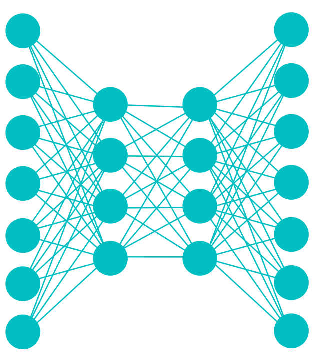
Lecture Notes for

Neural Networks and Machine Learning

Adaptive, Self-supervised, Multi-modal, & Multi-task Learning



Logistics and Agenda

- Logistics
  - Lab three uses multi-task and multi-modal learning
- Agenda
  - Adaptive Learning
  - Self-Supervised Learning
  - Paper Presentation
  - Multi-modal/task Learning
    - Techniques
    - Applications and domains
- Next Time:
  - Paper Presentation: Speaker Verification with X-Vectors and SincNet

Paper Presentation: The Lottery Hypothesis

Published as a conference paper at ICLR 2019

THE LOTTERY TICKET HYPOTHESIS: FINDING SPARSE, TRAINABLE NEURAL NETWORKS

Jonathan Frankle
MIT CSAIL
jfrankle@csail.mit.edu

Michael Carbin
MIT CSAIL
mcarbin@csail.mit.edu

ABSTRACT

Neural network pruning techniques can reduce the parameter counts of trained networks by over 90%, decreasing storage requirements and improving computational performance of inference without compromising accuracy. However, contemporary experience is that the sparse architectures produced by pruning are difficult to train from the start, which would similarly improve training performance.

We find that a standard pruning technique naturally uncovers subnetworks whose initializations made them capable of training effectively. Based on these results, we articulate the *lottery ticket hypothesis*: dense, randomly-initialized, feed-forward networks contain subnetworks (*winning tickets*) that—when trained in isolation—reach test accuracy comparable to the original network in a similar number of iterations. The winning tickets we find have won the initialization lottery: their connections have initial weights that make training particularly effective.

We present an algorithm to identify winning tickets and a series of experiments that support the lottery ticket hypothesis and the importance of these fortuitous initializations. We consistently find winning tickets that are less than 10-20% of the size of several fully-connected and convolutional feed-forward architectures for MNIST and CIFAR10. Above this size, the winning tickets that we find learn faster than the original network and reach higher test accuracy.

Last Time

$$X = x_1, x_2, \ldots x_N \in \mathcal{X}$$

$$\mathcal{D} = \{\mathcal{X}, p(X)\}$$

Domain — Feature Space — Probability Observation

- Domain defines the features used
- Marginal Distribution of observing instances in the feature space
  - Typically intractable to calculate (generative)

$$Y = y_1, y_2, \ldots y_N \in \mathcal{Y}$$

$$\mathcal{T} = \{\mathcal{Y}, p(Y|X)\}$$

Task — Label Space — Learned Probability

- Task is within a domain
- Label space is typically one specific classification or regression task
- Probability of observing label given the feature space:
  - Not intractable (discriminative)

- Feature Extraction Transfer
  - Most well known: use learned parameters from one task in another task in same domain
  - Most useful when labels for target domain are sparse

	Training		Testing	
Transfer Learning	Task 1		Task 2	
Multi-task Learning	Task 1 ···	Task N	Task 1 ···	Task N
Lifelong Learning	Task 1 ···	Task N	Task N+1	

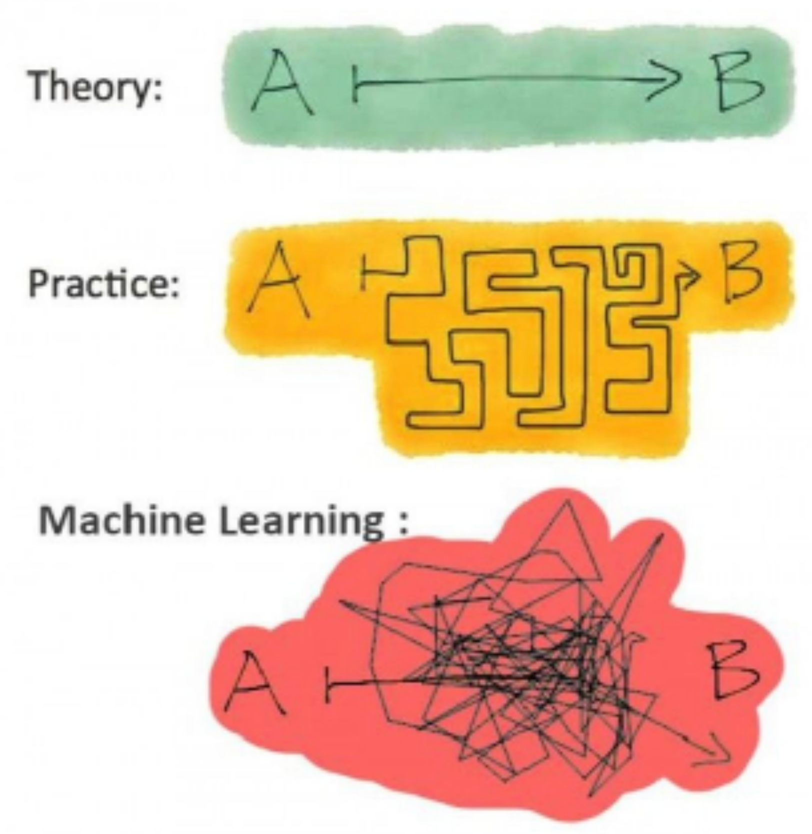
Humans can learn to ride a bike and use that to understand better about driving a car. Machine Learning in its current form is far from this capability. How can we move our siloed version of artificial intelligence closer to the process of human based learning? How can we accumulate knowledge from model to model?

Does biology of human learning hold any clues to success? How does a human learn to crawl? To talk? To ride a bike? What is a human's motivation to learn?

Ian Goodfellow's Definition:

"Transfer learning refers to any situation where what has been learned in one setting is exploited to improve generalization in another setting."

