#DLUPC

Day 5 Lecture 2

Transfer learning and domain adaptation

Ramon Morros
ramon.morros@upc.edu

Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

Many slides from:

Kevin McGuinness
kevin.mcguinness@dcu.ie

Research Fellow
Insight Centre for Data Analytics
Dublin City University

Eric Arazo
eric.arazosanchez@dcu.ie

PhD Candidate
Insight Centre for Data Analytics
Dublin City University

[course site]

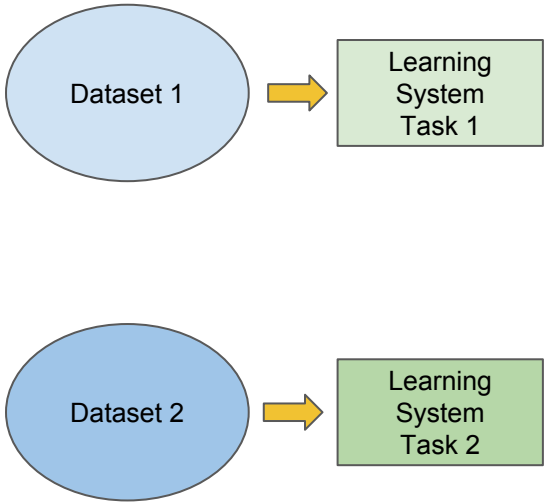
Transfer Learning

The ability to apply knowledge learned in previous tasks to novel tasks

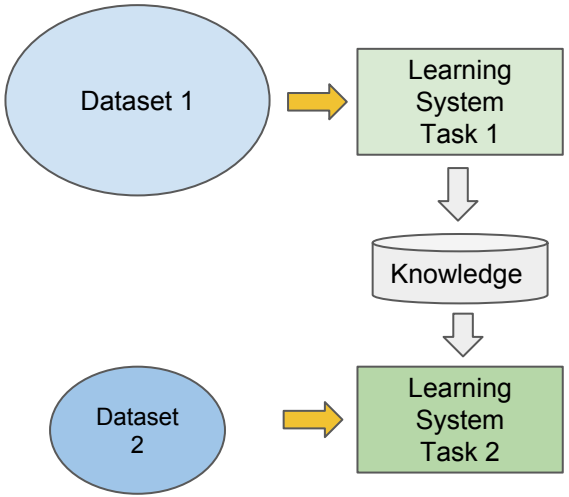
- Based on human learning. People can often transfer knowledge learnt previously to novel situations
    - Play classic piano ➔ Play jazz piano
    - Maths ➔ Machine Learning
    - Ride motorbike ➔ Drive a car

Traditional ML vs Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



Transfer learning in DL

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

Transfer learning: idea

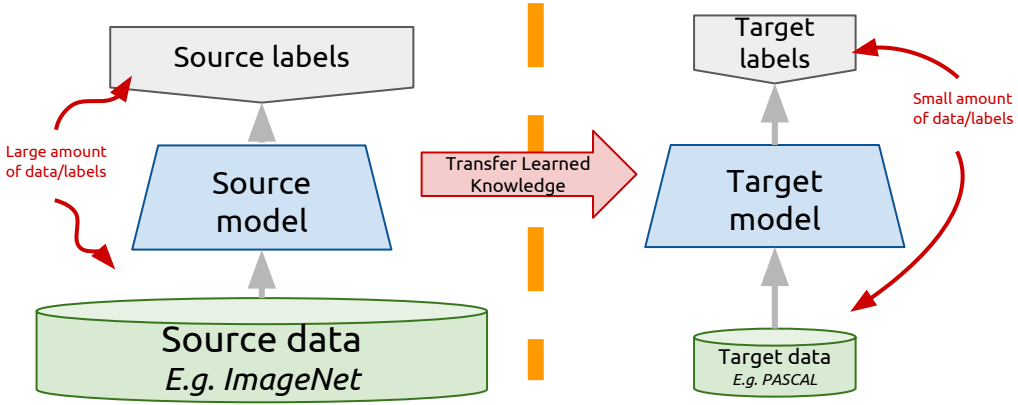
Instead of training a deep network from scratch for your task:

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

This lecture will talk about how to do this.

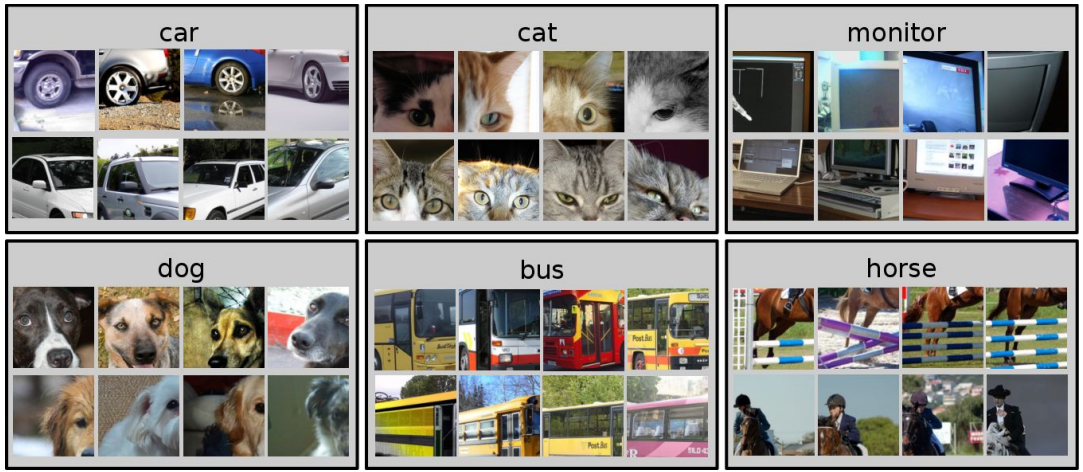
Variations:

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



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?



