Deep Multi-Task and Meta Learning

CS 330

What will you learn in this course?

1. The foundations of modern deep learning methods for learning across tasks
2. How to implement and work with practical multi-task & transfer learning systems (in PyTorch)
3. A glimpse into the scientific and engineering process of building and understanding new algorithms

Topics

1. Multi-task learning, transfer learning basics
2. Meta-learning algorithms
   (black-box approaches, optimization-based meta-learning, metric learning)
3. Advanced meta-learning topics
   (meta-overfitting, unsupervised meta-learning, Bayesian models)
4. Unsupervised pre-training for few-shot learning
5. Relation to foundation models & in-context learning
6. Domain adaptation & generalization
7. Lifelong learning
8. Open problems

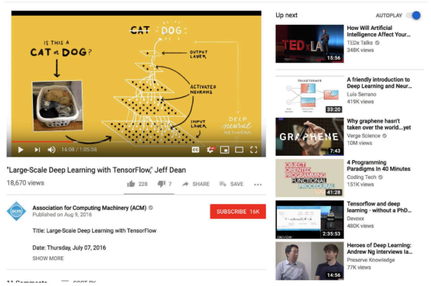
} New!

New: No RL lectures or HWs.

Emphasis on **deep learning** techniques.

Case studies of important & timely applications

- Multi-task learning in recommender systems
- Meta-learning for land cover classification, education
- Few-shot learning in large language models



Zhao et al. Recommending What Video to Watch Next. 2019

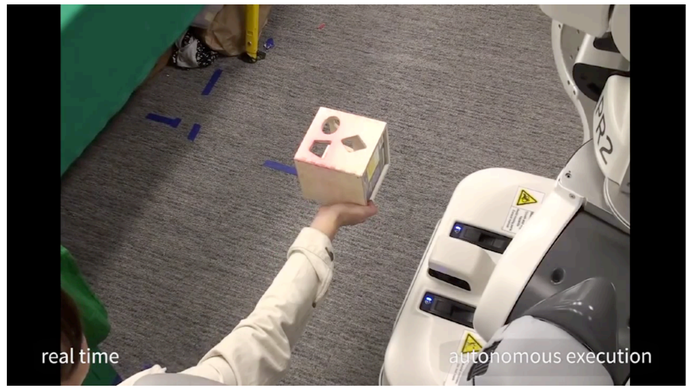
Brown et al. Language Models are Few-Shot Learners. 2020

Some of Chelsea's Research

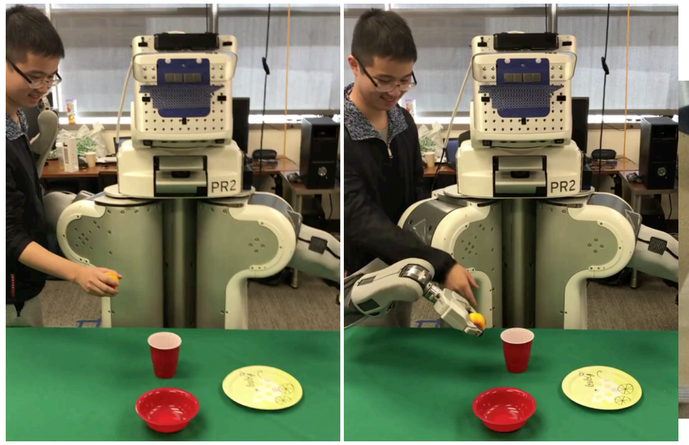
(and why I care about multi-task learning and meta-learning)

How can we enable agents to learn a breadth of skills in the real world?

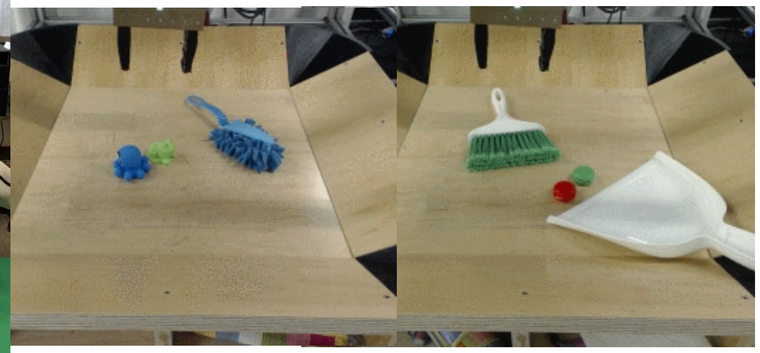
Robots.



Levine*, Finn*, Darrell, Abbeel. JMLR ‘16



Yu*, Finn*, Xie, Dasari, Zhang, Abbeel, Levine, RSS ‘18



Xie, Ebert, Levine, Finn, RSS ‘19

Why robots? Robots can teach us things about intelligence.

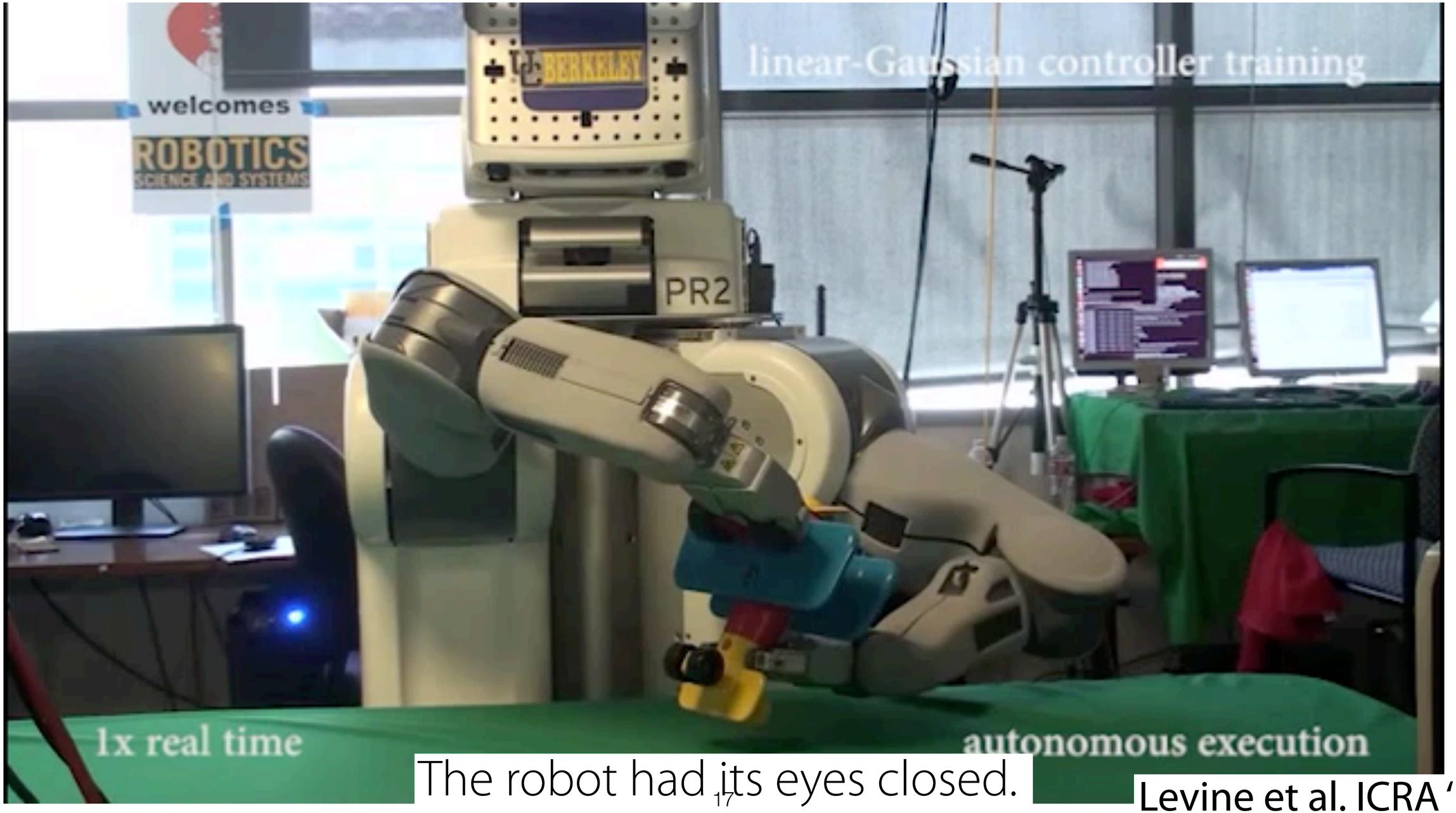
faced with the **real world**

must **generalize** across tasks, objects, environments, etc

need some **common sense understanding** to do well

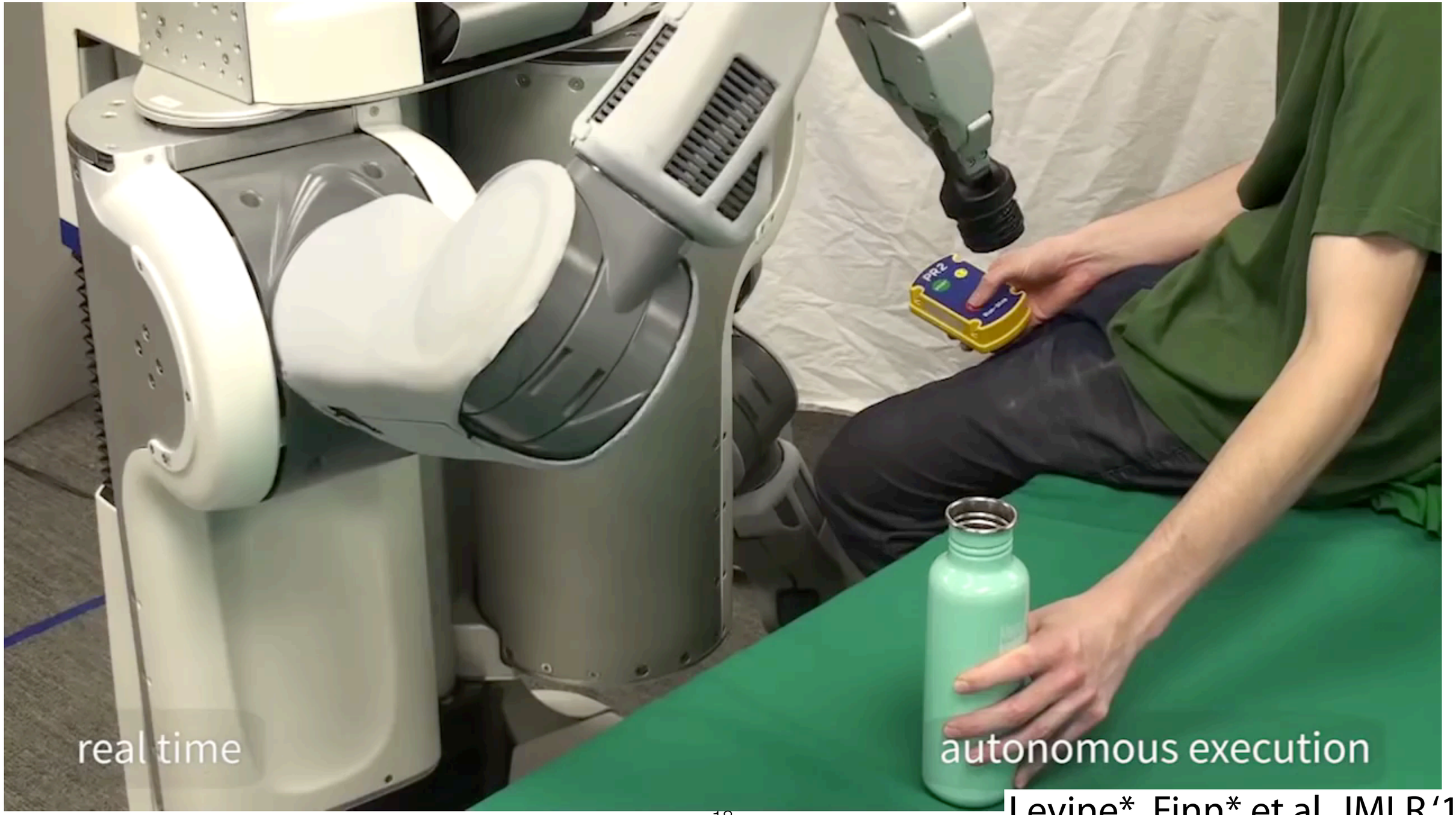
supervision can’t be taken for granted

Beginning of my PhD

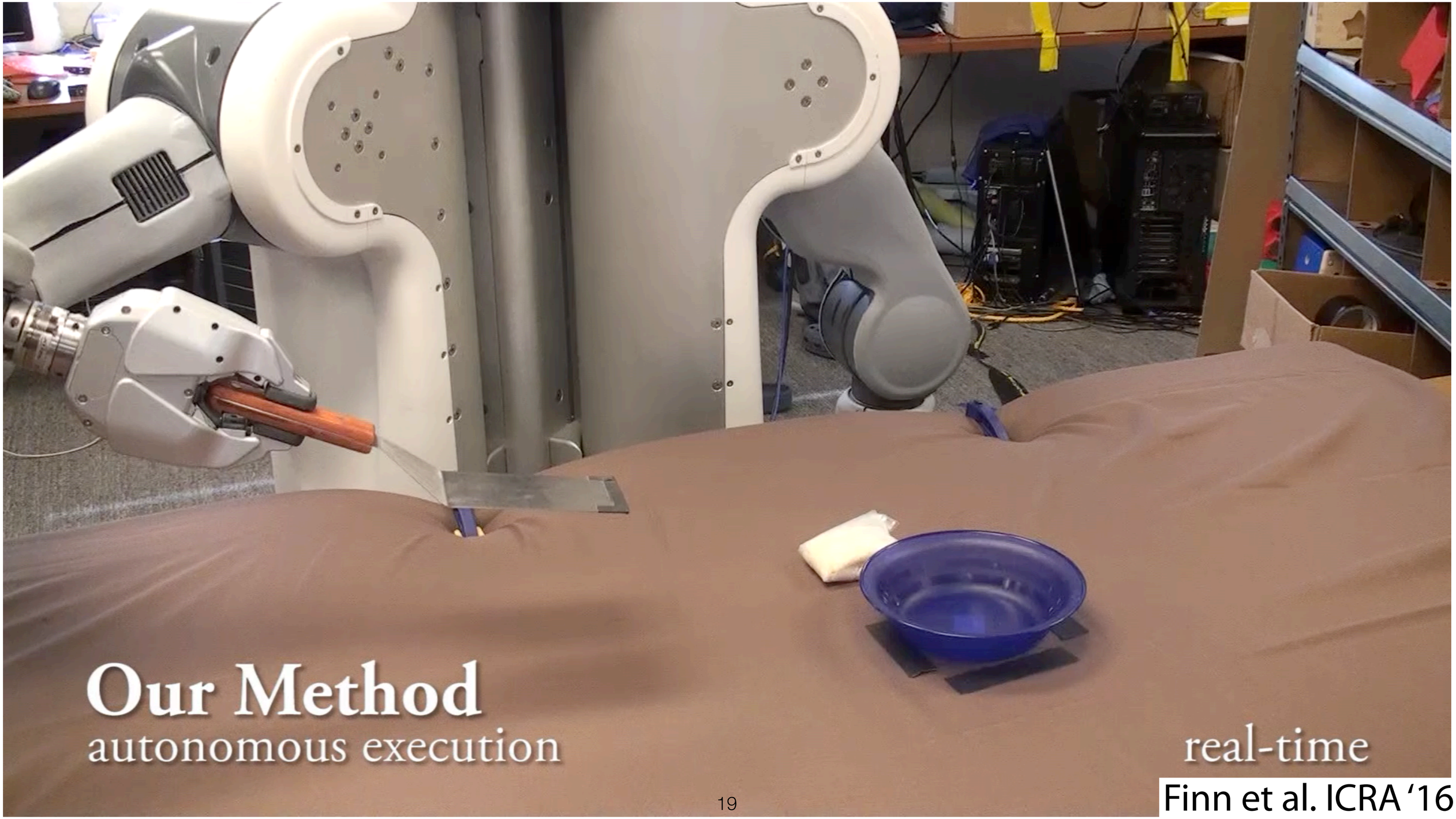


The robot had its eyes closed.

