

INTRODUCTION TO DEEP LEARNING

Winter School at UPC TelecomBCN Barcelona. 22-30 January 2018.



Instructors



Xavier
Giró-i-Nieto

Marta R.
Costa-jussà

Noé
Casas

Elisa
Sayrol

Antonio
Bonafonte

Verónica
Vilaplana

Ramon
Morros

Javier
Ruiz

Organizers



Supporters

aws educate

GitHub Education

+ info: <https://telecombcn-dl.github.io/2018-idl/>

[\[course site\]](#)



#DLUPC

Day 3 Lecture 2

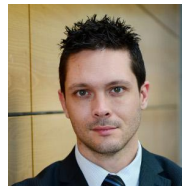
Transfer learning



Ramon Morros

ramon.morros@upc.edu

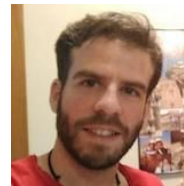
Associate Professor
Universitat Politècnica de Catalunya
Technical University of Catalonia



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



Transfer Learning

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

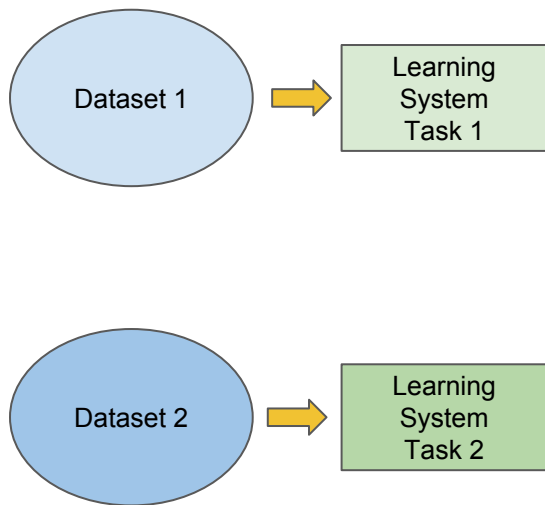
- Based on human learning. People can often transfer knowledge learnt previously to novel situations
 - Play classic piano → Play jazz piano
 - Maths → Machine Learning
 - Ride motorbike → Drive a car

Traditional ML

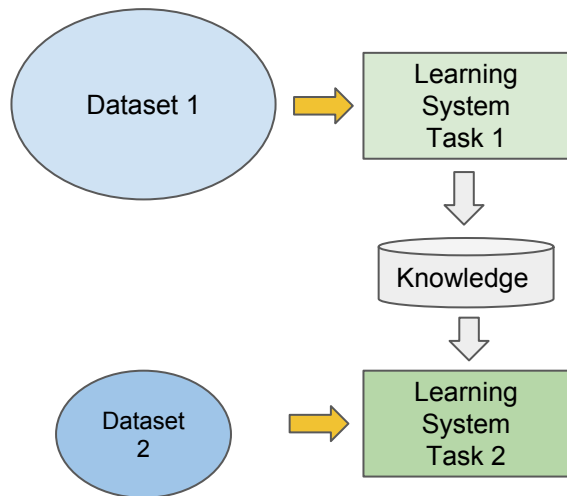
vs

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

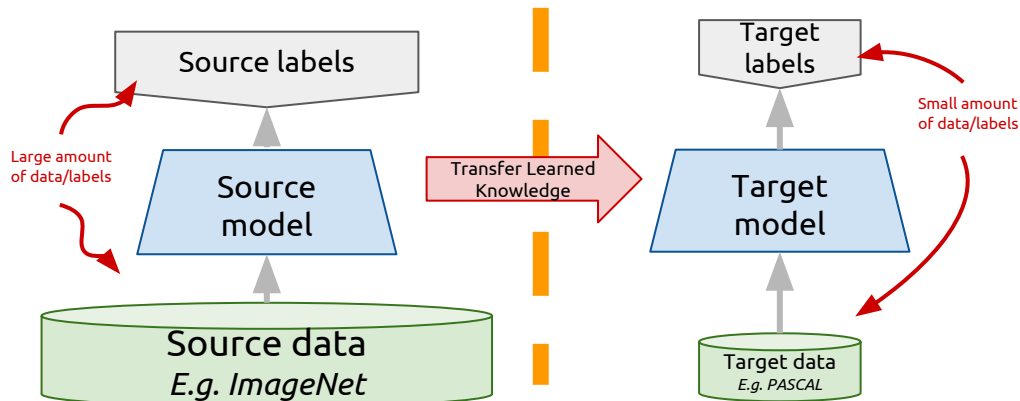
- You can learn useful representations from **unlabelled data**
- You can train on a nearby **surrogate objective** for which it is easy to generate labels
- You can **transfer** learned representations from a related task