Transfer Learning

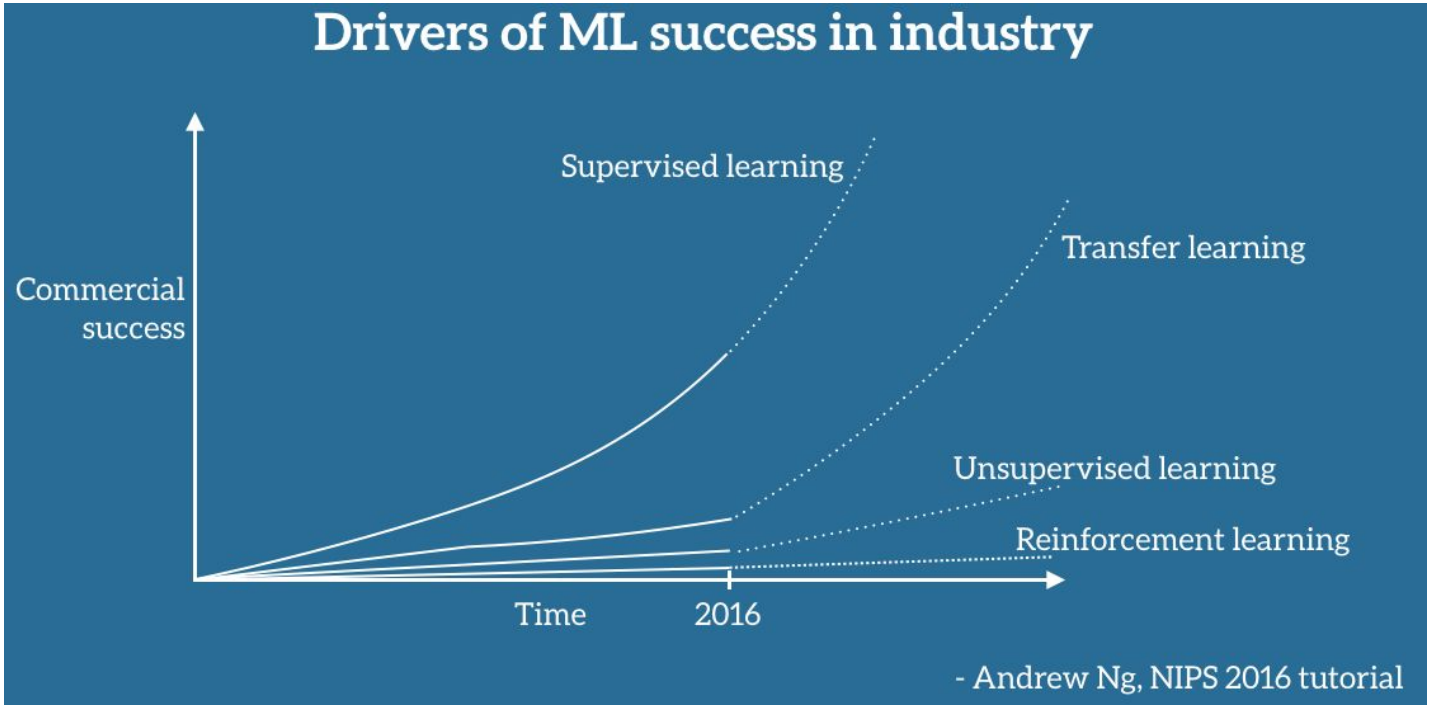


Figure extracted from Sebastian Rudder's blog: "Transfer Learning - Machine Learning's Next Frontier" http://ruder.io/transfer-learning/index.html

Notation (I)

A **Domain** consists of two components: $D = \{\mathcal{X}, P(X)\}$

- Feature space: $\mathcal{X}$
- Marginal distribution: $P(X),\ X = \{x_1, \cdots, x_n\}, x_i \in \mathcal{X}$

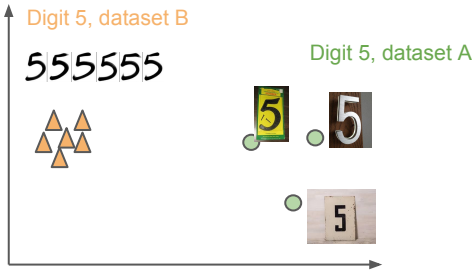
For a given domain **D**, a **Task** is defined by two components:

$$T = \{\mathcal{Y}, P(Y|X)\} = \{\mathcal{Y}, \eta\} \qquad Y = \{y_1, \cdots, y_n\}, y_i \in \mathcal{Y}$$

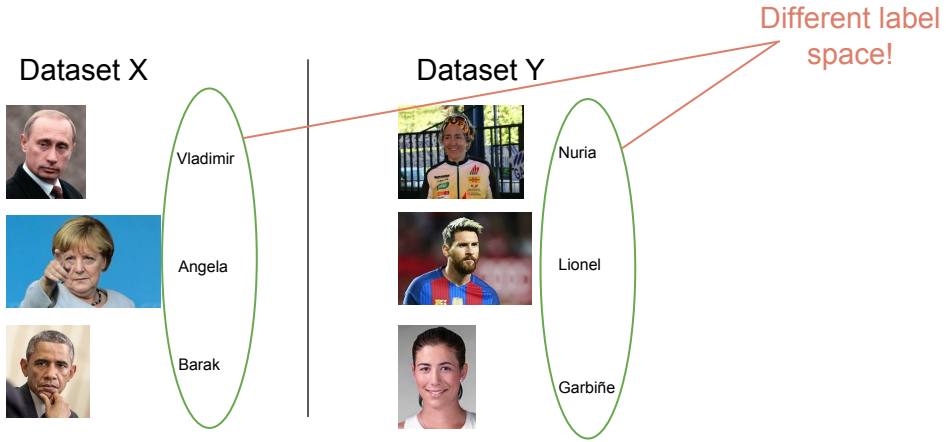
- A label space: $\mathcal{Y}$
- A predictive function η, learned from *feature vector/label* pairs, $(x_i, y_i),\ x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- For each feature vector in the domain, η predicts its corresponding label: $\eta(x_i) = y_i$

Notation (II)

If two domains are different, they may have different feature spaces or different marginal distributions



If two tasks are different, they may have different label spaces or different conditional distributions



Notation (III)

For simplicity, only two domains or two tasks are usually considered

- Source domain

$$D_S = \{\mathcal{X}_S, P(X_S)\},\ X_S = \{x_{S_1}, \cdots, x_{S_n}\}, \quad x_i \in \mathcal{X}_S$$

- Task on the source domain

$$T_S = \{\mathcal{Y}_S, P(Y_S|X_S)\},\ Y_S = \{y_{S_1}, \cdots, y_{S_n}\}, \quad y_i \in \mathcal{Y}_S$$

- Target domain

$$D_T = \{\mathcal{X}_T, P(X_T)\},\ X_T = \{x_{T_1}, \cdots, x_{T_n}\}, \quad x_i \in \mathcal{X}_T$$

- Task on the target domain

$$T_T = \{\mathcal{Y}_T, P(Y_S|X_T)\},\ Y_T = \{y_{T_1}, \cdots, y_{T_n}\}, \quad y_i \in \mathcal{Y}_T$$

Domain adaptation

Consider a classification task where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the set of labels. Given two sets of samples drawn from the source and target domains:

$$D_S = \{(x_i, y_i)\}_{i=1}^{n} \sim P(X_S), \quad D_T = \{(x_i, y_i)\}_{i=n+1}^{N} \sim P(X_T)$$

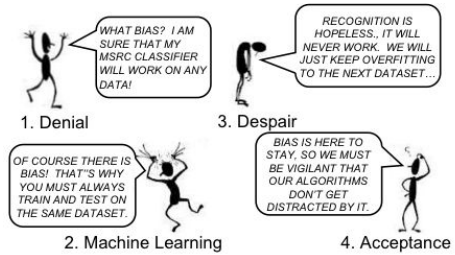
(Target space labeled data may not be present in unsupervised case)