Levine et al. ICRA ‘15

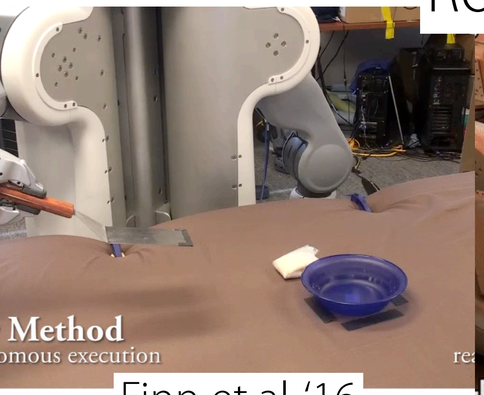


Levine*, Finn* et al. JMLR ‘16



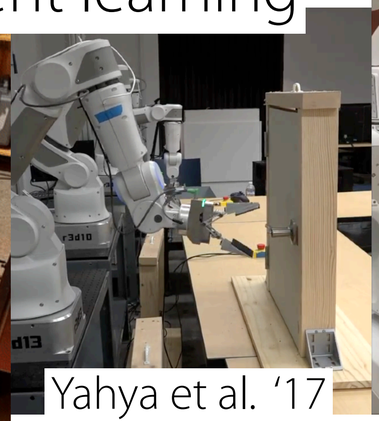
Finn et al. ICRA '16

Robot reinforcement learning

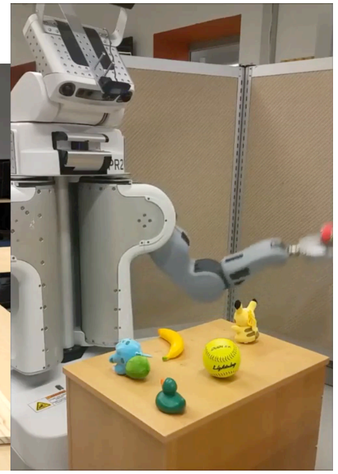


Finn et al. ‘16

Chebotar et al. ’17



Yahya et al. ‘17

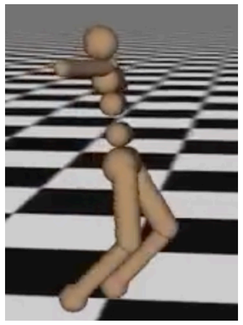


Ghadirzadeh et al. ’17

Reinforcement learning

AlphaGo

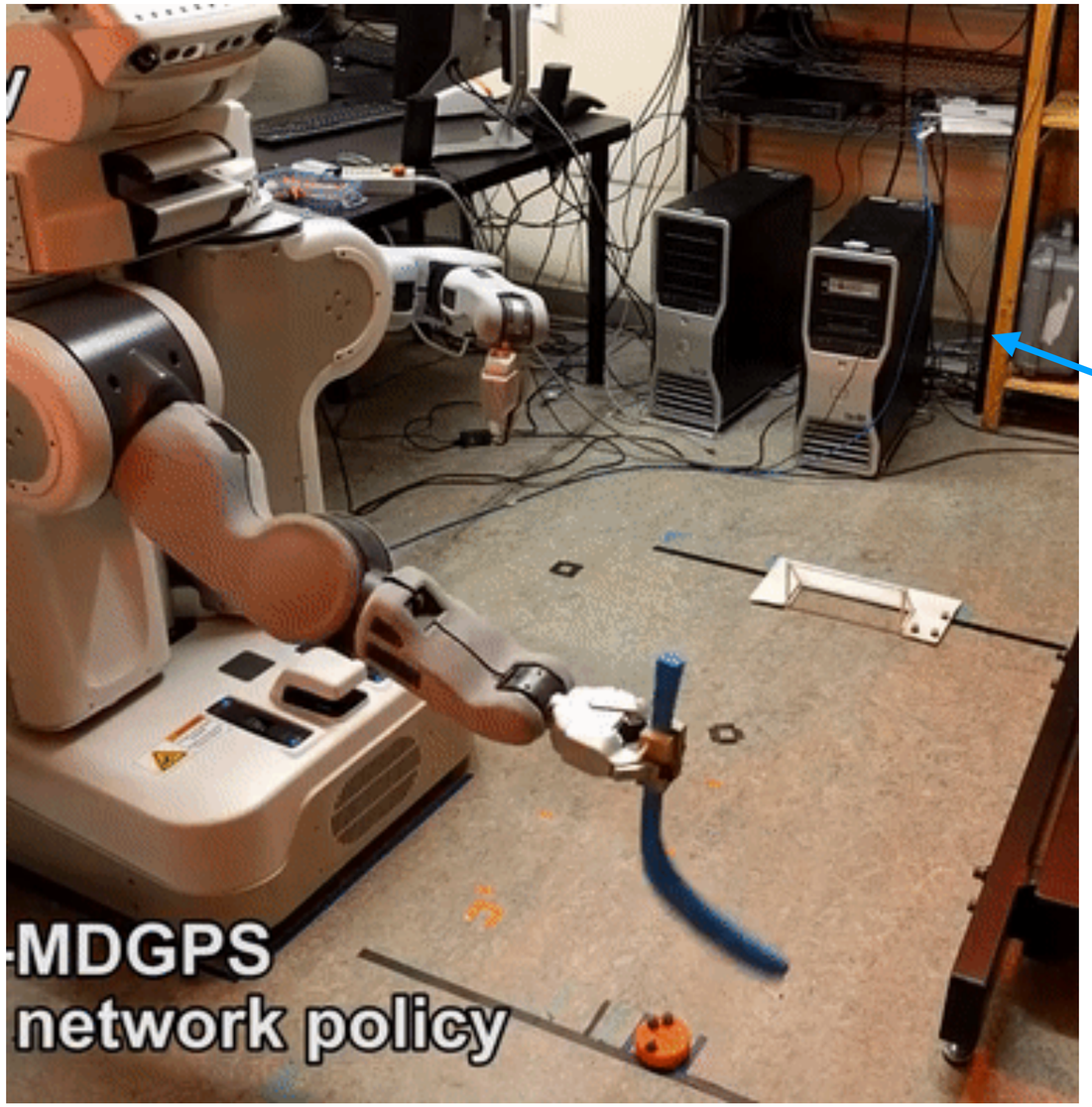
ATARI



locomotion

Learn **one task** in **one environment**, **starting from scratch**

Behind the scenes…

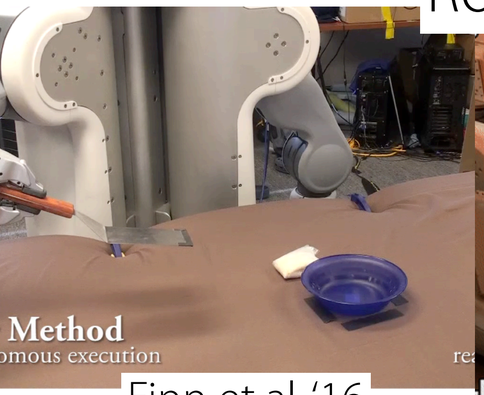


Yevgen

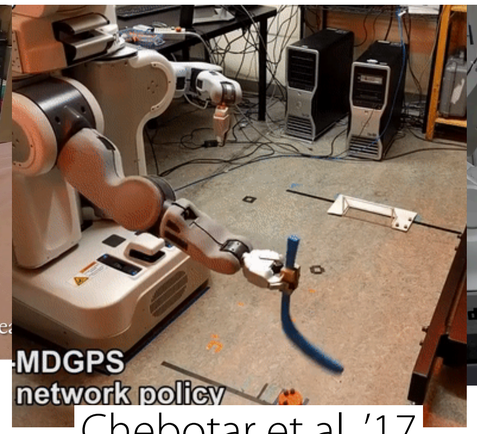
Yevgen is doing more work than the robot!

It’s not practical to collect a lot of data this way.

Robot reinforcement learning

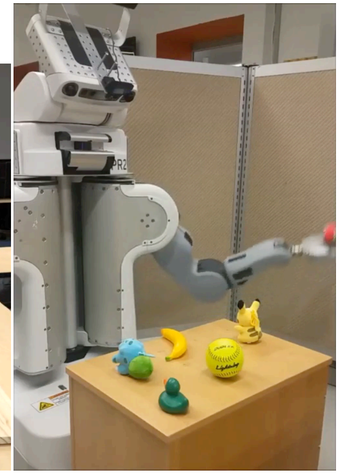


Finn et al. ‘16



Chebotar et al. ’17

Yahya et al. ‘17

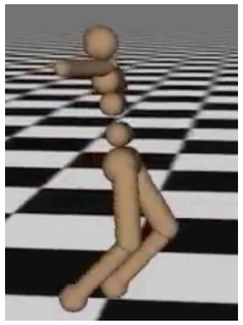


Ghadirzadeh et al. ’17

Reinforcement learning

AlphaGo

ATARI



locomotion

Learn **one task** in **one environment**, **starting from scratch** using **detailed supervision**

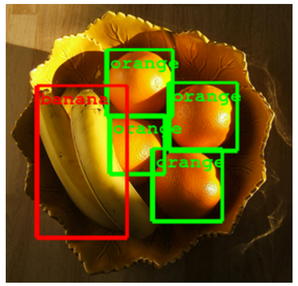
Not just a problem with reinforcement learning & robotics.

specialists

[single task]

machine translation

speech recognition



object detection

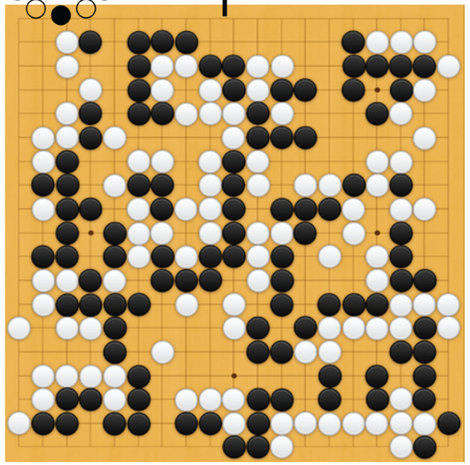
More diverse, yet still **one task**, **from scratch**, with **detailed supervision**

As an example, watch the following video:

https://youtu.be/8vNxjwt2AqY

Humans are ***generalists***.

AlphaGo

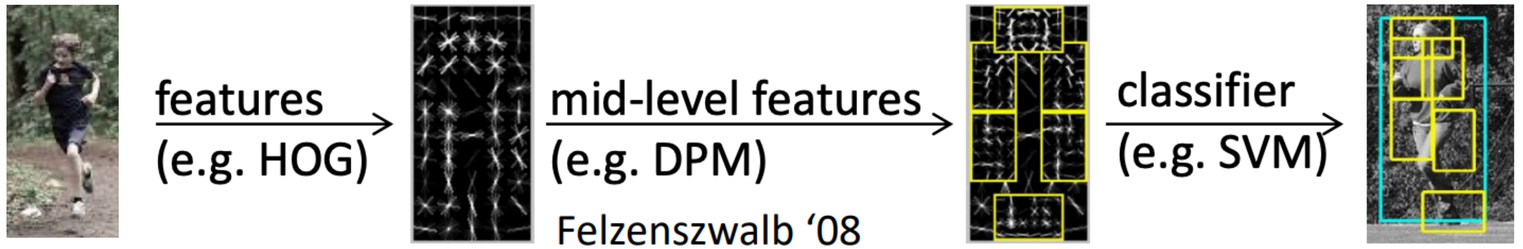


VS.

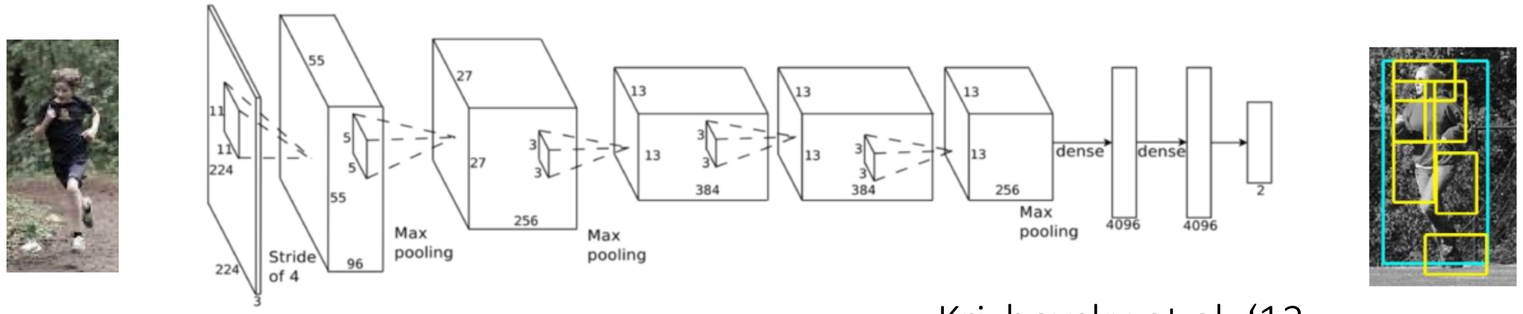
Why should we care about multi-task & meta-learning?

…beyond the robots and general-purpose ML systems

Standard computer vision:
hand-designed features



Modern computer vision:
end-to-end training

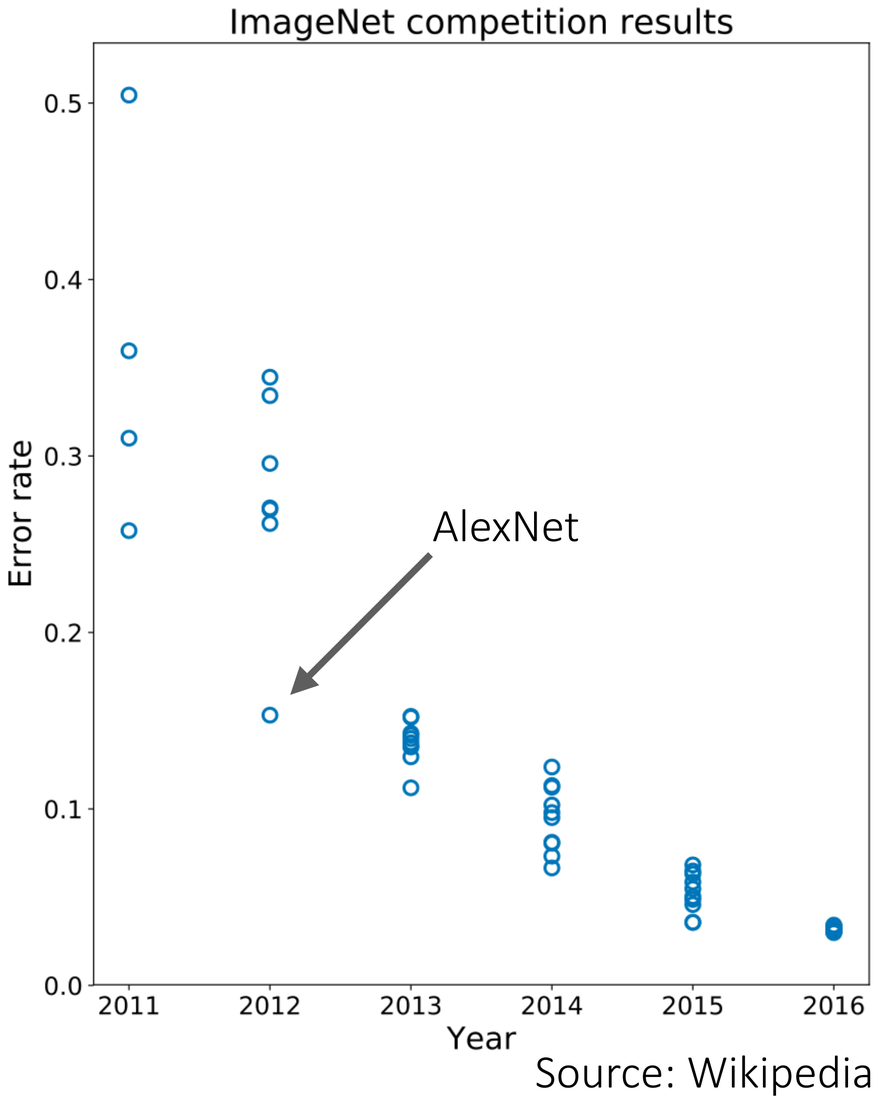


Krizhevsky et al. '12

Deep learning allows us to handle *unstructured inputs* (pixels, language, sensor readings, etc.) without hand-engineering features, with less domain knowledge

Slide adapted from Sergey Levine

Deep learning for object classification



Source: Wikipedia

Deep learning for machine translation

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

Table 10: Mean of side-by-side scores on production data

	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.504	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

Human evaluation scores on scale of 0 to 6

PBMT: Phrase-based machine translation

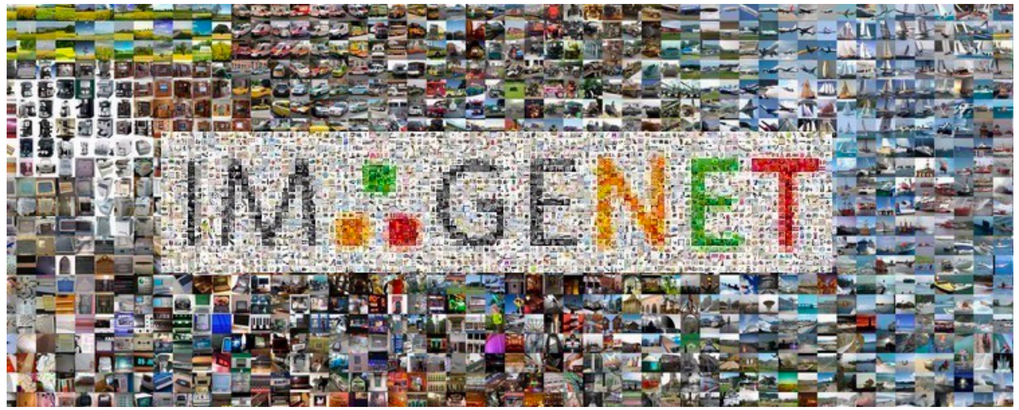
GNMT: Google's neural machine translation (in 2016)

Why deep **multi-task** and **meta-learning**?

Large, diverse data
(+ large models)

deep learning →

Broad generalization



Russakovsky et al. ‘14

Wu et al. ‘16

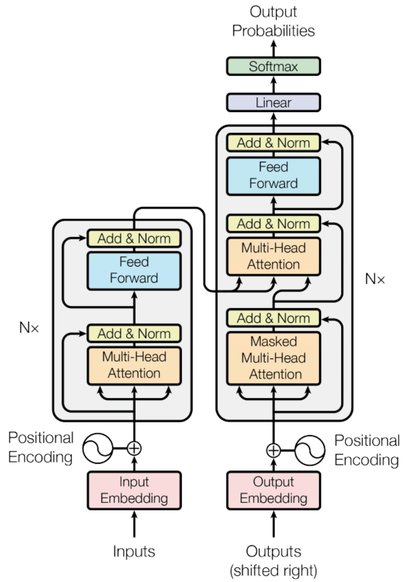


Figure 1: The Transformer - model architecture.

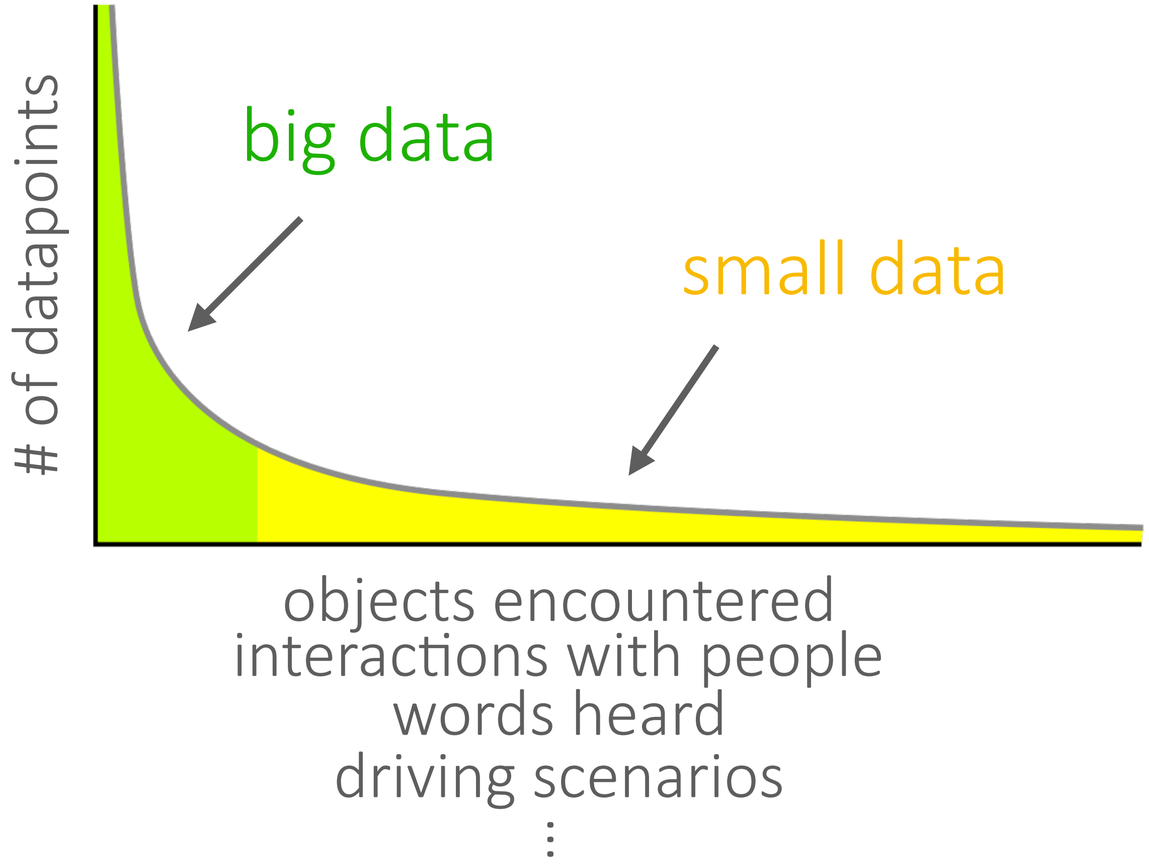
Vaswani et al. ‘18

What if you don’t have a large dataset?

medical imaging, robotics, personalized education, tr…ations

Impractical to learn from scratch for each disease, each robot, each person, each language, each **task**

What if your data has a long tail?



This setting breaks standard machine learning paradigms.

What if you need to quickly learn something new?

