Lecture 14:
Detection and Segmentation
Administrative

- A2 grades and Quiz 2 grades will be out this week
Transfer learning
“You need a lot of a data if you want to train/use CNNs”
“You need a lot of data if you want to train/use CNNs” BUSTED
Transfer Learning with CNNs
Transfer Learning with CNNs

AlexNet:
64 x 3 x 11 x 11

(More on this in Lecture 13)
Transfer Learning with CNNs

Test image  L2 Nearest neighbors in feature space

(More on this in Lecture 13)
Transfer Learning with CNNs

1. Train on Imagenet

- FC-1000
- FC-4096
- Conv-512
- Conv-512
- MaxPool
- Conv-512
- Conv-512
- MaxPool
- Conv-256
- Conv-256
- MaxPool
- Conv-128
- Conv-128
- MaxPool
- Conv-64
- Conv-64

Razavian et al., “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014
Transfer Learning with CNNs

1. Train on Imagenet
   - Conv-64
   - Conv-64
   - MaxPool
   - Conv-128
   - Conv-128
   - MaxPool
   - Conv-256
   - Conv-256
   - MaxPool
   - Conv-512
   - Conv-512
   - MaxPool
   - Conv-512
   - Conv-512
   - MaxPool
   - FC-4096
   - FC-4096
   - FC-1000

2. Small Dataset (C classes)
   - Conv-64
   - Conv-64
   - MaxPool
   - Conv-128
   - Conv-128
   - MaxPool
   - Conv-256
   - Conv-256
   - MaxPool
   - Conv-512
   - Conv-512
   - MaxPool
   - Conv-512
   - Conv-512
   - MaxPool
   - FC-4096
   - FC-4096

   - FC-C
   - Conv-128
   - Conv-128
   - MaxPool
   - Conv-256
   - Conv-256
   - MaxPool
   - Conv-512
   - Conv-512
   - MaxPool
   - FC-4096
   - FC-4096

   - Reinitialize this and train
   - Freeze these

Razavian et al., "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014
Transfer Learning with CNNs

1. Train on Imagenet

2. Small Dataset (C classes)

Reinitialize this and train

Freeze these


Transfer Learning with CNNs

1. Train on Imagenet

<table>
<thead>
<tr>
<th>FC-1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC-4096</td>
</tr>
<tr>
<td>FC-4096</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-512</td>
</tr>
<tr>
<td>Conv-512</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-256</td>
</tr>
<tr>
<td>Conv-256</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-128</td>
</tr>
<tr>
<td>Conv-128</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-64</td>
</tr>
<tr>
<td>Conv-64</td>
</tr>
<tr>
<td>Image</td>
</tr>
</tbody>
</table>

2. Small Dataset (C classes)

<table>
<thead>
<tr>
<th>FC-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC-4096</td>
</tr>
<tr>
<td>FC-4096</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-512</td>
</tr>
<tr>
<td>Conv-512</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-256</td>
</tr>
<tr>
<td>Conv-256</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-128</td>
</tr>
<tr>
<td>Conv-128</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-64</td>
</tr>
<tr>
<td>Conv-64</td>
</tr>
<tr>
<td>Image</td>
</tr>
</tbody>
</table>

3. Bigger dataset

<table>
<thead>
<tr>
<th>FC-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC-4096</td>
</tr>
<tr>
<td>FC-4096</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-512</td>
</tr>
<tr>
<td>Conv-512</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-256</td>
</tr>
<tr>
<td>Conv-256</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-128</td>
</tr>
<tr>
<td>Conv-128</td>
</tr>
<tr>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-64</td>
</tr>
<tr>
<td>Conv-64</td>
</tr>
<tr>
<td>Image</td>
</tr>
</tbody>
</table>

Reinitialize this and train

Freeze these

Train these

Freeze these

With bigger dataset, train more layers

Lower learning rate when finetuning; 1/10 of original LR is a good starting point

<table>
<thead>
<tr>
<th></th>
<th>Very similar dataset</th>
<th>Very different dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Very little data</strong></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><strong>Quite a lot of data</strong></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- **More specific**
- **More generic**
More specific

More generic

<table>
<thead>
<tr>
<th></th>
<th>very similar dataset</th>
<th>very different dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>very little data</strong></td>
<td>Use Linear Classifier on top layer</td>
<td>?</td>
</tr>
<tr>
<td><strong>quite a lot of data</strong></td>
<td>Finetune a few layers</td>
<td>?</td>
</tr>
</tbody>
</table>

