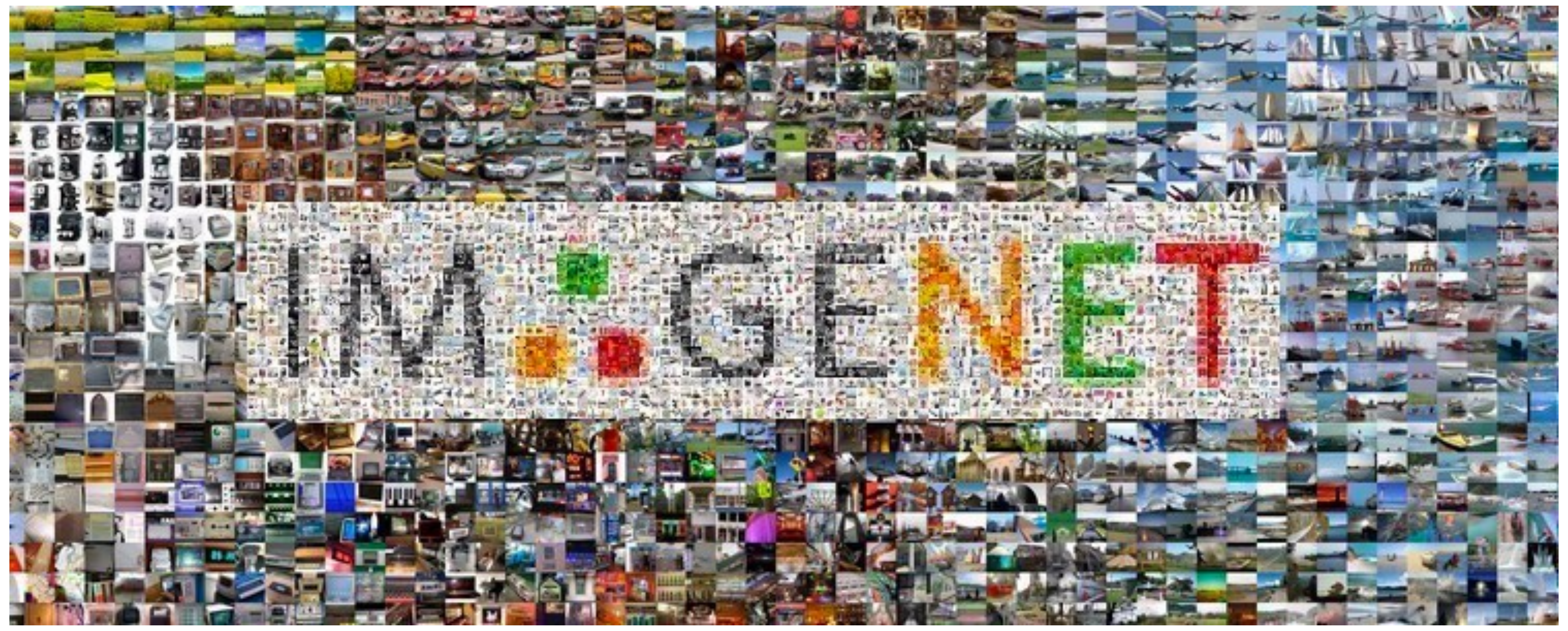
Meta-Learning

Harsh Agrawal

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Large, diverse data
(+ large models) → Broad generalization



Russakovsky et al. '14

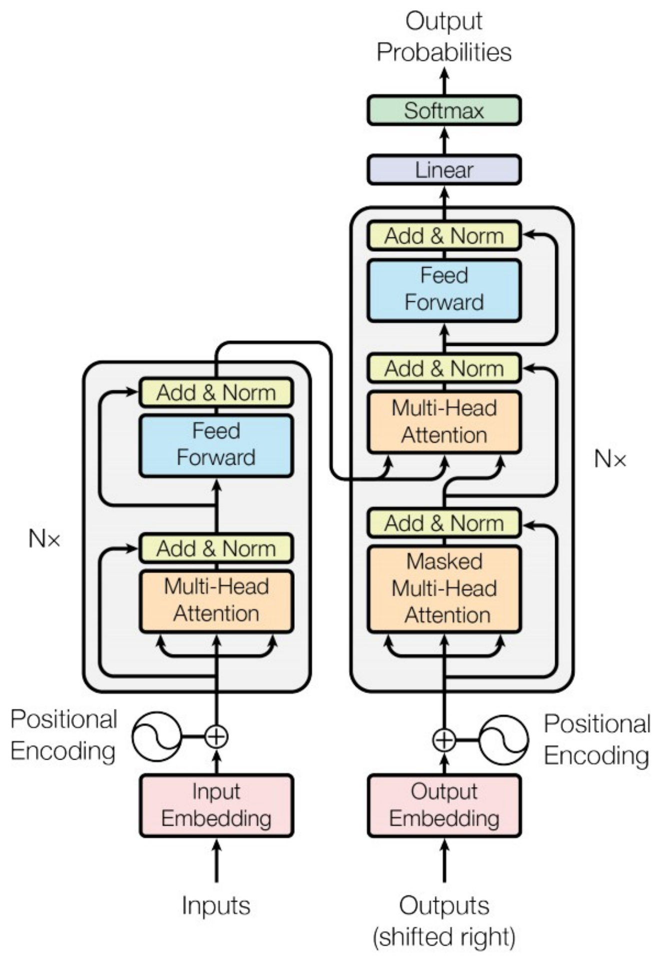


Figure 1: The Transformer - model architecture.

GPT-2

Radford et al. '19

Vaswani et al. '18

Under the paradigm of supervised learning.

Slide Credit: ICML 2019 Meta-Learning Tutorial

What if you don't have a large dataset?

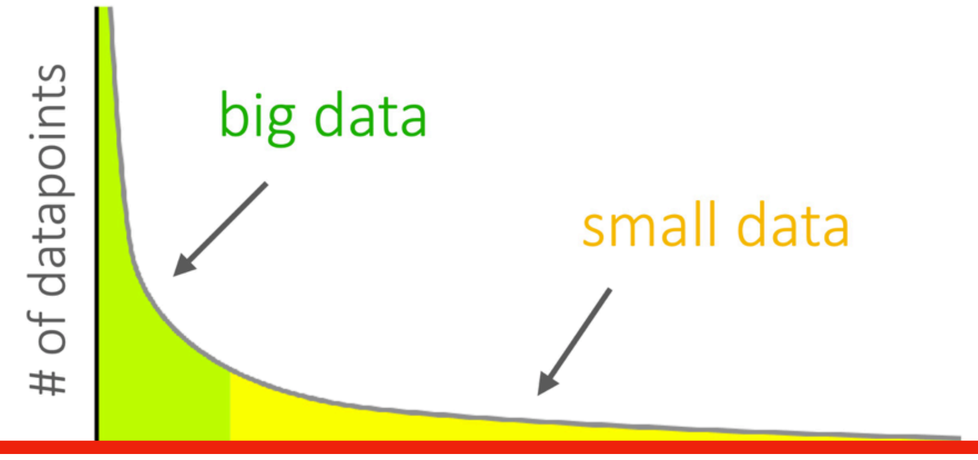
medical imaging   robotics   personalized education

translation for rare languages   recommendations

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

- Need to continuously adapt and learn on the job.
- Learning each thing from scratch won't cut it.

What if your data has a long tail?



Impractical to collect lots of data for each task, and learn specialized networks for each task

Slide Credit: ICML 2019 Meta-Learning Tutorial

Humans are generalists

training data

Braque | **Cezanne**

test datapoint

By Braque or Cezanne?

Slide Credit: ICML 2019 Meta-Learning Tutorial

What if you need to quickly learn something new?

about a new person, for a new task, about a new environment, etc.

“few-shot learning”

How did you accomplish this?

by leveraging prior experience!

Slide Credit: ICML 2019 Meta-Learning Tutorial

Why now?

Why should we study deep multi-task & meta-learning now?

Multitask Learning*

RICH CARUANA

Multitask Learning (MTL) is an inductive transfer mechanism whose principle goal is to improve generalization performance. MTL improves generalization by leveraging the domain-specific information contained in the training signals of *related* tasks. It does this by training tasks in parallel while using a shared representation. In effect, the training signals for the extra tasks serve as an inductive bias. Section 1.2 argues that inductive transfer is important if we wish to scale tabula rasa learning to complex, real-world tasks. Section 1.3 presents the simplest method we know for doing multitask inductive transfer, adding extra tasks (i.e., extra outputs) to a backpropagation net. Because the MTL net uses a shared hidden layer trained in parallel on all the tasks, what is learned for each task can help other tasks be learned better. Section 1.4 argues that it is reasonable to view training signals as an inductive bias when they are used this way.

Caruana, 1997

Is Learning The n-th Thing Any Easier Than Learning The First?

Sebastian Thrun[1]

They are often able to generalize correctly even from a single training example [2, 10]. One of the key aspects of the learning problem faced by humans, which differs from the vast majority of problems studied in the field of neural network learning, is the fact that humans encounter a whole stream of learning problems over their entire lifetime. When faced with a new thing to learn, humans can usually exploit an enormous amount of training data and experiences that stem from other, related learning tasks. For example, when learning to drive a car, years of learning experience with basic motor skills, typical traffic patterns, logical reasoning, language and much more precede and influence this learning task. The transfer of knowledge across learning tasks seems to play an essential role for generalizing accurately, particularly when training data is scarce.

Thrun, 1998

On the Optimization of a Synaptic Learning Rule

Samy Bengio Yoshua Bengio Jocelyn Cloutier Jan Gecsei

Université de Montréal, Département IRO

This paper presents a new approach to neural modeling based on the idea of using an automated method to optimize the parameters of a synaptic learning rule. The synaptic modification rule is considered as a parametric function. This function has *local* inputs and is the same in many neurons. We can use standard optimization methods to select appropriate parameters for a given type of task. We also present a theoretical analysis permitting to study the *generalization* property of such parametric learning rules. By generalization, we mean the possibility for the learning rule to learn to solve *new* tasks. Experiments were performed on three types of problems: a

Bengio et al. 1992

These algorithms are continuing to play a fundamental role in machine learning research.

Multilingual machine translation

Massively Multilingual Neural Machine Translation

Roee Aharoni[*]
Bar Ilan University
Ramat-Gan
Israel
roee.aharoni@gmail.com

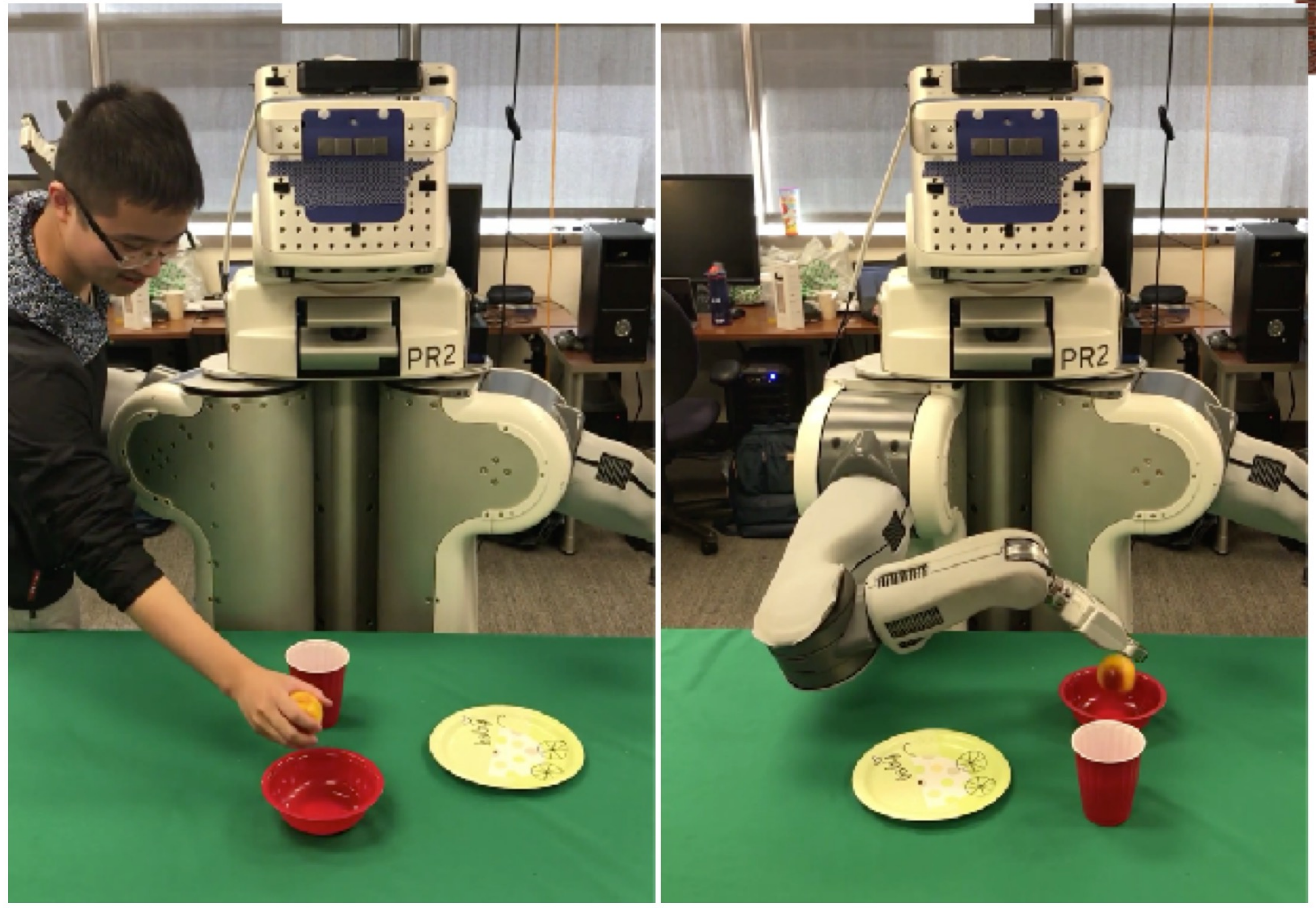
Melvin Johnson and **Orhan Firat**
Google AI
Mountain View
California
melvinp,orhanf@google.com

while supporting up to 59 languages. Our experiments on a large-scale dataset with 102 languages to and from English and up to one million examples per direction also show promising results, surpassing strong bilingual baselines and encouraging future work on massively multilingual NMT.

2019

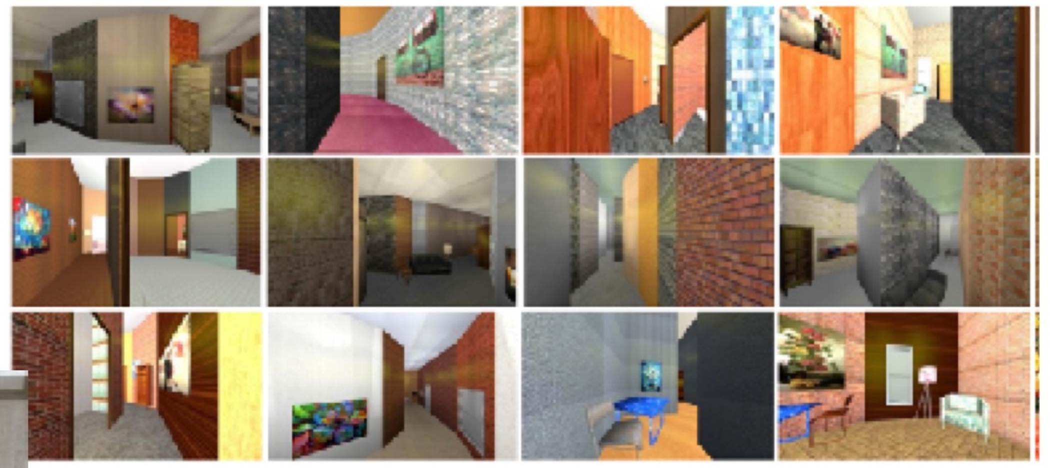
One-shot imitation learning from humans

DAML Yu et al. RSS 2018



Multi-domain learning for sim2real transfer

CAD^{2}RL Sadeghi & Levine, 2016



YouTube recommendations

Recommending What Video to Watch Next: A Multitask Ranking System

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, Ed Chi
Google, Inc.
{zhezhao,lichan,liwei,jilinc,aniruddhnath,shawnandrews,aditeek,nlogn,xinyang,edchi}@google.com

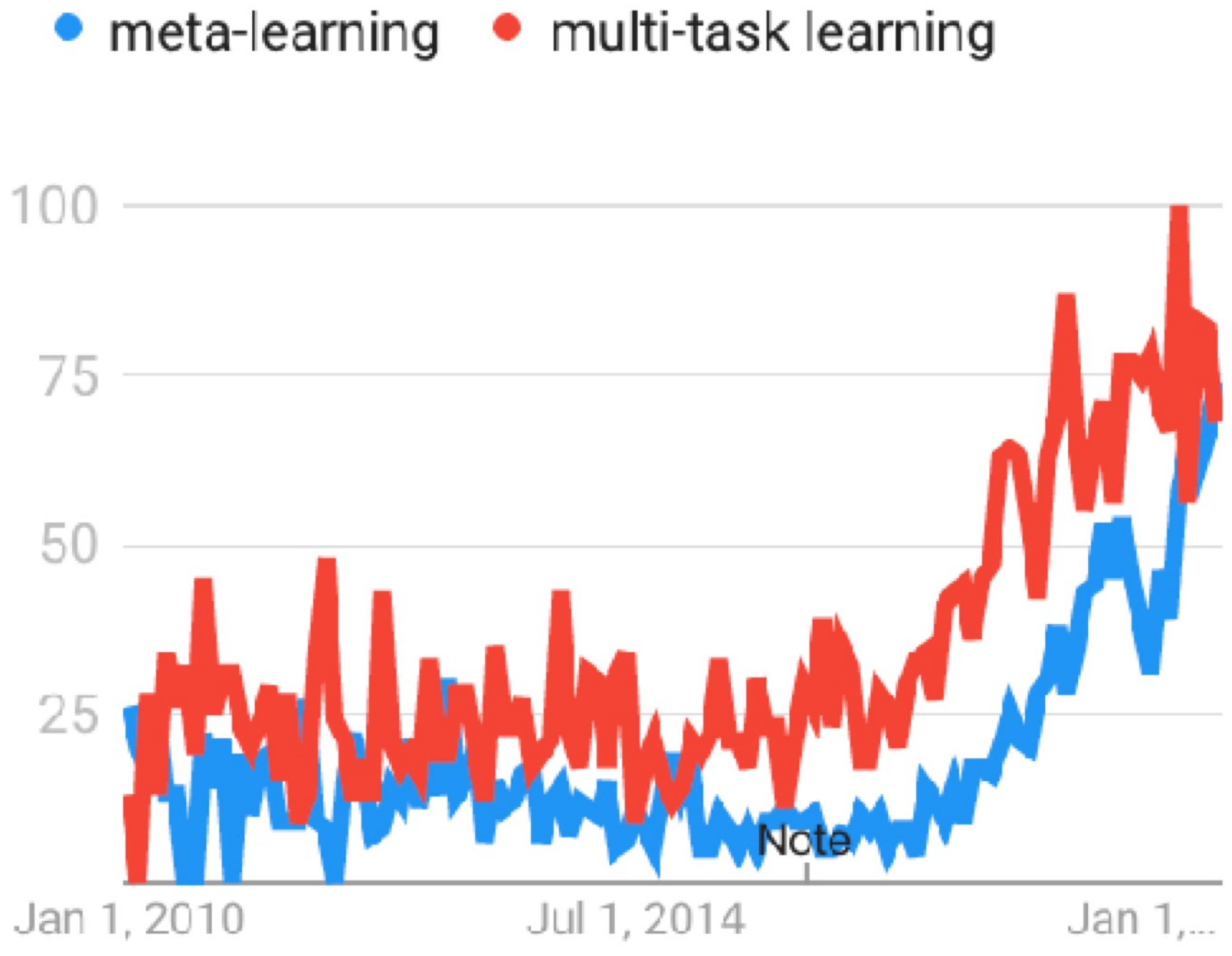
In this paper, we introduce a large scale multi-objective ranking system for recommending what video to watch next on an industrial video sharing platform. The system faces many real-world challenges, including the presence of multiple competing ranking objectives, as well as implicit selection biases in user feedback. To

2019

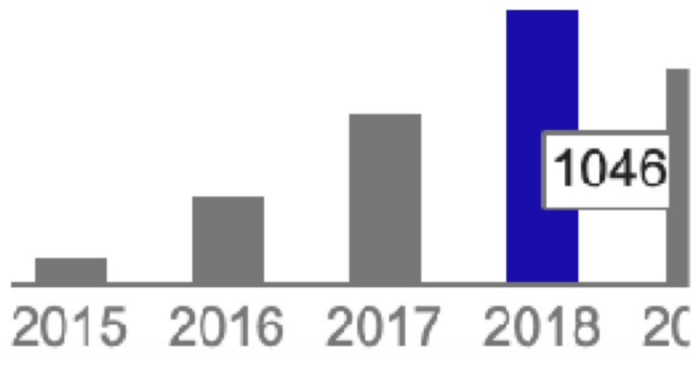
Slide Credit: CS 330: Deep Multi-Task and Meta Learning

These algorithms are playing a fundamental, and **increasing** role in machine learning research.

