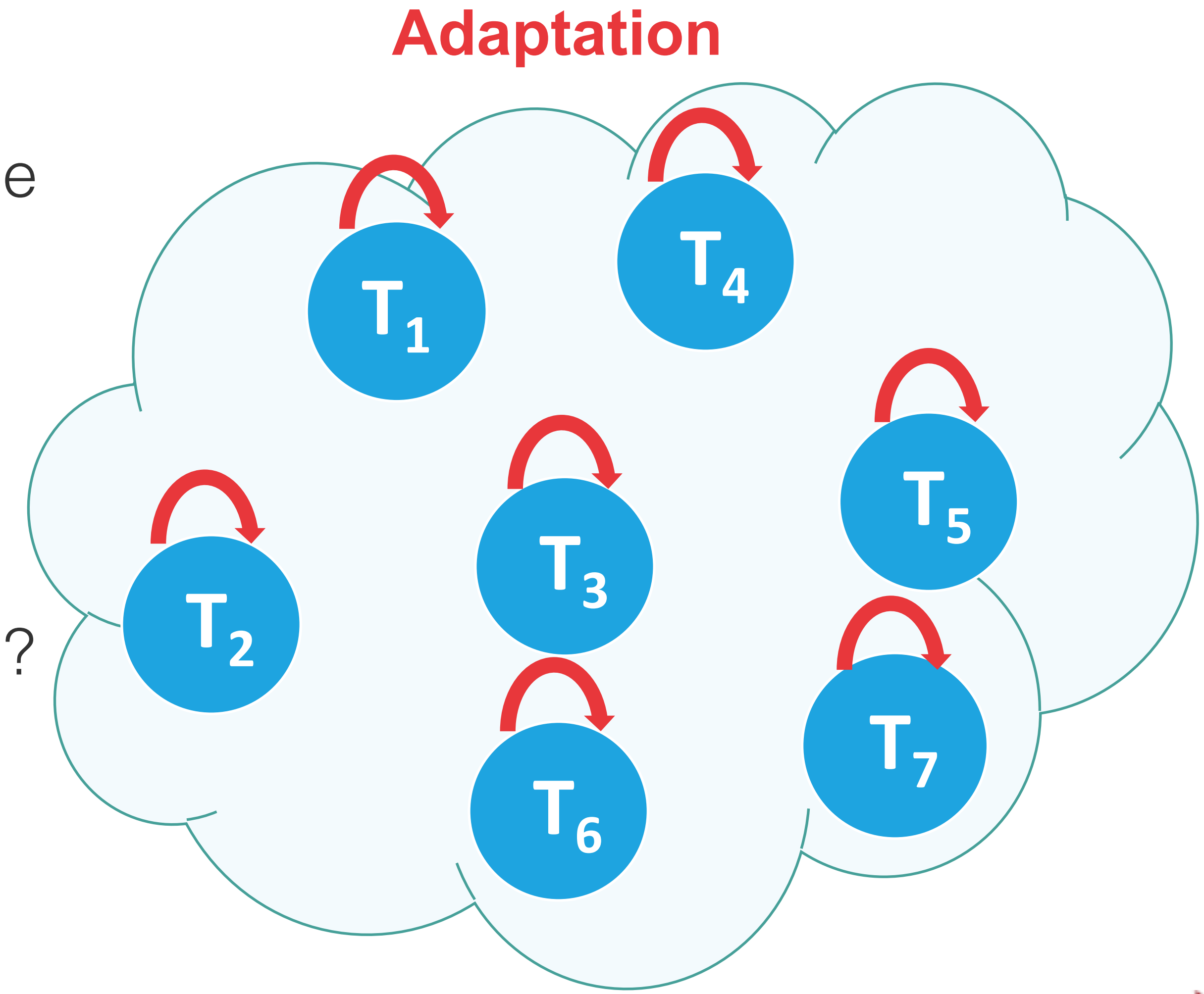


Adaptation in nonstationary environments

Classical few-shot learning setup:

- The tasks are i.i.d. samples from some underlying distribution.
- Given a new task, we get to interact with it before adapting.

- What if we are in a nonstationary environment (i.e. changing over time)? Can we still use meta-learning?

