




Transfer learning & domain adaptation

Hakan Bilen

Machine Learning Practical - MLP Lecture 13
30 January 2019

Human level machine performance

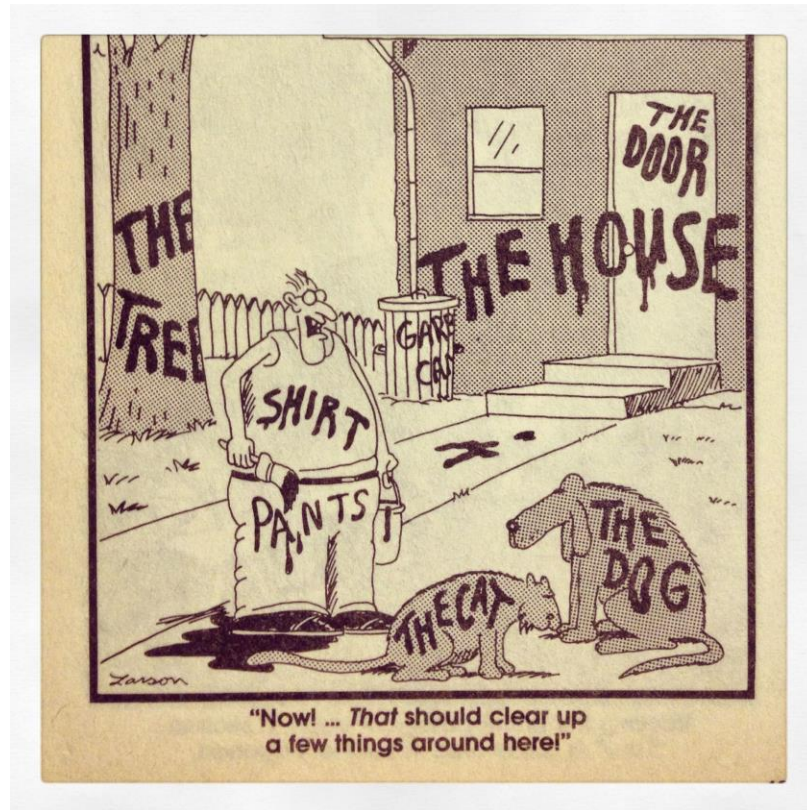
Task		Human	Machine	Measure
Object classification [He etal CVPR'16]		5.1%	3.75%	Top5 error ↓
Person identification [Taigman etal CVPR'14]		2.5%	2.6%	Top1 error ↓
Lip reading [Chung etal CVPR'17]		23.8%	54.9%	BLEU score ↑

Human level machine performance but machines

- Require **millions** of **labeled** images
 - Labeling is costly and time-consuming
 - We cannot label everything
- Do not generalize well to **new domains**



Image credit: Saenko et al.



Cartoon credit: Gary Larson

Human learning

- Humans have an ability to transfer knowledge across
 - Related tasks: E.g. knowing math and statistics helps to learn machine learning
 - Same task but different domain: E.g. knowing to drive on the left helps to learn driving on the right

This lecture focuses on

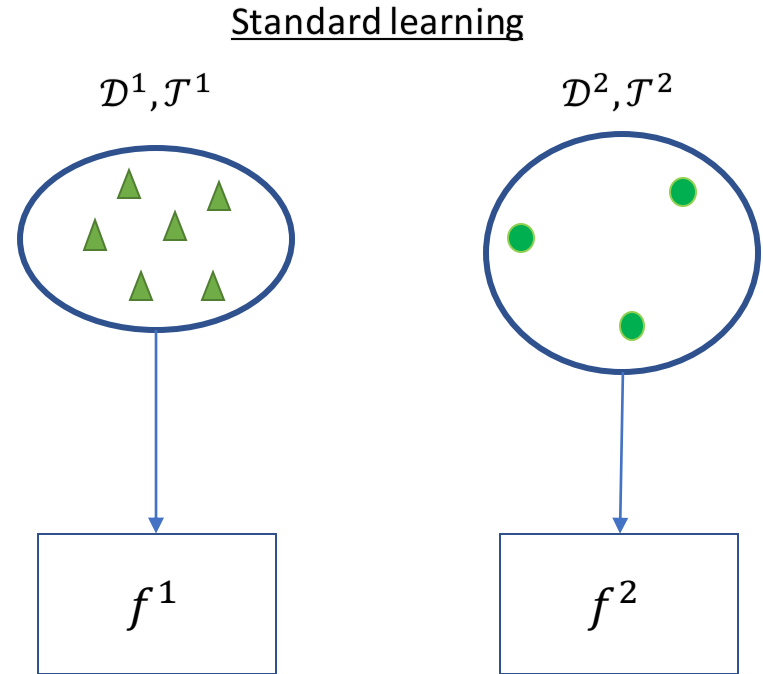
Leveraging previous knowledge from one task to solve related ones in machine learning

