Transfer Learning
Transfer learning with convolutional networks
Transfer learning with convolutional networks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Non-Convnet Method</th>
<th>Non-Convnet perf</th>
<th>Pretrained convnet + classifier</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caltech 101</td>
<td>MKL</td>
<td>84.3</td>
<td>87.7</td>
<td>+3.4</td>
</tr>
<tr>
<td>VOC 2007</td>
<td>SIFT+FK</td>
<td>61.7</td>
<td>79.7</td>
<td>+18</td>
</tr>
<tr>
<td>CUB 200</td>
<td>SIFT+FK</td>
<td>18.8</td>
<td>61.0</td>
<td>+42.2</td>
</tr>
<tr>
<td>Aircraft</td>
<td>SIFT+FK</td>
<td>61.0</td>
<td>45.0</td>
<td>-16</td>
</tr>
<tr>
<td>Cars</td>
<td>SIFT+FK</td>
<td>59.2</td>
<td>36.5</td>
<td>-22.7</td>
</tr>
</tbody>
</table>
Why transfer learning?

• Availability of training data

• Computational cost

• Ability to pre-compute feature vectors and use for multiple tasks

• Con: NO end-to-end learning
Finetuning
Initialize with pre-trained, then train with low learning rate
## Finetuning

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Non-Convnet Method</th>
<th>Non-Convnet perf</th>
<th>Pretrained convnet + classifier</th>
<th>Finetuned convnet</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caltech 101</td>
<td>MKL</td>
<td>84.3</td>
<td>87.7</td>
<td>88.4</td>
<td>+4.1</td>
</tr>
<tr>
<td>VOC 2007</td>
<td>SIFT+FK</td>
<td>61.7</td>
<td>79.7</td>
<td>82.4</td>
<td>+20.7</td>
</tr>
<tr>
<td>CUB 200</td>
<td>SIFT+FK</td>
<td>18.8</td>
<td>61.0</td>
<td>70.4</td>
<td>+51.6</td>
</tr>
<tr>
<td>Aircraft</td>
<td>SIFT+FK</td>
<td>61.0</td>
<td>45.0</td>
<td>74.1</td>
<td>+13.1</td>
</tr>
<tr>
<td>Cars</td>
<td>SIFT+FK</td>
<td>59.2</td>
<td>36.5</td>
<td>79.8</td>
<td>+20.6</td>
</tr>
</tbody>
</table>
What impacts transfer accuracies?

• Relationship between pre-training and target task?

• Unclear: sometimes transfer works across very different domains
  • E.g., ImageNet -> Satellite images

• Very limited work on understanding this
What impacts transfer accuracy?

• Size of the pre-training dataset
• Size of the model
• Bigger is better

Concerns about big transfer

• Opaque datasets?
• Uncurated datasets?
• Bias in the datasets?
  • Do biased datasets affect transfer? Turns out yes.
Computational complexity
Analyzing computational complexity

• What is the computational complexity of a single convolutional layer?
  • $h \times w \times c$ input and output
  • $k \times k$ kernel

• Space:
  • Input/output: $hwc$
  • Filters: $k^2c^2$

• Time (Flops): $hwk^2c^2$
Reducing computational complexity

• ...while maintaining accuracy?

• Multiple ways:
  • Make architecture *a priori* cheaper
  • Make *weights* and *operations* cheaper
  • Make inference adaptive
Cheaper convolutional blocks

• **Standard convolution:**
  - Each filter operates on all channels
  - Single $k \times k$ filter operating on $c$ channels producing one output channel: $k^2c$ parameters, cost
  - $c$ such filters: $k^2c^2$ parameters, cost

• **Depthwise separable convolution**
  - Each filter operates on a single channel
  - $c$ filters operating on $c$ channels: $k^2c$ parameters, cost
  - But each channel is independently processed
  - Add a 1x1 convolution at the end with cost $c^2 : k^2c + c^2$ parameters
Cheaper convolutional blocks

• Depthwise separable convolutions are specific instance of more general idea: *grouped convolutions*

• Grouped convolutions in original AlexNet network

• Grouped convolution:
  • Divide input channels into $g$ groups
  • Apply convolutional layers on each group independently
  • Concatenate
Grouped and depth-wise convolutions

Table 4. Depthwise Separable vs Full Convolution MobileNet

<table>
<thead>
<tr>
<th>Model</th>
<th>ImageNet</th>
<th>Million</th>
<th>Million</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv MobileNet</td>
<td>71.7%</td>
<td>4866</td>
<td>29.3</td>
</tr>
<tr>
<td>MobileNet</td>
<td>70.6%</td>
<td>569</td>
<td>4.2</td>
</tr>
</tbody>
</table>


Other architectural changes

• Biggest memory consumption: large feature maps
Other architectural changes

• Biggest memory consumption: large feature maps

• Simple solution: reduce resolution early
Other architectural changes

- Biggest memory consumption: large feature maps
- Simple solution (ResNet):
  - Reduce resolution drastically (/4) early
- More sophisticated changes: Inverted residuals (MobileNet v2)

---

Other kinds of connections

- **DenseNets**
  - Replace addition of residuals with concatenation
  - Alternative to solving vanishing gradient problem
  - Should *increase* number of parameters, but *decreases* them
  - Better re-use of features

*Figure 1*: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.
Adaptive inference

• Some examples are harder than others
• Should be able to use different amounts of computation for different examples
• Version 1: skip some residuals


Adaptive inference

• Some examples are harder than others
• Should be able to use different amounts of computation for different examples
• Version 1: skip some residuals


Adaptive inference

• Some examples are harder than others
• Should be able to use different amounts of computation for different examples
• Version 2: reduce resolution at different rates

Huang, Gao, et al. "Multi-scale dense networks for resource efficient image classification." ICLR 2018
Compressing model weights

• All of model storage: filters
• Flops also scale with non-zero entries in filters (in principle)
• Compress filters
  • Sparsify them
  • Represent them with fewer bits
Pruning network connections

• Simple approach: prune weights that are below a threshold
• Retrain rest of the weights
• Repeat

Filter quantization

• Two questions:
  • How do we quantize?
  • Quantization → discrete values. How do we optimize?

• Example 1: cluster
  • Weights → indices into dictionary
  • Update dictionary elements as parameters.

Filter quantization

• Two questions:
  • How do we quantize?
  • Quantization \(\rightarrow\) discrete values. How do we optimize?

• Example 2: \textit{binarize/ternarize}
  • Weights \(\rightarrow\) binary/ternary, + real-valued scale
  • Parameter updates happen in real space

Beyond ConvNets
Attention (Transformers)

- Comes from the NLP community
- Originally an approach for language generation
- Motivation: need to attend to different parts of the sentence for, e.g., translation

He went to the river

वह नदी के पास गया
Attention (Transformers)

• Comes from the NLP community
• Is an approach for processing sets

Attention (Transformers)

• Comes from the NLP community
• Is an approach for processing sets

Queries

Keys

Values

Queries
Attention (Transformers)

• Comes from the NLP community
• Is an approach for processing sets
Attention (Transformers)

Input

Queries

Keys

Values

Output

Attention
Attention (Transformers)


Transformers in computer vision

• An evolving frontier
  • First key results only late last year

• Preliminary conclusions
  • Transformers can outperform convnets, but require much stronger data augmentation
  • Transformers seem to transfer better.