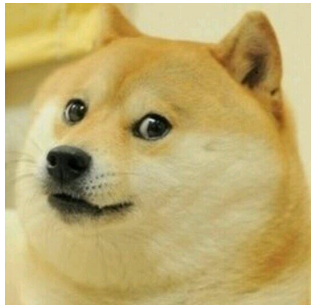
Simultaneous Deep Transfer Across Domains and Tasks

E Tzeng, J Hoffmann, T Darrell, K Saenko

Presentation by E J Crowley

Domain Adaptation for dummies

- Also called transfer learning
- Given a domain **X** with images and labels, can we learn a model that correctly predicts labels on images in domain **Y**?

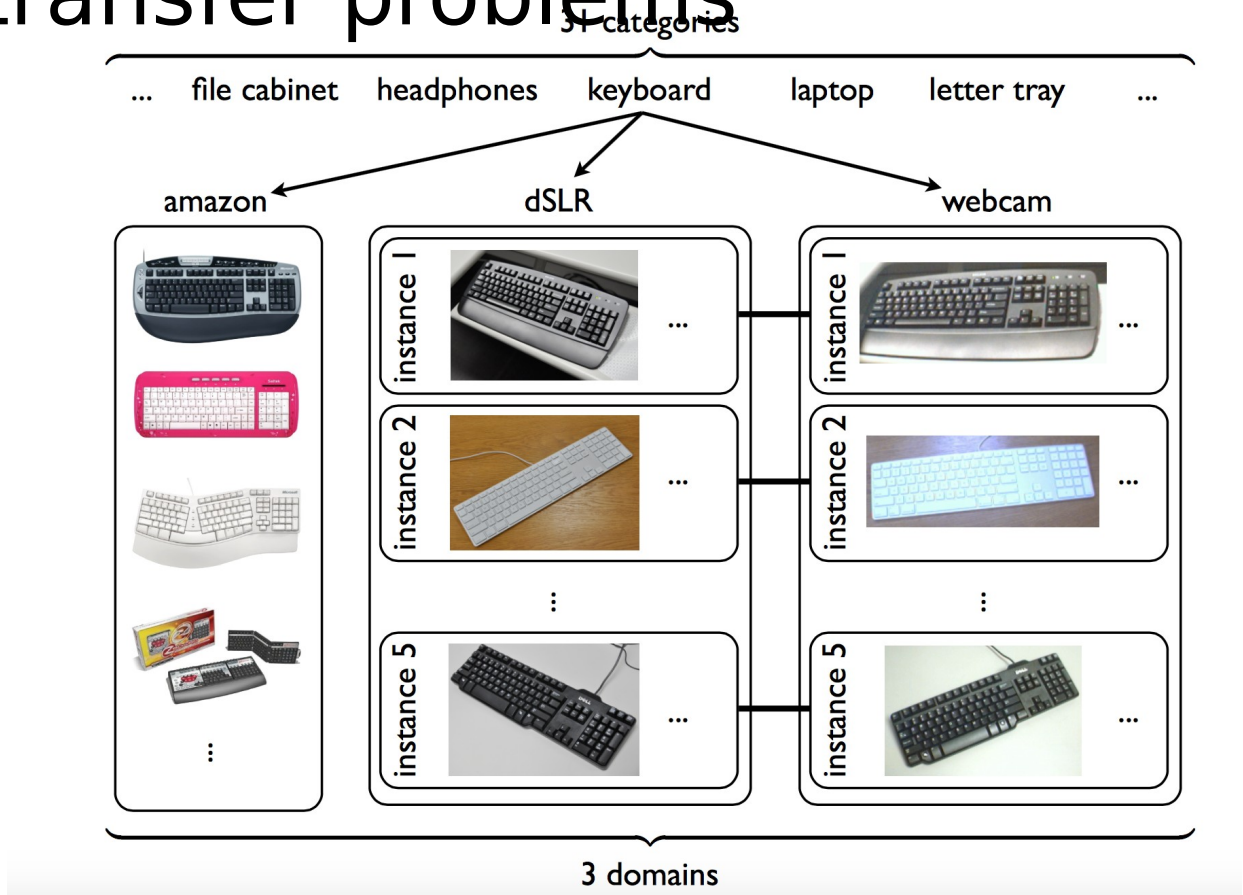







The Setup in this paper

- Office Dataset: 3 domains allowing for 6 transfer problems



Setup

For a given transfer problem (A-> B) we have training data for A, fully annotated with for categories

This can be supplemented with:

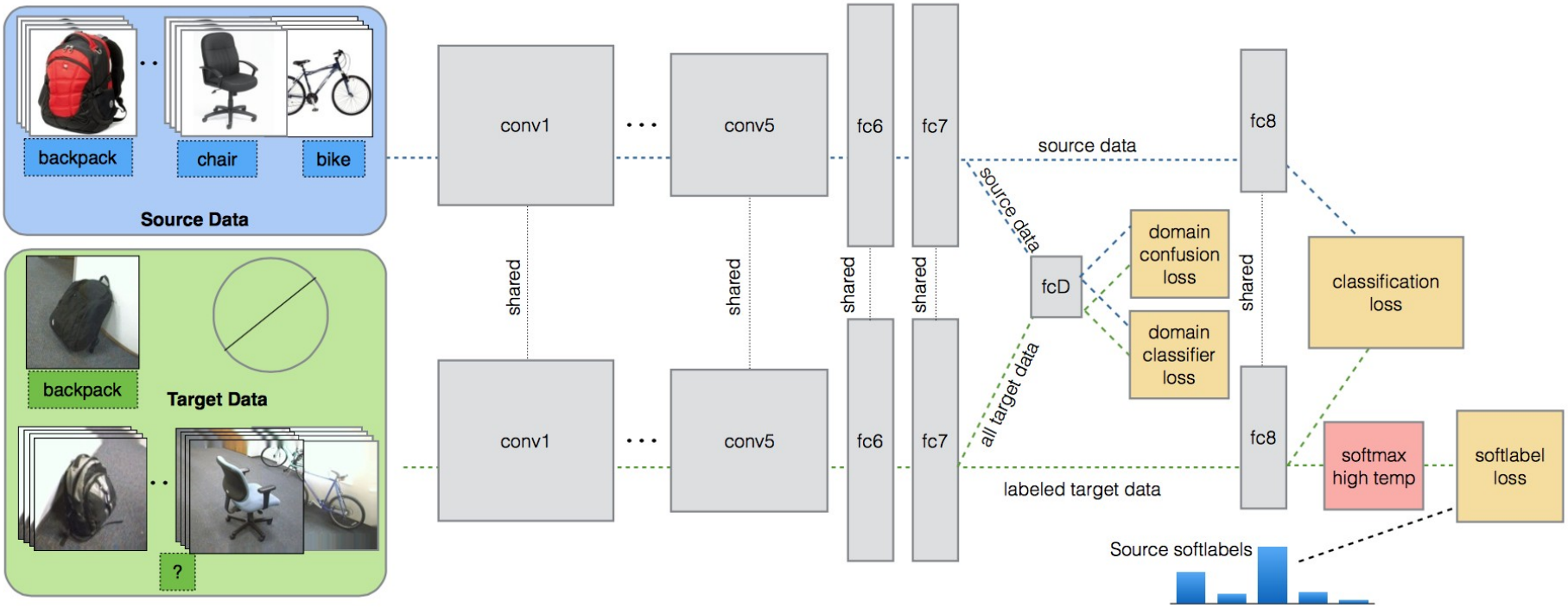
1) A small amount of labeled training data from B for every category (**supervised adaptation**)

or

2) A small amount of labeled training data from B for a subset of the categories (**semi-supervised adaptation**)

