Domain adaptation and deep learning for large scale object recognition and detection

Prof. Trevor Darrell (UCB)
with
Judy Hoffman, Eric Tzeng, Jeff Donahue, Ross Girshick (UCB)
Kate Saenko (UML)
Practical image classification

1. Collect Data
2. Compute Features
3. Learn Models

Images at test time differ from the images used to learn the model!
The Adaptation Problem

- Images from different visual domains have different appearances
- These differences are what we refer to as “domain shift”
- Given an abundance of data in a source domain (e.g. Amazon), how can we perform a task in a new target domain with very little training data (e.g. Webcam)?
Background: Feature-space Transformations

source ↓ target
Background: Joint feature and Parameter Adaptation

(a) SOURCE

(b) TARGET, no adaptation

\[
\min_{W, \Theta} \frac{1}{2} \|W\|_F^2 + \frac{1}{2} \|\Theta\|_F^2 + \lambda \mathcal{L}(W, \Theta, Z, h) + \lambda_X \mathcal{L}(\Theta, X, y)
\]

 transformation parameter parameter vs target parameter vs source

regularizer loss

(Hoffman et al., ICLR 13)
Conventional Performance is limited by visual features.
Outline

• What does deep learning offer domain adaptation and vice-versa?
• A domain-adaptation perspective on large-scale ImageNet detection
• Beyond binary domain adaptation
Outline

• What does deep learning offer domain adaptation and vice-versa?
• A domain-adaptation perspective on large-scale ImageNet detection
• Beyond binary domain adaptation
First attempt: unsupervised deep learning for DA?

- Let’s try a simple idea in the spirit of [1]
  1) Pre-train unsupervised on both domains to model the data
  2) Then, supervised backprop on the source domain to learn the labels
- But, the pre-training doesn’t seem to help:
  Worse performance for MNIST->SVHN
  Negligibly improved performance for SVHN->MNIST

<table>
<thead>
<tr>
<th>Method</th>
<th>Source</th>
<th>Error Rate</th>
<th>Target</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (pre-trained)</td>
<td>MNIST</td>
<td>2.79%</td>
<td>SVHN</td>
<td>66.79%</td>
</tr>
<tr>
<td>Baseline (no pre-training)</td>
<td>MNIST</td>
<td>1.56%</td>
<td>SVHN</td>
<td>59.20%</td>
</tr>
<tr>
<td>Ours (pre-trained)</td>
<td>SVHN</td>
<td>19.33%</td>
<td>MNIST</td>
<td>41.98%</td>
</tr>
<tr>
<td>Baseline (no pre-training)</td>
<td>SVHN</td>
<td>12.38%</td>
<td>MNIST</td>
<td>42.63%</td>
</tr>
</tbody>
</table>

Still has significant domain shift despite cross domain pre-training ...

What about supervised CNNs?

• CNNs trained on large amounts of data yield state of the art performance on classification tasks; c.f. AlexNet:

• Does using a representation learned on large amounts of data remove domain shift?
“DeCAF” baselines

<table>
<thead>
<tr>
<th></th>
<th>Amazon → Webcam</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SURF</td>
<td>DeCAF₆</td>
<td>DeCAF₇</td>
</tr>
<tr>
<td>Logistic Reg. (S)</td>
<td>9.63 ± 1.4</td>
<td>48.58 ± 1.3</td>
<td>53.56 ± 1.5</td>
</tr>
<tr>
<td>SVM (S)</td>
<td>11.05 ± 2.3</td>
<td>52.22 ± 1.7</td>
<td>53.90 ± 2.2</td>
</tr>
<tr>
<td>Logistic Reg. (T)</td>
<td>24.33 ± 2.1</td>
<td>72.56 ± 2.1</td>
<td>74.19 ± 2.8</td>
</tr>
<tr>
<td>SVM (T)</td>
<td>51.05 ± 2.0</td>
<td>78.26 ± 2.6</td>
<td>78.72 ± 2.3</td>
</tr>
<tr>
<td>Logistic Reg. (ST)</td>
<td>19.89 ± 1.7</td>
<td>75.30 ± 2.0</td>
<td>76.32 ± 2.0</td>
</tr>
<tr>
<td>SVM (ST)</td>
<td>23.19 ± 3.5</td>
<td>80.66 ± 2.3</td>
<td>79.12 ± 2.1</td>
</tr>
</tbody>
</table>

monitors grouped together

...kind of!

