Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Large deep convnets
4. Beyond ImageNet
   1. Fully Convolutional Networks (FCNs)
   2. Segmentation
3. Transfer
Transfer from ImageNet (source)

Transfer as generic features
Brut Deep features (learned from ImageNet)
(== a learned embedding from Image to vector representation)

Transfer learning (from source to target)
Frozen features + SVM => solution to small datasets
Frozen features + Deep
Fine tuning not easy in that case (small datasets)
Transfer from source (=ImageNet task) to target task

**Source:** ImageNet (dataset + 100 classes) => AlexNet trained

**Target:** new dataset Cal-101 and new classification task with 101 classes => Chopped

AlexNet (layer i) + SVM trained on

> Results better than SoA CV methods on Cal-101!
Transfer: fine-tuning of a deep model on target task

Train a deep (AlexNet) on source (ImageNet)
Keep the deep params. for target and complete with a small deep on top (fully trained on target task)
Fine-tune the whole model on target data
  Challenge: only limited target data, careful about overfitting
  Solution: Freeze the gradient’s update for AlexNet part
Transfer: fine-tuning of a deep model on target task

Train a deep (AlexNet) on source (ImageNet)
Keep the deep params. for target and complete with a small deep on top (fully trained on target task)
Fine-tune the whole model on target data
  Challenge: only limited target data, careful about overfitting
  Solution: Freeze the gradient’s update for AlexNet part
Other solution: use smaller gradient’s update for AlexNet part
Transfer: which parts of the deep?

Which layer(s) can be transferred (copied)?

- Speech: usually copy the last few layers
- Image: usually copy the first few layers
Transfer: which supervision?

• Task description
  • Source data: \((x^s, y^s)\) \hspace{1cm} A large amount
  • Target data: \((x^t, y^t)\) \hspace{1cm} (Very) little

Rq: Few/One-shot learning: only a few/one examples in target domain

Many different contexts:

In vision: from large dataset (ImageNet) to small datasets (VOC2007)

In speech: (supervised) speaker adaption
  • Source data: audio data and transcriptions from many speakers
  • Target data: audio data and its transcriptions of specific user
More on transfer framework

<table>
<thead>
<tr>
<th>Target Data</th>
<th>Source Data (ImageNet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelled</td>
<td>Frozen or fine-tuning</td>
</tr>
<tr>
<td></td>
<td>Few</td>
</tr>
<tr>
<td></td>
<td>One</td>
</tr>
<tr>
<td></td>
<td>Zero</td>
</tr>
</tbody>
</table>

Main purposes:
Similar visual domain?
Same tasks (ie class)?
Similar domain: ImageNet task => Dog/Cat task

Target: Dog/Cat Classifier

cat
dog

Data *not directly related to* the task considered

ImageNet: Similar domain, different task (1000 classes but NOT Dog and Cat classes)
General Framework for Transfer Learning

Target: Dog/Cat Classifier

Data *not directly related to* the task considered

Similar domain, completely different tasks

Different domains, same task
## General Framework for Transfer Learning

<table>
<thead>
<tr>
<th>Source Data (not directly related to the task)</th>
<th>Target Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>labelled</td>
</tr>
<tr>
<td></td>
<td>Fine-tuning</td>
</tr>
<tr>
<td></td>
<td>Multitask Learning</td>
</tr>
<tr>
<td></td>
<td>Domain-adversarial training</td>
</tr>
<tr>
<td></td>
<td>Zero-shot learning</td>
</tr>
</tbody>
</table>
## General Framework for Transfer Learning

<table>
<thead>
<tr>
<th>Source Data (not directly related to the task)</th>
<th>Target Data</th>
<th>labelled</th>
<th>Unlabelled</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelled</td>
<td></td>
<td><strong>Fine-tuning</strong></td>
<td></td>
</tr>
<tr>
<td>unlabelled</td>
<td></td>
<td><strong>Multitask Learning</strong></td>
<td>Not considered here</td>
</tr>
</tbody>
</table>

Not considered here
Multitask Learning

- The multi-layer structure makes NN suitable for multitask learning.
<table>
<thead>
<tr>
<th>Source Data (not directly related to the task)</th>
<th>labelled</th>
<th>unlabeled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>labelled</td>
<td><strong>Fine-tuning</strong></td>
<td>Not considered here</td>
</tr>
<tr>
<td><strong>Multitask Learning</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unlabeled</td>
<td><strong>Domain adaptation-adversarial training</strong></td>
<td>Not considered here</td>
</tr>
</tbody>
</table>
Unsupervised Domain Adaptation (UDA)

Source data: \((x^s, y^s)\)  \(\rightarrow\)  Training data
Target data: \((x^t)\)

\{ Same task, domain mismatch \}

MNIST

SOURCE  
\(4 \ 0 \ 1\)  with labels

TARGET  
\(8 \ 1 \ 6\)  without labels

Final test on target domain!
Main principle: diminish the domain shift in the learned features, encourage domain confusion
UDA strategy: align both domains
UDA strategy: 1/ domain-adversarial training

Add to the feature generator (G) a domain classifier (discriminant D) for which labels are available!