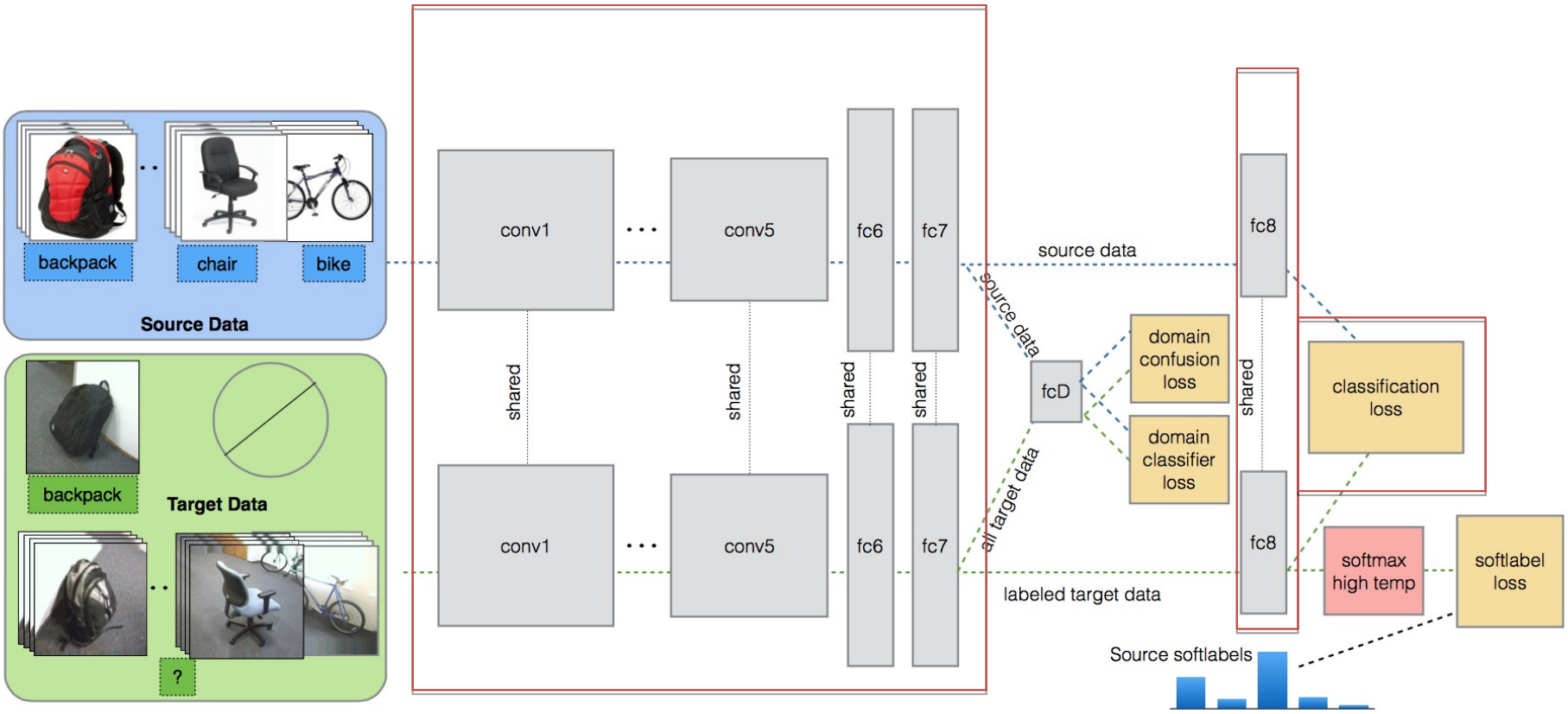
Goal: Maximize classification accuracy on test data (from B)

The Network



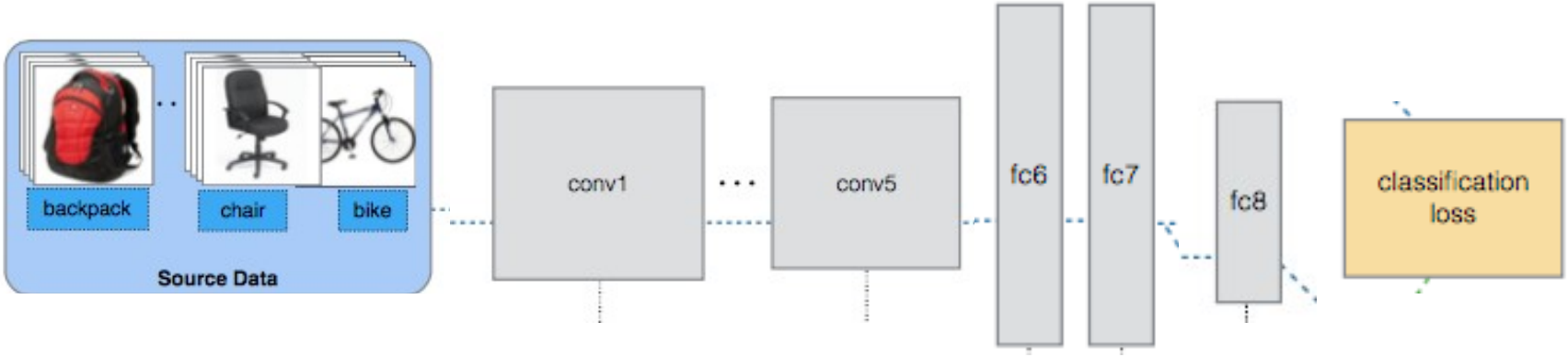
$$\mathcal{L}(x_S, y_S, x_T, y_T, \theta_D; \theta_{\text{repr}}, \theta_C) =$$
$$\mathcal{L}_C(x_S, y_S, x_T, y_T; \theta_{\text{repr}}, \theta_C)$$
$$+ \lambda \mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}})$$
$$+ \nu \mathcal{L}_{\text{soft}}(x_T, y_T; \theta_{\text{repr}}, \theta_C).$$

