

# Learning to Learn from small amounts of data

Chelsea Finn



BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH



**This class:**

**This class:**

**1. Get large dataset**

**This class:**

- 1. Get large dataset**
- 2. Get large compute (i.e. GPUs)**

**This class:**

- 1. Get large dataset**
- 2. Get large compute (i.e. GPUs)**
- 3. ?????**

**This class:**

- 1. Get large dataset**
- 2. Get large compute (i.e. GPUs)**
- 3. ?????**
- 4. Profit!**

**This class:**

- 1. Get large dataset**
- 2. Get large compute (i.e. GPUs)**
- 3. ?????**
- 4. Profit!**

***Do neural networks need a large dataset?***

**This class:**

- 1. Get large dataset**
- 2. Get large compute (i.e. GPUs)**
- 3. ?????**
- 4. Profit!**

***Do neural networks need a large dataset?***

Humans can learn with a very small amount of data. How?

**This class:**

- 1. Get large dataset**
- 2. Get large compute (i.e. GPUs)**
- 3. ?????**
- 4. Profit!**

***Do neural networks need a large dataset?***

Humans can learn with a very small amount of data. How?

**Humans don't start from scratch.**

**This class:**

- 1. Get large dataset**
- 2. Get large compute (i.e. GPUs)**
- 3. ?????**
- 4. Profit!**

***Do neural networks need a large dataset?***

Humans can learn with a very small amount of data. How?

**Humans don't start from scratch.**

**Humans accumulate prior experience, acquire common sense**



# **How can we move away from training from scratch?**

**How can we move away from training from scratch?  
Can we develop systems that accumulate prior experience?**

**How can we move away from training from scratch?  
Can we develop systems that accumulate prior experience?**

**Idea: learn the structure underlying previous  
data/tasks/experiences; use that structure to quickly  
learn new tasks.**

**How can we move away from training from scratch?  
Can we develop systems that accumulate prior experience?**

**Idea: learn the structure underlying previous  
data/tasks/experiences; use that structure to quickly  
learn new tasks.**

**How? One way to do so is by learning to learn.**

**How can we move away from training from scratch?  
Can we develop systems that accumulate prior experience?**

**Idea: learn the structure underlying previous  
data/tasks/experiences; use that structure to quickly  
learn new tasks.**

**How? One way to do so is by learning to learn.**

**Disclaimer 1:** meta-learning is still in its infancy

**How can we move away from training from scratch?  
Can we develop systems that accumulate prior experience?**

**Idea: learn the structure underlying previous  
data/tasks/experiences; use that structure to quickly  
learn new tasks.**

**How? One way to do so is by learning to learn.**

**Disclaimer 1:** meta-learning is still in its infancy  
(though it dates back to the 80/90s in the context of AI).

**How can we move away from training from scratch?  
Can we develop systems that accumulate prior experience?**

**Idea: learn the structure underlying previous  
data/tasks/experiences; use that structure to quickly  
learn new tasks.**

**How? One way to do so is by learning to learn.**

**Disclaimer 1:** meta-learning is still in its infancy  
(though it dates back to the 80/90s in the context of AI).

**Disclaimer 2:** Could teach an entire course on meta-learning

# Terminology

# Terminology

**Learning to learn, meta-learning:** overarching terms

# Terminology

**Learning to learn, meta-learning:** overarching terms

**AutoML:** automating the learning process

# Terminology

**Learning to learn, meta-learning:** overarching terms

**AutoML:** automating the learning process

- hyperparameter optimization

# Terminology

**Learning to learn, meta-learning:** overarching terms

**AutoML:** automating the learning process

- hyperparameter optimization
- architecture search

# Terminology

**Learning to learn, meta-learning:** overarching terms

**AutoML:** automating the learning process

- hyperparameter optimization
- architecture search

**One-shot/few-shot learning:** learning to learn from small amounts of data

# Terminology

**Learning to learn, meta-learning:** overarching terms

**AutoML:** automating the learning process

- hyperparameter optimization
- architecture search

**One-shot/few-shot learning:** learning to learn from small amounts of data

**Meta-reinforcement learning:** learning a reinforcement learning algorithm

# Terminology

**Learning to learn, meta-learning:** overarching terms

**AutoML:** automating the learning process

- hyperparameter optimization
- architecture search

**One-shot/few-shot learning:** learning to learn from small amounts of data

**Meta-reinforcement learning:** learning a reinforcement learning algorithm

**Meta-X learning:** X=imitation, unsupervised, semi-supervised

# Terminology

**Learning to learn, meta-learning:** overarching terms

**AutoML:** automating the learning process

- hyperparameter optimization
- architecture search

**One-shot/few-shot learning:** learning to learn from small amounts of data

**Meta-reinforcement learning:** learning a reinforcement learning algorithm

**Meta-X learning:** X=imitation, unsupervised, semi-supervised

**This lecture:** focus on few-shot learning

# Outline

# Outline

## 1. Applications of learning to learn

# Outline

1. Applications of learning to learn
2. Problem formulation

# Outline

1. Applications of learning to learn
2. Problem formulation
3. Solution Classes:

# Outline

1. Applications of learning to learn
2. Problem formulation
3. Solution Classes:
  - a) metric-learning approach

# Outline

1. Applications of learning to learn
2. Problem formulation
3. Solution Classes:
  - a) metric-learning approach
  - b) direct black-box approach

# Outline

1. Applications of learning to learn
2. Problem formulation
3. Solution Classes:
  - a) metric-learning approach
  - b) direct black-box approach
  - c) gradient-based approach

# Outline

1. Applications of learning to learn
2. Problem formulation
3. Solution Classes:
  - a) metric-learning approach
  - b) direct black-box approach
  - c) gradient-based approach
4. Open Questions / Problems

# Outline

- 1. Applications of learning to learn**
2. Problem formulation
3. Solution Classes:
  - a) metric-learning approach
  - b) direct black-box approach
  - c) gradient-based approach
4. Open Questions / Problems

# Application: Few-Shot Image Classification

# Application: Few-Shot Image Classification

Given 1 example of 5 classes:



Classify new examples

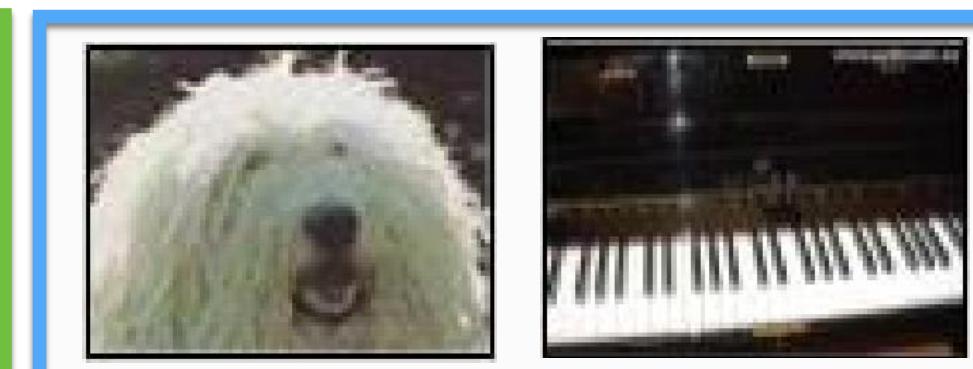
# Application: Few-Shot Image Classification

Given 1 example of 5 classes:



Classify new examples

$\mathcal{T}_1$



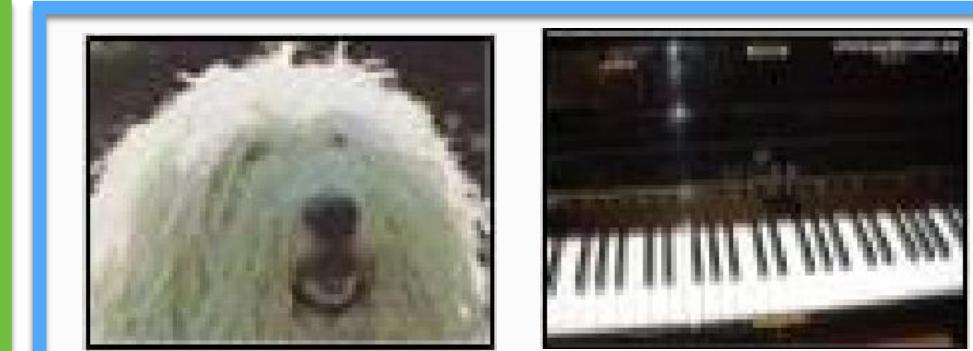
# Application: Few-Shot Image Classification

Given 1 example of 5 classes:

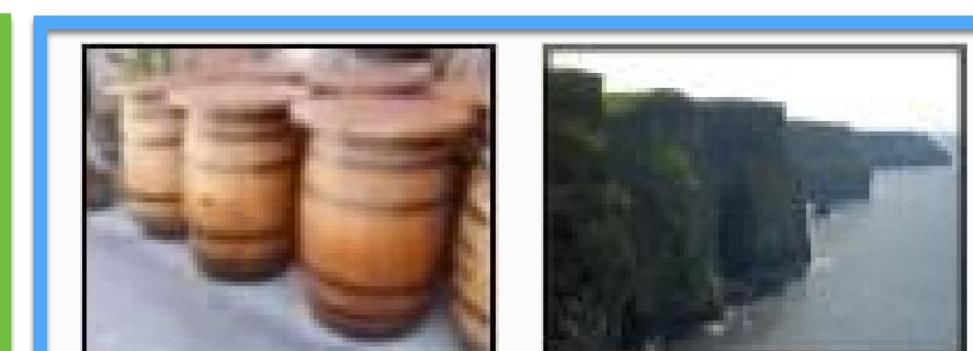


Classify new examples

$\mathcal{T}_1$



$\mathcal{T}_2$



# Application: Few-Shot Image Classification

Given 1 example of 5 classes:

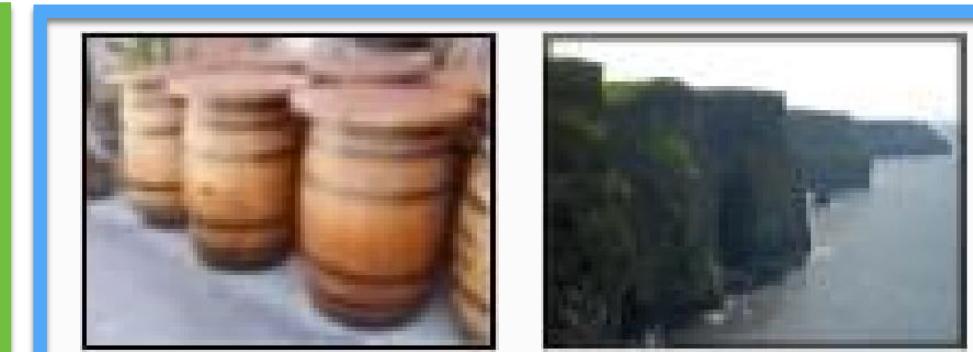


Classify new examples

$\mathcal{T}_1$



$\mathcal{T}_2$



:

:

# Application: Few-Shot Image Classification

Given 1 example of 5 classes:



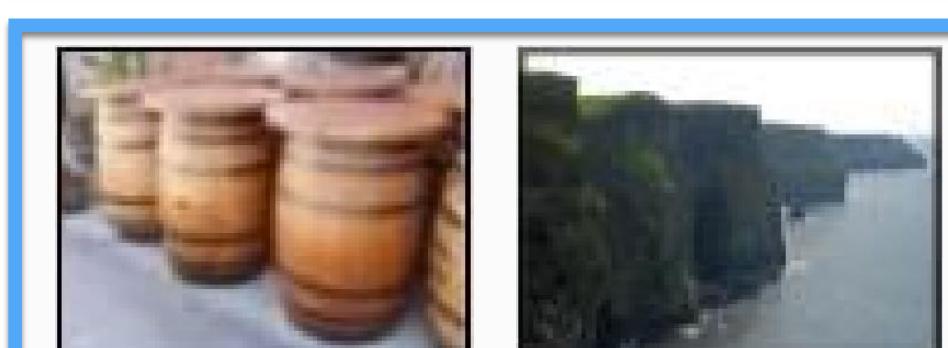
Classify new examples

$\mathcal{T}_1$



training classes

$\mathcal{T}_2$



:

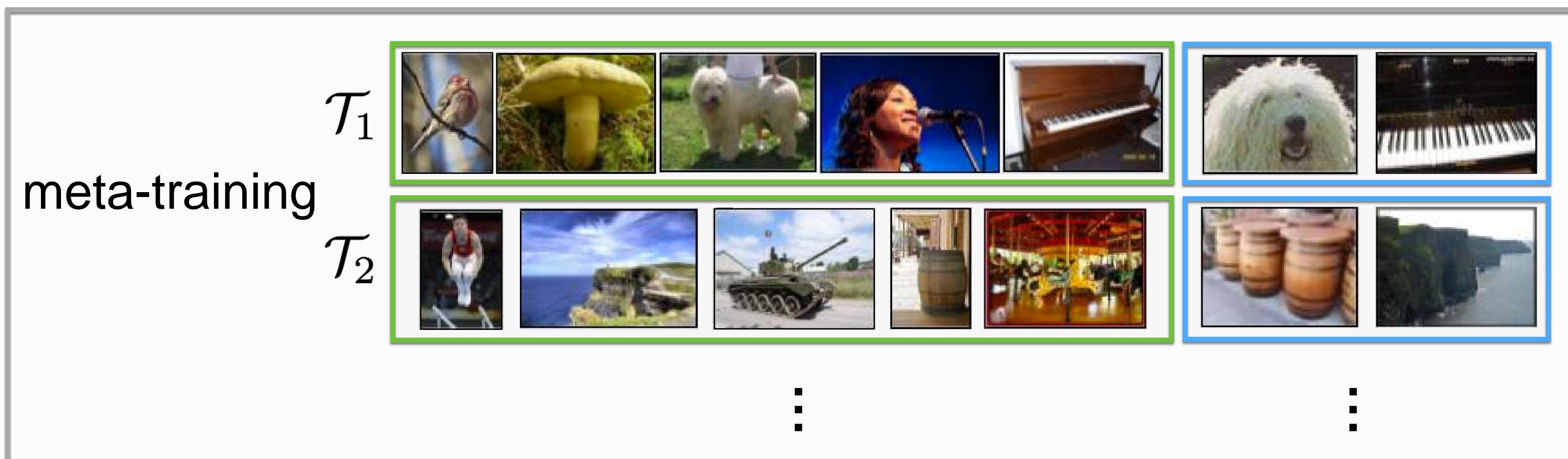
:

# Application: Few-Shot Image Classification

Given 1 example of 5 classes:



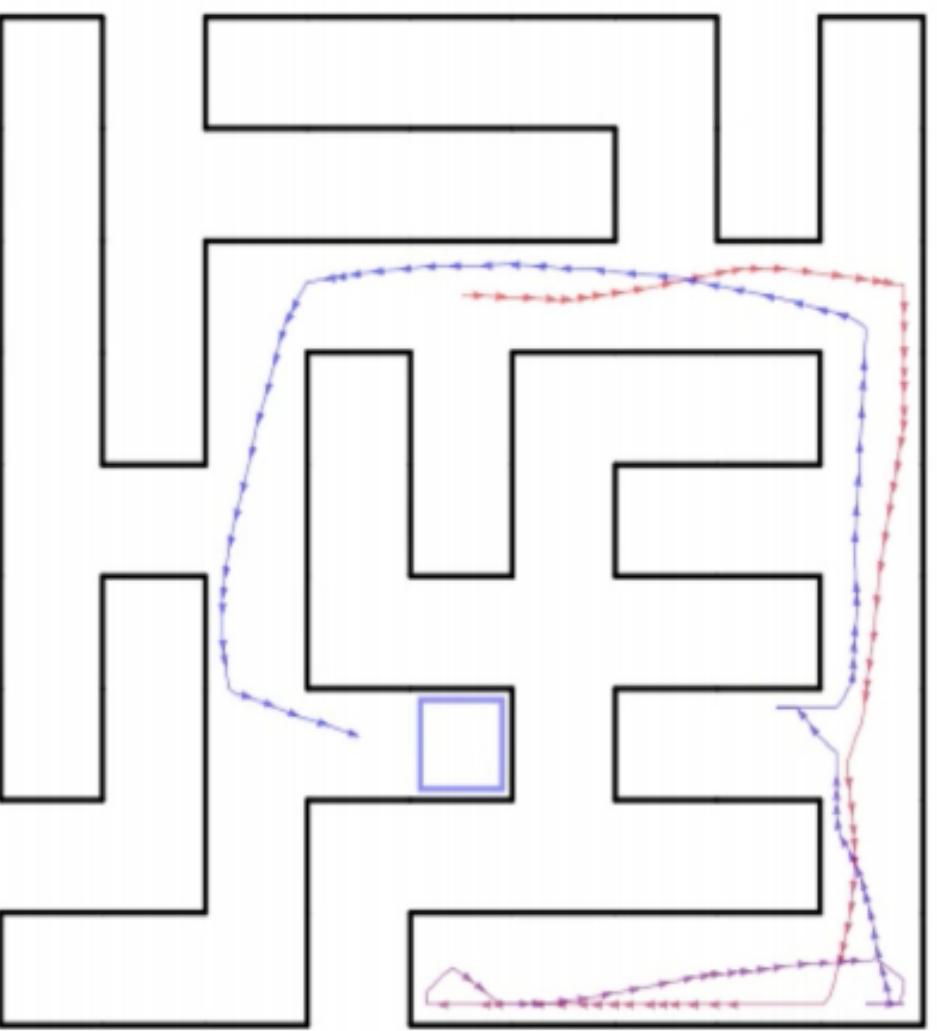
Classify new examples



# Application: Fast Reinforcement Learning

# Application: Fast Reinforcement

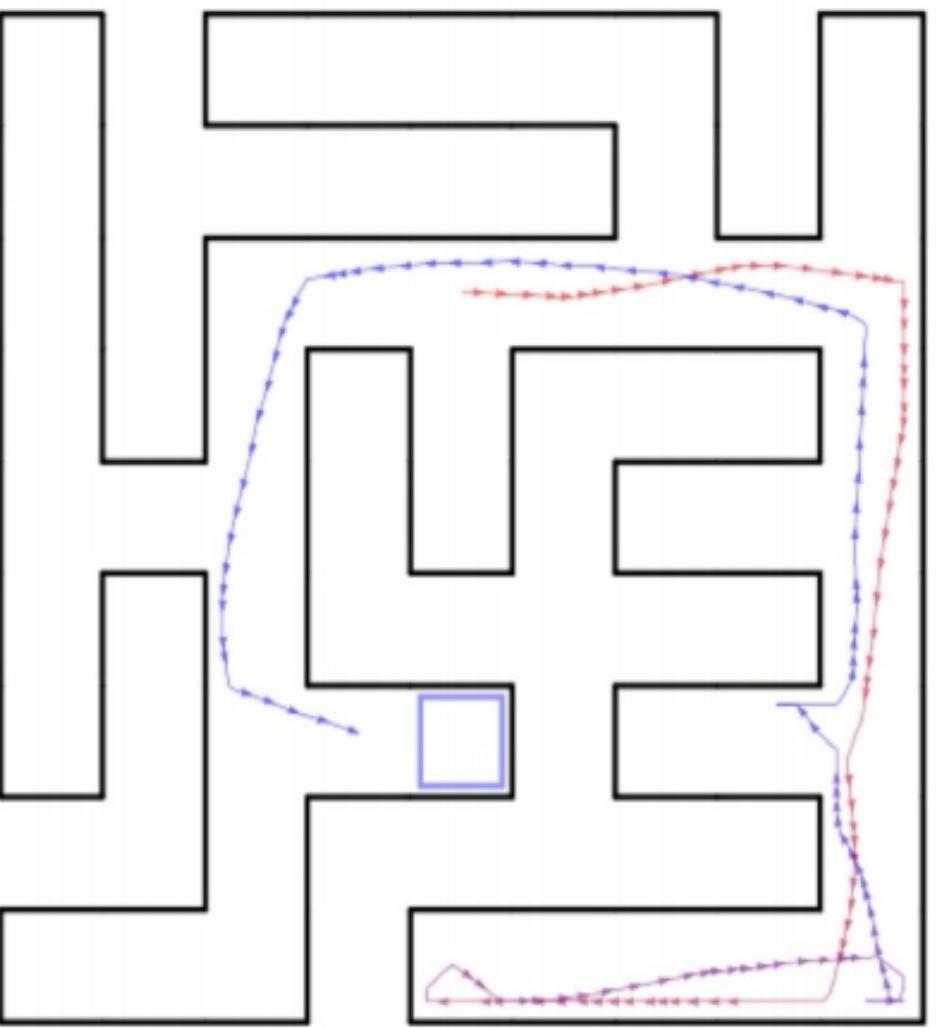
Given a small amount of learning experience



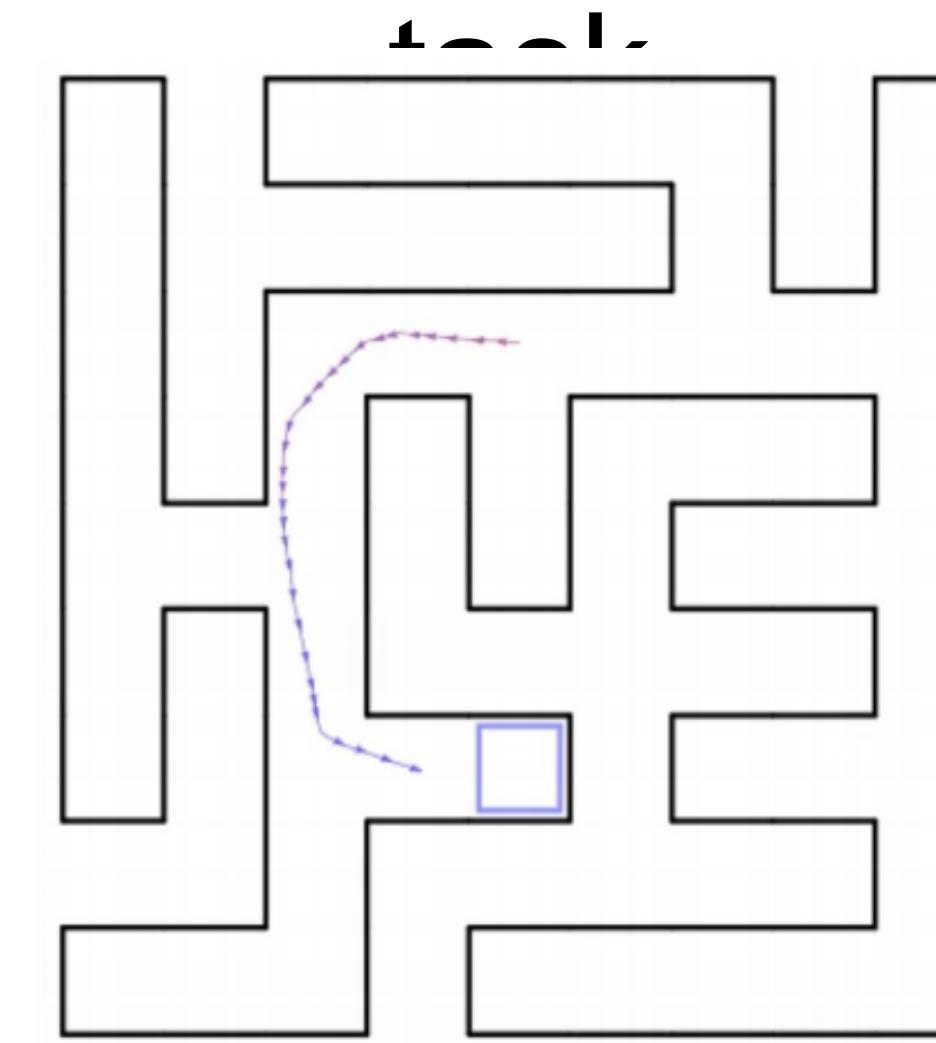
# Application: Fast Reinforcement

Given a small amount of experience

## Learning

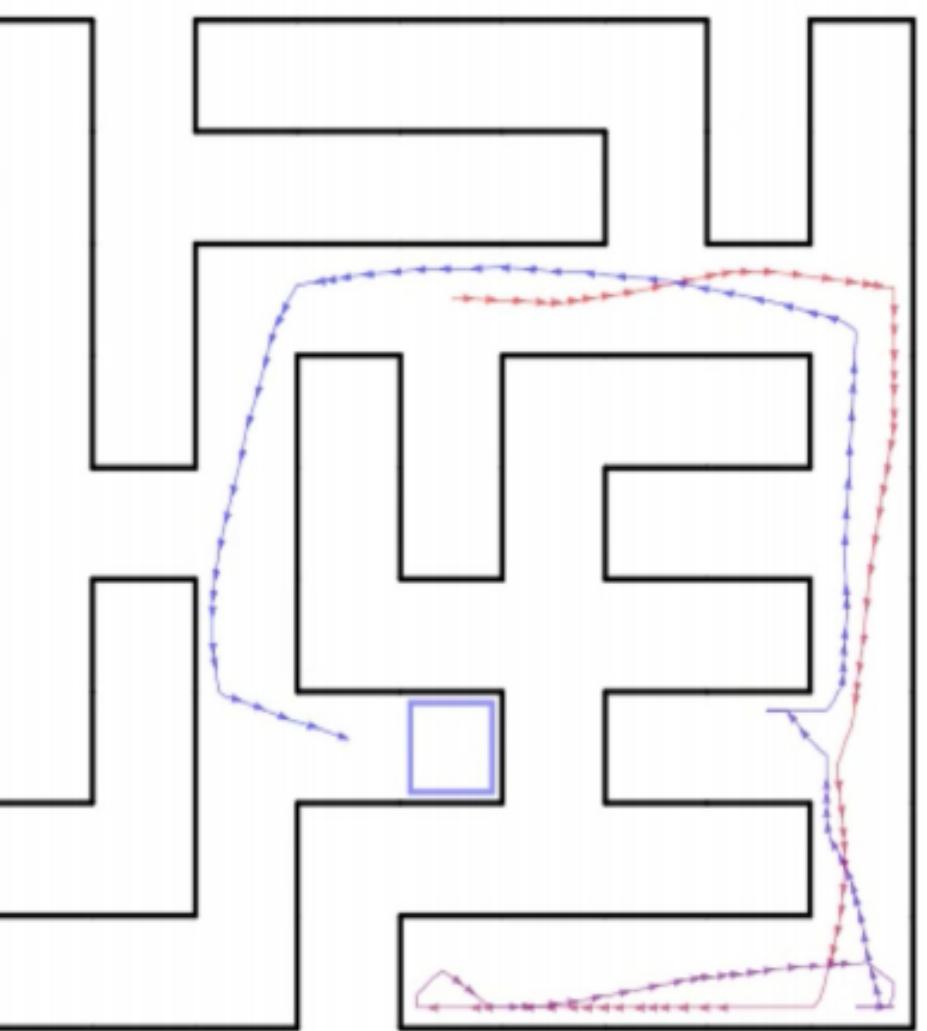


Learn to solve a



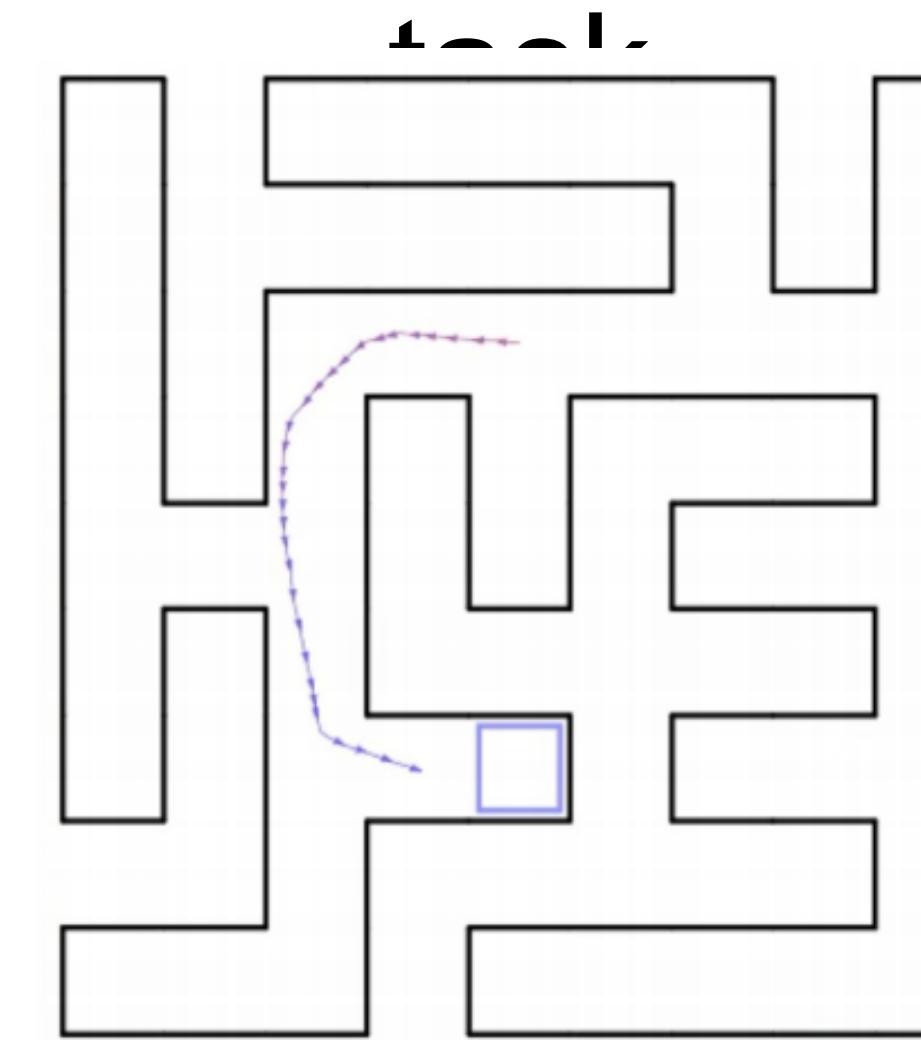
# Application: Fast Reinforcement Learning

Given a small amount of experience



Learning

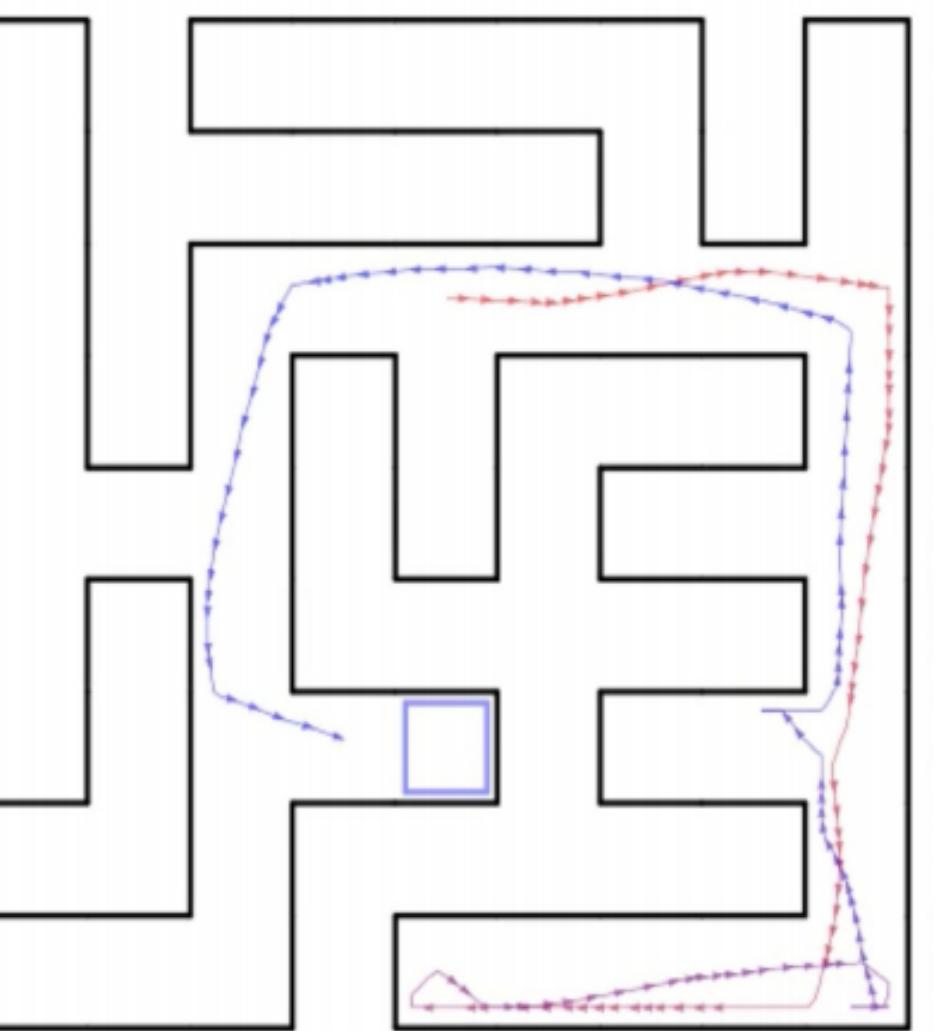
Learn to solve a



By learning how to learn many other tasks:

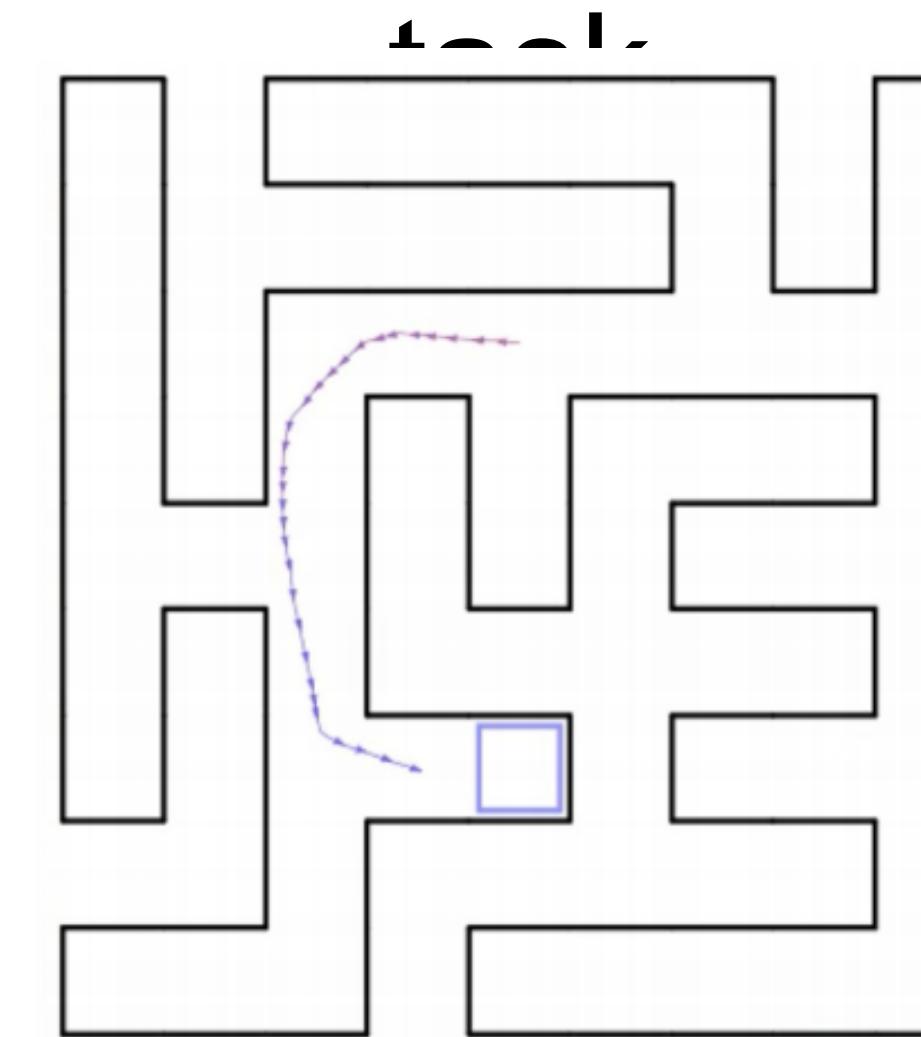
# Application: Fast Reinforcement Learning

Given a small amount of experience

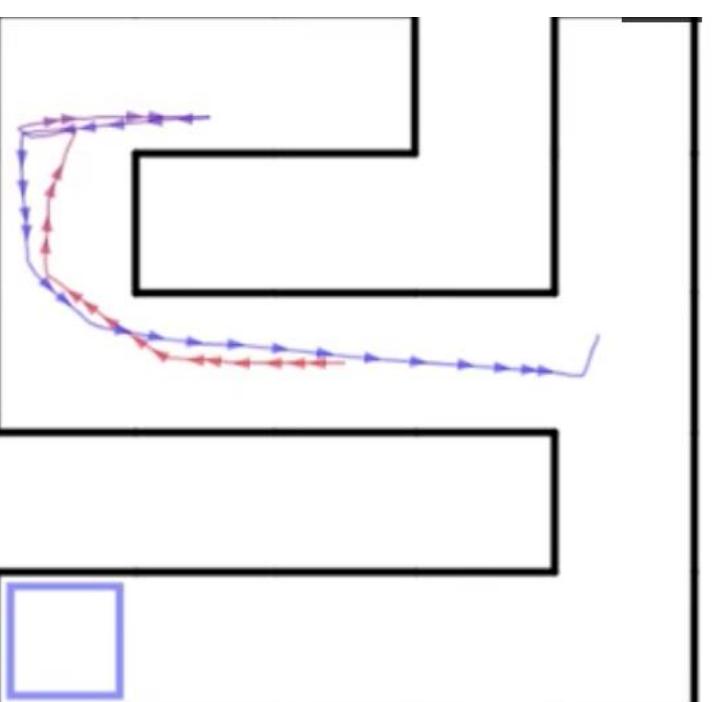


## Learning

Learn to solve a

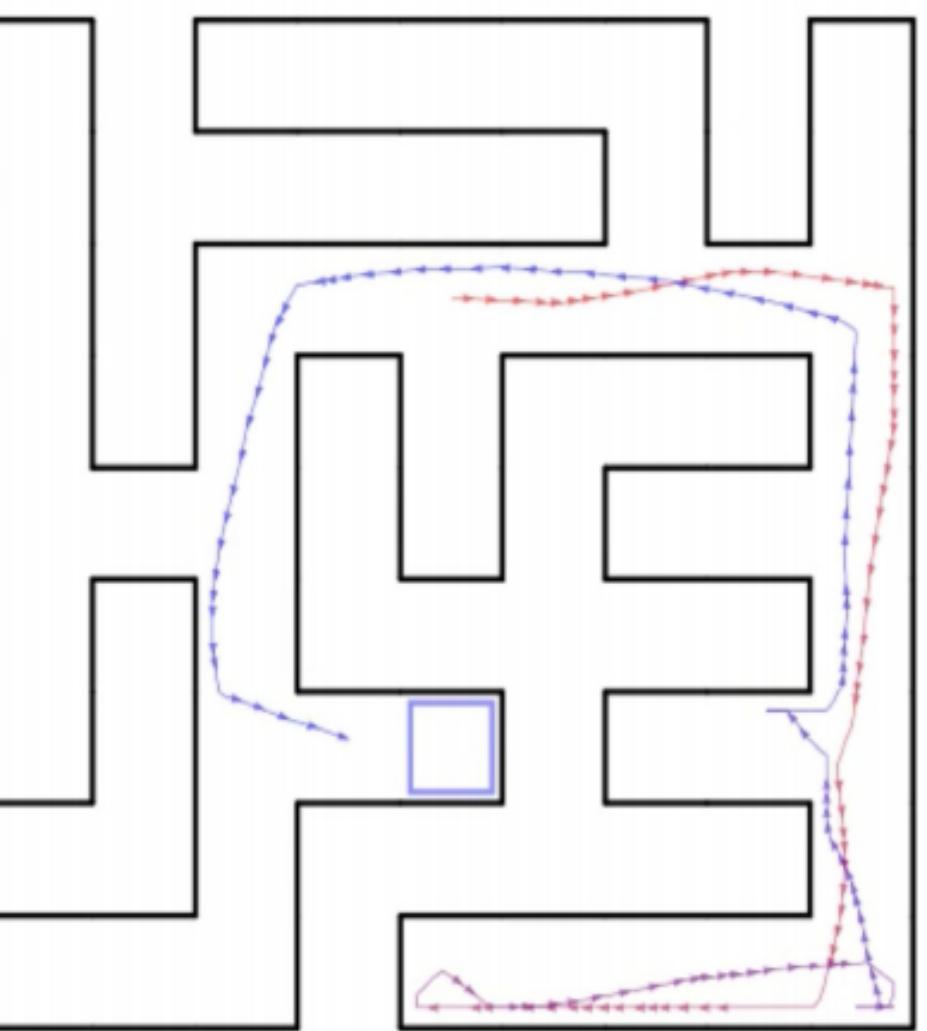


By learning how to learn many other tasks:



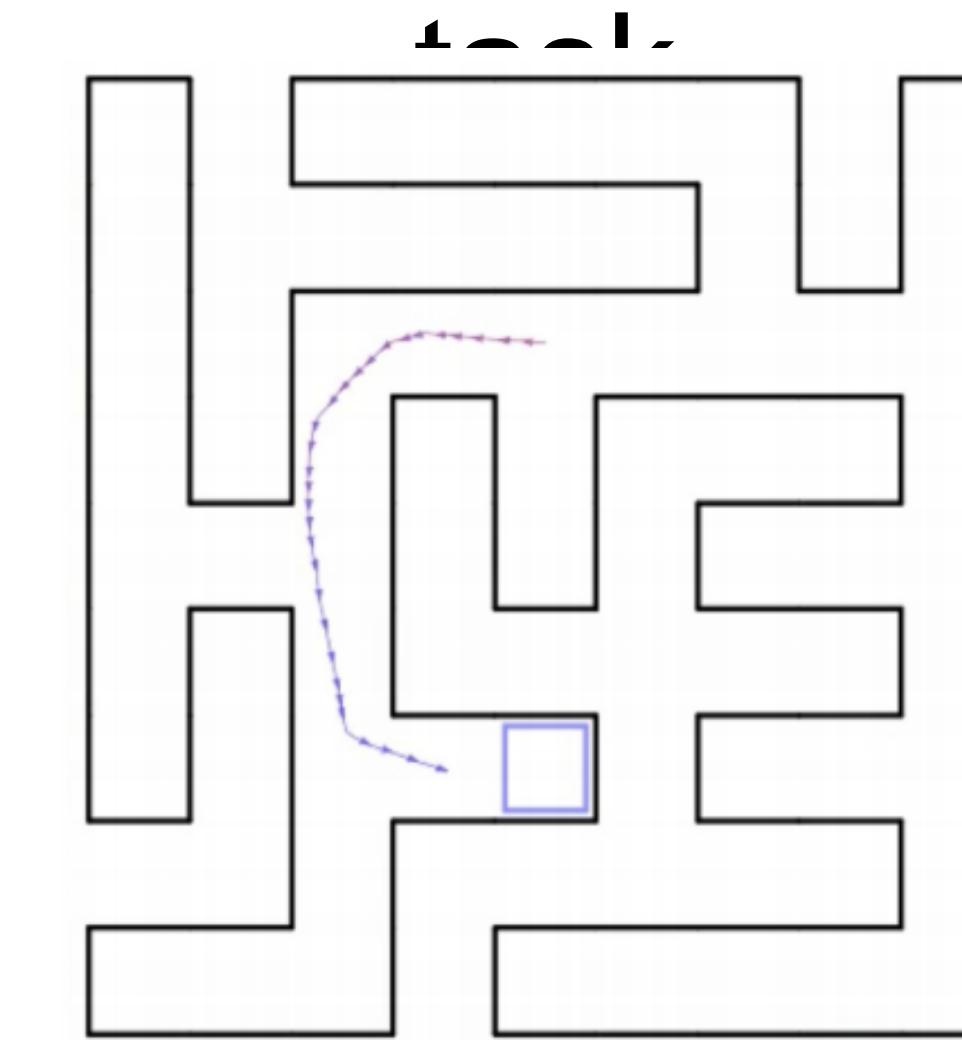
# Application: Fast Reinforcement Learning

Given a small amount of experience

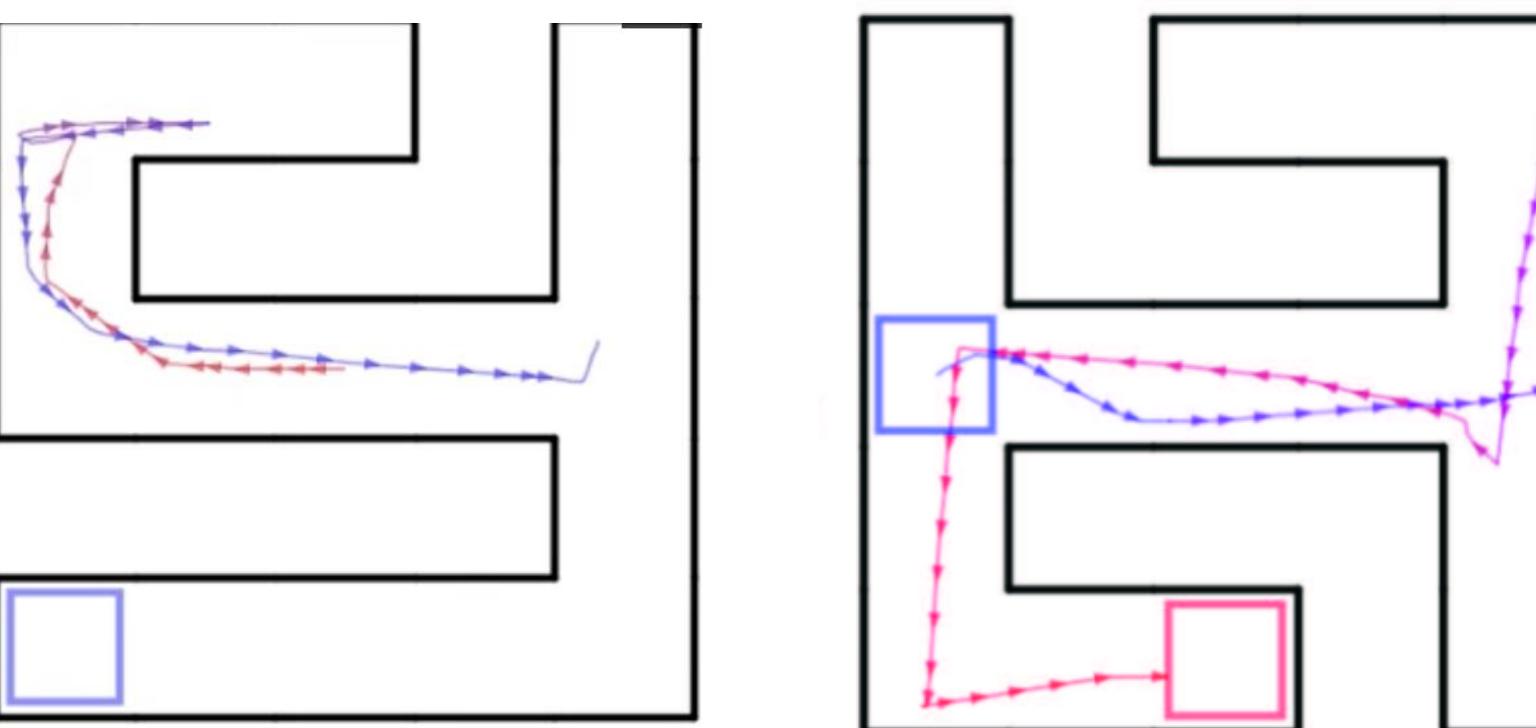


## Learning

Learn to solve a

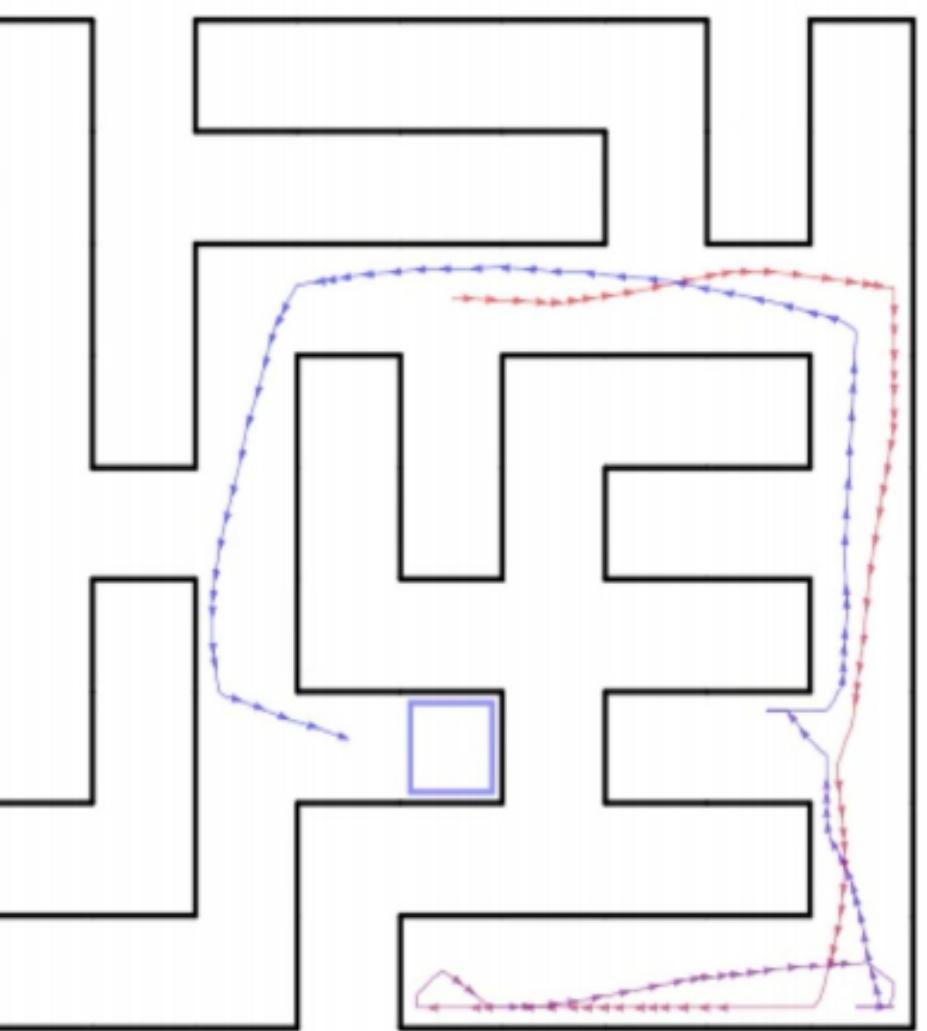


By learning how to learn many other tasks:



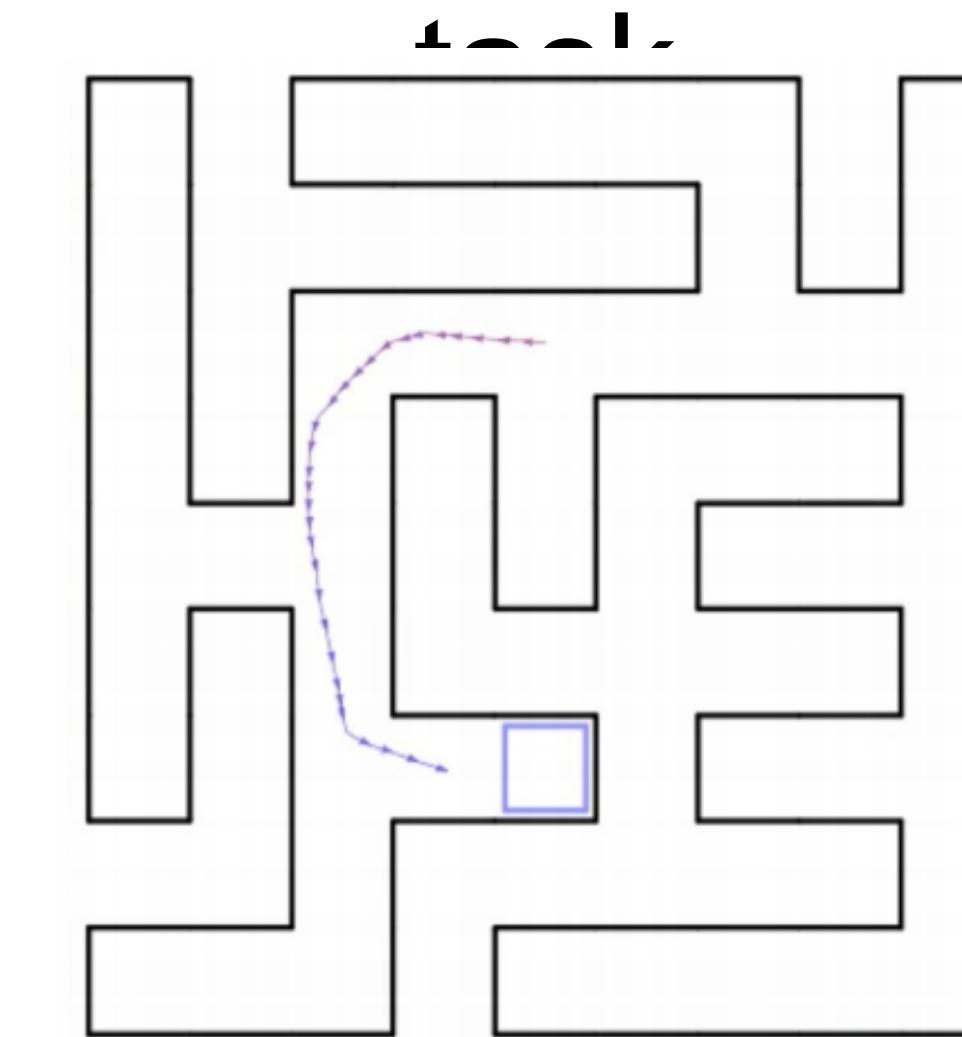
# Application: Fast Reinforcement Learning

Given a small amount of experience

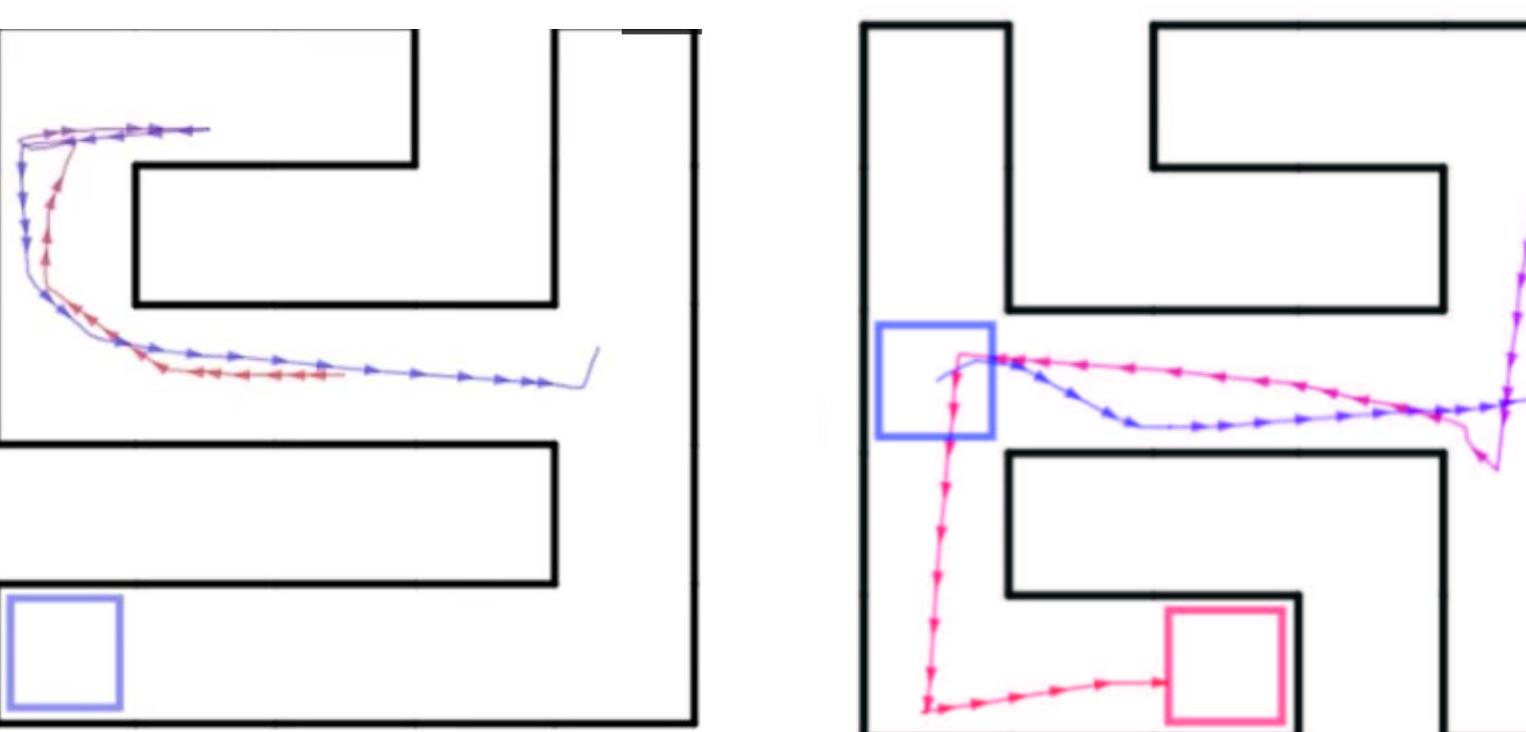


## Learning

Learn to solve a



By learning how to learn many other tasks:



...