The goal of the learning algorithm is to build a classifier $\eta : \mathcal{X} \to \mathcal{Y}$ with a low target risk on the target domain

$$R_{D_T}(\eta) = \Pr_{(x,y)\sim D_T}(\eta(x) \neq y)$$

Domain biass

- Datasets are samples of the world
- In many cases, there is a shift or bias in the distributions of the source and target data representations

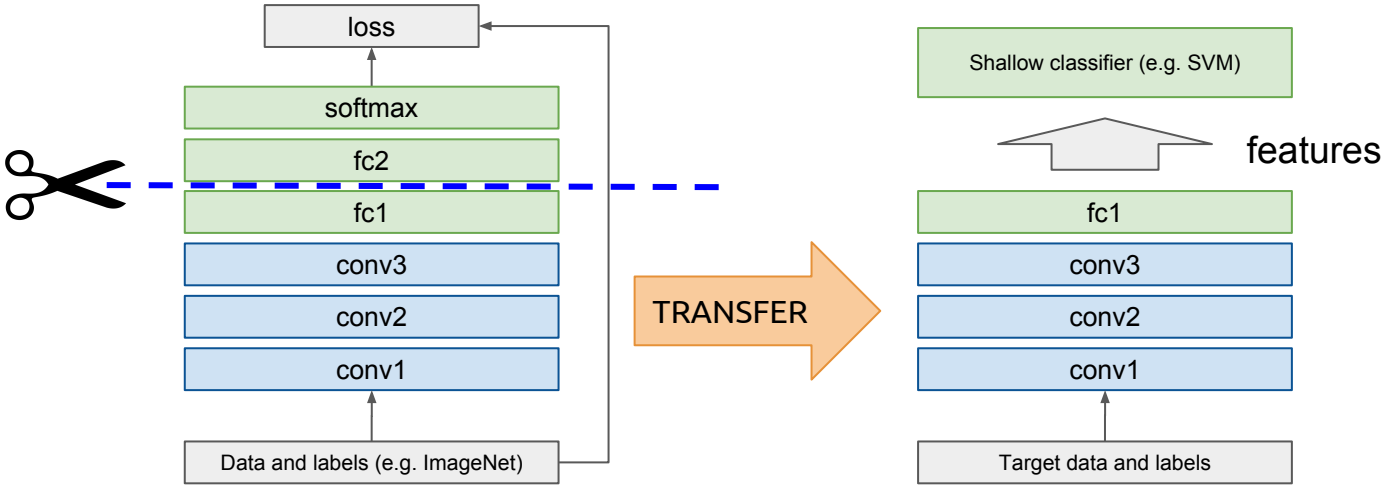


A. Torralba, A. Efros. Unbiased Look at Dataset Bias. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

“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.

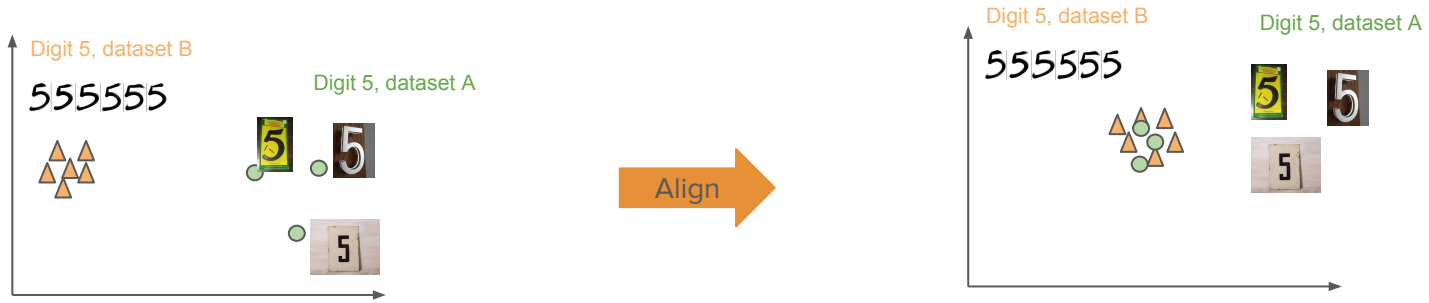
Assumes that $D_S = D_T$



Domain adaptation

When there is a domain shift

- The size of this shift is often measured by the distance between source and target subspaces
- A typical approach is to learn a feature space transformation to align the source and target representations (reduce domain divergence)



Off-the-shelf features

Works surprisingly well in practice!

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

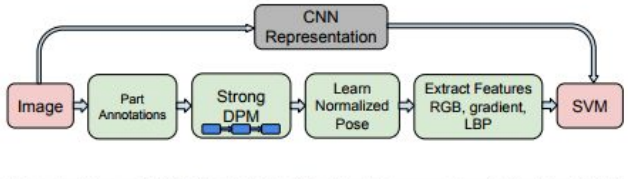
Image classification:

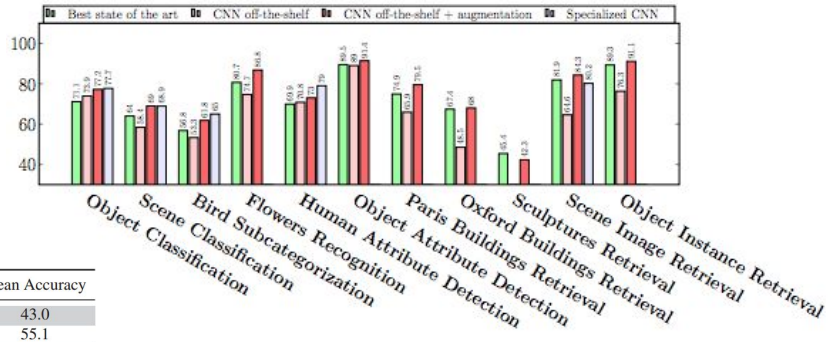
- PASCAL VOC 2007
- Oxford flowers
- CUB Bird dataset
- MIT indoors

Image retrieval:

- Paris 6k
- Holidays
- UKBench

(Trained to perform object classification on ILSVRC13)