The Network

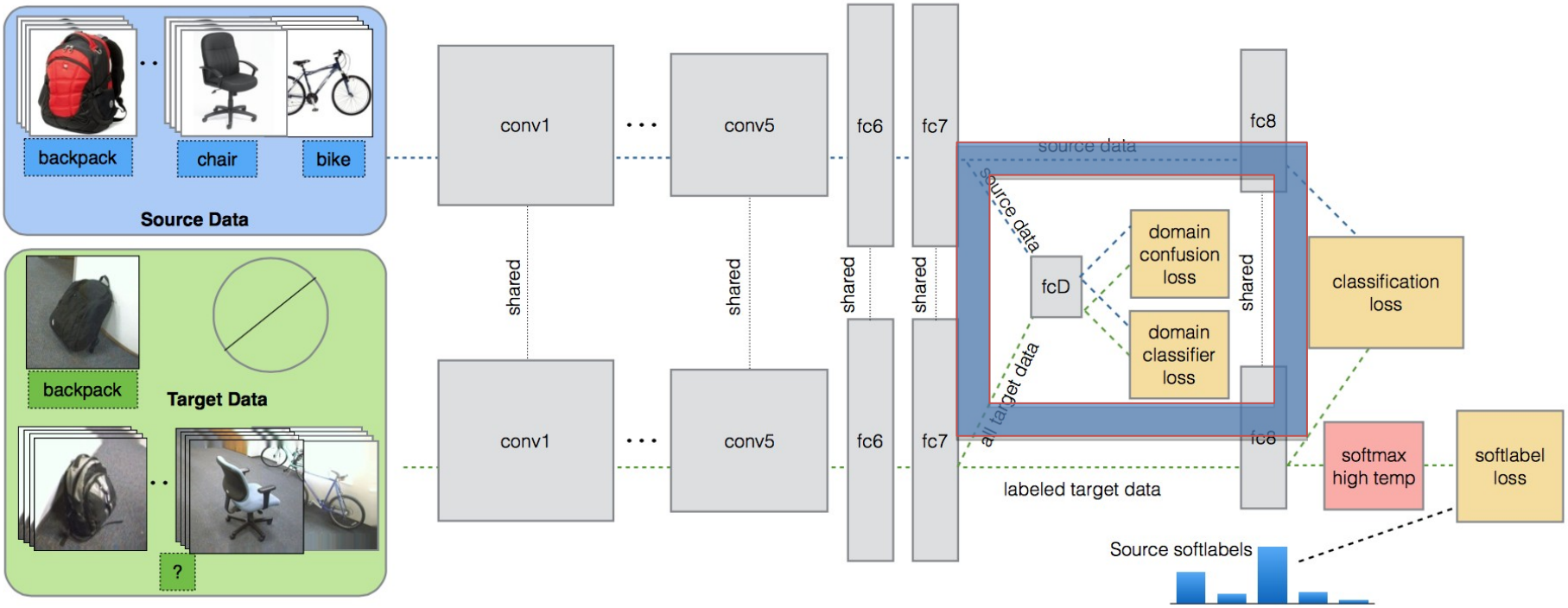


$$\mathcal{L}(x_S, y_S, x_T, y_T, \theta_D; \theta_{\text{repr}}, \theta_C) =$$
$$\mathcal{L}_C(x_S, y_S, x_T, y_T; \theta_{\text{repr}}, \theta_C)$$
$$+ \lambda \mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}})$$
$$+ \nu \mathcal{L}_{\text{soft}}(x_T, y_T; \theta_{\text{repr}}, \theta_C).$$