- Conv-64
- Conv-64
- MaxPool
- Conv-128
- Conv-128
- MaxPool
- Conv-256
- Conv-256
- MaxPool
- Conv-512
- Conv-512
- MaxPool
- Conv-128
- Conv-128
- MaxPool
- Conv-64
- Conv-64
- Image

- FC-1000
- FC-4096
- FC-4096
### More generic vs. More specific

<table>
<thead>
<tr>
<th>More generic</th>
<th>More specific</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image</strong></td>
<td><strong>Conv-64</strong></td>
</tr>
<tr>
<td>Conv-512</td>
<td>Conv-512</td>
</tr>
<tr>
<td>MaxPool</td>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-256</td>
<td>Conv-256</td>
</tr>
<tr>
<td>MaxPool</td>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-128</td>
<td>Conv-128</td>
</tr>
<tr>
<td>MaxPool</td>
<td>MaxPool</td>
</tr>
<tr>
<td>Conv-64</td>
<td>Conv-64</td>
</tr>
<tr>
<td>Conv-64</td>
<td>Conv-64</td>
</tr>
</tbody>
</table>

### Dataset Differences

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th>Action (very similar dataset)</th>
<th>Action (very different dataset)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>very little data</strong></td>
<td>Use Linear Classifier on top layer</td>
<td>You’re in trouble… Try linear classifier from different stages</td>
</tr>
<tr>
<td><strong>quite a lot of data</strong></td>
<td>Finetune a few layers</td>
<td>Finetune a larger number of layers</td>
</tr>
</tbody>
</table>
Transfer learning with CNNs is pervasive…
(it’s the norm, not an exception)

Object Detection
(Fast R-CNN)

Image Captioning: CNN + RNN

Figure copyright Ross Girshick, 2015. Reproduced with permission.

Figure copyright IEEE, 2015. Reproduced for educational purposes.
Transfer learning with CNNs is pervasive…
(it’s the norm, not an exception)

Object Detection
(Fast R-CNN)

CNN pretrained on ImageNet

Image Captioning: CNN + RNN

Figure copyright Ross Girshick, 2015. Reproduced with permission.

Figure copyright IEEE, 2015. Reproduced for educational purposes.
Transfer learning with CNNs is pervasive…
(it’s the norm, not an exception)

Object Detection
(Fast R-CNN)

CNN pretrained
on ImageNet

Image Captioning: CNN + RNN

Word embedding layer
pretrained with word2vec

Figure copyright Ross Girshick, 2015. Reproduced with permission.

Figure copyright IEEE, 2015. Reproduced for educational purposes.
Transfer learning with CNNs is pervasive…
(it’s the norm, not an exception)

1. Train CNN on ImageNet
2. Fine-Tune (1) for object detection on Visual Genome
   1. Train BERT language model on lots of text
   2. Combine(2) and (3), train for joint image / language modeling
3. Fine-tune (4) for image captioning, visual question answering, etc.

Zhou et al, “Unified Vision-Language Pre-Training for Image Captioning and VQA” CVPR 2020
Figure copyright Luowei Zhou, 2020. Reproduced with permission.

Krishna et al, “Visual genome: Connecting language and vision using crowdsourced dense image annotations” IJCV 2017
Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” ArXiv 2018
Transfer learning is pervasive…
But recent results show it might not always be necessary!

Training from scratch can work just as well as training from a pretrained ImageNet model for object detection

But it takes 2-3x as long to train.

They also find that collecting more data is better than finetuning on a related task.

He et al, “Rethinking ImageNet Pre-training”, ICCV 2019
Figure copyright Kaiming He, 2019. Reproduced with permission.
Takeaway for your projects and beyond:

Transfer learning be like

Source: AI & Deep Learning Memes For Back-propagated Poets
Takeaway for your projects and beyond:
Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big neural network there
2. Transfer learn to your dataset

Deep learning frameworks provide a “Model Zoo” of pretrained models so you don’t need to train your own

TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision
Image Classification: A core task in Computer Vision

(assume given a set of possible labels)
{dog, cat, truck, plane, ...}
Structured prediction tasks in vision

- Classification
- Semantic Segmentation
- Object Detection
- Instance Segmentation

**Classification**
- CAT

**Semantic Segmentation**
- GRASS, CAT, TREE, SKY

**Object Detection**
- DOG, DOG, CAT

**Instance Segmentation**
- DOG, DOG, CAT

No spatial extent
No objects, just pixels
Multiple Object

This image is CC0 public domain
Semantic Segmentation

- Classification
- Semantic Segmentation
- Object Detection
- Instance Segmentation

GRASS, CAT, TREE, SKY

No spatial extent
No objects, just pixels
Multiple Object
Semantic Segmentation: The Problem

Paired training data: for each training image, each pixel is labeled with a semantic category.