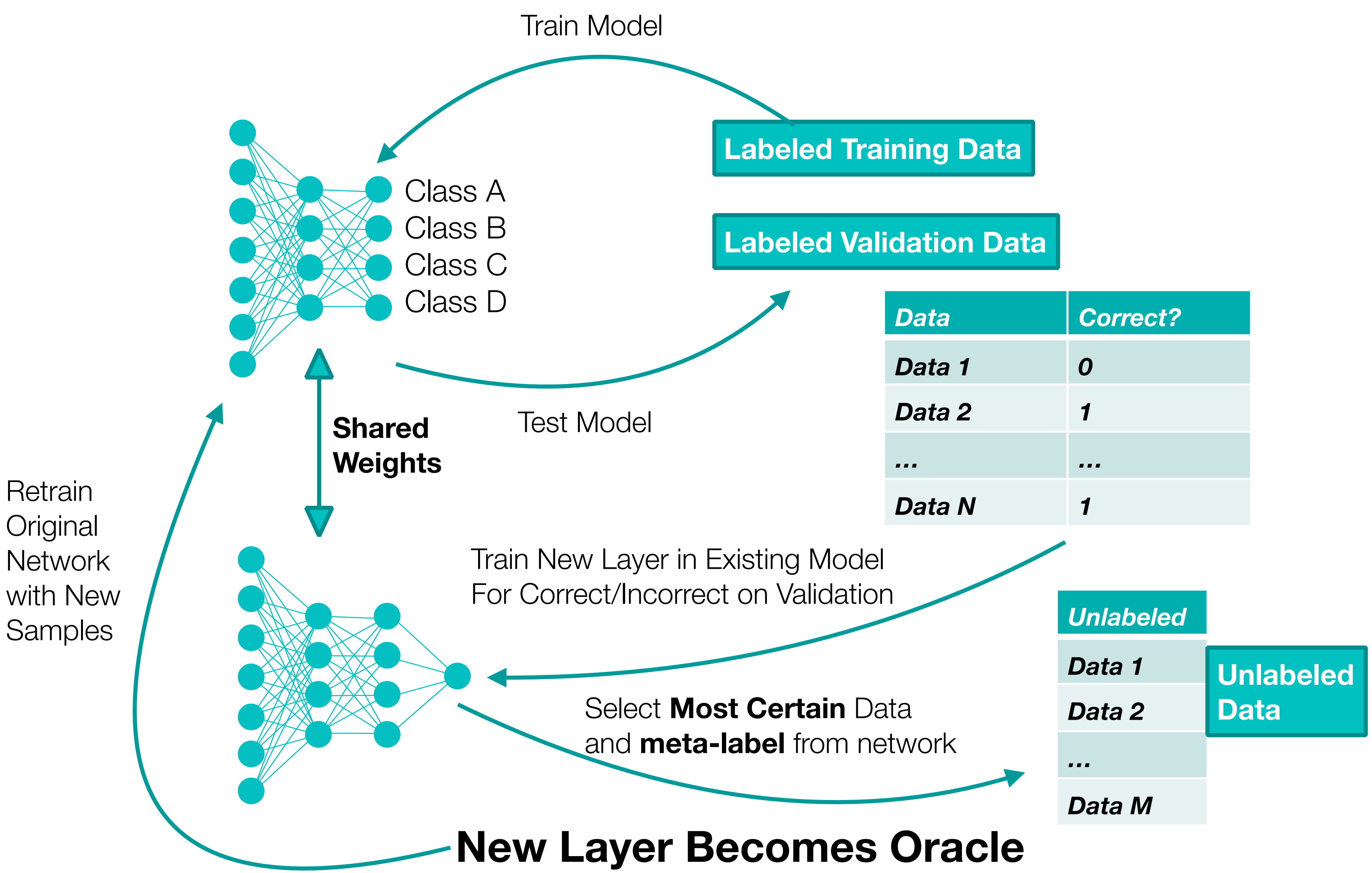
Active Transfer Learning



Active Learning Overview

- **Basic Idea**: Use a trained model to sample from an oracle that can magically give you a new label
  - We are asking:
    What labels should we ask the oracle about?
- Uncertainty Sampling
  - Choose instances where the model is most uncertain or most certain
  - Various ways to measure certainty
- Diversity Sampling
  - Choose instances that are similar or different from training distribution

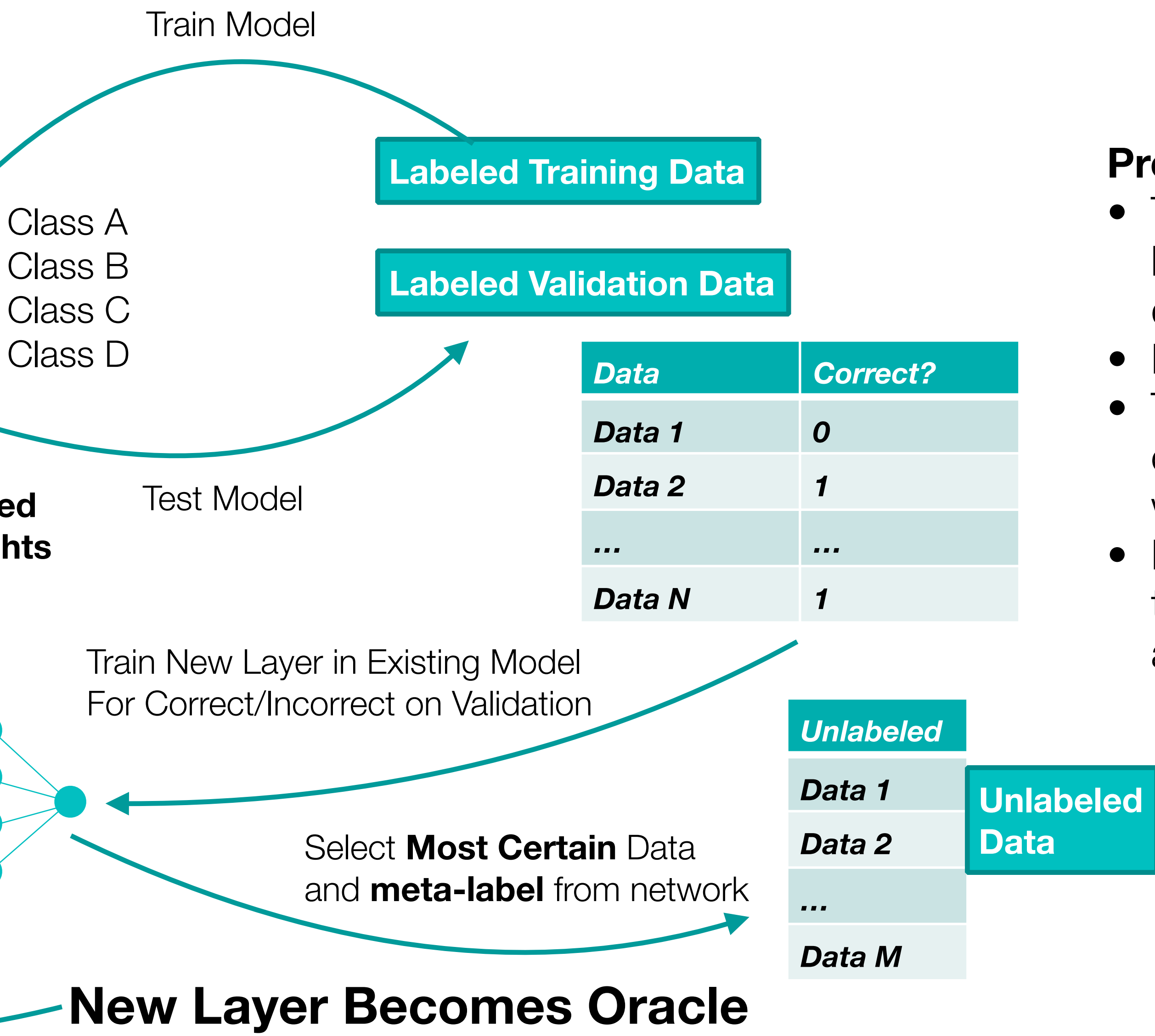
Uncertainty Sampling with a Neural Network