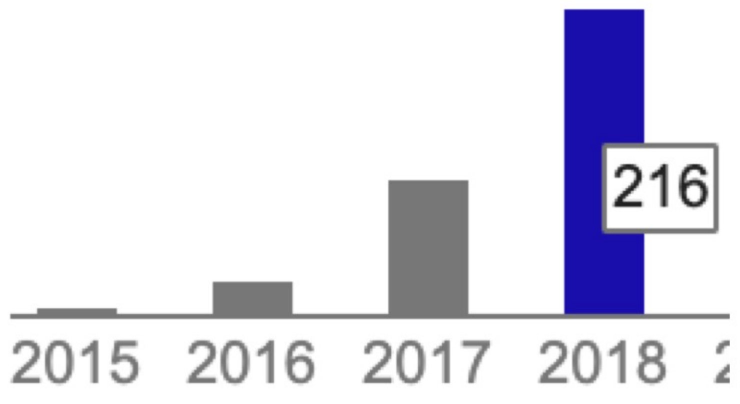
Interest level via search queries



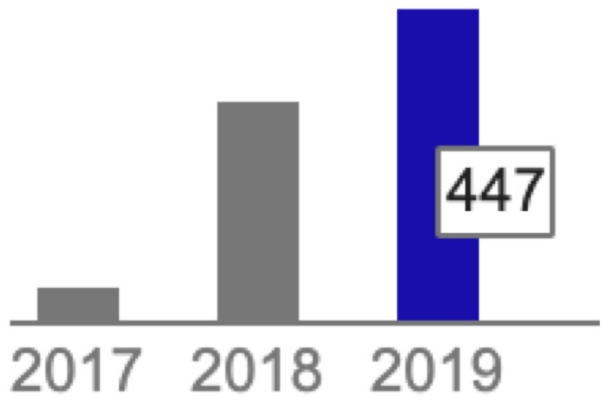
How transferable are features in a deep neural network?
Yosinski et al. '15



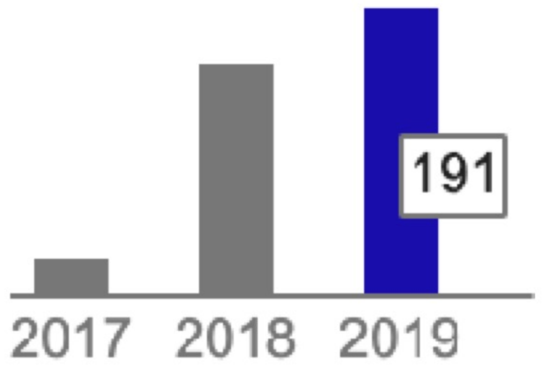
Learning to learn by gradient descent by gradient descent
Andrychowicz et al. '15



Model-agnostic meta-learning for fast adaptation of deep networks
Finn et al. '17



An overview of multi-task learning in neural networks
Ruder '17



Graph sources: Google scholar, Google trends

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

What if you don’t have a large dataset?

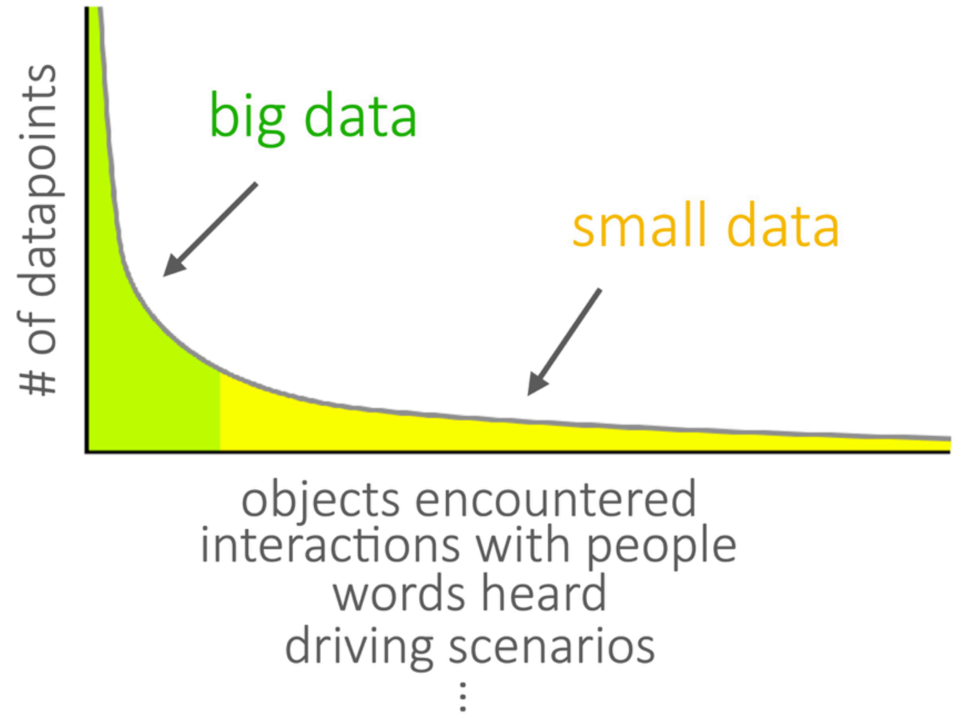
medical imaging robotics personalized education

translation for rare languages recommendations

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

- Need to continuously adapt and learn on the job.
- Learning each thing from scratch won’t cut it.

What if your data has a long tail?



Slide Credit: CS 330: Deep Multi-Task and Meta Learning

What is a task?

What is a task?

For now: dataset $\mathcal{D}$, loss function $\mathcal{L}$ $\longrightarrow$ model f_θ

Different tasks can vary based on:

- different objects
- different people
- different objectives
- different lighting conditions
- different words
- different languages
- ...

Not *just* different "tasks"

Critical Assumption

The bad news: Different tasks need to share some structure.

If this doesn't hold, you are better off using single-task learning.

The good news: There are many tasks with shared structure!

Even if the tasks are seemingly unrelated:

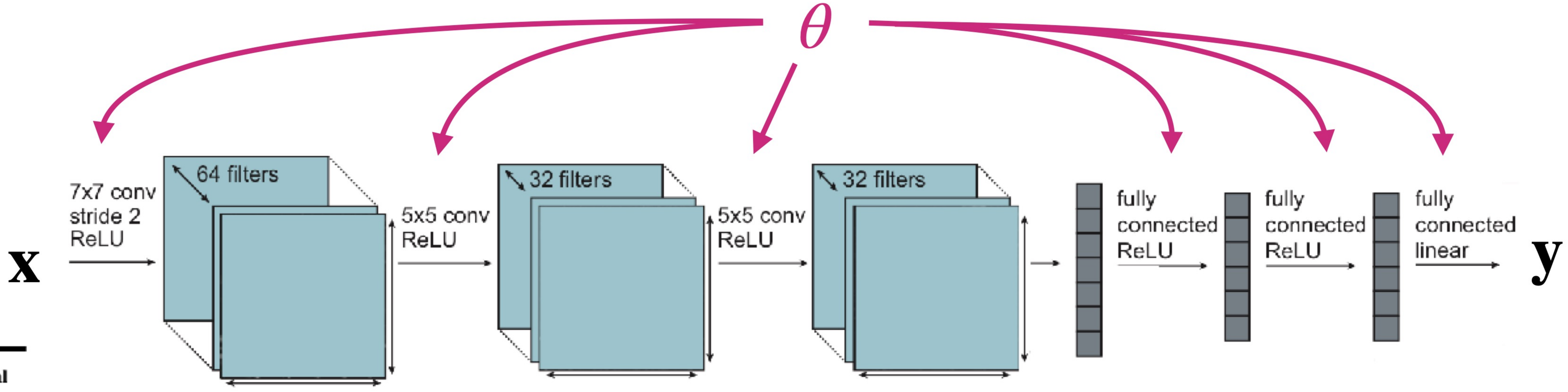
- The **laws of physics** underly real data.
- People are **all organisms with intentions**.
- The **rules of English** underly English language data.
- Languages all develop for **similar purposes**.

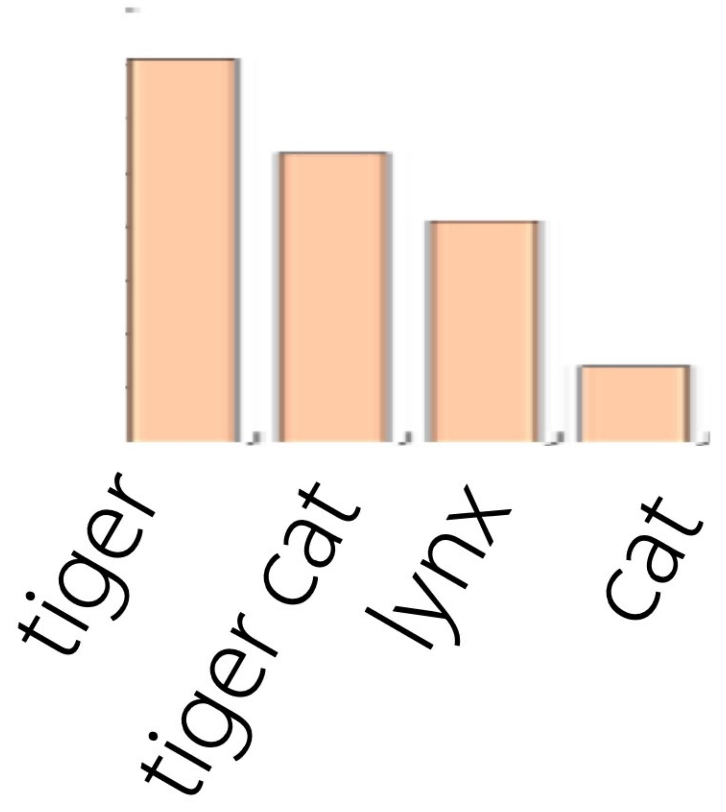
This leads to far greater structure than random tasks.

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Some notation

ImageNet Classification with Deep Convolutional Neural Networks





length of paper

$$f_\theta(\mathbf{y} \mid \mathbf{x})$$

Single-task learning: $\mathscr{D} = \{(\mathbf{x}, \mathbf{y})_k\}$

[supervised] $\min_\theta \mathscr{L}(\theta, \mathscr{D})$

Typical loss: negative log likelihood

$$\mathscr{L}(\theta, \mathscr{D}) = -\mathbb{E}_{(x,y)\sim\mathscr{D}}[\log f_\theta(\mathbf{y} \mid \mathbf{x})]$$

What is a task? (more formally this time)

A task: $\mathscr{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathscr{L}_i\}$

data generating distributions

Corresponding datasets: $\mathscr{D}_i^{tr}$ $\mathscr{D}_i^{test}$

will use $\mathscr{D}_i$ as shorthand for $\mathscr{D}_i^{tr}$:

Informal Problem Definitions

The multi-task learning problem: Learn all of the tasks more quickly or more proficiently than learning them independently.

The meta-learning problem: Given data/experience on previous tasks, learn a new task more quickly and/or more proficiently.

Meta-Learning Basics

General Recipe: How to train/evaluate meta-learning algorithms

Training task 1

Support set

K=2

N=3

Query set

Training task 2 . . .

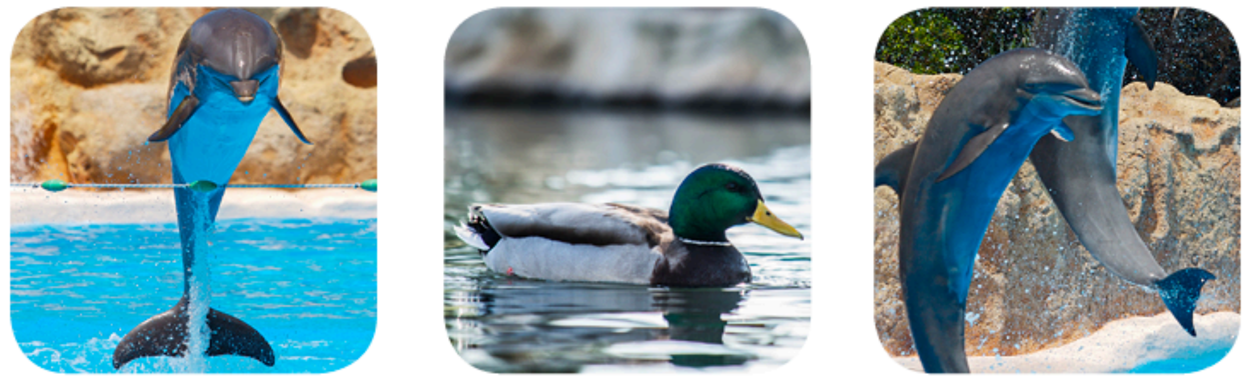
Support set

Query set

Test task 1 . . .

Support set - Held out classes

Query set



General Recipe: How to train/evaluate meta-learning algorithms

the Omniglot dataset Lake et al. Science 2015

1623 characters from **50 different alphabets**

many classes, few examples

the "transpose" of MNIST

statistics more reflective of **the real world**

20 instances of each character

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

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

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

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

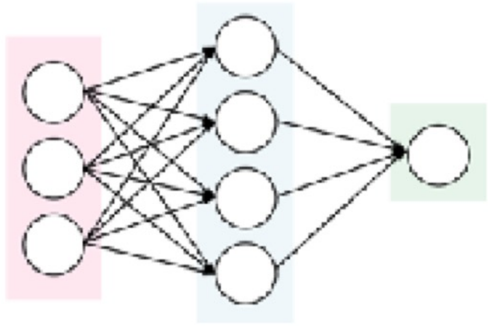
Two ways to view meta-learning algorithms

Mechanistic view

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

Problem definitions

supervised learning:



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

model parameters training data

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

input (e.g., image) label

Slide Credit: ICML 2019 Meta-Learning Tutorial

Problem definitions

supervised learning:

$$\arg\max_{\phi} \log p(\phi|\mathcal{D}) \qquad \mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

The meta-learning problem

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

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

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

meta-learning:

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

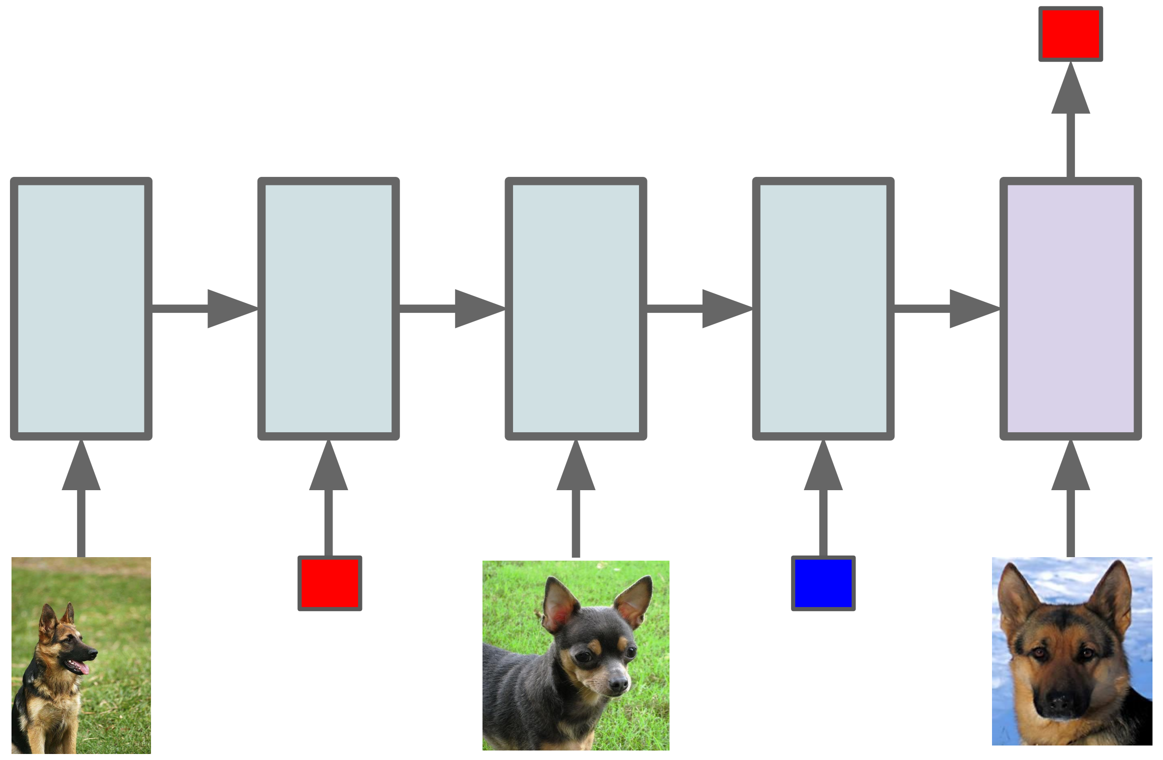
what if we don't want to keep $\mathcal{D}_{\text{meta-train}}$ around forever?

