Lecture 9: Multi-Task and Transfer Learning

Multi-Task and Transfer Learning

- Many datasets and tasks have strongly overlap
- Many features are re-usable across tasks
- Leveraging these similarities can improve performance on new tasks with limited samples
- Humans adapt rapidly (in terms of samples complexity) to constantly changing environments and tasks

a) Transferring Learned knowledge from Java to Python.

b) Transferring Learned knowledge Table Tennis to Tennis.

Image Credit: Azin Asgarian

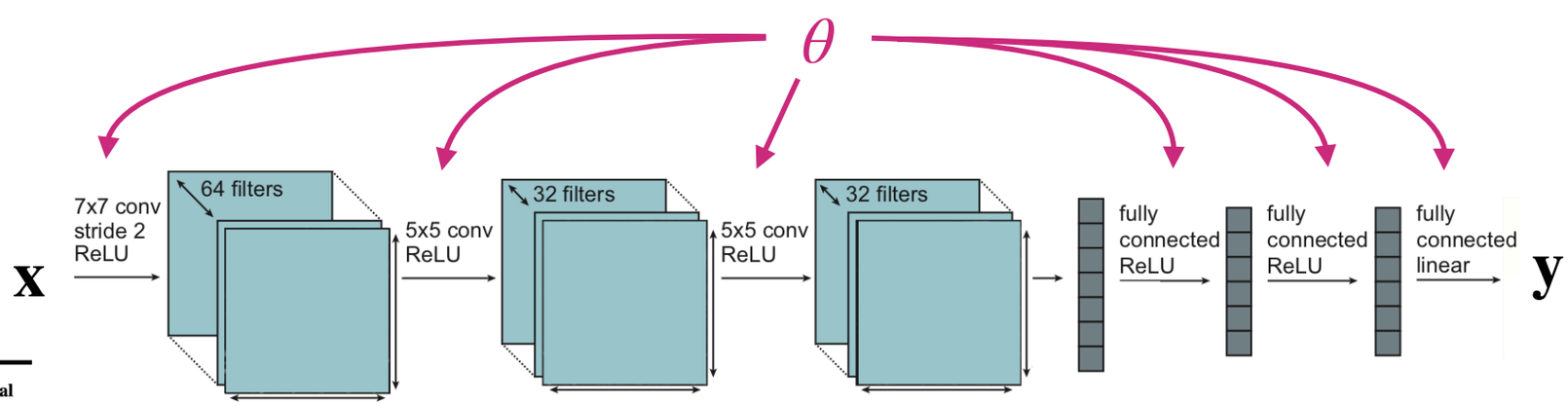
Slide Credits

- Part of this lecture slides (generously) from Chelsea Finn
- Some notation differences

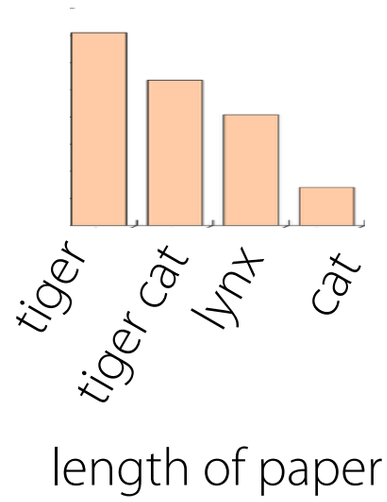
Notations and Definitions

Some notation

ImageNet Classification with Deep Convolutional Neural Networks



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



length of paper

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

Slides Credit: Chelsea Finn

Examples

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

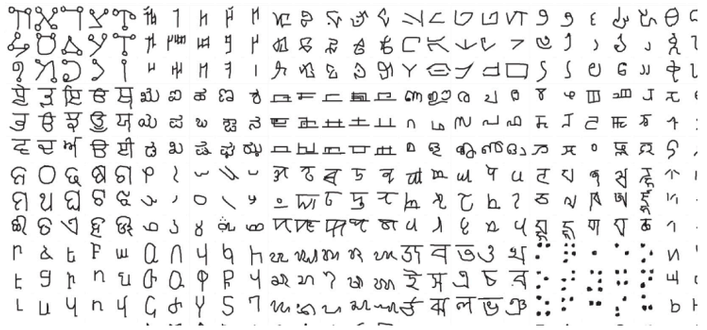
will use $\mathcal{D}_i$ as shorthand for $\mathcal{D}_i^{tr}$:

This is often called just multi-task in the literature

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. CelebA attribute recognition