Data	*Correct?*
Data 1	*0*
Data 2	*1*
...	*...*
Data N	*1*

Unlabeled
Data 1
Data 2
...
Data M

Uncertainty Sampling with a Neural Network



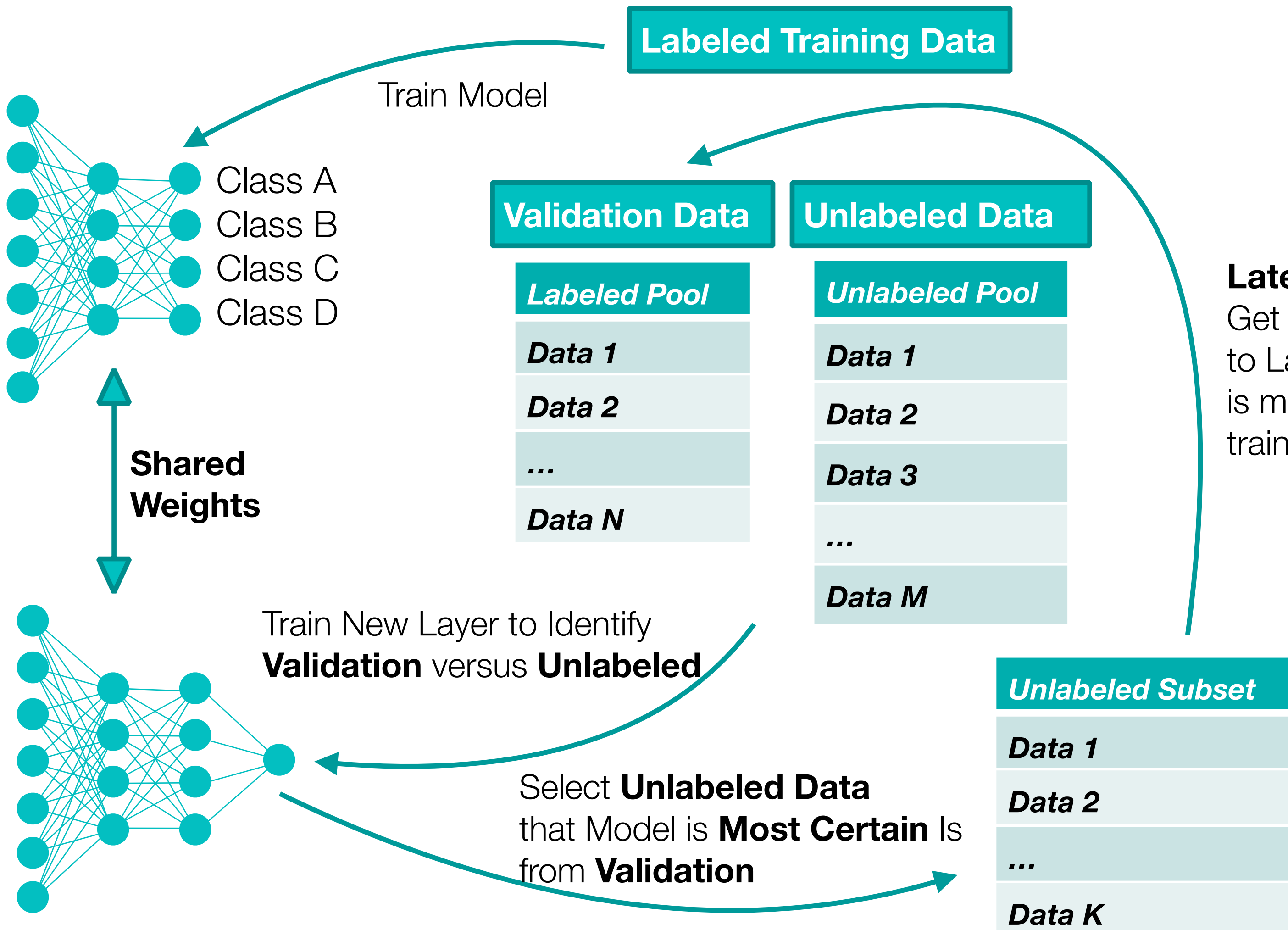
Data	*Correct?*
Data 1	*0*
Data 2	*1*
...	*...*
Data N	*1*

Unlabeled
Data 1
Data 2
...
Data M

Problems:

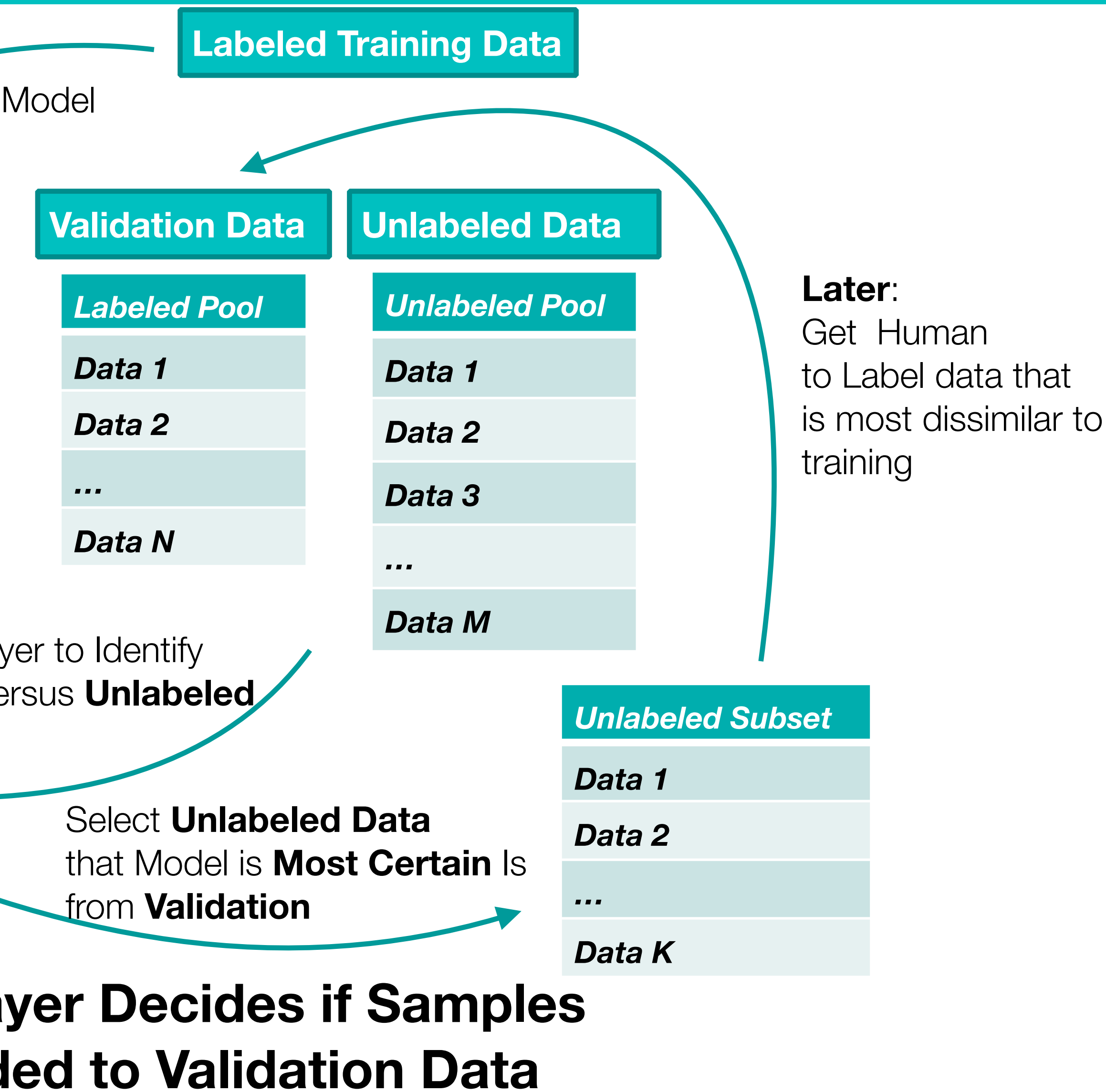
- Training pool is represented by classes the model already does well predicting
- Limited diversity of Samples
- Training pool can become contaminated easily from a few wrong predictions
- For Oracle: we might be asking to get labels that the model is already good at classifying