By applying **transfer learning** and **domain adaptation** techniques

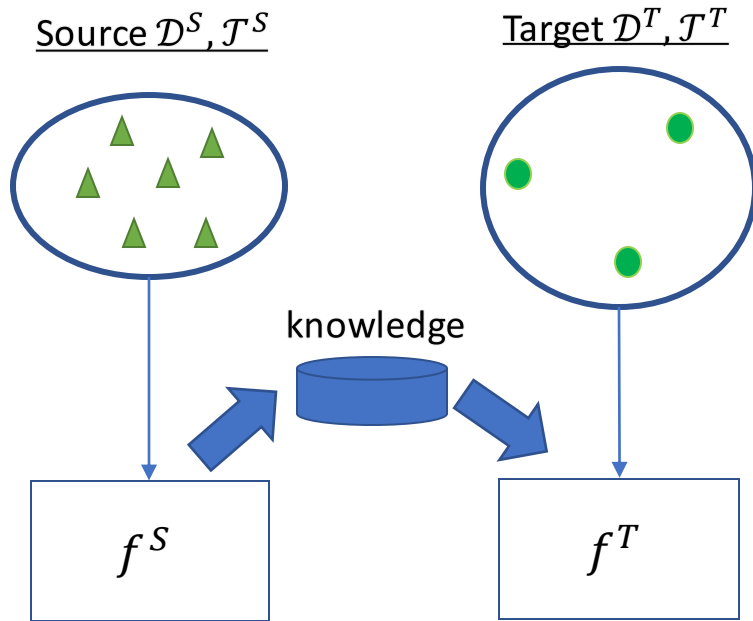
Related but **not** about unsupervised, multi-task, zero-shot learning methods

Notations and definitions

- Domain \mathcal{D} has two components
 - Feature space \mathcal{X} where $x_1, \dots, x_n \in \mathcal{X}$
 - Marginal probability distribution $P(X)$
- Task \mathcal{T} has two components
 - Label space \mathcal{Y} where $y_1, \dots, y_n \in \mathcal{Y}$
 - Predictive function $f(x): \mathcal{X} \rightarrow \mathcal{Y}$



Knowledge transfer



- Source and target domains/tasks
- Typically significantly more labeled training samples for source task, few or none for target
- Goal is to transfer knowledge from source to target task

Scenarios

1. If two domains are different $D^S \neq D^T$, they may have different

a) input spaces $\mathcal{X}^S \neq \mathcal{X}^T$

b) marginal probability $P(X^S) \neq P(X^T)$

2. If two tasks are different $T^S \neq T^T$, they may have different

a) label spaces $\mathcal{Y}^S \neq \mathcal{Y}^T$

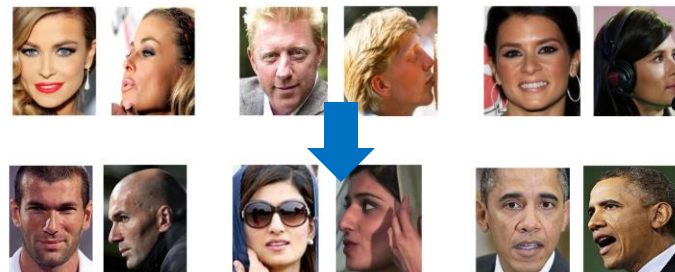
b) conditional probability $P(Y^S|X^S) \neq P(Y^T|X^T)$



1. a)



1. b)



2. a)

