

GENERALIZING FROM FEW EXAMPLES WITH META-LEARNING



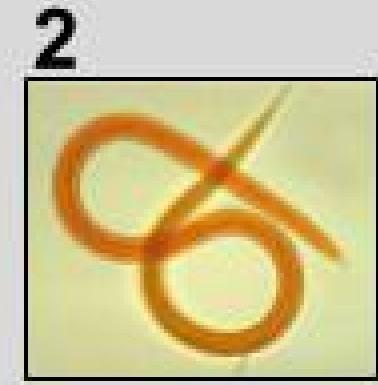
Hugo Larochelle
Google Brain

A RESEARCH AGENDA

- Deep learning successes have required a lot of labeled training data
 - ▶ collecting and labeling such data requires significant human labor
 - ▶ practically, is that really how we'll solve AI ?
 - ▶ scientifically, this means there is a gap with ability of humans to learn, which we should try to understand
- Alternative solution : exploit other sources of data that are imperfect but plentiful
 - ▶ unlabeled data (unsupervised learning)
 - ▶ multimodal data (multimodal learning)
 - ▶ multidomain data (transfer learning, domain adaptation)

A RESEARCH AGENDA

- Deep learning successes have required a lot of labeled training data
 - ▶ collecting and labeling such data requires significant human labor
 - ▶ practically, is that really how we'll solve AI ?
 - ▶ scientifically, this means there is a gap with ability of humans to learn, which we should try to understand
- Alternative solution : exploit other sources of data that are imperfect but plentiful
 - ▶ unlabeled data (unsupervised learning)
 - ▶ multimodal data (multimodal learning)
 - ▶ multidomain data (transfer learning, domain adaptation)



D_{train}



D_{test}

**People are
good at it**



Human-level concept learning through probabilistic program induction

Brenden M. Lake,^{1*} Ruslan Salakhutdinov,² Joshua B. Tenenbaum³



People are
good at it



Machines are
getting
better at it

പ	ി	ബ	ഡ	സ
കു	നി	ന	ബ	രു
ഭ	പ	ണ	തേ	ദ
ന	മ്പ	ല	കൈ	ജ

LEARNING

0 EXAMPLES

CONFIDENCE



TRAIN GREEN



0 EXAMPLES

CONFIDENCE



TRAIN PURPLE

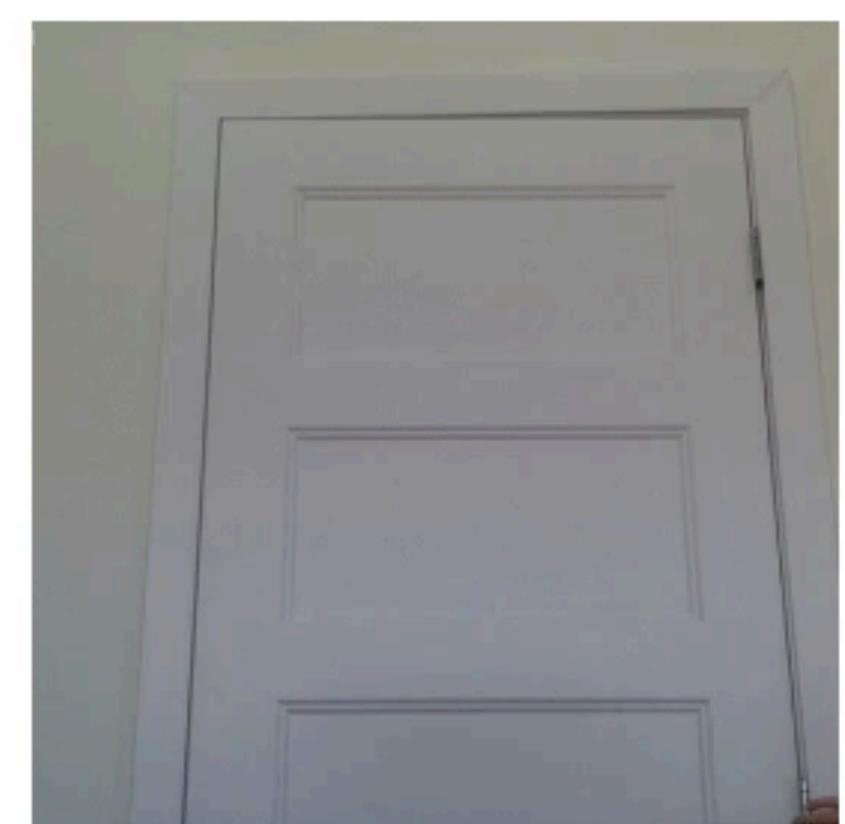
0 EXAMPLES

CONFIDENCE



TRAIN ORANGE

INPUT

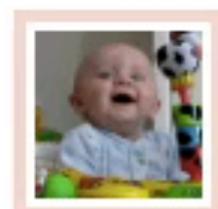


OUTPUT

GIF

Sound

Speech



LEARNING

0 EXAMPLES

CONFIDENCE



TRAIN GREEN



0 EXAMPLES

CONFIDENCE



TRAIN PURPLE

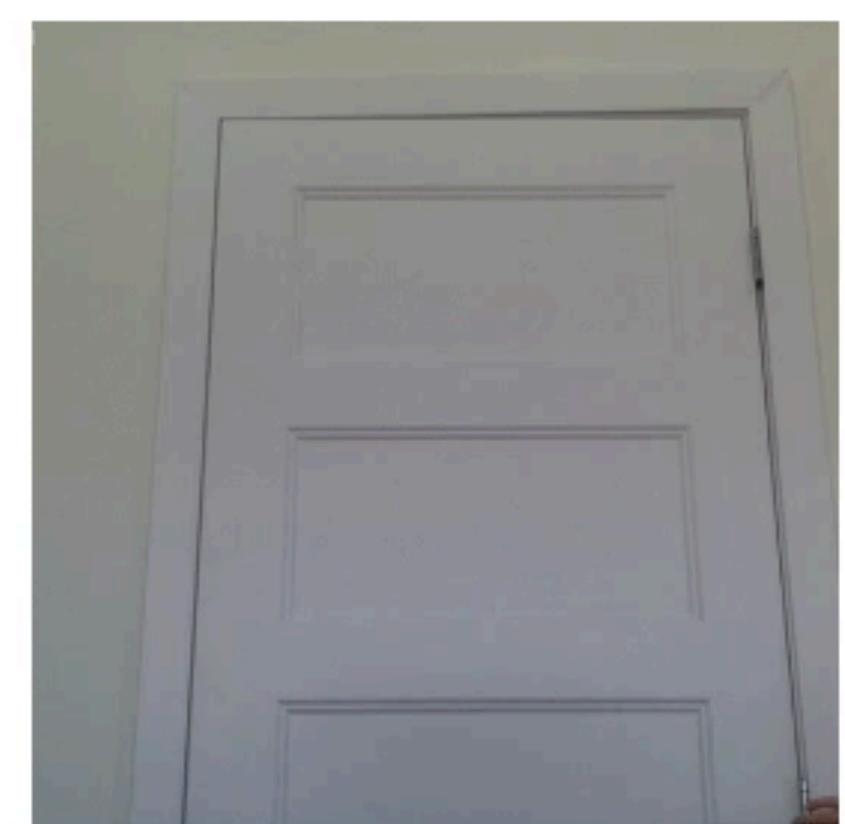
0 EXAMPLES

CONFIDENCE



TRAIN ORANGE

INPUT

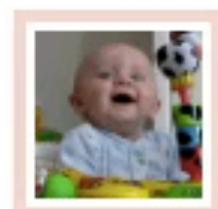


OUTPUT

GIF

Sound

Speech



A RESEARCH AGENDA

- Let's attack directly the problem of **few-shot learning**
 - ▶ we want to design a learning algorithm A that outputs a good parameters θ of a model M , when fed a small dataset $D_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^L$
- Idea: let's learn that algorithm A , end-to-end
 - ▶ this is known as **meta-learning** or **learning to learn**

RELATED WORK: TRANSFER LEARNING

- Large image datasets (e.g. ImageNet) have been shown to allow training representations that transfer to other problems
 - ▶ DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition (2014)
Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng and Trevor Darrell
 - ▶ CNN Features off-the-shelf: an Astounding Baseline for Recognition (2014)
Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, Stefan Carlsson
 - ▶ ...some have even reported some positive transfer on medical imaging datasets!
- 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

RELATED WORK: ONE-SHOT LEARNING

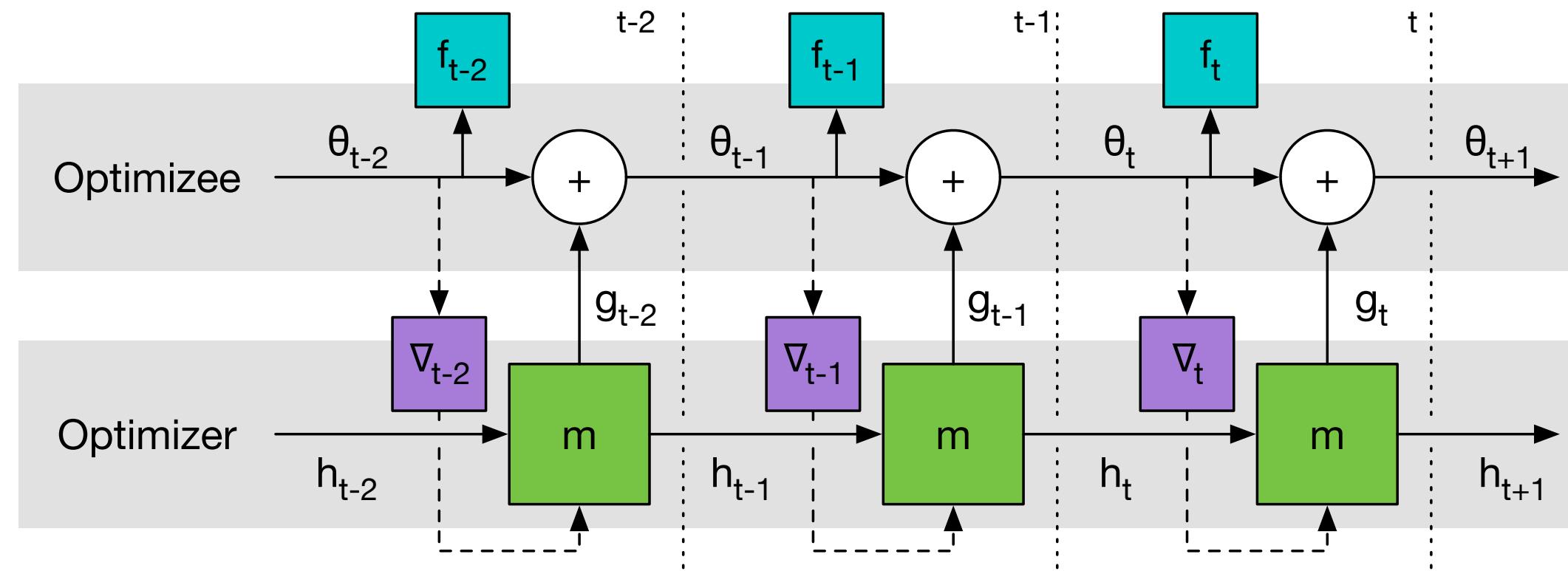
- One-shot learning has been studied before
 - ▶ One-Shot learning of object categories (2006)
Fei-Fei Li, Rob Fergus and Pietro Perona
 - ▶ Knowledge transfer in learning to recognize visual objects classes (2004)
Fei-Fei Li
 - ▶ Object classification from a single example utilizing class relevance 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

RELATED WORK: META-LEARNING

- 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

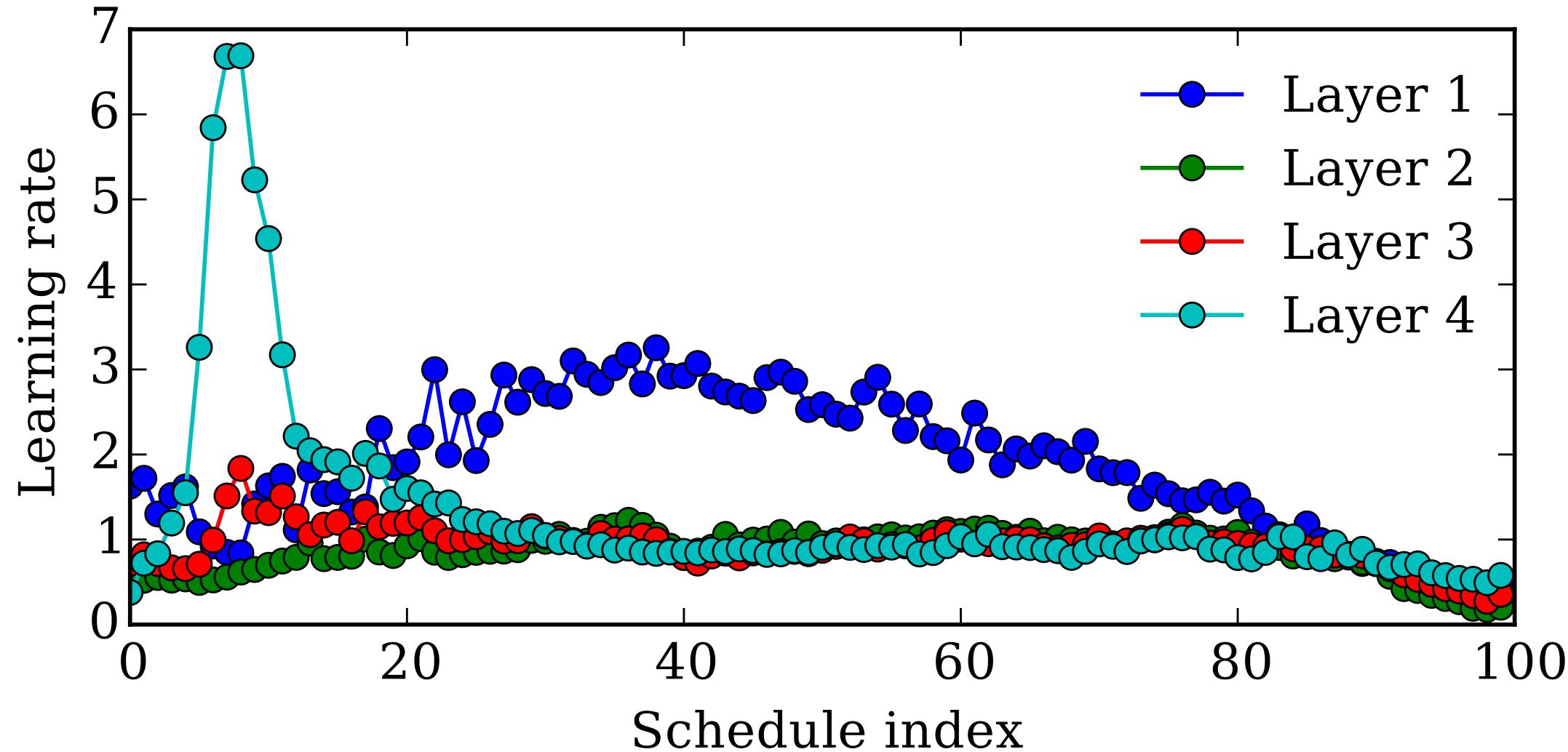


- Learning to learn by gradient descent by gradient descent (2016)
Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas
- Learning to learn using gradient descent (2001)
Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell

RELATED WORK: META-LEARNING

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

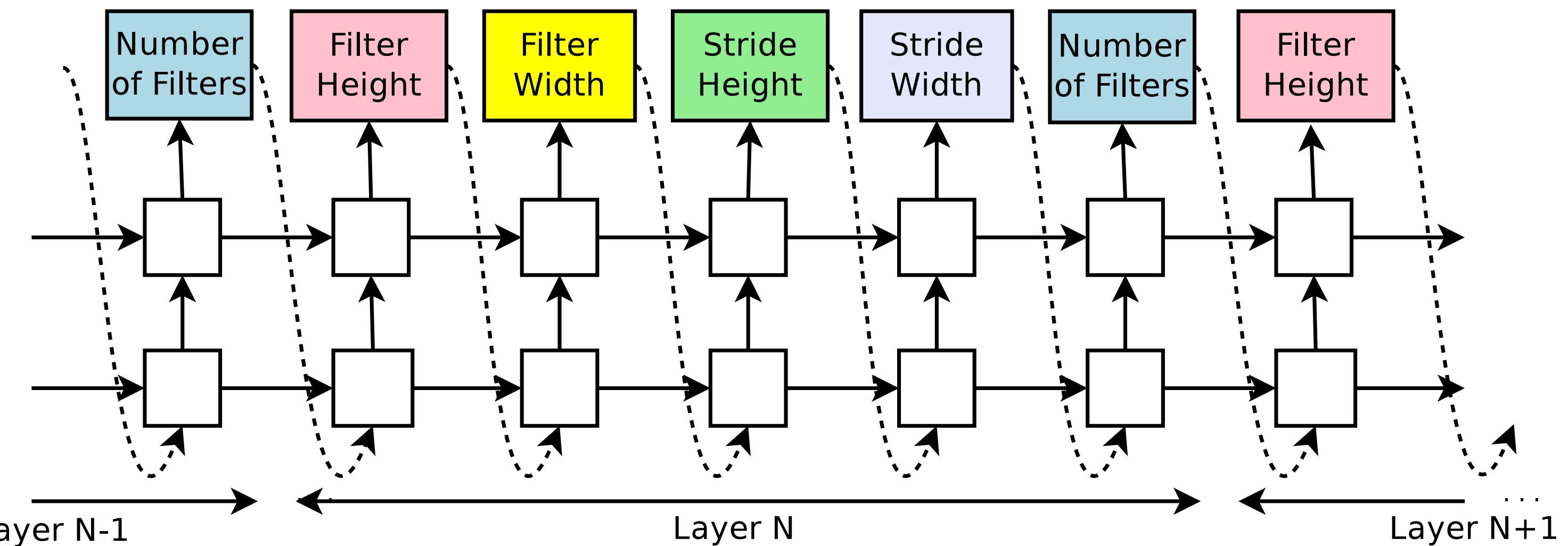
Optimized learning rate schedule



- Gradient-based hyperparameter optimization through reversible learning (2015)
Dougal Maclaurin, 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

If you don't evaluate on never-seen problems/datasets...

If you don't evaluate on never-seen problems/datasets...

... it's not meta-learning!

