

# Transfer Learning + Start of Meta-Learning

CS 330

Lecture 3



Credit: Arthur Pesah & Antoine Wehenkel

# Logistics

Homework 0 due **tonight at 11:59 pm.**

Homework 1 posted today,  
due **Wednesday, October 12**

Project resources to be posted today:

- community **project ideas** list
- **example projects** from last year
- form for **finding project groups**

Note on using code-completion tools like Github Autopilot  
*Okay for project, not okay for assignments*

## Weekly feedback form

- Starting this week
- Sent to random subset of class
- Will use to improve course

## One more CA!



Daniel Zeng

## Guest lectures!



Hanie Sedghi



Percy Liang

# Recap from Last Time

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

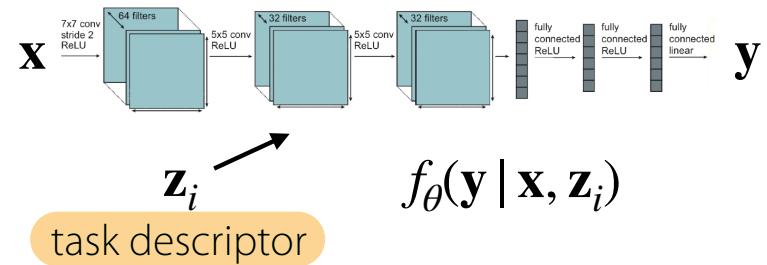
Corresponding datasets:  $\mathcal{D}_i^{tr}$   $\mathcal{D}_i^{test}$

Learning a task:  $\mathcal{D}_i^{tr} \rightarrow \theta$

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

- Choice of task weighting  $w_i$  affects **prioritization of tasks.**

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



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  $\mathcal{T}_1, \dots, \mathcal{T}_T$  at once.

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

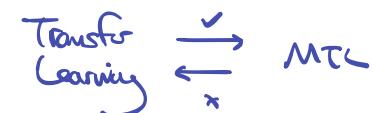
Solve target task  $\mathcal{T}_b$  after solving source task(s)  $\mathcal{T}_a$  by *transferring* knowledge learned from  $\mathcal{T}_a$

\* Common assumption: Cannot access data  $\mathcal{D}_a$  during transfer.

Transfer learning is a valid solution to multi-task learning.

(but not vice versa)

∴ you can't access  $\mathcal{D}_a$



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

- ① when  $\mathcal{D}_a$  is very large  
(don't want to retain & retrain on  $\mathcal{D}_a$ )

- ② when you don't care about solving  
 $\mathcal{T}_a$  &  $\mathcal{T}_b$  simultaneously

# Transfer learning via fine-tuning

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

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

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

(typically for many gradient steps)

Params suited for task  $b$ .

Pre-trained Dataset	PASCAL	SUN	
ImageNet	58.3	52.2	← Pre-trained
Random	41.3 [21]	35.7 [2]	← From scratch

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

Where do you get the pre-trained parameters?

- ImageNet classification → gives v. general-purpose visual features
- Models trained on large language corpora (BERT, LMs)
- Other unsupervised learning techniques → will cover later
- Whatever large, diverse dataset you might have

Pre-trained models often available online. → People have distilled prior knowledge into pre-trained models