Feature transfer

$Y^S \neq Y^T, N^T$ is small

Recipe

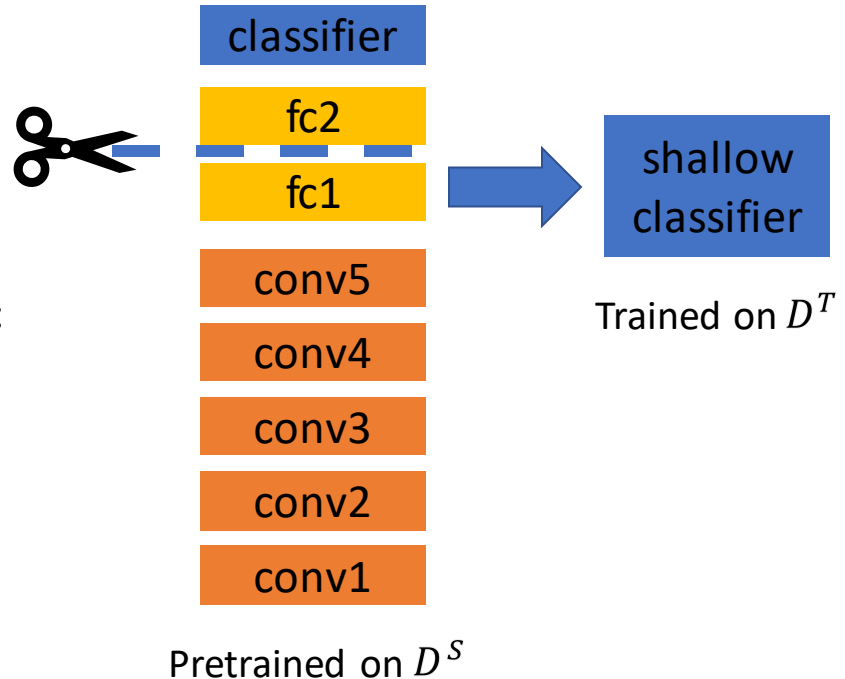
- Take a pretrained network on D^S
- Feed training data from D^T to extract features
- Train a shallow model on these features

Assumption

- Pretrained features from D^S is generic and useful for D^T too

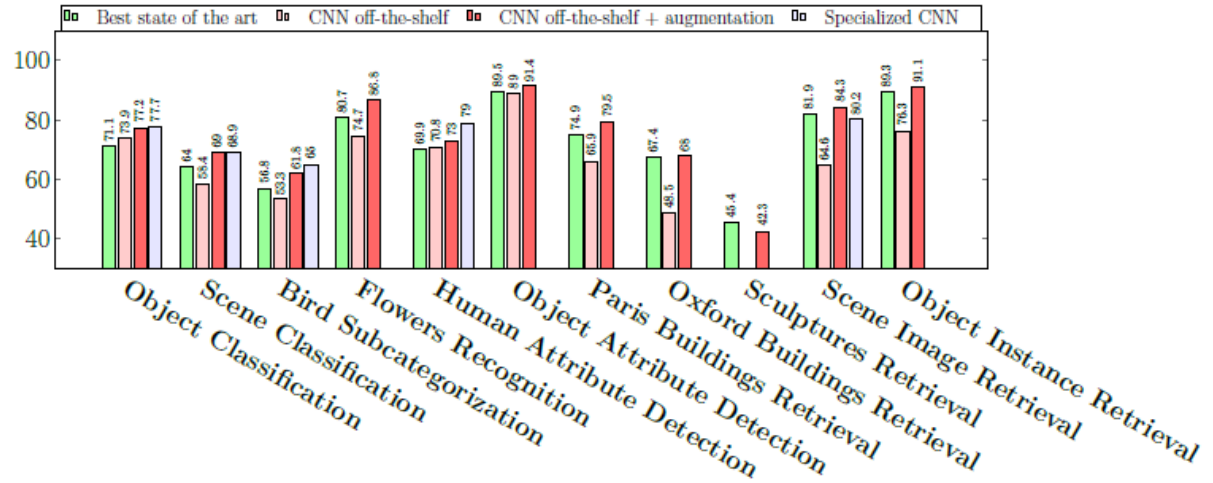
Design choices

- Which source task?
- Which network architecture?
- Which layer?