e.g. scene understanding

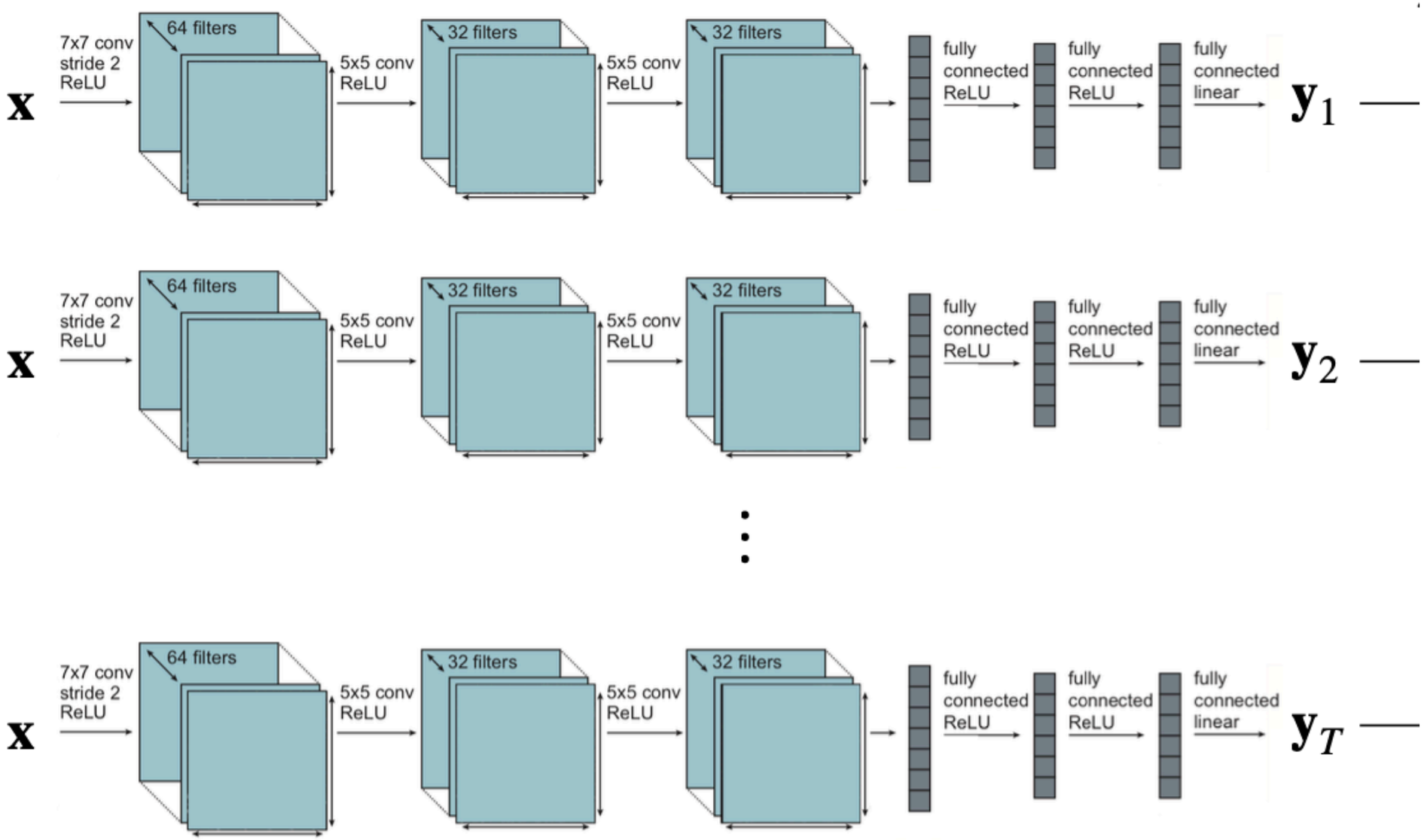
$$L_{\text{tot}} = w_{\text{depth}} L_{\text{depth}} + w_{\text{kpt}} L_{\text{kpt}} + w_{\text{normals}} L_{\text{normals}}$$

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

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

Slides Credit: Chelsea Finn

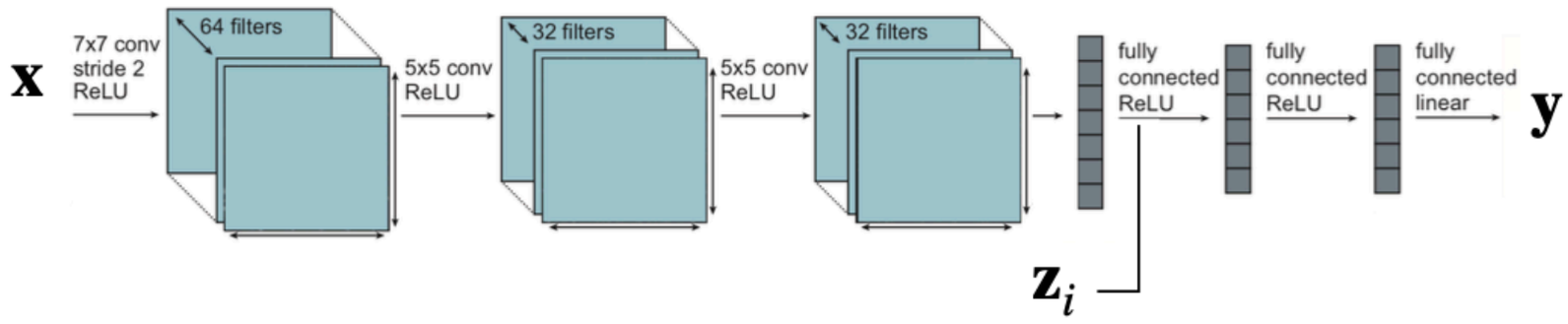
Simple Multi-Task Model



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

Slides Credit: Chelsea Finn

Classic Multi-Task Model



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)