# Outline

1. Applications of learning to learn
2. **Problem formulation**
3. Solution Classes:
  - a) metric-learning approach
  - b) direct black-box approach
  - c) gradient-based approach
4. Open Questions / Problems

# The Meta-Learning Problem

# The Meta-Learning Problem

**Supervised Learning:**

# The Meta-Learning Problem

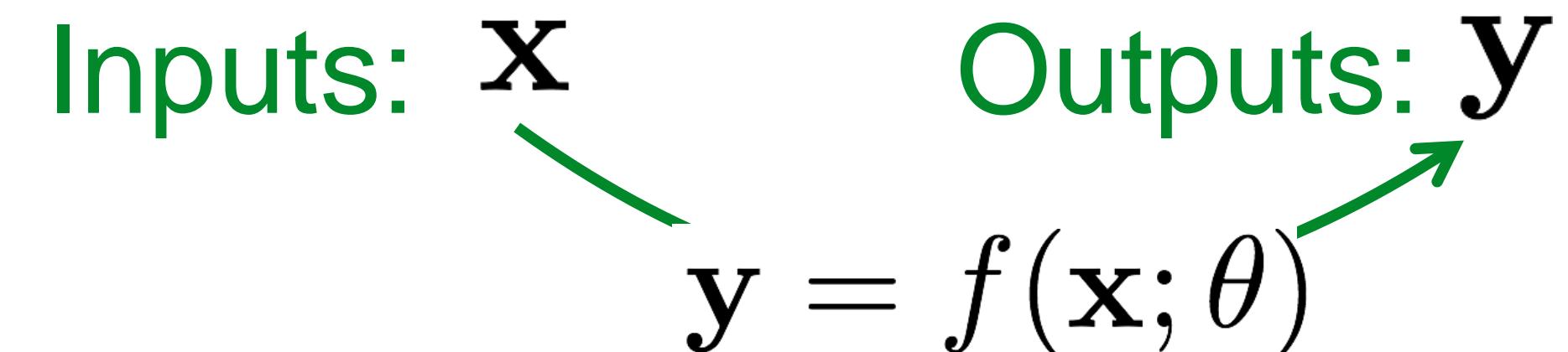
**Supervised Learning:**

Inputs:  $\mathbf{x}$

Outputs:  $\mathbf{y}$

# The Meta-Learning Problem

## Supervised Learning:



# The Meta-Learning Problem

## Supervised Learning:



# The Meta-Learning Problem

**Supervised Learning:**



**Meta-Supervised Learning:**

# The Meta-Learning Problem

## Supervised Learning:



## Meta-Supervised Learning:

Inputs:

# The Meta-Learning Problem

## Supervised Learning:

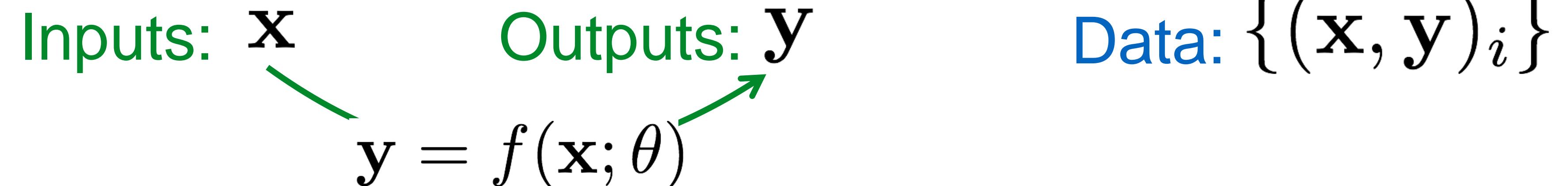


## Meta-Supervised Learning:

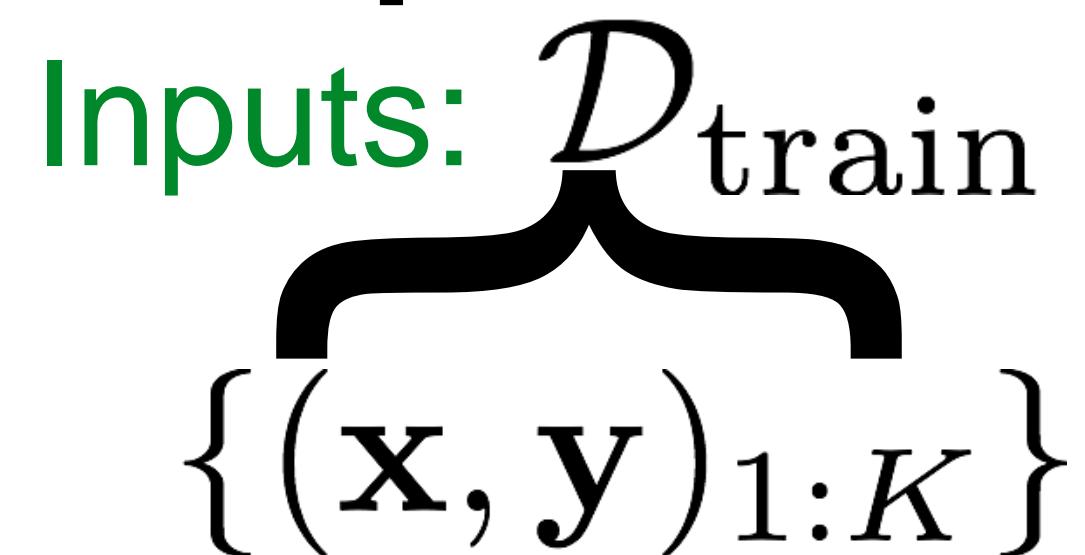
Inputs:  $\mathcal{D}_{\text{train}}$

# The Meta-Learning Problem

## Supervised Learning:

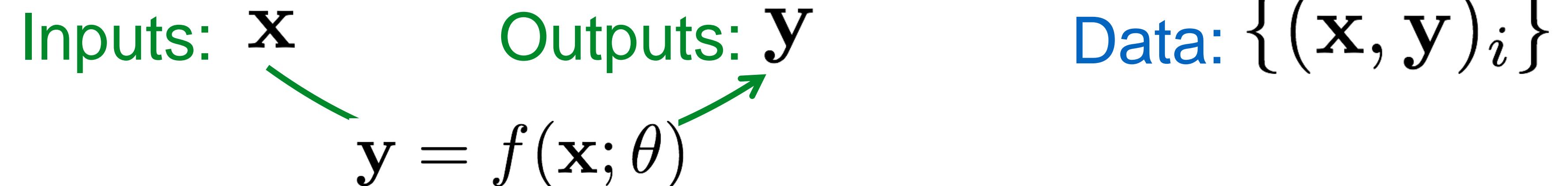


## Meta-Supervised Learning:

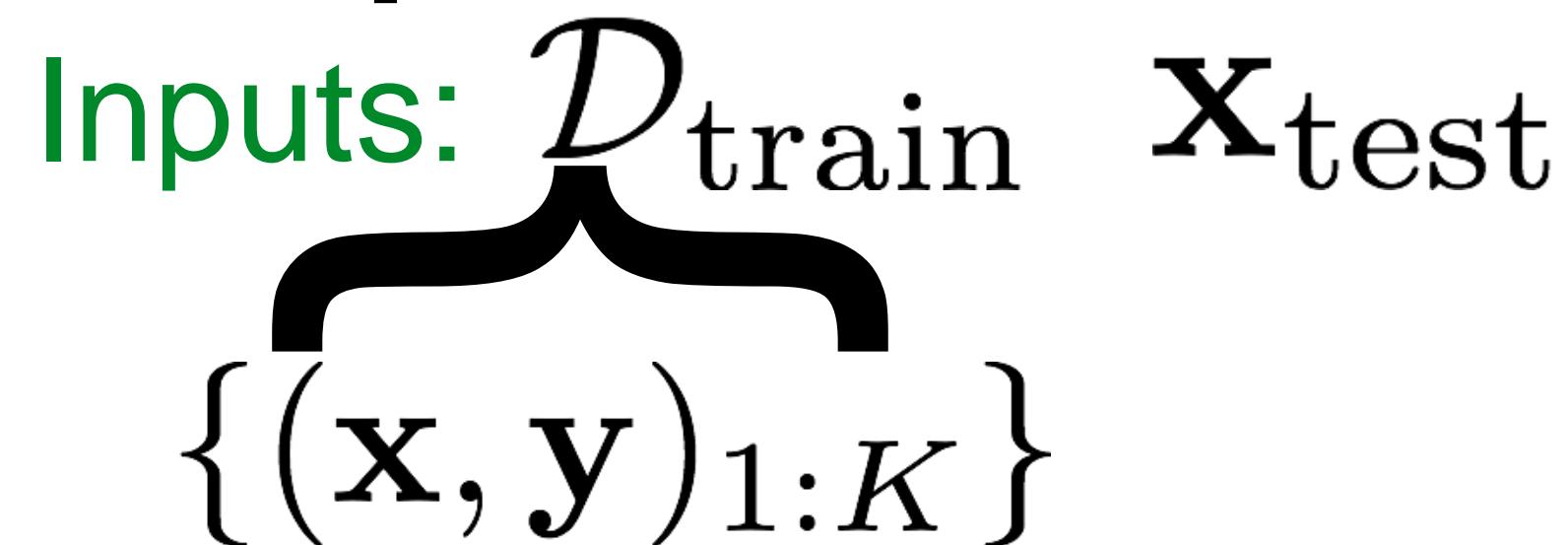


# The Meta-Learning Problem

## Supervised Learning:

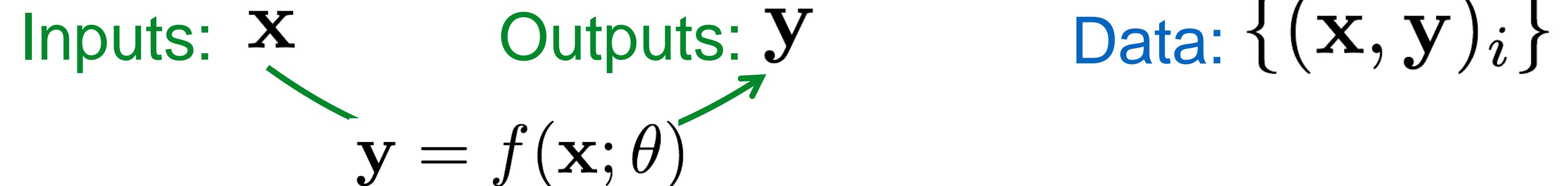


## Meta-Supervised Learning:

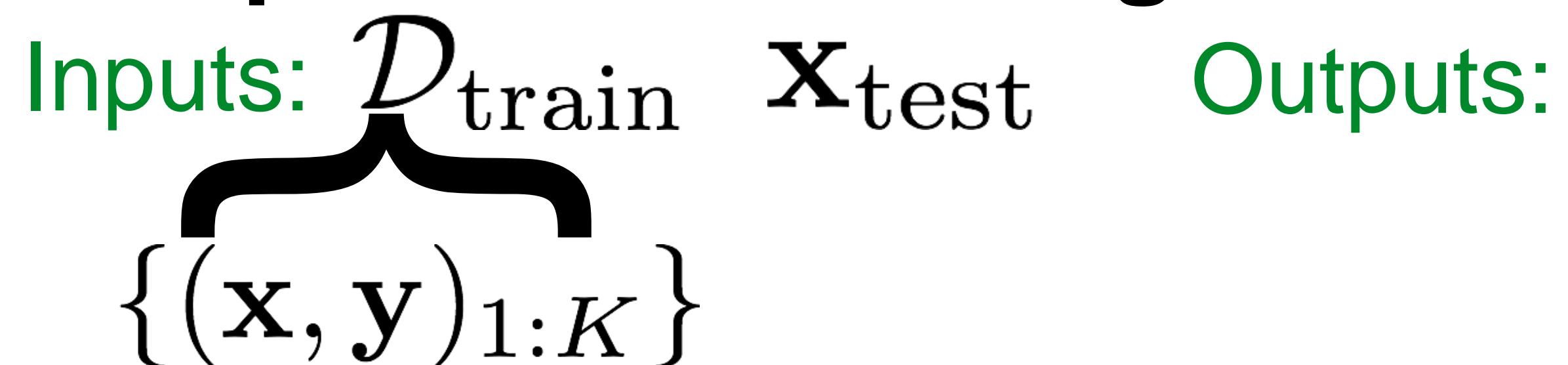


# The Meta-Learning Problem

## Supervised Learning:



## Meta-Supervised Learning:



# The Meta-Learning Problem

## Supervised Learning:

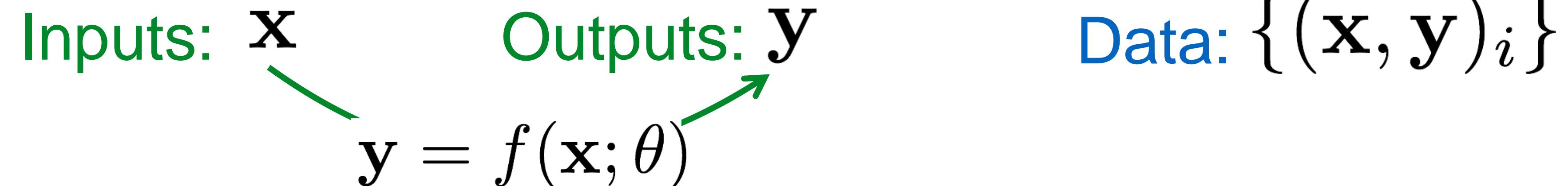


## Meta-Supervised Learning:

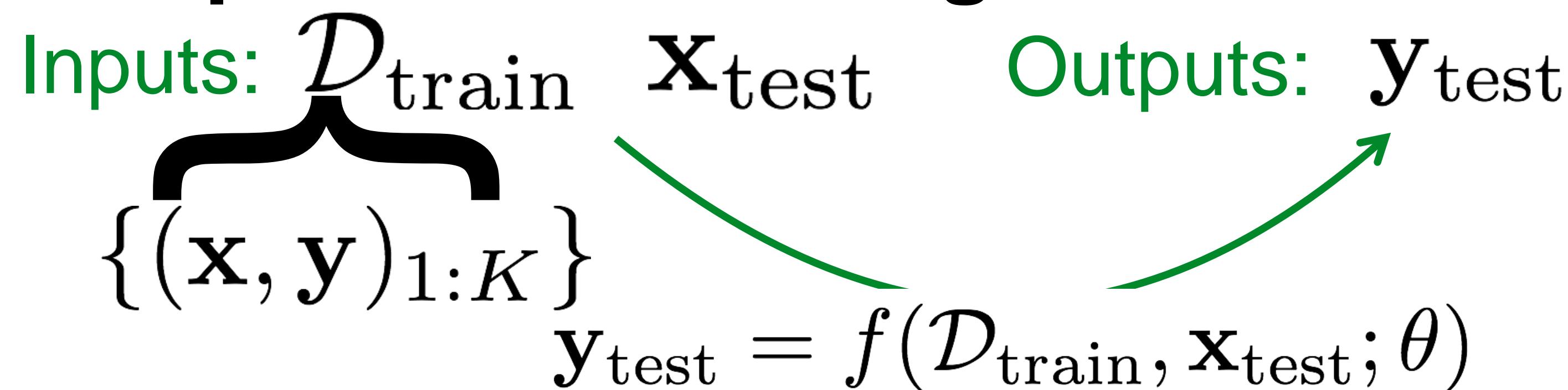


# The Meta-Learning Problem

## Supervised Learning:

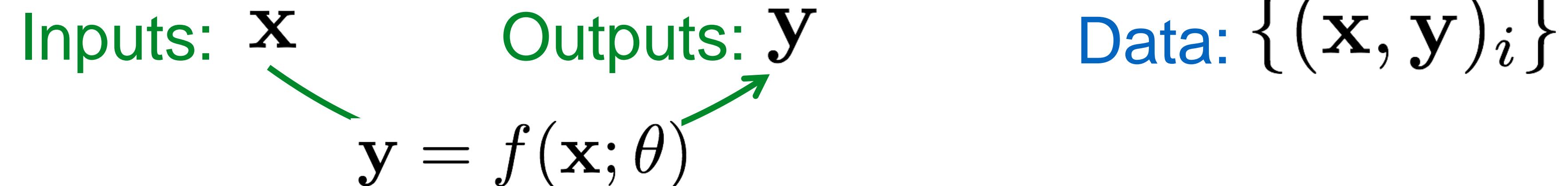


## Meta-Supervised Learning:

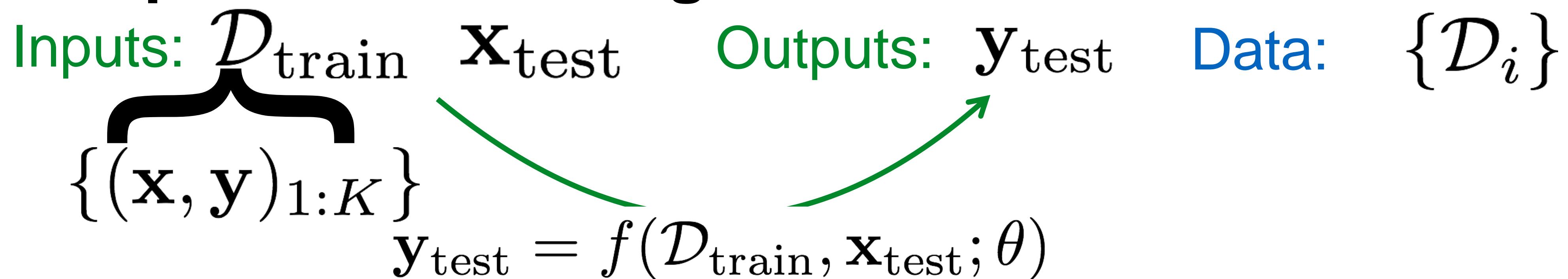


# The Meta-Learning Problem

## Supervised Learning:

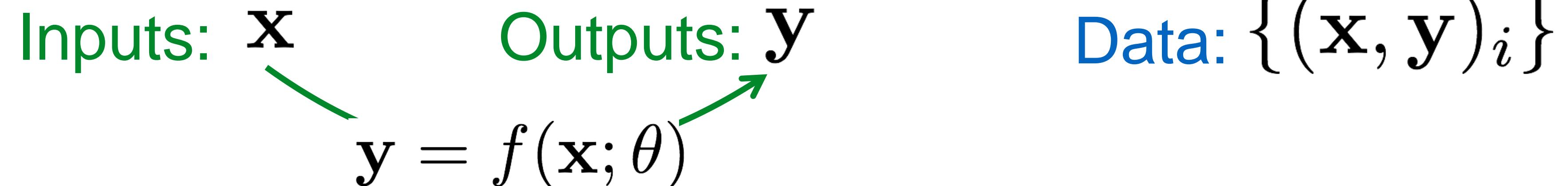


## Meta-Supervised Learning:

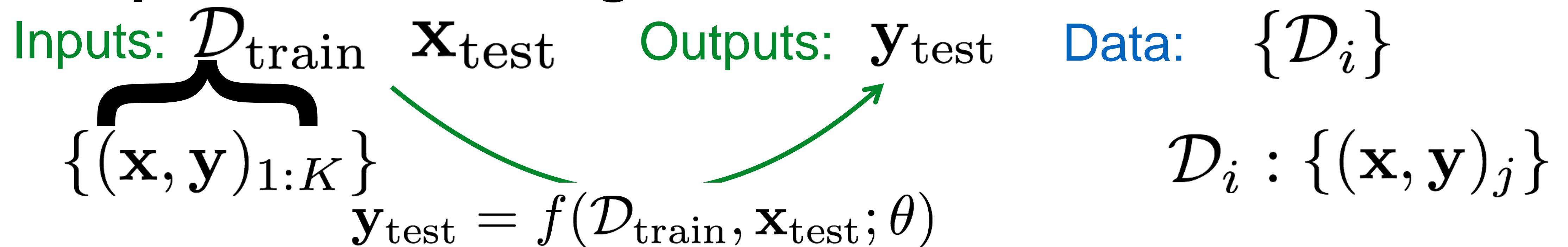


# The Meta-Learning Problem

## Supervised Learning:

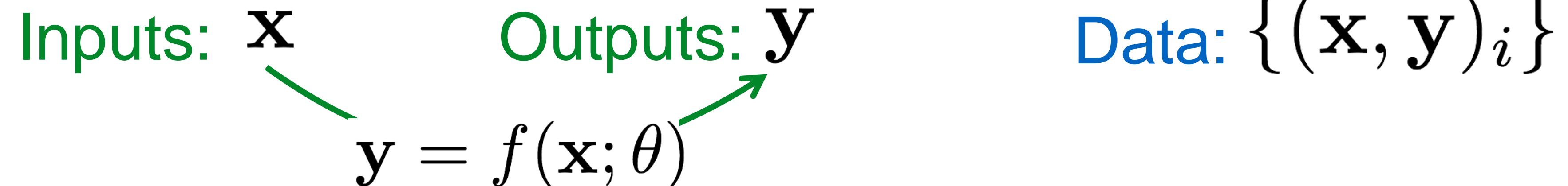


## Meta-Supervised Learning:

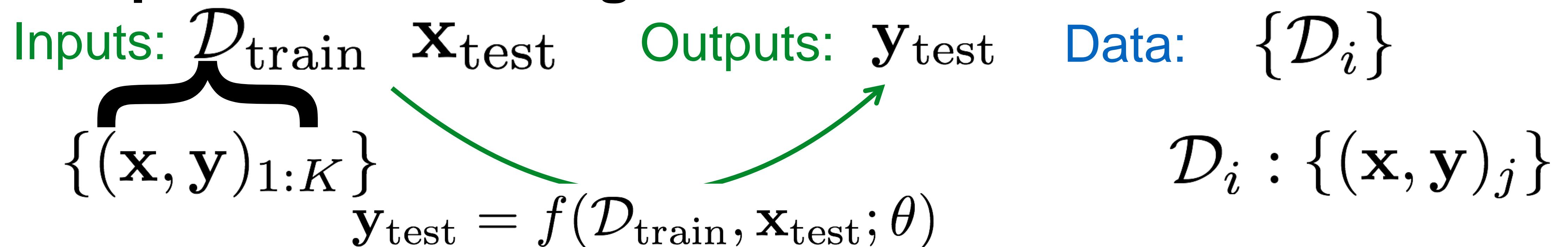


# The Meta-Learning Problem

## Supervised Learning:



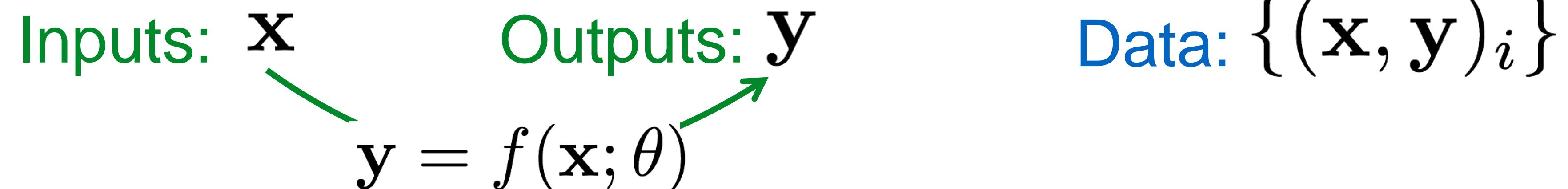
## Meta-Supervised Learning:



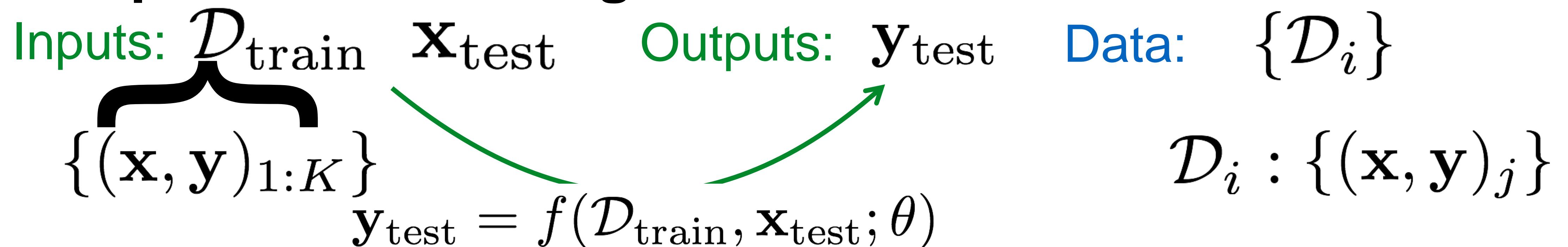
Why is this view useful?

# The Meta-Learning Problem

## Supervised Learning:



## Meta-Supervised Learning:



**Why is this view useful?**

Reduces the problem to the design & optimization of  $f$ .

# Application: Few-Shot Image

Given 1 example of 5 classes:



Classify new  
examples



# Application: Few-Shot Image

Given 1 example of 5 classes:



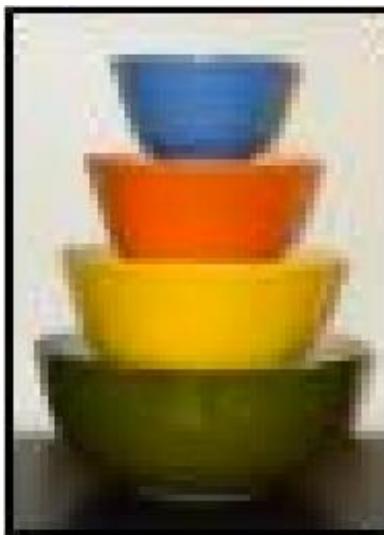
training data  $\mathcal{D}_{\text{train}}$

Classify new  
examples



# Application: Few-Shot Image

Given 1 example of 5 classes:



training data  $\mathcal{D}_{\text{train}}$

Classify new examples



test set  $\mathbf{X}_{\text{test}}$

# Application: Few-Shot Image Classification

Given 1 example of 5 classes:

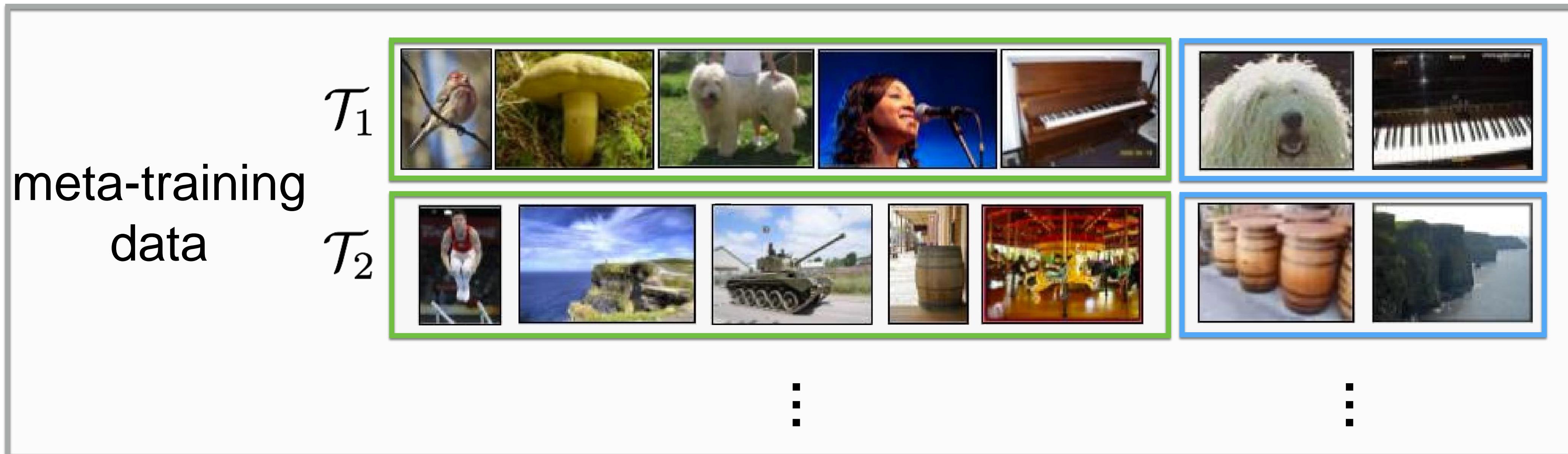


training data  $\mathcal{D}_{\text{train}}$

Classify new examples



test set  $\mathbf{X}_{\text{test}}$



# Outline

1. Applications of learning to learn
2. Problem formulation
3. **Solution Classes:**
  - a) **metric-learning approach**
  - b) direct black-box approach
  - c) gradient-based approach
4. Open Questions / Problems

# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$$

training data  $\mathcal{D}_{\text{train}}$



test set  $\mathbf{x}_{\text{test}}$



# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$$

training data  $\mathcal{D}_{\text{train}}$



test set  $\mathbf{x}_{\text{test}}$



# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$$

training data  $\mathcal{D}_{\text{train}}$



test set  $\mathbf{x}_{\text{test}}$



# Metric Learning Approach

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$

training data  $\mathcal{D}_{\text{train}}$



test set  $\mathbf{x}_{\text{test}}$



# Metric Learning Approach

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$

training data  $\mathcal{D}_{\text{train}}$



test set  $\mathbf{x}_{\text{test}}$



# Metric Learning Approach

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$

training data  $\mathcal{D}_{\text{train}}$

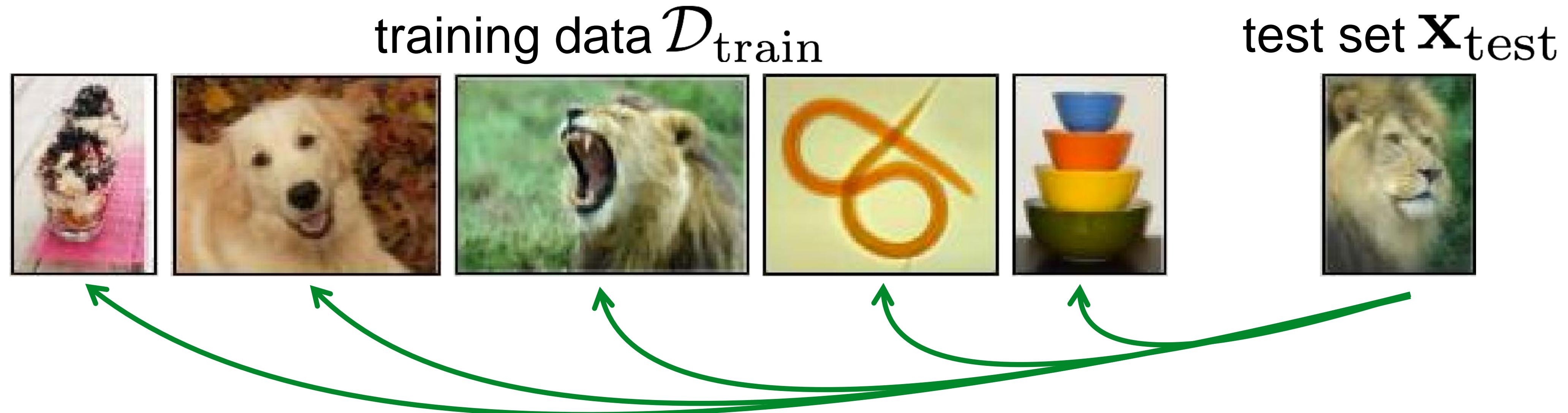


test set  $\mathbf{x}_{\text{test}}$



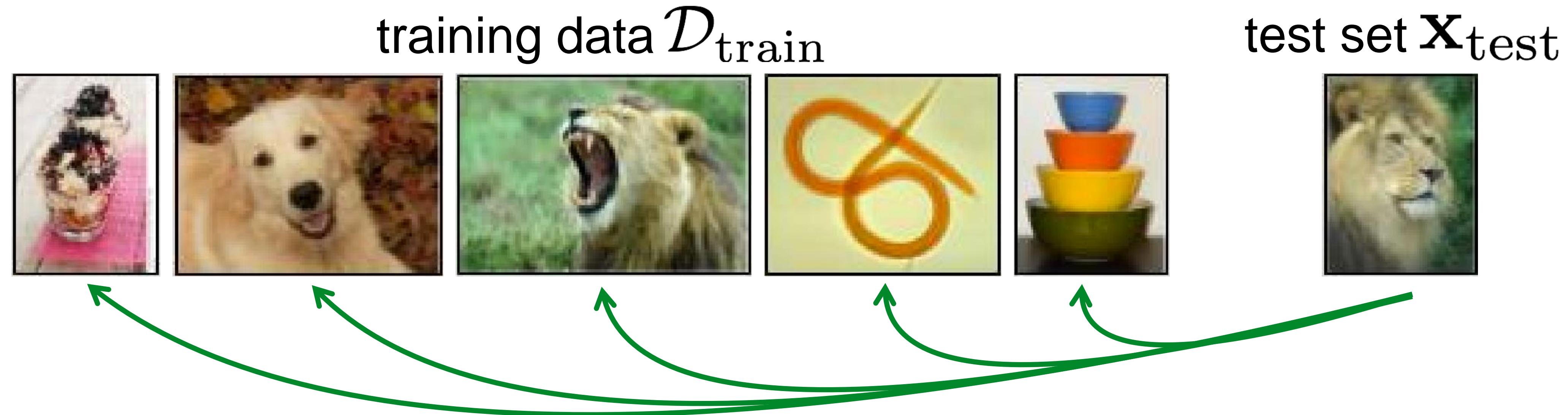
# Metric Learning Approach

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$



**Key idea:** compare test image with training images

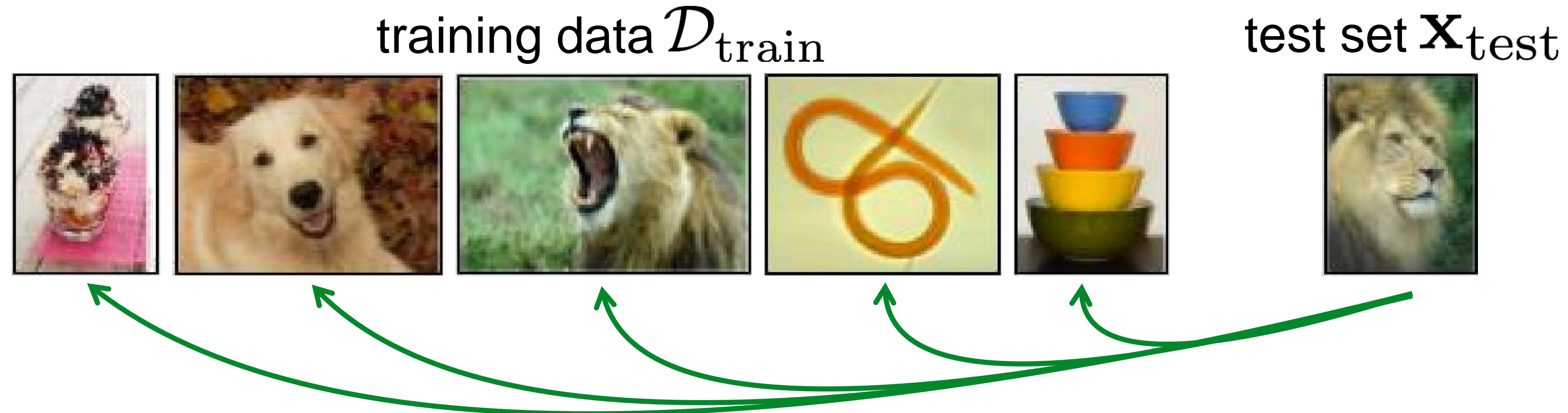
# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$


**Key idea:** compare test image with training images  
 $f :=$  nearest neighbors

# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$

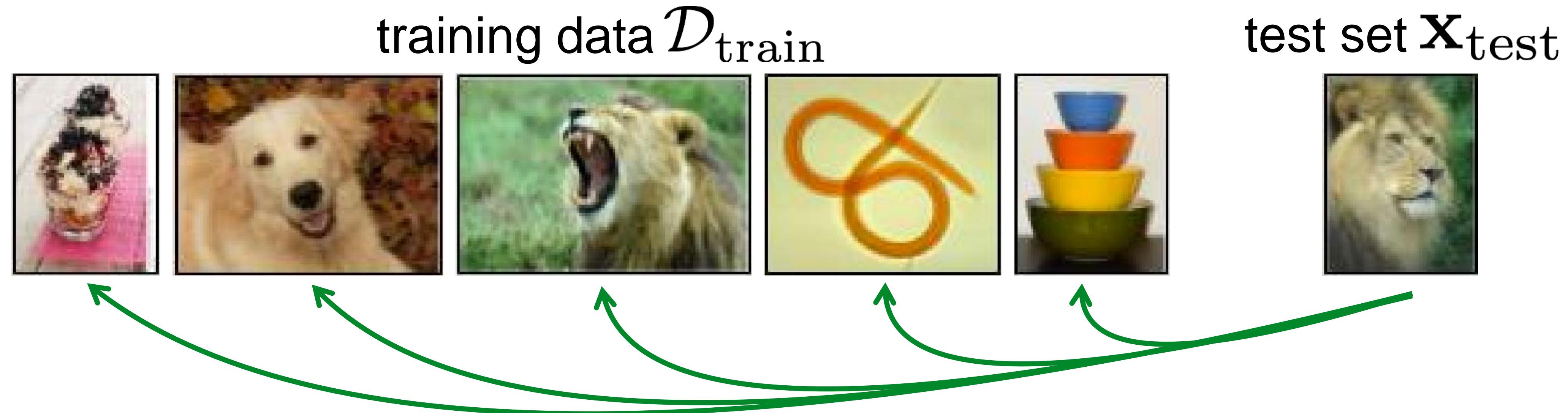


**Key idea:** compare test image with training images  
 $f :=$  nearest neighbors

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

# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$



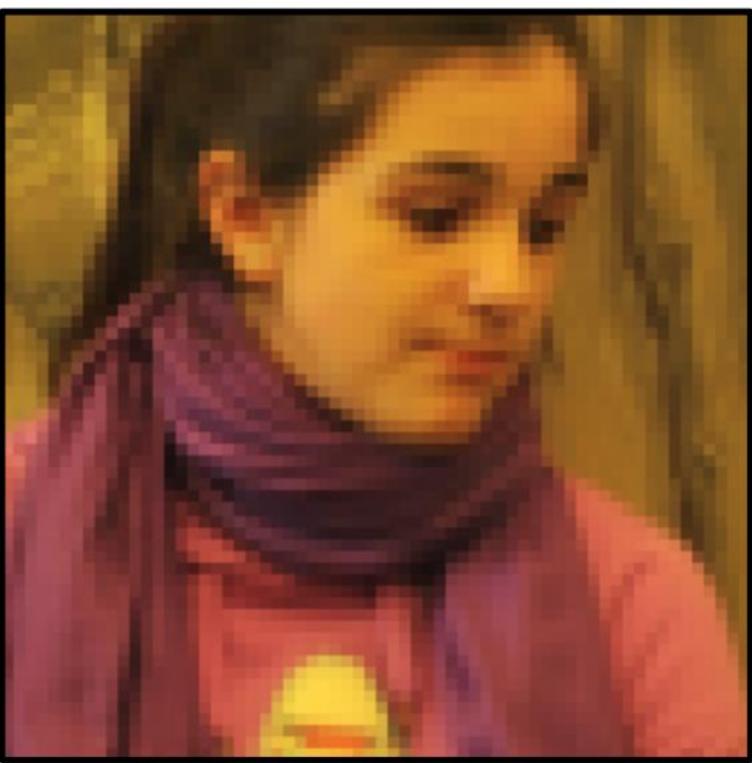
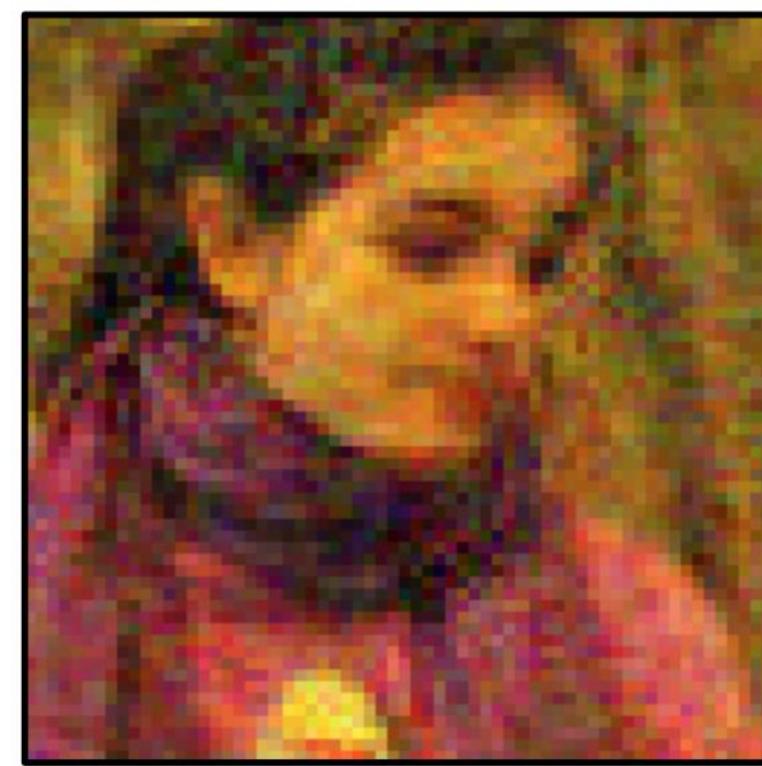
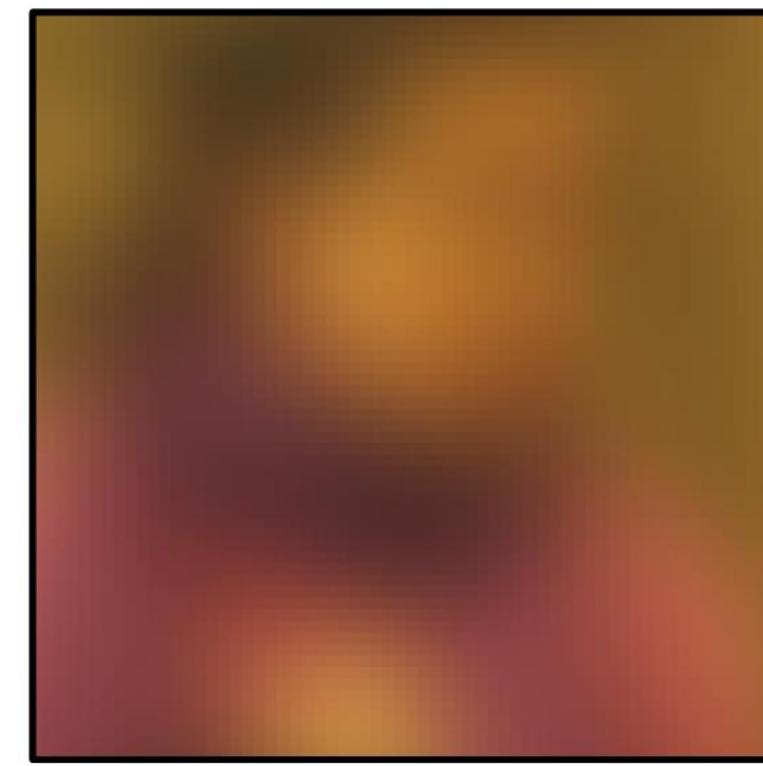
**Key idea:** compare test image with training images  
 $f :=$  nearest neighbors

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

pixel space,  $\ell_2$  distance?

pixel space,  $\ell_2$  distance?

pixel space,  $\| \cdot \|_2$  distance?



Zhang et al. arXiv '18 (1801.03924)

pixel space,  $\ell_2$  distance?



Zhang et al. arXiv '18 (1801.03924)

Chelsea Finn, UC Berkeley

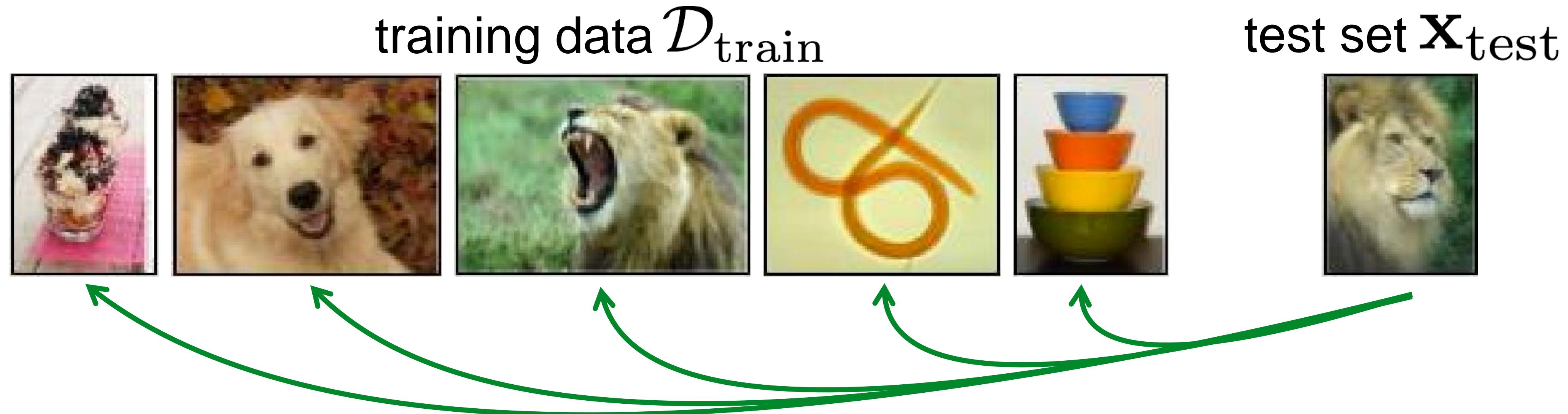
pixel space,  $\ell_2$  distance?



Zhang et al. arXiv '18 (1801.03924)

# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$



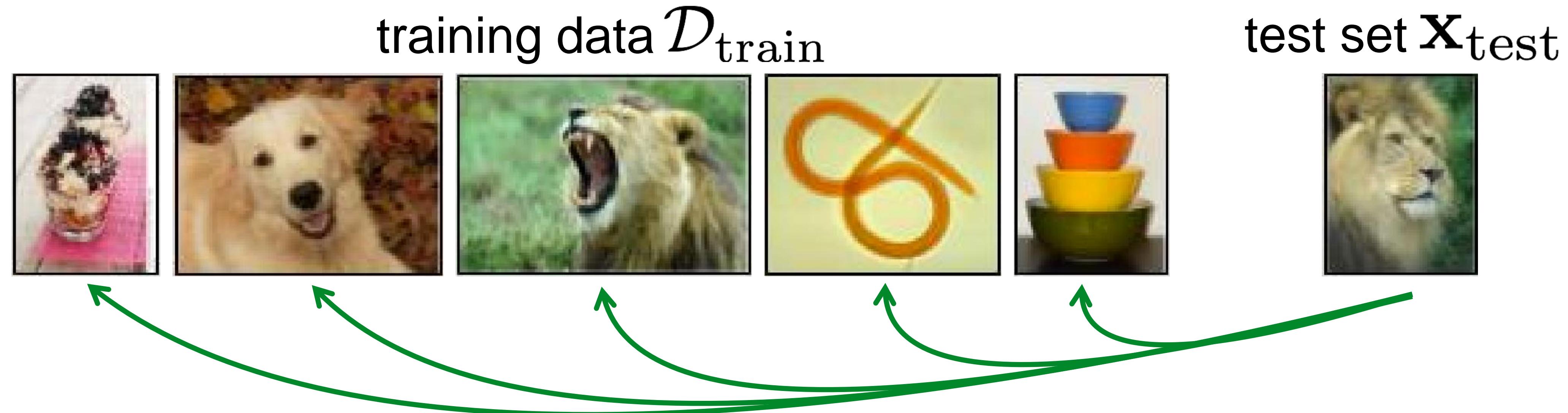
**Key idea:** compare test image with training images

$f :=$  nearest neighbors

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

# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$



**Key idea:** compare test image with training images  
 $f :=$  nearest neighbors

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

learn how to compare using data

# Metric Learning Example: Siamese Networks

Koch et al., ICML '15

---

## Siamese Neural Networks for One-shot Image Recognition

---

**Gregory Koch**

**Richard Zemel**

**Ruslan Salakhutdinov**

Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.

GKOCH@CS.TORONTO.EDU

ZEMEL@CS.TORONTO.EDU

RSALAKHU@CS.TORONTO.EDU

# Siamese Networks for One-Shot Image Recognition

Koch et al., ICML '15

# Siamese Networks for One-Shot Image Recognition

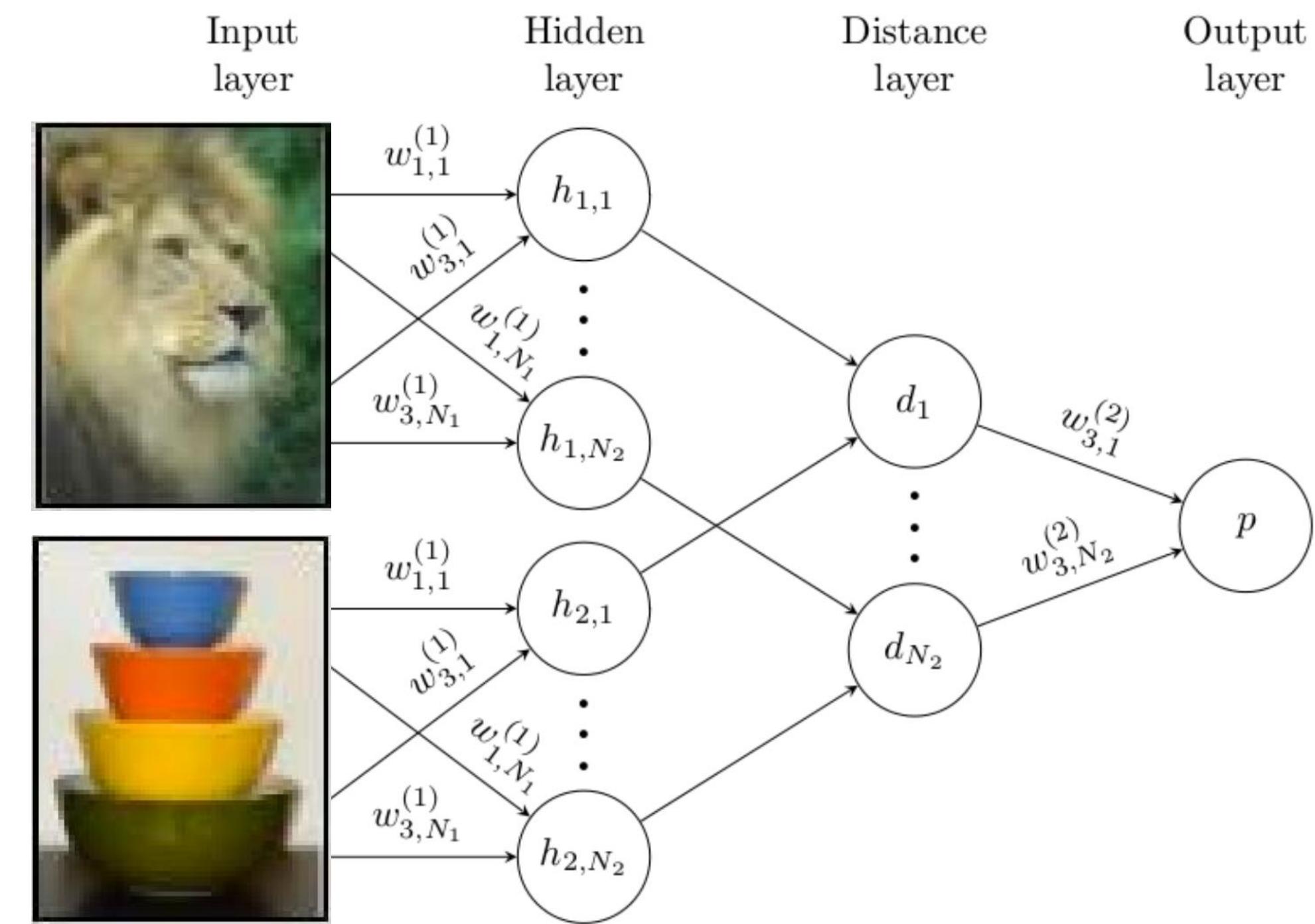
Koch et al., ICML '15

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

# Siamese Networks for One-Shot Image Recognition

Koch et al., ICML '15

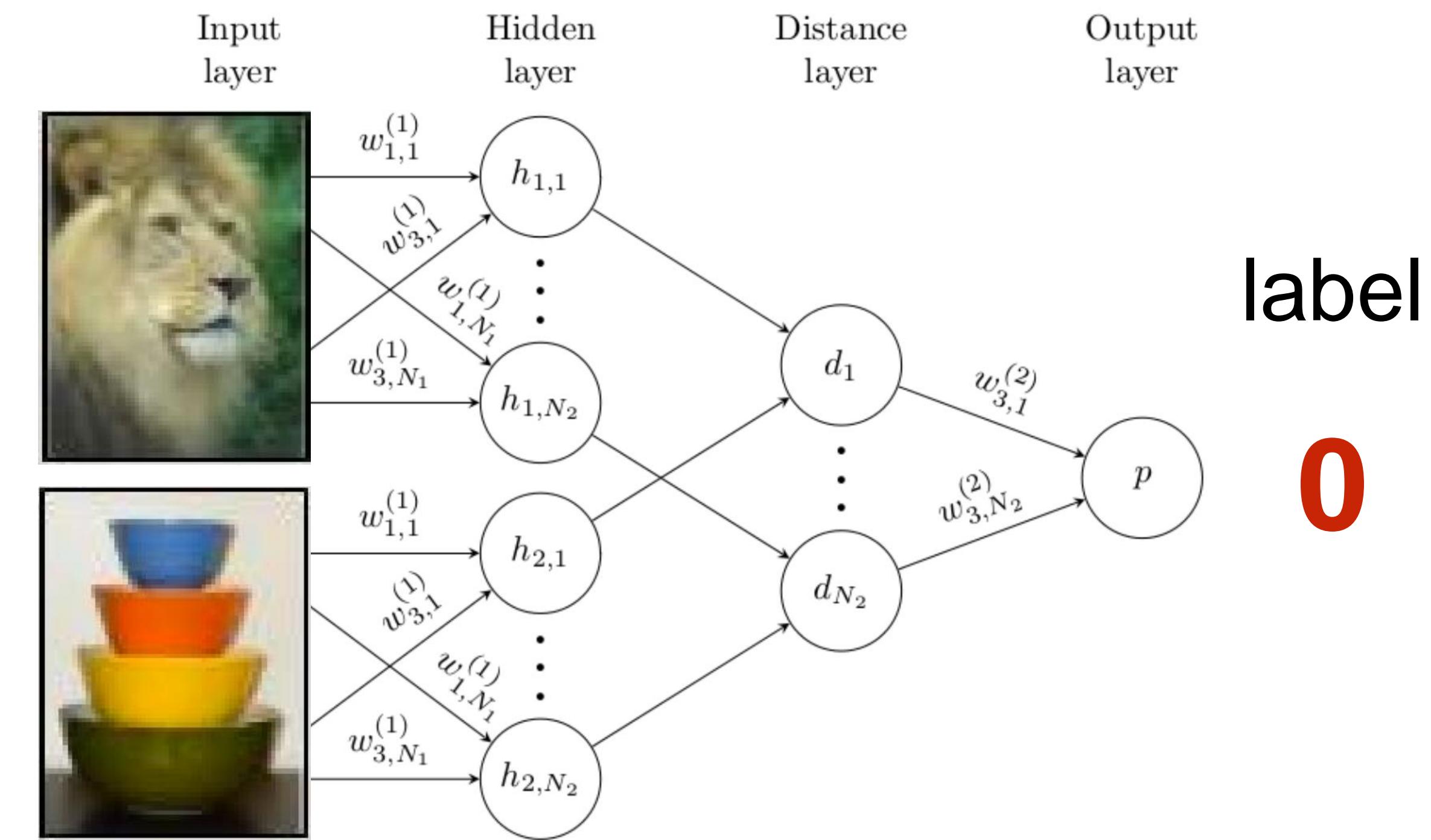
train network to predict whether or not two images are the same class



# Siamese Networks for One-Shot Image Recognition

Koch et al., ICML '15

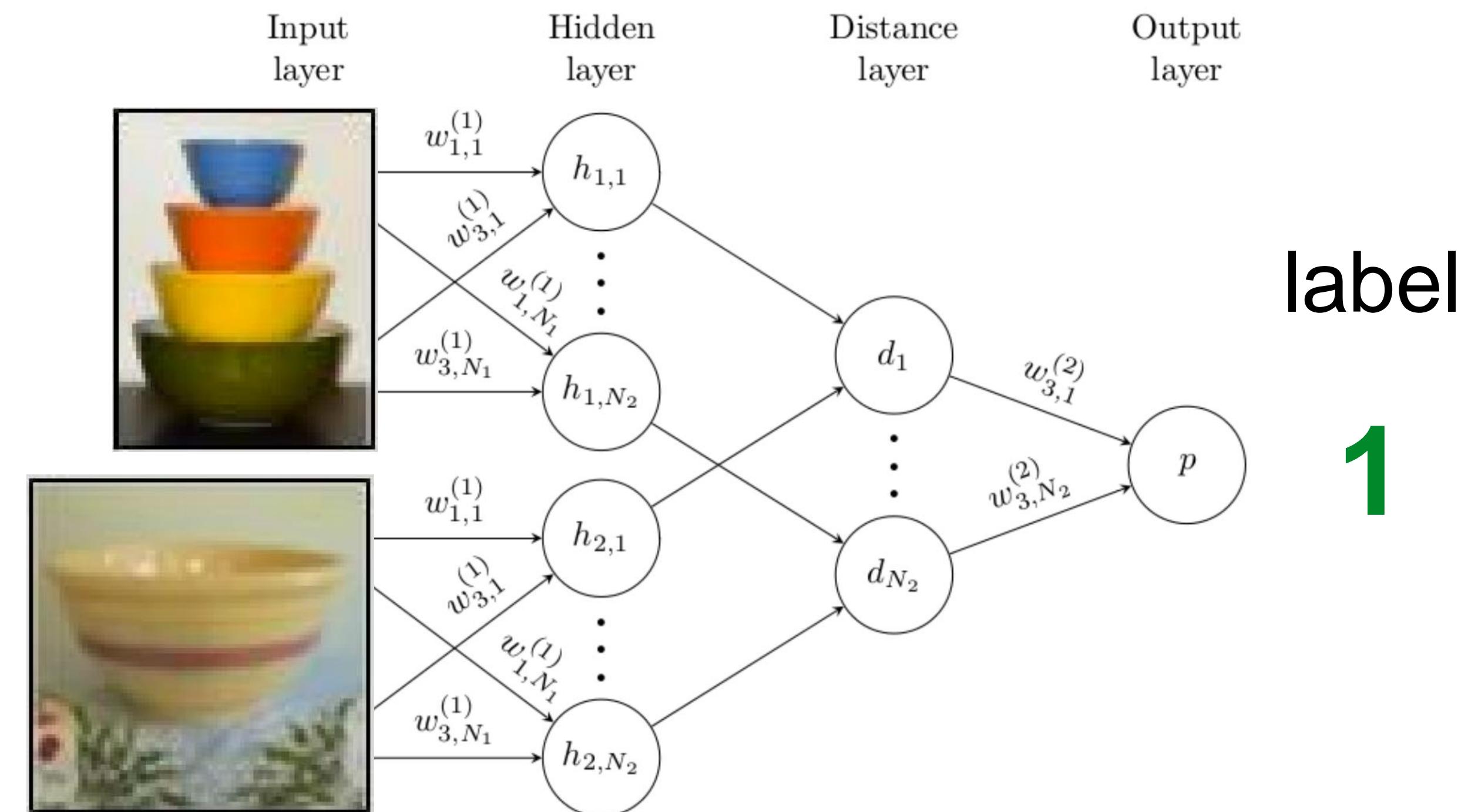
train network to predict whether or not two images are the same class