T-SNE embedding of FC8 representations
Blue = Amazon
Green = Webcam

...but not completely.

backpacks are separated
Solving domain shift?

CNNs already remove a lot of the domain shift in the Office dataset [1]

• But, can we get even more juice out of our network by tweaking it for the target domain?

• And, what if we just use a really big source domain?
Adaptation of Supervised Deep ConvNets with Limited Training Data

• Fully training a CNN takes a long time
• It would be nice to be able to reuse a trained model on a different task
• Typical solution: fine-tune!
  – But this only works when there is a lot of training data
• Hoffman et al. [arXiv 2013 / ICLR Workshop 2014]
  – We provide the first analysis of standard domain adaptation techniques with CNNs to handle the case where target domain training data is severely limited
  – We also provide the first investigation of domain adaptation using a large-scale source domain (ImageNet!)
Network Adaptation Framework

[ Hoffman et al. arXiv 2013 / ICLR Workshop 2014]
Adaptation Methods

• Three proposed methods of adaptation:
  – Deep and Frustratingly Easy (DFE)
  – Deep Late Fusion (DLF)
  – Deep Subspace Alignment (DSA)

• Each has its own advantages and disadvantages...

[ Hoffman et al. arXiv 2013 / ICLR Workshop 2014]
Deep and Frustratingly Easy (DFE)

- Supervised adaptation technique introduced by Daumé III
- Augment feature space by introducing three components: source-specific, target-specific, and common
  - e.g. Source data has the original feature vector replicated in the source-specific and common components and zeros in the target-specific component
- Then train a classifier on the augmented features
- Usually yields good performance
- ...But requires all source and target data at train time

Deep Late Fusion (DLF)

- Train classifiers for source and target domains separately
- Final classifier score is a linear interpolation of the two separate source/target classifier scores
- Source classifier can be trained in advance and saved
- ...But the linear interpolation coefficient must be set
  - (0.8 works pretty well in practice)
Deep Subspace Alignment (DSA)

- Unsupervised adaptation
- Compute source and target subspaces, and find an alignment between them
  - Train a classifier on the source data
  - At test time, project target data into source subspace and use source classifier
- Requires no labels, source subspace can be computed in advance

Supervised setting results

<table>
<thead>
<tr>
<th>Source: 20 images/category for Webcam and DSLR</th>
<th>8 images/category for Amazon</th>
<th>Target: 3 images/category</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image_url" alt="Image" /></td>
<td><img src="image_url" alt="Image" /></td>
<td><img src="image_url" alt="Image" /></td>
</tr>
</tbody>
</table>

[ Hoffman et al. arXiv 2013 / ICLR Workshop 2014]
Unsupervised setting results

<table>
<thead>
<tr>
<th></th>
<th>A → D</th>
<th>A → W</th>
<th>D → A</th>
<th>D → W</th>
<th>W → A</th>
<th>W → D</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFK(PLS,PCA) [13]</td>
<td>-</td>
<td>15.0 ± 0.4</td>
<td>-</td>
<td>44.6 ± 0.3</td>
<td>-</td>
<td>49.7 ± 0.5</td>
<td>36.4</td>
</tr>
<tr>
<td>DA-NBNN [23]</td>
<td>-</td>
<td>23.3 ± 2.7</td>
<td>-</td>
<td>67.2 ± 1.9</td>
<td>-</td>
<td>67.4 ± 3.0</td>
<td>52.6</td>
</tr>
<tr>
<td>DLID [6]</td>
<td>-</td>
<td>26.1</td>
<td>-</td>
<td>68.9</td>
<td>-</td>
<td>84.9</td>
<td>60.0</td>
</tr>
<tr>
<td>Ours - source only</td>
<td>50.2 ± 0.6</td>
<td>45.6 ± 0.5</td>
<td>45.3 ± 0.3</td>
<td>86.5 ± 0.3</td>
<td>44.2 ± 0.3</td>
<td>88.0 ± 0.4</td>
<td>73.4</td>
</tr>
<tr>
<td>Ours - DSA</td>
<td>50.1 ± 0.4</td>
<td>46.6 ± 0.7</td>
<td>45.5 ± 0.4</td>
<td>86.2 ± 0.3</td>
<td>43.0 ± 0.4</td>
<td>86.7 ± 0.5</td>
<td>73.2</td>
</tr>
</tbody>
</table>

Source: 20 images/category for Webcam and DSLR
8 images/category for Amazon

[ Hoffman et al. arXiv 2013 / ICLR Workshop 2014 ]
Large-scale Source Domains

• What happens if we try using a much bigger source dataset instead?
• Intuitively: a sufficiently large dataset should contain images like the target domain too!

