CS 4803 / 7643: Deep Learning

Topics:

- Low-label ML Formulations

Zsolt Kira
Georgia Tech

Administrativia

- Projects!
- Project Check-in due **April 11th**
  - Will be graded pass/fail, if fail then you can address the issues
  - Counts for 5 points of project score
- Poster due date moved to **April 23rd** (last day of class)
  - No presentations
- Final submission due date **April 30th**

Types of Learning

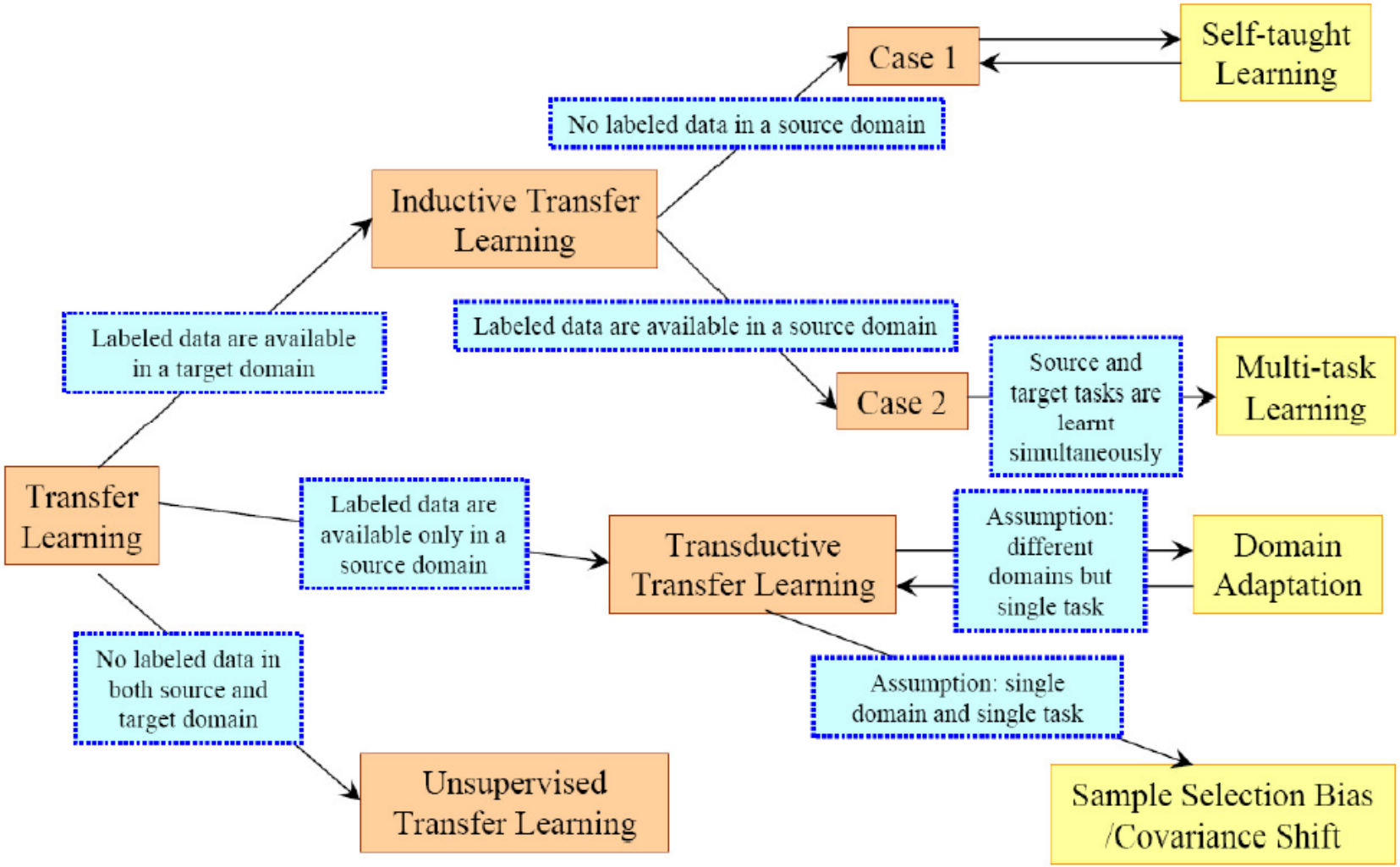
- Important note:
  - Your project should include doing something beyond just downloading open-source code and tuning hyper-parameters.
  - This can include:
    - implementation of additional approaches (if leveraging open-source code),
    - theoretical analysis, or
    - a thorough investigation of some phenomena.

- When using external resources, provide references to anything you used in the write-up!

But wait, there's more!

- **Transfer Learning**
- Domain adaptation
- Semi-supervised learning
- Zero-shot learning
- One/Few-shot learning
- Meta-Learning
- Continual / Lifelong-learning
- Multi-modal learning
- Multi-task learning
- Active learning
- …

Transfer Learning

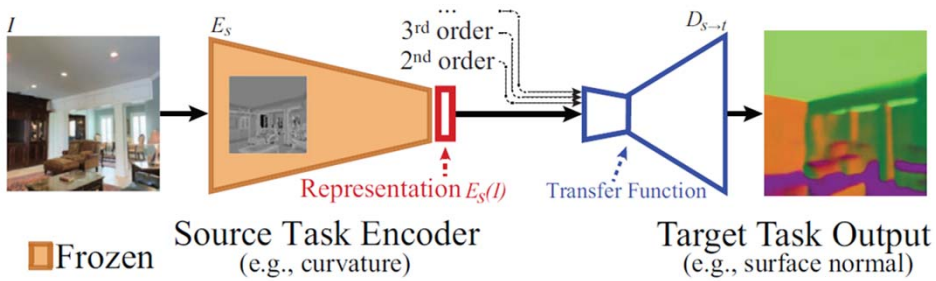


A Survey on Transfer Learning Sinno Jialin Pan and Qiang Yang *Fellow, IEEE*

Taskonomy

Builds graph of transferability between computer vision tasks:

1. Collect dataset of 4 million input images and labels for 26 vision tasks
   a. Surface normal, Depth estimation, Segmentation, 2D Keypoints, 3D pose estimation
2. Train convolutional autoencoder architecture for each tasks



http://taskonomy.stanford.edu/

Disentangling Task Transfer Learning, Amir R. Zamir, Alexander Sax*, William B. Shen*, Leonidas Guibas, Jitendra Malik, Silvio Savarese

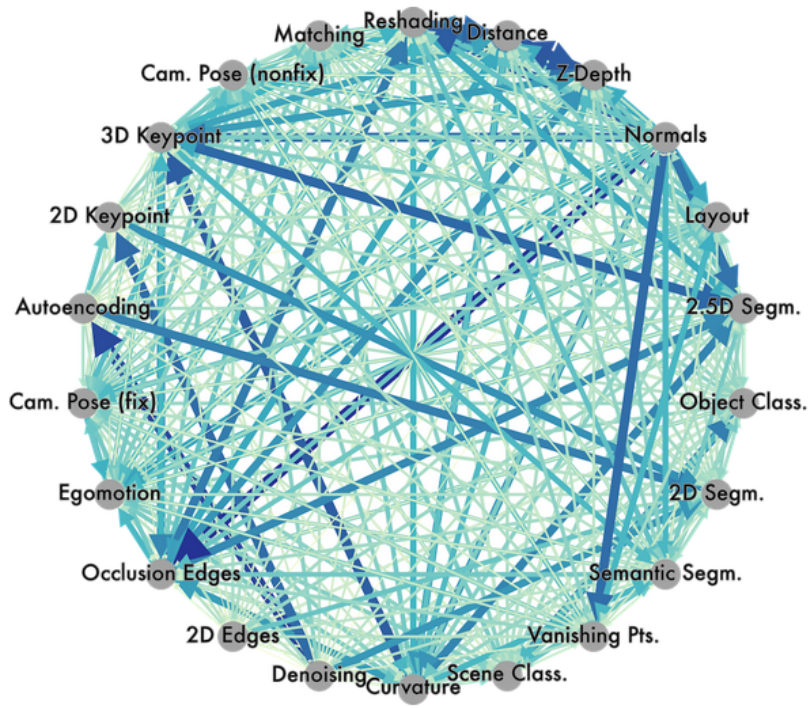
Slide Credit: Camilo & Higuera

Taskonomy

Builds graph of transferability between computer vision tasks:

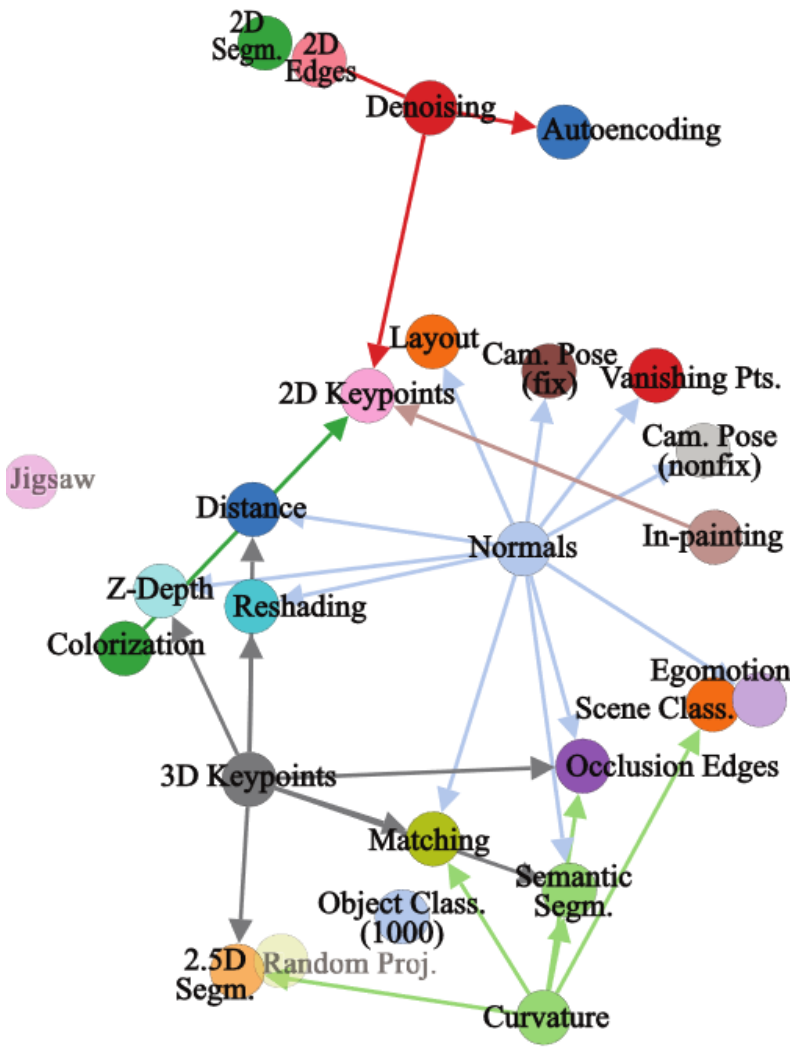
3. Transferability obtained by Analytic Hierarchy Process (from pairwise comparisons between all possible sources for each target task)
4. Final graph obtained by subgraph selection optimization (best performance from a limited set of source tasks): *transfer policy*

Empirical study on performance and data-efficiency gains from transfer using different datasets (Places and Imagenet)



Slide Credit: Camilo & Higuera

Taskonomy



Slide Credit: Camilo & Higuera

But wait, there's more!

