DSC190: Machine Learning with Few Labels

Meta learning

Zhiting Hu

Lecture 15, November 16, 2021

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Outline

- Meta learning (50mins)

- 2 Paper presentations (30 mins)
  - **Yuan Gao:** NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
  - **Lechuan Wang**: UNIPELT: A Unified Framework for Parameter-Efficient Language Model Tuning

Learning with multiple tasks

- Multi-task learning
- Transfer learning
- Meta learning
- (Continual/Lifelong learning, ..)

Learning with multiple tasks

Multi-Task Learning

Solve multiple tasks $\mathscr{T}_1, \cdots, \mathscr{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathscr{L}_i(\theta, \mathscr{D}_i)$$

Transfer Learning

Solve target task $\mathscr{T}_b$ after solving source task $\mathscr{T}_a$ by *transferring* knowledge learned from $\mathscr{T}_a$

assumption: Cannot access data $\mathscr{D}_a$ during transfer.

Transfer learning is a valid solution to multi-task learning.
(but not vice versa)

Question: What are some problems/applications where transfer learning might make sense?

when $\mathscr{D}_a$ is very large
(don't want to retain & retrain on $\mathscr{D}_a$)

when you don't care about solving $\mathscr{T}_a$ & $\mathscr{T}_b$ simultaneously

Transfer learning

Parameters pre-trained on $\mathscr{D}_a$

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

training data for new task $\mathscr{T}_b$

(typically for many gradient steps)

Pre-trained Dataset	PASCAL	SUN
Original	58.3	52.2
Random	41.3 [21]	35.7 [2]

What makes ImageNet good for transfer learning? Huh, Agrawal, Efros. '16

Where do you get the pre-trained parameters?

- ImageNet classification
- Models trained on large language corpora (BERT, LMs)
- Other unsupervised learning techniques
- Whatever large, diverse dataset you might have

Pre-trained models often available online.

Some common practices

- Fine-tune with a smaller learning rate
- Smaller learning rate for earlier layers
- Freeze earlier layers, gradually unfreeze
- Reinitialize last layer
- Search over hyperparameters via cross-val
- Architecture choices matter (e.g. ResNets)

Universal Language Model Fine-Tuning for Text Classification. Howard, Ruder. '18

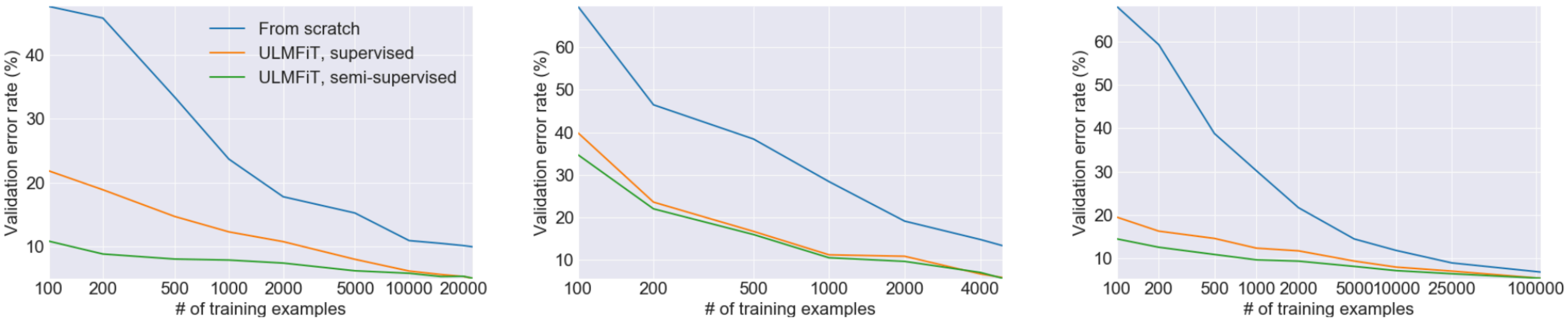


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

Fine-tuning doesn't work well with very small target task datasets

This is where meta-learning can help.

Slide courtesy: Finn, Stanford CS 330, Fall 2021

Example: Meta-learning for few-shot learning

Given 1 example of 5 classes:

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

Classify new examples

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

Slide courtesy: Finn, Stanford CS 330, Fall 2021

Example: Meta-learning for few-shot learning

Can replace image classification with: regression, language generation, skill learning, **any ML problem**

Two views of meta-learning

- **Mechanistic view**
  - Deep network that reads an entire dataset and then makes predictions for new datapoints
  - Dataset → datapoint; therefore we now have meta-dataset of datasets
- **Probabilistic view**
  - Extract prior from a set of (meta-training) tasks that allows efficient learning of new tasks
  - A new task uses this prior plus small training set to infer most likely parameters

Leveraging related tasks, either in terms of data or computations

- Learning to learn from few examples (few-shot learning)
- Learning to optimize
- AutoML, architecture search, meta-learning new algorithms
- …

Slide courtesy: Andreas, MIT 6.884

Two views of meta-learning

- **Mechanistic view**
  - Deep network that reads an entire dataset and then makes predictions for new datapoints
  - Dataset → datapoint; therefore we now have meta-dataset of datasets
- **Probabilistic view**
  - Extract prior from a set of (meta-training) tasks that allows efficient learning of new tasks
  - A new task uses this prior plus small training set to infer most likely parameters

Leveraging related tasks, either in terms of data or computations

- Learning to learn from few examples (few-shot learning)
- Learning to optimize
- AutoML, architecture search, meta-learning new algorithms
- …

Slide courtesy: Andreas, MIT 6.884

The Meta-Learning Problem

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, quickly solve new task $\mathcal{T}_{\text{test}}$

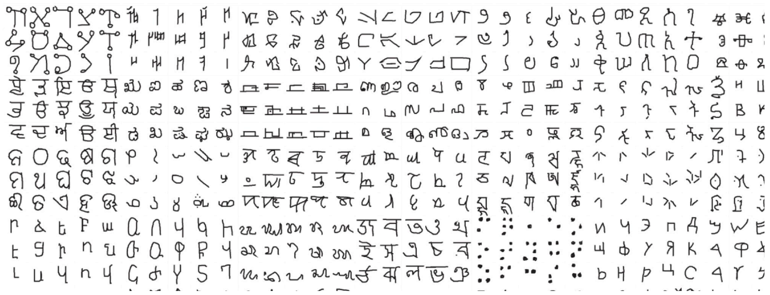
Key assumption: meta-training tasks and meta-test task drawn i.i.d. from same task distribution

$$\mathcal{T}_1, \ldots, \mathcal{T}_n \sim p(\mathcal{T}), \mathcal{T}_j \sim p(\mathcal{T})$$

Like before, tasks must share structure.

What do the tasks correspond to?