# Siamese Networks for One-Shot Image Recognition

Koch et al., ICML '15

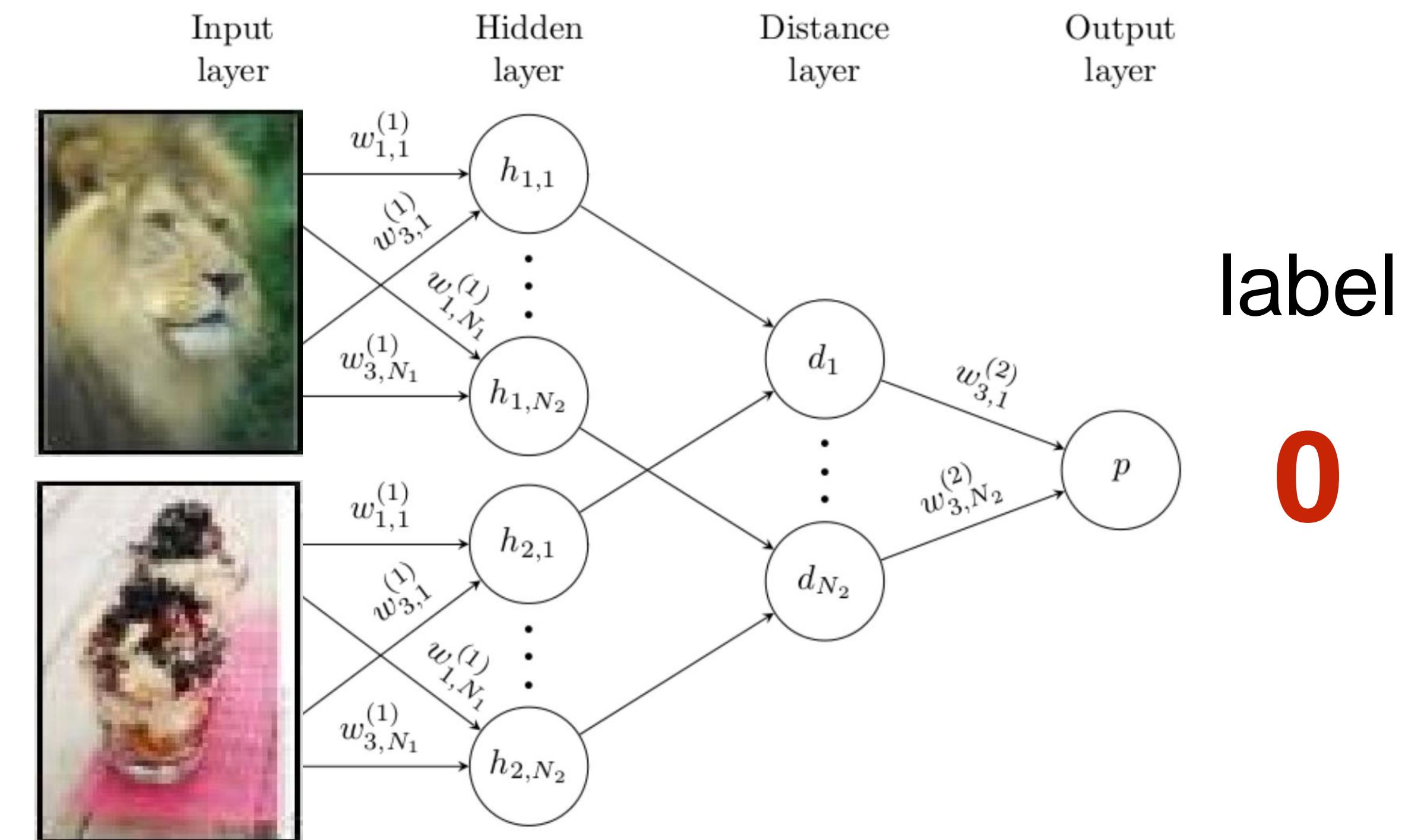
train network to predict whether or not two images are the same class



# Siamese Networks for One-Shot Image Recognition

Koch et al., ICML '15

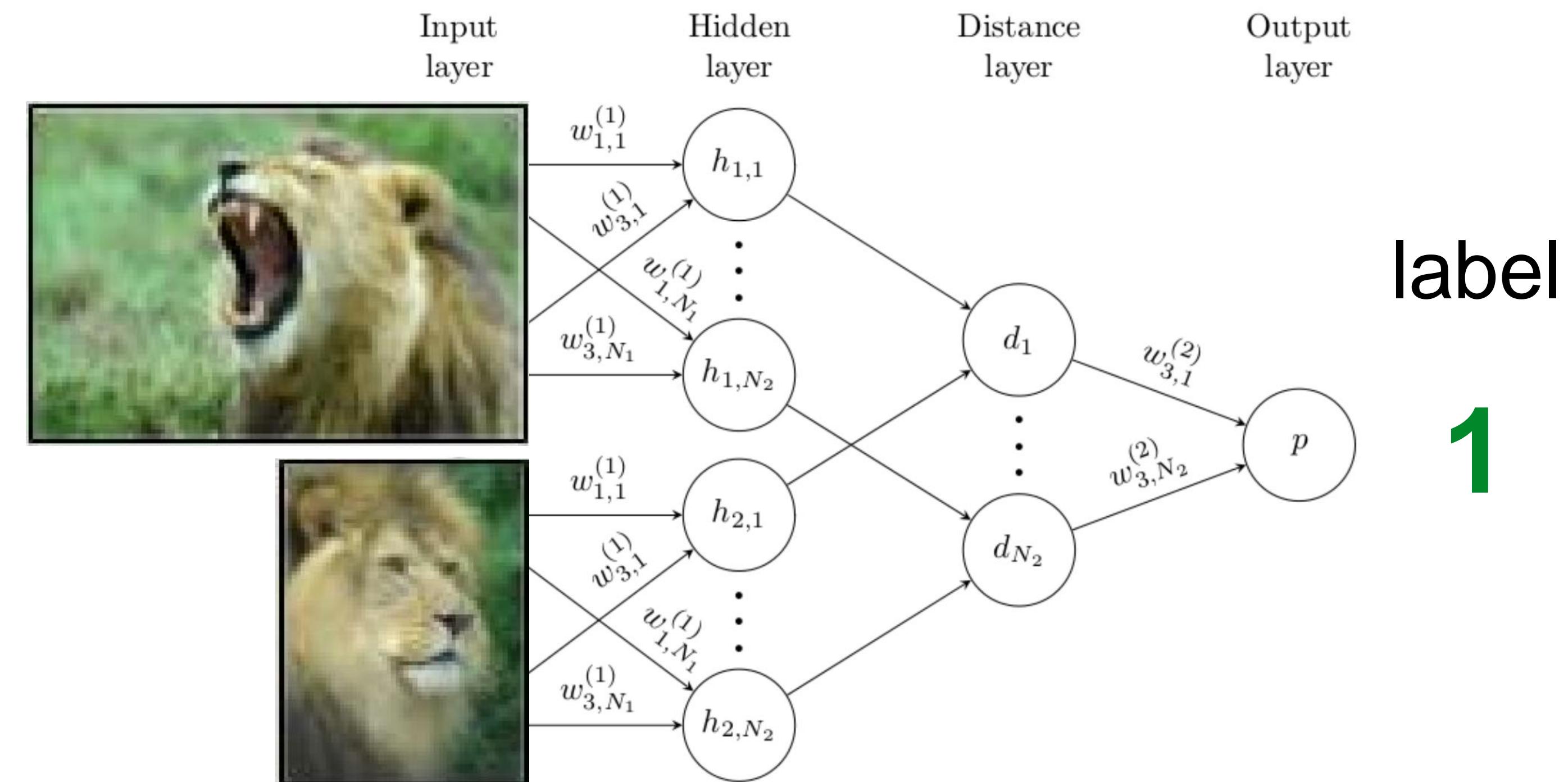
train network to predict whether or not two images are the same class



# Siamese Networks for One-Shot Image Recognition

Koch et al., ICML '15

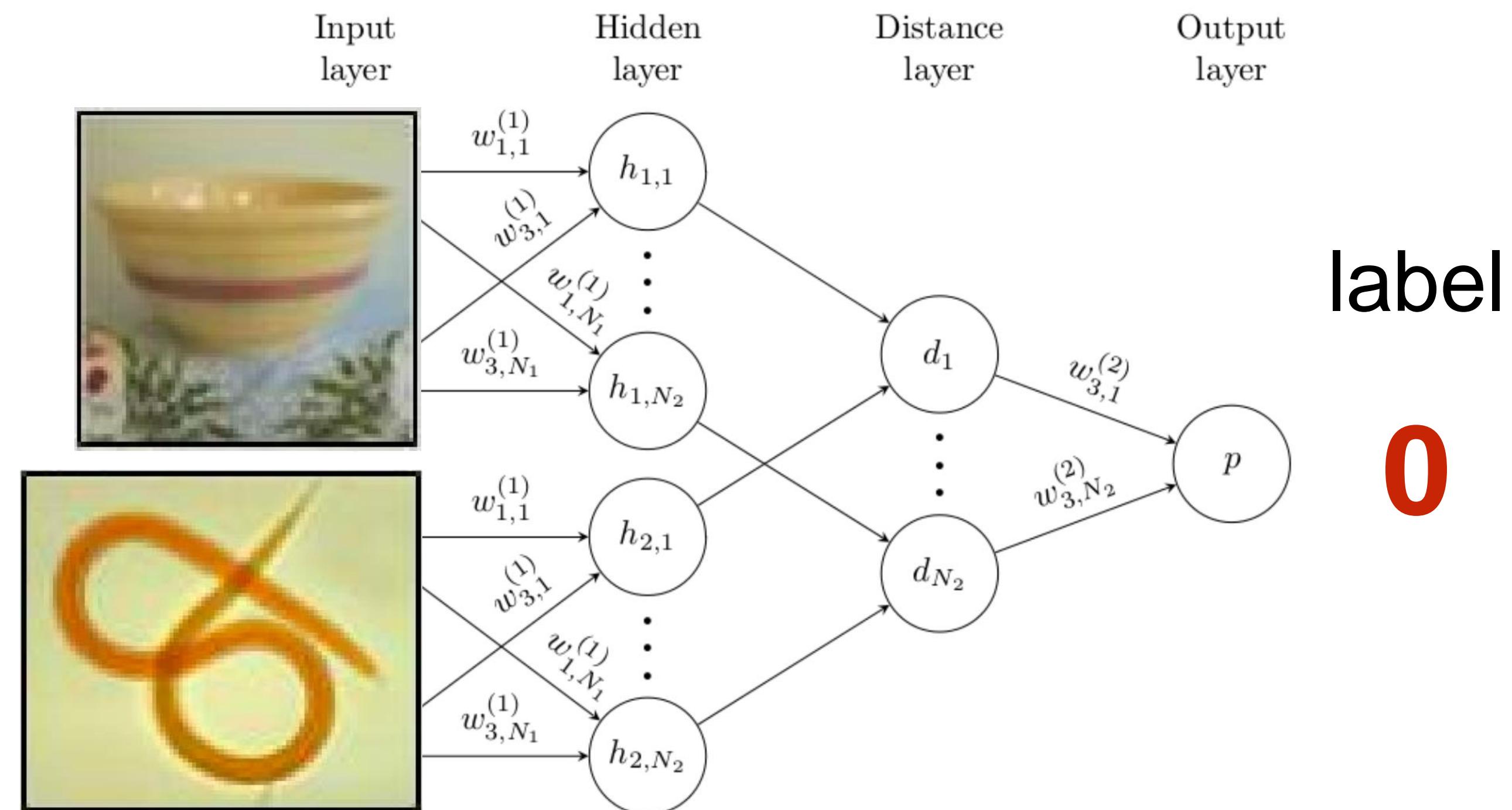
train network to predict whether or not two images are the same class



# Siamese Networks for One-Shot Image Recognition

Koch et al., ICML '15

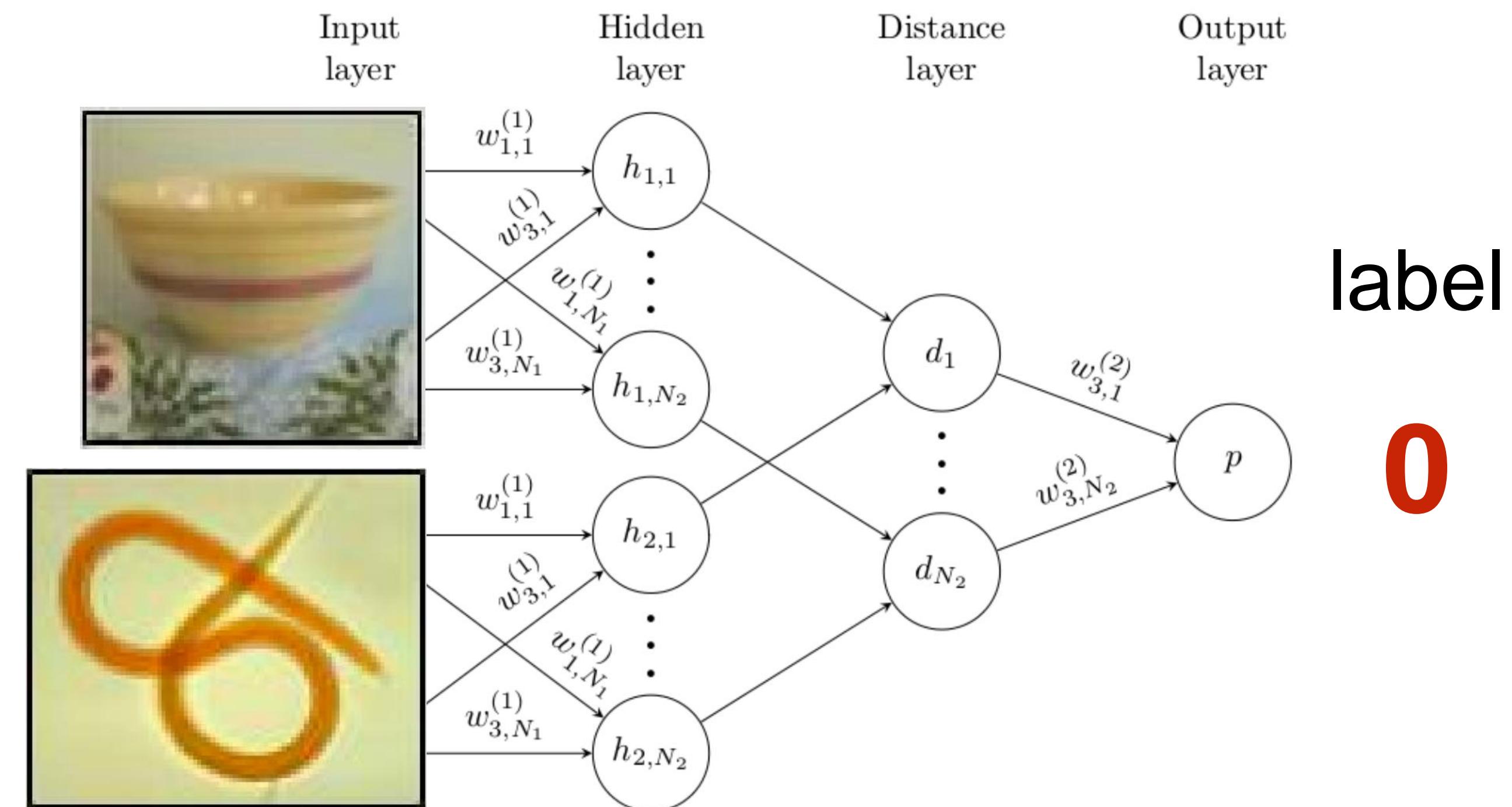
train network to predict whether or not two images are the same class



# Siamese Networks for One-Shot Image Recognition

Koch et al., ICML '15

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



Test time: compare image

$\mathbf{x}_{\text{test}}$  to each image in

$\mathcal{D}_{\text{train}}$

# Metric Learning Example: Matching Networks

Vinyals et al., NIPS '16

---

## Matching Networks for One Shot Learning

---

**Oriol Vinyals**  
Google DeepMind  
[vinyals@google.com](mailto:vinyals@google.com)

**Charles Blundell**  
Google DeepMind  
[cblundell@google.com](mailto:cblundell@google.com)

**Timothy Lillicrap**  
Google DeepMind  
[countzero@google.com](mailto:countzero@google.com)

**Koray Kavukcuoglu**  
Google DeepMind  
[korayk@google.com](mailto:korayk@google.com)

**Daan Wierstra**  
Google DeepMind  
[wierstra@google.com](mailto:wierstra@google.com)

# Matching Networks for One-Shot Learning

Vinyals et al., NIPS '16

# Matching Networks for One-Shot Learning

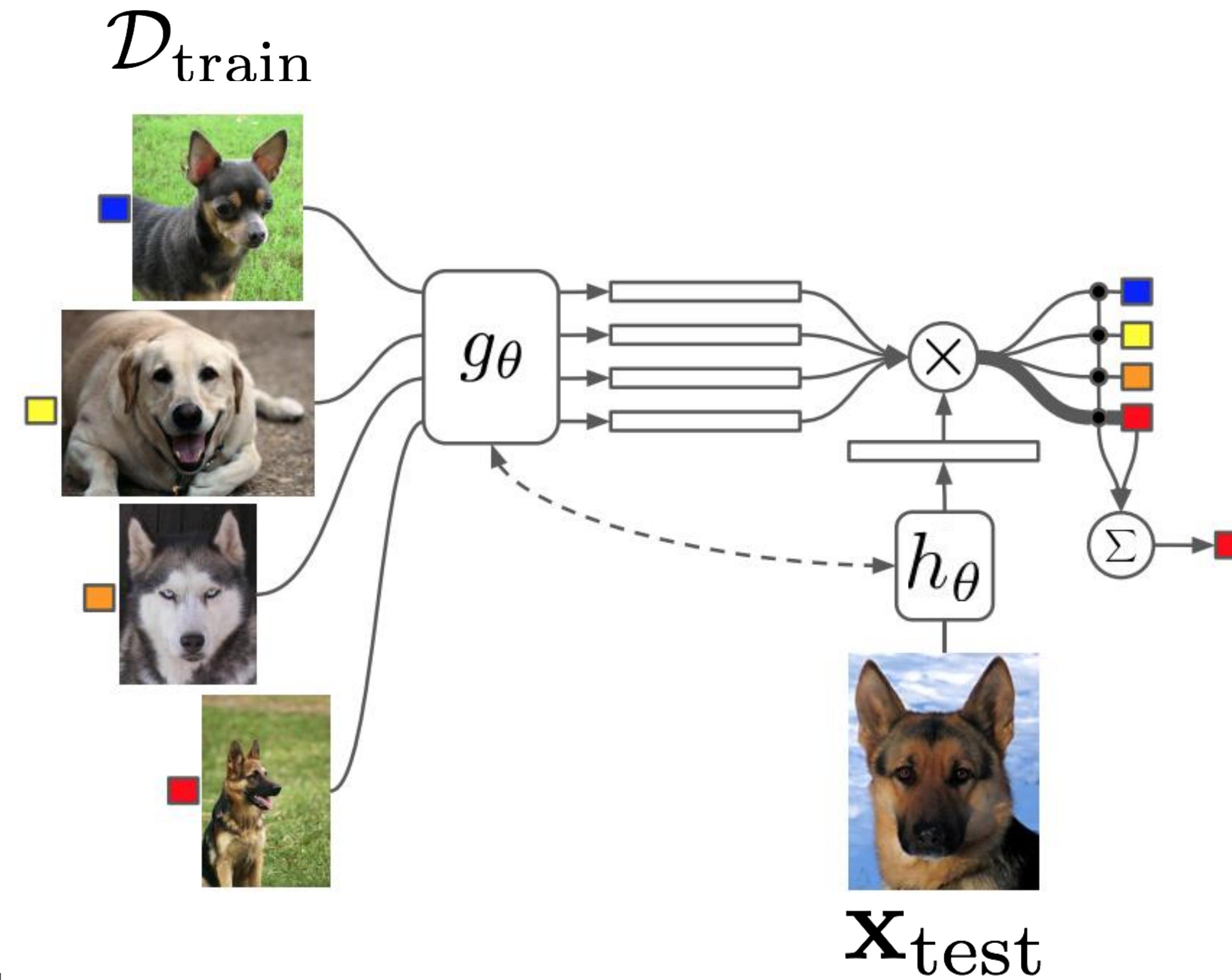
Vinyals et al., NIPS '16

motivating principle: train and test conditions must match

# Matching Networks for One-Shot Learning

Vinyals et al., NIPS '16

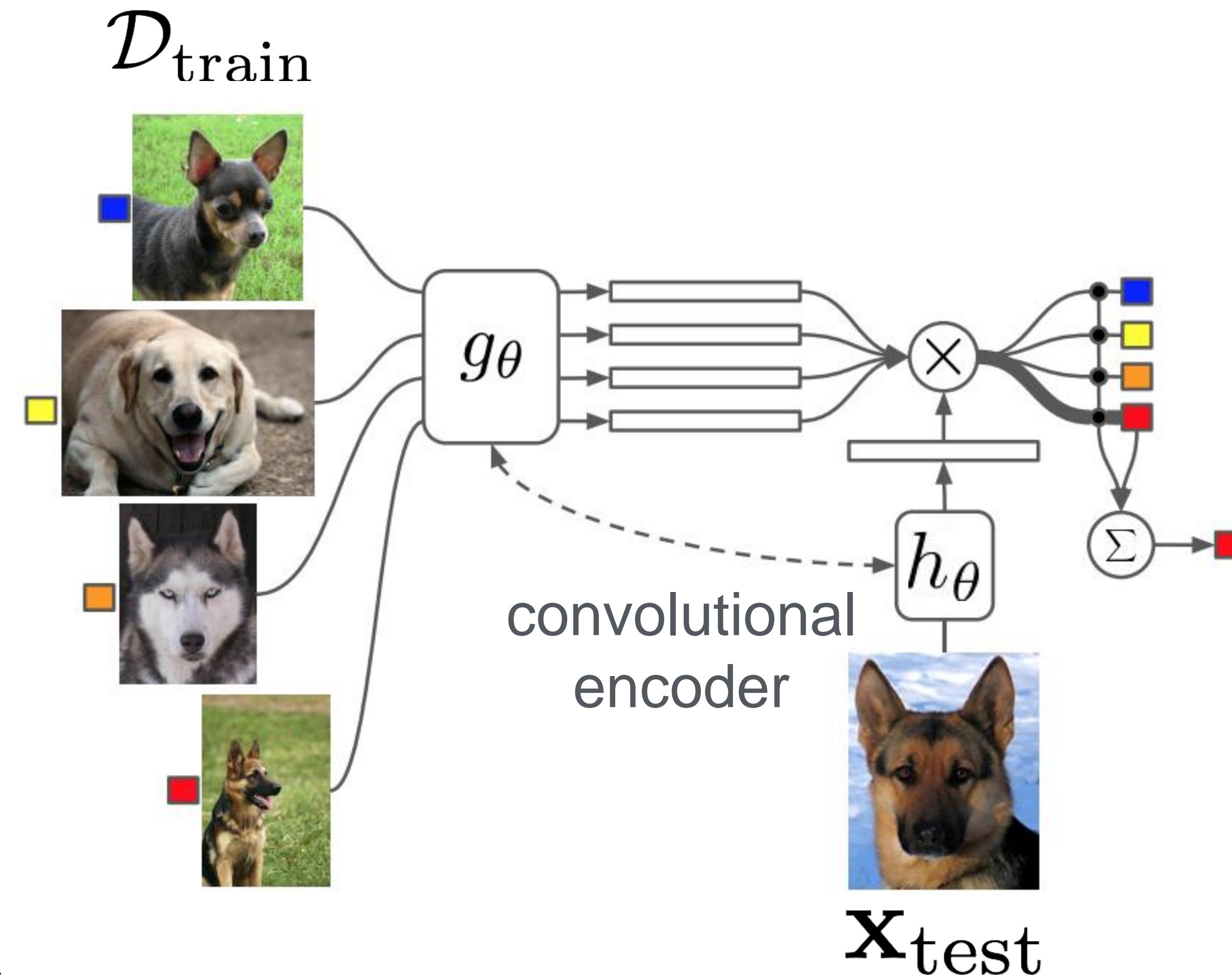
motivating principle: train and test conditions must match



# Matching Networks for One-Shot Learning

Vinyals et al., NIPS '16

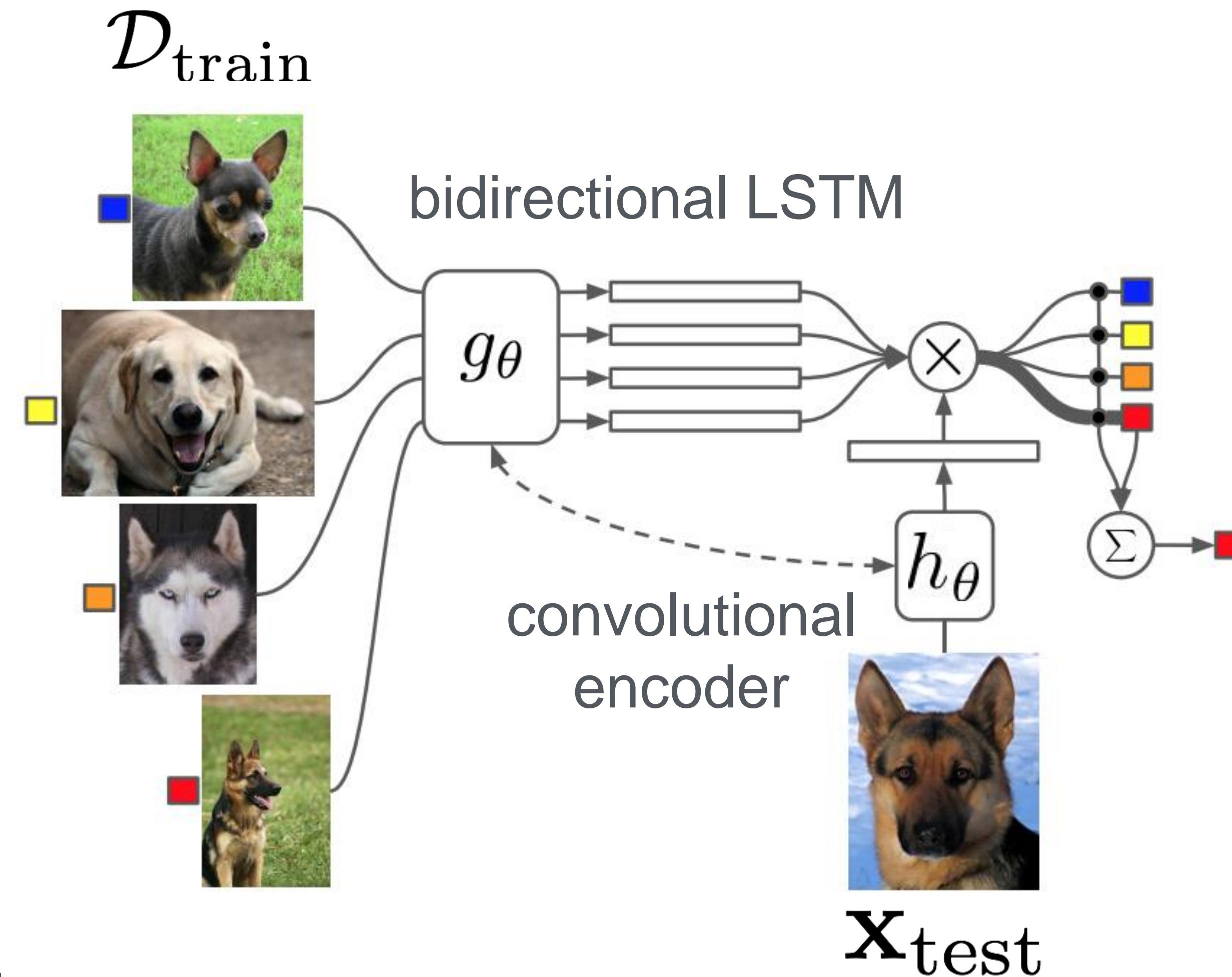
motivating principle: train and test conditions must match



# Matching Networks for One-Shot Learning

Vinyals et al., NIPS '16

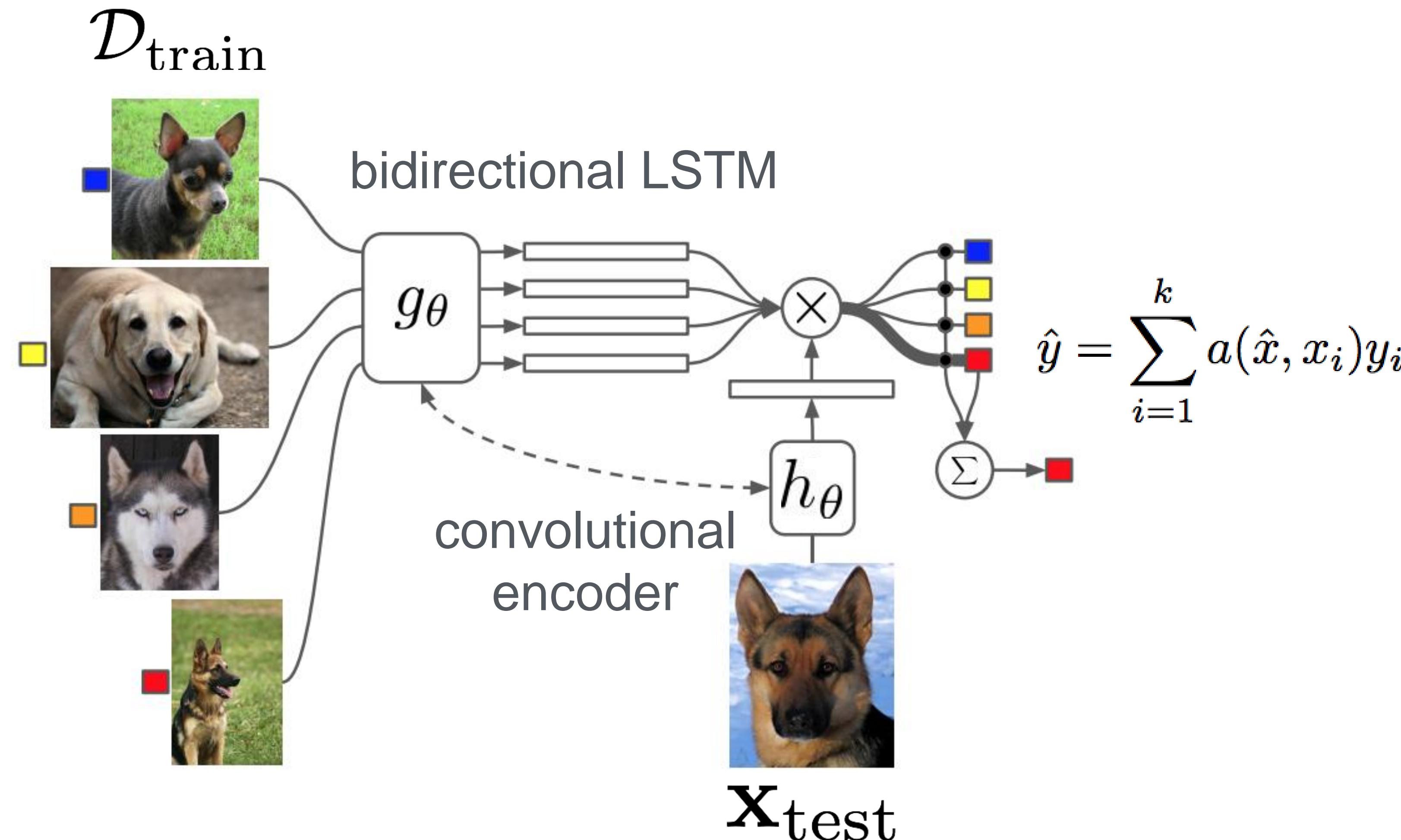
motivating principle: train and test conditions must match



# Matching Networks for One-Shot Learning

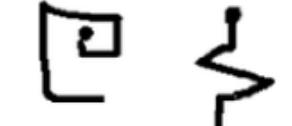
Vinyals et al., NIPS '16

motivating principle: train and test conditions must match



# Few-Shot Classification Results

# Few-Shot Classification Results

 Omniglot dataset  
 Lake et al. '15  
 1623 character classes  
 each written by 20 different people

# Few-Shot Classification Results



Omniglot dataset

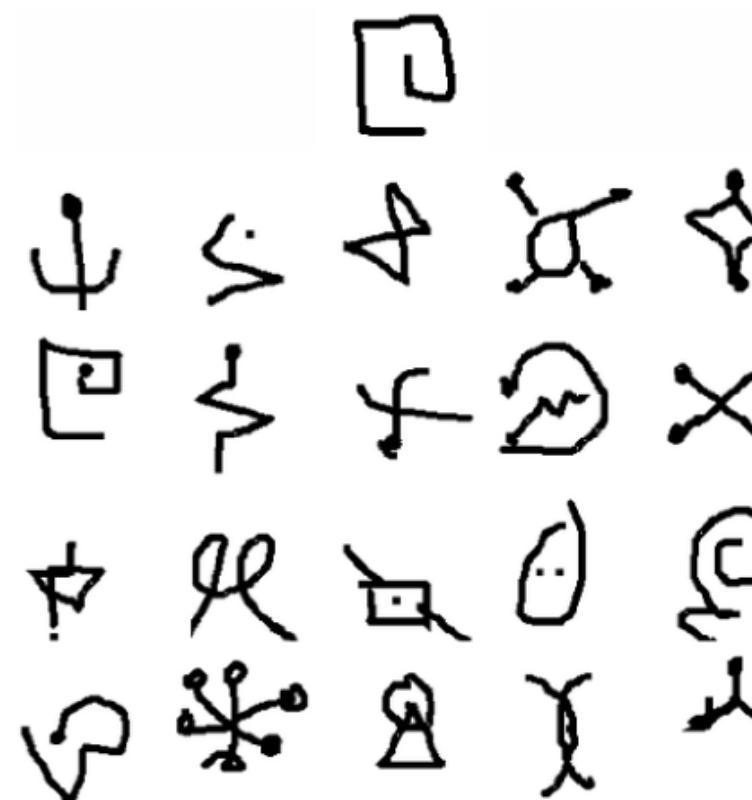
Lake et al. '15

*“MNIST transpose”*

**1623** character classes

each written by **20** different people

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
<b>PIXELS</b>	Cosine	N	41.7%	63.2%	26.7%	42.6%
<b>BASELINE CLASSIFIER</b>	Cosine	N	80.0%	95.0%	69.5%	89.1%
<b>BASELINE CLASSIFIER</b>	Cosine	Y	82.3%	98.4%	70.6%	92.0%
<b>BASELINE CLASSIFIER</b>	Softmax	Y	86.0%	97.6%	72.9%	92.3%
<b>MANN (NO CONV) [21]</b>	Cosine	N	82.8%	94.9%	—	—
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	N	96.7%	98.4%	88.0%	96.5%
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	Y	97.3%	98.4%	88.1%	97.0%
<b>MATCHING NETS (OURS)</b>	Cosine	N	<b>98.1%</b>	<b>98.9%</b>	<b>93.8%</b>	98.5%
<b>MATCHING NETS (OURS)</b>	Cosine	Y	97.9%	98.7%	93.5%	<b>98.7%</b>

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
<b>PIXELS</b>	Cosine	N	41.7%	63.2%	26.7%	42.6%
<b>BASELINE CLASSIFIER</b>	Cosine	N	80.0%	95.0%	69.5%	89.1%
<b>BASELINE CLASSIFIER</b>	Cosine	Y	82.3%	98.4%	70.6%	92.0%
<b>BASELINE CLASSIFIER</b>	Softmax	Y	86.0%	97.6%	72.9%	92.3%
<b>MANN (NO CONV) [21]</b>	Cosine	N	82.8%	94.9%	—	—
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	N	96.7%	98.4%	88.0%	96.5%
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	Y	97.3%	98.4%	88.1%	97.0%
<b>MATCHING NETS (OURS)</b>	Cosine	N	<b>98.1%</b>	<b>98.9%</b>	<b>93.8%</b>	98.5%
<b>MATCHING NETS (OURS)</b>	Cosine	Y	97.9%	98.7%	93.5%	<b>98.7%</b>

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
PIXELS	Cosine	N	41.7%	63.2%	26.7%	42.6%
BASELINE CLASSIFIER	Cosine	N	80.0%	95.0%	69.5%	89.1%
BASELINE CLASSIFIER	Cosine	Y	82.3%	98.4%	70.6%	92.0%
BASELINE CLASSIFIER	Softmax	Y	86.0%	97.6%	72.9%	92.3%
MANN (NO CONV) [21]	Cosine	N	82.8%	94.9%	—	—
CONVOLUTIONAL SIAMESE NET [11]	Cosine	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NET [11]	Cosine	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS (OURS)	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETS (OURS)	Cosine	Y	97.9%	98.7%	93.5%	98.7%

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
<b>PIXELS</b>	Cosine	N	41.7%	63.2%	26.7%	42.6%
<b>BASELINE CLASSIFIER</b>	Cosine	N	80.0%	95.0%	69.5%	89.1%
<b>BASELINE CLASSIFIER</b>	Cosine	Y	82.3%	98.4%	70.6%	92.0%
<b>BASELINE CLASSIFIER</b>	Softmax	Y	86.0%	97.6%	72.9%	92.3%
<b>MANN (NO CONV) [21]</b>	Cosine	N	82.8%	94.9%	—	—
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	N	96.7%	98.4%	88.0%	96.5%
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	Y	97.3%	98.4%	88.1%	97.0%
<b>MATCHING NETS (OURS)</b>	Cosine	N	<b>98.1%</b>	<b>98.9%</b>	<b>93.8%</b>	98.5%
<b>MATCHING NETS (OURS)</b>	Cosine	Y	97.9%	98.7%	93.5%	<b>98.7%</b>

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
<b>PIXELS</b>	Cosine	N	41.7%	63.2%	26.7%	42.6%
<b>BASELINE CLASSIFIER</b>	Cosine	N	80.0%	95.0%	69.5%	89.1%
<b>BASELINE CLASSIFIER</b>	Cosine	Y	82.3%	98.4%	70.6%	92.0%
<b>BASELINE CLASSIFIER</b>	Softmax	Y	86.0%	97.6%	72.9%	92.3%
<b>MANN (NO CONV) [21]</b>	Cosine	N	82.8%	94.9%	—	—
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	N	96.7%	98.4%	88.0%	96.5%
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	Y	97.3%	98.4%	88.1%	97.0%
<b>MATCHING NETS (OURS)</b>	Cosine	N	<b>98.1%</b>	<b>98.9%</b>	<b>93.8%</b>	98.5%
<b>MATCHING NETS (OURS)</b>	Cosine	Y	97.9%	98.7%	93.5%	<b>98.7%</b>

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
<b>PIXELS</b>	Cosine	N	41.7%	63.2%	26.7%	42.6%
<b>BASELINE CLASSIFIER</b>	Cosine	N	80.0%	95.0%	69.5%	89.1%
<b>BASELINE CLASSIFIER</b>	Cosine	Y	82.3%	98.4%	70.6%	92.0%
<b>BASELINE CLASSIFIER</b>	Softmax	Y	86.0%	97.6%	72.9%	92.3%
<b>MANN (NO CONV) [21]</b>	Cosine	N	82.8%	94.9%	—	—
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	N	96.7%	98.4%	88.0%	96.5%
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	Y	97.3%	98.4%	88.1%	97.0%
<b>MATCHING NETS (OURS)</b>	Cosine	N	<b>98.1%</b>	<b>98.9%</b>	<b>93.8%</b>	98.5%
<b>MATCHING NETS (OURS)</b>	Cosine	Y	97.9%	98.7%	93.5%	<b>98.7%</b>

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
PIXELS	Cosine	N	41.7%	63.2%	26.7%	42.6%
BASELINE CLASSIFIER	Cosine	N	80.0%	95.0%	69.5%	89.1%
BASELINE CLASSIFIER	Cosine	Y	82.3%	98.4%	70.6%	92.0%
BASELINE CLASSIFIER	Softmax	Y	86.0%	97.6%	72.9%	92.3%
MANN (NO CONV) [21]	Cosine	N	82.8%	94.9%	—	—
CONVOLUTIONAL SIAMESE NET [11]	Cosine	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NET [11]	Cosine	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS (OURS)	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETS (OURS)	Cosine	Y	97.9%	98.7%	93.5%	98.7%

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

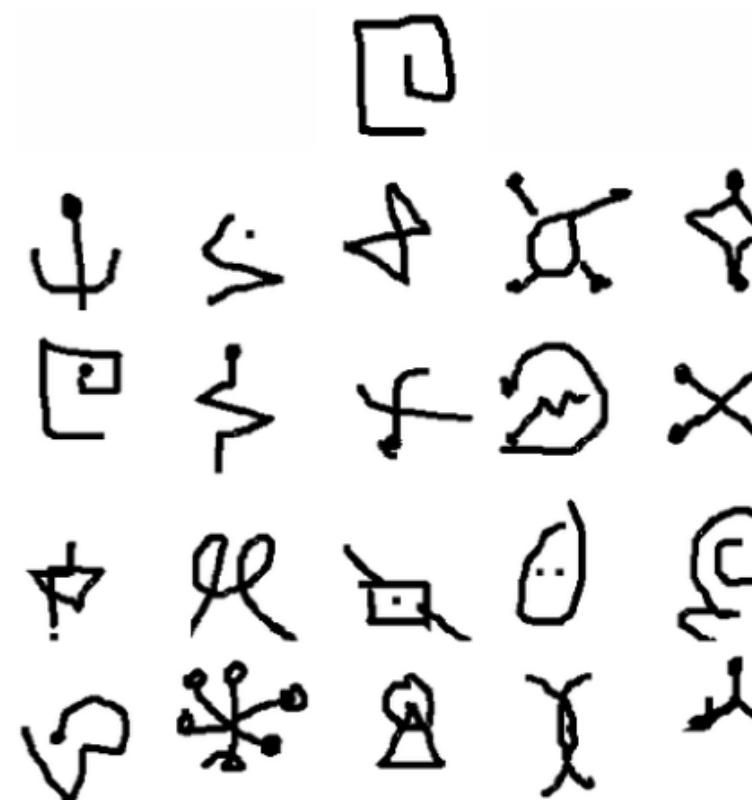
“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
PIXELS	Cosine	N	41.7%	63.2%	26.7%	42.6%
BASELINE CLASSIFIER	Cosine	N	80.0%	95.0%	69.5%	89.1%
BASELINE CLASSIFIER	Cosine	Y	82.3%	98.4%	70.6%	92.0%
BASELINE CLASSIFIER	Softmax	Y	86.0%	97.6%	72.9%	92.3%
MANN (NO CONV) [21]	Cosine	N	82.8%	94.9%	—	—
CONVOLUTIONAL SIAMESE NET [11]	Cosine	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NET [11]	Cosine	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS (OURS)	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETS (OURS)	Cosine	Y	97.9%	98.7%	93.5%	98.7%

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
PIXELS	Cosine	N	41.7%	63.2%	26.7%	42.6%
BASELINE CLASSIFIER	Cosine	N	80.0%	95.0%	69.5%	89.1%
BASELINE CLASSIFIER	Cosine	Y	82.3%	98.4%	70.6%	92.0%
BASELINE CLASSIFIER	Softmax	Y	86.0%	97.6%	72.9%	92.3%
MANN (NO CONV) [21]	Cosine	N	82.8%	94.9%	—	—
CONVOLUTIONAL SIAMESE NET [11]	Cosine	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NET [11]	Cosine	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS (OURS)	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETS (OURS)	Cosine	Y	97.9%	98.7%	93.5%	98.7%

# Few-Shot Classification Results



Omniglot dataset

Lake et al. '15

“MNIST transpose”

1623 character classes

each written by 20 different people

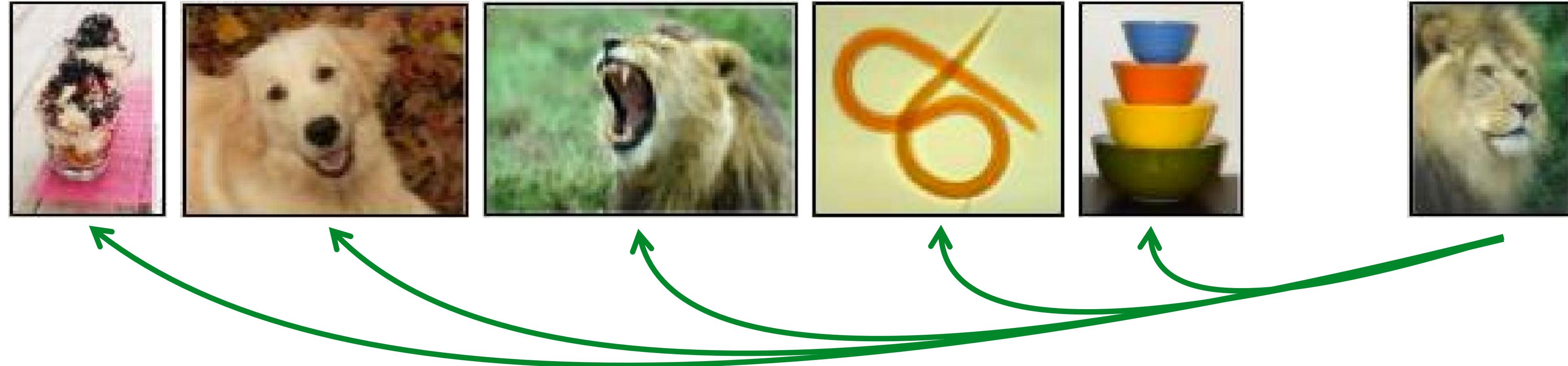
Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
<b>PIXELS</b>	Cosine	N	41.7%	63.2%	26.7%	42.6%
<b>BASELINE CLASSIFIER</b>	Cosine	N	80.0%	95.0%	69.5%	89.1%
<b>BASELINE CLASSIFIER</b>	Cosine	Y	82.3%	98.4%	70.6%	92.0%
<b>BASELINE CLASSIFIER</b>	Softmax	Y	86.0%	97.6%	72.9%	92.3%
<b>MANN (NO CONV) [21]</b>	Cosine	N	82.8%	94.9%	—	—
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	N	96.7%	98.4%	88.0%	96.5%
<b>CONVOLUTIONAL SIAMESE NET [11]</b>	Cosine	Y	97.3%	98.4%	88.1%	97.0%
<b>MATCHING NETS (OURS)</b>	Cosine	N	<b>98.1%</b>	<b>98.9%</b>	<b>93.8%</b>	98.5%
<b>MATCHING NETS (OURS)</b>	Cosine	Y	97.9%	98.7%	93.5%	<b>98.7%</b>

# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$

training data  $\mathcal{D}_{\text{train}}$

test set  $\mathbf{x}_{\text{test}}$



**Key idea:** compare test image with training images  
 $f :=$  nearest neighbors

**Takeaways:**

# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$

training data  $\mathcal{D}_{\text{train}}$



test set  $\mathbf{x}_{\text{test}}$



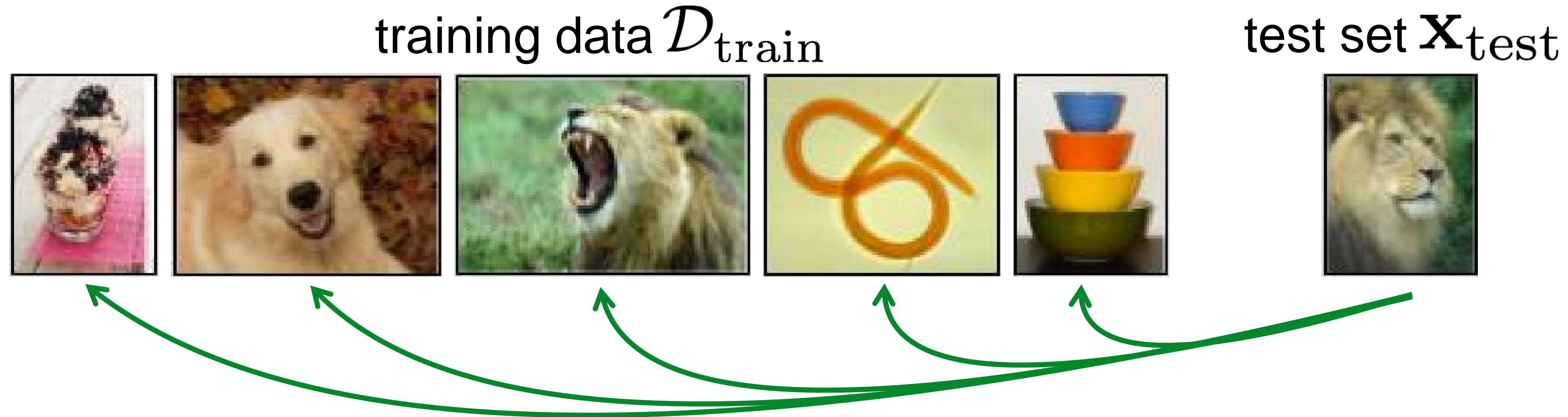
**Key idea:** compare test image with training images  
 $f :=$  nearest neighbors

**Takeaways:**

+ Successful approach for few-shot classification

# Metric Learning Approach

$$\mathcal{D}_{\text{train}} \quad \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$



**Key idea:** compare test image with training images  
 $f :=$  nearest neighbors

## Takeaways:

- + Successful approach for few-shot classification
- Hasn't been demonstrated on non few-shot classification problems

# Outline

1. Applications of learning to learn
2. Problem formulation
3. **Solution Classes:**
  - a) metric-learning approach
  - b) **direct black-box approach**
  - c) gradient-based approach
4. Open Questions / Problems

# Design of $f$ ?

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$

# Design of $f$ ?

Recurrent network  $\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$   
(LSTM, NTM, Conv)

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}}$    $\mathbf{y}_{\text{test}}$

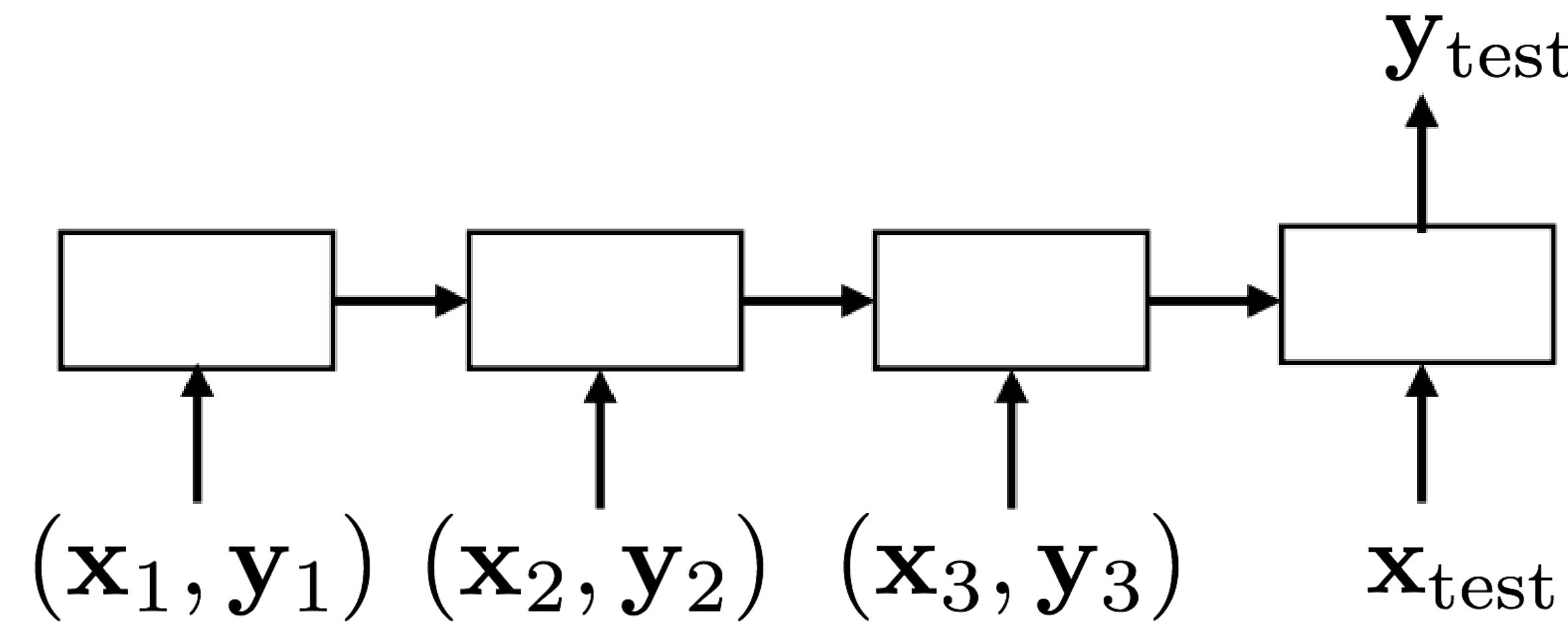
Santoro et al. '16, Duan et al. '17, Wang et al.  
'17,  
Munkhdalai & Yu '17, Mishra et al. '17, ...

# Design of $f$ ?

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$

**Recurrent network**  $\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$   
(LSTM, NTM, Conv)

Santoro et al. '16, Duan et al. '17, Wang et al.  
'17,  
Munkhdalai & Yu '17, Mishra et al. '17, ...



# Design of $f$ ?

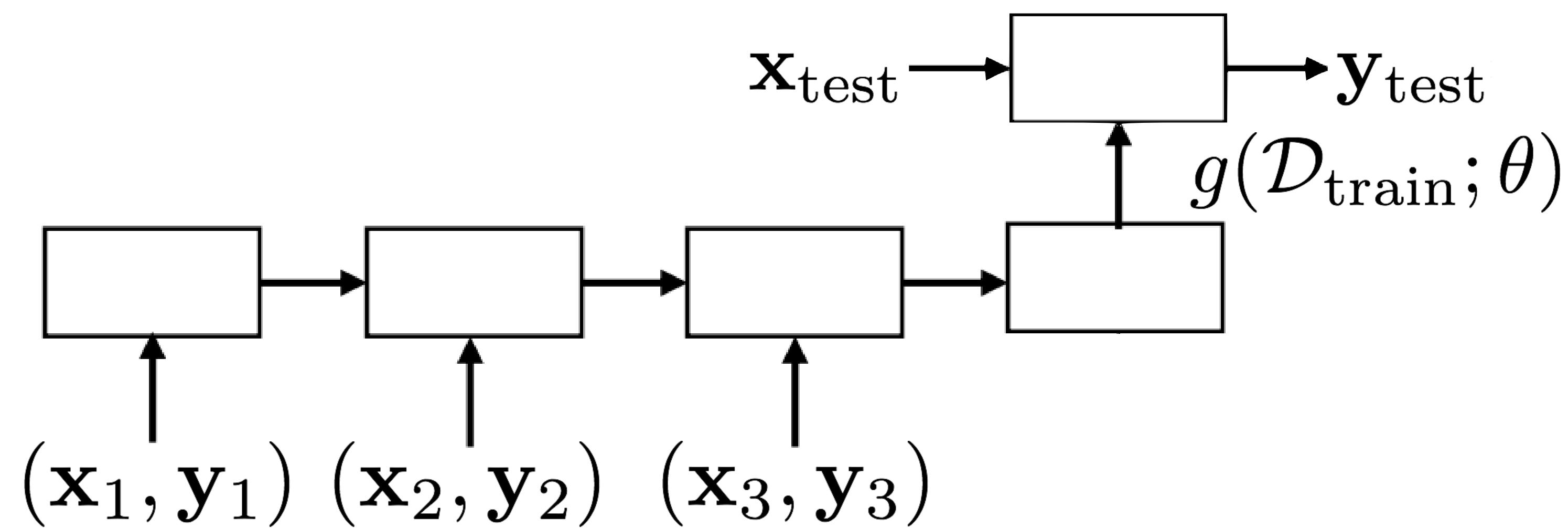
$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$

**Recurrent network**  $\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$   
(LSTM, NTM, Conv)

Santoro et al. '16, Duan et al. '17, Wang et al.  
'17,  
Munkhdalai & Yu '17, Mishra et al. '17, ...

**Learned optimizer**  $\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$   
(often uses recurrence)

Schmidhuber et al. '87, Bengio et al. '90,  
Hochreiter et al. '01, Li & Malik '16,  
Andrychowicz et al. '16, Ha et al. '17, Ravi &  
Larochelle '17, ...



