Lecture 32: Applications of CNNs
Menu for Today (December 1, 2021)

Topics:

— Image classification with CNNs
— Segmentation with CNNs
— Object detection with CNNs

Readings:

— Today’s Lecture:  N/A
— Next Lecture:  N/A

Reminders:

— Assignment 6: Deep Learning due Dec 6
— Quiz 6 is Dec 7
— Practice Quiz 6 available Thu Dec 2 to Mon Dec 6
Today’s “fun” Example: Adversarial Examples for CNNs

Bus → Chicken → Building → Soap dispenser → Praying mantis → Dog

[ Szegedy et. al., 2013 ]
Today’s “fun” Example: Adversarial Examples for CNNs

[ Szegedy et. al., 2013 ]
Today’s “fun” Example: Adversarial Examples for CNNs

[ Papernot et. al. ]
Today’s “fun” Example: Adversarial Examples for CNNs
Today’s “fun” Example: Adversarial Examples for CNNs
Convolutional Neural Networks

VGG-16 Network
Convolutional Layer: Closer Look at **Spatial Dimensions**

- **32 x 32 x 3 image**
- **5 x 5 x 3 filter** $(W)$
- **32 width**
- **3 depth**

Convolve (slide) over all spatial locations to get the activation map.

- **activation map**
  - **28 width**
  - **1 depth**
  - **28 height**

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n* Stanford
Convolutional Layer: **1x1 convolutions**

56 x 56 x 64 **image**

56 height

56 width

64 depth

---

32 **filters** of size, 1 x 1 x 64

56 x 56 x 32 **image**

56 height

56 width

32 depth

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**
**Convolutional Neural Network (ConvNet)**

- **32 height**
  - **CONV, ReLU**
  - e.g. 6 5x5x3 filters

- **28 height**
  - **CONV, ReLU**
  - e.g. 10 5x5x6 filters

- **24 height**
  - **CONV, ReLU**

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer Summary

Accepts a volume of size: $W_i \times H_i \times D_i$
Convolutional Layer Summary

Accepts a volume of size: $W_i \times H_i \times D_i$  (for mini-batch $N \times W_i \times H_i \times D_i$)
Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$  
(for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

— Number of filters: $K$  
(for typical networks $K \in \{32, 64, 128, 256, 512\}$)

— Spatial extent of filters: $F$  
(for a typical networks $F \in \{1, 3, 5, \ldots\}$)

— Stride of application: $S$  
(for a typical network $S \in \{1, 2\}$)

— Zero padding: $P$  
(for a typical network $P \in \{0, 1, 2\}$)
Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$  (for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

- Number of filters: $K$  (for typical networks $K \in \{32, 64, 128, 256, 512\}$

- Spatial extent of filters: $F$  (for a typical networks $F \in \{1, 3, 5, \ldots\}$

- Stride of application: $S$  (for a typical network $S \in \{1, 2\}$

- Zero padding: $P$  (for a typical network $P \in \{0, 1, 2\}$

Produces a volume of size: $W_o \times H_o \times D_o$  (for mini-batch $N \times W_o \times H_o \times D_o$)
Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$  
(for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

— Number of filters: $K$  
(for typical networks $K \in \{32, 64, 128, 256, 512\}$)

— Spatial extent of filters: $F$  
(for a typical networks $F \in \{1, 3, 5, \ldots\}$)

— Stride of application: $S$  
(for a typical network $S \in \{1, 2\}$)

— Zero padding: $P$  
(for a typical network $P \in \{0, 1, 2\}$)

Produces a volume of size: $W_o \times H_o \times D_o$  
(for mini-batch $N \times W_o \times H_o \times D_o$)

\[
W_o = (W_i - F + 2P)/S + 1 \quad H_o = (H_i - F + 2P)/S + 1 \quad D_o = K
\]
Convolutional Layer Summary

Accepts a volume of size: \( W_i \times H_i \times D_i \)  
(for mini-batch \( N \times W_i \times H_i \times D_i \))

Requires hyperparameters:

- Number of filters: \( K \)  
  (for typical networks \( K \in \{32, 64, 128, 256, 512\} \))
- Spatial extent of filters: \( F \)  
  (for a typical networks \( F \in \{1, 3, 5, \ldots\} \))
- Stride of application: \( S \)  
  (for a typical network \( S \in \{1, 2\} \))
- Zero padding: \( P \)  
  (for a typical network \( P \in \{0, 1, 2\} \))

