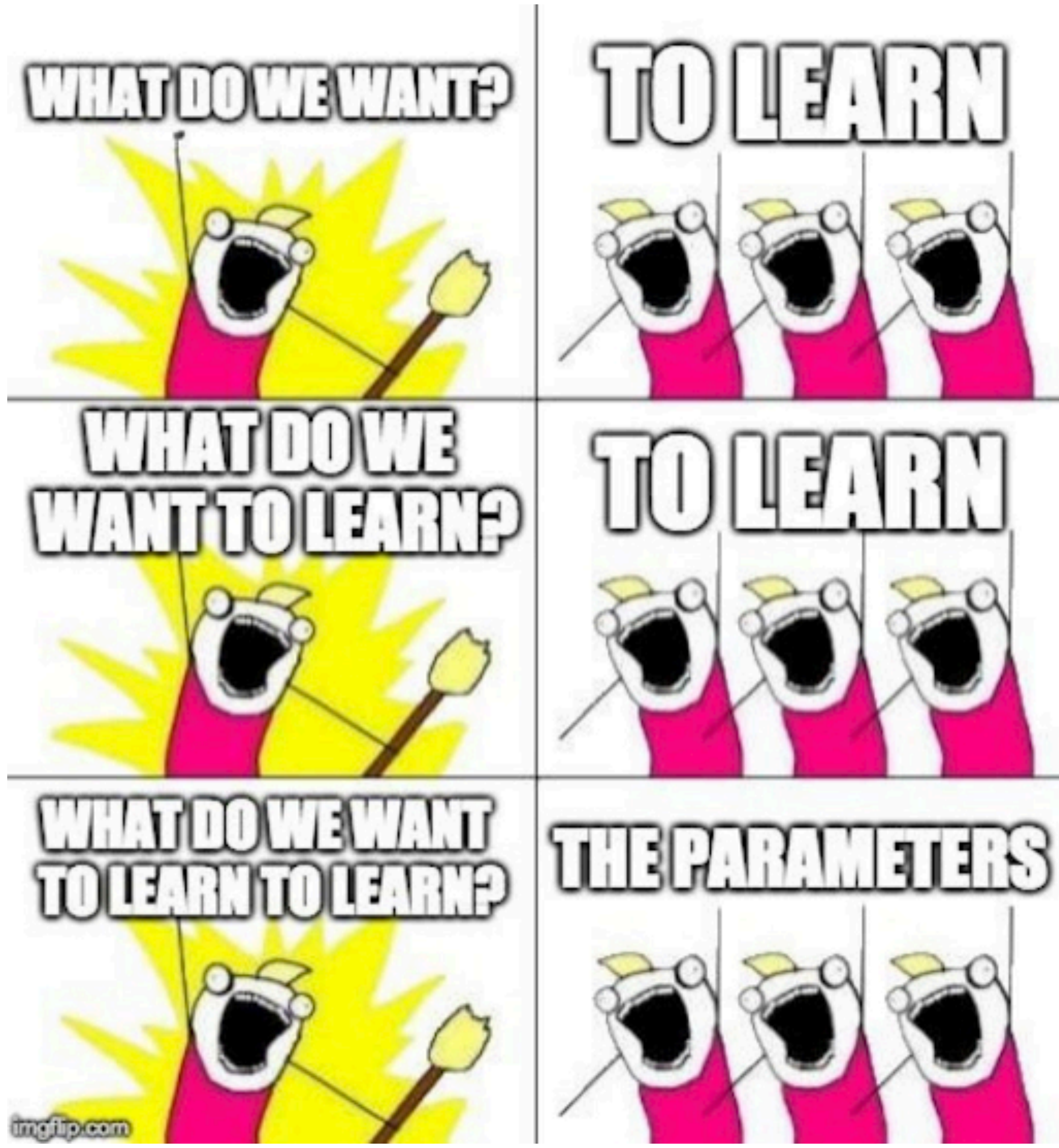
Transfer Learning + Start of Meta-Learning

CS 330

Lecture 3



Credit: Arthur Pesah & Antoine Wehenkel

Logistics

Homework 0 due **Wednesday 10/4** at **11:59 pm PT**.

Homework 1 out on Wednesday.

Project ideas posted on Ed.

Recap from Last Time

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

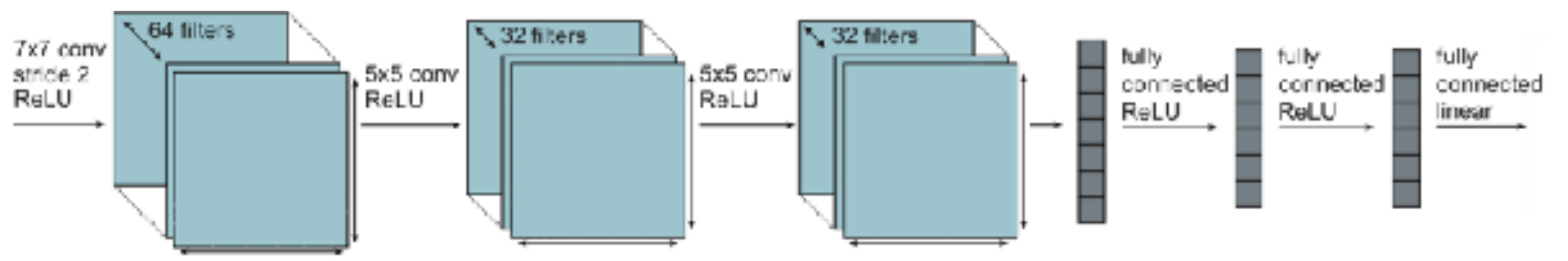
Corresponding datasets: $\mathscr{D}_i^{tr}$ $\mathscr{D}_i^{test}$

Learning a task: $\mathscr{D}_i^{tr} \longrightarrow \theta$

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

- Choice of task weighting w_i affects prioritization of tasks.