At test time, classify each pixel of a new image.

GRASS, CAT, TREE, SKY, ...

?
Semantic Segmentation Idea: Sliding Window

Full image
Semantic Segmentation Idea: Sliding Window

Full image

Impossible to classify without context

Q: how do we include context?
Semantic Segmentation Idea: Sliding Window

Q: how do we model this?
Semantic Segmentation Idea: Sliding Window

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014
Semantic Segmentation Idea: Sliding Window

Problem: Very inefficient! Not reusing shared features between overlapping patches

Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014
Semantic Segmentation Idea: Convolution

An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.
Semantic Segmentation Idea: Fully Convolutional

- Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!
Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!

Input: $3 \times H \times W$

Convolutions: $D \times H \times W$

Scores: $C \times H \times W$

Predictions: $H \times W$

Problem: convolutions at original image resolution will be very expensive ...
Semantic Segmentation Idea: Fully Convolutional

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

Semantic Segmentation Idea: Fully Convolutional

**Downsampling:**
Pooling, strided convolution

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

**Input:**
3 x H x W

**High-res:**
D₁ x H/2 x W/2

**Low-res:**
D₃ x H/4 x W/4

**Med-res:**
D₂ x H/4 x W/4

**High-res:**
C x H x W

**Predictions:**
H x W

---

### In-Network upsampling: “Unpooling”

#### Nearest Neighbor

<table>
<thead>
<tr>
<th>Input: 2 x 2</th>
<th>Output: 4 x 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>1 1 2 2</td>
</tr>
<tr>
<td>3 4</td>
<td>1 1 2 2</td>
</tr>
<tr>
<td></td>
<td>3 3 4 4</td>
</tr>
<tr>
<td></td>
<td>3 3 4 4</td>
</tr>
</tbody>
</table>

#### “Bed of Nails”

<table>
<thead>
<tr>
<th>Input: 2 x 2</th>
<th>Output: 4 x 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>1 0 2 0</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0</td>
<td>3 0 4 0</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>
In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

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

Input: 4 x 4

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

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

Input: 2 x 2

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers
Learnable Upsampling

**Recall:** Normal 3 x 3 convolution, stride 1 pad 1

Input: 4 x 4

Output: 4 x 4
Learnable Upsampling

**Recall**: Normal 3 x 3 convolution, stride 1 pad 1

Input: 4 x 4

Dot product between filter and input

Output: 4 x 4
Learnable Upsampling

**Recall:** Normal 3 x 3 convolution, stride 1 pad 1

Input: 4 x 4

Dot product between filter and input

Output: 4 x 4
Learnable Upsampling

**Recall:** Normal 3 x 3 convolution, **stride 2** pad 1

**Input:** 4 x 4  
**Output:** 2 x 2
Learnable Upsampling

**Recall:** Normal 3 x 3 convolution, **stride 2** pad 1

- Input: 4 x 4
- Output: 2 x 2

Dot product between filter and input
Learnable Upsampling

Recall: Normal 3 x 3 convolution, **stride** 2 pad 1

Input: 4 x 4

Dot product between filter and input

Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

We can interpret strided convolution as “learnable downsampling”.

Learnable Upsampling: Transposed Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4
Learnable Upsampling: Transposed Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4
Learnable Upsampling: Transposed Convolution

3 x 3 **transpose** convolution, stride 2 pad 1

Input gives weight for filter

Input: 2 x 2  
Output: 4 x 4

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input
Learnable Upsampling: Transposed Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input gives weight for filter

Input: 2 x 2

Output: 4 x 4

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Sum where output overlaps

Input gives weight for filter

Nov 21, 2023

Ali Farhadi, Aditya Kusupati
Learnable Upsampling: Transposed Convolution

Q: Why is it called transpose convolution?