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 domain adaptation

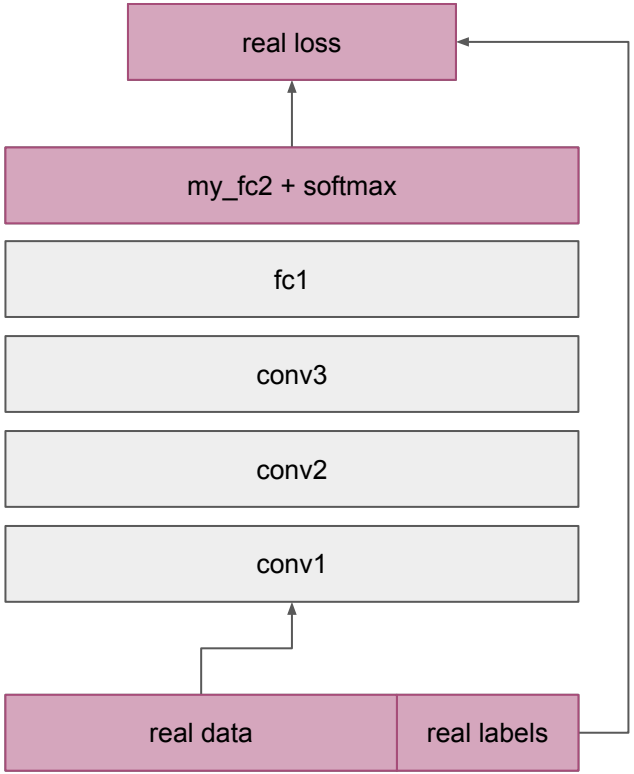
Train deep net on “nearby” task for which it is easy to get labels using standard backprop

- 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

Aligns D_S with D_T



Freeze or fine-tune?

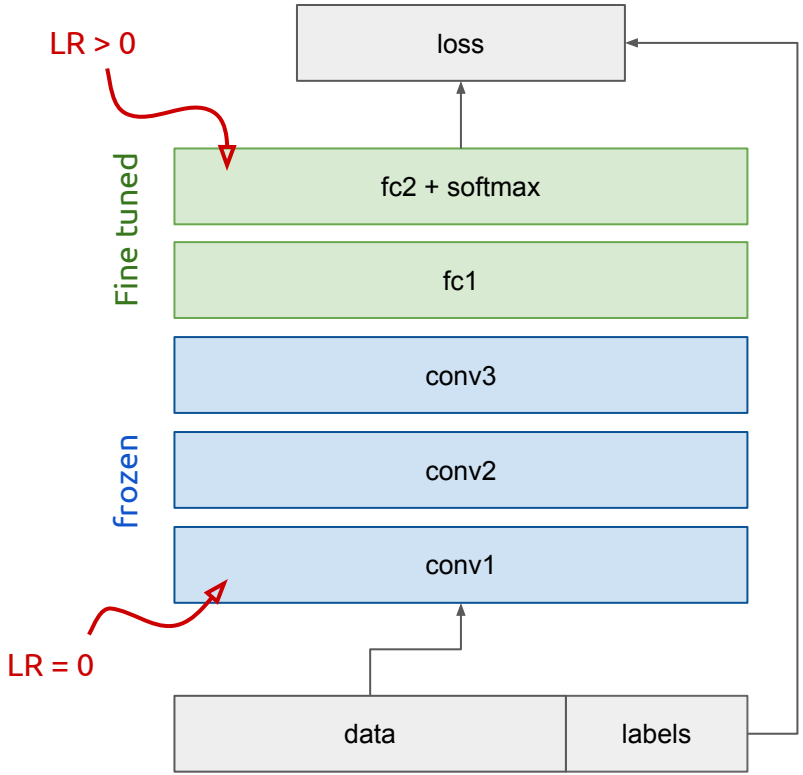
Bottom *n* layers can be frozen or fine tuned.

- **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

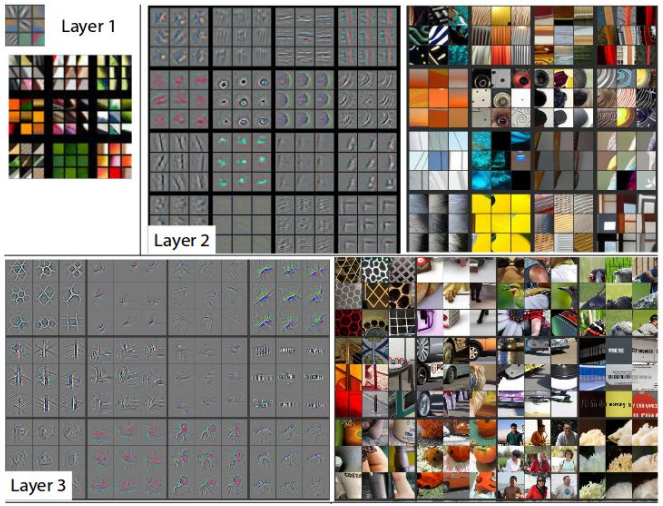
In general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine tuning

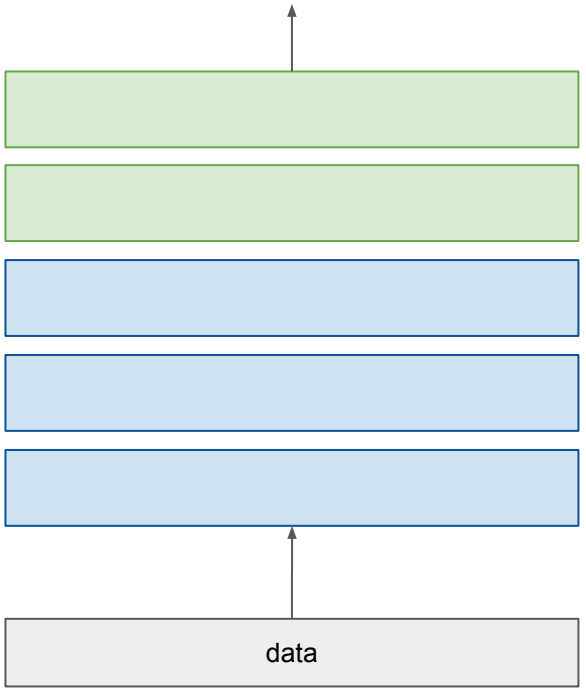


How transferable are features?

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

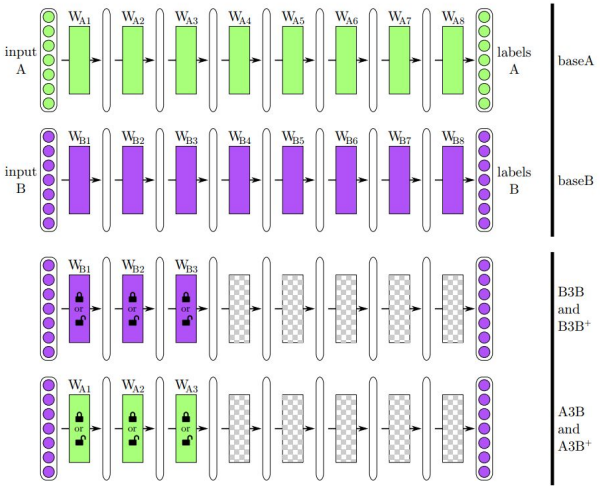
Higher layers: more task specific.