Nice practical tips

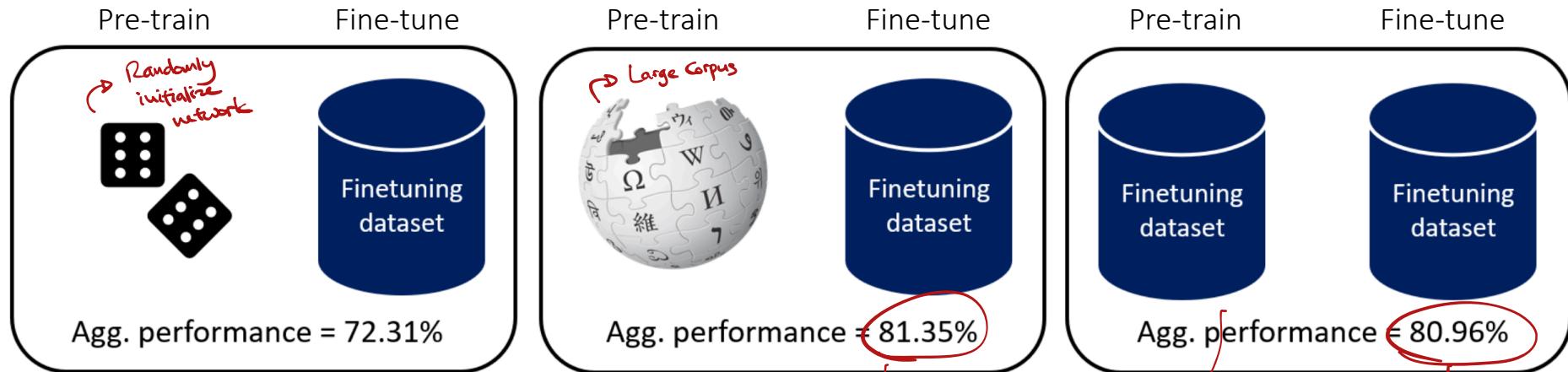
## Common design choices

- 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 knowledge break?



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



Krishna, Garg, Bingham, Lipton. Downstream Datasets Make Surprisingly Good Pretraining Corpora. arXiv 09/28/22.

Breaks the common wisdom

that typically we need a v. diverse pretraining dataset

If you're using unsupervised methods for pre-training, you may not need a v. diverse pre-training dataset

Do see a benefit in pre-training

There's another similar paper that did sth similar in computer vision... so check that out

Pre-train in an Unsupervised way using fin-tuning dataset → supervised pre-tr. won't work

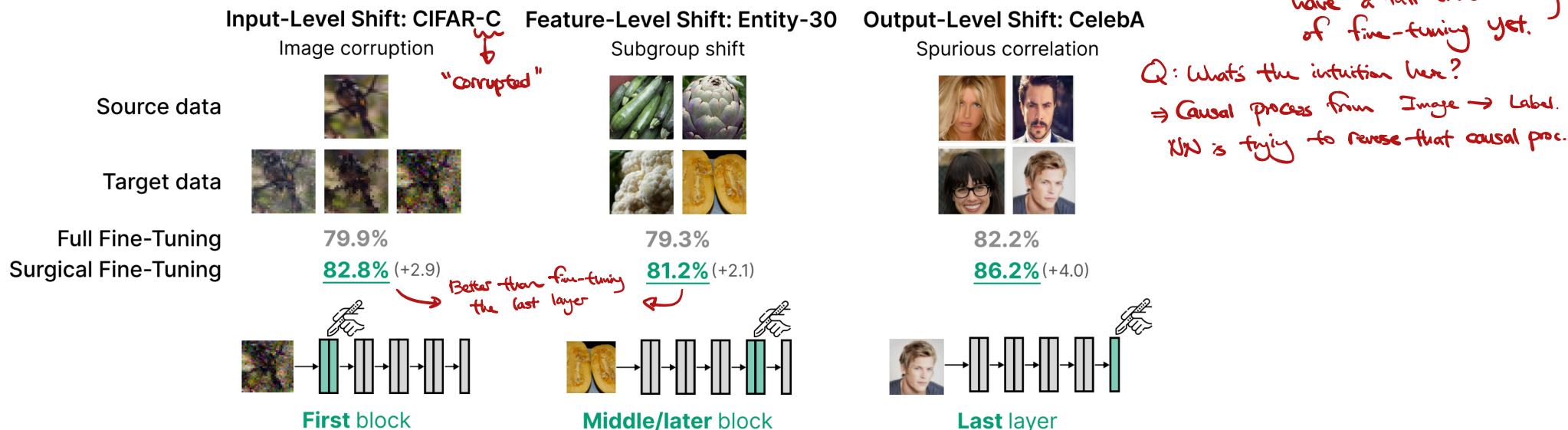
# When might this common knowledge break?