# Case Study: Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

## A SIMPLE NEURAL ATTENTIVE META-LEARNER

**Nikhil Mishra** \*†

UC Berkeley, Department of Electrical Engineering and Computer Science  
Embodied Intelligence

{nmishra, rohaninejadm, c.xi, pabbeel}@berkeley.edu

**Mostafa Rohaninejad**\*

**Xi Chen**†

**Pieter Abbeel**†

# Simple Neural Attentive Meta-Learner

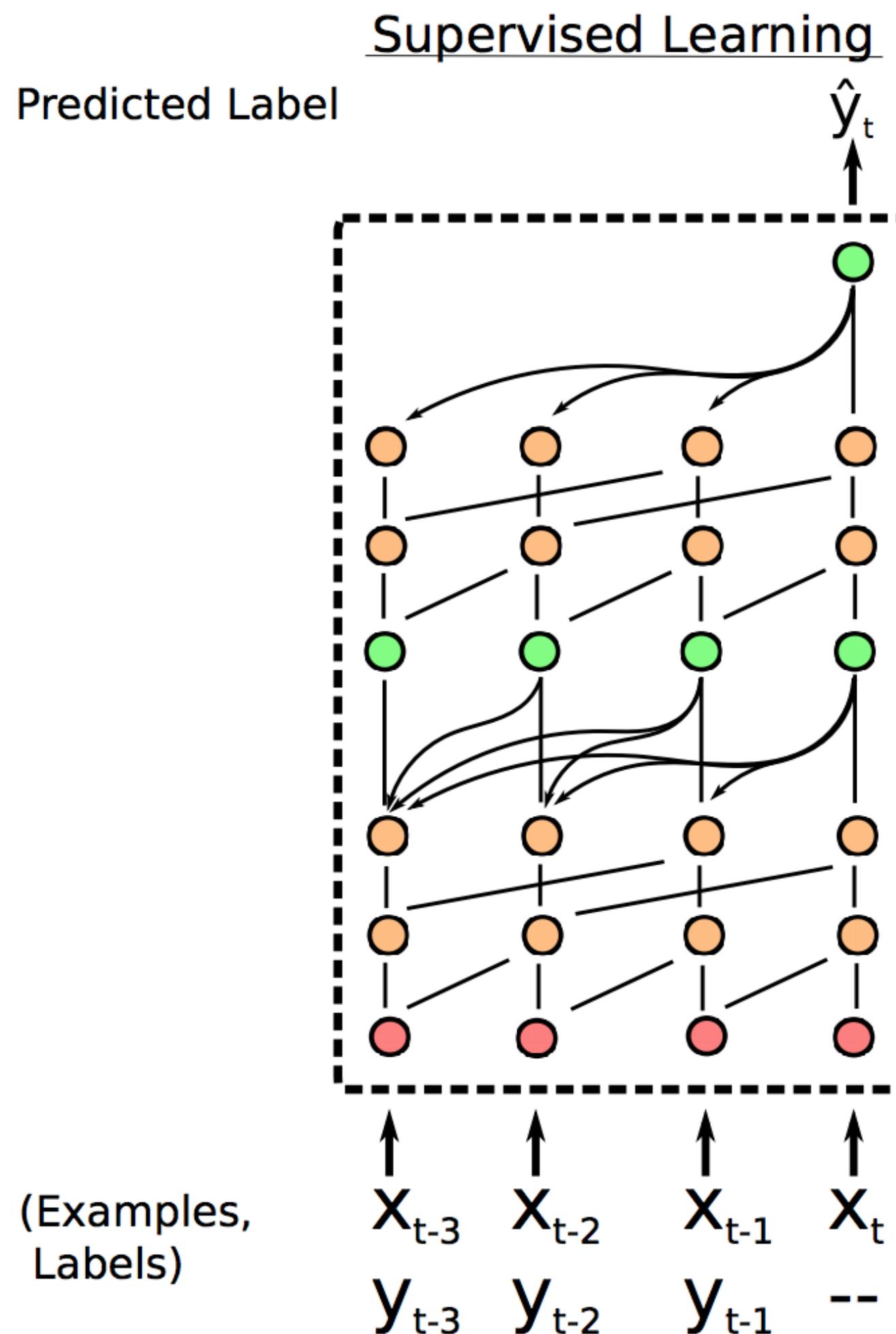
Mishra et al, ICLR '18

**interleave 1D convolutions and attention**

# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

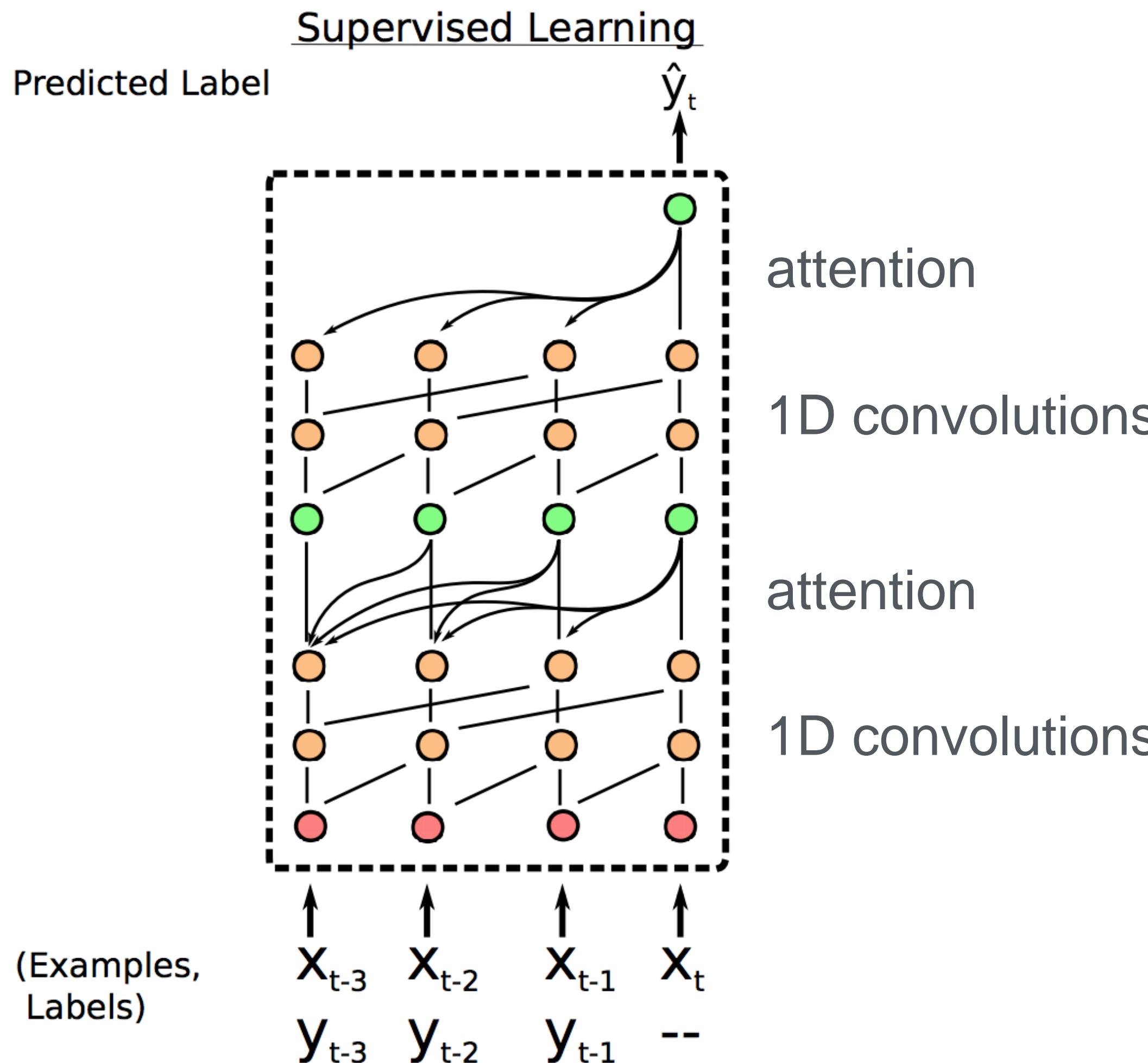
**interleave 1D convolutions and attention**



# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

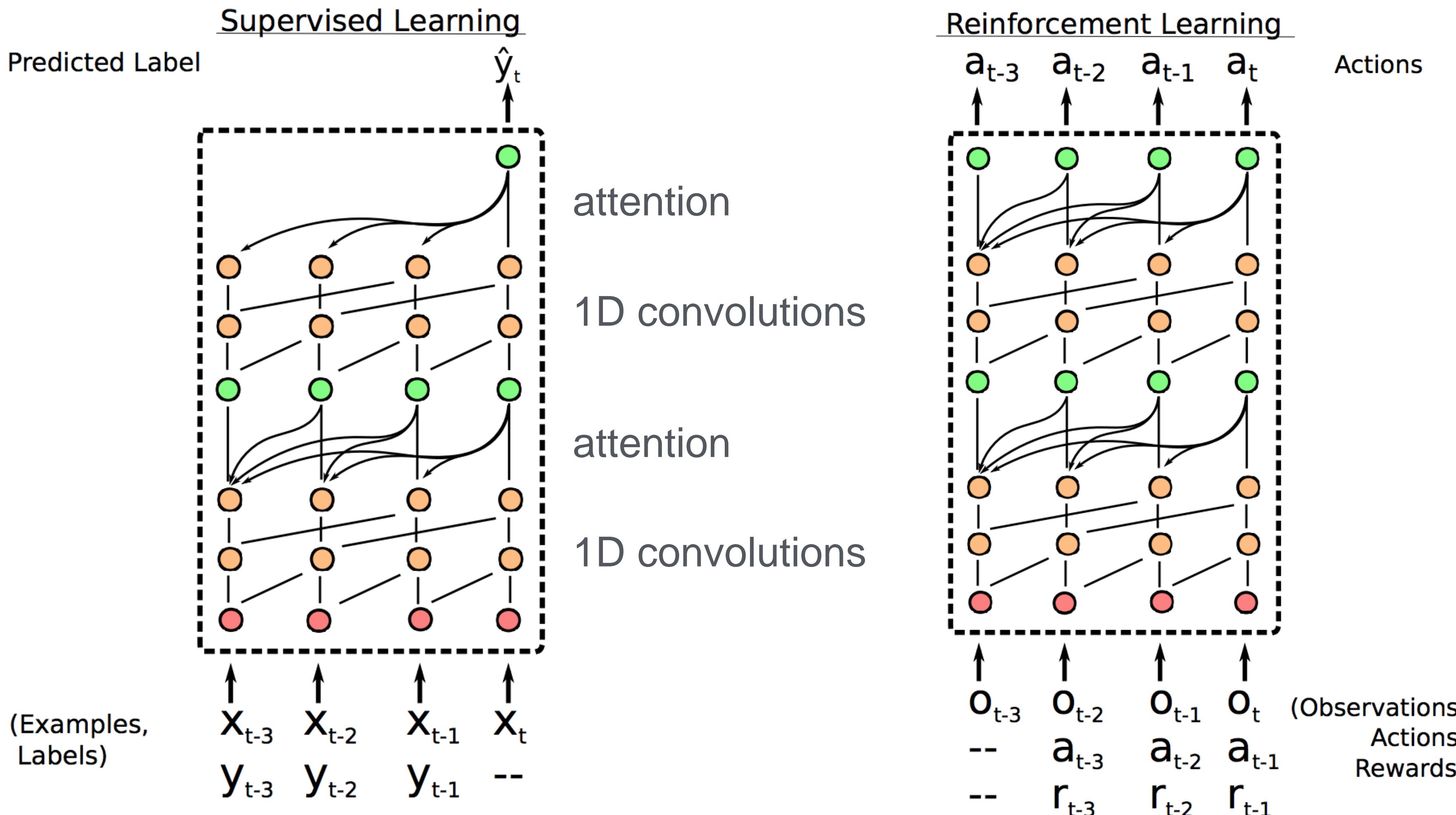
**interleave 1D convolutions and attention**



# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

**interleave 1D convolutions and attention**



# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

**Experiment:** Learning to visually navigate a maze

# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

**Experiment:** Learning to visually navigate a maze

- train on 1000 small mazes

# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

**Experiment:** Learning to visually navigate a maze

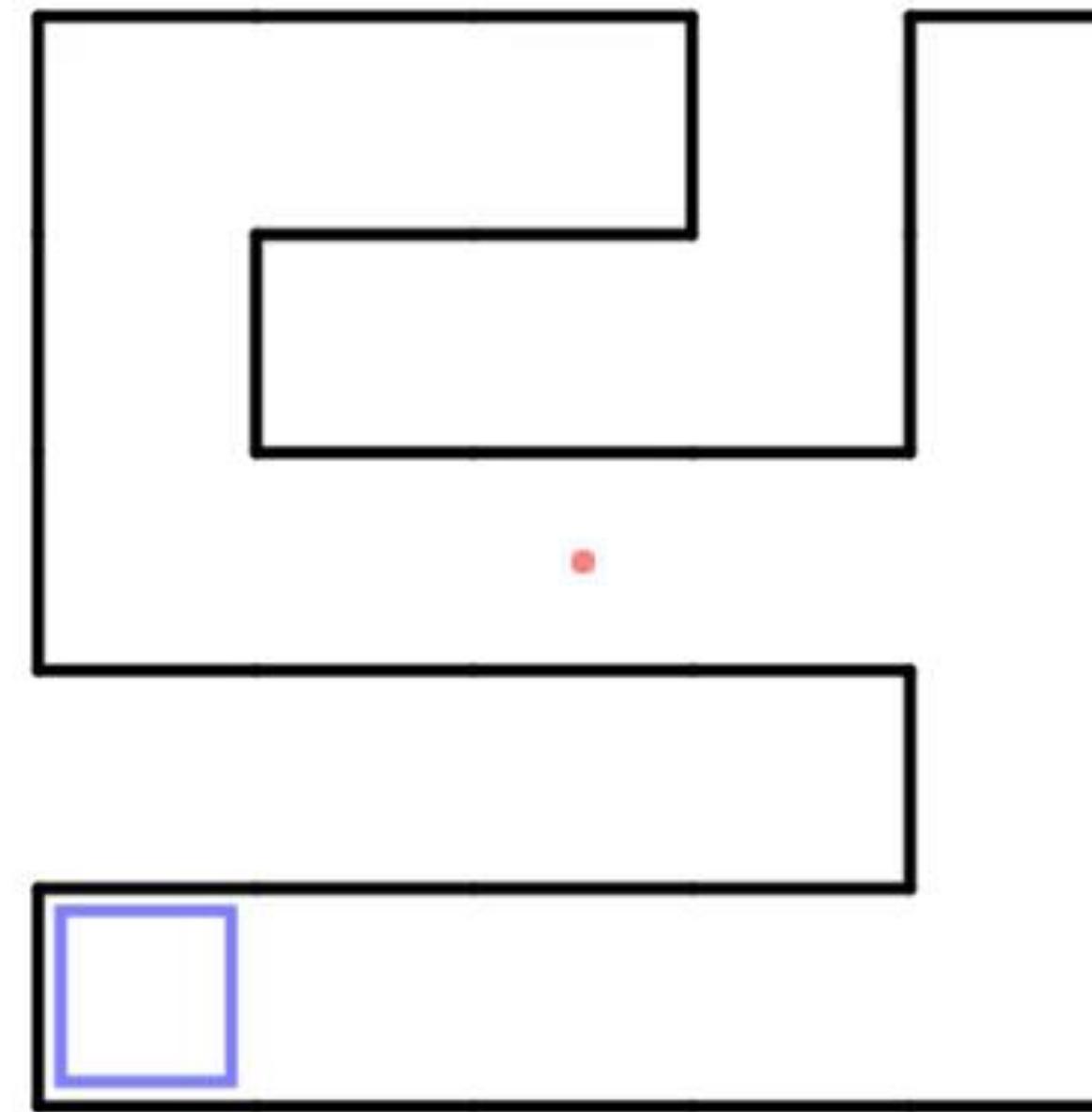
- train on 1000 small mazes
- test on held-out small mazes and large mazes

# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

**Experiment:** Learning to visually navigate a maze

- train on 1000 small mazes
- test on held-out small mazes and large mazes

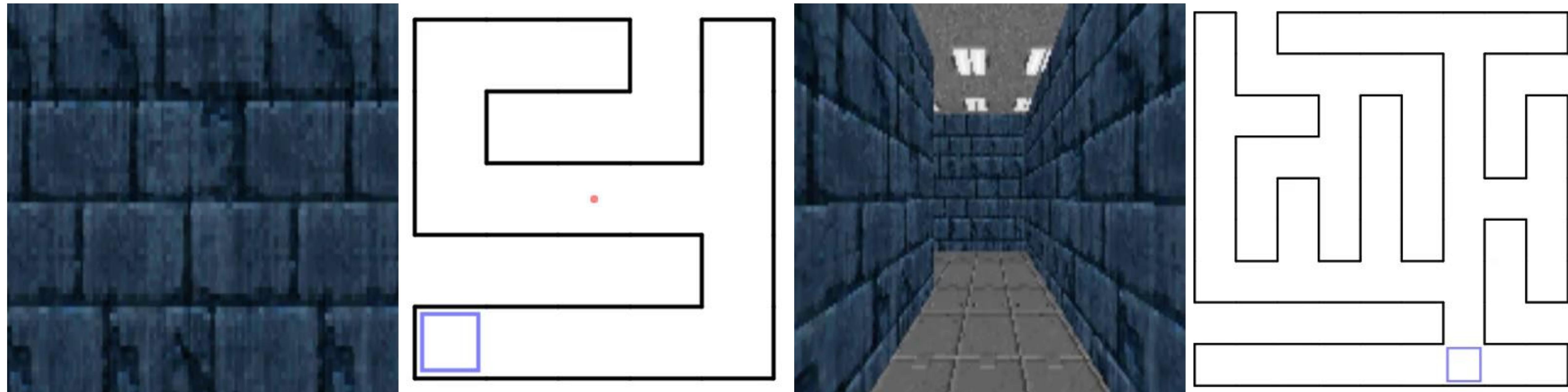


# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

**Experiment:** Learning to visually navigate a maze

- train on 1000 small mazes
- test on held-out small mazes and large mazes



# Simple Neural Attentive Meta-Learner

Mishra et al, ICLR '18

**Experiment:** Learning to visually navigate a maze

- train on 1000 small mazes
- test on held-out small mazes and large mazes

Method	Small Maze		Large Maze	
	Episode 1	Episode 2	Episode 1	Episode 2
Random	$188.6 \pm 3.5$	$187.7 \pm 3.5$	$420.2 \pm 1.2$	$420.8 \pm 1.2$
LSTM	$52.4 \pm 1.3$	$39.1 \pm 0.9$	$180.1 \pm 6.0$	$150.6 \pm 5.9$
SNAIL (ours)	<b><math>50.3 \pm 0.3</math></b>	<b><math>34.8 \pm 0.2</math></b>	<b><math>140.5 \pm 4.2</math></b>	<b><math>105.9 \pm 2.4</math></b>

Table 5: Average time to find the goal on each episode

# Design of $f$ ?

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$

**Recurrent network**  $\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$   
(LSTM, NTM, Conv)

Santoro et al. '16, Duan et al. '17, Wang et al.  
'17,  
Munkhdalai & Yu '17, Mishra et al. '17, ...

**Learned optimizer**  $\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$   
(often uses recurrence)

Schmidhuber et al. '87, Bengio et al. '90,  
Hochreiter et al. '01, Li & Malik '16,  
Andrychowicz et al. '16, Ha et al. '17, Ravi &  
Larochelle '17, ...

# Design of $f$ ?

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$

**Recurrent network**  $\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$   
(LSTM, NTM, Conv)

Santoro et al. '16, Duan et al. '17, Wang et al.  
'17,  
Munkhdalai & Yu '17, Mishra et al. '17, ...

**Learned optimizer**  $\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$   
(often uses recurrence)

Schmidhuber et al. '87, Bengio et al. '90,  
Hochreiter et al. '01, Li & Malik '16,  
Andrychowicz et al. '16, Ha et al. '17, Ravi &  
Larochelle '17, ...

**Takeaways:**

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

**Recurrent network**  $y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$   
(LSTM, NTM, Conv)

Santoro et al. '16, Duan et al. '17, Wang et al.  
'17,  
Munkhdalai & Yu '17, Mishra et al. '17, ...

**Learned optimizer**  $y_{\text{test}} = f(x_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$   
(often uses recurrence)

Schmidhuber et al. '87, Bengio et al. '90,  
Hochreiter et al. '01, Li & Malik '16,  
Andrychowicz et al. '16, Ha et al. '17, Ravi &  
Larochelle '17, ...

## Takeaways:

- + General and powerful approach to meta-learning, has been demonstrated for a range of meta-learning problems

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

**Recurrent network**  $y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$   
(LSTM, NTM, Conv)

Santoro et al. '16, Duan et al. '17, Wang et al.  
'17,  
Munkhdalai & Yu '17, Mishra et al. '17, ...

**Learned optimizer**  $y_{\text{test}} = f(x_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$   
(often uses recurrence)

Schmidhuber et al. '87, Bengio et al. '90,  
Hochreiter et al. '01, Li & Malik '16,  
Andrychowicz et al. '16, Ha et al. '17, Ravi &  
Larochelle '17, ...

## Takeaways:

- + General and powerful approach to meta-learning, has been demonstrated for a range of meta-learning problems
- complex model and complex task (very little structure)

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

**Recurrent network**  $y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$   
(LSTM, NTM, Conv)

Santoro et al. '16, Duan et al. '17, Wang et al.  
'17,  
Munkhdalai & Yu '17, Mishra et al. '17, ...

**Learned optimizer**  $y_{\text{test}} = f(x_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$   
(often uses recurrence)

Schmidhuber et al. '87, Bengio et al. '90,  
Hochreiter et al. '01, Li & Malik '16,  
Andrychowicz et al. '16, Ha et al. '17, Ravi &  
Larochelle '17, ...

## Takeaways:

- + General and powerful approach to meta-learning, has been demonstrated for a range of meta-learning problems
- complex model and complex task (very little structure)
- typically data inefficient

# Outline

1. Applications of learning to learn
2. Problem formulation
3. **Solution Classes:**
  - a) metric-learning approach
  - b) direct black-box approach
  - c) **gradient-based approach**
4. Open Questions / Problems

# Learning Few-Shot Adaptation

# Learning Few-Shot Adaptation

**Fine-tuning**

# Learning Few-Shot Adaptation

Fine-tuning

$\theta$  ↗ pretrained parameters

# Learning Few-Shot Adaptation

Fine-tuning

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{train}}(\theta)$$

pretrained parameters

training data  
for new task

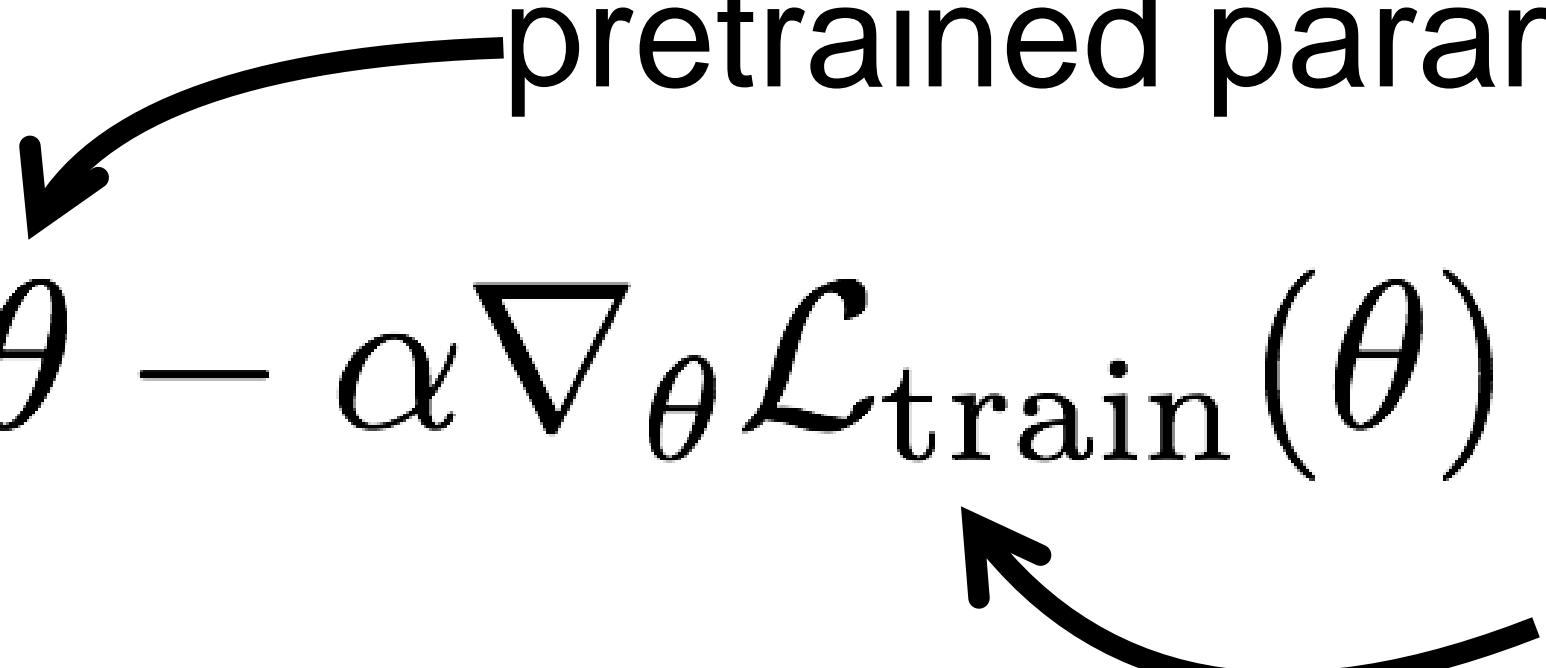
The diagram illustrates the fine-tuning process. It starts with the equation  $\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{train}}(\theta)$ . A curved arrow points from the term  $\theta$  in the equation to the text "pretrained parameters". Another curved arrow points from the term  $\mathcal{L}_{\text{train}}(\theta)$  to the text "training data for new task".

# Learning Few-Shot Adaptation

**Fine-tuning**  
*[test-time]*

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{train}}(\theta)$$

pretrained parameters  
training data  
for new task



# Learning Few-Shot Adaptation

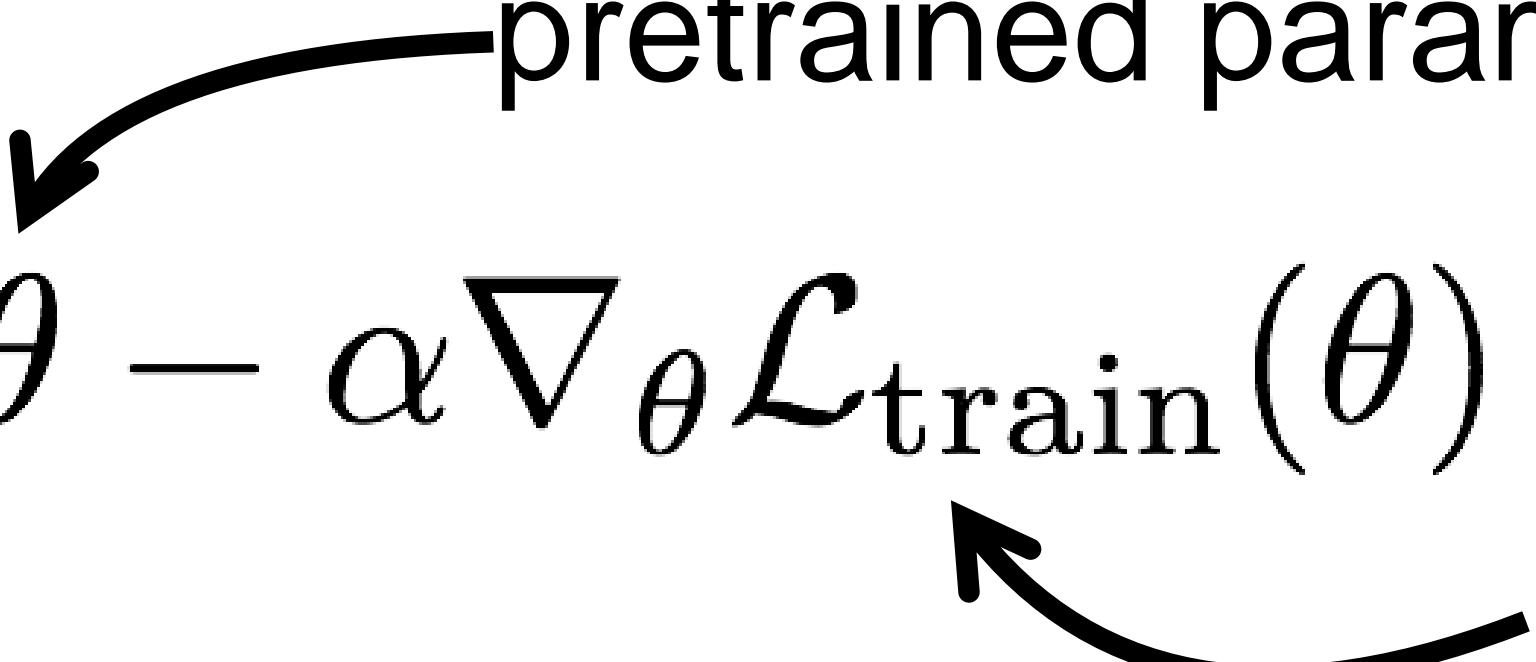
**Fine-tuning**  
*[test-time]*

**Our method**

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{train}}(\theta)$$

pretrained parameters

training data  
for new task



# Learning Few-Shot Adaptation

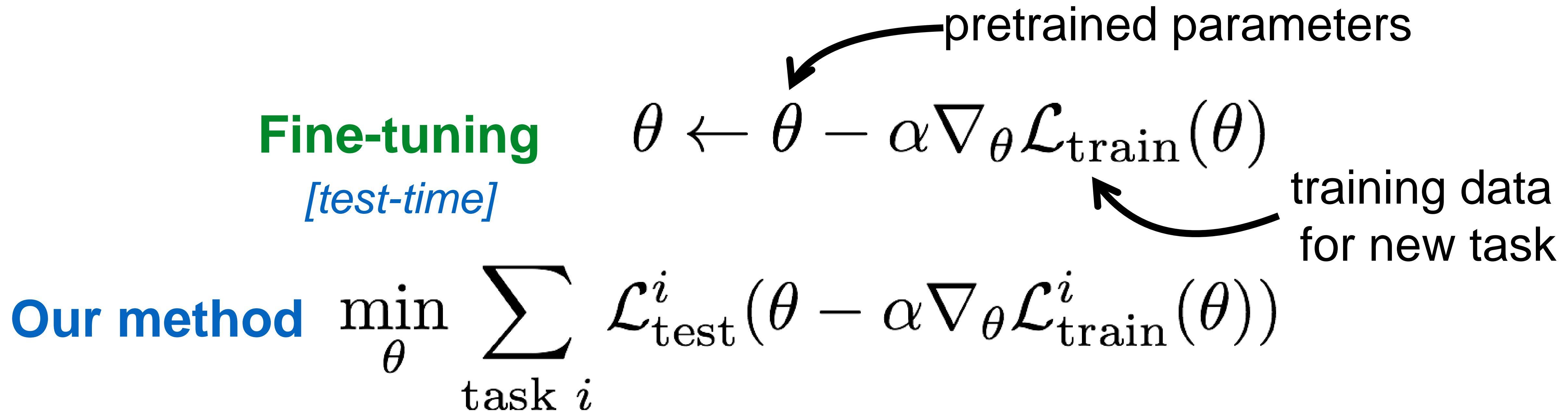
**Fine-tuning**  
*[test-time]*

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{train}}(\theta)$$

pretrained parameters  
training data  
for new task

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

# Learning Few-Shot Adaptation



**Key idea:** Train over many tasks, to learn parameter vector  $\theta$  that transfers

# Learning Few-Shot Adaptation

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

# Learning Few-Shot Adaptation

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

$\theta$  parameter vector  
being meta-learned

# Learning Few-Shot Adaptation

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

$\theta$  parameter vector  
being meta-learned

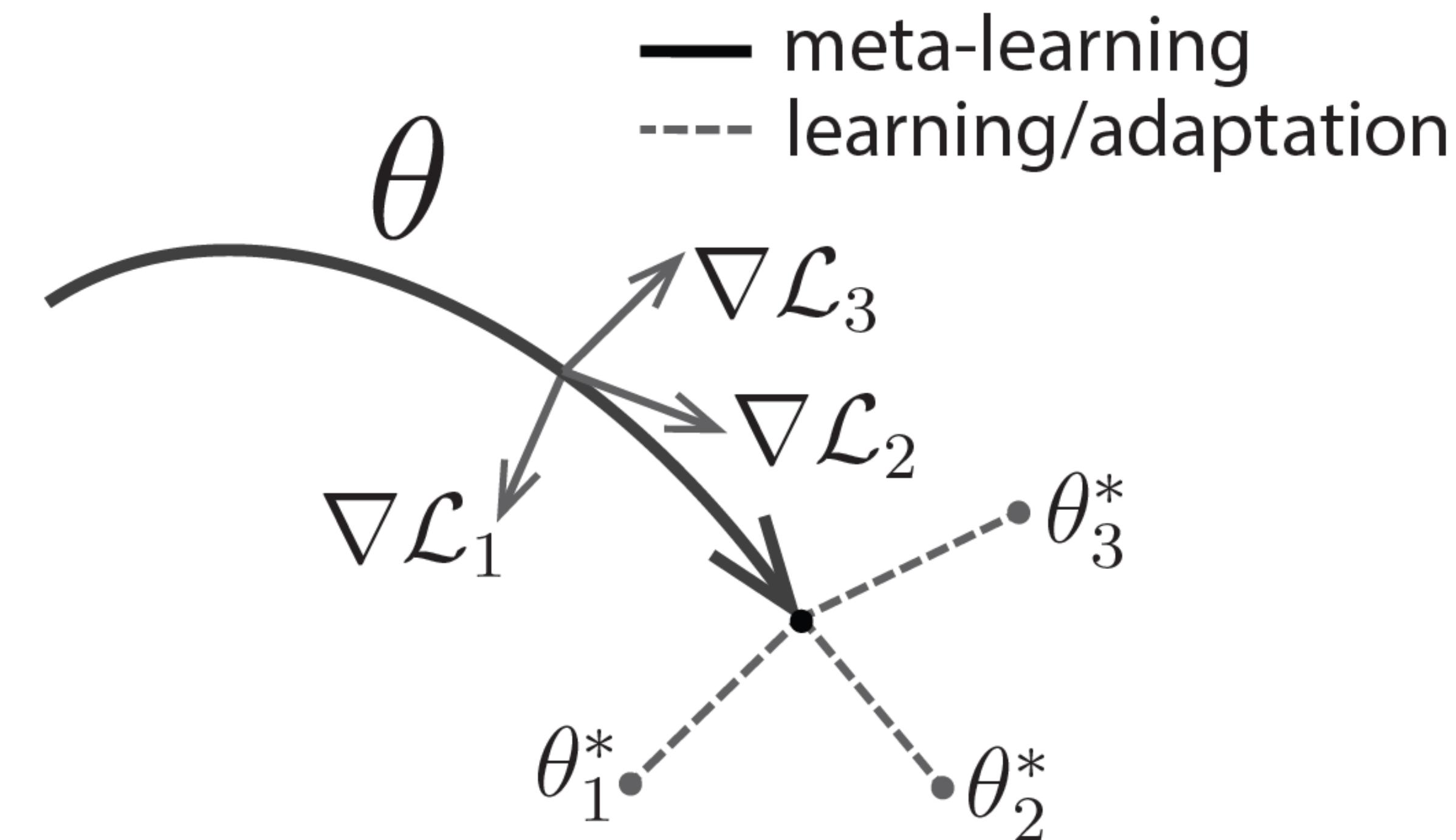
$\theta_i^*$  optimal parameter  
vector for task i

# Learning Few-Shot Adaptation

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

$\theta$  parameter vector  
being meta-learned

$\theta_i^*$  optimal parameter  
vector for task  $i$

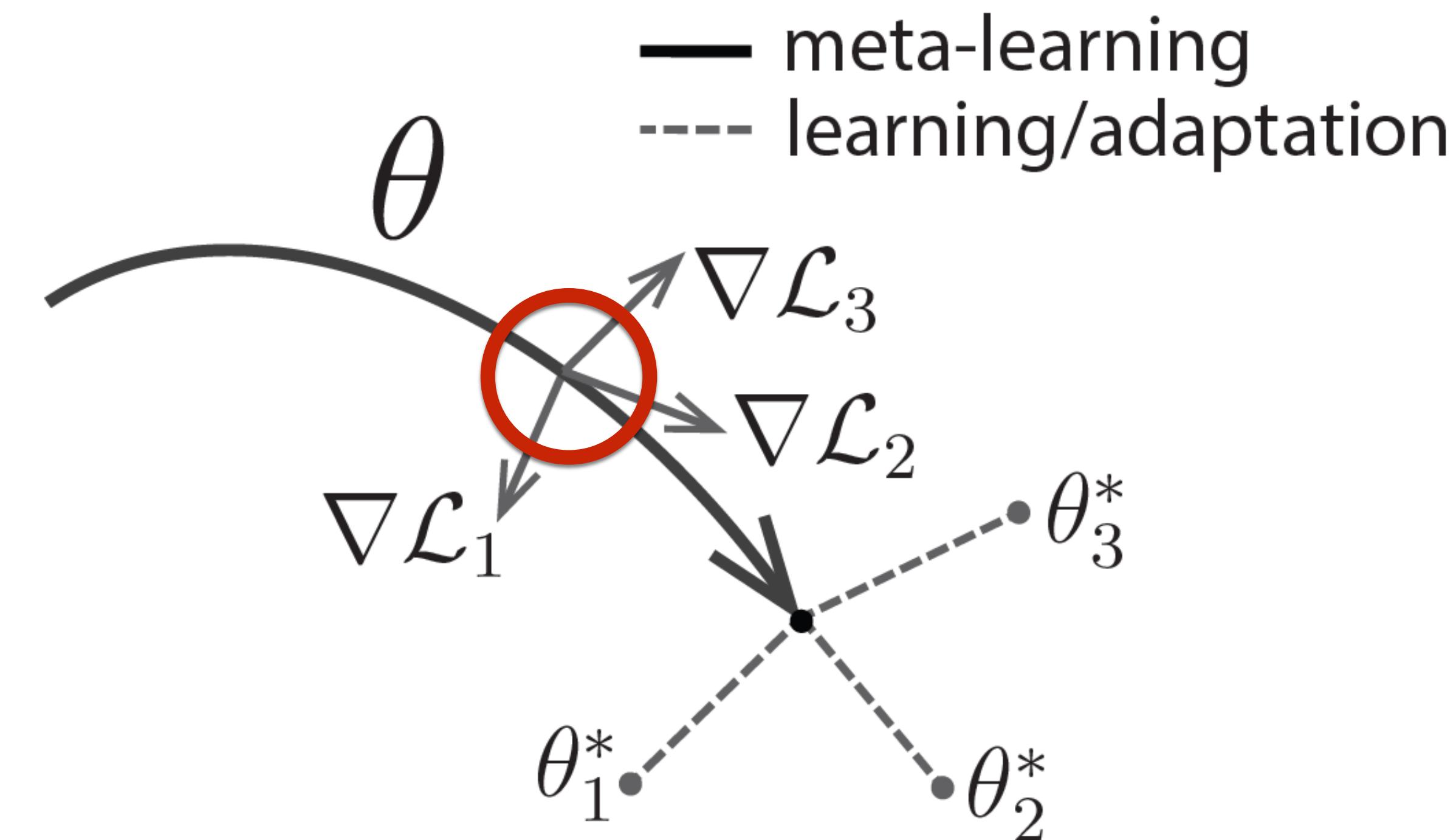


# Learning Few-Shot Adaptation

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

$\theta$  parameter vector  
being meta-learned

$\theta_i^*$  optimal parameter  
vector for task  $i$

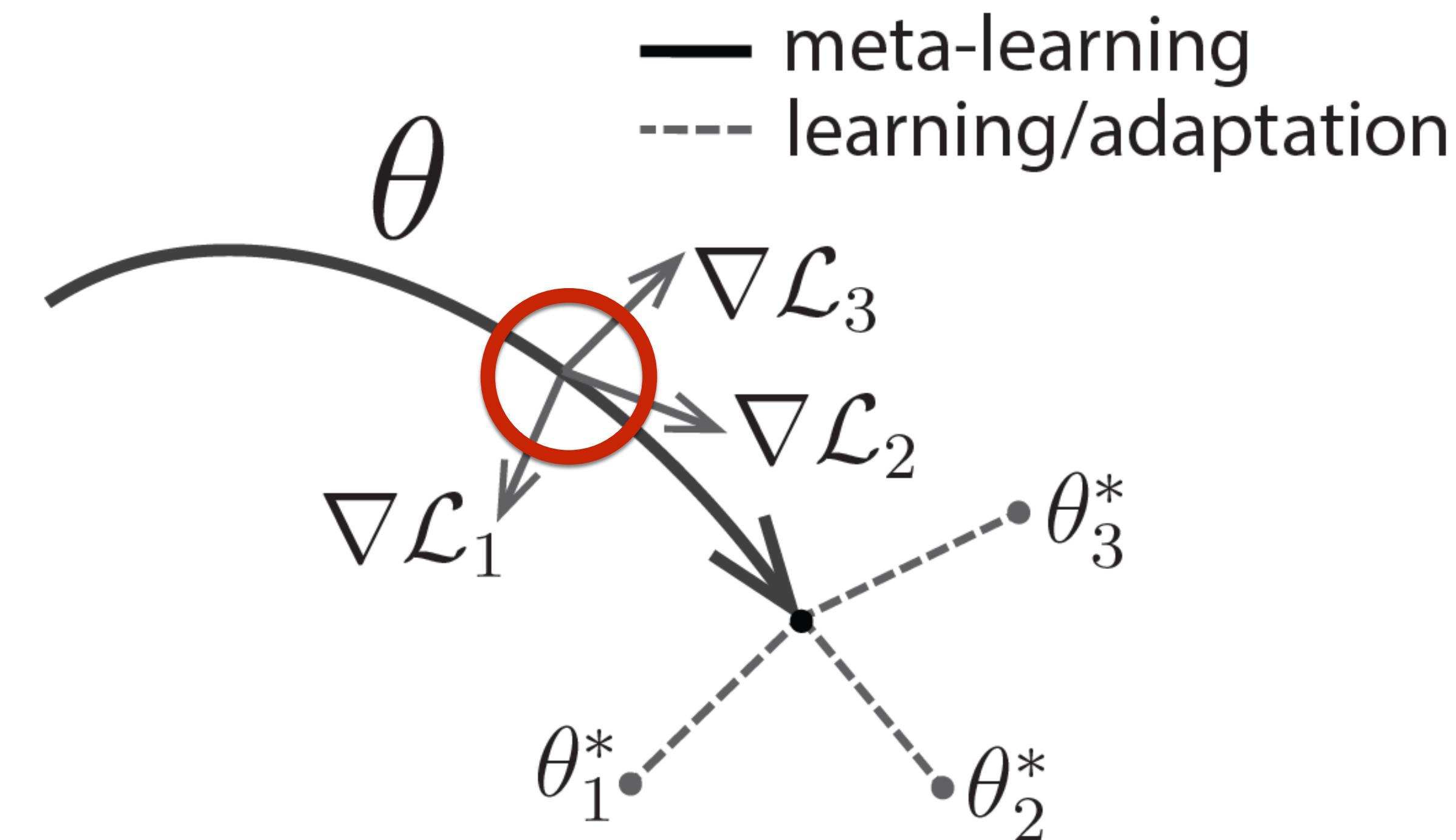


# Learning Few-Shot Adaptation

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

$\theta$  parameter vector  
being meta-learned

$\theta_i^*$  optimal parameter  
vector for task  $i$

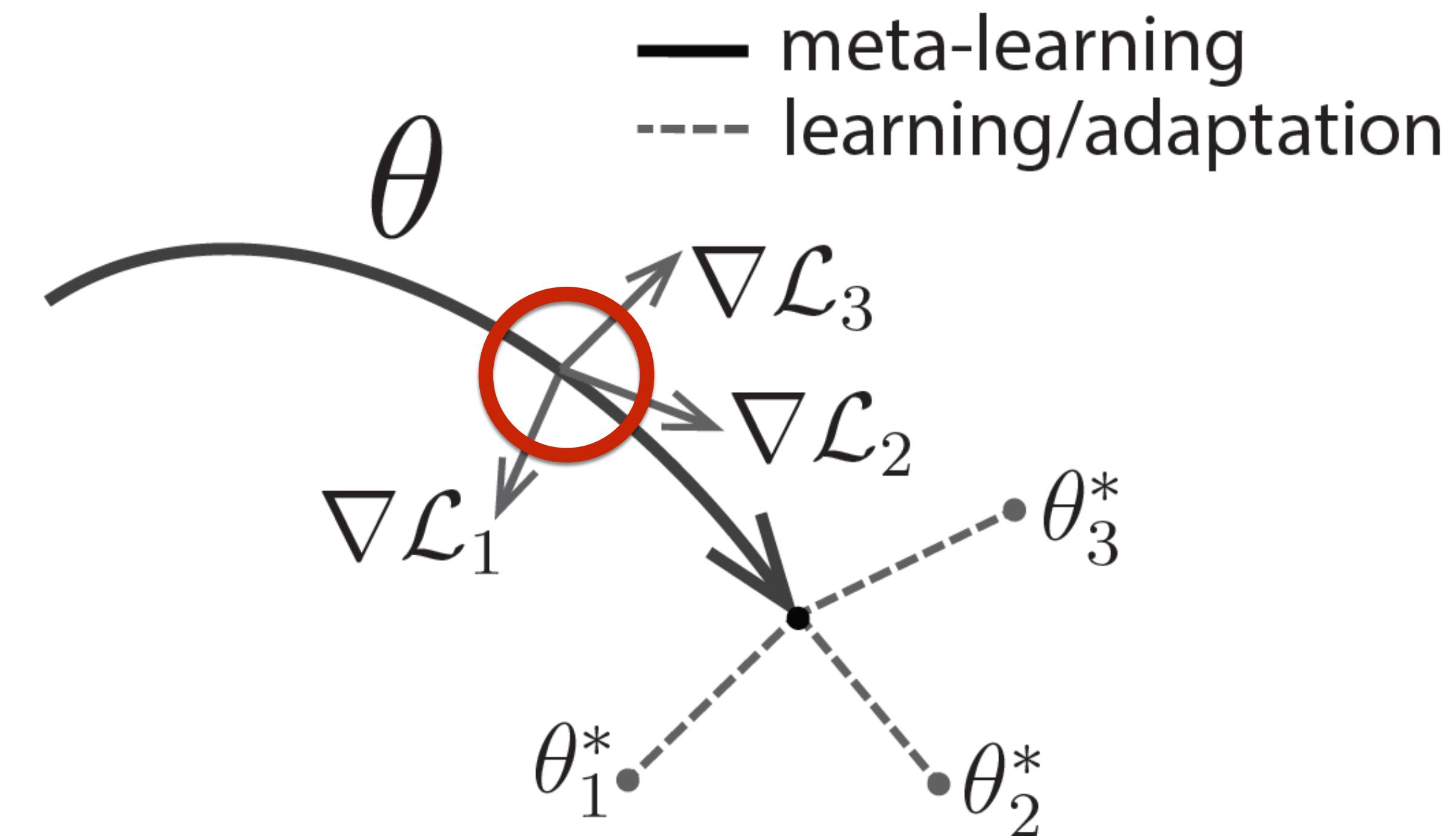


# Learning Few-Shot Adaptation

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

$\theta$  parameter vector  
being meta-learned

$\theta_i^*$  optimal parameter  
vector for task  $i$

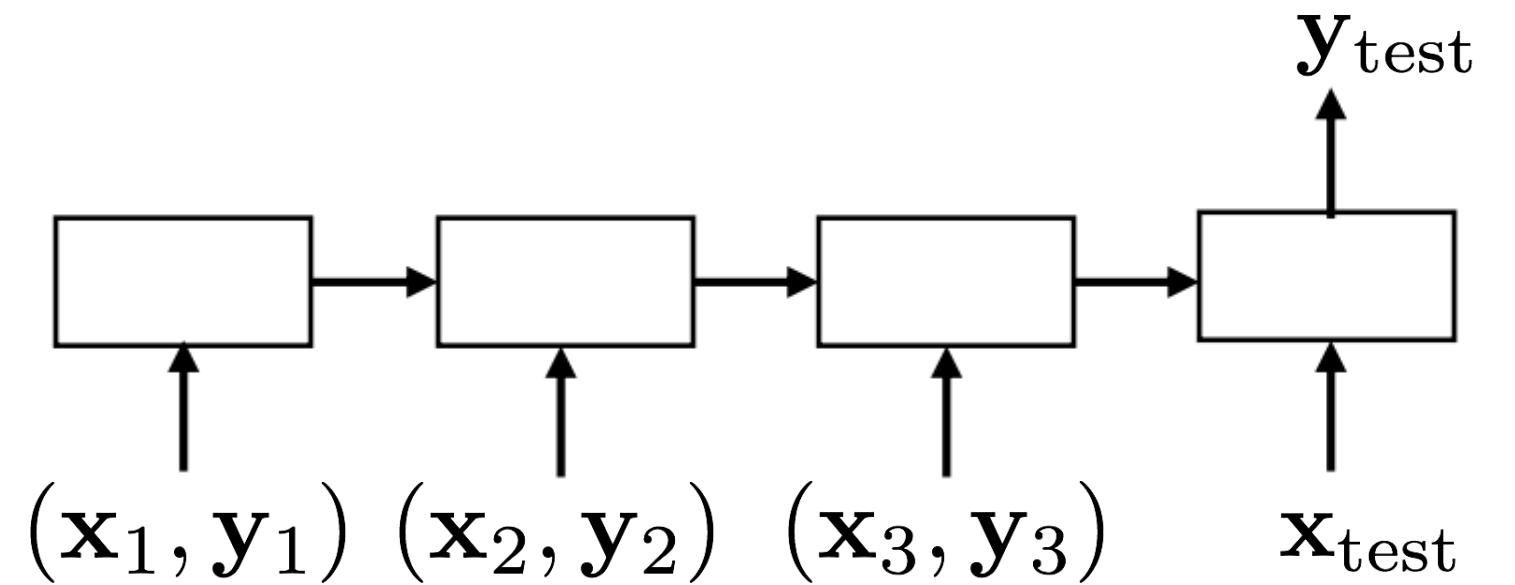


# Design of $f$ ?

$\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$

Recurrent network

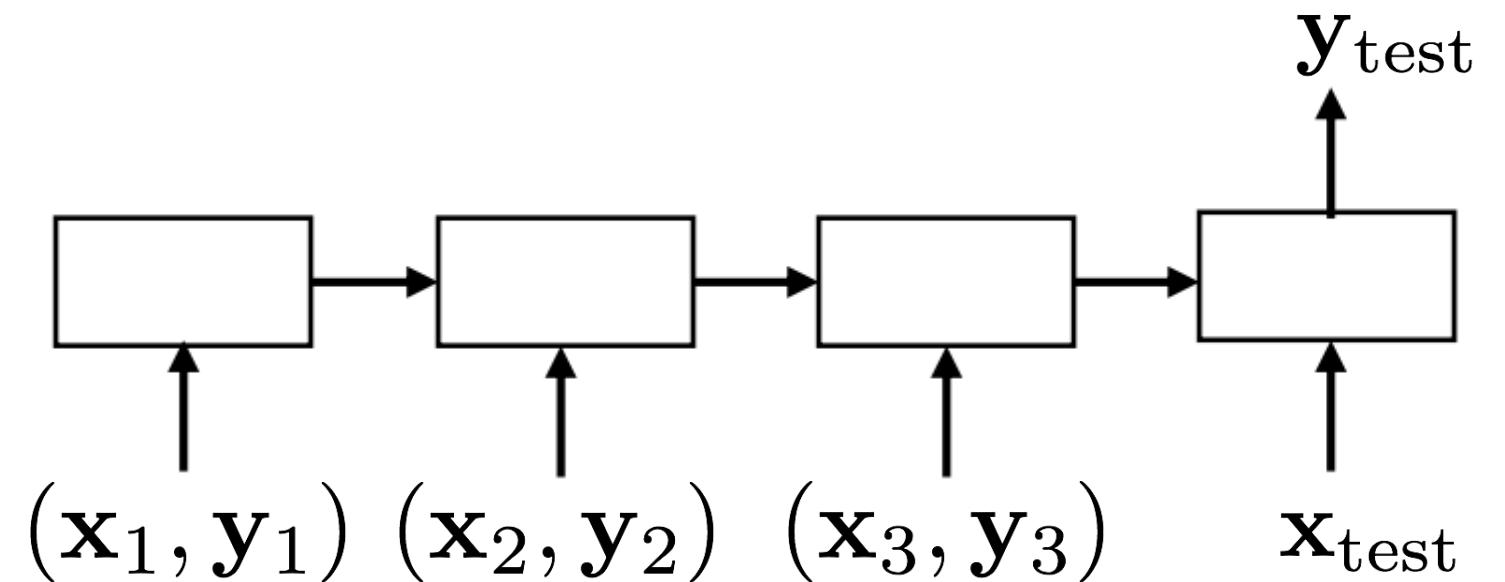
$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$



# Design of $f$ ?

Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$



$$\mathcal{D}_{\text{train}} \ \mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$$

MAML

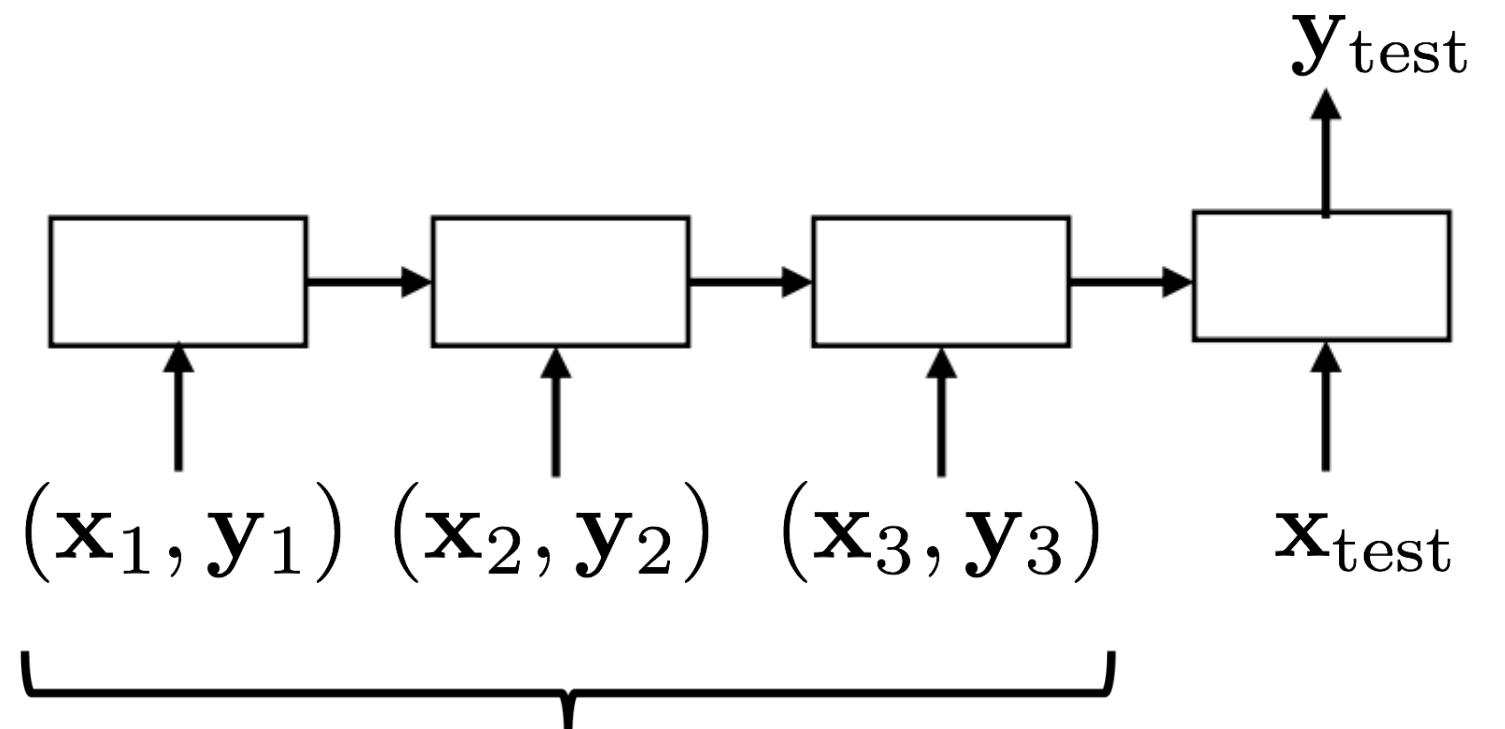
$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

# Design of $f$ ?

$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\text{green arrow}} y_{\text{test}}$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



MAML

$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

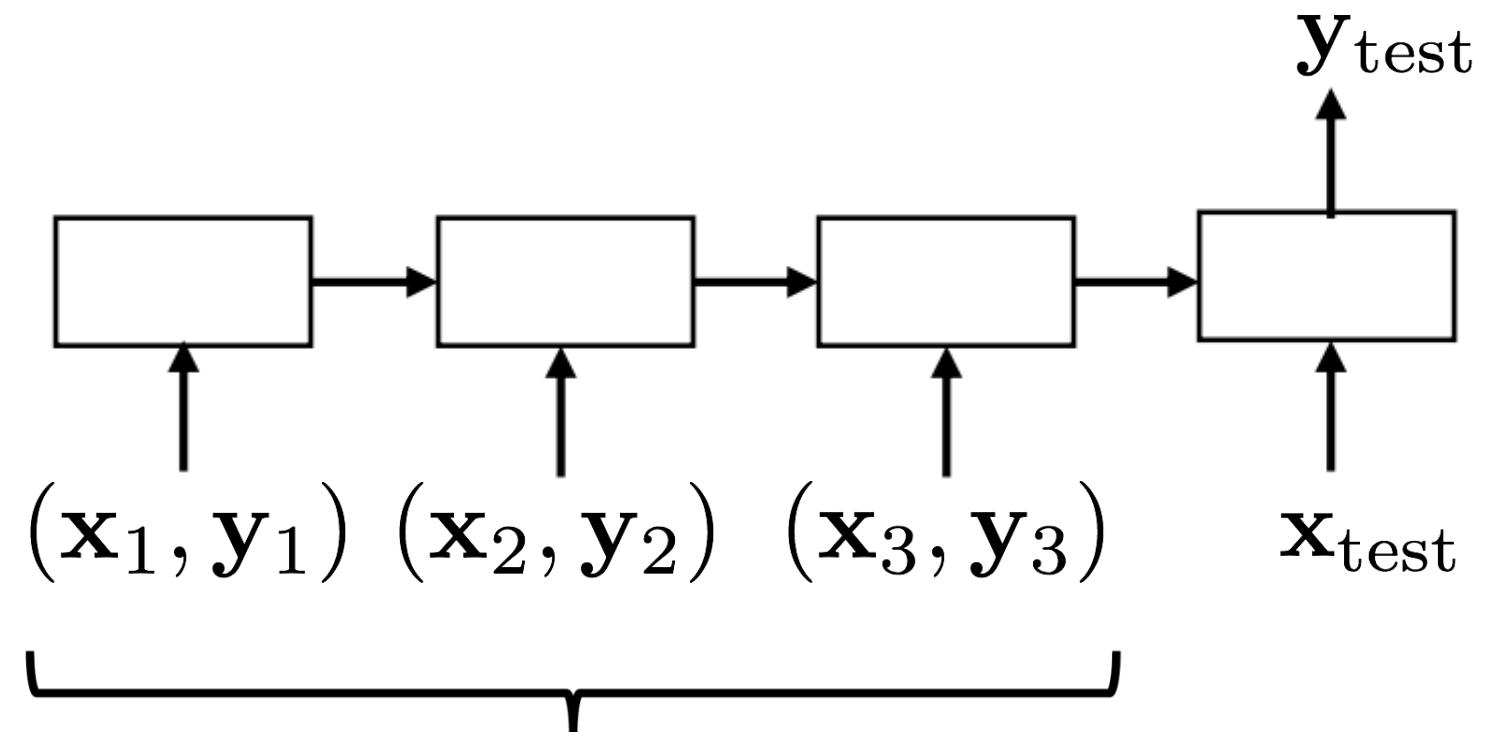
network implements the  
“learned learning procedure”

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



MAML

$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

network implements the  
“learned learning procedure”

Does it converge?

What does it converge to?