Produces a volume of size: \( W_o \times H_o \times D_o \)  
(for mini-batch \( N \times W_o \times H_o \times D_o \))

\[
W_o = (W_i - F + 2P)/S + 1 \quad H_o = (H_i - F + 2P)/S + 1 \quad D_o = K
\]

Number of total learnable parameters: \((F \times F \times D_i) \times K + K\)
Convolutional Neural Networks

VGG-16 Network
**CNNs**: Reminder Fully Connected Layers

- **Input**: 3072 (32 x 32 x 3 image -> stretches to 3072 x 1)
- **Activation**: $W^T x + b$, where $W \in \mathbb{R}^{10 \times 3072}$
- Each neuron looks at the full input volume

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**
Convolutional Neural Networks

VGG-16 Network
Convolutional Neural Networks

VGG-16 Network
Pooling Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently

How many parameters? None!

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Max Pooling

activation map

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

max pool with 2 x 2 filter and stride of 2

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Average **Pooling**

activation map

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

avg pool with 2 x 2 filter and stride of 2

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.25</td>
<td>5.25</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
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: $(P+K-1) \times (P+K-1)$
Pooling Layer **Receptive Field**

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: $(P+K-1) \times (P+K-1)$

* slide from Marc’Aurelio Renzato
Pooling Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$  
(for mini-batch $N \times W_i \times H_i \times D_i$)

Requires hyperparameters:

— Spatial extent of filters: $F$

— Stride of application: $S$

Produces a volume of size: $W_o \times H_o \times D_o$

$$W_o = (W_i - F)/S + 1 \quad H_o = (H_i - F)/S + 1 \quad D_o = D_i$$

Number of total learnable parameters: 0  
(for mini-batch $N \times W_o \times H_o \times D_o$)
Convolutional Neural Networks

VGG-16 Network
Computer Vision Problems
Computer Vision Problems

Categorization

Single-label: Horse
Church
Toothbrush
Person

Multi-label: Horse
Church
Toothbrush
Person
Computer Vision Problems

Categorization

Single-label: Horse
Church
Toothbrush
Person

Multi-label: Horse
Church
Toothbrush
Person

Detection

Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)
Computer Vision Problems

Categorization

Single-label: Horse
Church
Toothbrush
Person

Detection

Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

Segmentation

Multi-label: Horse
Church
Toothbrush
Person

COCO Common Objects in Context
Computer Vision Problems

Categorization

Single-label: Horse Church Toothbrush Person

Detection

Horse (x, y, w, h) Horse (x, y, w, h) Person (x, y, w, h) Person (x, y, w, h)

Segmentation

Horse Person

Instance Segmentation

Horse1 Horse2 Person1 Person2

Multi-label: Horse Church Toothbrush Person
Computer Vision Problems

Categorization

Single-label: Horse
              Church
              Toothbrush
              Person

Multi-label: Horse
            Church
            Toothbrush
            Person
Object **Classification**

**Problem:** For each image predict which category it belongs to out of a fixed set

<table>
<thead>
<tr>
<th>Category</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>No</td>
</tr>
<tr>
<td>Cat</td>
<td>No</td>
</tr>
<tr>
<td>Couch</td>
<td>No</td>
</tr>
<tr>
<td>Flowers</td>
<td>No</td>
</tr>
<tr>
<td>Leopard</td>
<td>Yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Object Classification

Problem: For each image predict which category it belongs to out of a fixed set
Object **Classification**

Problem: For each image predict which category it belongs to out of a fixed set
Object Classification

- ILSVRC'15 ResNet: 3.57
- ILSVRC'14 GoogleNet: 6.7 layers
- ILSVRC'14 VGG: 7.3 layers
- ILSVRC'13: 8 layers
- ILSVRC'12 AlexNet: 11.7 layers
- ILSVRC'11: 16.4 layers
- ILSVRC'10: 25.8 layers
- Shallow: 28.2 layers

152 layers
Comparing Complexity


* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Computer Vision Problems

Categorization
- Single-label: Horse, Church, Toothbrush, Person
- Multi-label: Horse, Church, Toothbrush, Person