Feature transfer

- The authors extract features from ImageNet pretrained OverFeat network and train an SVM
- Works surprisingly better than handcrafted methods (“Best state-of-the-art”)
- In contrast to crafted features, deep features transfers the knowledge from source task
- Language modeling examples: word2vec and GloVe



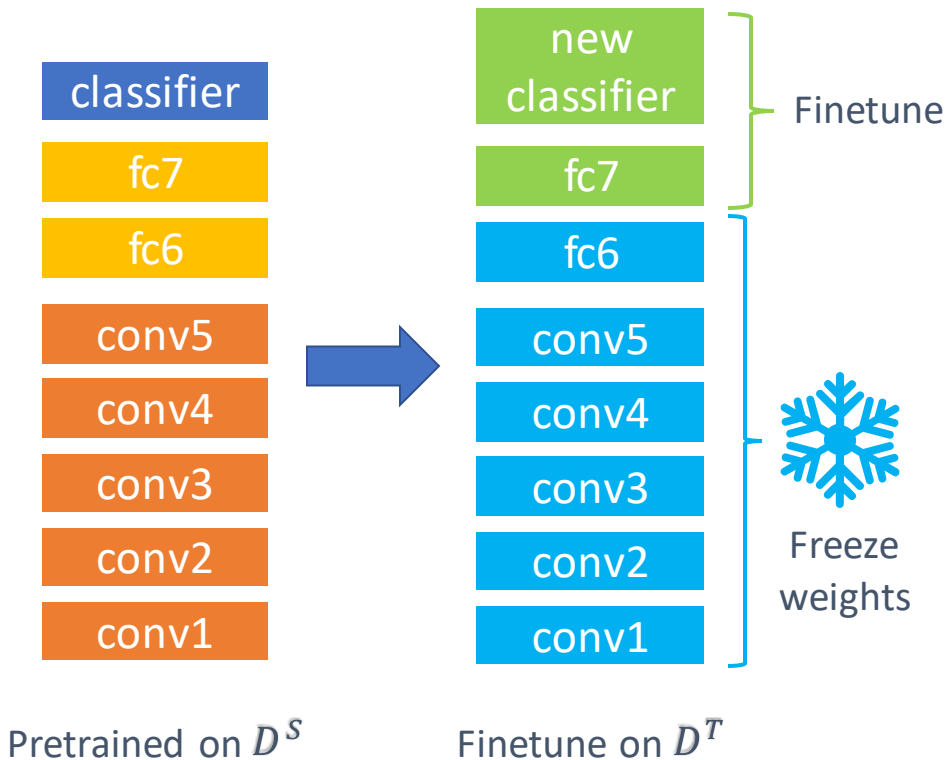
Finetuning

Idea

- Train a network trained on source task
- Clone the source network for target task
- Cut off top layer(s) and replace classifier
- Freeze bottom n layers
- Finetune remaining layers (usually with a low learning rate)

Challenge

- Which layers to freeze or to finetune?
- How to prevent overfitting?



How transferable are features in deep nets?

baseA: Train all layers from scratch on dataset A

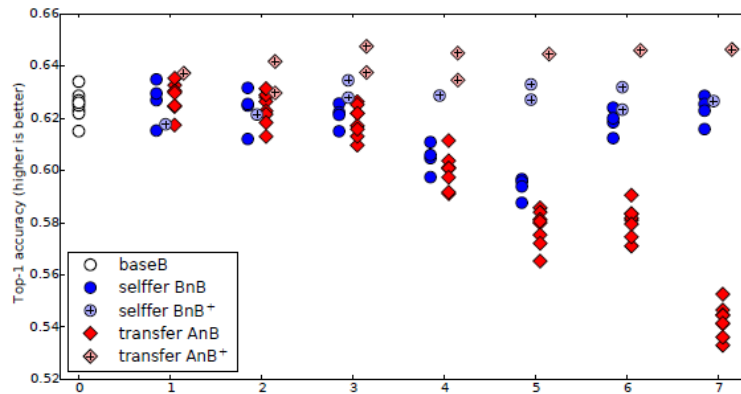
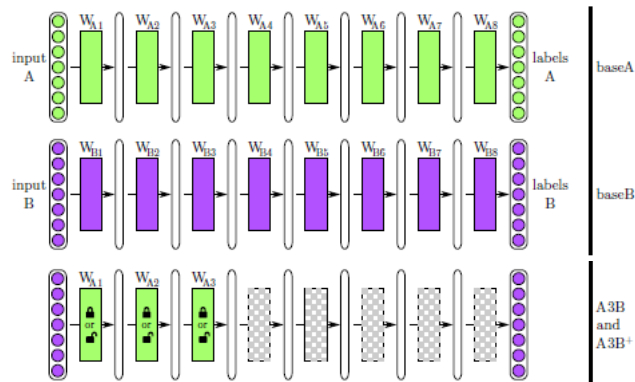
baseB: Train all layers from scratch on dataset B

Scenario I (transfer AnB)

- Train a network on dataset A
- Freeze first n layers, randomly initialize the rest
- Train on dataset B

Scenario II (transfer AnB+):

- Train a network on dataset A
- Do not freeze first n layers, randomly initialize the rest
- Train on dataset B



Regularization & finetuning

Problem: Finetuning a pretrained network on a small (target) dataset can also overfit to it!