Multi-task learning learns neural network conditioned on task descriptor $\mathbf{z}_i$

$\mathbf{x}$  $\mathbf{y}$

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

task descriptor

- Choice of how to condition on $\mathbf{z}_i$ affects how parameters are shared.
  - If you observe negative transfer, share less.
  - If you observe overfitting, try sharing more.

Plan for Today

Transfer Learning

- Problem formulation
- Fine-tuning

Start of Meta-Learning

- Problem formulation
- General recipe of meta-learning algorithms

} Part of Homework 1!

What you'll learn:

- How can you transfer things learned from one task to another?
- What does it mean for two tasks to have *"shared structure"*?
- What is meta-learning?

Multi-Task Learning vs. Transfer Learning

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(s) $\mathscr{T}_a$

by *transferring* knowledge learned from $\mathscr{T}_a$

Common 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 via fine-tuning

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

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

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

(typically for many gradient steps)

Pre-trained Dataset	PASCAL	SUN
ImageNet	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

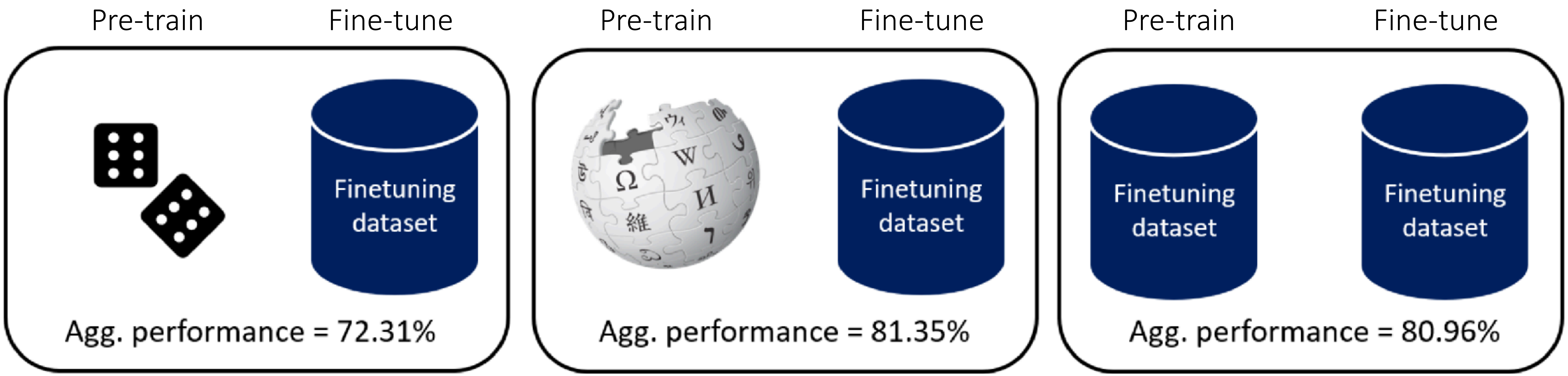
Pre-trained models often ***available online***.

Common design choices

- 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)

When might this common wisdom break?

Unsupervised pre-training objectives may not require diverse data for pre-training.