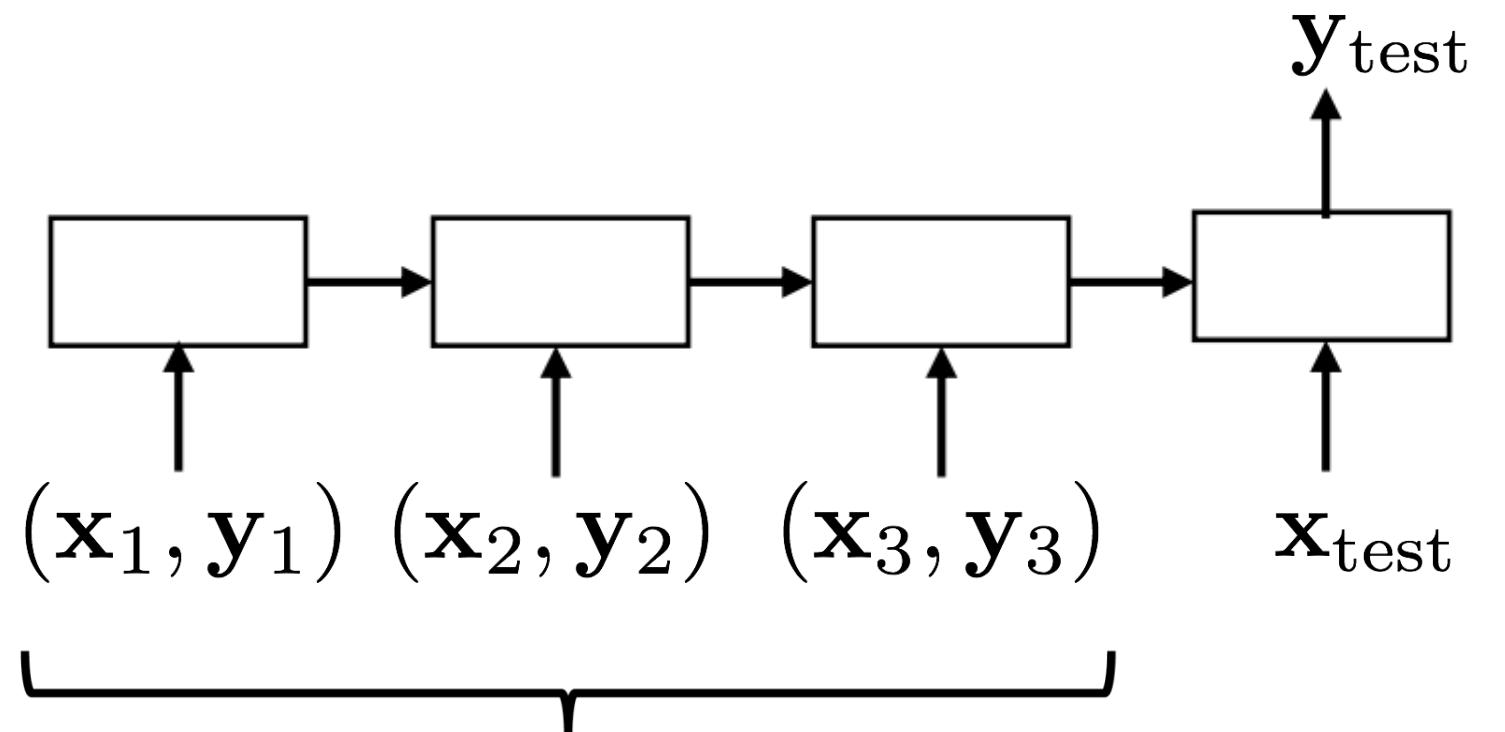
What to do if not good enough?

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

Does it converge?  
- Sort of?

What does it converge to?

What to do if not good enough?

MAML

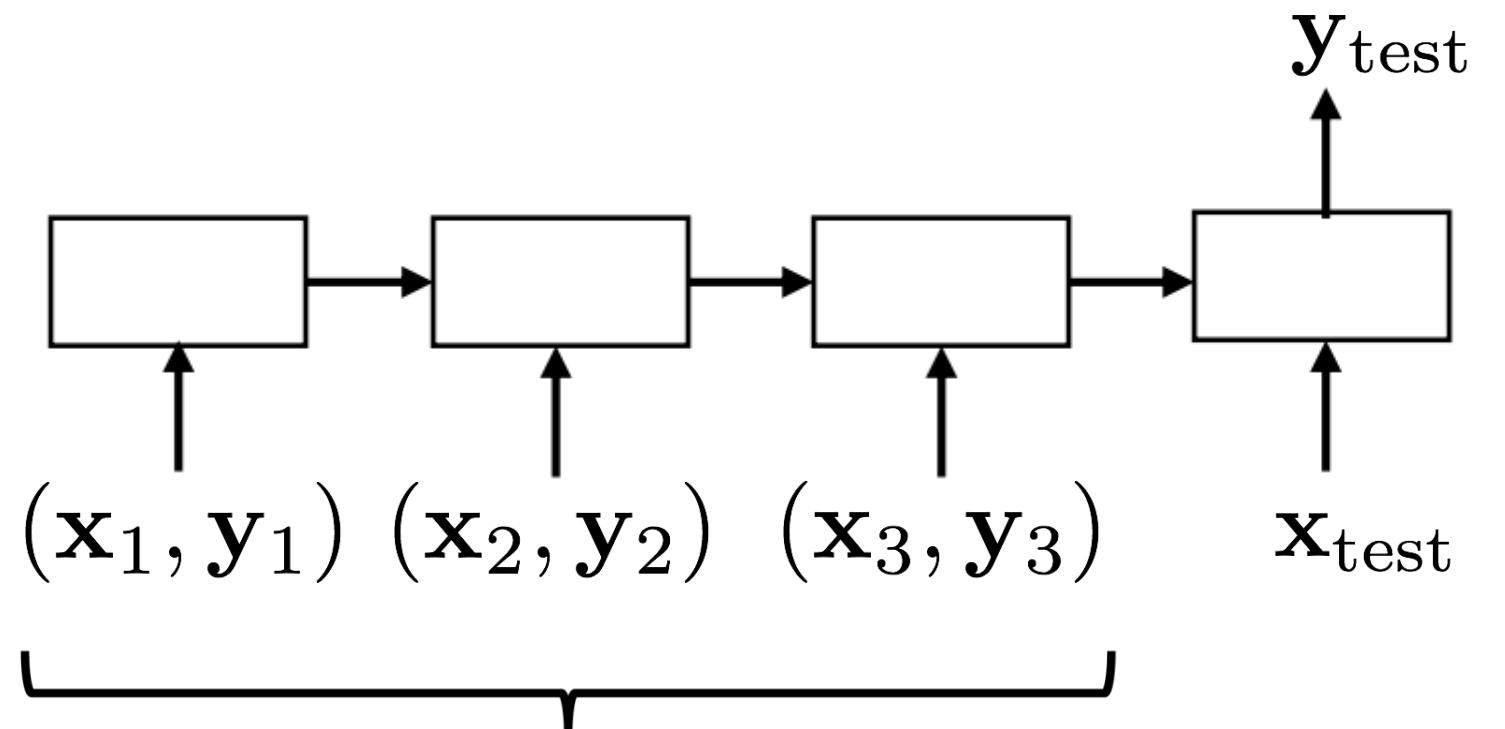
$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

Does it converge?  
- Sort of?

What does it converge to?  
- Who knows...

What to do if not good enough?

MAML

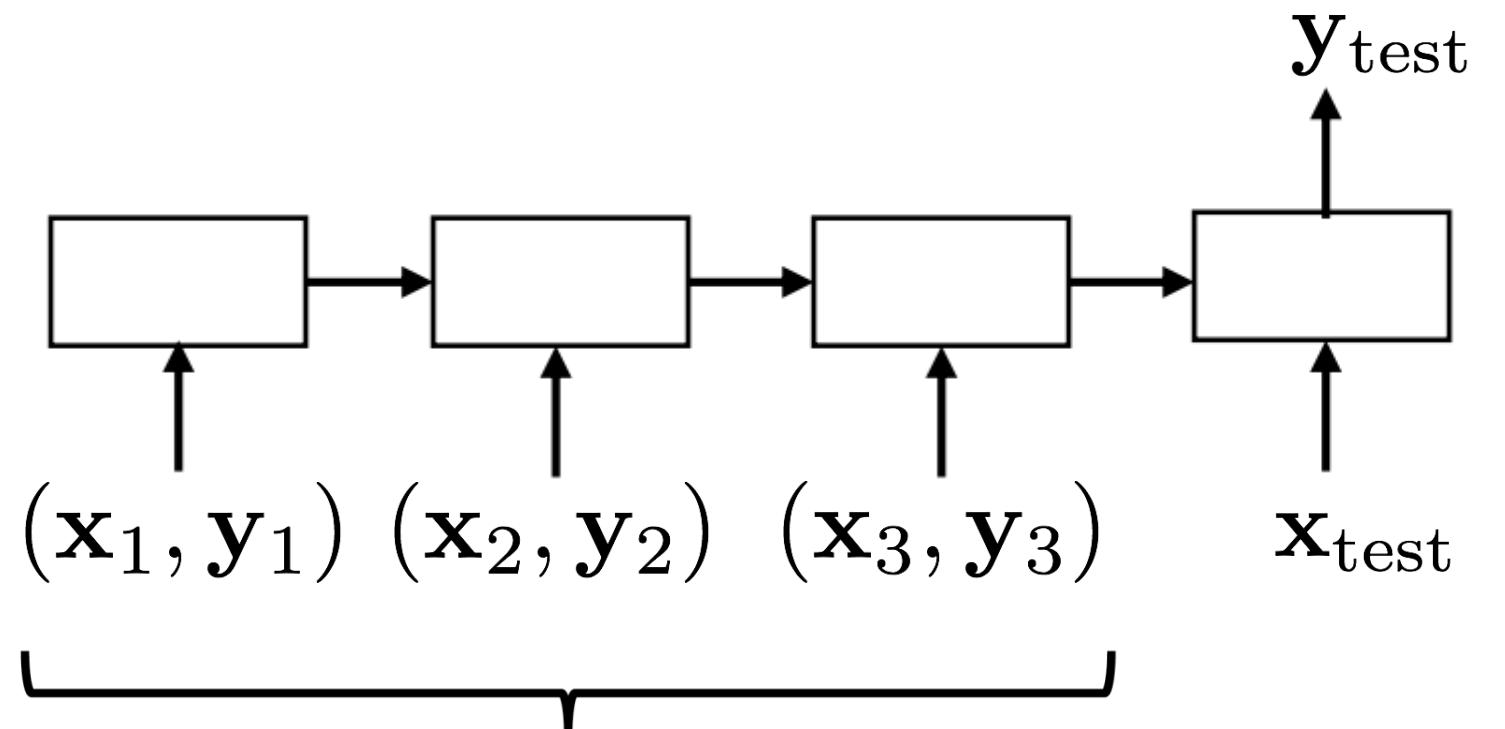
$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\text{green arrow}} y_{\text{test}}$$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



MAML

$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

network implements the  
“learned learning procedure”

Does it converge?  
- Sort of?

What does it converge to?  
- Who knows...

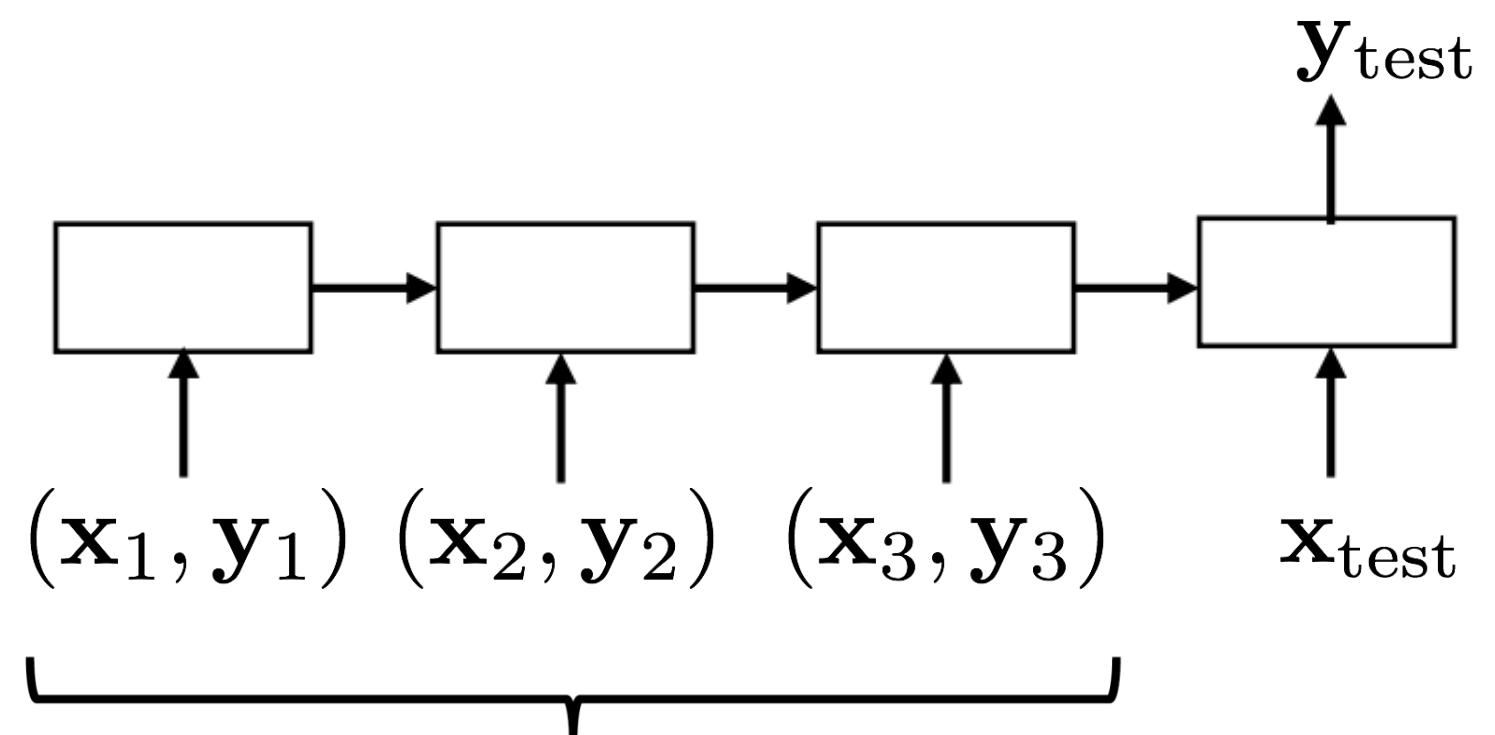
What to do if not good enough?  
- Nothing

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

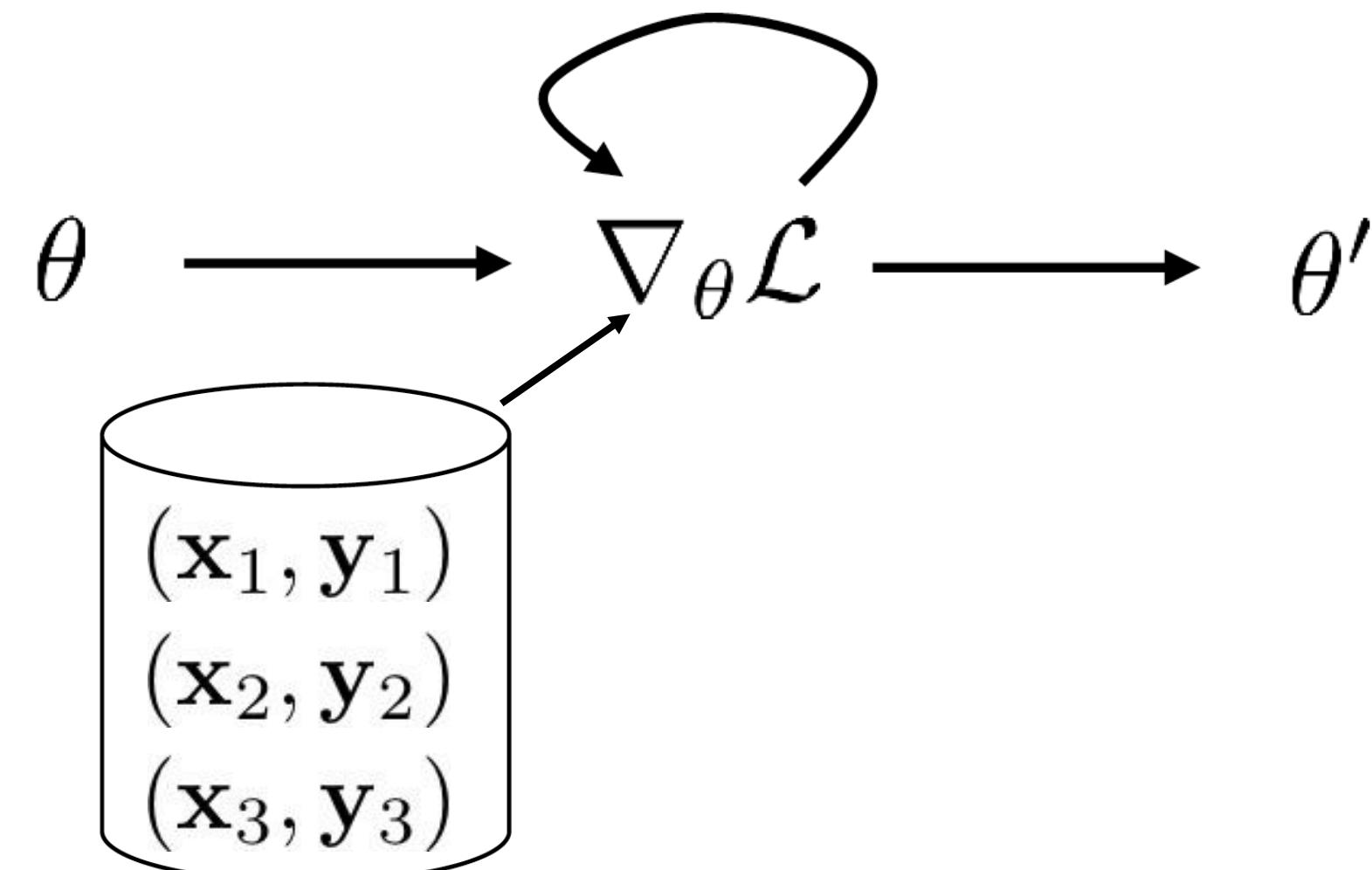
Does it converge?  
- Sort of?

What does it converge to?  
- Who knows...

What to do if not good enough?  
- Nothing

MAML

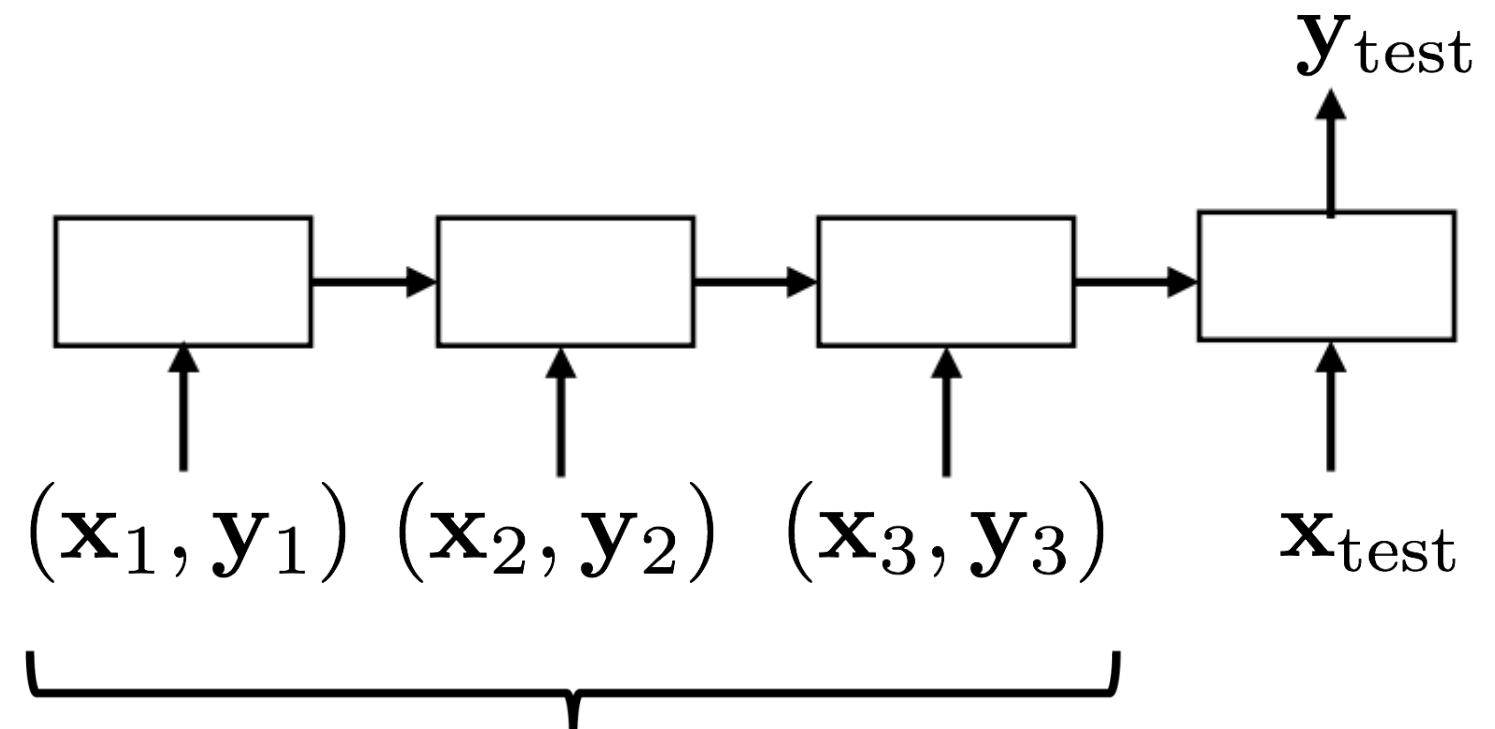
$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$



# Design of $f$ ?

Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

Does it converge?  
- Sort of?

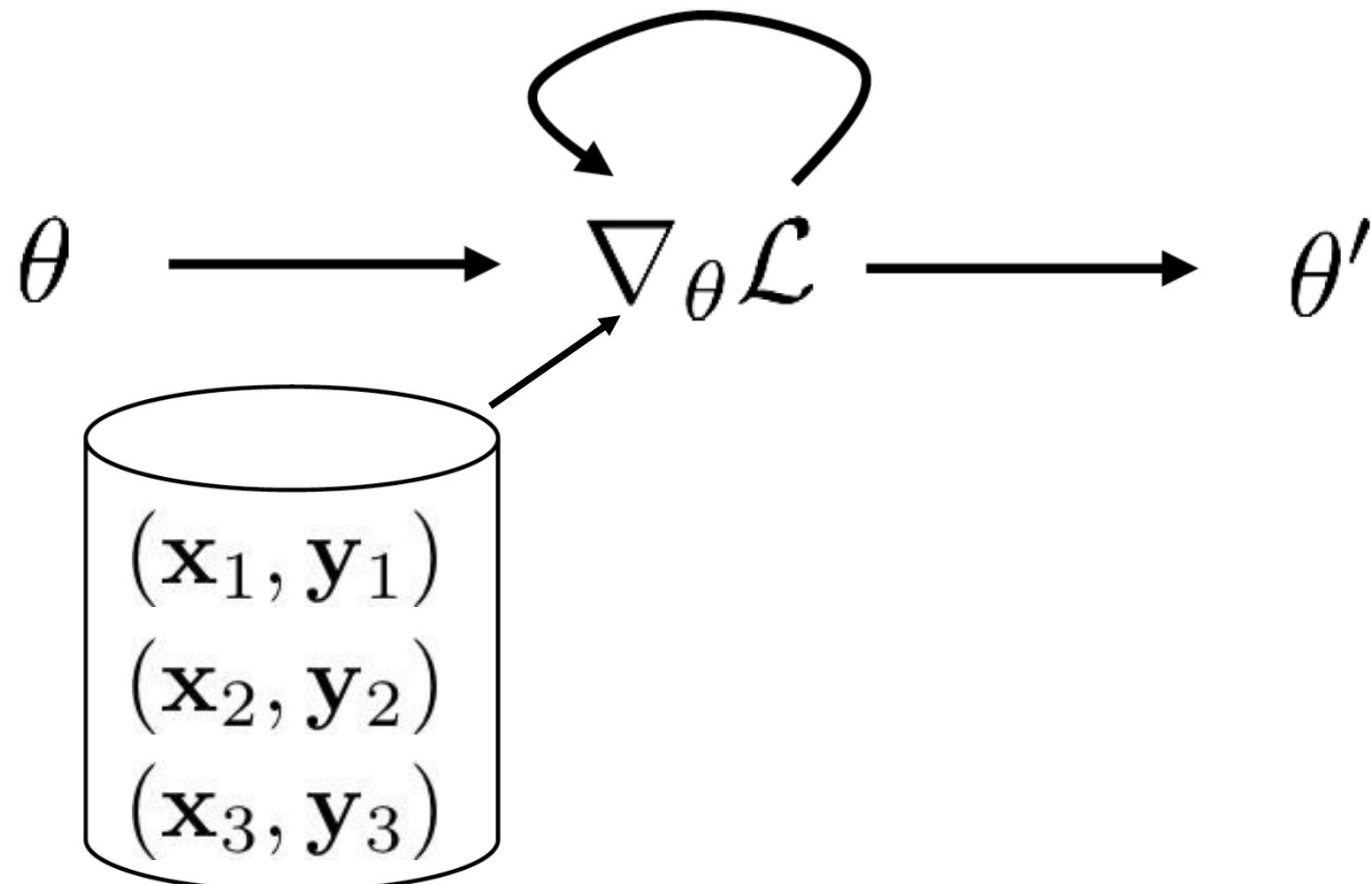
What does it converge to?  
- Who knows...

What to do if not good enough?  
- Nothing

$$\mathcal{D}_{\text{train}} \ \mathbf{x}_{\text{test}} \xrightarrow{\quad} \mathbf{y}_{\text{test}}$$

MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$



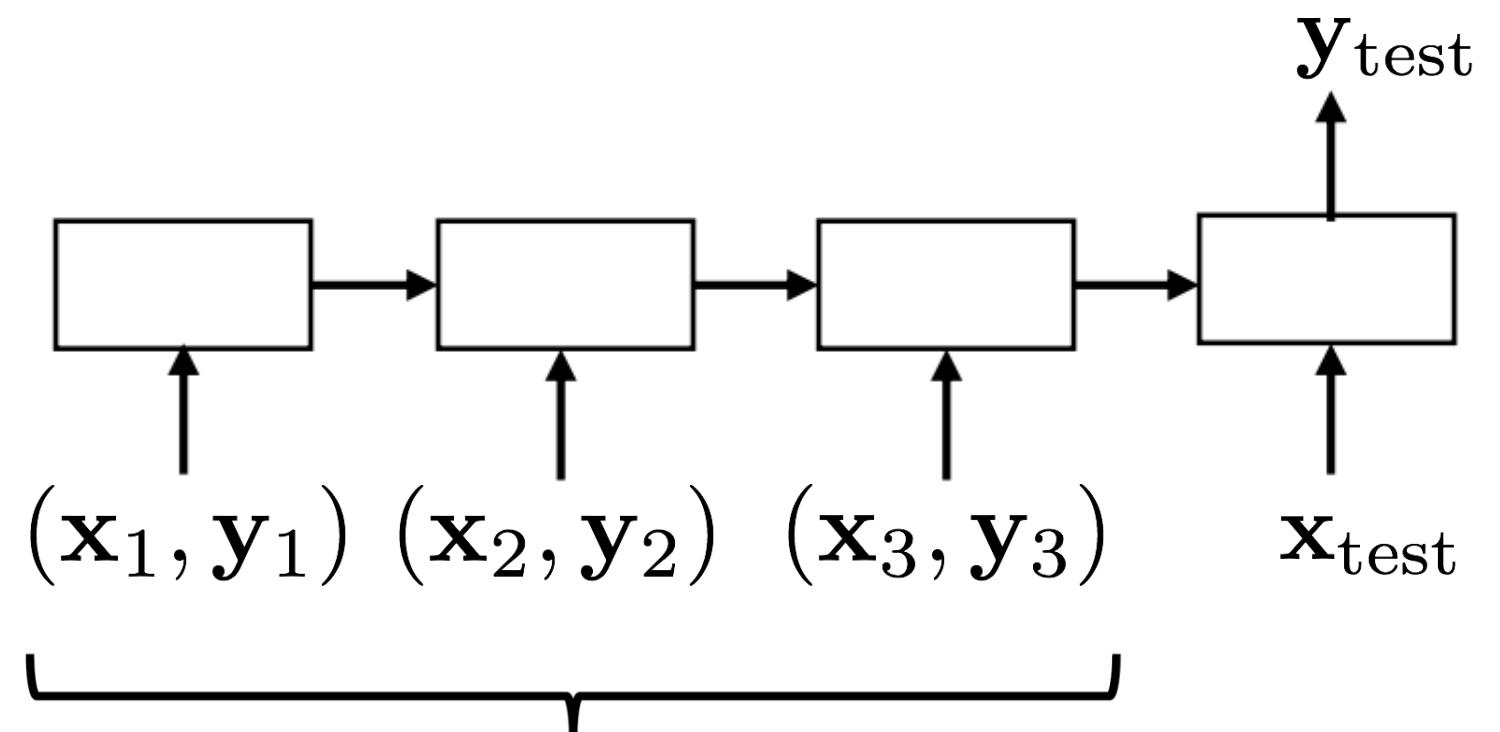
Does it converge?

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\text{green arrow}} y_{\text{test}}$$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

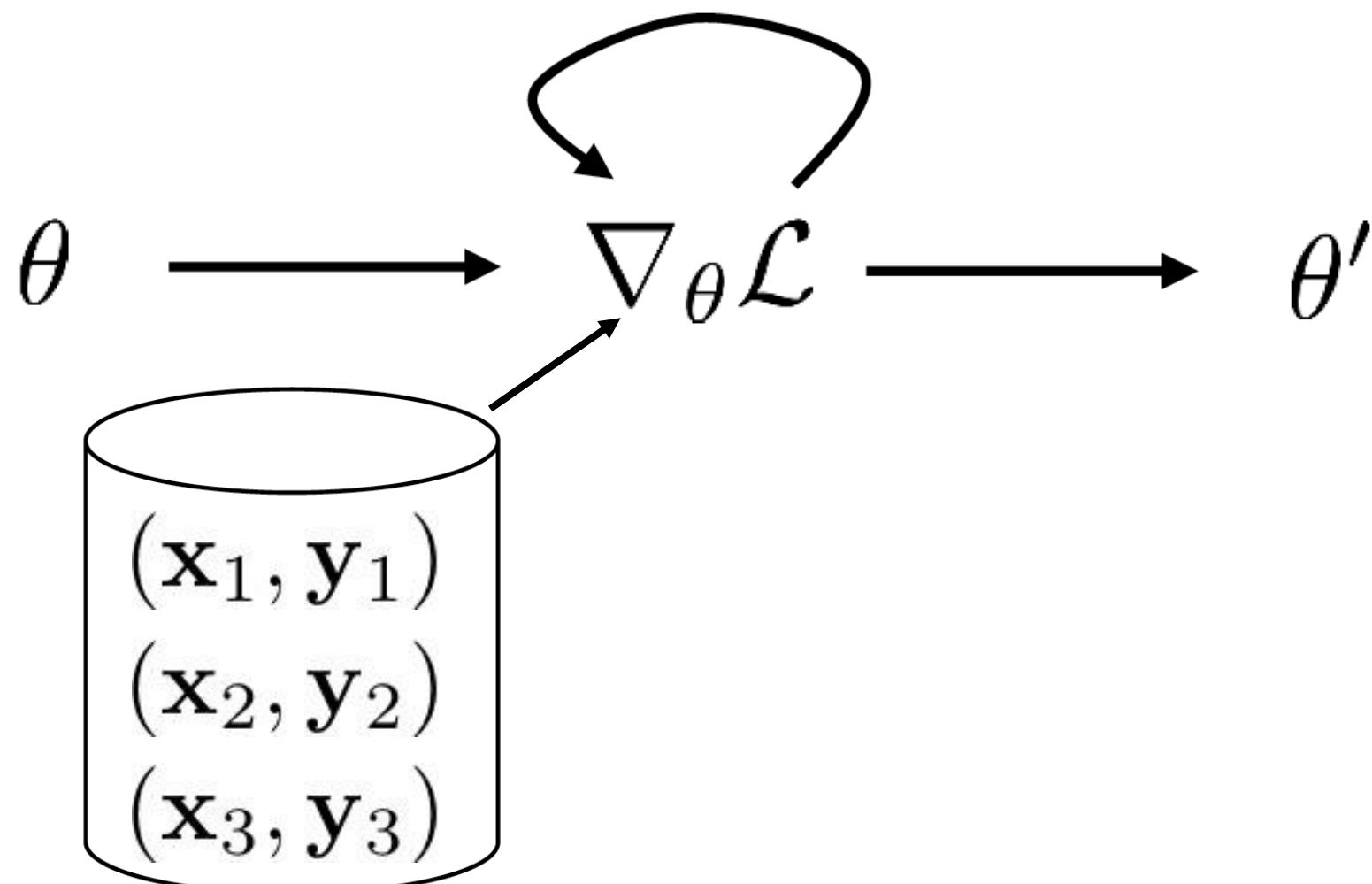
Does it converge?  
- Sort of?

What does it converge to?  
- Who knows...

What to do if not good enough?  
- Nothing

MAML

$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$



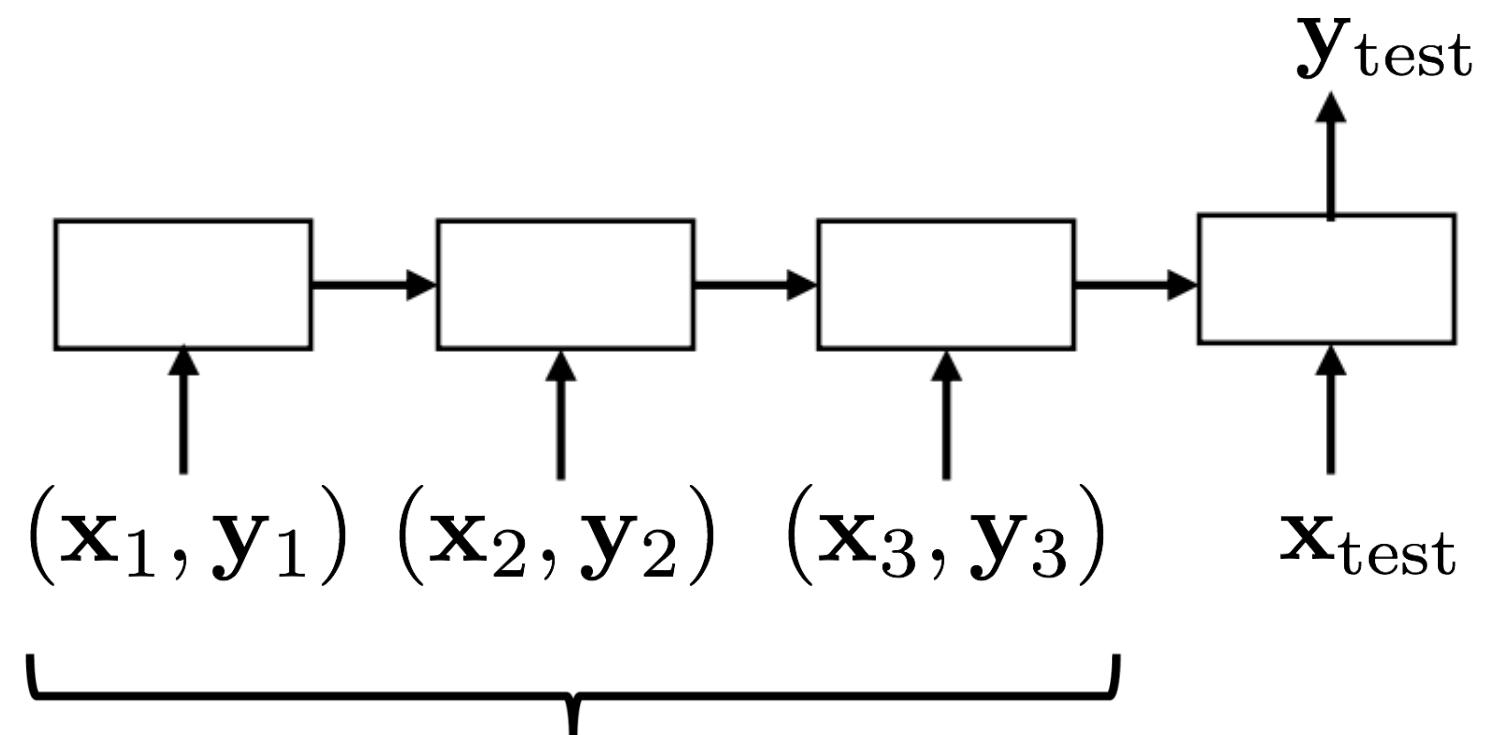
Does it converge?  
- Yes (it's gradient descent...)

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

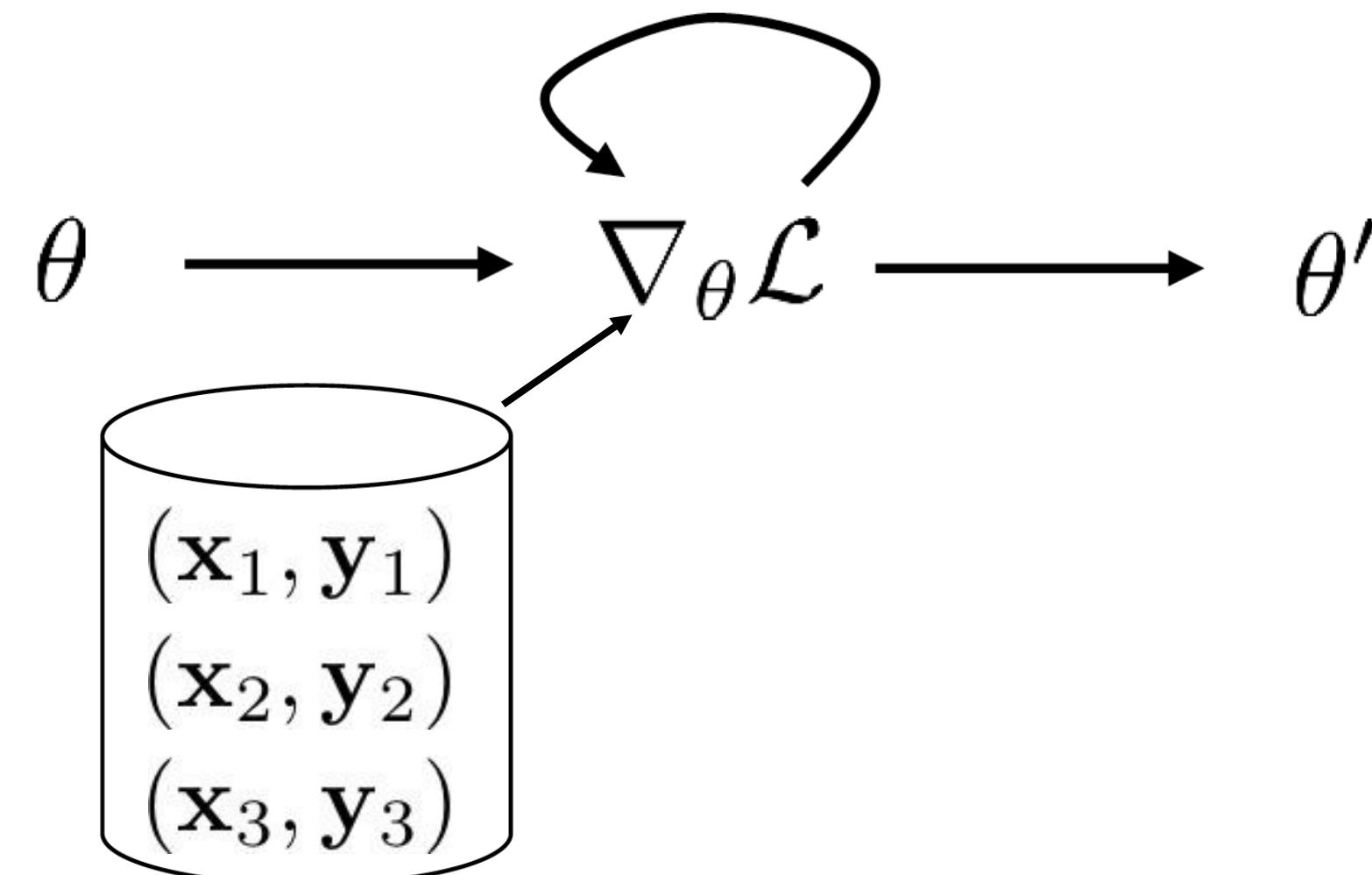
Does it converge?  
- Sort of?

What does it converge to?  
- Who knows...

What to do if not good enough?  
- Nothing

MAML

$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$



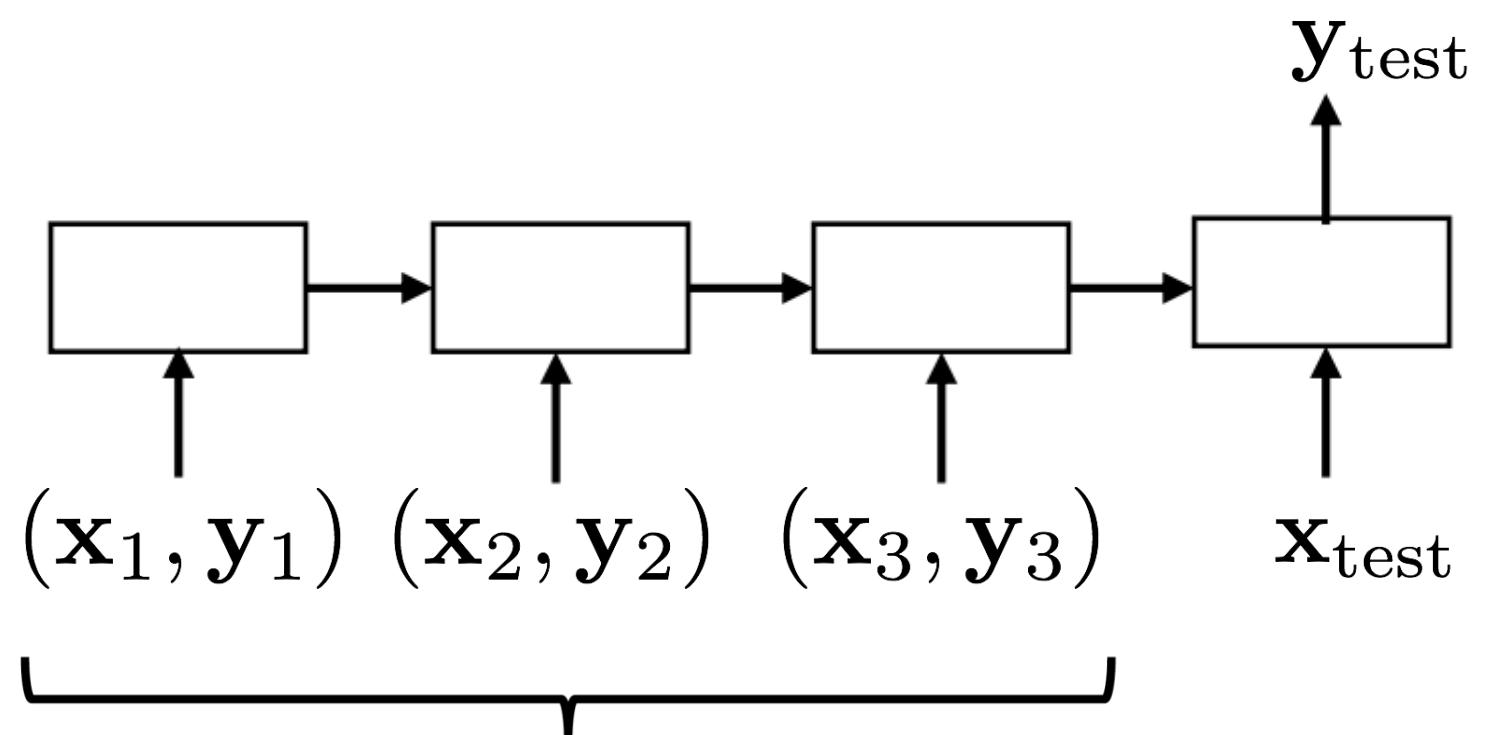
Does it converge?  
- Yes (it's gradient descent...)  
What does it converge to?

# Design of $f$ ?

$$\mathcal{D}_{\text{train}} \ x_{\text{test}} \xrightarrow{\quad} y_{\text{test}}$$

Recurrent network

$$y_{\text{test}} = f(\mathcal{D}_{\text{train}}, x_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

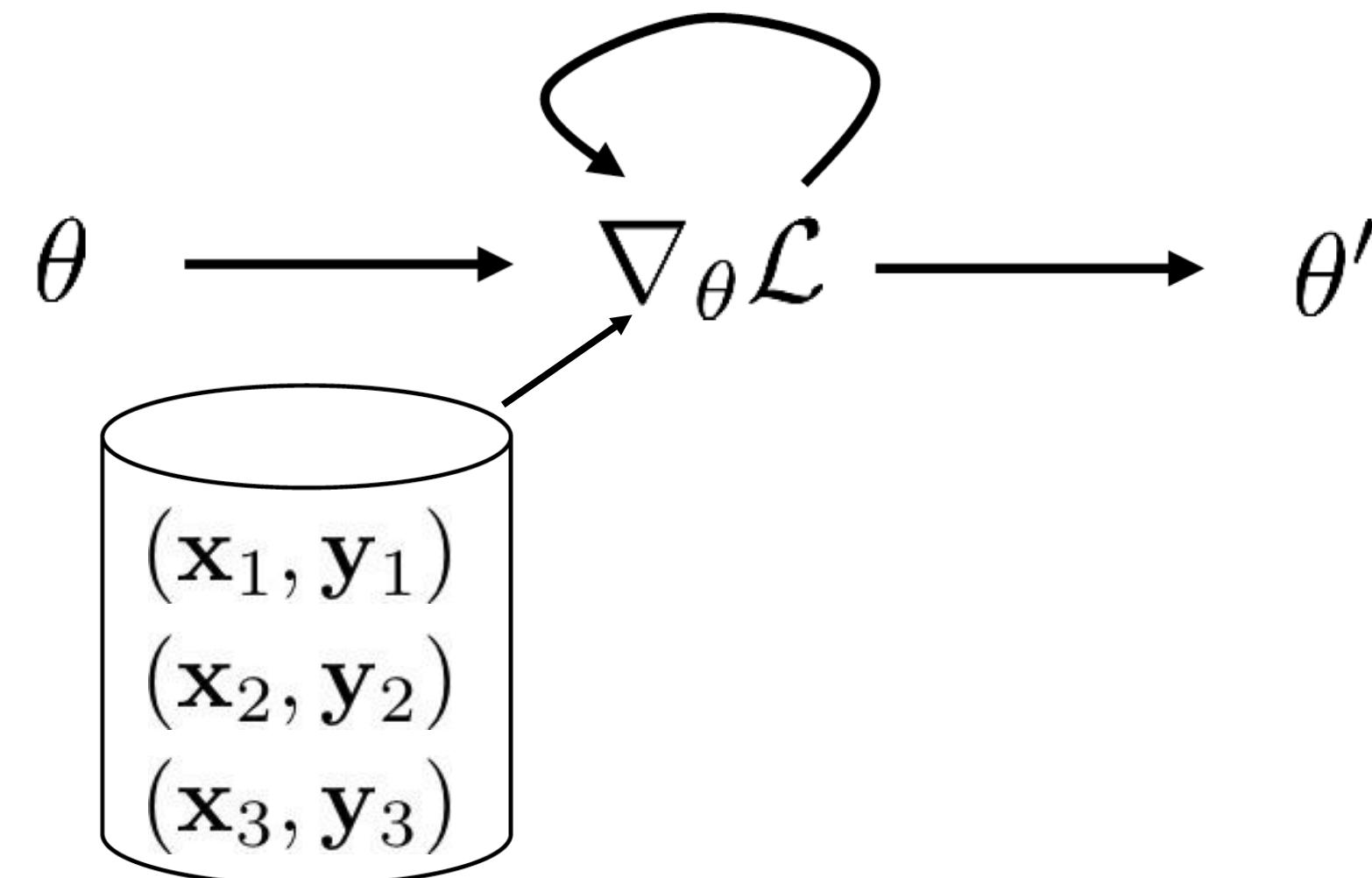
Does it converge?  
- Sort of?

What does it converge to?  
- Who knows...

What to do if not good enough?  
- Nothing

MAML

$$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$



Does it converge?

- Yes (it's gradient descent...)

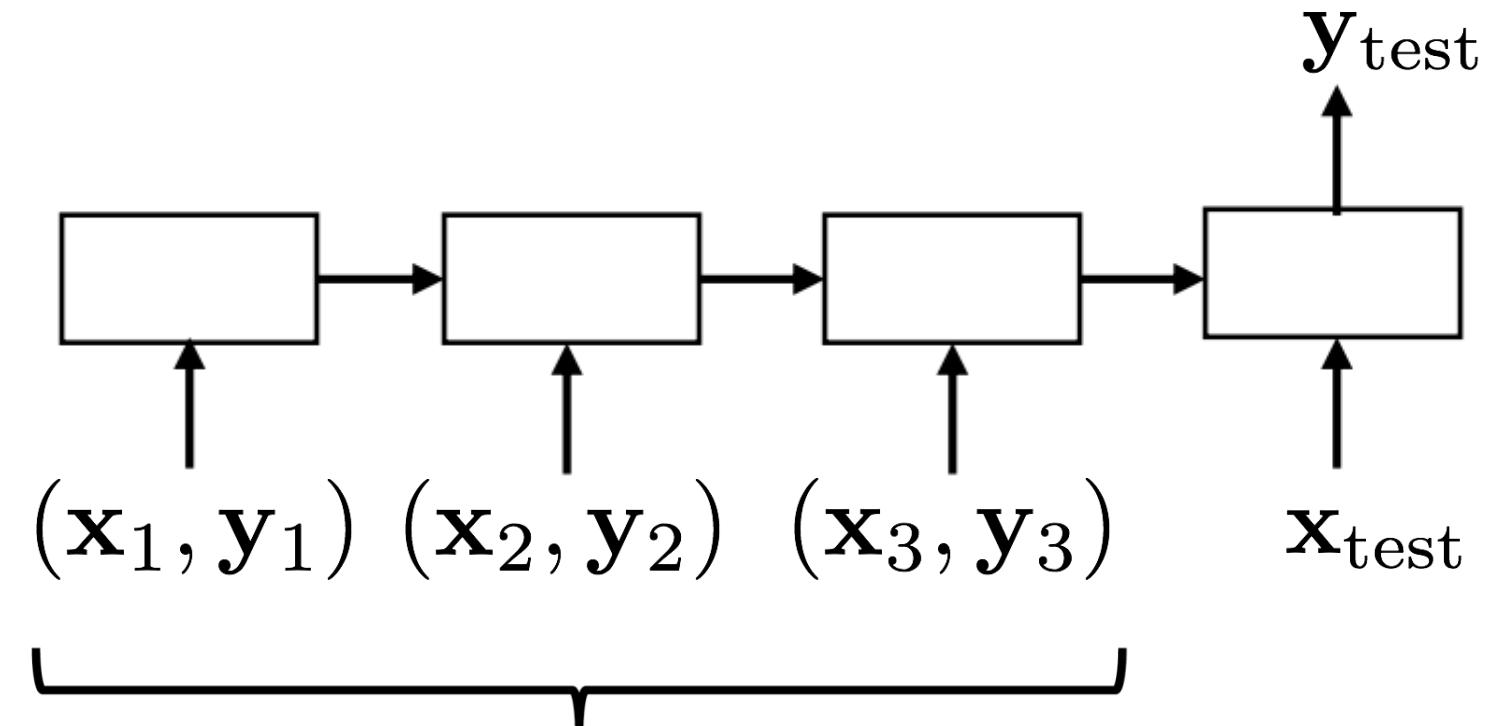
What does it converge to?

- A local optimum (it's gradient descent...)

# Design of $f$ ?

Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

Does it converge?  
- Sort of?

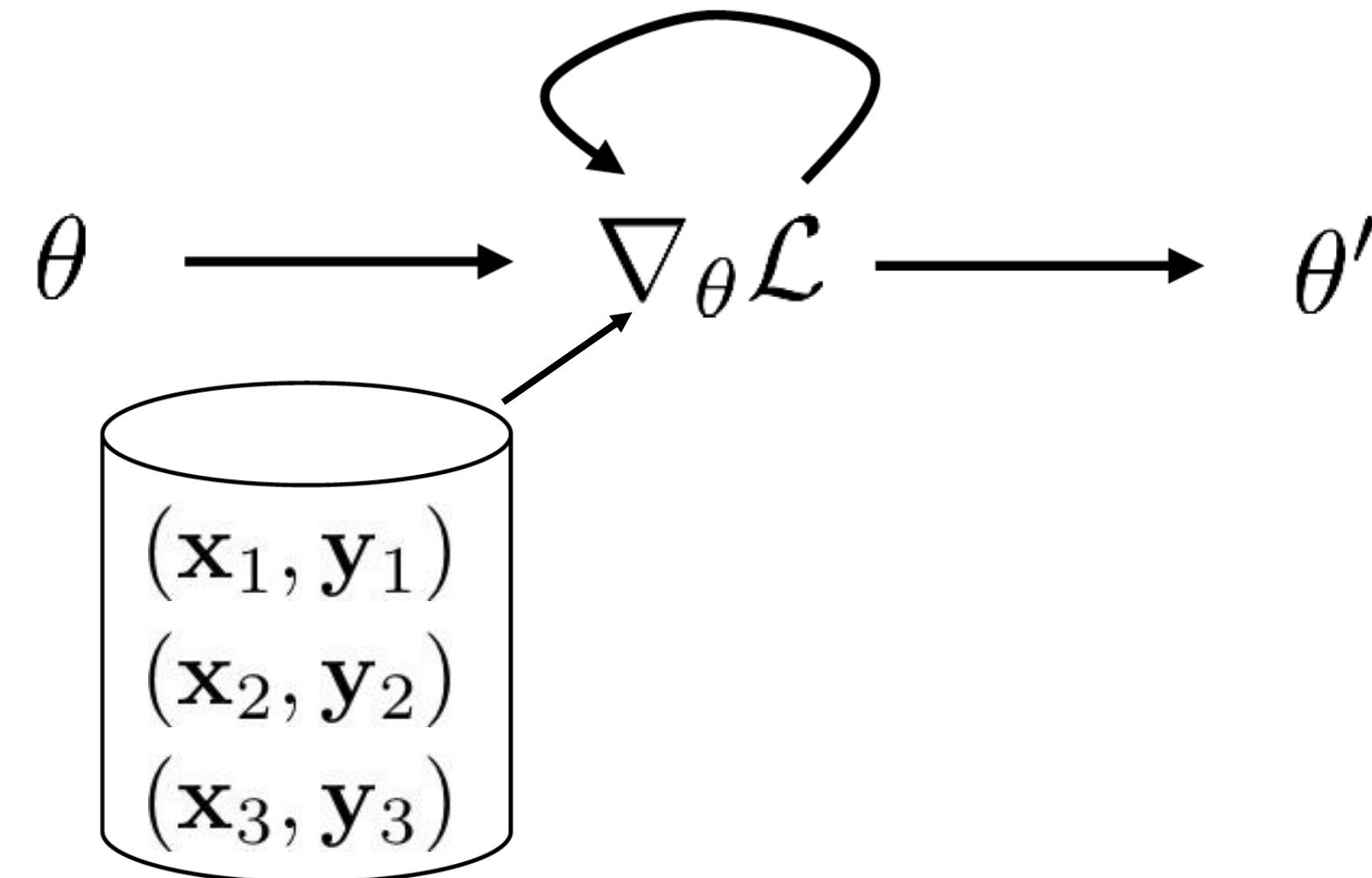
What does it converge to?  
- Who knows...

What to do if not good enough?  
- Nothing

$$\mathcal{D}_{\text{train}} \ \mathbf{x}_{\text{test}} \longrightarrow \mathbf{y}_{\text{test}}$$

MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$



Does it converge?

- Yes (it's gradient descent...)

What does it converge to?

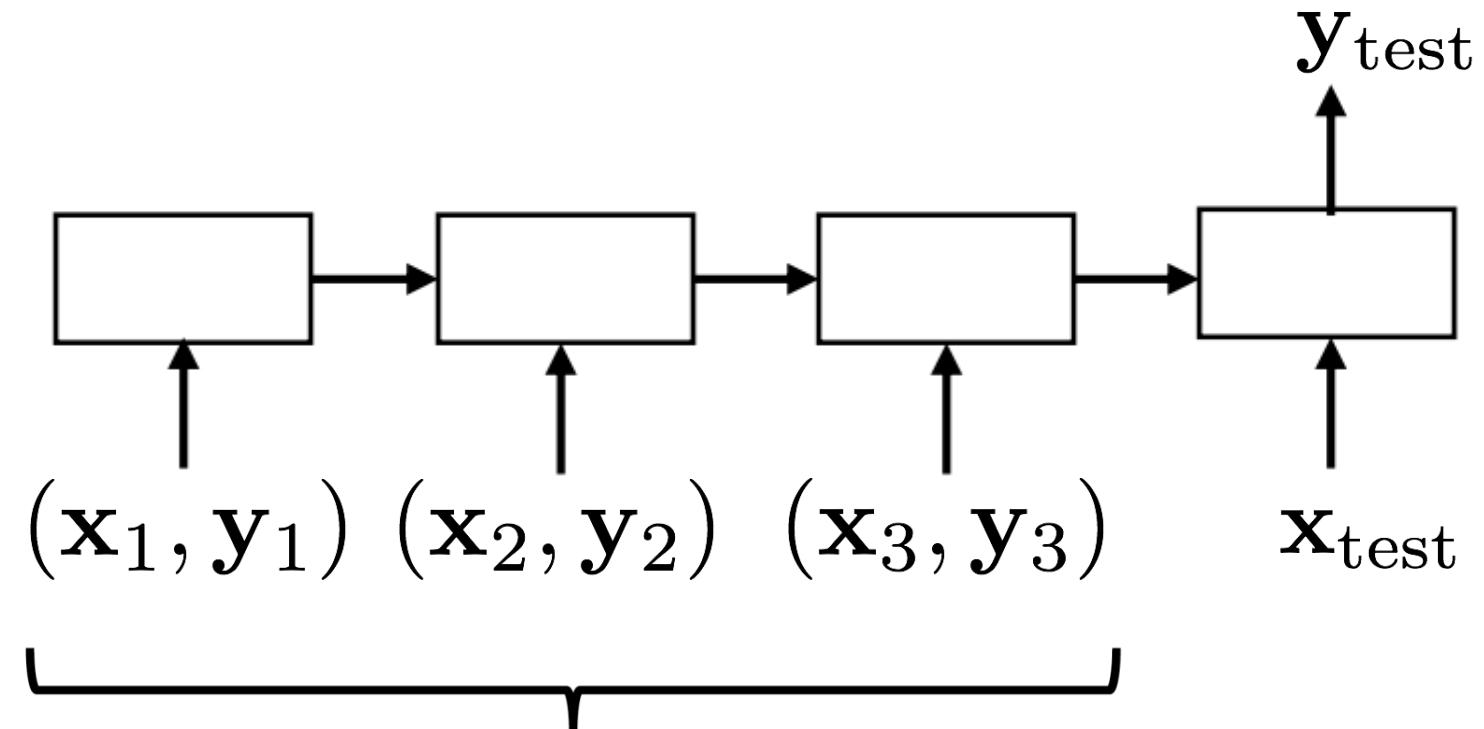
- A local optimum (it's gradient descent...)

What to do if not good enough?

# Design of $f$ ?

Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$



network implements the  
“learned learning procedure”

Does it converge?

- Sort of?

What does it converge to?

- Who knows...