META-LEARNING

- Learning algorithm A
 - ▶ *input*: training set $D_{train} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$
 - ▶ *output*: parameters θ 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 $\mathcal{D}_{meta-train} = \{(D_{train}^{(n)}, D_{test}^{(n)})\}_{n=1}^N$ of episodes
 - ▶ *output*: parameters Θ algorithm A (the **meta-learner**)
 - ▶ *objective*: good performance on meta-test set $\mathcal{D}_{meta-test} = \{(D'^{({n})}_{train}, D'^{({n})}_{test})\}_{n=1}^{N'}$

META-LEARNING



META-LEARNING

D_{train} | D_{test}

II
episode

**Meta-
Train**
 $\mathcal{D}_{meta-train}$



D_{train}

D_{test}

D_{train}

D_{test}

⋮

⋮

**Meta-
Test**
 $\mathcal{D}_{meta-test}$



D_{train}

D_{test}

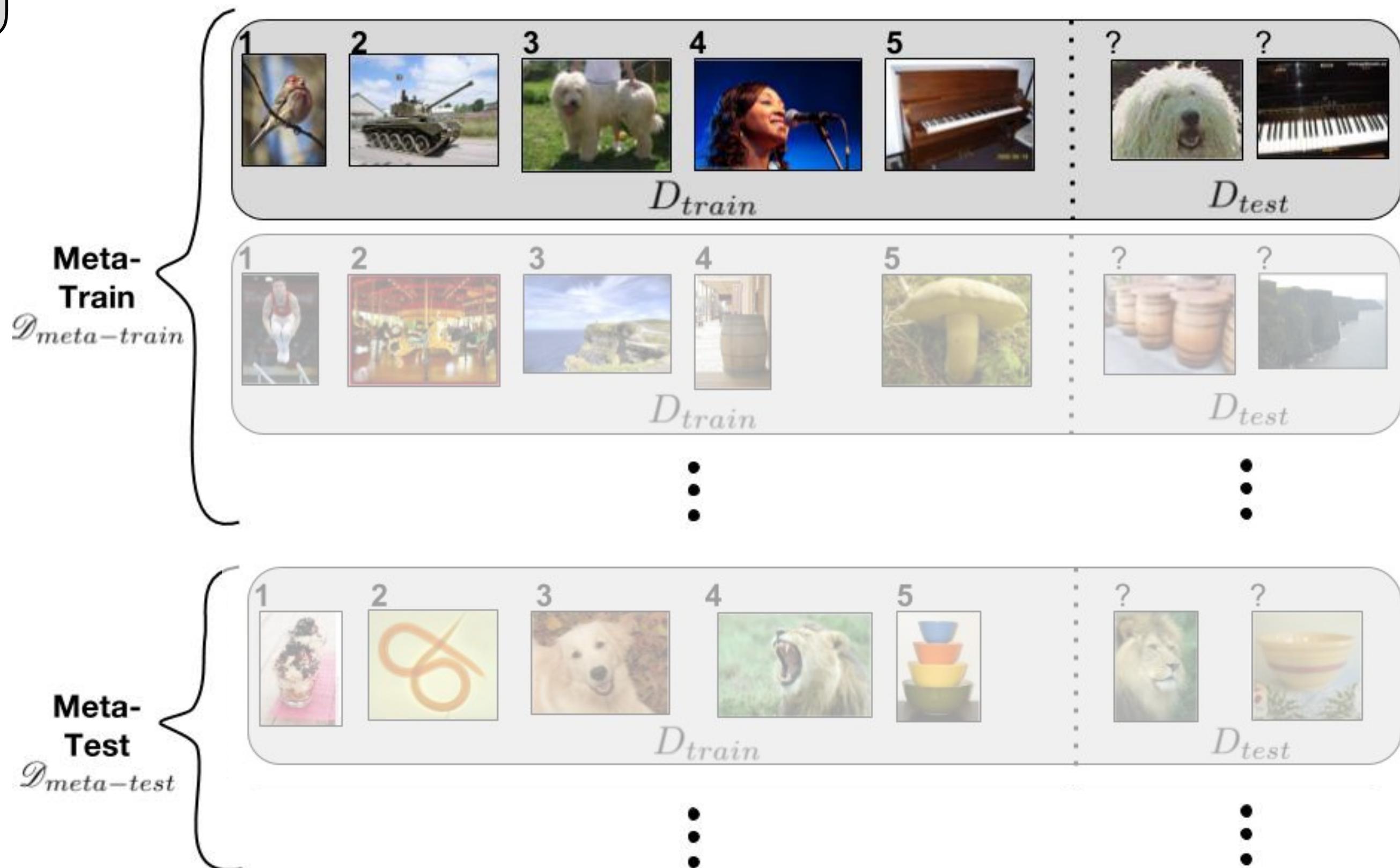
⋮

⋮

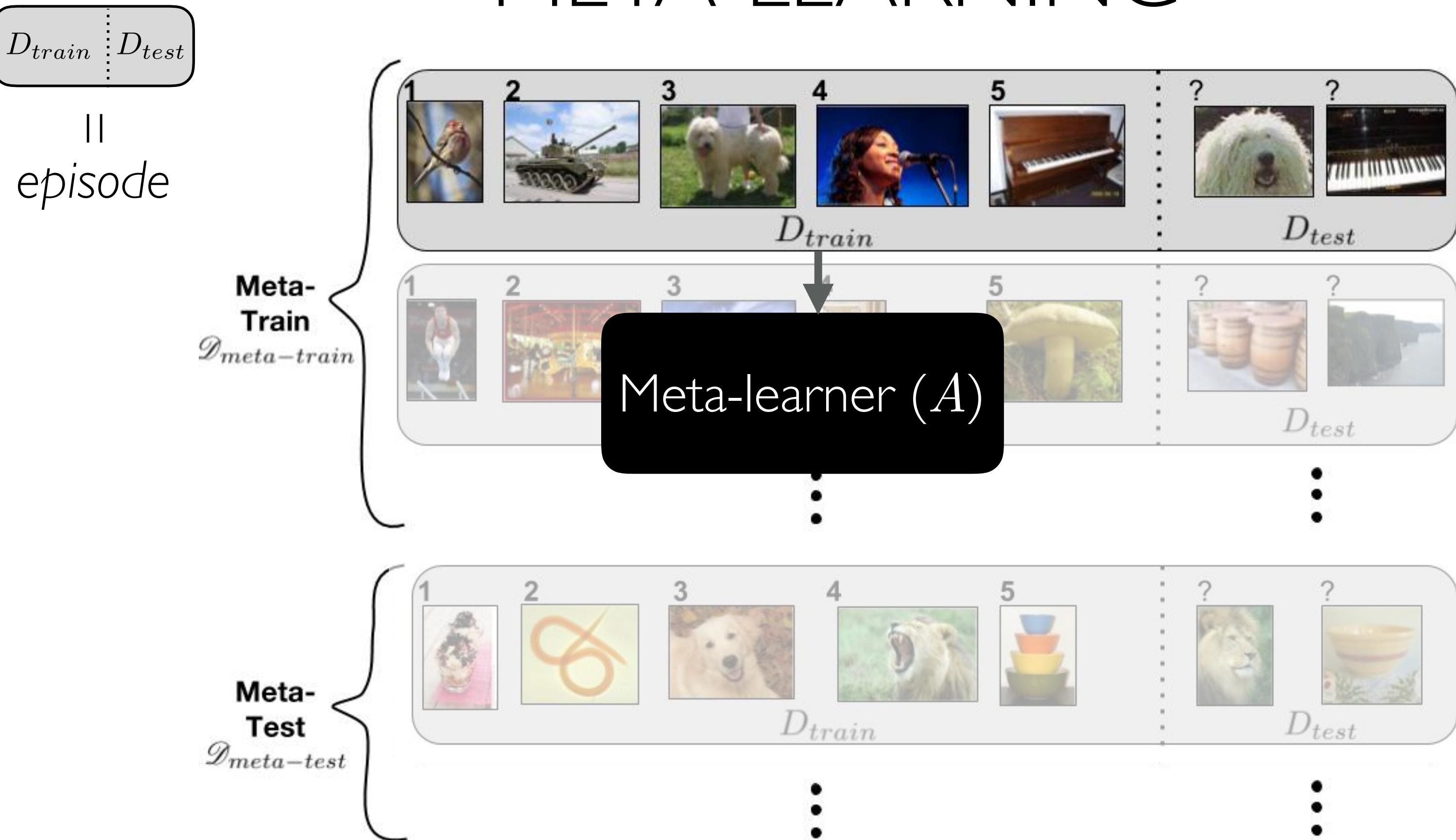
META-LEARNING

D_{train} | D_{test}

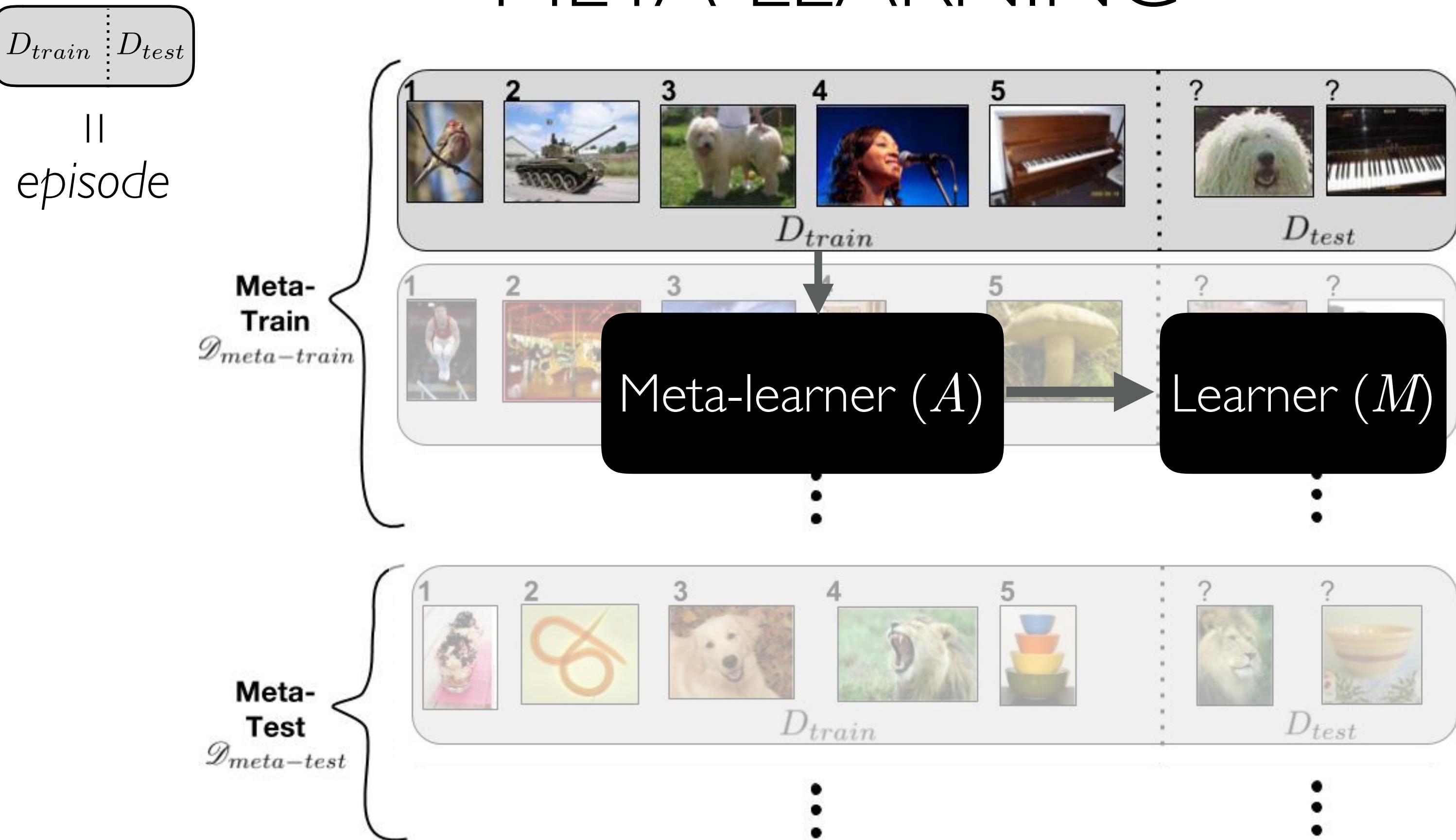
II
episode



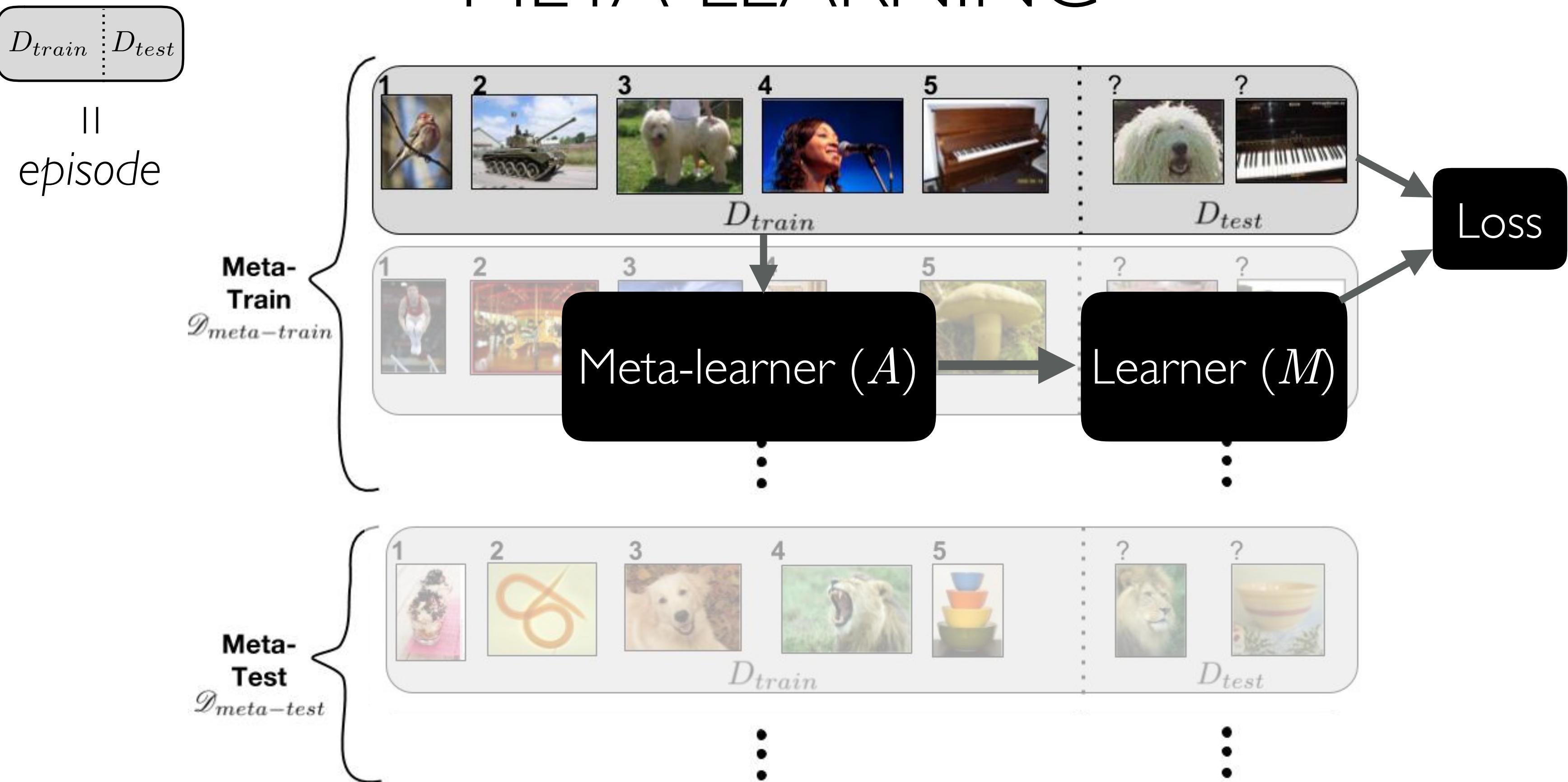
META-LEARNING



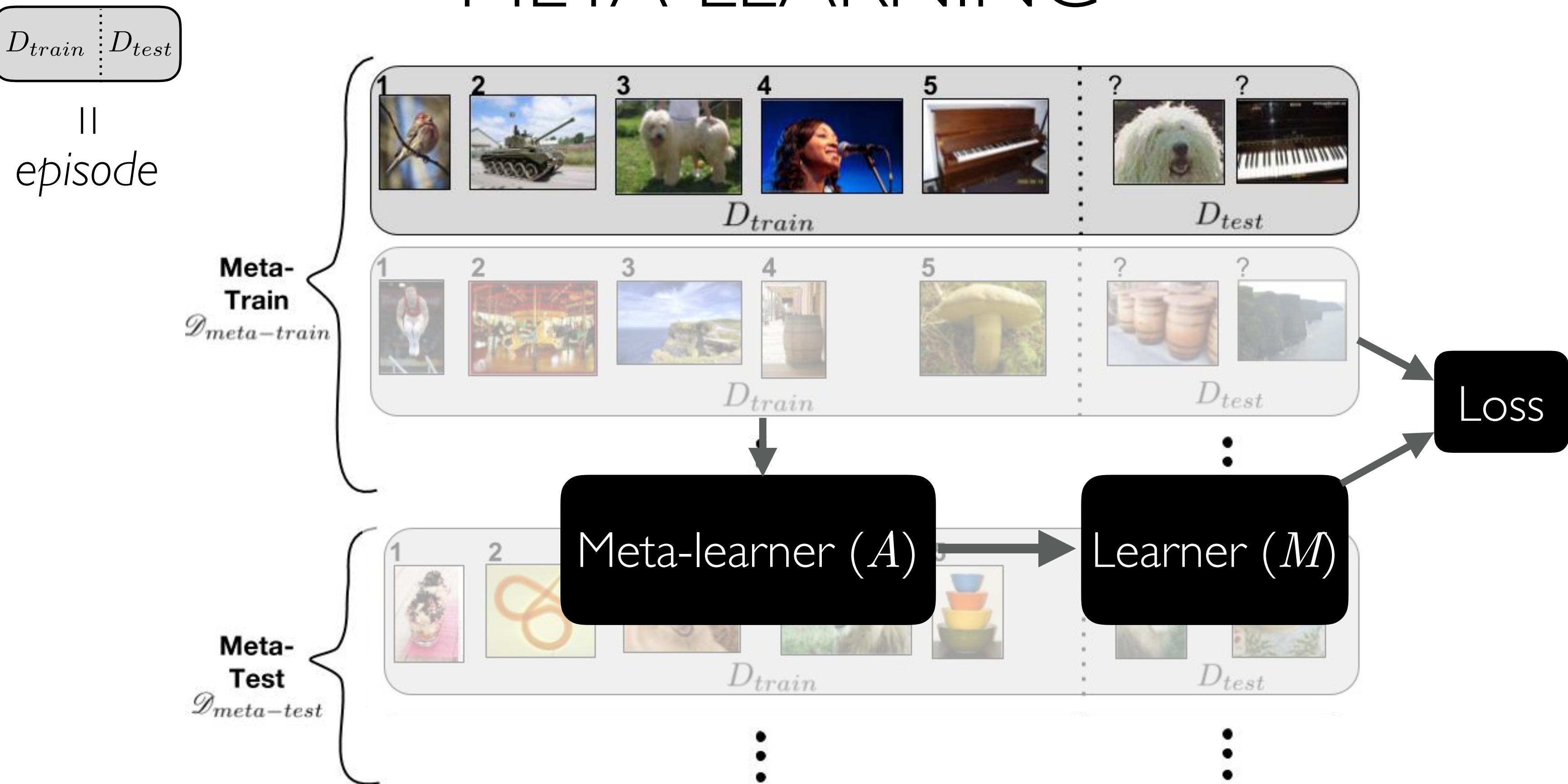
META-LEARNING



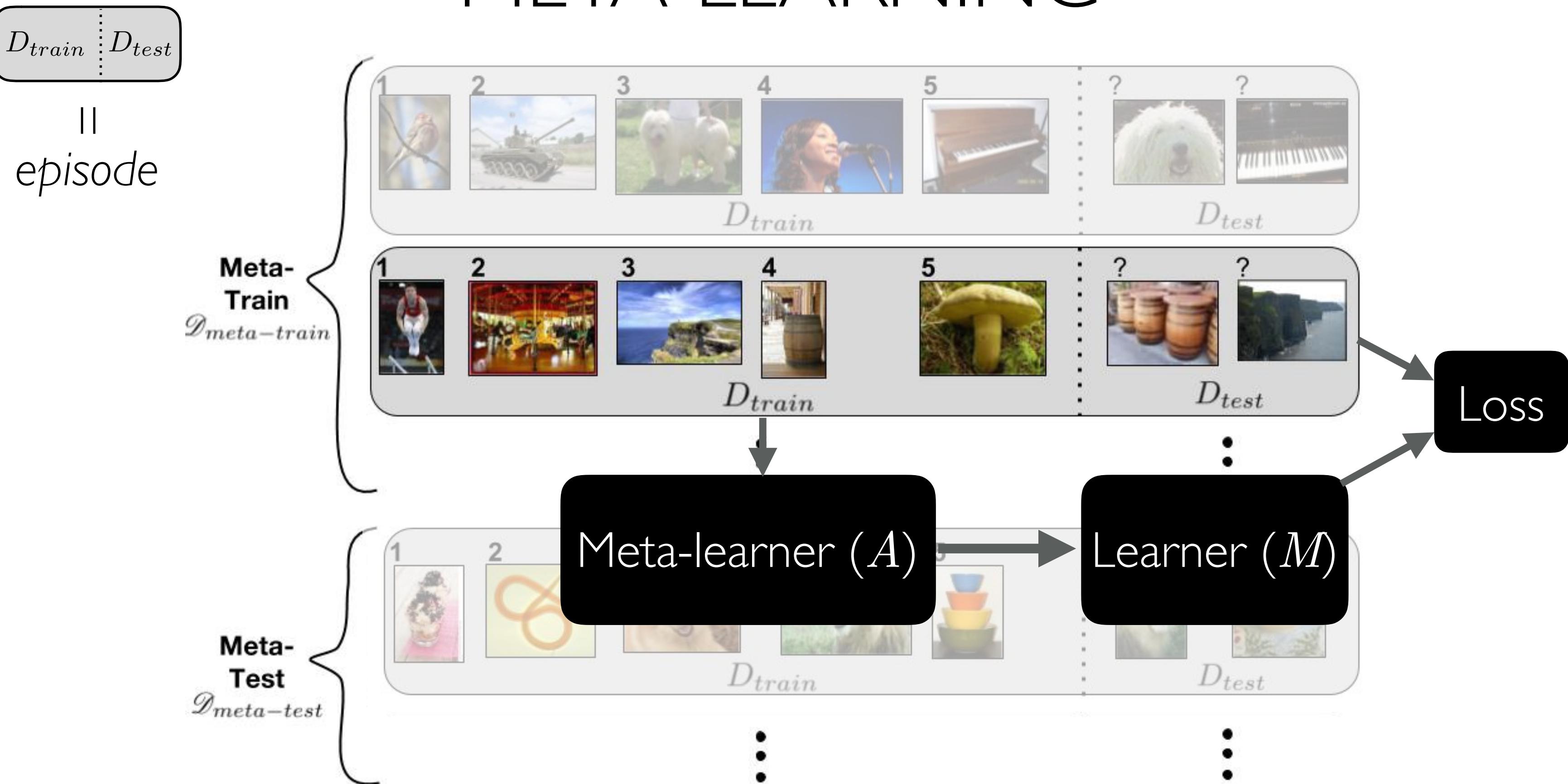