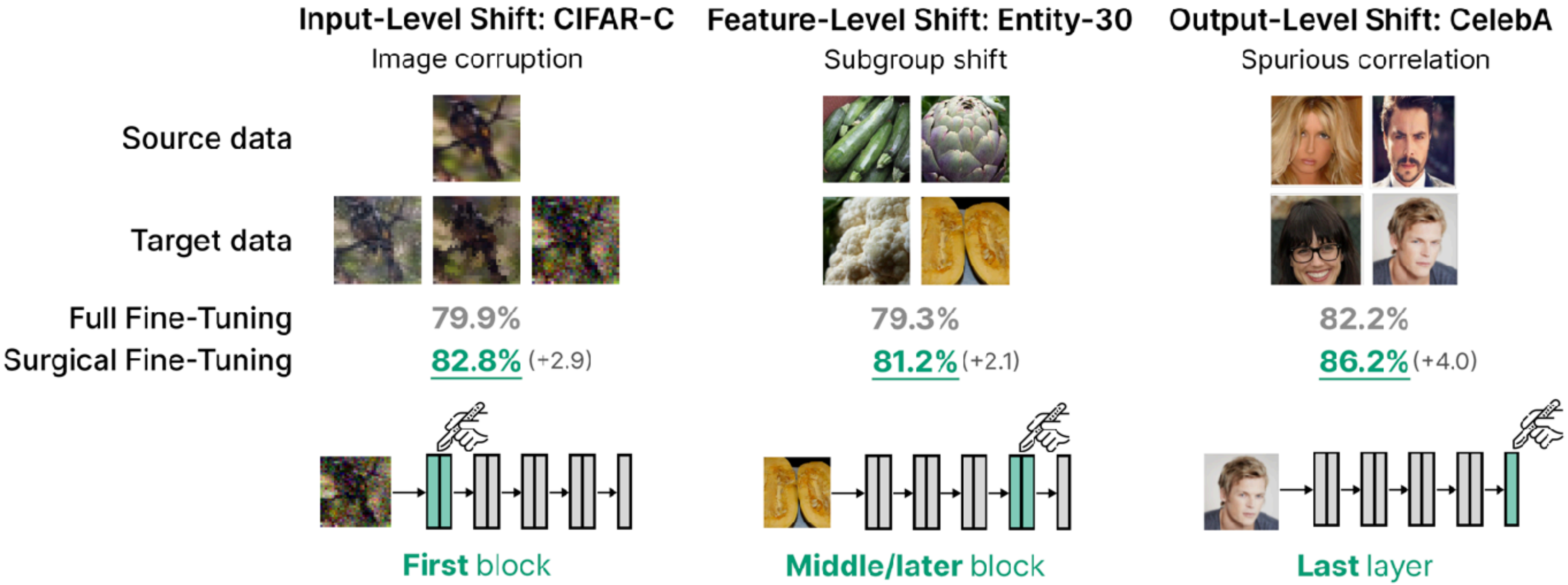
Krishna, Garg, Bingham, Lipton. Downstream Datasets Make Surprisingly Good Pretraining Corpora. ACL 2023.

When might this common wisdom break?

Yoonho's (rough) thought process

1. Fine-tuning **only the last layer** works well.
2. Is there anything special about the last layer?
3. For fine-tuning to low-level image corruptions, maybe the first layer might be better?

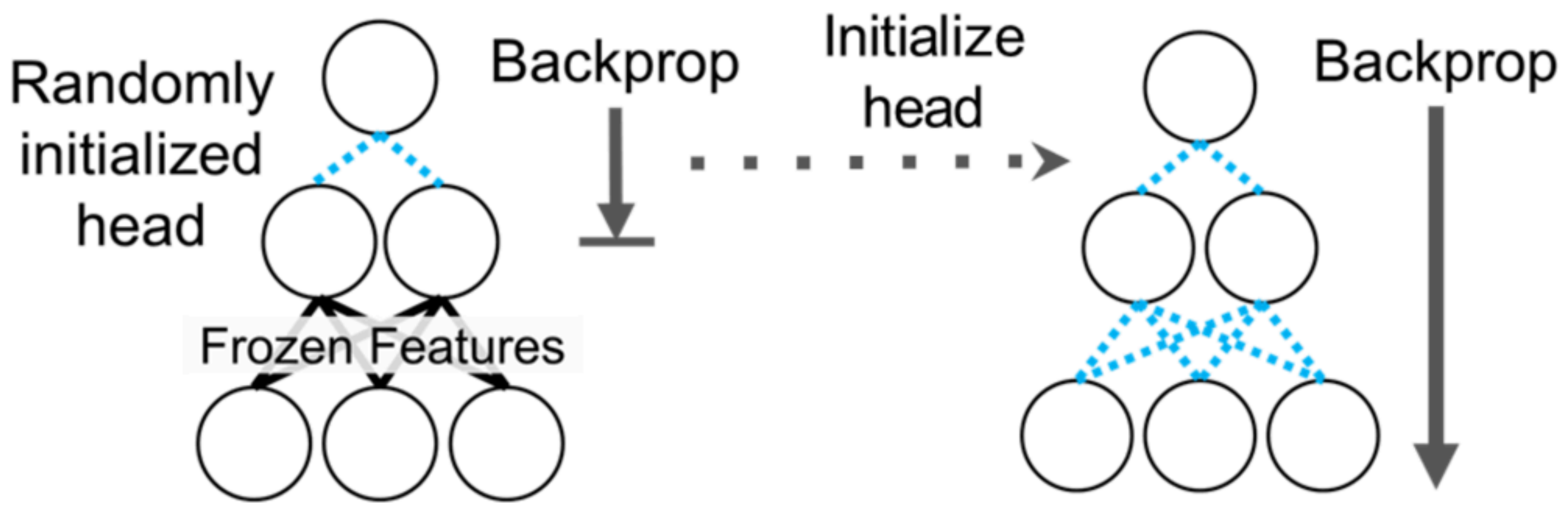
Result: Fine-tuning the first or middle layers can work better than the last layers.



Lee*, Chen*, Tajwar, Kumar, Yao, Liang, Finn. Surgical Fine-Tuning Improves Adaptation to Distribution Shifts. ICLR 2023.

Chelsea's recommended default

Train last layer, then **fine-tune entire network**



Kumar, Raghunathan, Jones, Ma, Liang. Fine-Tuning Can Distort Pre-Trained Features and Underperform Out-of-Distribution. ICLR 2022

How does fine-tuning work with varying target dataset sizes?