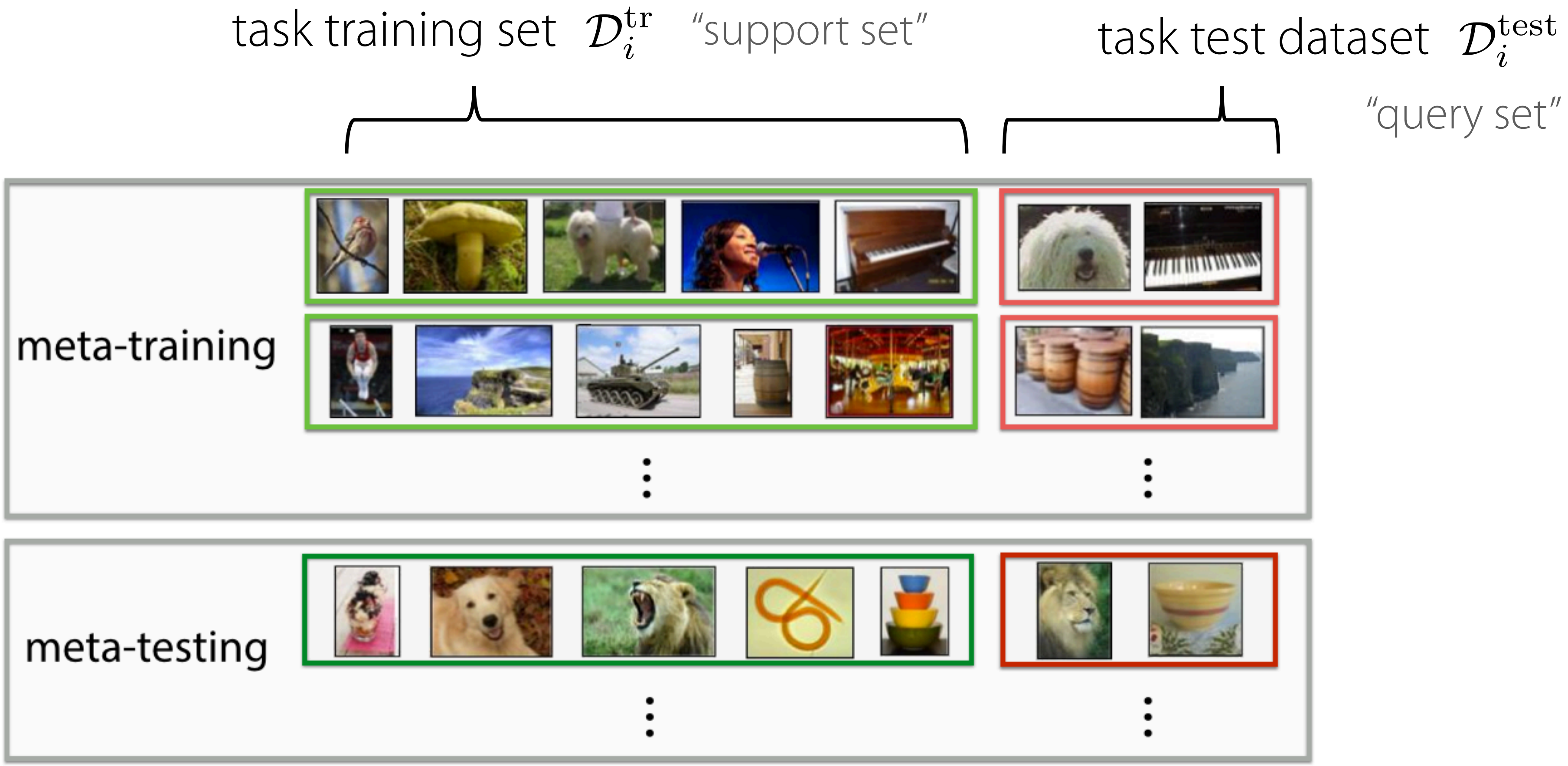
- recognizing handwritten digits from different languages
- giving feedback to students on different exams
- classifying species in different regions of the world
- a robot performing different tasks



How many tasks do you need? The more the better. (analogous to more data in ML)

Slide courtesy: Finn, Stanford CS 330, Fall 2021

Terminology



k-shot learning: learning with **k** examples per class (or **k** examples total for regression)

N-way classification: choosing between N classes

Slide courtesy: Finn, Stanford CS 330, Fall 2021

Problem Settings Recap

Multi-Task Learning

Solve multiple tasks $\mathcal{T}_1, \cdots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$$

Transfer Learning

Solve target task $\mathcal{T}_b$ after solving source task $\mathcal{T}_a$ by *transferring* knowledge learned from $\mathcal{T}_a$

The Meta-Learning Problem

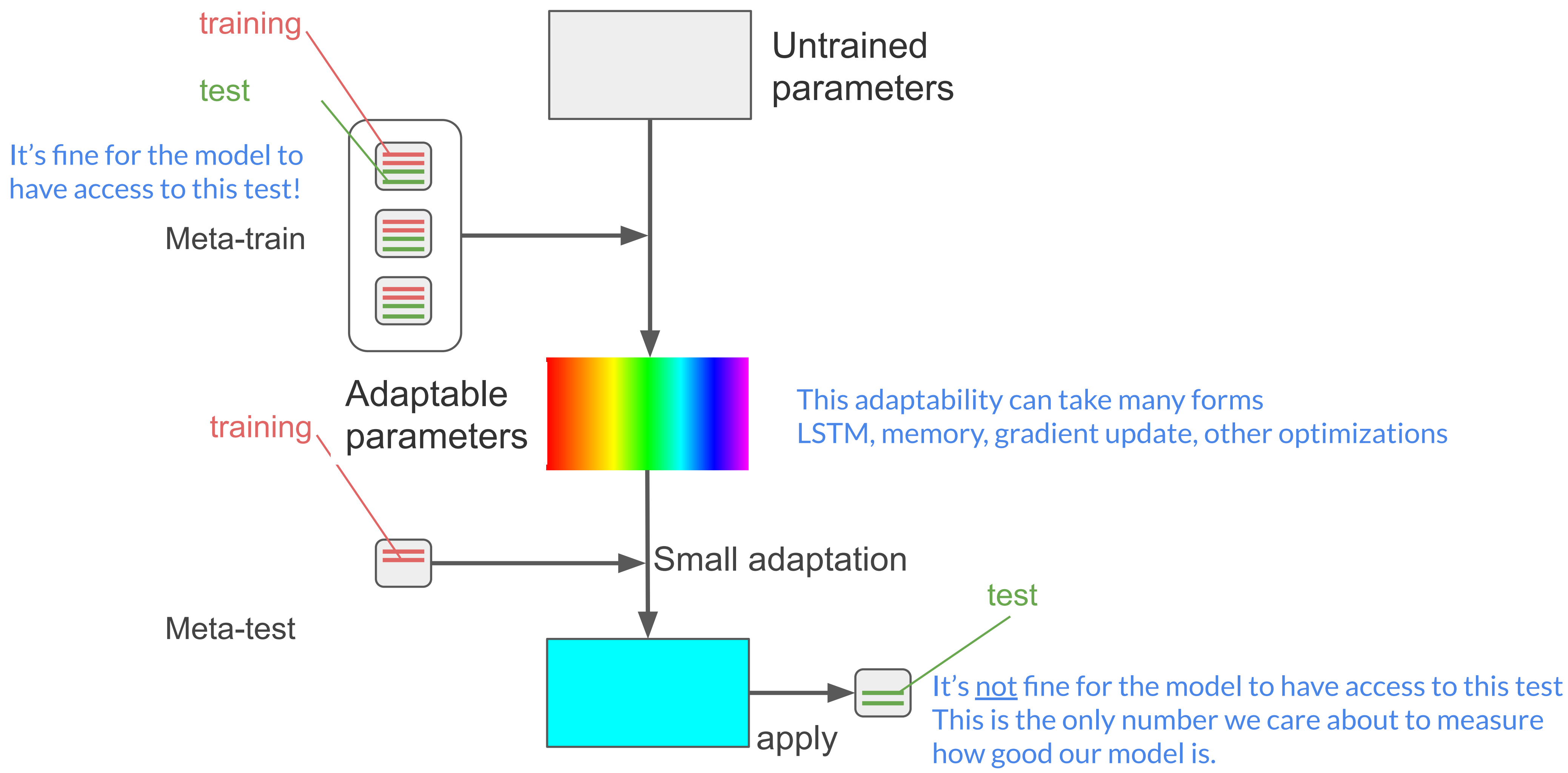
Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, quickly solve new task $\mathcal{T}_{\text{test}}$

In transfer learning and meta-learning:
generally impractical to access prior tasks

In all settings: tasks must share structure.