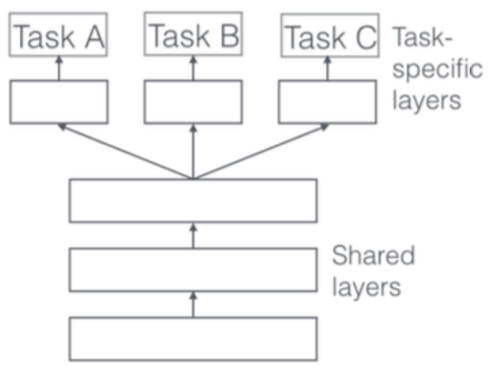
Slides Credit: Chelsea Finn

Multitask Learning

Rich Caruana

23 September 1997

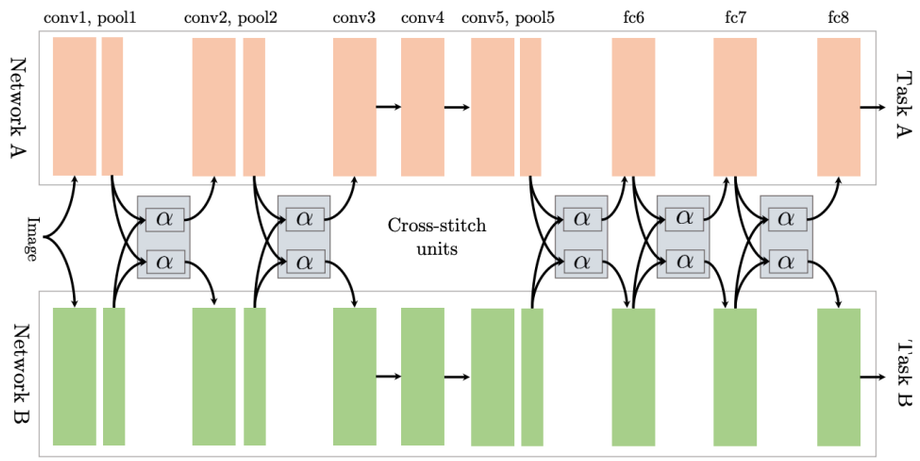
Rich Caruana



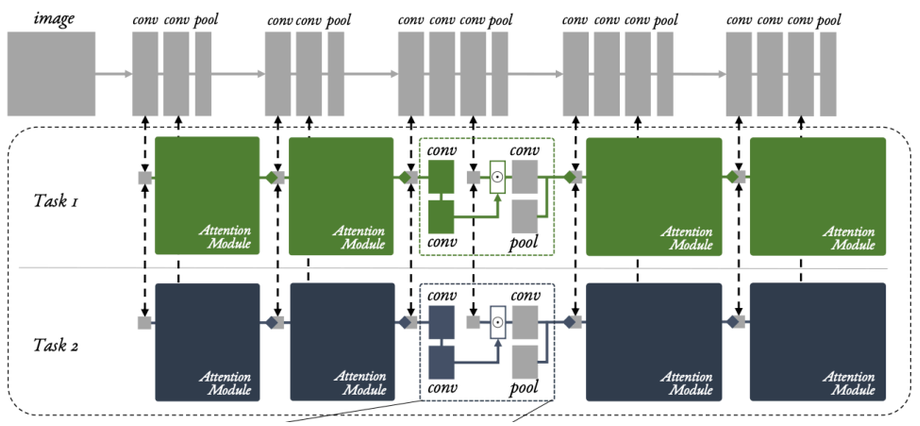
Task heads

Classic Multi-Task Model

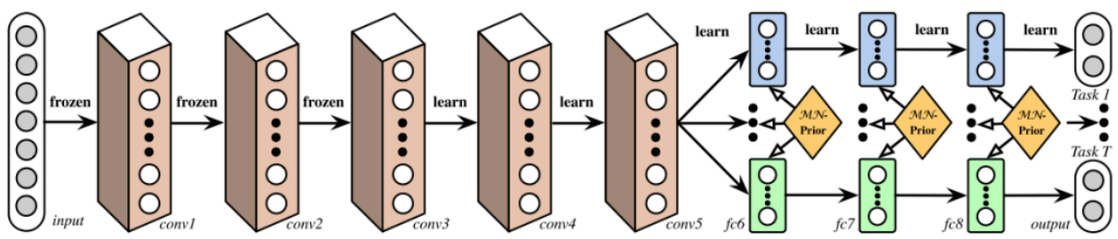
Conditioning: More Complex Choices



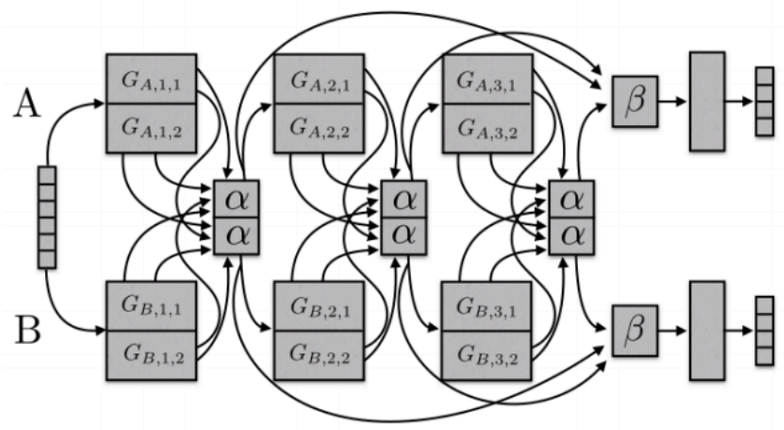
Cross-Stitch Networks. Misra, Shrivastava, Gupta, Hebert '16



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



Deep Relation Networks. Long, Wang '15



Sluice Networks. Ruder, Bingel, Augenstein, Sogaard '17

Various Degrees of Sharing and Independence

Slides Credit: Chelsea Finn

Weighting each differently

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?

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

a. various heuristics

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

b. use task uncertainty

(e.g. see Kendall et al. CVPR 2018)

c. aim for monotonic improvement towards Pareto optimal solution

(e.g. see Sener et al. NeurIPS 2018)

Slides Credit: Chelsea Finn

Negative Transfer

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

Slides Credit: Chelsea Finn

Overfitting

Challenge #2: Overfitting

You may not be sharing enough!

Multi-task learning <-> a form of regularization

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

Solution: Share more.