Transfer learning: idea

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

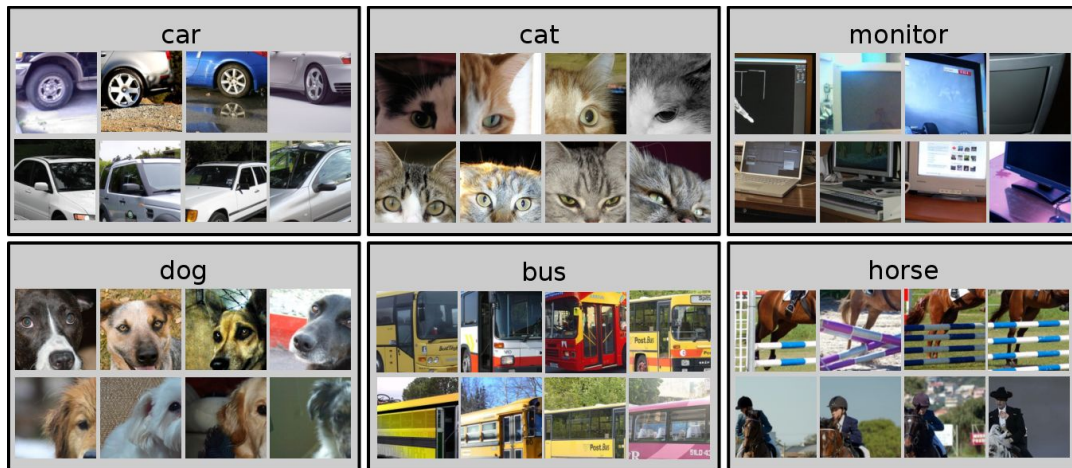
- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

This lecture will talk about how to do this.



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?



Notation (I)

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

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

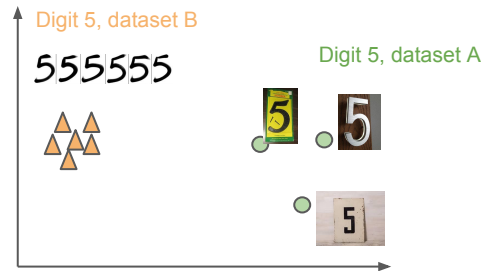
For a given domain **D**, a **Task** is defined by two components:

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

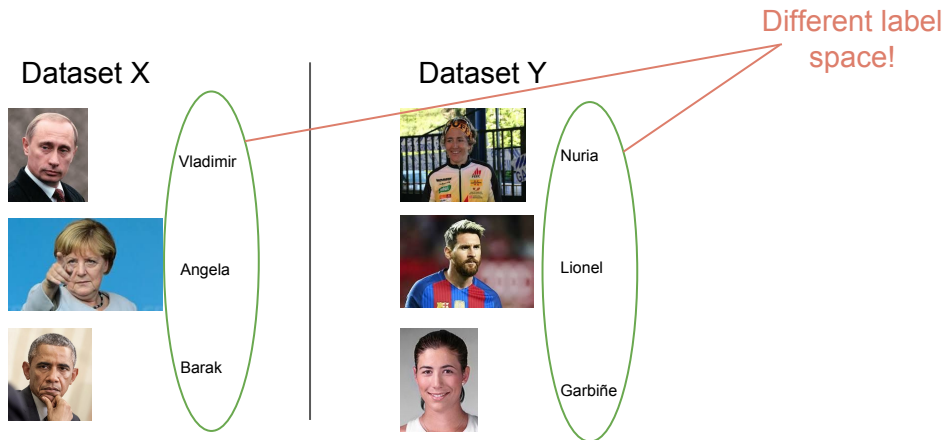
- A label space: \mathcal{Y}
- A predictive function η , learned from *feature vector/label* pairs, (x_i, y_i) , $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- For each feature vector in the domain, η predicts its corresponding label: $\eta(x_i) = y_i$

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

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

- Task on the source domain

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

- Target domain


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

- Task on the target domain

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

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:

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


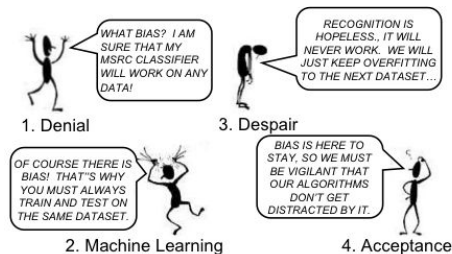
(Target space labeled data may not be present in unsupervised case)

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

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

Domain bias

- 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.

