Introduction to Convolutional Networks

Lecture 3

Rob Fergus

New York University
Convolutional Neural Networks

- LeCun et al. 1989
- Neural network with specialized connectivity structure
Multistage Hubel-Wiesel Architecture

- Stack multiple stages of simple cells / complex cells layers
- Higher stages compute more global, more invariant features
- Classification layer on top

History:
- Neocognitron [Fukushima 1971-1982]
- Convolutional Nets [LeCun 1988-2007]
- HMAX [Poggio 2002-2006]
- Many others….
Overview of Convnets

• **Feed-forward:**
  – Convolve input
  – Non-linearity (rectified linear)
  – Pooling (local max)

• **Supervised**

• **Train convolutional filters by back-propagating classification error**

![Diagram of Convolutional Neural Networks](image)

LeCun et al. 1998
Convnet Successes

- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011])
  - Arabic & Chinese [Ciresan et al. 2012]

- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 2011]

- But less good at more complex datasets
  - E.g. Caltech-101/256 (few training examples)
Application to ImageNet

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk

[Deng et al. CVPR 2009]

ImageNet Classification with Deep Convolutional Neural Networks

[NIPS 2012]
**Goal**

- **Image Recognition**
  - Pixels $\rightarrow$ Class Label

[Image of lenses, abacus, slug, and hen with class labels](#)

[Krizhevsky et al. NIPS 2012](#)
Krizhevsky et al. [NIPS2012]

- Same model as LeCun’98 but:
  - Bigger model (8 layers)
  - More data (10^6 vs 10^3 images)
  - GPU implementation (50x speedup over CPU)
  - Better regularization (DropOut)

- 7 hidden layers, 650,000 neurons, 60,000,000 parameters
- Trained on 2 GPUs for a week
Examples

• From Clarifai.com
Examples

• From Clarifai.com

Predicted Tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>2.30%</td>
</tr>
<tr>
<td>helsinki</td>
<td>1.80%</td>
</tr>
<tr>
<td>fish</td>
<td>1.40%</td>
</tr>
<tr>
<td>port</td>
<td>1.10%</td>
</tr>
<tr>
<td>istanbul</td>
<td>1.10%</td>
</tr>
<tr>
<td>beach</td>
<td>1.00%</td>
</tr>
<tr>
<td>denmark</td>
<td>1.00%</td>
</tr>
<tr>
<td>copenhagen</td>
<td>0.90%</td>
</tr>
<tr>
<td>sea</td>
<td>0.80%</td>
</tr>
<tr>
<td>boat</td>
<td>0.80%</td>
</tr>
</tbody>
</table>
Examples

• From Clarifai.com
Using Features on Other Datasets

• Train model on ImageNet 2012 training set

• Re-train classifier on new dataset
  – Just the top layer (softmax)

• Classify test set of new dataset
Training Images per-class

Accuracy %

6 training examples

The Details

• Operations in each layer

• Architecture

• Training

• Results
Components of Each Layer

Pixels / Features

Filter with learned dictionary

Non-linearity

Spatial local max pooling

Output Features
Defining Convolution

• Let $f$ be the image and $g$ be the kernel. The output of convolving $f$ with $g$ is denoted $f \ast g$.

$$(f \ast g)[m, n] = \sum_{k, l} f[m - k, n - l] g[k, l]$$

• Convention: kernel is “flipped”
• MATLAB: conv2 (also imfilter)

Source: F. Durand
Key properties

• **Linearity**: \( \text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2) \)

• **Shift invariance**: same behavior regardless of pixel location: \( \text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f)) \)

• Theoretical result: any linear shift-invariant operator can be represented as a convolution
Annoying details

• What is the size of the output?
• MATLAB: `conv2(f, g, shape)`
  
  – `shape = 'full'`: output size is sum of sizes of `f` and `g`
  – `shape = 'same'`: output size is same as `f`
  – `shape = 'valid'`: output size is difference of sizes of `f` and `g`
ConvNet Architecture

• Exploits two properties of images:

• 1. Dependencies are local
  – No need to have each unit connect to every pixel

• 2. Spatially stationary statistics
  – Translation invariant dependencies
  – Only approximately true
Filtering

- **Convolution**
  - Filter is learned during training
  - Same filter at each location
Filtering