[ Hoffman et al. arXiv 2013 / ICLR Workshop 2014]
Using ImageNet as Source

Evaluation on Webcam domain

<table>
<thead>
<tr>
<th>Adaptation Method</th>
<th>Source Domain</th>
<th>Source Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Imagenet</td>
<td>Amazon</td>
</tr>
<tr>
<td>source only</td>
<td>55.5 ± 0.2</td>
<td>45.2 ± 1.4</td>
</tr>
<tr>
<td>DSA</td>
<td>55.4 ± 0.4</td>
<td>46.1 ± 2.0</td>
</tr>
</tbody>
</table>

(a) Unsupervised Adaptation

<table>
<thead>
<tr>
<th>Adaptation Method</th>
<th>Source Domain</th>
<th>Source Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ImageNet</td>
<td>Amazon</td>
</tr>
<tr>
<td>source only</td>
<td>55.5 ± 0.2</td>
<td>45.2 ± 1.4</td>
</tr>
<tr>
<td>target only</td>
<td>55.5 ± 2.7</td>
<td>55.5 ± 2.7</td>
</tr>
<tr>
<td>DLF</td>
<td>62.8 ± 1.2</td>
<td>56.8 ± 2.3</td>
</tr>
<tr>
<td>DFE</td>
<td>69.9 ± 1.5</td>
<td>64.5 ± 1.3</td>
</tr>
</tbody>
</table>

(b) Supervised Adaptation (One-Shot Adaptation)

[ Hoffman et al. arXiv 2013 / ICLR Workshop 2014]
When to Adapt?

ImageNet to Webcam

[Hoffman et al. arXiv 2013 / ICLR Workshop 2014]
Outline

• What does deep learning offer domain adaptation and vice-versa?
• A domain-adaptation perspective on large-scale ImageNet detection
• Beyond binary domain adaptation
Outline

• What does deep learning offer domain adaptation and vice-versa?
• A domain-adaptation perspective on large-scale ImageNet detection
• Beyond binary domain adaptation
Large-scale detection

- ImageNet provides huge amounts of classification data, so classification performance is very good even with a large number of categories.
- But there is comparatively little detection data, since bounding boxes are hard to come by.
  - How to learn detectors for thousands of categories?

<table>
<thead>
<tr>
<th>Maximally accurate</th>
<th>Maximally specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>tabby</td>
<td>0.58003</td>
</tr>
<tr>
<td>tiger cat</td>
<td>0.18860</td>
</tr>
<tr>
<td>Egyptian cat</td>
<td>0.18018</td>
</tr>
<tr>
<td>lynx</td>
<td>0.04025</td>
</tr>
<tr>
<td>cougar</td>
<td>0.00370</td>
</tr>
</tbody>
</table>

CNN took 0.068 seconds.
Idea: adapt classifiers for detection

PASCAL VOC Challenge

Dataset: 22k images, 50k objects, 20 classes

Detect: people, horses, sofas, bicycles, potted plants...
Progress on PASCAL VOC

- R-CNN
- DPM
- DPM++, MKL
- DPM++, MKL, Selective Search
- DPM, HOG+BOW
- DPM, MKL

VOC dataset year:
- 2006
- 2007
- 2008
- 2009
- 2010
- 2011
- 2012
- 2013

mAP (%):
- 0
- 10
- 20
- 30
- 40
- 50
- 60
- 70

Competition results:
Progress on PASCAL VOC

- DPM, MKL
- DPM++
- MKL
- Selective Search
- SegDPM [Fidler et al. 2013]
- Regionlets [Wang et al. 2013]
- DPM, HOG+BOW
- DPM, MKL
- DPM++, MKL
Progress on PASCAL VOC

- R-CNN (this work)
- Regionlets [Wang et al. 2013]
- SegDPM [Fidler et al. 2013]
- DPM, HOG+BOW
- DPM, MKL
- DPM++, MKL
- DPM++, MKL, Selective Search
ImageNet LSVR Challenge

- 1000 classes (vs. 20)
- 1.2 million training images (vs. 10k)
- Image classification (not detection)

[Deng et al. CVPR’09]
Multi-layer feature learning

“SuperVision” Convolutional Neural Network (CNN)


cf. LeCun et al. Neural Comp. ’89 & Proc. of the IEEE ‘98
Impressive ImageNet results!