Existing solutions: Dropout and L^2 regularization (or weight decay, not always the same!)

- Dropout prevents co-adapting hidden units
- L^2 regularization penalizes complex models (distance between model and origin $||\mathbf{w} - \mathbf{0}||^2$)

Observation: There is no mechanism in fine-tuning for retaining the features learned on the source task

Solution: Penalize the distance to the pretrained model or the starting point (SP) $||\mathbf{w} - \mathbf{w}_0||^2$

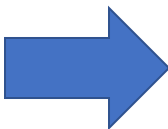
	MIT Indoors 67	Stanford Dogs 120	Caltech 256 – 30	Caltech 256 – 60
L^2	79.6±0.5	81.4±0.2	81.5±0.2	85.3±0.2
L^2 -SP	84.2±0.3	85.1±0.2	83.5±0.1	86.4±0.2

Domain adaptation

amazon.com



Source domain
Lots of labeled data



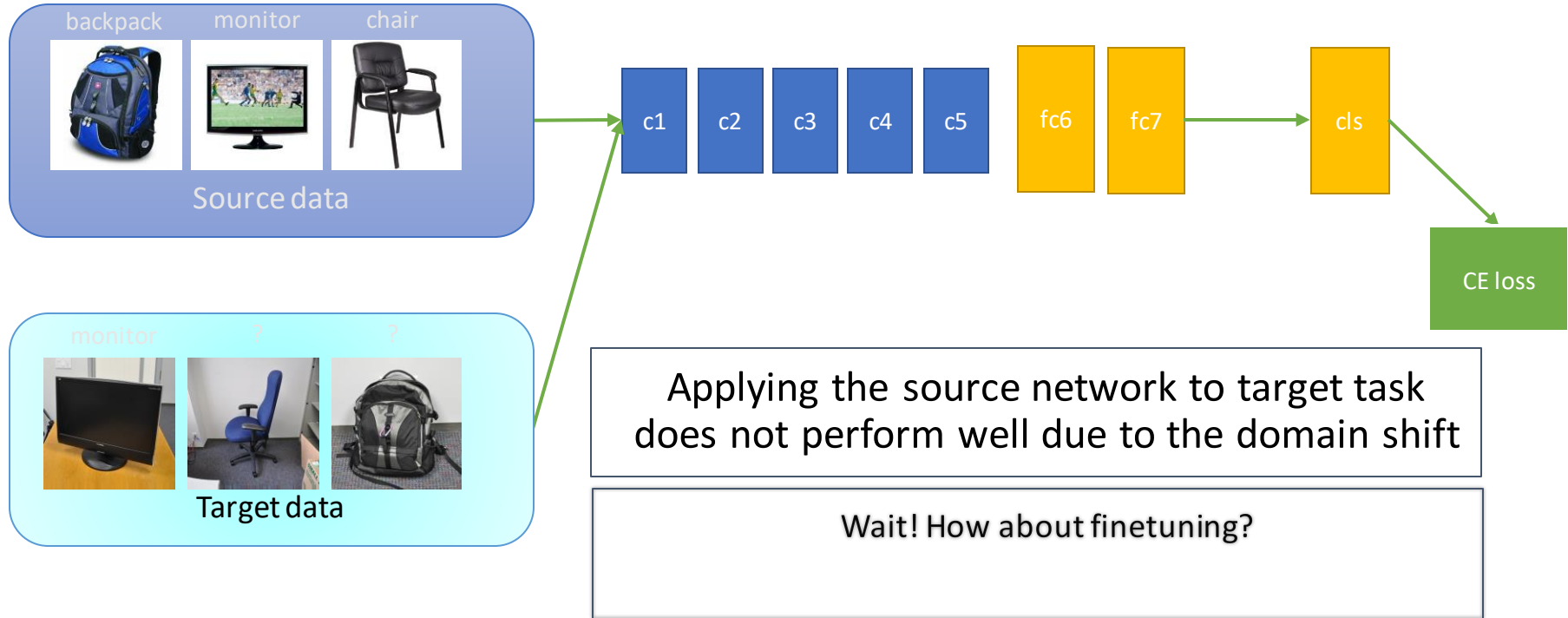
The Office



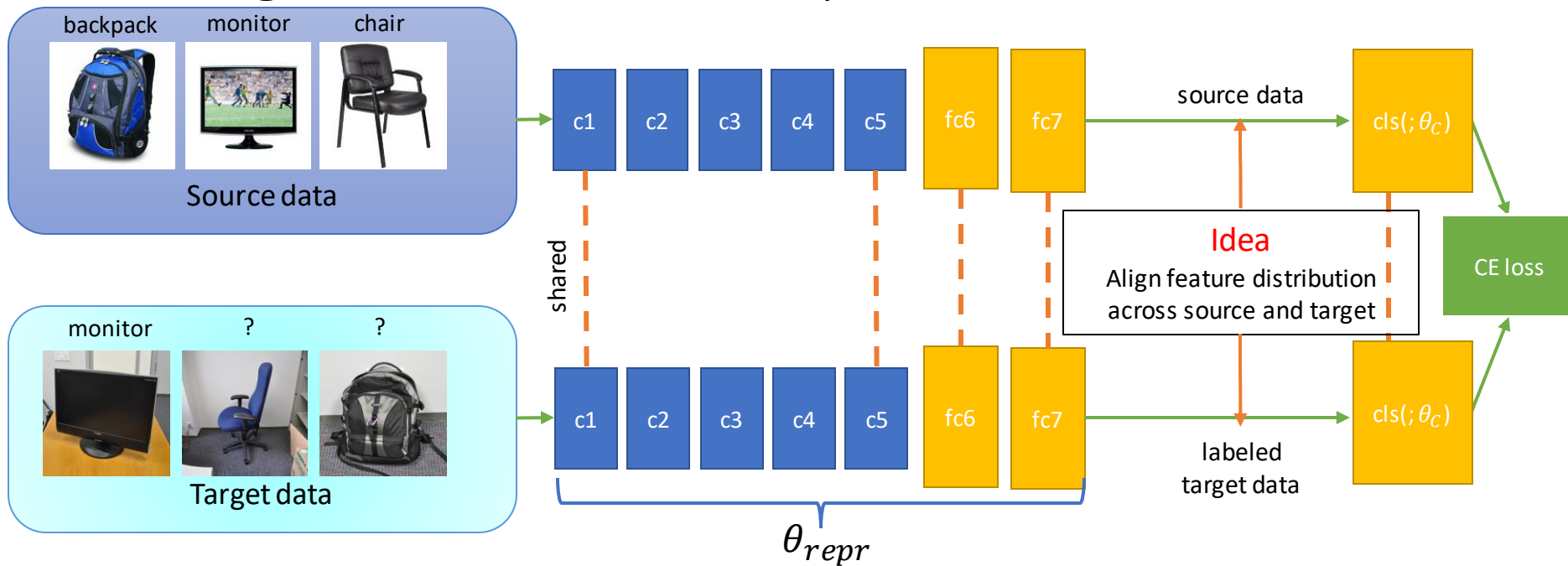
Target domain
Unlabeled or limited labels

$$\mathcal{X}^S = \mathcal{X}^T \text{ and } \mathcal{Y}^S = \mathcal{Y}^T \text{ but } P(\mathcal{X}^S) \neq P(\mathcal{X}^T) \text{ and/or } P(\mathcal{Y}^S|\mathcal{X}^S) \neq P(\mathcal{Y}^T|\mathcal{X}^T)$$