- Local
  - Each unit layer above look at local window
  - But no weight tying

- E.g. face recognition

Filters

Input
Filtering

- **Tiled**
  - Filters repeat every $n$
  - More filters than convolution for given # features
Non-Linearity

• Rectified linear function
  − Applied per-pixel
  − output = max(0, input)

Input feature map

Output feature map

Black = negative; white = positive values

Only non-negative values
• Other choices:
  – Tanh
  – Sigmoid: $1/(1+\exp(-x))$
  – PReLU

Pooling

• Spatial Pooling
  – Non-overlapping / overlapping regions
  – Sum or max
  – Boureau et al. ICML’10 for theoretical analysis
Pooling

- Pooling across feature groups
  - Additional form of inter-feature competition
  - MaxOut Networks [Goodfellow et al. ICML 2013]
Role of Pooling

- Spatial pooling
  - Invariance to small transformations
  - Larger receptive fields
    (see more of input)

Visualization technique from [Le et al. NIPS’10]:

Videos from: http://ai.stanford.edu/~quocle/TCNNweb

Zeiler, Fergus [arXiv 2013]
Alternative to Pooling

- Replace pooling with strided convolution
  - i.e. filters applied every \( r \) pixels \((r>1)\)
  - [Striving for Simplicity: the all Convolutional Net, Spingenberg et al. ICL 2015]

<table>
<thead>
<tr>
<th>Model</th>
<th>Strided-CNN-C</th>
<th>ConvPool-CNN-C</th>
<th>All-CNN-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 32 × 32 RGB image</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 × 3 conv. 96 ReLU</td>
<td>3 × 3 conv. 96 ReLU</td>
<td>3 × 3 conv. 96 ReLU</td>
<td>3 × 3 conv. 96 ReLU</td>
</tr>
<tr>
<td>3 × 3 conv. 96 ReLU with stride ( r = 2 )</td>
<td>3 × 3 max-pooling stride 2</td>
<td>3 × 3 conv. 96 ReLU with stride ( r = 2 )</td>
<td></td>
</tr>
<tr>
<td>3 × 3 conv. 192 ReLU</td>
<td>3 × 3 conv. 192 ReLU</td>
<td>3 × 3 conv. 192 ReLU</td>
<td>3 × 3 conv. 192 ReLU</td>
</tr>
<tr>
<td>3 × 3 conv. 192 ReLU with stride ( r = 2 )</td>
<td>3 × 3 max-pooling stride 2</td>
<td>3 × 3 conv. 192 ReLU with stride ( r = 2 )</td>
<td></td>
</tr>
</tbody>
</table>

CIFAR-10 classification error

- Model C: 9.74\% \( \approx 1.3 \) M
- Strided-CNN-C: 10.19\% \( \approx 1.3 \) M
- ConvPool-CNN-C: 9.31\% \( \approx 1.4 \) M
- ALL-CNN-C: 9.08\% \( \approx 1.4 \) M
Components of Each Layer

Pixels / Features

Filter with learned dictionary

Non-linearity

Spatial local max pooling

[Optional] Normalization across data/features

Output Features
Normalization

- Lots of different normalization approaches

- Basic idea:
  - Make mean = 0
  - Make standard deviation = 1
  - Question: which dimensions?

---

![Diagram of normalization methods](https://arxiv.org/pdf/1803.08494.pdf)
Normalization across Data

- **Batch Normalization**


**Algorithm 1:** Batch Normalizing Transform, applied to activation $x$ over a mini-batch.

| Input: Values of $x$ over a mini-batch: $B = \{x_1...m\};$  
| Parameters to be learned: $\gamma, \beta$ 
| Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$ 

$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i$ \hspace{1cm} // mini-batch mean

$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2$ \hspace{1cm} // mini-batch variance

$\tilde{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ \hspace{1cm} // normalize

$y_i \leftarrow \gamma \tilde{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i)$ \hspace{1cm} // scale and shift

**Figure 2:** Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.
• But batch normalization has issues, e.g. when batch size is small or 1.
• But batch normalization has issues, e.g. when batch size is small or 1
Normalization

- Instance Norm, Layer Norm, Group Norm

![Graph showing error (%) vs epochs for different normalization methods: Batch Norm (BN), Layer Norm (LN), Instance Norm (IN), and Group Norm (GN). The graph illustrates the performance improvement with different normalization techniques, with Group Norm showing the best performance.](https://arxiv.org/pdf/1803.08494.pdf)
Normalization