META-LEARNING



META-LEARNING



META-LEARNING



META-LEARNING NOMENCLATURE

Training set

Test set

Meta-training set

Meta-test set

META-LEARNING NOMENCLATURE

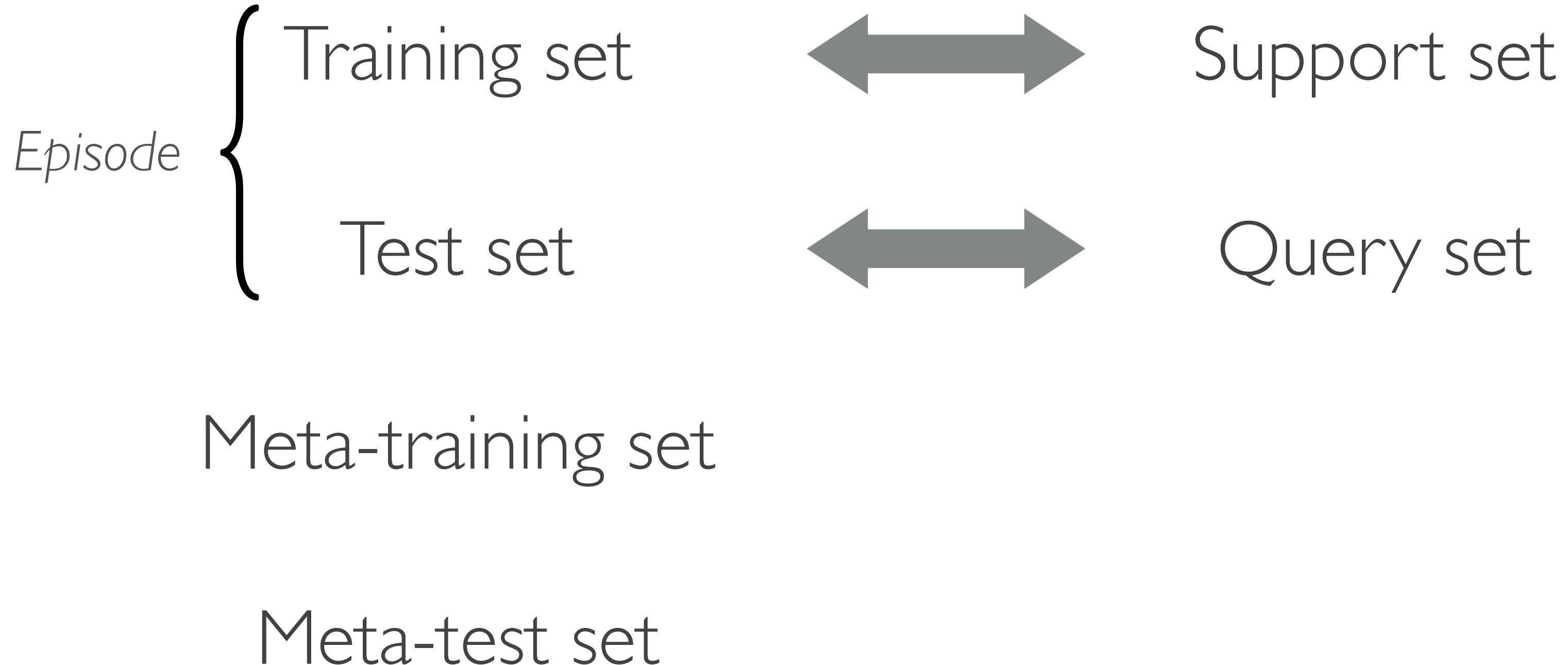
Episode {

- Training set
- Test set

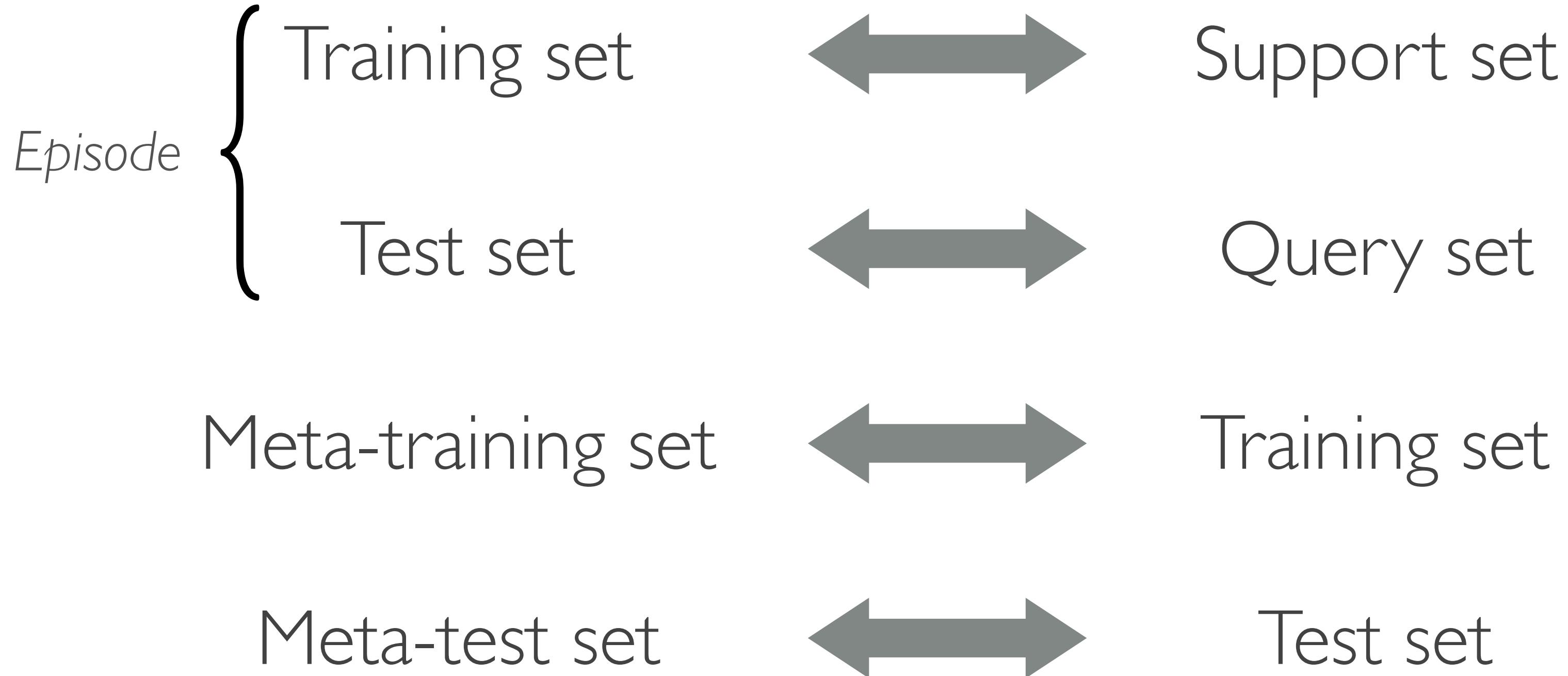
Meta-training set

Meta-test set

META-LEARNING NOMENCLATURE



META-LEARNING NOMENCLATURE



LEARNING PROBLEM STATEMENT

- 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

CHOOSING A 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)

CHOOSING A META-LEARNER

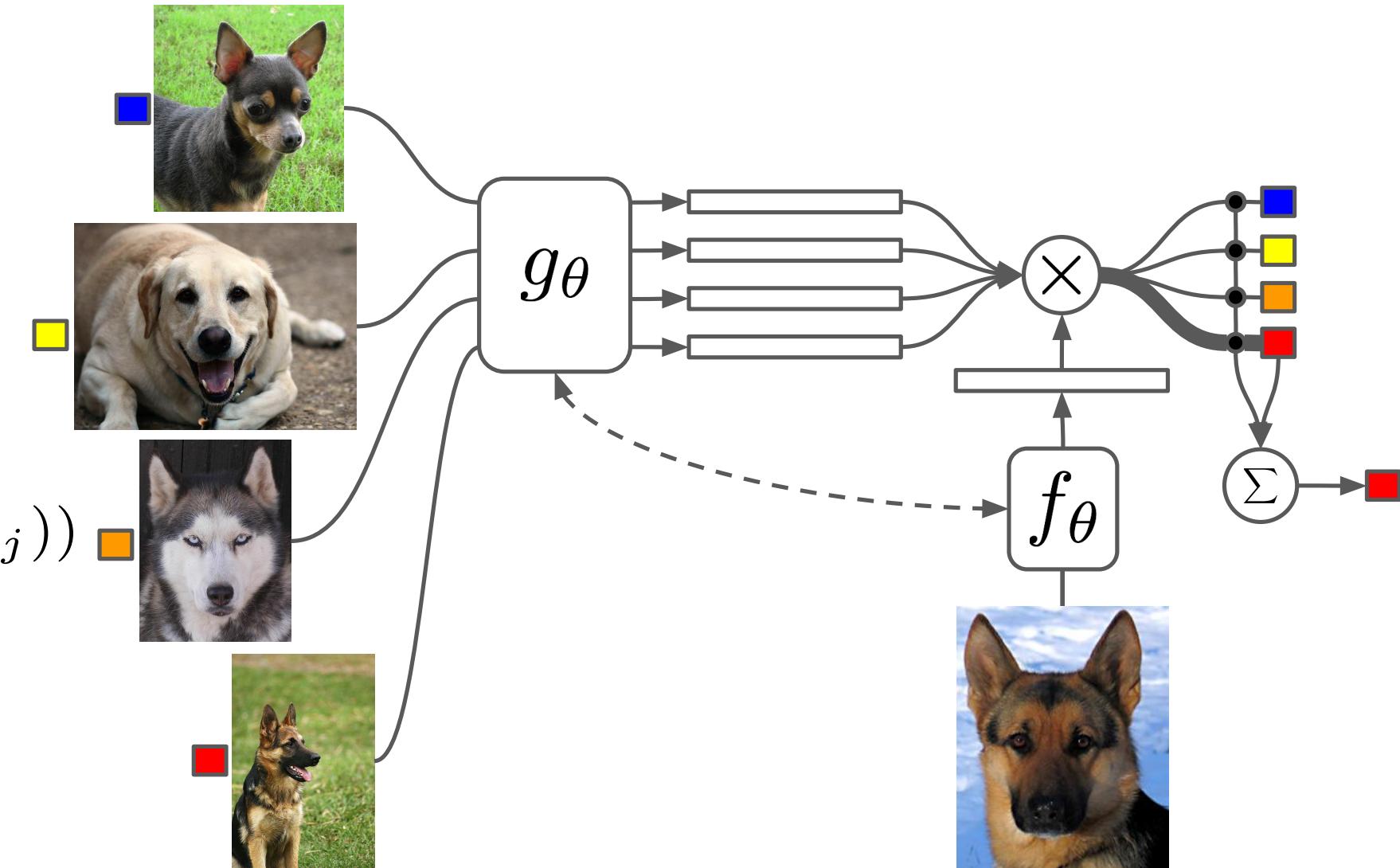
- How to parametrize learning algorithms?
- Two approaches to defining a meta-learner
 - ▶ **Take inspiration from a known learning algorithm**
 - kNN/kernel machine: Matching networks (Vinyals et al. 2016)
 - Gaussian classifier: Prototypical Networks (Snell et al. 2017)
 - Gradient Descent: Meta-Learner 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

PROTOTYPICAL NETWORKS

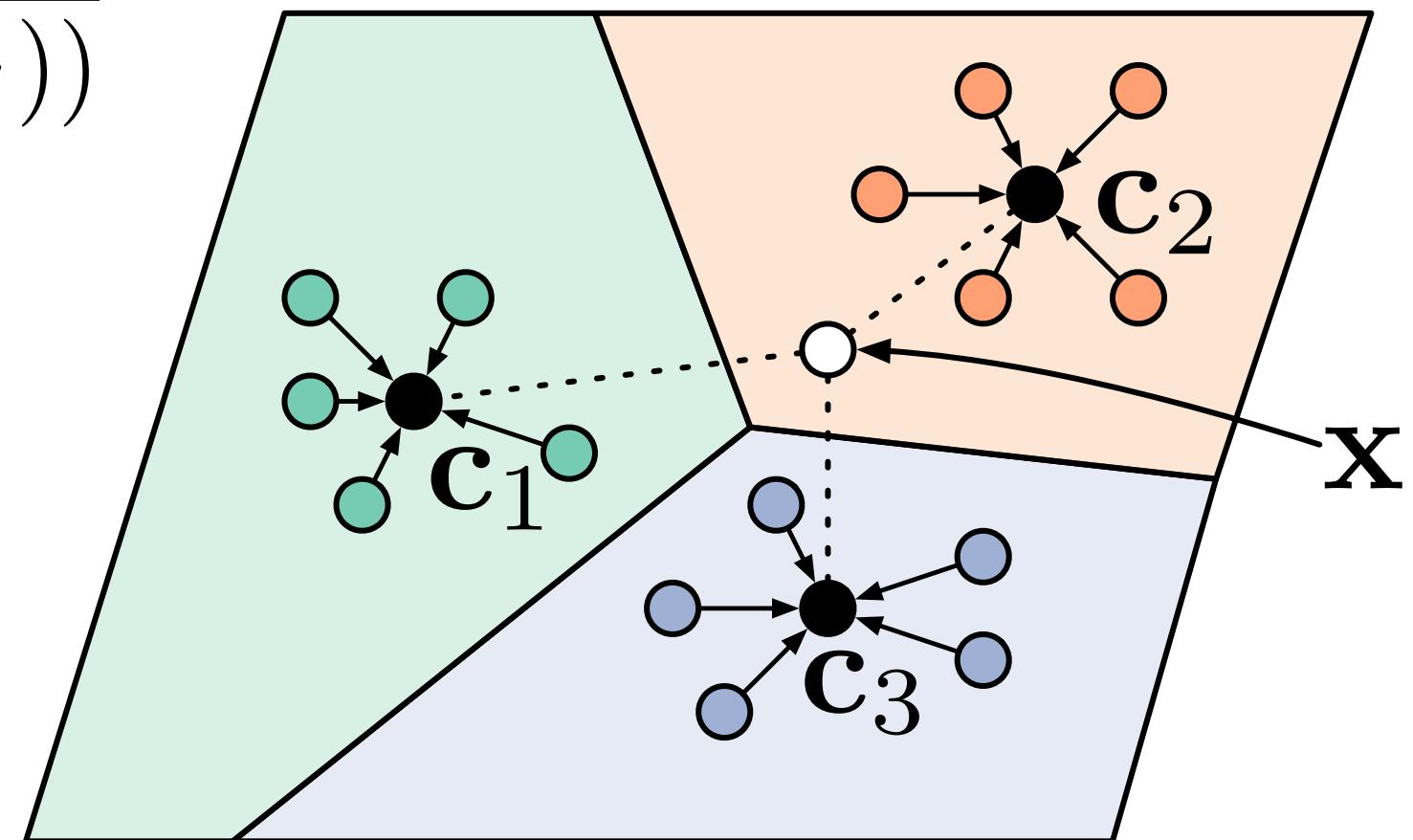
- Training a “**prototype extractor**”