Input: 2 x 2

Output: 4 x 4

3 x 3 transpose convolution, stride 2 pad 1

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Sum where output overlaps

Input gives weight for filter

Learnable Upsampling: Transposed Convolution
Learnable Upsampling: 1D Example

Output contains copies of the filter weighted by the input, summing at where at overlaps in the output.
Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} \ast \vec{a} = X \vec{a}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$
Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication:

$$\vec{x} \ast \vec{a} = X \vec{a}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} \ast^T \vec{a} = X^T \vec{a}$$

Example: 1D transpose conv, kernel size=3, stride=2, padding=0

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$
Semantic Segmentation Idea: Fully Convolutional

**Downsampling:**
- Pooling, strided convolution

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

- **Input:** $3 \times H \times W$
- **High-res:** $D_1 \times H/2 \times W/2$
- **Med-res:** $D_2 \times H/4 \times W/4$
- **Low-res:** $D_3 \times H/4 \times W/4$
- **High-res:** $D_1 \times H/2 \times W/2$
- **Predictions:** $H \times W$

**Upsampling:**
- Unpooling or strided transpose convolution

---

Today: UNet with residual connections

Input: $3 \times H \times W$

Predictions: $H \times W$

Newell et al. Stacked Hourglass Networks for Human Pose Estimation. ECCV 2016
Semantic Segmentation: Summary
Semantic Segmentation

Label each pixel in the image with a category label.

Don’t differentiate instances, only care about pixels.
Object Detection

Classification

Semantic Segmentation

Object Detection

Instance Segmentation

CAT

GRASS, CAT, TREE, SKY

DOG, DOG, CAT

DOG, DOG, CAT

No spatial extent

No objects, just pixels

Multiple Object

Ali Farhadi, Aditya Kusupati  Lecture 14 -  57  Nov 21, 2023
Object Detection

Classification

Semantic Segmentation

Object Detection

Instance Segmentation

CAT

GRASS, CAT, TREE, SKY

No spatial extent

No objects, just pixels

DOG, DOG, CAT

Multiple Object

DOG, DOG, CAT
Object Detection: Single Object
(Classification + Localization)

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Vector: 4096
Fully Connected: 4096 to 4
Box Coordinates
(x, y, w, h)

Fully Connected: 4096 to 1000

x, y

This image is CC0 public domain
Object Detection: Single Object
(Classification + Localization)

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label:
Cat

Fully Connected:
4096 to 1000

Correct box:
(x’, y’, w’, h’)

Vector:
4096

L2 Loss

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

Treat localization as a regression problem!
Object Detection: Single Object  
(Classification + Localization)

Class Scores  
Cat: 0.9  
Dog: 0.05  
Car: 0.01  
...

Vector: 4096

Fully Connected: 4096 to 1000  
Class Scores  
Cat: 0.9  
Dog: 0.05  
Car: 0.01  
...

Softmax Loss

Multitask Loss

Correct label:  
Cat

Box Coordinates  
(x, y, w, h)

L2 Loss

Correct box:  
(x', y', w', h')

Treat localization as a regression problem!
Object Detection: Multiple Objects

CAT: (x, y, w, h)

DOG: (x, y, w, h)

DUCK: (x, y, w, h)

....
Object Detection: Multiple Objects

Each image needs a different number of outputs!

CAT: \((x, y, w, h)\) 
4 numbers

DOG: \((x, y, w, h)\) 
12 numbers

CAT: \((x, y, w, h)\)

DUCK: \((x, y, w, h)\) 
Many numbers!

DUCK: \((x, y, w, h)\)

....
Object Detection: Multiple Objects

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog? NO
Cat? NO
Background? YES
Object Detection: Multiple Objects

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog? YES
Cat? NO
Background? NO
Object Detection: Multiple Objects

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background.

Dog? YES
Cat? NO
Background? NO
Object Detection: Multiple Objects

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Q: What’s the problem with this approach?

