Neural network training
Vagaries of optimization

• Non-convex
  • Local optima
  • Sensitivity to initialization

• Vanishing / exploding gradients

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

• If each term is (much) greater than 1 \(\rightarrow\) explosion of gradients
• If each term is (much) less than 1 \(\rightarrow\) vanishing gradients
Vanishing and exploding gradients

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

\[
\frac{\partial L}{\partial z_i} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial z_i}
\]

\[
\lambda_{\text{min}}\left(\frac{\partial z}{\partial z_i}\right) \frac{\partial L}{\partial z} \leq \frac{\partial L}{\partial z} \frac{\partial z}{\partial z_i} \leq \lambda_{\text{max}}\left(\frac{\partial z}{\partial z_i}\right) \frac{\partial L}{\partial z}
\]

\[
\lambda_{\text{max}}(UV) \leq \lambda_{\text{max}}(U)\lambda_{\text{max}}(V)
\]

\[
\lambda_{\text{min}}(UV) \geq \lambda_{\text{min}}(U)\lambda_{\text{min}}(V)
\]

\[
\lambda_{\text{max}}(A^n) = \lambda_{\text{max}}(A)^n
\]

\[
\lambda_{\text{min}}(A^n) = \lambda_{\text{min}}(A)^n
\]
Sigmoids cause vanishing gradients
Convolution subsampling convolution

- Convolution
- Subsampling
- Convolution
Rectified Linear Unit (ReLU)

• \( \text{max} (x,0) \)
• Also called half-wave rectification (signal processing)
Image Classification
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

$h(x) = s$

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

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

$L(h(x), y) = -\log \hat{p}(y|x)$
## 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>
</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: Convolutional Networks
- 2011: Convolutional Networks
- 2012: Convolutional Networks
Why do convnets work?

• Claim: ConvNets have way more parameters than traditional models
  • Wrong: contemporary models had same or more parameters

• Claim: Deep models are more expressive than shallow models
  • Wrong: 3 layer neural networks are *universal function approximators*

• What does depth provide?
  • More non-linearities: many ways of expressing non-linear functions
  • More module reuse: really long switch-case vs functions
  • More parameter sharing: most computation is shared amongst categories
Visualizing convolutional networks I

Visualizing convolutional networks II

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

Convolutional Networks and the Brain
Transfer learning
Transfer learning with convolutional networks

Trained feature extractor $\phi$
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
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>
Exploring convnet architectures
Deeper is better
Deeper is better

Challenge winner's accuracy

- 2010: Alexnet
- 2011: Alexnet
- 2012: Alexnet
- 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 same number of channels
• Every subsampling $\rightarrow$ double the number of channels
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
Challenges in training: dependence on init
Solutions

• Careful init

• Batch normalization

• Residual connections
Careful initialization

- Key idea: want variance to remain approx. constant
  - Variance increases in backward pass => exploding gradient
  - Variance decreases in backward pass => vanishing gradient
- “MSRA initialization”
  - weights = Gaussian with 0 mean and variance = $2/(k\times k \times d)$
Batch normalization

• Key idea: normalize so that each layer output has zero mean and unit variance
  • Compute mean and variance for each channel
  • Aggregate over batch
  • Subtract mean, divide by std

• Need to reconcile train and test
  • No “batches” during test
  • After training, compute means and variances on train set and store

Residual connections

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

\[
\begin{align*}
  z_{i+1} &= f_{i+1}(z_i, w_{i+1}) \\
  \frac{\partial z_{i+1}}{\partial z_i} &= \frac{\partial f_{i+1}(z_i, w_{i+1})}{\partial z_i} \\
  \frac{\partial z}{\partial z_i} &= \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i}
\end{align*}
\]
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 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

$$z_{i+1} = g_{i+1}(z_i, w_{i+1}) + \text{subsample}(Wz_i)$$
A residual block

• Instead of single layers, have residual connections over block
Bottleneck blocks

• Problem: When channels increases, 3x3 convolutions introduce many parameters
  • $3 \times 3 \times c^2$

• Key idea: use 1x1 to project to lower dimensionality, do convolution, then come back
  • $c \times d + 3 \times 3 \times d^2 + d \times c$
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
Semantic Segmentation
The Task
Evaluation metric

• Pixel classification!
• Accuracy?
  • Heavily unbalanced
• Intersection over Union
  • Average across classes and images
• Per-class accuracy
  • Average across classes and images
Things vs Stuff

THINGS
- Person, cat, horse, etc
- Constrained shape
- Individual instances with separate identity
- May need to look at objects

STUFF
- Road, grass, sky etc
- Amorphous, no shape
- No notion of instances
- Can be done at pixel level
- “texture”
Challenges in data collection

• Precise localization is hard to annotate

• Annotating every pixel leads to heavy tails

• Common solution: annotate few classes (often things), mark rest as “Other”

• Common datasets: PASCAL VOC 2012 (~1500 images, 20 categories), COCO (~100k images, 20 categories)
Pre-convnet semantic segmentation

• Things
  • Do object detection, then segment out detected objects

• Stuff
  • ”Texture classification”
  • Compute histograms of filter responses
  • Classify local image patches
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks

Convolve with \#classes 1x1 filters

\( \frac{w}{4} \) \hspace{1cm} \( \frac{h}{4} \)
Semantic segmentation using convolutional networks

• Pass image through convolution and subsampling layers
• Final convolution with #classes outputs
• Get scores for \textit{subsampled} image
• Upsample back to original size
Semantic segmentation using convolutional networks
The resolution issue

• Problem: Need fine details!
• Shallower network / earlier layers?
  • Not very semantic!
• Remove subsampling?
  • Looks at only a small window!
Solution 1: Image pyramids

Solution 2: Skip connections
Solution 2: Skip connections
Skip connections

Skip connections

- Problem: early layers not semantic