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

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

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

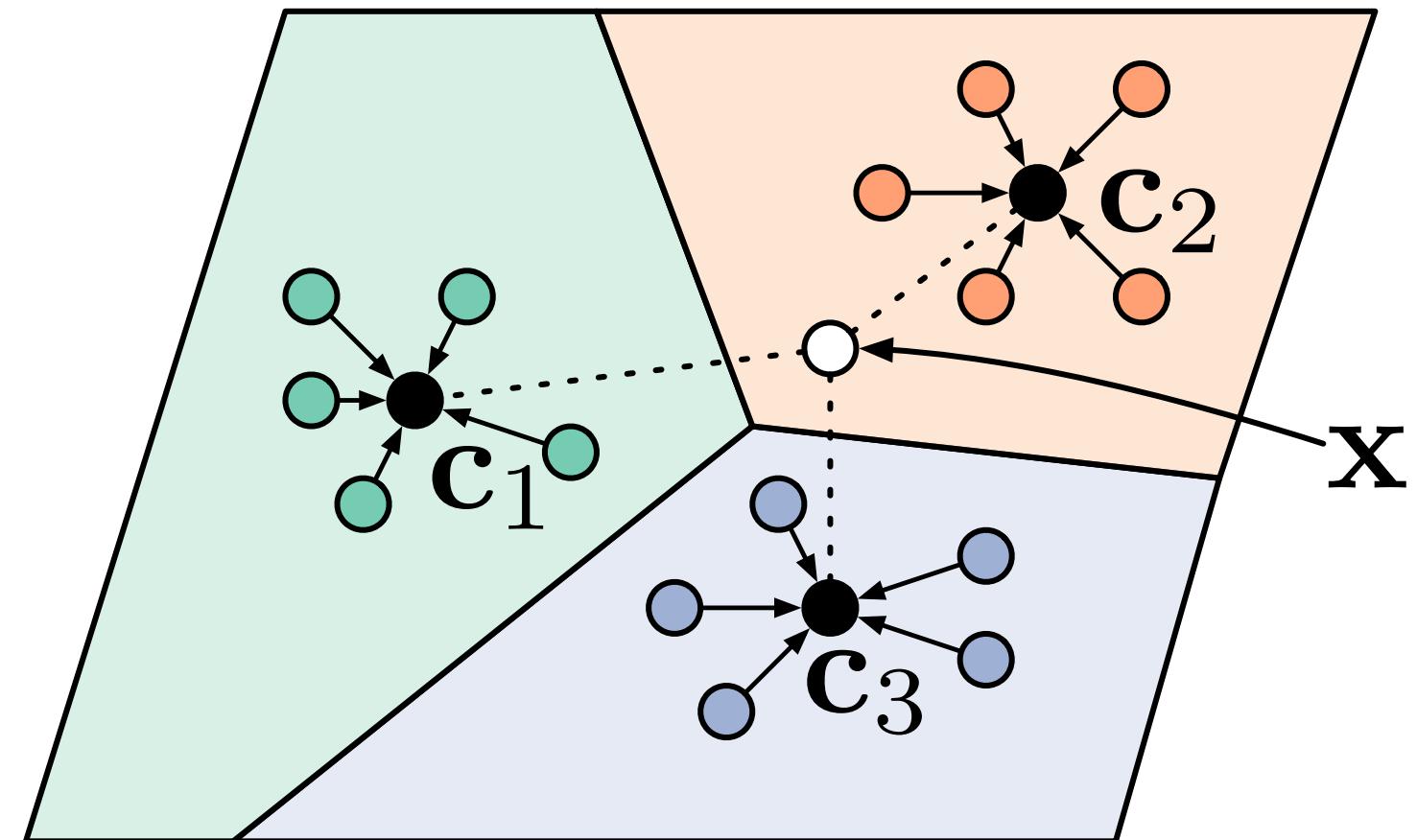
$$\phi \equiv \Theta$$



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

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

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

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

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

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

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

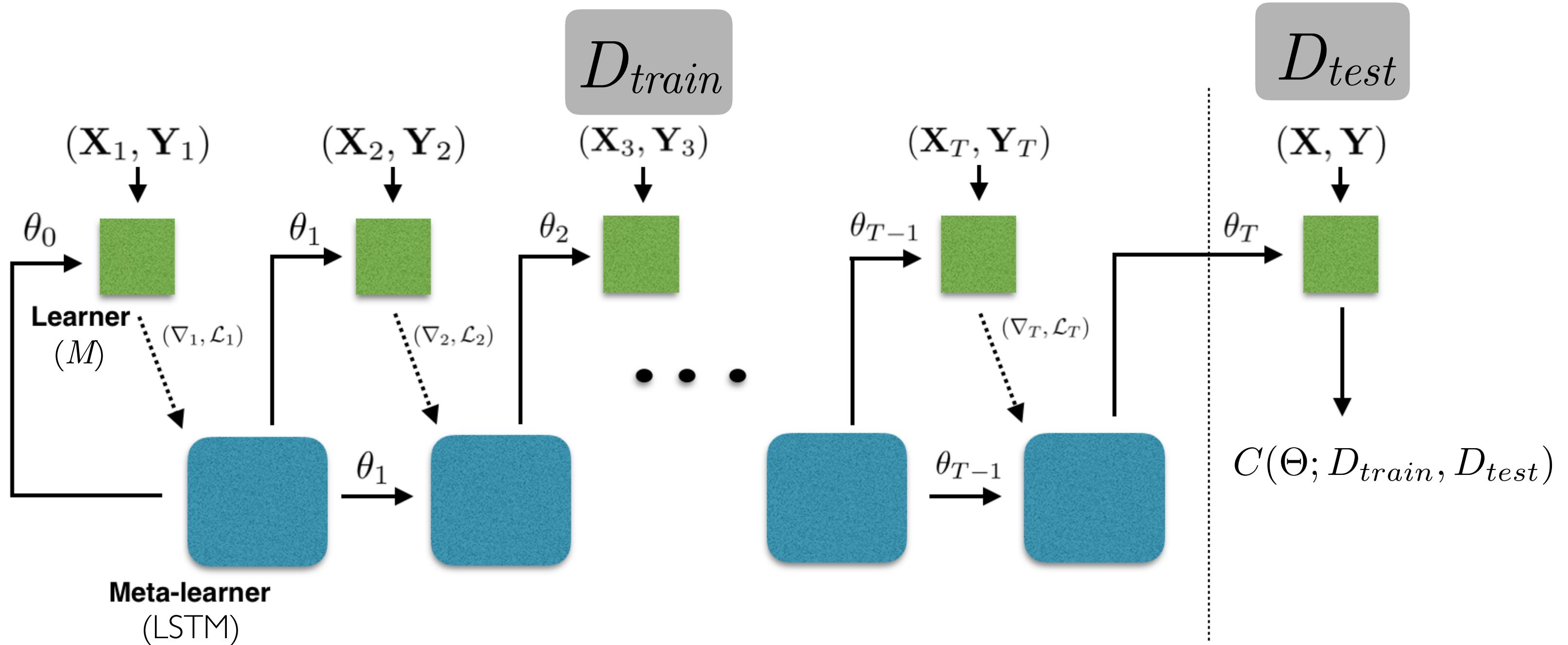
- state c_t is model M 's parameter space θ_t \leftarrow c_0 becomes a *learned initialization*
- state update \tilde{c}_t is the negative gradient $-\nabla_{\theta_{t-1}} \mathcal{L}_t$
- f_t and i_t are LSTM gates: $i_t = \sigma(\mathbf{W}_I \cdot [\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}] + \mathbf{b}_I)$ \leftarrow *adaptive learning rate*
 $f_t = \sigma(\mathbf{W}_F \cdot [\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}] + \mathbf{b}_F)$ \leftarrow *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-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)

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

- 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

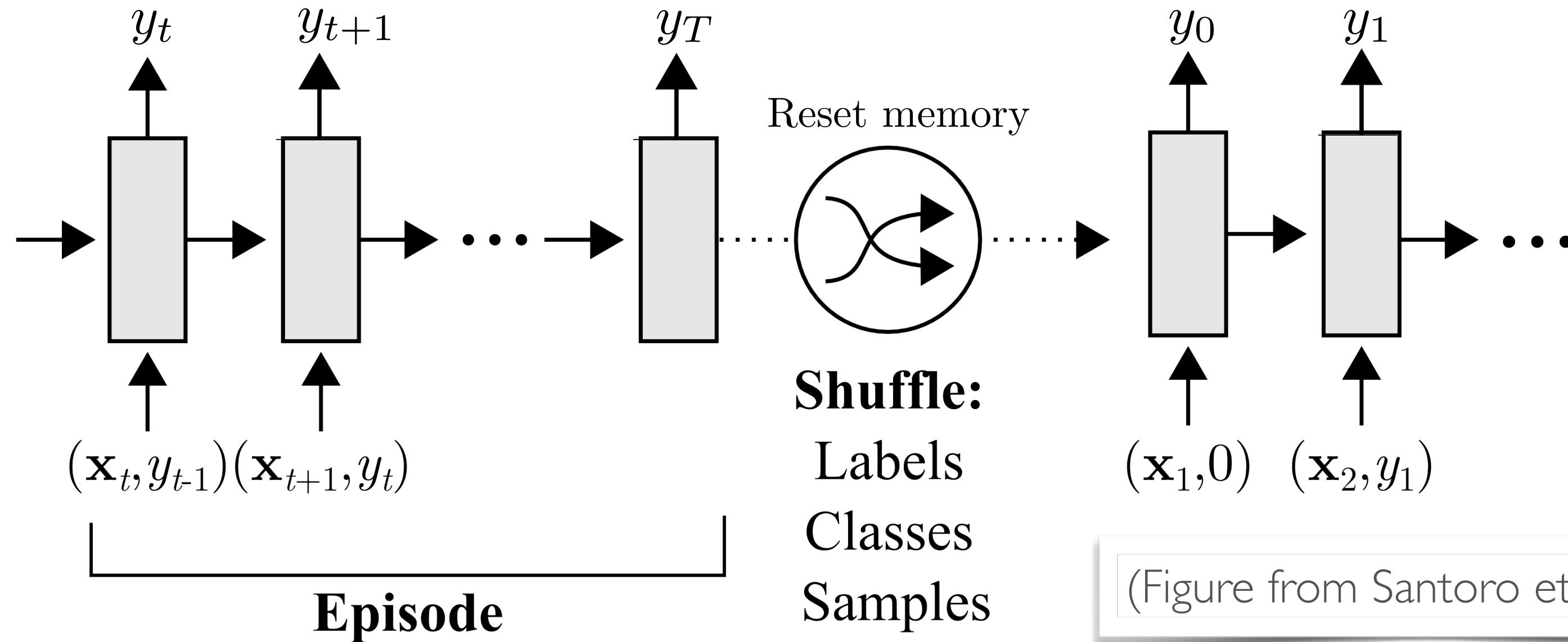
- Training a “**gradient descent procedure**” applied on some learner M
 - ▶ MAML proposes not to bother with training an LSTM for the gradient descent updates and constant step-size updates
 - ▶ better results are also reported by the so-called *bias transformation* architecture (*One-Shot Visual Imitation 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

CHOOSING A 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

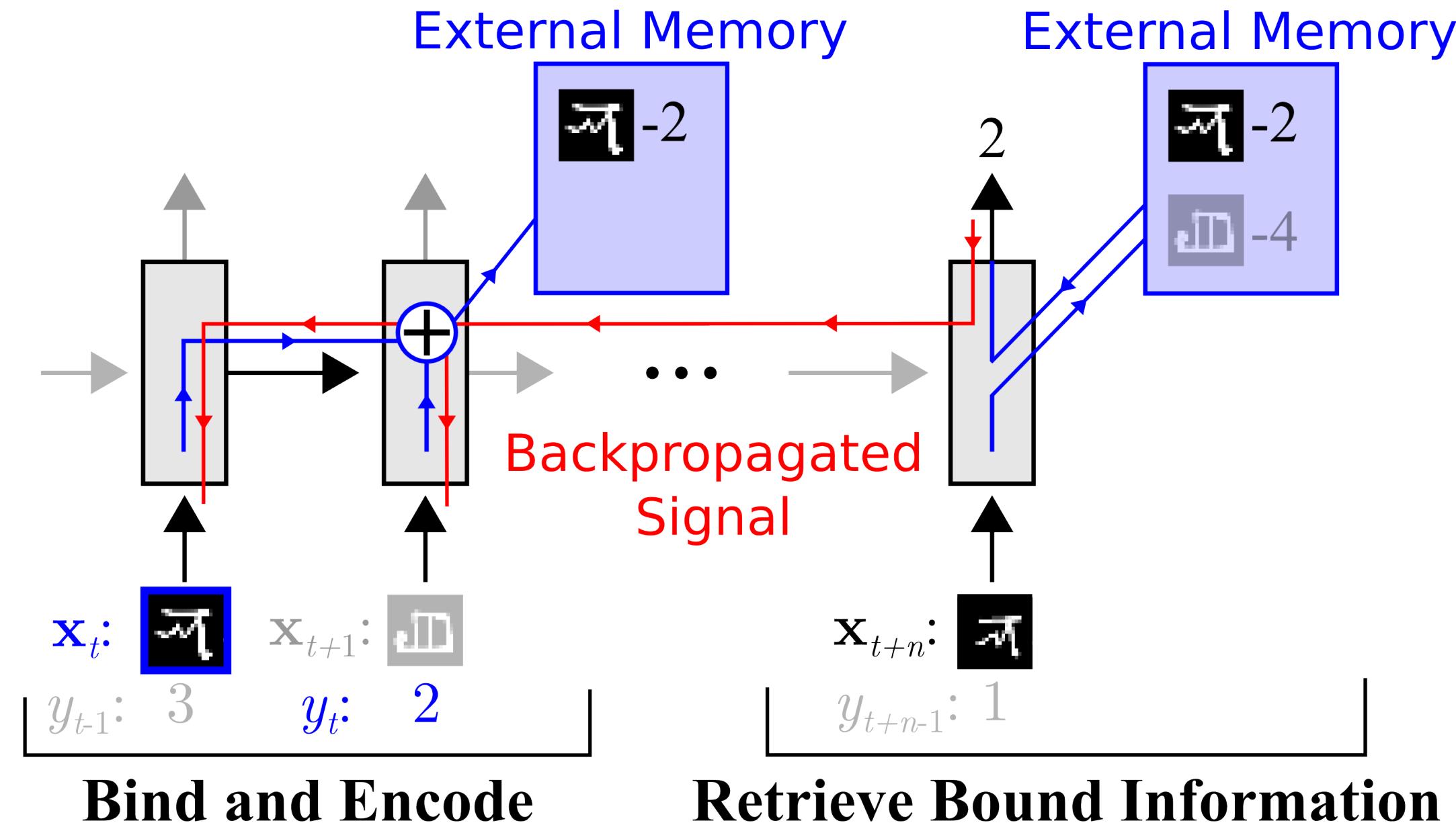
- Frame meta-learning as sequence labeling with correct labels as delayed inputs



- 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



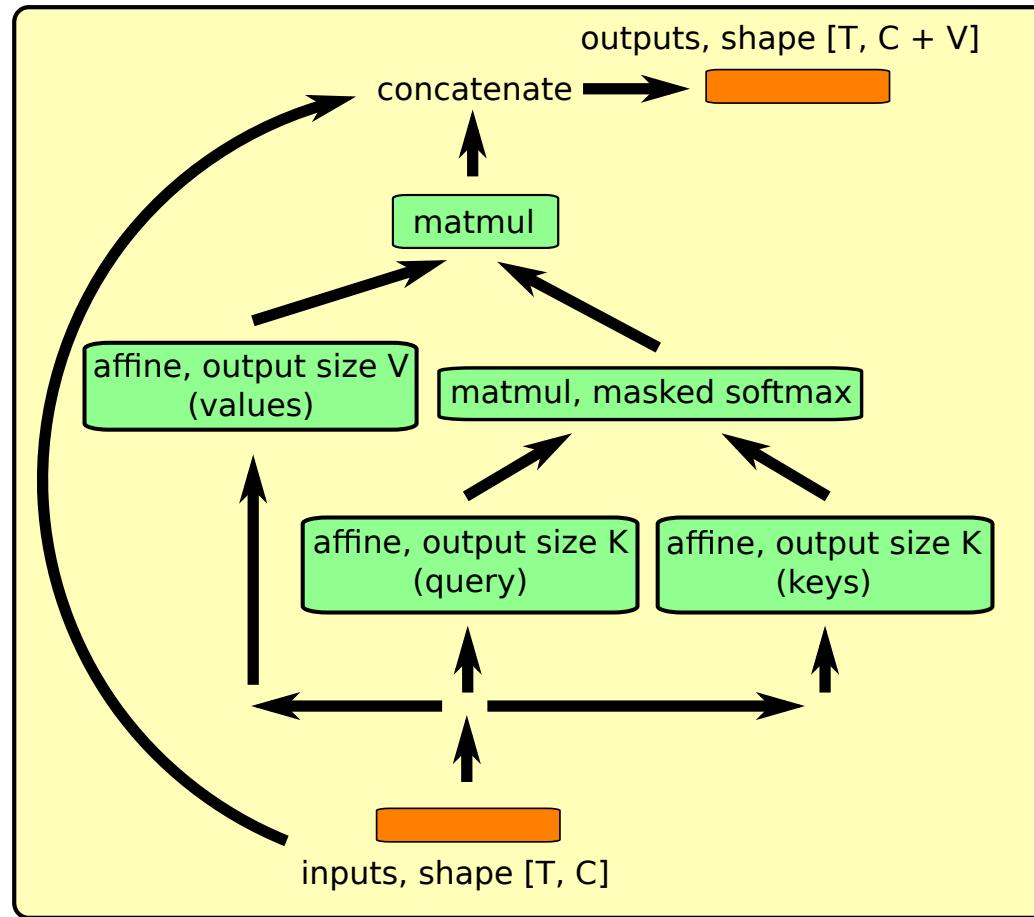
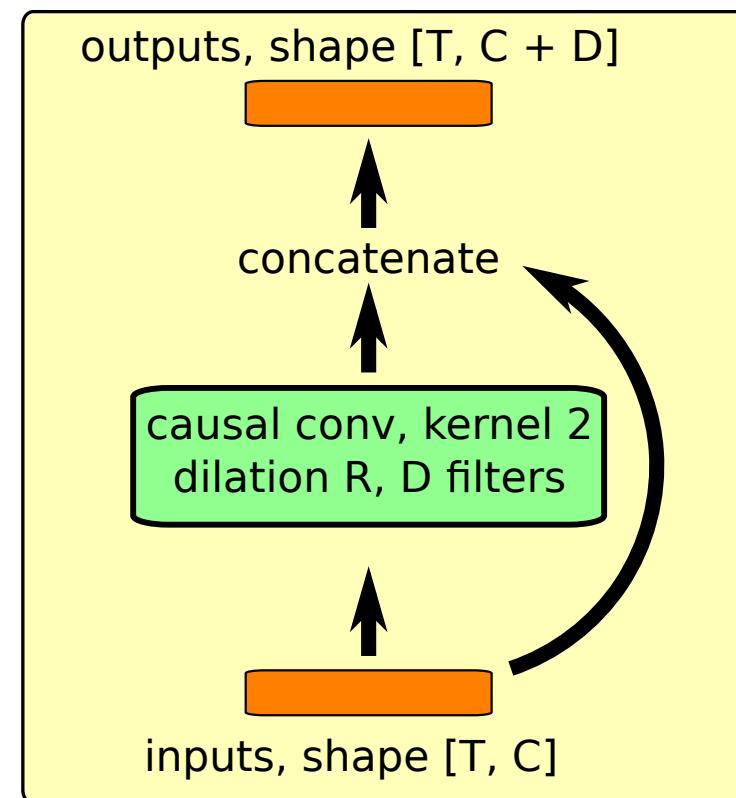
- One-shot learning with memory-augmented neural networks (2016)
Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap

MEMORY-AUGMENTED NEURAL NETWORK

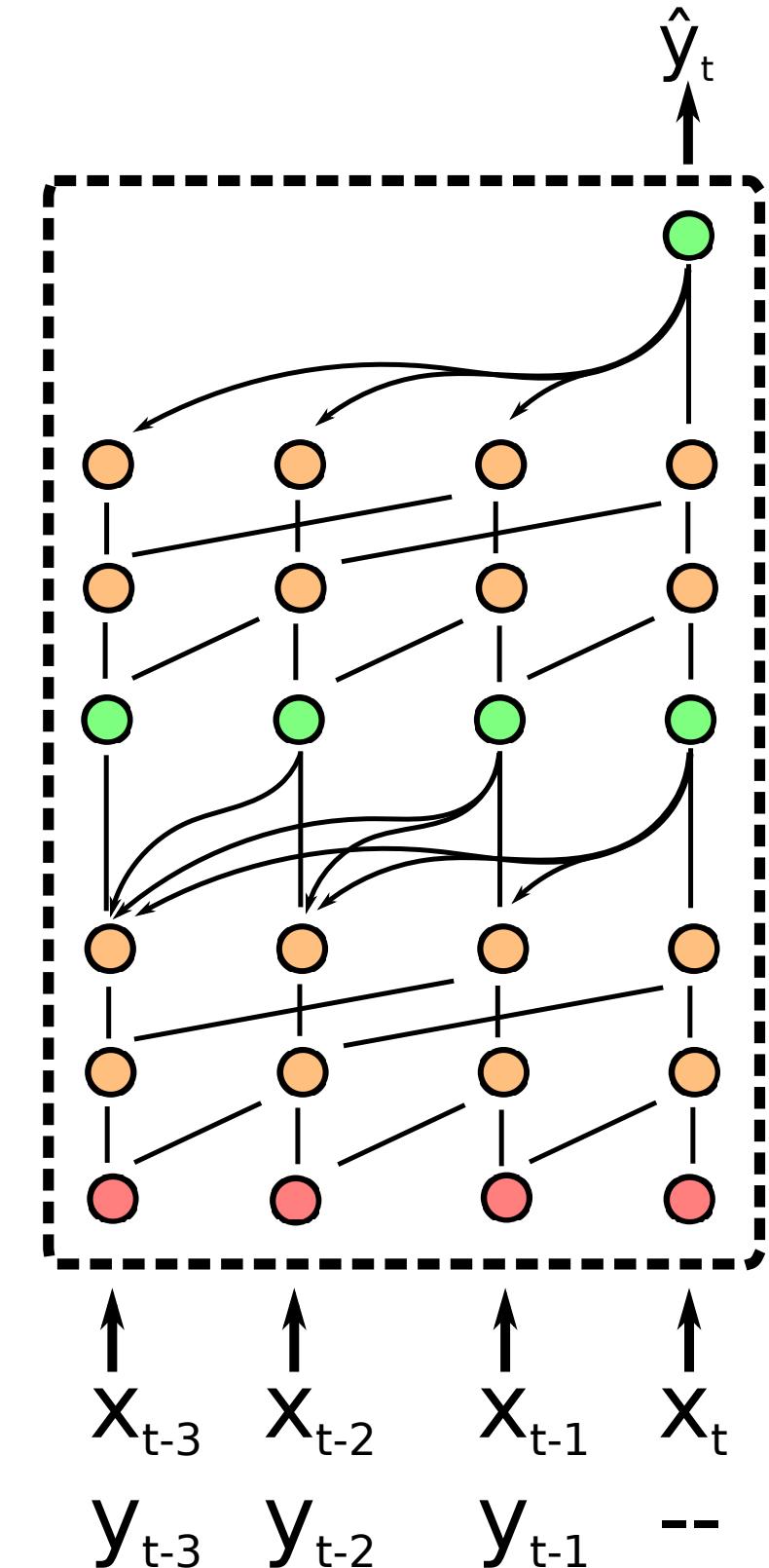
- Training a **neural Turing machine** to learn a learning algorithm
 - ▶ required adapting the original neural Turing machine, using a fairly non-trivial Least Recently Used Access (LRUA) writing mechanism
 - ▶ only shown to work on Omniglot dataset (and not on the harder split used by Lake et al. 2015)
- One-shot learning with memory-augmented neural networks (2016)
Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap

SIMPLE NEURAL ATTENTIVE LEARNER

- Training a **convolutional/attentional network** to learn
 - ▶ alternates between **dilated convolutional layers** and **attentional layers**
 - ▶ when inputs are images, an convolutional embedding network is used to map to a vector space

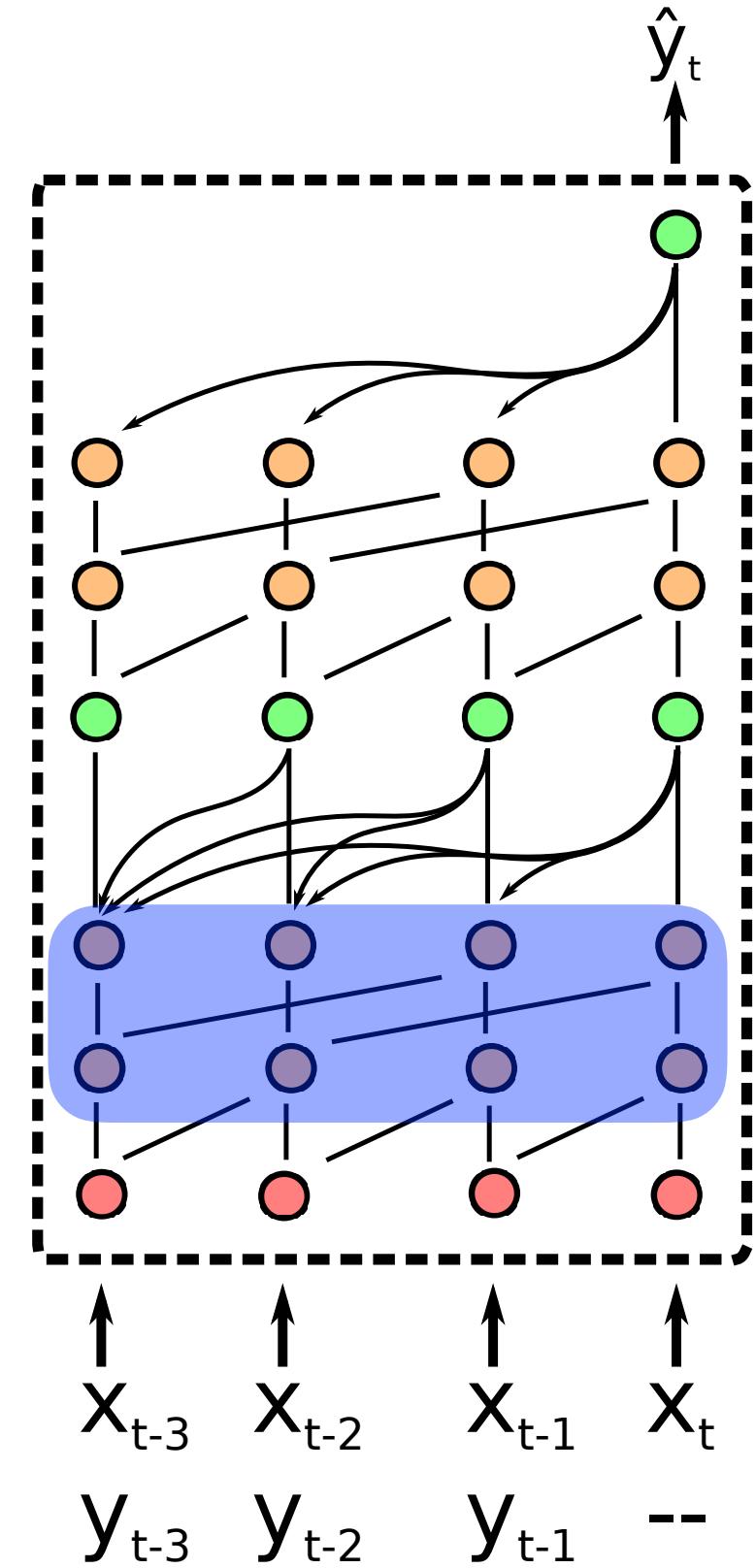
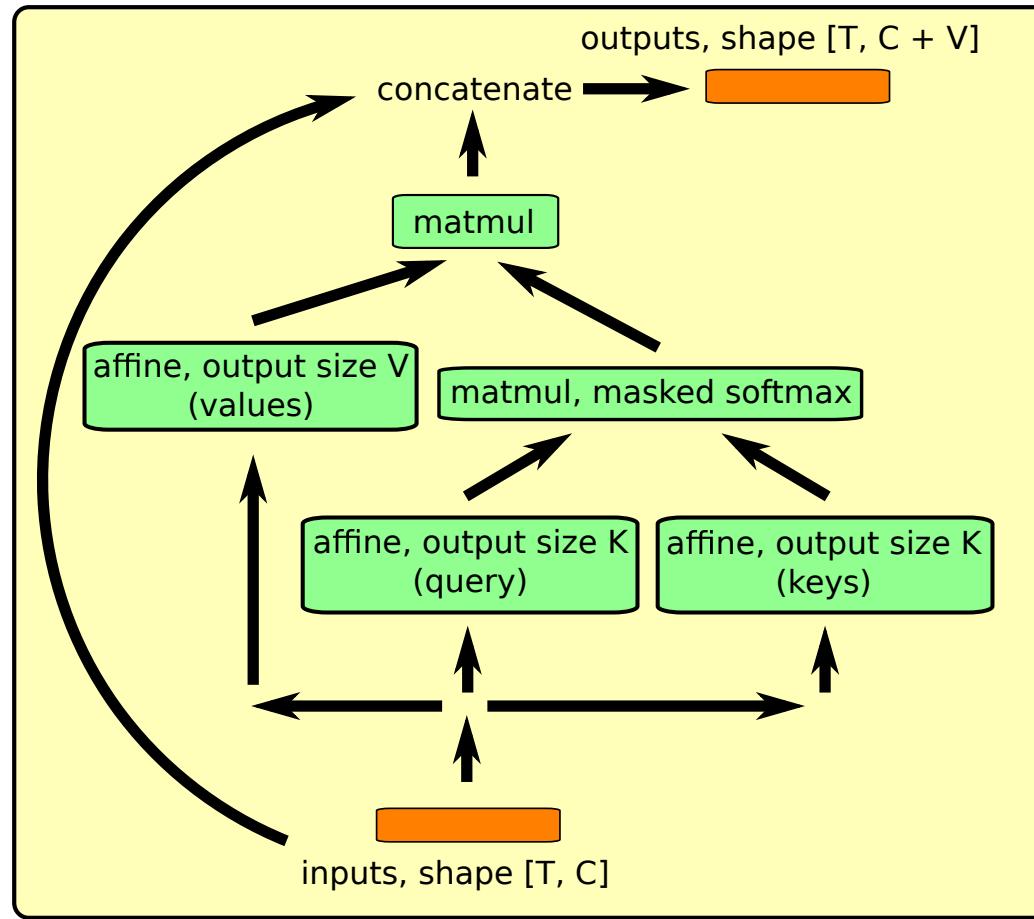
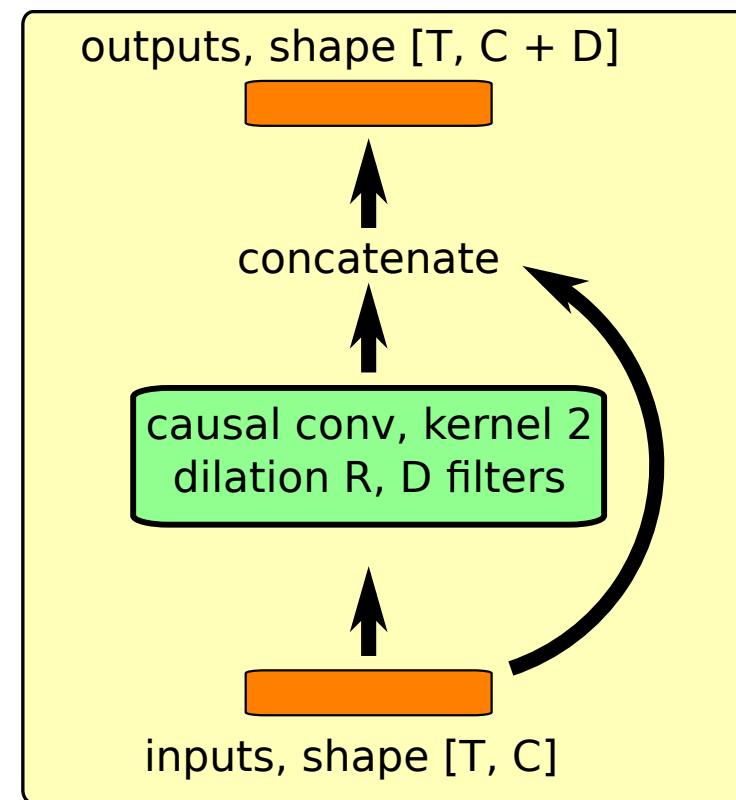


- A Simple Neural Attentive Meta-Learner (2018)
Nikhil Mishra, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel



SIMPLE NEURAL ATTENTIVE LEARNER

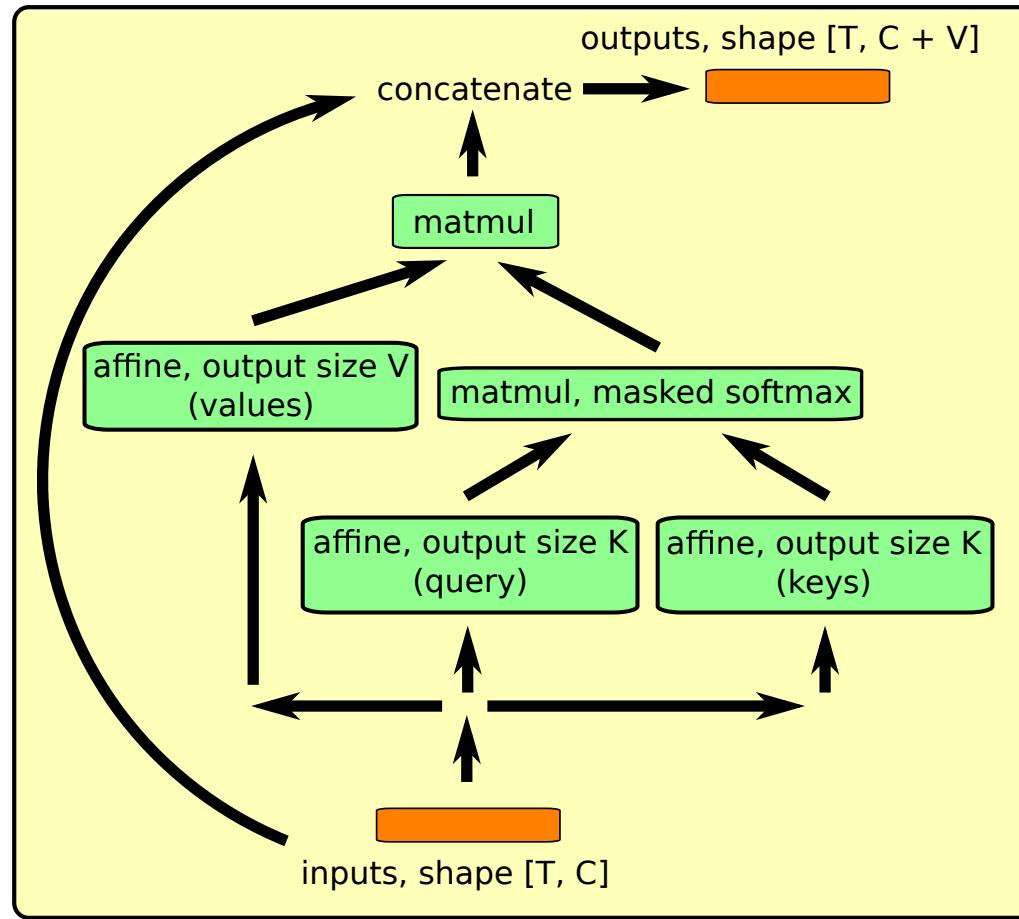
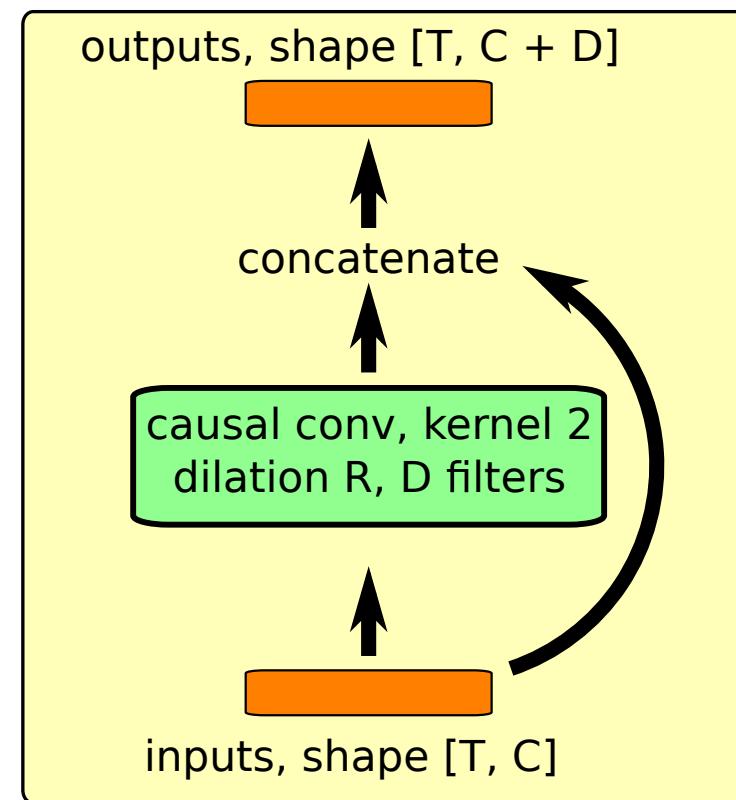
- Training a **convolutional/attentional network** to learn
 - ▶ alternates between **dilated convolutional layers** and **attentional layers**
 - ▶ when inputs are images, an convolutional embedding network is used to map to a vector space



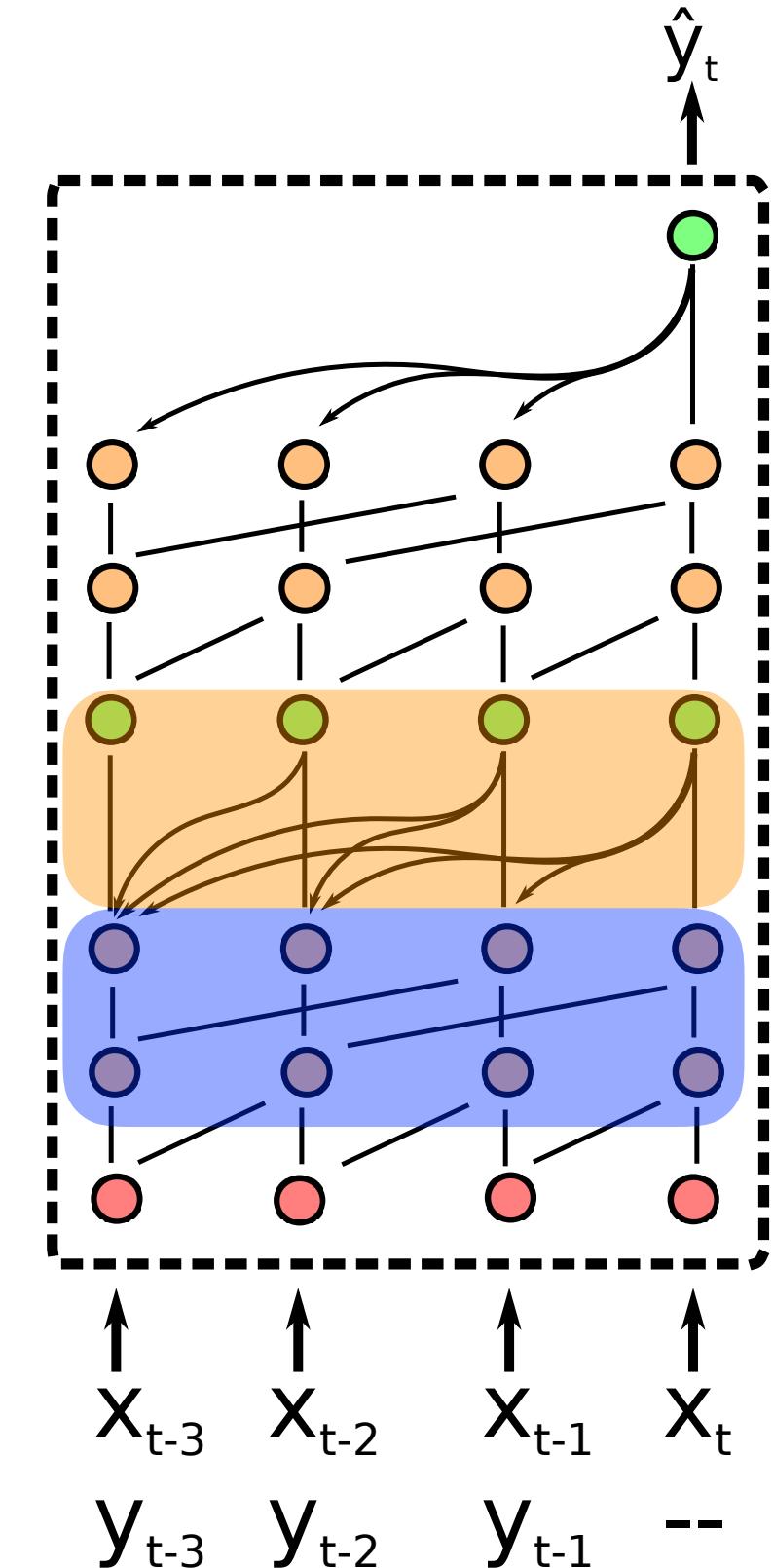
- A Simple Neural Attentive Meta-Learner (2018)
Nikhil Mishra, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel

SIMPLE NEURAL ATTENTIVE LEARNER

- Training a **convolutional/attentional network** to learn
 - ▶ alternates between **dilated convolutional layers** and **attentional layers**
 - ▶ when inputs are images, an convolutional embedding network is used to map to a vector space

