Deep Learning - MAI

Transfer Learning
“Don’t be a hero” - Andrej Karpathy

The Transfer Learning philosophy
Learning from scratch

- Trying to learn from scratch is difficult and arduous
  - You have to learn many fundamental things before getting to learn complex aspects of your task
- It’s easier to learn if you already know something beforehand
  - There are some basic things needed to learn anything
  - In image processing, learning to “see”: Characterize images based on fundamental visual features
Why Transfer Learning

- You can learn **faster**
  - If I know that much, I’m that much closer to my goal

- You can learn **better**
  - There is a limited amount of things you can learn from data before getting trapped in spurious patterns.
  - What would you rather learn from your data?
Putting things in perspective

The ImageNet ¿success?
Growing up

- **1998** LeNet-5
- **2012** AlexNet
- **2014** VGG19
- **2014** GoogLeNet
- **2015** Inception-V3
- **2015** ResNet-56
What we get

- We solved ImageNet
What we pay

- Data labeling, transfer & storage
  - e.g., 1,000 images per class

- Training cost
  - Money (hardware, energy, salaries)
  - Environmental cost (CO$_2$ emissions)
  - Human effort
    - Highly skilled professionals
    - Architecture design
    - Hyper-parameter fine tuning
The ImageNet way is no way

- We **cannot** do that for every single problem out there
  - The cost is too high. But more importantly...
The ImageNet way is no way

- We **cannot** do that for every single problem out there
  - The cost is too high. But more importantly...
- We **do not want to** do that for every single problem out there
  - TL to the rescue
- Transfer learning reduces the requirements on...
  - Data (implicit reuse of data)
  - Cost (faster convergence)
  - Effort (initial design & parametrization)
The essence of Transfer Learning

Learning it's all about generalization
What is learning about?
What is learning about?

Train set

Test set
What is learning about?

Train set

Test set
What is learning about?

Train set

FAIL

Test set

MODEL

TRAIN

PREDICT
What is the bias here?

Train set

Test set
What is the bias here?

Train set

GREEN BACKGROUND

DOG
What is the bias here?

Test set

CAT ← GREEN BACKGROUND
What is the bias here?

Train and test sets have different conditional probability distributions.
Fixing bias

Train set

Solution?
Fixing bias

Train set

Solution?

Randomizing ensures that train and test sets have similar conditional probability distributions
Fixing bias

Train set

Solution?

Randomizing ensures that train and test sets have similar conditional probability distributions. They will never be exactly equal. This is why overfitting exists.
Fixing bias

Train set

More similarity means better generalization. But generalization to what?
What is learning about?

Generalization between samples **from the same source** can be (approximately) ensured through randomization.
What is learning about?

Is the same source enough?
What do we really want to generalize to?
What is the **real purpose** of the model?
What should we test on?

Train set

GENERALIZATION

25
A realistic scenario
A realistic scenario

Generalization in this case is less certain
Error is expected to rise
Is it fixable?

Train set

Test set
Formalizing Transfer Learning

Tasks and Domains

Formalizing transfer learning

Domain:

What is the nature of data?
Which is it manifold?

Task:
Formalizing transfer learning

**Domain:** \( \mathcal{D} = \{X, P(X)\} \)

- A feature space \( X \)

- A marginal probability distribution \( P(X), \text{where } X = \{x_1, \ldots, x_n\} \in X \)

**Task:**

- Bag of words
- Content vector

"The Elgar Concert Hall at the University of Birmingham for our third conference"
Formalizing transfer learning

Domain:

Task:

What is the mapping of data?
How is it computed?
Formalizing transfer learning

**Domain:** \( \mathcal{D} = \{ \mathcal{X}, P(X) \} \)
- A feature space \( \mathcal{X} \)

- A marginal probability distribution \( P(X) \), where \( X = \{ x_1, \ldots, x_n \} \in \mathcal{X} \)

**Task:** \( \mathcal{T} = \{ y, f(\cdot) \} \)
- A label space \( y \)
  - \( \text{CAT, DOG} \neq \text{LION, WOLF} \)

- An objective predictive function \( f(\cdot) \Leftrightarrow P(y|x) \)

- Bag of words
- Content vector

"The Elgar Concert Hall at the University of Birmingham for our third conference"
Formalizing transfer learning

**Domain:** $\mathcal{D} = \{\mathcal{X}, P(X)\}$
- A feature space $\mathcal{X}$
  - The Same (different)
- A marginal probability distribution $P(X)$
  - Different
  - Similar

**Task:** $\mathcal{T} = \{y, f(\cdot)\}$
Formalizing transfer learning

Domain: $\mathcal{D} = \{\mathcal{X}, P(X)\}$
- A feature space $\mathcal{X}$
  - The Same (different)
- A marginal probability distribution $P(X)$
  - Different
  - Similar

Task: $\mathcal{T} = \{y, f(\cdot)\}$
- A label space $y$
  - Different
  - The same

- An objective predictive function
  - Different (but similar?)