Yoonho's (rough) thought process

1. Fine-tuning only the last layer works well.
2. Is there anything special about the last layer?  
*↪ a lot of times*
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.



→ Suggests we don't have a full understanding of fine-tuning yet.

Q: What's the intuition here?  
⇒ Causal process from Image → Label.  
NN is trying to reverse that causal proc.



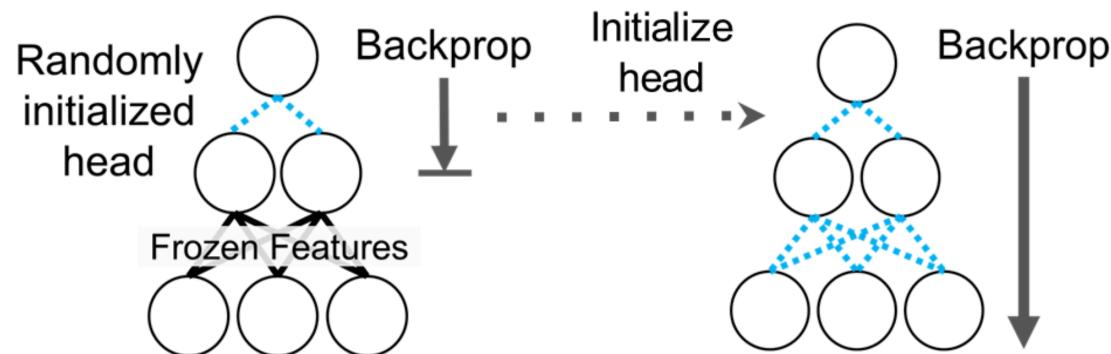
# Chelsea's recommended default

Practical Tip!

Train last layer, then fine-tune entire network

★★★ Generally a good place to start.

↳ This is what I do generally



Q: When do I unfreeze?

A: When training the last layer → typically don't overfit!  
( $\because$  it's a pretty small no. of params.)

↳ Unfreeze when  $\frac{\text{L}_{\text{fin}}}{\text{L}_{\text{tot}}}$  stops going down

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

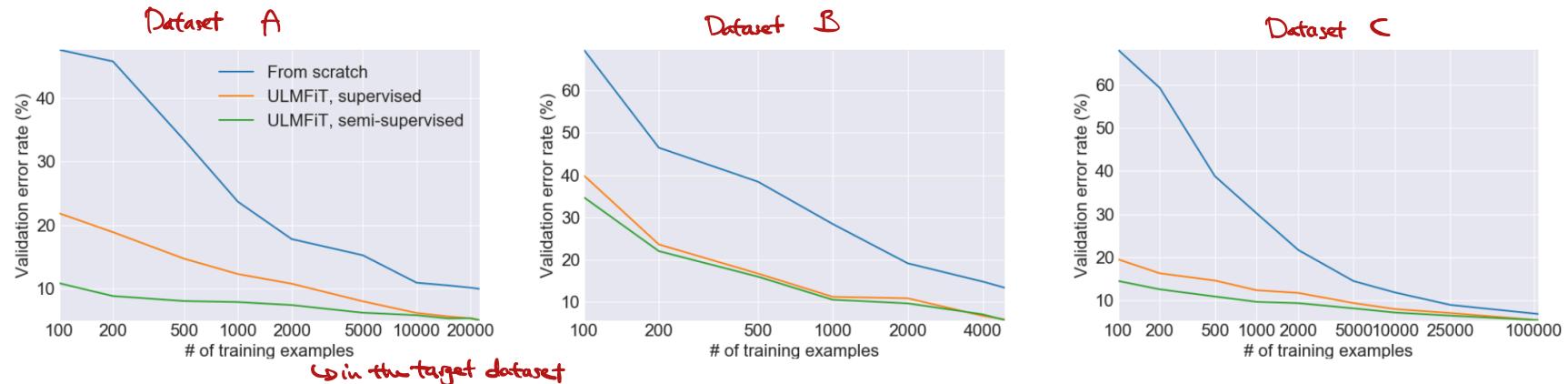


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

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?  *Learning to learn* 

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:  $\mathcal{D}_i^{tr} \rightarrow \theta$