A familiar sight

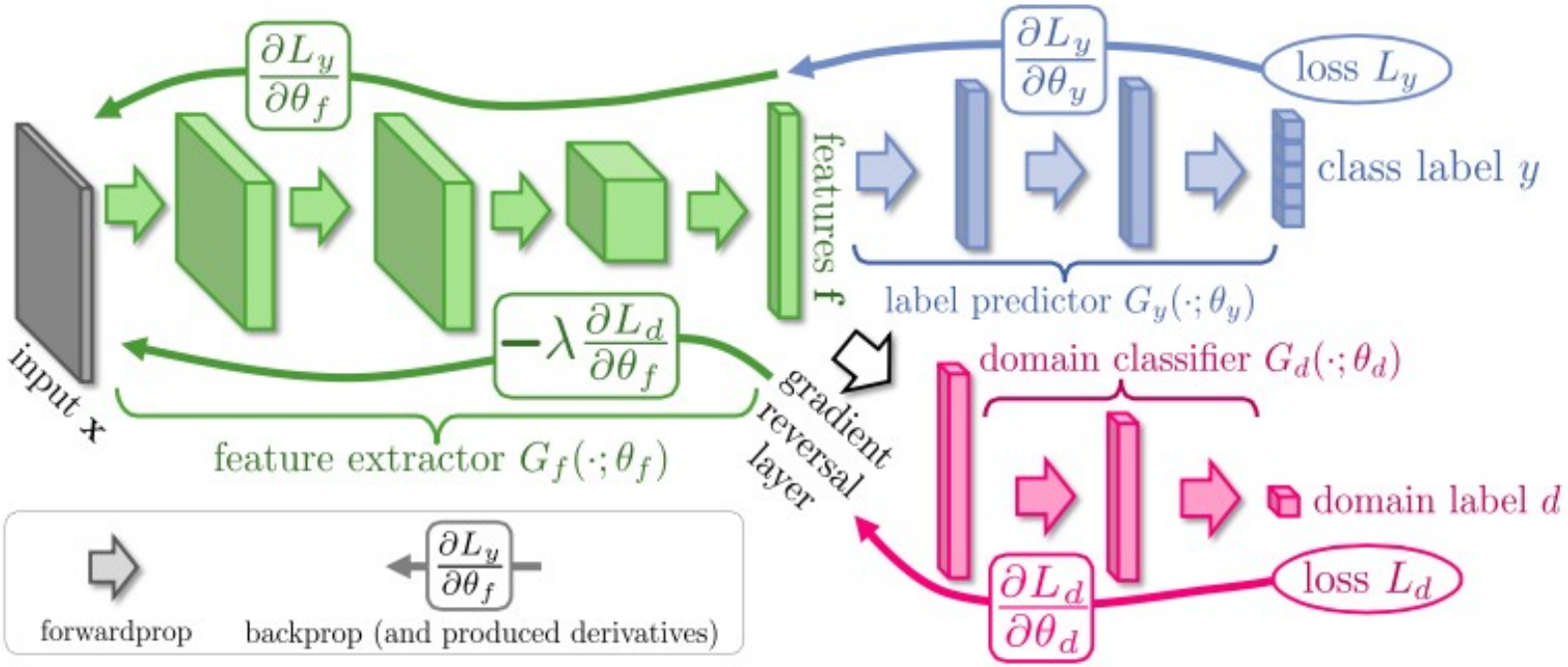


The Network



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

Credit to Ganin and Lempitsky



Domain Confusion

Their idea:

Seek parameters of features that **maximize** loss of domain classifier while simultaneously seeking the parameters of the classifier that **minimize** the loss of domain classifier

An aside on softmax:

- **Softmax**: Transforms a vector into probabilities. Useful for multi-class classification where each image only has one class label. Always sums to 1

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

x = [1 2 3]

y = exp(x)/sum(exp(x)) = [0.0900 0.2447 0.6652]

- **Logloss:** Loss function that punishes the probability on the correct class label being low

$$y = \ell(\mathbf{x}, c) = -\sum_{ij} \log x_{ijc}$$

Only the softmax value corresponding the correct label contributions
e.g. for y above with label vector [0 1 0] the loss is –log(0.2247)

Softmax loss is softmax followed by logloss

Opposing costs

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

q is a 2D vector that sums to 1

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

This loss is low if the classifier predicts the domain label correctly
e.g. q = [0.9 0.1]

