Image Classification
Convolutional networks - Why

• Convolutions
  • Reduce parameters
  • Capture shift-invariance: location of patch in image should not matter

• Subsampling
  • Allows greater invariance to deformations
  • Allows the capture of large patterns with small filters
How to do machine learning

• Create training / validation sets
• Identify loss functions
• Choose hypothesis class
• Find best hypothesis by minimizing training loss
How to do machine learning

- Create training / validation sets
- Identify loss functions
- Choose hypothesis class
- Find best hypothesis by minimizing training loss

\[ h(x) = s \]

\[ \hat{p}(y = k|x) \propto e^{s_k} \]

\[ \hat{p}(y = k|x) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

\[ L(h(x), y) = -\log \hat{p}(y|x) \]

Negative log likelihood for multiclass classification

Multiclass classification !!
Negative log likelihood for multiclass classification

\[ L(h(x), y) = -\log \hat{p}(y|x) \]

- Often represent label as a ``one-hot'' vector \( y \)
  - \( y = [0, 0, ..., 1, ..., 0] \)
  - \( y_k = 1 \) if label is \( k \), 0 otherwise

\[ L(h(x), y) = - \sum_k y_k \log \hat{p}(y = k|x) \]
Building a convolutional network

conv + relu + subsample

conv + relu + subsample

conv + relu + subsample

average pool

linear

10 classes
Building a convolutional network
Building a convolutional network

5x5 conv, no subsample

5x5 conv, subsample by 2

Flatten+ Linear

Linear

Linear(classifier)
Training the network

![Graph showing training and test error over epochs with a note on overfitting.](image)
Controlling overfitting in convolutional networks

- Reduce parameters?
- Increase dataset size?
  - Automatically by jittering examples - “Data augmentation”
Controlling overfitting in convolutional networks

• Dropout: Internally create data augmentations
  • Randomly zero out some fraction of values before a layer
  • Can be thought of as per-layer data augmentation
  • Typically applied on inputs to linear layers (since linear layers have tons of parameters)
Dropout

Without dropout
Train error: 0%
Test error: 1%

With dropout
Train error: 0.7%
Test error: 0.85%
## MNIST Classification

<table>
<thead>
<tr>
<th>Method</th>
<th>Error rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear classifier over pixels</td>
<td>12</td>
</tr>
<tr>
<td>Non-linear classifier over pixels</td>
<td>1.41</td>
</tr>
<tr>
<td>Linear classifier over HOG</td>
<td>1.44</td>
</tr>
<tr>
<td>Kernel SVM over HOG</td>
<td>0.79</td>
</tr>
<tr>
<td>Convolutional Network</td>
<td>0.95</td>
</tr>
</tbody>
</table>
ImageNet

• 1000 categories
• ~1000 instances per category

ImageNet

- Top-5 error: algorithm makes 5 predictions, true label must be in top 5
- Useful for incomplete labelings
Challenge winner's accuracy

- 2010: 26.3%
- 2011: 27.3%
- 2012: 37.8%
- 2014: 7.9%
- 2015: 8.2%

Layer counts:
- 7-layer Convolutional Networks
- 19 layers
- 152 layers
Exploring convnet architectures
Deeper is better

Challenge winner's accuracy

- 7 layers
- 16 layers
Deeper is better

Challenge winner's accuracy

- 2010: Alexnet
- 2011: Alexnet
- 2012: VGG16
- 2013: VGG16
- 2014: VGG16
The VGG pattern

- Every convolution is 3x3, padded by 1
- Every convolution followed by ReLU
- ConvNet is divided into “stages”
  - Layers within a stage: no subsampling
  - Subsampling by 2 at the end of each stage
- Layers within stage have the same number of channels
- Every subsampling → double the number of channels
Example network
Challenges in training: exploding / vanishing gradients

• Vanishing / exploding gradients

\[
\frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \ldots \frac{\partial z_{i+1}}{\partial z_i}
\]

• If each term is (much) greater than 1 \(\rightarrow\) explosion of gradients
• If each term is (much) less than 1 \(\rightarrow\) vanishing gradients
Residual connections