Dog? NO
Cat? YES
Background? NO
Object Detection: Multiple Objects

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Dog? NO
Cat? YES
Background? NO
Region Proposals: Selective Search

- Find “bloppy” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU

Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014
R-CNN

Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
R-CNN

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

Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
R-CNN

Input image

Warped image regions (224x224 pixels)

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

Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
R-CNN

Input image

ConvNet

ConvNet

ConvNet

Warped image regions (224x224 pixels)

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

Forward each region through ConvNet (ImageNet-pretrained)

R-CNN

Input image

ConvNet

ConvNet

ConvNet

SVMs

SVMs

SVMs

Warped image regions (224x224 pixels)

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

Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretrained)

Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
R-CNN

Input image

ConvNet

ConvNet

ConvNet

SVMs

SVMs

SVMs

Bbox reg

Bbox reg

Bbox reg

Warped image regions (224x224 pixels)

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

Forward each region through ConvNet (ImageNet-pretrained)

Classify regions with SVMs

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)

Classify regions with SVMs

R-CNN

Problem: Very slow! Need to do ~2k independent forward passes for each image!

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

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

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)

“Slow” R-CNN

Problem: Very slow! Need to do ~2k independent forward passes for each image!

Idea: Pass the image through convnet before cropping! Crop the conv feature instead!

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

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

Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
Fast R-CNN

Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source, Reproduced with permission.
Fast R-CNN

"Backbone" network: AlexNet, VGG, ResNet, etc

Run whole image through ConvNet

"conv5" features

ConvNet

Input image

"Slow" R-CNN

ConvNet

ConvNet

ConvNet

SVMs

SVMs

Input image

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

“Backbone” network: AlexNet, VGG, ResNet, etc

“conv5” features

Run whole image through ConvNet

ConvNet

Input image

“Slow” R-CNN

SVMs

ConvNet

SVMs

ConvNet

Input image

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

“Backbone” network: AlexNet, VGG, ResNet, etc

Run whole image through ConvNet

Crop + Resize features

“conv5” features

Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source, Reproduced with permission.
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

“Backbone” network: AlexNet, VGG, ResNet, etc

ConvNet

Input image

CNN

Per-Region Network

Crop + Resize features

“conv5” features

Run whole image through ConvNet

Linear + softmax

Linear

Box offset

Object category

Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

“Backbone” network: AlexNet, VGG, ResNet, etc

Input image

ConvNet

“Slow” R-CNN

Regions of Interest (RoIs) from a proposal method

“Backbone” network: AlexNet, VGG, ResNet, etc

Run whole image through ConvNet

Crop + Resize features

“conv5” features

Per-Region Network

Object category

Linear + softmax

Linear

Box offset

Fast R-CNN

Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
Cropping Features: RoI Pool

Input Image
(e.g. 3 x 640 x 480)

Image features: C x H x W
(e.g. 512 x 20 x 15)

Cropping Features: RoI Pool

Input Image (e.g. 3 x 640 x 480)

Project proposal onto features

Image features: C x H x W (e.g. 512 x 20 x 15)

Cropping Features: RoI Pool

“Snap” to grid cells

Project proposal onto features

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features: C x H x W
(e.g. 512 x 20 x 15)

Cropping Features: RoI Pool

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features: C x H x W
(e.g. 512 x 20 x 15)

Project proposal onto features

“Snap” to grid cells

Q: how do we resize the 512 x 20 x 15 region to, e.g., a 512 x 2 x 2 tensor?

Cropping Features: RoI Pool

Project proposal onto features

“Snap” to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Q: how do we resize the 512 x 20 x 15 region to, e.g., a 512 x 2 x 2 tensor?

Input Image (e.g. 3 x 640 x 480)

CNN

Image features: C x H x W (e.g. 512 x 20 x 15)

Cropping Features: RoI Pool

Project proposal onto features

"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion

Region features (here 512 x 2 x 2; In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

Input Image (e.g. 3 x 640 x 480)

CNN

Image features: C x H x W (e.g. 512 x 20 x 15)

Cropping Features: RoI Pool

- Project proposal onto features
- “Snap” to grid cells
- Divide into 2x2 grid of (roughly) equal subregions
- Max-pool within each subregion

Input Image
(e.g. 3 x 640 x 480)

Image features: C x H x W
(e.g. 512 x 20 x 15)

Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

Problem: Region features slightly misaligned

Cropping Features: RoI Align

Project proposal onto features

No “snapping”!

He et al, "Mask R-CNN", ICCV 2017

Input Image
(e.g. 3 x 640 x 480)

Image features: C x H x W
(e.g. 512 x 20 x 15)
Cropping Features: RoI **Align**

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features: C x H x W
(e.g. 512 x 20 x 15)

He et al, "Mask R-CNN", ICCV 2017

Sample at regular points in each subregion using bilinear interpolation

No “snapping”!