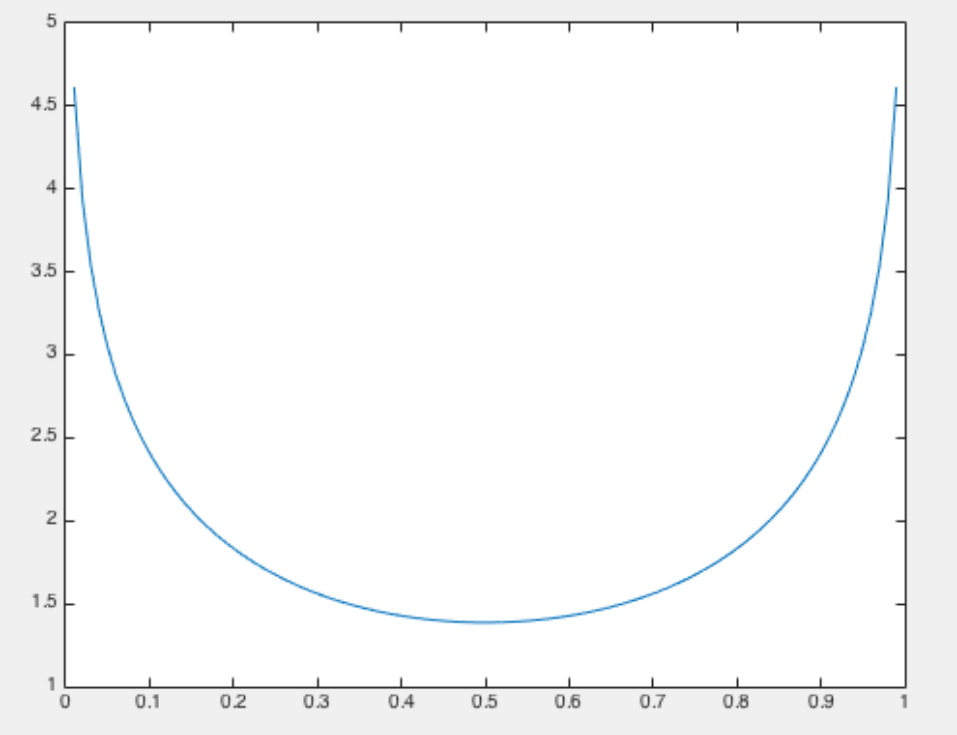
$$\mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}}) = -\sum_d \frac{1}{D} \log q_d.$$

This loss is low if the classifier cannot predict the domain label
e.g. q = [0.5 0.5]

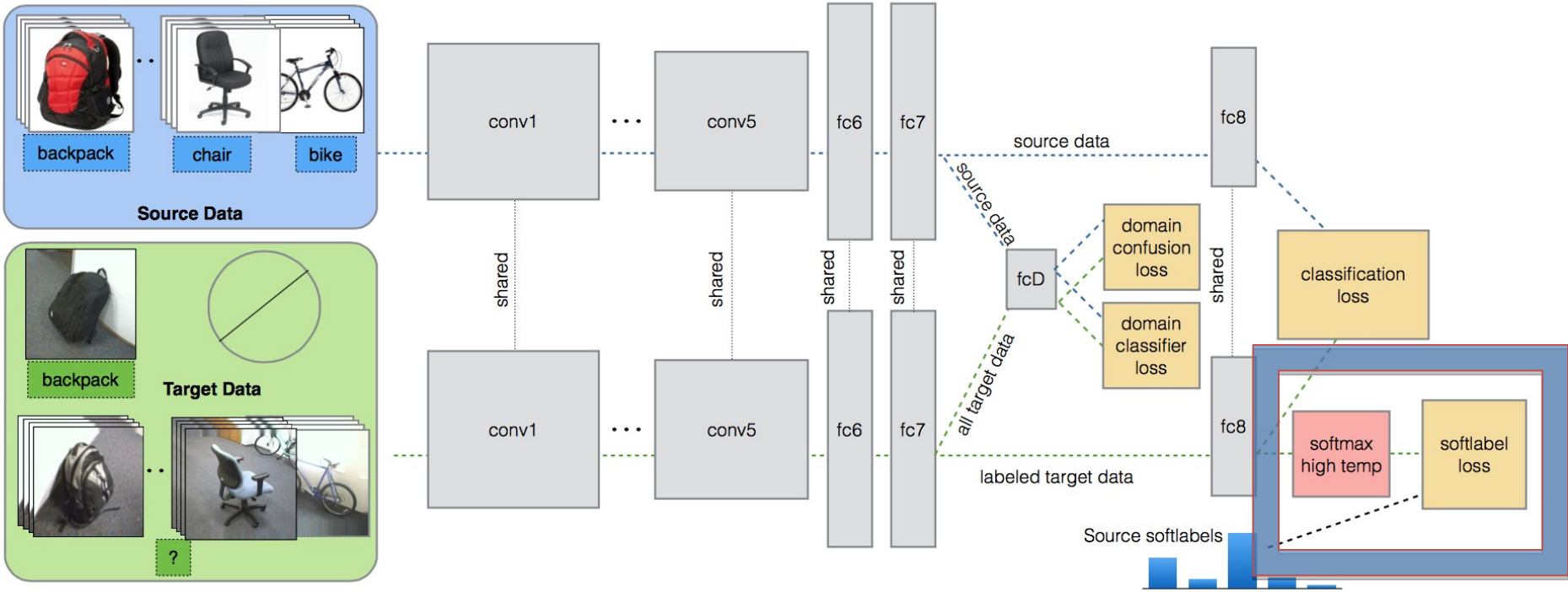
The power of MATLAB

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

```
x=0:.01:1
y = -log(x) - log(1-x)
plot(x , y)
```