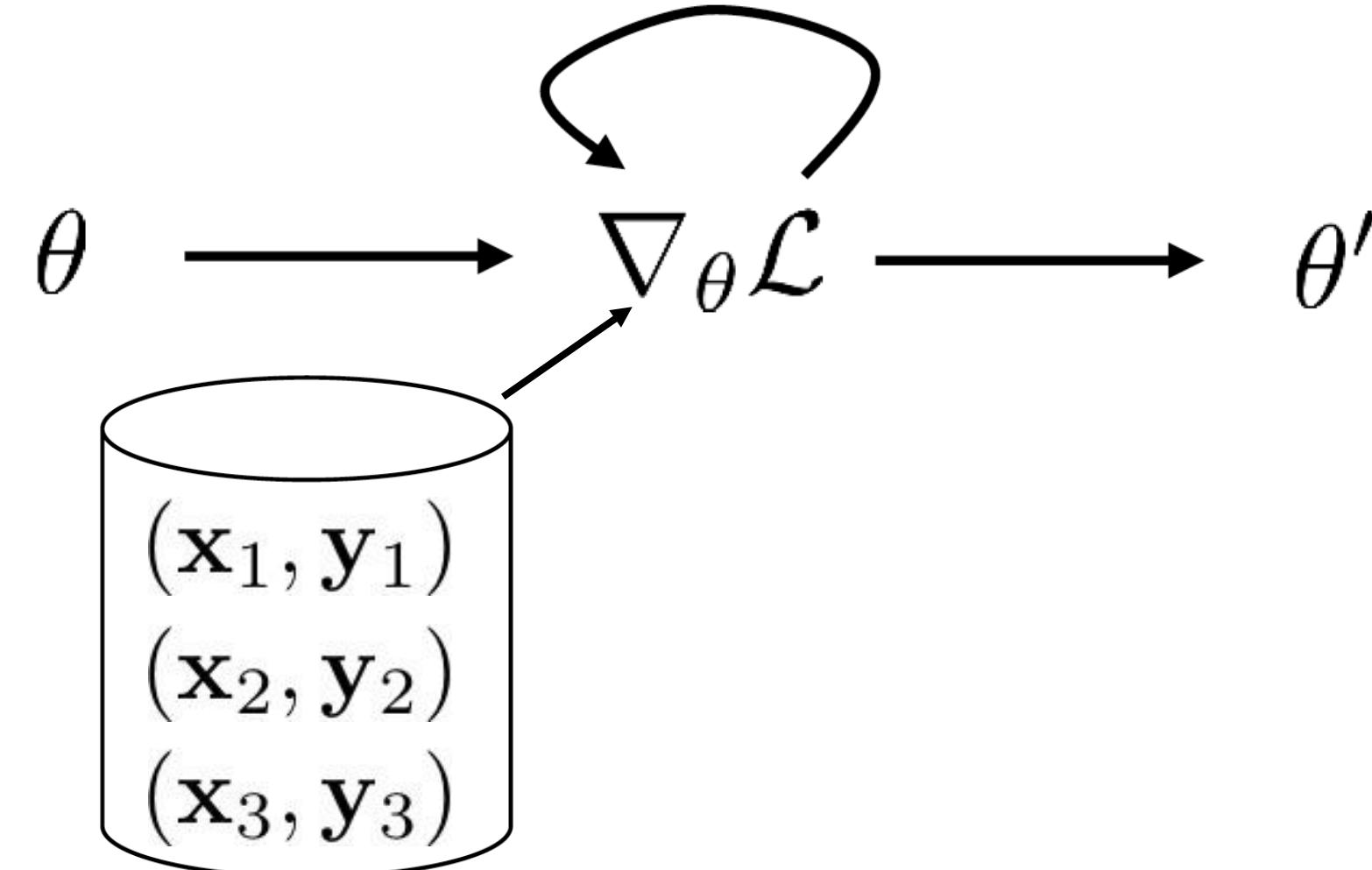
What to do if not good enough?

- Nothing

$$\mathcal{D}_{\text{train}} \ \mathbf{x}_{\text{test}} \xrightarrow{\text{green arrow}} \mathbf{y}_{\text{test}}$$

MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$



Does it converge?

- Yes (it's gradient descent...)

What does it converge to?

- A local optimum (it's gradient descent...)

What to do if not good enough?

- Keep taking gradient steps (it's gradient descent..)

# How well can methods generalize to similar, but extrapolated tasks?

How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

MAML

How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML SAIL, MetaNetworks**

How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**, **MetaNetworks**

**Omniglot image classification**

How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**, **MetaNetworks**

Omniglot image classification

performance

task variability

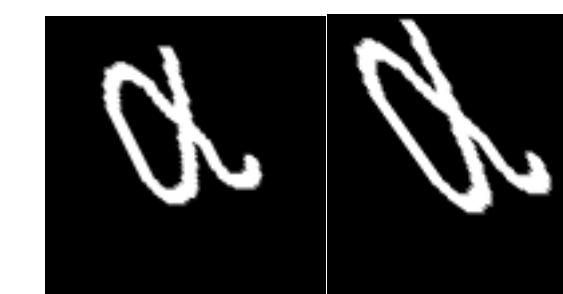
How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**, **MetaNetworks**

Omniglot image classification

performance



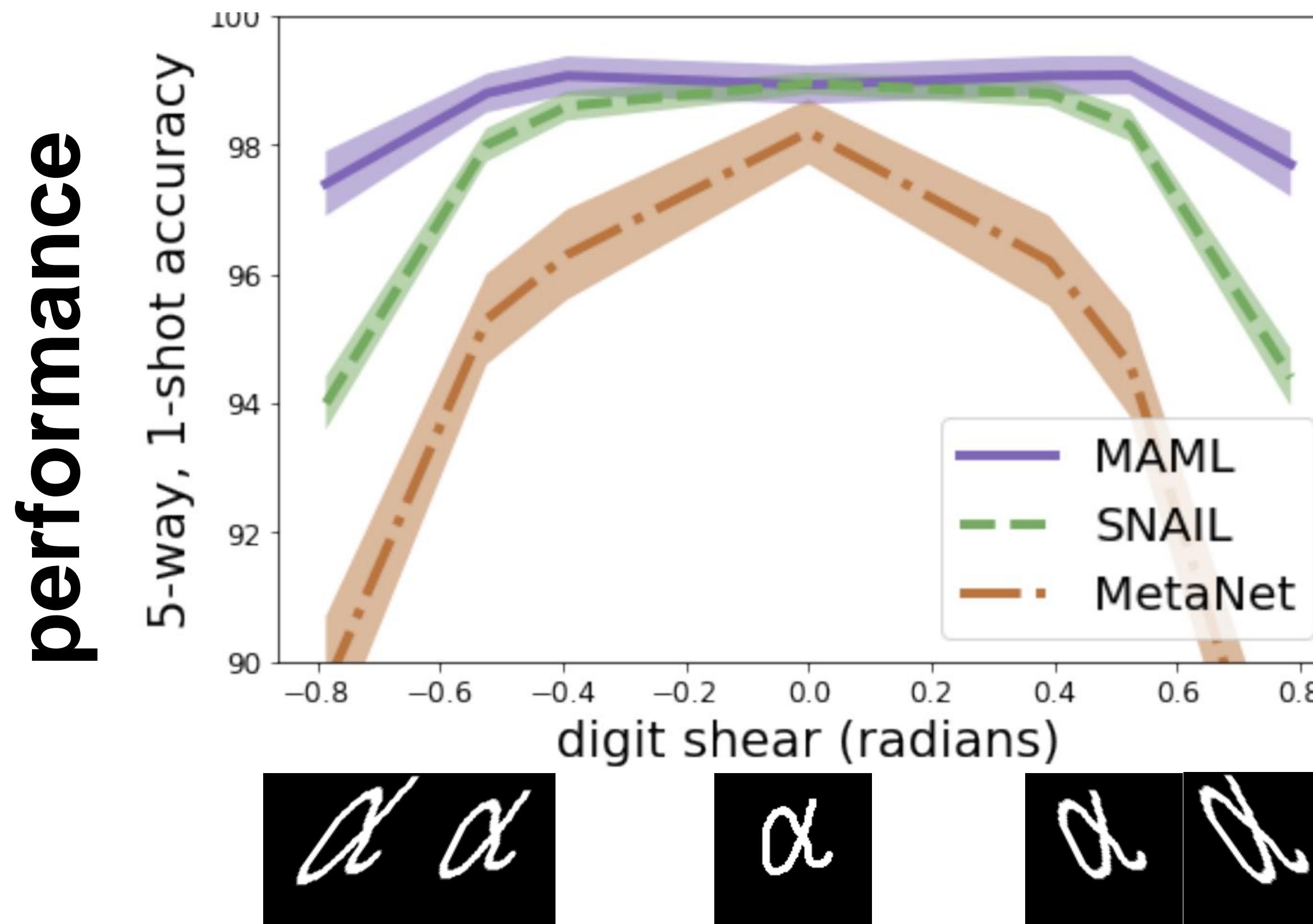
**task variability**

# How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML SAIL, MetaNetworks**

## Omniglot image classification



## task variability

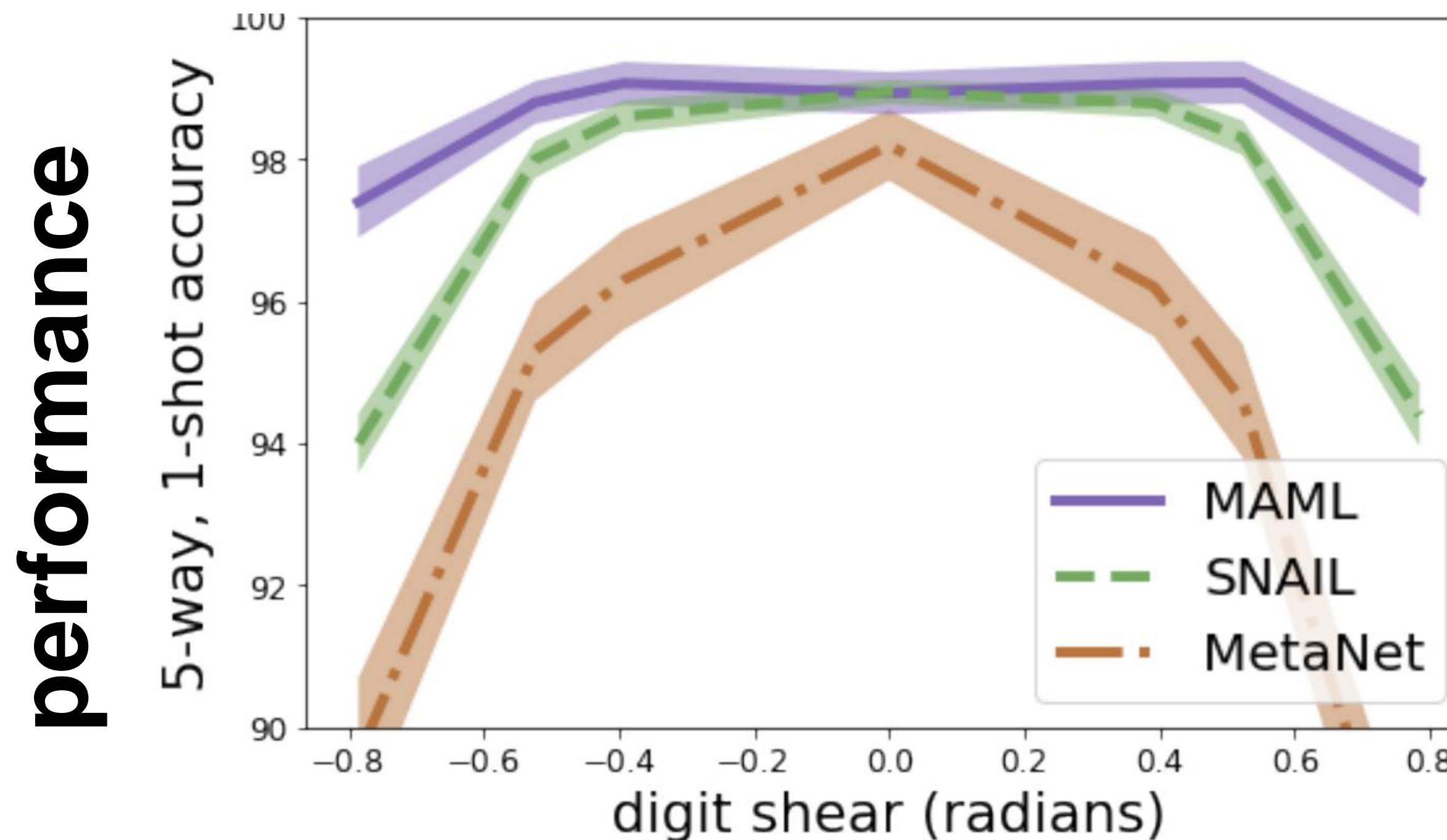


# How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML SAIL, MetaNetworks**

## Omniglot image classification



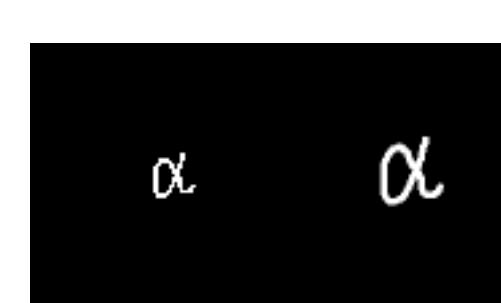
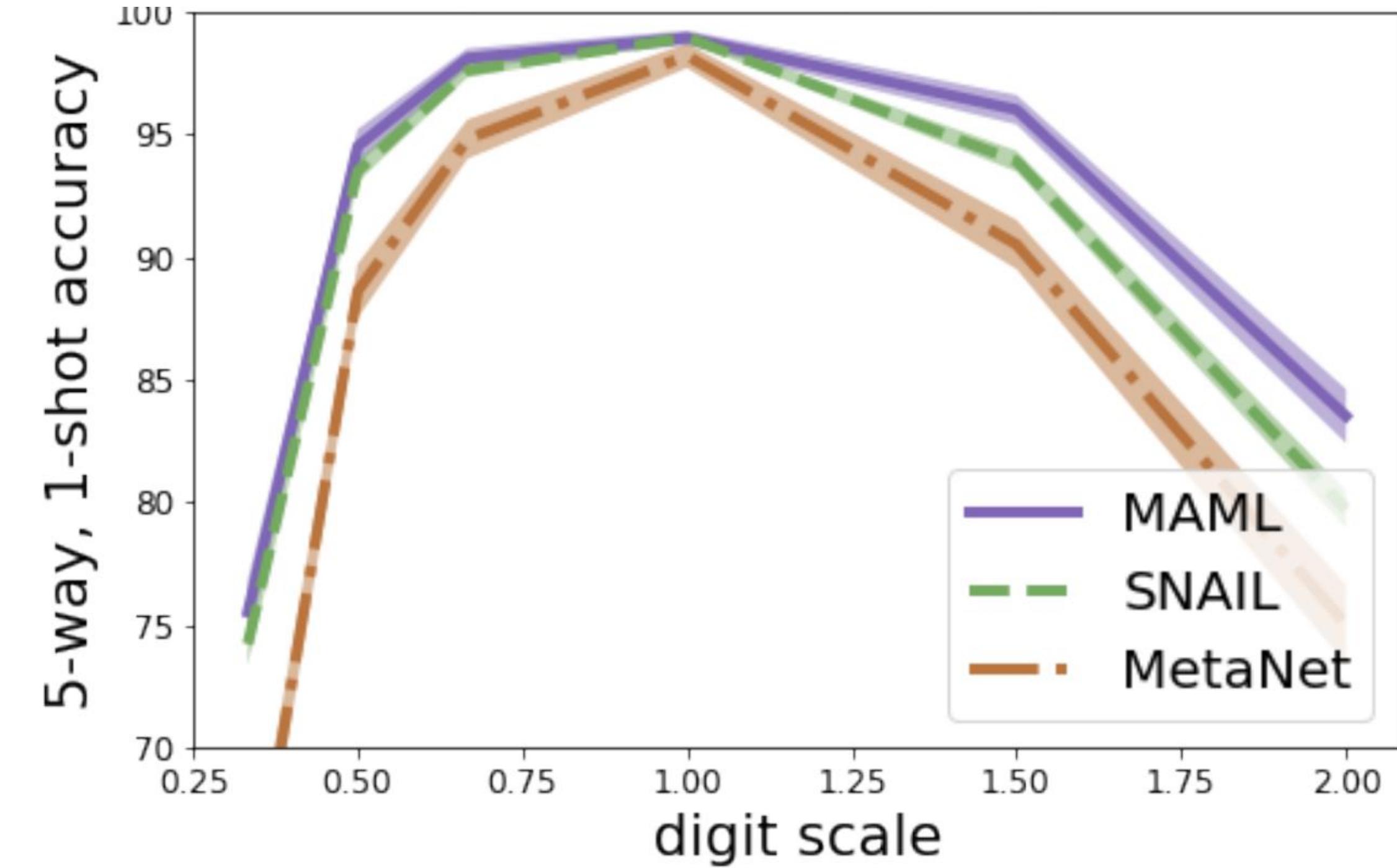
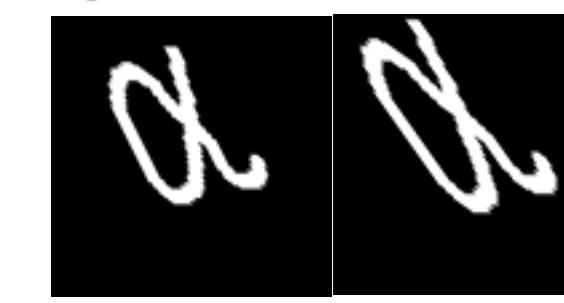
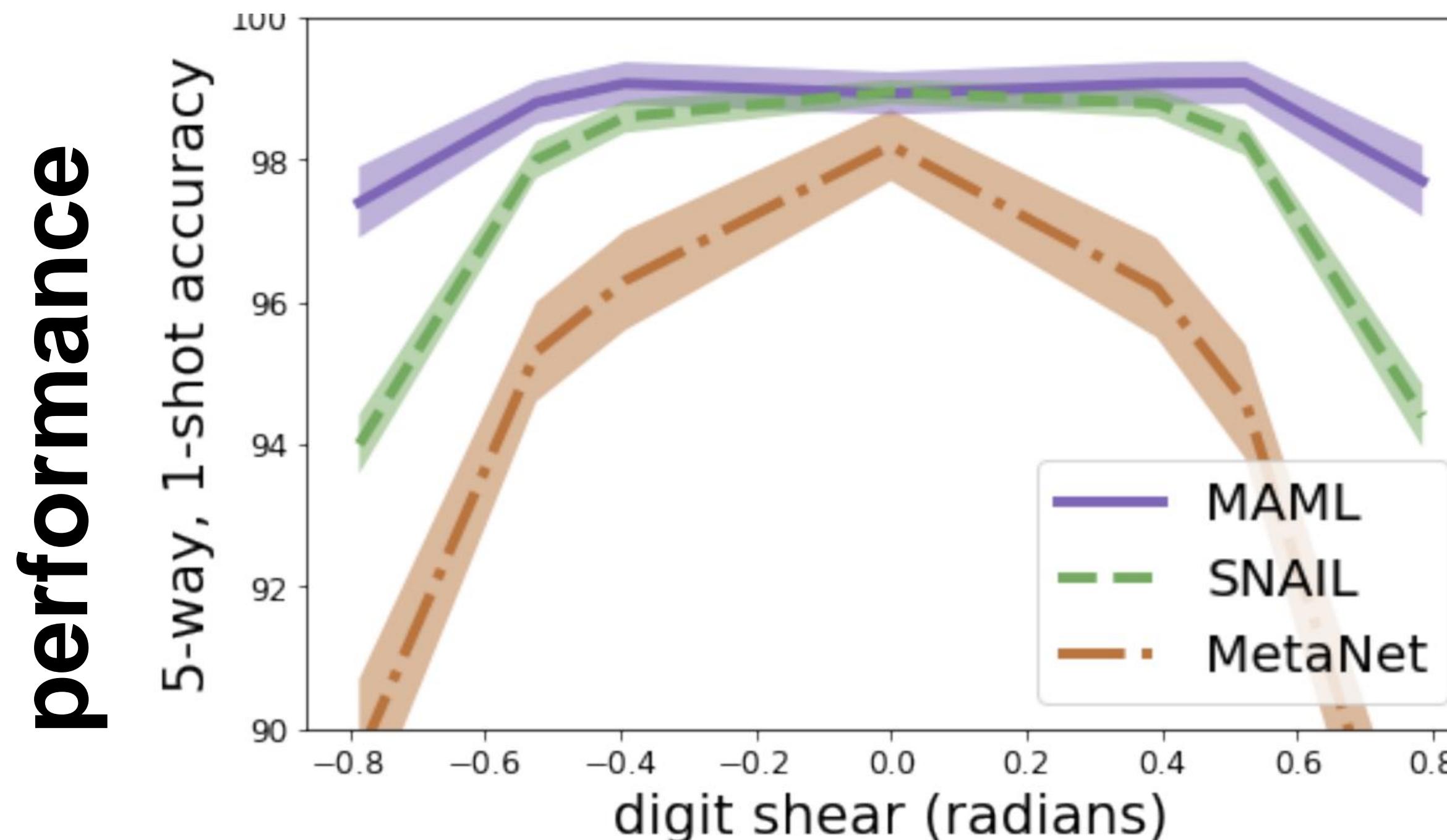
## task variability

# How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML SAIL, MetaNetworks**

## Omniglot image classification



## task variability

How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**

**Sinusoid curve regression**

How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**

**Sinusoid curve regression**

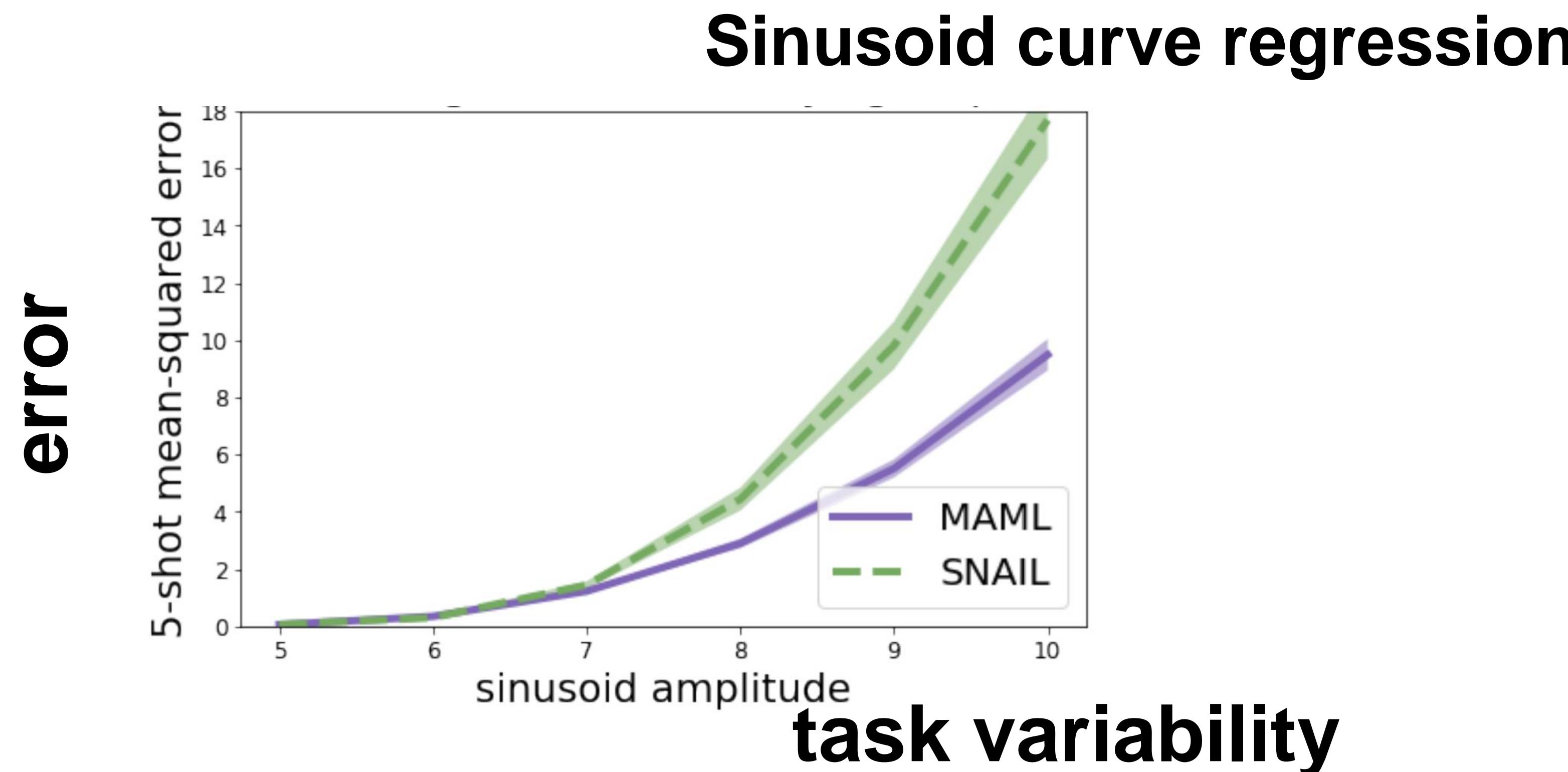
error

**task variability**

# How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**

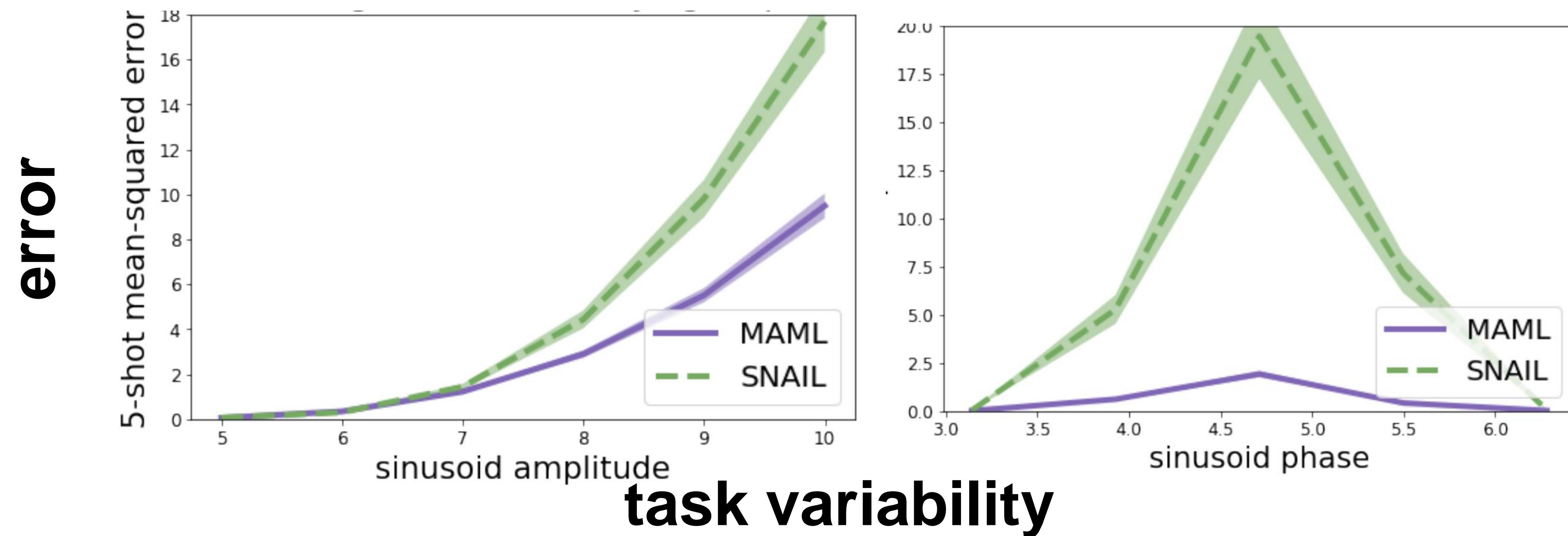


# How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**

## Sinusoid curve regression

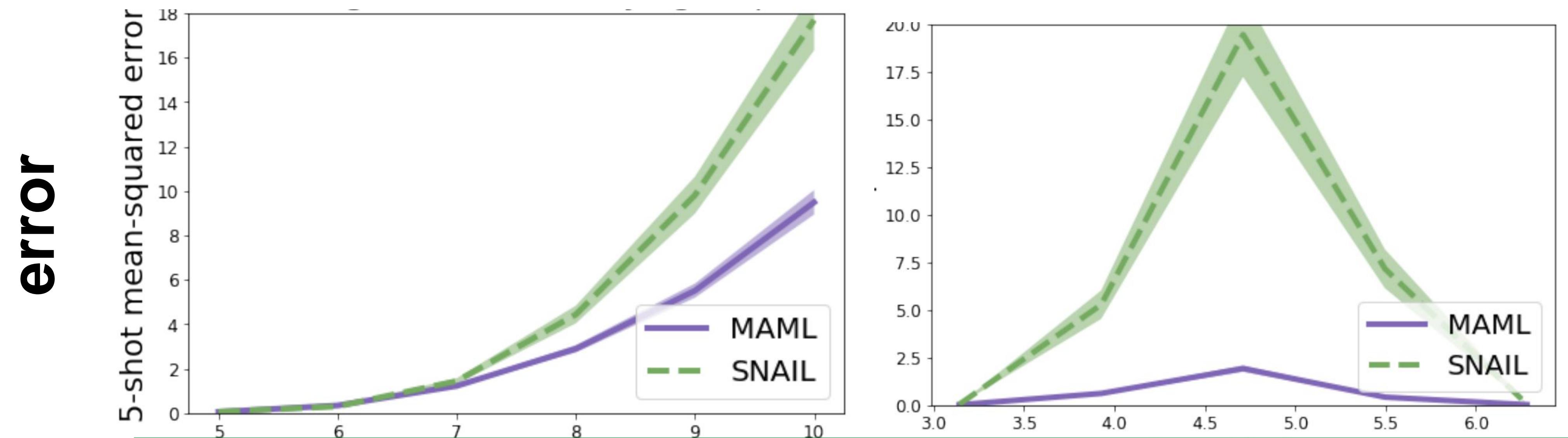


# How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**

## Sinusoid curve regression



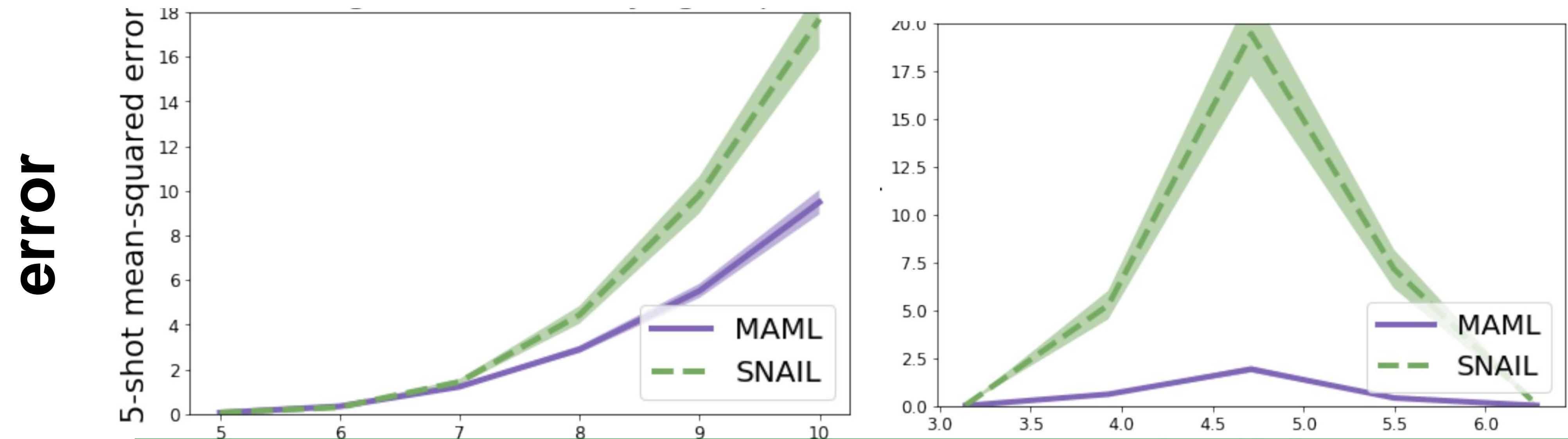
**Takeaway:** Strategies learned with MAML consistently generalize better to out-of-distribution tasks

# How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**MAML** **SNAIL**

## Sinusoid curve regression



**Takeaway:** Strategies learned with MAML consistently generalize better to out-of-distribution tasks

## Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

## MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

Does this structure come at a cost?

## Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

## MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

Does this structure come at a cost?

For a sufficiently deep  $f$ ,

MAML function can approximate any function  $\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}$

Finn & Levine, ICLR 2018

## Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

## MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

Does this structure come at a cost?

For a sufficiently deep  $f$ ,

MAML function can approximate any function  $\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}$

Finn & Levine, ICLR 2018

Assumptions:

## Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

## MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

Does this structure come at a cost?

For a sufficiently deep  $f$ ,

MAML function can approximate any function  $\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}$

Finn & Levine, ICLR 2018

Assumptions:

- nonzero  $\alpha$

## Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

## MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

Does this structure come at a cost?

For a sufficiently deep  $f$ ,

MAML function can approximate any function  $\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}$

Finn & Levine, ICLR 2018

Assumptions:

- nonzero  $\alpha$
- loss function gradient does not lose information about the label

## Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

## MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

Does this structure come at a cost?

For a sufficiently deep  $f$ ,

MAML function can approximate any function  $\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}$

Finn & Levine, ICLR 2018

Assumptions:

- nonzero  $\alpha$
- loss function gradient does not lose information about the label
- datapoints in  $\mathcal{D}_{\text{train}}$  are unique

## Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

## MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

Does this structure come at a cost?

For a sufficiently deep  $f$ ,

MAML function can approximate any function  $\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}$

Finn & Levine, ICLR 2018

Assumptions:

- nonzero  $\alpha$
- loss function gradient does not lose information about the label
- datapoints in  $\mathcal{D}_{\text{train}}$  are unique

Why is this interesting?

## Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

## MAML

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_{\theta} \mathcal{L}(\mathcal{D}_{\text{train}}))$$

Does this structure come at a cost?

For a sufficiently deep  $f$ ,

MAML function can approximate any function  $\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}$

Finn & Levine, ICLR 2018

Assumptions:

- nonzero  $\alpha$
- loss function gradient does not lose information about the label
- datapoints in  $\mathcal{D}_{\text{train}}$  are unique

Why is this interesting?

MAML has benefit of inductive bias without losing expressive power.

# Application: One-Shot Visual Imitation Learning

## One-Shot Visual Imitation Learning via Meta-Learning

**Chelsea Finn<sup>\*1</sup>, Tianhe Yu<sup>\*1</sup>, Tianhao Zhang<sup>1</sup>, Pieter Abbeel<sup>1,2</sup>, Sergey Levine<sup>1</sup>**

<sup>1</sup>University of California, Berkeley, <sup>2</sup>OpenAI

`cbfinn,tianhe.yu,tianhao.z,pabbeel,svlevine@berkeley.edu`

## One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning

Tianhe Yu\*, Chelsea Finn\*, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, Sergey Levine

University of California, Berkeley

Email: `{tianhe.yu,cbfinn,annie.xie,sdasari,tianhao.z,pabbeel,svlevine}@berkeley.edu`

\* denotes equal contribution

# One-Shot Visual Imitation Learning

# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

Visual imitation is expensive.

# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

Visual imitation is expensive.



Rahmanizadeh et al. '17  
hang et al. '17  
learns from raw pixels,  
but requires many demonstrations

# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

Visual imitation is expensive.



Rahmanizadeh et al. '17  
hang et al. '17  
learns from raw pixels,  
but requires many demonstrations

Through meta-learning: reuse data from other  
tasks/objects/environments

# One-Shot Visual Imitation Learning

# One-Shot Visual Imitation Learning

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

# One-Shot Visual Imitation Learning

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

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

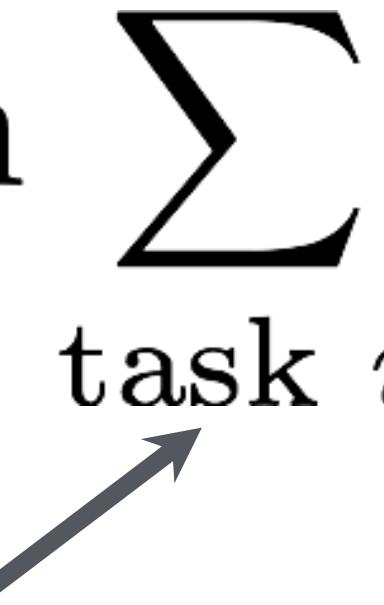
# One-Shot Visual Imitation Learning

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

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

meta-training  
tasks



# One-Shot Visual Imitation Learning

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

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

**meta-training tasks**

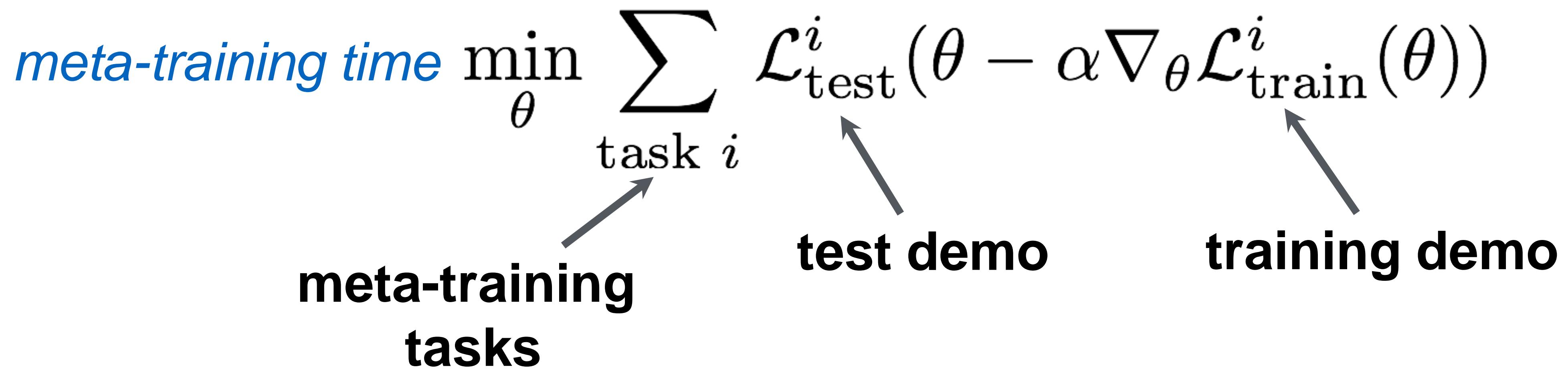
**training demo**

```
graph TD; Eq["meta-training time minθ ∑ task i Litest(θ - α ∇θ Litrain(θ))"] -- "meta-training tasks" --> Sum["∑ task i"]; Eq -- "training demo" --> Grad["α ∇θ Litrain(θ)"]
```

# One-Shot Visual Imitation Learning

imitation loss

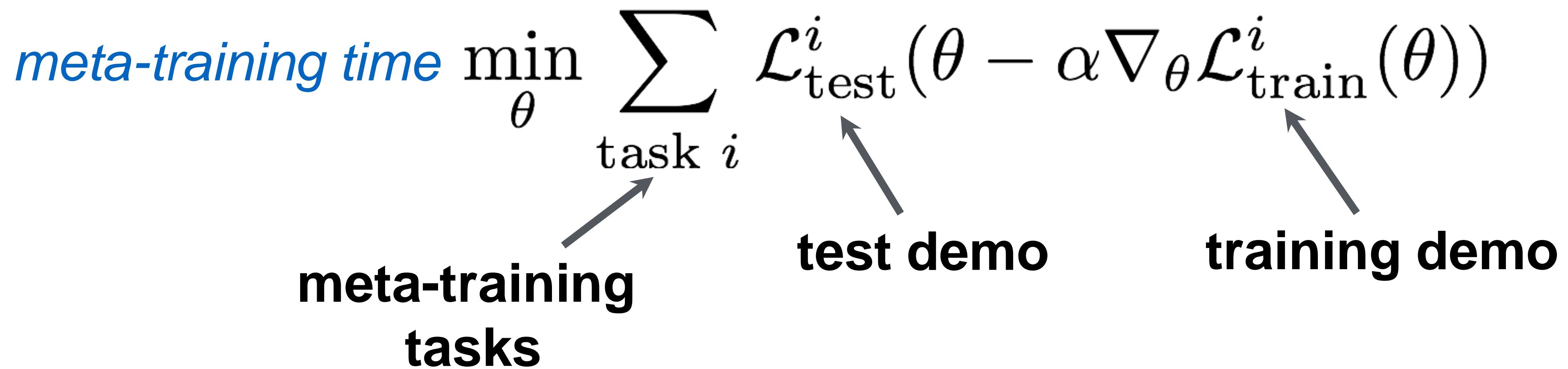
$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$



# One-Shot Visual Imitation Learning

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$



*meta-test time*  $\theta' \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta)$

# One-Shot Visual Imitation Learning

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

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

meta-training  
tasks

test demo

training demo

*meta-test time*  $\theta' \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta)$

demo of meta-test task

# Object placing from pixels



subset of  
training objects



held-out test objects

# Object placing from pixels



subset of  
training objects



held-out test objects

input demo  
(via teleoperation)

# Object placing from pixels



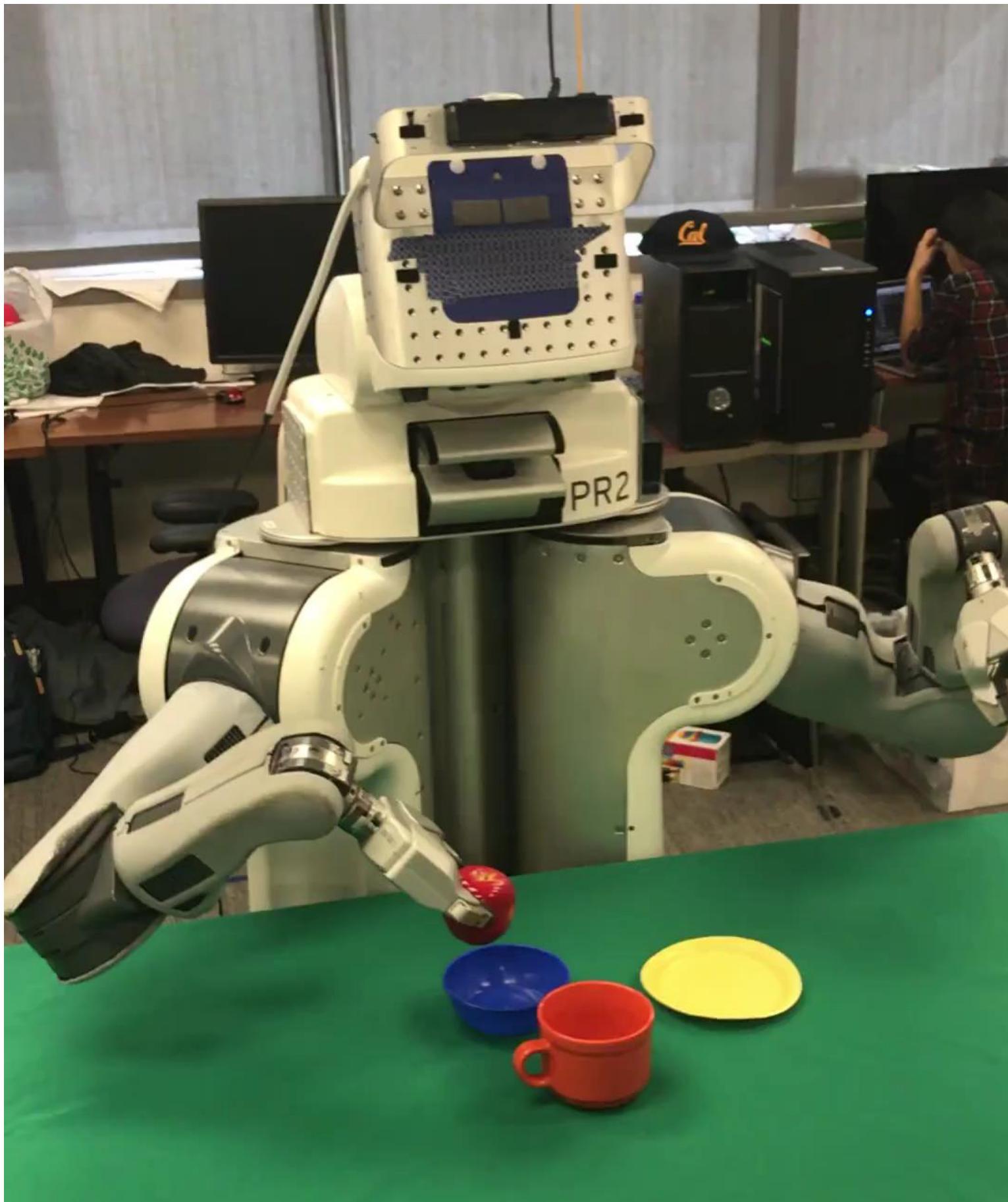
subset of  
training objects



held-out test objects

Chelsea Finn, UC Berkeley

input demo  
(via teleoperation)



Finn\*, Yu\*, Zhang, Abbeel, Levine CoR

# Object placing from pixels



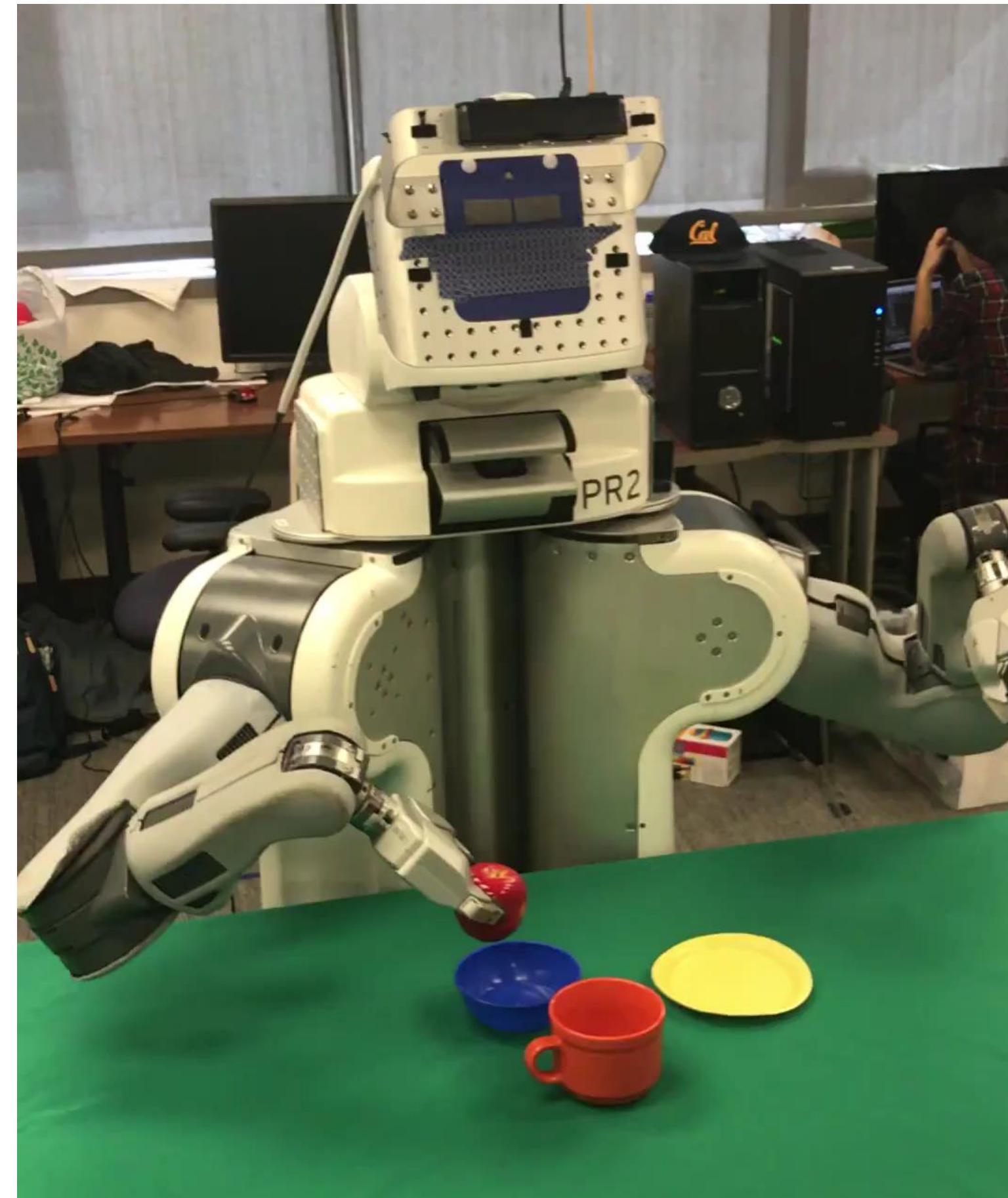
subset of  
training objects



held-out test objects

Chelsea Finn, UC Berkeley

input demo  
(via teleoperation)



resulting policy

Finn\*, Yu\*, Zhang, Abbeel, Levine CoR

# Object placing from pixels



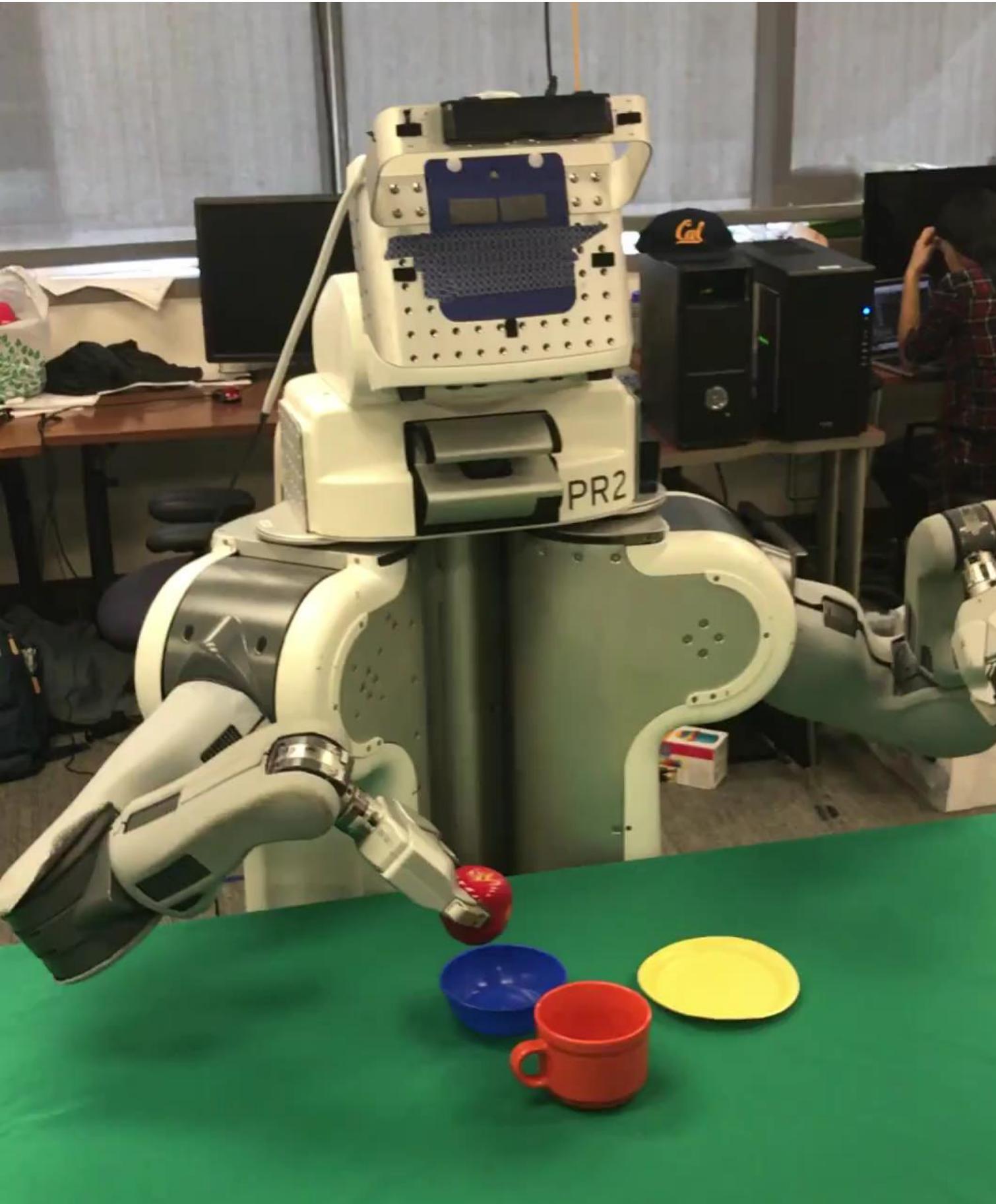
subset of training objects



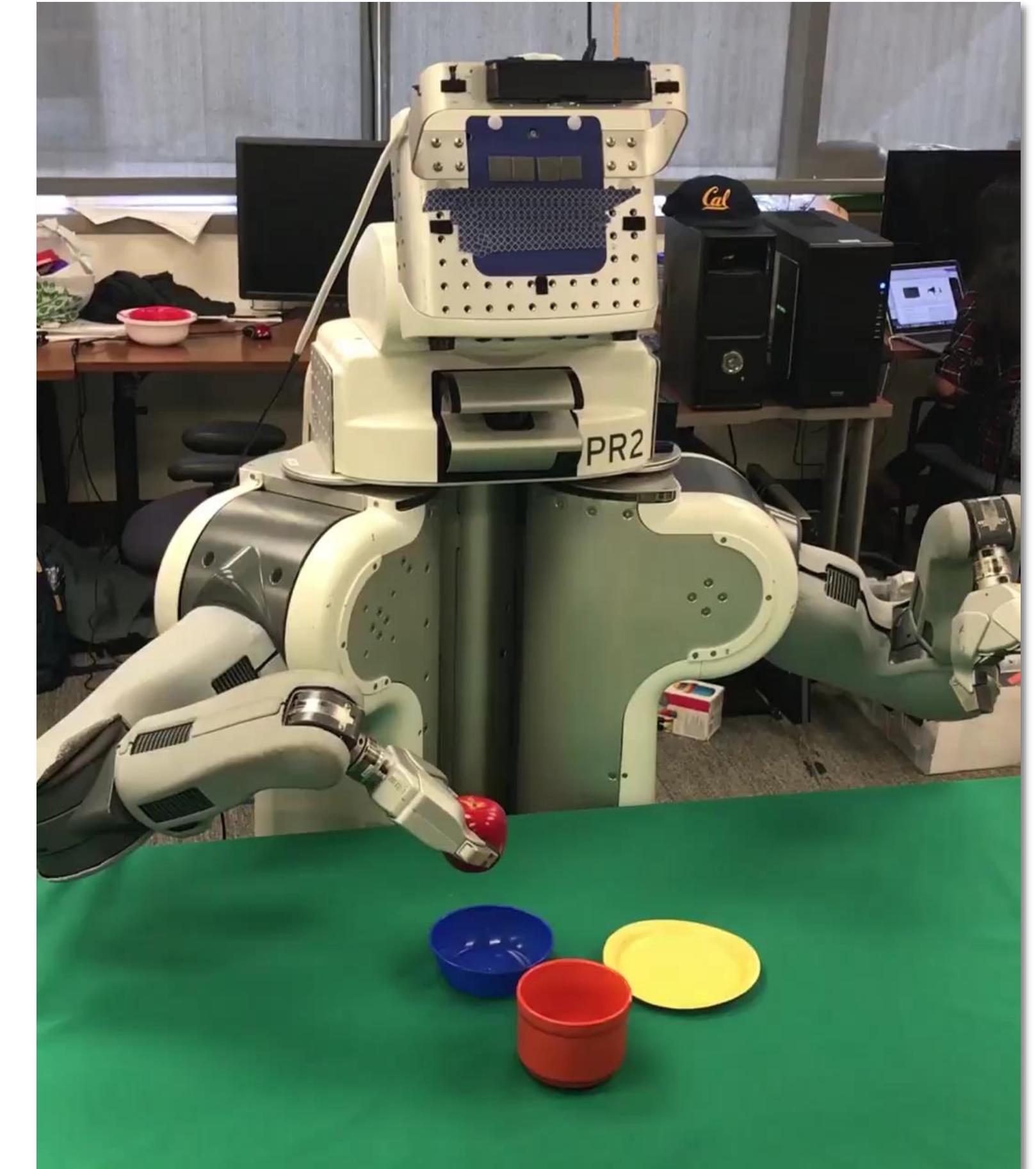
held-out test objects

Chelsea Finn, UC Berkeley

input demo  
(via teleoperation)



resulting policy



[real-time execution]

Finn\*, Yu\*, Zhang, Abbeel, Levine CoR

# Few-Shot Imitation Learning

# Few-Shot Imitation Learning *from Weak Supervision*

# Few-Shot Imitation Learning from Weak Supervision

Given one teleoperated demonstration:



Learn a policy.



# Few-Shot Imitation Learning from Weak Supervision

Given one teleoperated demonstration:  
Given a video of a human:



Learn a policy.



# Learning to Learn from Weak Supervision

*meta-training*

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

# Learning to Learn from Weak Supervision

*meta-training*

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

**fully supervised**

# Learning to Learn from Weak Supervision

*meta-training*

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

**fully supervised**      **weakly supervised**

Yu\*, Finn\*, Xie, Dasari, Zhang,  
Abbeel, Levine RSS '18  
Grant, Finn, Peterson, Abbott,  
Levine, Darrell, Griffiths NIPS CIAI  
Workshop '17

# Learning to Learn from Weak Supervision

*meta-training*

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

**fully supervised**

*meta-test*

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

**weakly supervised**

# Learning to Learn from Weak Supervision

*meta-training*

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

**fully supervised**

**weakly supervised**

*meta-test*

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

**weakly supervised**

# Learning to Learn from Weak Supervision

*meta-training*

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

**fully supervised**

**weakly supervised**

*meta-test*

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

**weakly supervised**

What if the weakly supervised loss is unavailable?

Yu\*, Finn\*, Xie, Dasari, Zhang,  
Abbeel, Levine RSS '18  
Grant, Finn, Peterson, Abbott,  
Levine, Darrell, Griffiths NIPS CIAI  
Workshop '17

# Learning to Learn from Weak Supervision

**meta-training**

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

**fully supervised**

**meta-test**

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

**weakly supervised**

What if the weakly supervised loss is unavailable?