Diversity Sampling with a Neural Network



New Layer Decides if Samples are Added to Validation Data

Diversity Sampling with a Neural Network



Discussion:
- Training pool is not contaminated
- Expands validation data in well mannered way, not adding too "far away" samples
- Validation versus Unlabeled might not be the best comparison, because it ignores confusions in the training data
- For Oracle: we can get labels to inputs that the model is likely to be unsure about
- But… this only helps us when we have an Oracle to give us labels

ATLAS: Active Transfer Learning for Adaptive Sampling

Advantages:

- Doesn't contaminate the Labeled Training Pool
- Advises on what samples should be labeled later
- Allows for some fine tuning even before Oracle gives labels

Train Model

Labeled Training Data

Class A
Class B
Class C
Class D

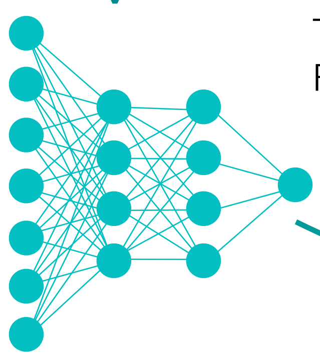
Labeled Validation Data

Test Model

Shared Weights

Data	*Correct?*
Data 1	***0***
Data 2	***1***
...	***...***
Data N	***1***

Subset	*Correct?*
Data 1	***1***
Data 2	***1***
...	***1***
Data K	***1***



Train New Layer in Existing Model
For Correct/Incorrect on Validation

Select **Most Certain** Data and **mark as "correct"**

Unlabeled
Data 1
Data 2
...
Data M

Unlabeled Data

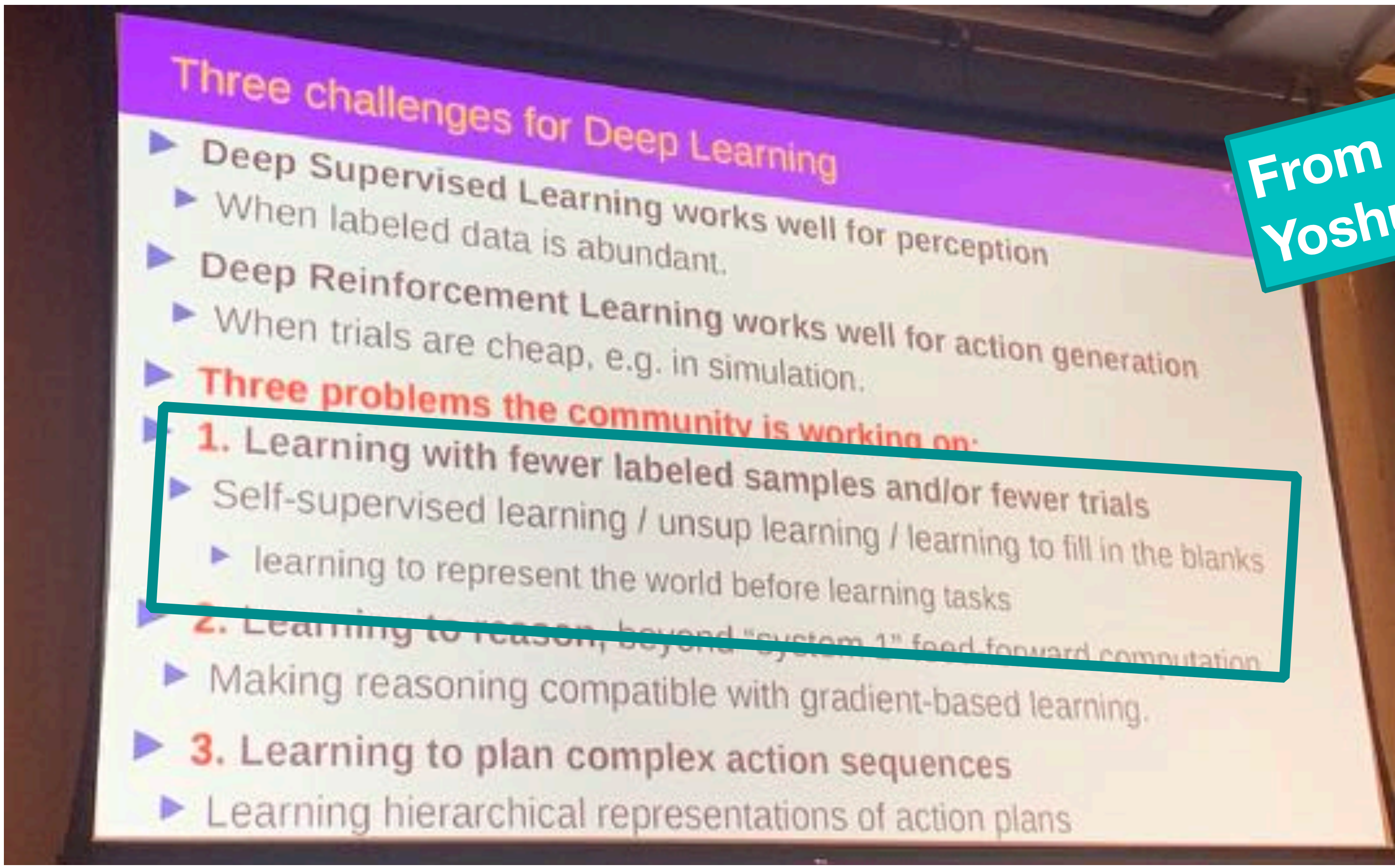
Retrain **Validation Layer And a few shared layers** With "Correct" Samples