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

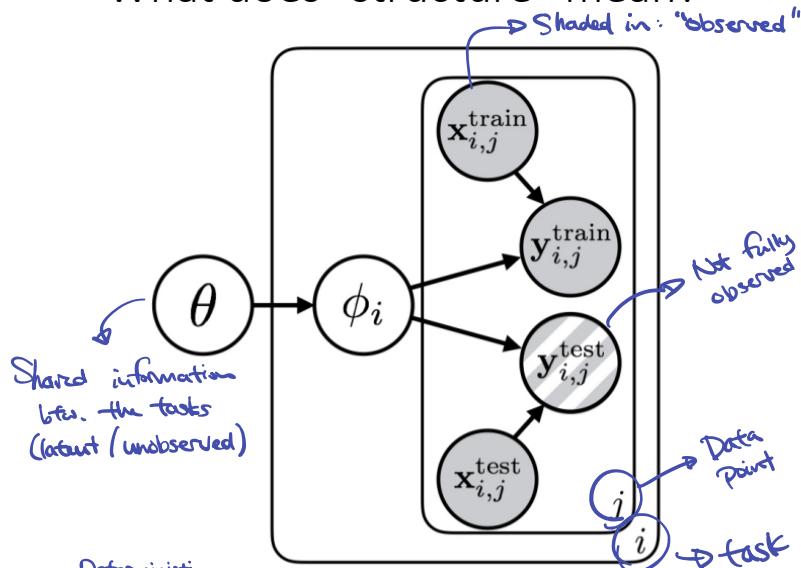
Kind of a  
Bayesian  
approach

# How would Bayes view it?



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

What does “structure” mean?



statistical dependence on shared latent information  $\theta$

unobserved  $\rightarrow$  Not shaded nodes

If you condition on that information,

- task parameters become independent

$$\text{i.e. } \phi_{i_1} \perp\!\!\!\perp \phi_{i_2} \mid \theta$$

and are not otherwise independent  $\phi_{i_1} \perp\!\!\!\perp \phi_{i_2}$

- hence, you have a lower entropy

$$\text{i.e. } \mathcal{H}(p(\phi_i \mid \theta)) < \mathcal{H}(p(\phi_i))$$

$\rightarrow$  model that's more predictable.  
 $\rightarrow$  model is better at representing  
the data it was trained on.

Q: 어떤 특성이 있나요??

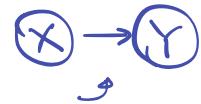
Thought exercise #1: If you can identify  $\theta$  (i.e. with meta-learning), when should learning  $\phi_i$  be faster than learning from scratch?

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

there's only one value  
 $\hookrightarrow$  if we have  $\theta$ , then for  $\phi$

$\rightarrow$  If you have info on  $\theta$ ,  
 $\rightarrow$  lowers the entropy estimate  
of  $\phi$  (you can narrow down  
the value of  $\phi$  more easily)  
 $\rightarrow$  we have fewer bits to  
uncertainty from our training  
data points  
 $\rightarrow$  if you have info on  $\theta$ ,  $\rightarrow$   
 $\rightarrow$  this  $\theta \rightarrow \phi$  dependency exists,  
 $\rightarrow$  you understand that dep.,  
then learning  $\phi$  would be faster  
than if you didn't have that info

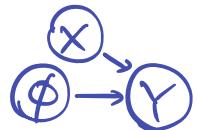
Random Variable  $\rightarrow$  Represented as a Node