~~this is the meta-learning problem~~

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

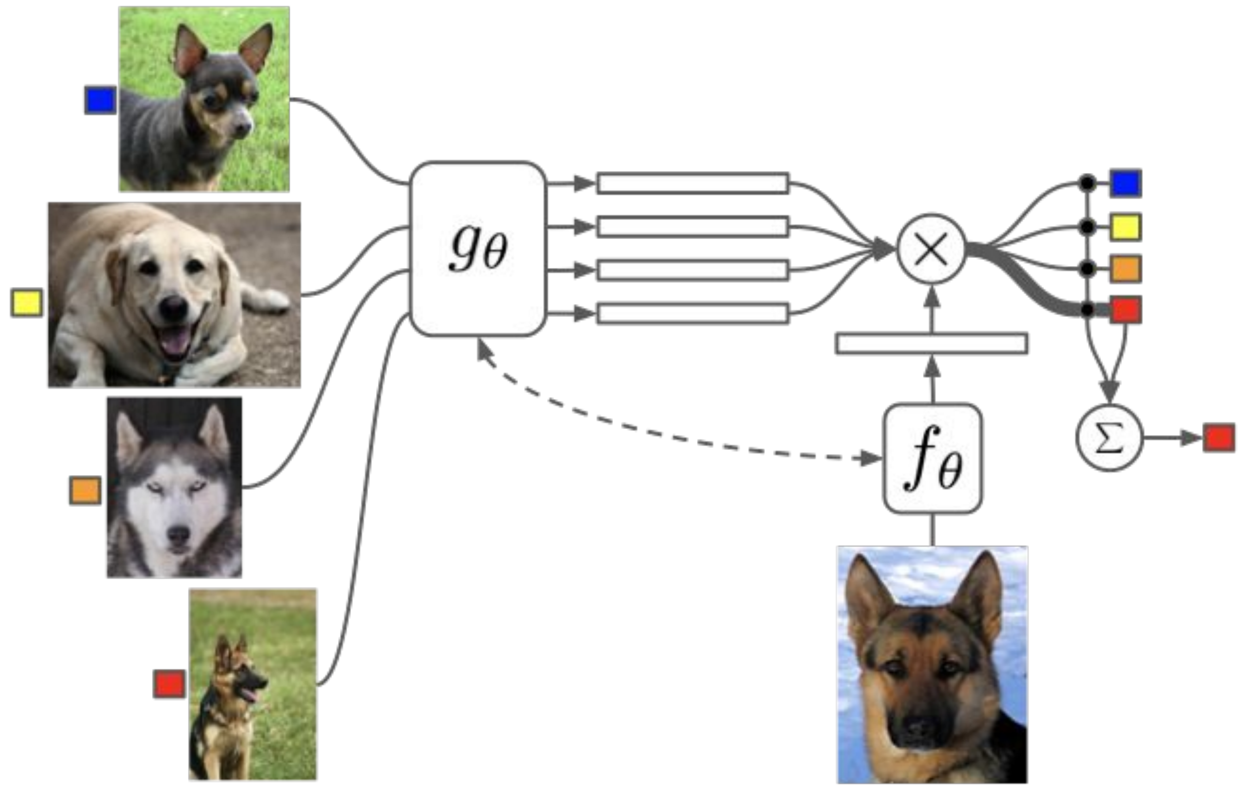
Meta Learning Algorithms Taxonomy

Model Based



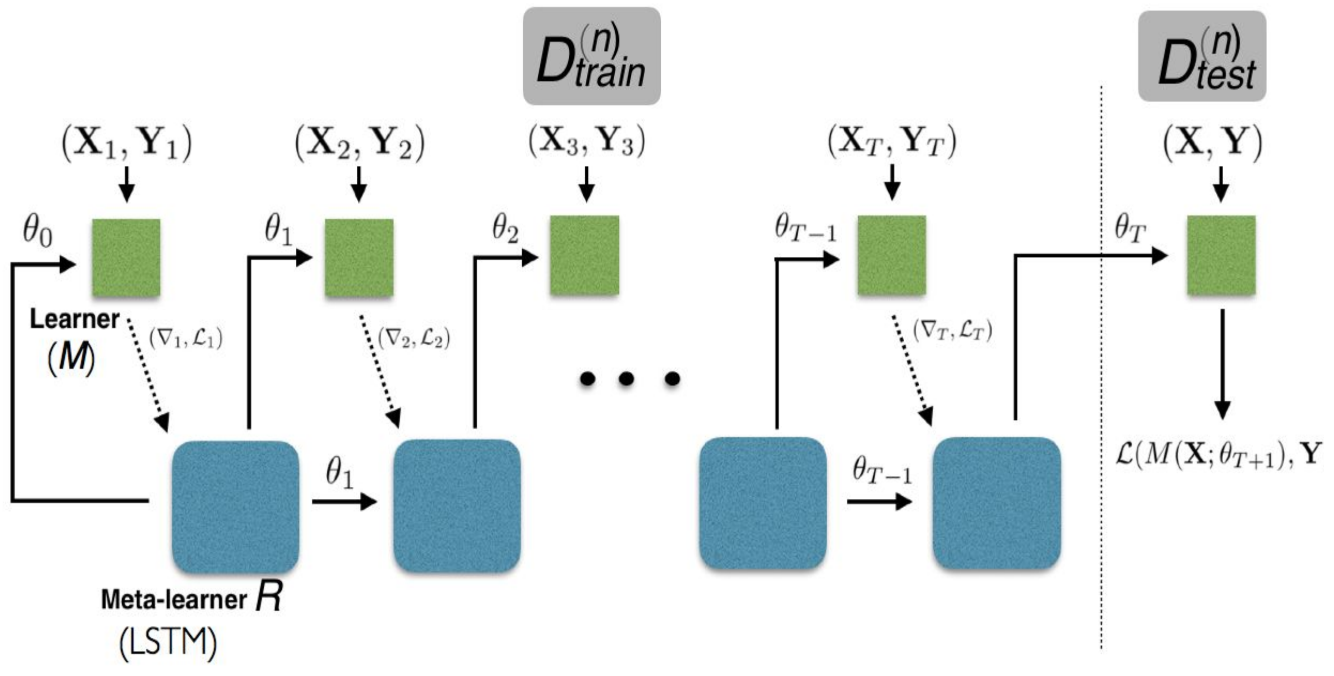
- Santoro et al. ’16
- Duan et al. ’17
- Wang et al. ‘17
- Munkhdalai & Yu ‘17
- Mishra et al. ‘17

Metric Based



- Koch ’15
- Vinyals et al. ‘16
- Snell et al. ‘17
- Shyam et al. ‘17
- Sung et al. ‘17

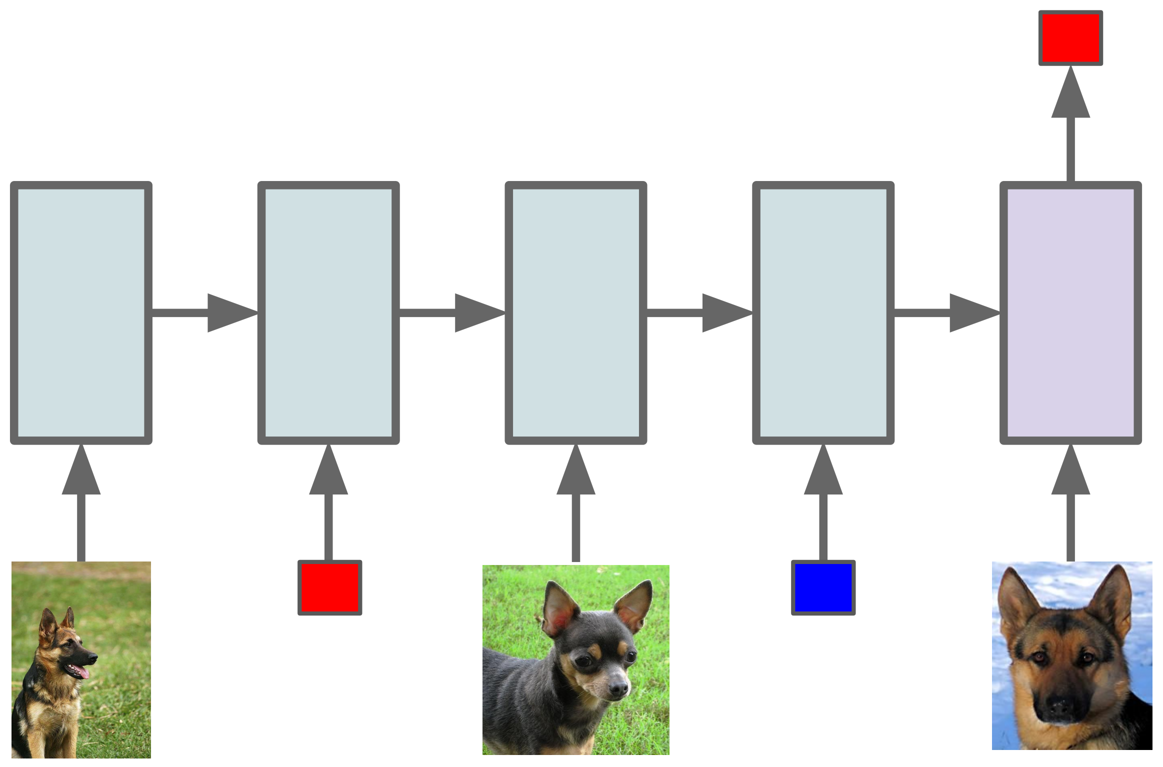
Optimization Based



- Schmidhuber ’87, ’92
- Bengio et al. ’90, ‘92
- Hochreiter et al. ’01
- Li & Malik ‘16
- Andrychowicz et al. ’16
- Ravi & Larochelle ‘17
- Finn et al. ‘17

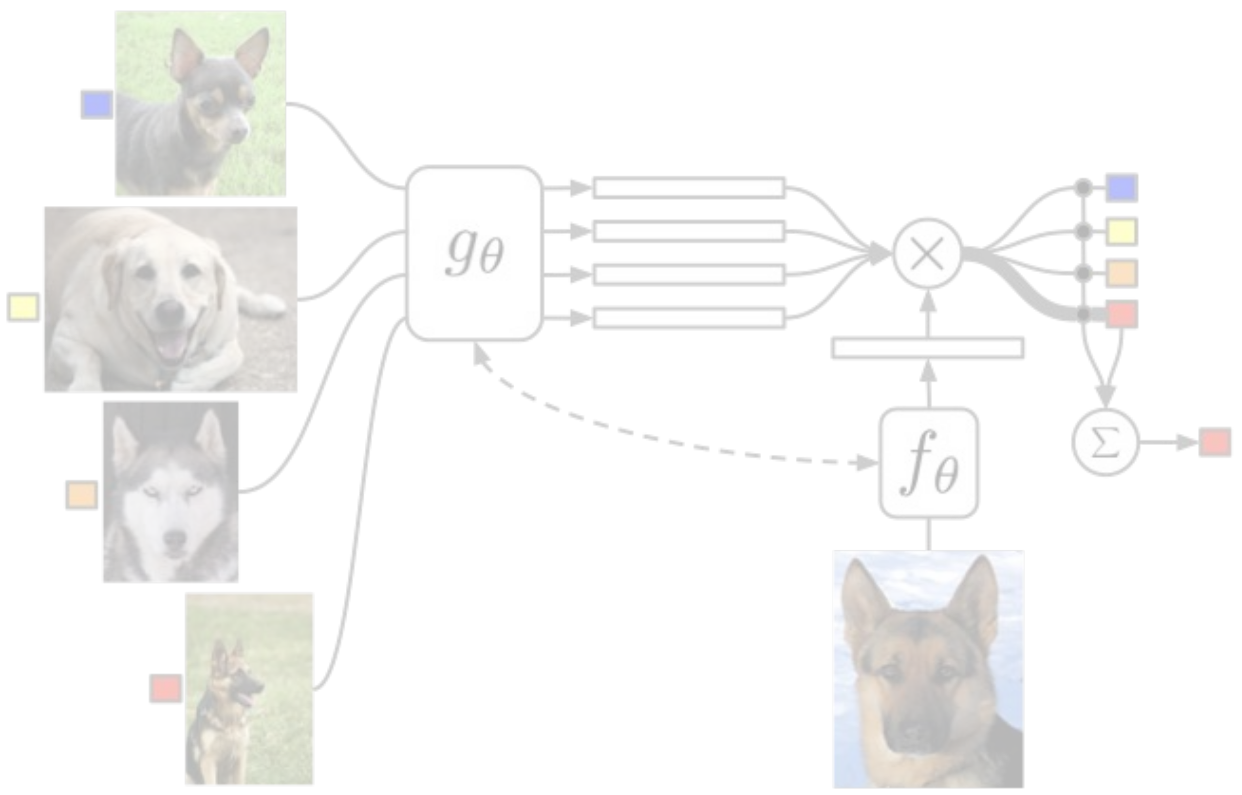
Meta Learning Algorithms Taxonomy

Model Based



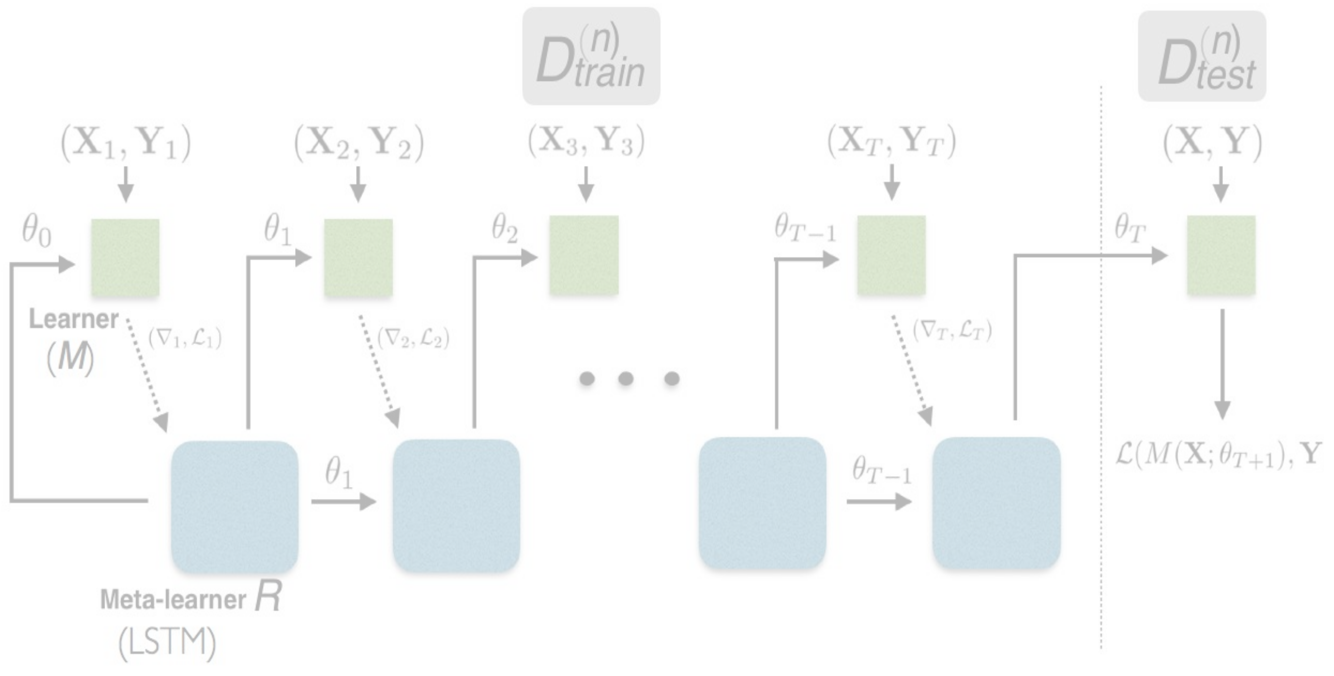
- Santoro et al. ’16
- Duan et al. ’17
- Wang et al. ‘17
- Munkhdalai & Yu ‘17
- Mishra et al. ‘17

Metric Based



- Koch ’15
- Vinyals et al. ‘16
- Snell et al. ‘17
- Shyam et al. ‘17
- Sung et al. ‘17

Optimization Based



- Schmidhuber ’87, ’92
- Bengio et al. ’90, ‘92
- Hochreiter et al. ’01
- Li & Malik ‘16
- Andrychowicz et al. ’16
- Ravi & Larochelle ‘17
- Finn et al. ‘17

A Quick Example

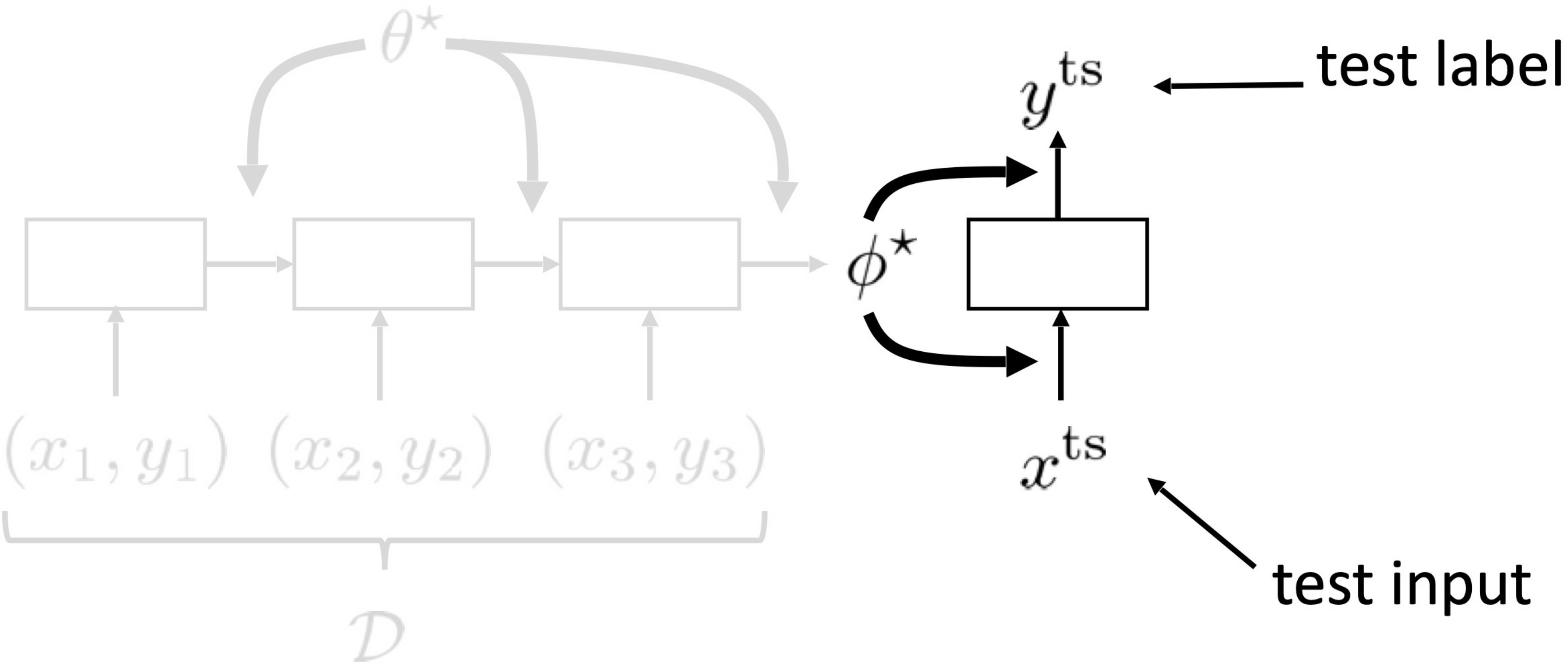
$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

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

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

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

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



How do we train this thing?