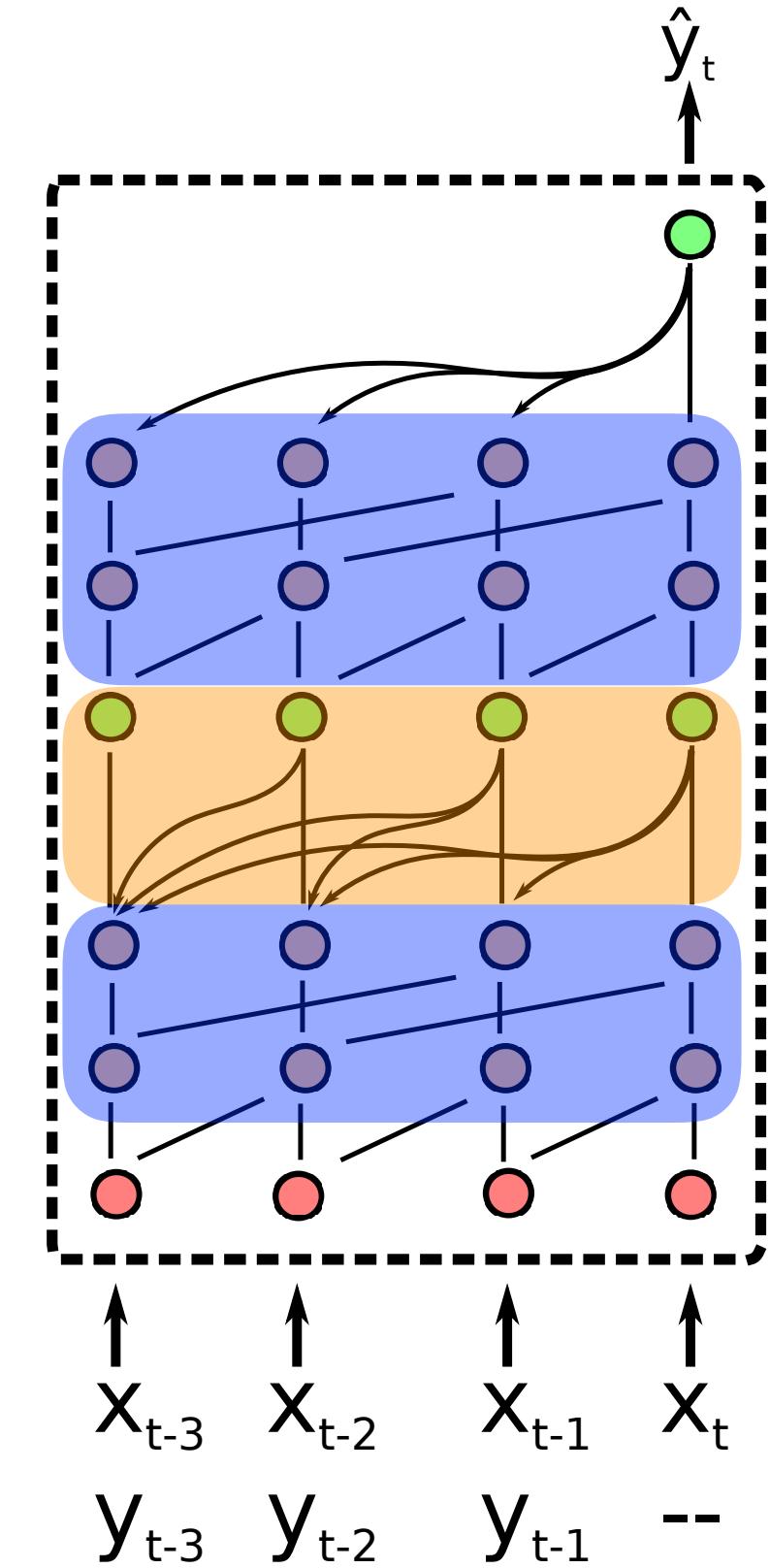
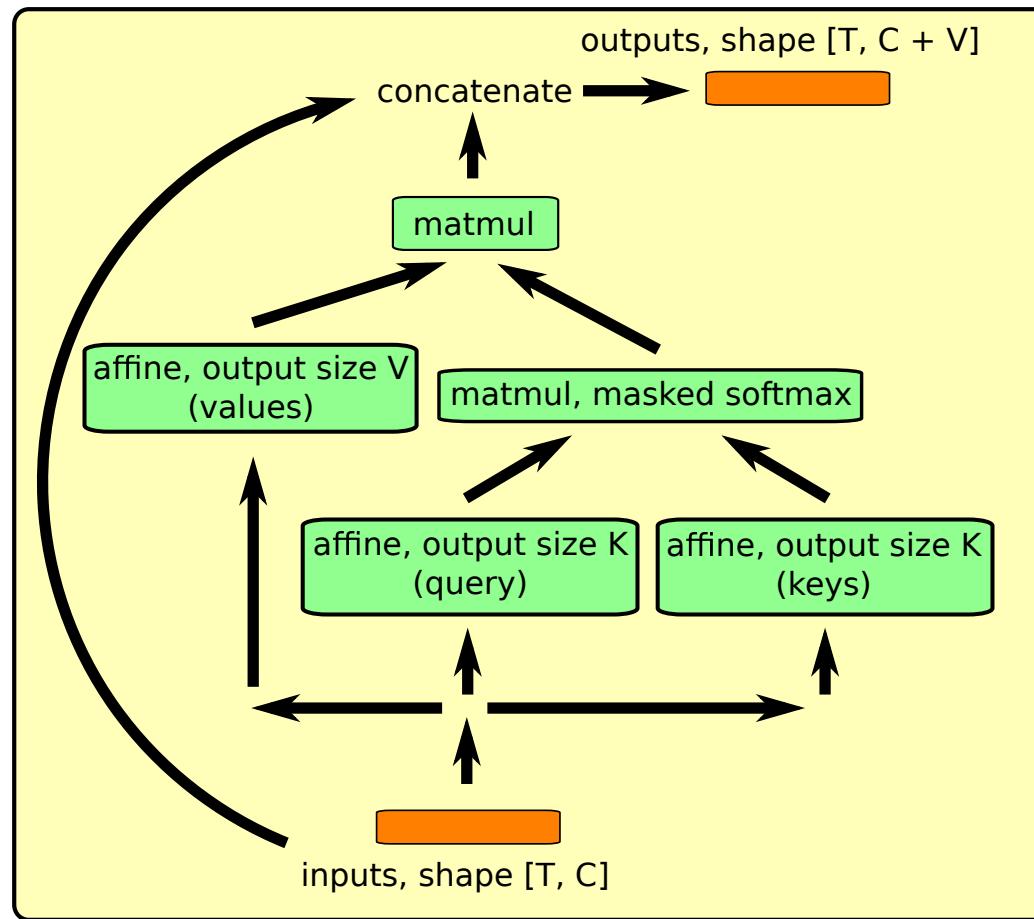
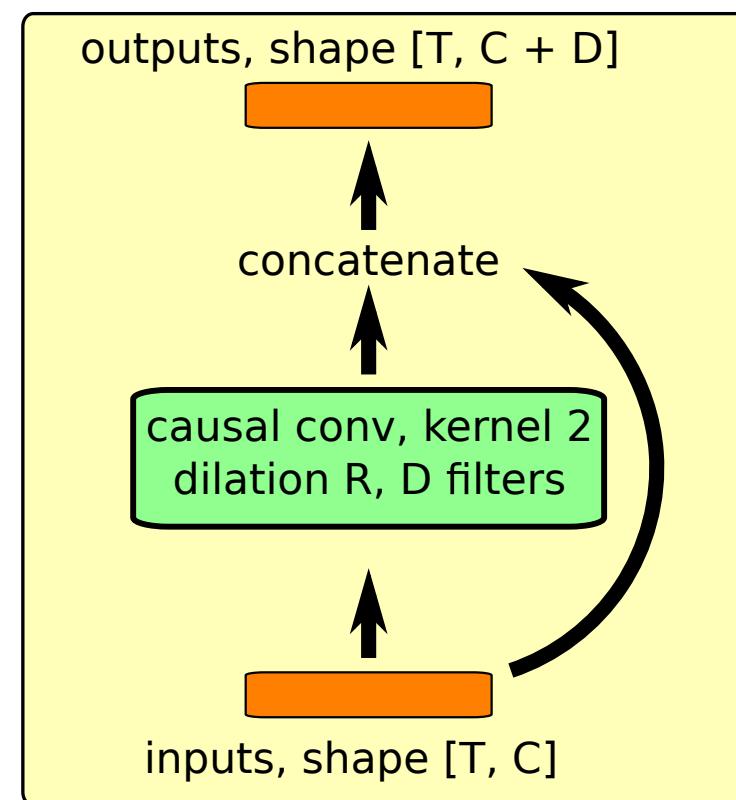


- A Simple Neural Attentive Meta-Learner (2018)
Nikhil Mishra, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel



SIMPLE NEURAL ATTENTIVE LEARNER

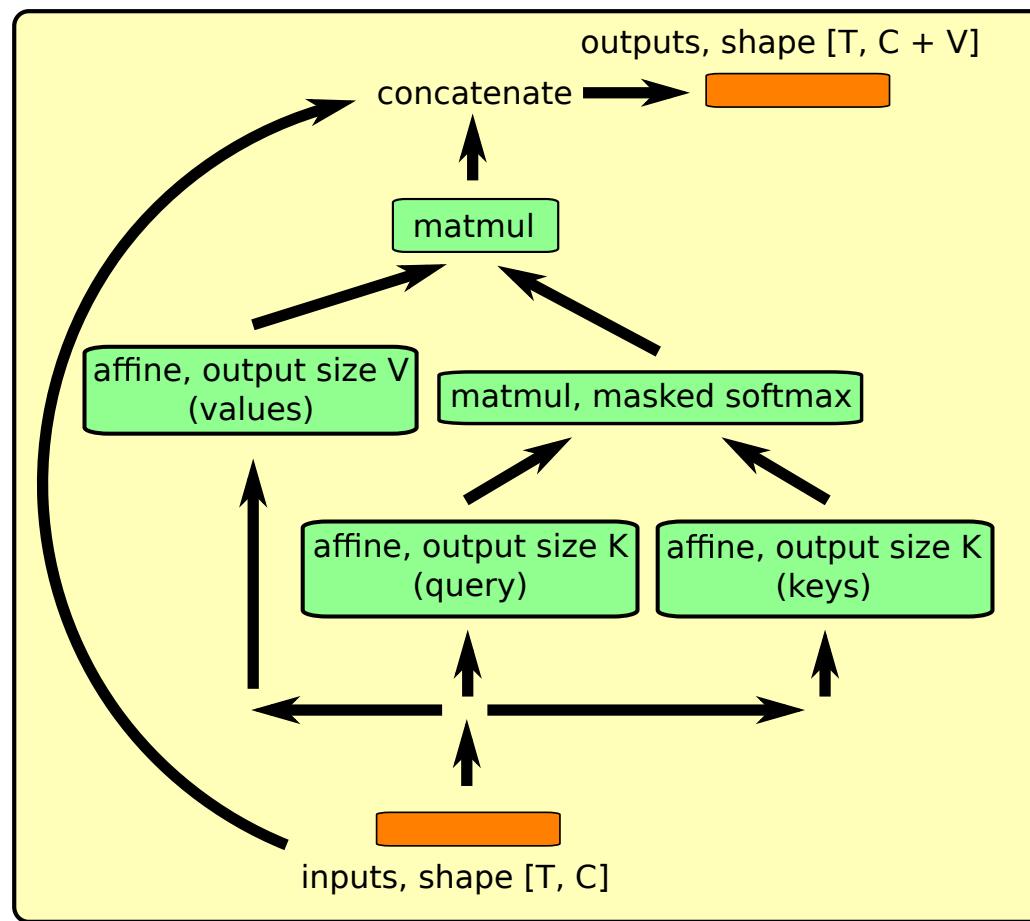
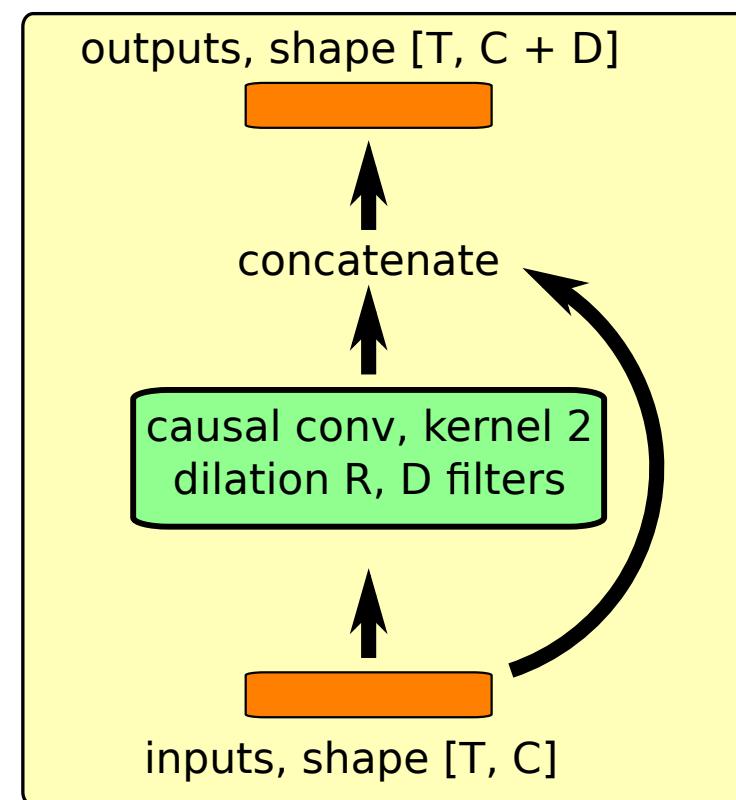
- Training a **convolutional/attentional network** to learn
 - ▶ alternates between **dilated convolutional layers** and **attentional layers**
 - ▶ when inputs are images, an convolutional embedding network is used to map to a vector space



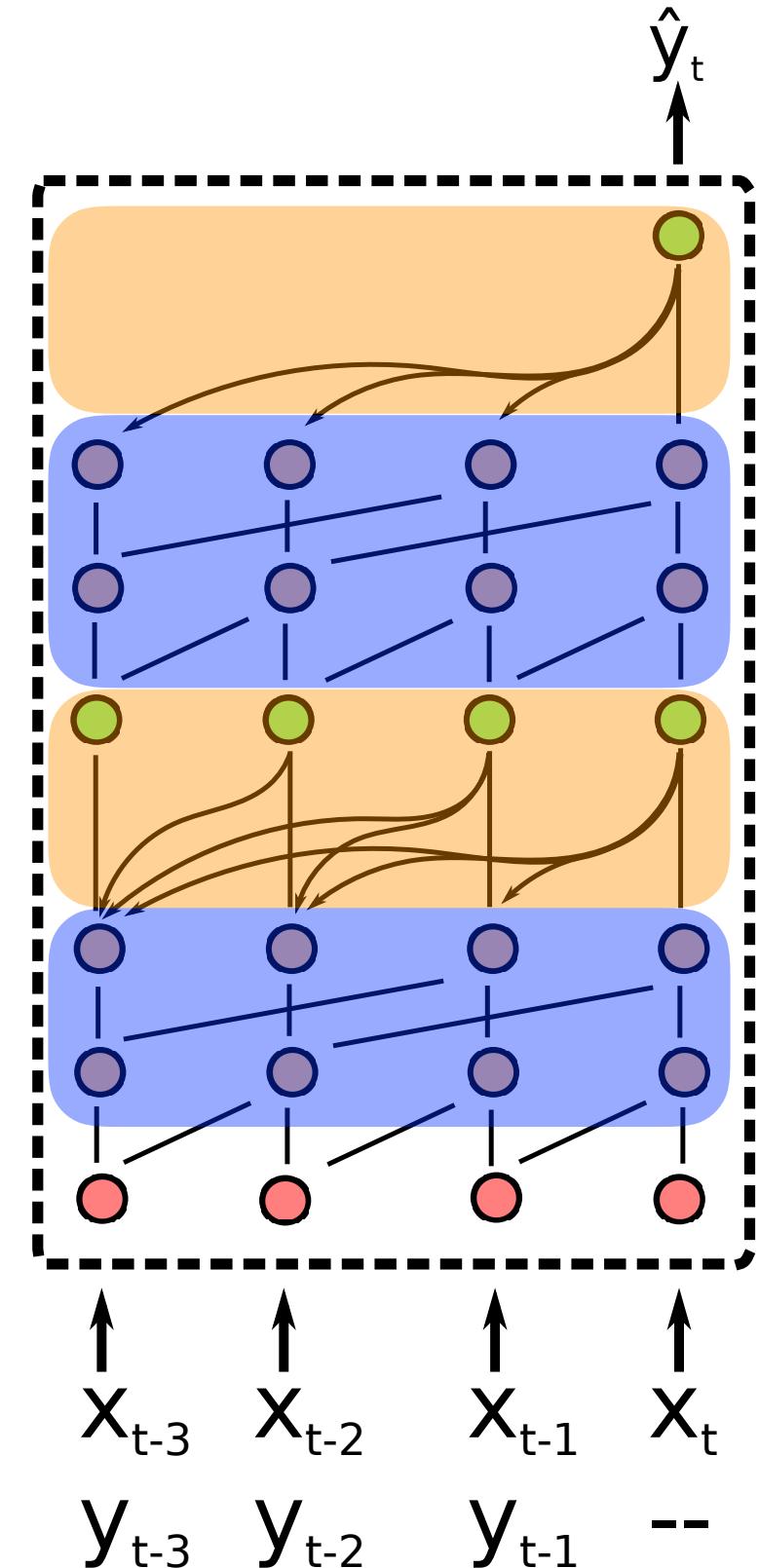
- A Simple Neural Attentive Meta-Learner (2018)
Nikhil Mishra, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel

SIMPLE NEURAL ATTENTIVE LEARNER

- Training a **convolutional/attentional network** to learn
 - ▶ alternates between **dilated convolutional layers** and **attentional layers**
 - ▶ when inputs are images, an convolutional embedding network is used to map to a vector space



- A Simple Neural Attentive Meta-Learner (2018)
Nikhil Mishra, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel



EXPERIMENT

- 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	$43.40 \pm 0.78\%$	$51.09 \pm 0.71\%$
Matching Network FCE	$43.56\% \pm 0.84\%$	$55.31\% \pm 0.73\%$
Meta-Learner LSTM (OURS)	$43.44\% \pm 0.77\%$	$60.60\% \pm 0.71\%$

EXPERIMENT

- 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

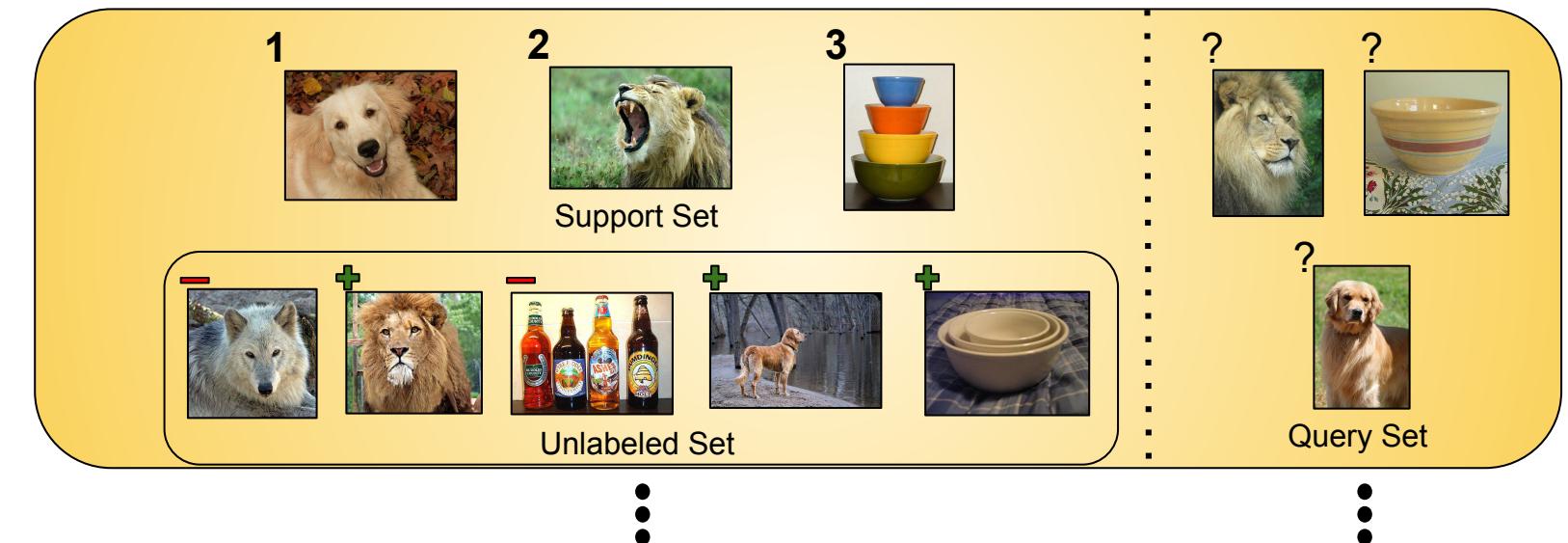
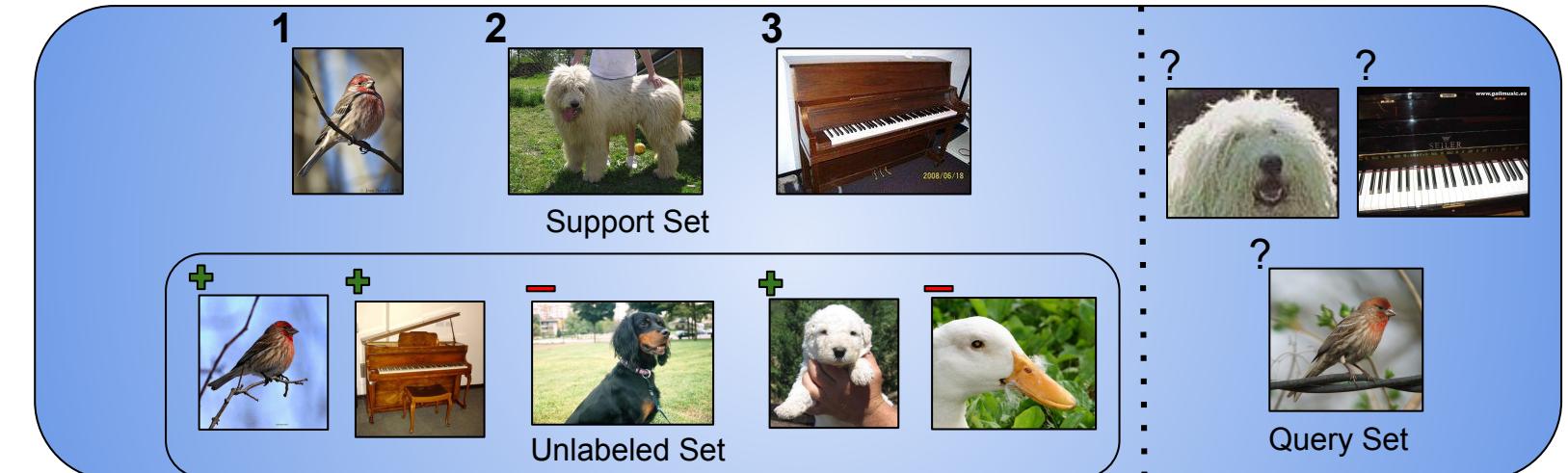
Model	5-class	
	1-shot	5-shot
Prototypical Nets (Snell et al.)	$49.42\% \pm 0.78\%$	$68.20\% \pm 0.66\%$
MAML (Finn et al.)	$48.70\% \pm 1.84\%$	$63.10\% \pm 0.92\%$
SNAIL (Mishra et al.)	$55.71\% \pm 0.99\%$	$68.88\% \pm 0.98\%$
Matching Network FCE	$43.56\% \pm 0.84\%$	$55.31\% \pm 0.73\%$
Meta-Learner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