Slide courtesy: Finn, Stanford CS 330, Fall 2021

Meta-Learning Methods



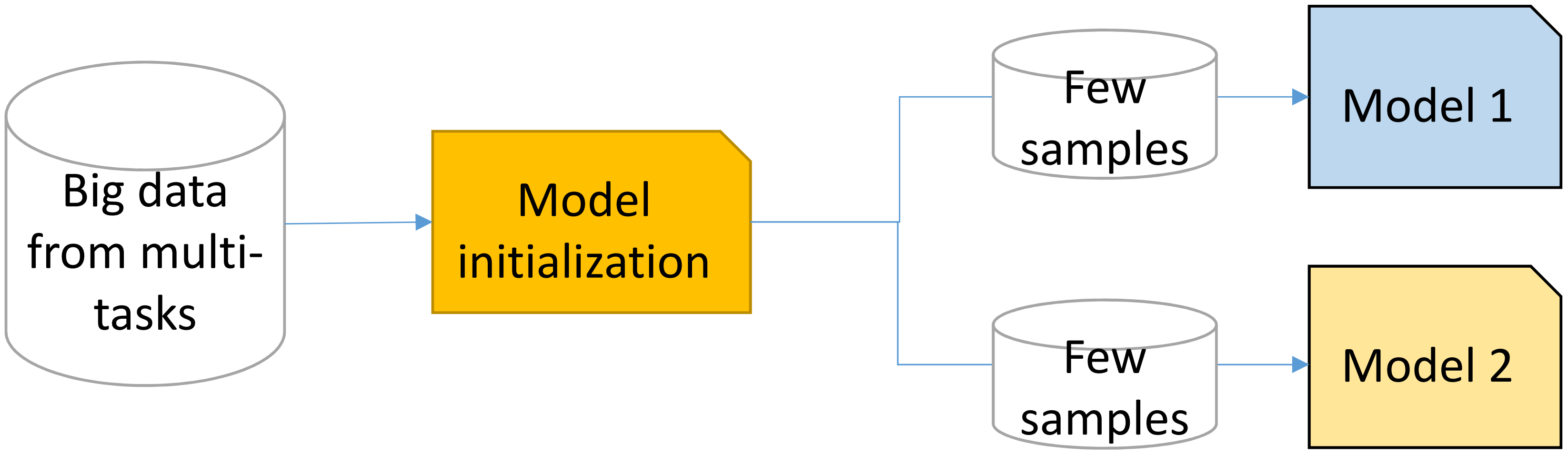
Slide courtesy: Andreas, MIT 6.884

Meta-Learning Methods

- Initialization based methods
  - Learning how to initialize the model for the new task
- Black-box adaptation methods
- Non-parametric methods

Model-Agnostic Meta Learning (MAML)

- Goal: train a model that can be fast adapted to different tasks via few shots
- MAML idea: directly optimize for an initial representation that can be effectively fine-tuned from a small number of examples



Finn et al. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.

Slide courtesy: Zhang, SJTU CS420, 2019

Recall: Fine-tuning

pre-trained parameters

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

training data for new task

Finn et al. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.

Model-Agnostic Meta Learning (MAML)

pre-trained parameters

Fine-tuning
[test-time]

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

training data
for new task

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

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

Finn et al. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.

Algorithm 2 MAML for Few-Shot Supervised Learning

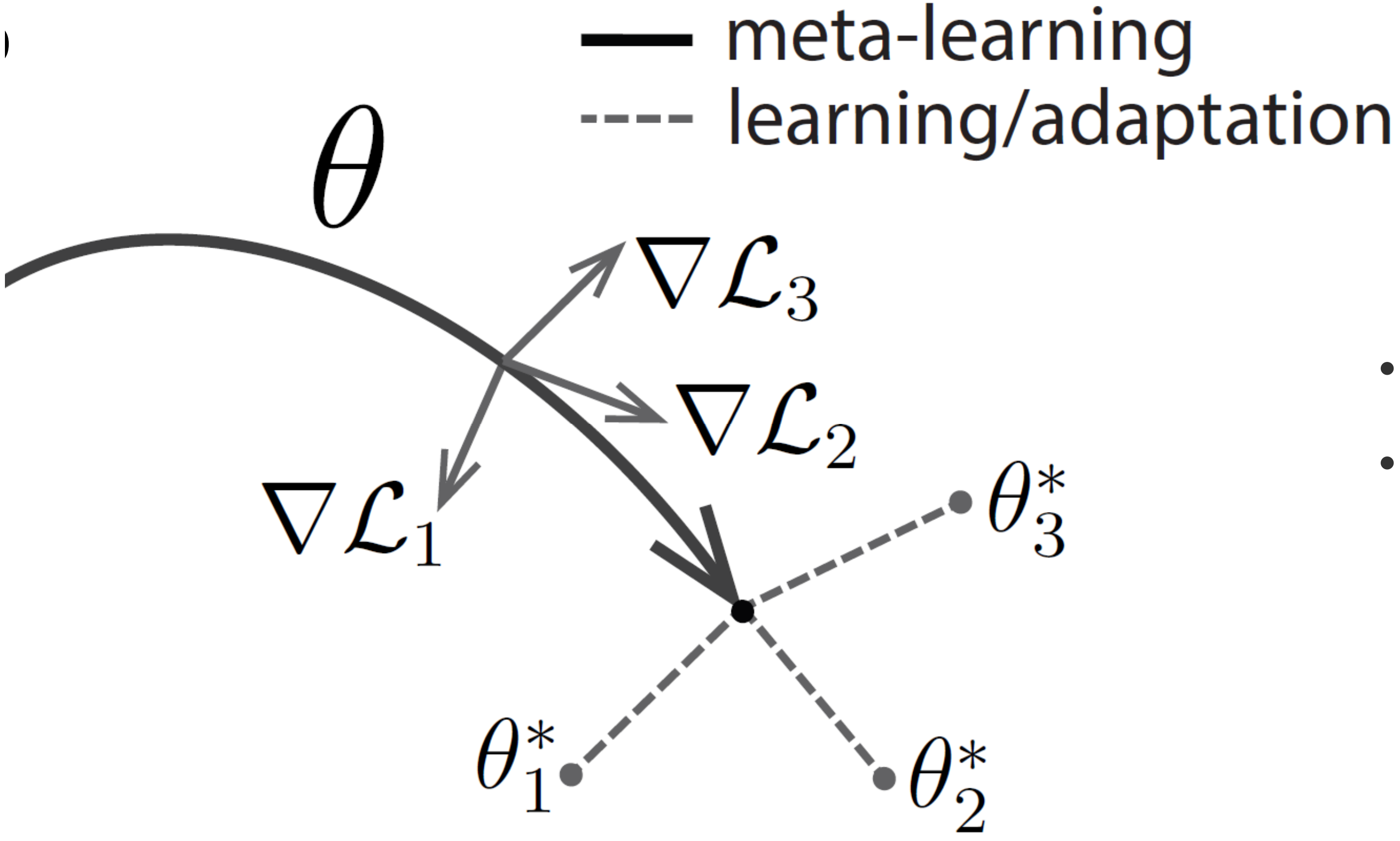
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
end while

Finn et al. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.

Model-Agnostic Meta Learning (MAML)

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



- This brings up **second-order** derivatives
- Supported by standard deep learning libraries such as PyTorch/TensorFlow

Slide courtesy: Zhang, SJTU CS420, 2019

Recall Lecture#7: Meta-learning data weights/augmentation

- Can we learn ϕ_i automatically?

$$\min_{\theta} - \mathbb{E}_{x_i \sim \mathcal{D}} \left[\phi_i \log p_\theta(x_i) \right]$$

- Training set $\mathcal{D}$, a held-out "validation" set $\mathcal{D}_v$
- Intuition: after training the model θ on the weighted data, the model gets better performance on the validation set

$$\theta' = \operatorname{argmin}_{\theta} - \mathbb{E}_{x_i \sim \mathcal{D}} \left[\phi_i \log p_\theta(x_i) \right]$$

  - θ' is a function of ϕ, i.e., $\theta' = \theta'(\phi)$

$$\phi' = \operatorname{argmin}_{\phi} - \mathbb{E}_{x_i \sim \mathcal{D}_v} \left[\log p_{\theta'(\phi)}(x_i) \right]$$

Ren et al., "Learning to reweight examples for robust deep learning"

Hu et al., "Learning Data Manipulation for Augmentation and Weighting"

Meta-Learning Methods

- Initialization based methods
  - Learning how to initialize the model for the new task