The Network

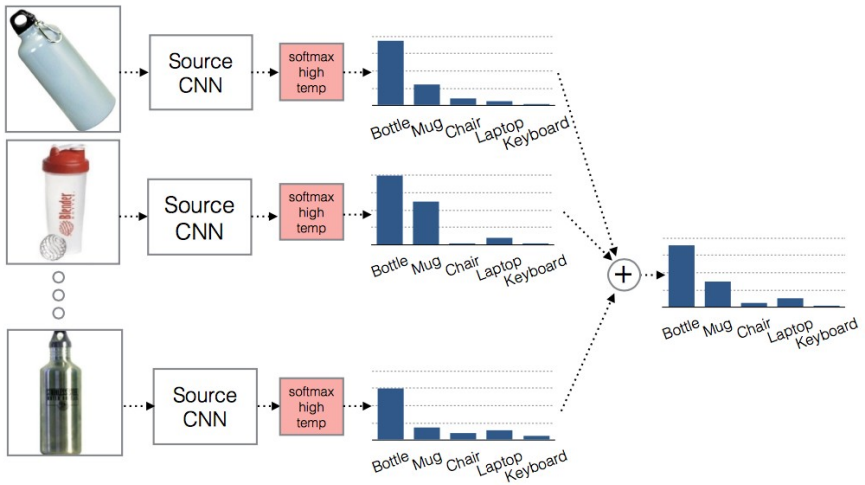


$$\mathcal{L}(x_S, y_S, x_T, y_T, \theta_D; \theta_{\text{repr}}, \theta_C) =$$
$$\mathcal{L}_C(x_S, y_S, x_T, y_T; \theta_{\text{repr}}, \theta_C)$$
$$+ \lambda \mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}})$$
$$+ \nu \mathcal{L}_{\text{soft}}(x_T, y_T; \theta_{\text{repr}}, \theta_C).$$

Soft Labels

Each class is described as a “softmax with high temperature” distribution of all classes

Softmax with temperature: Divide each value by a constant before softmaxing to reduce the effect of large values (akin to rootsift) e.g. x = [1 2 3], temp = 2 giv



This allows classes in the target domain without labels to be represented

Soft Labels

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

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

logloss now considers the probabilities for all classes
e.g. for p = [0.2 0.3 0.5], and soft label [0.1 0.1 0.9]

Loss = -0.2 * log(0.1) – 0.3*log(0.1) – 0.5*log(0.9)

Results

	$A \to W$	$A \to D$	$W \to A$	$W \to D$	$D \to A$	$D \to W$	Average
DLID [7]	51.9	–	–	89.9	–	78.2	–
DeCAF$_6$ S+T [9]	80.7 ± 2.3	–	–	–	–	94.8 ± 1.2	–
DaNN [13]	53.6 ± 0.2	–	–	83.5 ± 0.0	–	71.2 ± 0.0	–
Source CNN	56.5 ± 0.3	64.6 ± 0.4	42.7 ± 0.1	93.6 ± 0.2	47.6 ± 0.1	92.4 ± 0.3	66.22
Target CNN	80.5 ± 0.5	81.8 ± 1.0	59.9 ± 0.3	81.8 ± 1.0	59.9 ± 0.3	80.5 ± 0.5	74.05
Source+Target CNN	82.5 ± 0.9	85.2 ± 1.1	$\mathbf{65.2 \pm 0.7}$	96.3 ± 0.5	65.8 ± 0.5	93.9 ± 0.5	81.50
Ours: dom confusion only	$\mathbf{82.8 \pm 0.9}$	85.9 ± 1.1	64.9 ± 0.5	97.5 ± 0.2	$\mathbf{66.2 \pm 0.4}$	95.6 ± 0.4	82.13
Ours: soft labels only	82.7 ± 0.7	84.9 ± 1.2	$\mathbf{65.2 \pm 0.6}$	$\mathbf{98.3 \pm 0.3}$	66.0 ± 0.5	$\mathbf{95.9 \pm 0.6}$	82.17
Ours: dom confusion+soft labels	82.7 ± 0.8	$\mathbf{86.1 \pm 1.2}$	65.0 ± 0.5	97.6 ± 0.2	$\mathbf{66.2 \pm 0.3}$	95.7 ± 0.5	**82.22**