Dependencies btw. RVs  $\rightarrow$  Represented as an Arrow

e.g) No dependency :  $P(Y|X) = P(Y)$   $\Rightarrow$    
(No arrow)

e.g)   
: . No path btw.  $A \leftarrow B \rightarrow A \perp B$  (independent)  
·  $A \sim Y$  are dependent

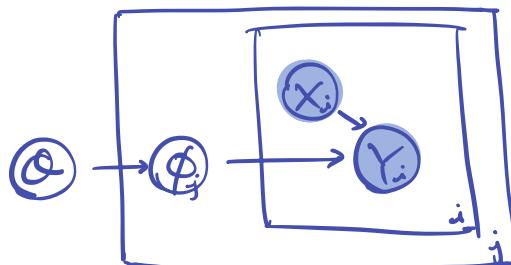


$$f(Y|X; \phi)$$

$\begin{cases} Y: \text{Target} \\ X: \text{Input} \\ \phi: \text{Params} \end{cases}$

The Box

Plate Notation : Makes copies of everything inside the plate, indexed by  $i$  (or  $j$ )



$i$  : index for data points

$j$  : index for tasks

$\phi$  : Shared information between the tasks (only btw tasks)

: Latent / Undeserved Random Variable

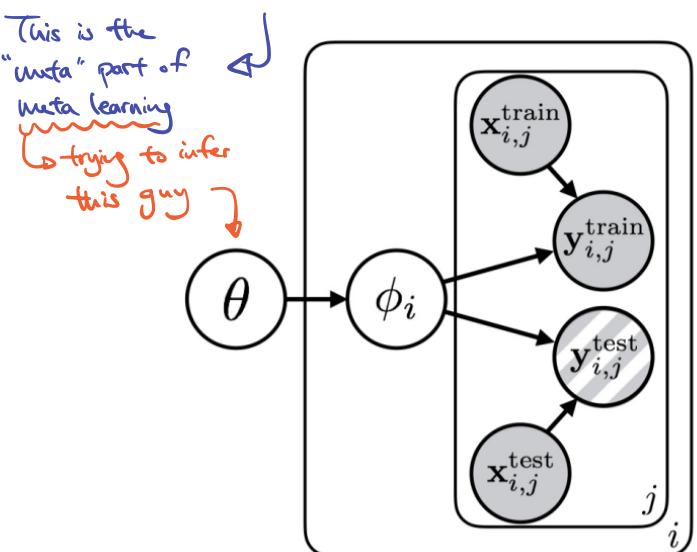
: Observed Random Variable

# How would Bayes view it?



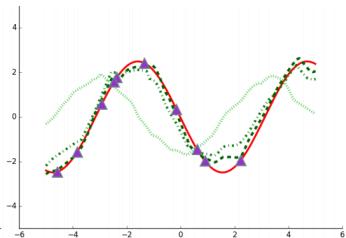
Graphical model for multi-task learning & meta-learning. (whiteboard)

What does “structure” mean?



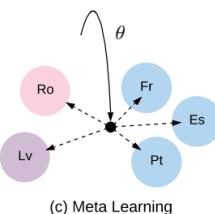
statistical dependence on shared latent information  $\theta$

What information might  $\theta$  contain...



...in a multi-task sinusoid problem?

$\theta$  corresponds to family of sinusoid functions  
(everything but phase and amplitude)



...in multi-language machine translation?

$\theta$  corresponds to the family of all language pairs

Note that  $\theta$  is narrower than the space of all possible functions.

i.e. not universal language, but the family of all language pairs

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

Classify new examples



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

↳ if we're trying to solve this task from scratch, it won't work v. well.

⇒ We want to leverage previous information.

↳ if we have data from other tasks, we  
should be able to use that to help us  
solve this few-shot CLF problem.

# How does meta-learning work? An example.



Learn how to quickly learn each of these tasks, s.t. when we're given new image classes, we can quickly learn a classifier.

meta-testing  $T_{\text{test}}$

Given 1 example of 5 classes:



target task

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

new tasks that the meta-learner has never seen before.