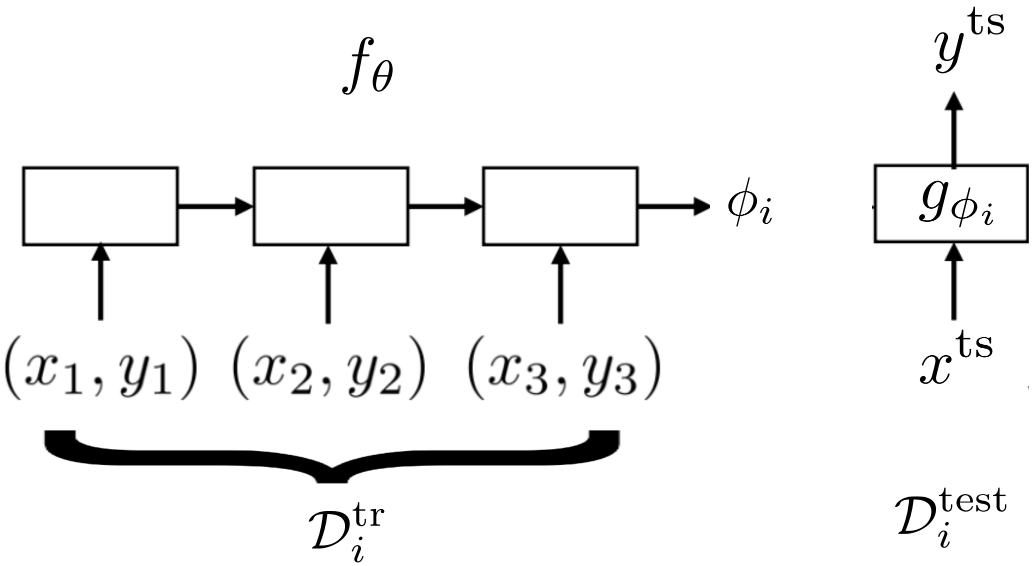
- Black-box adaptation methods

- Non-parametric methods

Black-Box Adaptation

Key idea: Train a neural network to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$ "learner"

Predict test points with $\mathbf{y}^{\text{ts}} = g_{\phi_i}(\mathbf{x}^{\text{ts}})$



Train with standard supervised learning!

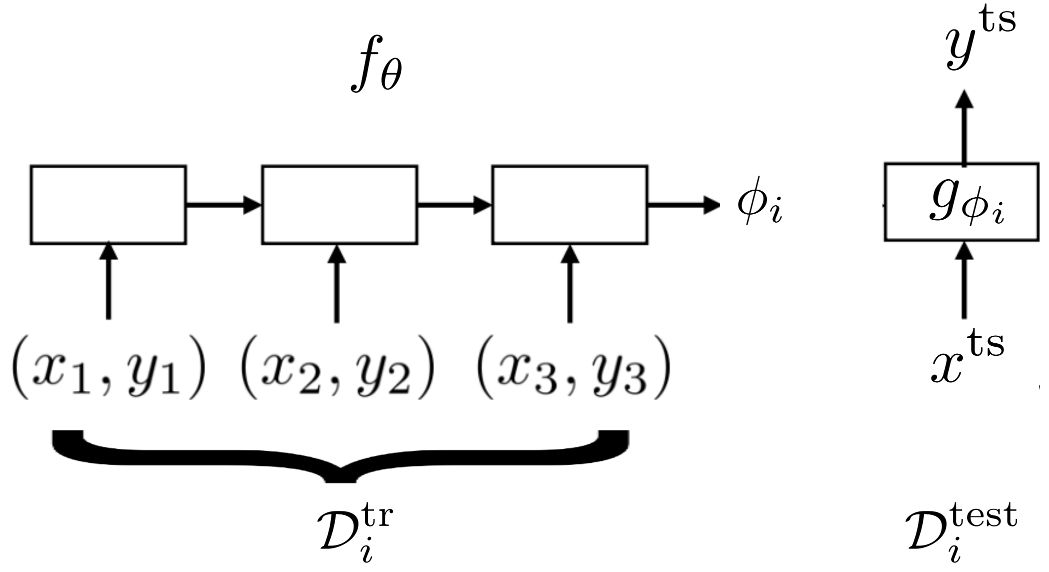
$$\max_\theta \sum_{\mathcal{T}_i} \underbrace{\sum_{(x,y)\sim\mathcal{D}_i^{\text{test}}} \log g_{\phi_i}(y|x)}_{\mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})}$$

$$\max_\theta \sum_{\mathcal{T}_i} \mathcal{L}(f_\theta(\mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{test}})$$

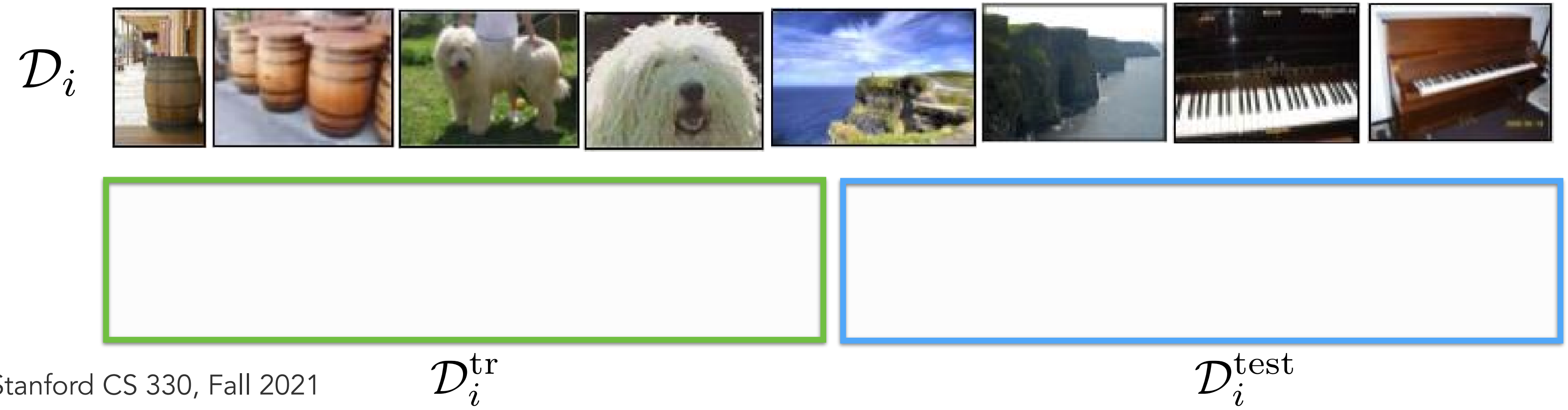
Slide courtesy: Finn, Stanford CS 330, Fall 2021

Black-Box Adaptation

Key idea: Train a neural network to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$.



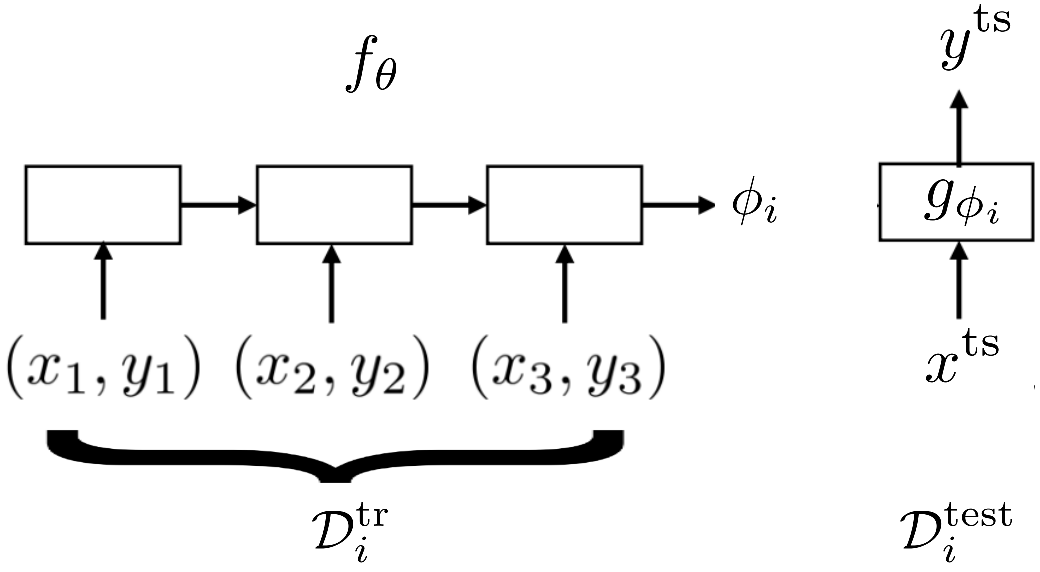
1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$



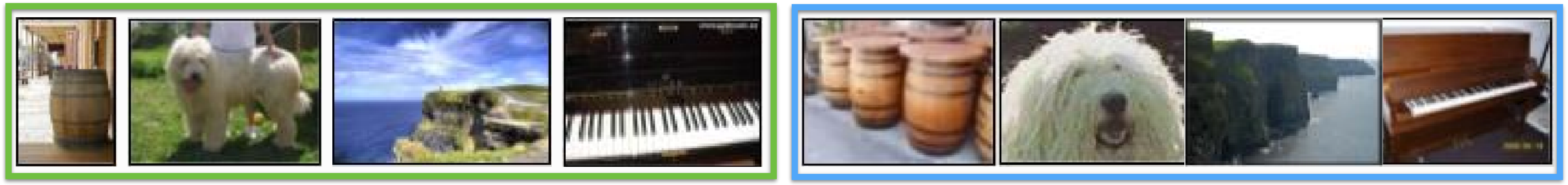
Slide courtesy: Finn, Stanford CS 330, Fall 2021

Black-Box Adaptation

Key idea: Train a neural network to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$.



