Insight
Centre for Data Analytics

DEEP
LEARNING
WORKSHOP

Dublin City University
21-22 May 2018

DCU

Day 2 Lecture 2

Unsupervised, semi-supervised, and transfer learning

Kevin McGuinness
kevin.mcguinness@dcu.ie

Assistant Professor
School of Electronic Engineering
Dublin City University

Semi-supervised and transfer learning

Myth: you can't do deep learning unless you have a million labelled examples for your problem.

Reality

- You can learn useful representations from **unlabelled data**
- You can **transfer** learned representations from a related task
- You can train on a nearby **surrogate objective** for which it is easy to generate labels

Supervised and unsupervised learning

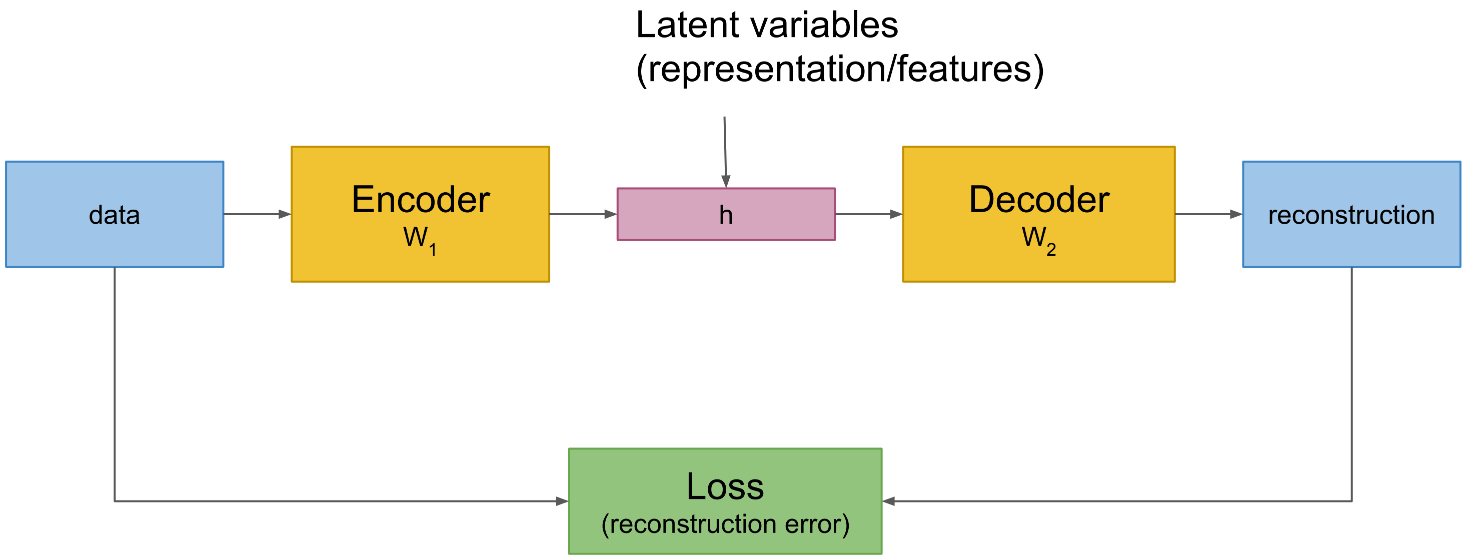
Supervised learning: Given features X and targets Y. Goal: find a function that maps features to targets s.t. the function **generalizes well to unseen examples.**

Unsupervised learning:

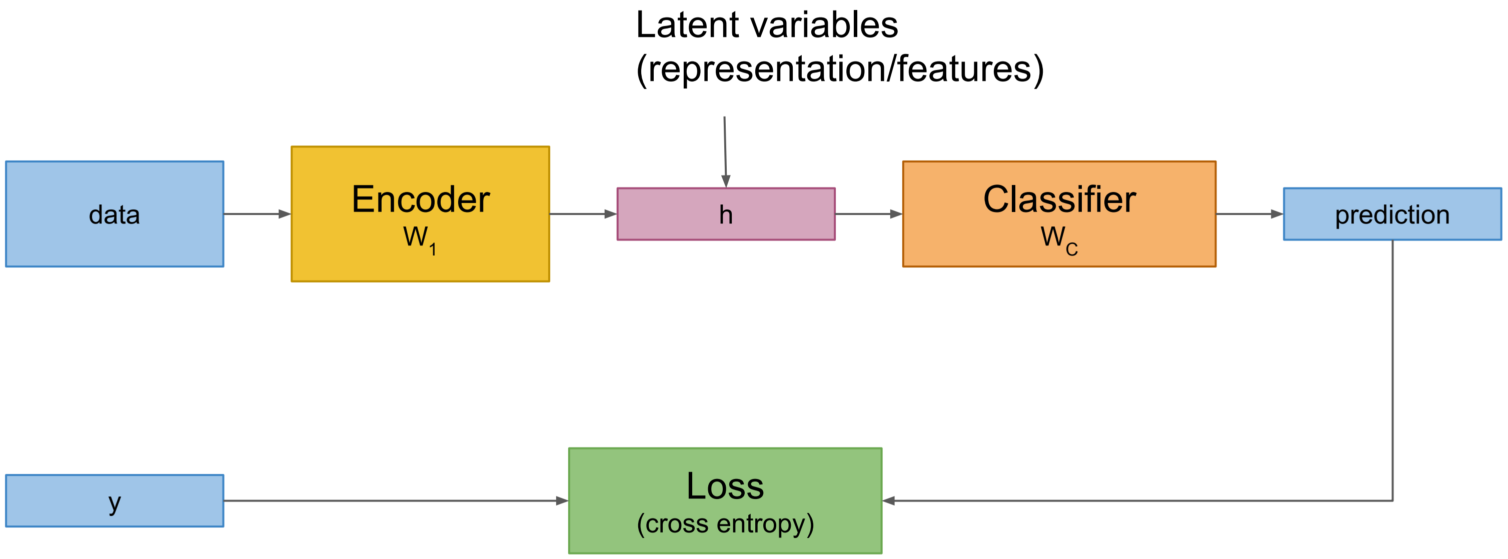
1. Only given the features X.
2. Goal is less clear. Discover structure in the data.
3. Model data distribution p(X) in some way.

Often in practice we do unsupervised learning because we want to solve some supervised task but do not have sufficient data.