**imitation loss**

$$\mathcal{L} = \sum_t \|\pi_{\theta}(o_t) - a_t^*\|^2$$



Yu\*, Finn\*, Xie, Dasari, Zhang,  
Abbeel, Levine RSS '18  
Grant, Finn, Peterson, Abbott,  
Levine, Darrell, Griffiths NIPS CIAI  
Workshop '17

# Learning to Learn from Weak Supervision

**meta-training**

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

**fully supervised**

**meta-test**

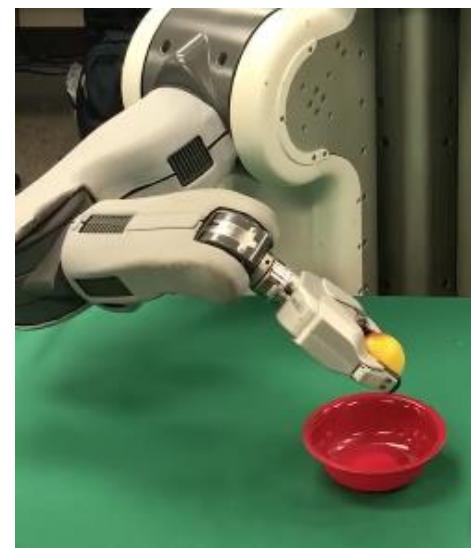
$$\theta' \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta)$$

**weakly supervised**

What if the weakly supervised loss is unavailable?

**imitation loss**

$$\mathcal{L} = \sum_t \|\pi_{\theta}(o_t) - a_t^*\|^2$$



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

# Learning to Learn from Weak Supervision

**meta-training**

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

**fully supervised**

**meta-test**

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

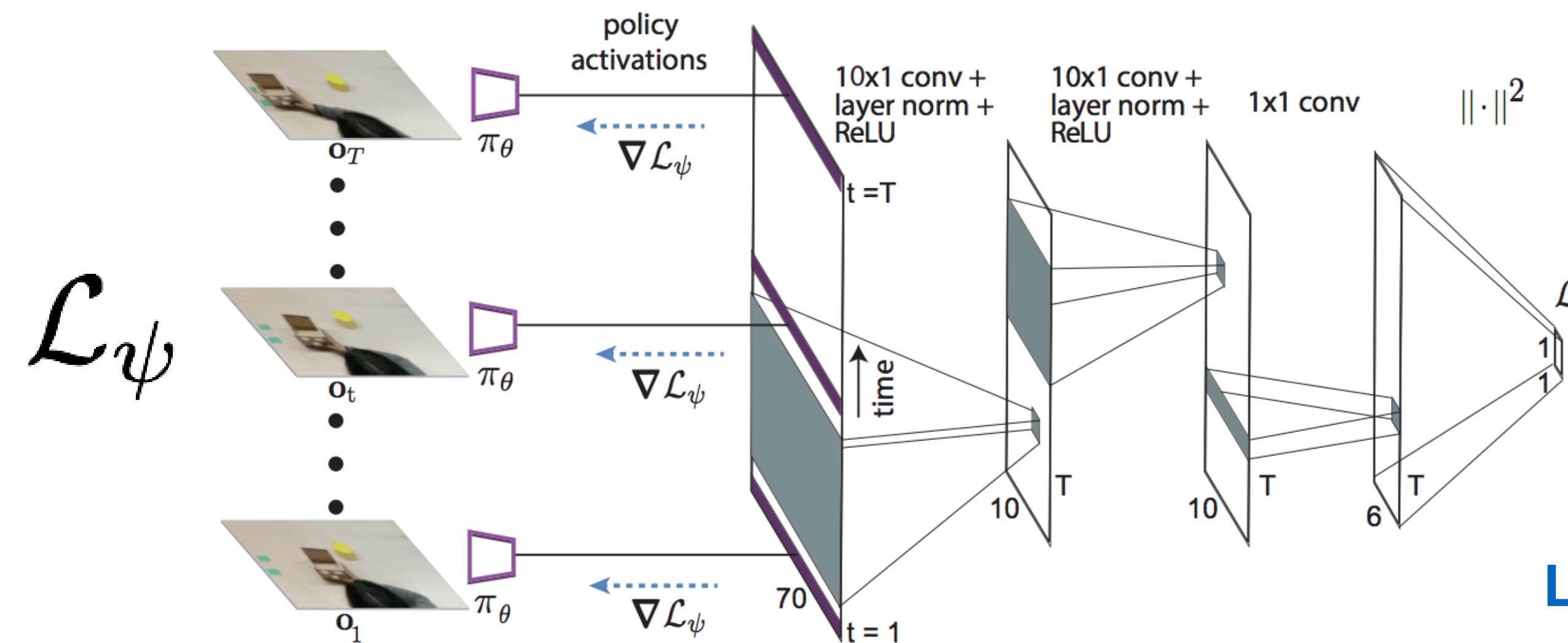
**weakly supervised**

What if the weakly supervised loss is unavailable?

**imitation loss**

$$\mathcal{L} = \sum_t \|\pi_{\theta}(o_t) - a_t^*\|^2$$

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



Yu\*, Finn\*, Xie, Dasari, Zhang,  
Abbeel, Levine RSS '18  
Grant, Finn, Peterson, Abbott,  
Levine, Darrell, Griffiths NIPS CIAI  
Workshop '17

# One-shot imitation from human video

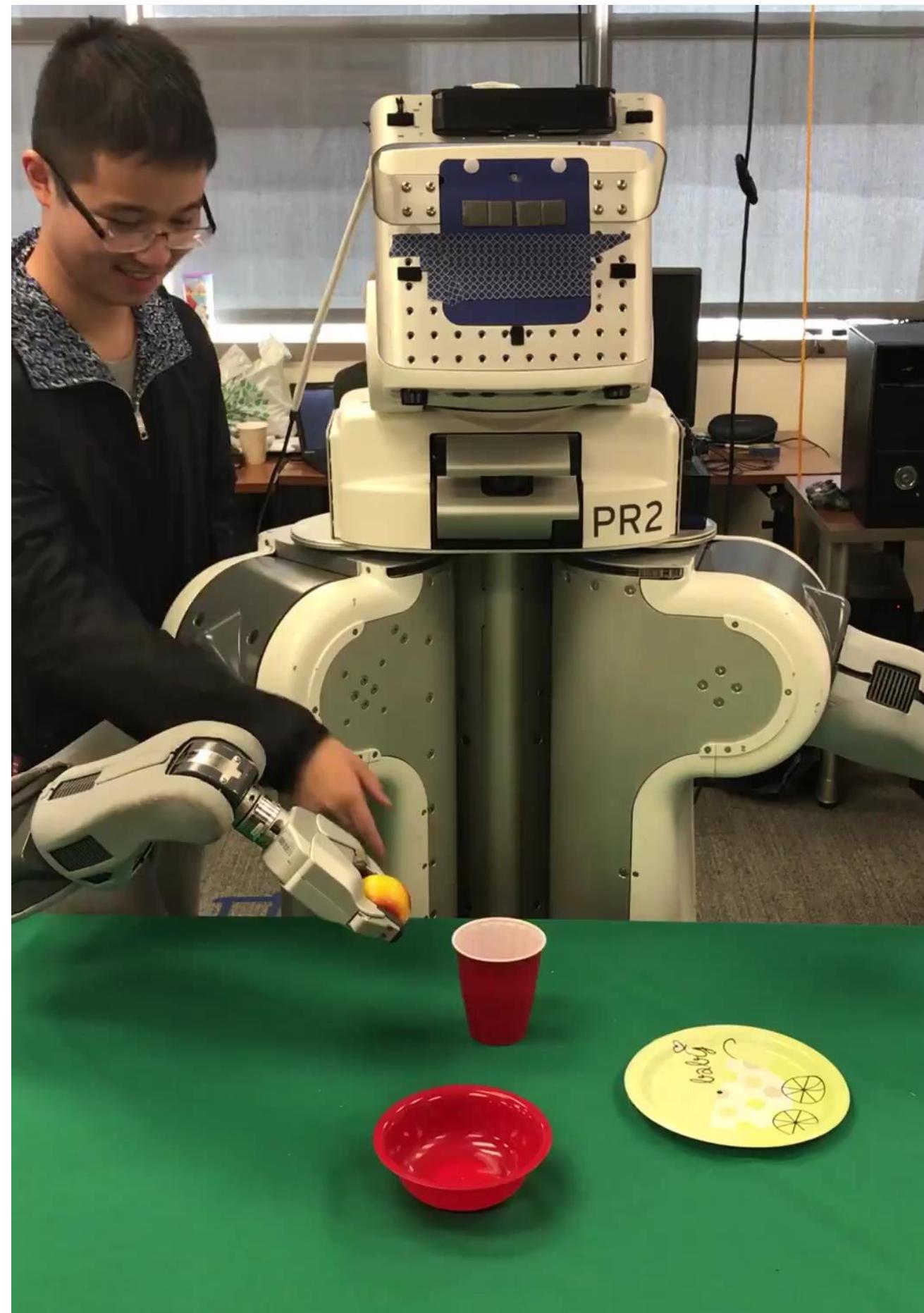
# One-shot imitation from human video

## input human demo



# One-shot imitation from human video

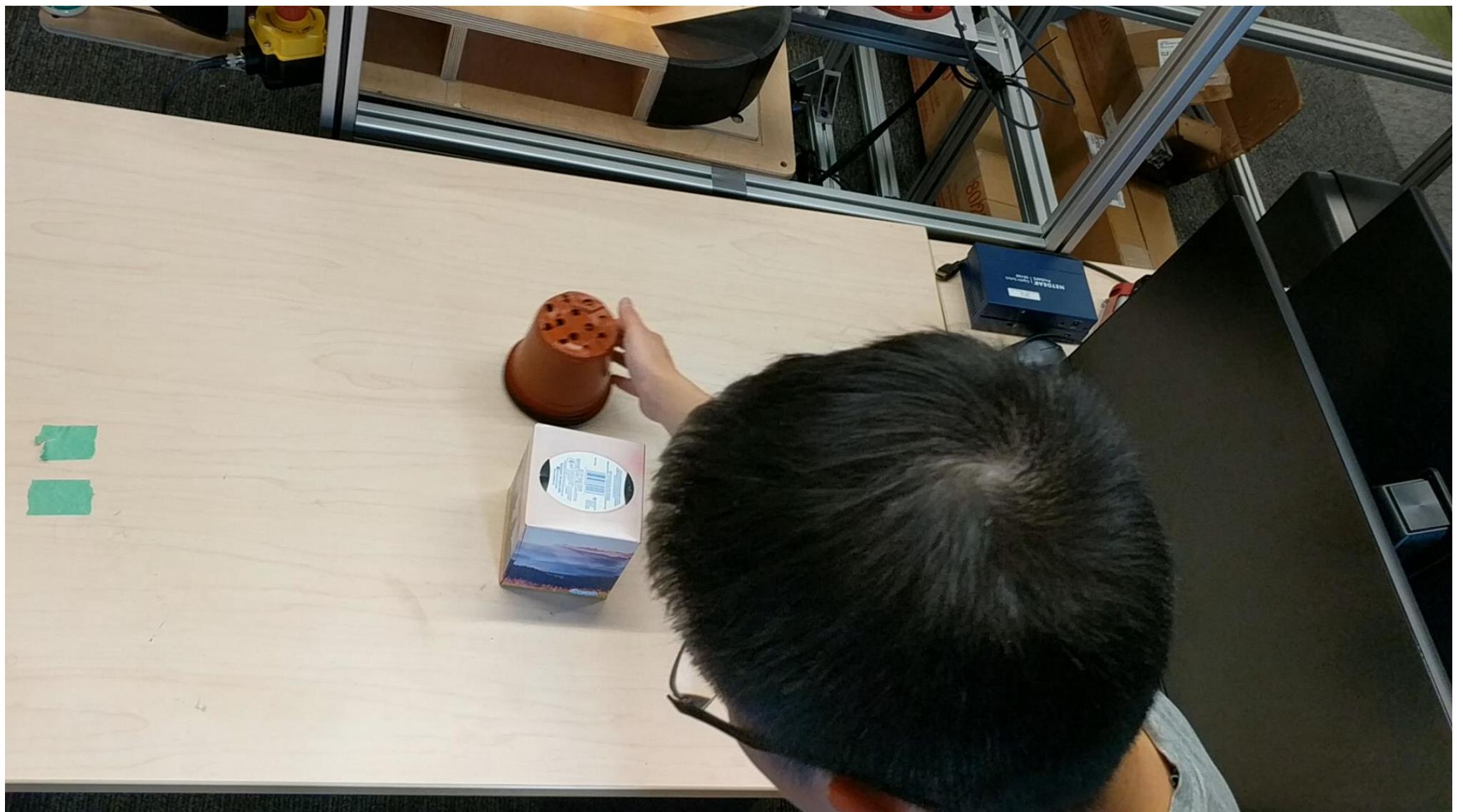
input human demo      resulting policy



# One-shot imitation from human video

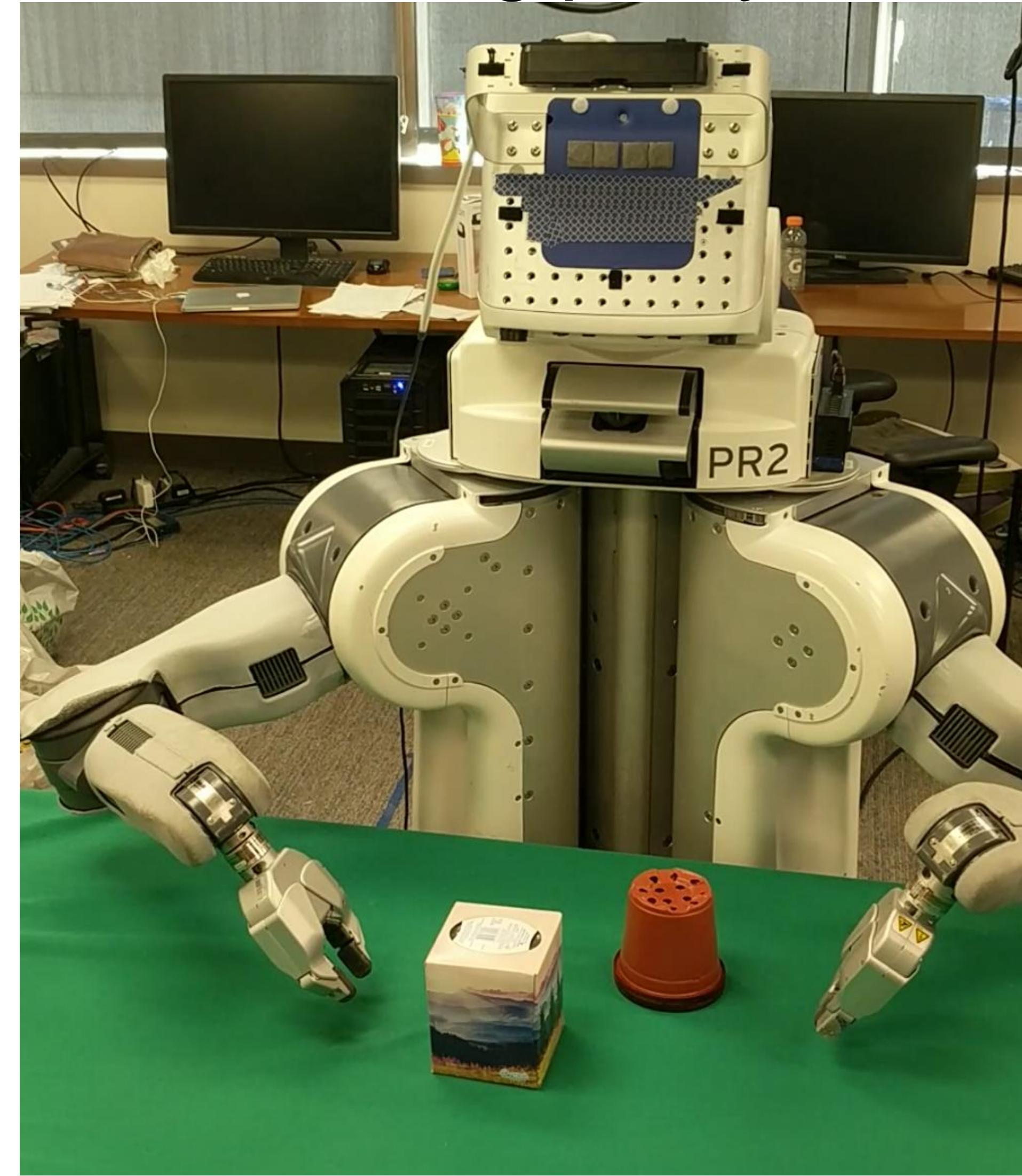
# One-shot imitation from human video

input human demo



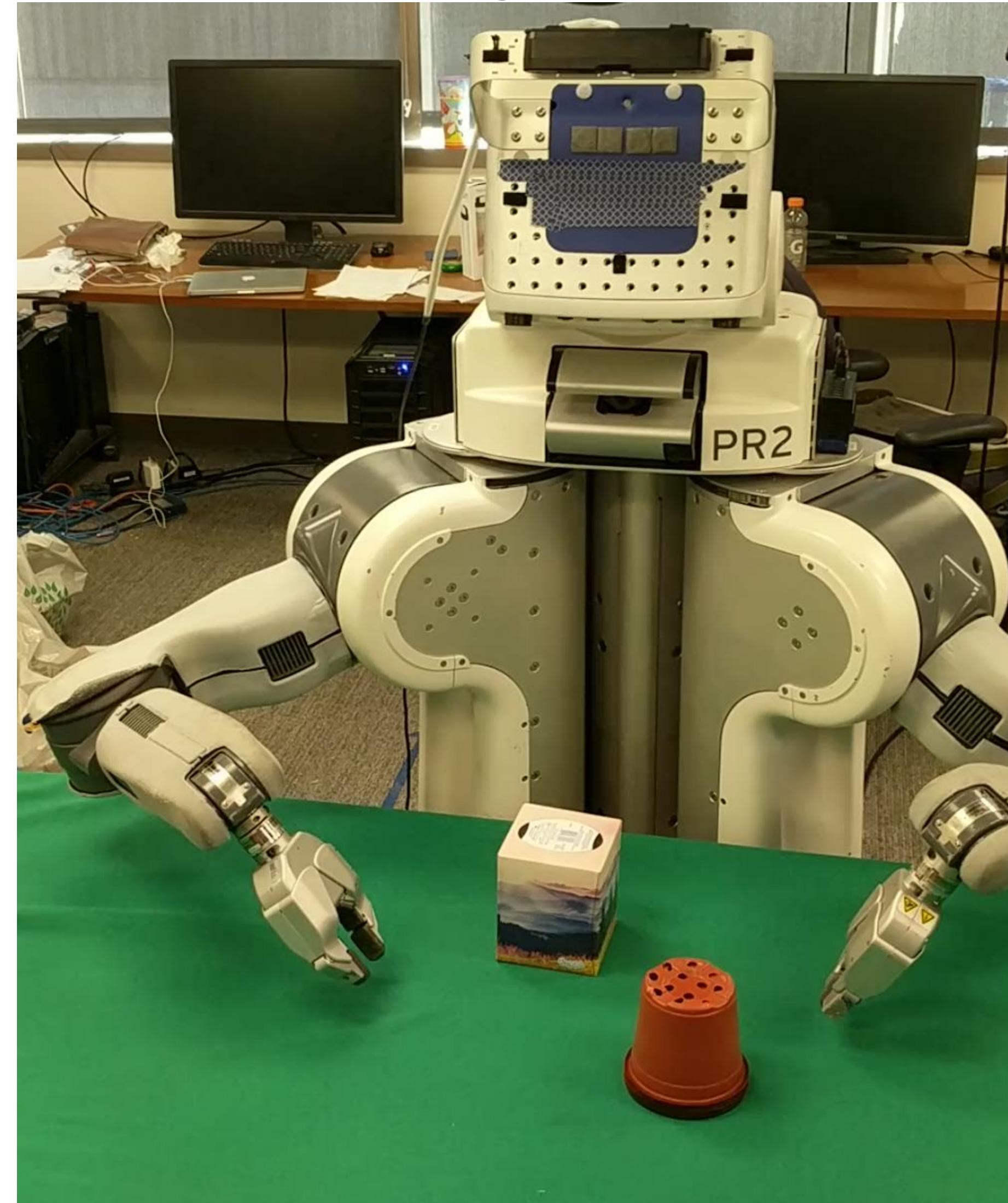
# One-shot imitation from human video resulting policy

input human demo



# One-shot imitation from human video resulting policy

input human demo



# Takeaways

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{y}_{\text{test}}$

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches
  - metric learning approach

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{x}_{\text{test}} \rightarrow \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches
  - metric learning approach
    - + works well for few-shot classification

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches
  - metric learning approach
    - + works well for few-shot classification
  - direct black box approach

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches
  - metric learning approach
    - + works well for few-shot classification
  - direct black box approach
    - + general, powerful

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches
  - metric learning approach
    - + works well for few-shot classification
  - direct black box approach
    - + general, powerful
    - data inefficient, poor extrapolation

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches
  - metric learning approach
    - + works well for few-shot classification
  - direct black box approach
    - + general, powerful
    - data inefficient, poor extrapolation
  - gradient-based approach

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{x}_{\text{test}} \rightarrow \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches
  - metric learning approach
    - + works well for few-shot classification
  - direct black box approach
    - + general, powerful
    - data inefficient, poor extrapolation
  - gradient-based approach
    - + general, good extrapolation

# Takeaways

- Meta-learning can be seen as learning a  
 $\mathcal{D}_{\text{train}} \xrightarrow{\text{function}} \mathbf{x}_{\text{test}} \rightarrow \mathbf{y}_{\text{test}}$
- Discussed three major classes of approaches
  - metric learning approach
    - + works well for few-shot classification
  - direct black box approach
    - + general, powerful
    - data inefficient, poor extrapolation
  - gradient-based approach
    - + general, good extrapolation
    - somewhat difficult optimization

**This class:**

1. Get large dataset
2. Get large compute (i.e. GPUs)
3. ?????
4. Profit!

Humans can learn with a very small amount of data. How?

***Do neural networks need a large dataset?***

## This class:

1. Get large dataset
2. Get large compute (i.e. GPUs)
3. ?????
4. Profit!

Humans can learn with a very small amount of data. How?

***Do neural networks need a large dataset?***

**Sort of.**

**This class:**

1. Get large dataset
2. Get large compute (i.e. GPUs)
3. ?????
4. Profit!

Humans can learn with a very small amount of data. How?

***Do neural networks need a large dataset?***

**Sort of.**

**Humans still require a lot of data, but not for each and every task.**

**This class:**

1. Get large dataset
2. Get large compute (i.e. GPUs)
3. ?????
4. Profit!

Humans can learn with a very small amount of data. How?

***Do neural networks need a large dataset?***

**Sort of.**

**Humans still require a lot of data, but not for each and every task.  
Data is amortized across tasks.**

# Outline

1. Applications of learning to learn
2. Problem formulation
3. Solution Classes:
  - a) metric-learning approach
  - b) direct black-box approach
  - c) gradient-based approach
4. **Open Questions / Problems**

# Open Problems in Meta-Learning

# Open Problems in Meta-Learning

- Where do the **tasks** come from?

# Open Problems in Meta-Learning

- Where do the **tasks** come from?
- Theoretically study of learning performance on **similar, but extrapolated tasks**

# Open Problems in Meta-Learning

- Where do the **tasks** come from?
- Theoretically study of learning performance on **similar, but extrapolated tasks**
- **Meta-RL** algorithms that are **consistent & universal**

# Open Problems in Meta-Learning

- Where do the **tasks** come from?
- Theoretically study of learning performance on **similar, but extrapolated tasks**
- **Meta-RL** algorithms that are **consistent & universal**
- **Benchmarks, datasets, environments** for meta-learning

# Open Problems in Meta-Learning

- Where do the **tasks** come from?
- Theoretically study of learning performance on **similar, but extrapolated tasks**
- **Meta-RL** algorithms that are **consistent & universal**
- **Benchmarks, datasets, environments** for meta-learning
- Meta-learning for **continual learning** (NIPS 2018 competition)

# Further Reading on Meta-Learning

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18
- **Gradient-based approach:** Finn et al. ICML '17

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18
- **Gradient-based approach:** Finn et al. ICML '17
- **Bayesian concept learning:** Tenenbaum thesis '99, Lake et al. Science '15

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18
- **Gradient-based approach:** Finn et al. ICML '17
- **Bayesian concept learning:** Tenenbaum thesis '99, Lake et al. Science '15
- **Meta-reinforcement learning:** J. Wang et al. '16, Y. Duan et al. '16,  
A. Gupta et al. '18, Clavera et al. '18, Houthooft et al. '18

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18
- **Gradient-based approach:** Finn et al. ICML '17
- **Bayesian concept learning:** Tenenbaum thesis '99, Lake et al. Science '15
- **Meta-reinforcement learning:** J. Wang et al. '16, Y. Duan et al. '16, A. Gupta et al. '18, Clavera et al. '18, Houthooft et al. '18
- **Few-shot generative modeling:** Reed et al. ICLR '18

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18
- **Gradient-based approach:** Finn et al. ICML '17
- **Bayesian concept learning:** Tenenbaum thesis '99, Lake et al. Science '15
- **Meta-reinforcement learning:** J. Wang et al. '16, Y. Duan et al. '16, A. Gupta et al. '18, Clavera et al. '18, Houthooft et al. '18
- **Few-shot generative modeling:** Reed et al. ICLR '18
- **One-shot visual imitation:** Yu et al. RSS '18

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18
- **Gradient-based approach:** Finn et al. ICML '17
- **Bayesian concept learning:** Tenenbaum thesis '99, Lake et al. Science '15
- **Meta-reinforcement learning:** J. Wang et al. '16, Y. Duan et al. '16, A. Gupta et al. '18, Clavera et al. '18, Houthooft et al. '18
- **Few-shot generative modeling:** Reed et al. ICLR '18
- **One-shot visual imitation:** Yu et al. RSS '18
- **Semi-supervised few-shot learning:** Ren et al. ICLR '18

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18
- **Gradient-based approach:** Finn et al. ICML '17
- **Bayesian concept learning:** Tenenbaum thesis '99, Lake et al. Science '15
- **Meta-reinforcement learning:** J. Wang et al. '16, Y. Duan et al. '16, A. Gupta et al. '18, Clavera et al. '18, Houthooft et al. '18
- **Few-shot generative modeling:** Reed et al. ICLR '18
- **One-shot visual imitation:** Yu et al. RSS '18
- **Semi-supervised few-shot learning:** Ren et al. ICLR '18
- **Learning unsupervised learning rules:** L. Metz et al. '18

# Further Reading on Meta-Learning

- **Metric learning:** Vinyals et al., Snell et al.
- **Black-box approach:** Hochreiter et al. '01, Santoro et al. ICML '16, Li & Malik arXiv '18, Ha et al. ICLR '17, Mishra et al. ICLR '18
- **Gradient-based approach:** Finn et al. ICML '17
- **Bayesian concept learning:** Tenenbaum thesis '99, Lake et al. Science '15
- **Meta-reinforcement learning:** J. Wang et al. '16, Y. Duan et al. '16, A. Gupta et al. '18, Clavera et al. '18, Houthooft et al. '18
- **Few-shot generative modeling:** Reed et al. ICLR '18
- **One-shot visual imitation:** Yu et al. RSS '18
- **Semi-supervised few-shot learning:** Ren et al. ICLR '18
- **Learning unsupervised learning rules:** L. Metz et al. '18  
excludes architecture search, learning update rules, hyperparameter optimization

# Questions?

## Further Resources:

Learning to Learn blog post:

<http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/>

Learning to Optimize blog post:

<http://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/>

Meta-RL lecture (Deep RL course): <https://youtu.be/Xe9bktyYB34>







# Universal Function Approximation Theorem

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

A neural network with one hidden layer of finite width can approximate any continuous function

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

A neural network with one hidden layer of finite width can approximate any continuous function

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

“universal function approximator”

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

A neural network with one hidden layer of finite width can approximate any continuous function

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

“universal function approximator”

How can we define a notion of universality / expressive power for meta-learning

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

A neural network with one hidden layer of finite width can approximate any continuous function

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

“universal function approximator”

How can we define a notion of universality / expressive power for meta-learning

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

A neural network with one hidden layer of finite width can approximate any continuous function

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

“universal function approximator”

How can we define a notion of universality / expressive power for meta-learning

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

“universal learning procedure approximator”

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

A neural network with one hidden layer of finite width can approximate any continuous function

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

“universal function approximator”

How can we define a notion of universality / expressive power for meta-learning

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

“universal learning procedure approximator”

Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

Learned optimizer

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$$

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

A neural network with one hidden layer of finite width can approximate any continuous function

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

“universal function approximator”

How can we define a notion of universality / expressive power for meta-learning

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

“universal learning procedure approximator”

Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

Learned optimizer

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$$

With sufficient depth, both are universal learning procedure approximators.

# Universal Function Approximation Theorem

Hornik et al. '89, Cybenko '89, Funahashi '89

A neural network with one hidden layer of finite width can approximate any continuous function

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

“universal function approximator”

How can we define a notion of universality / expressive power for meta-learning

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

“universal learning procedure approximator”

Recurrent network

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

Learned optimizer

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; g(\mathcal{D}_{\text{train}}; \theta))$$

With sufficient depth, both are universal learning procedure approximators.

Are we losing expressive power when using MAML?

# Can we interpret MAML in a probabilistic framework?

# Can we interpret MAML in a probabilistic framework?

meta-learning  $\approx$  learning a prior

# Can we interpret MAML in a probabilistic framework?

meta-learning  $\approx$  learning a prior

***Bayesian concept learning***

# Can we interpret MAML in a probabilistic framework?

meta-learning  $\approx$  learning a prior

***Bayesian concept learning***

[Tenenbaum '99, Fei-Fei et al. '03, Lawrence & Platt '04, ...]

# Can we interpret MAML in a probabilistic framework?

meta-learning  $\approx$  learning a prior

## ***Bayesian concept learning***

[Tenenbaum '99, Fei-Fei et al. '03, Lawrence & Platt '04, ...]

formulate few-shot learning as probabilistic inference  
problem

# Can we interpret MAML in a probabilistic framework?

meta-learning  $\approx$  learning a prior

## ***Bayesian concept learning***

[Tenenbaum '99, Fei-Fei et al. '03, Lawrence & Platt '04, ...]

formulate few-shot learning as probabilistic inference  
problem

+ can effectively generalize from limited evidence

# Can we interpret MAML in a probabilistic framework?

meta-learning  $\approx$  learning a prior

## ***Bayesian concept learning***

[Tenenbaum '99, Fei-Fei et al. '03, Lawrence & Platt '04, ...]

formulate few-shot learning as probabilistic inference  
problem

+ can effectively generalize from limited evidence

- hard to scale to complex models

# Can we interpret MAML in a probabilistic framework?

# Can we interpret MAML in a probabilistic framework?



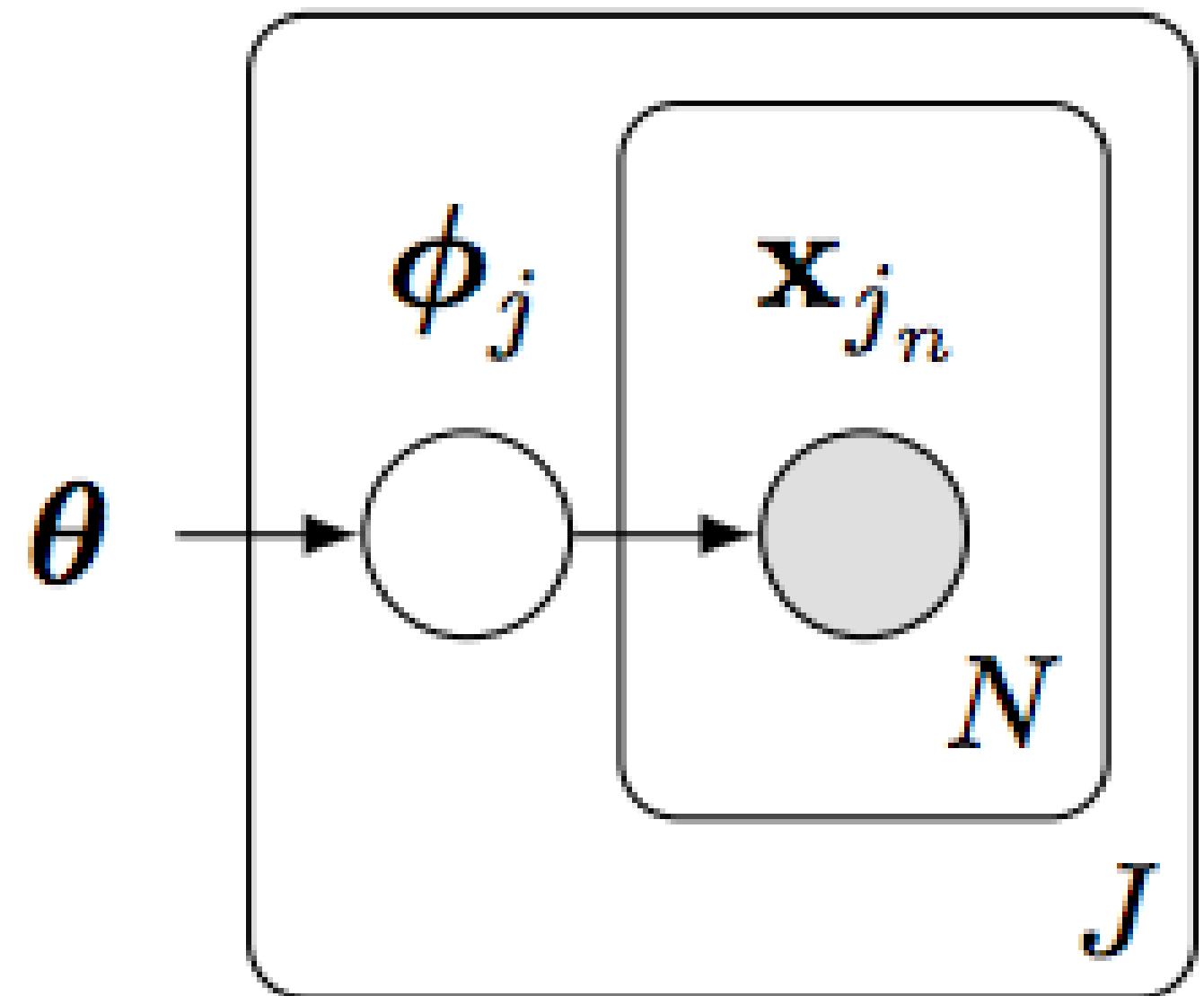
# Can we interpret MAML in a probabilistic framework?

## Bayesian meta-learning approach



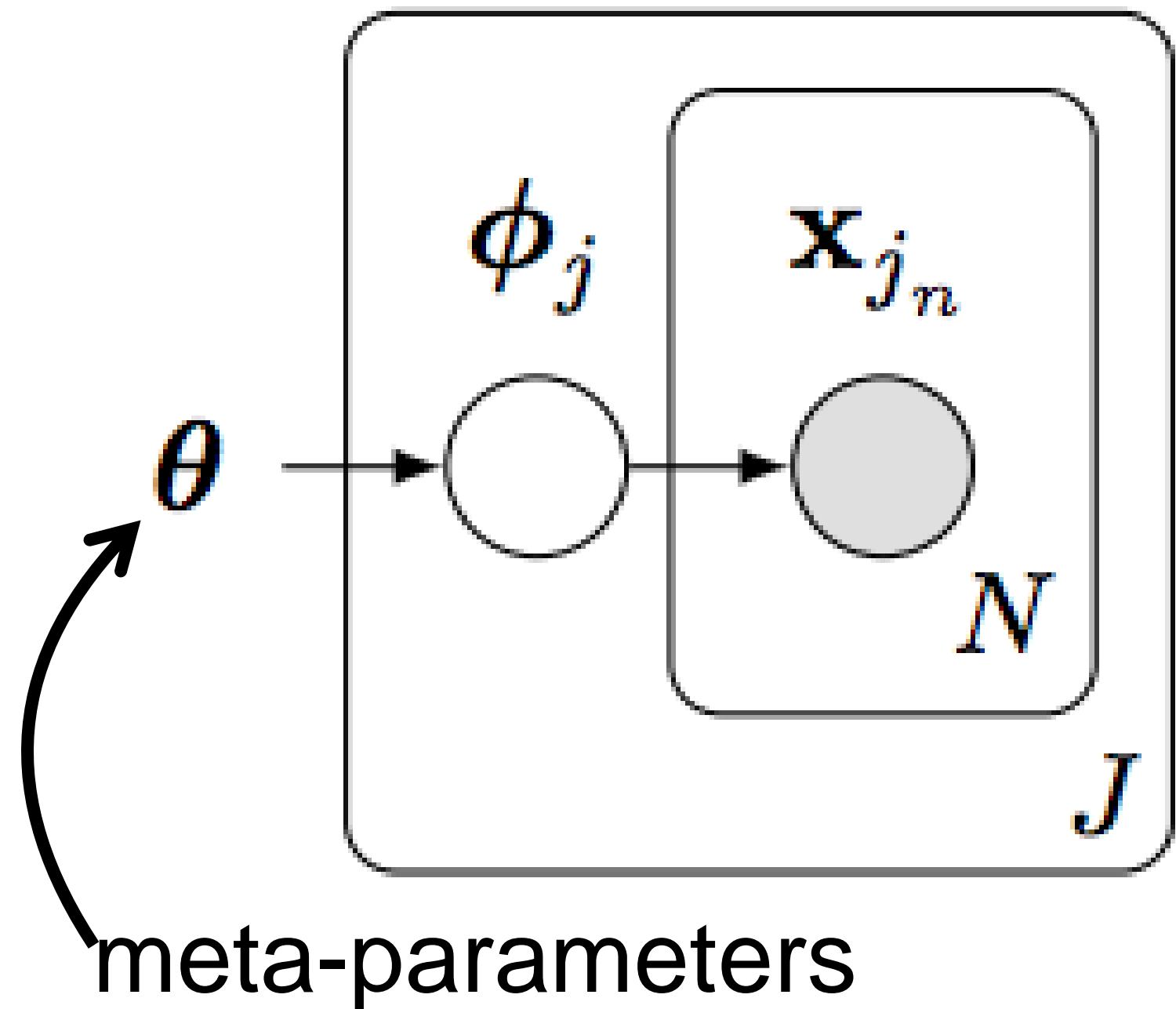
# Can we interpret MAML in a probabilistic framework?

Bayesian meta-learning approach



# Can we interpret MAML in a probabilistic framework?

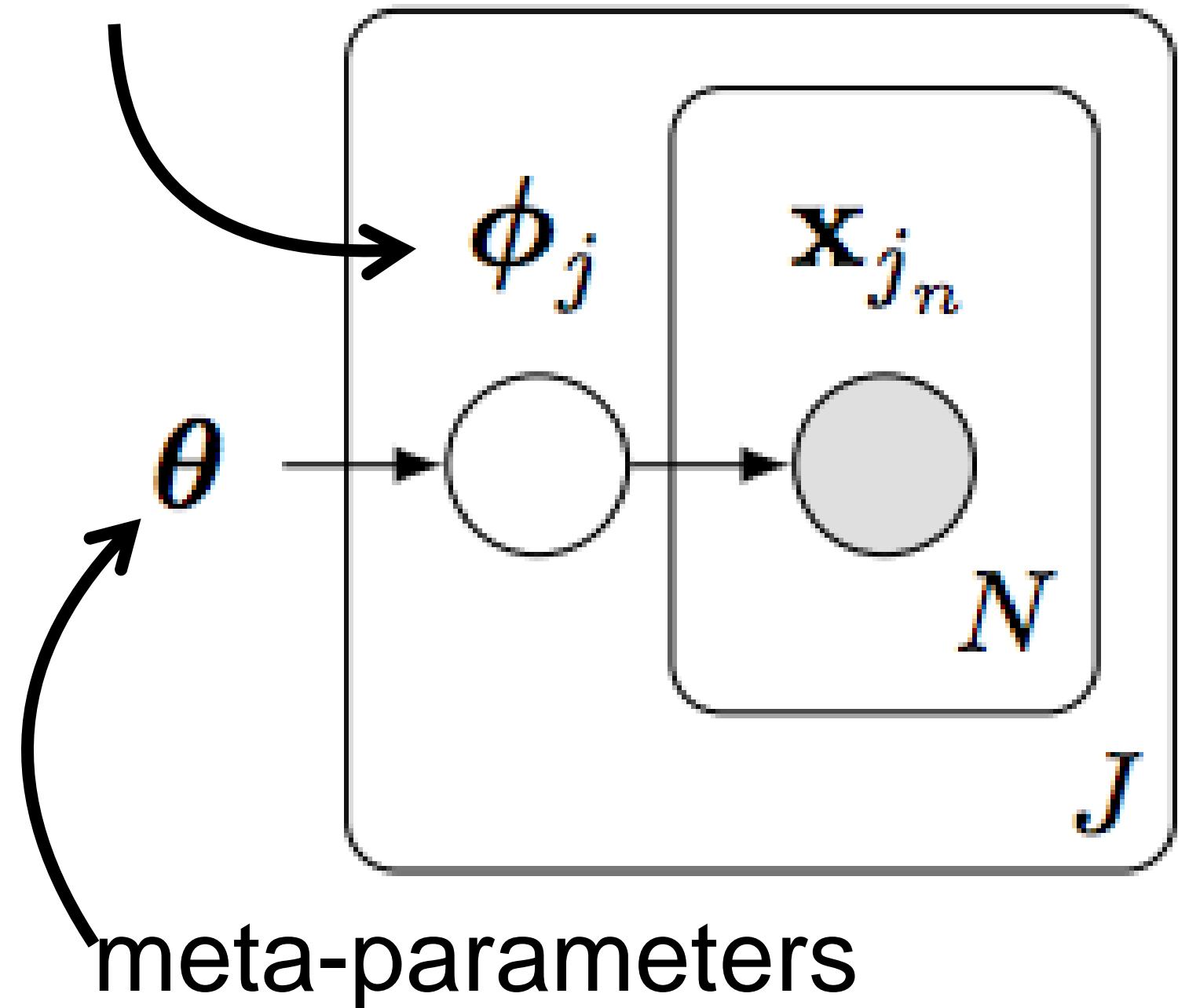
Bayesian meta-learning approach



# Can we interpret MAML in a probabilistic framework?

Bayesian meta-learning approach

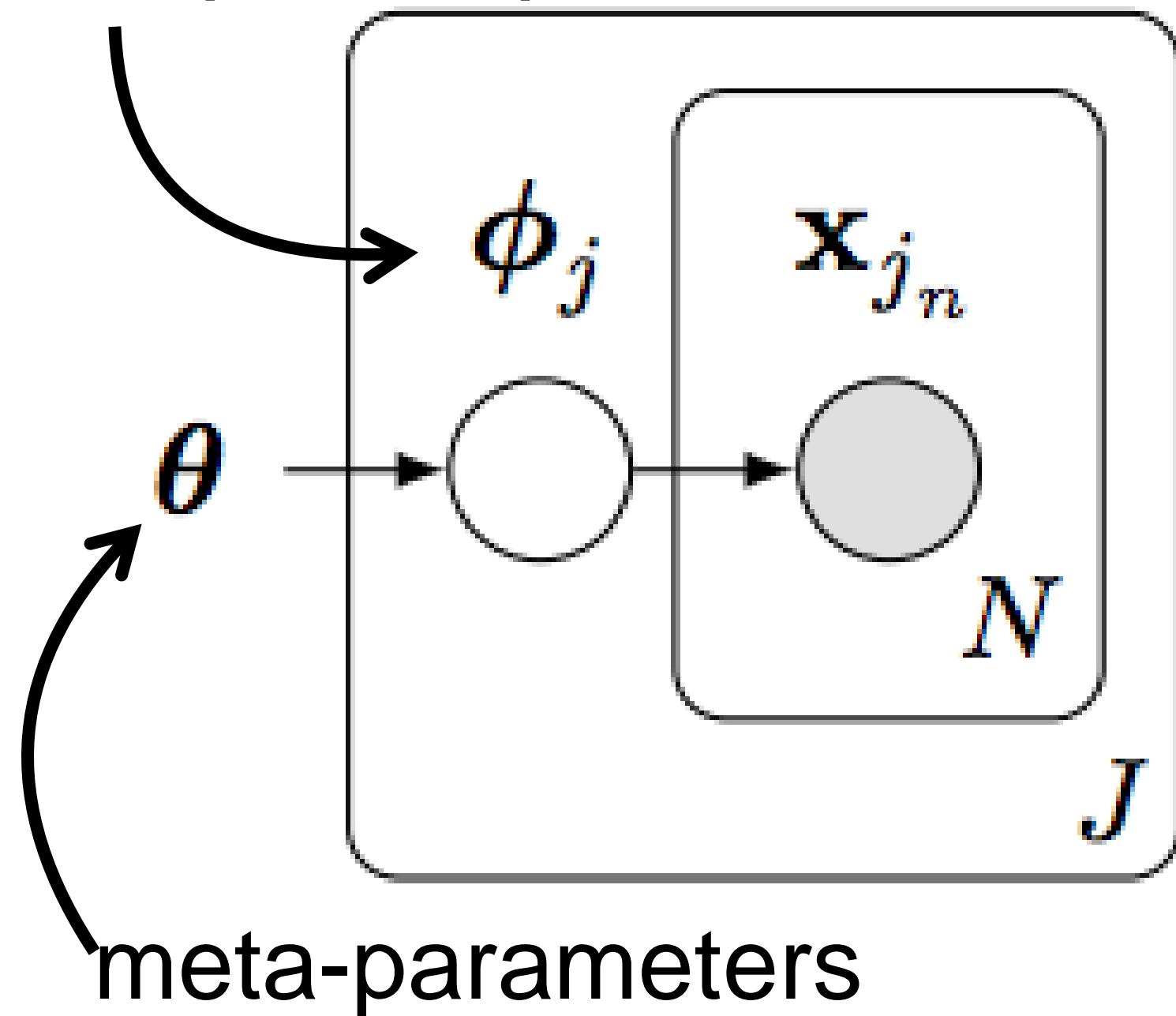
task-specific parameters



# Can we interpret MAML in a probabilistic framework?

## Bayesian meta-learning approach

task-specific parameters



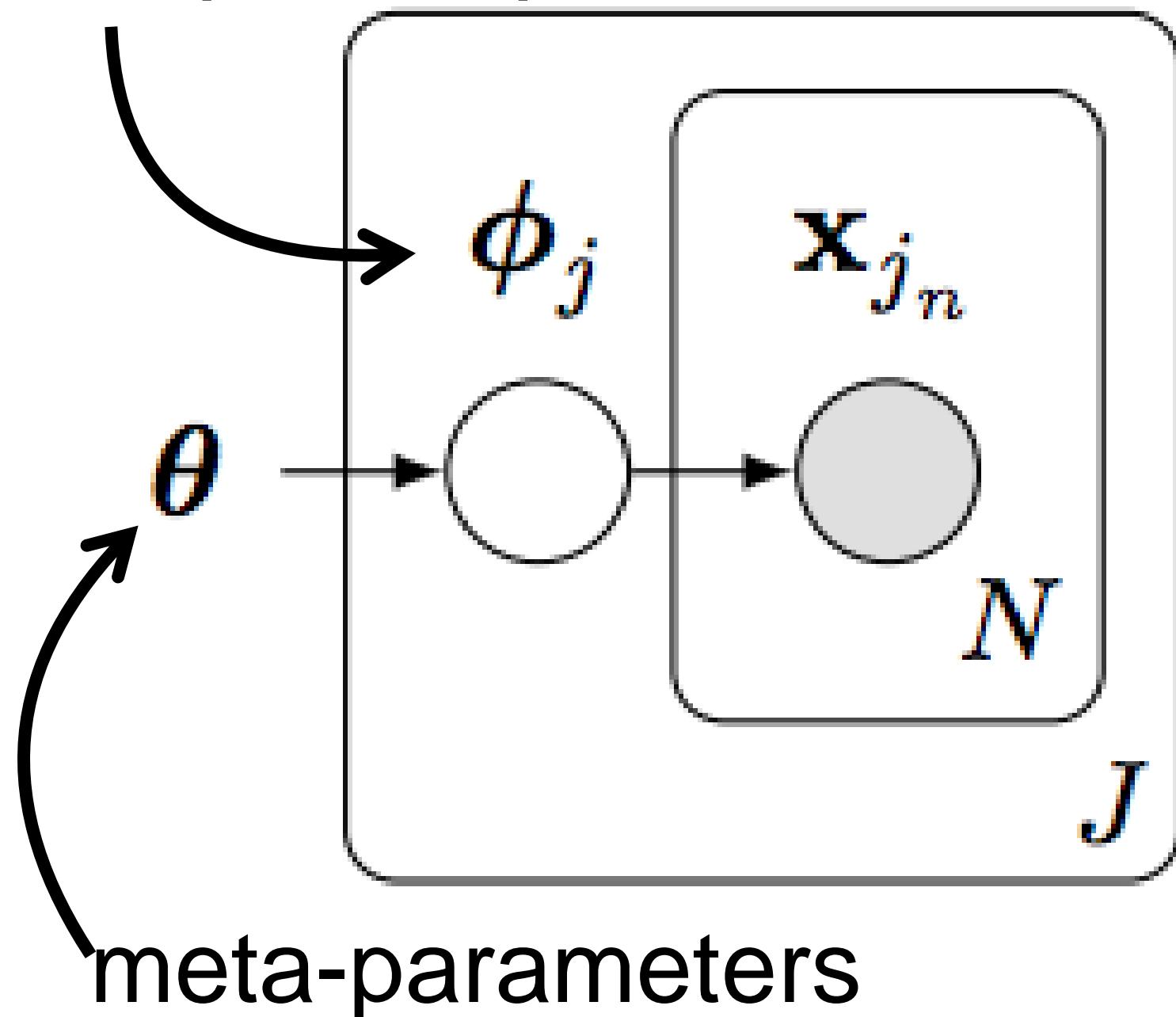
$$\max_{\theta} \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \theta)$$



# Can we interpret MAML in a probabilistic framework?

## Bayesian meta-learning approach

task-specific parameters



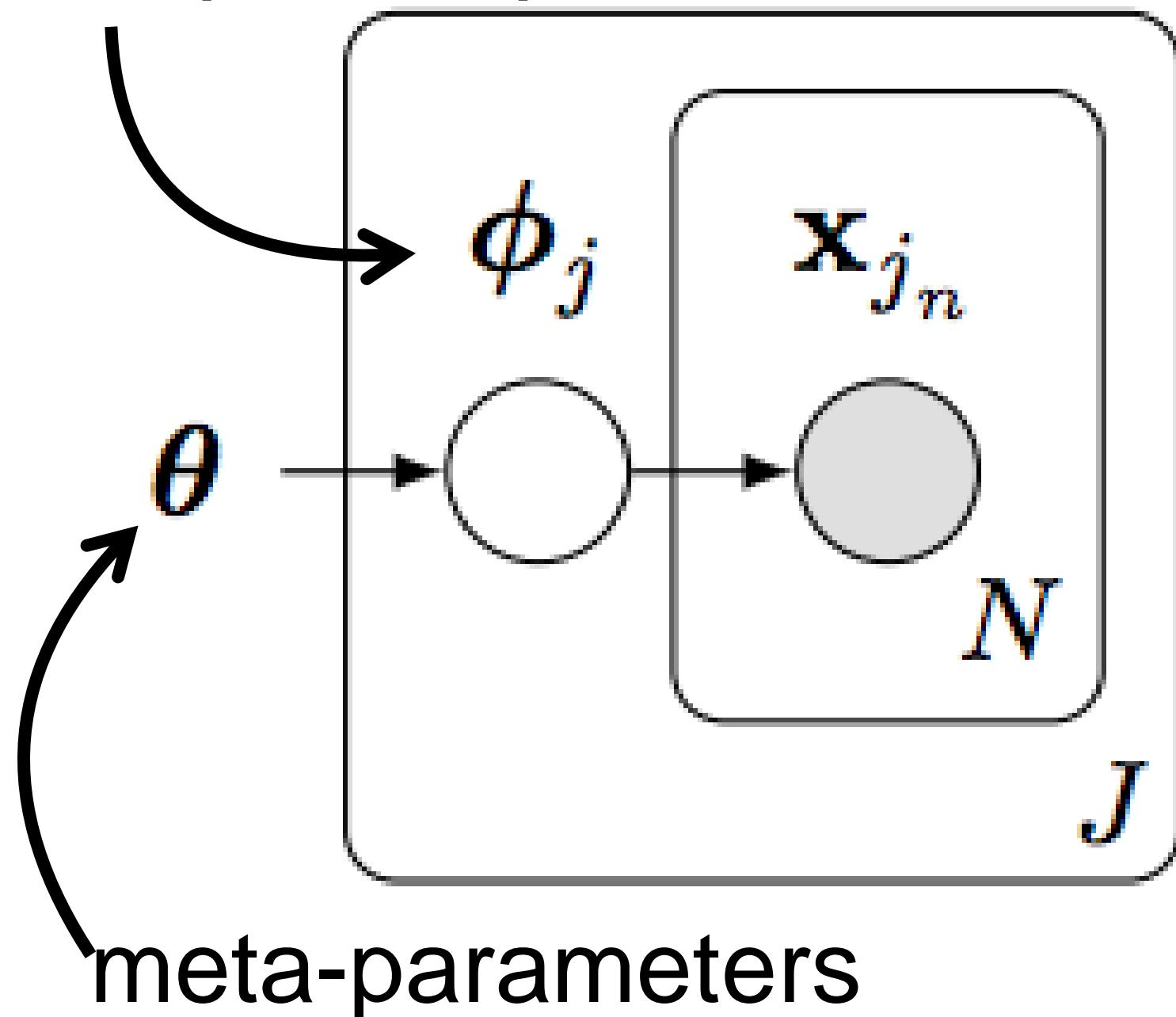
$$\begin{aligned} \max_{\theta} \prod p(\mathcal{D}_{\text{train}}^{(j)} | \theta) \\ = \prod_j \int p(\mathcal{D}_{\text{train}}^{(j)} | \phi_j) p(\phi_j | \theta) d\phi_j \end{aligned}$$



# Can we interpret MAML in a probabilistic framework?

## Bayesian meta-learning approach

task-specific parameters



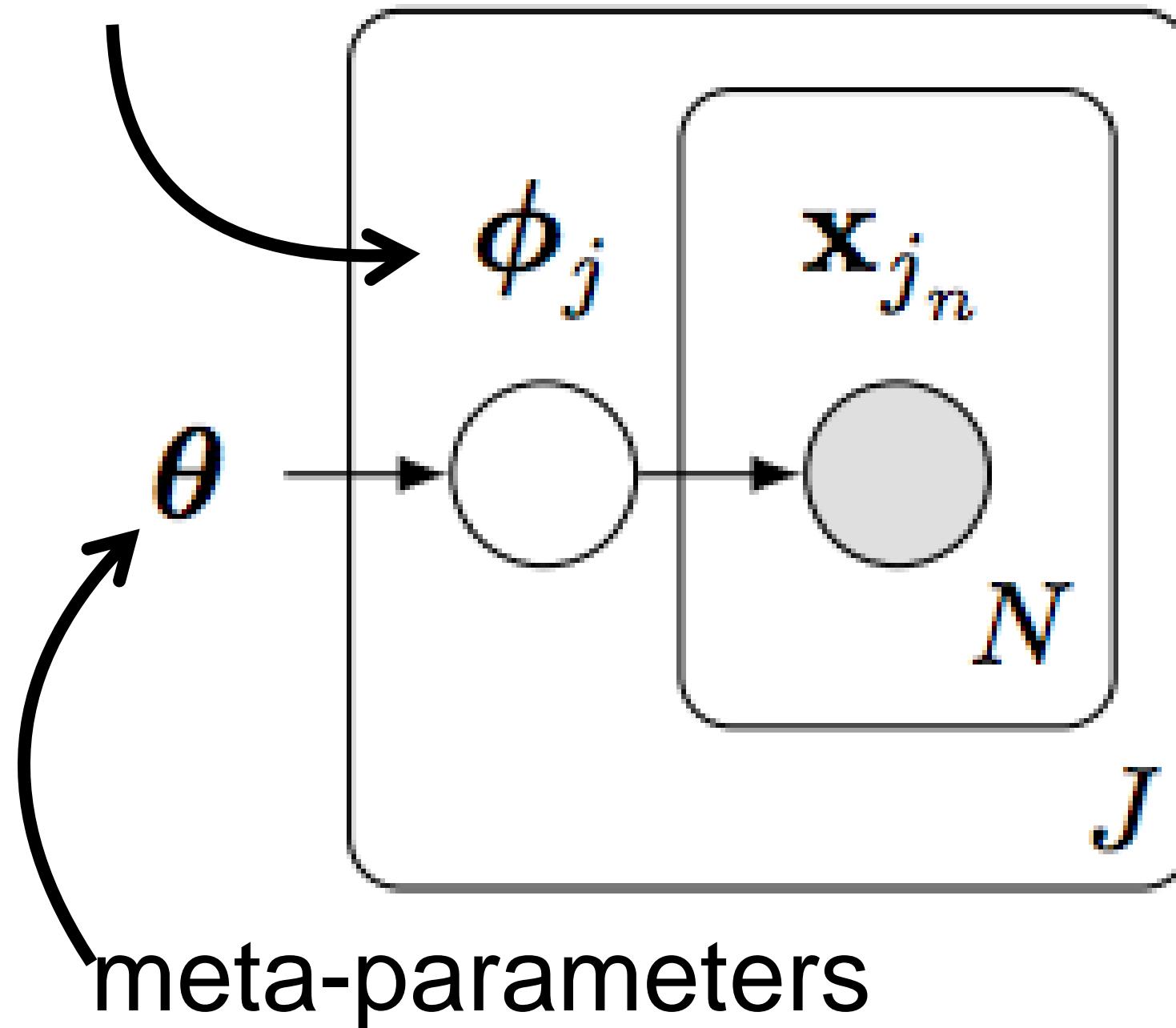
$$\begin{aligned} \max_{\theta} \prod p(\mathcal{D}_{\text{train}}^{(j)} | \theta) \\ = \prod_j \int p(\mathcal{D}_{\text{train}}^{(j)} | \phi_j) p(\phi_j | \theta) d\phi_j \end{aligned} \quad (\text{empirical Bayes})$$



# Can we interpret MAML in a probabilistic framework?

## Bayesian meta-learning approach

task-specific parameters



$$\begin{aligned} \max_{\theta} \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \theta) \\ = \prod_j \int p(\mathcal{D}_{\text{train}}^{(j)} | \phi_j) p(\phi_j | \theta) d\phi_j \\ \approx \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \hat{\phi}_j) p(\hat{\phi}_j | \theta) \end{aligned} \quad (\text{empirical Bayes})$$

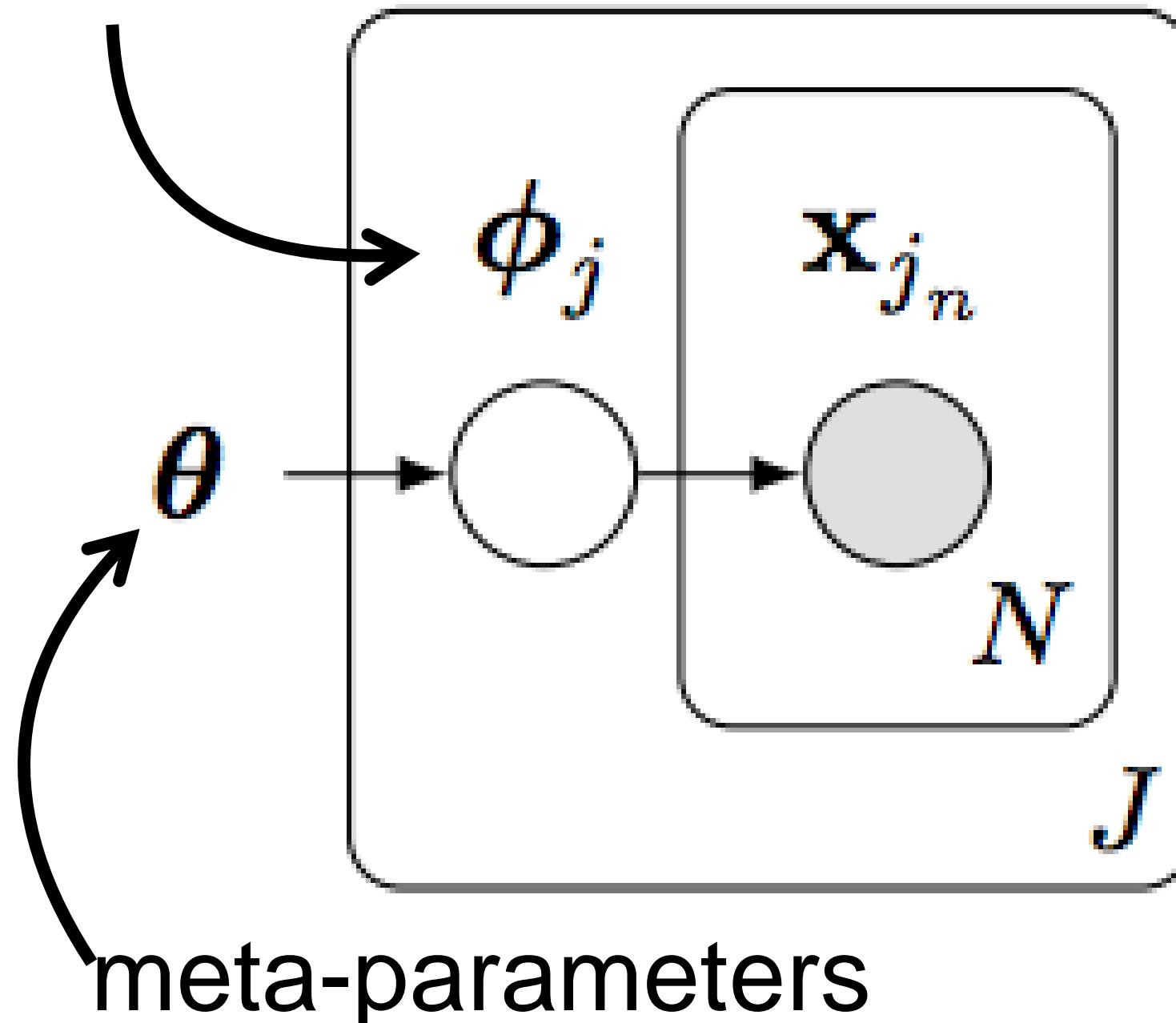
MAP estimate



# Can we interpret MAML in a probabilistic framework?

## Bayesian meta-learning approach

task-specific parameters



$$\begin{aligned} \max_{\theta} \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \theta) \\ = \prod_j \int p(\mathcal{D}_{\text{train}}^{(j)} | \phi_j) p(\phi_j | \theta) d\phi_j \\ \approx \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \hat{\phi}_j) p(\hat{\phi}_j | \theta) \end{aligned} \quad (\text{empirical Bayes})$$

MAP estimate

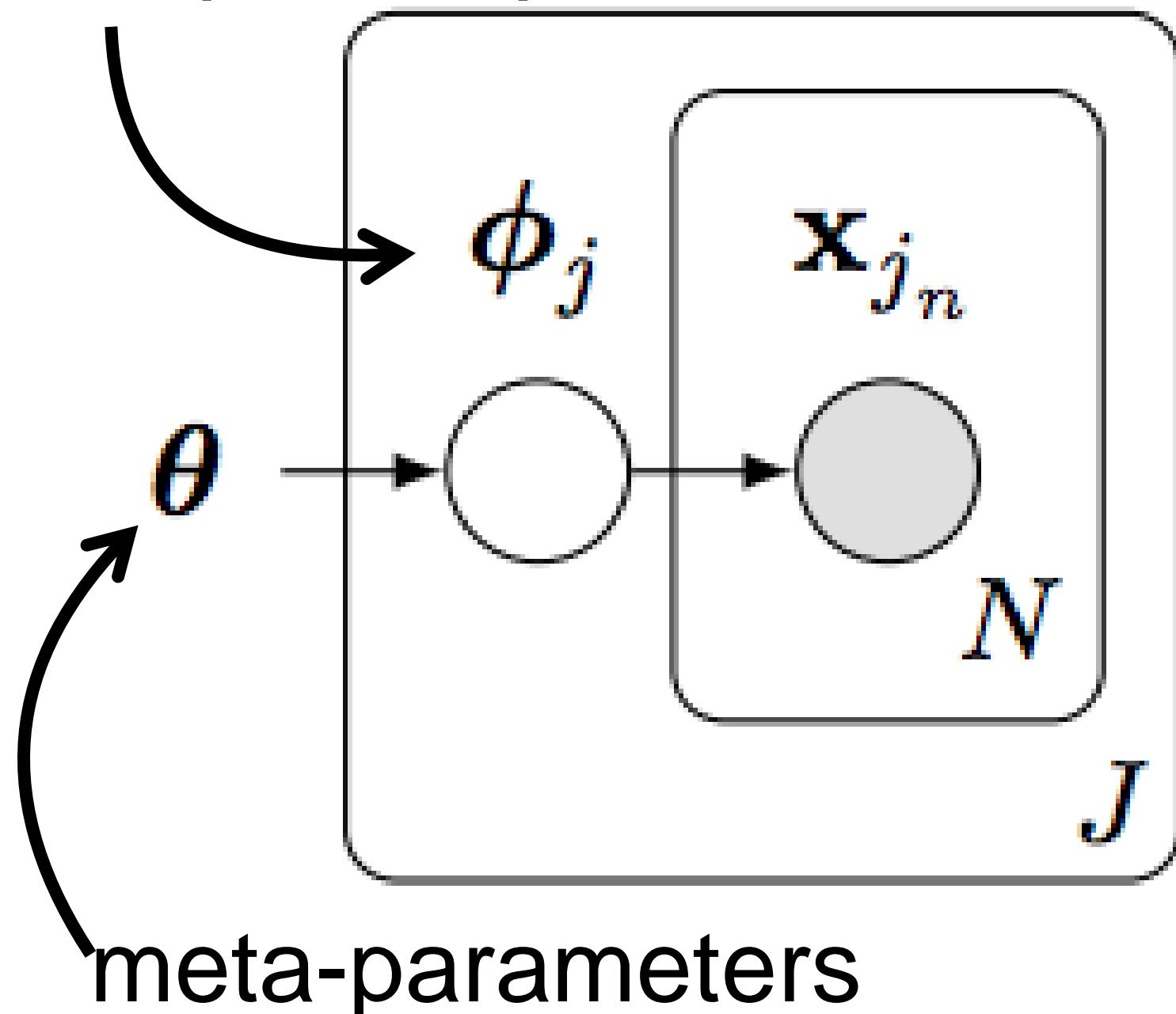
How to compute MAP estimate?