Project proposal onto features
Cropping Features: RoI Align

Project proposal onto features

No “snapping”!

Sample at regular points in each subregion using bilinear interpolation

Feature $f_{xy}$ for point $(x, y)$ is a linear combination of features at its four neighboring grid cells:

He et al, “Mask R-CNN”, ICCV 2017
Cropping Features: RoI Align

He et al, "Mask R-CNN", ICCV 2017

Sample at regular points in each subregion using bilinear interpolation

No “snapping”!

Feature $f_{xy}$ for point $(x, y)$ is a linear combination of features at its four neighboring grid cells:

$$f_{xy} = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$
Cropping Features: RoI Align

Input Image (e.g. 3 x 640 x 480)

CNN

He et al, “Mask R-CNN”, ICCV 2017

Project proposal onto features

Sample at regular points in each subregion using bilinear interpolation

No “snapping”!

Max-pool within each subregion

Region features (here 512 x 2 x 2; In practice e.g 512 x 7 x 7)

Image features: C x H x W (e.g. 512 x 20 x 15)
R-CNN vs Fast R-CNN

Girshick, "Fast R-CNN", ICCV 2015
R-CNN vs Fast R-CNN

![Graphs showing training and test time comparisons between R-CNN, SPP-Net, and Fast R-CNN. The graphs indicate that Fast R-CNN has significantly lower training and test times compared to R-CNN and SPP-Net.]

Problem:
Runtime dominated by region proposals!

**Faster R-CNN:**
Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN:
Crop features for each proposal, classify each one

---


Figure copyright 2015, Ross Girshick; reproduced with permission
Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features
(e.g. 512 x 20 x 15)
Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map.
Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map.

At each point, predict whether the corresponding anchor contains an object (binary classification).

Input Image (e.g. 3 x 640 x 480)

CNN

Image features (e.g. 512 x 20 x 15)

Conv

Anchor is an object? 1 x 20 x 15
Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map.

- **Input Image**
  - Example: 3 x 640 x 480

- **CNN**
  - Image features
    - Example: 512 x 20 x 15

- **Conv**
  - Anchor is an object?
    - 1 x 20 x 15
  - Box corrections
    - 4 x 20 x 15

For positive boxes, also predict corrections from the anchor to the ground-truth box (regress 4 numbers per pixel).
Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

CNN

Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

In practice use K different anchor boxes of different size / scale at each point
Region Proposal Network

In practice use $K$ different anchor boxes of different size / scale at each point

Anchor is an object? $K \times 20 \times 15$

Box transforms $4K \times 20 \times 15$

Sort the $K \times 20 \times 15$ boxes by their “objectness” score, take top $\sim 300$ as our proposals

**Input Image**
(e.g. $3 \times 640 \times 480$)

**CNN**

**Image features**
(e.g. $512 \times 20 \times 15$)

**Conv**

**Output Boxes**

$K \times 20 \times 15$
Faster R-CNN:
Make CNN do proposals!

Jointly train with 4 losses:
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

Figure copyright 2015, Ross Girshick; reproduced with permission
Faster R-CNN:
Make CNN do proposals!
Faster R-CNN:
Make CNN do proposals!

Glossing over many details:
- Ignore overlapping proposals with **non-max suppression**
- How are anchors determined?
- How do we sample positive / negative samples for training the RPN?
- How to parameterize bounding box regression?

Figure copyright 2015, Ross Girshick; reproduced with permission
Faster R-CNN: Make CNN do proposals!

Faster R-CNN is a **Two-stage object detector**

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset
Faster R-CNN: Make CNN do proposals!

Faster R-CNN is a Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

Do we really need the second stage?

Classification loss
Bounding-box regression loss

Rol pooling
Single-Stage Object Detectors: YOLO / SSD / RetinaNet

Divide image into grid $7 \times 7$

Image a set of **base boxes** centered at each grid cell

Here $B = 3$

Within each grid cell:
- Regress from each of the $B$ base boxes to a final box with 5 numbers: $(dx, dy, dh, dw, \text{confidence})$
- Predict scores for each of $C$ classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:

$7 \times 7 \times (5 \times B + C)$

Lin et al., “Focal Loss for Dense Object Detection”, ICCV 2017
Object Detection: Lots of variables ...