- Transfer Learning
- Domain adaptation
- Semi-supervised learning
- Zero-shot learning
- **One/Few-shot learning**
- **Meta-Learning**
- Continual / Lifelong-learning
- Multi-modal learning
- Multi-task learning
- Active learning
- …

Reducing Label Requirements

- Alternative solution to gathering more data: exploit other sources of data that are imperfect but plentiful
  - unlabeled data (unsupervised learning)
  - Multi-modal data (multimodal learning)
  - Multi-domain data (transfer learning, domain adaptation)

Few-Shot Learning

Few-Shot Learning

People are good at it

Human-level concept learning through probabilistic program induction

Brenden M. Lake,[1*] Ruslan Salakhutdinov,[2] Joshua B. Tenenbaum[3]

Machines are getting better at it

Few-Shot Learning

- Let's attack directly the problem of **few-shot learning**
  - we want to design a learning algorithm A that outputs good parameters $\boldsymbol{\theta}$ of a model M, when fed a small dataset $D_{\text{train}}=\{(\mathbf{x}_i,y_i)\}\ i=1$

- Idea: let's *learn* that algorithm A, end-to-end
  - this is known as **meta-learning** or **learning to learn**

- Rather than features, in few-shot learning, we aim at transferring the complete training of the model on new datasets (not just transferring the features or initialization)
  - ideally there should be no human involved in producing a model for new datasets

Prior Methods

- One-shot learning has been studied before
  - One-Shot learning of object categories (2006) *Fei-Fei Li, Rob Fergus and Pietro Perona*
  - Knowledge transfer in learning to recognize visual objects classes (2004) *Fei-Fei Li*
  - Object classification from a single example utilizing class relevance pseudo-metrics (2004) *Michael Fink*
  - Cross-generalization: learning novel classes from a single example by feature replacement (2005) *Evgeniy Bart and Shimon Ullman*
- These largely relied on hand-engineered features
  - with recent progress in end-to-end deep learning, we hope to learn a representation better suited for few-shot learning

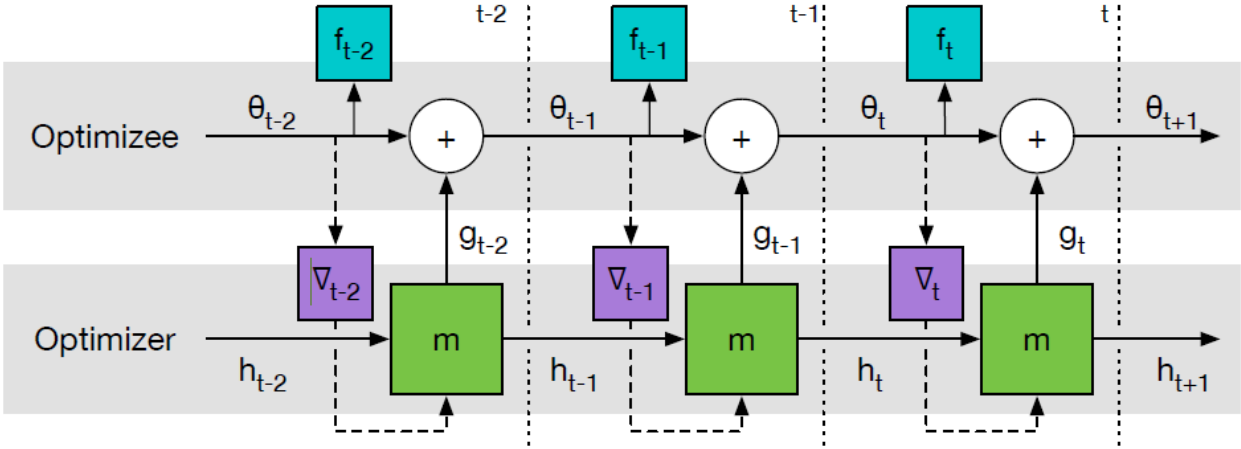
Prior Meta-Learning Methods

- Early work on learning an update rule
  - Learning a synaptic learning rule (1990) *Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier*
  - The Evolution of Learning: An Experiment in Genetic Connectionism (1990) *David Chalmers*
  - On the search for new learning rules for ANNs (1995) *Samy Bengio, Yoshua Bengio, and Jocelyn Cloutier*

- Early work on recurrent networks modifying their weights
  - Learning to control fast-weight memories: An alternative to dynamic recurrent networks (1992) *Jürgen Schmidhuber*
  - A neural network that embeds its own meta-levels (1993) *Jürgen Schmidhuber*

Related Work: Meta-Learning

- Training a recurrent neural network to optimize
  - outputs update, so can decide to do something else than gradient descent

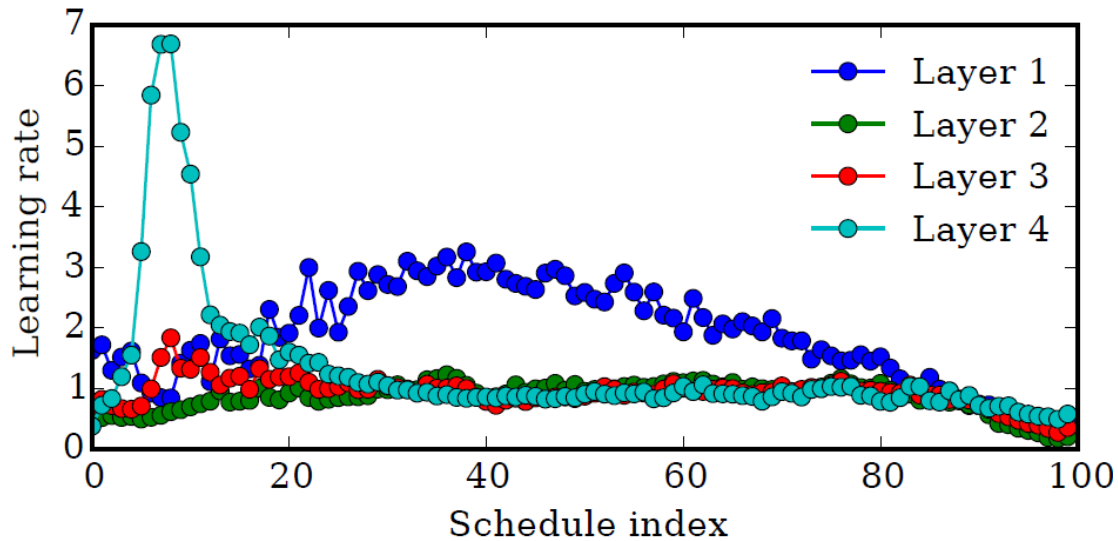


- Learning to learn by gradient descent by gradient descent (2016) *Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas*
- Learning to learn using gradient descent (2001) *Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell*

Slide Credit: Hugo Larochelle

Related Work: Meta-Learning

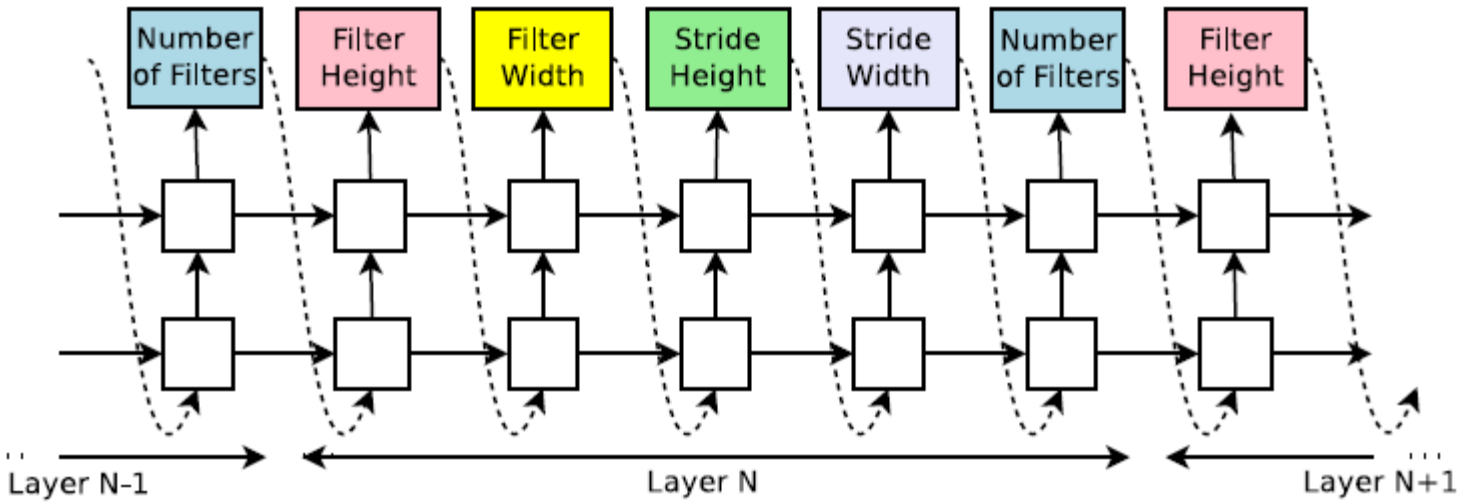
- Hyper-parameter optimization
  - idea of learning the learning rates and the initialization conditions



- Gradient-based hyperparameter optimization through reversible learning (2015) *Dougal Maclourin, David Duvenaud, and Ryan P. Adams*

Related Work: Meta-Learning

- AutoML (Bayesian optimization, reinforcement learning)



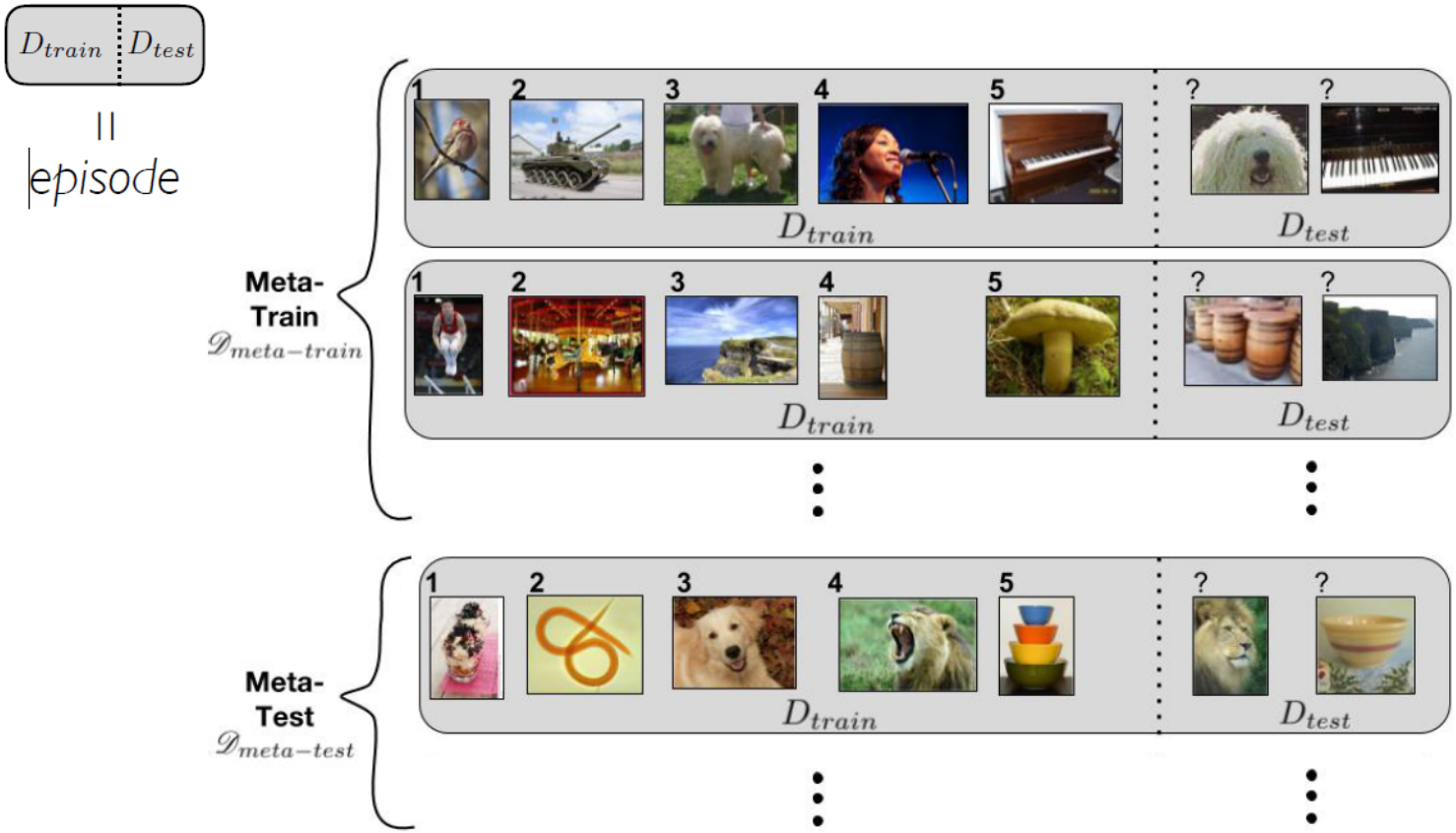
- Neural Architecture Search with Reinforcement Learning (2017) *Barret Zoph and Quoc Le*