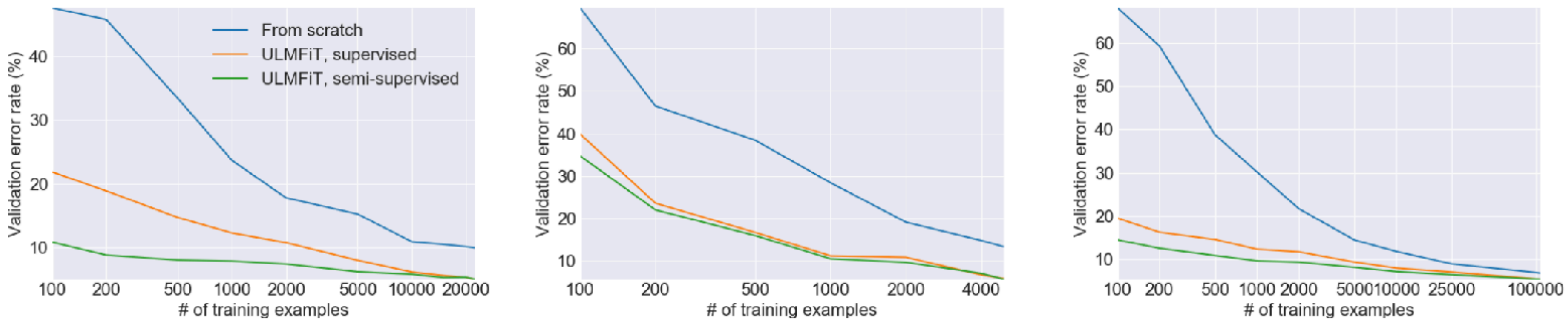


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

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

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

This is where meta-learning can help.

Plan for Today

Transfer Learning

- Problem formulation
- Fine-tuning

Meta-Learning

- Problem formulation
- General recipe of meta-learning algorithms

From Transfer Learning to Meta-Learning

Transfer learning: Initialize model. Hope that it helps the target task.

Meta-learning: Can we explicitly ***optimize*** for transferability?

Given a set of training tasks, can we optimize for the ability to learn these tasks quickly?

so that we can learn ***new*** tasks quickly too

Learning a task: $\mathcal{D}_i^{tr} \;\; \text{—>} \, \theta$

Can we optimize this function?

(for small $\mathcal{D}_i^{tr}$)