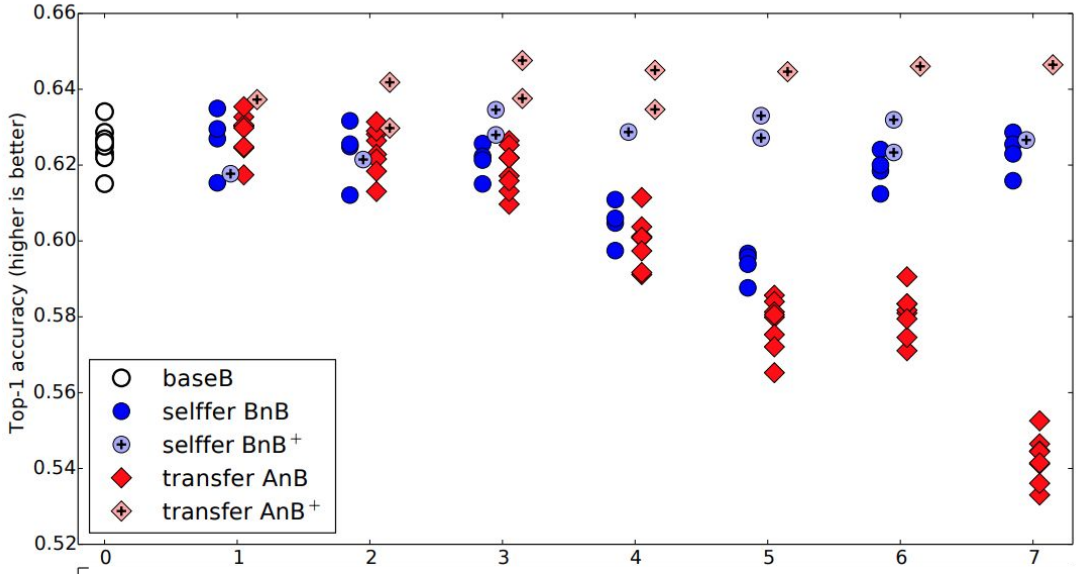




Matthew D Zeiler, Rob Fergus, "Visualizing and Understanding Convolutional Networks", CVPR 2013

How transferable are features?





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

How transferable are features?

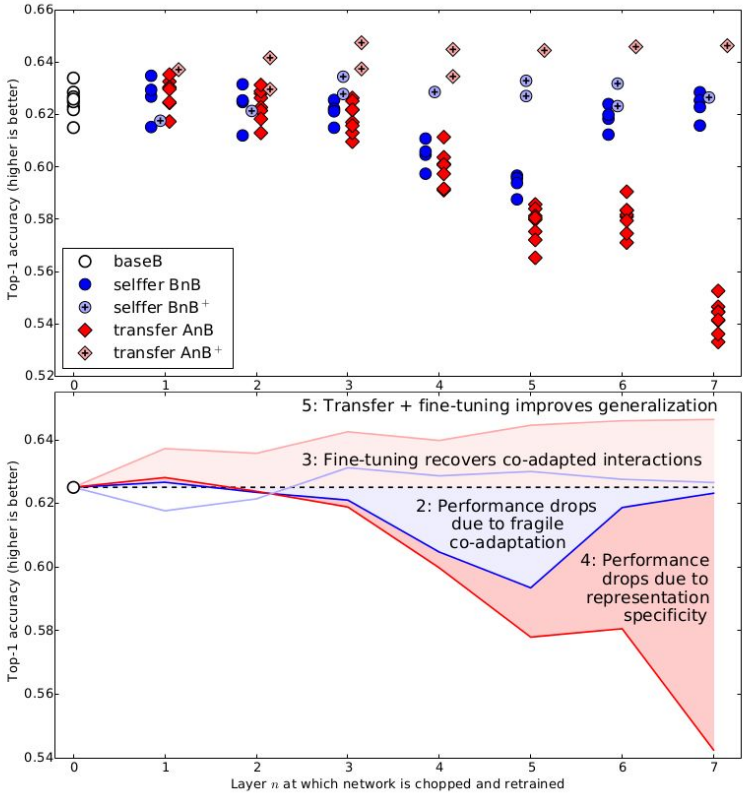
Transferability is negatively affected by two distinct issues:

- The specialization of higher layer neurons
- Optimization difficulties related to splitting networks between co-adapted neurons

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.

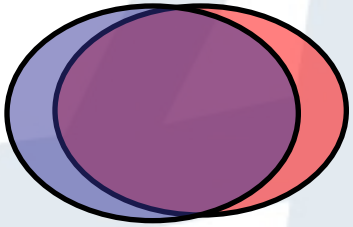
Even features transferred from distant tasks are often better than random initial weights!



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

Single-Task Transfer Learning

Case 1



Sample Selection Bias / Covariate Shift

Instance-based Transfer Learning Approaches

Problem Setting

Given $\mathrm{D}_S = \{x_{S_i}, y_{S_i}\}_{i=1}^{n_S}$, $\mathrm{D}_T = \{x_{T_i}\}_{i=1}^{n_T}$,

Learn f_T, s.t. $\sum_i \epsilon(f_T(x_{T_i}), y_{T_i})$ is small,

where y_{T_i} is unknown.

Assumption

- $\mathcal{Y}_S = \mathcal{Y}_T$, and $P(Y_S|X_S) = P(Y_T|X_T)$,
- $\mathcal{X}_S \approx \mathcal{X}_T$,
- $P(X_S) \neq P(X_T)$.

A*STAR I²R

Single-Task Transfer Learning

Instance-based Approaches

Recall, given a target task,

$$
\begin{aligned}
\theta^* &= \arg\min \mathbb{E}_{(x,y)\sim P_T}[l(x,y,\theta)] \\
&= \arg\min \mathbb{E}_{(x,y)\sim P_T}\left[\frac{P_S(x,y)}{P_S(x,y)} l(x,y,\theta)\right] \\
&= \arg\min \int_y \int_x P_T(x,y)\left(\frac{P_S(x,y)}{P_S(x,y)} l(x,y,\theta)\right) dxdy \\
&= \arg\min \int_y \int_x P_S(x,y)\left(\frac{P_T(x,y)}{P_S(x,y)} l(x,y,\theta)\right) dxdy \\
&= \arg\min \mathbb{E}_{(x,y)\sim P_S}\left[\frac{P_T(x,y)}{P_S(x,y)} l(x,y,\theta)\right]
\end{aligned}
$$

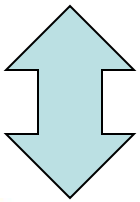
A*STAR I²R

Single-Task Transfer Learning

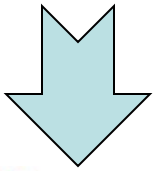
Instance-based Approaches (cont.)

If $P_S(x, y) = P_T(x, y)$

$$\theta^* = \arg\min \mathbb{E}_{(x_T, y_T) \sim P_T}[l(x_T, y_T, \theta)]$$



$$\theta^* = \arg\min \mathbb{E}_{(x_S, y_S) \sim P_S}[l(x_S, y_S, \theta)]$$



$$\theta^* = \arg\min \sum_{i=1}^{n_S} l(x_{S_i}, y_{S_i}, \theta) + \lambda\Omega(\theta)$$

A*STAR I²R

Single-Task Transfer Learning

Instance-based Approaches (cont.)