Source

Target

{CAT, DOG}
{FELINE, CANINE}

{LION, WOLF}
{FELINE, CANINE}

$f_S(\cdot)$

$f_T(\cdot)$
What is transfer learning about?

Train set

Test set
What is transfer learning about?

Source domain

Target domain
What is transfer learning about?

Source domain
- Task: \( y = \{\text{CAT, DOG}\}, f_s(\cdot) \)

Target domain
- Task: \( y = \{\text{LION, WOLF}\}, f_T(\cdot) \)
Formalizing transfer learning

**Domain:** \( \mathcal{D} = \{ \mathcal{X}, P(X) \} \)
- A feature space \( \mathcal{X} \)
  - The Same (different)
- A marginal probability distribution \( P(X) \)
  - Different
  - Similar

**Task:** \( \mathcal{T} = \{ y, f(\cdot) \} \)
- A label space \( y \)
  - Different
  - The same

- An objective predictive function
  - Different (but similar?)

\( f_S(\cdot) \) \quad f_T(\cdot)
Formalizing transfer learning

Source
\( \hat{f}_S(\cdot) \)

Target
\( \hat{f}_T(\cdot) \)

- Are they similar?
- Can we just use \( \hat{f}_S(\cdot) \) to approximate \( f_T(\cdot) \)?
- Can we reuse part of it?
Representation Learning & Classifiers

Learning to describe
A typical classifier

- **Support Vector Machine (SVM)** is just a classifier (a very good one).

- SVM find the best boundary separating the data instances into different classes in a **given** feature space.
A good classifier

- SVMs using the **kernel trick** can overcome the linear limitation through an **implicit** mapping to a higher dimensional feature space.
Deep Neural Networks and classifiers

Intermediate representations

Linear Classifier

Input

Output
Classifiers and Representations

- Classifiers are **Task**-specific
  - We can rarely reuse them for a different task, as they are bounded to the **label space**

- Representations are **Domain**-specific
  - We can often reuse them for a different Task if we remain in the same **feature space**!
Reusing Deep Representations

Save the Earth - Reuse DNNs
What can be saved?

Intermediate representations

Linear Classifier

Input

Output
What can be saved?

Intermediate representations

Input

Linear Classifier

Output
Feature extraction

Intermediate representations

Input
Feature extraction

Intermediate representations

Input

Target Task Labels

SV M

Feature extraction

Intermediate representations

Input

Target Task Labels

SV M
reuse all

- DNN last layer features + SVM
  - Feature extraction
  - Very similar task and same domain
Fine tuning

What if the tasks are quite different?

Intermediate representations

Input

Source Task

Linear Classifier

Output
Fine tuning

Intermediate representations

Features learned for the Source Task

Can we make them better?

Input

Input Features learned for the Source Task

Can we make them better?
Fine tuning

Intermediate representations

Input

Target Task Labels
Fine tuning

Effect of fine tuning is diminished by depth
Fine tuning

- Difficult to control how much do we re-train.
  - Reduced learning rate (1/10)
  - Early stopping
  - Alternate source/target sampling
Retrain

- DNN last layer features + SVM
  - Feature extraction
  - Very similar task and same domain

- Train one or several NN layers + pre-trained layers
  - Fine tuning
  - Sort of similar task and same domain
  - Data volume
Knowledge inside DNN

- Conv1: depth=64, 3x3 conv, conv1_1, conv1_2
- Conv2: depth=128, 3x3 conv, conv2_1, conv2_2
- Conv3: depth=256, 3x3 conv, conv3_1, conv3_2, conv3_3, conv3_4
- Conv4: depth=512, 3x3 conv, conv4_1, conv4_2, conv4_3, conv4_4
- Conv5: depth=512, 3x3 conv, conv5_1, conv5_2, conv5_3, conv5_4
- FC1: size=4096
- FC2: size=1000
- Softmax
Knowledge inside DNN

More influenced by domain

More influenced by task
Knowledge inside DNN

Fine Tuning

To improve, to remember, to forget
The choices in fine tuning

- **Reuse and freeze**
  - Use source task status
  - “It's good as it is”

- **Reuse and fine tune**
  - Start from source task status, adjust with target task
  - “It’s a good starting point”

- **Train from scratch**
  - Reinitialize weights randomly, train with target task only
  - “It’s pretty much useless”
The order of fine tuning

Freeze

Fine tune

Random

- Freeze
  - depth=64
  - 3x3 conv
  - conv1_1
  - conv1_2

- Fine tune
  - depth=128
  - 3x3 conv
  - conv2_1
  - conv2_2
  - depth=256
  - 3x3 conv
  - conv3_1
  - conv3_2
  - conv3_3
  - conv3_4
  - depth=512
  - 3x3 conv
  - conv4_1
  - conv4_2
  - conv4_3
  - conv4_4
  - depth=512
  - 3x3 conv
  - conv5_1
  - conv5_2
  - conv5_3
  - conv5_4