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 $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$



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

Black-Box Adaptation

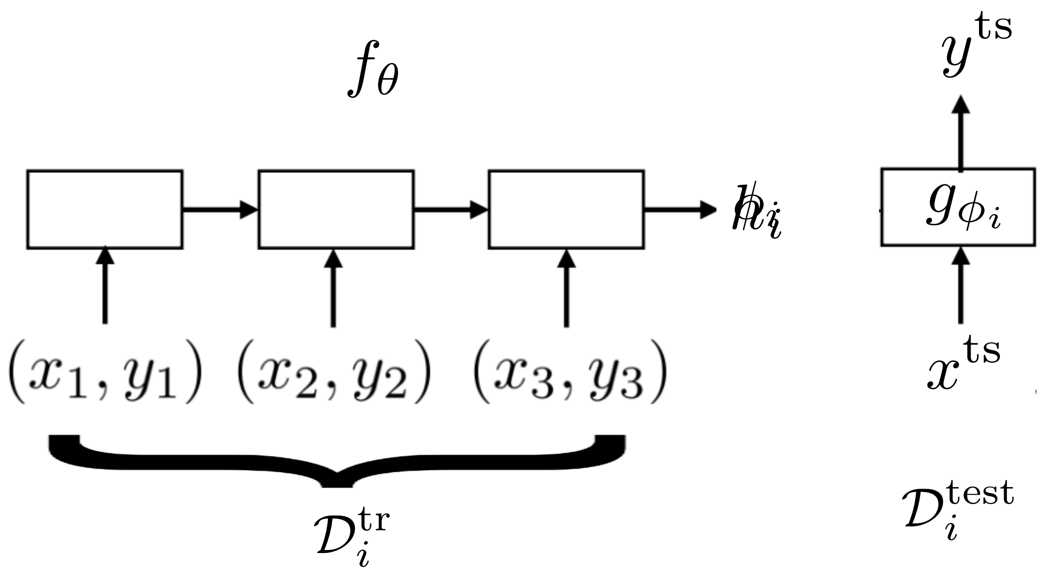
Key idea: Train a neural network to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$.

Challenge

Outputting all neural net parameters does not seem scalable?

Idea: Do not need to output **all** parameters of neural net, only sufficient statistics

(Santoro et al. MANN, Mishra et al. SNAIL)

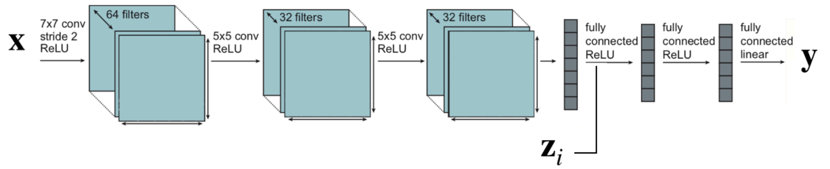


low-dimensional vector h_i

represents contextual task information

$$\phi_i = \{h_i, \theta_g\}$$

recall:



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

What architecture should we use for f_θ?

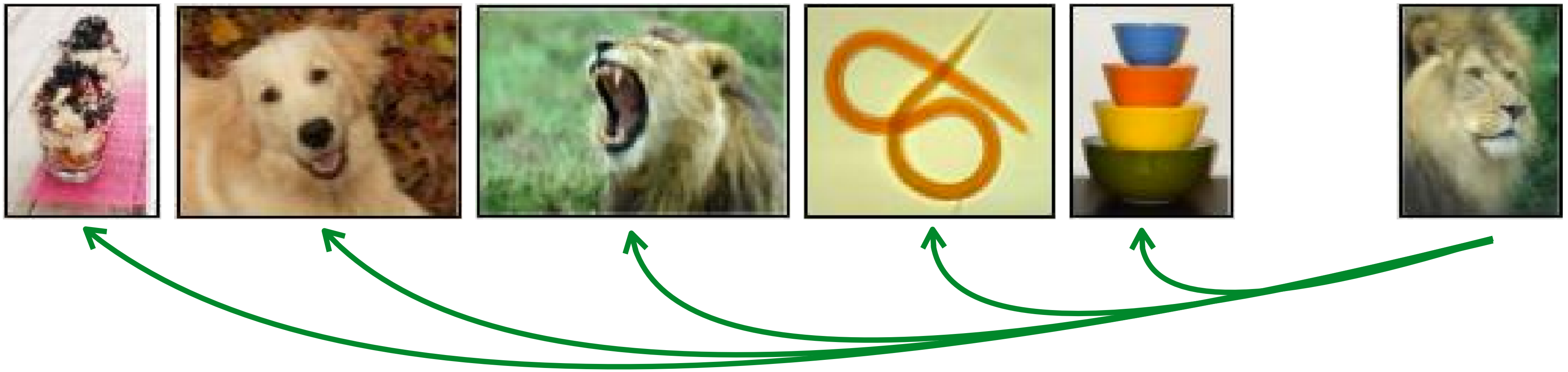
Meta-Learning Methods

- Initialization based methods
  - Learning how to initialize the model for the new task
- Black-box adaptation methods
- Non-parametric methods

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?

ℓ_2 distance in pixel space?

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

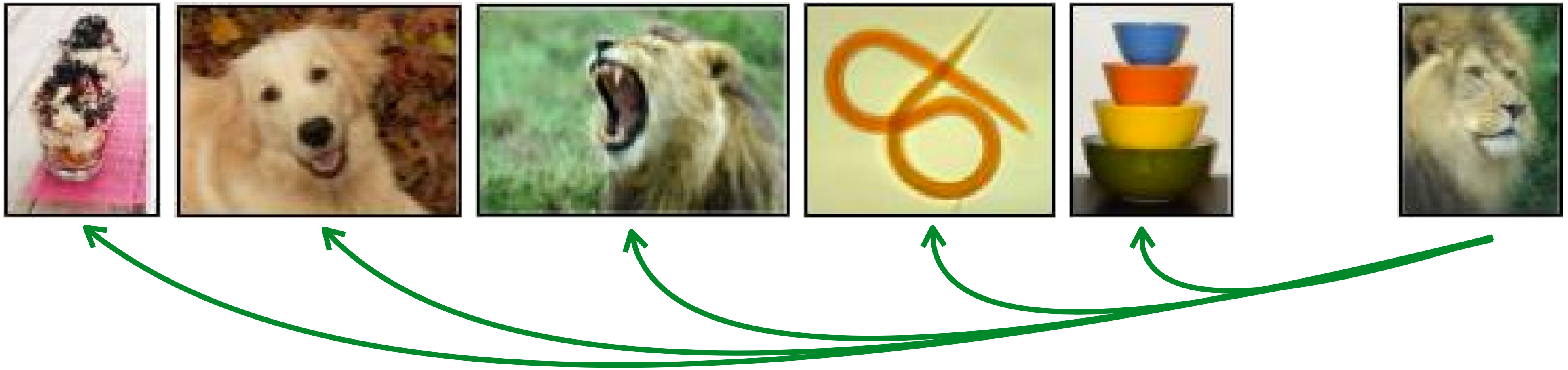
ℓ_2 distance in pixel space?