Example: autoencoders



Example: autoencoders



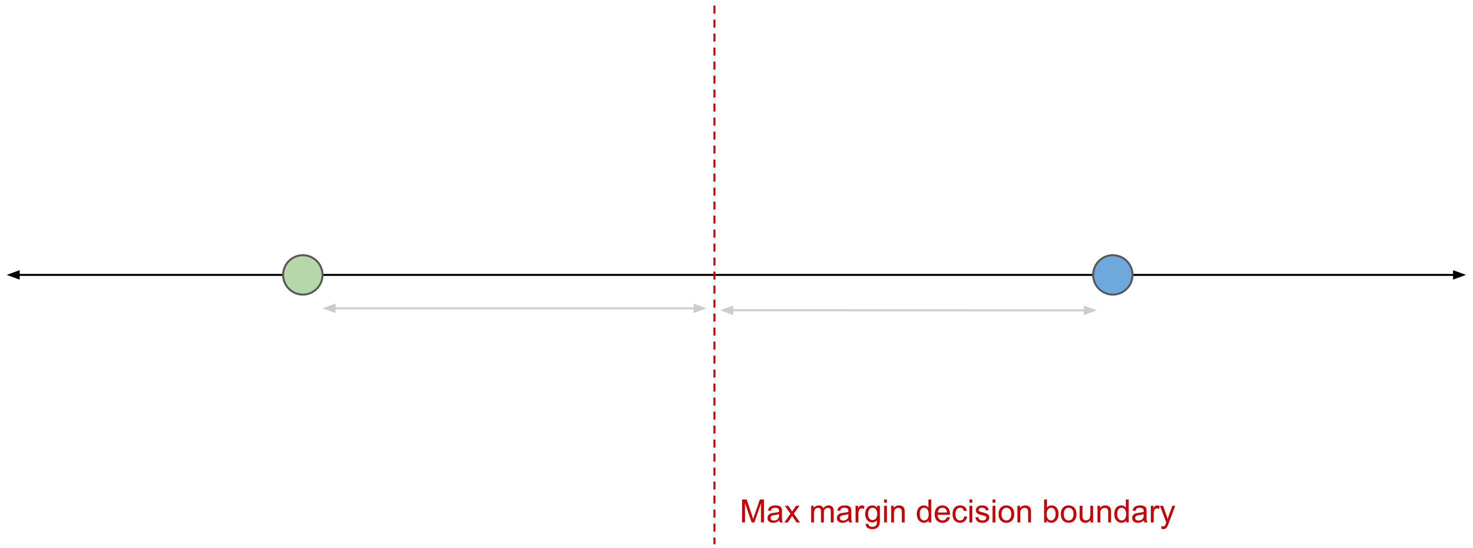
Semi-supervised learning

The previous example is actually a form of semi-supervised learning.

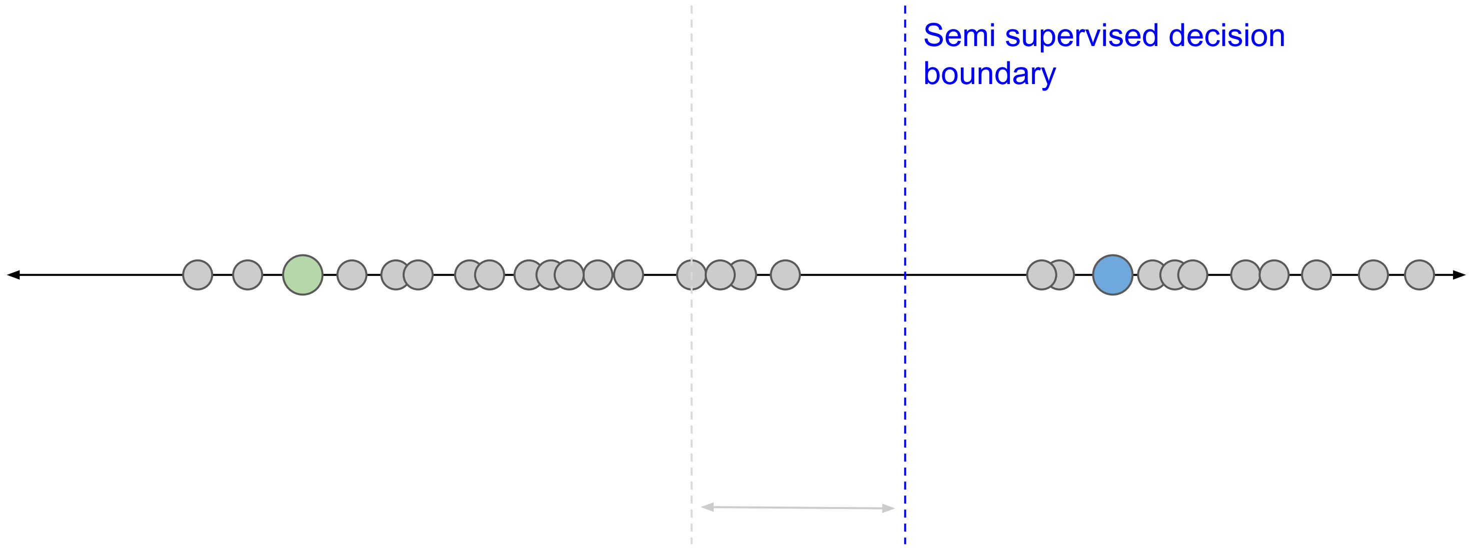
You have some labelled examples and many unlabelled examples.

This is often the typical case in the real world.