Meta-Learning

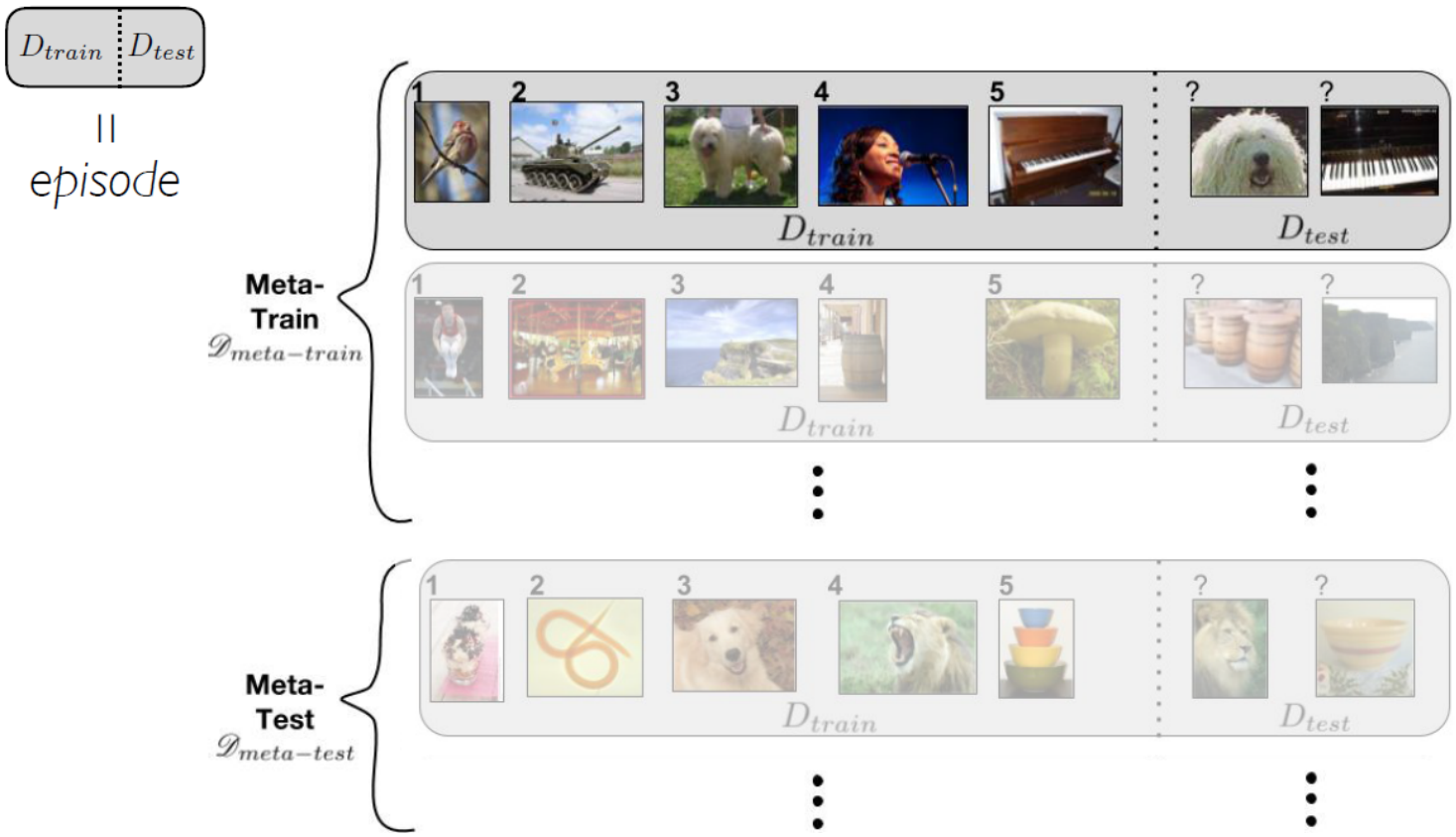
- Learning algorithm *A*
  - *input*: training set $D_{train}=\{(\mathbf{x}_i,\mathbf{y}_i)\}$
  - *output*: parameters $\boldsymbol{\theta}$ model *M* (the **learner**)
  - *objective*: good performance on test set $D_{test}=\{(\mathbf{x}'_i,\mathbf{y}'_i)\}$

- Meta-learning algorithm
  - *input*: meta-training set $\mathscr{D}_{meta-train}=\{(D^{(n)}_{train},D^{(n)}_{test})\}_{n=1}^{N}$ of *episodes*
  - *output*: parameters $\boldsymbol{\Theta}$ algorithm *A* (the **meta-learner**)
  - *objective*: good performance on meta-test set $\mathscr{D}_{meta-test}=\{(D'^{(n)}_{train},D'^{(n)}_{test})\}_{n=1}^{N'}$