# Can we interpret MAML in a probabilistic framework?

## Bayesian meta-learning approach

task-specific parameters



meta-parameters

$$\begin{aligned} \max_{\theta} \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \theta) \\ = \prod_j \int p(\mathcal{D}_{\text{train}}^{(j)} | \phi_j) p(\phi_j | \theta) d\phi_j \\ (\text{empirical Bayes}) \\ \approx \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \hat{\phi}_j) p(\hat{\phi}_j | \theta) \end{aligned}$$

MAP estimate

## How to compute MAP estimate?

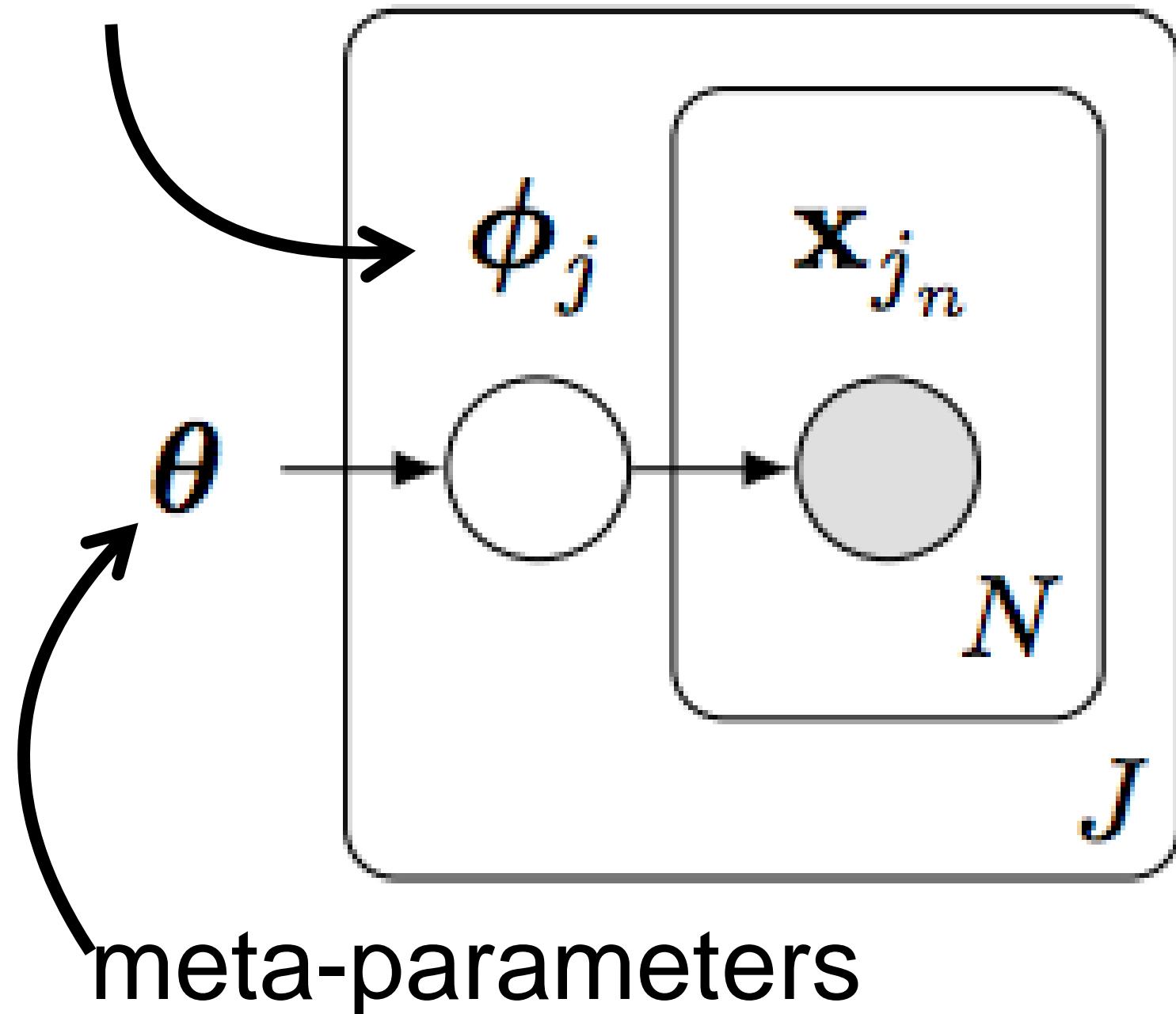
Gradient descent with early stopping = MAP inference under Gaussian prior with mean at initial parameters [Santos '96]  
(exact in linear case, approximate in nonlinear case)



# Can we interpret MAML in a probabilistic framework?

## Bayesian meta-learning approach

task-specific parameters



meta-parameters

$$\begin{aligned} \max_{\theta} \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \theta) \\ = \prod_j \int p(\mathcal{D}_{\text{train}}^{(j)} | \phi_j) p(\phi_j | \theta) d\phi_j \\ \approx \prod_j p(\mathcal{D}_{\text{train}}^{(j)} | \hat{\phi}_j) p(\hat{\phi}_j | \theta) \end{aligned} \quad (\text{empirical Bayes})$$

MAP estimate

## How to compute MAP estimate?

Gradient descent with early stopping = MAP inference under Gaussian prior with mean at initial parameters [Santos '96]  
(exact in linear case, approximate in nonlinear case)

MAML approximates hierarchical Bayesian inference. [Grant et al. '17]



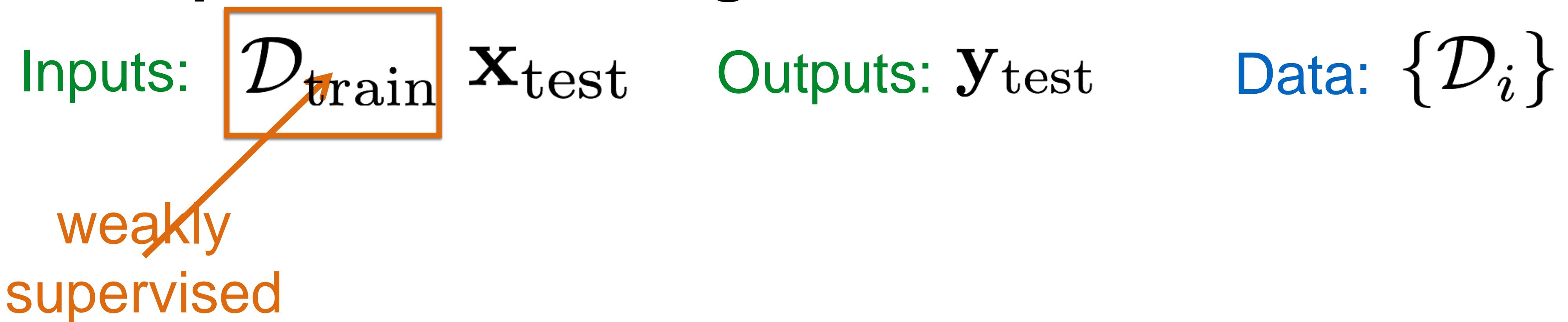
# Learning to Learn from Weak Supervision

## Meta-Supervised Learning:

Inputs:  $\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}}$     Outputs:  $\mathbf{y}_{\text{test}}$     Data:  $\{\mathcal{D}_i\}$

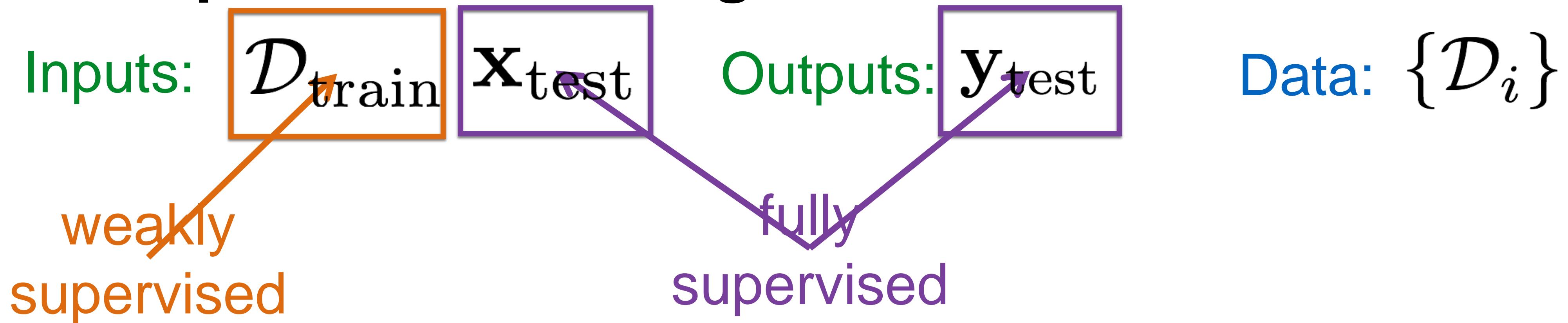
# Learning to Learn from Weak Supervision

## Meta-Supervised Learning:



# Learning to Learn from Weak Supervision

## Meta-Supervised Learning:



# Learning to Learn from Weak Supervision

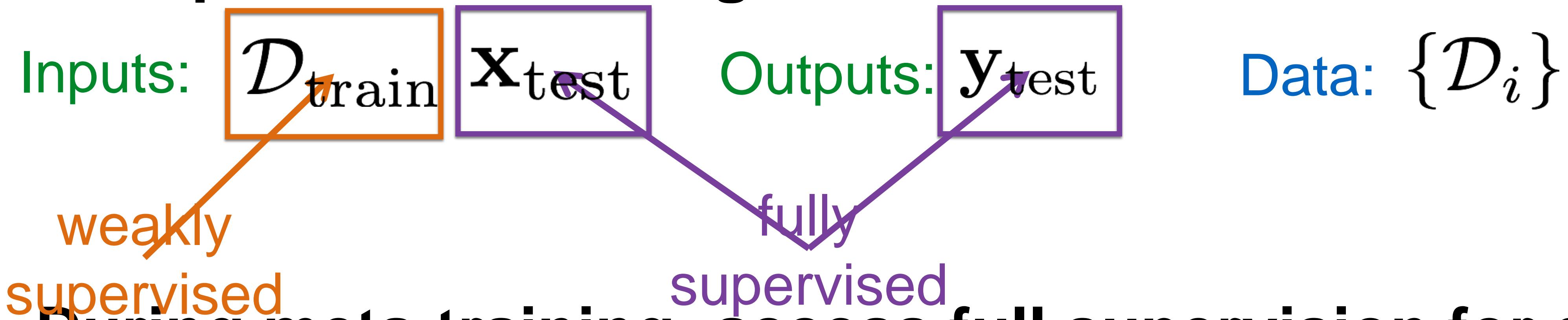
## Meta-Supervised Learning:

Inputs:  $\mathcal{D}_{\text{train}}$   $\mathbf{x}_{\text{test}}$  Outputs:  $\mathbf{y}_{\text{test}}$  Data:  $\{\mathcal{D}_i\}$

~~weakly supervised~~ **During meta-training, access full supervision for each task**

# Learning to Learn from Weak Supervision

## Meta-Supervised Learning:

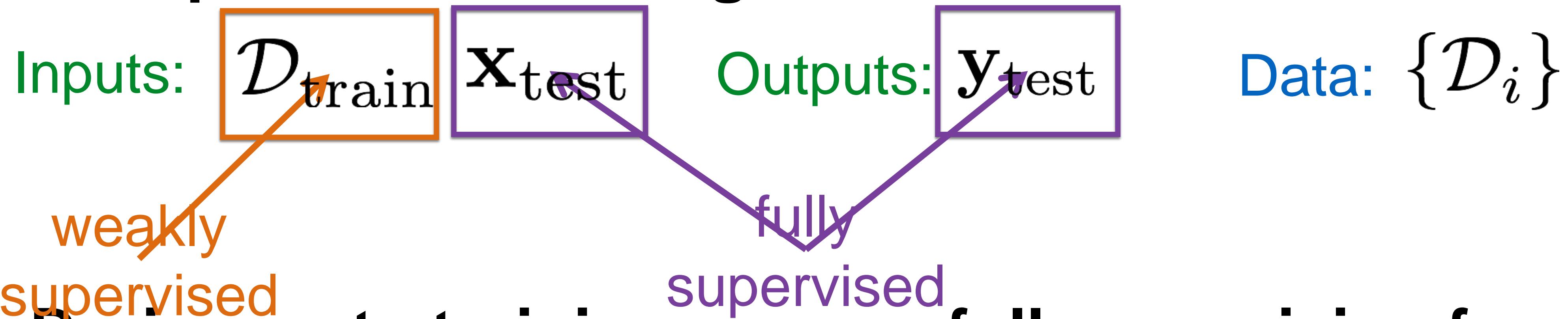


**During meta-training: access full supervision for each task**

**During meta-testing: only use weakly-supervised datapoints**

# Learning to Learn from Weak Supervision

## Meta-Supervised Learning:



**During meta-training: access full supervision for each task**

**During meta-testing: only use weakly-supervised datapoints**

*With MAML:* 
$$\min_{\theta} \sum \mathcal{L}_v(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$$

*Key insight* inner loss can be different than outer loss

# Weak Supervision Results

# Weak Supervision Results

- Learning from positive examples  
Grant, Finn, Peterson, Abbott, Levine, Darrell, Griffiths, NIPS '17 CIAI workshop

# Weak Supervision Results

- Learning from positive examples
  - Grant, Finn, Peterson, Abbott, Levine, Darrell, Griffiths, NIPS '17 CIAI workshop
- One-shot Imitation from human video
  - (in preparation, with Yu, Abbeel, Levine)

# Typical Objective of Few-Shot Learning

**Image recognition**

Given 1 example of 5 classes:



Classify new  
examples



# Typical Objective of Few-Shot Learning

**Image recognition**

Given 1 example of 5 classes:



Classify new  
examples



**Human Concept Learning**

# Typical Objective of Few-Shot Learning

## Image recognition

Given 1 example of 5 classes:



Classify new examples



## Human Concept Learning

Given 1 positive example:



Classify new examples:



# Typical Objective of Few-Shot Learning

## Image recognition

Given 1 example of 5 classes:



Classify new examples



## Human Concept Learning

Given 1 positive example:



Classify new examples:



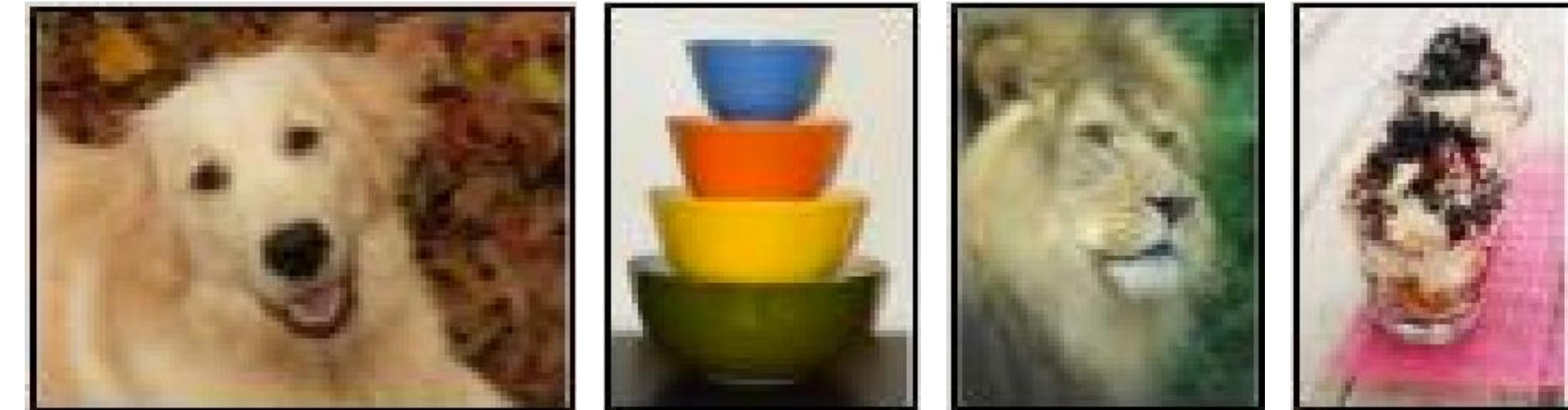
Beyond how humans learn, this setting is also more interesting.

# Human Concept Learning

Given 1 positive  
example:



Classify new examples:

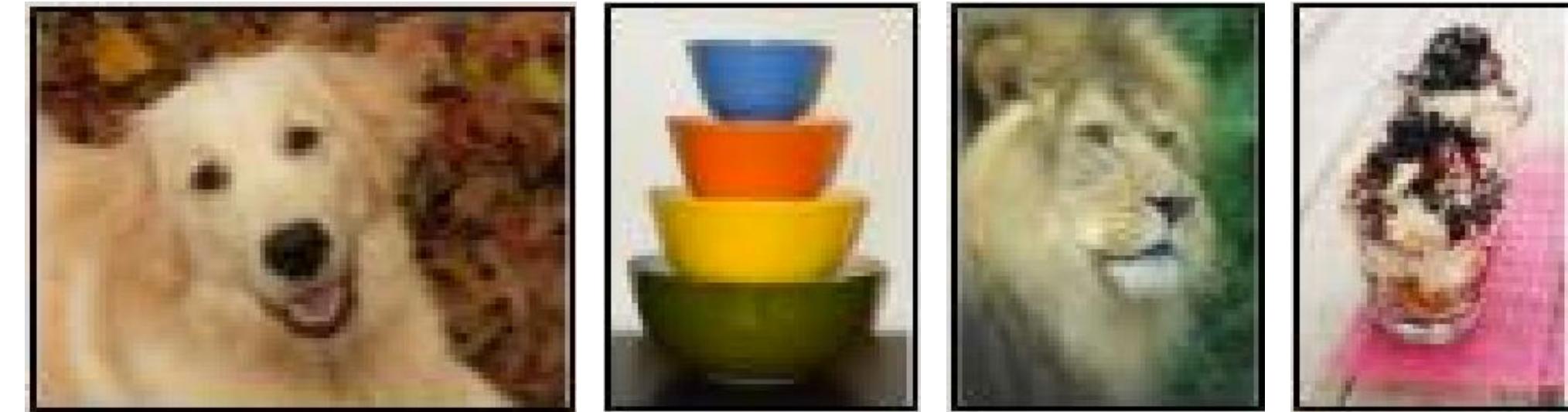


# Human Concept Learning

Given 1 positive  
example:



Classify new examples:



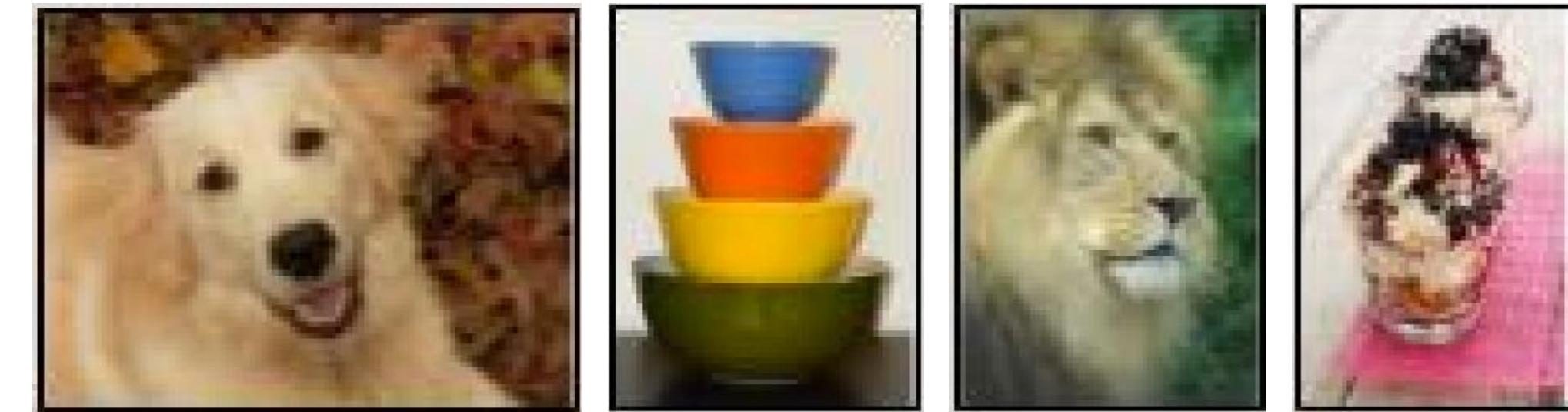
$$\min_{\theta} \sum \mathcal{L}_{\text{v}}(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$$

# Human Concept Learning

Given 1 positive example:



Classify new examples:



$$\min_{\theta} \sum \mathcal{L}_v(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$$



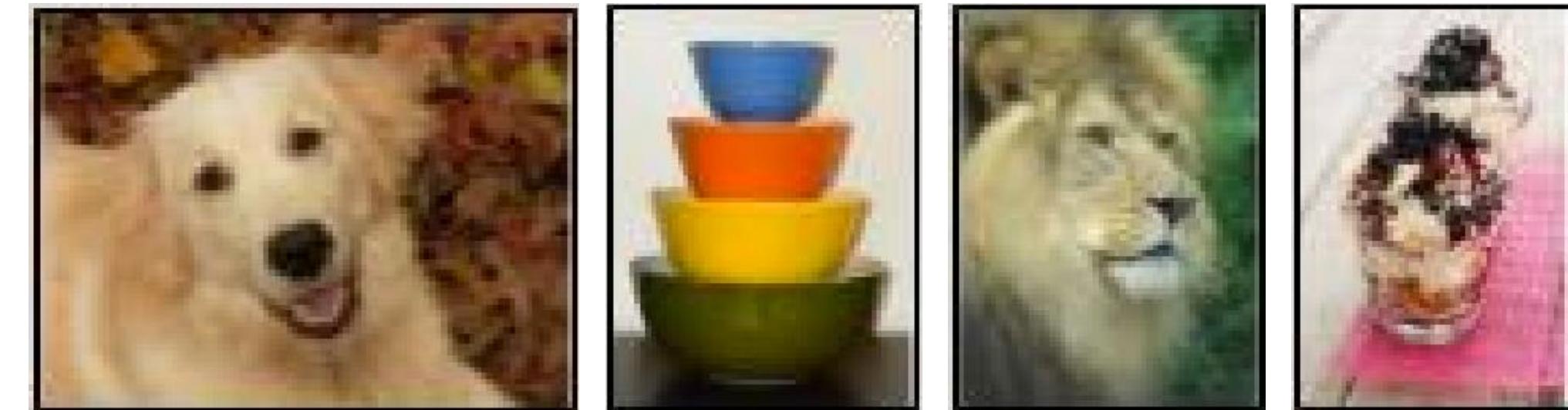
only positive examples

# Human Concept Learning

Given 1 positive example:



Classify new examples:



$$\min_{\theta} \sum \mathcal{L}_v(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$$

both positive & negatives

only positive examples

# Human Concept Learning

Given 1 positive example:



Classify new examples:



$$\min_{\theta} \sum \mathcal{L}_v(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$$

both positive & negatives

only positive examples

A large curly brace is positioned under the summation symbol in the equation, spanning from the first term to the last term. Two arrows point upwards from the text "both positive & negatives" and "only positive examples" to the brace, indicating that the first term in the sum represents training on both types of examples while the second term represents training on only positive examples.

**Why does this make sense?**

# Human Concept Learning

Given 1 positive example:



Classify new examples:



$$\min_{\theta} \sum \mathcal{L}_v(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$$

both positive & negatives

only positive examples

**Why does this make sense?**

MAML approximates hierarchical Bayesian inference

# Human Concept Learning

Given 1 positive example:



Classify new examples:



$$\min_{\theta} \sum \mathcal{L}_v(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$$

both positive & negatives

only positive examples

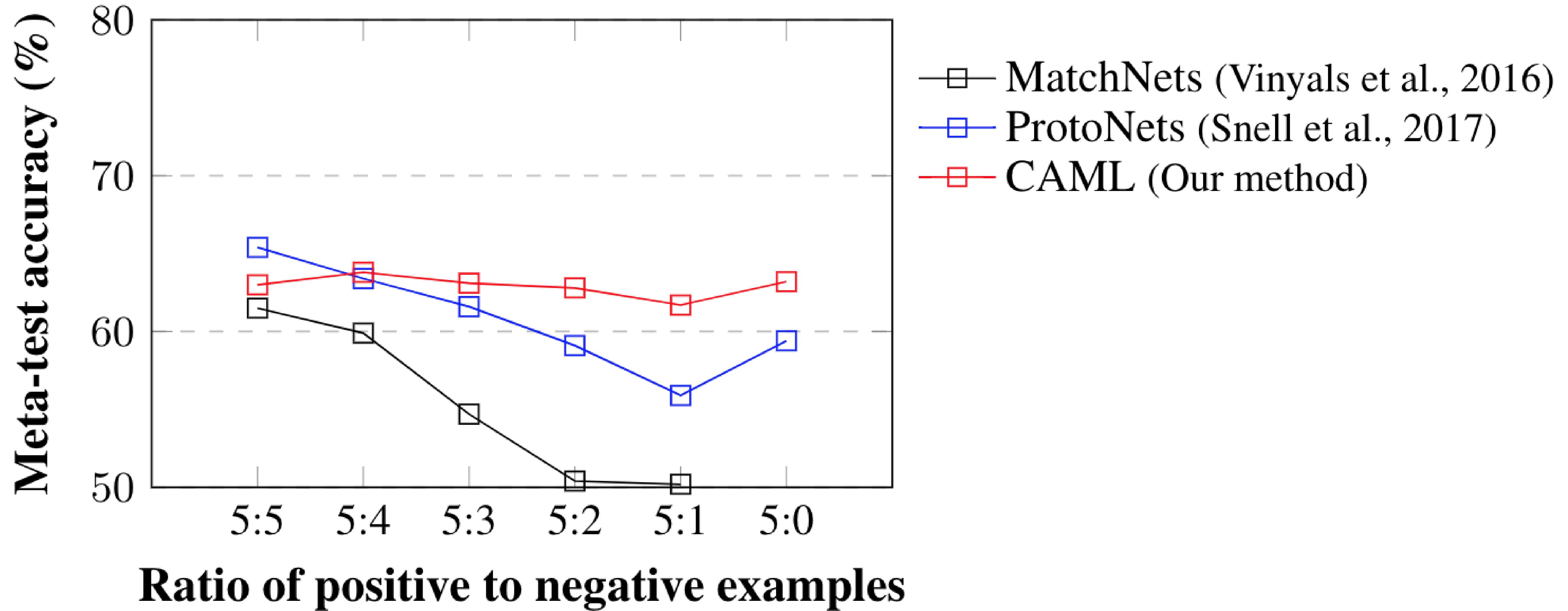
**Why does this make sense?**

MAML approximates hierarchical Bayesian inference

**Concept Acquisition through Meta-Learning (CAML)**

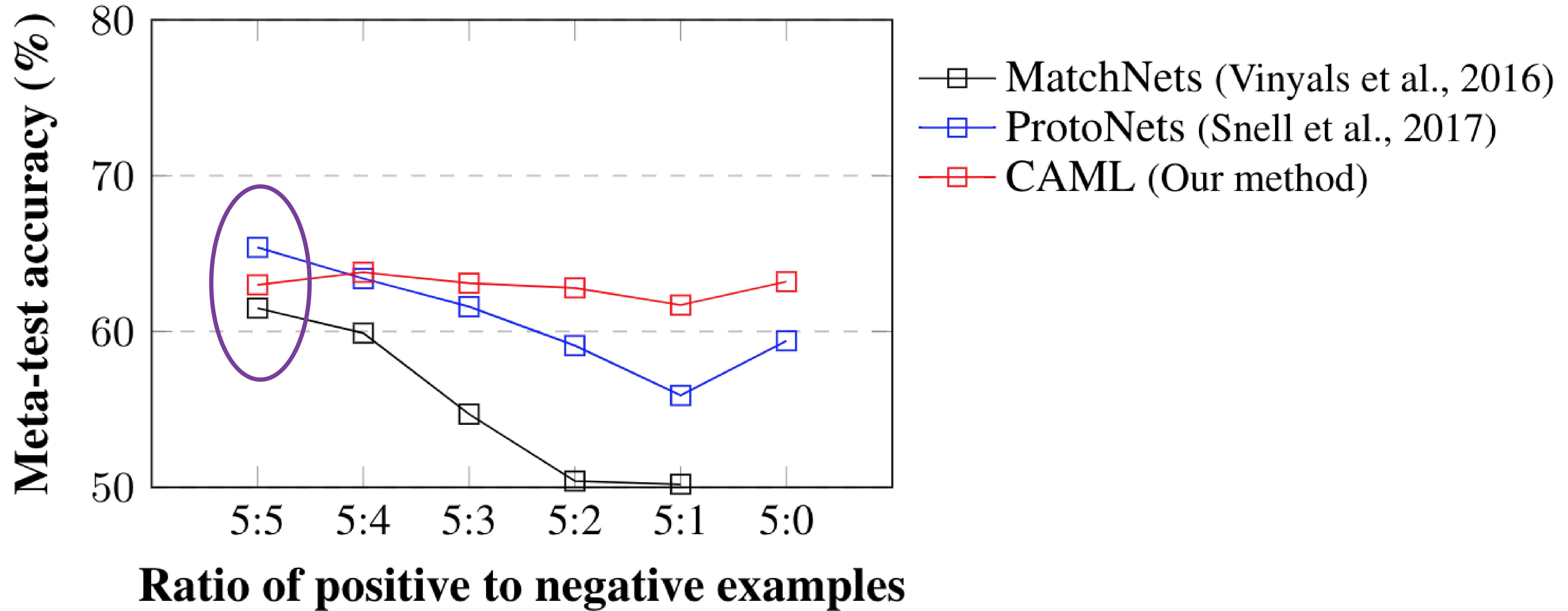
# Few-Shot Image Classification from Positive Examples

## Minilmagenet dataset



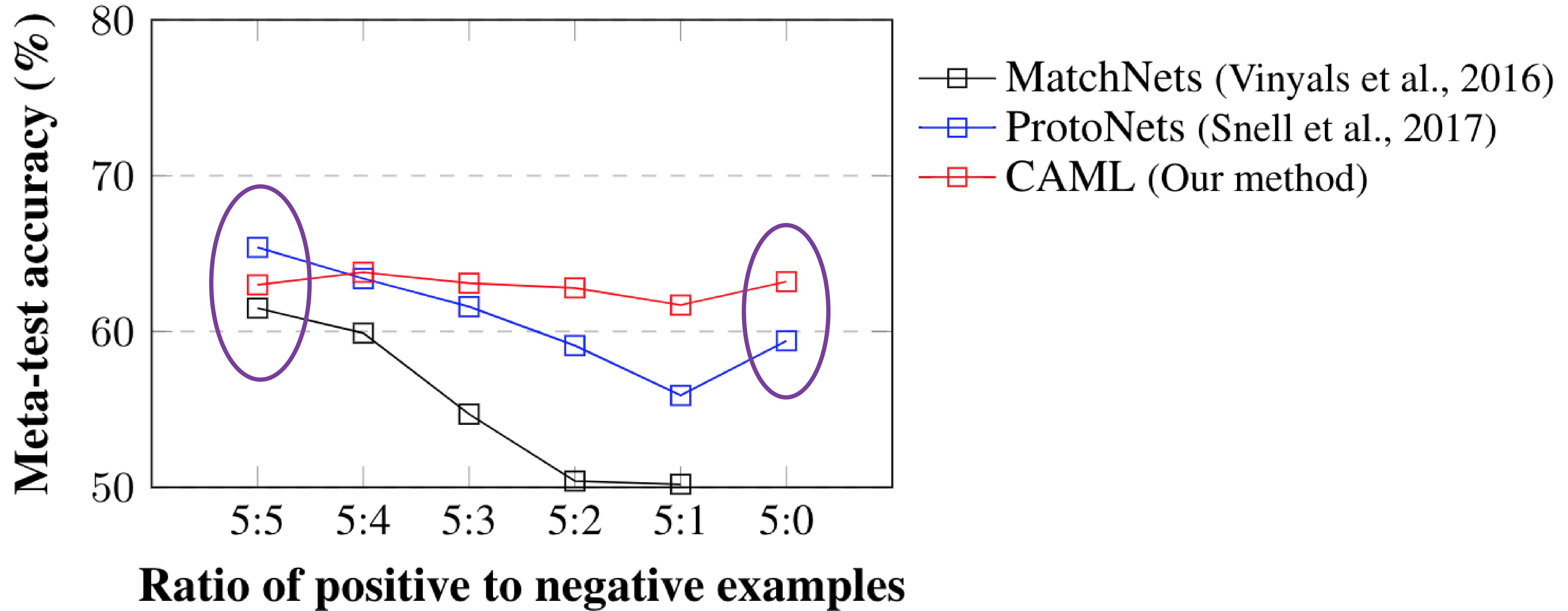
# Few-Shot Image Classification from Positive Examples

## Minilmagenet dataset



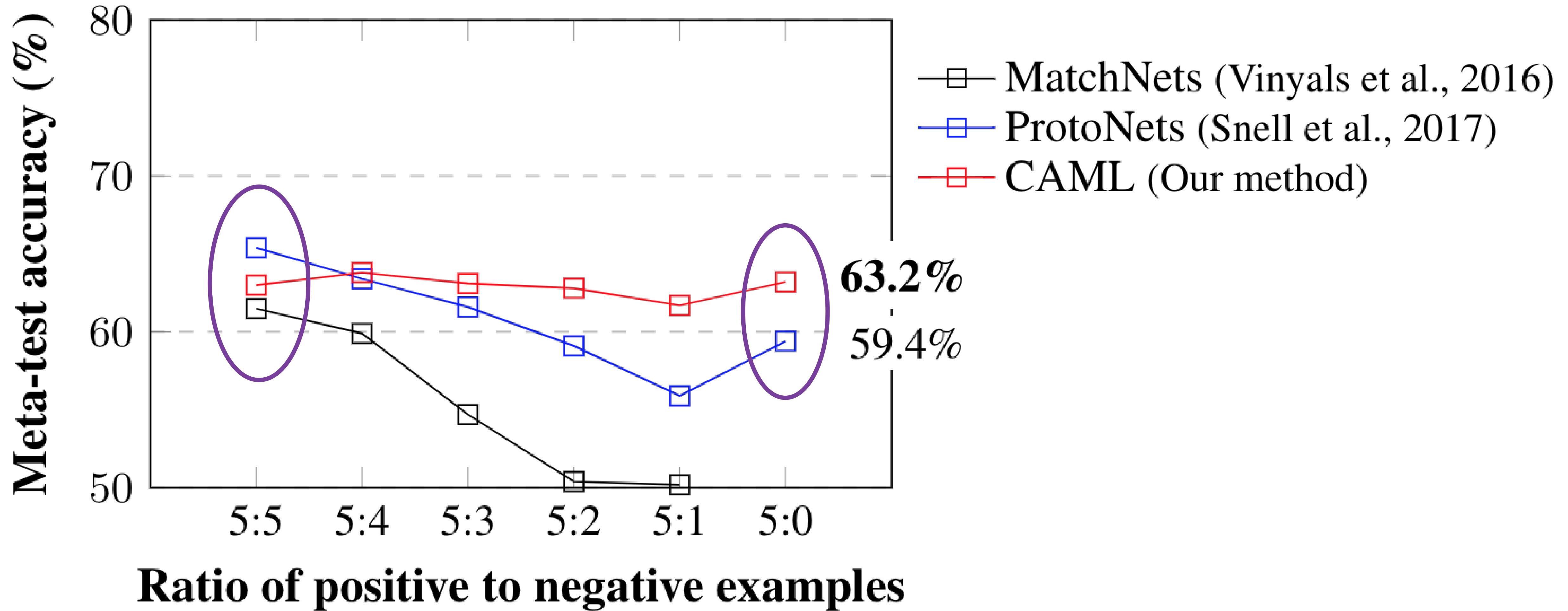
# Few-Shot Image Classification from Positive Examples

## Minilmagenet dataset



# Few-Shot Image Classification from Positive Examples

Minilmagenet dataset



# One-Shot Visual Imitation Learning

# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

Visual imitation is expensive.

# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

Visual imitation is expensive.

behavior cloning / supervised learning



Rahmanizadeh et al. '17'hang et al. '17

learns from raw pixels,

but requires many demonstrations

# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

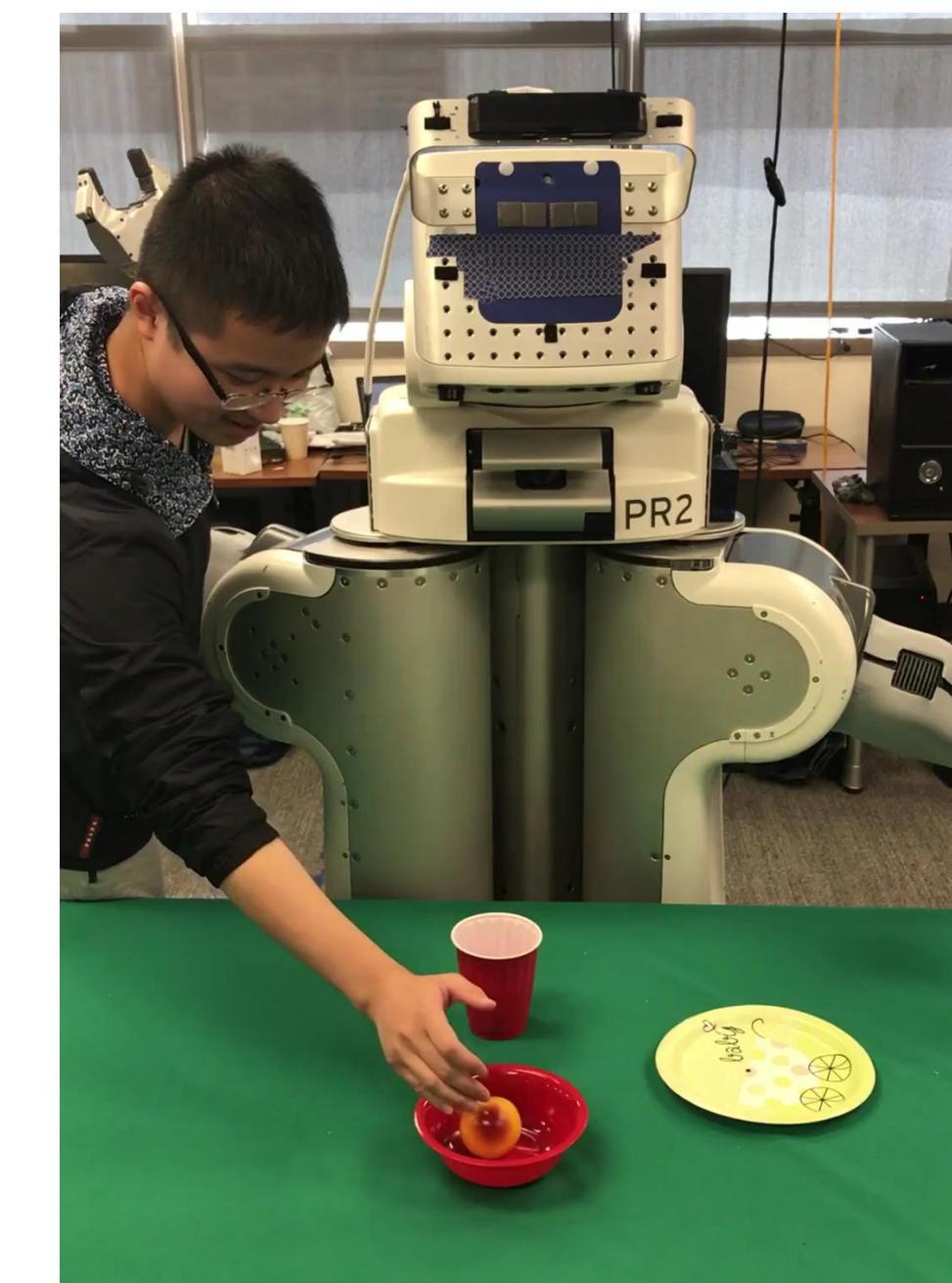
Visual imitation is expensive.

behavior cloning / supervised learning



Rahmanizadeh et al. '17'hang et al. '17  
learns from raw pixels,  
but requires many demonstrations

No direct supervision signal  
in video of human.



# One-Shot Visual Imitation Learning

**Goal:** Given one visual demonstration of a new task, learn a policy

Visual imitation is expensive.

behavior cloning / supervised learning

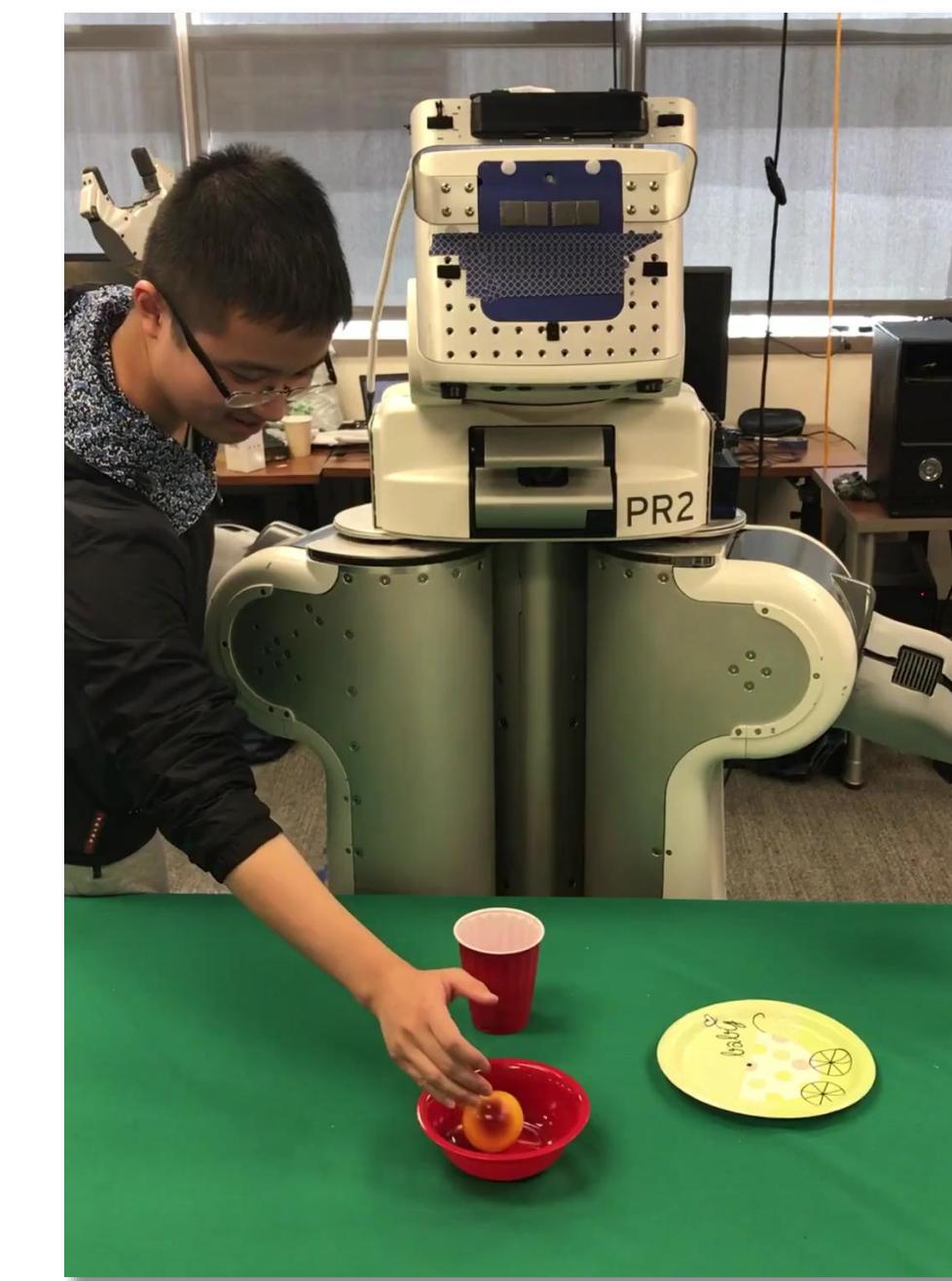


Rahmanizadeh et al. '17'hang et al. '17

learns from raw pixels,

but requires many demonstrations

No direct supervision signal  
in video of human.



Through meta-learning: reuse data from other  
tasks/objects/envionrments

# One-Shot Visual Imitation from Humans

# One-Shot Visual Imitation from Humans

**imitation loss**

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

# One-Shot Visual Imitation from Humans

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

*meta-training time*  $\min_{\theta} \sum_{\text{tasks}} \mathcal{L}_{\text{v}}(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$

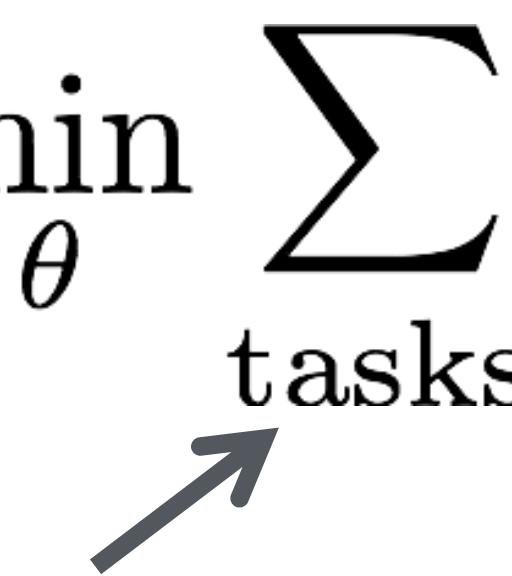
# One-Shot Visual Imitation from Humans

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

*meta-training time*  $\min_{\theta} \sum_{\text{tasks}} \mathcal{L}_v(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$

meta-training  
tasks



# One-Shot Visual Imitation from Humans

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$

*meta-training time*    $\min_{\theta} \sum_{\text{tasks}} \mathcal{L}_v(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{tr}}(\theta))$

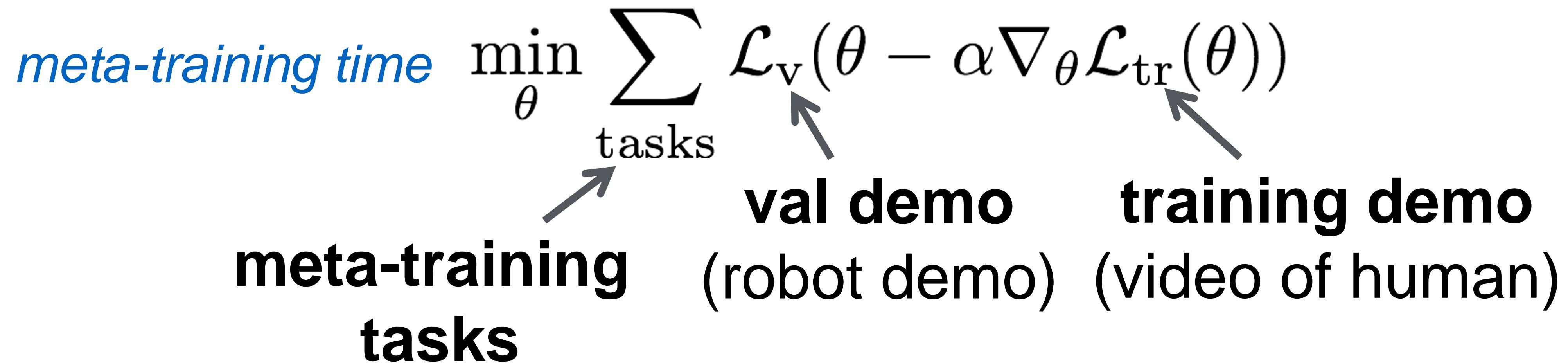
meta-training  
tasks

training demo  
(video of human)

# One-Shot Visual Imitation from Humans

imitation loss

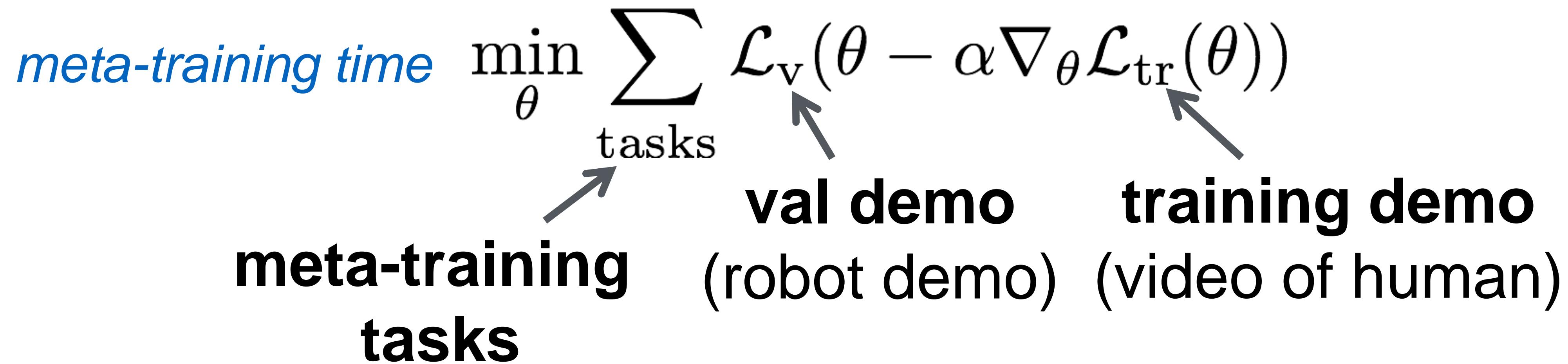
$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$



# One-Shot Visual Imitation from Humans

imitation loss

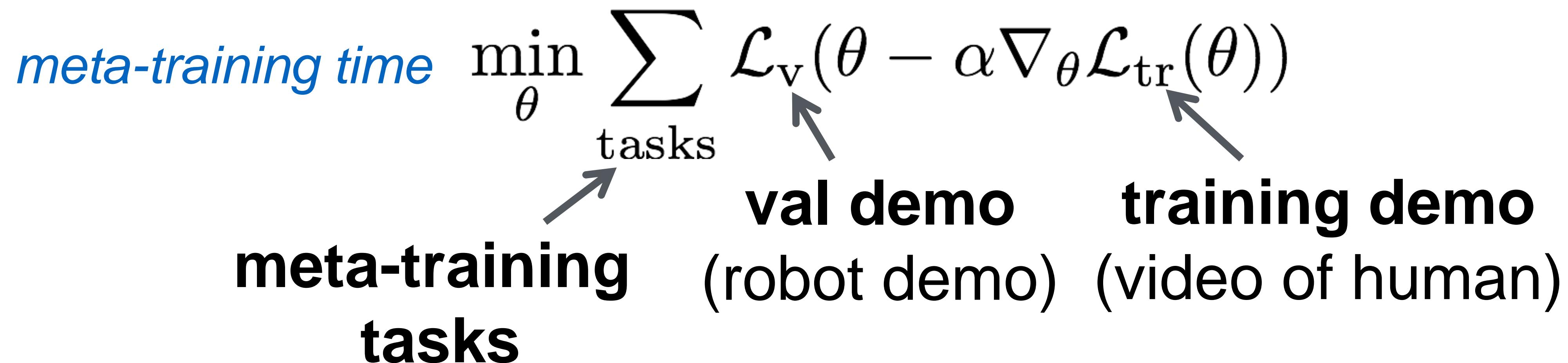
$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$



# One-Shot Visual Imitation from Humans

imitation loss

$$\mathcal{L} = \sum_t \|\pi_\theta(o_t) - a_t^*\|^2$$



*meta-test time*

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

**demo of meta-test task**  
(video of human)