Assumption: $\{P_S(x) \neq P_T(x),\ P_S(y|x) = P_T(y|x)\} \Rightarrow P_S(x,y) \neq P_T(x,y)$

$$\begin{aligned}
\theta^* &= \arg\min \mathbb{E}_{(x,y)\sim P_S}\left[\frac{P_T(x,y)}{P_S(x,y)} l(x,y,\theta)\right] \\
&= \arg\min \mathbb{E}_{(x,y)\sim P_S}\left[\frac{P_T(x)P_T(y|x)}{P_S(x)P_S(y|x)} l(x,y,\theta)\right] \\
&= \arg\min \mathbb{E}_{(x,y)\sim P_S}\left[\frac{P_T(x)}{P_S(x)} l(x,y,\theta)\right]
\end{aligned}$$

$$\text{Denote } \beta_i = \frac{P_T(x_{S_i})}{P_S(x_{S_i})},$$

$$\theta^* = \arg\min \sum_{i=1}^{n_S} \beta_i l(x_{S_i}, y_{S_i}, \theta) + \lambda\Omega(\theta)$$

Single-Task Transfer Learning

Instance-based Approaches (cont.)

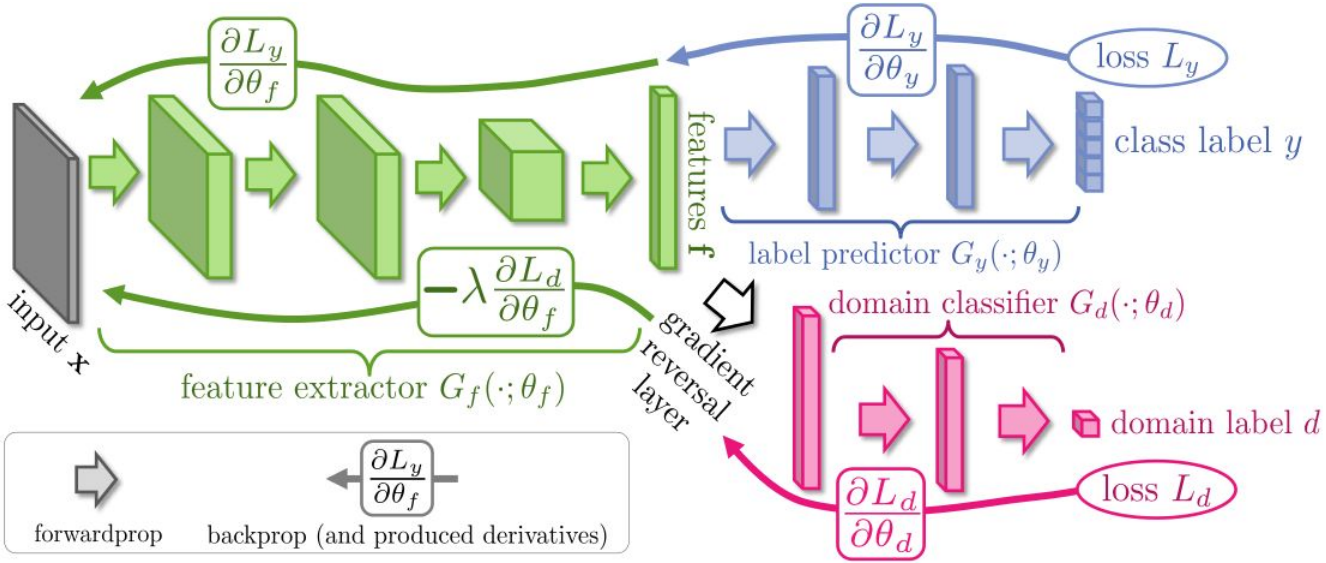
How to estimate $\beta_i = \frac{P_T(x_{S_i})}{P_S(x_{S_i})}$?

A simple solution is to first estimate $P_T(x)$, $P_S(x)$, respectively, and calculate $\frac{P_T(x_{S_i})}{P_S(x_{S_i})}$.

Sample Selection Bias / Covariate Shift
[Quionero-Candela, *etal,* Data Shift in Machine Learning, MIT Press 2009]

Unsupervised domain adaptation

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



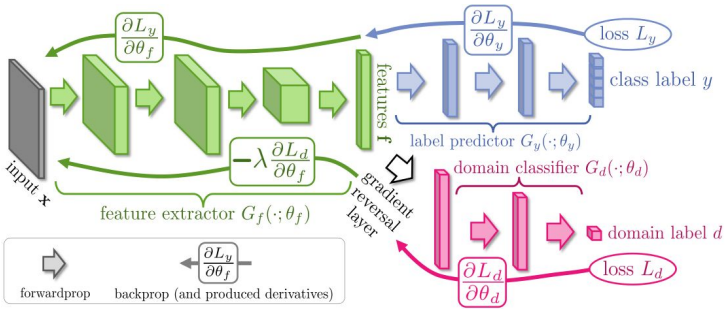
(Source domain)

(Source and target domains)

Y Ganin and V Lempitsky, **Unsupervised Domain Adaptation by Backpropagation,** ICML 2015 https://arxiv.org/abs/1409.7495
Y. Ganin et al. "Domain-Adversarial Training of Neural Networks." Journal of Machine Learning research 17 (2016).

Unsupervised domain adaptation

The approach promotes the emergence of "deep" features that are (i) discriminative for the main learning task on D_S and (ii) invariant with respect to the shift between the domains



loss for label prediction

$$E(\theta_f,\theta_y,\theta_d)=\sum_{\substack{i=1..N\\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y),y_i\right)-$$

$$\lambda\sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d),y_i\right)$$

loss for the domain classification

$$(\hat{\theta}_f,\hat{\theta}_y)=\arg\min_{\theta_f,\theta_y} E(\theta_f,\theta_y,\hat{\theta}_d)$$

Minimize the loss of the domain classifier and the label predictor

$$\hat{\theta}_d=\arg\max_{\theta_d} E(\hat{\theta}_f,\hat{\theta}_y,\theta_d)\,.$$

maximize the loss of the domain classifier (by making the two feature distributions as similar as possible)

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

Y. Ganin,et al. "Domain-Adversarial Training of Neural Networks." Journal of Machine Learning research 17 (2016).

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

Task transfer

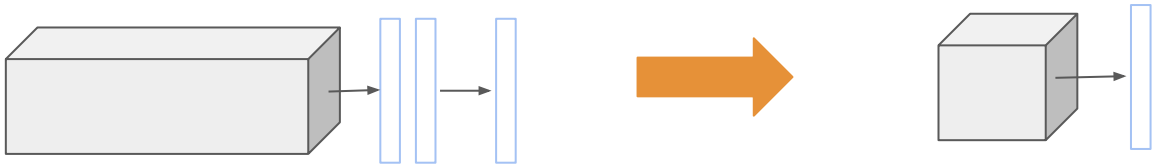
Maximizing domain confusion does not necessarily align the classes in the target with those in the source