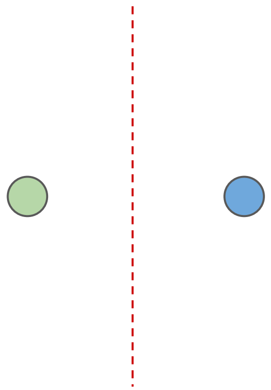
Using unlabelled examples: 1D example



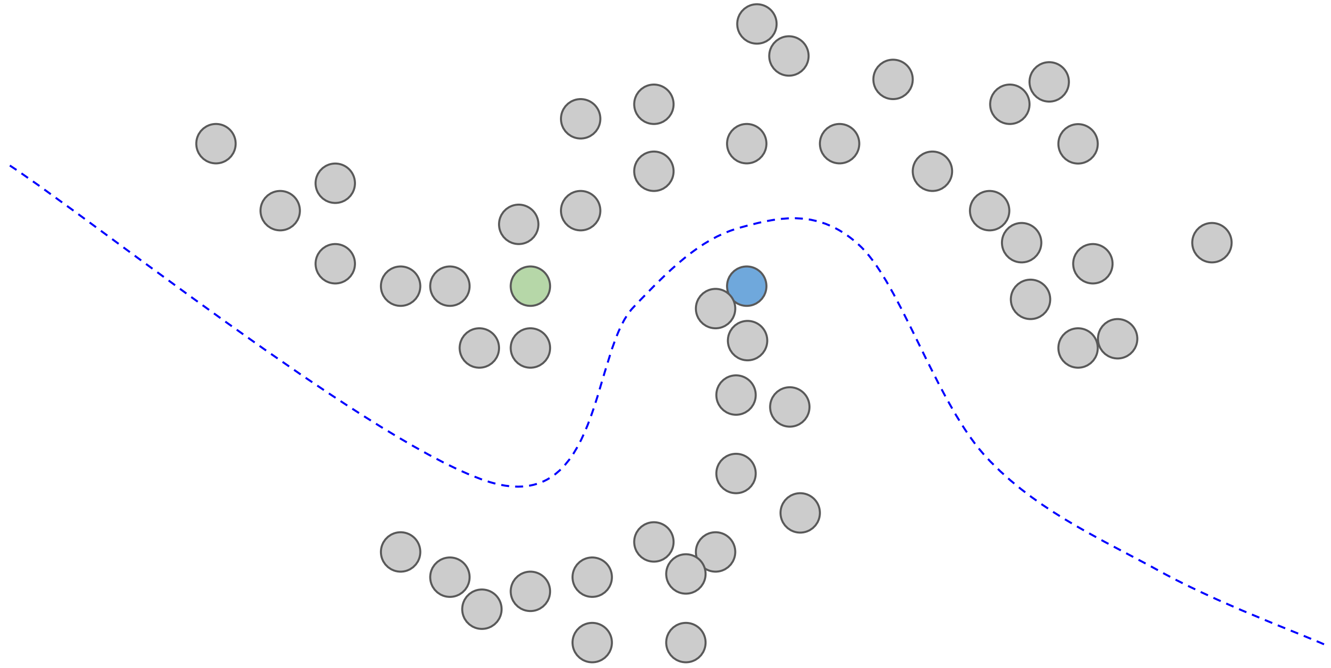
Using unlabelled examples: 1D example



Using unlabelled examples: 2D example



Using unlabelled examples: 2D example

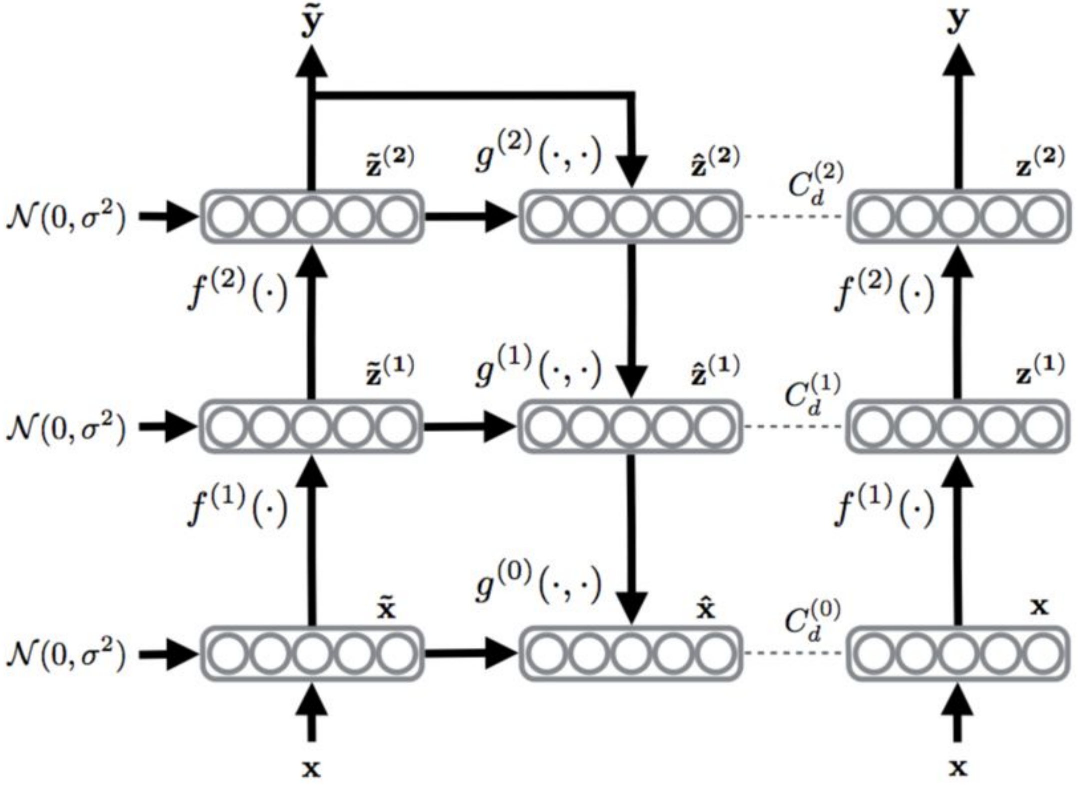


Semi-supervised example: ladder networks

Combine supervised and unsupervised objectives and train together

- Clean path and noisy path
- Decoder which can invert the mappings on each layer
- Loss is weighted sum of supervised and unsupervised cost

1.13% error on permutation invariant MNIST with only 100 examples



Rasmus et al. **Semi-Supervised Learning with Ladder Networks.** NIPS 2015. http://arxiv.org/abs/1507.02672

Transfer learning: idea

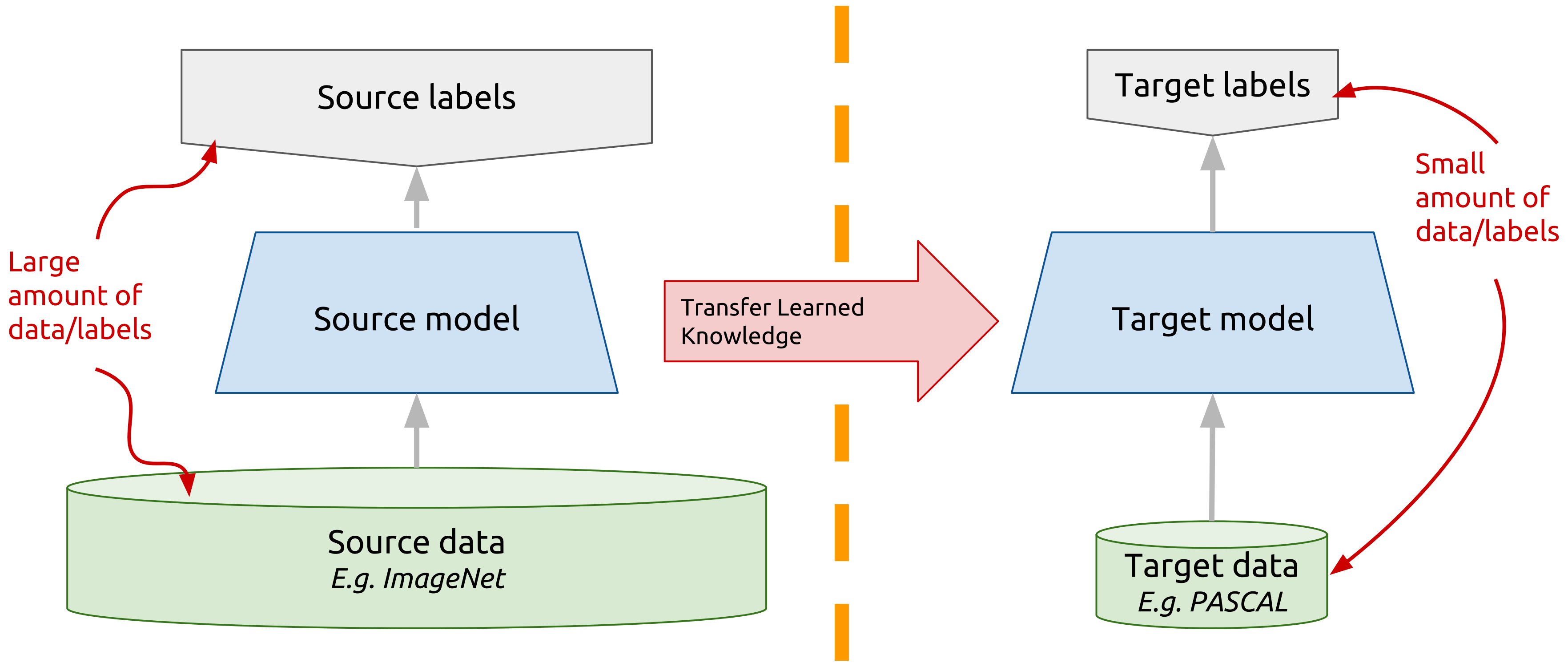
Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

Variations:

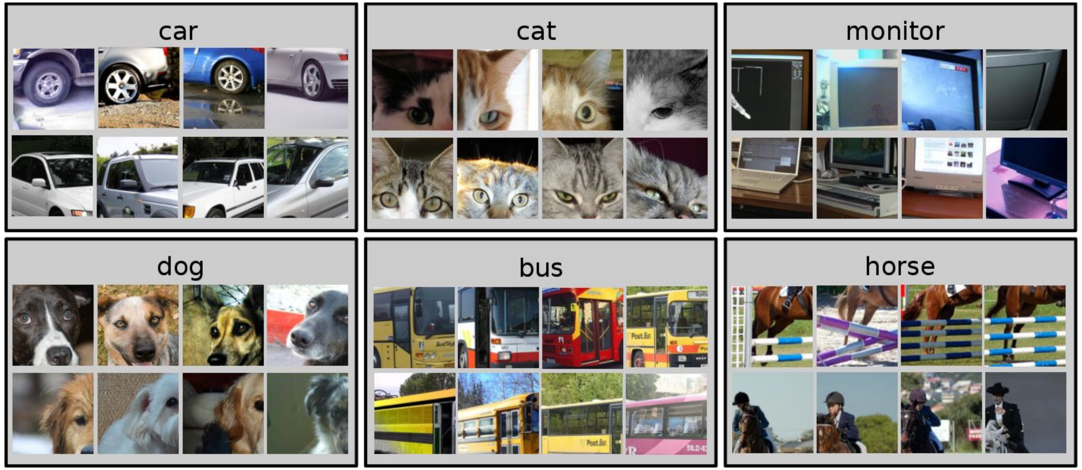
- Same domain, different task
- Different domain, same task