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

training data

Braque

Cezanne

test datapoint

By Braque or Cezanne?

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!

What if you want a more general-purpose AI system?

Learning each task from scratch won't cut it.

What if you don't have a large dataset?

medical imaging

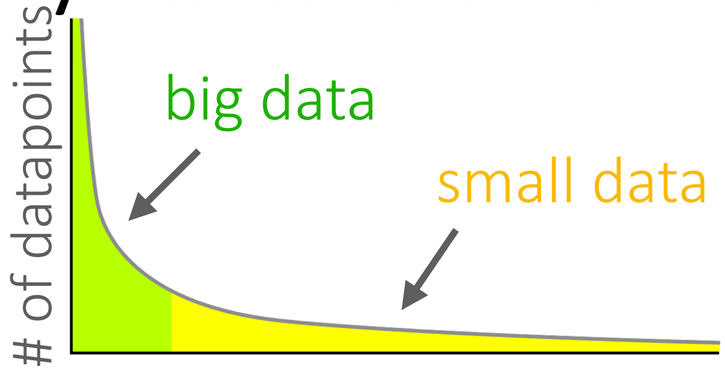
robotics

personalized education,

translation for rare languages

medicine, recommendations

What if your data has a long tail?



What if you need to quickly learn something new?

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

This is where elements of multi-task learning can come into play.

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 methods are continuing to play a major role in AI.

Visual language models can learn many distinct tasks.

Object recognition:

Reading & arithmetic:

Counting:

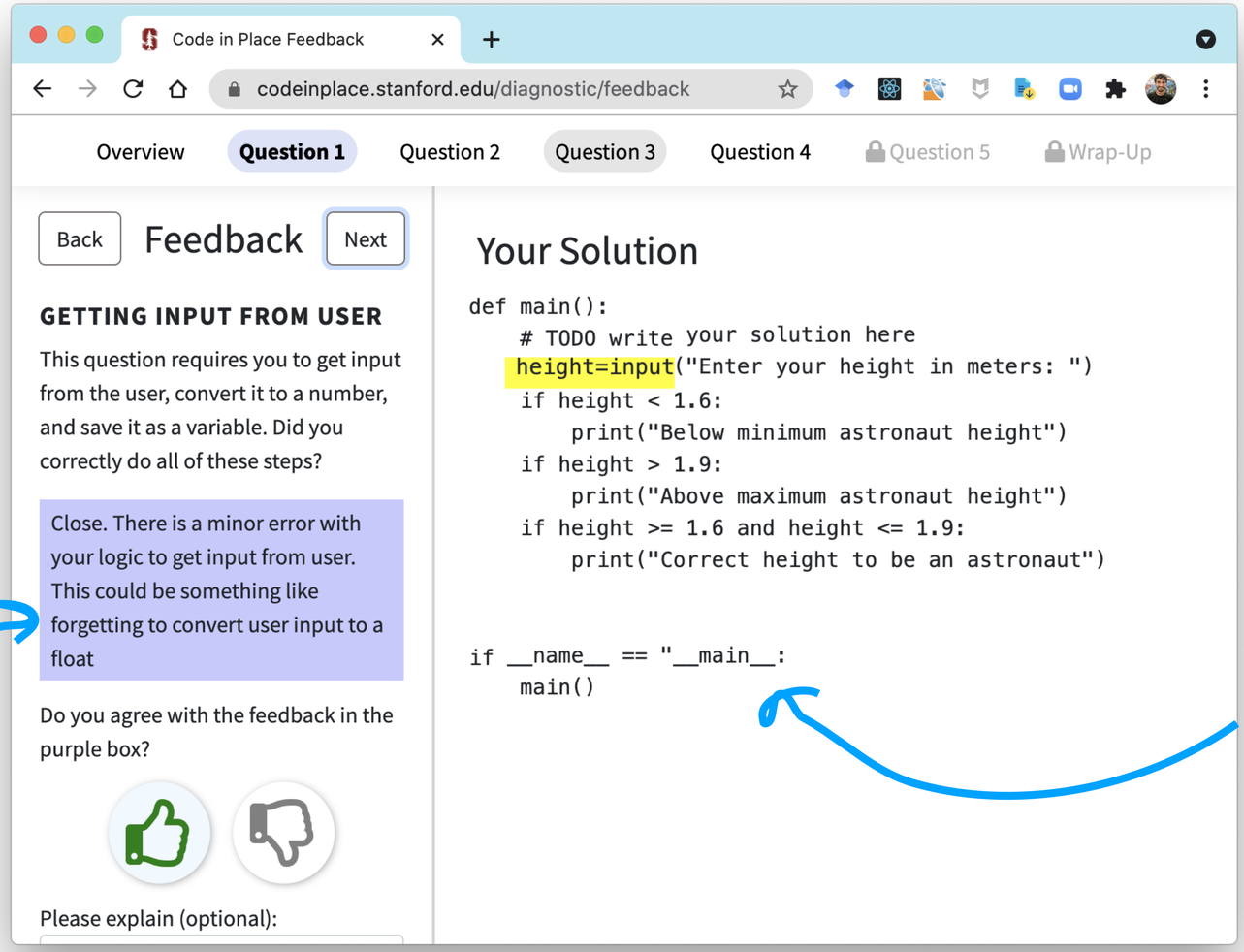
Alayrac*, Donahue*, Luc*, Miech* et al. Flamingo: a Visual Language Model for Few-Shot Learning. 2022

These methods are continuing to play a major role in AI.

Meta-learning enables automatic feedback on student work on new problems.

Code-in-Place 2021

Humans gave feedback on ~1k student programs.
Meta-learning system gave feedback on the remaining ~15k.



Generated feedback

Syntax error here would prevent unit tests from being useful

Wu, Goodman, Piech, Finn. ProtoTransformer: A Meta-Learning Approach to Providing Student Feedback. 2021

These methods are continuing to play a major role in AI.

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.

NAACL, 2019

YouTube recommendations

Recommending What Video to Watch Next: A Multitask Ranking System

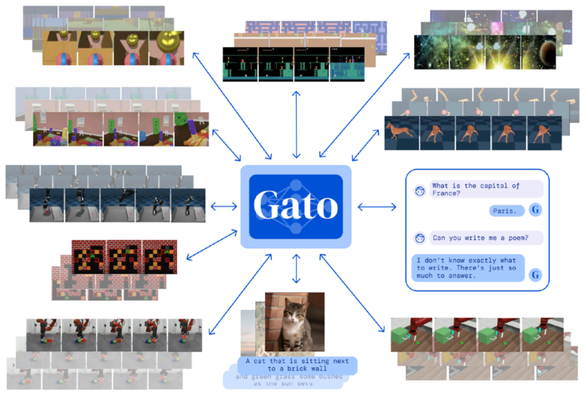
Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, Ed Chi
Google, Inc.
{zhezhao,lichan,liwei,jilinc,aniruddhnath,shawnandrews,aditeek,nlogn,xinyang,edchi}@google.com

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

RecSys 2019

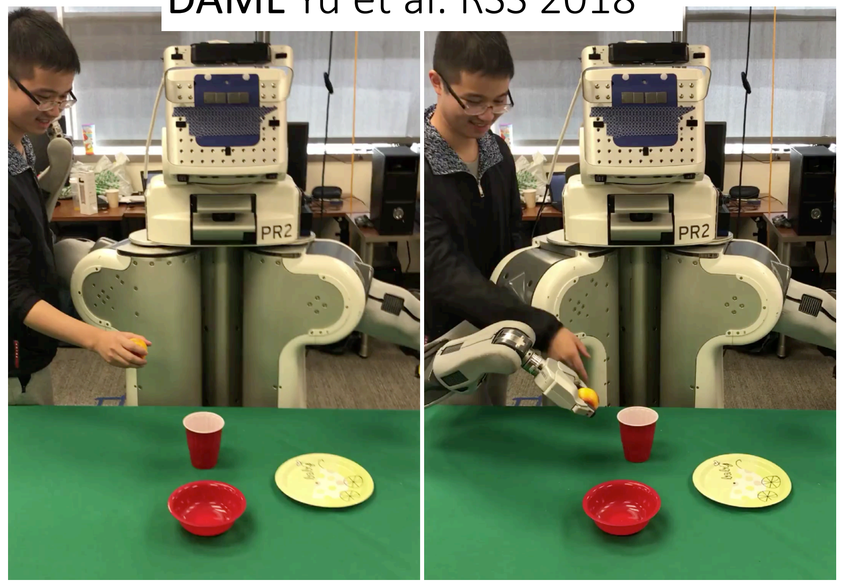
A Generalist Agent

Reed et al. 2022



One-shot imitation from humans

DAML Yu et al. RSS 2018



Its success is important for the **democratization** of deep learning.

ImageNet	1.2 million images and labels
WMT ’14 English - French	40.8 million paired sentences
Switchboard Speech Dataset	300 hours of labeled data

Kaggle’s Diabetic Retinopathy Detection dataset	35K labeled images
Adaptive epilepsy treatment with RL Guez et al. ‘08	< 1 hour of data
Learning for robotic manipulation Finn et al. ‘16	< 15 min of data

What is a task?

Informally: dataset $\mathcal{D}$, loss function $\mathcal{L}$ → 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.

Informal Problem Definitions

We'll define these more formally later.

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

The transfer learning problem: Given data on previous task(s), learn a new task more quickly and/or more proficiently.

This course: anything that solves these problem statements.

Doesn’t multi-task learning reduce to single-task learning?

$$\mathcal{D} = \bigcup \mathcal{D}_i \qquad\qquad \mathcal{L} = \sum \mathcal{L}_i$$

Are we done with the course?

Doesn’t multi-task learning reduce to single-task learning?

Yes, it can!

Aggregating the data across tasks & learning a single model is one approach to multi-task learning.

But, what if you want to learn *new* tasks?

And, how do we tell the model what task to do?

And, what if aggregating doesn’t work?

Plan for Today

Multi-Task Learning

- Problem statement
- Models, objectives, optimization
- Challenges
- Case study of real-world multi-task learning

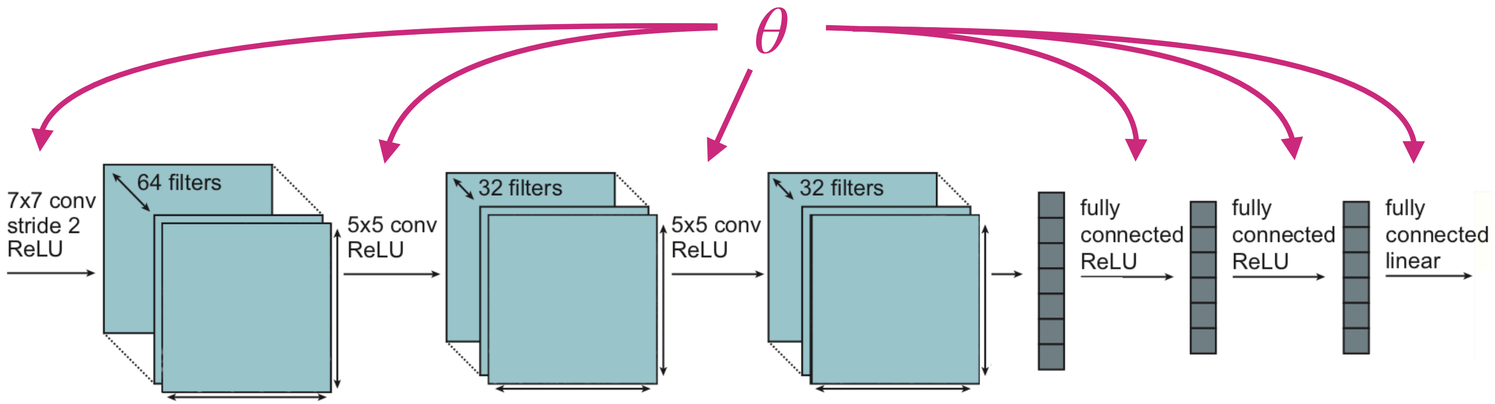
Goals for by the end of lecture:

- Understand the key design decisions when building multi-task learning systems

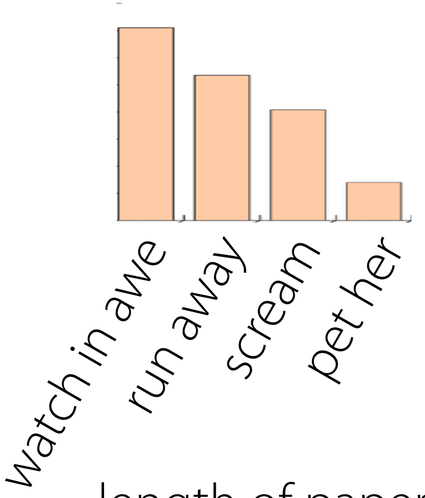
Some notation

ImageNet Classification with Deep Convolutional Neural Networks

$\mathbf{x}$



$\mathbf{y}$



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 $\mathscr{D}_i^{tr}$:

Examples of Tasks

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

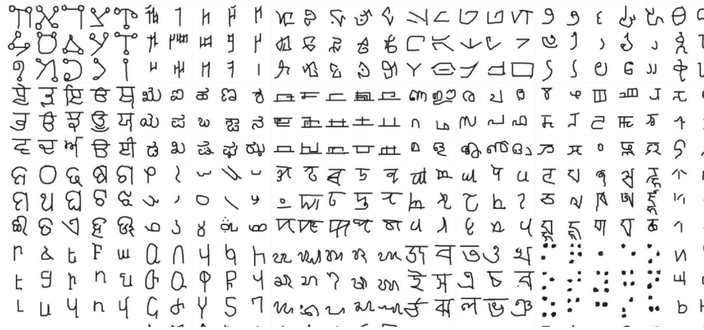
data generating distributions

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

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

Multi-task classification: $\mathcal{L}_i$ same across all tasks

e.g. per-language handwriting recognition

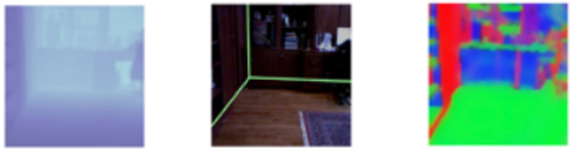


e.g. personalized spam filter

Multi-label learning: $\mathcal{L}_i$, $p_i(\mathbf{x})$ same across all tasks

e.g. face attribute recognition

e.g. scene understanding

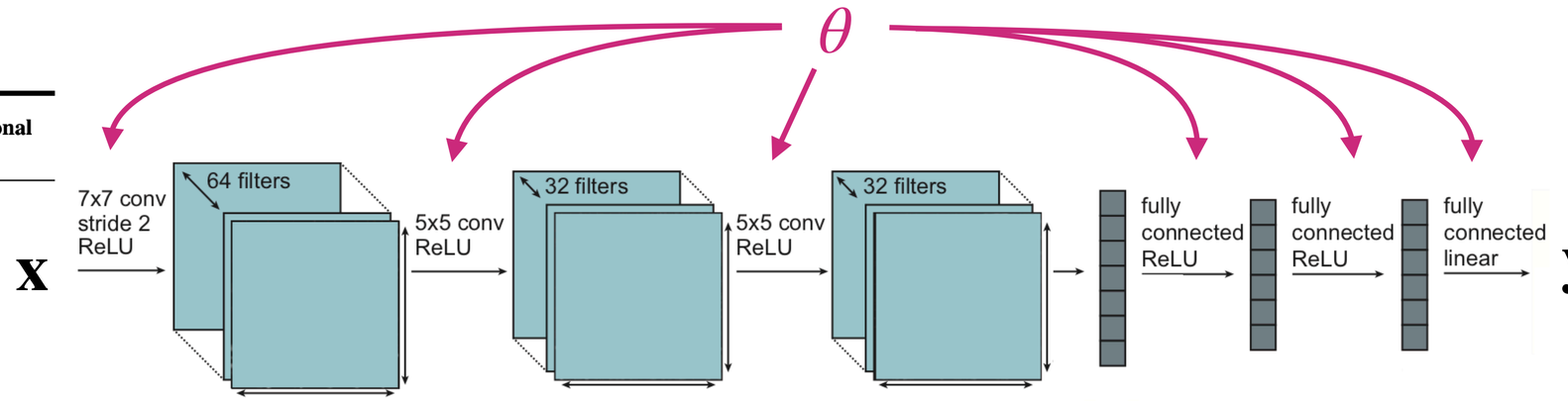


$$L_{\text{tot}} = w_{\text{depth}} L_{\text{depth}} + w_{\text{kpt}} L_{\text{kpt}} + w_{\text{normals}} L_{\text{normals}}$$

When might $\mathcal{L}_i$ vary across tasks?

- mixed discrete, continuous labels across tasks
- multiple metrics that you care about

ImageNet Classification with Deep Convolutional Neural Networks



length of paper

summary of paper

paper review

$\mathbf{z}_i$

task descriptor

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

e.g. one-hot encoding of the task index

or, whatever meta-data you have

- personalization: user features/attributes
- language description of the task
- formal specifications of the task

Vanilla MTL Objective

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

Decisions on the model, the objective, and the optimization.

How should we condition on $\mathbf{z}_i$? What objective should we use?

How to optimize our objective?

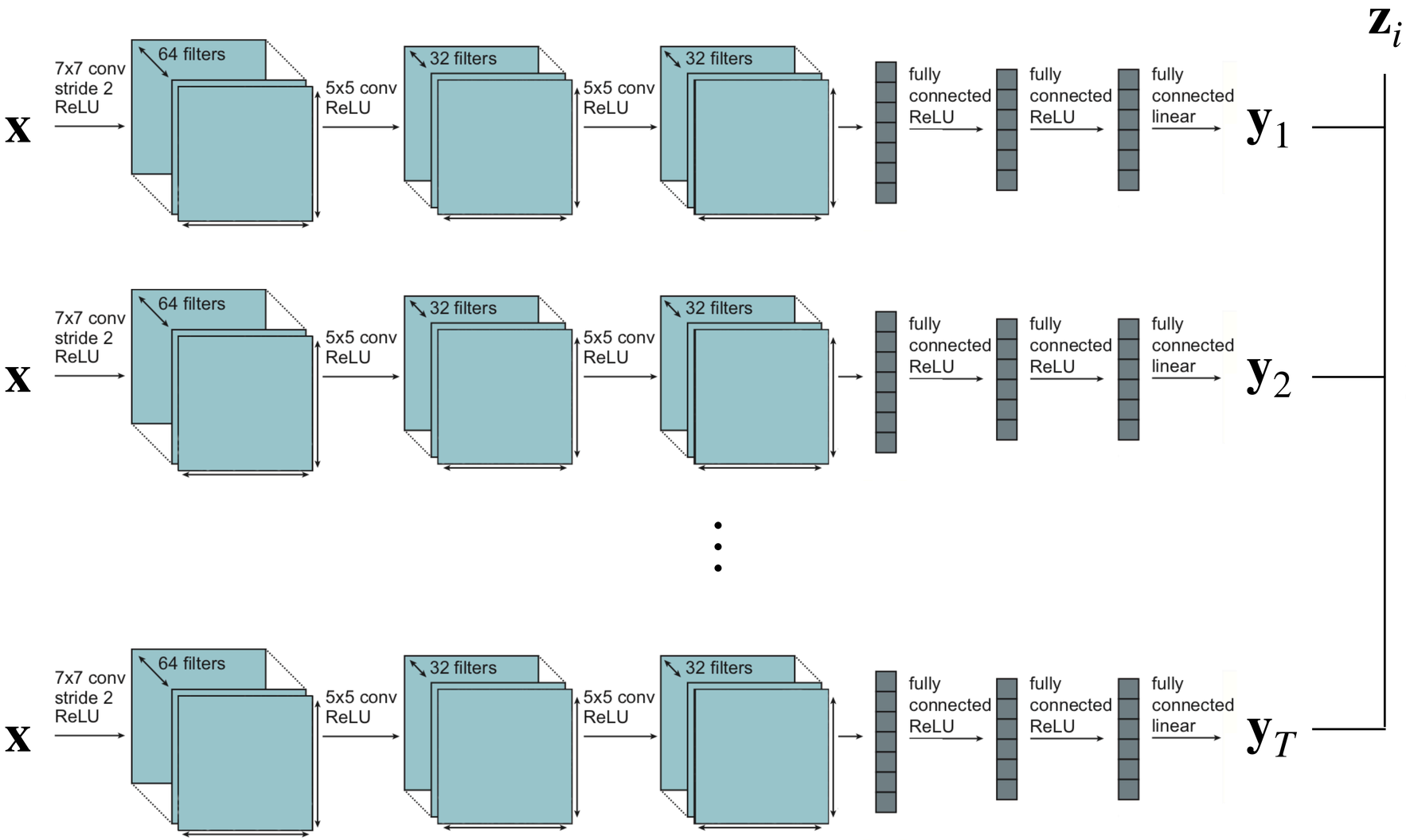
Model	How should the model be conditioned on $\mathbf{z}_i$? What parameters of the model should be shared?
Objective	How should the objective be formed?
Optimization	How should the objective be optimized?