Detection
- Horse \((x, y, w, h)\)
- Horse \((x, y, w, h)\)
- Person \((x, y, w, h)\)
- Person \((x, y, w, h)\)

Segmentation
- Horse
- Person

Instance Segmentation
- Horse1
- Horse2
- Person1
- Person2
Computer **Vision Problems** (no language for now)

Segmentation

![Image with labels: Horse, Person]
Semantic **Segmentation**

Label *every pixel* with a category label (without differentiating instances)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Semantic **Segmentation**: Sliding Window

**Problem:** VERY inefficient, no reuse of computations for overlapping patches

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n Stanford*
Semantic **Segmentation**: Fully Convolutional CNNs

Design a network as a number of convolutional layers to make predictions for all pixels at once!

**Problem**: Convolutions at the original image scale will be very expensive

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Semantic **Segmentation**: Fully Convolutional CNNs

Design a network as a number of convolutional layers with **downsampling** and **upsampling** inside the network!

**Input Image**  
$3 \times H \times W$

**High-res:**  
$D_1 \times H/2 \times W/2$

**Low-res:**  
$D_3 \times H/4 \times W/4$

**Med-res:**  
$D_2 \times H/4 \times W/4$

**Predicted Labels**  
$H \times W$

**Downsampling** = Pooling

**Upsampling** = ???

[ Long et al, CVPR 2015 ]
[ Noh et al, ICCV 2015 ]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
In-network **Up Sampling** (a.k.a “Unpooling”)

Nearest Neighbor

<table>
<thead>
<tr>
<th>1 2</th>
<th>1 1</th>
<th>2 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 4</td>
<td>3 3</td>
<td>4 4</td>
</tr>
</tbody>
</table>

Input: 2 x 2  
Output: 4 x 4

“Bed of Nails”

<table>
<thead>
<tr>
<th>1 2</th>
<th>1 0</th>
<th>2 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 4</td>
<td>0 0</td>
<td>0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 4</th>
<th>3 0</th>
<th>4 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 0</td>
<td></td>
</tr>
</tbody>
</table>

Input: 2 x 2  
Output: 4 x 4

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
In-network **Up Sampling:** Max Unpooling

**Max Pooling**
Remember which element was max!

**Max Unpooling**
Use positions from pooling layer

![Diagram showing max pooling and unpooling](image)

**Input:** 4 x 4

**Output:** 2 x 2

**Input:** 2 x 2

**Output:** 4 x 4

* Corresponding pairs of downsampling and upsampling layers

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Computer Vision Problems

Categorization

Detection

Segmentation

Multi-class:
- Horse
- Church
- Toothbrush
- Person

Multi-label:
- Horse
- Church
- Toothbrush
- Person

IMAGENET

Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

Horse
Person

Horse1
Horse2
Person1
Person2

Common Objects in Context
Computer **Vision Problems** (no language for now)

Detection

Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)
Object **Detection** as Regression Problem

**Problem:** each image needs a different number of outputs

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Object **Detection** as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background.

<table>
<thead>
<tr>
<th>Category</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>No</td>
</tr>
<tr>
<td>Cat</td>
<td>No</td>
</tr>
<tr>
<td>Couch</td>
<td>No</td>
</tr>
<tr>
<td>Flowers</td>
<td>No</td>
</tr>
<tr>
<td>Background</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Object **Detection** as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n Stanford*
Object **Detection** as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Object Detection as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background.

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Object **Detection** as Classification Problem

**Problem:** Need to apply CNN to **many** patches in each image

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

*slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n Stanford*
**Region Proposals** (older idea in vision)

Find image **regions that are likely to contain objects** (any object at all)
- typically works by looking at histogram distributions, region aspect ratio, closed contours, coherent color

Relatively **fast to run** (Selective Search gives 1000 region proposals in a few seconds on a CPU)

---

**Goal:** Get “true” object regions to be in as few top K proposals as possible

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**
R-CNN

[ Girshick et al, CVPR 2014 ]

* image from Ross Girshick
R-CNN

Regions of Interest from a proposal method (~2k)