Learning domain invariant representations

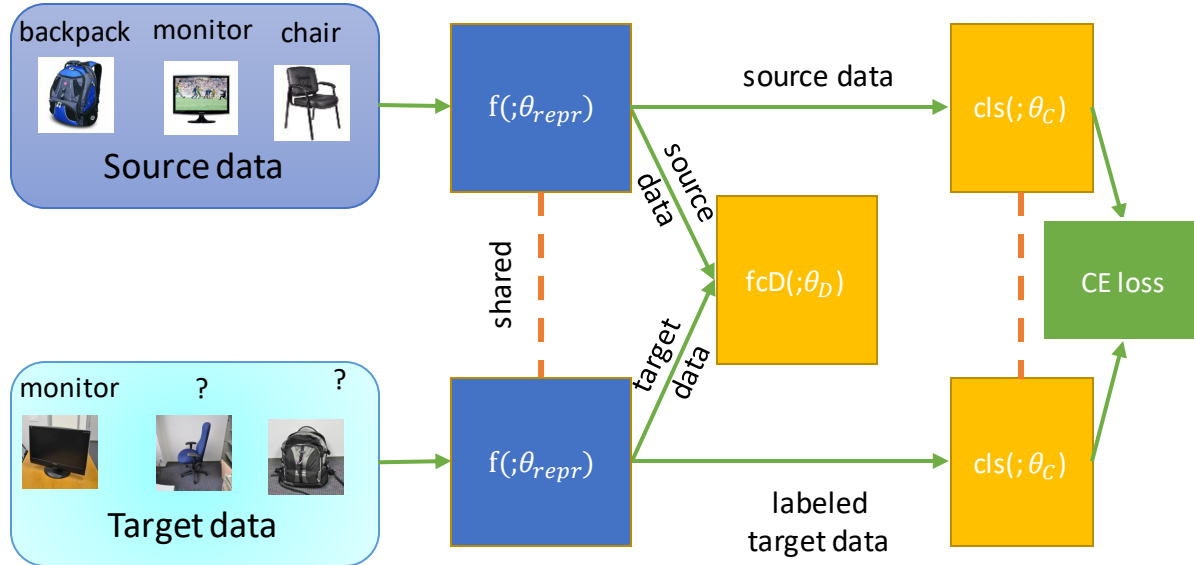


Learning domain invariant representations



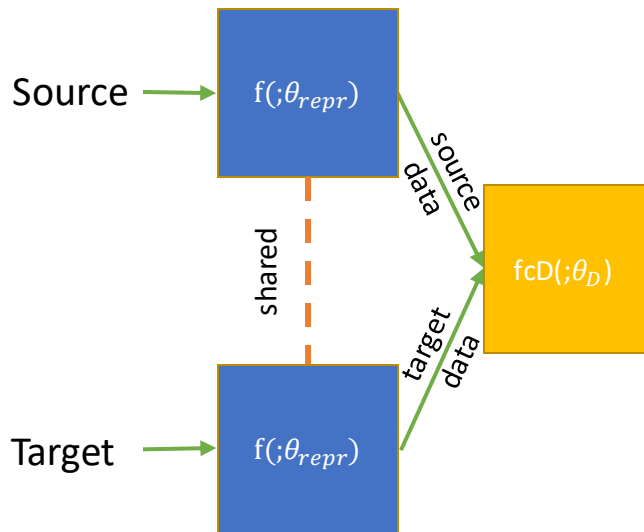
$$\text{Output: } p = \text{softmax}(\theta_C^T f(x; \theta_{repr}))$$
$$\text{Cross entropy loss: } \min_{\theta_{repr}, \theta_c} L_D(x, y; \theta_{repr}, \theta_c) = - \sum_k 1[y = k] \log p_k$$

Learning domain invariant representations



- Goal: Learn a domain invariant representation
- Idea: Add a domain classifier $fcD(\cdot; \theta_D)$ and learn to fool it
- Similar to GANs
 - Generator $f(\cdot; \theta_{repr})$
 - Discriminator $f(\cdot; \theta_D)$

Aligning domain distributions



- Output of domain classifier $fcD(\cdot; \theta_D)$ is $q = \text{softmax}(\theta_D^T f(x; \theta_{repr}))$
- Domain classifier can only control θ_D and aims to minimize

$$\begin{aligned} \min_{\theta_D} L_D(x_S, x_T, \theta_{repr}; \theta_D) \\ = - \sum_d 1[y_D = d] \log q_d \end{aligned}$$

- Network $f(x; \theta_{repr})$ can only control θ_{repr} and aims to fool domain classifier

$$\min_{\theta_{repr}} L_{Conf}(x_S, x_T, \theta_D; \theta_{repr}) = -\frac{1}{D} \sum_d \log q_d$$

Aligning domain distributions

We have two contradicting objectives as in GANs

$$\min_{\theta_D} L_D(x_S, x_T, \theta_{repr}; \theta_D) = - \sum_d 1[y_D = d] \log q_d$$
$$\min_{\theta_{repr}} L_{Conf}(x_S, x_T, \theta_D; \theta_{repr}) = - \frac{1}{D} \sum_d \log q_d$$