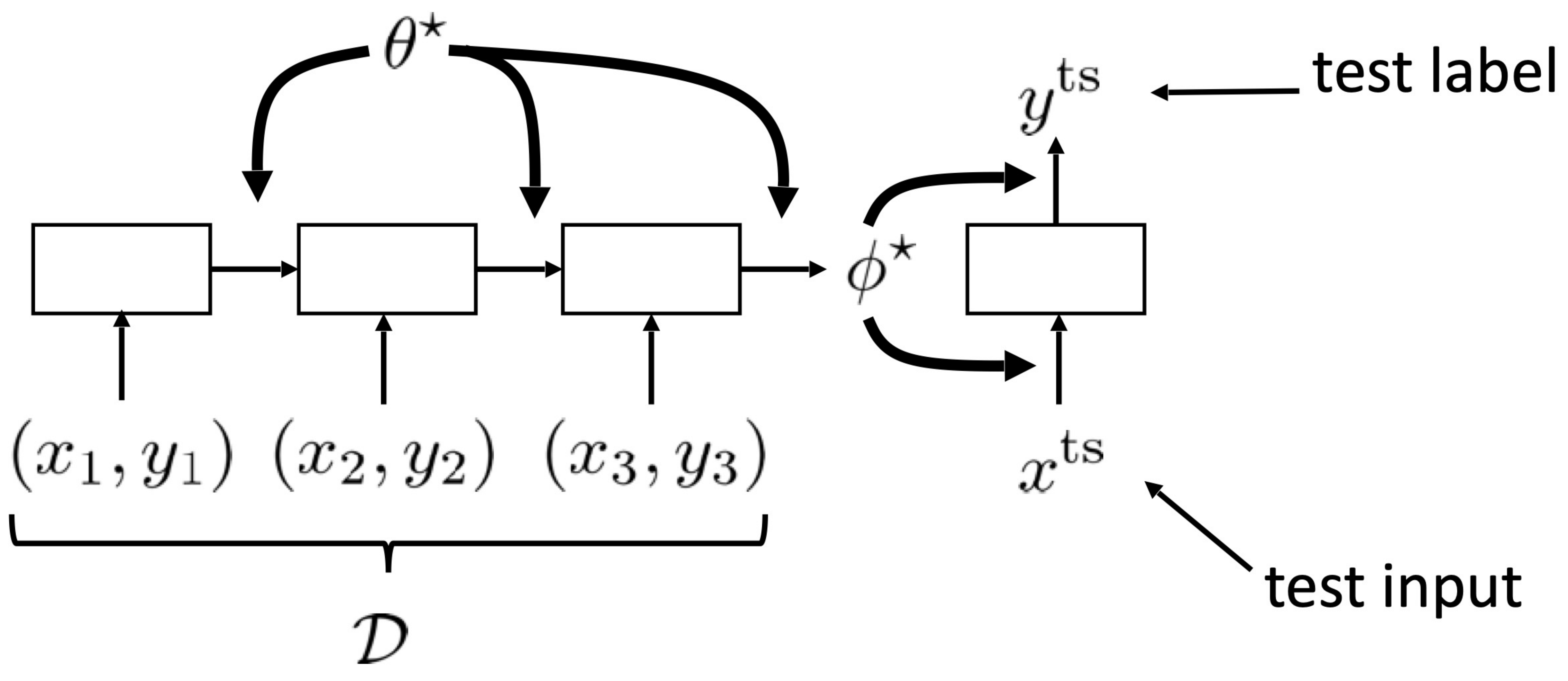
$$\boxed{\text{meta-learning: } \theta^\star = \arg\max_\theta \log p(\theta|\mathcal{D}_{\text{meta-train}})}$$

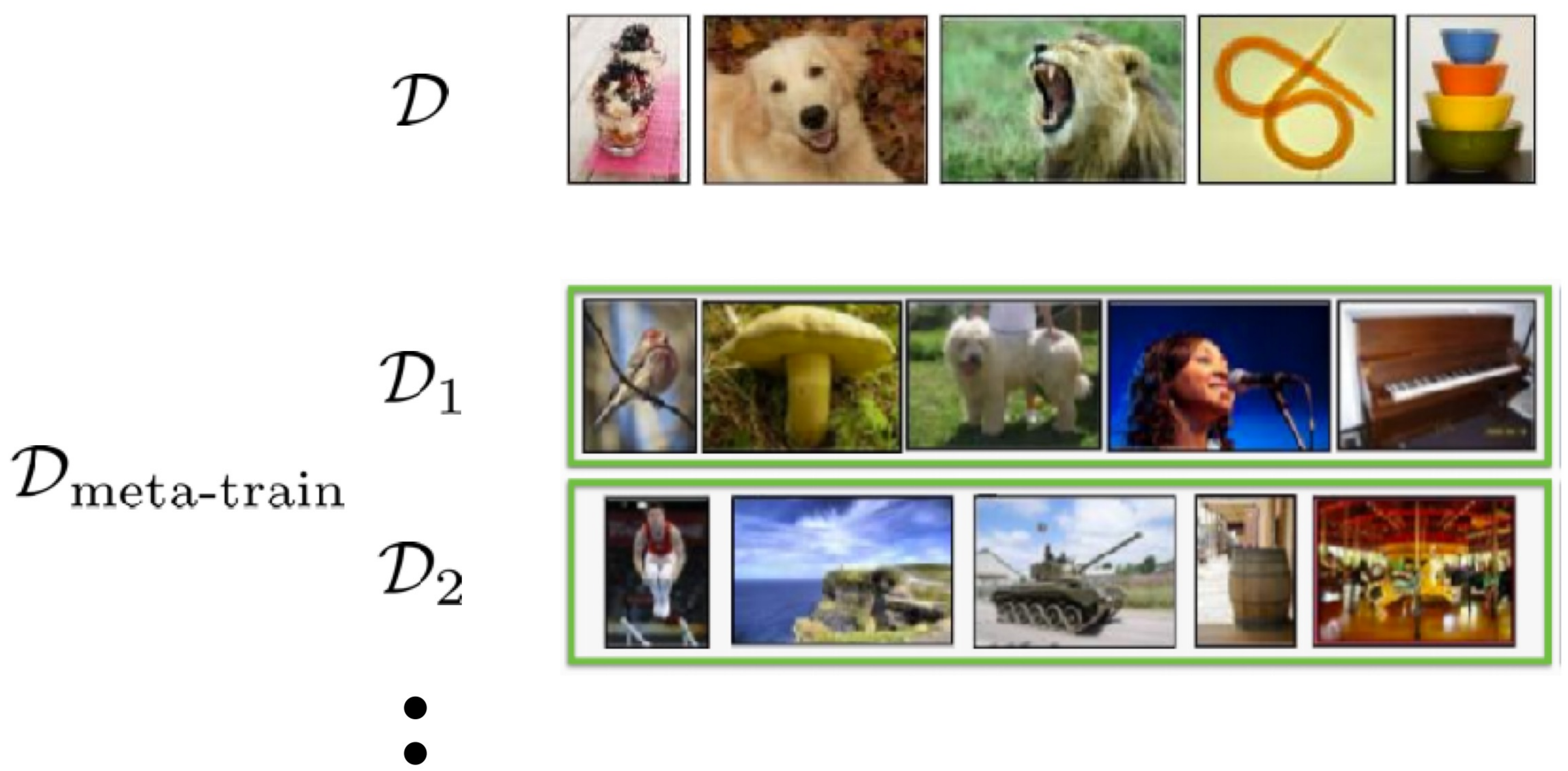
$$\text{adaptation: } \phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}, \theta^\star)$$

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

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

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



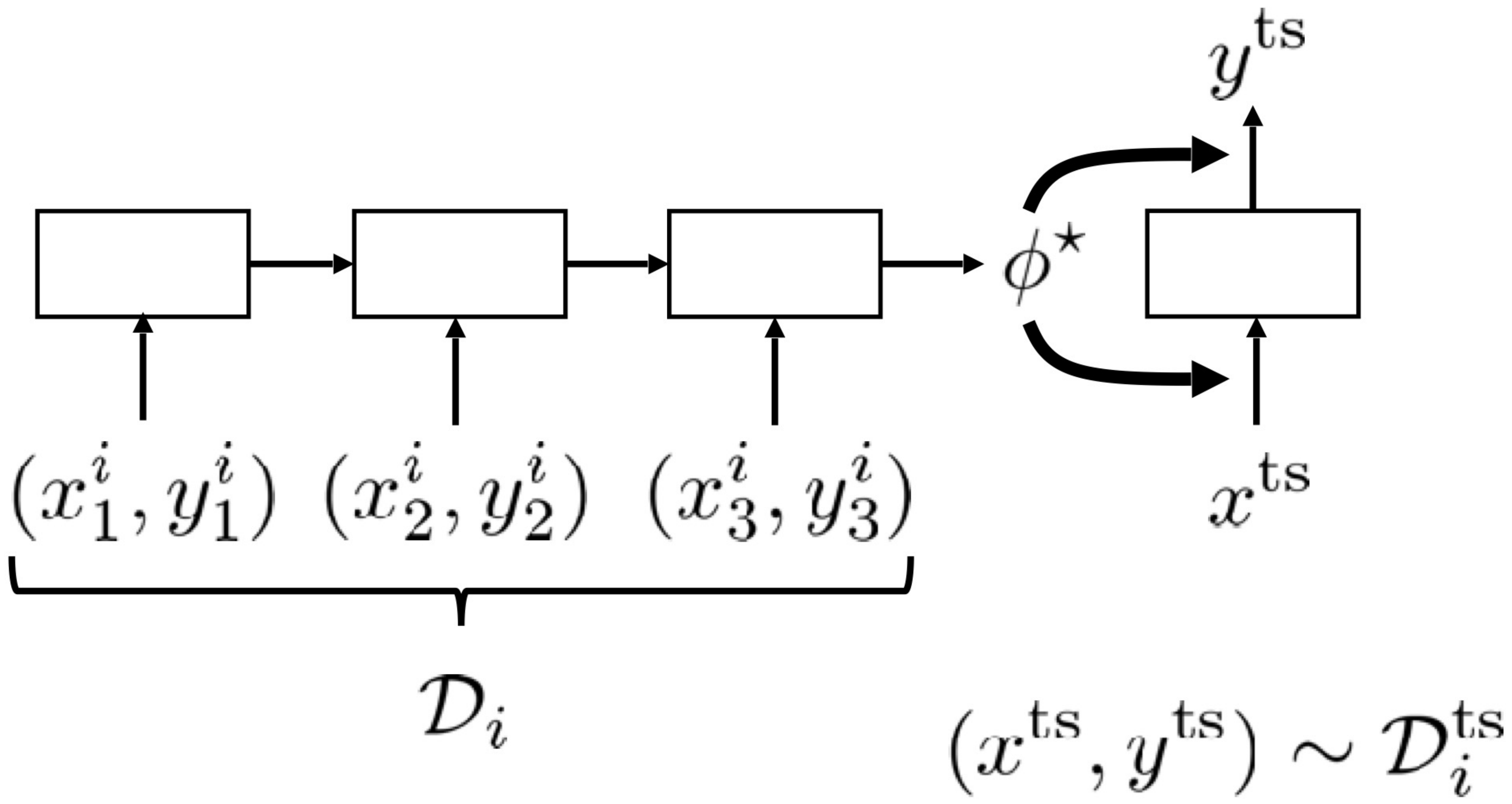


Key idea:
"our training procedure is based on a simple machine learning principle: test and train conditions must match"
Vinyals et al., Matching Networks for One-Shot Learning

Reserve a test set for each task!

$\mathcal{D}_{\text{meta-train}}$ $\mathcal{D}_1$ $\mathcal{D}_2$ $\vdots$

(meta) training-time



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

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

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

Key idea:
"our training procedure is based on a simple machine learning principle: test and train conditions must match"
Vinyals et al., Matching Networks for One-Shot Learning

Slide Credit: ICML 2019 Meta-Learning Tutorial

How to perform meta-training - data loader

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

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

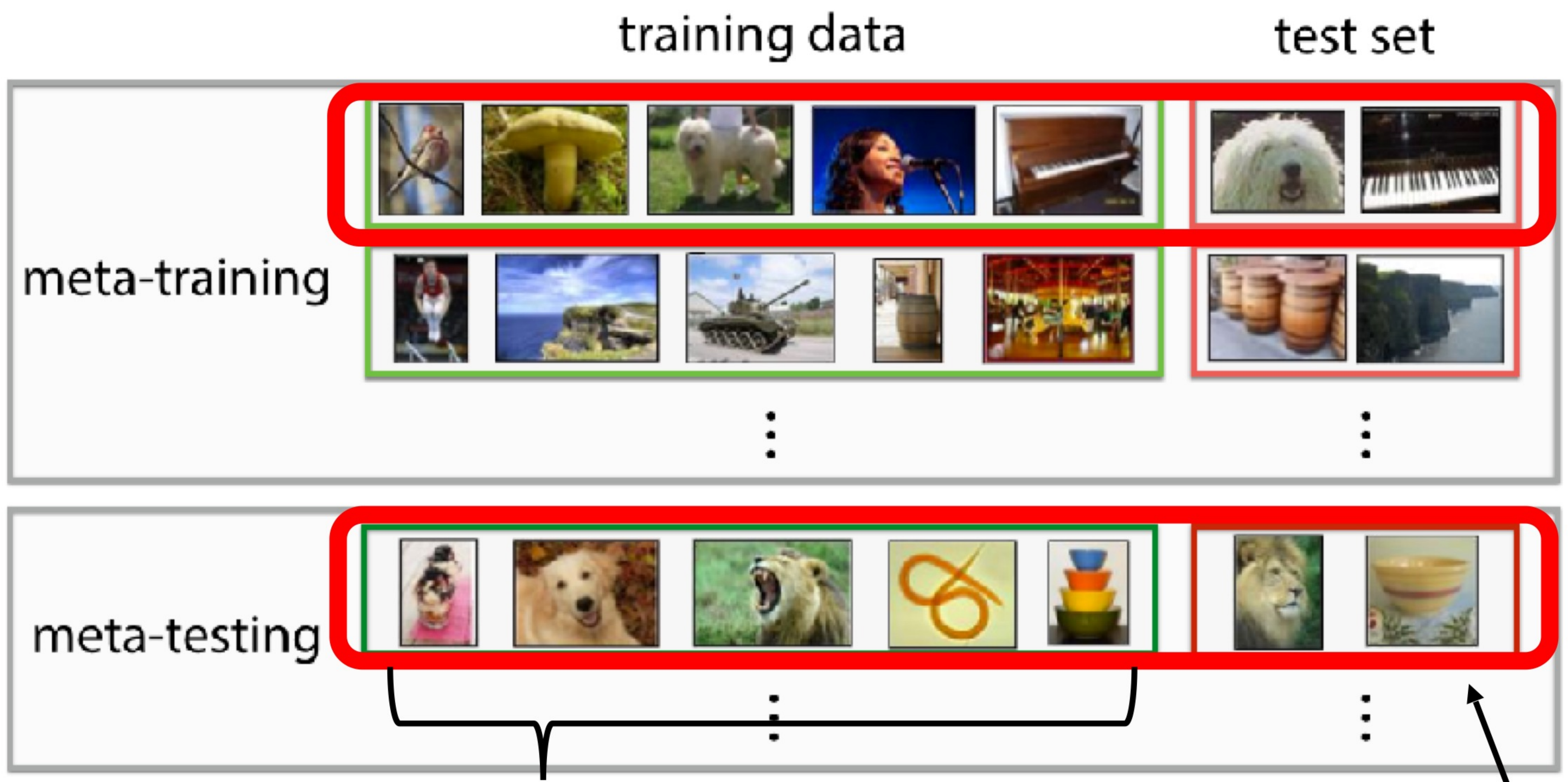
where $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$

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

$$\mathcal{T}_i \begin{cases} \mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\} \\ \mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \ldots, (x_l^i, y_l^i)\} \end{cases}$$

shot
(i.e., k-shot, 5-shot)

(meta-training) task $\mathcal{T}_i$



$\mathcal{D}_1$

$\mathcal{D}_2$

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

(meta-test) task

support (set)

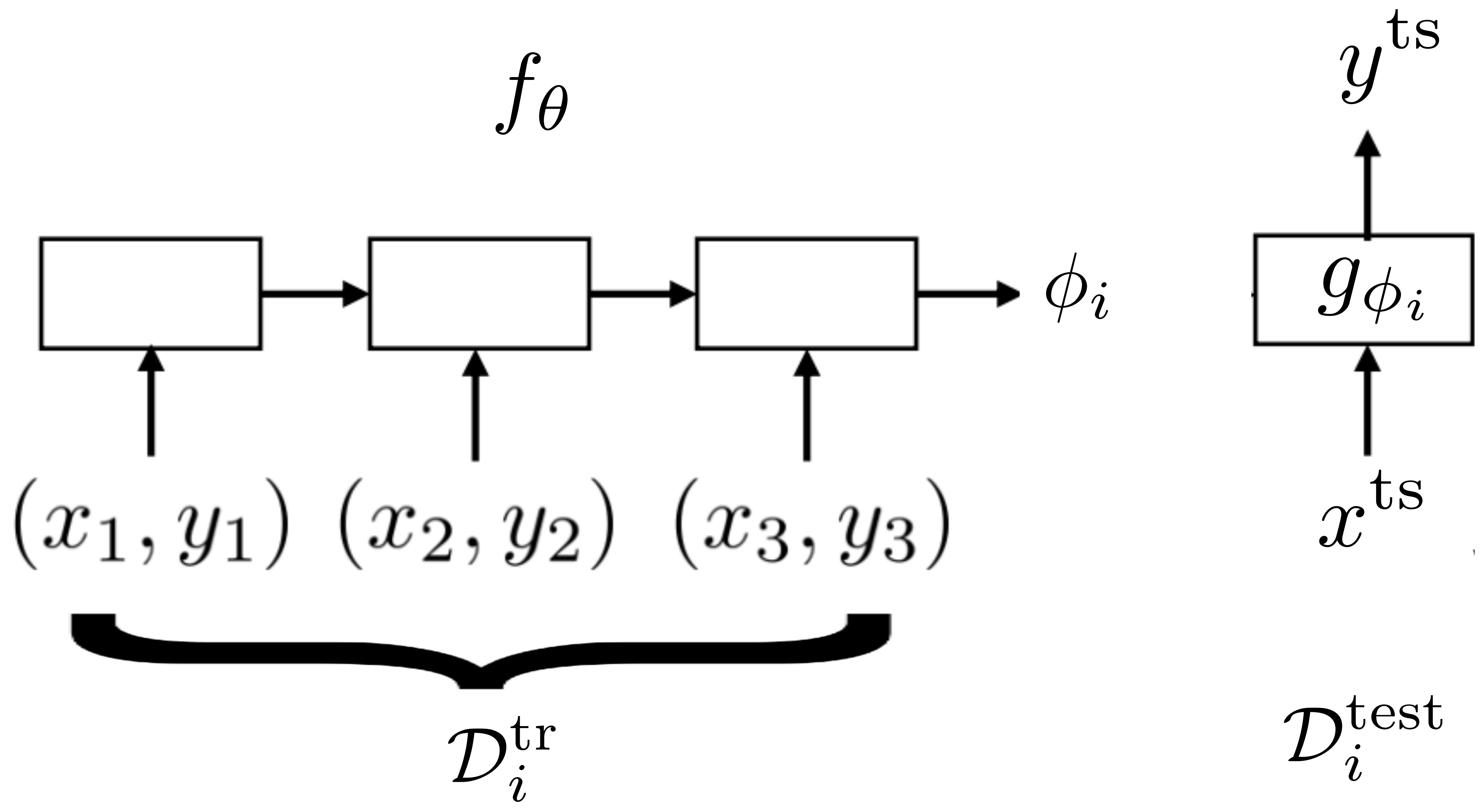
query

image credit: Ravi & Larochelle '17

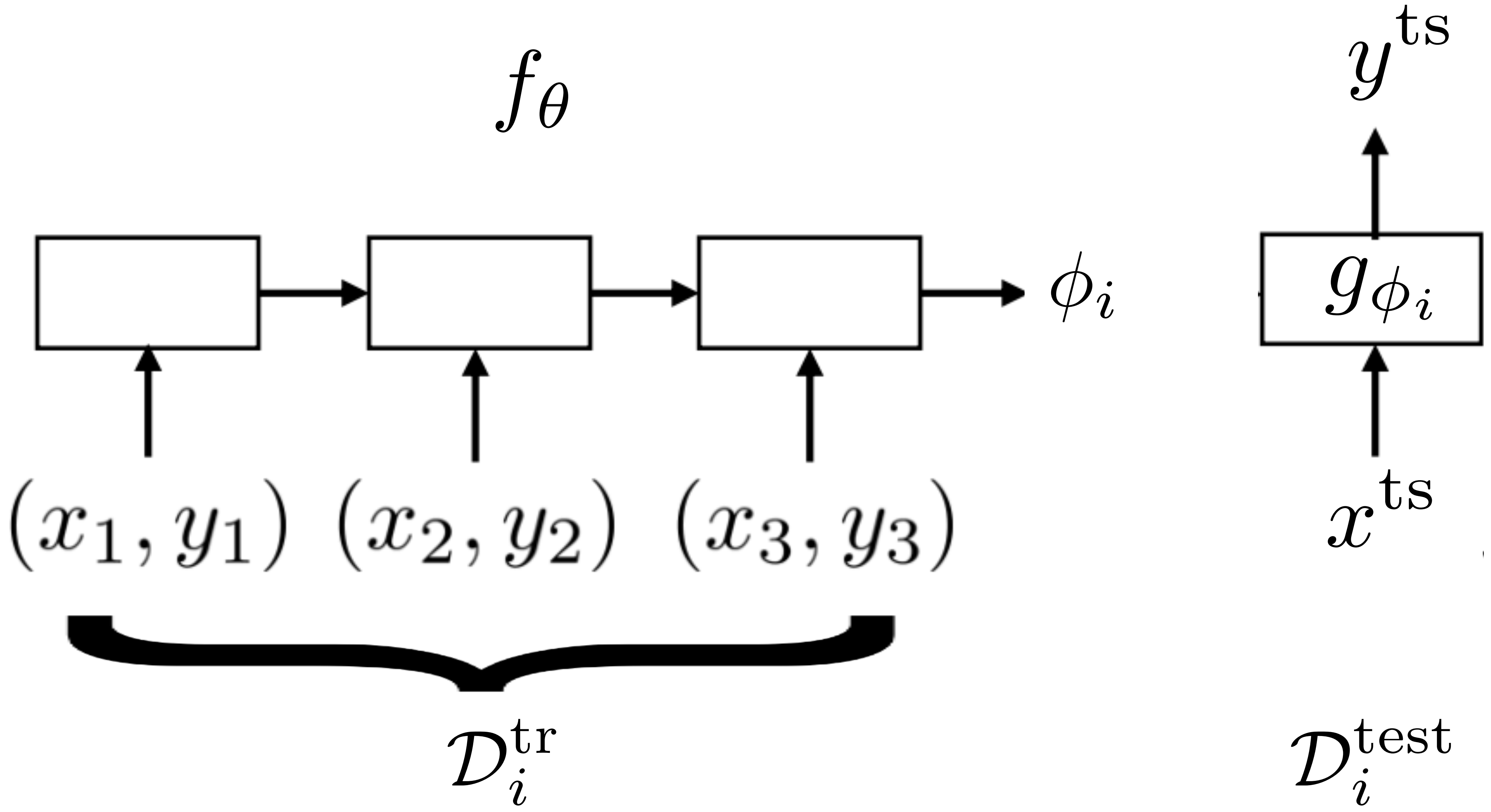
Slide Credit: ICML 2019 Meta-Learning Tutorial