New Layer Selects Data that Needs Label

Time Period	Protocol	Expected Feedback
First Week	Homeowner provides 8-20 examples over the first week: • 1-2 Shower usages • 1 run of the dishwasher • 1 run of the laundry machine • 2 examples of each toilet • 1 example of hot and cold water use for each dual handle faucet • 1 example of hot, cold, and mixed water use for each single handle faucet (2 examples if in kitchen)	HydroSense relies on the rule based classifier for the first week. Pressure waves are saved in order to create a sparse codebook of features. Results are displayed at the fixture category for dishwashers, showers, and washing machines.
Start of Second Week	Homeowner provides 2-4 labels every other day when the system messages them on their mobile device	Results are displayed at the full fixture category level from the CoDBN-VE algorithm. Expected accuracy: • 85% at fixture category level
End of Second Week	Homeowner has supplied 9-12 examples that were flagged by active learning.	HydroSense now displays results at the Lumped Fixture level. Expected accuracy: • 82% at fixture level • 87% at fixture category level
End of Third Week	Homeowner continues to supply sparsely selected examples every other day. About 9-12 additional examples provided.	Valve level accuracy now provided. Expected accuracy: • 80% at valve level • 87% at fixture level • 92% at fixture category level
Fourth Week	Homeowner can optionally continue to provide examples to the system for increased accuracy.	Expected accuracy: • 81% at valve level • 89% at fixture level • 93% at fixture category level

Table 8-2. Expected feedback and calibration protocol for semi-supervised HydroSense system

Self-Supervised Learning

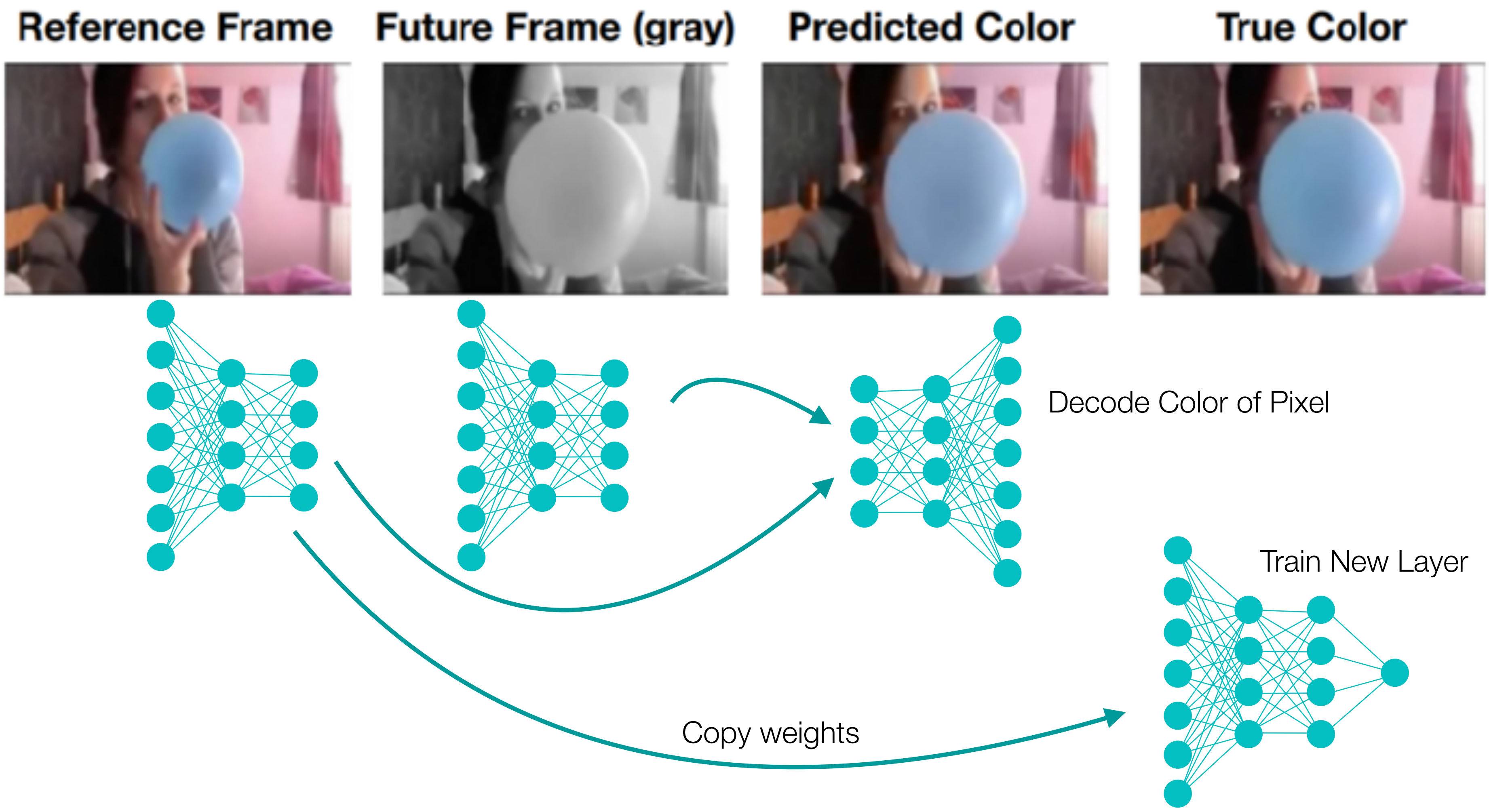


From Yoshua Bengio

Self-supervised Learning

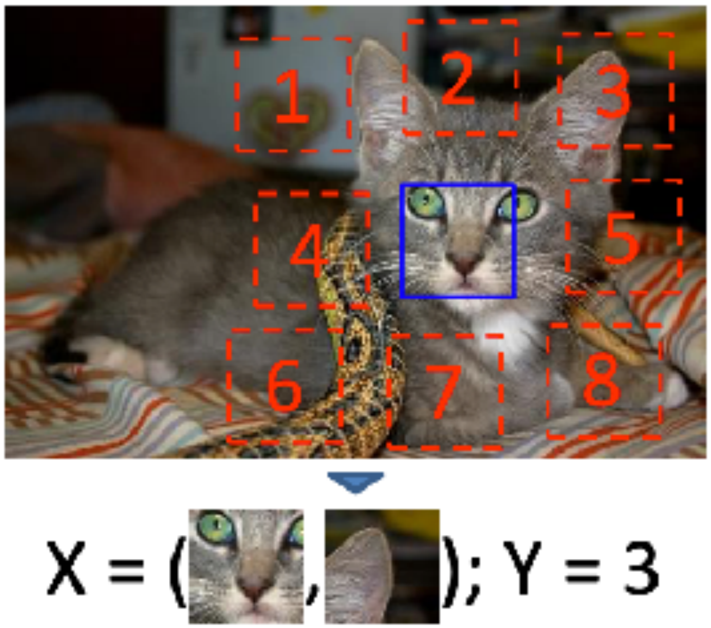
- **Problem**: deep learning is not sample efficient
- **Idea**: learn about the world before learning the task
- **New Problem**: how do we learn about the world?
- **Solution**: transfer learning on toy problem
  - 1. train on auxiliary task that is easy to label
  - 2. throw away anything specific to auxiliary task
  - 3. train new network with task of interest, transferring knowledge (downstream task)
  - 4. profit

