Transfer learning and domain adaptation
Semi-supervised and transfer learning

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

Source labels

Source model

Source data
E.g. ImageNet

Target labels

Target model

Target data
E.g. PASCAL

Transfer Learned Knowledge

Large amount of data/labels

Small amount of data/labels
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?
“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.
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

<table>
<thead>
<tr>
<th>Method</th>
<th>mean Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSV [27]</td>
<td>43.0</td>
</tr>
<tr>
<td>SIFT internal [27]</td>
<td>55.1</td>
</tr>
<tr>
<td>SIFT boundary [27]</td>
<td>32.0</td>
</tr>
<tr>
<td>HOG [27]</td>
<td>49.6</td>
</tr>
<tr>
<td>HSV+SIFTi+SIFTb+HOG(MKL) [27]</td>
<td>72.8</td>
</tr>
<tr>
<td>BOW(4000) [14]</td>
<td>65.5</td>
</tr>
<tr>
<td>SPM(4000) [14]</td>
<td>67.4</td>
</tr>
<tr>
<td>FLH(100) [14]</td>
<td>72.7</td>
</tr>
<tr>
<td>BiCos seg [7]</td>
<td>79.4</td>
</tr>
<tr>
<td>Dense HOG+Coding+Pooling[2] w/o seg</td>
<td>76.7</td>
</tr>
<tr>
<td>Seg+Dense HOG+Coding+Pooling[2]</td>
<td>80.7</td>
</tr>
<tr>
<td>CNN-SVM w/o seg</td>
<td>74.7</td>
</tr>
<tr>
<td>CNNaug-SVM w/o seg</td>
<td><strong>86.8</strong></td>
</tr>
</tbody>
</table>

Can we do better than off the shelf features?

**Domain adaptation**
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
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.**

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!

Unsupervised domain adaptation

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

## Unsupervised domain adaptation

<table>
<thead>
<tr>
<th>Method</th>
<th>Source</th>
<th>Source</th>
<th>Target</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MNIST</td>
<td>MNIST-M</td>
<td>SVHN</td>
<td>SVHN</td>
<td>MNIST</td>
</tr>
<tr>
<td><strong>SOURCE ONLY</strong></td>
<td>.5749</td>
<td>.8665</td>
<td>.5919</td>
<td>.7400</td>
<td></td>
</tr>
<tr>
<td><strong>SA (Fernando et al., 2013)</strong></td>
<td>.6078 (7.9%)</td>
<td>.8672 (1.3%)</td>
<td>.6157 (5.9%)</td>
<td>.7635 (9.1%)</td>
<td></td>
</tr>
<tr>
<td><strong>PROPOSED APPROACH</strong></td>
<td>.8149 (57.9%)</td>
<td>.9048 (66.1%)</td>
<td>.7107 (29.3%)</td>
<td><strong>.8866 (56.7%)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TRAIN ON TARGET</strong></td>
<td>.9891</td>
<td>.9244</td>
<td>.9951</td>
<td>.9987</td>
<td></td>
</tr>
</tbody>
</table>

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