1000-way image classification

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-5 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fisher Vectors (ISI)</td>
<td>26.2%</td>
</tr>
<tr>
<td>5 SuperVision CNNs</td>
<td>16.4%</td>
</tr>
<tr>
<td>7 SuperVision CNNs</td>
<td>15.3%</td>
</tr>
</tbody>
</table>

metric: classification error rate  (lower is better)

now: 12%

But... does it generalize to other datasets and tasks?

Spirited debate at ECCV 2012
Objective

Can the SuperVision CNN detect objects?
Proposed system

R-CNN: “Regions with CNN features”

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

[Girshick, Donahue, Darrell, Malik to appear in CVPR’14]

“selective search” [van de Sande et al. 2011]
R-CNN results on PASCAL

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC 2007</th>
<th>VOC 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5 (Girshick et al. 2011)</td>
<td>33.7%</td>
<td>29.6%</td>
</tr>
<tr>
<td>UVA sel. search (Uijlings et al. 2012)</td>
<td></td>
<td>35.1%</td>
</tr>
<tr>
<td>Regionlets (Wang et al. 2013)</td>
<td>41.7%</td>
<td>39.7%</td>
</tr>
</tbody>
</table>

metric: mean average precision (higher is better)
# R-CNN results on PASCAL

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC 2007</th>
<th>VOC 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5 (Girshick et al. 2011)</td>
<td>33.7%</td>
<td>29.6%</td>
</tr>
<tr>
<td>UVA sel. search (Uijlings et al. 2012)</td>
<td></td>
<td>35.1%</td>
</tr>
<tr>
<td>Regionlets (Wang et al. 2013)</td>
<td>41.7%</td>
<td>39.7%</td>
</tr>
<tr>
<td>R-CNN</td>
<td>54.2%</td>
<td>50.2%</td>
</tr>
<tr>
<td>R-CNN + bbox regression</td>
<td>58.5%</td>
<td>53.7%</td>
</tr>
</tbody>
</table>
(a) Classification CNN

Classification data from categories A and B

Train Classification CNN

Detection data from categories B

Labeled warped region

Train adapted detection CNN

Final Combined and fully adapted CNN

(b) Hidden Layer Adaptation


(c) Output Layer Adaptation
Classification CNN

• Train a classification network using your favorite architecture

• We begin with Krizhevsky et al.’s network and train it on the 1000-way ImageNet classification task

• We then replace the final layer with a new multinomial logistic regression layer to recognize the categories we are interested in

Hidden Layer Adaptation

• For the classes with detection data, we now train detectors
• Begin with weights from classification network
• Add a background class
  – Gathered by sampling selective search windows with low overlap with ground truth bounding boxes
• Fine-tune network using detection data + background patches

Output Layer Adaptation

- Directly transferring the output layer for the classes without detection data is a bad idea – preceding layers have changed!
- Simple idea: compute average classification-to-detection change for each class in the set of classes with detection data $B$:

$$ \Delta W_{avg} = \frac{1}{|B|} \sum_{i \in B} W^d_i - W^c_i. $$

Output Layer Adaptation

• Intuitively: some classes may transform differently than others
• Use the “closest” classes to determine the transformation
  – “Close” means small Euclidean distance between the L2 normalized FC7 layer parameters
• Letting $N_B(j,k)$ denote the $k$-th nearest neighbor to class $j$, we have:

$$\forall j \in A : W^d_j = W^c_j + \frac{1}{k} \sum_k \left[ W^d_{NB(j,k)} - W^c_{NB(j,k)} \right].$$