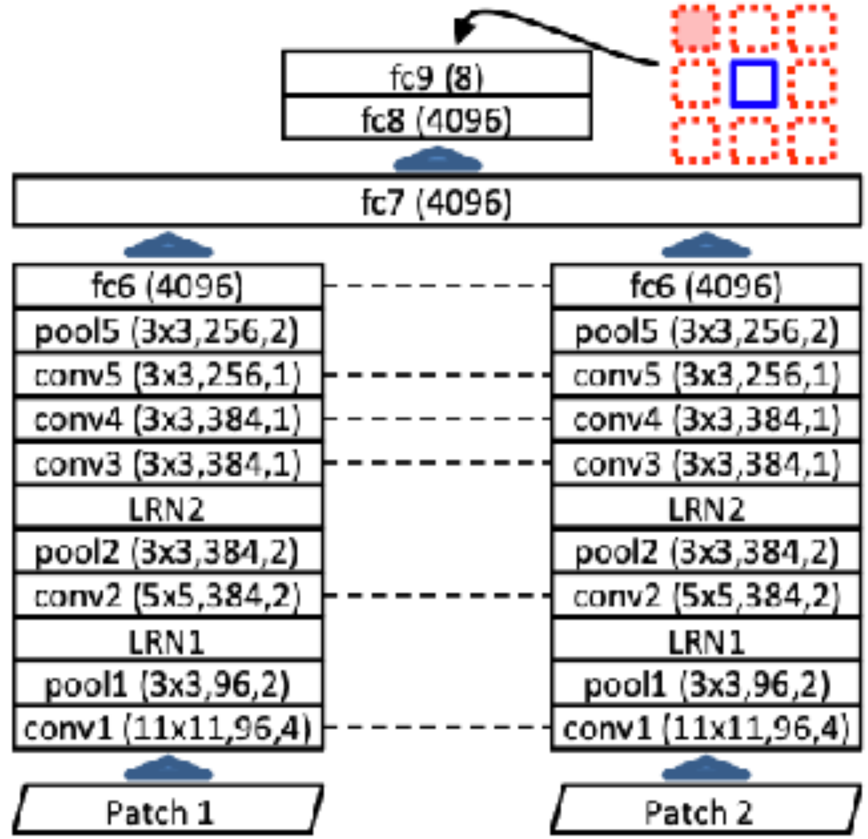
Examples of Self Supervised Learning



https://www.fast.ai/2020/01/13/self_supervised/

Examples of Self Supervised Learning





Unsupervised Visual Representation Learning by Context Prediction

Carl Doersch[1,2] Abhinav Gupta[1] Alexei A. Efros[2]
[1] School of Computer Science, Carnegie Mellon University
[2] Dept. of Electrical Engineering and Computer Science, University of California, Berkeley

https://www.fast.ai/2020/01/13/self_supervised/

Examples of SSL

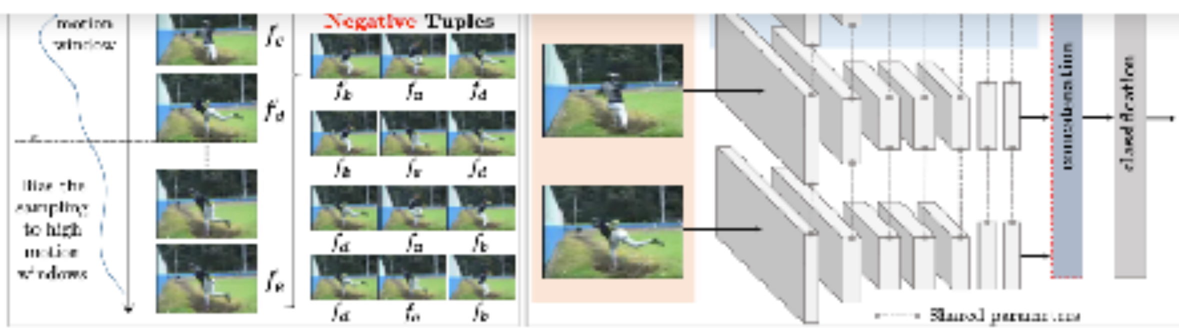
Shuffle and Learn: Unsupervised Learning using Temporal Order Verification

Ishan Misra[1] C. Lawrence Zitnick[2] Martial Hebert[1]

[1] The Robotics Institute, Carnegie Mellon University
[2] Facebook AI Research

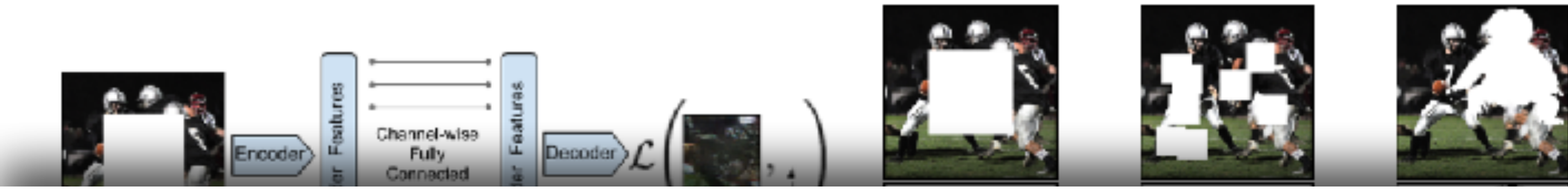
Table 2: Mean classification accuracies over the 3 splits of UCF101 and HMDB51 datasets. We compare different initializations and finetune them for action recognition.