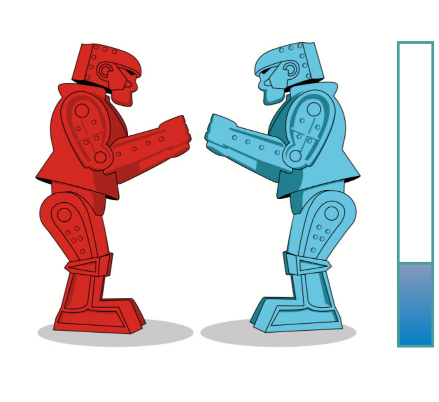


Al-Shedivat et al., ICLR 2018

Adaptation in nonstationary environments

Example: adaptation to a learning opponent

Agent Opponent

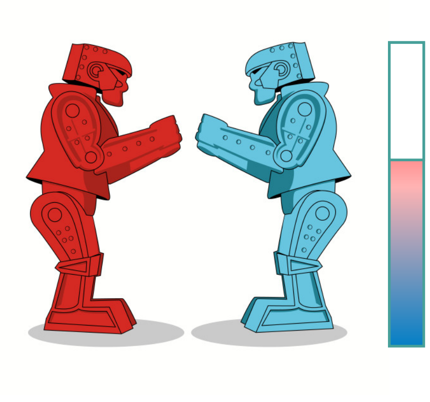


Opponent's competence

Easy

Opponent is learning

Round 1



Medium

Round 2

Each new round is a new task. Nonstationary environment is a sequence of tasks.

Figure source: https://www.teepublic.com/tapestry/631249-rock-em-

Adaptation in nonstationary environments

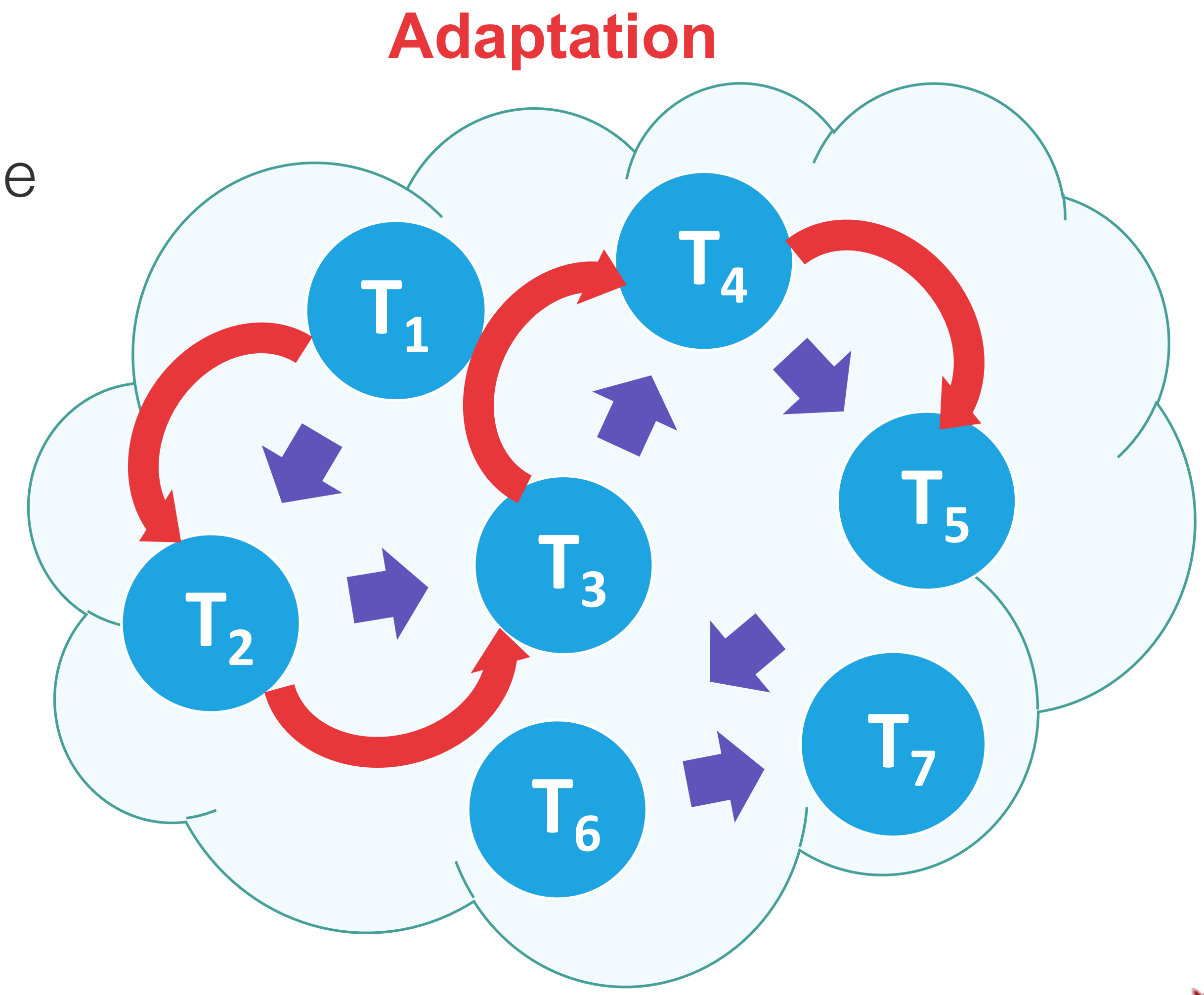
Classical few-shot learning setup:

- The tasks are i.i.d. samples from some underlying distribution.

Continuous adaptation setup:

- The tasks are sequentially dependent.

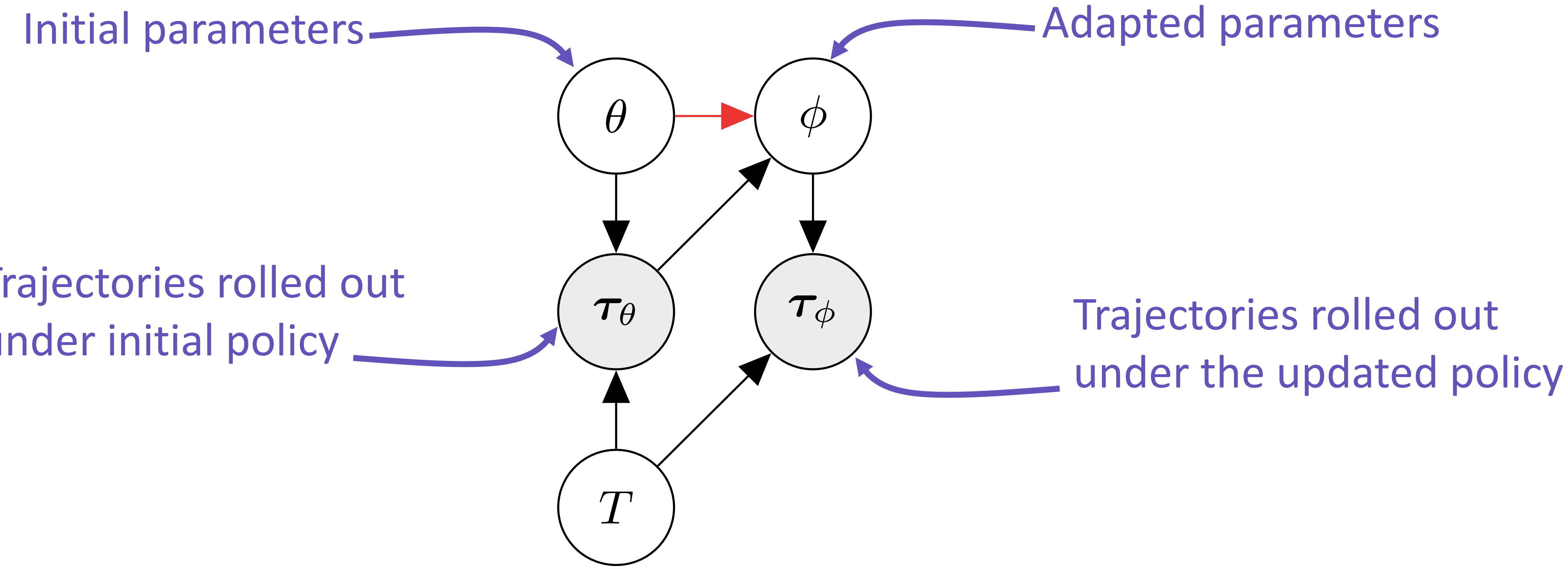
⇒ meta-learn to exploit dependencies



Al-Shedivat et al., ICLR 2018

Adaptation as Inference

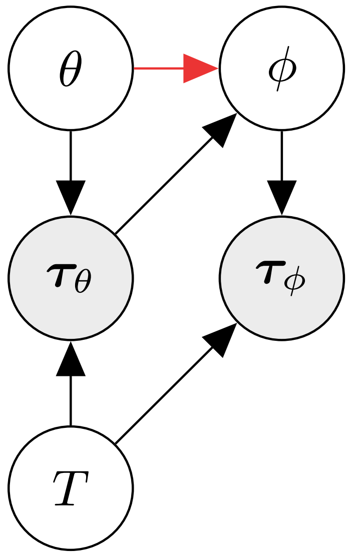
Treat policy parameters, tasks, and all trajectories as **random variables**