Possible solution: Transfer the similarity structure amongst categories from the source to the target, using **distillation**

(Hinton, G., Vinyals, O., & Dean, J. "Distilling the Knowledge in a Neural Network". *NIPS 2014 DL Workshop*, 1–9)

Distillation

Original application was to transfer the knowledge from a large, easy to train model into a smaller/faster model more suitable for deployment



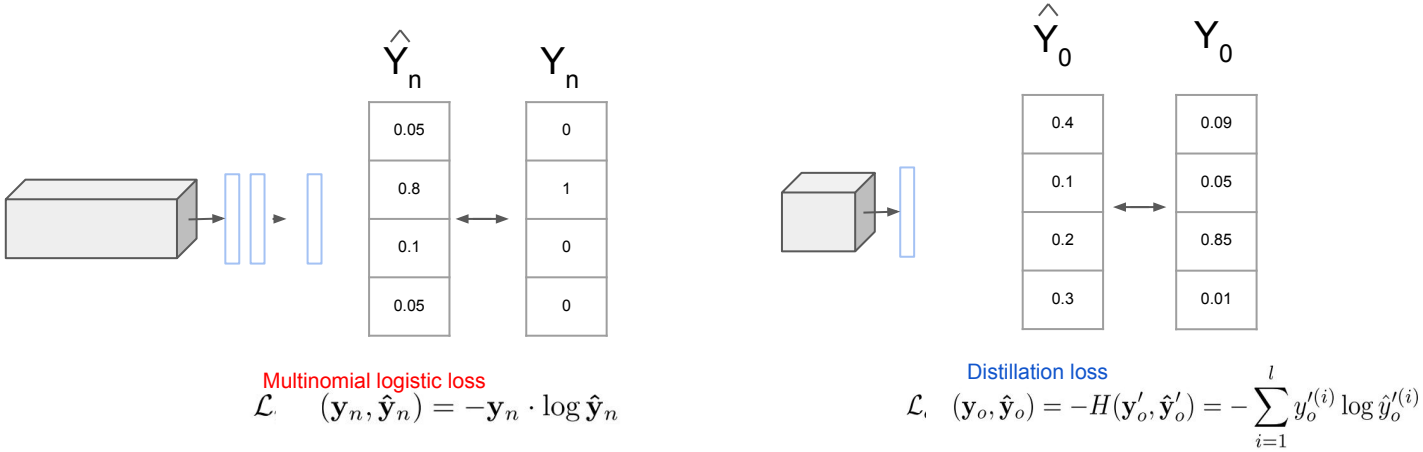
Bucilua[1] demonstrated that this can be done reliably when transferring from a large ensemble of models to a single small model

[1]C.Bucilua, R. Caruana, and A. Niculescu-Mizil. “Model compression”. In ACMSIG KDD ’06, 2006

Distillation

Idea: use the class probabilities produced by the large model as “soft targets” for training the small model

- The ratios of probabilities in the soft targets provide information about the learned function
- These ratios carry information about the structure of the data
- Train by replacing the hard labels with the **softmax activations from the original large model**



Hinton, G., Vinyals, O., & Dean, J. “Distilling the Knowledge in a Neural Network”. *NIPS 2014 DL Workshop*, 1–9.

Distillation

- To increase the influence of non-target class probabilities in the cross entropy, the temperature of the final softmax is raised to “soften” the final probability distribution over classes
- Transfer can be obtained by using the same large model training set or a separate training set
- If the ground-truth labels of the transfer set are known, standard loss and distillation loss can be combined

$$y_o'^{(i)} = \frac{(y_o^{(i)})^{1/T}}{\sum_j (y_o^{(j)})^{1/T}}, \quad \hat{y}_o'^{(i)} = \frac{(\hat{y}_o^{(i)})^{1/T}}{\sum_j (\hat{y}_o^{(j)})^{1/T}}.$$

0.09
0.05
0.85
0.01

T=1

0.15
0.10
0.70
0.05

T>1

Hinton, G., Vinyals, O., & Dean, J. “Distilling the Knowledge in a Neural Network”. *NIPS 2014 DL Workshop*, 1–9.

Semi-supervised task & domain adaptation

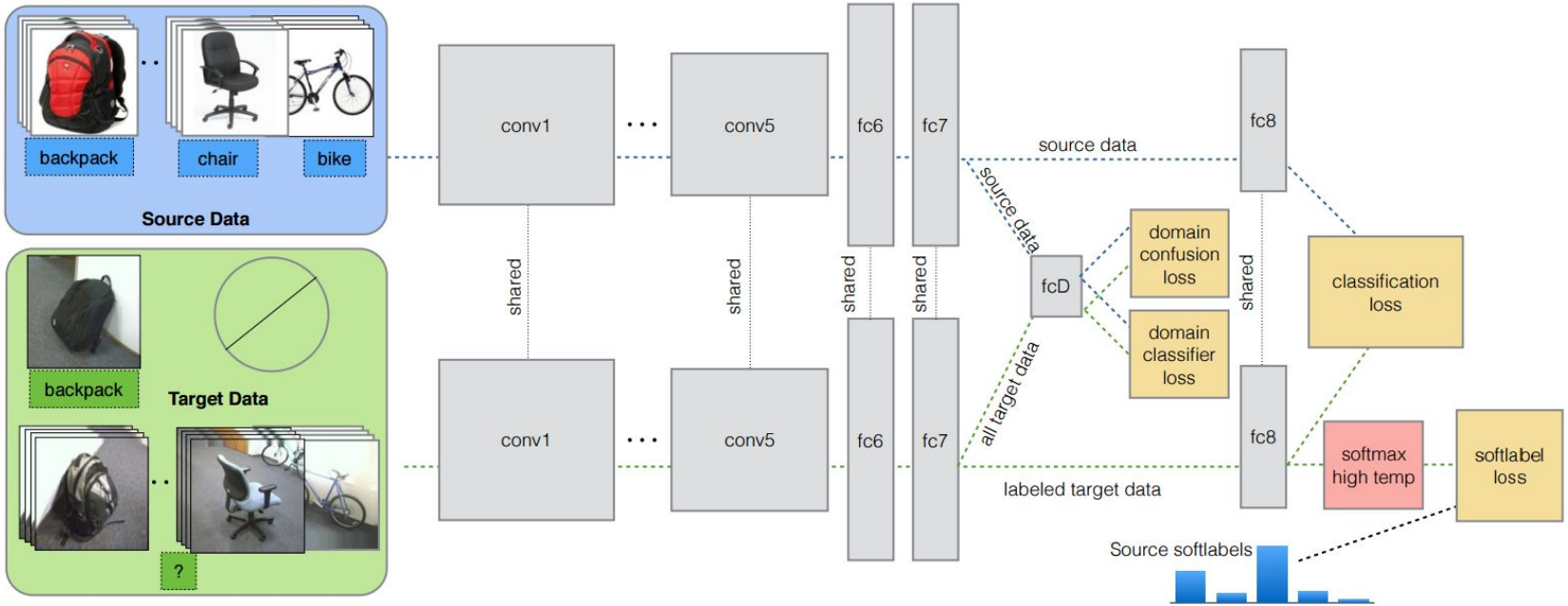
When some labels are available in the target domain, then we can use these when doing domain adaptation. I.e. combine *fine tuning* and *unsupervised domain adaptation*.