Dataset	Initialization	Mean Accuracy
UCF101	Random	38.6
	(Ours) Tuple verification	**50.2**
HMDB51	Random	13.3
	UCF Supervised	15.2
	(Ours) Tuple verification	**18.1**



https://www.fast.ai/2020/01/13/self_supervised/

Examples of Self Supervised Learning



Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	78.2%	56.8%	48.0%
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal *et al.* [1]	egomotion	10 hours	52.9%	41.8%	-
Wang *et al.* [39]	motion	1 week	58.7%	47.4%	-
Doersch *et al.* [7]	relative context	4 weeks	55.3%	46.6%	-
Ours	context	14 hours	56.5%	44.5%	30.0%

Context Encoders: Feature Learning by Inpainting

Deepak Pathak Philipp Krähenbühl Jeff Donahue Trevor Darrell Alexei A. Efros
University of California, Berkeley

https://www.fast.ai/2020/01/13/self_supervised/

Unsupervised Consistency Loss

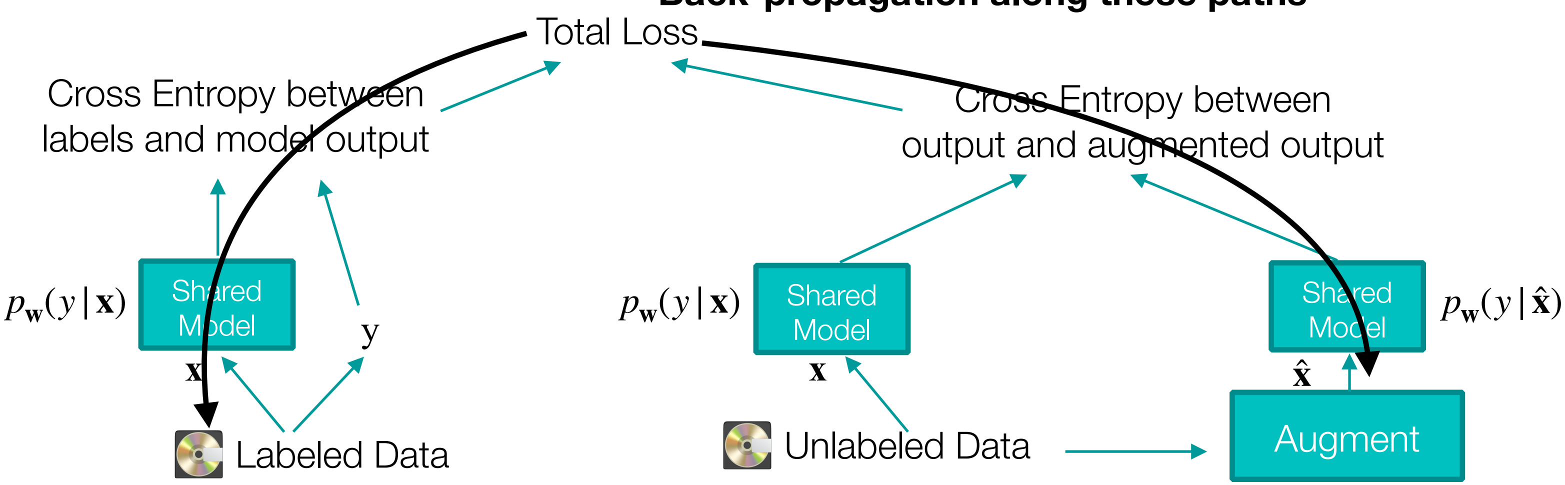
$$\min_{\mathbf{w}} \overbrace{\mathbf{E}_{\mathbf{x},y\in L}[-\log p_{\mathbf{w}}(y\,|\,\mathbf{x})]}^{\text{cross entropy}} + \lambda\, \overbrace{\mathbf{E}_{\mathbf{x}\in U}\mathbf{E}_{\hat{\mathbf{x}}\leftarrow q(\hat{\mathbf{x}}|\mathbf{x})}\left[\mathcal{D}_{KL}\left(\underset{\textbf{no}\text{ back prop}}{p_{\mathbf{w}}(y\,|\,\mathbf{x})}\,||\,\underset{\textbf{yes}\text{ back prop}}{p_{\mathbf{w}}(y\,|\,\hat{\mathbf{x}})}\right)\right]}^{\text{consistency in augmentation}}$$

Neural Network approximates $p(y|\mathbf{x})$ by $\mathbf{w}$
Use labeled data to minimize network

Sample new $\mathbf{x}$ from unlabeled pool with function q
function q is augmentation procedure
Minimize cross entropy of two models

Get accustomed to this notation

Update Model with
Back-propagation along these paths



Unsupervised Data Augmentation (UDA) for Consistency Training, Xie et al., NeurIps 2019

Unsupervised Consistency Loss

$$\min_{\mathbf{w}} \overbrace{\mathbf{E}_{\mathbf{x},y\in L}[-\log p_{\mathbf{w}}(y\,|\,\mathbf{x})]}^{\text{cross entropy}} + \lambda\, \overbrace{\mathbf{E}_{\mathbf{x}\in U}\mathbf{E}_{\hat{\mathbf{x}}\leftarrow q(\hat{\mathbf{x}}|\mathbf{x})}\left[\mathscr{D}_{KL}\left(p_{\mathbf{w}}(y\,|\,\mathbf{x})\,||\,p_{\mathbf{w}}(y\,|\,\hat{\mathbf{x}})\right)\right]}^{\text{consistency in augmentation}}$$

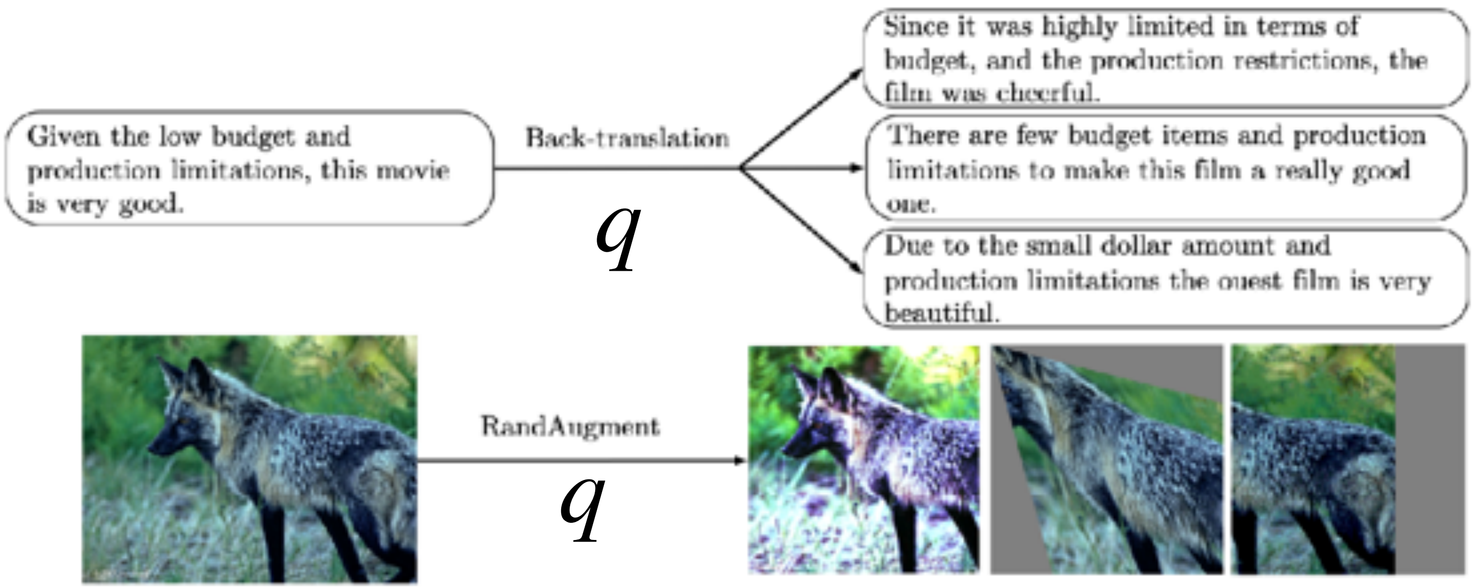


Figure 2: Augmented examples using back-translation and RandAugment.