Conditioning on the task

Let's assume $\mathbf{z}_i$ is the one-hot task index.

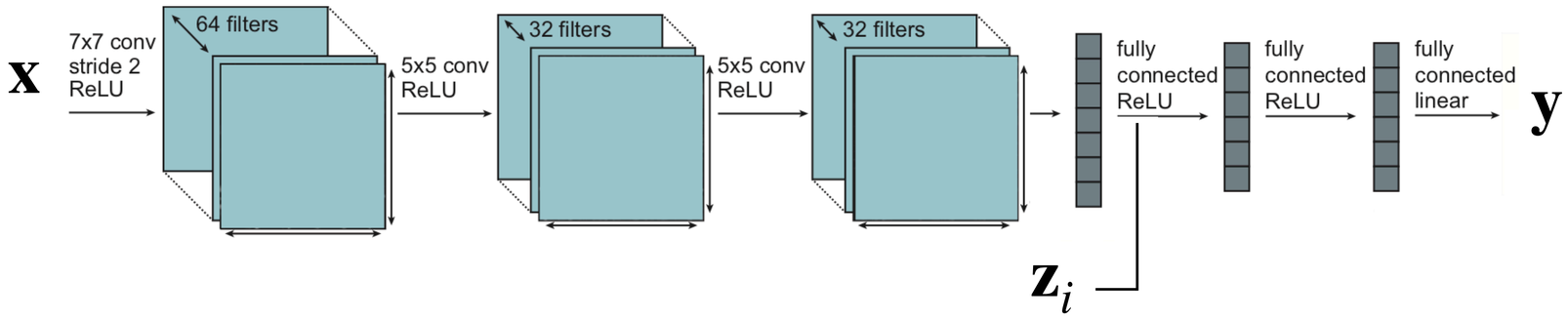
Question: How should you condition on the task in order to share as little as possible?

Conditioning on the task



—> independent training within a single network!
with no shared parameters

The other extreme



Concatenate $\mathbf{z}_i$ with input and/or activations

all parameters are shared
(except the parameters directly following $\mathbf{z}_i$, if $\mathbf{z}_i$ is one-hot)

An Alternative View on the Multi-Task Architecture

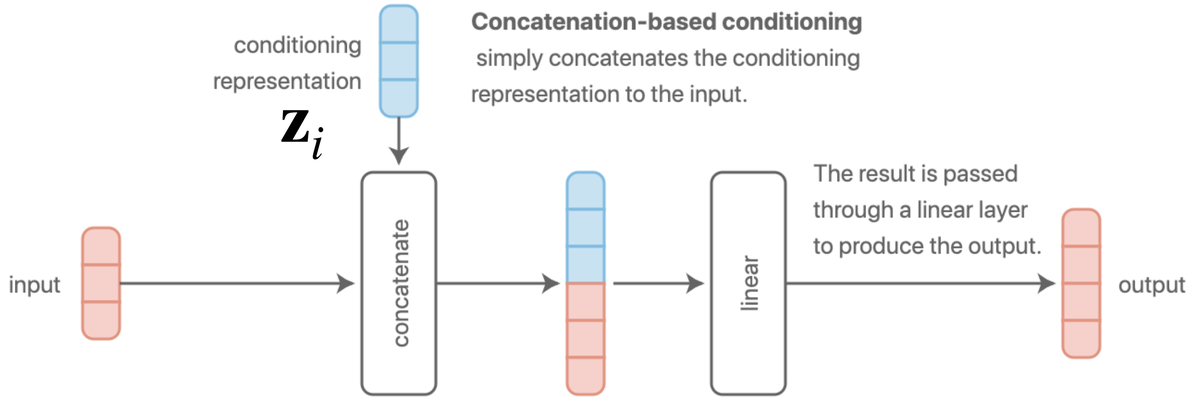
Split θ into shared parameters θ^{sh} and task-specific parameters θ^{i}

Then, our objective is: $\min_{\theta^{sh},\theta^{1},\ldots,\theta^{T}} \sum_{i=1}^{T} \mathscr{L}_i(\{\theta^{sh},\theta^{i}\},\mathscr{D}_i)$

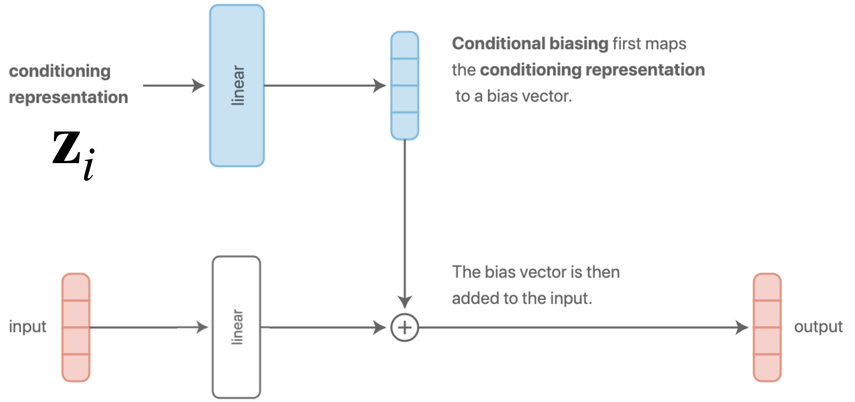
Choosing how to condition on $\mathbf{z}_i$ equivalent to Choosing how & where to share parameters

Conditioning: Some Common Choices

1. **Concatenation-based** conditioning



2. **Additive** conditioning



These are actually equivalent!

Question: why are they the same thing? (raise your hand)

Concat followed by a fully-connected layer:

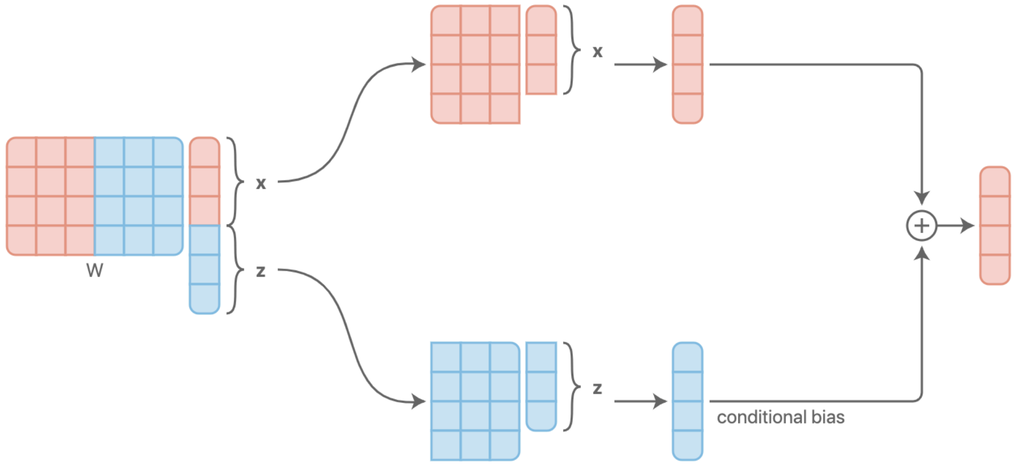
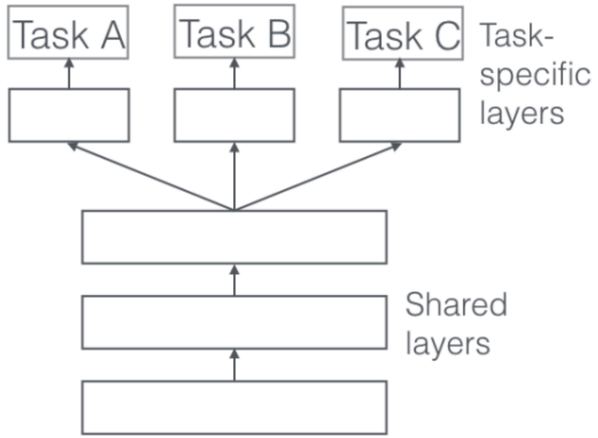


Diagram sources: distill.pub/2018/feature-wise-transformations/

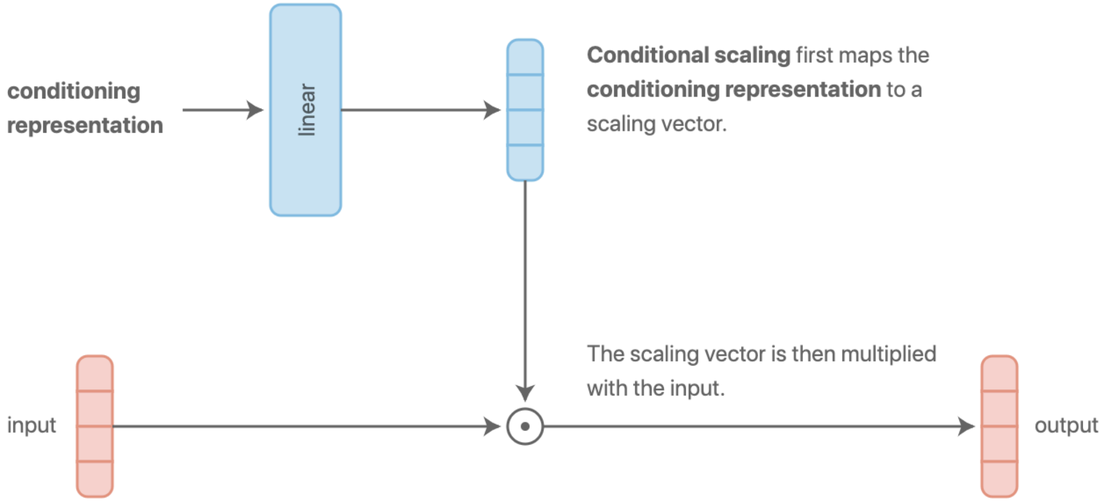
Conditioning: Some Common Choices

3. **Multi-head architecture**



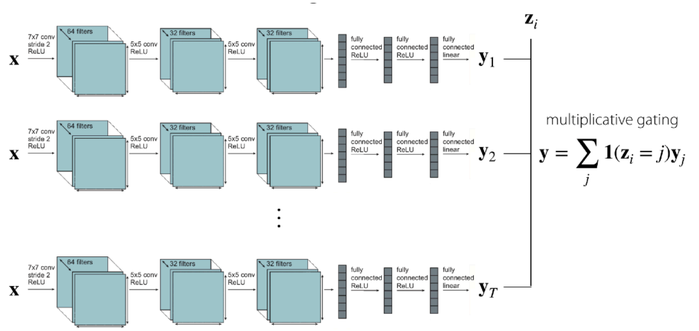
Ruder '17

4. **Multiplicative** conditioning



Why might multiplicative conditioning be a good idea?

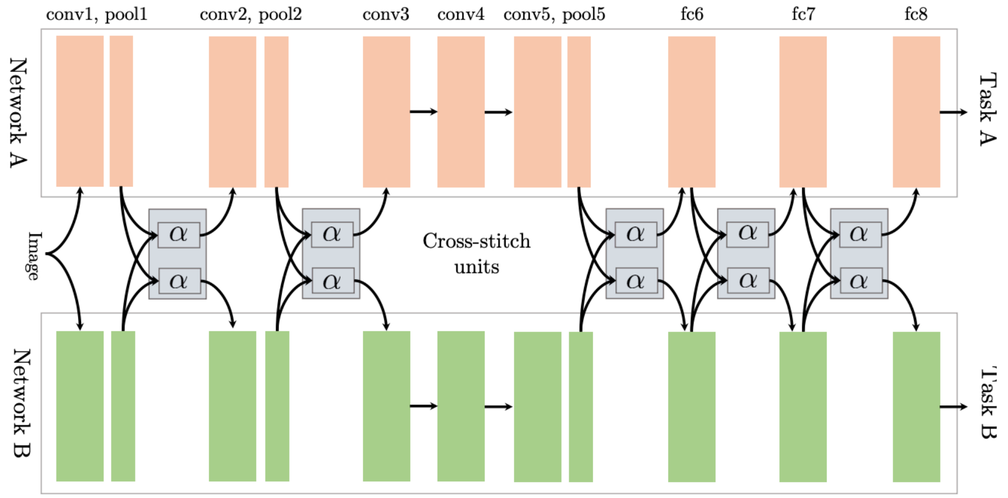
- more expressive per layer
- recall: multiplicative gating



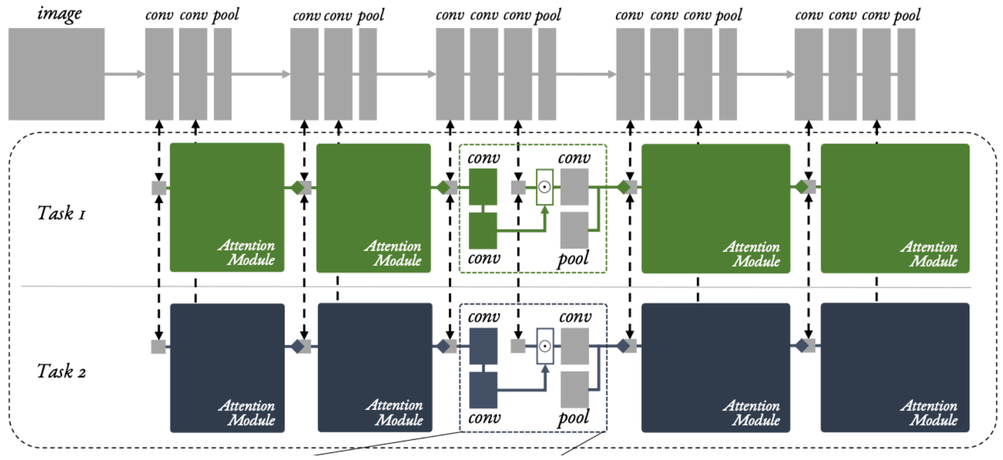
Multiplicative conditioning **generalizes** independent networks and independent heads.

Diagram sources: distill.pub/2018/feature-wise-transformations/

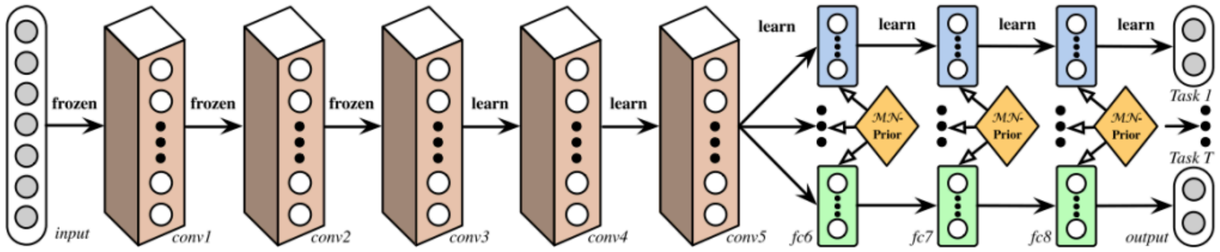
Conditioning: More Complex Choices



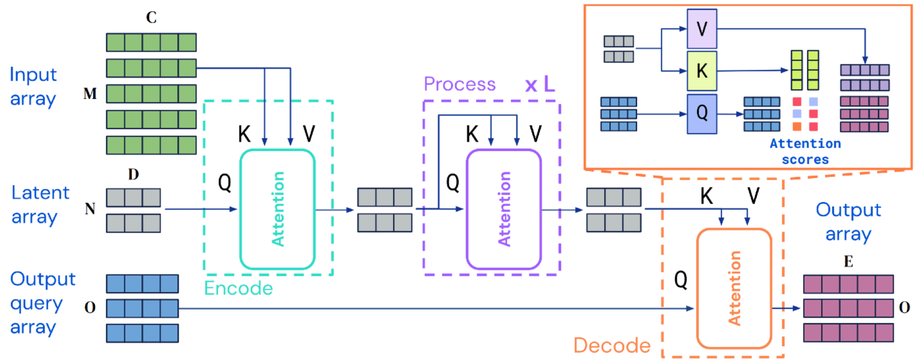
Cross-Stitch Networks. Misra, Shrivastava, Gupta, Hebert '16



Multi-Task Attention Network. Liu, Johns, Davison '18



Deep Relation Networks. Long, Wang '15



Perceiver IO. Jaegle et al. '21

Conditioning Choices

Unfortunately, these design decisions are like neural network architecture tuning:

- **problem dependent**
- largely guided by **intuition** or **knowledge** of the problem
- currently more of an **art** than a science

Model	How should the model be conditioned on $\mathbf{z}_i$? What parameters of the model should be shared?
Objective	How should the objective be formed?
Optimization	How should the objective be optimized?

Vanilla MTL objective:

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

Often want to weight tasks differently:

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

How to choose w_i?

- manually based on importance or priority
- *dynamically* adjust throughout training

a. various heuristics

encourage gradients to have similar magnitudes
(Chen et al. GradNorm. ICML 2018)

b. optimize for the worst-case task loss

$$\min_{\theta} \max_{i} \mathscr{L}_i(\theta, \mathscr{D}_i)$$

(e.g. for task robustness, or for fairness)

Model	How should the model be conditioned on $\mathbf{z}_i$? What parameters of the model should be shared?
Objective	How should the objective be formed?
Optimization	How should the objective be optimized?

Optimizing the objective

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

Basic Version:

1. Sample mini-batch of tasks $\mathscr{B} \sim \{\mathscr{T}_i\}$
2. Sample mini-batch datapoints for each task $\mathscr{D}_i^b \sim \mathscr{D}_i$
3. Compute loss on the mini-batch: $\hat{\mathscr{L}}(\theta, \mathscr{B}) = \sum_{\mathscr{T}_k \in \mathscr{B}} \mathscr{L}_k(\theta, \mathscr{D}_k^b)$
4. Backpropagate loss to compute gradient $\nabla_\theta \hat{\mathscr{L}}$
5. Apply gradient with your favorite neural net optimizer (e.g. Adam)

Note: This ensures that tasks are sampled uniformly, regardless of data quantities.