- Local contrast normalization across features
  - See Divisive Normalization in Neuroscience
  - Local version of Layer Norm
• Local Contrast normalization (across feature maps)
  – Local mean = 0, local std. = 1, “Local” → 7x7 Gaussian
  – Equalizes the features maps
Image Whitening

- Convariance matrix of 32x32 real-world images
- Compute whitening matrix $W$ via ZCA transform
- Rows of $W$, reshaped to 32x32 images
  - Reveals local dependencies
- Whitened image
ZCA Transform

- Convariance matrix $C = 1/(n-1) \, X \, X^T$
- Want linear transform $W$: $Y = W \, X$
  such that $YY^T = (n-1) \, I$
- [Some math] $W = (1/(n-1) \, (XX^T))^{-1/2}$
- Compute $W$ using SVD

- Note: only applicable to small images
- For large images, use local contrast normalization

How important is Depth

- “Deep” in Deep Learning
- Ablation study
- Tap off features
Architecture of Krizhevsky et al.

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR’09]
- 18.2% top-5 error
- Our reimplementation: 18.1% top-5 error
Architecture of Krizhevsky et al.

- Remove top fully connected layer — Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!
Architecture of Krizhevsky et al.

• Remove both fully connected layers
  – Layer 6 & 7

• Drop ~50 million parameters

• 5.7% drop in performance
• Now try removing upper feature extractor layers:
  – Layers 3 & 4
• Drop ~1 million parameters
• 3.0% drop in performance
Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6, 7

- Now only 4 layers

- 33.5% drop in performance

→ Depth of network is key
Tapping off Features at each Layer

Plug features from each layer into linear SVM or soft-max

<table>
<thead>
<tr>
<th></th>
<th>Cal-101 (30/class)</th>
<th>Cal-256 (60/class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (1)</td>
<td>44.8 ± 0.7</td>
<td>24.6 ± 0.4</td>
</tr>
<tr>
<td>SVM (2)</td>
<td>66.2 ± 0.5</td>
<td>39.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (3)</td>
<td>72.3 ± 0.4</td>
<td>46.0 ± 0.3</td>
</tr>
<tr>
<td>SVM (4)</td>
<td>76.6 ± 0.4</td>
<td>51.3 ± 0.1</td>
</tr>
<tr>
<td>SVM (5)</td>
<td>86.2 ± 0.8</td>
<td>65.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (7)</td>
<td>85.5 ± 0.4</td>
<td>71.7 ± 0.2</td>
</tr>
<tr>
<td>Softmax (5)</td>
<td>82.9 ± 0.4</td>
<td>65.7 ± 0.5</td>
</tr>
<tr>
<td>Softmax (7)</td>
<td>85.4 ± 0.4</td>
<td>72.6 ± 0.1</td>
</tr>
</tbody>
</table>
Translation (Vertical)
Scale Invariance

Layer 1

Layer 7

Output

Lawn Mower
Shih-Tzu
African Crocodile
African Grey
Entertrainment Center

P(true class)
Rotation Invariance

Layer 1

Layer 7

Output
Very Deep Models (1)


- Lots of 3x3 conv layers: more non-linearity than single 7x7 layer
- Close to SOA results on Imagenet: 6.8% top-5 val
- Can be hard to train

<table>
<thead>
<tr>
<th>ConvNet Configuration</th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 weight layers</td>
<td>11 weight layers</td>
<td>13 weight layers</td>
<td>16 weight layers</td>
<td>16 weight layers</td>
<td>19 weight layers</td>
<td></td>
</tr>
<tr>
<td>input (224 × 224 RGB image)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td></td>
</tr>
<tr>
<td>LRN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td></td>
</tr>
<tr>
<td>conv3-128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td></td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td></td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-4096</td>
<td>FC-4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-4096</td>
<td>FC-4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-4096</td>
<td>FC-4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-4096</td>
<td>FC-4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Number of parameters (in millions).

<table>
<thead>
<tr>
<th>Network</th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters</td>
<td>133</td>
<td>133</td>
<td>134</td>
<td>138</td>
<td>144</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: ConvNet performance at a single test scale.