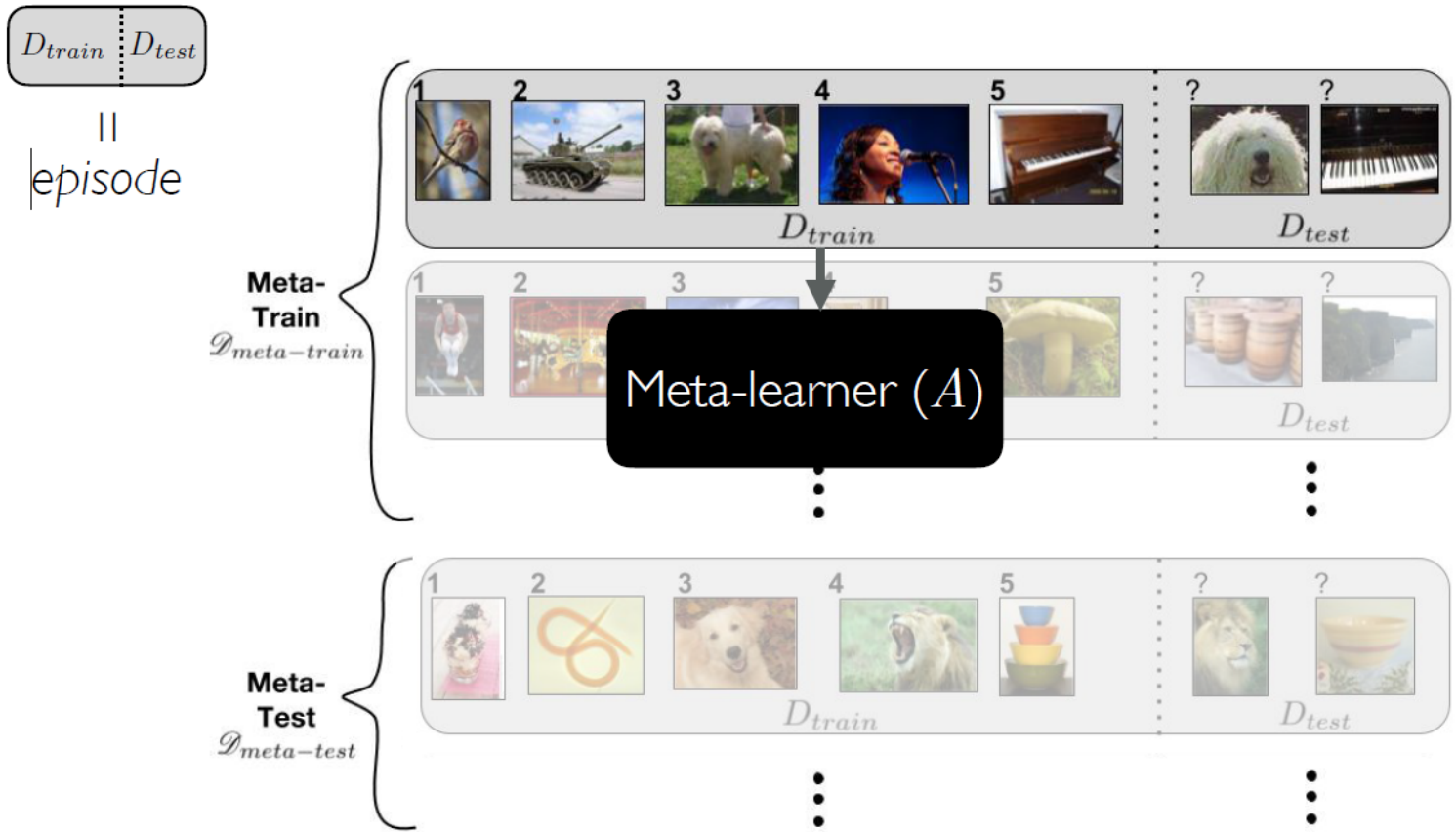
Meta-Learning



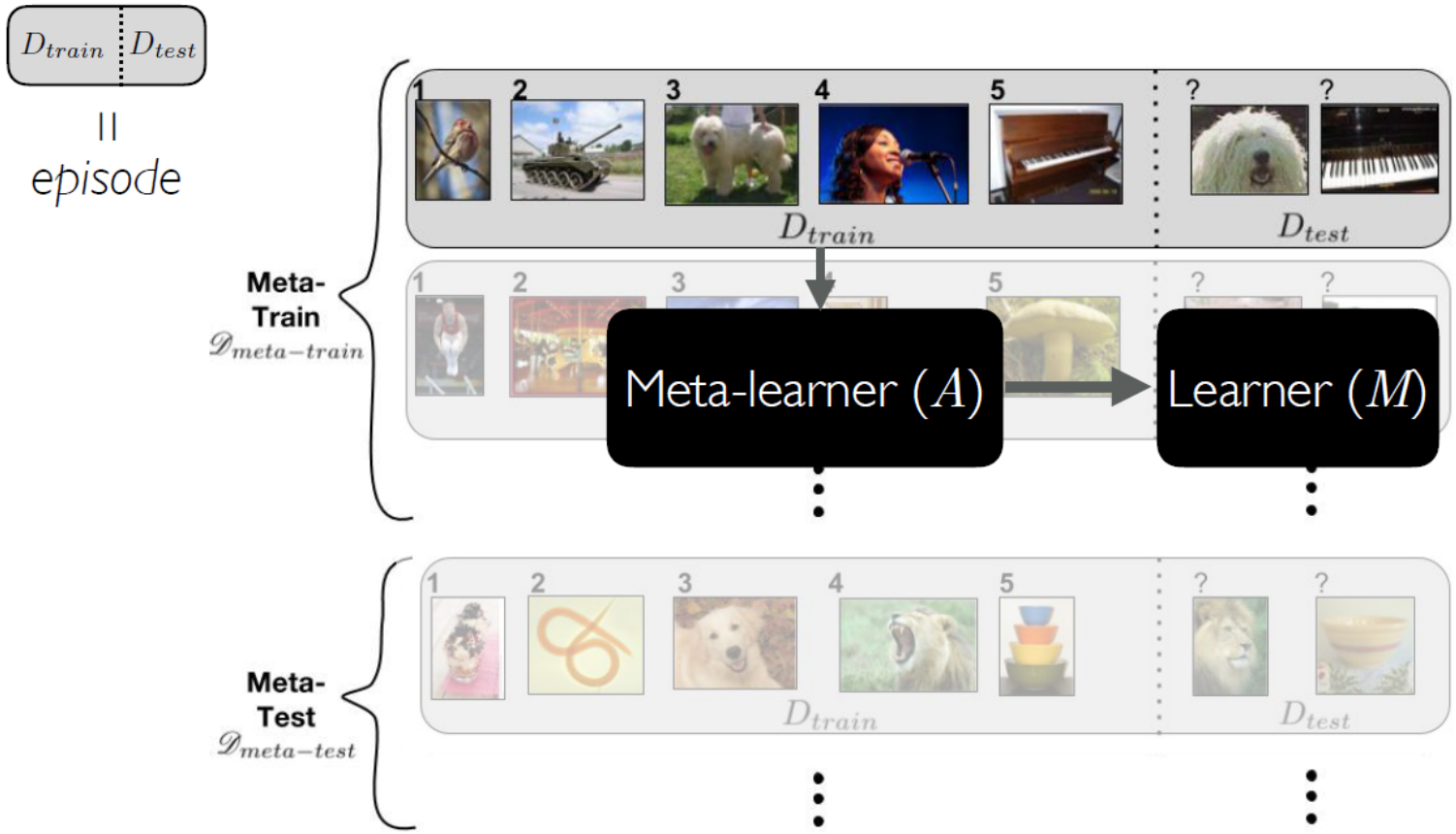
Meta-Learning



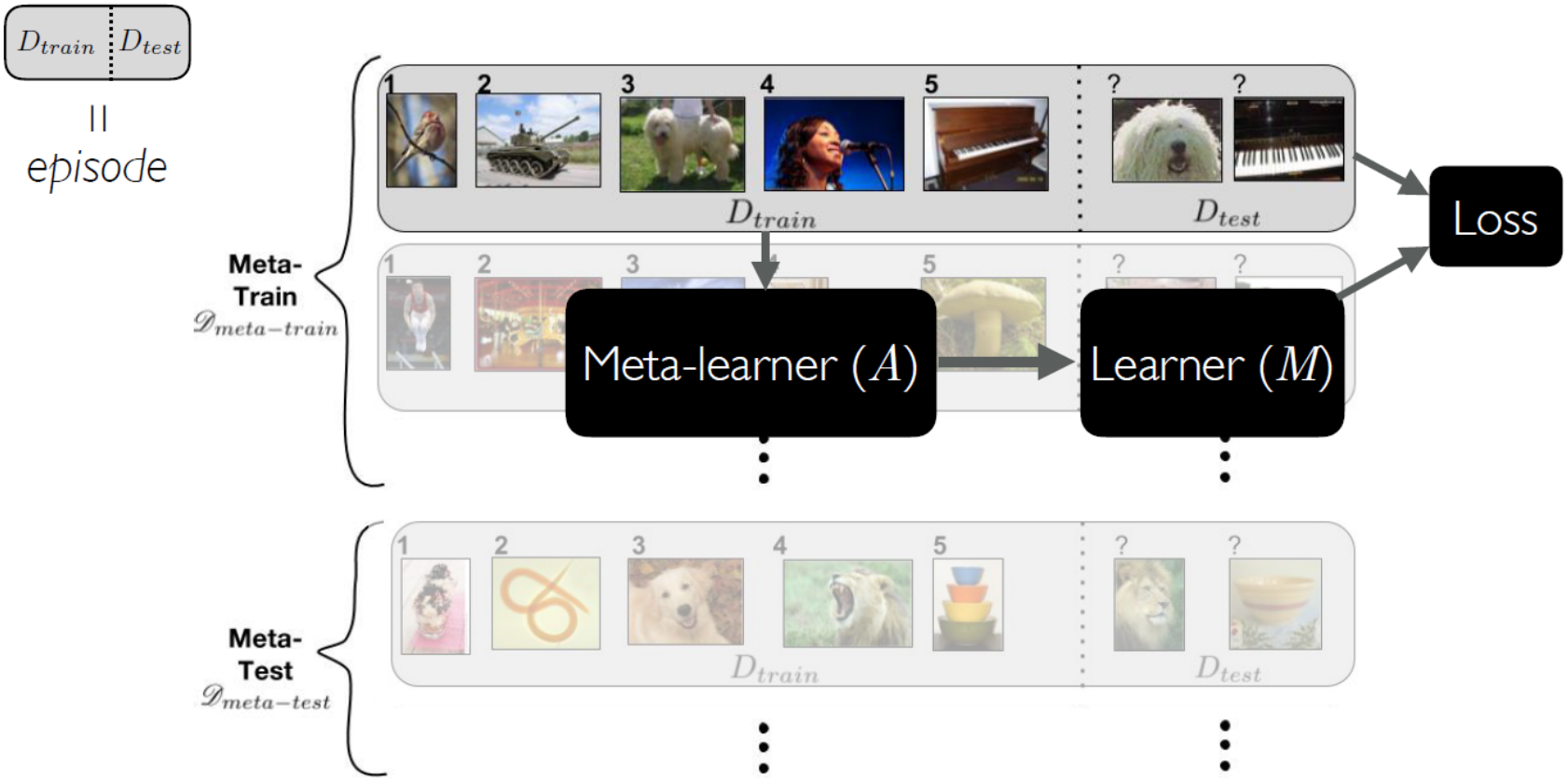
Meta-Learning



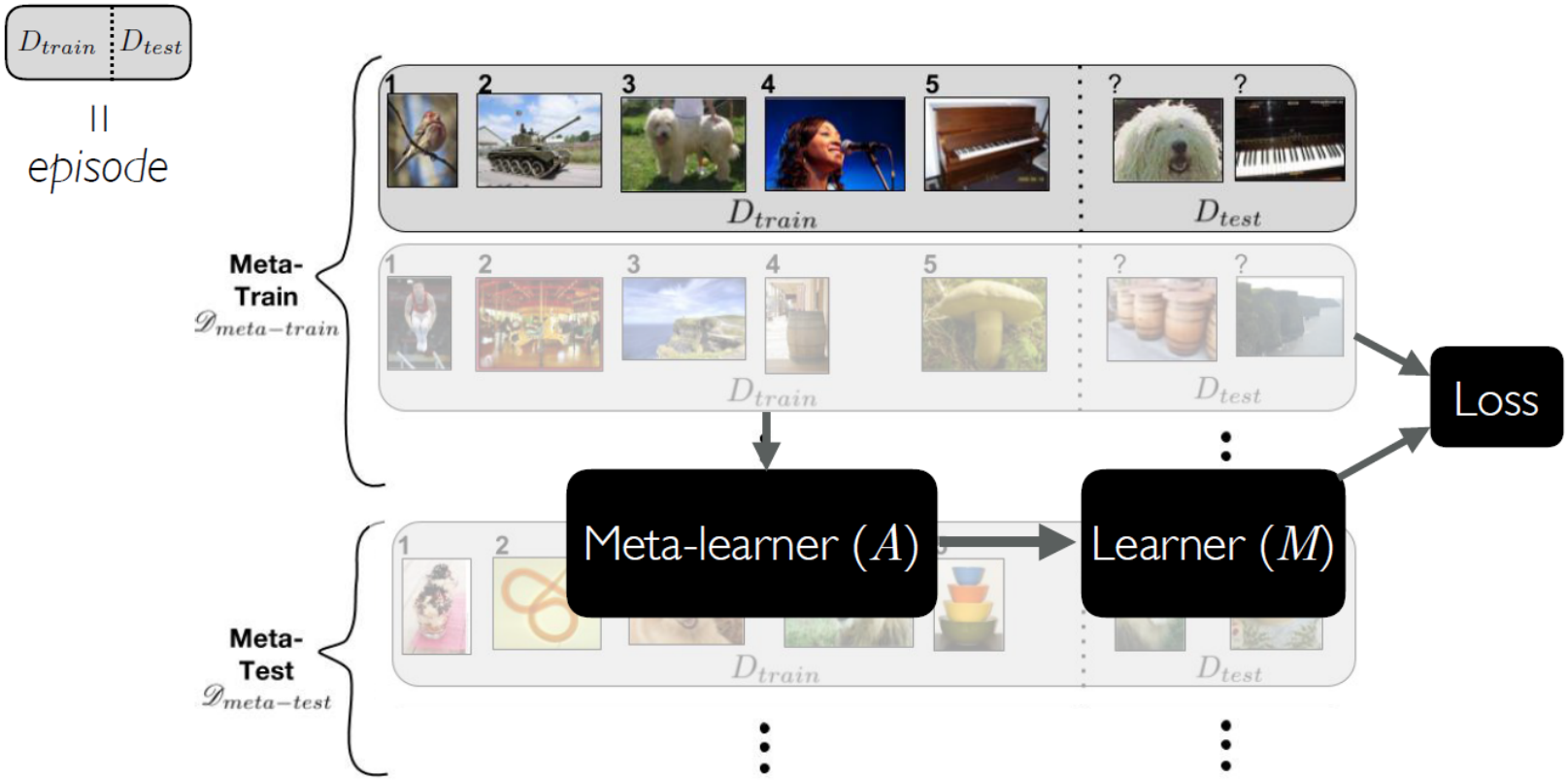
Meta-Learning



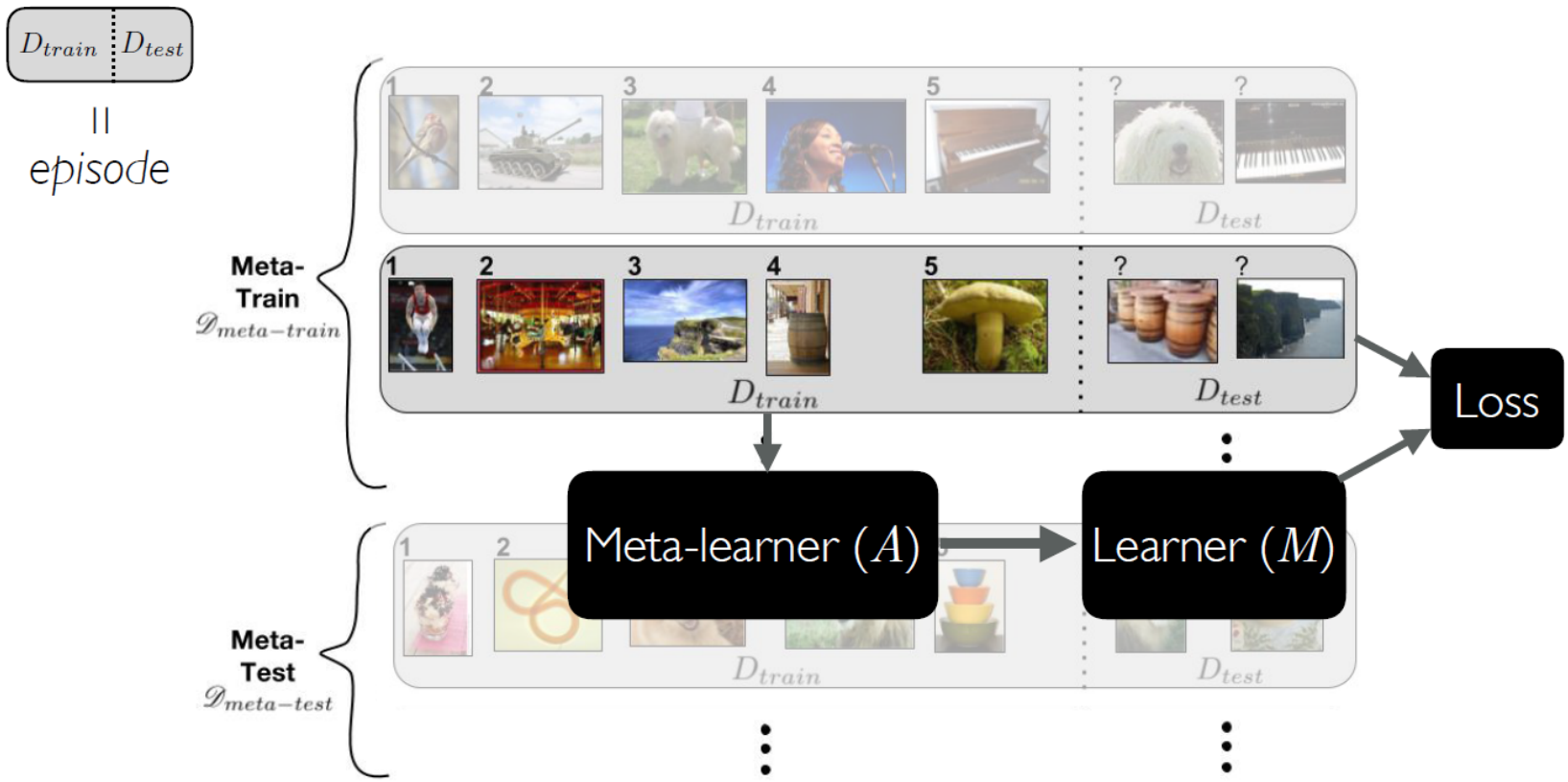
Meta-Learning



Meta-Learning



Meta-Learning



Meta-Learning Nomenclature

Episode

Training set ⟷ Support set

Test set ⟷ Query set

Meta-training set ⟷ Training set

Meta-test set ⟷ Test set

Meta-Learning Nomenclature

- Assuming a probabilistic model *M* over labels, the cost per episode can become

$$C(\Theta; D_{train}, D_{test}) = \frac{1}{|D_{test}|} \sum_{\substack{(\mathbf{x}'_i, y'_i) \\ \in D_{test}}} -\log p_\Theta(y'_i | \mathbf{x}'_i, D_{train})$$

- Depending on the choice of meta-learner, $p_\Theta(y|\mathbf{x}, D_{train})$ will take a different form

Meta-Learner

- How to parametrize learning algorithms?

- Two approaches to defining a meta-learner
  - Take inspiration from a known learning algorithm
    - kNN/kernel machine: Matching networks (Vinyals et al. 2016)
    - Gaussian classifier: Prototypical Networks (Snell et al. 2017)
    - Gradient Descent: Meta-Learner LSTM (Ravi & Larochelle, 2017) , MAML (Finn et al. 2017)
  - Derive it from a black box neural network
    - MANN (Santoro et al. 2016)
    - SNAIL (Mishra et al. 2018)

Meta-Learner

- How to parametrize learning algorithms?

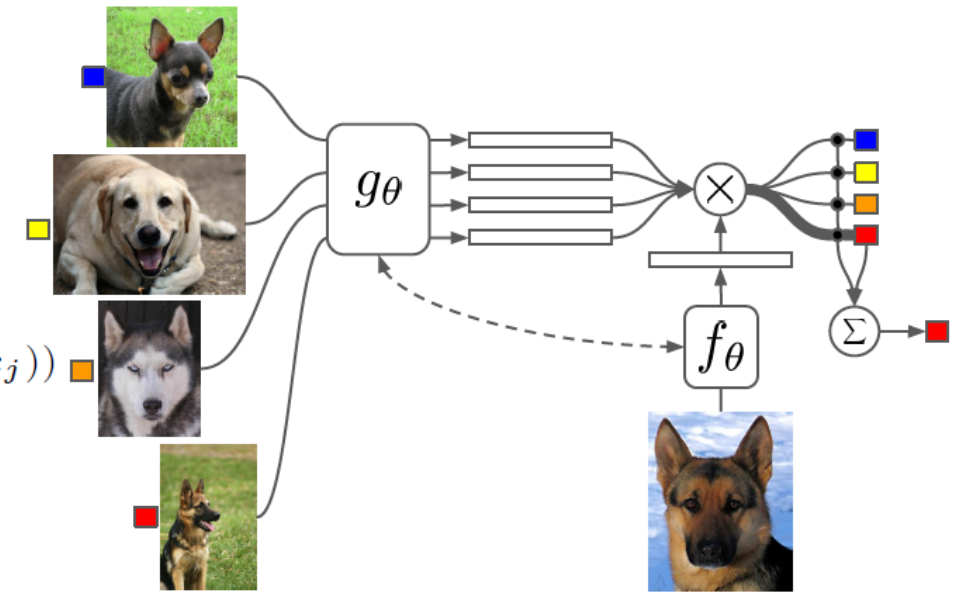
- Two approaches to defining a meta-learner
  - **Take inspiration from a known learning algorithm**
    - kNN/kernel machine: Matching networks (Vinyals et al. 2016)
    - Gaussian classifier: Prototypical Networks (Snell et al. 2017)
    - Gradient Descent: Meta-Learner LSTM (Ravi & Larochelle, 2017) , MAML (Finn et al. 2017)
  - Derive it from a black box neural network
    - MANN (Santoro et al. 2016)
    - SNAIL (Mishra et al. 2018)

Matching Networks

- Training a "**pattern matcher**"

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

$$a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}$$



- Matching networks for one shot learning (2016)
  Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra

Slide Credit: Hugo Larochelle

Prototypical Networks

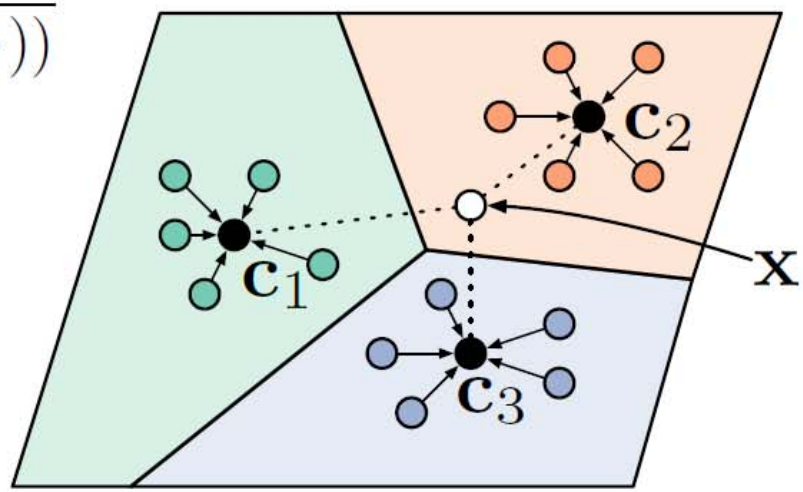
- Training a "**prototype extractor**"

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

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