How to perform meta-training

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

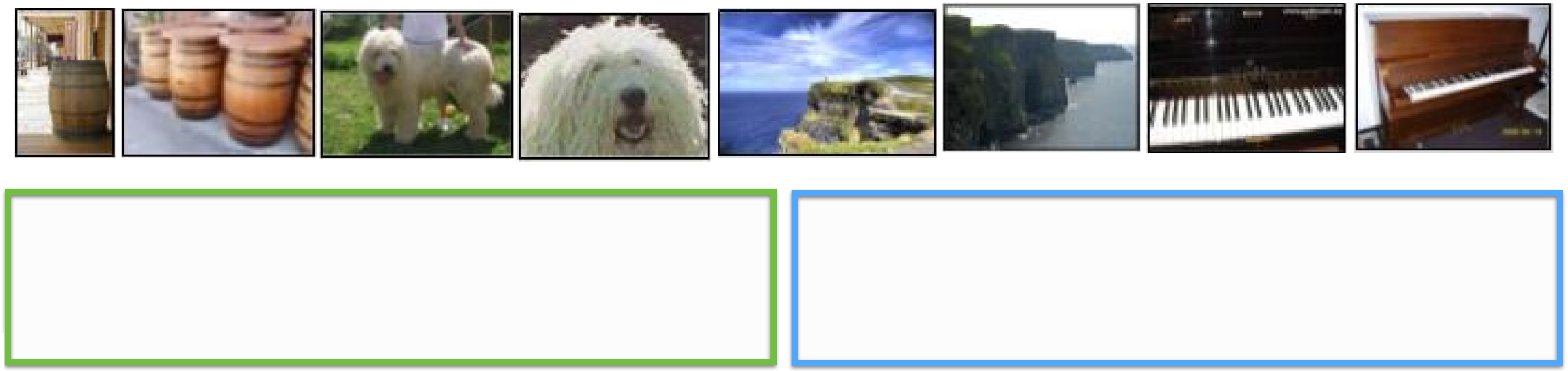


Key idea: Train a neural network to represent $p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$



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

$\mathcal{D}_i$

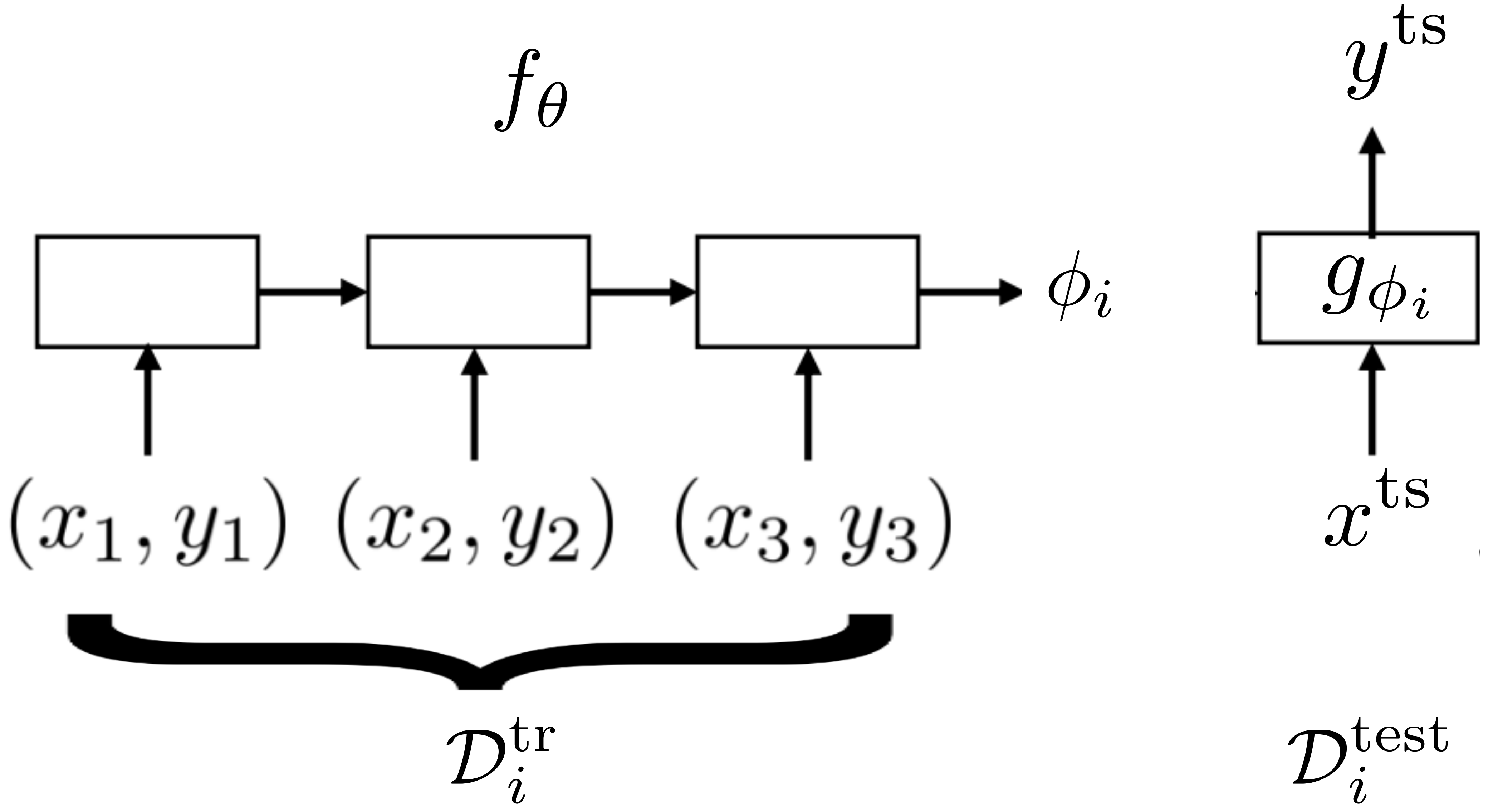


$\mathcal{D}_i^{\mathrm{test}}$

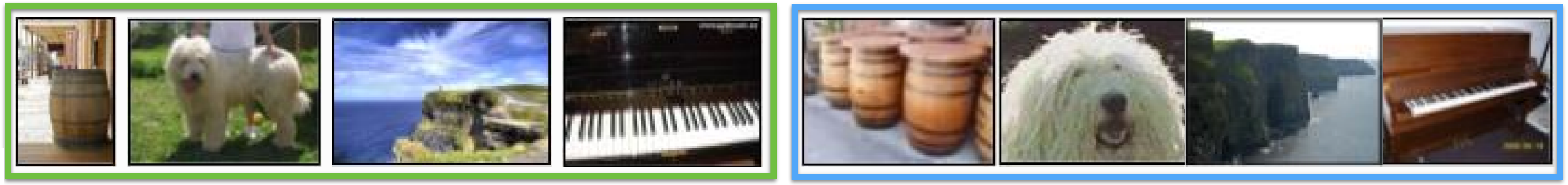
Slide Credit: ICML 2019 Meta-Learning Tutorial

Black-Box Adaptation

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



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

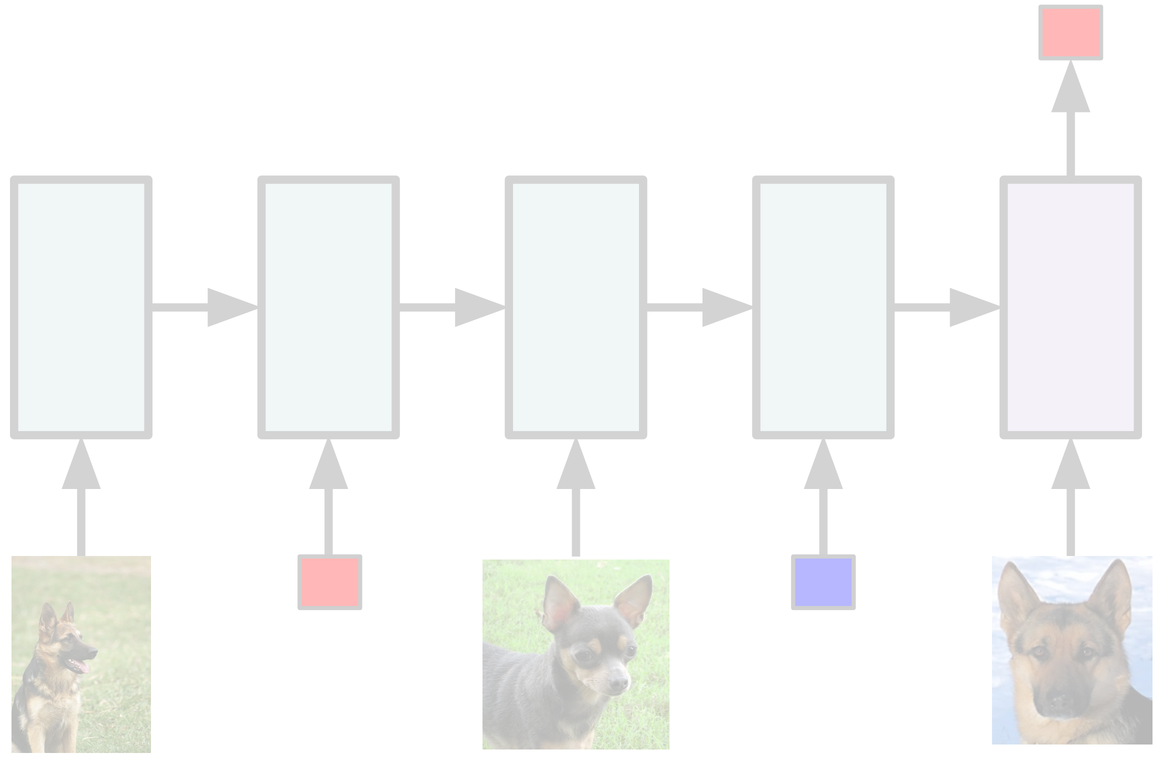


$\mathcal{D}_i^{\text{test}}$

Slide Credit: ICML 2019 Meta-Learning Tutorial

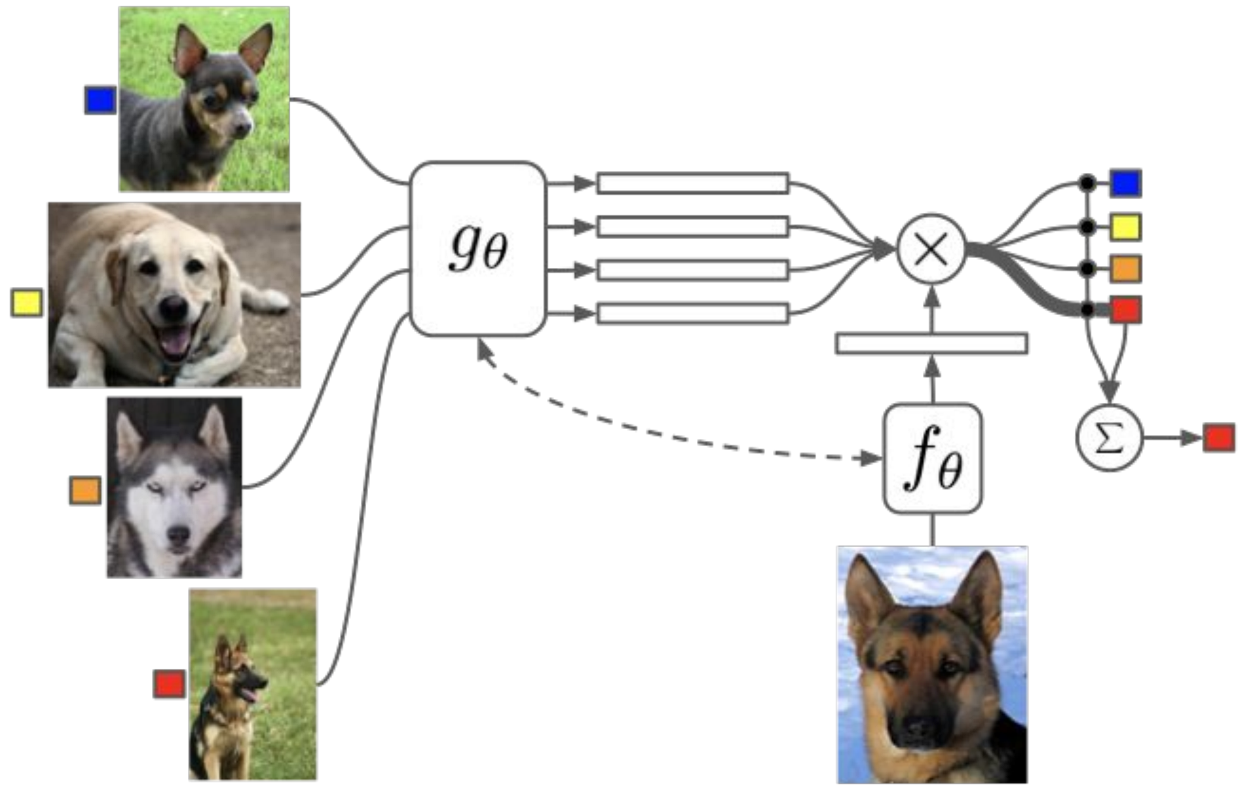
Meta Learning Algorithms Taxonomy

Model Based



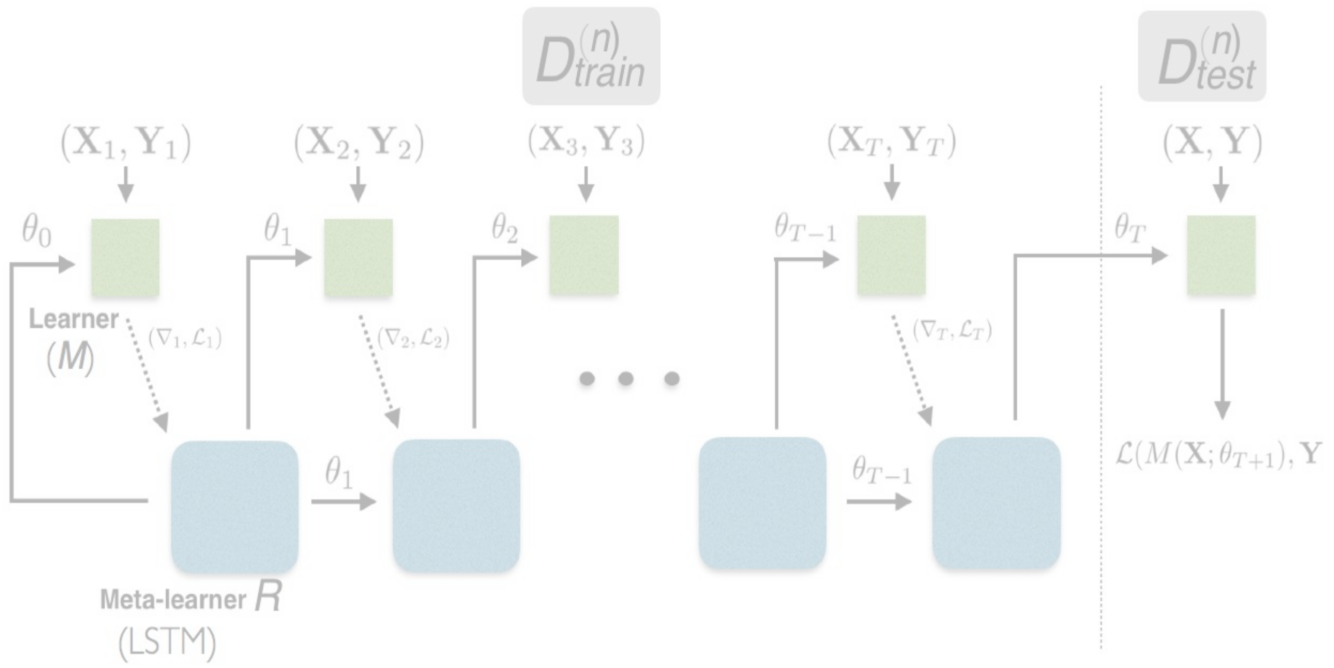
- Santoro et al. '16
- Duan et al. '17
- Wang et al. '17
- Munkhdalai & Yu '17
- Mishra et al. '17

Metric Based



- Koch '15
- Vinyals et al. '16
- Snell et al. '17
- Shyam et al. '17
- Sung et al. '17

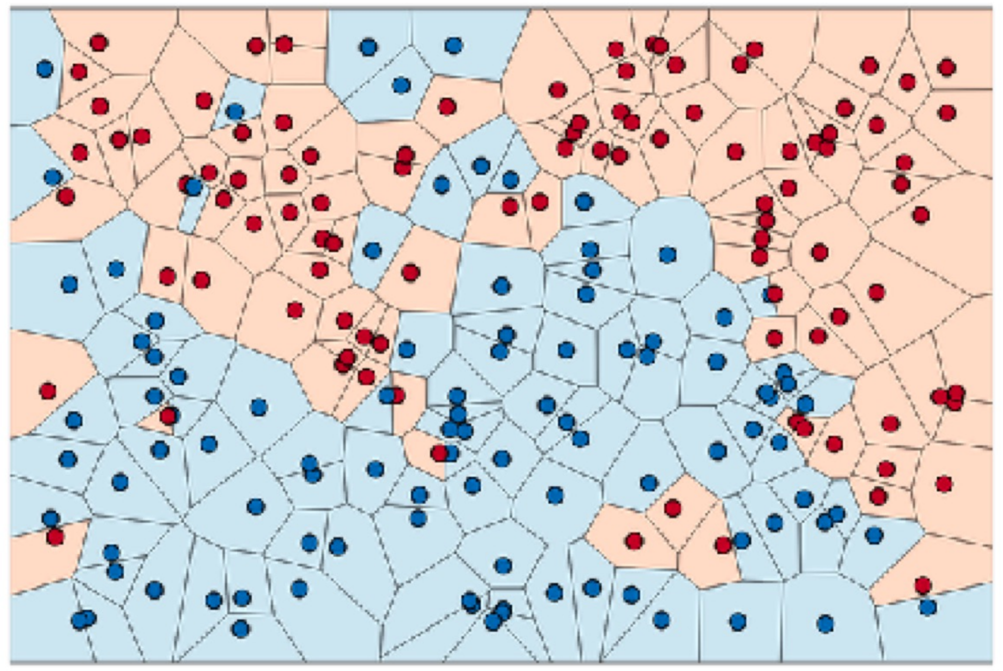
Optimization Based



- Schmidhuber '87, '92
- Bengio et al. '90, '92
- Hochreiter et al. '01
- Li & Malik '16
- Andrychowicz et al. '16
- Ravi & Larochelle '17
- Finn et al. '17

Non-Parametric (Metric-Based) Models

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

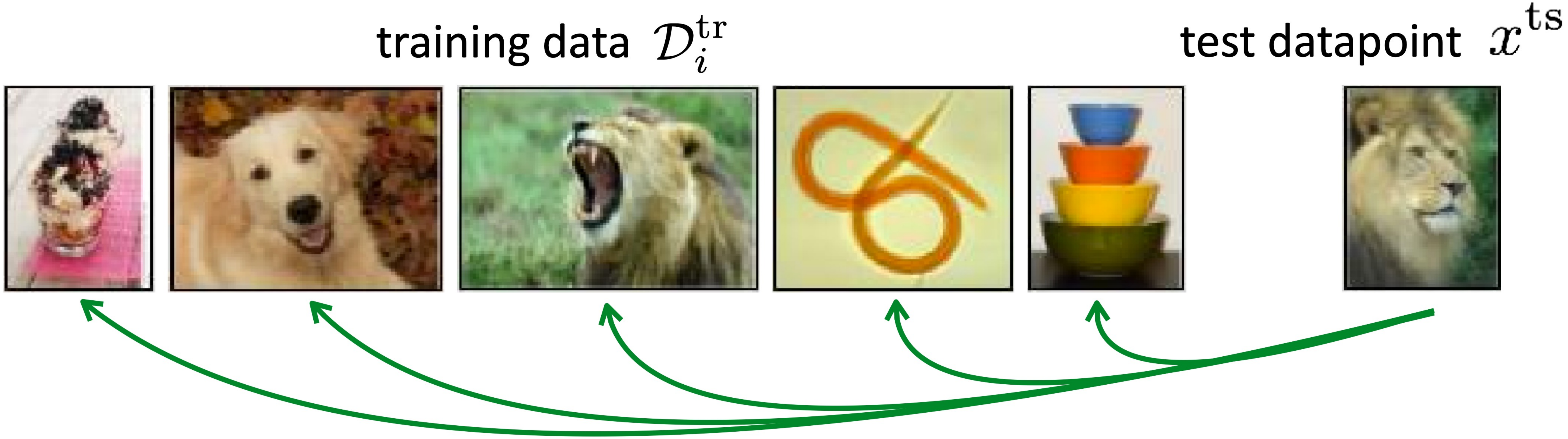


During **meta-test time**: few-shot learning <-> low data regime

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Non-parametric methods

Key Idea: Use non-parametric learner.