Table 1. Multi-class accuracy evaluation on the standard supervised adaptation setting with the *Office* dataset. We evaluate on all 31 categories using the standard experimental protocol from [28]. Here, we compare against three state-of-the-art domain adaptation methods as well as a CNN trained using only source data, only target data, or both source and target data together.

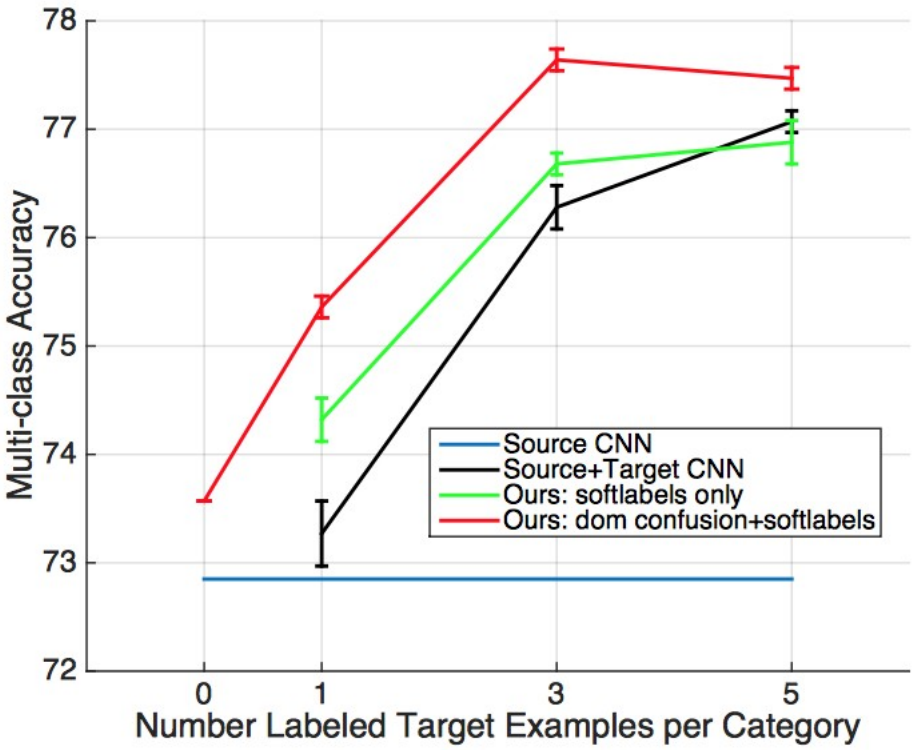
	$A \to W$	$A \to D$	$W \to A$	$W \to D$	$D \to A$	$D \to W$	Average
MMDT [18]	–	44.6 ± 0.3	–	58.3 ± 0.5	–	–	–
Source CNN	54.2 ± 0.6	63.2 ± 0.4	34.7 ± 0.1	94.5 ± 0.2	36.4 ± 0.1	89.3 ± 0.5	62.0
Ours: dom confusion only	55.2 ± 0.6	63.7 ± 0.9	$\mathbf{41.1 \pm 0.0}$	96.5 ± 0.1	41.2 ± 0.1	$\mathbf{91.3 \pm 0.4}$	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	$\mathbf{59.3 \pm 0.6}$	$\mathbf{68.0 \pm 0.5}$	40.5 ± 0.2	$\mathbf{97.5 \pm 0.1}$	$\mathbf{43.1 \pm 0.2}$	90.0 ± 0.2	**66.4**

Table 2. Multi-class accuracy evaluation on the standard semi-supervised adaptation setting with the *Office* dataset. We evaluate on 16 held-out categories for which we have no access to target labeled data. We show results on these unsupervised categories for the source only model, our model trained using only soft labels for the 15 auxiliary categories, and finally using domain confusion together with soft labels on the 15 auxiliary categories.

Copes better with colour

Figure 5. Examples from the Amazon→Webcam shift in the semi-supervised adaptation setting, where our method (the bottom turquoise label) correctly classifies images while the baseline (the top purple label) does not.

Works in transferring from ImageNet -> Caltech



Discussion