Tip: For regression problems, make sure your task labels are on the same scale!

Challenge #1: Negative transfer

Negative transfer: Sometimes independent networks work the best.

Multi-Task CIFAR-100

recent approaches

	% accuracy	
task specific, 1-fc (Rosenbaum et al., 2018)	42	} multi-head architectures
task specific, all-fc (Rosenbaum et al., 2018)	49	} multi-head architectures
cross stitch, all-fc (Misra et al., 2016b)	53	} cross-stitch architecture
independent	67.7	} independent training

(Yu et al. Gradient Surgery for Multi-Task Learning. 2020)

Why?

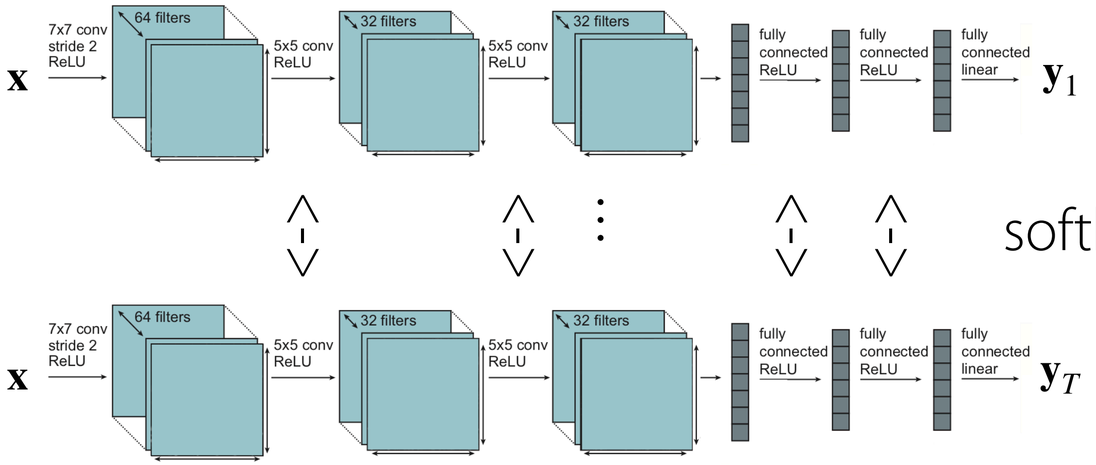
- **optimization challenges**
  - caused by cross-task interference
  - tasks may learn at different rates
- **limited representational capacity**
  - multi-task networks often need to be *much larger* than their single-task counterparts

If you have negative transfer, **share less** across tasks.

It's not just a binary decision!

$$\min_{\theta^{sh},\theta^1,\ldots,\theta^T} \sum_{i=1}^{T} \mathscr{L}_i(\{\theta^{sh},\theta^i\},\mathscr{D}_i) + \lambda \underbrace{\sum_{i'=1}^{T} \|\theta^i - \theta^{i'}\|}$$

"soft parameter sharing"



softly constrained weights

+ allows for more fluid degrees of parameter sharing

- yet another set of design decisions / hyperparameters

- more memory intensive

Challenge #2: Overfitting

You may not be sharing enough!

Multi-task learning <-> a form of regularization

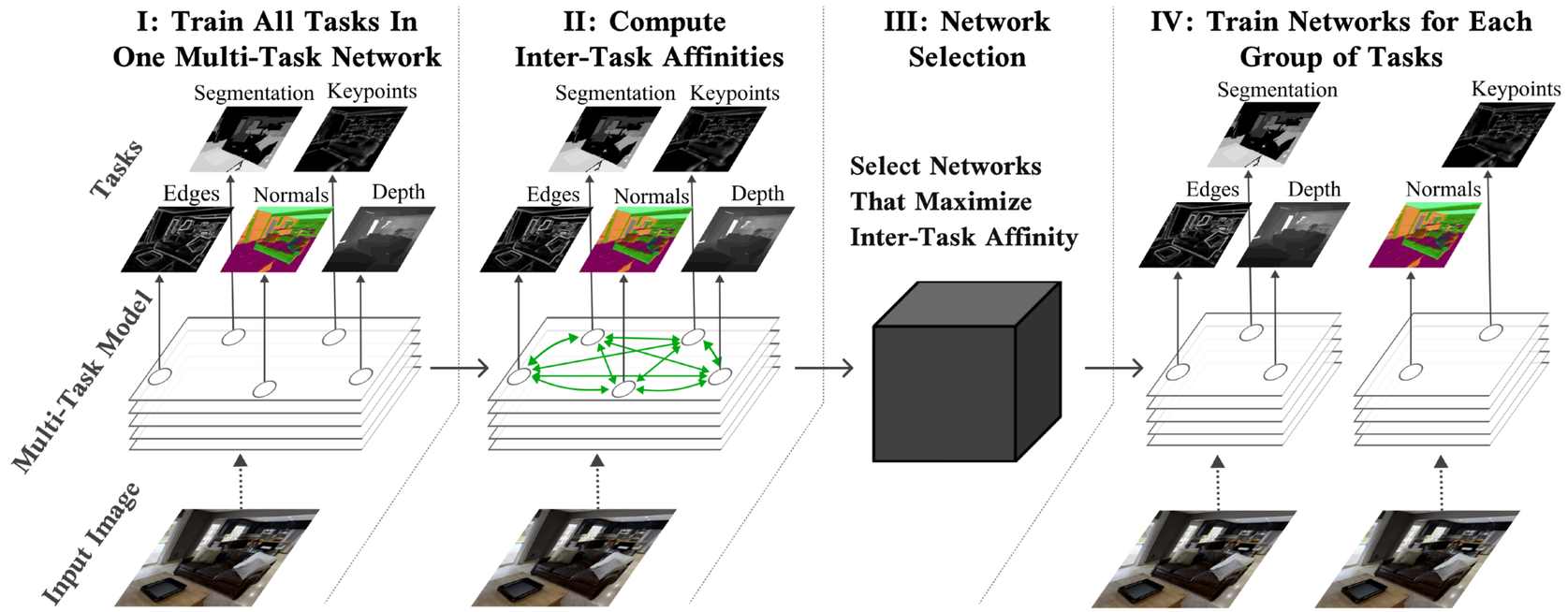
Solution: Share more.

Challenge #3: What if you have a lot of tasks?

Should you train all of them together? Which ones will be complementary?

The bad news: No closed-form solution for measuring task similarity.

The good news: There are ways to approximate it from one training run.



Fifty, Amid, Zhao, Yu, Anil, Finn. ***Efficiently Identifying Task Groupings for Multi-Task Learning***. NeurIPS 2021

Multi-Task Learning Recap

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

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

Model Architecture

- multiplicative vs. additive conditioning on $\mathbf{z}_i$
- share more vs. less depending on observed transfer

Objective & Optimization

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

- choosing task weights
- stratified mini-batches

Plan for Today

Multi-Task Learning

- Problem statement
- Models, objectives, optimization
- Challenges
- **Case study of real-world multi-task learning**

Case study

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

Goal: Make recommendations for YouTube

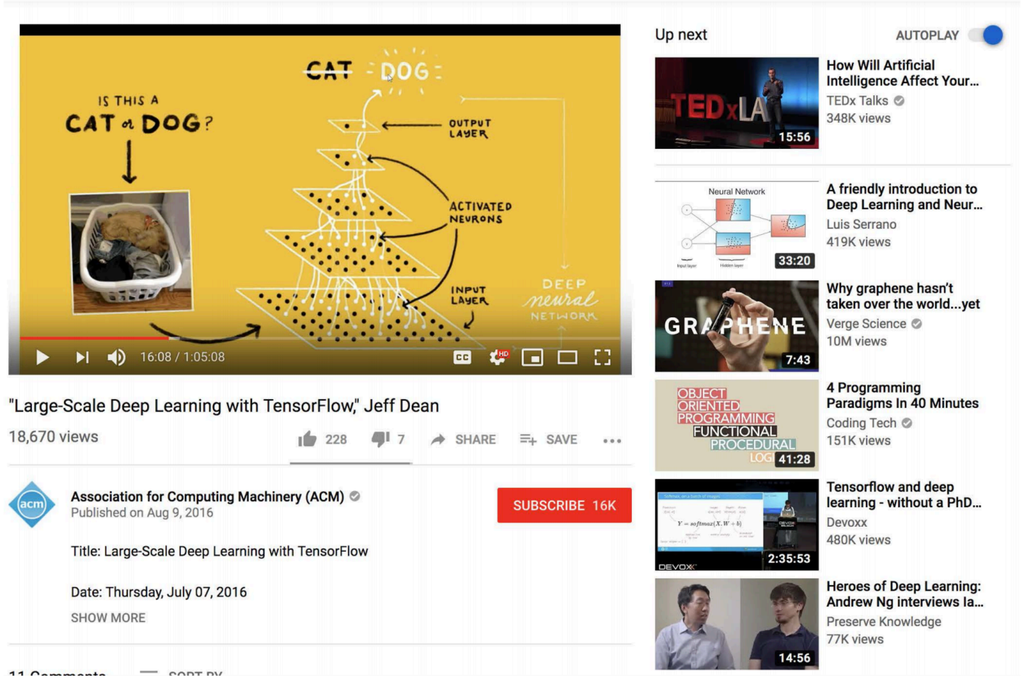


Figure 4: Recommending what to watch next on YouTube.

Framework Set-Up

Input: what the user is currently watching (query video) + user features

1. Generate a few hundred of candidate videos
2. Rank candidates
3. Serve top ranking videos to the user

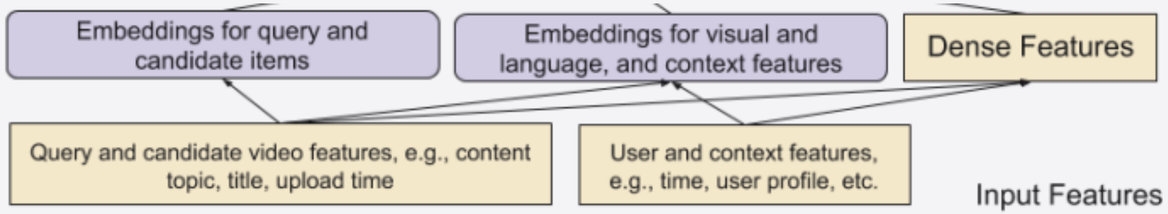
Candidate videos: pool videos from multiple candidate generation algorithms

- matching topics of query video
- videos most frequently watched with query video
- And others

Ranking: central topic of this paper

The Ranking Problem

Input: query video, candidate video, user & context features



Model output: engagement and satisfaction with candidate video

Engagement:

- binary classification tasks like **clicks**
- regression tasks related to **time spent**

Satisfaction:

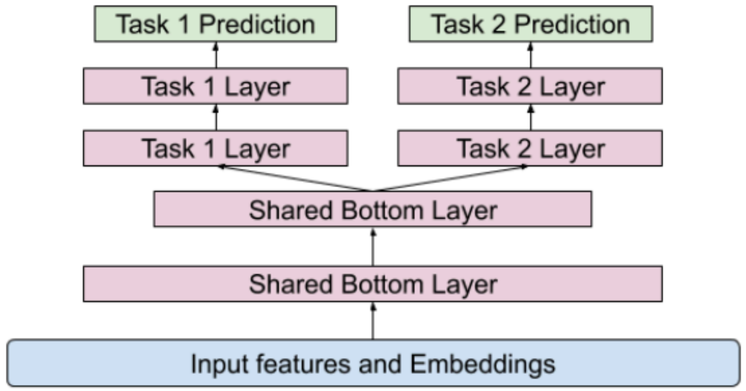
- binary classification tasks like **clicking "like"**
- regression tasks such as **rating**

Weighted combination of **engagement** & **satisfaction** predictions -> **ranking score**
score weights manually tuned

Question: Are these objectives reasonable? What are some of the issues that might come up?

The Architecture

Basic option: "Shared-Bottom Model"
(i.e. multi-head architecture)

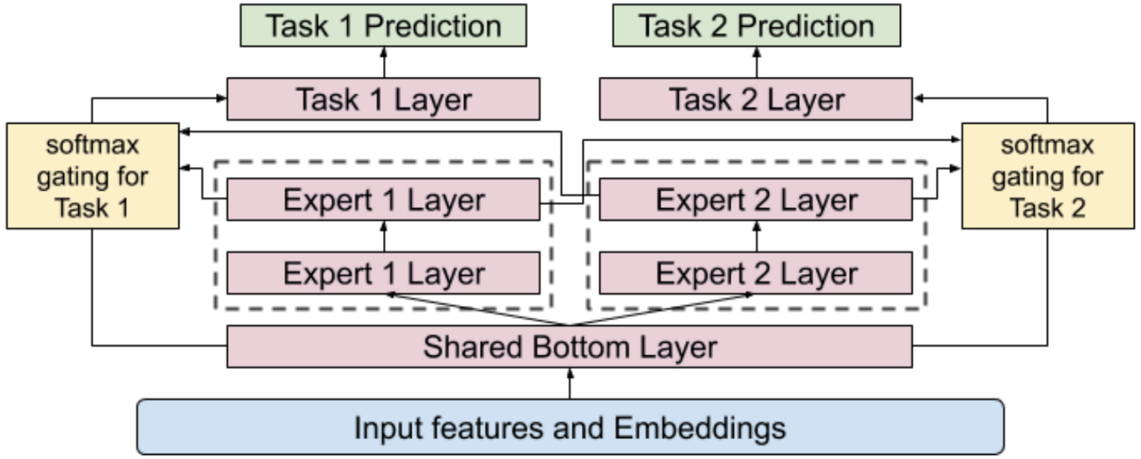


(a) Shared-Bottom Model with shared bottom hidden layers and separate towers for two tasks.

-> harms learning when correlation between tasks is low

The Architecture

Instead: use a form of soft-parameter sharing
"**Multi-gate Mixture-of-Experts (MMoE)**"



(b) Multi-gate Mixture-of-Expert Model with one shared bottom layer and separate hidden layers for two tasks.

Allow different parts of the network to "specialize"

expert neural networks $f_i(x)$

Decide which expert to use for input x, task k:

$$g^k(x) = \mathbf{softmax}(W_{g^k} x)$$

Compute features from selected expert:

$$f^k(x) = \sum_{i=1}^{n} g^k_{(i)}(x) f_i(x)$$

Compute output: $y_k = h^k(f^k(x)),$

Experiments

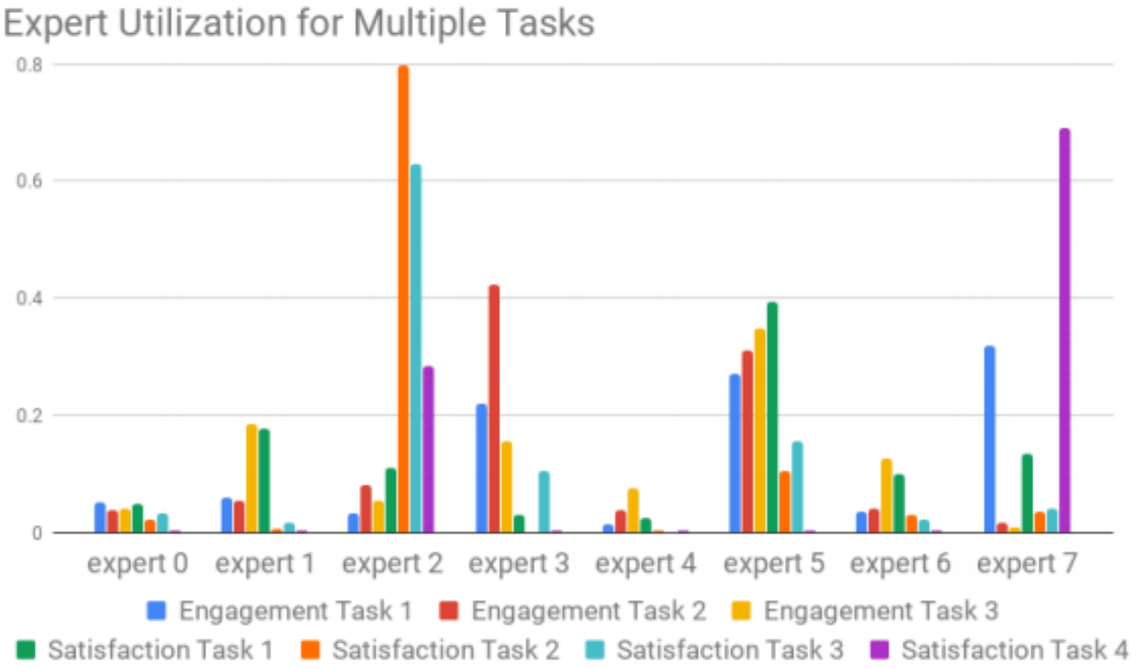
Set-Up

- Implementation in TensorFlow, TPUs
- Train in ***temporal order***, running training continuously to consume newly arriving data
- **Online A/B testing** in comparison to production system
  - live metrics based on time spent, survey responses, rate of dismissals
- Model **computational efficiency** matters

Results

Model Architecture	Number of Multiplications	Engagement Metric	Satisfaction Metric
Shared-Bottom	3.7M	/	/
Shared-Bottom	6.1M	+0.1%	+ 1.89%
MMoE (4 experts)	3.7M	+0.20%	+ 1.22%
MMoE (8 Experts)	6.1M	+0.45%	+ 3.07%

Table 1: YouTube live experiment results for MMoE.



Found 20% chance of gating polarization during distributed training -> use drop-out on experts

Recap from Last Time

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

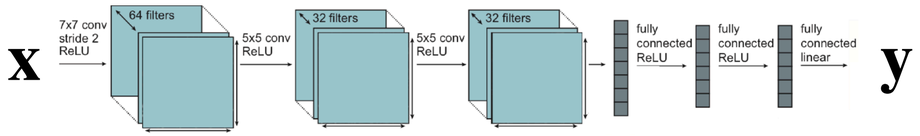
Corresponding datasets: $\mathcal{D}_i^{tr}$ $\mathcal{D}_i^{test}$

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 $\mathbf{z}_i$



$\mathbf{z}_i$ task descriptor

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

- 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

Multi-Task Learning

- Problem statement
- Models, objectives, optimization
- Challenges
- Case study of real-world multi-task learning

Transfer Learning

- Pre-training & fine-tuning

Goals for by the end of lecture:

- Know the key design decisions when building multi-task learning systems
- Understand the difference between multi-task learning and transfer learning
- Understand the basics of transfer learning

Multi-Task Learning vs. Transfer Learning

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 $\mathcal{T}_a$

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

Side note: $\mathcal{T}_a$ may include multiple tasks itself.