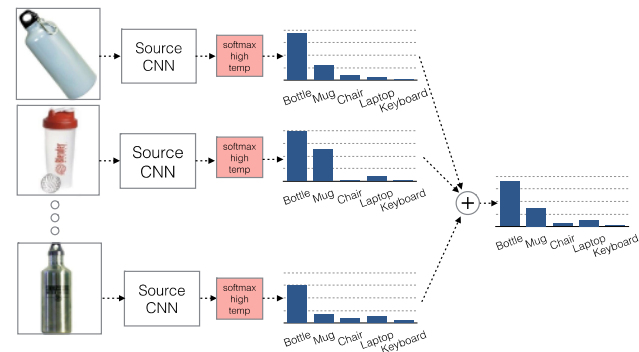
Optimize them iteratively in alternating steps

☺ This guarantees that feature distributions are aligned

☹ It does not ensure that same features represent the same categories

Aligning source and target classes

- Goal: Encourage class probabilities to be aligned across domains
- Observation: A bottle is more similar to a mug than a chair
- Idea: Calculate average “soft-labels” for source images and enforce the same relation for target images



$$p = \text{softmax}(\theta_C^T f(x; \theta_{repr}))$$

- $l^{bottle} = \frac{1}{N_{bottle}} \sum_i p_i$
- $L_{Soft}(x_T, y_T, \theta_{repr}; \theta_C) = - \sum_i l_i^{y^T} \log p_i$

Experimental setup

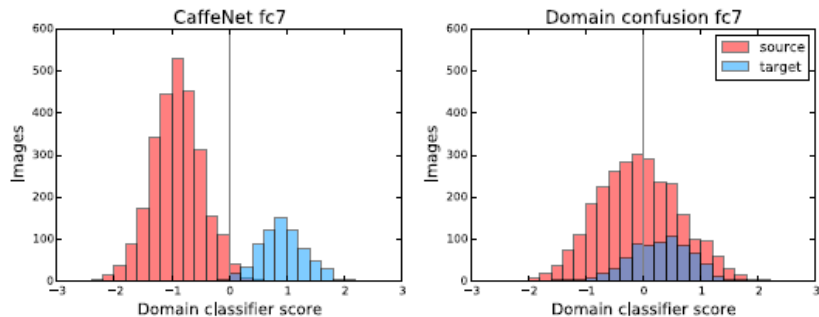


- 3 domains (D, A, W), each contains 31 categories
- 6 transfer scenarios (A→D, A→W, W→A, W→D, D→A, D→W)
- # training samples per category for source domain (20 for A, 8 for D and W)
- # training samples per category for target domain (3 for A, D and W)
- Only 15 out of 31 categories for target domain have labels

Results

	$A \rightarrow W$	$A \rightarrow D$	$W \rightarrow A$	$W \rightarrow D$	$D \rightarrow A$	$D \rightarrow W$	Average
MMDT [18]	–	44.6 ± 0.3	–	58.3 ± 0.5	–	–	–
Source CNN	54.2 ± 0.6	63.2 ± 0.4	34.7 ± 0.1	94.5 ± 0.2	36.4 ± 0.1	89.3 ± 0.5	62.0
Ours: dom confusion only	55.2 ± 0.6	63.7 ± 0.9	41.1 ± 0.0	96.5 ± 0.1	41.2 ± 0.1	91.3 ± 0.4	64.8
Ours: soft labels only	56.8 ± 0.4	65.2 ± 0.9	38.8 ± 0.4	96.5 ± 0.2	41.7 ± 0.3	89.6 ± 0.1	64.8
Ours: dom confusion+soft labels	59.3 ± 0.6	68.0 ± 0.5	40.5 ± 0.2	97.5 ± 0.1	43.1 ± 0.2	90.0 ± 0.2	66.4

Adding domain confusion and soft labels improve the performance



Domain labels get difficult to predict

Summary

- Networks as feature extractor
- Finetuning pretrained networks
- Domain adaptation

Recommended

- [Tzeng et al \(2015\), Simultaneous Deep Transfer Across Domains and Tasks, ICCV](#)

Additional

- [Yosinski et al \(2014\), How transferable are features in deep neural networks, NIPS](#)
- [Li et al \(2018\), Explicit Inductive Bias for Transfer Learning with Convolutional Networks, ICML](#)