Learn Gand D:
- G tries to align domains
- D tries to identify domains

Rq: Similar to GAN (coming soon)
UDA strategy: 1/ domain-adversarial training
   2/ classification task (same for source and target here)

Maximize label classification accuracy +
minimize domain classification accuracy

Not only cheat the domain classifier, but satisfying label classifier at the same time

Maximize label classification accuracy

Maximize domain classification accuracy
UDA strategy: joint learning

Domain classifier fails in the end
It should struggle ......

Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016
Domain-adversarial training

<table>
<thead>
<tr>
<th>METHOD</th>
<th>SOURCE</th>
<th>TARGET</th>
<th>MNIST</th>
<th>SYN NUMBERS</th>
<th>SVHN</th>
<th>SYN SIGNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE ONLY</td>
<td>MNIST-M</td>
<td>SVHN</td>
<td>.5749</td>
<td>.8665</td>
<td>.5919</td>
<td>.7400</td>
</tr>
<tr>
<td>SA (FERNANDO ET AL., 2013)</td>
<td>MNIST-M</td>
<td>SVHN</td>
<td>.6078 (7.9%)</td>
<td>.8672 (1.3%)</td>
<td>.6157 (5.9%)</td>
<td>.7635 (9.1%)</td>
</tr>
<tr>
<td>PROPOSED APPROACH</td>
<td>MNIST</td>
<td>SVHN</td>
<td>.8149 (57.9%)</td>
<td>.9048 (66.1%)</td>
<td>.7107 (29.3%)</td>
<td>.8866 (56.7%)</td>
</tr>
<tr>
<td>TRAIN ON TARGET</td>
<td>MNIST</td>
<td>SVHN</td>
<td>.9891</td>
<td>.9244</td>
<td>.9951</td>
<td>.9987</td>
</tr>
</tbody>
</table>

Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016
Domain adaptation

Main principle: diminish the domain shift in the learned features, encourage domain confusion

Another example: Adversarial Discriminative Domain Adaptation [Tzeng et al. 2017]
Domain adaptation

Other architecture
Domain adaptation

Other architecture: Image translation for Domain adaptation [Murez 2017]
Domain adaptation

Other architecture: Image translation for Domain adaptation [Murez 2017]
## Transfer Learning - Overview

<table>
<thead>
<tr>
<th>Target Data</th>
<th>Source Data (not directly related to the task)</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelled</td>
<td>labelled</td>
</tr>
<tr>
<td></td>
<td>Fine-tuning</td>
</tr>
<tr>
<td></td>
<td><em>Multitask Learning</em></td>
</tr>
<tr>
<td></td>
<td>Domain adaptation-adversarial training</td>
</tr>
<tr>
<td></td>
<td><em>Zero-shot learning</em></td>
</tr>
<tr>
<td>unlabeled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>unlabeled</td>
</tr>
<tr>
<td></td>
<td>Not considered here</td>
</tr>
<tr>
<td></td>
<td>Not considered here</td>
</tr>
</tbody>
</table>
Zero-shot Learning

- Source data: \((x^S, y^S)\) ➔ Training data
- Target data: \((\emptyset)\) usually same domain

Different tasks

Training time:

- \(x^S:\) cat, dog, ...
- \(y^S:\) 

Test time \(x^T:\)

- => Fish class!
Zero-shot Learning

- Representing each class by its attributes

- **Training**
  - NN
  - 1
  - furry
  - 0
  - 4 legs
  - 1
  - tail

- Database attributes

<table>
<thead>
<tr>
<th></th>
<th>furry</th>
<th>4 legs</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Fish</td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>Chimp</td>
<td>O</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- sufficient attributes for one to one mapping
Zero-shot Learning

- Representing each class by its attributes

**Testing**

Find the class with the most similar attributes

<table>
<thead>
<tr>
<th>attributes</th>
<th>furry</th>
<th>4 legs</th>
<th>tail</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Fish</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Chimp</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

sufficient attributes for one to one mapping
Zero-shot Learning

• Attribute embedding + class (word name) embedding
Zero-shot Learning

- Attribute embedding

\[ f(\ast) \text{ and } g(\ast) \text{ can be NN.} \]

Training target:

\[ f(x^n) \text{ and } g(y^n) \text{ as close as possible} \]

\[ f(x^1) \quad g(y^1) \]

\[ f(x^2) \quad g(y^2) \]

\[ g(y^3) \quad f(y^3) \]

\[ y_i \text{ are linked together by a class relationship (e.g. class name embedding as W2v)} ]