<table>
<thead>
<tr>
<th>ConvNet config. (Table 1)</th>
<th>Smallest image size</th>
<th>Top-1 val. error (%)</th>
<th>Top-5 val. error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>256</td>
<td>256</td>
<td>29.6</td>
</tr>
<tr>
<td>A-LRN</td>
<td>256</td>
<td>256</td>
<td>29.7</td>
</tr>
<tr>
<td>B</td>
<td>256</td>
<td>256</td>
<td>28.7</td>
</tr>
<tr>
<td>C</td>
<td>256</td>
<td>256</td>
<td>28.1</td>
</tr>
<tr>
<td>[256;512]</td>
<td>384</td>
<td>384</td>
<td>27.3</td>
</tr>
<tr>
<td>D</td>
<td>256</td>
<td>256</td>
<td>27.0</td>
</tr>
<tr>
<td>[256;512]</td>
<td>384</td>
<td>384</td>
<td>26.8</td>
</tr>
<tr>
<td>E</td>
<td>256</td>
<td>256</td>
<td>25.6</td>
</tr>
<tr>
<td>[256;512]</td>
<td>384</td>
<td>384</td>
<td>25.0</td>
</tr>
</tbody>
</table>

Published as a conference paper at ICLR 2015
GoogLeNet inception module:

1. Multiple filter scales at each layer

2. Dimensionality reduction to keep computational requirements down
GoogLeNet vs Previous Models

Width of inception modules ranges from 256 filters (in early modules) to 1024 in top inception modules.

Can remove fully connected layers on top completely

Number of parameters is reduced to 5 million

6.7% top-5 validation error on Imagnet

Computational cost is increased by less than 2X compared to Krizhevsky’s network. (<1.5Bn operations/evaluation)
Residual Networks

[He, Zhang, Ren, Sun, CVPR 2016]

Really, really deep convnets don’t train well, E.g. CIFAR10:

Key idea: introduce “pass through” into each layer

Thus only residual now needs to be learned

With ensembling, 3.57% top-5 test error on ImageNet

Table 4. Error rates (%) of single-model results on the ImageNet validation set (except † reported on the test set).
Visualizing Convnets

• Want to know what they are learning

• Raw coefficients of learned filters in higher layers difficult to interpret

• Two classes of method:
  1. Project activations back to pixel space
  2. Optimize input image to maximize a particular feature map or class
Visualizing Convnets

• Projection from higher layers back to input
  – Several similar approaches:
  – Visualizing and Understanding Convolutional Networks, Matt Zeiler & Rob Fergus, ECCV 2014
Projection from Higher Layers

[Zeiler et al. ECCV14]
Details of Operation

Deconvnet layer

Convnet layer
Unpooling Operation
Layer 1 Filters
Visualizations of Higher Layers

- Use ImageNet 2012 validation set
- Push each image through network

- Take max activation from feature map associated with each filter
- Use Deconvnet to project back to pixel space
- Use pooling “switches” peculiar to that activation
Layer 1: Top-9 Patches
Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map
Layer 2: Top-9

- **Not samples from model**
- Just parts of input image that give strong activation of this feature map
- Non-parametric view on invariances learned by model
Receptive Field

- Receptive field of the first layer is the filter size.
- Receptive field (w.r.t. input image) of a deeper layer depends on all previous layers’ filter size and strides.

- **Correspondence** between a feature map pixel and an image pixel is not unique.
- Map a feature map pixel to the **center of the receptive field** on the image in the SPP-net paper.

Layer 3: Top-9
Layer 4: Top-9 Patches
Layer 4: Top-9
Layer 5: Top-9 Patches
Layer 5: Top-9
Visualizing Convnets

• Optimize input to maximize particular output
  – Lots of approaches, e.g. Erhan et al. [Tech Report 2009], Le et al. [NIPS 2010].
  – Depend on initialization

• Google DeepDream
  [http://googleresearch.blogspot.ch/2015/06/inceptionism-going-deeper-into-neural.html]
  – Maximize “banana” output
Google DeepDream

https://photos.google.com/share/F1QipPX0SCl7OzWilt9LnuQliattX4OUcJ_8EP65_c1TvBmS1jnYgsGQAieQUc1VQWdgQ/photo/AF1QipMYTXpt0TvZ0Q5kubkGw8VAq2isxBuL02wKZafB?key=aVBxWjhwSzego2RjWLRuVFBBZEN1d205bUdEMnhB
Training Big ConvNets