Input Image

* image from Ross Girshick
R-CNN

* image from Ross Girshick

[ Girshick et al, CVPR 2014 ]
R-CNN

Forward each region through a **CNN**

**Warped** image regions

**Regions of Interest** from a proposal method (~2k)

Input **Image**

* image from Ross Girshick
R-CNN

Classify regions with SVM

Forward each region through a **CNN**

Warped image regions

Regions of Interest from a proposal method (~2k)

Input Image

* image from Ross Girshick
R-CNN

Linear Regression for bounding box offsets

Classify regions with SVM

Forward each region through a CNN

Warped image regions

Regions of Interest from a proposal method (~2k)

Input Image

* image from Ross Girshick
R-CNN (Regions with CNN features) algorithm:

- Extract promising candidate regions using an object proposals algorithm
- Resize each proposal window to the size of the input layer of a trained convolutional neural network
- Input each resized image patch to the convolutional neural network

Implementation detail: Instead of using the classification scores of the network directly, the output of the final fully-connected layer can be used as an input feature to a trained support vector machine (SVM)
Fast R-CNN

Input Image

[ Girshick et al, ICCV 2015 ]

* image from Ross Girshick
Fast **R-CNN**

[ Girshick et al, ICCV 2015 ]

* image from Ross Girshick
Fast **R-CNN**

[ Girshick et al, ICCV 2015 ]

**Input Image**

Forward prop the **whole image** through CNN

“conv5” feature map

* image from Ross Girshick
Fast R-CNN

Regions of Interest from the proposal method

“conv5” feature map

Forward prop the whole image through CNN

Input Image

* image from Ross Girshick
Fast R-CNN

Regions of Interest from the proposal method

ConvNet

Forward prop the whole image through CNN

“RoI Pooling” layer
“conv5” feature map

Input Image

* image from Ross Girshick
Fast R-CNN

Input Image

Forward prop the whole image through CNN

Regions of Interest from the proposal method

“conv5” feature map

“RoI Pooling” layer

Bounding box regression

Object classification

Log loss + Smooth L1 loss

Multi-task loss

Linear + softmax

Linear

FCs

[ Girshick et al, ICCV 2015 ]

* image from Ross Girshick
Fast **R-CNN**: Training

- **Object classification**
  - Log loss + Smooth L1 loss
  - Multi-task loss
  - Linear + softmax
  - Linear
  - FCs
  - Bounding box regression
  - “RoI Pooling” layer
  - “conv5” feature map
  - Forward prop the **whole image** through CNN

- **Regions of Interest** from the proposal method

**Input Image**

---

* image from Ross Girshick

[ Girshick et al, ICCV 2015 ]
R-CNN vs. SPP vs. Fast R-CNN

Observation: Performance dominated by the region proposals at this point!

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Neural Image Captioning

* slide from Dhruv Batra
Neural Image Captioning

Image Embedding (VGGNet)
Neural Image Captioning

Image Embedding (VGGNet)

* slide from Dhruv Batra
Neural Image Captioning

Linear

RNN

RNN

RNN

RNN

RNN

P(next)
P(next)
P(next)
P(next)
P(next)
P(next)

<start> Two people and two horses.

Image Embedding (VGGNet)

* slide from Dhruv Batra
Neural Image Captioning

Good results

- A cat sitting on a suitcase on the floor
- A cat is sitting on a tree branch
- A dog is running in the grass with a frisbee
- A white teddy bear sitting in the grass
- Two people walking on the beach with surfboards
- A tennis player in action on the court
- Two giraffes standing in a grassy field
- A man riding a dirt bike on a dirt track

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Neural Image Captioning

Failure cases

A woman is holding a cat in her hand

A person holding a computer mouse on a desk

A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Summary
Common types of layers:

1. **Convolutional** Layer
   — Parameters define a set of learnable filters

2. **Pooling** Layer
   — Performs a downsampling along the spatial dimensions

3. **Fully-Connected** Layer
   — As in a regular neural network

Each layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function.
Summary

The parameters of a neural network are learned using backpropagation, which computes gradients via recursive application of the chain rule.

A convolutional neural network assumes inputs are images, and constrains the network architecture to reduce the number of parameters.

A convolutional layer applies a set of learnable filters.

A pooling layer performs spatial downsampling.

A fully-connected layer is the same as in a regular neural network.

Convolutional neural networks can be seen as learning a hierarchy of filters.