Unsupervised Data Augmentation (UDA) for Consistency Training, Xie et al., NeurIps 2019

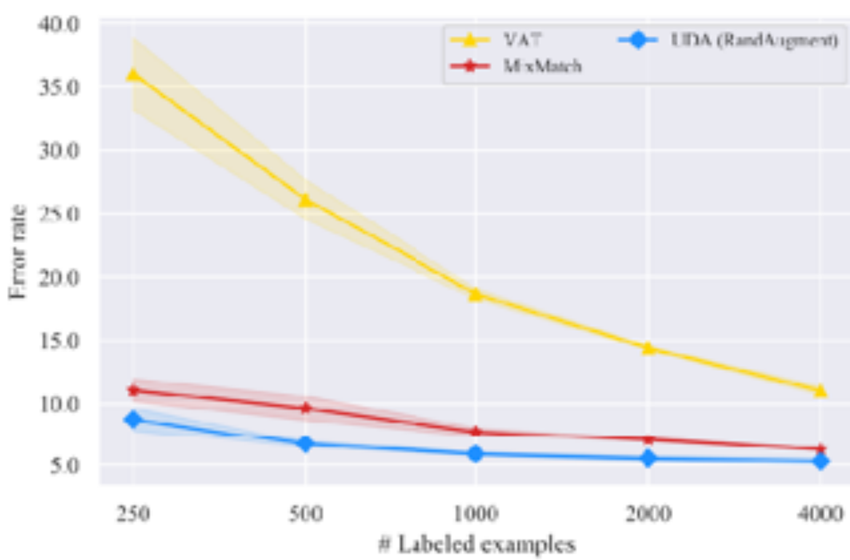
Unsupervised Consistency Loss

Augmentation (# Sup examples)	Sup (50k)	Semi-Sup (4k)
Crop & flip	5.36	16.17
Cutout	4.42	6.42
RandAugment	**4.23**	**5.29**

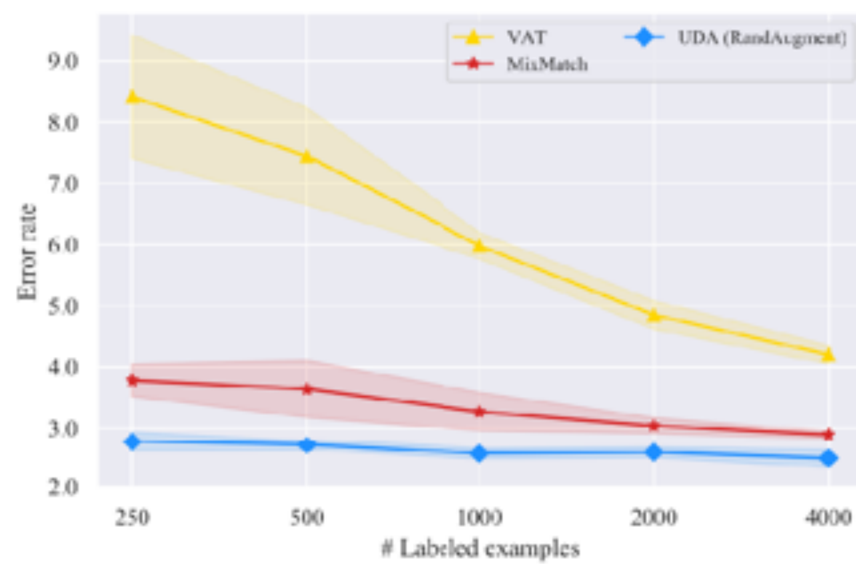
Table 1: Error rates on CIFAR-10.

Augmentation (# Sup examples)	Sup (650k)	Semi-sup (2.5k)
✗	38.36	50.80
Switchout	37.24	43.38
Back-translation	**36.71**	**41.35**

Table 2: Error rate on Yelp-5.



(a) CIFAR-10



(b) SVHN

Unsupervised Data Augmentation (UDA) for Consistency Training, Xie et al., NeurIps 2019

Unsupervised Consistency Loss

Method	Model	# Param	CIFAR-10 (4k)	SVHN (1k)
Π-Model (Laine & Aila, 2016)	Conv-Large	3.1M	12.36 ± 0.31	4.82 ± 0.17
Mean Teacher (Tarvainen & Valpola, 2017)	Conv-Large	3.1M	12.31 ± 0.28	3.95 ± 0.19
VAT + EntMin (Miyato et al., 2018)	Conv-Large	3.1M	10.55 ± 0.05	3.86 ± 0.11
SNTG (Luo et al., 2018)	Conv-Large	3.1M	10.93 ± 0.14	3.86 ± 0.27
VAdD (Park et al., 2018)	Conv-Large	3.1M	11.32 ± 0.11	4.16 ± 0.08
Fast-SWA (Athiwaratkun et al., 2018)	Conv-Large	3.1M	9.05	-
ICT (Verma et al., 2019)	Conv-Large	3.1M	7.29 ± 0.02	3.89 ± 0.04
Pseudo-Label (Lee, 2013)	WRN-28-2	1.5M	16.21 ± 0.11	7.62 ± 0.29
LGA + VAT (Jackson & Schulman, 2019)	WRN-28-2	1.5M	12.06 ± 0.19	6.58 ± 0.36
mixmixup (Hataya & Nakayama, 2019)	WRN-28-2	1.5M	10	-
ICT (Verma et al., 2019)	WRN-28-2	1.5M	7.66 ± 0.17	3.53 ± 0.07
MixMatch (Berthelot et al., 2019)	WRN-28-2	1.5M	6.24 ± 0.06	2.89 ± 0.06

Methods	SSL	10%	100%
ResNet-50	✗	55.09 / 77.26	77.28 / 93.73
w. RandAugment	✗	58.84 / 80.56	78.43 / 94.37
UDA (RandAugment)	✓	**68.78 / 88.80**	**79.05 / 94.49**

Table 5: Top-1 / top-5 accuracy on ImageNet with 10% and 100% of the labeled set. We use image size 224 and 331 for the 10% and 100% experiments respectively.

Lecture Notes for

Neural Networks and Machine Learning

Ada, SSL,

Next Time:

M-Modal/task

Reading: Papers