Zhang et al. (arXiv 1801.03924)

Non-parametric methods

Key Idea: Use non-parametric learner.

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



Compare test image with training images

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

~~ℓ_2 distance in pixel space?~~

Question: What distance metric would you use instead?

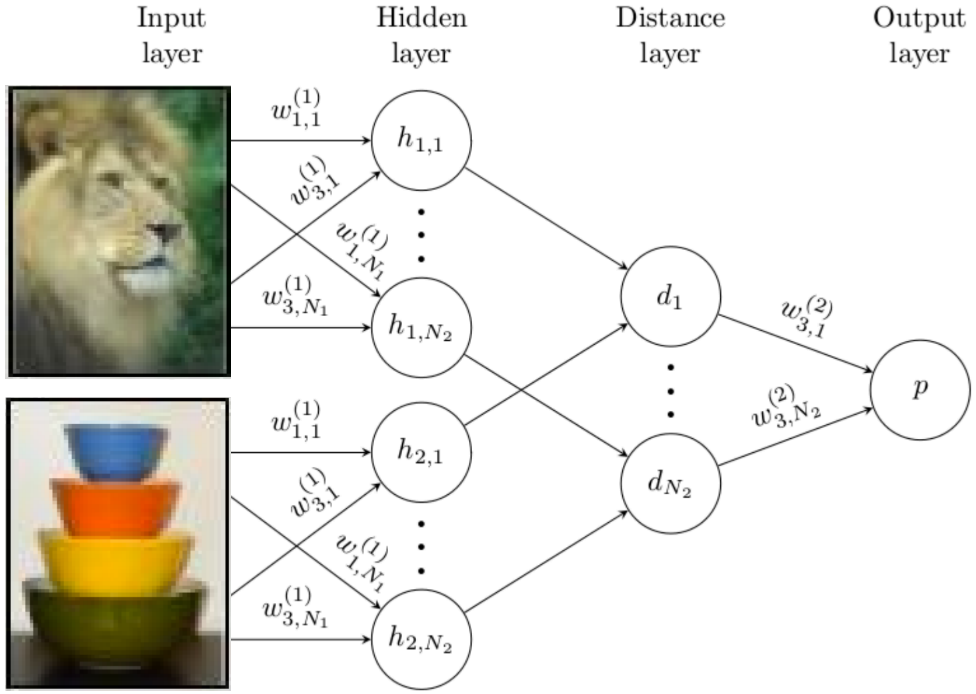
Idea: Learn to compare using meta-training data

Slide courtesy: Finn, Stanford CS 330, Fall 2021

Non-parametric methods

Key Idea: Use non-parametric learner.