Two ways to view meta-learning algorithms

Mechanistic view

- ➢ Deep network 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

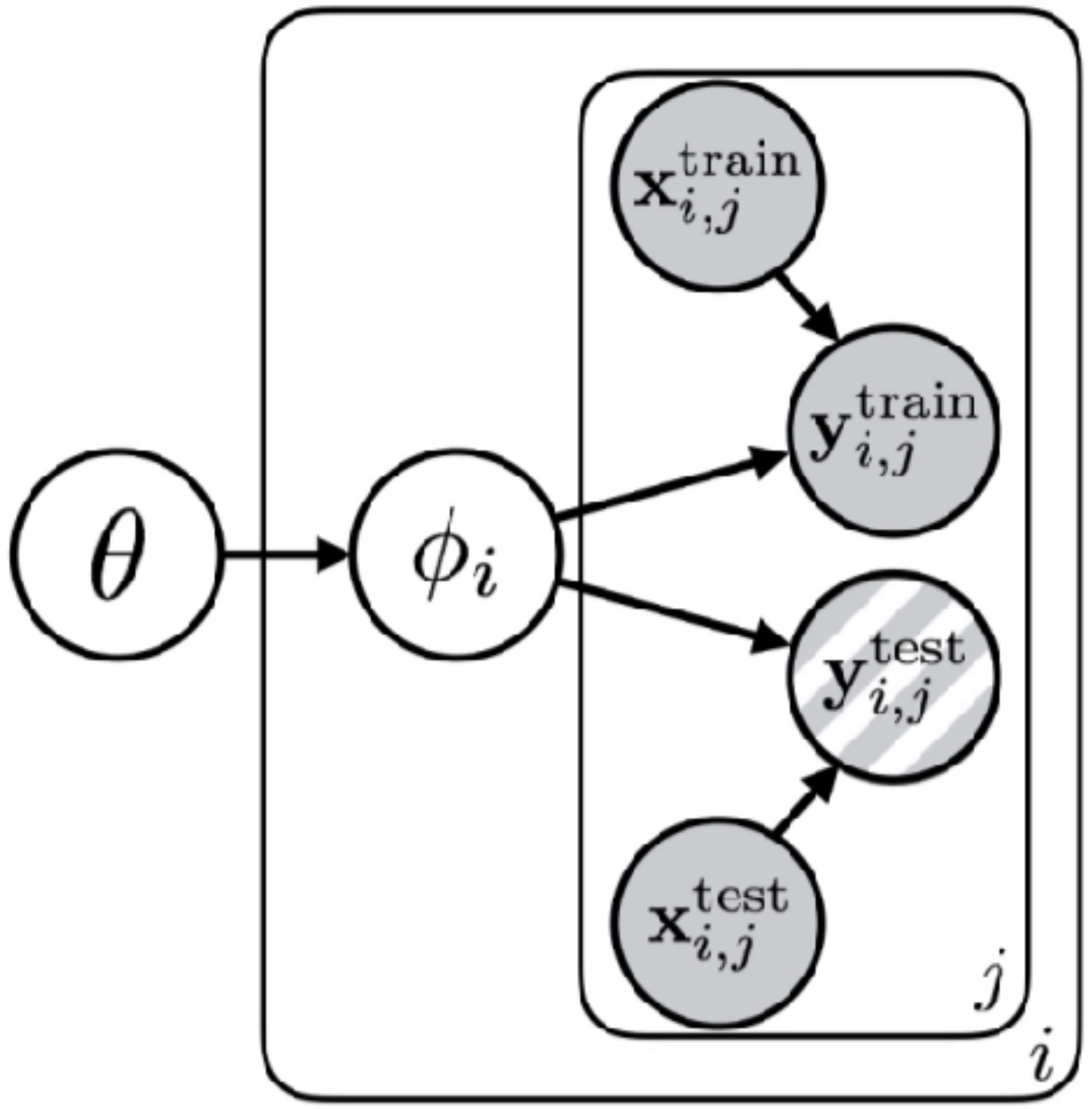
Probabilistic view

- ➢ Extract shared prior knowledge from a set of tasks that allows efficient learning of new tasks
- ➢ Learning a new task uses this prior and (small) training set to infer most likely posterior parameters

How would Bayes view it?

Graphical model for multi-task learning & meta-learning. (whiteboard)

What does "structure" mean? statistical dependence on shared latent information θ



If you condition on that information,

- task parameters become independent
  i.e. $\phi_{i_1} \perp\!\!\!\perp \phi_{i_2} \mid \theta$
  and are not otherwise independent $\phi_{i_1} \not\!\perp\!\!\!\perp \phi_{i_2}$
- hence, you have a lower entropy
  i.e. $\mathscr{H}(p(\phi_i \mid \theta)) < \mathscr{H}(p(\phi_i))$

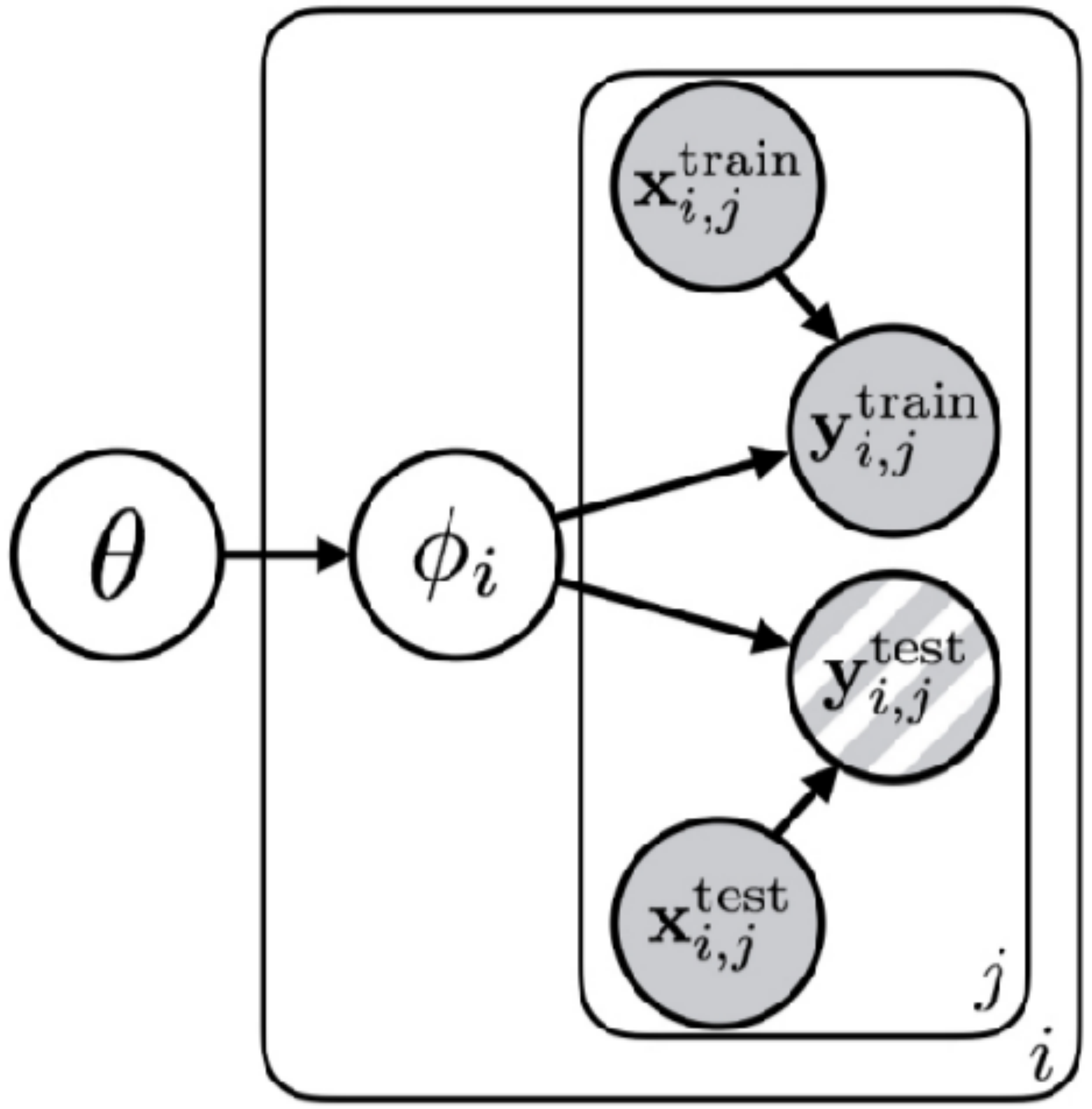
Thought exercise #1: If you can identify θ (i.e. with meta-learning), when should learning ϕ_i be faster than learning from scratch?

