Convolutional Neural Networks

Computer Vision
James Hays
Outline

• Neural Networks (covered in previous lecture)
• Convolutional Neural Networks
• Visualization and interpretation of Deep Networks
Key Idea: Wiggle To Decrease Loss

Let's say we want to decrease the loss by adjusting $W_{i,j}^1$. We could consider a very small $\epsilon = 1e-6$ and compute:

$$L(x, y; \theta)$$

$$L(x, y; \theta \setminus W_{i,j}^1, W_{i,j}^1 + \epsilon)$$

Then, update:

$$W_{i,j}^1 \leftarrow W_{i,j}^1 + \epsilon \text{sgn}(L(x, y; \theta) - L(x, y; \theta \setminus W_{i,j}^1, W_{i,j}^1 + \epsilon))$$
Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips
Fully Connected Layer

Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..
Locally Connected Layer

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).
Locally Connected Layer

**STATIONARITY?** Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

**Note:** This parameterization is good when input image is registered (e.g., face recognition).
Convolutional Layer

Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer

\[ \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \times \text{image} = \text{filtered image} \]
Convolutional Layer

Learn multiple filters.

E.g.: 200x200 image
    100 Filters
    Filter size: 10x10
    10K parameters
Convolutional Layer

\[ h_j^n = \max(0, \sum_{k=1}^{K} h_{k}^{n-1} \ast w_{kj}^n) \]

output feature map
input feature map
kernel

Ranzato
Convolutional Layer

\[ h^n_j = \max(0, \sum_{k=1}^{K} h^{n-1}_k \ast w^n_{kj}) \]

output feature map  input feature map  kernel
Convolutional Layer

\[ h^n_j = \max(0, \sum_{k=1}^{K} h^{n-1}_k \ast w^n_{kj}) \]

- output feature map
- input feature map
- kernel
**Convolutional Layer**

**Question:** What is the size of the output? What's the computational cost?

**Answer:** It is proportional to the number of filters and depends on the stride. If kernels have size KxK, input has size DxD, stride is 1, and there are M input feature maps and N output feature maps then:
- the input has size M@DxD
- the output has size N@(D-K+1)x(D-K+1)
- the kernels have MxNxKxK coefficients (which have to be learned)
- cost: M*K*K*N*(D-K+1)*(D-K+1)

**Question:** How many feature maps? What's the size of the filters?

**Answer:** Usually, there are more output feature maps than input feature maps. Convolutional layers can increase the number of hidden units by big factors (and are expensive to compute). The size of the filters has to match the size/scale of the patterns we want to detect (task dependent).
Key Ideas

A standard neural net applied to images:
- scales quadratically with the size of the input
- does not leverage stationarity

Solution:
- connect each hidden unit to a small patch of the input
- share the weight across space

This is called: convolutional layer.
A network with convolutional layers is called convolutional network.

LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998
Pooling Layer

Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?
Pooling Layer

By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.
Pooling Layer: Examples

Max-pooling:

\[ h_j^n(x, y) = \max_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y}) \]

Average-pooling:

\[ h_j^n(x, y) = 1/K \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y}) \]

L2-pooling:

\[ h_j^n(x, y) = \sqrt{\sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})^2} \]

L2-pooling over features:

\[ h_j^n(x, y) = \sqrt{\sum_{k \in N(j)} h_k^{n-1}(x, y)^2} \]
Pooling Layer

**Question:** What is the size of the output? What's the computational cost?

**Answer:** The size of the output depends on the stride between the pools. For instance, if pools do not overlap and have size KxK, and the input has size DxD with M input feature maps, then:
- output is M@(D/K)x(D/K)
- the computational cost is proportional to the size of the input (negligible compared to a convolutional layer)

**Question:** How should I set the size of the pools?

**Answer:** It depends on how much “invariant” or robust to distortions we want the representation to be. It is best to pool slowly (via a few stacks of conv-pooling layers).
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: 
\((P+K-1) \times (P+K-1)\)
If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: $(P+K-1) \times (P+K-1)$.
Local Contrast Normalization

$$h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))}$$
Local Contrast Normalization

\[ h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))} \]

We want the same response.
Local Contrast Normalization

\[ h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))} \]

Performed also across features and in the higher layers.