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



label

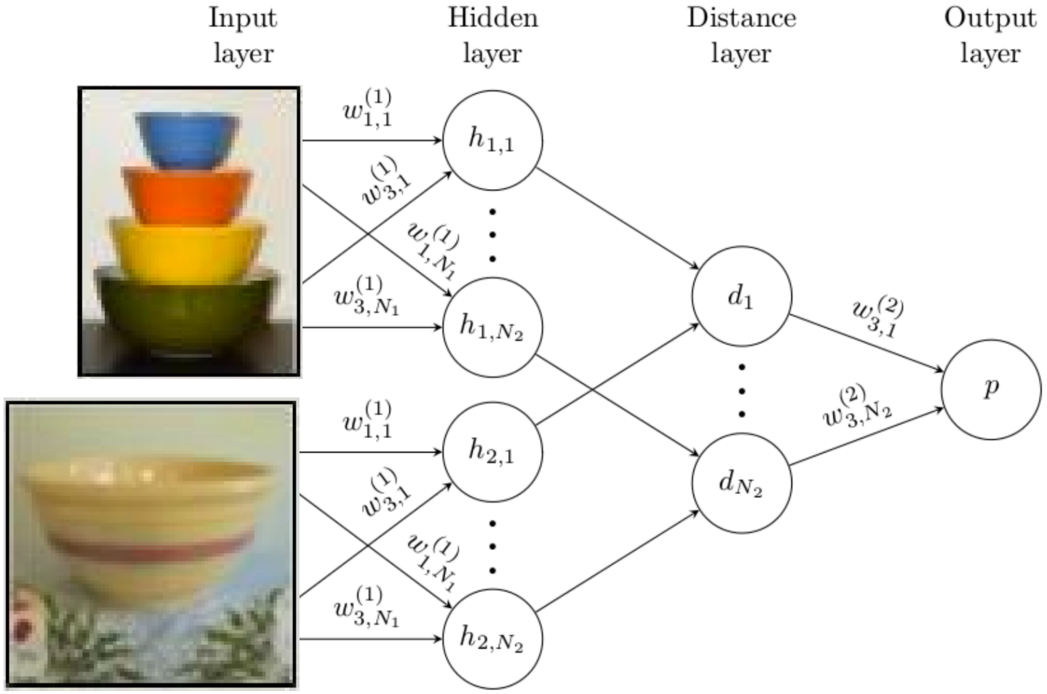
0

Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

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



label

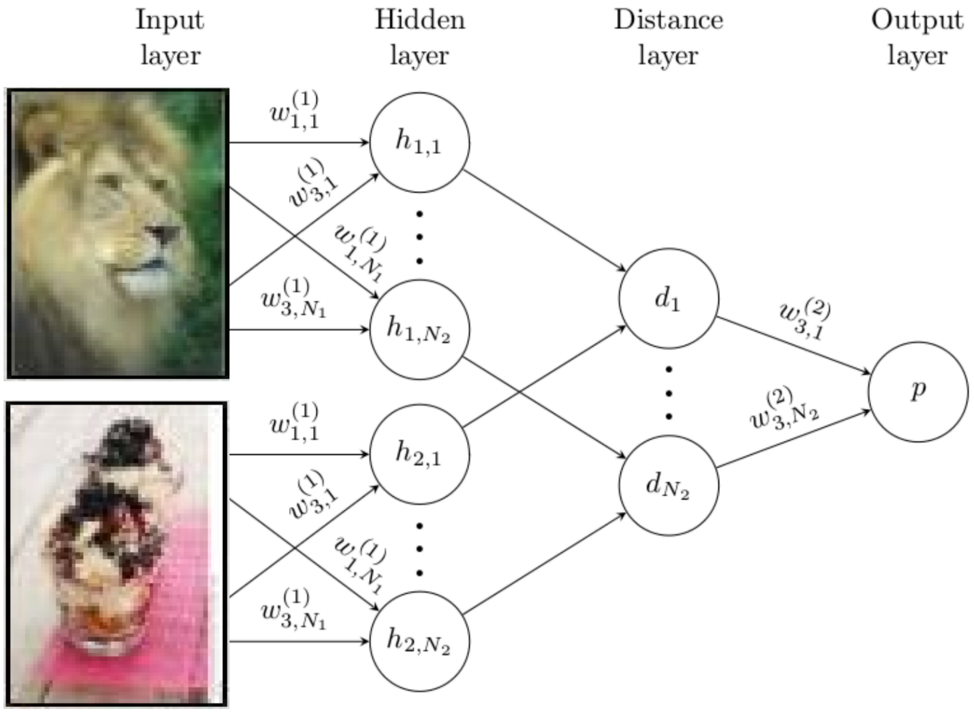
1

Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

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



label

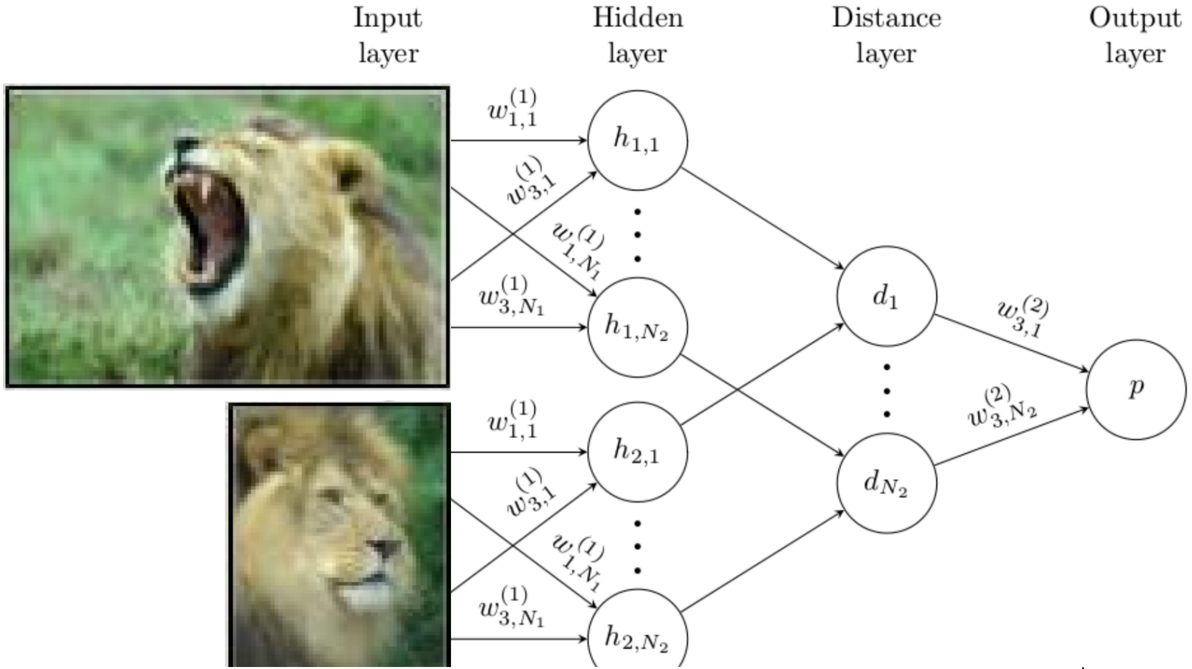
0

Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

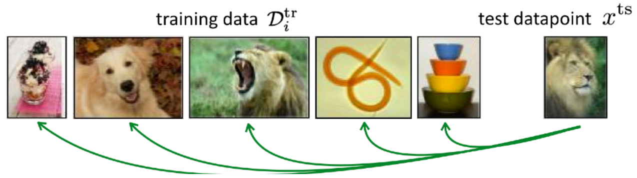
train Siamese network to predict whether or not two images are the same class



label

1

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

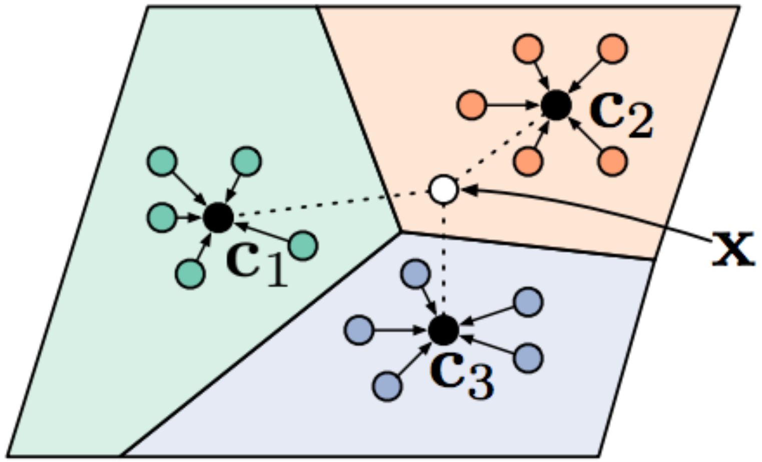


Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

Prototypes: aggregate class information to create a prototypical embedding



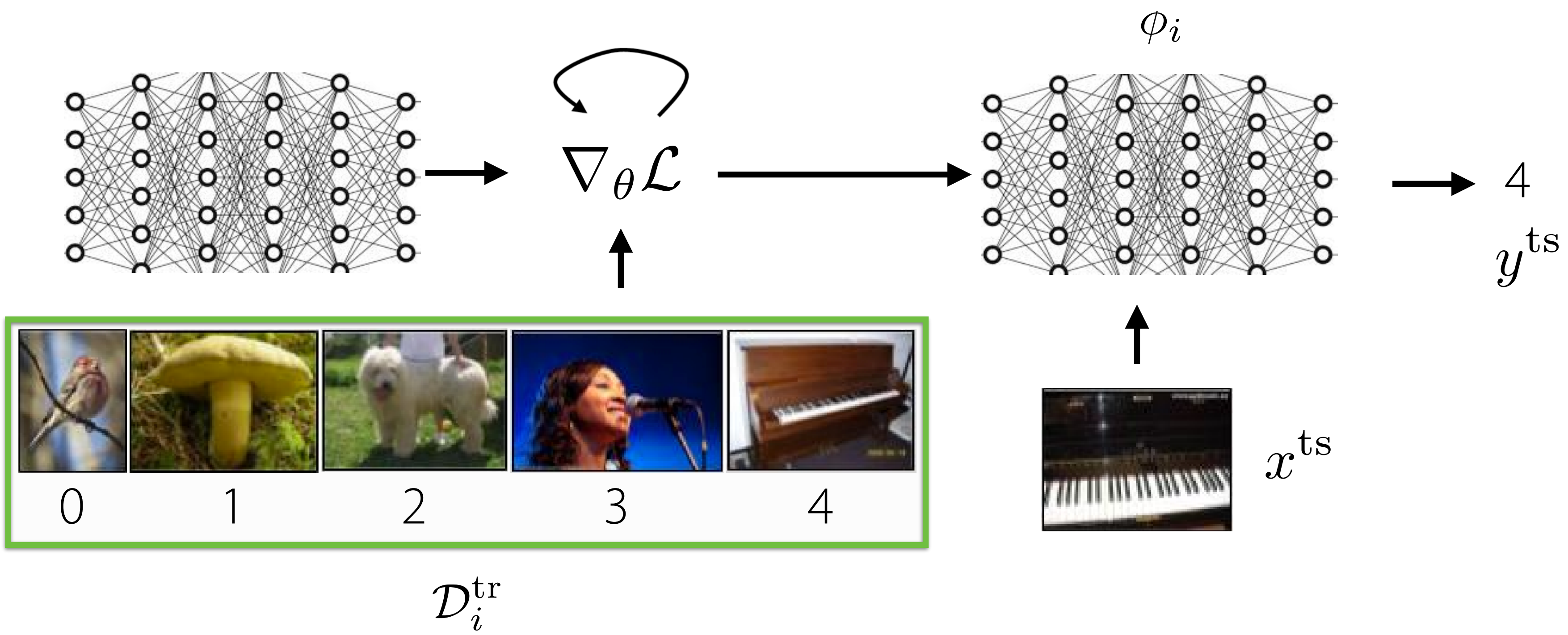
$$\mathbf{c}_n = \frac{1}{K} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} \mathbb{1}(y = n) f_\theta(x)$$