Transfer learning: idea



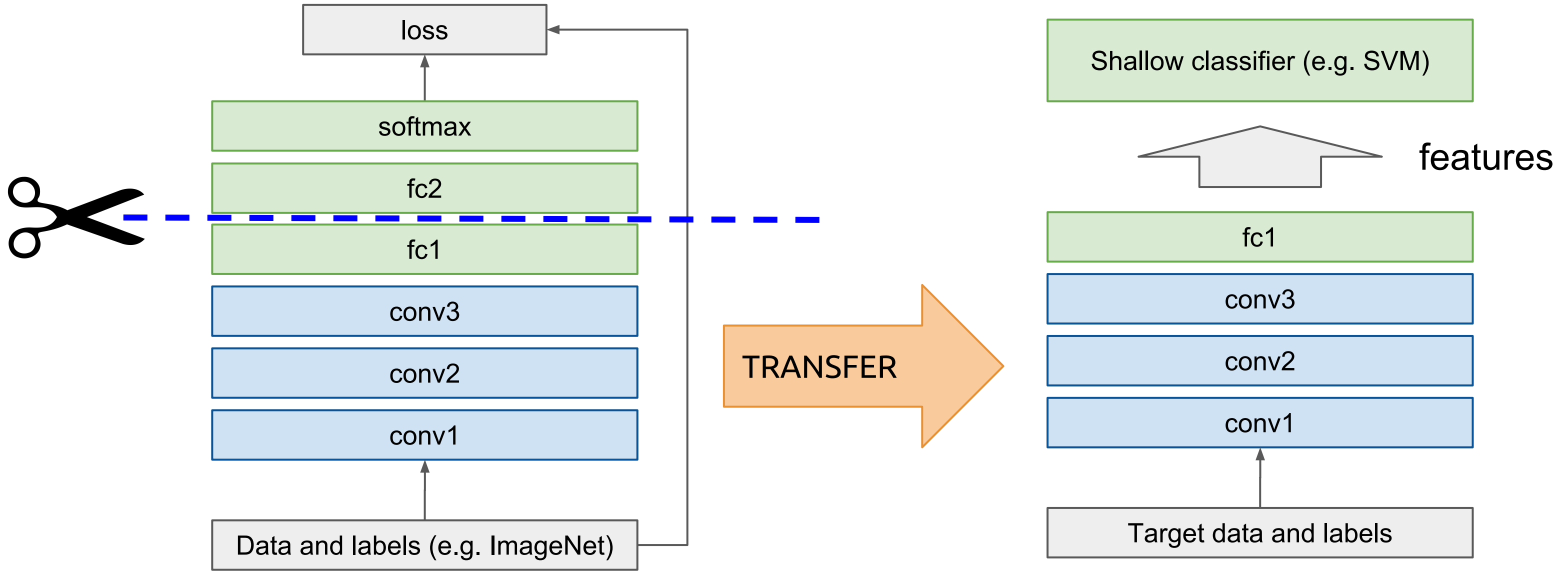
Example: PASCAL VOC 2007

- Standard classification benchmark, 20 classes, ~10K images, 50% train, 50% test
- Deep networks can have many parameters (e.g. 60M in Alexnet)
- Direct training (from scratch) using only 5K training images can be problematic. Model overfits.
- How can we use deep networks in this setting?



“Off-the-shelf”

Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.



Off-the-shelf features

Works surprisingly well in practice!

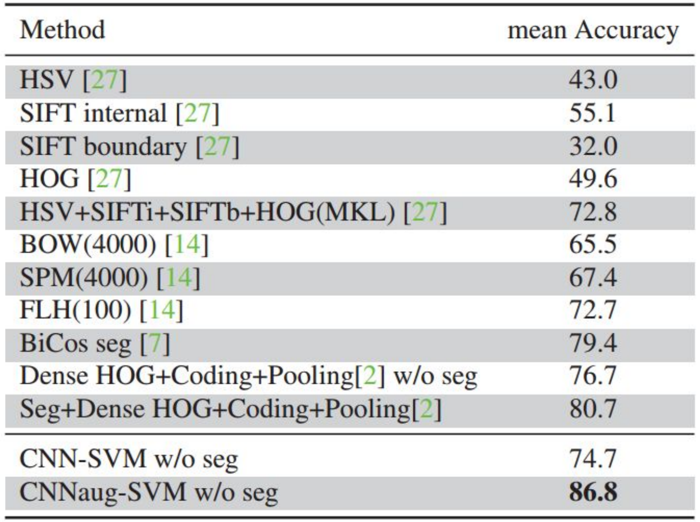
Surpassed or on par with state-of-the-art in several tasks in 2014

Image classification:

- PASCAL VOC 2007
- Oxford flowers
- CUB Bird dataset
- MIT indoors

Image retrieval:

- Paris 6k
- Holidays
- UKBench



Method	mean Accuracy
HSV [27]	43.0
SIFT internal [27]	55.1
SIFT boundary [27]	32.0
HOG [27]	49.6
HSV+SIFTi+SIFTb+HOG(MKL) [27]	72.8
BOW(4000) [14]	65.5
SPM(4000) [14]	67.4
FLH(100) [14]	72.7
BiCos seg [7]	79.4
Dense HOG+Coding+Pooling[2] w/o seg	76.7
Seg+Dense HOG+Coding+Pooling[2]	80.7
CNN-SVM w/o seg	74.7
CNNaug-SVM w/o seg	**86.8**

Oxford 102 flowers dataset

Razavian et al, **CNN Features off-the-shelf: an Astounding Baseline for Recognition,** CVPRW 2014 http://arxiv.org/abs/1403.6382

Can we do better than off the shelf features?

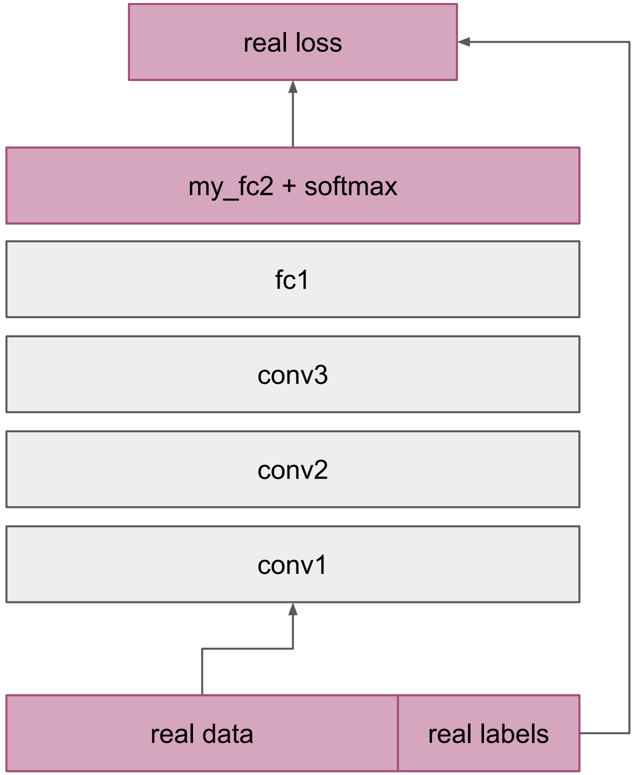
Fine-tuning: supervised task adaptation

Train deep net on “nearby” task for which it is easy to get labels using standard backprop

- E.g. ImageNet classification
- Pseudo classes from augmented data
- Slow feature learning, ego-motion

Cut off top layer(s) of network and replace with supervised objective for target domain

Fine-tune network using backprop with labels for target domain until validation loss starts to increase



Freeze or fine-tune?