$$S_k = \{(\mathbf{x}_i, y_i) | y_i = k, (\mathbf{x}_i, y_i) \in D_{train}\}$$

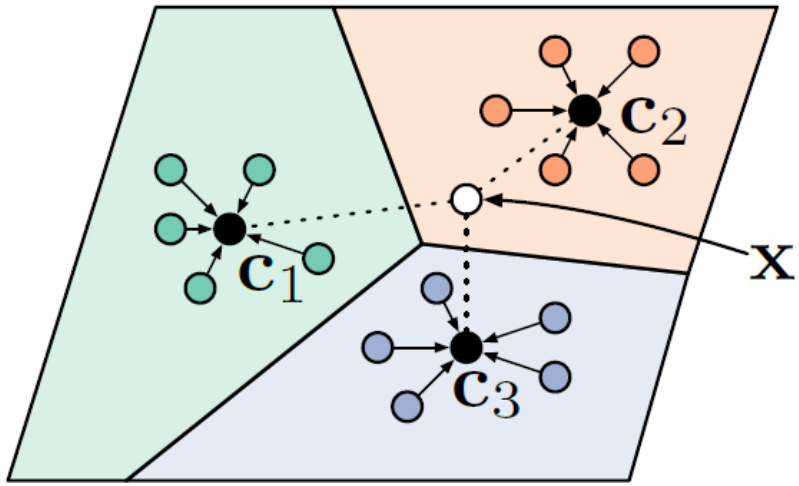
$$\phi \equiv \Theta$$



- Prototypical Networks for Few-shot Learning (2017)
  Jake Snell, Kevin Swersky and Richard Zemel

Prototypical Networks

- Training a "**prototype extractor**"
  - distance function $d(\cdot, \cdot)$ can be anything (euclidean squared, negative cosine)
  - if distance is euclidean squared, equivalent to learning an embedding network $f_\phi(\cdot)$ such that a Gaussian classifier works well
  - prototype vectors are equivalent to output weights of a neural network
  - Snell et al. find that using more classes in the meta-training episodes compared to meta-testing works better



- Prototypical Networks for Few-shot Learning (2017)
  Jake Snell, Kevin Swersky and Richard Zemel

Meta-Learner LSTM

- Training a "**gradient descent procedure**" applied on some learner M
  - gradient descent starts from some initial parameters θ_0 and then performs the following updates:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Meta-Learner LSTM

- Training a "**gradient descent procedure**" applied on some learner M
  - gradient descent starts from some initial parameters θ_0 and then performs the following updates:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$

  - this is quite similar to LSTM cell state updates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

    - state c_t is model M's parameter space θ_t
    - state update $\tilde{c}_t$ is the negative gradient $-\nabla_{\theta_{t-1}} \mathcal{L}_t$
    - f_t and i_t are LSTM gates:

$$i_t = \sigma\left(\mathbf{W}_I \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}\right] + \mathbf{b}_I\right)$$

$$f_t = \sigma\left(\mathbf{W}_F \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}\right] + \mathbf{b}_F\right)$$

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Meta-Learner LSTM

- Training a "**gradient descent procedure**" applied on some learner M
  - gradient descent starts from some initial parameters θ_0 and then performs the following updates:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$

  - this is quite similar to LSTM cell state updates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

    - state c_t is model M's parameter space θ_t ← c_0 becomes a *learned initialization*
    - state update $\tilde{c}_t$ is the negative gradient $-\nabla_{\theta_{t-1}} \mathcal{L}_t$
    - f_t and i_t are LSTM gates:

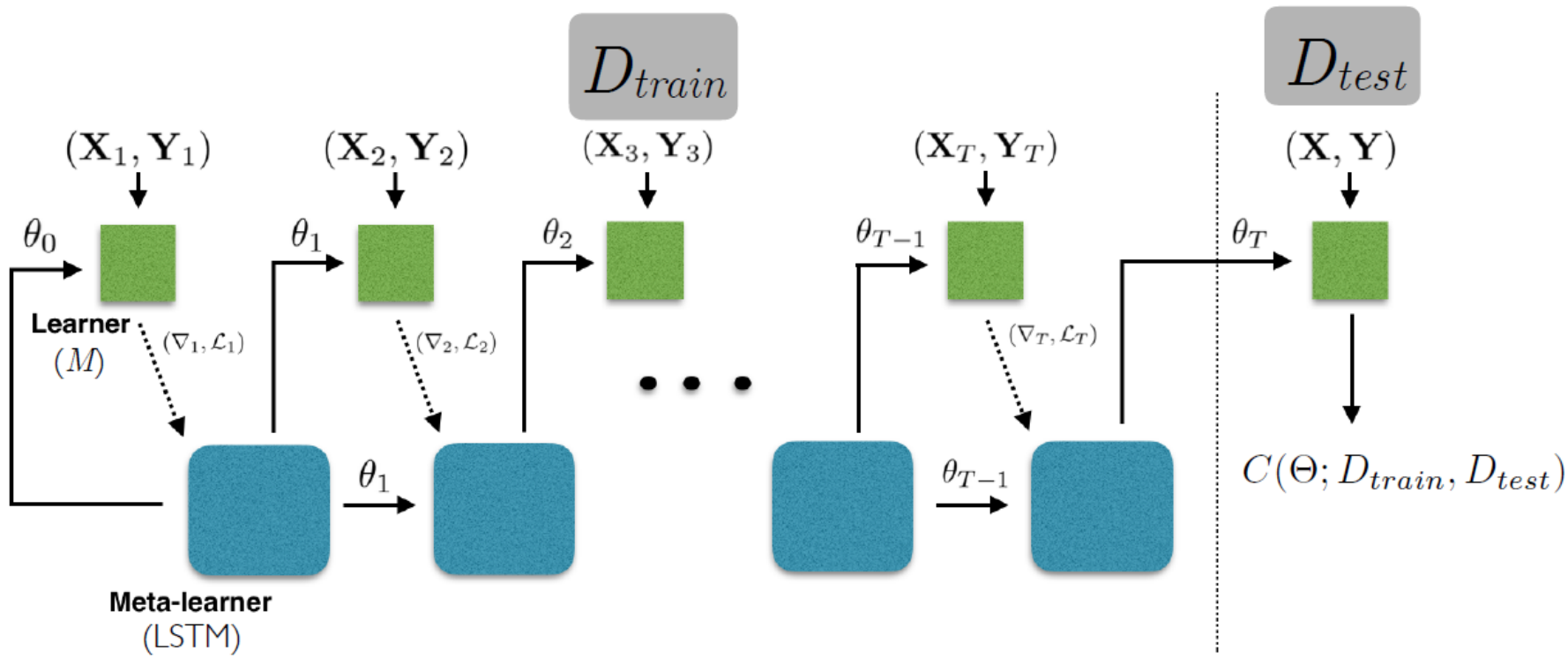
$$i_t = \sigma\left(\mathbf{W}_I \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}\right] + \mathbf{b}_I\right)$$ ← *adaptive learning rate*

$$f_t = \sigma\left(\mathbf{W}_F \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}\right] + \mathbf{b}_F\right)$$ ← *adaptive weight decay*

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Meta-Learner LSTM

- Training a "**gradient descent procedure**" applied on some learner M



- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Meta-Learning Algorithm

Algorithm 1 Train Meta-Learner

Input: Meta-training set $\mathscr{D}_{meta-train}$, Learner M with parameters θ, Meta-Learner R with parameters Θ.

$\Theta_0 \leftarrow$ random initialization

for $d = 1, n$ **do**

  $D_{train}, D_{test} \leftarrow$ random dataset from $\mathscr{D}_{meta-train}$

  $\theta_0 \leftarrow c_0$ ▷ Intialize learner parameters

  for $t = 1, T$ **do**

    $\mathbf{X}_t, \mathbf{Y}_t \leftarrow$ random batch from D_{train}

    $\mathcal{L}_t \leftarrow \mathcal{L}(M(\mathbf{X}_t; \theta_{t-1}), \mathbf{Y}_t)$ ▷ Get loss of learner on train batch

    $c_t \leftarrow R((\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t); \Theta_{d-1})$ ▷ Get output of meta-learner using Equation 2

    $\theta_t \leftarrow c_t$ ▷ Update learner parameters

  end for

  $\mathbf{X}, \mathbf{Y} \leftarrow D_{test}$

  $\mathcal{L}_{test} \leftarrow \mathcal{L}(M(\mathbf{X}; \theta_T), \mathbf{Y})$ ▷ Get loss of learner on test batch

  Update Θ_d using $\nabla_{\Theta_{d-1}} \mathcal{L}_{test}$ ▷ Update meta-learner parameters

end for

Meta-Learner LSTM

- Training a "**gradient descent procedure**" applied on some learner M
  - LSTM parameters are shared across M's parameters (i.e. treated like a large minibatch)
  - can ignore (stop) gradients through the inputs of the LSTM
  - gradient (and loss) inputs to the Meta-LSTM preprocessed as proposed by Andrychowicz et al. (2016)

$$\nabla^k \rightarrow \begin{cases} \left(\frac{\log(|\nabla|)}{p}, \mathrm{sgn}(\nabla)\right) & \text{if } |\nabla| \geq e^{-p} \\ (-1, e^p \nabla) & \text{otherwise} \end{cases}$$

  - we are careful to avoid "leakage" from batchnorm statistics between meta-train / meta-test sets (sometimes referred to as the "transductive setting")

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

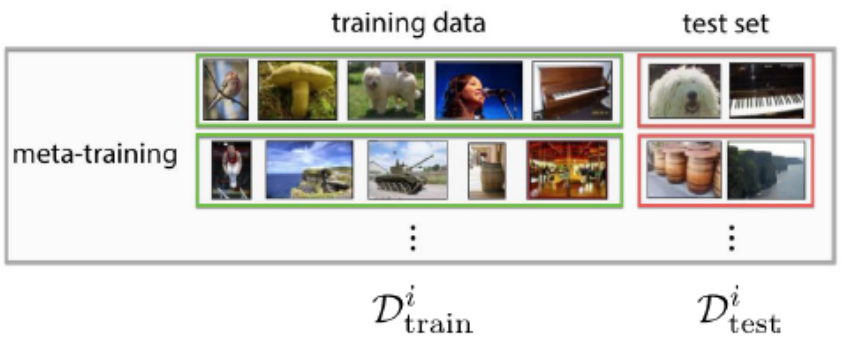
Model-Agnostic Meta-Learning (MAML)

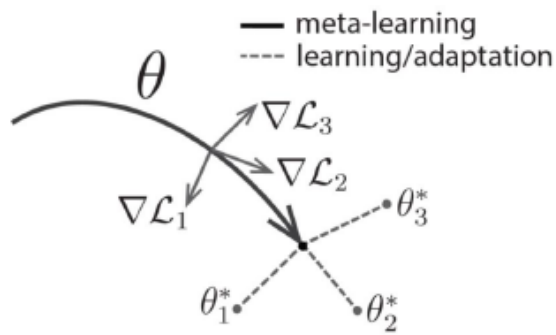
- Training a "**gradient descent procedure**" applied on some learner M
  - MAML proposes not to bother with training an LSTM for the gradient descent updates and constant step-size updates
  - better results are also reported by the so-called *bias transformation* architecture (*One-Shot Visual Imitation Learning via Meta-Learning*, Finn et al. 2017)
    - concatenates to one of the layers a trainable parameter vector, for instance to the input layer $[\mathbf{x}_i, \theta_b]$
    - decouples the updates of the bias and weights of that layer
    - with it, can be shown that even a single gradient descent update yields a universal approximator over functions mapping D_{train} and $\mathbf{x}$ to any label y, for a sufficiently deep ReLU network and certain losses (*Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm*, Finn and Levine, 2018)

- Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks (2017)
  Chelsea Finn, Pieter Abbeel and Sergey Levine

Model-Agnostic Meta-Learning (MAML)

a general recipe:





$$\theta \leftarrow \theta - \beta \sum_i \nabla_\theta \underbrace{\mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^i_{\text{train}}), \mathcal{D}^i_{\text{test}})}_{\text{“meta-loss” for task } i}$$

* in general, can take more than one gradient step here

** we often use 4 – 10 steps

Chelsea Finn

Finn et al., “Model-Agnostic Meta-Learning”

Model-Agnostic Meta-Learning (MAML)

supervised learning: $f(x) \rightarrow y$

supervised meta-learning: $f(\mathcal{D}_{\text{train}}, x) \rightarrow y$

model-agnostic meta-learning: $f_{\text{MAML}}(\mathcal{D}_{\text{train}}, x) \rightarrow y$

$$f_{\text{MAML}}(\mathcal{D}_{\text{train}}, x) = f_{\theta'}(x)$$

$$\theta' = \theta - \alpha \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \nabla_\theta \mathcal{L}(f_\theta(x), y)$$

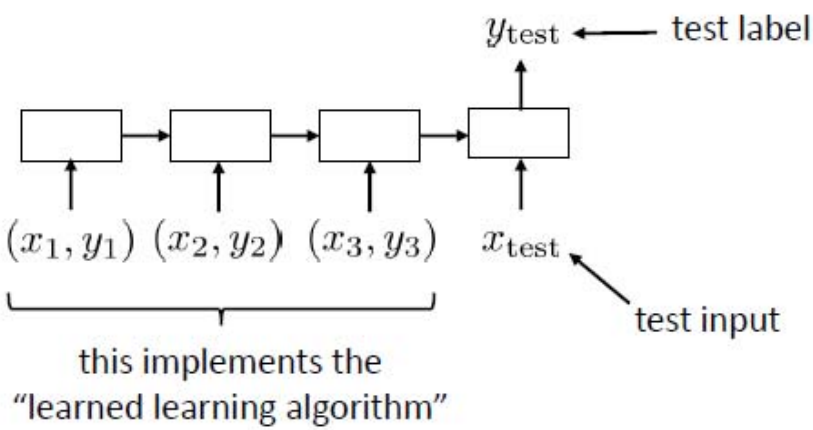
Just another computation graph…

Can implement with any autodiff package (e.g., TensorFlow)

Slide Credit: Sergey Levine

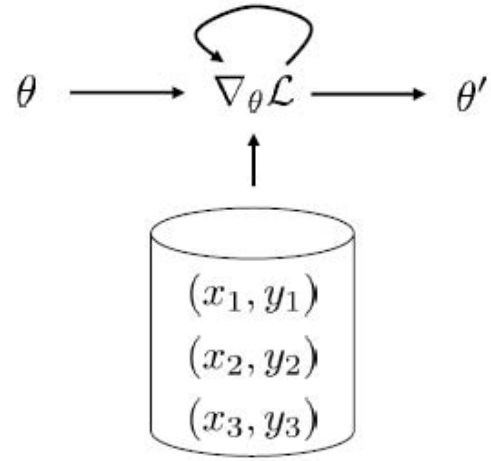
Comparison

RNN-based meta-learning



- Does it converge?
  - Kind of?
- What does it converge to?
  - Who knows…
- What to do if it's not good enough?
  - Nothing…

MAML



- Does it converge?
  - Yes (it's gradient descent…)
- What does it converge to?
  - A local optimum (it's gradient descent…)
- What to do if it's not good enough?
  - Keep taking gradient steps (it's gradient descent…)