Slides Credit: Chelsea Finn

Transfer Learning

- In some cases we have a single problem “task” we are concerned with and we want to use related data and tasks to improve it
- Multi-task learning can still help but ..
- As datasets and tasks become huge it might be impractical to do multi-task learning
  - e.g. 100 image dataset, multi-task learning would involve
- Challenges of negative transfer and overfitting add extra cumbersome parameters
- Fortunately representation from large scale deep learning models are often useful for other datasets

Transfer Learning

Multi-Task Learning vs. Transfer Learning

Multi-Task Learning

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

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

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)

Slides Credit: Chelsea Finn

Transfer learning via fine-tuning

pre-trained parameters

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

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

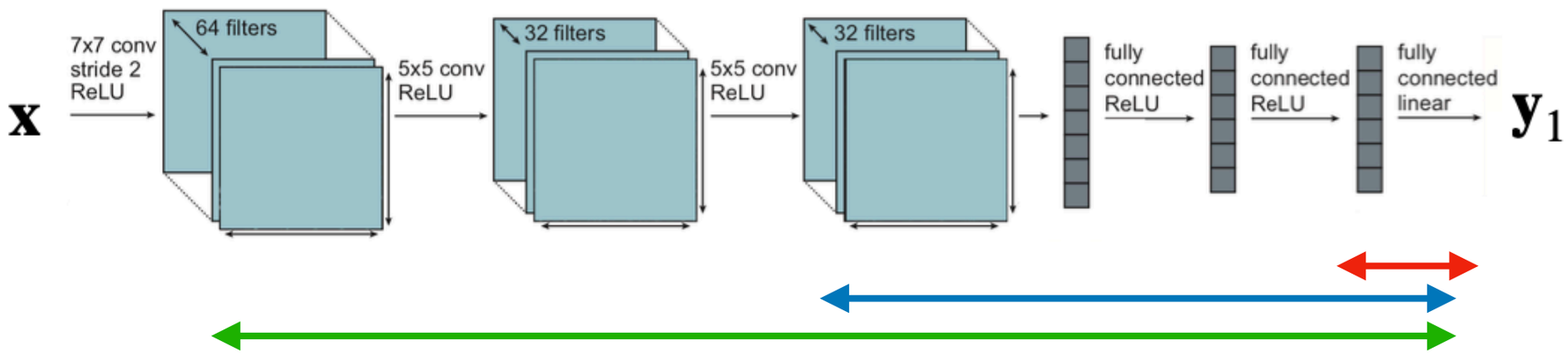
Where do you get the pre-trained parameters?

- 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.

Slides Credit: Chelsea Finn

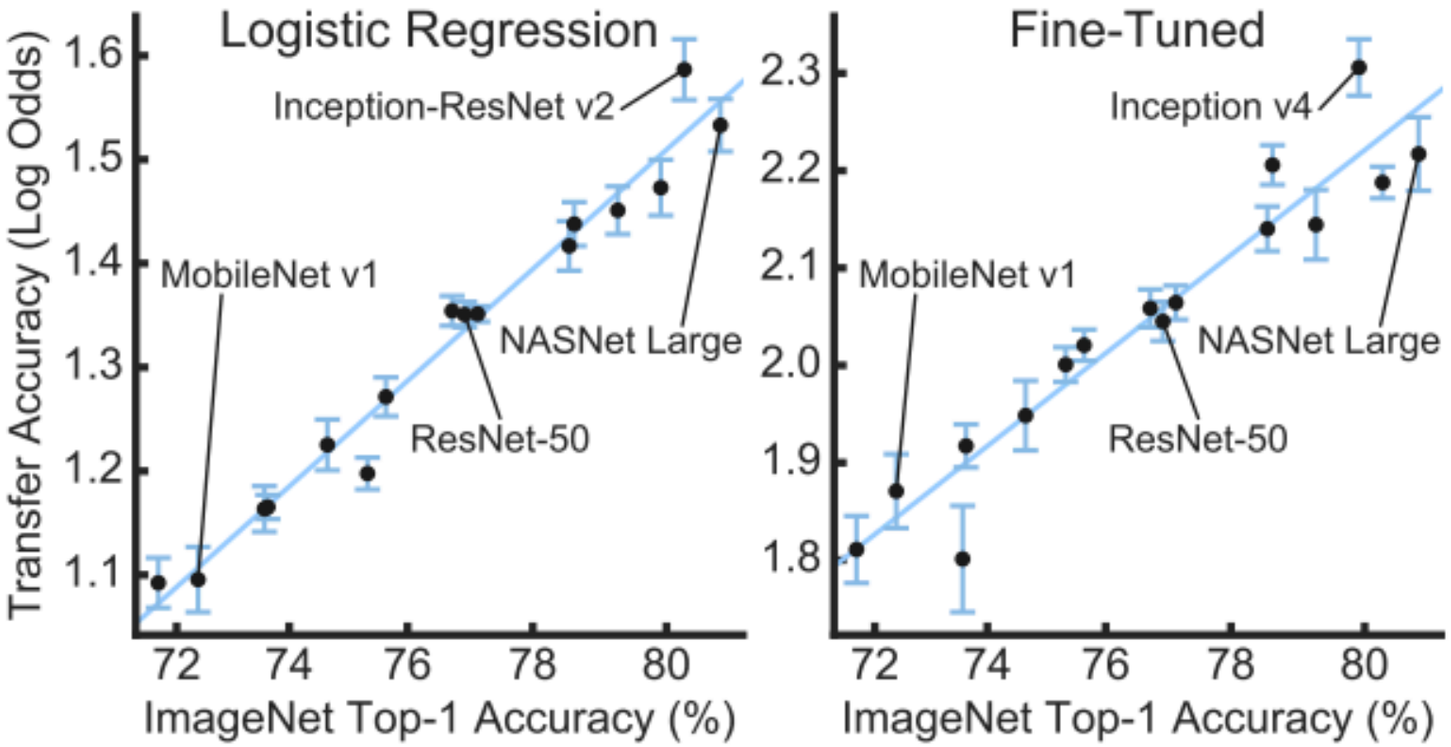
Transfer Learning Variants for ImageNet



- Fine tune only the last layer
- Fine tune top layers
    - Assumption that lower layers might be more general e.g. edge filters are common amongst all images
- “Fine-tune” entire model — here the initial model acts as a good initialization
- Choice is often dependent on data size
    - For very larger target datasets transfer learning can sometimes give no benefit

Which Models are better for transfer

- Often larger models on bigger datasets transfer better



Do Better ImageNet Models Transfer Better?