Classify new examples



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

\*\*\*

any ML problem

Can replace image classification with: regression, language generation, skill learning,

# Meta-Learning Problem

Transfer Learning with Many Source Tasks

with fewer learning examples.

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

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

↳ from the same distribution!

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

Like before, tasks must share structure.

What do the tasks correspond to?

- recognizing handwritten digits from different languages (see homework 1!)
- giving feedback to students on different exams
- classifying species in different regions of the world
- a robot performing different tasks



任务1：识别手写数字  
任务2：从不同语言中识别手写数字  
任务3：从不同地区识别物种  
任务4：机器人执行不同任务

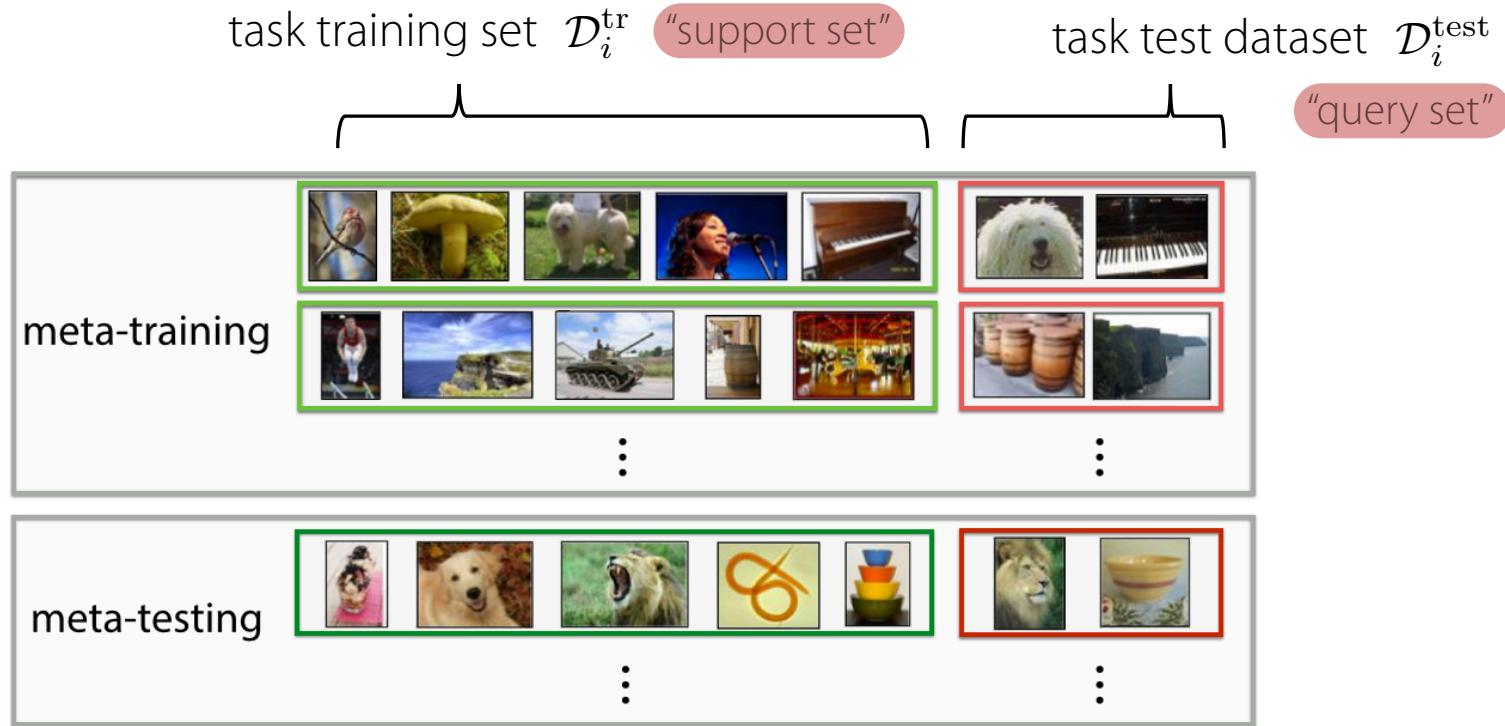
How many tasks do you need?