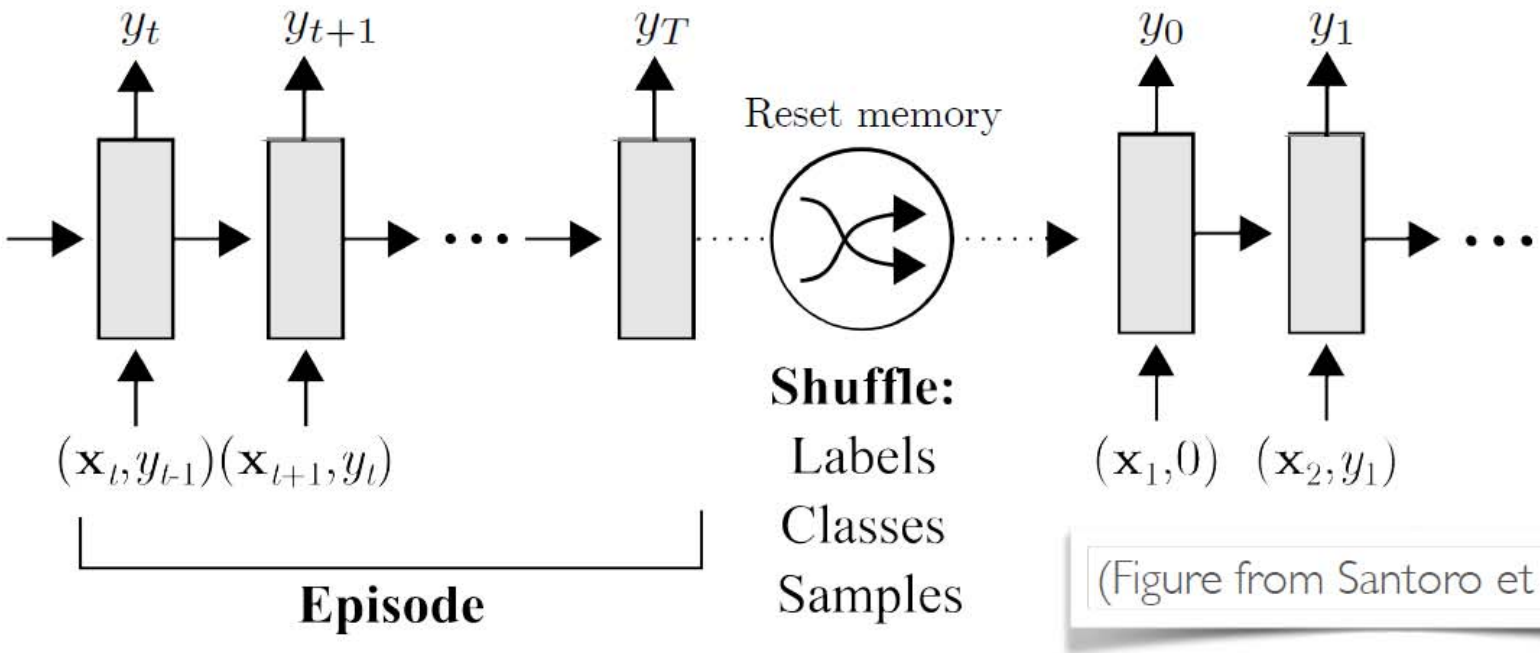
Meta-Learner

- How to parametrize learning algorithms?

- Two approaches to defining a meta-learner
  - Take inspiration from a known learning algorithm
    - kNN/kernel machine: Matching networks (Vinyals et al. 2016)
    - Gaussian classifier: Prototypical Networks (Snell et al. 2017)
    - Gradient Descent: Meta-Learner LSTM (Ravi & Larochelle, 2017) , MAML (Finn et al. 2017)
  - **Derive it from a black box neural network**
    - MANN (Santoro et al. 2016)
    - SNAIL (Mishra et al. 2018)

Black-Box Meta-Learner

- Frame meta-learning as sequence labeling with correct labels as delayed inputs

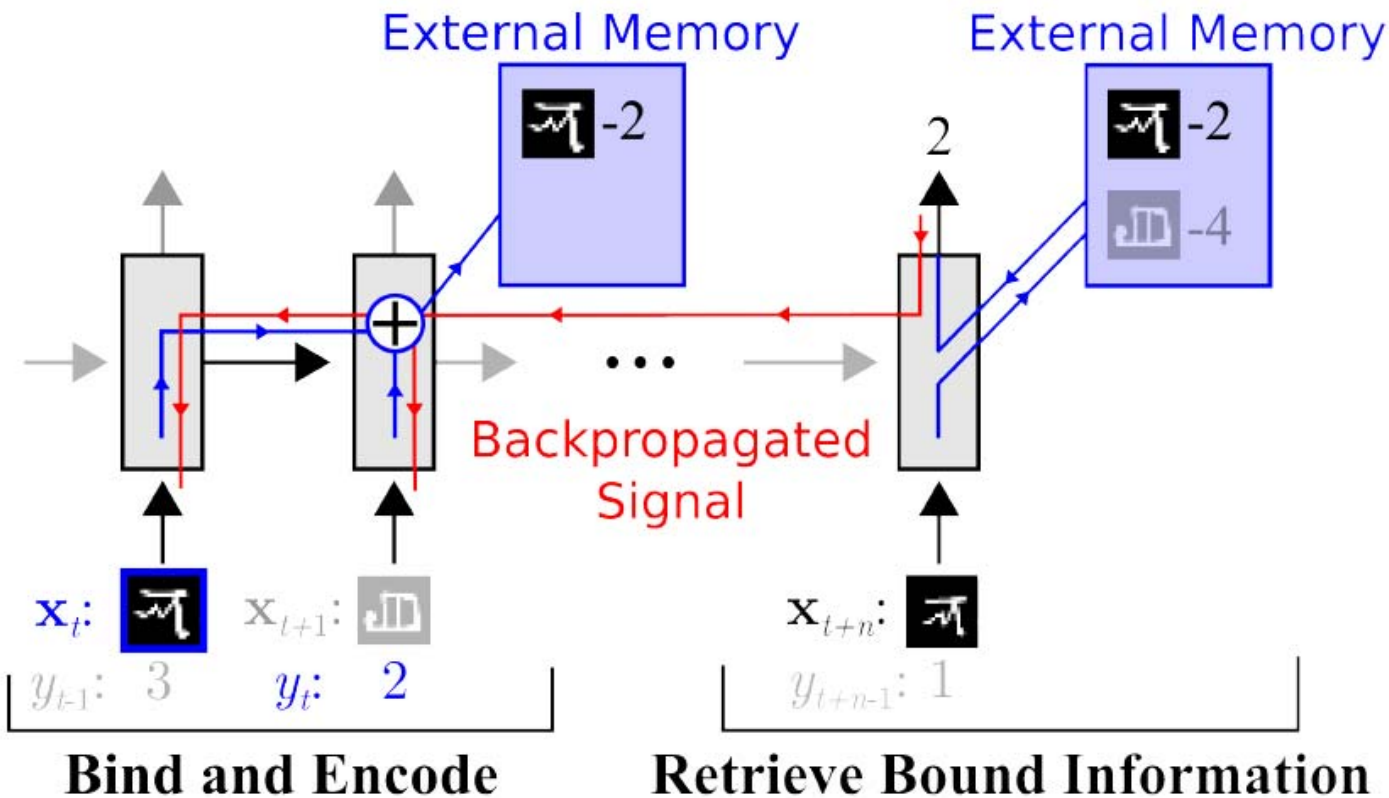


(Figure from Santoro et al. 2016)

- Learning to learn using gradient descent (2001)
  Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell

Memory-Augmented Neural Network

- Training a **neural Turing machine** to learn a learning algorithm



- One-shot learning with memory-augmented neural networks (2016)
  Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap

Experiments

- Mini-ImageNet (split used in Ravi & Larochelle, 2017)
  - random subset of 100 classes (64 training, 16 validation, 20 testing)
  - random sets D_{train} are generated by randomly picking 5 classes from class subset

Model	5-class	
	1-shot	5-shot
Baseline-finetune	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
Baseline-nearest-neighbor	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
Matching Network	$\mathbf{43.40 \pm 0.78\%}$	$51.09 \pm 0.71\%$
Matching Network FCE	$\mathbf{43.56\% \pm 0.84\%}$	$55.31\% \pm 0.73\%$
Meta-Learner LSTM (OURS)	$\mathbf{43.44\% \pm 0.77\%}$	$\mathbf{60.60\% \pm 0.71\%}$

Experiments

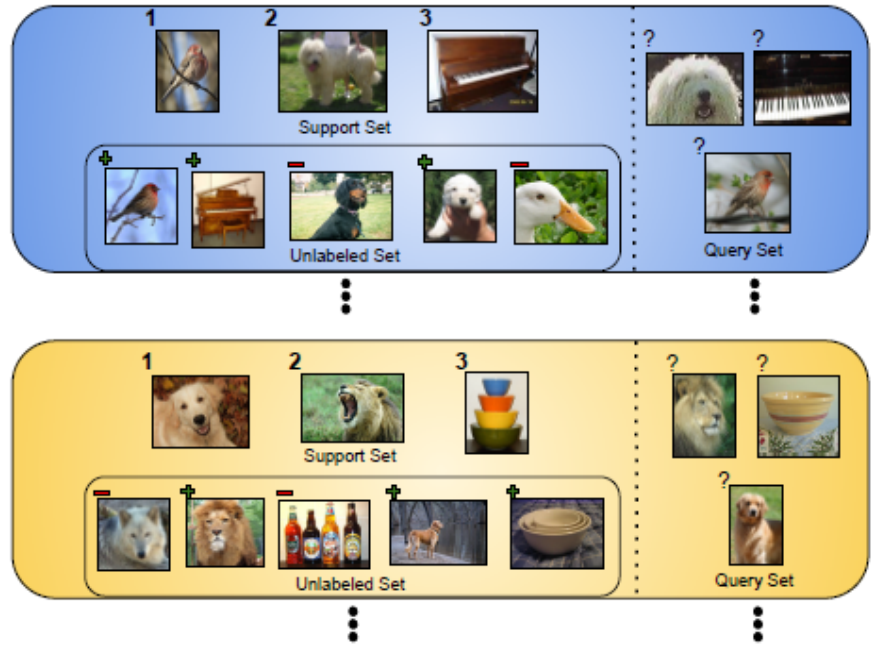
- Mini-ImageNet (split used in Ravi & Larochelle, 2017)
  - random subset of 100 classes (64 training, 16 validation, 20 testing)
  - random sets D_{train} are generated by randomly picking 5 classes from class subset

Model	5-class	
	1-shot	5-shot
Prototypical Nets (Snell et al.)	49.42% ± 0.78%	**68.20% ± 0.66%**
MAML (Finn et al.)	48.70% ± 1.84%	63.10% ± 0.92%
SNAIL (Mishra et al.)	**55.71% ± 0.99%**	**68.88% ± 0.98%**
Matching Network FCE	43.56% ± 0.84%	55.31% ± 0.73%
Meta-Learner LSTM (OURS)	43.44% ± 0.77%	60.60% ± 0.71%