Thought exercise #2: what if $\mathscr{H}(p(\phi_i \mid \theta)) = 0 \quad \forall i$?

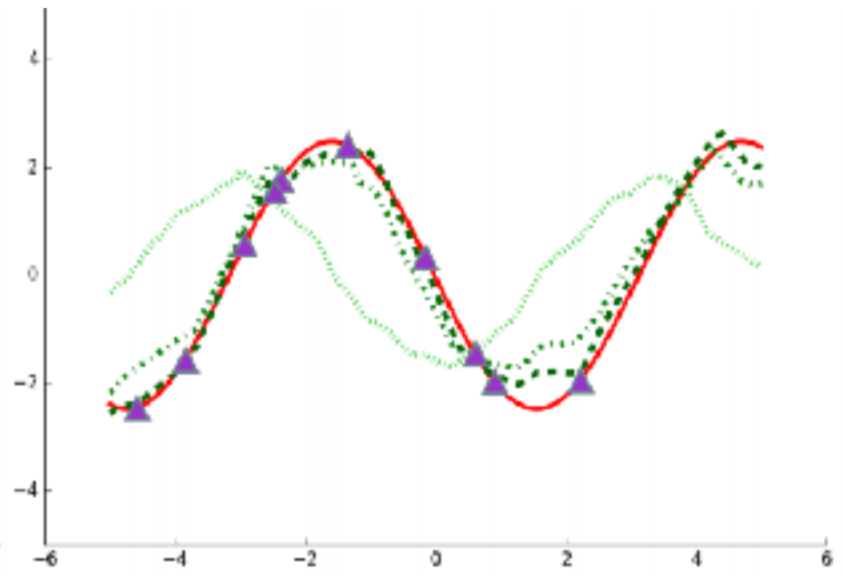
How would Bayes view it?

Graphical model for multi-task learning & meta-learning. (whiteboard)

What does “structure” mean? statistical dependence on shared latent information θ

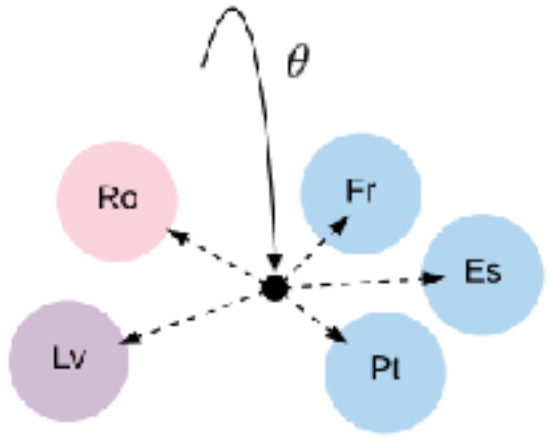


What information might θ contain…



…in a multi-task sinusoid problem?

θ corresponds to family of sinusoid functions (everything but phase and amplitude)



(c) Meta Learning

…in multi-language machine translation?

θ corresponds to the family of all language pairs

Note that θ is narrower than the space of all possible functions.

Two ways to view meta-learning algorithms

Mechanistic view

- ➢ Deep network 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

Probabilistic view

- ➢ Extract shared prior knowledge from a set of tasks that allows efficient learning of new tasks
- ➢ Learning a new task uses this prior and (small) training set to infer most likely posterior parameters

For rest of lecture: Focus primarily on the mechanistic view.

(Bayes will be back later)

How does meta-learning work? An example.

Given 1 example of 5 classes:

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

Classify new examples

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

How does meta-learning work? An example.

training classes

Given 1 example of 5 classes:

Classify new examples

meta-testing $\mathcal{T}_{\text{test}}$

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

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

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

Meta-Learning Problem

Transfer Learning with Many Source Tasks

Given data from $\mathscr{T}_1, \ldots, \mathscr{T}_n$, solve new task $\mathscr{T}_{\text{test}}$ more quickly / proficiently / stably