• Stochastic Gradient Descent
  – Compute (noisy estimate of) gradient on small batch of data & make step
  – Take as many steps as possible (even if they are noisy)
  – Large initial learning rate
  – Anneal learning rate

• Momentum
  – Variants [Sutskever ICML 2012]
Annealing of Learning Rate

- Start large, slowly reduce
- Explore different scales of energy surface
Evolution of Features During Training
Evolution of Features During Training
Fooling Convnets

• Search for images that are misclassified by the network


• Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images, Anh Nguyen, Jason Yosinski, Jeff Clune, arXiv 1412.1897.

• Problem common to any discriminative method

Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq 99.6\%$ certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects.
Adversarial Examples

Adversarial examples: the formulation

- $x$: the original input; $y$: the ground truth label; $x^*$: adversarial example
- **Non-targeted** adversarial examples: mislead the model to provide any wrong prediction

$$\max_{x^*} \ell(f_\theta(x^*), y)$$
$$\text{s.t. } d(x, x^*) \leq B$$

- **Targeted** adversarial examples: mislead the model to provide the target prediction $y^* \neq y$ specified by the adversary

$$\min_{x^*} \ell(f_\theta(x^*), y^*)$$
$$\text{s.t. } d(x, x^*) \leq B$$

- $d(x, x^*)$ is an $\ell_p$ norm in most existing work
- $B$ is a constant to make sure that $x^*$ is visually similar to $x$

More material:

- Blog: http://karpathy.github.io/2015/03/30/breaking-convnets/
- CVPR 2021 Tutoal: https://advmlincv.github.io/cvpr21-tutorial/
Fast Gradient-Sign Method (FGSM): a one-step attack

- \( d(x, x^*) \) is the \( \ell_\infty \) norm
- \( x^* = x + B \text{sign}(\nabla_x \ell(f_\theta(x), y)) \)
- Simple yet effective attacks against models without defense
- Not effective against models with defense


Adversarial Attacks in Computer Vision: An Overview, Xinyun Chen, CVPR 2021 tutorial
Black-box attacks based on transferability

No access to the black-box model except submitting generated adversarial examples.

Adversarial Attacks in Computer Vision: An Overview, Xinyun Chen, CVPR 2021 tutorial
Non-targeted attacks on ImageNet

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSD</th>
<th>ResNet-152</th>
<th>ResNet-101</th>
<th>ResNet-50</th>
<th>VGG-16</th>
<th>GoogLeNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-152</td>
<td>22.83</td>
<td>0%</td>
<td>13%</td>
<td>18%</td>
<td>19%</td>
<td>11%</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>23.81</td>
<td>19%</td>
<td>0%</td>
<td>21%</td>
<td>21%</td>
<td>12%</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>22.86</td>
<td>23%</td>
<td>20%</td>
<td>0%</td>
<td>21%</td>
<td>18%</td>
</tr>
<tr>
<td>VGG-16</td>
<td>22.51</td>
<td>22%</td>
<td>17%</td>
<td>17%</td>
<td>0%</td>
<td>5%</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>22.58</td>
<td>39%</td>
<td>38%</td>
<td>34%</td>
<td>19%</td>
<td>0%</td>
</tr>
</tbody>
</table>

- RMSD: root mean square deviation $d(x, x^*) = \sqrt{\sum_i (x_i^* - x_i)^2 / M}$, $M$: image size
- All selected original images are predicted correctly by all models by top-1 accuracy.
- >60% adversarial examples are wrongly classified by different models.


Adversarial Attacks in Computer Vision: An Overview, Xinyun Chen, CVPR 2021 tutorial
Transferability of targeted attacks between two models is poor

<table>
<thead>
<tr>
<th>Model</th>
<th>ResNet152</th>
<th>ResNet101</th>
<th>ResNet50</th>
<th>VGG16</th>
<th>GoogLeNet</th>
<th>Incept-v3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet152</td>
<td>100%</td>
<td>2%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>ResNet101</td>
<td>3%</td>
<td>100%</td>
<td>3%</td>
<td>2%</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>ResNet50</td>
<td>4%</td>
<td>2%</td>
<td>100%</td>
<td>1%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>VGG16</td>
<td>2%</td>
<td>1%</td>
<td>2%</td>
<td>100%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>1%</td>
<td>1%</td>
<td>0%</td>
<td>1%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Incept-v3</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>100%</td>
</tr>
</tbody>
</table>

<5% adversarial examples are predicted with the same label by two models.