Simon Kornblith,* Jonathon Shlens, and Quoc V. Le
Google Brain
{skornblith,shlens,qvl}@google.com

Meta-Learning

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

The Meta-Learning Problem

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

(In sample complexity)

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

In all settings: tasks must share structure.

Transfer Learning can be a solution to Meta-Learning Problem

Slides Credit: Chelsea Finn

Meta-Learning Formulation

Supervised Learning:

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

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

Meta Supervised Learning:

Inputs: $\mathcal{D}^{\text{tr}}$ $\mathbf{x}_{\text{test}}$ Outputs: $\mathbf{y}_{\text{test}}$ Data: $\{\mathcal{D}_i\}$

$$\mathbf{y}_{\text{test}} = h(\underbrace{\{(\mathbf{x}, \mathbf{y})_{1:K}\}}_{\mathcal{D}^{\text{tr}}}, \mathbf{x}_{\text{test}}; \theta)$$

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

- One solution is to think of meta-learning as a function that takes in a whole dataset

Slides Credit: Chelsea Finn

Meta-Train and Meta-Test sets

How does meta-learning work? An example.

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

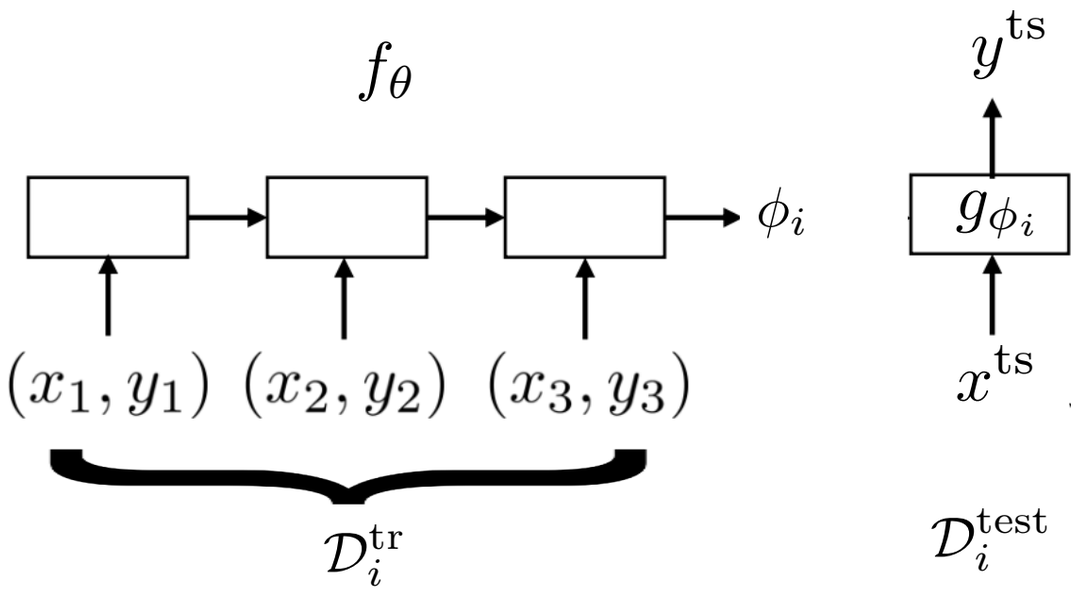
Slides Credit: Chelsea Finn

Meta-Learning with Black-Box

$$\mathbf{y}_{\text{test}} = h(\mathcal{D}^{\text{tr}}, \mathbf{x}_{\text{test}}; \theta)$$

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

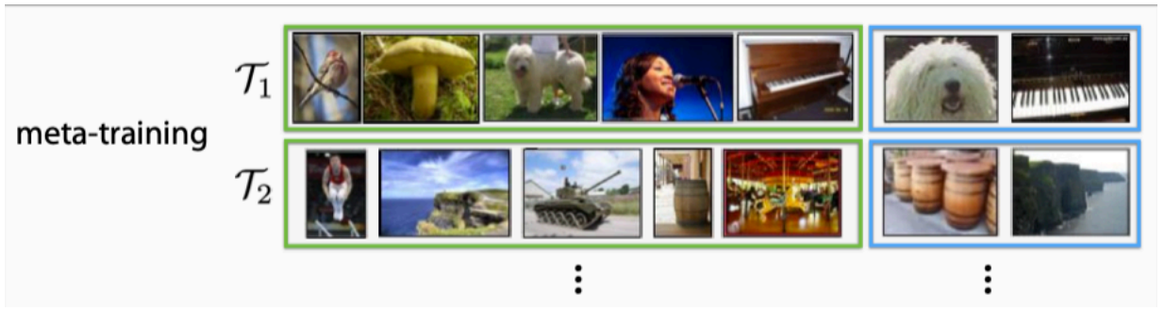
Predict test points with $\mathbf{y}^{\text{ts}} = g_{\phi_i}(\mathbf{x}^{\text{ts}})$



Train with standard supervised learning!