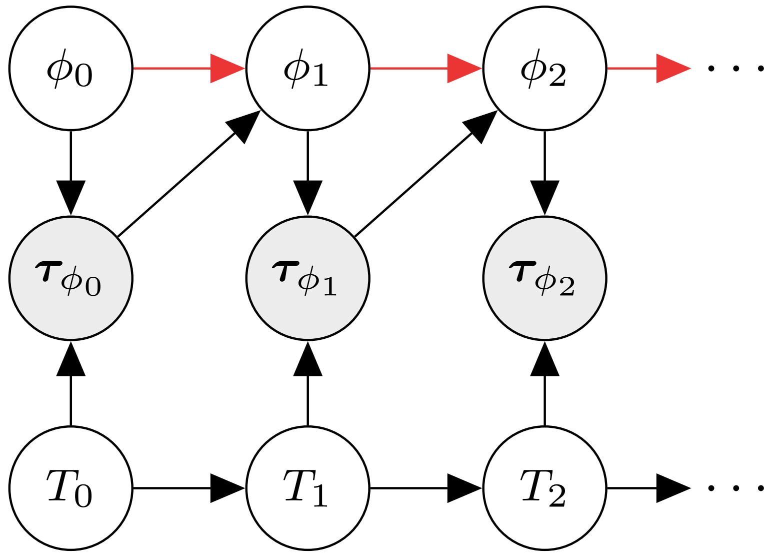
meta-learning = learning a prior and adaptation = inference