## Results on ILSVRC2013

<table>
<thead>
<tr>
<th>Detection Adaptation Layers</th>
<th>Output Layer Adaptation</th>
<th>mAP Trained 100 Categories</th>
<th>mAP Held-out 100 Categories</th>
<th>mAP All 200 Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Adapt (Classification Network)</td>
<td>-</td>
<td>12.63</td>
<td>10.31</td>
<td>11.90</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd}</td>
<td>-</td>
<td>14.93</td>
<td>12.22</td>
<td>13.60</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd},fc\textsubscript{6}</td>
<td>-</td>
<td>24.72</td>
<td>13.72</td>
<td>19.20</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd},fc\textsubscript{7}</td>
<td>-</td>
<td>23.41</td>
<td>14.57</td>
<td>19.00</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd},fc\textsubscript{B}</td>
<td>-</td>
<td>18.04</td>
<td>11.74</td>
<td>14.90</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd},fc\textsubscript{6},fc\textsubscript{7}</td>
<td>-</td>
<td>25.78</td>
<td>14.20</td>
<td>20.00</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd},fc\textsubscript{6},fc\textsubscript{7},fc\textsubscript{B}</td>
<td>-</td>
<td>26.33</td>
<td>14.42</td>
<td>20.40</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd},layers1-7,fc\textsubscript{B}</td>
<td>-</td>
<td>27.81</td>
<td>15.85</td>
<td>21.83</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd},layers1-7,fc\textsubscript{B} Avg NN (k=5)</td>
<td>-</td>
<td>28.12</td>
<td>15.97</td>
<td>22.05</td>
</tr>
<tr>
<td>fc\textsubscript{bgrnd},layers1-7,fc\textsubscript{B} Avg NN (k=100)</td>
<td>-</td>
<td>27.91</td>
<td>15.96</td>
<td>21.94</td>
</tr>
<tr>
<td>Oracle: Full Detection Network</td>
<td></td>
<td>29.72</td>
<td>26.25</td>
<td>28.00</td>
</tr>
</tbody>
</table>

Table 1: Ablation study for the pieces of DNN. We consider removing different pieces of our algorithm to determine which pieces are essential. We consider training with the first 100 (alphabetically) categories of the ILSVRC2013 detection validation set (on val1) and report mean average precision (mAP) over the 100 trained on and 100 held out categories (on val2). We find the best improvement is from fine-tuning all convolutional fully connected layers and using output layer adaptation.
Results on ILSVRC2013

- Oracle is our method trained with all 200 category detection data

Detailed analysis of error type

before adaptation    after adaptation

Held-out Categories

The adapted network makes fewer Loc and BG errors

Loc: location error   Oth: other error   BG: confused with background

Better localization

**Red**: classification network before adaptation

**Green**: adapted

7K class detector!

- Public release of a 7604 category detector trained using this method
  - 200 ILSVRC2013 classes trained with bounding box data
  - 7404 ImageNet leaf nodes trained with adaptation

http://lsda.berkeleyvision.org/
ECCV 2014 Demo...
Funny wagon
An ambulance used to transport patients to a mental hospital
Outline

• What does deep learning offer domain adaptation and vice-versa?
• A domain-adaptation perspective on large-scale ImageNet detection
• Beyond binary domain adaptation
Outline

• What does deep learning offer domain adaptation and vice-versa?
• A domain-adaptation perspective on large-scale ImageNet detection
• Beyond binary domain adaptation
Beyond Binary Domains...
Traffic Intersection Data

Labeled

Day 1
9:00

Day 1
15:00

Day 1
21:00

Unlabeled

Day 2
3:00

Day 2
9:00
Automobiles across decades
Unsupervised Adaptation

(Gopalan, ICCV 11), (Gong, CVPR 12), (Fernando, ICCV 13)
Continuous Unsupervised Adaptation

\[ \min_{P_t^T P_t = I, \tilde{A}_t, \tilde{B}_t} r(P_{t-1}, P_t) + R_{err}(z_t, P_t) + \psi(U \tilde{A}_t, P_t \tilde{B}_t) \]

where \( r(\cdot) \) is a regularizer that encourages the new subspace learned at time \( t \) to be close to the previous subspace of time \( t - 1 \).