Effects:
- improves invariance
- improves optimization
- increases sparsity

**Note:** Computational cost is negligible w.r.t. conv. layer.
ConvNets: Typical Stage

One stage (zoom)

Convol. → LCN → Pooling

Conceptually similar to: SIFT, HoG, etc.
ConvNets: Typical Architecture

One stage (zoom)

Whole system

Input Image

1st stage  2nd stage  3rd stage

Fully Conn. Layers

Class Labels
ConvNets: Typical Architecture

Whole system

Input Image → 1st stage → 2nd stage → 3rd stage → Fully Conn. Layers → Class Labels

Conceptually similar to:

SIFT → K-Means → Pyramid Pooling → SVM
Lazebnik et al. “...Spatial Pyramid Matching...” CVPR 2006

SIFT → Fisher Vect. → Pooling → SVM
Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips
CONV NETS: EXAMPLES

- OCR / House number & Traffic sign classification

Ciresan et al. “MCDNN for image classification” CVPR 2012
Jaderberg et al. “Synthetic data and ANN for natural scene text recognition” arXiv 2014
CONV NETS: EXAMPLES

- Texture classification

Sifre et al. “Rotation, scaling and deformation invariant scattering...” CVPR 2013
CONV NETS: EXAMPLES

- Pedestrian detection

Sermanet et al. “Pedestrian detection with unsupervised multi-stage..” CVPR 2013
CONV NETS: EXAMPLES

- Scene Parsing

Farabet et al. “Learning hierarchical features for scene labeling” PAMI 2013
Pinheiro et al. “Recurrent CNN for scene parsing” arxiv 2013
CONV NETS: EXAMPLES

- Segmentation 3D volumetric images

Ciresan et al. “DNN segment neuronal membranes...” NIPS 2012
Turaga et al. “Maximin learning of image segmentation” NIPS 2009
CONV NETS: EXAMPLES

- Action recognition from videos

Taylor et al. “Convolutional learning of spatio-temporal features” ECCV 2010
Karpathy et al. “Large-scale video classification with CNNs” CVPR 2014
CONV NETS: EXAMPLES

- Denoising

original  noised  denoised

Burger et al. “Can plain NNs compete with BM3D?” CVPR 2012
Dataset: ImageNet 2012

Deng et al. “Imagenet: a large scale hierarchical image database” CVPR 2009
ImageNet

Examples of hammer:
Architecture for Classification

- Linear
- Fully Connected
- Fully Connected
- Max Pooling
- Conv
- Conv
- Conv
- Max Pooling
- Local Contrast Norm
- Conv
- Max Pooling
- Local Contrast Norm
- Conv

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Architecture for Classification

Total nr. params: 60M

category prediction

Total nr. flops: 832M

4M
LINEAR

16M
FULLY CONNECTED

442K
MAX POOLING

74M
CONV

1.3M
CONV

224M
CONV

884K
CONV

149M
CONV

307K
MAX POOLING

223M
LOCAL CONTRAST NORM

35K
LOCAL CONTRAST NORM

105M
CONV

input

The first convolutional layer filters the $224 \times 224 \times 3$ input image with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring
Optimization

SGD with momentum:
- Learning rate = 0.01
- Momentum = 0.9

Improving generalization by:
- Weight sharing (convolution)
- Input distortions
- Dropout = 0.5
- Weight decay = 0.0005
Results: ILSVRC 2012

**TASK 1 - CLASSIFICATION**

- CNN: 15%
- SIFT+FV: 25%
- SVM1: 30%
- SVM2: 32%
- NCM: 34%

**TASK 2 - DETECTION**

- CNN: 30%
- DPM-SVM1: 50%
- DPM-SVM2: 52%

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
<table>
<thead>
<tr>
<th>mite</th>
<th>container ship</th>
<th>motor scooter</th>
<th>leopard</th>
</tr>
</thead>
<tbody>
<tr>
<td>mite</td>
<td>container ship</td>
<td>motor scooter</td>
<td>leopard</td>
</tr>
<tr>
<td>black widow</td>
<td>lifeboat</td>
<td>go-kart</td>
<td>jaguar</td>
</tr>
<tr>
<td>cockroach</td>
<td>amphibian</td>
<td>moped</td>
<td>cheetah</td>
</tr>
<tr>
<td>tick</td>
<td>fireboat</td>
<td>bumper car</td>
<td>cheetah</td>
</tr>
<tr>
<td>starfish</td>
<td>drilling platform</td>
<td>golfcart</td>
<td>snow leopard</td>
</tr>
<tr>
<td>grille</td>
<td>mushroom</td>
<td>cherry</td>
<td>Madagascar cat</td>
</tr>
<tr>
<td>convertible</td>
<td>agaric</td>
<td>dalmatian</td>
<td>squirrel monkey</td>
</tr>
<tr>
<td>grille</td>
<td>mushroom</td>
<td>grape</td>
<td>spider monkey</td>
</tr>
<tr>
<td>pickup</td>
<td>jelly fungus</td>
<td>elderberry</td>
<td>titi</td>
</tr>
<tr>
<td>beach wagon</td>
<td>gill fungus</td>
<td>fforde</td>
<td>indri</td>
</tr>
<tr>
<td>fire engine</td>
<td>dead-man's-fingers</td>
<td>affordshire bullterrier</td>
<td>howler monkey</td>
</tr>
</tbody>
</table>
Object Detectors Emerge in Deep Scene CNNs