Continuous Adaptation to Nonstationarity

Treat policy parameters, tasks, and all trajectories as **random variables**



Classical few-shot learning

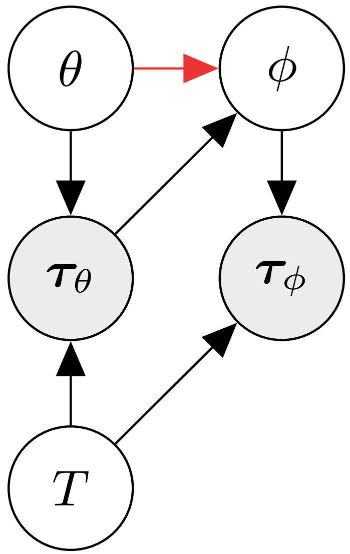


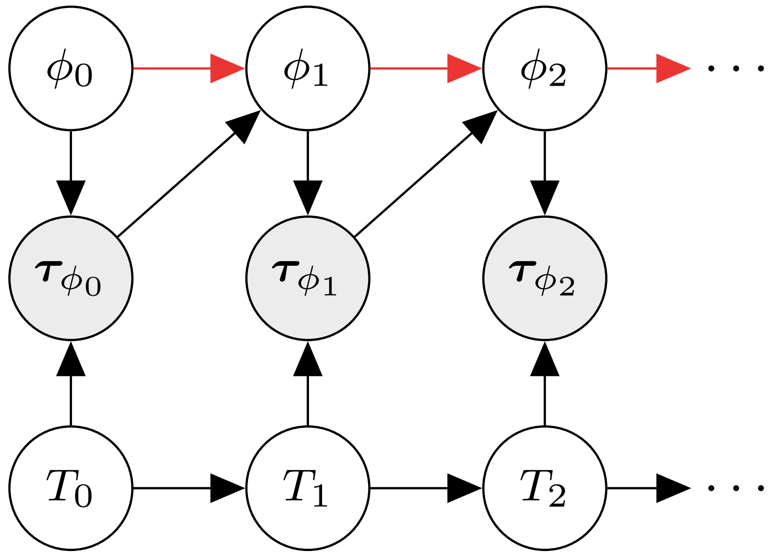
Continuous adaptation

Al-Shedivat et al., ICLR 2018

Continuous Adaptation to Nonstationarity

Treat policy parameters, tasks, and all trajectories as **random variables**





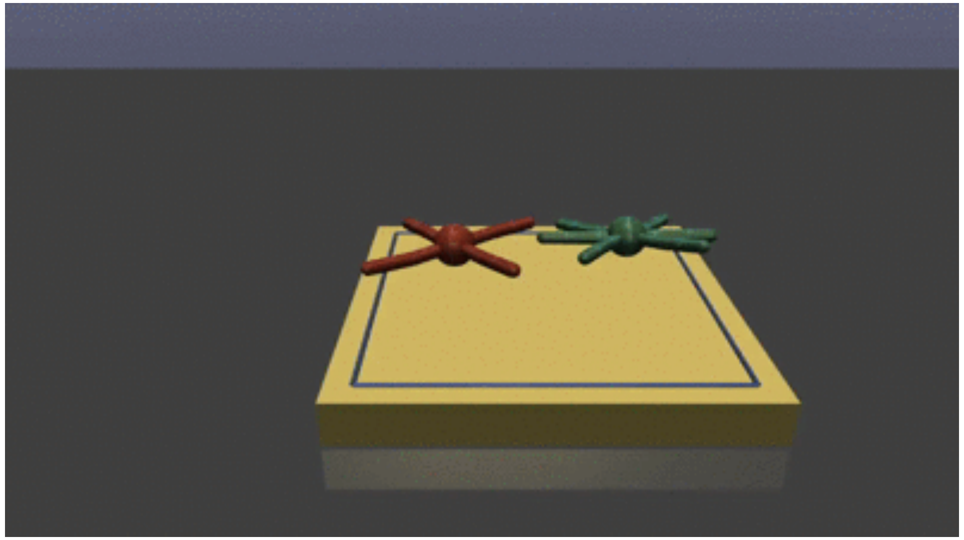
$$\min_{\theta} \mathbb{E}_{\mathcal{P}(T_i)} \left[\sum_{i=1}^{L} \mathcal{L}_{T_i}(\theta) \right]$$

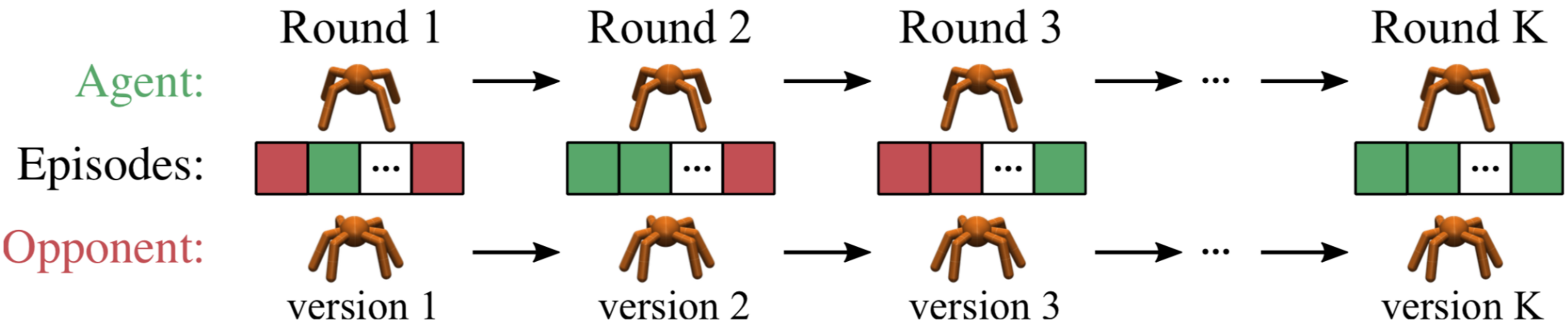
$$\min_{\theta} \mathbb{E}_{\mathcal{P}(T_0), \mathcal{P}(T_{i+1}|T_i)} \left[\sum_{i=1}^{L} \mathcal{L}_{T_i, T_{i+1}}(\theta) \right]$$