Continuous Manifold Based Adaptation for Evolving Visual Domains
[CVPR 2014]

Judy Hoffman
UC Berkeley, EECS
jhoffman@eecs.berkeley.edu

Trevor Darrell
UC Berkeley, EECS
trevor@eecs.berkeley.edu

Kate Saenko
UMass Lowell, CS
saenko@cs.uml.edu
Table 2. Our method, CMA, continues to provide improvement for the scene classification task even when testing over the 5 days following the labeled training data. We show here average precision (%) for the 2400 test images following the 50 available labeled training images.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>KNN</td>
<td>71.24±5.7</td>
<td>47.51±5.1</td>
<td>52.27±3.4</td>
<td>39.91±3.0</td>
</tr>
<tr>
<td>-</td>
<td>SVM</td>
<td>80.40±0.6</td>
<td>68.69±3.6</td>
<td>50.98±3.6</td>
<td>48.91±3.0</td>
</tr>
<tr>
<td>CMA+GFK</td>
<td>KNN</td>
<td>77.21±3.8</td>
<td>52.97±2.7</td>
<td>39.08±2.6</td>
<td>47.39±2.8</td>
</tr>
<tr>
<td>CMA+GFK</td>
<td>SVM</td>
<td>84.17±1.5</td>
<td>59.53±2.9</td>
<td>47.39±2.8</td>
<td></td>
</tr>
<tr>
<td>CMA+SA</td>
<td>KNN</td>
<td>78.61±3.3</td>
<td>51.33±4.2</td>
<td>38.21±2.6</td>
<td></td>
</tr>
<tr>
<td>CMA+SA</td>
<td>SVM</td>
<td>84.32±1.4</td>
<td>59.68±2.9</td>
<td>49.05±2.8</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Our method, CMA, improves performance independent of the feature choice for the scene classification task. Results here are shown with optimizing the unsupervised adaptation problem using either the geodesic flow kernel (GFK) [9] or the subspace alignment (SA) method [7]. Average precision (%) is recorded when training with 50 labeled images and testing on the immediately following 24 hours (480 images).
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>KNN</td>
<td>66.31±0.6</td>
<td>72.77±0.8</td>
<td>84.60±0.7</td>
</tr>
<tr>
<td>-</td>
<td>SVM</td>
<td>79.26±0.6</td>
<td>76.40±0.7</td>
<td>85.92±0.4</td>
</tr>
<tr>
<td>CMA+GFK</td>
<td>KNN</td>
<td>66.32±0.2</td>
<td>72.60±0.9</td>
<td>82.65±0.5</td>
</tr>
<tr>
<td>CMA+GFK</td>
<td>SVM</td>
<td>80.24±0.7</td>
<td>78.32±0.6</td>
<td>89.68±0.1</td>
</tr>
<tr>
<td>CMA+SA</td>
<td>KNN</td>
<td>65.06±1.1</td>
<td>71.44±1.3</td>
<td>81.97±0.6</td>
</tr>
<tr>
<td>CMA+SA</td>
<td>SVM</td>
<td>79.79±0.6</td>
<td>78.31±0.7</td>
<td>89.71±0.1</td>
</tr>
</tbody>
</table>

Table 3. Our algorithm improves performance on category recognition task. We evaluate our continuous manifold adaptation approach (CMA) on the task of labeling images of automobiles as either cars or trucks. We show results using two solutions to the unsupervised adaptation problem (GFK[9] and SA[7]) and two inner product based source classifiers (KNN and SVM). We compare across three types of features and demonstrate the benefit of using our algorithm for each feature choice, including a deep learning based feature that was tuned for object classification on all of ImageNet[3].

Figure 7. Our method clearly adapts to vehicle appearance as it evolves to look different from that in the labeled 50’s training data. We show example images misclassified by non-adaptive SVM (DeCAF features) and correctly classified by CMA followed by the same SVM classifier. The 5 sedans and 5 trucks for which the SVM had the highest confidence (though incorrect) are displayed here.
Conclusions

• What does deep learning offer domain adaptation and vice-versa?
• Detection is a profitable “domain”
• Generalization beyond the two-domain setting is important
Domain adaptation and deep learning for large scale object recognition and detection

Prof. Trevor Darrell (UCB)
with
Judy Hoffman, Eric Tzeng, Jeff Donahue, Ross Girshick (UCB)
Kate Saenko (UML)