• In general, gradients tend to vanish
• Key idea: allow gradients to flow unimpeded

\[ z_{i+1} = f_{i+1}(z_i, w_{i+1}) \]

\[ \frac{\partial z_{i+1}}{\partial z_i} = \frac{\partial f_{i+1}(z_i, w_{i+1})}{\partial z_i} \]

\[ \frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i} \]
Residual connections

- In general, gradients tend to vanish
- Key idea: allow gradients to flow unimpeded

\[ z_{i+1} = g_{i+1}(z_i, w_{i+1}) + z_i \]

\[ \frac{\partial z_{i+1}}{\partial z_i} = \frac{\partial g_{i+1}(z_i, w_{i+1})}{\partial z_i} + I \]

\[ \frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i} \]
Residual block
Residual connections

- Assumes all $z_i$ have the same size
- True within a stage
- Across stages?
  - Doubling of feature channels
  - Subsampling
- Increase channels by 1x1 convolution
- Decrease spatial resolution by subsampling

$$z_{i+1} = g_{i+1}(z_i, w_{i+1}) + \text{subsample}(Wz_i)$$
The ResNet pattern

- Decrease resolution substantially in first layer
  - Reduces memory consumption due to intermediate outputs
- Divide into stages
  - maintain resolution, channels in each stage
  - halve resolution, double channels between stages
- Divide each stage into residual blocks
- At the end, compute average value of each channel to feed linear classifier
Putting it all together - Residual networks

Challenge winner's accuracy

- 2010
- 2011
- 2012
- 2013
- 2014
- 2015
Transfer learning with convolutional networks

Trained feature extractor

Linear classifier

Horse
Transfer learning with convolutional networks

• What do we do for a new image classification problem?

• Key idea:
  • Freeze parameters in feature extractor
  • Retrain classifier

Linear classifier

Trained feature extractor
## Transfer learning with convolutional networks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Best Non-Convnet perf</th>
<th>Pretrained convnet + classifier</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caltech 101</td>
<td>84.3</td>
<td>87.7</td>
<td>+3.4</td>
</tr>
<tr>
<td>VOC 2007</td>
<td>61.7</td>
<td>79.7</td>
<td>+18</td>
</tr>
<tr>
<td>CUB 200</td>
<td>18.8</td>
<td>61.0</td>
<td>+42.2</td>
</tr>
<tr>
<td>Aircraft</td>
<td>61.0</td>
<td>45.0</td>
<td>-16</td>
</tr>
<tr>
<td>Cars</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
Finetuning

Initialize with pre-trained, then train with low learning rate
## Finetuning

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Best 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>84.3</td>
<td>87.7</td>
<td>88.4</td>
<td>+4.1</td>
</tr>
<tr>
<td>VOC 2007</td>
<td>61.7</td>
<td>79.7</td>
<td>82.4</td>
<td>+20.7</td>
</tr>
<tr>
<td>CUB 200</td>
<td>18.8</td>
<td>61.0</td>
<td>70.4</td>
<td>+51.6</td>
</tr>
<tr>
<td>Aircraft</td>
<td>61.0</td>
<td>45.0</td>
<td>74.1</td>
<td>+13.1</td>
</tr>
<tr>
<td>Cars</td>
<td>59.2</td>
<td>36.5</td>
<td>79.8</td>
<td>+20.6</td>
</tr>
</tbody>
</table>
Visualizing convolutional networks
Receptive field

- Which input pixels does a particular unit in a feature map depend on

convolve with 3 x 3 filter
Receptive field

- 5x5 receptive field
  - convolve with 3 x 3 filter
  - 3x3 receptive field
  - convolve with 3 x 3 filter
  - final output
Receptive field

convolve with 3 x 3 filter, subsample
Receptive field

7x7 receptive field: union of 9 3x3 fields with stride of 2

Convolve with 3 x 3 filter, subsample by factor 2

3x3 receptive field

Convolve with 3 x 3 filter
Visualizing convolutional networks

- Take images for which a given unit in a feature map scores high
- Identify the receptive field for each.

Visualizing convolutional networks II

• Block regions of the image and classify

Visualizing convolutional networks II

- Image pixels important for classification = pixels when blocked cause misclassification.