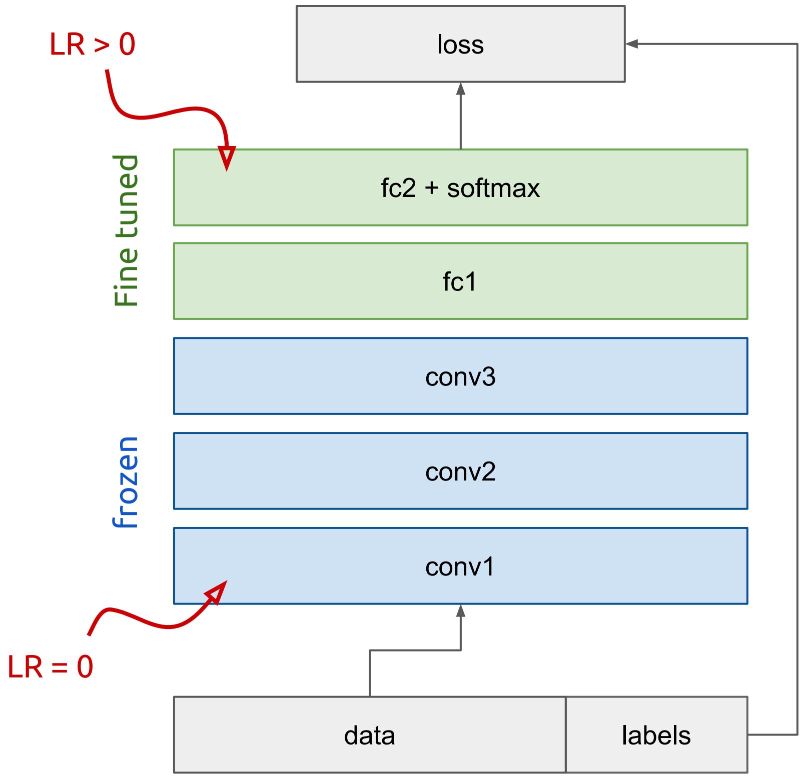
Bottom *n* layers can be frozen or fine tuned.

- **Frozen**: not updated during backprop
- **Fine-tuned**: updated during backprop

Which to do depends on target task:

- **Freeze**: target task labels are scarce, and we want to avoid overfitting
- **Fine-tune**: target task labels are more plentiful

In general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine tuning



How transferable are features?

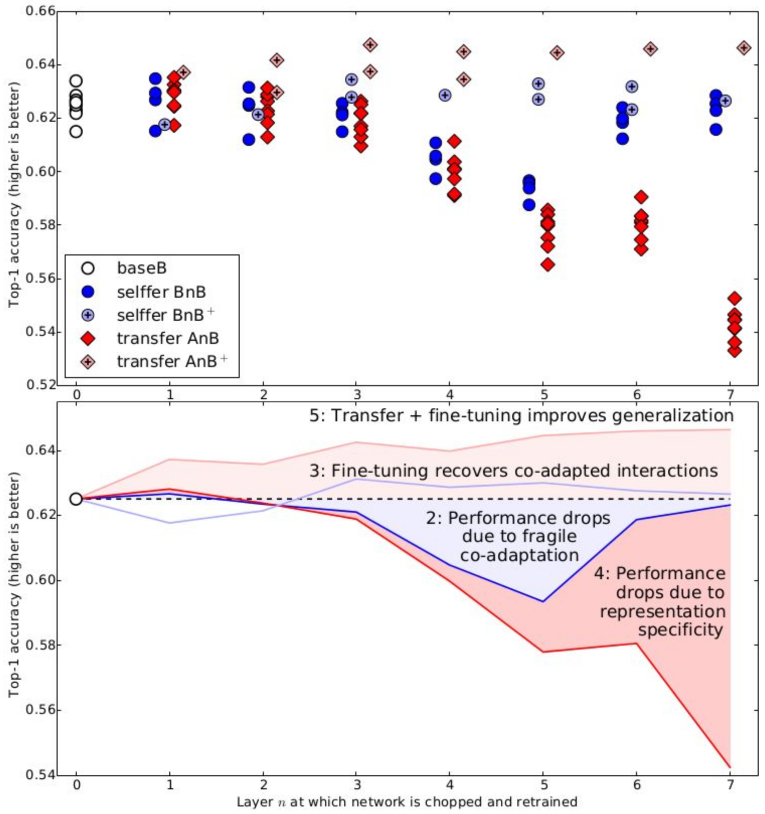
Lower layers: more general features. Transfer very well to other tasks.

Higher layers: more task specific.

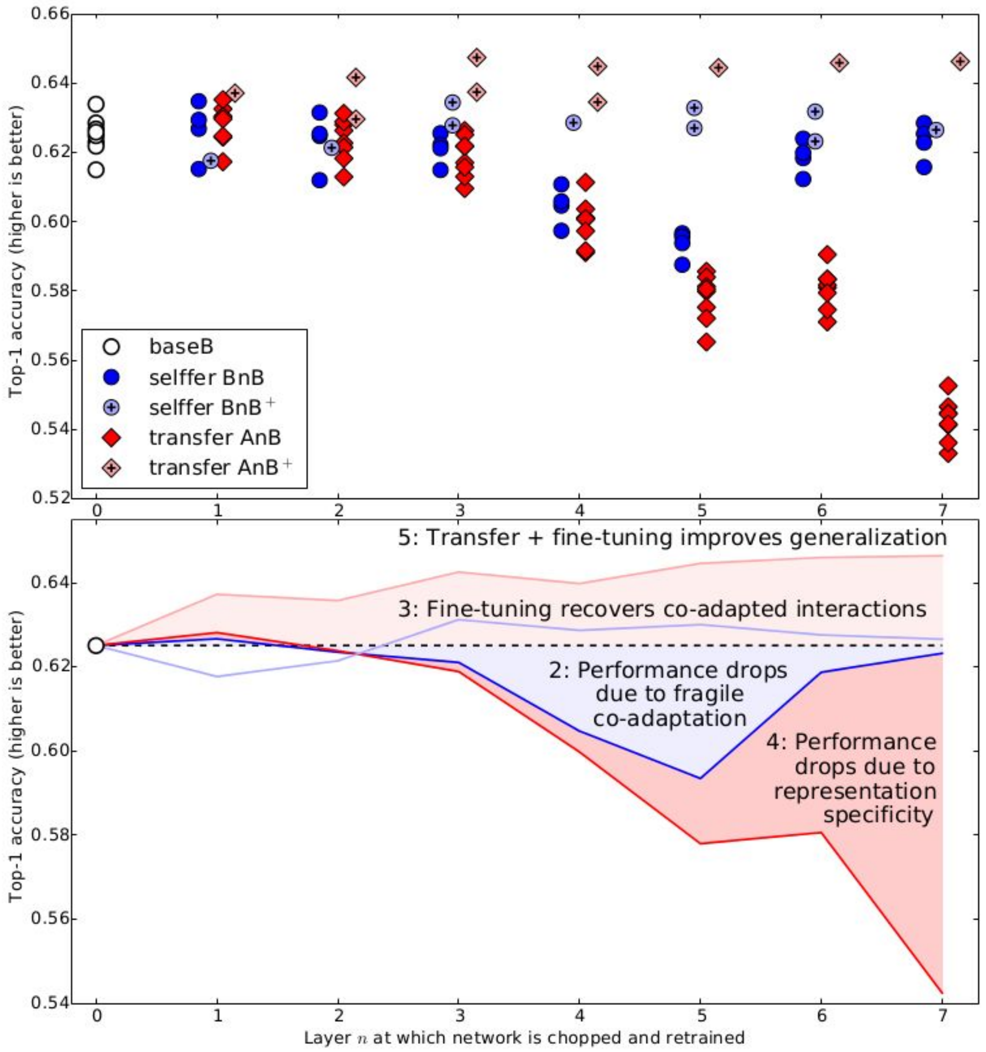
Fine-tuning improves generalization when sufficient examples are available.