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

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



Meta-train train set $\mathcal{D}_i^{\text{tr}}$

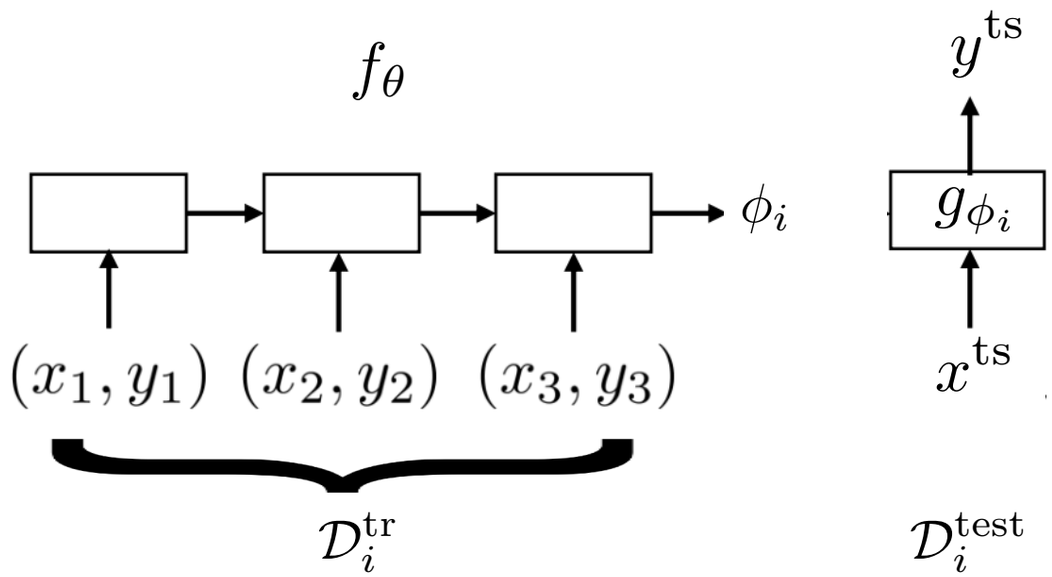
Meta-train test set $\mathcal{D}_i^{\text{test}}$

Slides Credit: Chelsea Finn

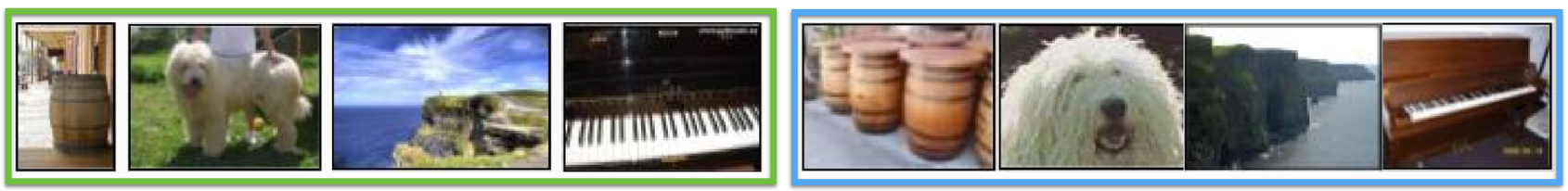
Meta-Learning with Black-Box Adaptation

Black-Box Adaptation

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



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



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

Slides Credit: Chelsea Finn

Meta-Learning Initialization

Optimization-Based Adaptation

Fine-tuning
[test-time]

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

pre-trained parameters

training data for new task

Meta-learning

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

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

Fine-tuning can include multiple steps, we interpret this as learning the initialization

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

Slides Credit: Chelsea Finn

Meta-Learning with Inner Optimization

Optimization-Based Adaptation

Key idea: Acquire ϕ_i through optimization.

General Algorithm:

~~Black box approach~~ Optimization-based approach

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

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

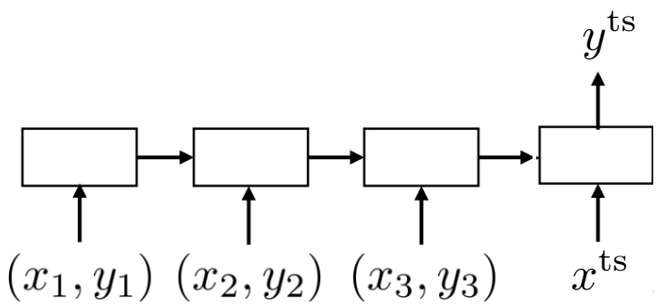
Slides Credit: Chelsea Finn

Meta-Learning with Inner Optimization

Optimization vs. Black-Box Adaptation

Black-box adaptation

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



Model-agnostic meta-learning

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

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

MAML can be viewed as **computation graph**, with embedded gradient operator

Note: Can mix & match components of computation graph

Learn initialization but replace gradient update with learned network

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

Ravi & Larochelle ICLR '17

(actually precedes MAML)

Slides Credit: Chelsea Finn

Some issues with Meta-Learning Algorithms

- Can be computationally very expensive
- Can only use a few iterations in the case of e.g. MAML
- Rigid restrictions on size of test data
  - What if I have 100 samples instead of the 10 used to train?
  - Many methods that work in 1 or 5 shot learning don’t apply well for more sample

Meta-Learning Failures

- Meta-learning is an interesting conceptual framework that generalizes transfer learning
- “Meta-learning” is often approached and associated to using a nested objective function which can be difficult to optimize particularly in the case where it is an inner optimization or RNN style model
- In practice we can bypass the meta-learning formulation to achieve good performance on the same problem using the transfer learning and other approaches which are unaware of their future use case
- Even in cases where it gives small performance gains this typically comes at huge computational cost
- Interesting idea with many active research directions

Selecting Relevant Features from a Multi-domain Representation for Few-shot Classification

Nikita Dvornik[1], Cordelia Schmid[2], and Julien Mairal[1]

[1] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
[2] Inria, École normale supérieure, CNRS, PSL Research Univ., 75005 Paris, France
firstname.lastname@inria.fr

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

d Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

A CLOSER LOOK AT FEW-SHOT CLASSIFICATION

Wei-Yu Chen
Carnegie Mellon University
weiyuc@andrew.cmu.edu

Yen-Cheng Liu & Zsolt Kira
Georgia Tech
{ycliu,zkira}@gatech.edu

Yu-Chiang Frank Wang
National Taiwan University
ycwang@ntu.edu.tw

Jia-Bin Huang
Virginia Tech
jbhuang@vt.edu

Meta-Learning Failures

Transfer Learning

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

The Meta-Learning Problem

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

- Fine-tuning baseline vs top meta-learning methods

- Fine-tuning baseline , how is it different from transfer learning problem statement?