- Random
  - size=4096
  - FC1
  - FC2
  - size=1000
  - softmax
The order of fine tuning

**Freeze**

Layers which are pretty stable and universal
Increase with source-target similarity

```
depth=64
3x3 conv
conv1_1
conv1_2
```

```
depth=128
3x3 conv
conv2_1
conv2_2
```

```
depth=256
3x3 conv
conv3_1
conv3_2
conv3_3
conv3_4
```

```
depth=512
3x3 conv
conv4_1
conv4_2
conv4_3
conv4_4
```

```
depth=512
3x3 conv
conv5_1
conv5_2
conv5_3
conv5_4
```

```
size=4096
FC1

size=1000
FC2
softmax
```
The order of fine tuning

**Fine tune** Layers which are pretty similar but improvable
Increase with source-target similarity & data volume
The order of fine tuning

Random: Layers which are pretty dissimilar
Increase with source-target dissimilarity & data volume
Trade-off of fine tuning

- **Reuse and freeze**
  - Remove parameters for target to learn (needs data but allows focus)
  - Adds noise

- **Reuse and fine tune**
  - Allows to focus learning (requires data)
  - Adds bias

- **Random init**
  - Again, from the top (cost, cost, cost)
  - Tailor made for target
Feature Extraction

To improve, to remember, to forget
Factors deep representations quality

- Source task
  - Total volume
  - Class variety

- Target task
  - Source-target similarity

- Starting Model
  - Capacity
  - Accuracy
Factors deep representations quality

- **Source task**
  - Total volume
  - Class variety

- **Target task**
  - Source-target similarity

- **Starting Model**
  - Capacity
  - Accuracy

If you have all of this, feature extraction plus a classifier will get you close to state-of-the-art in 10 minutes of CPU.
Which layers to use?

- If source & target task are VERY similar, use the “classifier” layers
Which layers to use?

- If source & target task are NOT very similar, broaden the scope.
Feature extraction normalization

- When doing feature extraction for a regular classifier (e.g., SVM) each feature is assumed to be i.i.d. (not even close!)

- Beware of size
  - FC layers have lots of activations
  - Conv layers activations are spatially dependent

- Beware of scale
  - Different layers activate with different strength

- Default solution: L2-norm (by layer)
  - Does not fix scale (careful if mixing layers!)

<table>
<thead>
<tr>
<th>VGG16</th>
<th>Convs</th>
<th>FCs</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers:</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>Activations:</td>
<td>33%</td>
<td>66%</td>
</tr>
</tbody>
</table>

Conv2, Conv13, FC2
Advanced feature normalization

- Normalizing features considering the target
  - Feature standardization (vertically instead of horizontally)

- For each feature:
  - Compute mean and std dev. on target training set
  - Normalize feature-wise to zero mean, one std dev.
  - Features are adapted to target domain

Best for multi-layer feature extraction
Advanced feature normalization

- Dimensionality of extracted features is an issue (12K in VGG16)
- Removing complexity without losing expressivity
  - Discretizing the space (-1,0,1)

*Full-Network Embedding (FNE)*
Feature extraction in action

- High similarity source - target

Network pre-trained on **Places2** for mit67 and on **ImageNet** for the rest.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>mit67</th>
<th>cub200</th>
<th>flowers102</th>
<th>cats-dogs</th>
<th>sdogs</th>
<th>caltech101</th>
<th>food101</th>
<th>textures</th>
<th>wood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline fc6</td>
<td>80.0</td>
<td>65.8</td>
<td>89.5</td>
<td>89.3</td>
<td>78.0</td>
<td>91.4±0.6</td>
<td>61.4±0.2</td>
<td>69.6</td>
<td>70.8±6.6</td>
</tr>
<tr>
<td>Baseline fc7</td>
<td>81.7</td>
<td>63.2</td>
<td>87.0</td>
<td>89.6</td>
<td>79.3</td>
<td>89.7±0.3</td>
<td>59.1±0.6</td>
<td>69.0</td>
<td>68.9±6.8</td>
</tr>
<tr>
<td>Full-network</td>
<td>83.6</td>
<td>65.5</td>
<td>93.3</td>
<td>89.2</td>
<td>78.8</td>
<td>91.4±0.6</td>
<td>67.0±0.7</td>
<td>73.0</td>
<td>74.1±6.9</td>
</tr>
<tr>
<td>SotA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ED</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>-</td>
</tr>
<tr>
<td>FT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>-</td>
</tr>
</tbody>
</table>
Feature extraction in action