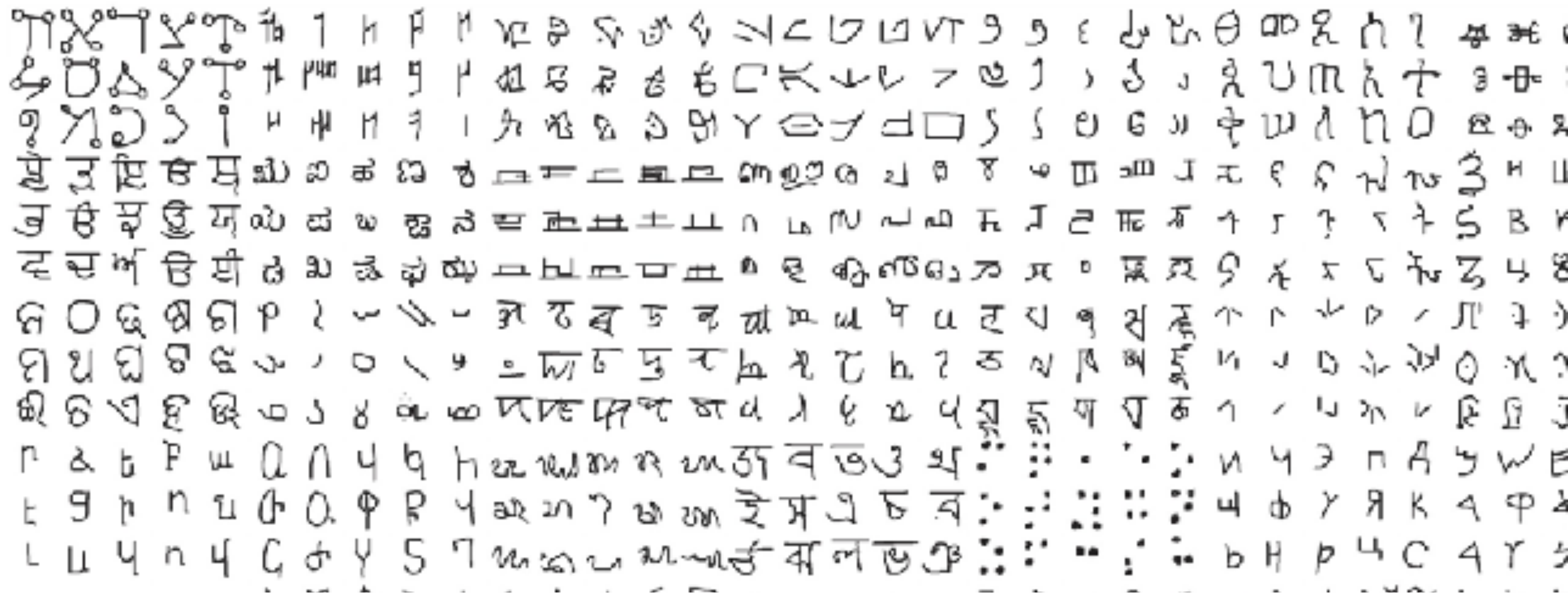
Key assumption: meta-training tasks and meta-test task drawn i.i.d. from same task distribution

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

Like before, tasks must share structure.

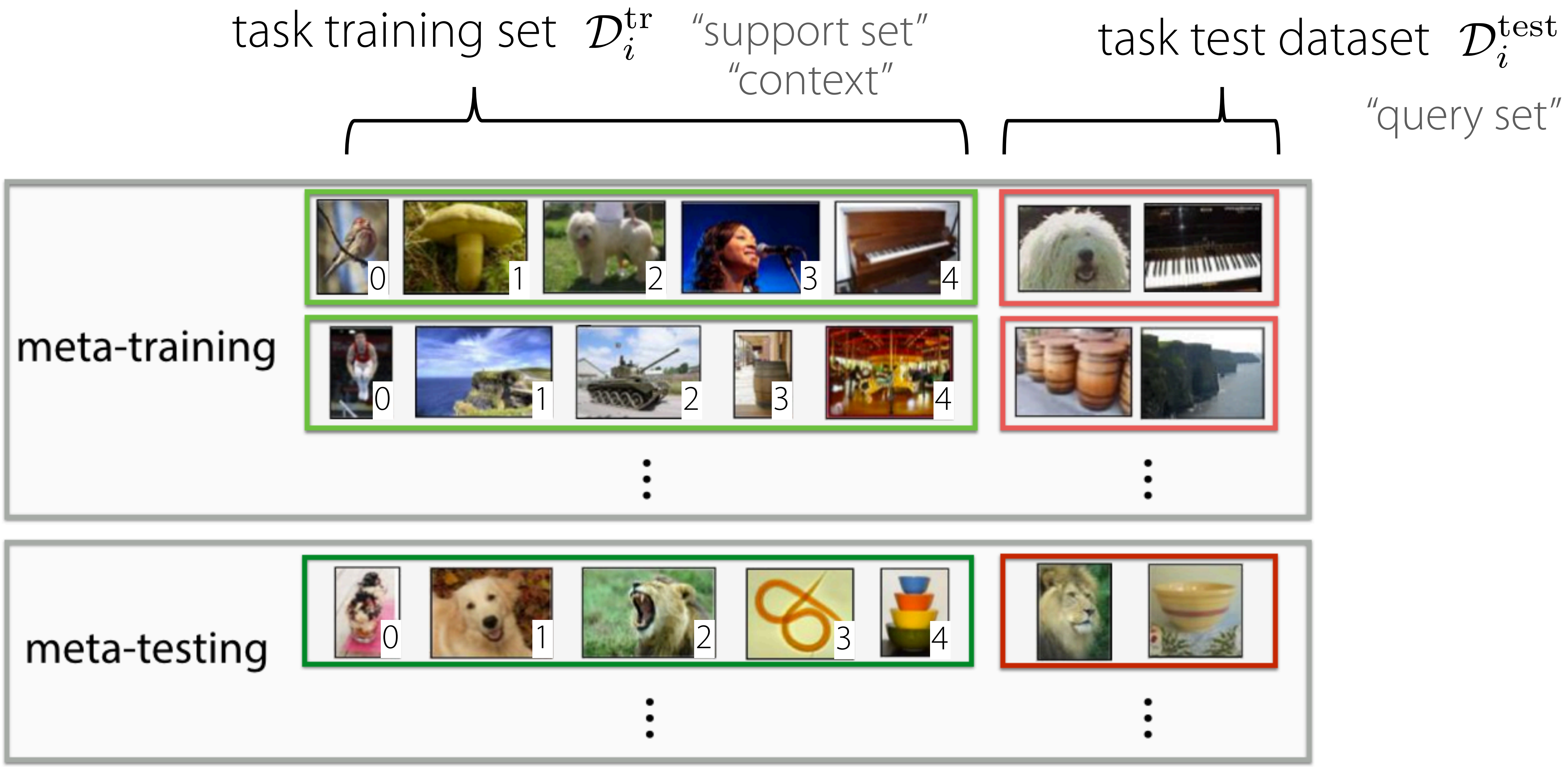
What do the tasks correspond to?