Compare test image with training images

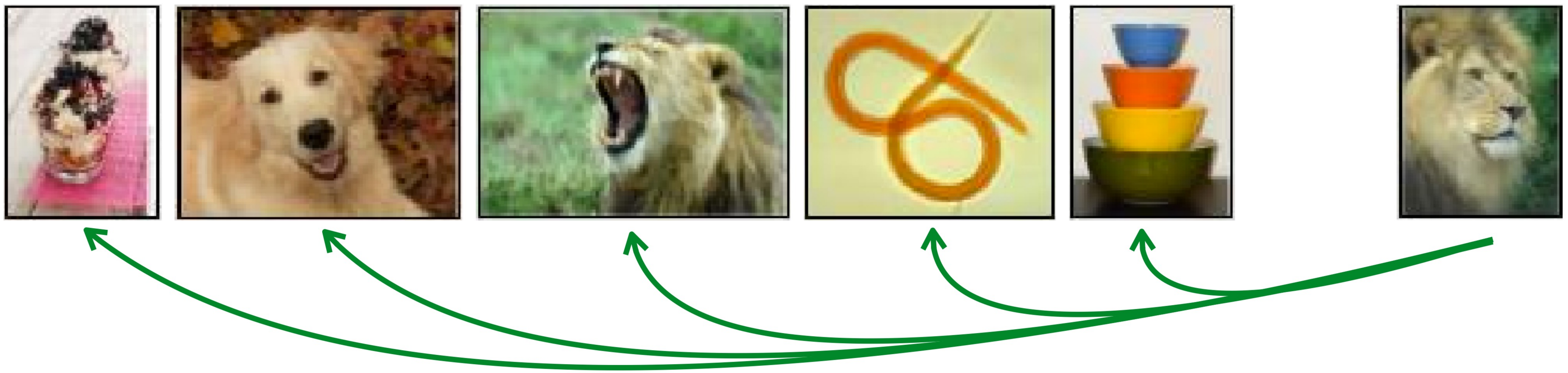
In what space do you compare? With what distance metric?

pixel space, l_2 distance?

Non-parametric methods

Key Idea: Use non-parametric learner.

training data $\mathcal{D}_i^{\text{tr}}$ test datapoint x^{ts}



Compare test image with training images

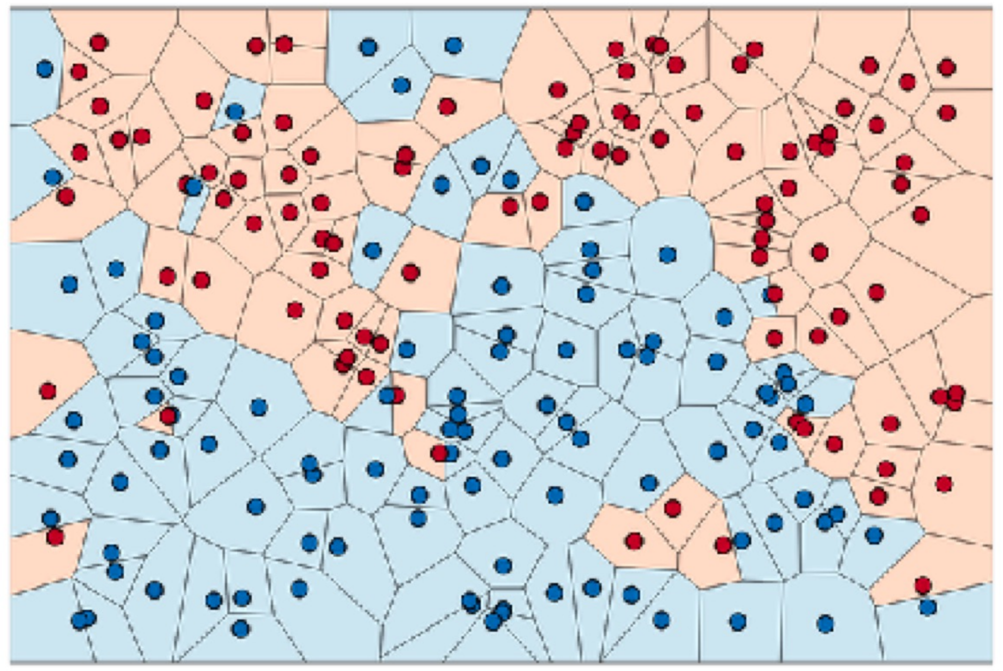
In what space do you compare? With what distance metric?

~~pixel space, l_2 distance?~~

Learn to compare using meta-training data!

Non-Parametric (Metric-Based) Models

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



During **meta-test time**: few-shot learning <-> low data regime

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

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

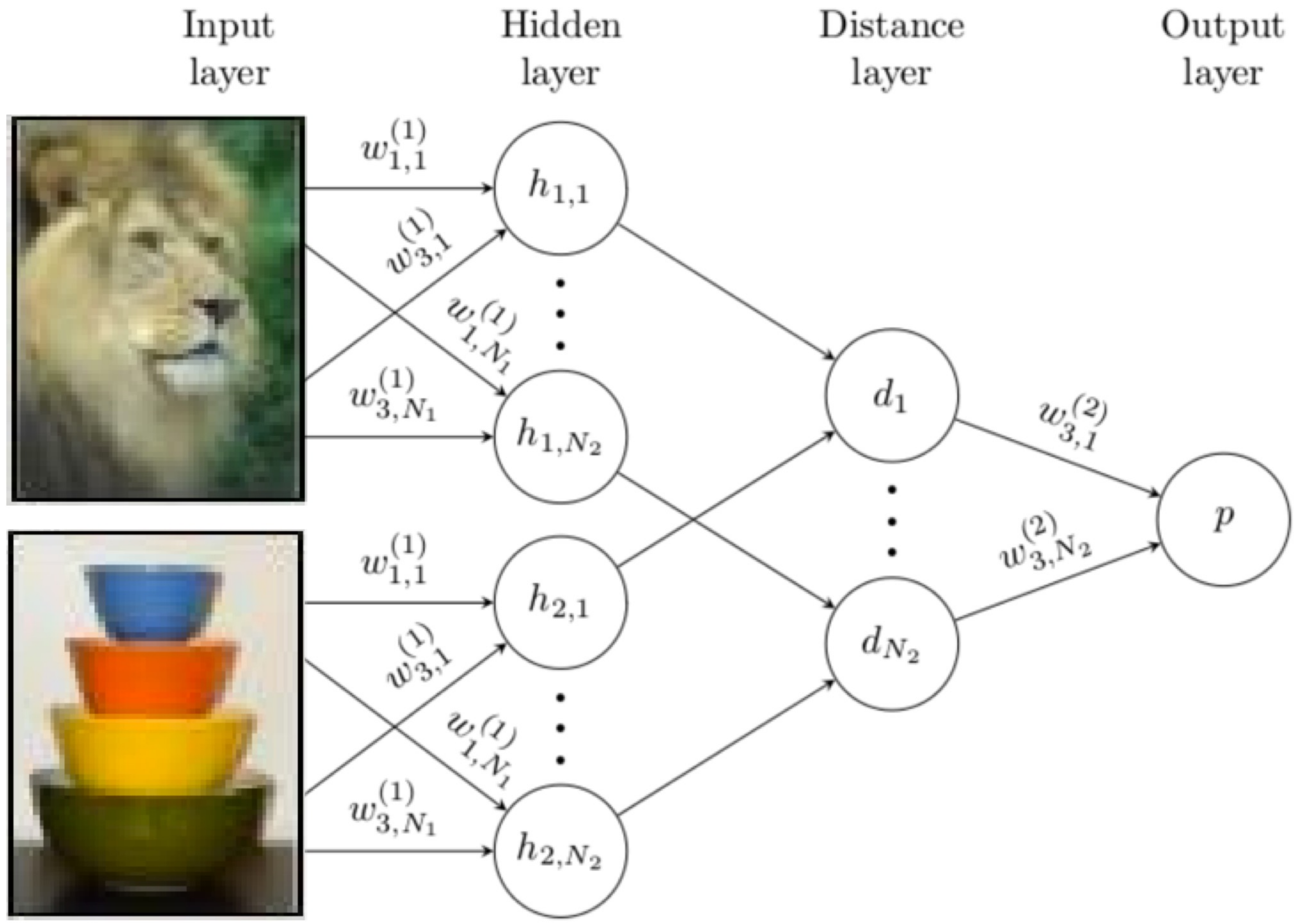
Note: some of these methods precede parametric approaches

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Non-parametric methods

Key Idea: Use non-parametric learner.

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



label

0

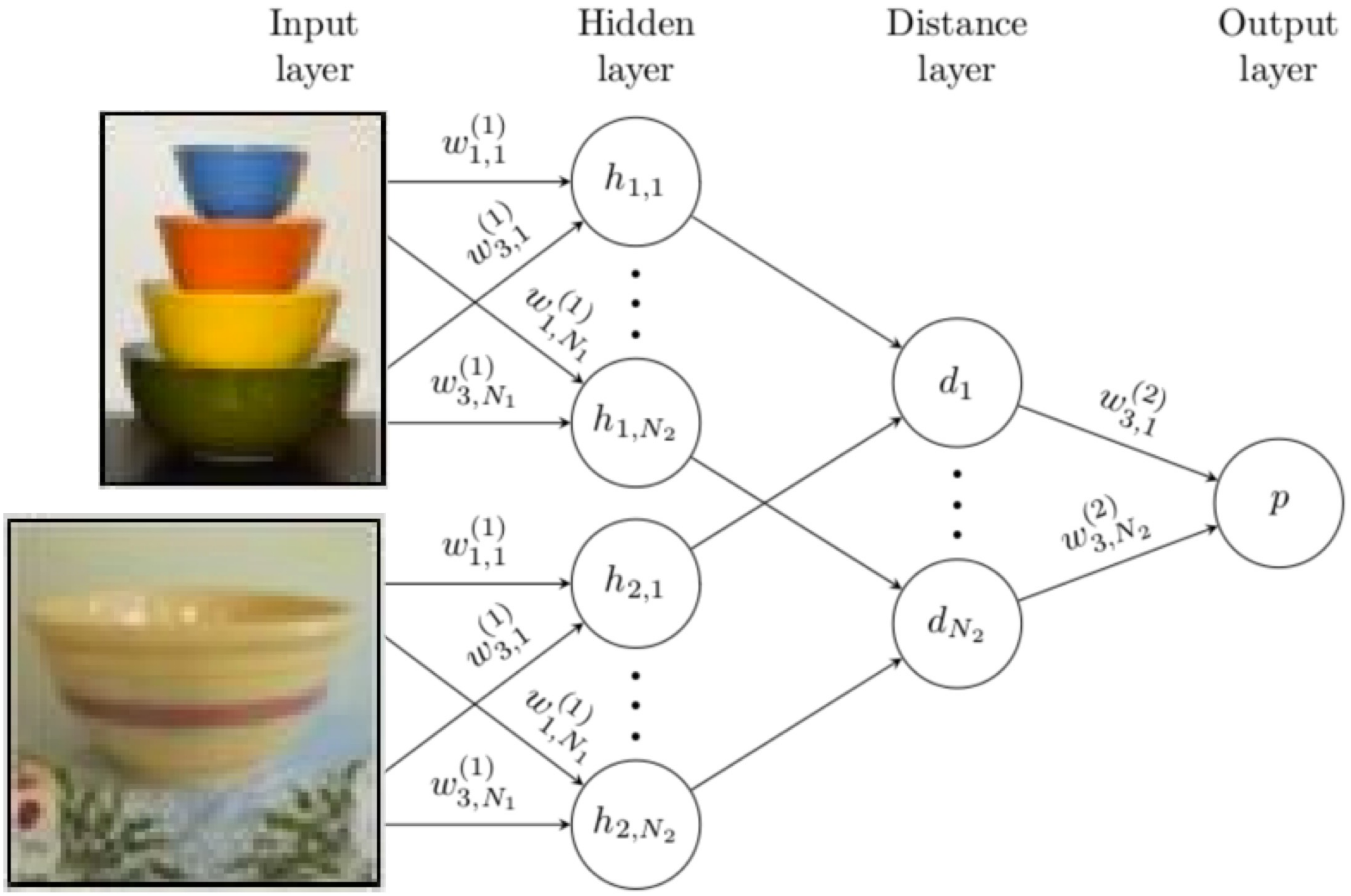
Koch et al., ICML ‘15

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Non-parametric methods

Key Idea: Use non-parametric learner.

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



label

1

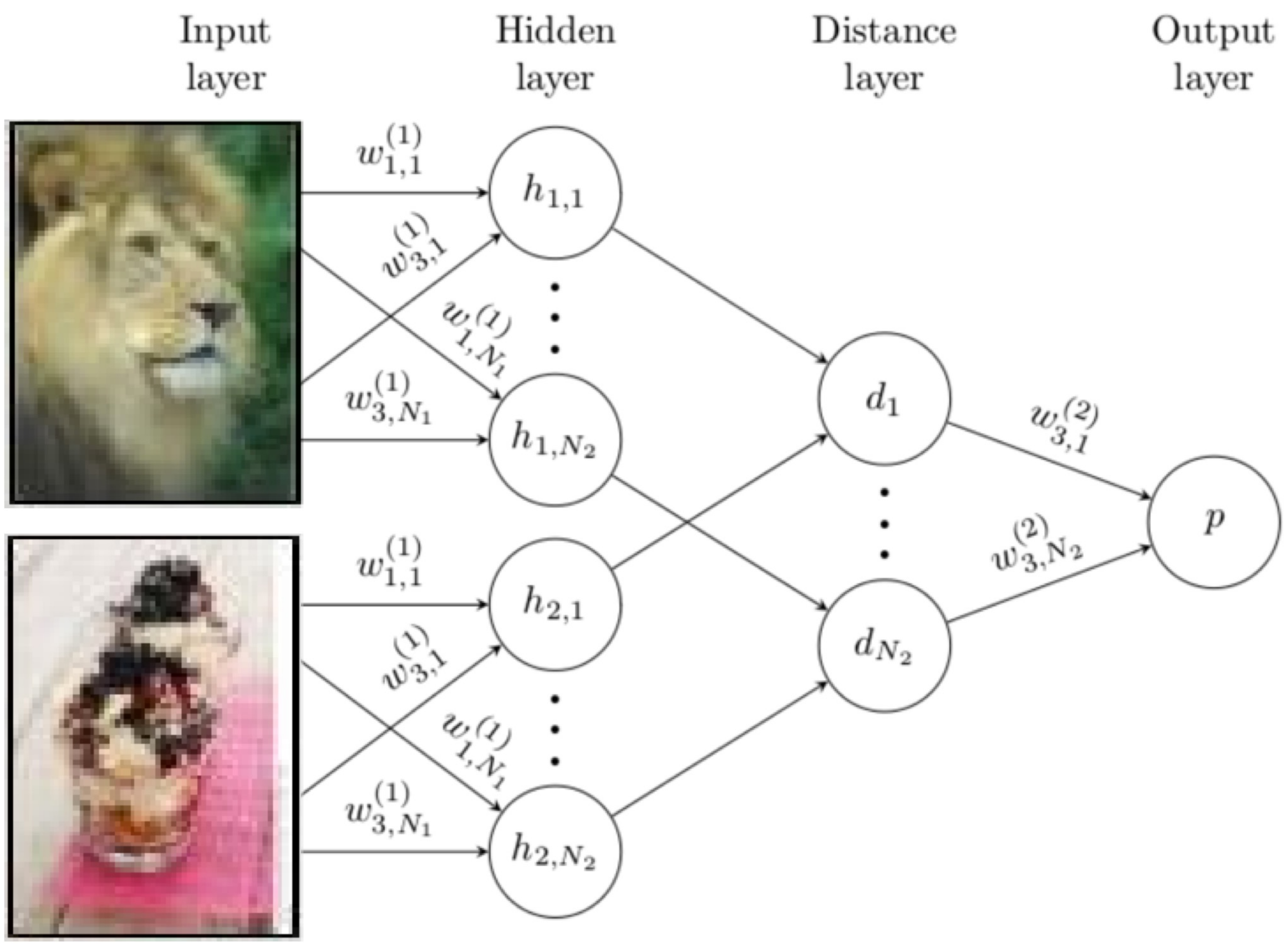
Koch et al., ICML '15

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Non-parametric methods

Key Idea: Use non-parametric learner.

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



label

0

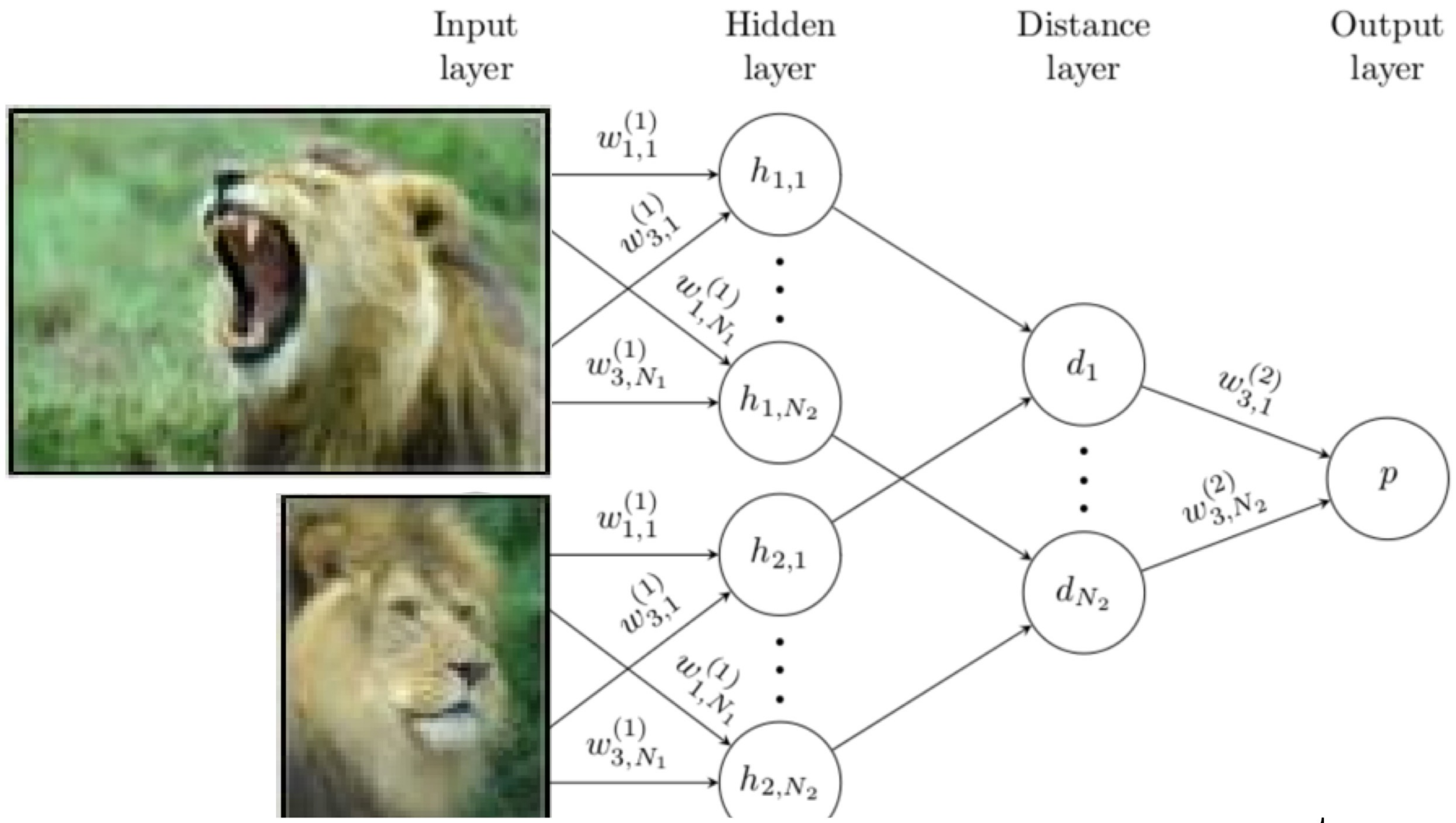
Koch et al., ICML '15

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Non-parametric methods

Key Idea: Use non-parametric learner.