Al-Shedivat et al., ICLR 2018

Nonstationary Environments

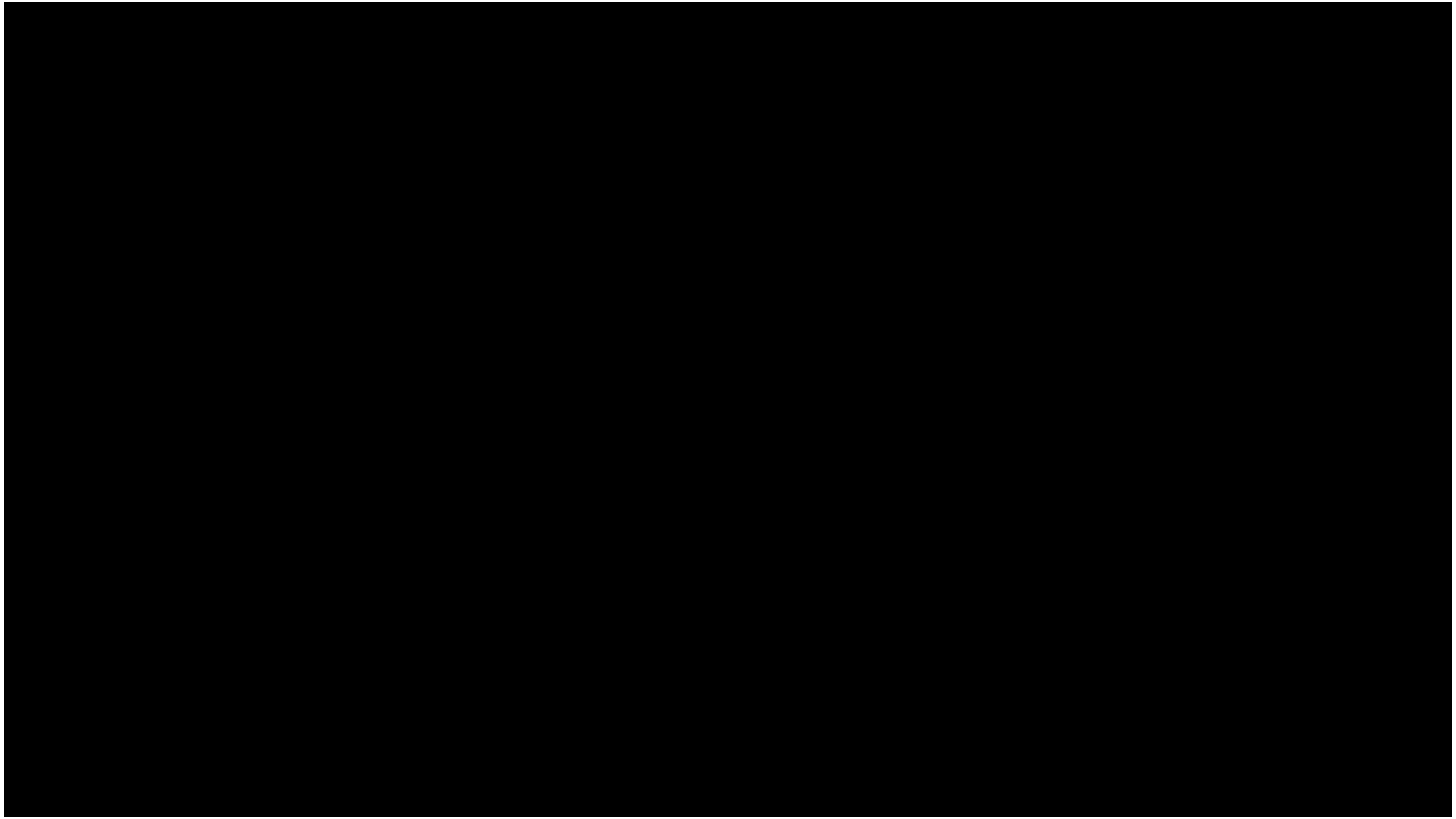
RoboSumo: a multiagent competitive env
an agent competes vs. an opponent, the opponent's behavior changes over time





Al-Shedivat et al., ICLR 2018

Continuous Adaptation Results



Al-Shedivat et al., ICLR 2018

Takeaways

- Learning-to-learn (or meta-learning) setup is particularly suitable for multi-task reinforcement learning
- Both on-policy and off-policy RL can be "upgraded" to meta-RL:
  - On-policy meta-RL is directly enabled by MAML
  - Decoupling task inference and policy learning enables off-policy methods
- Is it about fast adaptation or learning good multitask representations? (See discussion in Meta-Q-Learning: https://arxiv.org/abs/1910.00125)
- Probabilistic view of meta-learning allows to use meta-learning ideas beyond distributions of i.i.d. tasks, e.g., continuous adaptation.
- Very active area of research.

PETUUM

Carnegie Mellon University
School of Computer Science