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

d: Euclidean, or cosine distance

Summary: Meta-Learning Methods

- Initialization based methods



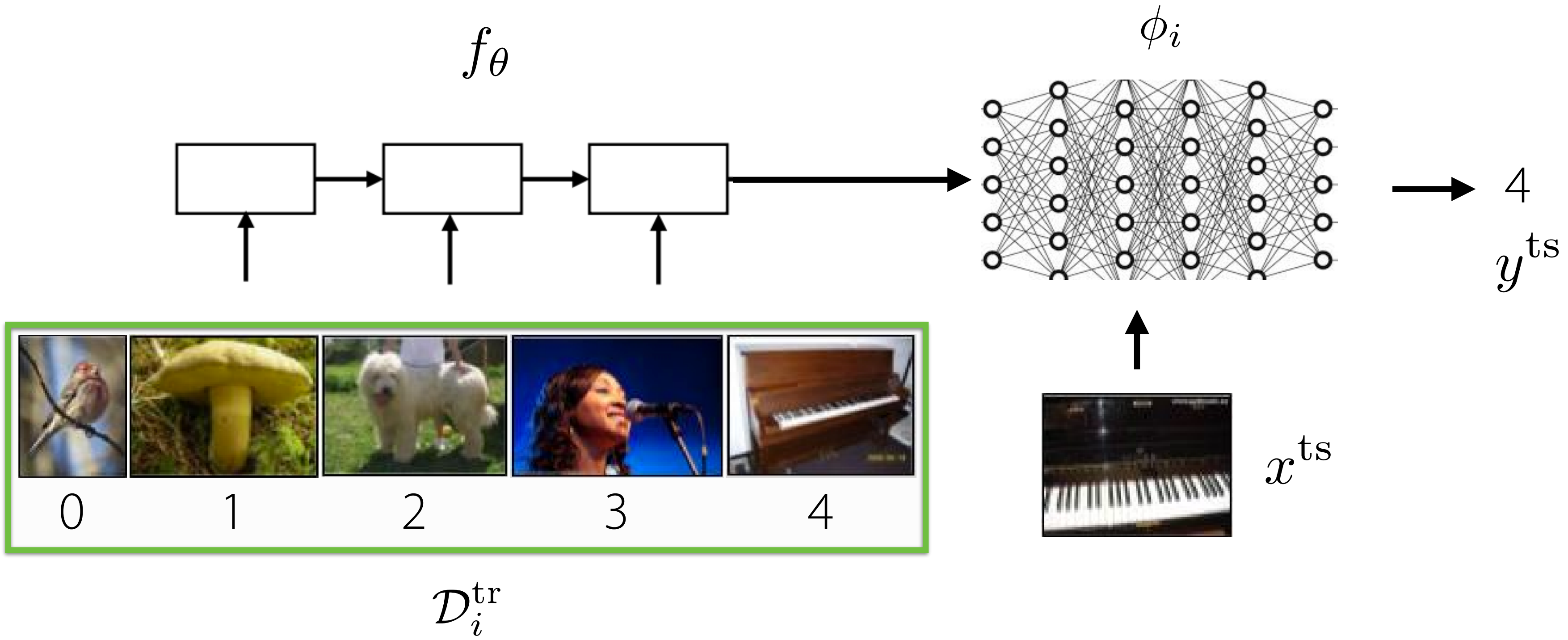
Key idea: embed optimization inside the inner learning process

+ **structure** of **optimization** embedded into meta-learner

- typically requires **second-order optimization**

Summary: Meta-Learning Methods

- Black-box adaptation methods



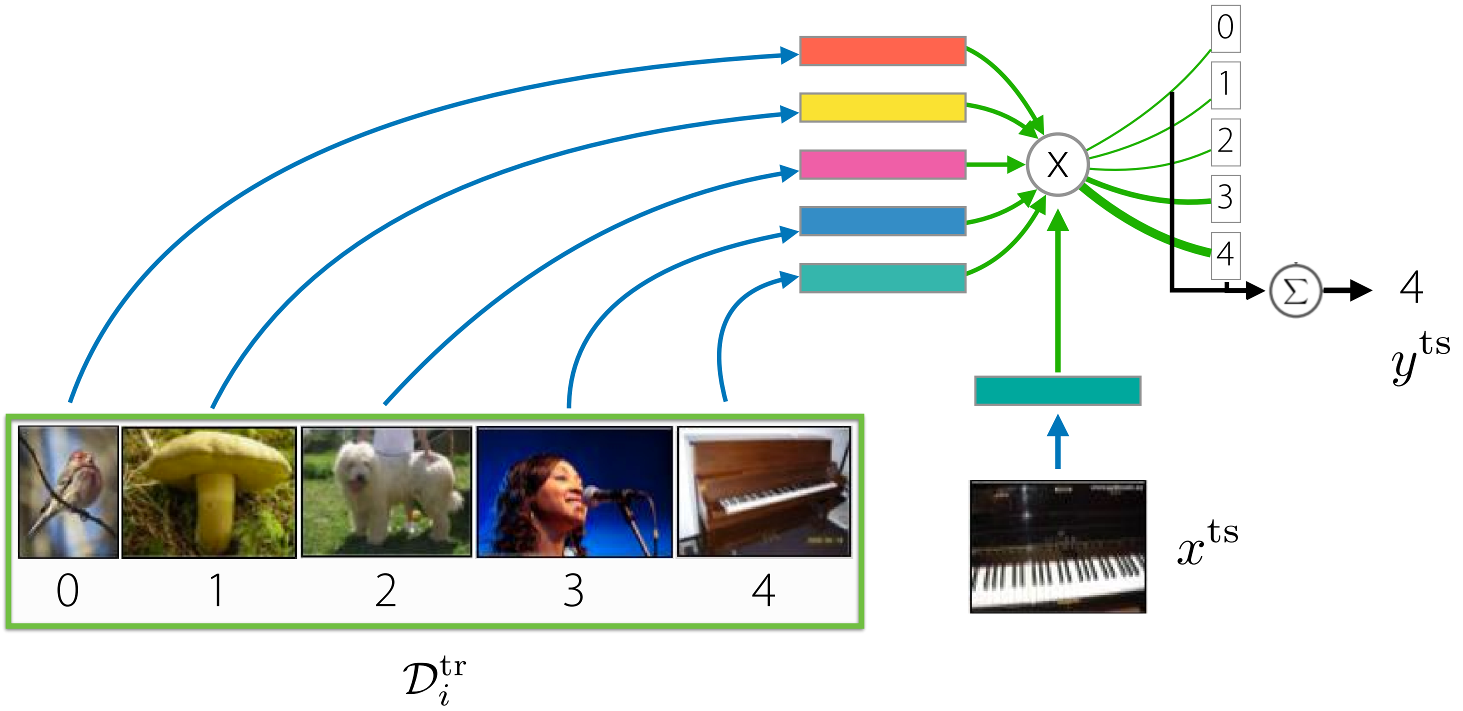
Key idea: parametrize learner as a neural network

+ **expressive**

- **challenging optimization** problem

Summary: Meta-Learning Methods

- Non-parametric methods



Key idea: *non-parametric learner* with *parametric* embedding / distance
(e.g. kNN to examples/prototypes)

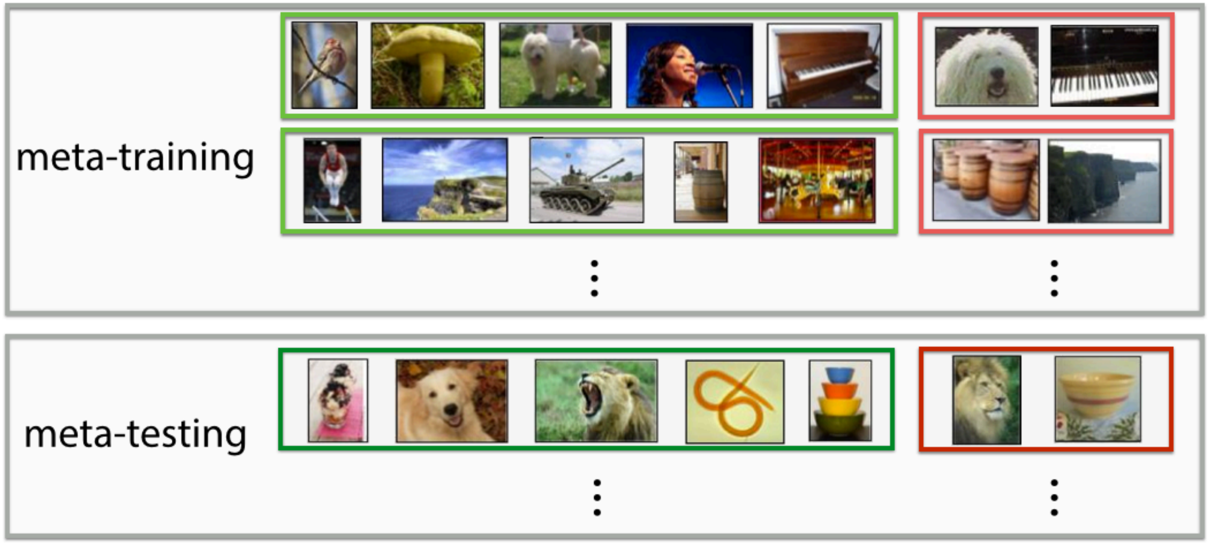
+ easy to optimize, computationally fast

- largely restricted to classification

Slide courtesy: Finn, Stanford CS 330, Fall 2021

Key Takeaways

- Learning with multiple tasks:
  - Multi-task learning, transfer learning, mete-learning
- Meta-learning problem setting
- Meta-learning methods
  - Initialization-based methods
  - Black-box adaptation
  - Non-parametric methods



Questions?