Transfer learning and fine tuning often lead to better performance than training from scratch on the target dataset.

Even features transferred from distant tasks are often better than random initial weights!

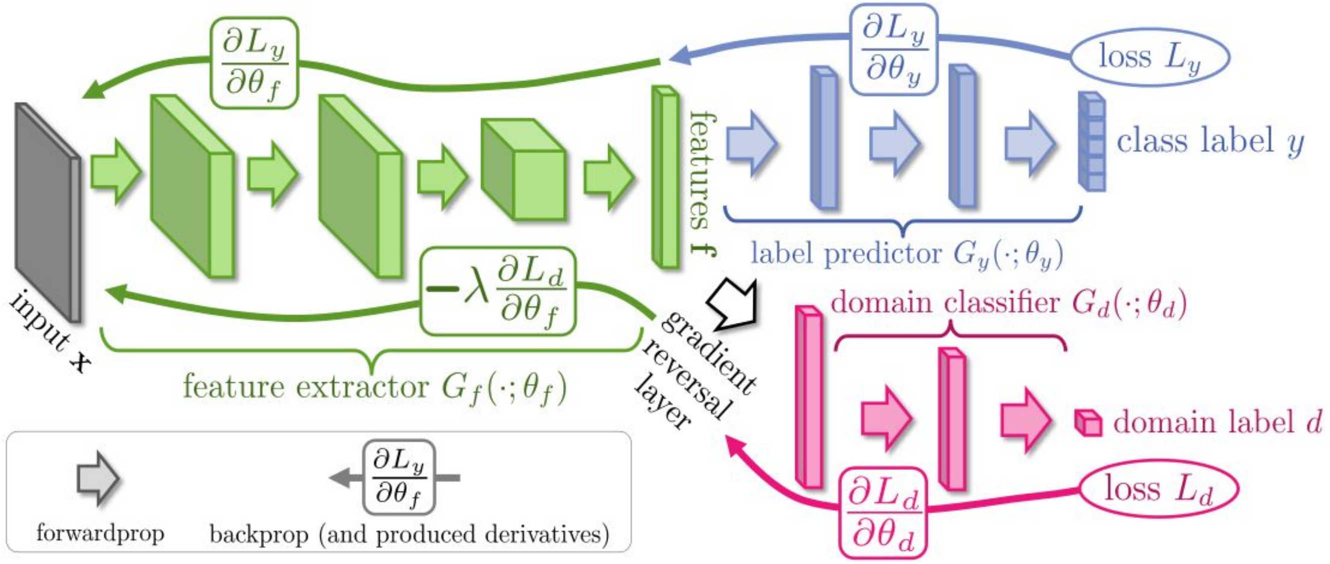


Yosinki et al. **How transferable are features in deep neural networks.** NIPS 2014. https://arxiv.org/abs/1411.1792

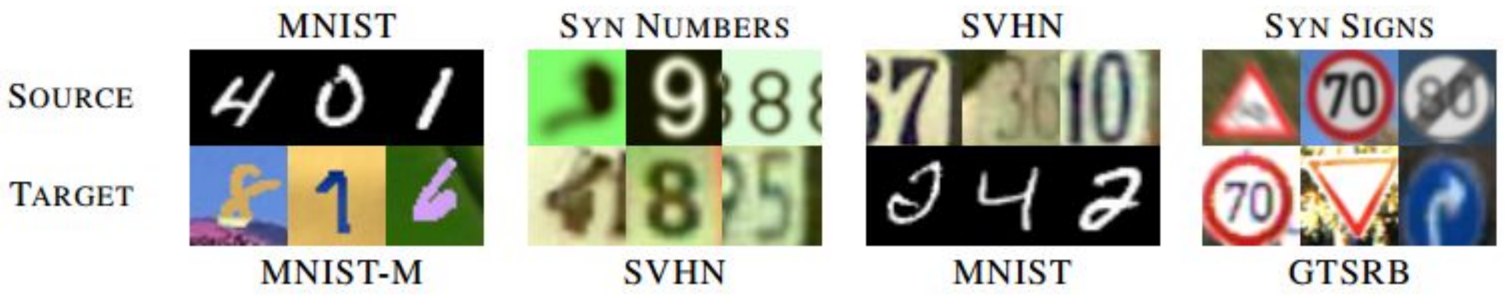

Top-1 accuracy (higher is better)
0.52
0.54
0.56
0.58
0.60
0.62
0.64
0.66
0
1
2
3
4
5
6
7
baseB
selffer BnB
selffer BnB$^+$
transfer AnB
transfer AnB$^+$
5: Transfer + fine-tuning improves generalization
3: Fine-tuning recovers co-adapted interactions
2: Performance drops due to fragile co-adaptation
4: Performance drops due to representation specificity
Layer n at which network is chopped and retrained

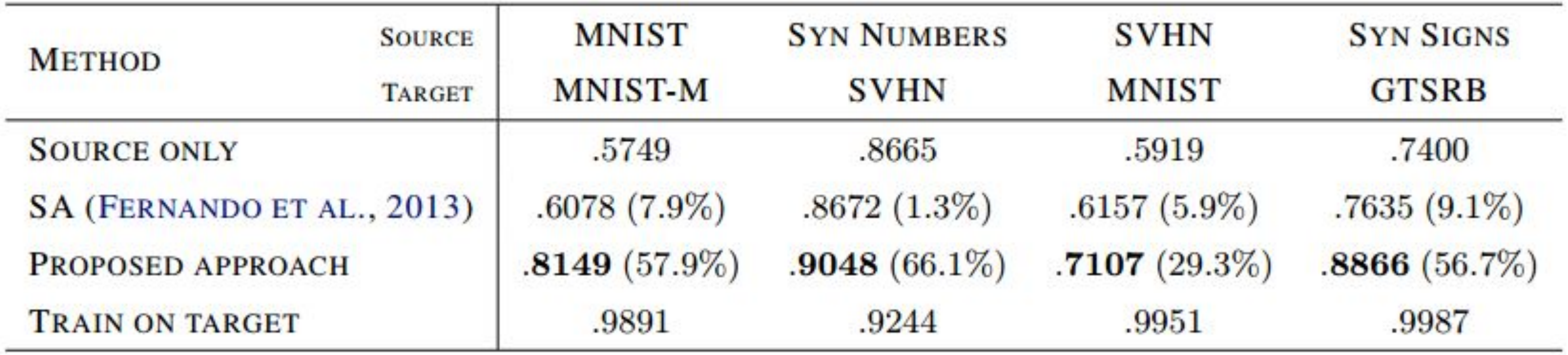
Unsupervised domain adaptation

Also possible to do domain adaptation without labels in target set.