Domain adaptation

When there is a domain shift ...

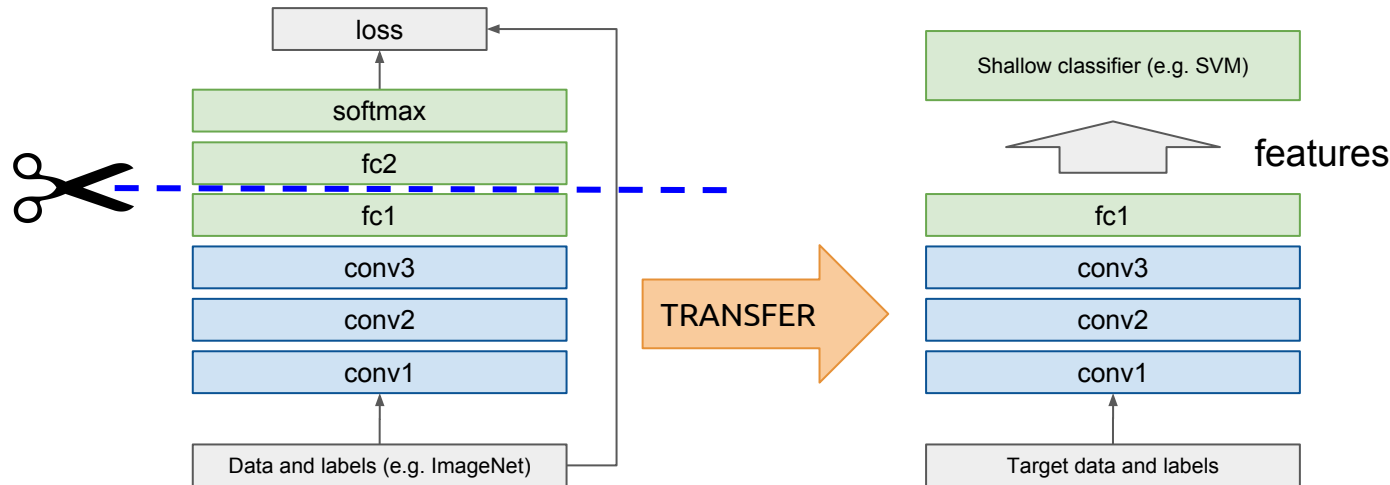
- The size of this shift is often measured by the distance between source and target subspaces
- A typical approach is to learn a feature space transformation to align the source and target representations (reduce domain divergence)



“Off-the-shelf”

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 $D_S = D_T$



Off-the-shelf features

Works surprisingly well in practice!

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

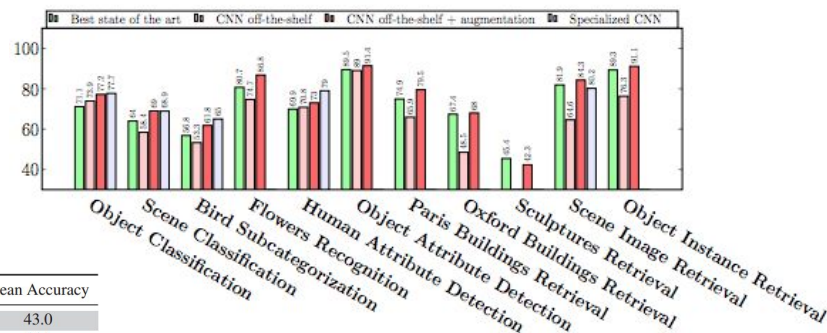
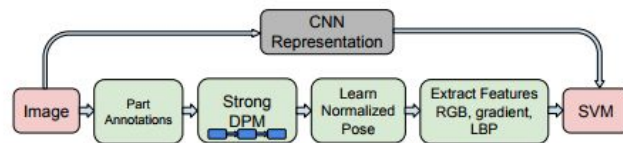
Image classification:

- 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+SIFT+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

Can we do better than off the shelf features?