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

Question: In what settings might transfer learning make sense?
(answer in chat or raise hand)

Transfer learning via fine-tuning

pre-trained parameters

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

training data for new task

(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

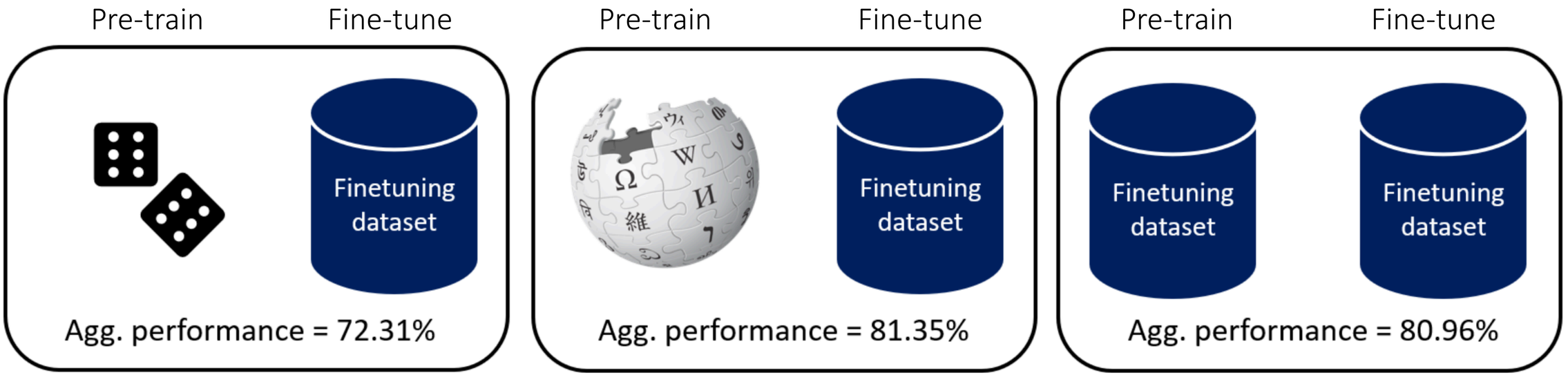
Pre-trained models often available online.

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)

When might this common knowledge break?

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



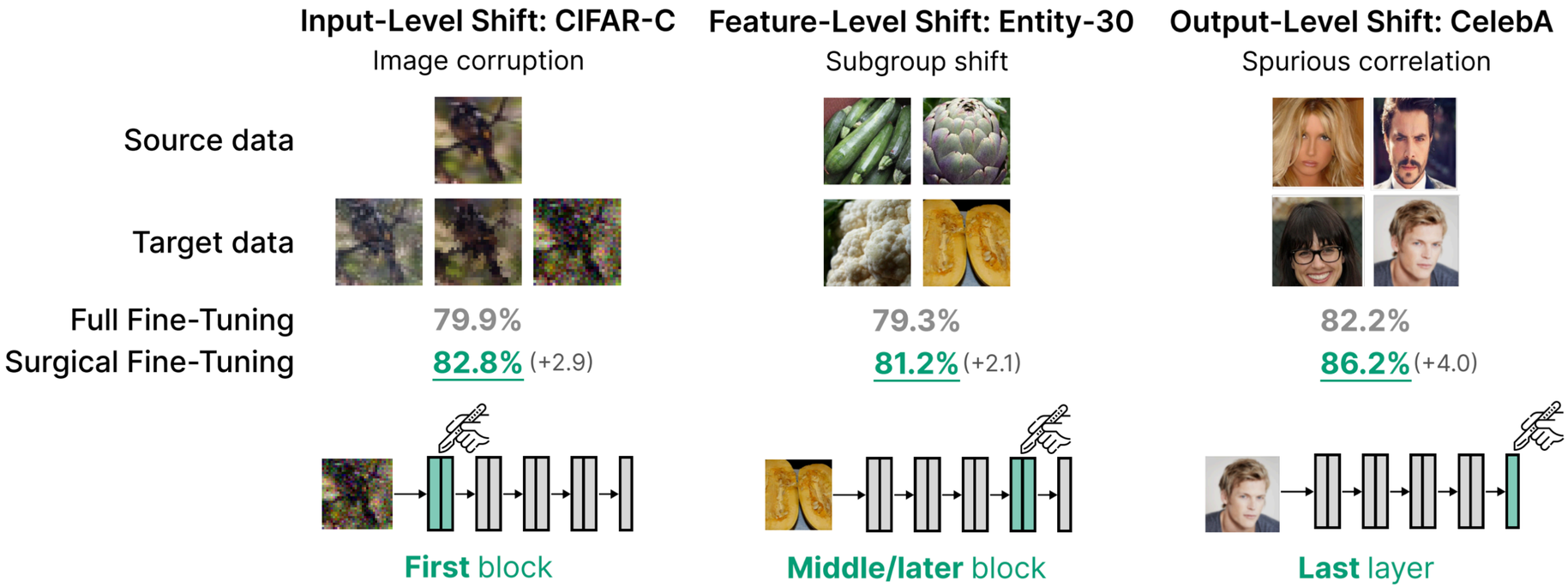
Krishna, Garg, Bingham, Lipton. Downstream Datasets Make Surprisingly Good Pretraining Corpora. arXiv 09/28/22.

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?
3. For fine-tuning to low-level image corruptions, maybe the first layer might be better?

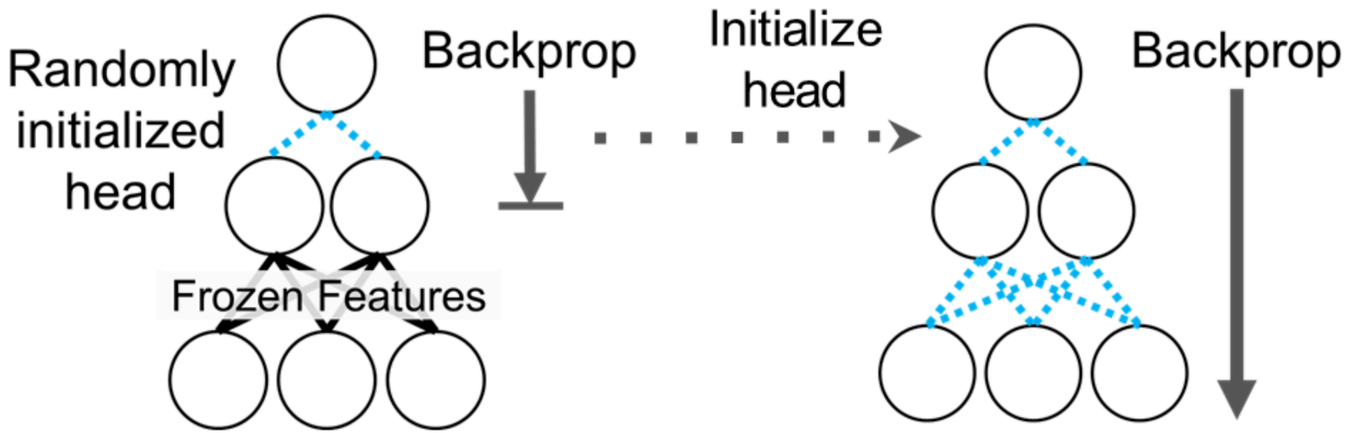
Result: Fine-tuning the first or middle layers can work better than the last layers.



Lee*, Chen*, Tajwar, Kumar, Yao, Liang, Finn. Surgical Fine-Tuning Improves Adaptation to Distribution Shifts. Openreview 09/28/22.

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?

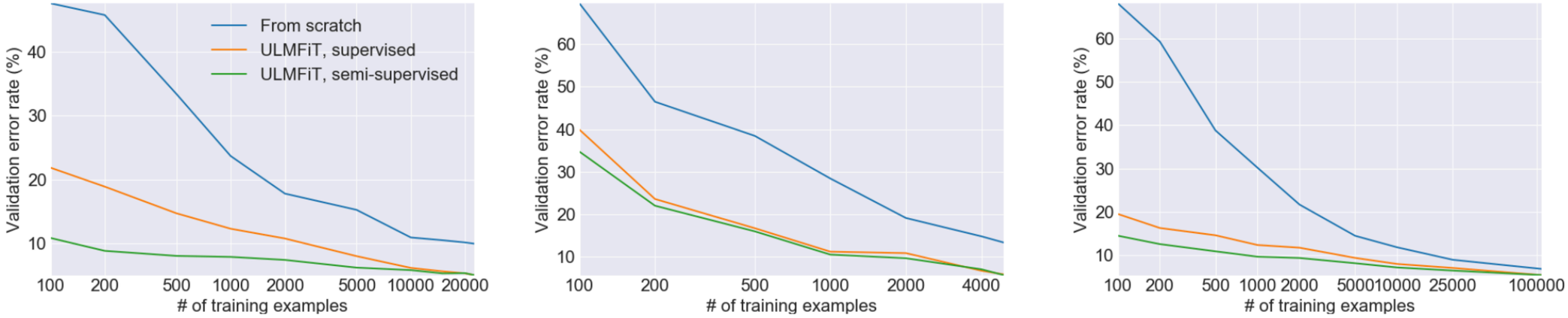


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

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.

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

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

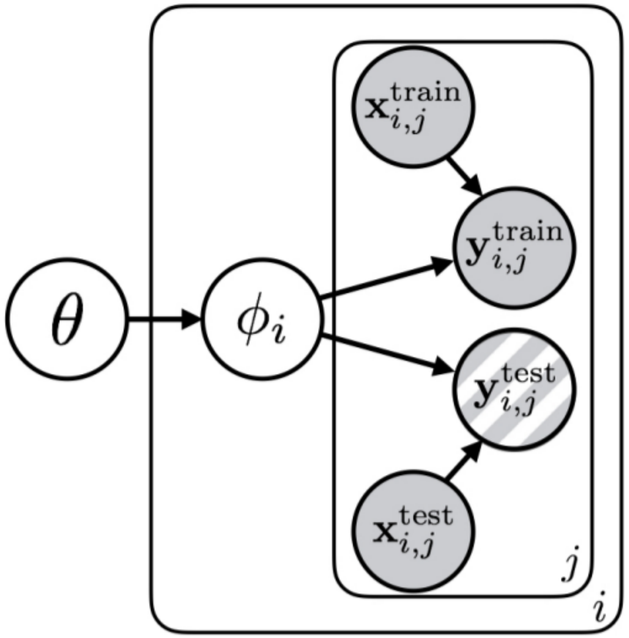
Probabilistic view

- 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

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. $\mathscr{H}(p(\phi_i \mid \theta)) < \mathscr{H}(p(\phi_i))$

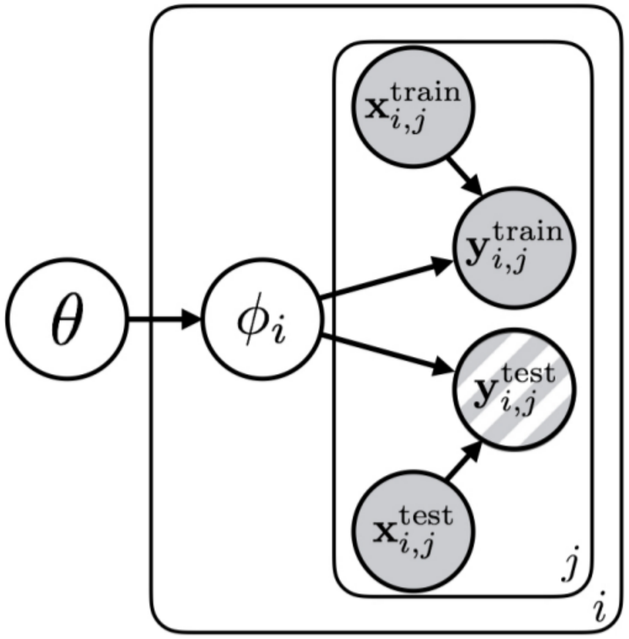
Thought exercise #1: If you can identify θ (i.e. with meta-learning), when should learning ϕ_i be faster than learning from scratch?

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

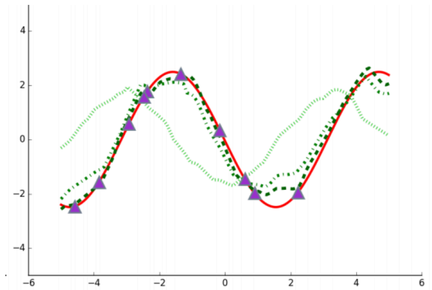
How would Bayes view it?

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

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

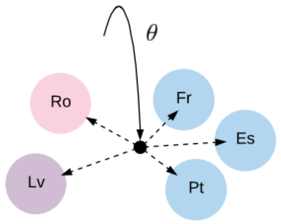


What information might θ contain…



…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 $\mathcal{D}_{\text{train}}$

Classify new examples

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

How does meta-learning work? An example.

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

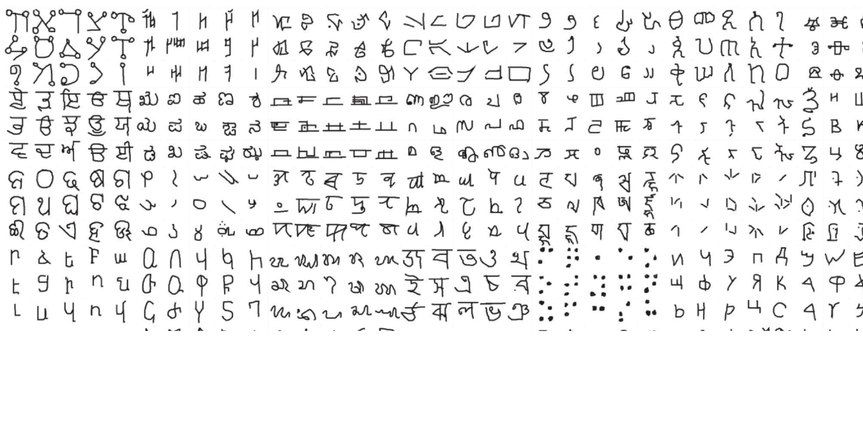
Key assumption: meta-training tasks and meta-test task drawn i.i.d. from same task distribution

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

Like before, tasks must share structure.

What do the tasks correspond to?

- recognizing handwritten digits from different languages (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)

Problem Settings Recap

Multi-Task Learning

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

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

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.

Lecture Recap

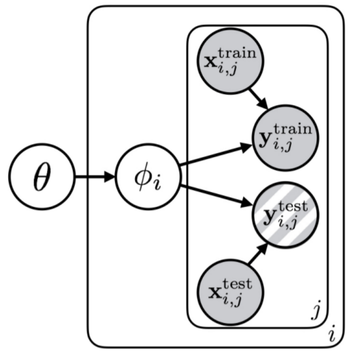
How can you transfer things learned from one task to another?

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.

Towards understanding transfer learning and its limits

Hanie Sedghi

Google Research, Brain Team

Science of Deep Learning

- Our understanding of modern neural networks lags behind their practical successes. This growing gap poses a challenge to the pace of progress.
- Although there has been some progress in this area, still we are far from answering many fundamental questions such as generalization capabilities of deep models and how to ensure successful transfer to new domains.
- I believe this understanding helps us extend beyond our current use of deep learning in a reliable way.

Science of Deep Learning

- Our understanding of modern neural networks lags behind their practical successes. This growing gap poses a challenge to the pace of progress.
- Although there has been some progress in this area, still we are far from answering many fundamental questions such as generalization capabilities of deep models and how to ensure successful transfer to new domains.
- I believe this understanding helps us extend beyond our current use of deep learning in a reliable way.
- Principled approaches to investigate deep learning phenomena.
- To understand when and why DNNs generalize, improve training and generalization performance in state of the art deep learning models and extend the current success of our models to new domains.

Part I

What is being transferred in transfer learning?

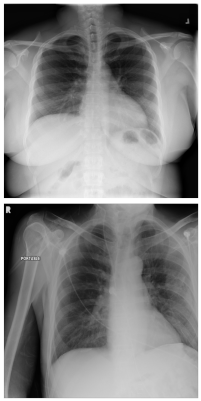
What is being transferred in transfer learning?

- One desired capability of machines is to transfer their knowledge or understanding of a domain it is trained on (source domain) to another domain (target domain) where data is (usually) scarce or a fast speed of convergence is needed.
- Plethora of works using transfer learning in different applications.
- We would like to understand:
  - what enables a successful transfer?
  - which parts of the network are responsible for that?

What is being transferred in transfer learning?, B. Neyshabur*, H. Sedghi*, C. Zhang* , NeurIPS 2020

Problem Setup

- Target domains that are intrinsically different and diverse:
  - CheXpert: a medical imaging dataset of chest x-rays considering 5 different diseases.
  - DomainNet: designed to probe transfer learning for diverse visual representations. The domains range from real images to sketches, clipart and painting samples. 345 classes
- Two initialization scenarios:
  - Pre-trained on ImageNet (Finetune)
  - Start from random initialization (RandInit)

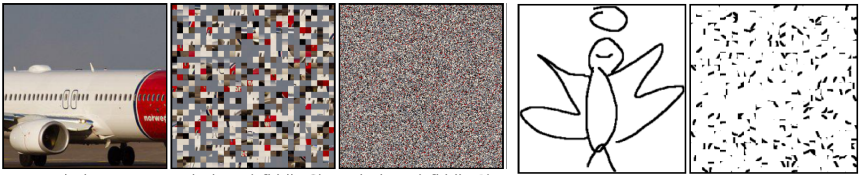


Role of feature reuse

- Comparing learning curves
  - Largest performance boost on the `real` domain, which contains natural images.
  - Even for the most distant target domains, we still observe performance boosts from transfer learning.
  - The optimization for Finetune also converges much faster than Randinit in all cases.
- The benefits of transfer learning are generally believed to come from reusing the pre-trained feature hierarchy.
- But, why in many successful applications of transfer learning, the target domain could be visually very dissimilar to the source domain?

Role of feature reuse

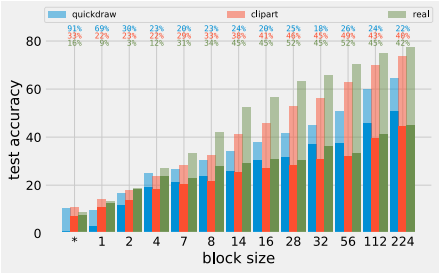
Experiment: We partition the image of the downstream tasks into equal sized blocks and shuffle the blocks randomly. The shuffling disrupts visual features in those images.

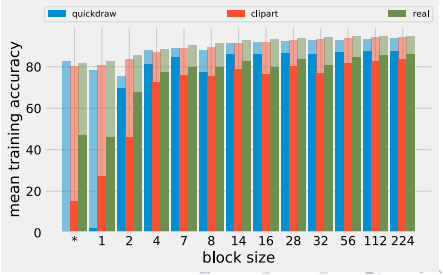


Role of feature reuse

- Feature reuse plays a very important role!
  especially when the downstream task shares similar visual features with the pre-training domain.

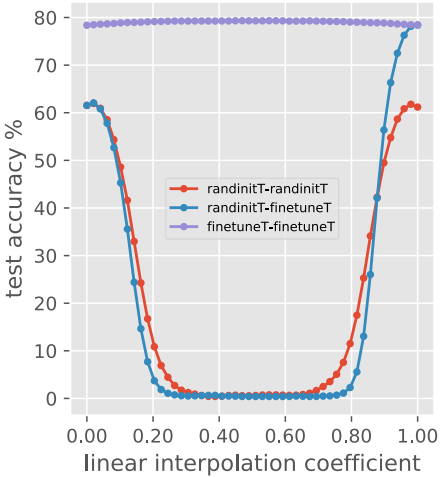
- There are other factors at play!
  low-level statistics of the data that are not ruined in the shuffling lead to the significant benefits of transfer learning, especially on optimization speed.





Performance barriers in the loss landscape

- *Any* two minimizers of a deep network can be connected via a *non-linear* low-loss path.
- We evaluate a series of models along the linear interpolation of the two weights.
- Performance barriers are generally expected between two unrelated NN models.
- When the two solutions belong to the same flat basin of the loss landscape, performance barrier is absent.
- Finetune models reside in the same basin.
- RandInits end up in a different basin, even if starting from same random seed.