$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features $\mathbf{f}$
class label y
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
loss L_d
$\frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Unsupervised domain adaptation





METHOD	SOURCE / TARGET	MNIST / MNIST-M	SYN NUMBERS / SVHN	SVHN / MNIST	SYN SIGNS / GTSRB
SOURCE ONLY		.5749	.8665	.5919	.7400
SA (FERNANDO ET AL., 2013)		.6078 (7.9%)	.8672 (1.3%)	.6157 (5.9%)	.7635 (9.1%)
PROPOSED APPROACH		**.8149** (57.9%)	**.9048** (66.1%)	**.7107** (29.3%)	**.8866** (56.7%)
TRAIN ON TARGET		.9891	.9244	.9951	.9987

Y Ganin and V Lempitsky, **Unsupervised Domain Adaptation by Backpropagation,** ICML 2015 https://arxiv.org/abs/1409.7495

Semi-supervised domain adaptation

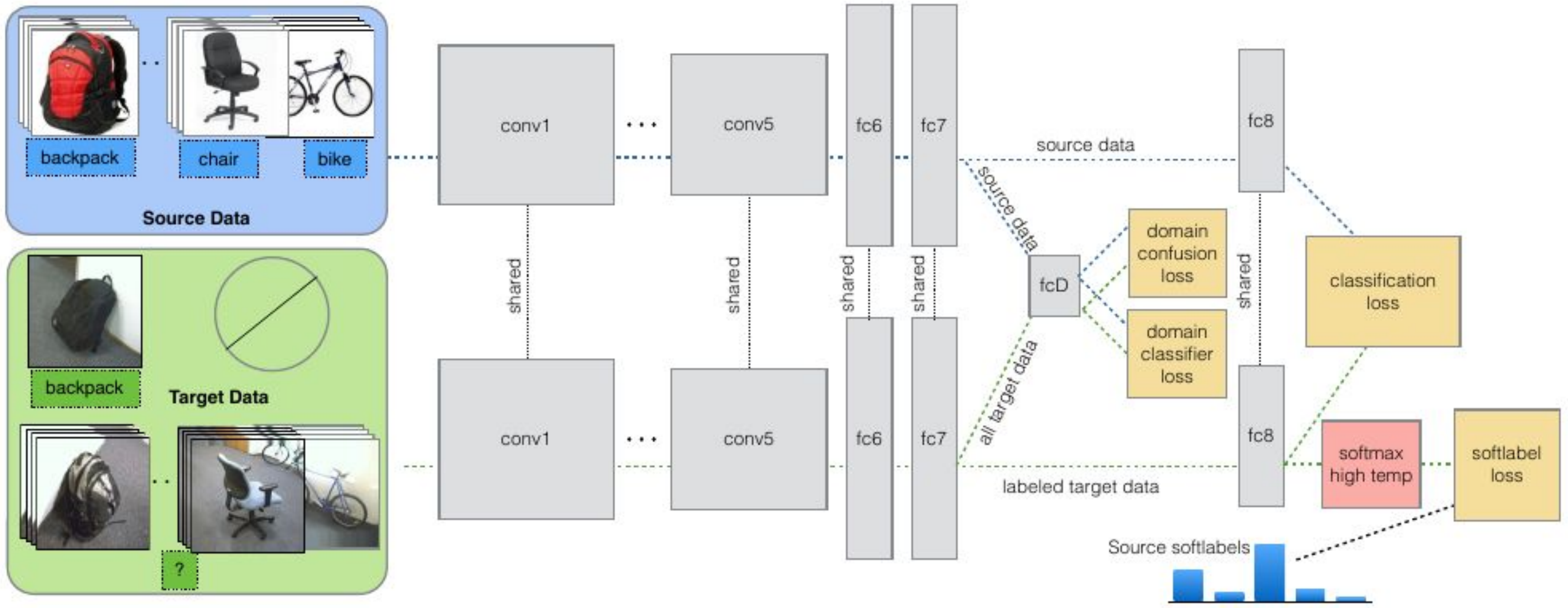
When some labels are available in the target domain, then we can use these when doing domain adaptation. I.e. combine fine tuning and unsupervised domain adaptation.

Tzeng *et al.* take this a step further and try to simultaneously optimize a loss that maximizes:

1. classification accuracy on both source and target datasets
2. domain confusion of a domain classifier
3. agreement of classifier score distributions across domains

Tzeng, Eric, et al. Simultaneous deep transfer across domains and tasks. ICCV. 2015.

Semi-supervised domain adaptation



Tzeng, Eric, et al. Simultaneous deep transfer across domains and tasks. ICCV. 2015.

Semi-supervised domain adaptation

$$\begin{aligned}\mathcal{L}(x_S, y_S, x_T, y_T, \theta_D; \theta_{\text{repr}}, \theta_C) = \\ & \mathcal{L}_C(x_S, y_S, x_T, y_T; \theta_{\text{repr}}, \theta_C) \\ & + \lambda \mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}}) \\ & + \nu \mathcal{L}_{\text{soft}}(x_T, y_T; \theta_{\text{repr}}, \theta_C).\end{aligned}$$

Domain confusion loss

Classifier loss

Soft label loss to align classifier scores across domains

Domain confusion loss

Alternate optimization of two objectives (like adversarial training). First makes domain classifier as good as possible. Standard **binary cross entropy** loss:

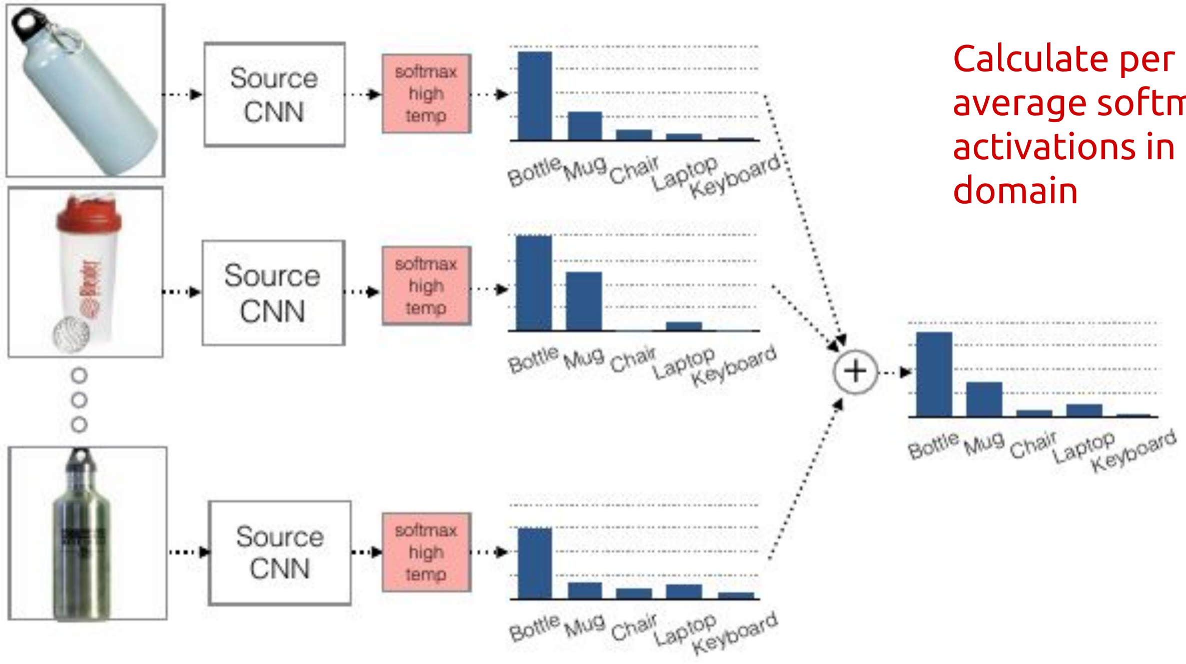
$$\mathcal{L}_D(x_S, x_T, \theta_{\text{repr}}; \theta_D) = -\sum_d \mathbb{1}[y_D = d] \log q_d$$