The more the better.

19

(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?

$$\begin{array}{l} \leftarrow 1 \\ N = 5 \end{array}$$

# Problem Settings Recap

## Multi-Task Learning

Solve multiple tasks  $\mathcal{T}_1, \dots, \mathcal{T}_T$  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  $\mathcal{T}_1, \dots, \mathcal{T}_n$ , 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*

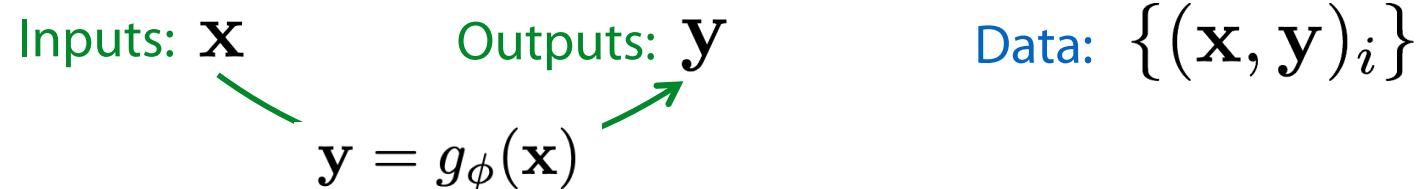
- Problem formulation
- Fine-tuning

## *Meta-Learning*

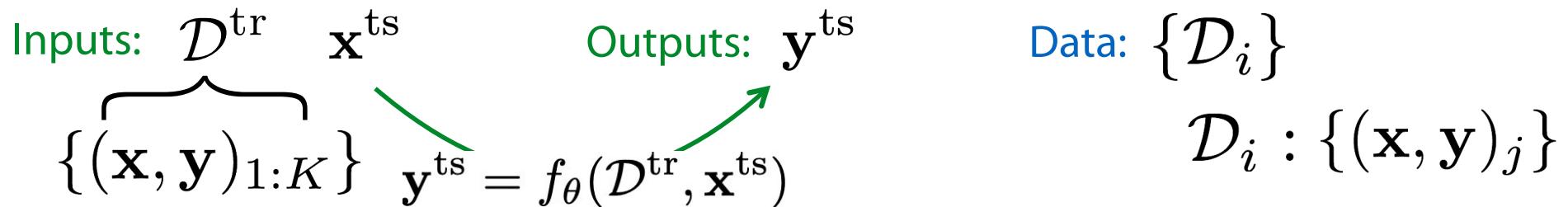
- Problem formulation
- **General recipe of meta-learning algorithms**

# One View on the Meta-Learning Problem

Supervised Learning:



Meta Supervised Learning:



Why is this view useful?

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

# General recipe

## How to design a meta-learning algorithm

1. Choose a form of  $f_{\theta}(\mathcal{D}^{\text{tr}}, \mathbf{x}^{\text{ts}})$
2. Choose how to optimize  $\theta$  w.r.t. max-likelihood objective using meta-training data  
  
meta-parameters

# Lecture Recap

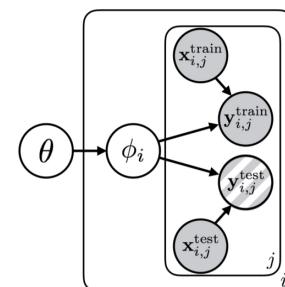
How can you **transfer** things learned from one task to another?

**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*  $\theta$



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

# 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)

## Reminders

Homework 0 due **tonight at 11:59 pm.**

Homework 1 posted today,  
due **Wednesday, October 12**

Project resources to be posted today:

- community **project ideas** list
- **example projects** from last year
- form for **finding project groups**