Performance barriers in the loss landscape

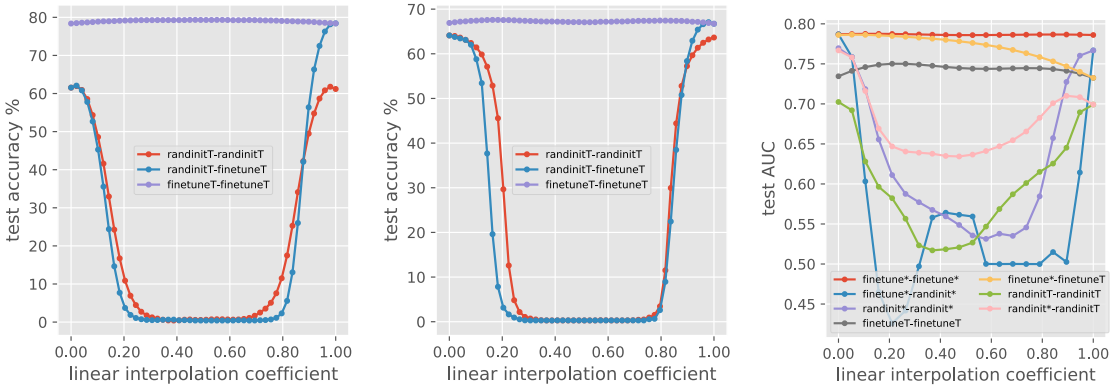
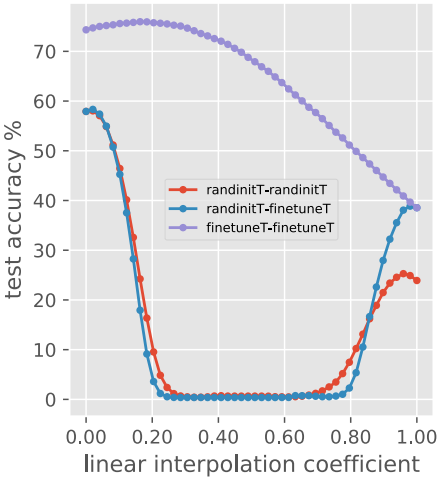


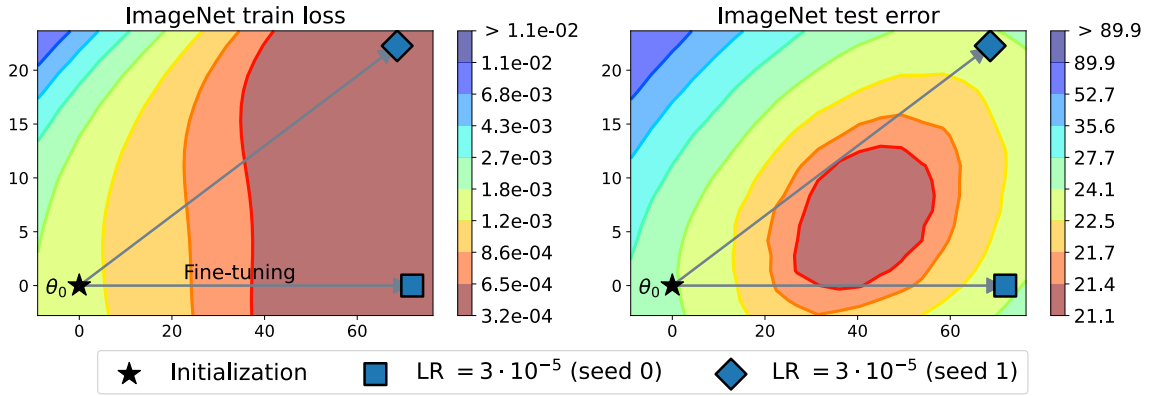
Figure: The left and middle panes show performance barrier measured by test accuracy on DomainNet `real` and `quickdraw`, respectively. The right pane shows the performance barrier measured by test AUC on CheXpert.

Cross-domain weight interpolation on DomainNet

- when directly evaluated on a different domain that the models are trained from, we could still get non-trivial test performance.
- P-T consistently outperforms RI-T even in the cross-domain cases.
- when interpolating between P-T models, (instead of performance barrier) we observe performance boost in the middle of the interpolation.
- This suggests that all the trained P-T models on all domains are in one shared basin.



Model Soups



Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, Wortsman et al 2022

Model Soups

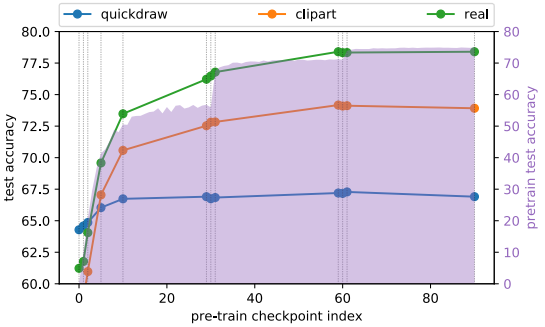
Method	ImageNet			Distribution shifts					
	Top-1	ReaL	Multilabel	IN-V2	IN-R	IN-Sketch	ObjectNet	IN-A	Avg shifts
ViT/G-14 (Zhai et al., 2021)	90.45	90.81	–	83.33	–	–	70.53	–	–
CoAtNet-7 (Dai et al., 2021)	90.88	–	–	–	–	–	–	–	–
Our models/evaluations based on ViT-G/14:									
ViT/G-14 (Zhai et al., 2021) (reevaluated)	90.47	90.86	96.89	83.39	94.38	72.37	71.16	89.00	82.06
Best model on held out val set	90.72	91.04	96.94	83.76	95.04	73.16	78.20	91.75	84.38
Best model on each test set (oracle)	90.78	**91.78**	**97.29**	**84.31**	95.04	73.73	**79.03**	92.16	84.68
Greedy ensemble	**90.93**	91.29	**97.23**	**84.14**	94.85	73.07	77.87	91.69	84.33
Greedy soup	**90.94**	91.20	**97.17**	**84.22**	**95.46**	**74.23**	78.52	**92.67**	**85.02**

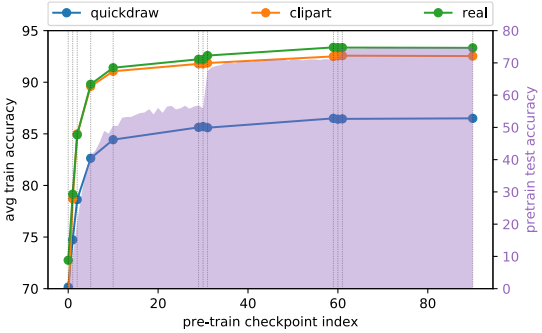
- No barrier between different fune-tuned model $\rightarrow$ possible to combine fine-tuned models by interpolating their weights.
- Simply averaging the weights of multiple models fine-tuned with different hyperparameters can improve performance
- Achieving most of the accuracy gain of ensembling outputs without any added computational cost at inference time.

Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, Wortsman et al 2022

Which pre-trained checkpoint is most useful for transfer learning?

- Significant improvements are observed when we start from the checkpoints where the pre-training performance has been plateauing.
- Independence between the improvements on optimization speed and final performance.
- You can start from earlier checkpoints in pre-training.





Not all layers are created equal!

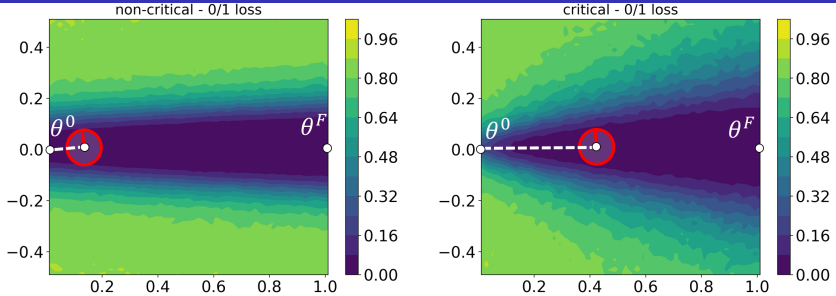
Experiment:

- Consider a deep neural network (at any training epoch).
- Pick one of the layers and rewind its value back to its value at initialization.
- Keep the value of all other layers fixed.
- Notice the change in performance.

Observation: In a deep neural network, some modules are more critical than others, i.e., rewinding their parameter values back to initialization, while keeping other modules fixed at the trained parameters, results in a large drop in the network's performance.

C. Zhang, S. Bengio, Y. Singer, *Are all layers created equal?*, Feb 2019

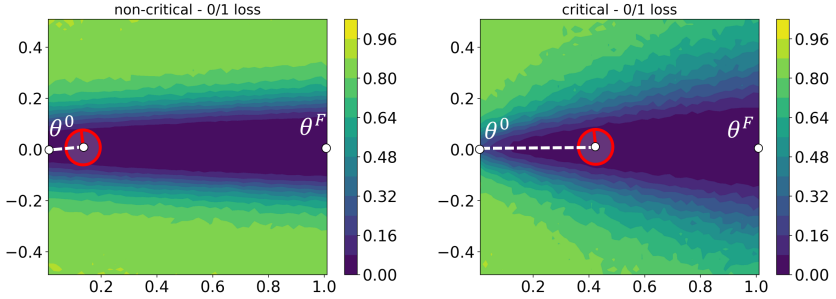
Role of different layers: Module Criticality



- Loss values in the valleys that connect the initial weights θ^0 to the final weights θ^F.
- Module criticality: how far one can push the ball of radius r in the valley towards initialization divided by the radius.

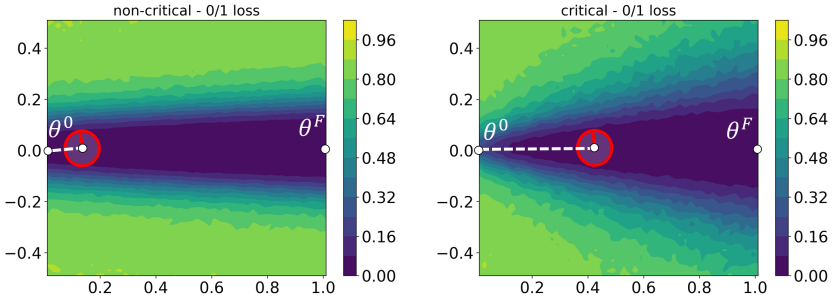
The intriguing role of module criticality in the generalization of deep networks,
N. Chatterji, B. Neyshabur, H. Sedghi, Spotlight in ICLR2020

Module Criticality



- Non-critical modules $\equiv$ wide valley
  Critical modules $\equiv$ sharp valley

Module Criticality

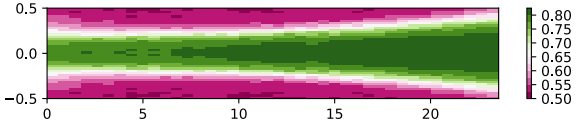


- Non-critical modules $\equiv$ wide valley
  Critical modules $\equiv$ sharp valley
- Module criticality as a generalization measure correlates well with model performance.

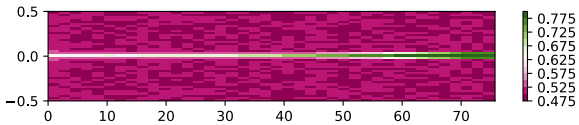
The intriguing role of module criticality in the generalization of deep networks,
N. Chatterji, B. Neyshabur, H. Sedghi, Spotlight in ICLR2020

Role of different layers

- As we move from the input towards the output, we see tighter valleys, i.e., modules become more critical.
- This is in agreement with observation of [Yosinski+2014, Raghu+2019] that lower layers are in charge of more general features while higher layers have features that are more specialized for the target domain.



(g) Module Criticality Layer1



(h) Module Criticality Layer4

So far...

- Both feature-reuse and low-level statistics of the data are important.
- There is no performance barriers between finetune models, while models trained from random initialization are in a different basins in the loss landscape.
- Lower layers are in charge of general features and higher layers are more sensitive to perturbation of their parameters.

What is being transferred in transfer learning?, B. Neyshabur*, H. Sedghi*, C. Zhang* , NeurIPS 2020

Part II

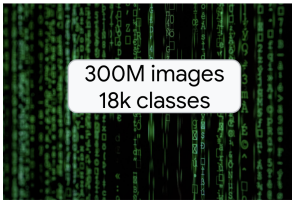
Exploring the limits of large scale pre-training

Effect of Scale: A prelude

- Recent impressive progress on transfer and few-shot learning: scaling up model and data
- Prominent examples: GPT-3, CLIP
- Massive datasets: Instagram images and JFT-300

IMAGENET

"JFT-300M"
Google internal dataset



Effect of scale: current narrative

- These developments implicitly encourage two consistent views:
  1. Scaling up the model and data size improves the performance significantly;
  2. The performance improvement transfers to downstream tasks in a desirable way.
- Non-saturating performance.
- Linear relationship between imagenet pre-training and downstream accuracy.

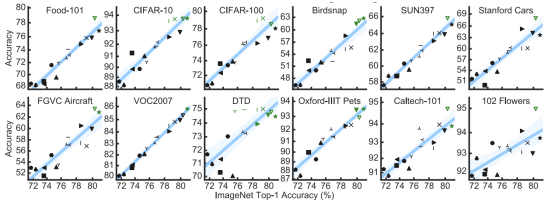


Figure: Kornblith et al, 2019

Shortcomings of earlier works

- Performance for different choices of hyper-parameter values are not reported.
- When studying scaling, we are concerned about the best performance of models given all possible values for the hyper-parameters!
- Limited accuracy range

Shortcomings of earlier works

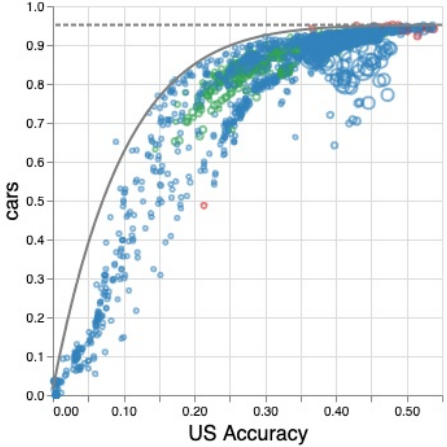
- Performance for different choices of hyper-parameter values are not reported.
- When studying scaling, we are concerned about the best performance of models given all possible values for the hyper-parameters!
- Limited accuracy range
- Focusing on improving SOTA and limited computational budget.
- Simply extrapolating scaling without understanding of the dynamics of scaling can be detrimental.

This work

- Systematic large scale study
- Investigate the transferability of improvements on a large-scale upstream task to a wide range of downstream tasks.
- More than 4800 experiments
- Image recognition task
- Vision Transformers, ResNets, Mixers of varying size (ten million to ten billion parameters)
- Trained on the largest scale of available image data (JFT, ImageNet21k)
- More than 20 downstream tasks
- Downstream tasks cover a wide range of standard datasets, e.g., VTAB, MetaDataset, Wilds and medical imaging.

Setting

- Goal: Predict downstream performance for a given model.
- DS-vs-US accuracy plot.
- Horizontal line = Predicted accuracy as US accuracy becomes 1.



Recap: Convex hull

Definition (Convex hull)

A convex hull $\mathcal{C}$ of N points a_j in a set $\mathcal{S}$ is given by

$$\mathcal{C} \equiv \left\{ \sum_{j=1}^{N} p_j a_j : p_j \geq 0 \quad \text{for all j,} \quad \sum_{j=1}^{N} p_j = 1 \right\}.$$

Recap: Convex hull

Definition (Convex hull)

A convex hull $\mathcal{C}$ of N points a_j in a set $\mathcal{S}$ is given by

$$\mathcal{C} \equiv \left\{ \sum_{j=1}^{N} p_j a_j : p_j \geq 0 \text{ for all j, } \sum_{j=1}^{N} p_j = 1 \right\}.$$

Lemma

Consider a group of models $\theta_j, j \in [N]$ that reaches accuracy $a_j = (a_j^{US}, a_j^{DS}), j \in [N]$ on some pair of tasks (US,DS). Construct a randomized model $\tilde{\theta}$ as follows: for each input x_i, with probability p_j pick model θ_j and output $\theta_j(x_i)$. Then the randomized model will demonstrate accuracy $\sum_{j=1}^{N} p_j a_j$.

Choice of data for fitting the power law

Lemma

Consider a group of models $\theta_j, j \in [N]$ that reaches accuracy $a_j = (a_j^{US}, a_j^{DS}), j \in [N]$ on some pair of tasks (US,DS). Construct a randomized model $\tilde{\theta}$ as follows: for each input x_i, with probability p_j pick model θ_j and output $\theta_j(x_i)$. Then the randomized model will demonstrate accuracy $\sum_{j=1}^{N} p_j a_j$.

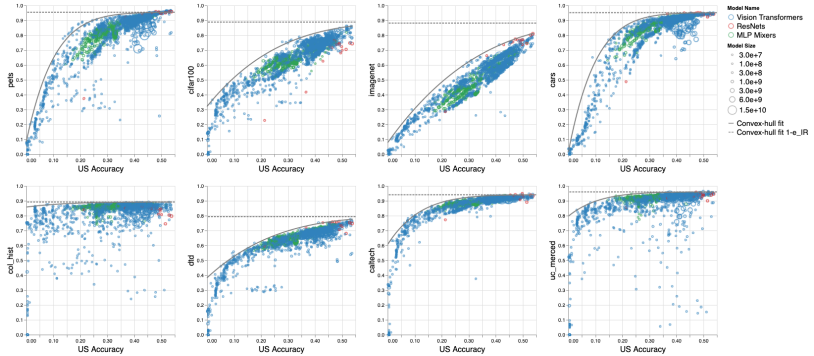
$\Rightarrow$ We consider the convex hull of the points in our analysis.

Large variance in DS-vs-US performance across models.

1. fit the existing points $\rightarrow$ fit average performance
2. fit the convex hull $\rightarrow$ fit best performing model. Robust to density of the points.

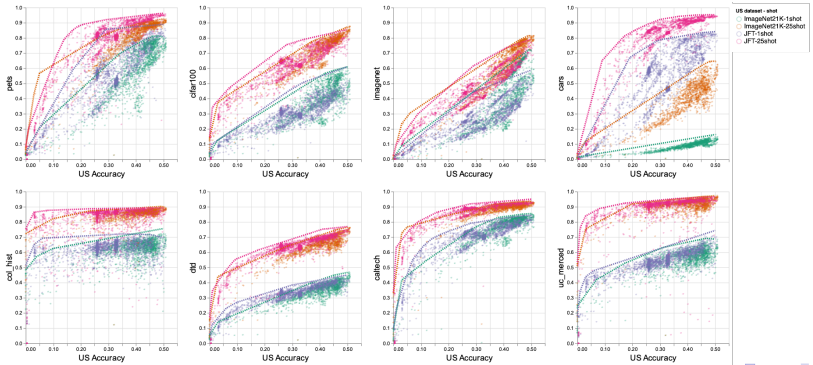
The diminishing benefit of scaling up in transfer learning

- Goal: Predict downstream performance for a given model.
- DS-vs-US accuracy plot.
- Saturation: even if US reaches accuracy of one, DS won't.
- Nonlinear relationship.



The diminishing benefit of scaling up in transfer learning

- DS-vs-US accuracy plot.
- Saturation
- Nonlinear relationship.
- Consistent across different US tasks & No. of shots.