Fine-tuning: supervised domain adaptation

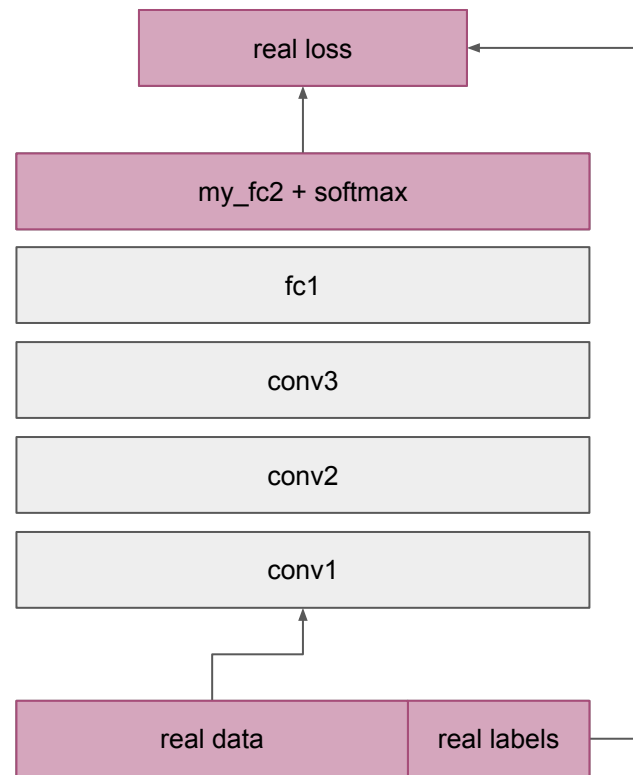
Train deep net on “nearby” task for which it is easy to get labels using standard backprop

- 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?

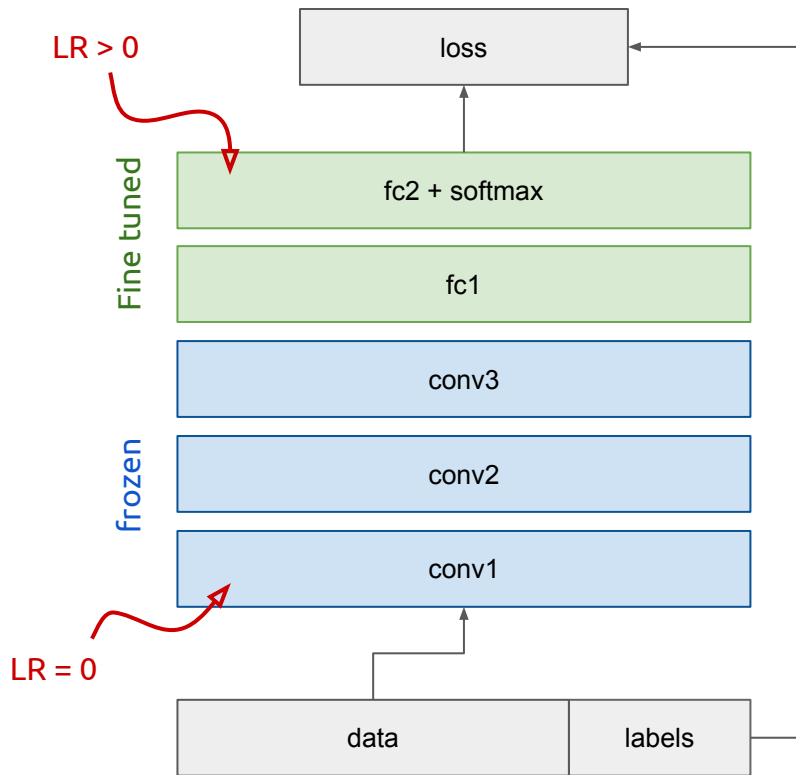
Bottom n layers can be frozen or fine tuned.