Method	CUB 1-shot	CUB 5-shot	*mini*-ImageNet 1-shot	*mini*-ImageNet 5-shot
Baseline	47.12 ± 0.74	64.16 ± 0.71	42.11 ± 0.71	62.53 ±0.69
Baseline++	60.53 ± 0.83	79.34 ± 0.61	48.24 ± 0.75	66.43 ±0.63
MatchingNet Vinyals et al. (2016)	60.52 ± 0.88	75.29 ± 0.75	48.14 ± 0.78	63.48 ±0.66
ProtoNet Snell et al. (2017)	50.46 ± 0.88	76.39 ± 0.64	44.42 ± 0.84	64.24 ±0.72
MAML Finn et al. (2017)	54.73 ± 0.97	75.75 ± 0.76	46.47 ± 0.82	62.71 ±0.71
RelationNet Sung et al. (2018)	62.34 ± 0.94	77.84 ± 0.68	49.31 ± 0.85	66.60 ±0.69

A CLOSER LOOK AT FEW-SHOT CLASSIFICATION

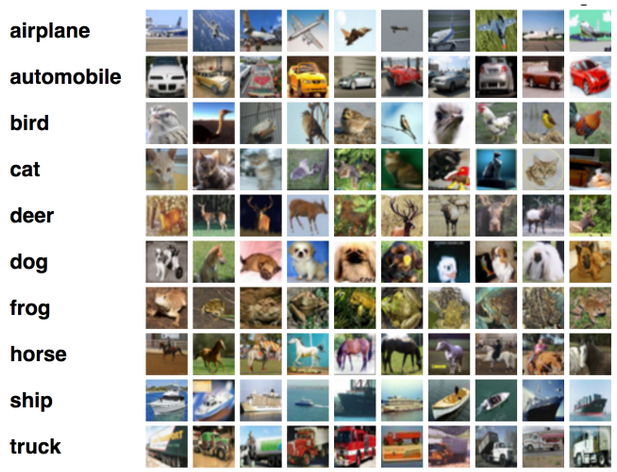
Wei-Yu Chen
Carnegie Mellon University
weiyuc@andrew.cmu.edu

Yen-Cheng Liu & Zsolt Kira
Georgia Tech
{ycliu,zkira}@gatech.edu

Yu-Chiang Frank Wang
National Taiwan University
ycwang@ntu.edu.tw

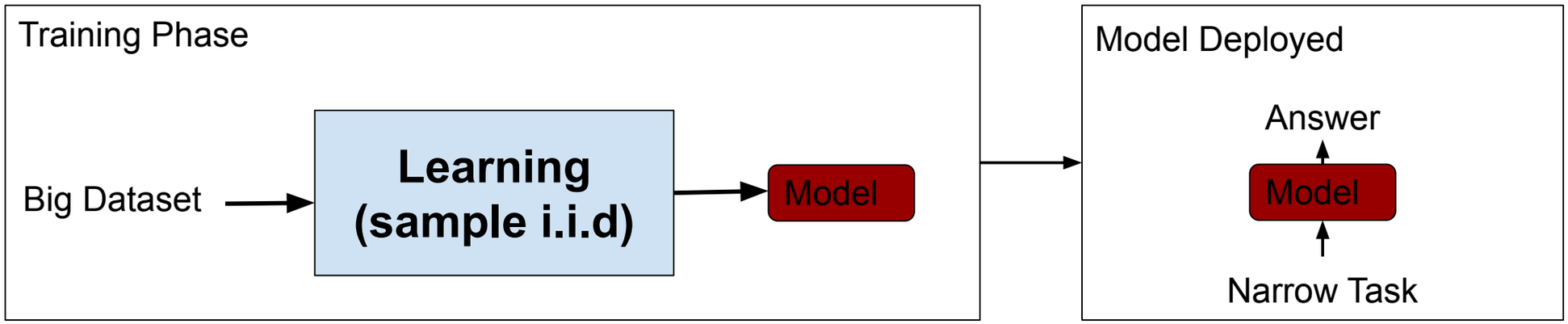
Jia-Bin Huang
Virginia Tech
jbhuang@vt.edu

Continual Learning

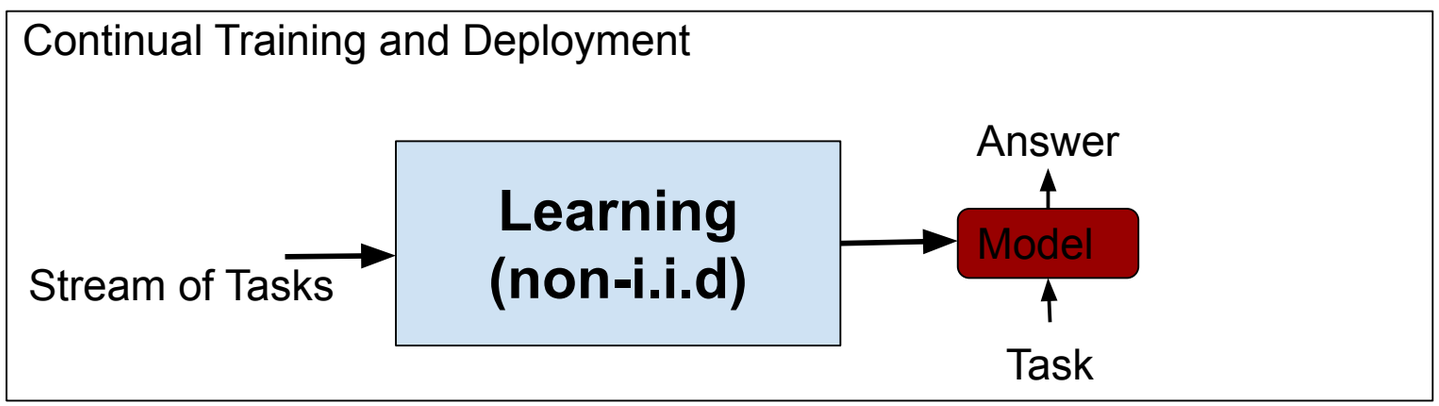


- Big dataset
- Train for long time (with SGD)
- Solve specific task
- New category? New Task?