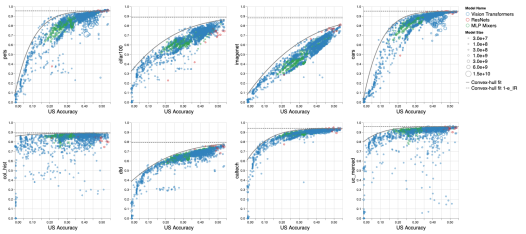


Scaling laws for downstream accuracy

- Goal: Predict DS performance
- Our proposed model

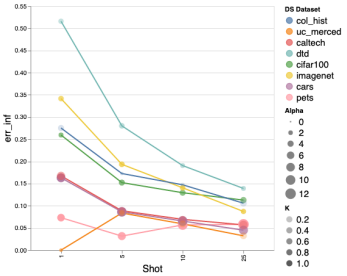
$$e_{DS} = k(e_{US})^{\alpha} + e_{\infty}$$

- e_{∞}
  - irreducible error.
  - captures the value of DS error if US error reaches zero.
  - captures the nonlinearity.
  - is not the Bayes error.



Effect of design choices on power law parameters

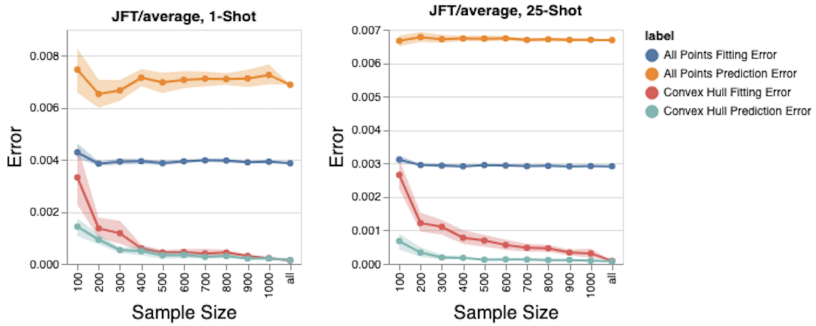
- k, e_∞ correlate negatively with number of shots.
- α is positively correlated with number of shots.
- Correlation values change drastically for different choices of US, DS tasks.



DS	US	Parameter	Correlation with Number of Shots
caltech	ImageNet21K	K	-0.777892
caltech	ImageNet21K	α	-0.582066
caltech	ImageNet21K	e_∞	-0.845368
caltech	JFT	K	-0.620526
caltech	JFT	α	0.259305
caltech	JFT	e_∞	-0.762856
cars	ImageNet21K	K	0.720391
cars	ImageNet21K	α	0.960490
cars	ImageNet21K	e_∞	-0.737273
cars	JFT	K	-0.976599
cars	JFT	α	-0.034033
cars	JFT	e_∞	-0.809016
cifar100	ImageNet21K	K	-0.918914
cifar100	ImageNet21K	α	0.683485
cifar100	ImageNet21K	e_∞	-0.587304
cifar100	JFT	K	-0.934455
cifar100	JFT	α	0.707966
cifar100	JFT	e_∞	-0.754030

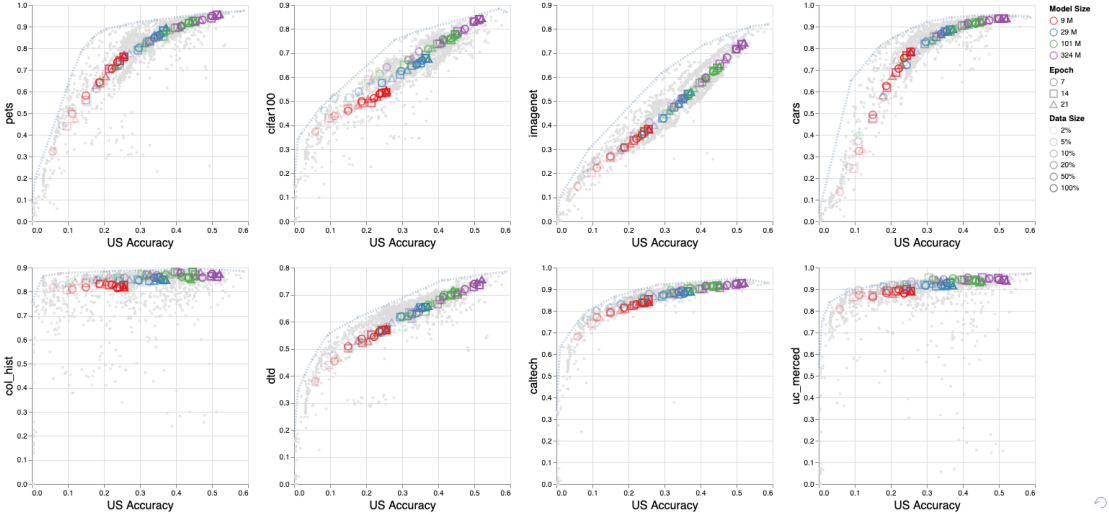
Sample size sensitivity analysis

- The prediction error is very small across all these choices.
- The proposed model will work well even when we have much smaller number of DS-vs-US points.
- The fitting error decreases by increasing the number of samples.



Effect of scale: A closer look

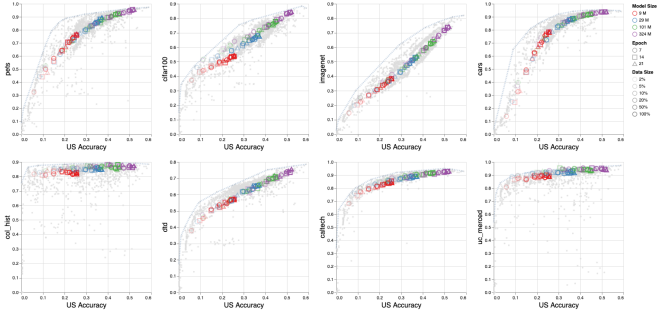
Controlled experiments : model size, data size, compute



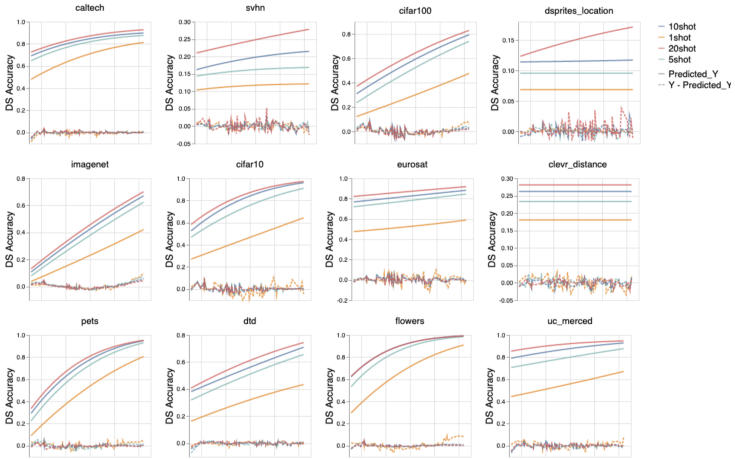
Effect of scale: A closer look

Controlled experiments : model size, data size, compute

- Same pattern.
- Same curve for the 3 parameters.
- Grid search equivalence.
- Variation is due to training hyper-parameters.



On the prediction power of US accuracy



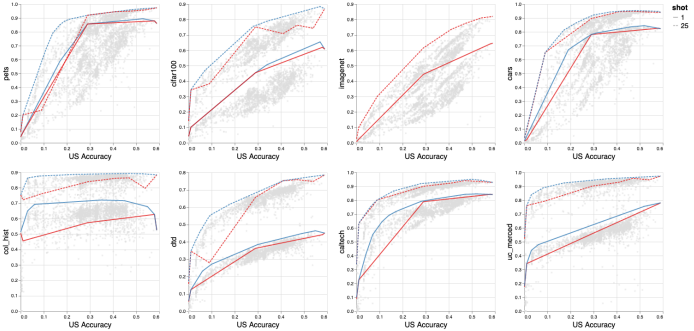
DS	error std
birds	0.1542
caltech	0.1020
cars	0.1979
col_hist	0.1552
dtd	0.0885
imagenet	0.1882
pets	0.1412
uc_merced	0.1581

Conditioned on US accuracy, not much is left for the rest of parameters altogether to predict!

Investigating different DS-vs-US trends

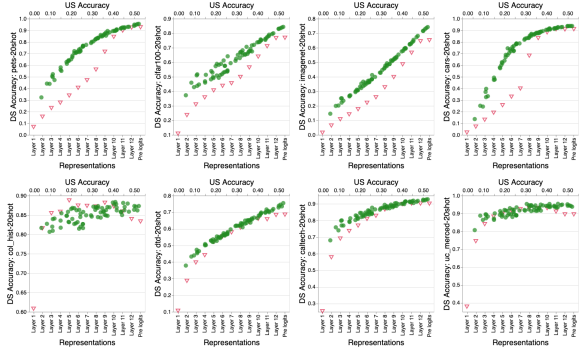
Overlay the convex hull of ImageNet DS-vs-US plot on all DS tasks. Observation

1. Best performing ImageNet models perform very similarly to best performing models in several but not all DS tasks.
2. As the US performance increases, the gap between best performing ImageNet models and best performing DS task models reduces significantly.



Investigating different DS-vs-US trends: Experiment

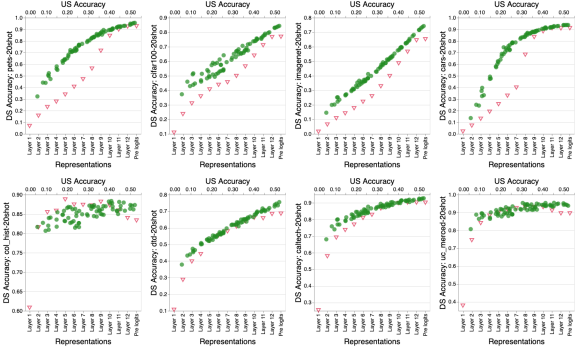
Experiment: Move the head to different layers



- DS versus US performance
- DS performance for representation taken from specific layer

Investigating different DS-vs-US trends: Experiment

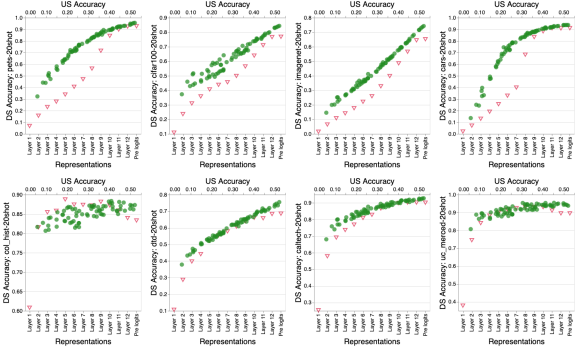
Experiment: Move the head to different layers



- DS versus US performance
- DS performance for representation taken from specific layer
- Plots show similar trend.
- For DS that saturate faster, higher layers are not needed.
- Lower layers capture lower level features that are more common across different dataset and tasks, whereas fine-grained features reside at top layers in the network

Investigating different DS-vs-US trends: Experiment

Experiment: Move the head to different layers

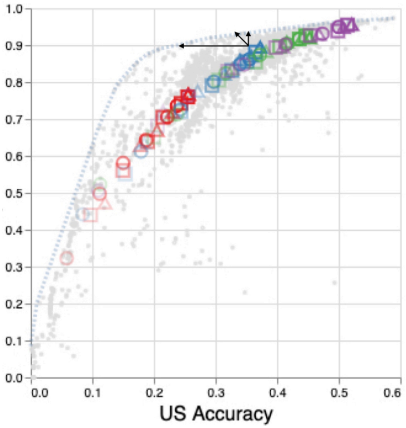


- DS versus US performance
- DS performance for representation taken from specific layer
- Plots show similar trend.
- For DS that saturate faster, higher layers are not needed.
- Lower layers capture lower level features that are more common across different dataset and tasks, whereas fine-grained features reside at top layers in the network

We need data diversity.

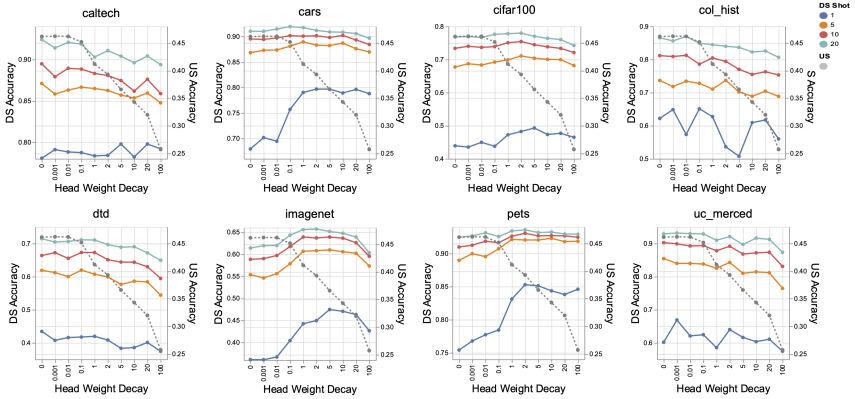
Discrepancies between US and DS performances: a case study

- Recap:
  training hyper-parameters cause variation from the curve.
- Now:
  focus on effect of head hyper-parameters.
- Decouple head from rest of network.
- weight decay, learning rate.



Effect of head weight decay

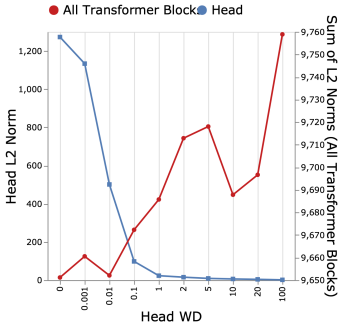
- Increasing head WD hurts US performance.
- Increasing head WD improves DS performance for some tasks.



Discrepancies between US and DS performances: why?

Increasing head WD

- decreases head margin, increases layer margin [Elsayed et al 2018].
- decreases head norm, increases layer norm for lower layers.
- pushes the information down to lower layers.



On generalization of the observed phenomena

- Number of shots
- Transfer vs. few-shot
- Scaling of plots: logit$(p) = \log(\frac{p}{1-p})$, $-\log(1-p)$, linear
- Architecture

Summary

- Large-scale systematic study.
- Performance saturation on DS does happen.
- Modeled DS-vs-US accuracy and predict DS accuracy by a power law curve.
- Our model predicts saturation point and is robust to low sample size.
- Data diversity matters.
- Scaling model size, pre-training data size, compute leads to the same curve.
- US performance has high prediction power.
- Hyper-parameters used in training matter and need to be DS-specific.
- Head hyper-parameters are important and can help improve DS performance.

Zooming in on the role of data

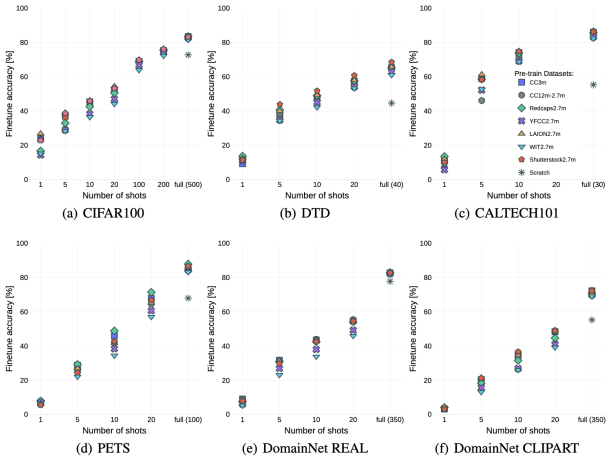
- Pre-training: CLIP, SimCLR
- Architecture: ResNet50
- 4000 trained networks.
- 7 upstream, 9 downstream datasets
- Downstream: CIFAR100, DTD, CALTECH101, PETS, Domainnet REAL, Domainnet CLIPART CameraTraps, Cassava Leaf Disease, EuroSAT

Dataset	Source	Total size
YFCC	Flickr	14,826,000
LAION	Common Crawl	15,504,742
CC-12M	Unspecified web pages	9,594,338
RedCaps	Reddit	11,882,403
WIT	Wikipedia	5,038,295
ShutterStock	ShutterStock	11,800,000
IN1K-Captions	ImageNet	463,622

The role of pretraining data in transfer learning, R. Entezari, M. Wortsman, O. Saukh, M. Shariatnia, H. Sedghi, Ludwig Schmidt, In submission

Role of pretraining data distribution

- CLIP
- the number of pretraining images is 2.7 million.
- Shutterstock is the best performing pre-training datasets
- pre-training dataset is important for low-shot transfer.

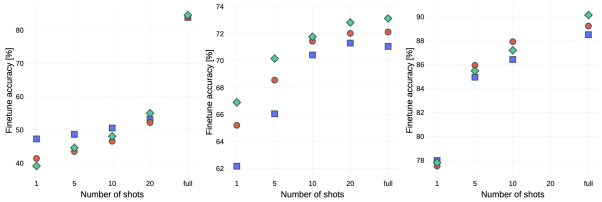


(a) CIFAR100 (b) DTD (c) CALTECH101

(d) PETS (e) DomainNet REAL (f) DomainNet CLIPART

Role of pretraining data distribution

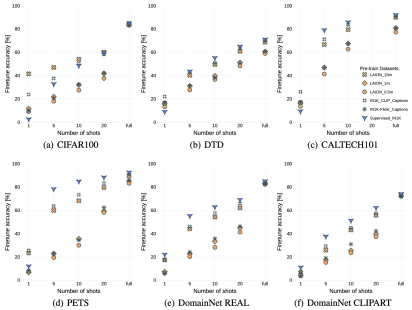
- Same setting as before
- Only change pretraining method to SimCLR
- Observe similar phenomena



(a) CIFAR100 (b) DTD (c) CALTECH101

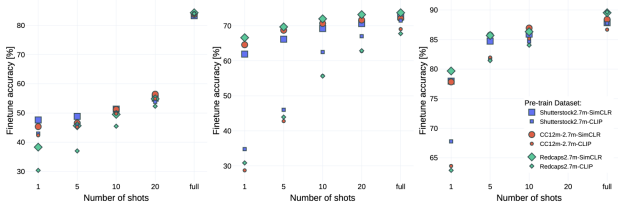
Role of data curation

- Recap:
  training hyper-parameters cause variation from the curve.



Role of pretraining method

- Contrastive > supervised for low shot setting
- image-image contrastive > image-text contrastive

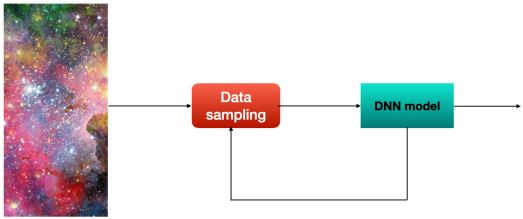


(a) CIFAR100 (b) DTD (c) CALTECH101

What's next?

Data sampling module

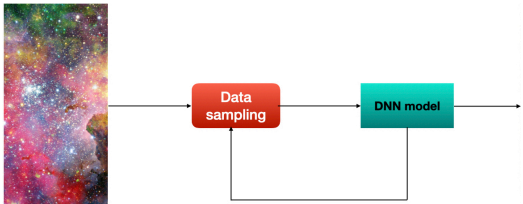
- Modeling data
- Ensuring data diversity
- Closing the loop
- Investigating the effect of curriculum learning



What's next?

Data sampling module

- Modeling data
- Ensuring data diversity
- Closing the loop
- Investigating the effect of curriculum learning



Thank you!