- **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

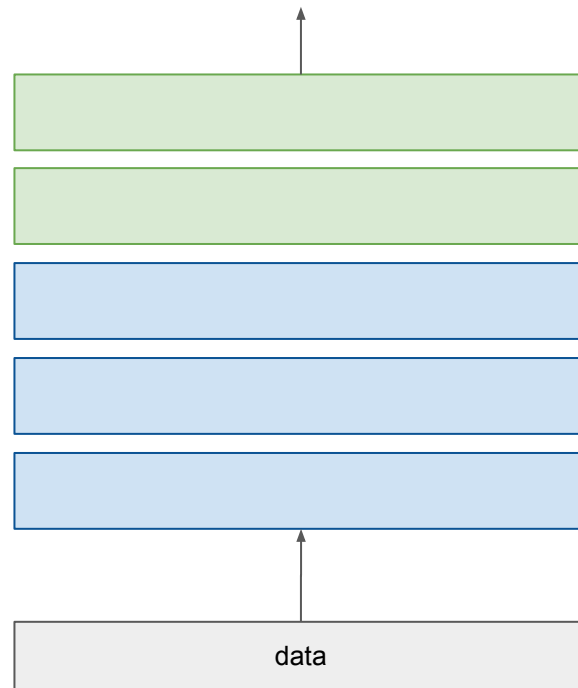
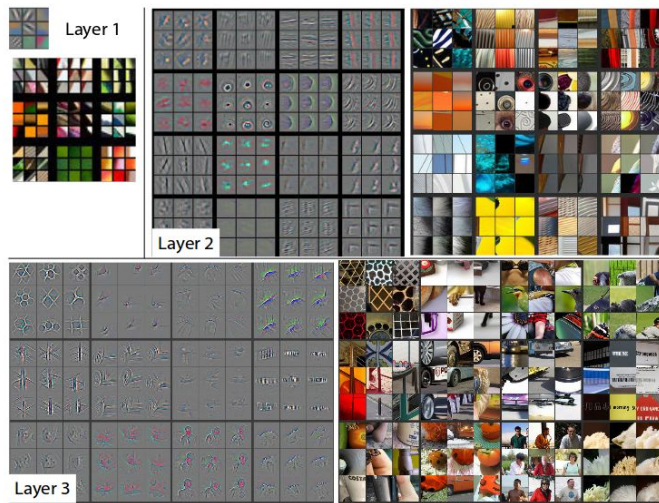
In general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine tuning



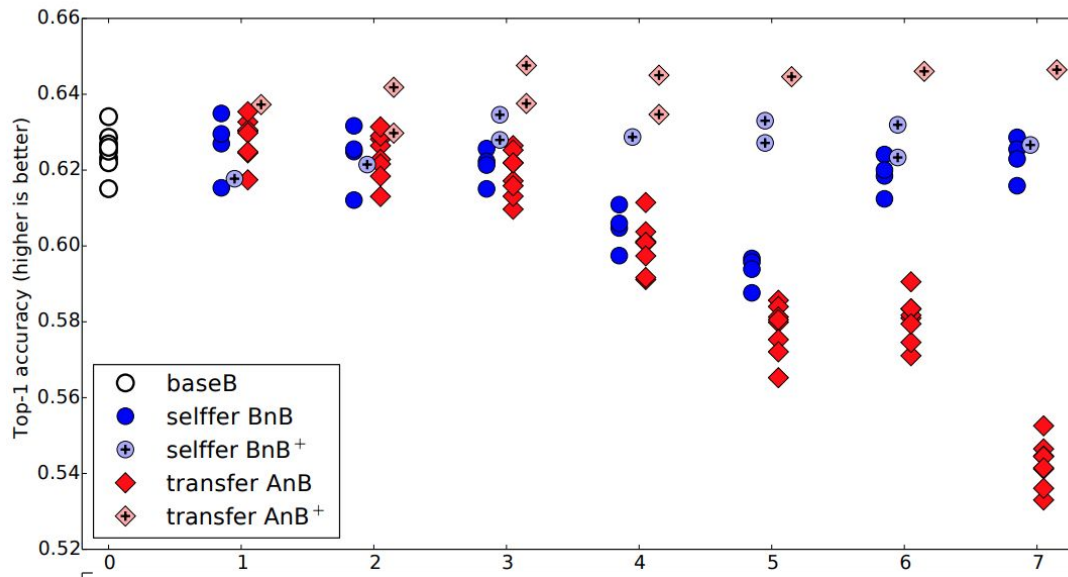
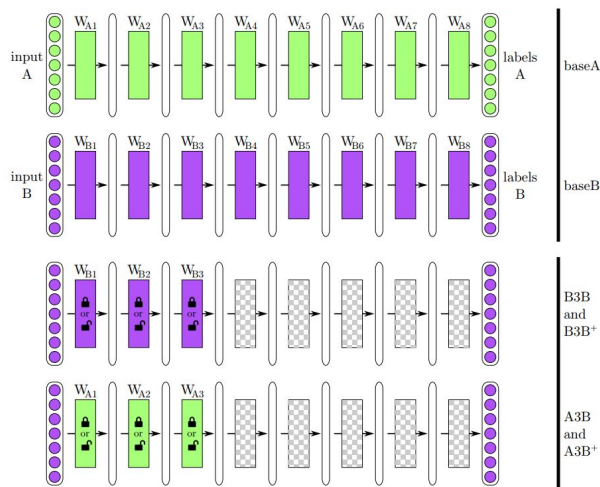
How transferable are features?