EXTENSIONS AND VARIATIONS

- Semi-supervised learning (with distractors)

- ▶ assign soft-labels to unlabeled examples
- ▶ use soft-labels to refine prototypes

$$p_c = \frac{\sum_i h(\mathbf{x}_i) z_{i,c}}{\sum_i z_{i,c}}$$



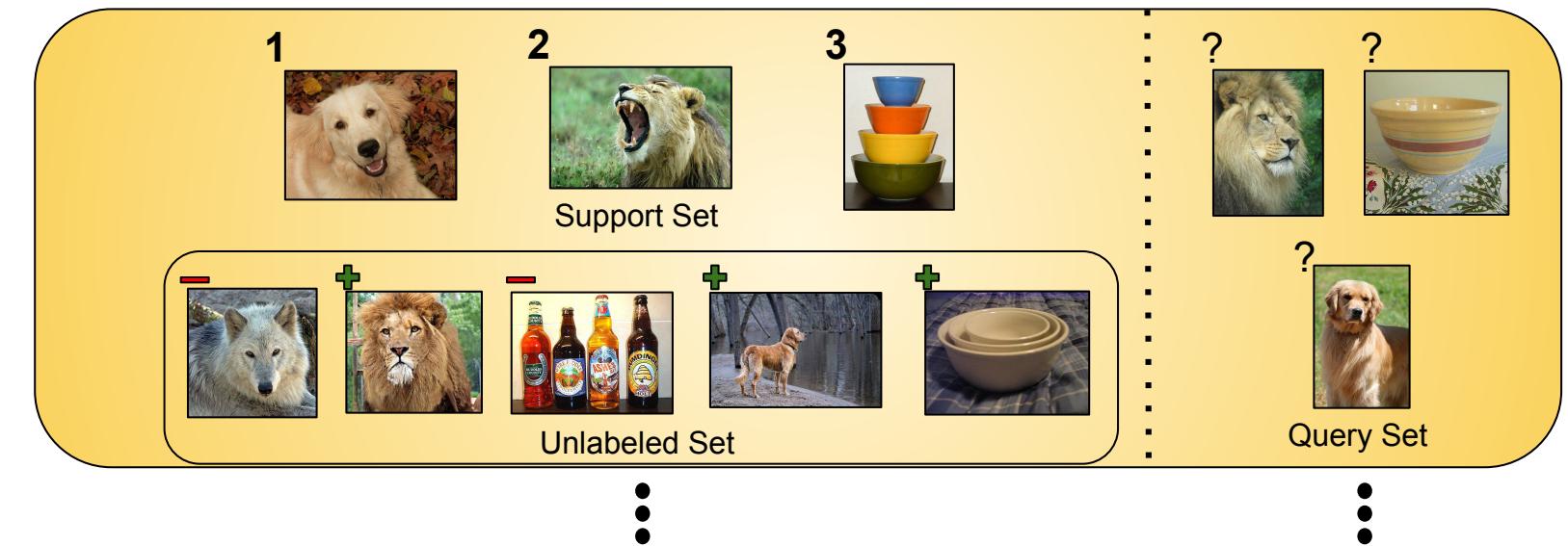
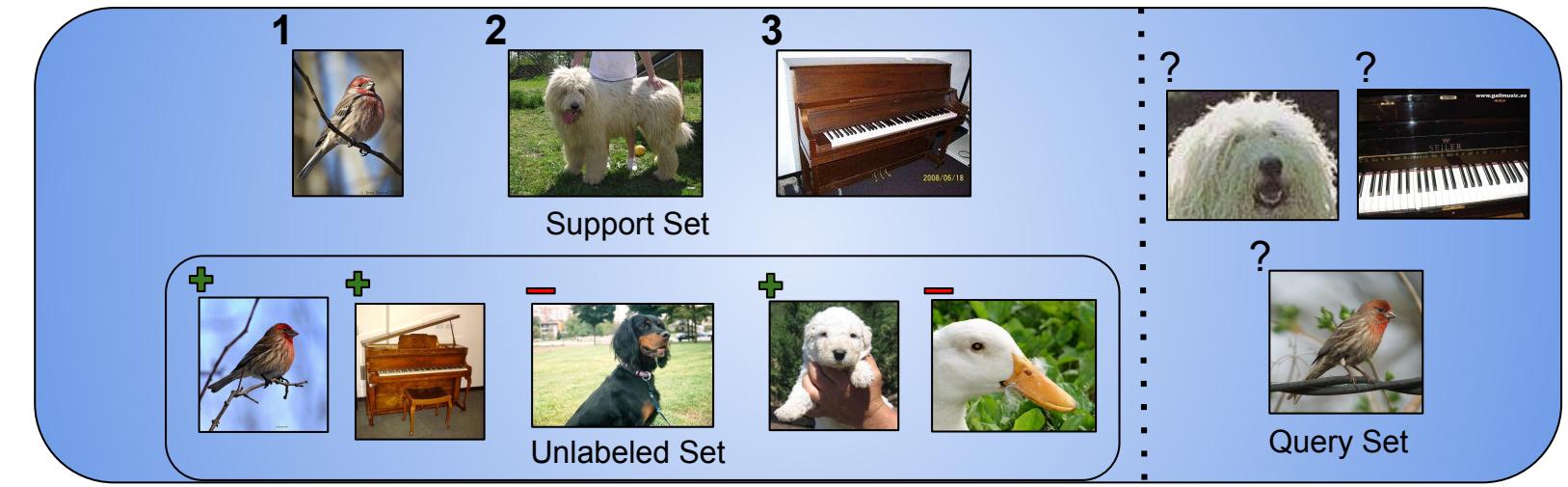
- Meta-Learning for Semi-supervised Few-Shot Classification (2018)
Ren, Triantafillou, Ravi, Snell, Swersky, Tenenbaum, Larochelle and Zemel

EXTENSIONS AND VARIATIONS

- Semi-supervised learning (with distractors)

- ▶ assign soft-labels to unlabeled examples
- ▶ use soft-labels to refine prototypes

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



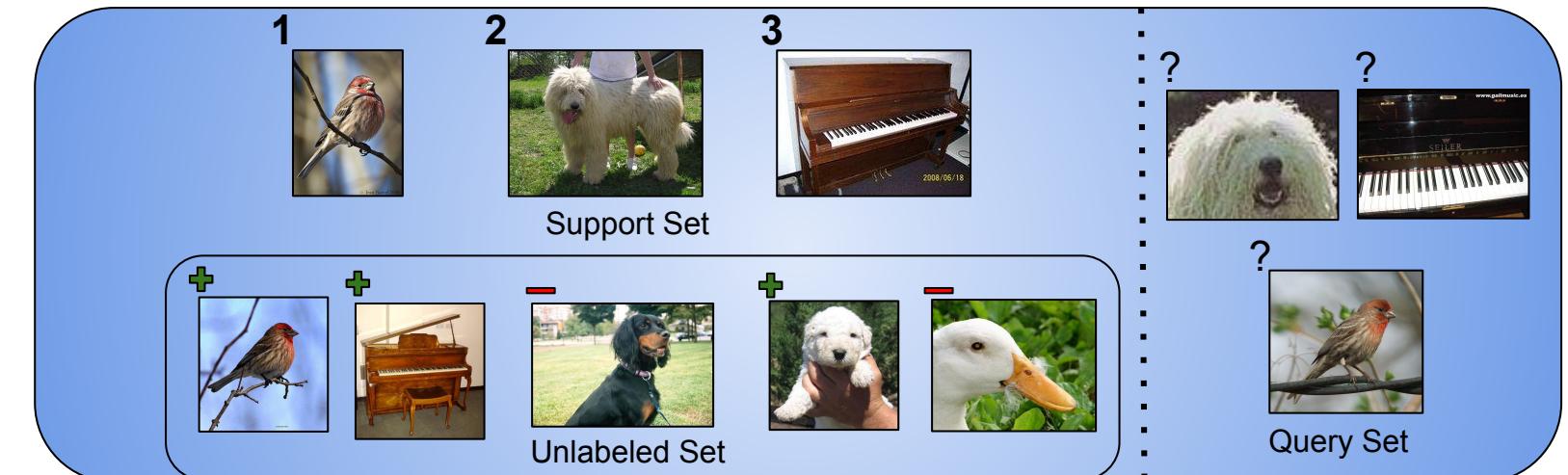
- Meta-Learning for Semi-supervised Few-Shot Classification (2018)
Ren, Triantafillou, Ravi, Snell, Swersky, Tenenbaum, Larochelle and Zemel

EXTENSIONS AND VARIATIONS

- Semi-supervised learning (with distractors)

- ▶ assign soft-labels to unlabeled examples
- ▶ use soft-labels to refine prototypes

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

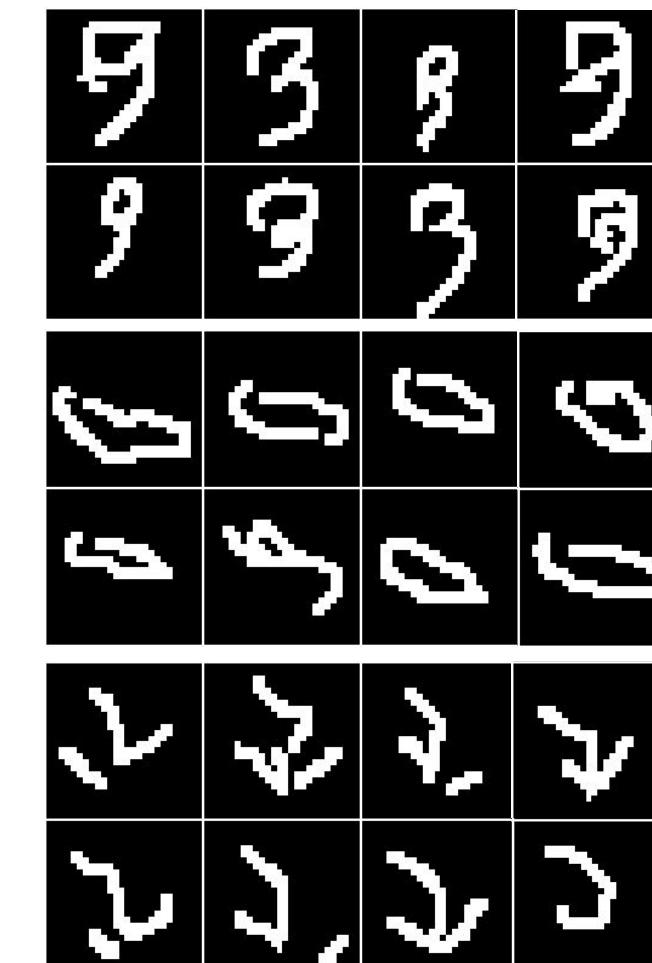
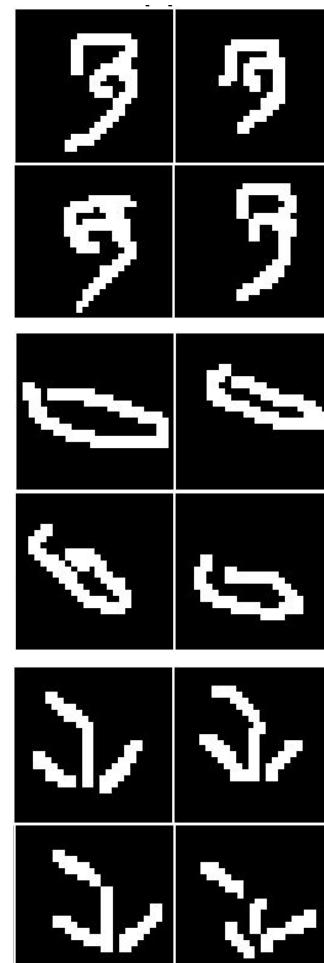


- Meta-Learning for Semi-supervised Few-Shot Classification (2018)
Ren, Triantafillou, Ravi, Snell, Swersky, Tenenbaum, Larochelle and Zemel

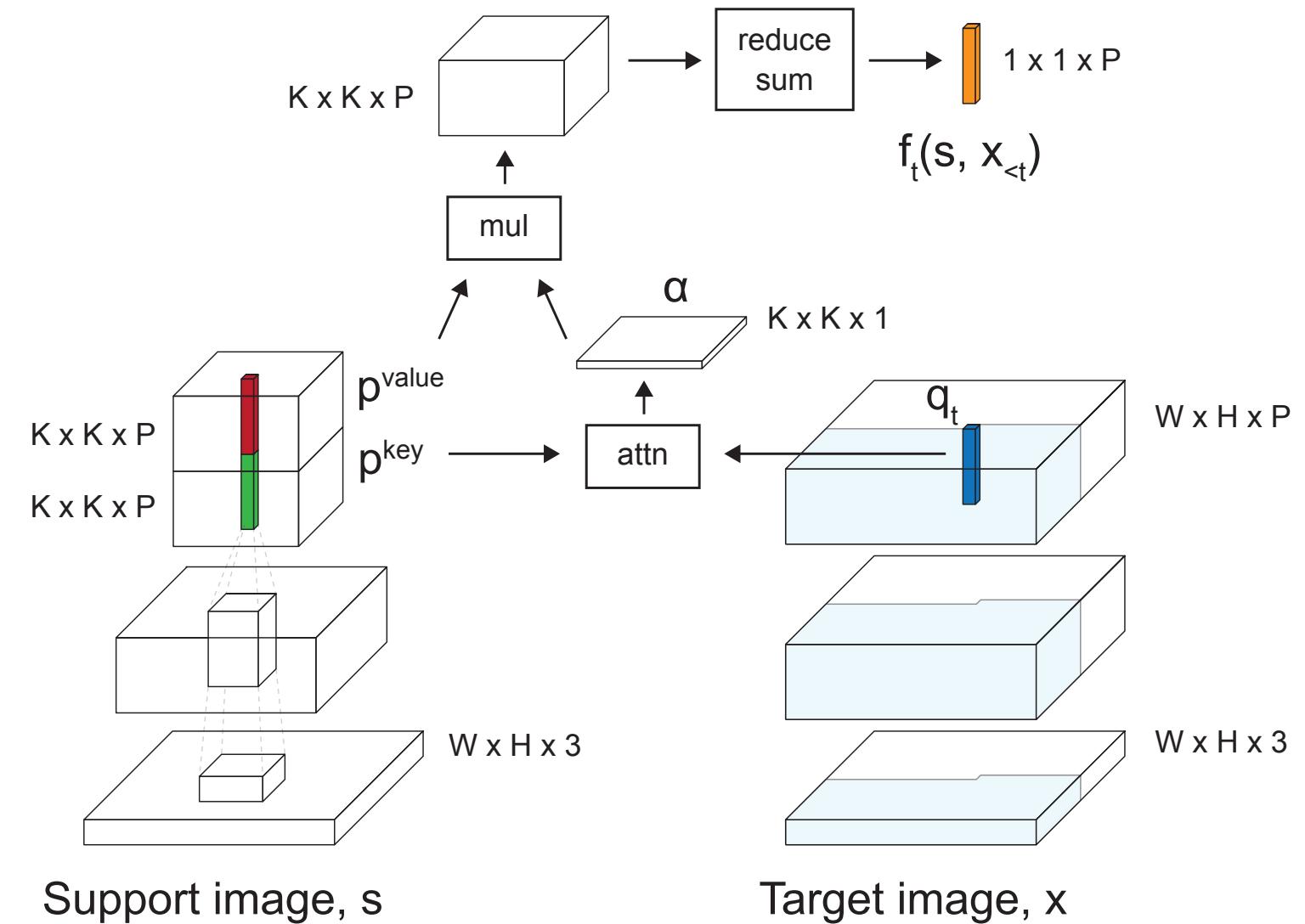
EXTENSIONS AND VARIATIONS

- Few-shot distribution estimation

- given $D_{train} = \{\mathbf{x}_i\}$ produce $p(\mathbf{x})$



Attention PixelCNN Mechanism



- Few-shot Autoregressive Density Estimation: Towards learning to learn distributions (2018)
Reed, Chen, Paine, van den Oord, Eslami, Rezende, Vinyals, de Freitas

FEW-SHOT DISTRIBUTION LEARNING FOR MUSIC GENERATION

Hugo Larochelle, Chelsea Finn, Sachin Ravi

Learn a generative model in the few-shot learning regime to generate MIDI sequences or lyrics.

Start date: November 2017

Category: Fundamental Research

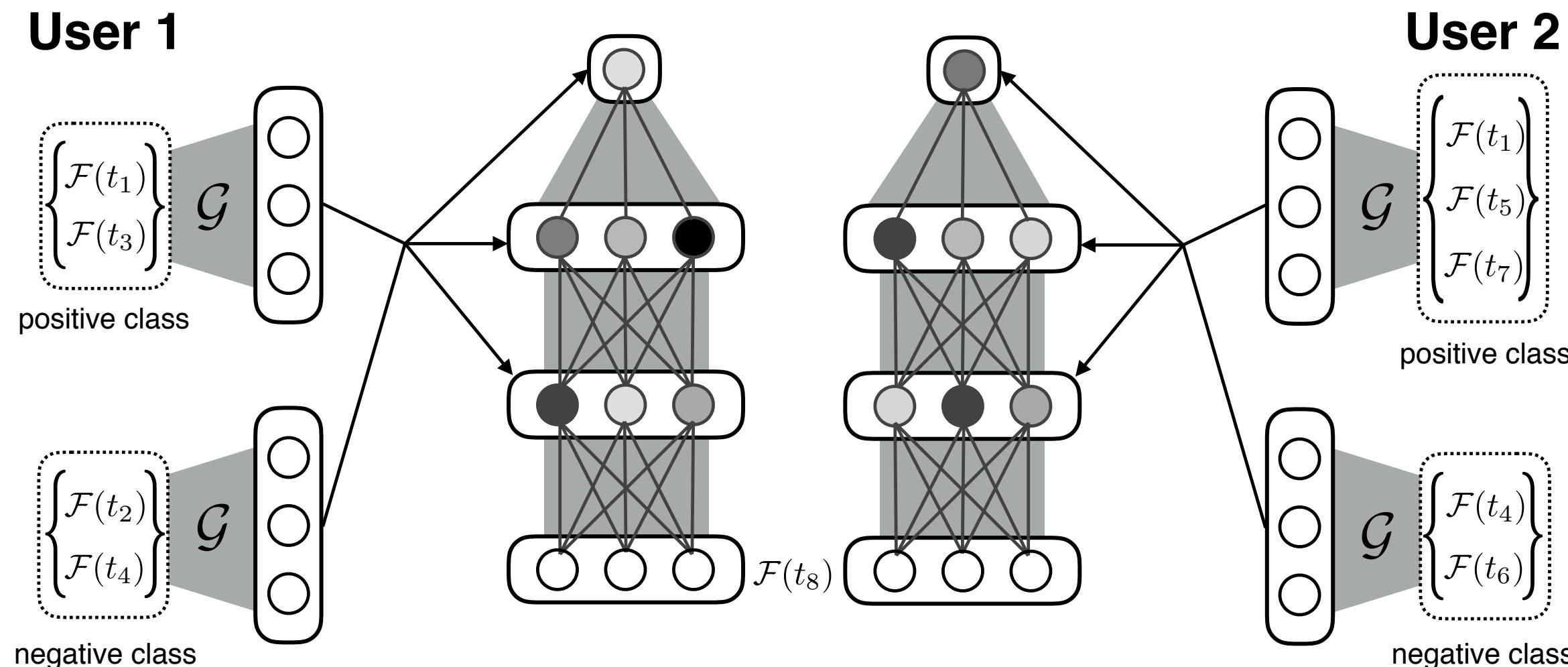
Contact point: Hugo Larochelle (hugolarochelle@google.com), Chelsea Finn (cbfinn@eecs.berkeley.edu), Sachin Ravi (sachinr@princeton.edu)

Abstract

Few-shot distribution learning refers to the problem of learning a generative model in the few-shot learning regime. We propose to investigate this problem in the context of generating music data, such as lyrics or MIDI sequences, using ideas from recent developments in adaptive language models, few-shot learning and meta-learning. We plan to collect and construct benchmarks for this problem and evaluate various solutions.