- recognizing handwritten digits from different languages (see homework 1!)
- 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)

Some terminology



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

N-way classification: choosing between N classes

Question: What are k and N for the above example?

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(s) $\mathcal{T}_a$

by *transferring* knowledge learned from $\mathcal{T}_a$

Meta-Learning Problem

Transfer Learning with Many Source Tasks

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

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

In all settings: tasks must share structure.

Plan for Today

Transfer Learning
- Problem formulation
- Fine-tuning

Meta-Learning
- Problem formulation
- **General recipe of meta-learning algorithms**

One View on the Meta-Learning Problem

Supervised Learning:

Inputs: $\mathbf{x}$ Outputs: $\mathbf{y}$ Data: $\{(\mathbf{x}, \mathbf{y})_i\}$

$$\mathbf{y} = g_\phi(\mathbf{x})$$

Meta Supervised Learning:

Inputs: $\underbrace{\mathcal{D}^{\mathrm{tr}}}_{\{(\mathbf{x}, \mathbf{y})_{1:K}\}}$ $\mathbf{x}^{\mathrm{ts}}$ Outputs: $\mathbf{y}^{\mathrm{ts}}$ Data: $\{\mathcal{D}_i\}$

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

$$\mathcal{D}_i : \{(\mathbf{x}, \mathbf{y})_j\}$$

Why is this view useful?

Reduces the meta-learning problem to the design & optimization of f.

Finn. *Learning to Learn with Gradients*. PhD Thesis. 2018

General recipe

How to design a meta-learning algorithm

1. Choose a form of $f_\theta(\mathcal{D}^{\mathrm{tr}}, \mathbf{x}^{\mathrm{ts}})$

2. Choose how to optimize θ w.r.t. max-likelihood objective using meta-training data

meta-parameters

Lecture Recap

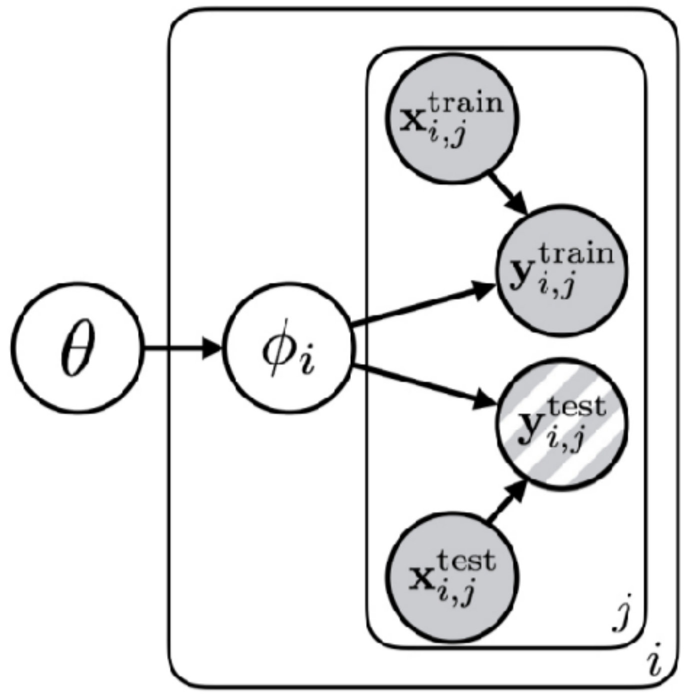
How can you transfer things learned from one task to another?

Fine-tuning: initialize on source task(s) then optimize on target task

Being careful to not destroy initialized features (e.g. smaller learning rate, train last layer first)

What does it mean for tasks to have *"shared structure"*?

Statistical dependence on shared latent information θ



Meta-learning aims to learn shared structure, use it to learn new tasks quickly.

Plan for Today

Transfer Learning
- Problem formulation
- Fine-tuning

Meta-Learning
- Problem formulation
- General recipe of meta-learning algorithms

Part of Homework 1!

What you'll learn:
- How can you transfer things learned from one task to another?
- What does it mean for two tasks to have *"shared structure"*?
- What is meta-learning?

Roadmap

Next five lectures on core methods

Meta-learning methods (3 lectures) (homework 1 & 2)

Unsupervised pre-training methods (2 lectures) (homework 3)

Homework 0 due **Wednesday 10/4** at **11:59 pm PT**.

Homework 1 out on Wednesday.

Project ideas posted on Ed.