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

Higher layers: more task specific.



How transferable are features?



How transferable are features?

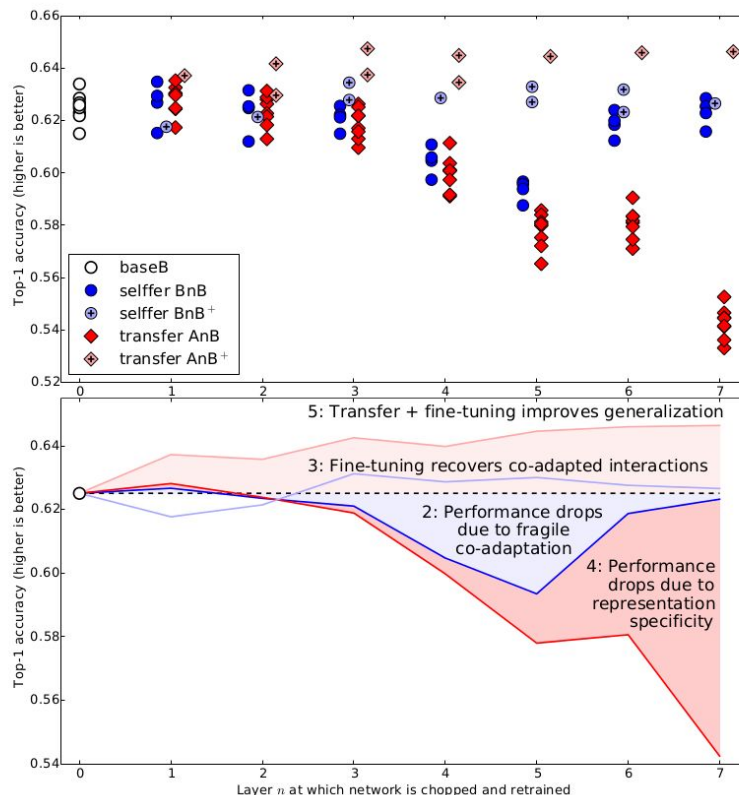
Transferability is negatively affected by two distinct issues:

- 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!



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



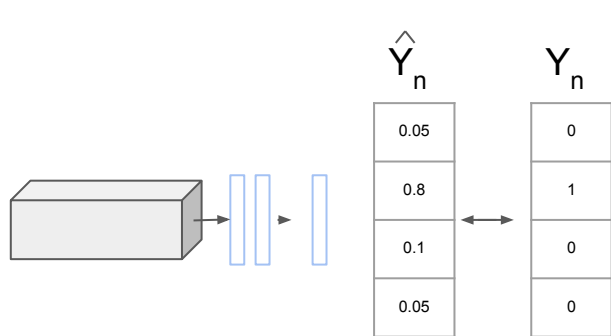
Bucilua¹ demonstrated that this can be done reliably when transferring from a large ensemble of models to a single small model

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

Distillation

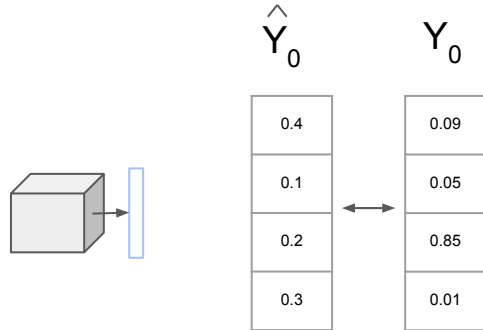
Idea: use the class probabilities produced by the large model as “soft targets” for training the small model

- 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**



Multinomial logistic loss

$$\mathcal{L}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n$$



Distillation loss

$$\mathcal{L}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) = -\sum_{i=1}^l y_o^{(i)} \log \hat{y}_o^{(i)}$$

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	0.15
0.05	0.10
0.85	0.70
0.01	0.05

T=1

T>1

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

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