EXTENSIONS AND VARIATIONS

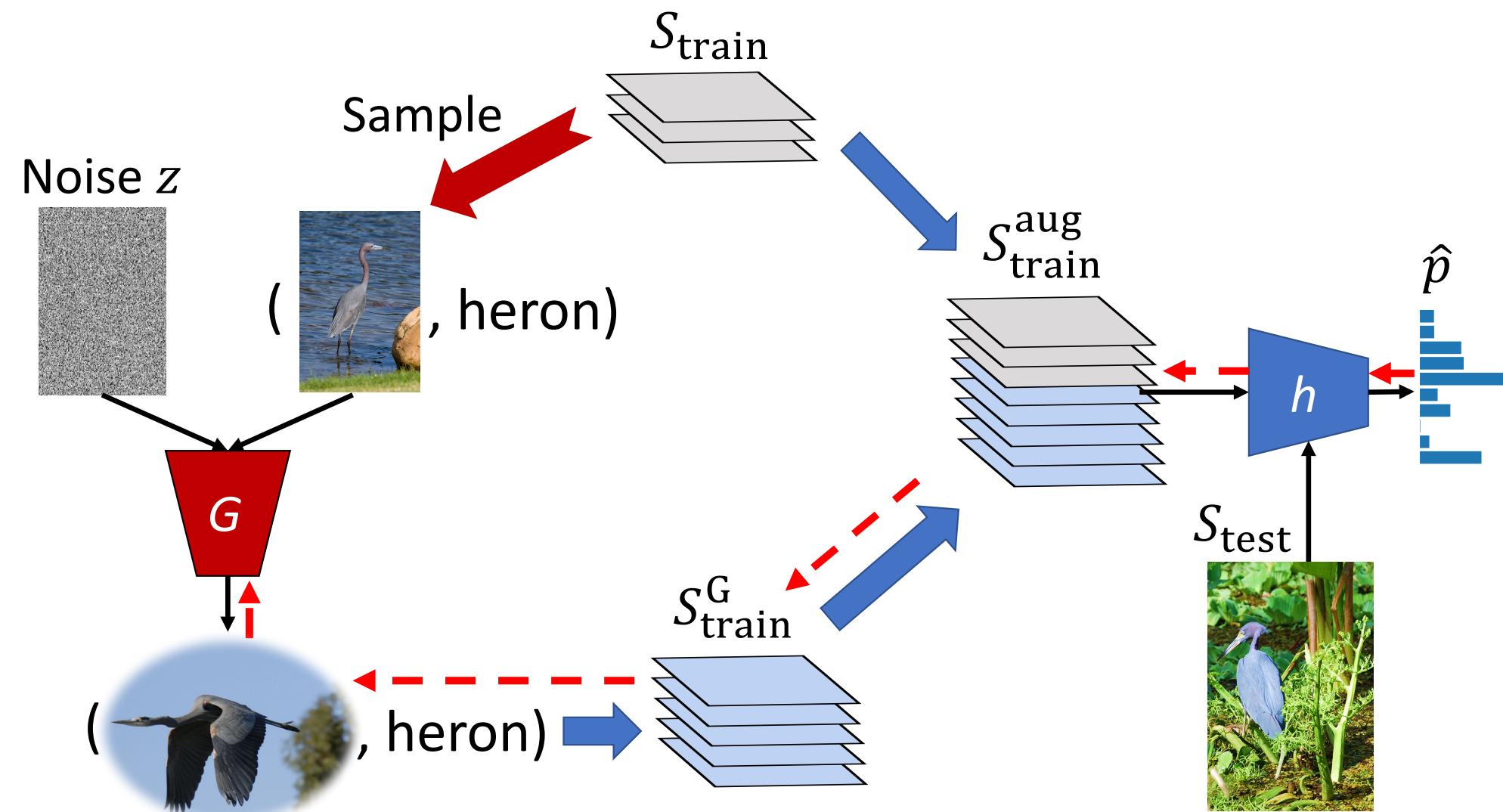
- Cold-start item recommendation
 - ▶ given positive/negative items for a user, produce bias parameters of engagement predictor for new item



- A Meta-Learning Perspective on Cold-Start Recommendations for Items (2017)
Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, Hugo Larochelle

EXTENSIONS AND VARIATIONS

- Learning a data-augmentation network
 - ▶ add to D_{train} examples produced by a generator network that also takes a random \mathbf{z} vector as input

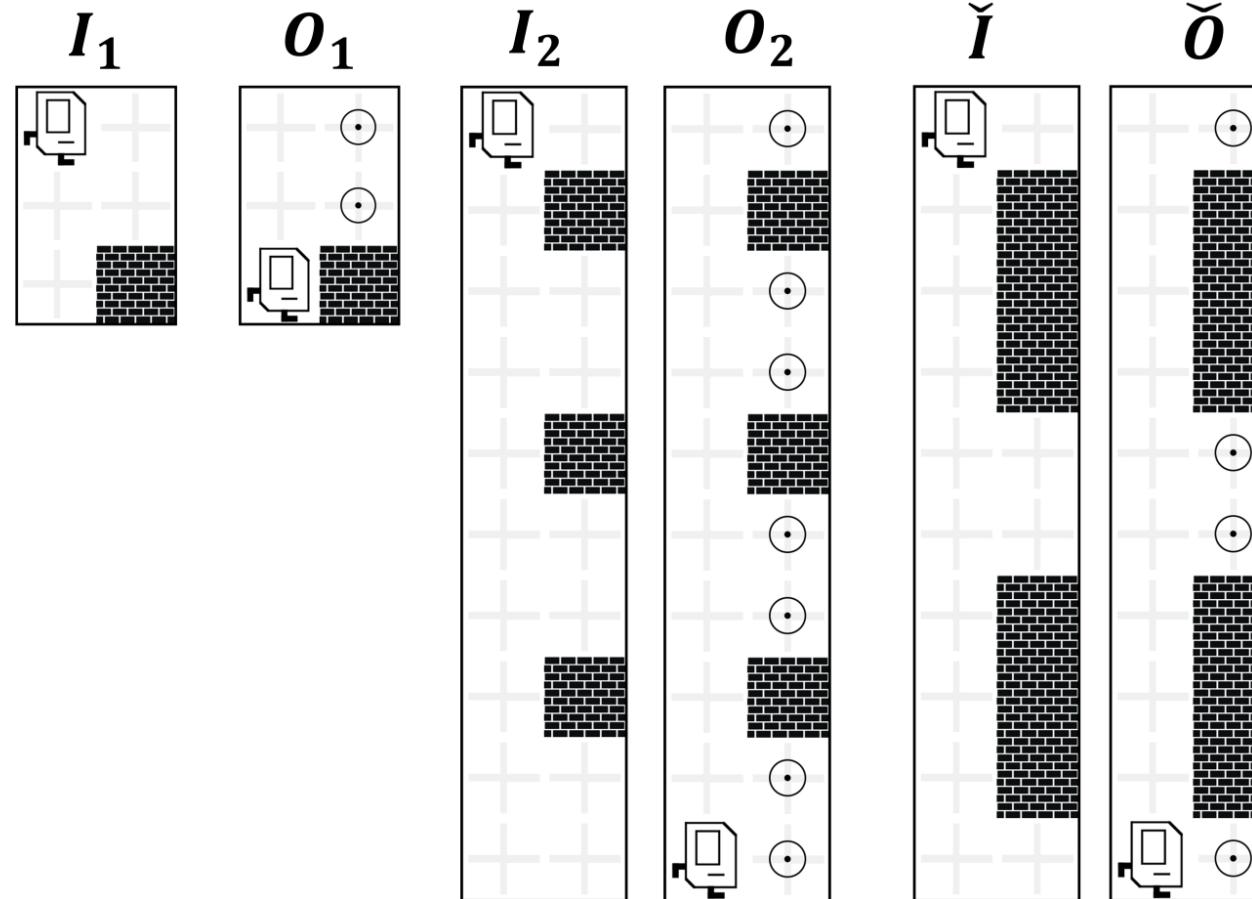


- Low-Shot Learning from Imaginary Data (2018)
Yu-Xiong Wang, Ross Girshick, Martial Hebert, Bharath Hariharan

EXTENSIONS AND VARIATIONS

- Few-shot program induction
 - ▶ learn to produce a neural network that implements a particular program, from demonstrations

Karel Sample Task



Underlying Program (Not used by model)

```
def run():
    if rightIsClear():
        turnRight()
        move()
        putMarker()
        turnLeft()
        turnLeft()
        move()
        turnRight()
    while frontIsClear():
        move()
        if rightIsClear():
            turnRight()
            move()
            putMarker()
            turnLeft()
            turnLeft()
            move()
            turnRight()
```

Karel DSL

Function:
def run() :
 block

Conditional:
if (condition) :
 block
if (condition) :
 block
else :
 block

Loops:
for i in range(count) :
 body
while (condition) :
 body
while (not condition) :
 body

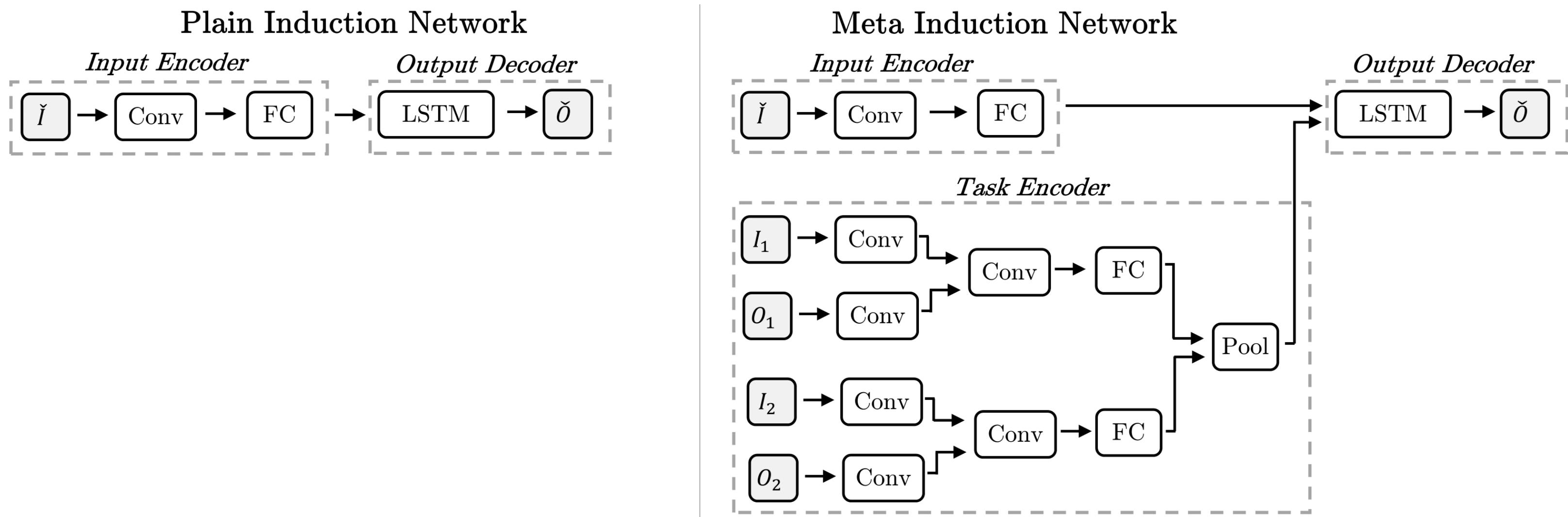
Actions:
move()
turnLeft()
turnRight()
putMarker()
pickMarker()

Conditions:
frontIsClear()
leftIsClear()
rightIsClear()
markerPresent()

- Neural Program Meta-Induction (2017)
Jacob Devlin, Rudy Bunel, Rishabh Singh, Matthew Hausknecht, Pushmeet Kohli

EXTENSIONS AND VARIATIONS

- Few-shot program induction
 - ▶ learn to produce a neural network that implements a particular program, from demonstrations

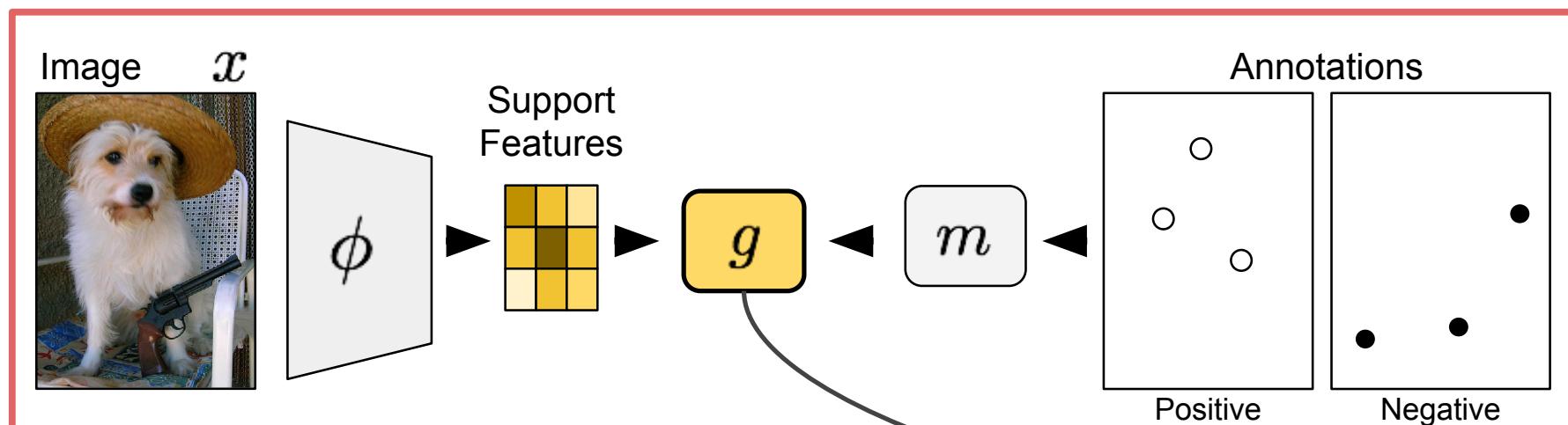


- Neural Program Meta-Induction (2017)
Jacob Devlin, Rudy Bunel, Rishabh Singh, Matthew Hausknecht, Pushmeet Kohli

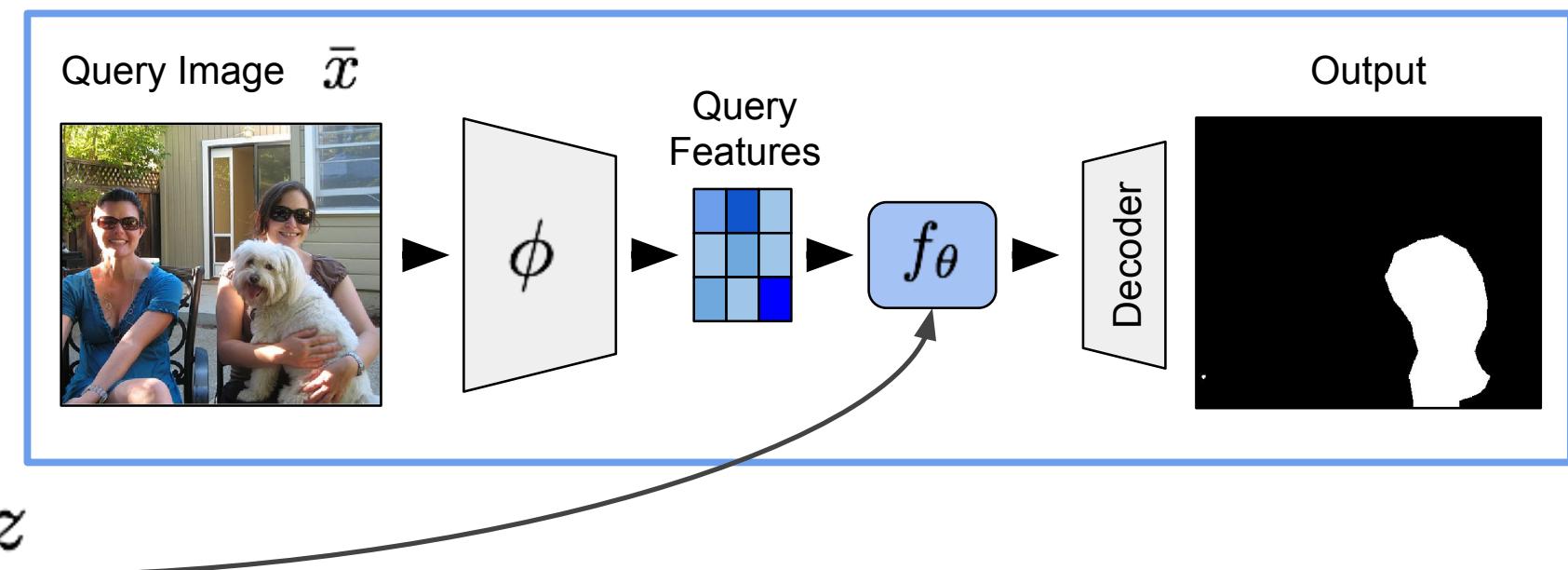
EXTENSIONS AND VARIATIONS

- Few-shot image segmentation
 - ▶ learn to produce the segmentation of a new image based on (sparsely) segmented data

Support Task Representation



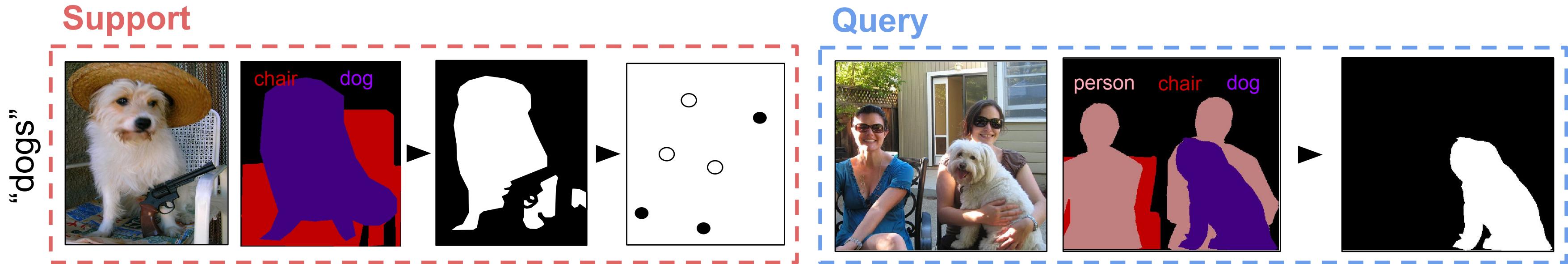
Guided Inference



- Few-Shot Segmentation Propagation with Guided Networks (2018)
Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alexei Efros and Sergey Levine

EXTENSIONS AND VARIATIONS

- Few-shot image segmentation
 - ▶ learn to produce the segmentation of a new image based on (sparsely) segmented data



- Few-Shot Segmentation Propagation with Guided Networks (2018)
Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alexei Efros and Sergey Levine

DISCUSSION

- 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)

MERCI !