Second makes features as confusing as possible for the discriminator:

$$\mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}}) = -\sum_d \frac{1}{D} \log q_d.$$

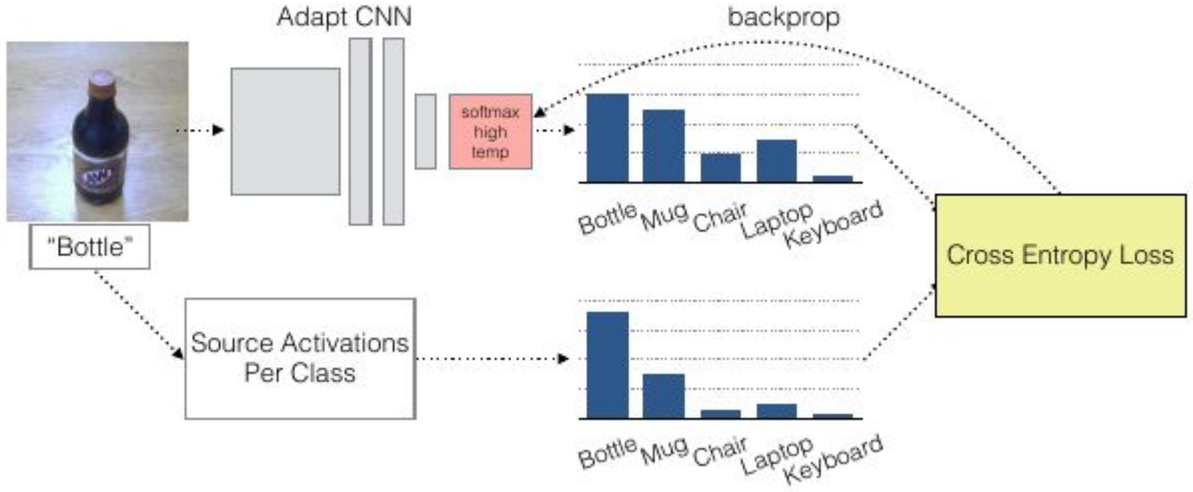
Alignment of source and target predictions

"bottle"



Calculate per class average softmax activations in source domain

Alignment of source and target predictions

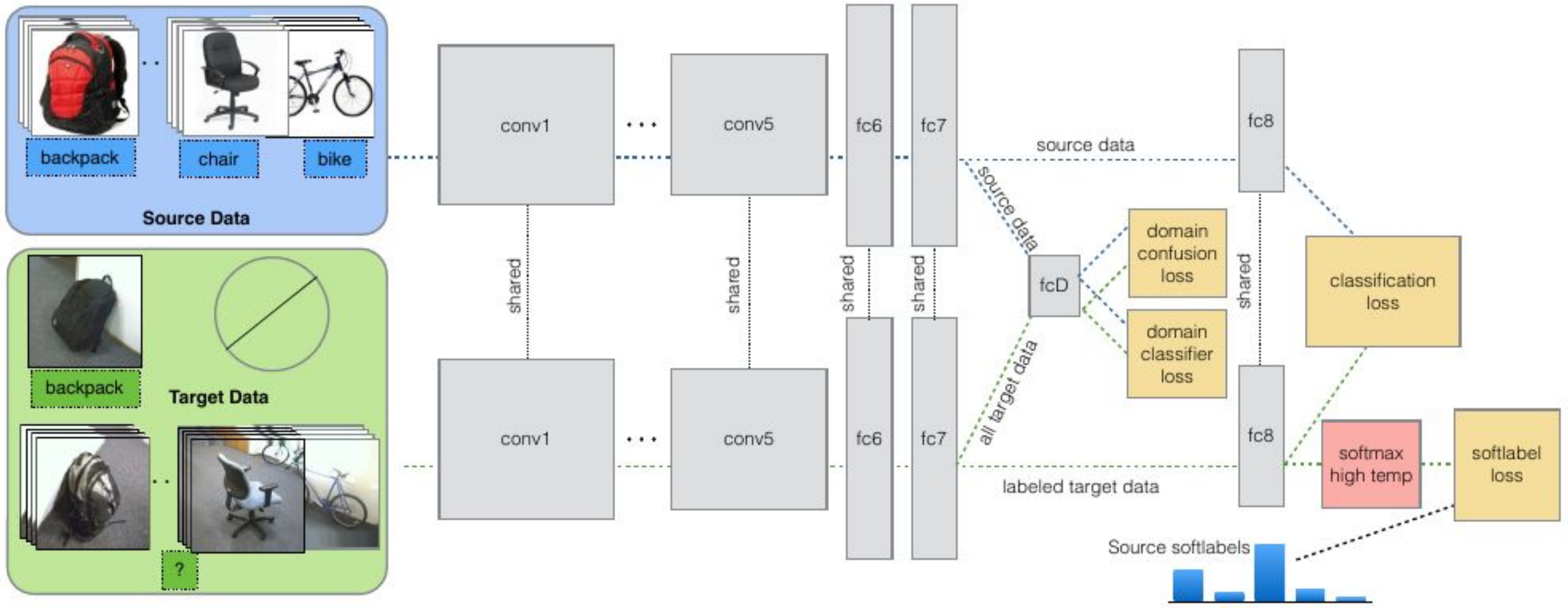


Use these as the target distribution for target domain.

Minimizing cross entropy loss same as minimizing KL divergence!

$$\mathcal{L}_{\text{soft}}(x_T, y_T; \theta_{\text{repr}}, \theta_C) = -\sum_i l_i^{(y_T)} \log p_i$$

Semi-supervised domain adaptation



Tzeng, Eric, et al. Simultaneous deep transfer across domains and tasks. ICCV. 2015.

How are they related?

What do unsupervised learning, semi-supervised learning, and transfer learning have in common?

We often do not have complete access to the *y*-value that we care about.

Use a **surrogate objective** when *y*-value unavailable:

- Loss you do not care directly about.
- You expect that improving this loss will help with what you do care about.

Transfer learning uses different dataset in which there are lots of *y*-values to create a surrogate objective.

Unsupervised tries to formulate surrogate objectives without any manual labels.

Summary

- Possible to train very large models on small data by using transfer learning and domain adaptation
- Off the shelf features work very well in various domains and tasks
- Lower layers of network contain very generic features, higher layers more task specific features
- Supervised domain adaptation via fine tuning almost always improves performance
- Possible to do unsupervised domain adaptation by matching feature distributions

Questions?

Additional resources

- Lluis Castrejon, “Domain adaptation and zero-shot learning”. University of Toronto 2016.
- Hoffman, J., Guadarrama, S., Tzeng, E. S., Hu, R., Donahue, J., Girshick, R., ... & Saenko, K. (2014). LSDA: Large scale detection through adaptation. NIPS 2014. (Slides by Xavier Giró-i-Nieto)
- Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?." In Advances in Neural Information Processing Systems, pp. 3320-3328. 2014.
- Shao, Ling, Fan Zhu, and Xuelong Li. "Transfer learning for visual categorization: A survey." Neural Networks and Learning Systems, IEEE Transactions on 26, no. 5 (2015): 1019-1034.
- Chen, Tianqi, Ian Goodfellow, and Jonathon Shlens. "Net2Net: Accelerating Learning via Knowledge Transfer." ICLR 2016. [code] [Notes by Hugo Larrochelle
- Gani, Yaroslav, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. "Domain-Adversarial Training of Neural Networks." arXiv preprint arXiv:1505.07818 (2015).