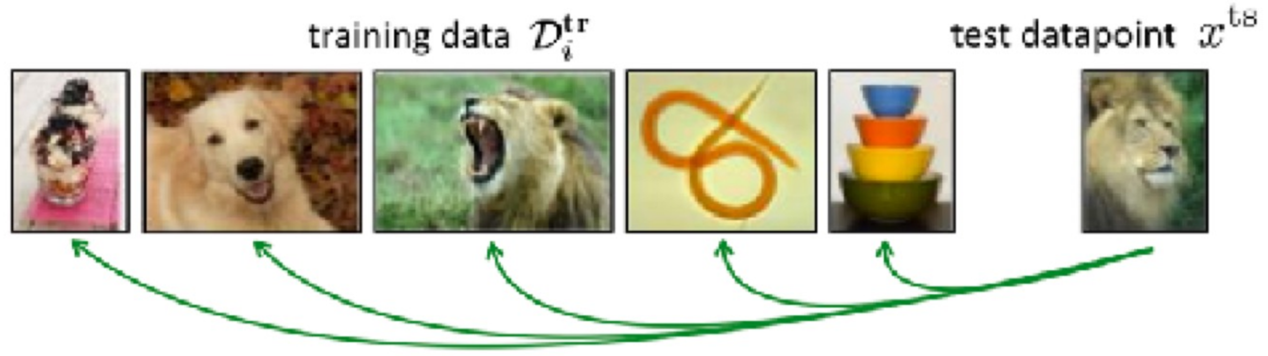
train Siamese network to predict whether or not two images are the same class



label

1

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



Meta-training: Binary classification
Meta-test: N-way classification

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

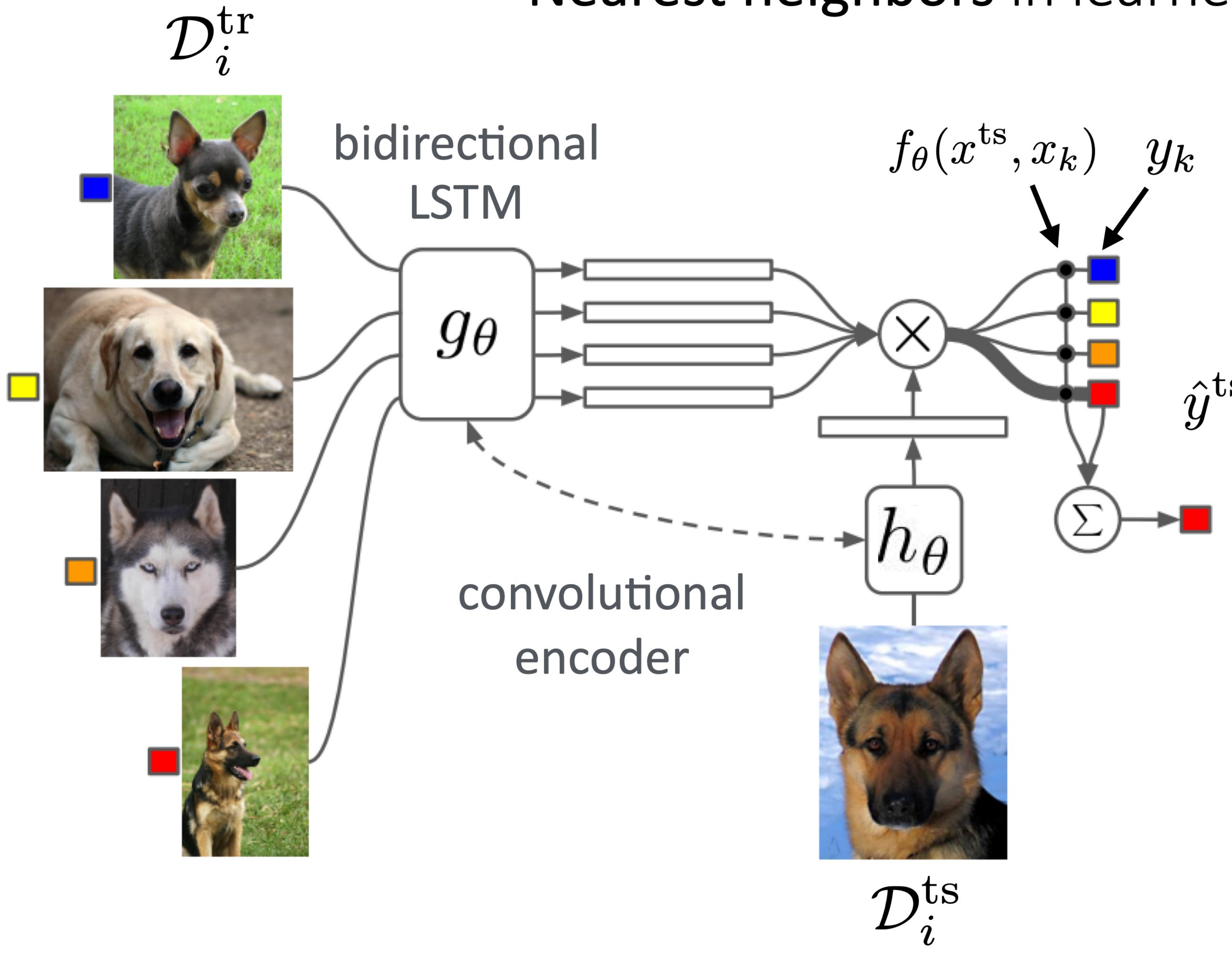
Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

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

Nearest neighbors in learned embedding space



$$\hat{y}^{\mathrm{ts}} = \sum_{x_k, y_k \in \mathcal{D}^{\mathrm{tr}}} f_\theta(x^{\mathrm{ts}}, x_k) y_k$$

Trained **end-to-end**.

Meta-train & **meta-test** time **match**.

Vinyals et al. Matching Networks, NeurIPS '16

Non-parametric methods

Key Idea: Use non-parametric learner.

General Algorithm:

~~Amortized approach~~ Non-parametric approach (matching networks)

1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\mathrm{tr}}, \mathcal{D}_i^{\mathrm{test}}$ from $\mathcal{D}_i$
3. ~~Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\mathrm{tr}})$~~ Compute $\hat{y}^{\mathrm{ts}} = \sum_{x_k, y_k \in \mathcal{D}^{\mathrm{tr}}} f_\theta(x^{\mathrm{ts}}, x_k) y_k$
4. ~~Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\mathrm{test}})$~~ Update θ using $\nabla_\theta \mathcal{L}(\hat{y}^{\mathrm{ts}}, y^{\mathrm{ts}})$

(Parameters ϕ integrated out, hence **non-parametric**)

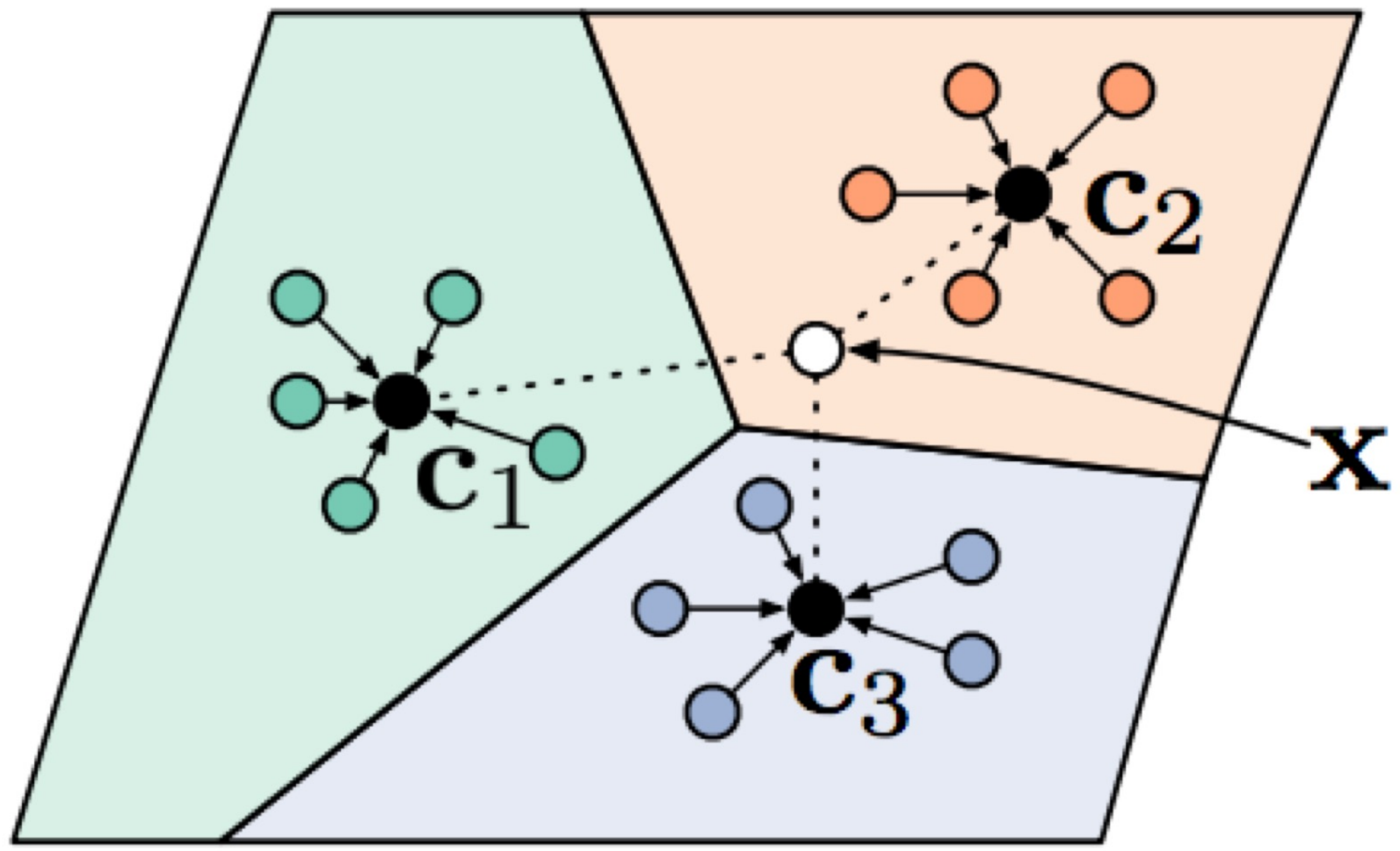
What if **>1 shot**?

Matching networks will perform comparisons **independently**

Can we **aggregate class information** to create a **prototypical embedding**?

Non-parametric methods

Key Idea: Use non-parametric learner.



(a) Few-shot

$$\mathbf{c}_n = \frac{1}{K} \sum_{(x,y)\in\mathcal{D}_i^{\mathrm{tr}}} \mathbb{1}(y=n) f_\theta(x)$$

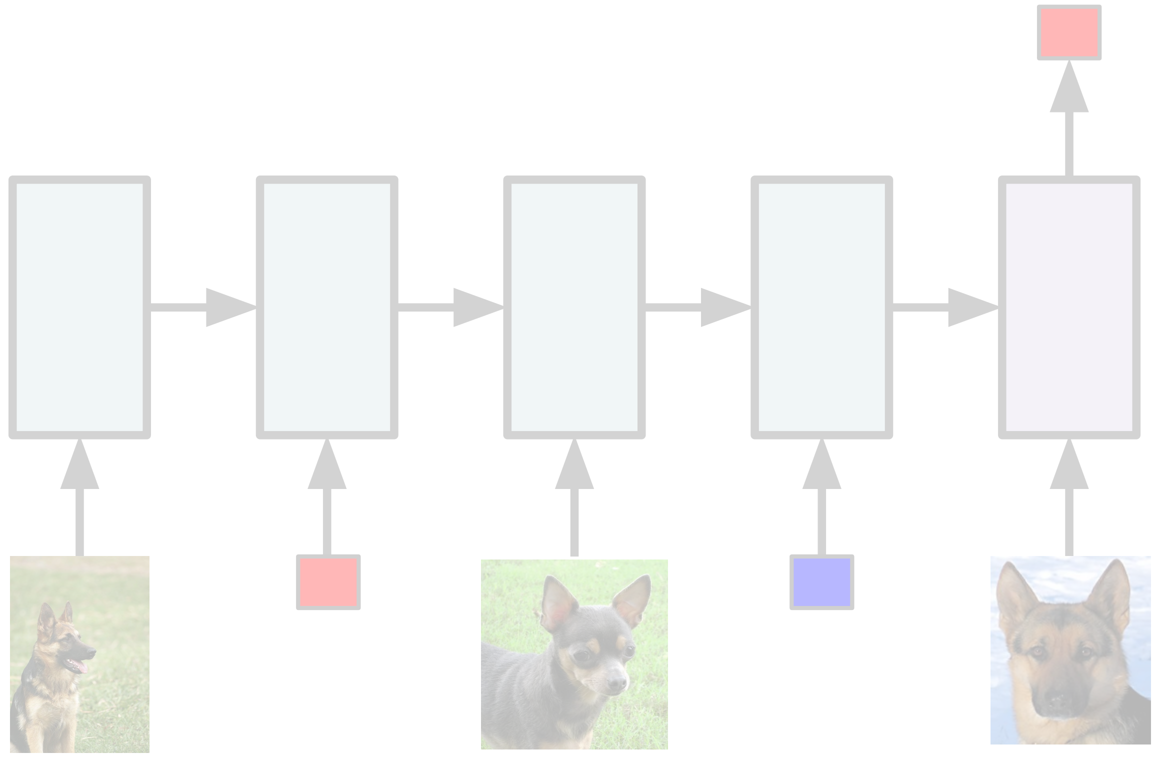
$$p_\theta(y=n|x) = \frac{\exp(-d\left(f_\theta(x), \mathbf{c}_n\right))}{\sum_{n'} \exp(d(f_\theta(x), \mathbf{c}_{n'}))}$$

d: Euclidean, or cosine distance

Snell et al. Prototypical Networks, NeurIPS '17

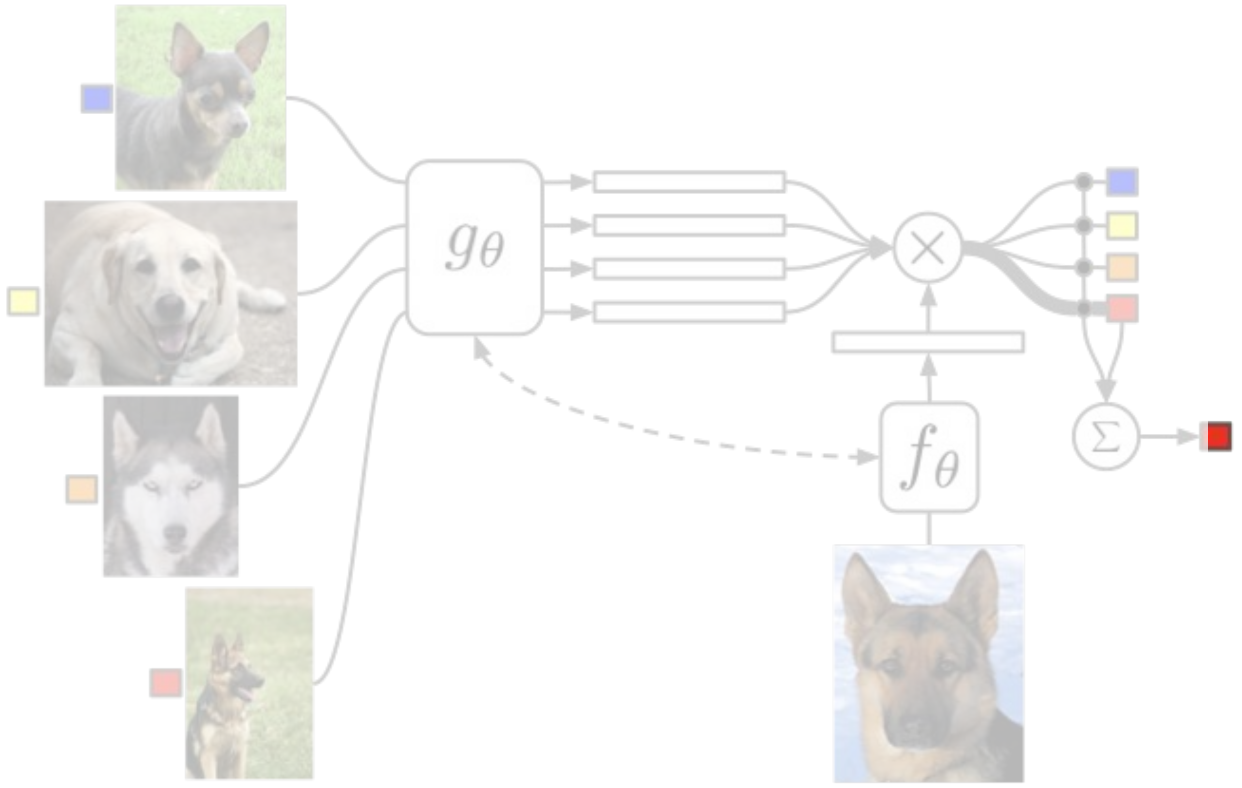
Meta Learning Algorithms Taxonomy

Model Based



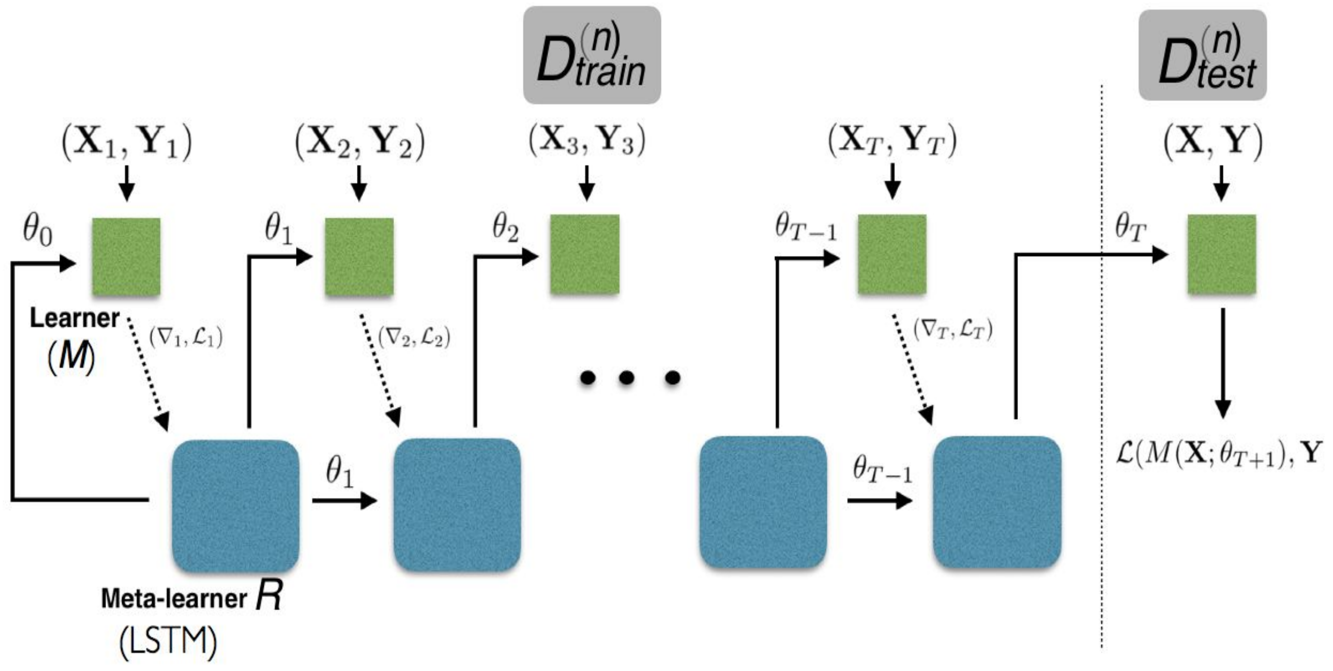
- Santoro et al. '16
- Duan et al. '17
- Wang et al. '17
- Munkhdalai & Yu '17
- Mishra et al. '17