Ground truth: running shoe

<table>
<thead>
<tr>
<th>VGG16</th>
<th>Military uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet50</td>
<td>Jigsaw puzzle</td>
</tr>
<tr>
<td>ResNet101</td>
<td>Motor scooter</td>
</tr>
<tr>
<td>ResNet152</td>
<td>Mask</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>Chainsaw</td>
</tr>
</tbody>
</table>
Universal Adversarial Examples

- Moosave-Dezfooli et al.  
  arXiv 1610.08401, Oct 2016
Our approach: attacking an ensemble of models

Intuition: If an adversarial example can fool N-1 white-box models, it might transfer better to the N-th black-box model.


Adversarial Attacks in Computer Vision: An Overview, Xinyun Chen, CVPR 2021 tutorial
Non-targeted attacks with ensemble

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSD</th>
<th>ResNet-152</th>
<th>ResNet-101</th>
<th>ResNet-50</th>
<th>VGG-16</th>
<th>GoogLeNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ResNet-152</td>
<td>17.17</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>-ResNet-101</td>
<td>17.25</td>
<td>0%</td>
<td>1%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>-ResNet-50</td>
<td>17.25</td>
<td>0%</td>
<td>0%</td>
<td>2%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>-VGG-16</td>
<td>17.80</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>6%</td>
<td>0%</td>
</tr>
<tr>
<td>-GoogLeNet</td>
<td>17.41</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>5%</td>
</tr>
</tbody>
</table>

- Model: the model architecture is not included in the white-box ensemble.

- Ensemble further decreases the accuracy on adversarial examples, and decreases the perturbation magnitude.
Invisibility Cloak

- https://www.cs.umd.edu/~tomg/projects/invisible/
- Adversarial attack on YOLO v2 person detector
References

• P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object Detection with Discriminatively Trained Part Based Models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, September 2010

• Zheng Song*, Qiang Chen*, Zhongyang Huang, Yang Hua, and Shuicheng Yan. Contextualizing Object Detection and Classification. In CVPR'11. (* indicates equal contribution) [No. 1 performance in VOC'10 classification task]


• Gehler and Nowozin, On Feature Combination for Multiclass Object Classification, ICCV'09

• Yoshua Bengio and Yann LeCun: Scaling learning algorithms towards AI, in Bottou, L. and Chapelle, O. and DeCoste, D. and Weston, J. (Eds), Large-Scale Kernel Machines, MIT Press, 2007
References

• S. Lazebnik, C. Schmid, and J. Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories, CVPR 2006
• Christoph H. Lampert, Hannes Nickisch, Stefan Harmeling: "Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer", IEEE Computer Vision and Pattern Recognition (CVPR), Miami, FL, 2009
• http://www.scholarpedia.org/article/Neocognitron
References

• Q.V. Le, J. Ngiam, Z. Chen, D. Chia, P. Koh, A.Y. Ng, Tiled Convolutional Neural Networks. NIPS, 2010
• http://ai.stanford.edu/~quocle/TCNNweb
• Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus, Adaptive Deconvolutional Networks for Mid and High Level Feature Learning, International Conference on Computer Vision (November 6-13, 2011)
References

References

• P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object Detection with Discriminatively Trained Part Based Models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, September 2010
• Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus, Adaptive Deconvolutional Networks for Mid and High Level Feature Learning, International Conference on Computer Vision (November 6-13, 2011)
References

• R. Girshick, P. Felzenszwalb, D. McAllester, Object Detection with Grammar Models, NIPS 2011
• P. Felzenszwalb, D. Huttenlocher, Pictorial Structures for Object Recognition, International Journal of Computer Vision, Vol. 61, No. 1, January 2005
• M. Fischler and R. Elschlager. The Representation and Matching of Pictorial Structures. (1973)
References

- Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus, Adaptive Deconvolutional Networks for Mid and High Level Feature Learning, International Conference on Computer Vision(November 6-13, 2011)
References

- Q.V. Le, W.Y. Zou, S.Y. Yeung, A.Y. Ng., Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis, CVPR 2011