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba

Massachusetts Institute of Technology
CNN for Object Recognition

Large-scale image classification result on ImageNet

Figure from Olga Russakovsky ECCV’14 workshop
How Objects are Represented in CNN?

DrawCNN: visualizing the units' connections
How Objects are Represented in CNN?

Deconvolution


Strong activation image


Back-propagation

Object Representations in Computer Vision

Part-based models are used to represent objects and visual patterns.

- Object as a set of parts
- Relative locations between parts

Figure from Fischler & Elschlager (1973)
Object Representations in Computer Vision

**Constellation model**
Weber, Welling & Perona (2000),

**Deformable Part model**

**Bag-of-word model**

**Class-specific graph model**
Kumar, Torr and Zisserman (2005), Felzenszwalb & Huttenlocher (2005)
Learning to Recognize Objects

Possible internal representations:
- Object parts
- Textures
- Attributes
How Objects are Represented in CNN?

CNN uses **distributed code** to represent objects.

Scene Recognition

Given an image, predict which place we are in.

Bedroom

Harbor
Learning to Recognize Scenes

Possible internal representations:

- Objects (scene parts?)
- Scene attributes
- Object parts
- Textures
CNN for Scene Recognition

**Places Database:** 7 million images from 400 scene categories

**Places-CNN:** AlexNet CNN on 2.5 million images from 205 scene categories.

<table>
<thead>
<tr>
<th></th>
<th>Places 205</th>
<th>SUN 205</th>
</tr>
</thead>
<tbody>
<tr>
<td>Places-CNN</td>
<td>50.0%</td>
<td>66.2%</td>
</tr>
<tr>
<td>ImageNet CNN feature+SVM</td>
<td>40.8%</td>
<td>49.6%</td>
</tr>
</tbody>
</table>

**Scene Recognition Demo:** 78% top-5 recognition accuracy in the wild

ImageNet CNN and Places CNN

ImageNet CNN for Object Classification

Places CNN for Scene Classification

Same architecture: AlexNet
Neuroscientists study brain

Data-Driven Approach to Study CNN

200,000 image stimuli of objects and scene categories (ImageNet TestSet+SUN database)
Estimating the Receptive Fields

Estimated receptive fields

- pool1
- conv3
- pool5

Actual size of RF is much smaller than the theoretic size

Segmentation using the RF of Units

- pool1
- pool2
- conv4
- pool5

More semantically meaningful
Annotating the Semantics of Units

Top ranked segmented images are cropped and sent to Amazon Turk for annotation.
Annotating the Semantics of Units

Pool5, unit 76; Label: ocean; Type: scene; Precision: 93%
Annotating the Semantics of Units

Pool5, unit 13; Label: Lamps; Type: object; Precision: 84%
Annotating the Semantics of Units

Pool5, unit 77; Label: legs; Type: object part; Precision: 96%
Annotating the Semantics of Units

Pool5, unit 112; Label: pool table; Type: object; Precision: 70%
Annotating the Semantics of Units

Pool5, unit 22; Label: dinner table; Type: scene; Precision: 60%
Object detectors emerge within CNN trained to classify scenes, without any object supervision!
Histogram of Emerged Objects in Pool5

ImageNet-CNN (59/256)
Histogram of Emerged Objects in Pool5

Places-CNN (151/256)
Evaluation on SUN Database

Evaluate the performance of the emerged object detectors
Evaluation on SUN Database

Object counts in SUN

Counts of CNN units discovering each object class.

Object counts of most informative objects for scene recognition

Correlation: 0.53

Correlation: 0.84
Conclusion

We show that object detectors emerge inside a CNN trained to classify scenes, without any object supervision.

Object detectors for free!

Places database, Places CNN, and unit annotations could be downloaded at

http://places.csail.mit.edu