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

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

Domain confusion loss

Classifier loss

Soft label loss to align classifier scores across domains

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

Domain confusion loss

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

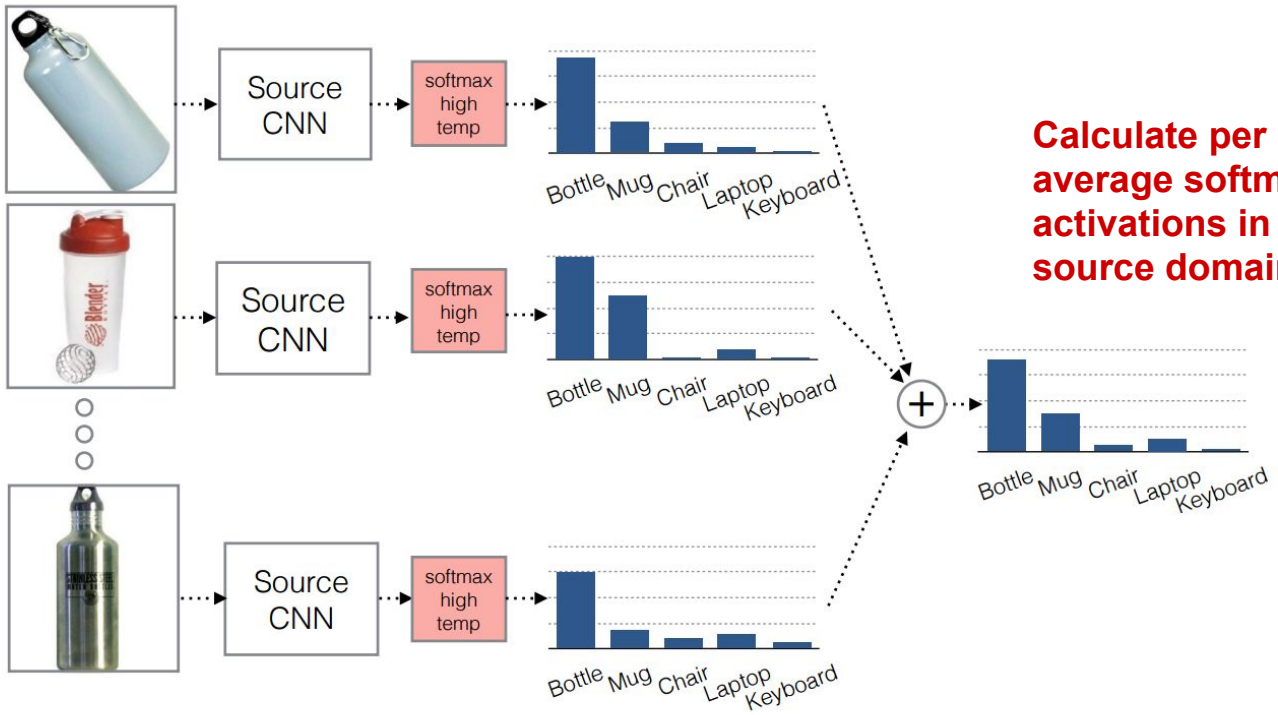
$$\mathcal{L}_D(x_S, x_T, \theta_{\text{repr}}; \theta_D) = -\sum_d \mathbb{1}[y_D = d] \log q_d$$

Second makes features as confusing as possible for the discriminator:

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

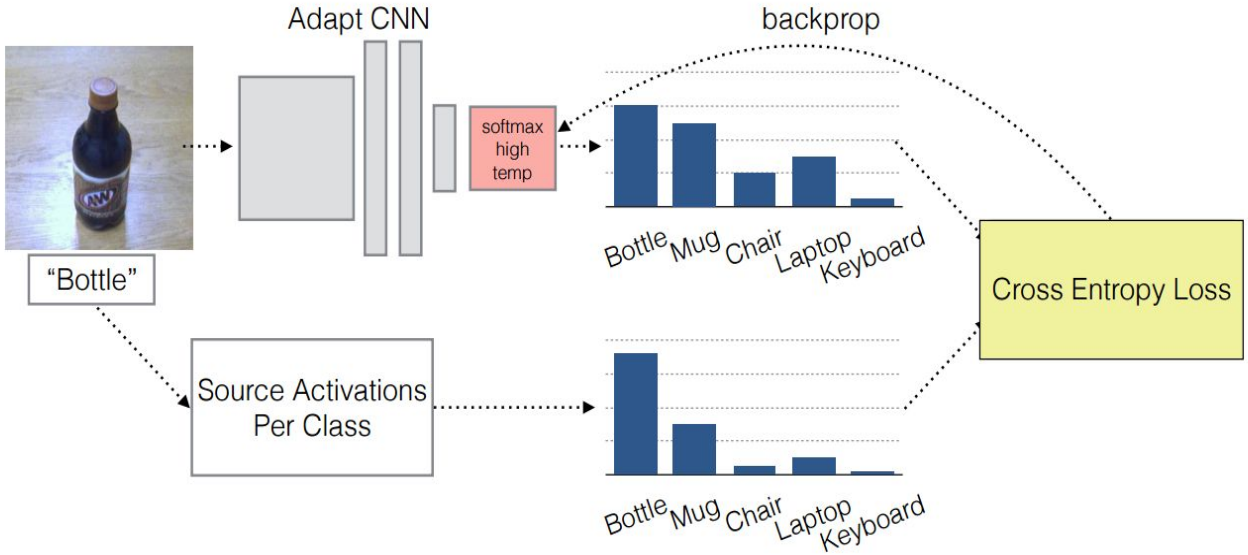
Tzeng, Eric, et al. Simultaneous deep transfer across domains and tasks. ICCV. 2015.

Alignment of source and target predictions



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

Alignment of source and target predictions



Use these as the target distribution for target domain.

Minimizing cross entropy loss same as minimizing KL divergence!

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

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

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).
- [Hinton2015] Hinton, G., Vinyals, O., & Dean, J. “Distilling the Knowledge in a Neural Network”. *NIPS 2014 DL Workshop*, 1–9.