Metric Based



- Koch '15
- Vinyals et al. '16
- Snell et al. '17
- Shyam et al. '17
- Sung et al. '17

Optimization Based



- Schmidhuber '87, '92
- Bengio et al. '90, '92
- Hochreiter et al. '01
- Li & Malik '16
- Andrychowicz et al. '16
- Ravi & Larochelle '17
- Finn et al. '17

Slide Credit: Oriol Vinyals, Meta Learning Symposium (NIPS 2017)

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

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

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

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

Optimization-Based Inference

pre-trained parameters

Fine-tuning $\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$

training data
for new task

(typically for many gradient steps)

Optimization-Based Inference

pre-trained parameters

Fine-tuning

[test-time]

$$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\mathrm{tr}})$$

training data
for new task

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

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

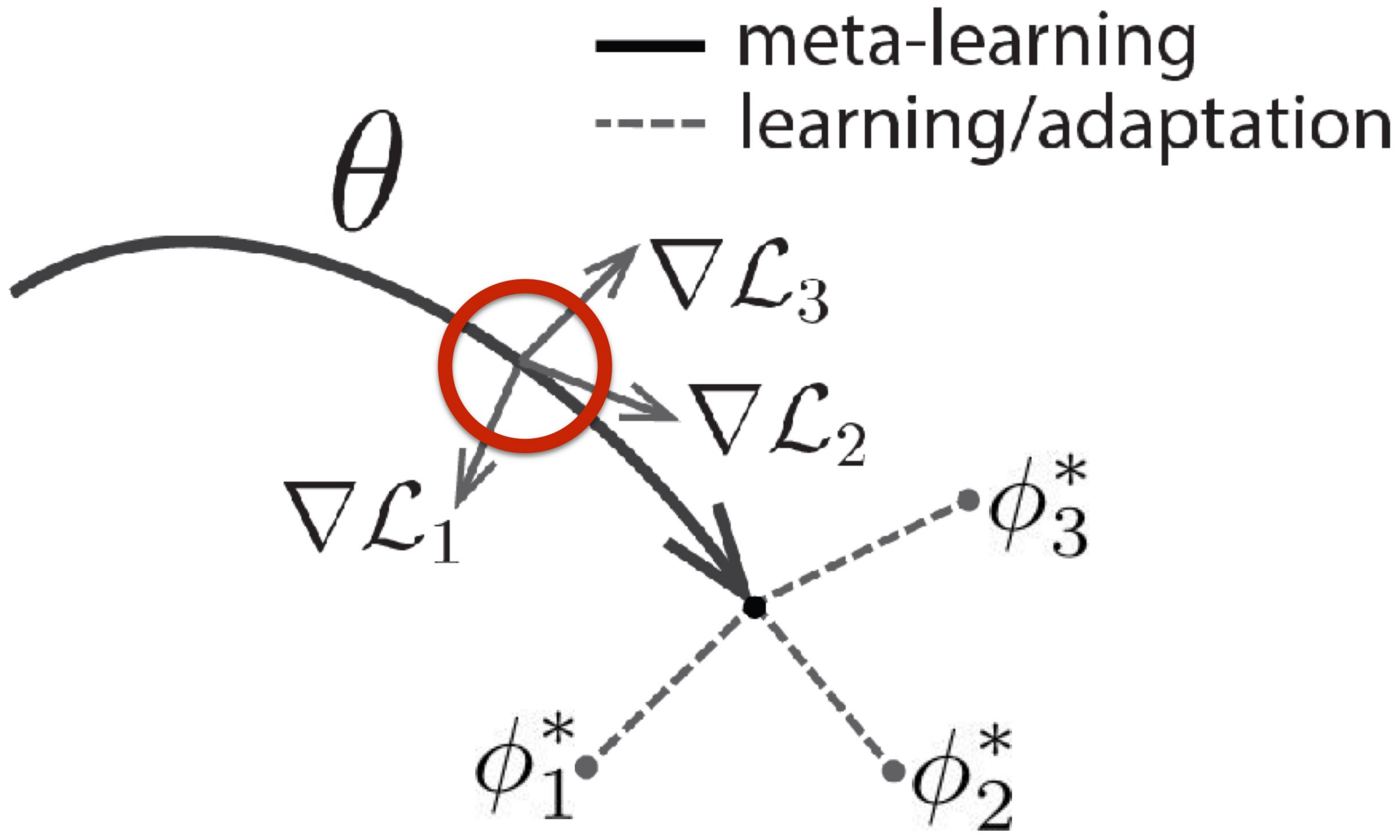
Finn, Abbeel, Levine. Model-Agnostic Meta-Learning. ICML 2017

Optimization-Based Inference

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

θ parameter vector being meta-learned

ϕ_i^* optimal parameter vector for task i



Model-**A**gnostic **M**eta-**L**earning

Finn, Abbeel, Levine. Model-Agnostic Meta-Learning. ICML 2017

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

General Algorithm:

~~Amortized approach~~ Optimization-based approach

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

—> brings up **second-order** derivatives

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Challenges

Backpropagating through many inner gradient steps is compute- & memory-intensive.

Idea: [Crudely] approximate $\frac{d\phi_i}{d\theta}$ as identity

(Finn et al. first-order MAML ‘17, Nichol et al. Reptile ’18)

Takeaway: works for simple few-shot problems, but (anecdotally) not for more complex meta-learning problems.

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Takeaways: Construct *bi-level optimization* problem.

+ positive inductive bias at the start of meta-learning
+ consistent procedure, tends to extrapolate better
+ maximally expressive with sufficiently deep network
+ model-agnostic (easy to combine with your favorite architecture)

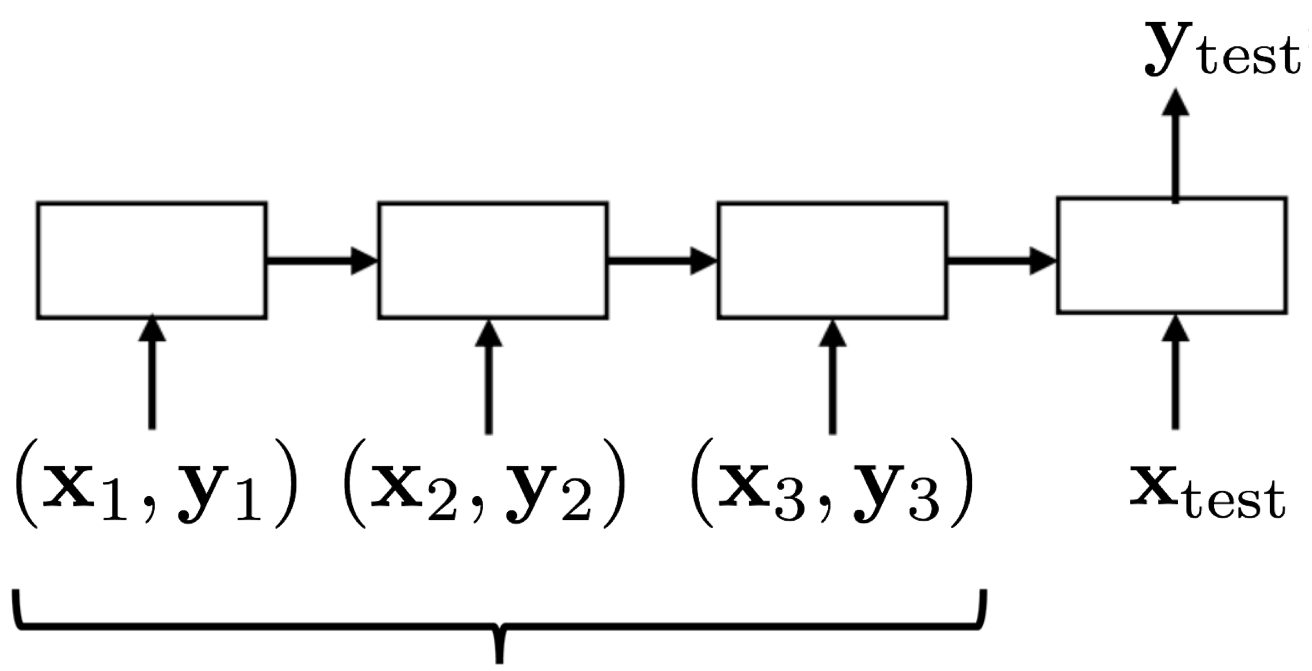
- typically requires second-order optimization
- usually compute and/or memory intensive

Design of ***f*** ?

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

Recurrent network

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



network implements the
"learned learning procedure"

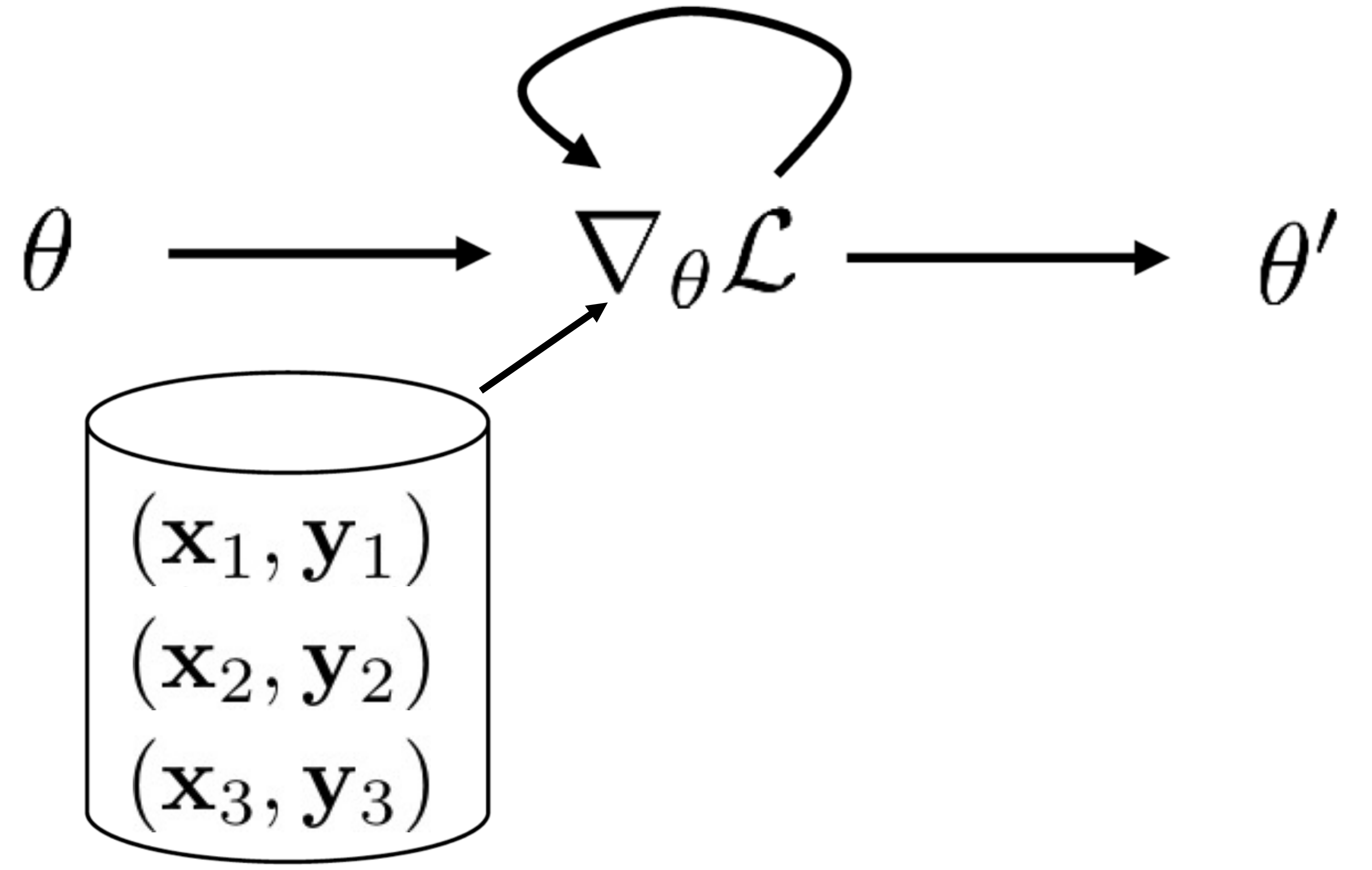
Does it converge?
- Sort of?

What does it converge to?
- Who knows…

What to do if not good enough?
- Nothing

MAML

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



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

What does it converge to?
- A local optimum (it's gradient descent…)

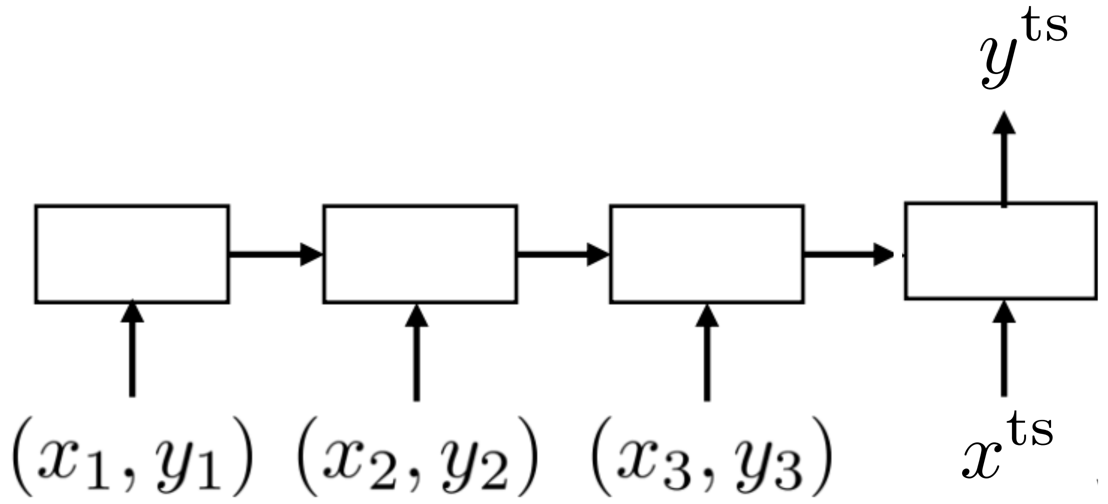
What to do if not good enough?
- Keep taking gradient steps (it's gradient descent..)

Meta learning algorithms taxonomy

Computation graph perspective

Black-box amortized

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



Optimization-based

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

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

Non-parametric

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

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

Black-box vs. Optimization vs. Non-Parametric

Black-box amortized

+ easy to combine with **variety of learning problems** (e.g. SL, RL)

- **challenging optimization** (no inductive bias at the initialization)
- often **data-inefficient**
- **model** & **architecture** intertwined

Optimization-based

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

- **second-order optimization**

Non-parametric

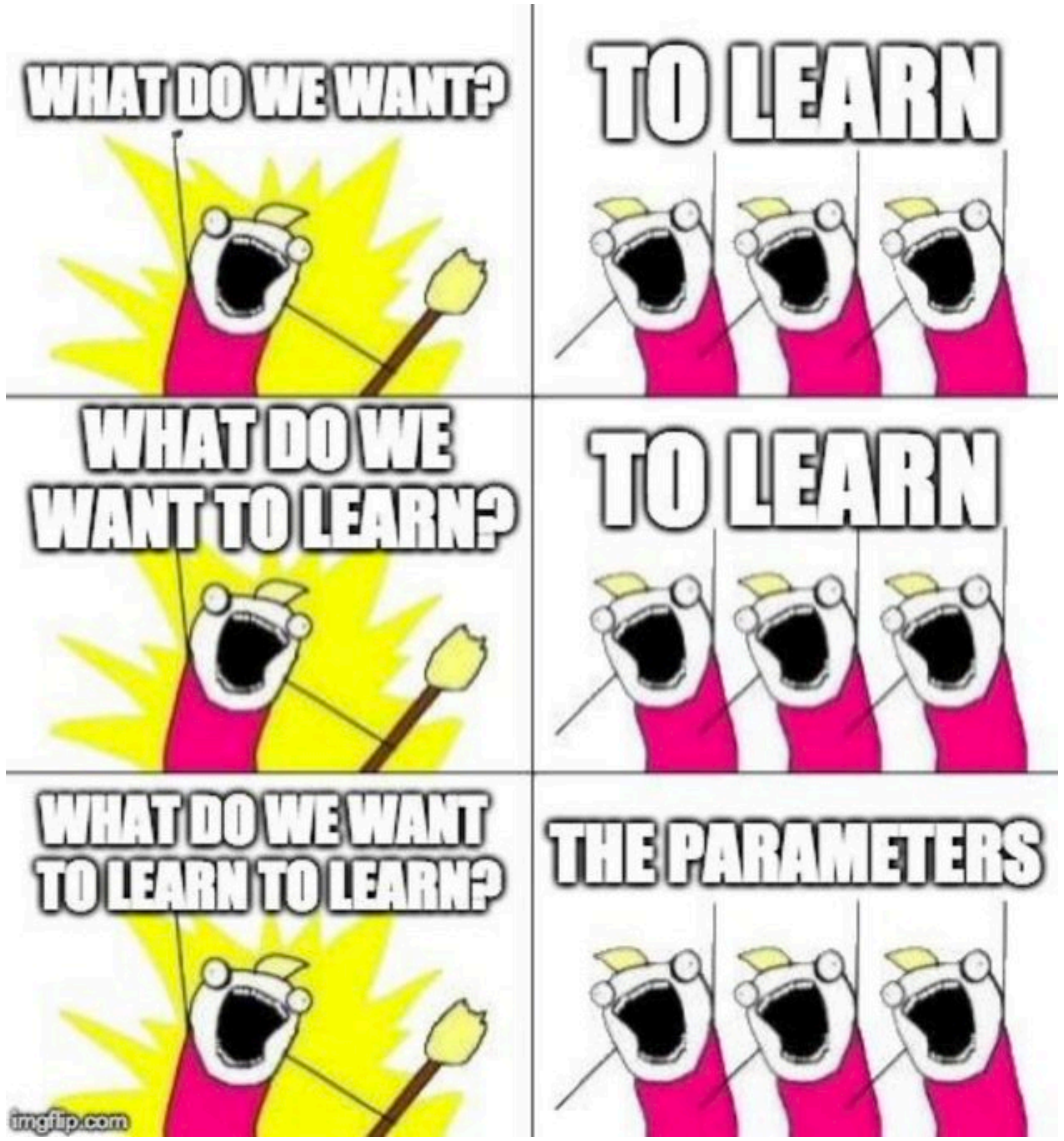
+ **simple**
+ entirely **feedforward**
+ **computationally fast** & **easy to optimize**

- **harder to generalize** to **varying K**
- hard to scale to **very large K**
- so far, **limited to classification**

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

Slide Credit: CS 330: Deep Multi-Task and Meta Learning

Thank you!


WHAT DO WE WANT?
TO LEARN
WHAT DO WE WANT TO LEARN?
TO LEARN
WHAT DO WE WANT TO LEARN TO LEARN?
THE PARAMETERS
imgflip.com