- High similarity source - target

Network pre-trained on **Places2** for mit67 and on **ImageNet** for the rest.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>mit67</th>
<th>cub200</th>
<th>flowers102</th>
<th>cats-dogs</th>
<th>sdogs</th>
<th>caltech101</th>
<th>food101</th>
<th>textures</th>
<th>wood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline fc6</td>
<td>80.0</td>
<td>65.8</td>
<td>89.5</td>
<td>89.3</td>
<td>78.0</td>
<td>91.4±0.6</td>
<td>61.4±0.2</td>
<td>69.6</td>
<td>70.8±6.6</td>
</tr>
<tr>
<td>Baseline fc7</td>
<td>81.7</td>
<td>63.2</td>
<td>87.0</td>
<td>89.6</td>
<td>79.3</td>
<td>89.7±0.3</td>
<td>59.1±0.6</td>
<td>69.0</td>
<td>68.9±6.8</td>
</tr>
<tr>
<td>Full-network</td>
<td>83.6</td>
<td>-0.3</td>
<td>93.3</td>
<td>-0.4</td>
<td>-0.5</td>
<td>91.4±0.6</td>
<td>67.0±0.7</td>
<td>73.0</td>
<td>74.1±6.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ED</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>FT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>-</td>
</tr>
</tbody>
</table>

Task similarity makes single layer l2-norm competitive

+2.9
+4.2
Feature extraction in action

- High similarity source - target

Network pre-trained on **Places2** for mit67 and on **ImageNet** for the rest.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>mit67</th>
<th>cub200</th>
<th>flowers102</th>
<th>cats-dogs</th>
<th>sdogs</th>
<th>caltech101</th>
<th>food101</th>
<th>textures</th>
<th>wood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline $f_6$</td>
<td>80.0</td>
<td>65.8</td>
<td>89.5</td>
<td>89.3</td>
<td>78.0</td>
<td>91.4±0.6</td>
<td>61.4±0.2</td>
<td>69.6</td>
<td>70.8±6.6</td>
</tr>
<tr>
<td>Baseline $f_7$</td>
<td>81.7</td>
<td>63.2</td>
<td>87.0</td>
<td>89.6</td>
<td>79.3</td>
<td>89.7±0.3</td>
<td>59.1±0.6</td>
<td>69.0</td>
<td>68.9±6.8</td>
</tr>
<tr>
<td>Full-network</td>
<td>83.6</td>
<td>65.5</td>
<td>93.3</td>
<td>89.2</td>
<td>78.8</td>
<td>91.4±0.6</td>
<td>67.0±0.7</td>
<td>73.0</td>
<td>74.1±6.9</td>
</tr>
<tr>
<td>SotA</td>
<td>86.9</td>
<td><strong>92.3</strong></td>
<td><strong>97.0</strong></td>
<td><strong>91.6</strong></td>
<td><strong>90.3</strong></td>
<td><strong>93.4</strong></td>
<td><strong>77.4</strong></td>
<td><strong>75.5</strong></td>
<td>-</td>
</tr>
<tr>
<td>ED</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Data (external or not) can make fine tuning worth the COST

+7.9
Feature extraction in action

- Low similarity source - target (*most real-world scenario!*)

Network pre-trained on **ImageNet** for mit67 and on **Places2** for the rest.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>mit67</th>
<th>cub200</th>
<th>flowers102</th>
<th>cats-dogs</th>
<th>caltech101</th>
<th>textures</th>
<th>wood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline fc7</td>
<td>72.2</td>
<td>23.6</td>
<td>73.3</td>
<td>38.7</td>
<td>72.0</td>
<td>55.8</td>
<td>65.3</td>
</tr>
<tr>
<td>Full-network</td>
<td>75.5</td>
<td>35.5</td>
<td>88.7</td>
<td>56.2</td>
<td>80.0</td>
<td>65.1</td>
<td>74.0</td>
</tr>
</tbody>
</table>

|       | +3.3 | +11.9 | +15.4 | +17.5 | +8.0 | +9.3 | +10.6 |

Data (external or not) makes fine tuning worth the COST
Key takeaways

● If possible, always use a pre-trained net
  ○ Don’t be a hero

● Consider the gradient of representations
  ○ From data to task

● Always analyze
  ○ Source/Target similarity
  ○ Data availability
Key takeaways

- Fine tune if possible
  - Freeze from the bottom
  - Fine tune the middle
  - Retrain from scratch at the top

- Feature extraction
  - Must-do baseline (cheap and easy!)
  - Best approach if data volume is short
● Is there a pre-trained model in a very similar domain?
  ○ Yes: Do FT. With 5-10 samples is already better than FE.
  ○ No. Do FE. Unless you have +100 samples/class and/or perform exhaustive hyperparameter tuning.

● What about human cost, time and environmental footprint?
Dario Garcia-Gasulla (BSC)
dario.garcia@bsc.es