Extensions and Variations

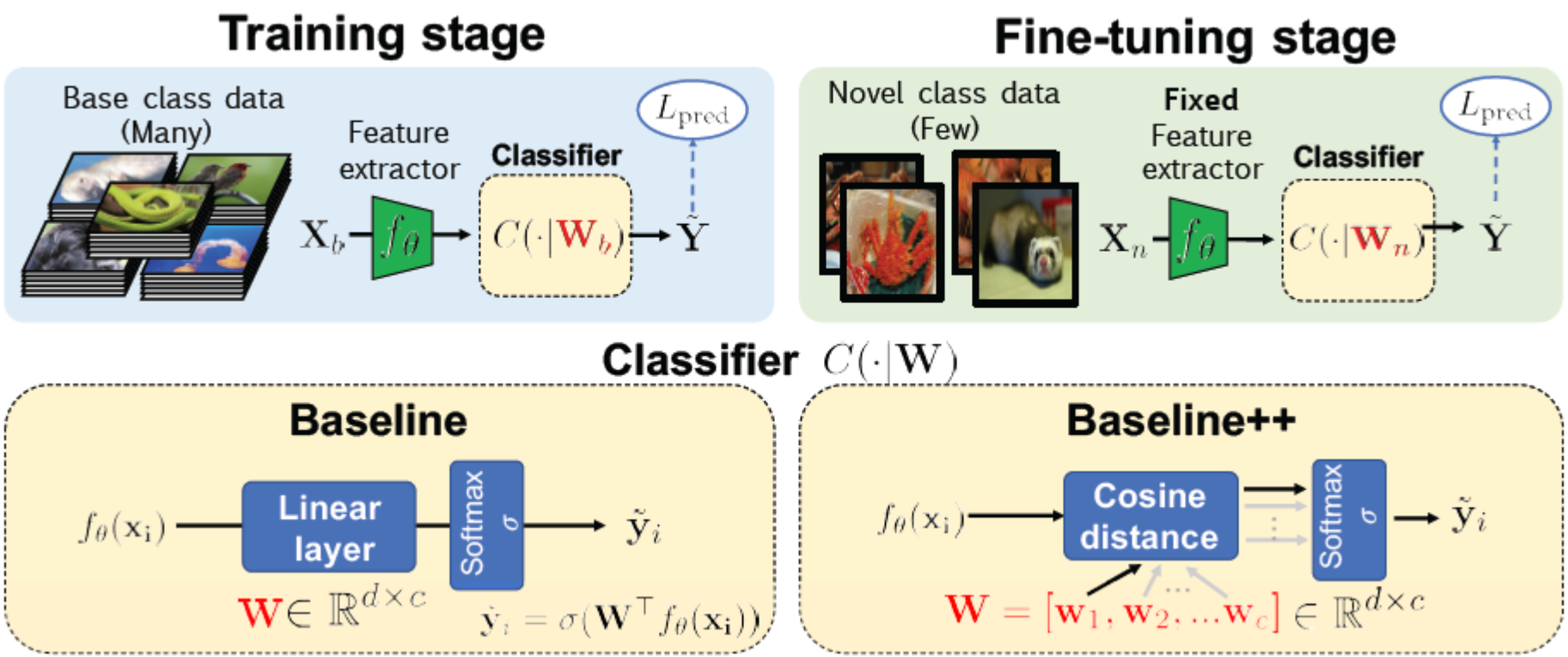
- Semi-supervised learning (with distractors)
  - assign soft-labels to unlabeled examples
  - use soft-labels to refine prototypes

$$\tilde{\boldsymbol{p}}_c = \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c} + \sum_j h(\tilde{\boldsymbol{x}}_j) \tilde{z}_{j,c} m_{j,c}}{\sum_i z_{i,c} + \sum_j \tilde{z}_{j,c} m_{j,c}}$$



- Meta-Learning for Semi-supervised Few-Shot Classification (2018)
  Ren, Triantafillou, Ravi, Snell, Swersky, Tenenbaum, Larochelle and Zemel

But beware



A Closer Look at Few-shot Classification,

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, Jia-Bin Huang

Table 2: **Few-shot classification results for both the *mini*-ImageNet and *CUB* datasets.** The Baseline++ consistently improves the Baseline model by a large margin and is competitive with the state-of-the-art meta-learning methods. All experiments are from 5-way classification with a Conv-4 backbone and data augmentation.

	CUB		***mini*-ImageNet**	
Method	**1-shot**	**5-shot**	**1-shot**	**5-shot**
Baseline	47.12 ± 0.74	64.16 ± 0.71	42.11 ± 0.71	62.53 ± 0.69
Baseline++	60.53 ± 0.83	79.34 ± 0.61	48.24 ± 0.75	66.43 ± 0.63
MatchingNet Vinyals et al. (2016)	61.16 ± 0.89	72.86 ± 0.70	48.14 ± 0.78	63.48 ± 0.66
ProtoNet Snell et al. (2017)	51.31 ± 0.91	70.77 ± 0.69	44.42 ± 0.84	64.24 ± 0.72
MAML Finn et al. (2017)	55.92 ± 0.95	72.09 ± 0.76	46.47 ± 0.82	62.71 ± 0.71
RelationNet Sung et al. (2018)	62.45 ± 0.98	76.11 ± 0.69	49.31 ± 0.85	66.60 ± 0.69

Distribution Shift

- What if there is a distribution shift (cross-domain)?

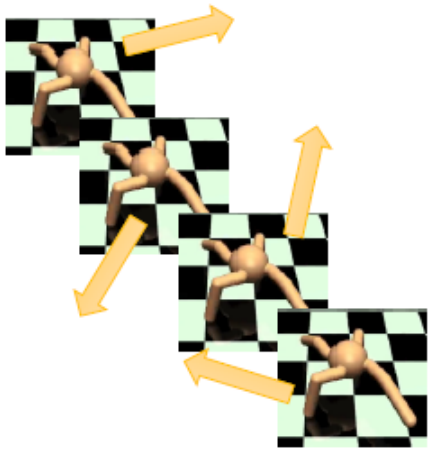
- **Lesson:** Methods that are successful *within-domain* might be worse *across domains*!

	mini-ImageNet →CUB
Baseline	**65.57±0.70**
Baseline++	62.04±0.76
MatchingNet	53.07±0.74
ProtoNet	62.02±0.70
MAML	51.34±0.72
RelationNet	57.71±0.73

Table 3: **5-shot accuracy under the cross-domain scenario with a ResNet-18 backbone.** Baseline outperforms all other methods under this scenario.

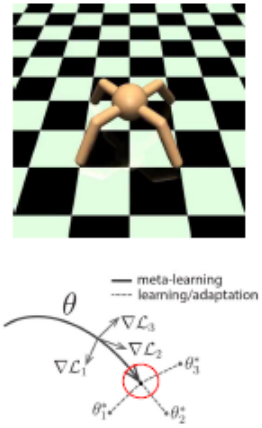
Distribution Shift

Let's Talk about Meta-Overfitting

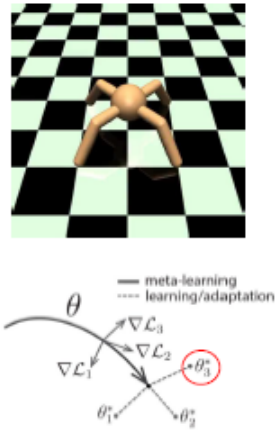


- Meta learning requires task distributions
- When there are too few meta-training tasks, we can *meta-overfit*
- Specifying task distributions is hard, especially for meta-RL!
- Can we propose tasks *automatically*?

after MAML training



after 1 gradient step

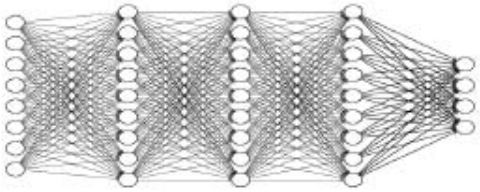


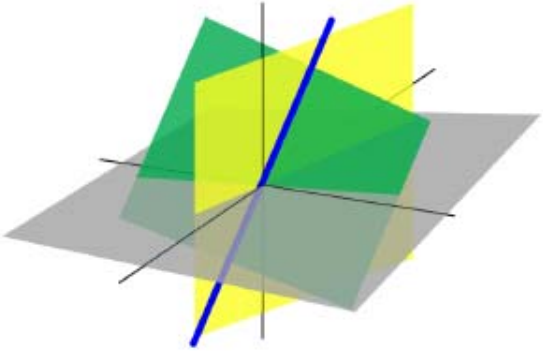
Random Task Proposals

- Use randomly initialize discriminators for reward functions

$$R(s, z) = \log p_D(z|s)$$

D → randomly initialized network

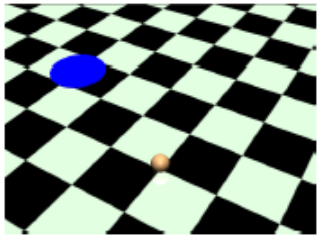


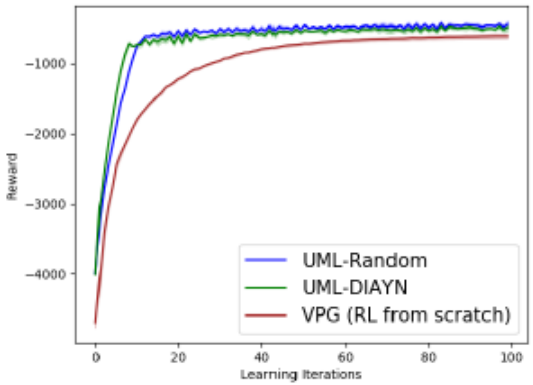


- Important: Random functions over state space, **not** random policies

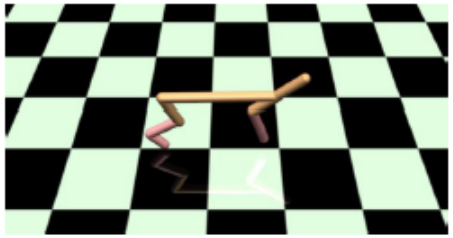
Does it Work?

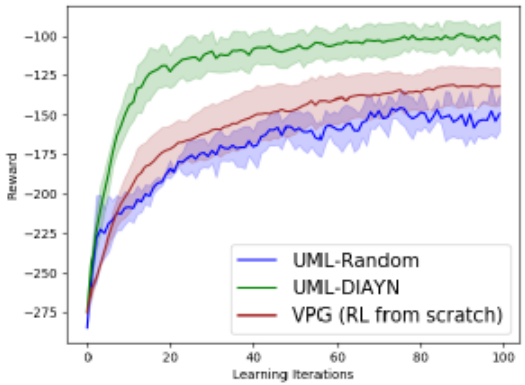
2D Navigation



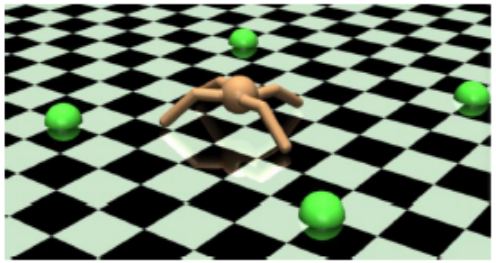


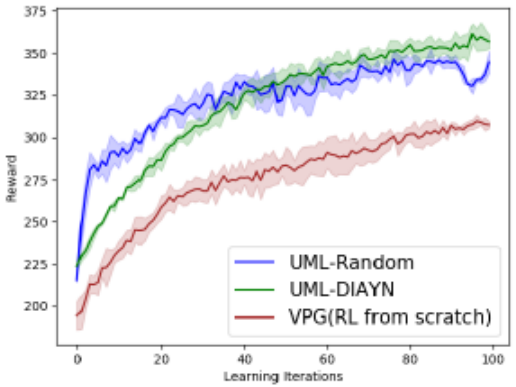
Cheetah





Ant





Meta-test performance with rewards

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

Discussions

- What is the right definition of distributions over problems?
  - varying number of classes / examples per class (meta-training vs. meta-testing) ?
  - semantic differences between meta-training vs. meta-testing classes ?
  - overlap in meta-training vs. meta-testing classes (see recent “low-shot” literature) ?
- Move from static to interactive learning
  - how should this impact how we generate episodes ?
  - meta-active learning ? (few successes so far)