Standard ML Paradigm

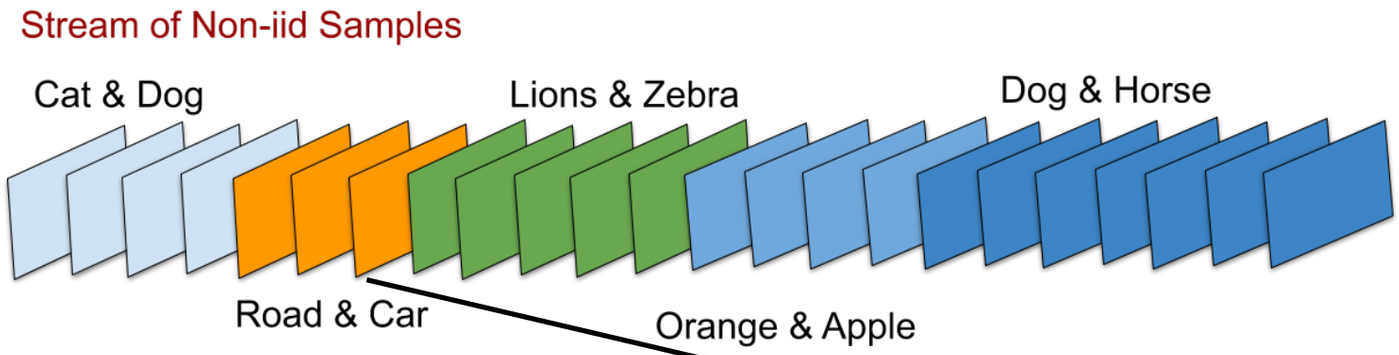


More Realistic

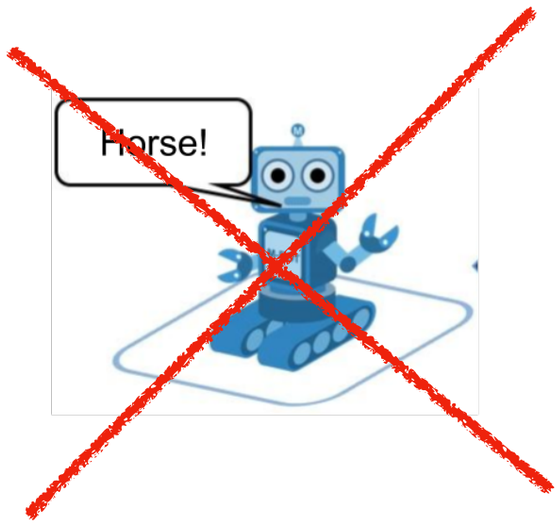


- Data constantly arriving
- New tasks / Distribution shift
- Re-training impractical

Continual Learning in Deep Networks



Old Data



Catastrophic forgetting

- Learner observers non-iid data stream
- We want models that perform
  - Fast transfer learning
  - Don’t forget how to solve old tasks (catastrophic forgetting)
  - Constrained memory and compute

Continual Learning

Multi-Task Learning

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

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

Transfer Learning

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

The Meta-Learning Problem

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

Continual Learning

Given $\mathcal{T}_1 .. \mathcal{T}_{n-1}$ quickly solve $\mathcal{T}_n$ Don't forget $\mathcal{T}_1 .. \mathcal{T}_{n-1}$

Repeat for n+1...

- Continual Learning can be seen as a superset of the above problems
  - Do well on new task while maintaining MTL performance on previous tasks
  - Limiting computation and memory (as in transfer learning)
  - Incorporates temporal notion
    - Meta-learning formulation assumes we care only about the specific task T given T-1 prior tasks
    - In CL we want the best anytime performance (algorithm that does well throughout all T observed tasks)

Approaches to Continual Learning

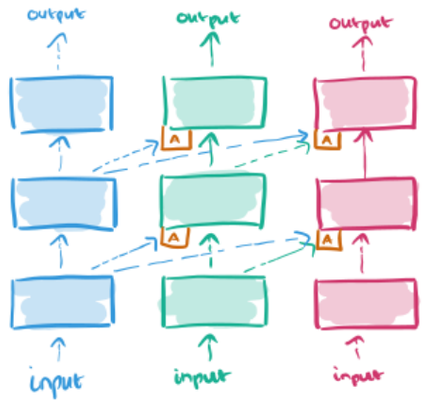
- Regularization methods
  - Keep weights near old model (e.g. EWC)

Kirkpatrick, James, et al. "Overcoming catastrophic forgetting in neural networks." *Proceedings of the national academy of sciences* 114.13 (2017): 3521-3526.

$$\mathscr{L}(X_{new}, Y_{new}, \theta) + R(\theta, \theta_{old})$$

  - Change more slowly weights important for old task

- Dynamic architecture / model
  - Add new components for new tasks
  - Allocate set of weights for new models



Rusu, Andrei A., et al. "Progressive neural networks." *arXiv preprint arXiv: 1606.04671* (2016).

- Rehearsal methods