<table>
<thead>
<tr>
<th>Backbone Network</th>
<th>“Meta-Architecture”</th>
<th>Takeaways</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>Two-stage: Faster R-CNN</td>
<td>Faster R-CNN is slower but more accurate</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>Single-stage: YOLO / SSD</td>
<td>SSD is much faster but not as accurate</td>
</tr>
<tr>
<td>Inception V2</td>
<td>Hybrid: R-FCN</td>
<td>Bigger / Deeper backbones work better</td>
</tr>
<tr>
<td>Inception V3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inception</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResNet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MobileNet</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Image Size

# Region Proposals

...
Object Detection: Lots of variables ...

**Backbone Network**
- VGG16
- ResNet-101
- Inception V2
- Inception V3
- ResNet
- MobileNet

**“Meta-Architecture”**
- Two-stage: Faster R-CNN
- Single-stage: YOLO / SSD
- Hybrid: R-FCN

**Image Size**

**# Region Proposals**

**Takeaways**
- Faster R-CNN is slower but more accurate
- SSD is much faster but not as accurate
- Bigger / Deeper backbones work better

Huang et al, “Speed/accuracy trade-offs for modern convolutional object detectors”, CVPR 2017

Instance Segmentation

Classification

Semantic Segmentation

Object Detection

Instance Segmentation

CAT

GRASS, CAT, TREE, SKY

DOG, DOG, CAT

NO spatial extent

No objects, just pixels

Multiple Object

Ali Farhadi, Aditya Kusupati

Lecture 14 - 113

Nov 21, 2023
Object Detection: Faster R-CNN

Classification loss
Bounding-box regression loss

Object Detection
Instance Segmentation

Region Proposal Network

CNN

image

feature map

proposals

Roll pooling
Instance Segmentation: Mask R-CNN

Add a small mask network that operates on each RoI and predicts a 28x28 binary mask.
Mask R-CNN

He et al, "Mask R-CNN", arXiv 2017
Mask R-CNN: Example Mask Training Targets
Mask R-CNN: Example Mask Training Targets
Mask R-CNN: Example Mask Training Targets
Mask R-CNN: Example Mask Training Targets
Mask R-CNN: Very Good Results!

He et al, "Mask R-CNN", ICCV 2017
Mask R-CNN
Also does pose

He et al, "Mask R-CNN", ICCV 2017
Open Source Frameworks

Lots of good implementations on GitHub!

TensorFlow Detection API:
https://github.com/tensorflow/models/tree/master/research/object_detection
Faster RCNN, SSD, RFCN, Mask R-CNN, ...

Detectron2 (PyTorch)
https://github.com/facebookresearch/detectron2
Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN, ...

Finetune on your own dataset with pre-trained models
Beyond 2D Object Detection...
Object Detection + Captioning
= Dense Captioning

Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016
Figure copyright IEEE, 2016. Reproduced for educational purposes.
Dense Video Captioning

Ranjay Krishna et al., "Dense-Captioning Events in Videos", ICCV 2017
Figure copyright IEEE, 2017. Reproduced with permission.
Objects + Relationships = Scene Graphs

Scene Graph Prediction

Krishna, Lu, Bernstein, and Fei-Fei, “Scene Graph Generation by Iterative Message Passing”, ECCV 2016
Figure copyright IEEE, 2018. Reproduced for educational purposes.
3D Object Detection

2D Object Detection:  
2D bounding box  
(x, y, w, h)

3D Object Detection:  
3D oriented bounding box  
(x, y, z, w, h, l, r, p, y)

Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!
3D Object Detection: Monocular Camera

Candidate sampling in 3D space

- Same idea as Faster RCNN, but proposals are in 3D
- 3D bounding box proposal, regress 3D box parameters + class score

Recap: Lots of computer vision tasks!

Classification

No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Multiple Object

Instance Segmentation

DOG, DOG, CAT

This image is CC0 public domain
Next time:
Generative Models
A point on the image plane corresponds to a ray in the 3D space.

A 2D bounding box on an image is a frustrum in the 3D space.

Localize an object in 3D:
The object can be anywhere in the camera viewing frustrum!
3D Shape Prediction: Mesh R-CNN

- Input Image
- 2D Recognition
- sofa
- chair
- 3D Meshes
- 3D Voxels

Gkioxari et al., Mesh RCNN, ICCV 2019