High Level Computer Vision

Object Detection & Segmentation

Bernt Schiele - schiele@mpi-inf.mpg.de
Mario Fritz - fritz@cispa.saarland

https://www.mpi-inf.mpg.de/hlcv
**Computer Vision**
- Scene Understanding
- Segmentation
- QA and Captioning

**Security & Privacy**
- Understanding & Controlling Privacy of Data & Models
- Adversarial Machine Learning
- Uncertainty
- Interpretability

**Machine Learning**
- Deep Learning
- Domain Adaptation
- Generative Adversarial Networks, Variational Autoencoders

Ours: a skier is headed down a steep slope

What sport is this man enjoying? snowboarding
So far: Image Classification

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

Vector: 4096

Fully-Connected: 4096 to 1000

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

This image is CC0 public domain

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Today: Detection, Segmentation

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Other Computer Vision Tasks

Semantic Segmentation

Classification + Localization

Object Detection

Instance Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

CAT

Single Object

DOG, DOG, CAT

Multiple Object

DOG, DOG, CAT

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 8 May 10, 2018

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Other Computer Vision Tasks

Semantic Segmentation

- GRASS, CAT, TREE, SKY
  - No objects, just pixels

2D Object Detection

- DOG, DOG, CAT
  - Object categories + 2D bounding boxes

3D Object Detection

- Car
  - Object categories + 3D bounding boxes

This image is CC0 public domain

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Semantic Segmentation

Semantic Segmentation

2D Object Detection

3D Object Detection

GRASS, CAT, TREE, SKY
No objects, just pixels

DOG, DOG, CAT
Object categories + 2D bounding boxes

Car
Object categories + 3D bounding boxes

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Semantic Segmentation

Label each pixel in the image with a category label.

Don’t differentiate instances, only care about pixels.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Semantic Segmentation Idea: Sliding Window

- Extract patch
- Classify center pixel with CNN

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

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers 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: $H \times W$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers 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 ...

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Semantic Segmentation Idea: Fully Convolutional

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:** \(D_4 \times H/4 \times W/4\)
- **Predictions:** \(H \times W\)


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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:** $D_1 \times H/2 \times W/2$
- **High-res:** $D_1 \times H/2 \times W/2$
- **Predictions:** $H \times W$


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
In-Network upsampling: “Unpooling”

Nearest Neighbor

Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

Input: 2 x 2

Output: 4 x 4

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

\[
\begin{array}{cccc}
1 & 2 & 6 & 3 \\ 
3 & 5 & 2 & 1 \\ 
1 & 2 & 2 & 1 \\ 
7 & 3 & 4 & 8 \\
\end{array}
\]

Input: 4 x 4

Output: 2 x 2

Max Unpooling
Use positions from pooling layer

\[
\begin{array}{cccc}
0 & 0 & 2 & 0 \\ 
0 & 1 & 0 & 0 \\ 
0 & 0 & 0 & 0 \\ 
3 & 0 & 0 & 4 \\
\end{array}
\]

Input: 2 x 2

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

Rest of the network

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

Recall: Typical 3 x 3 convolution, stride 1 pad 1

Input: 4 x 4

Output: 4 x 4

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

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

Input: 4 x 4

Dot product between filter and input

Output: 4 x 4

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

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

Input: 4 x 4

Output: 4 x 4

Dot product between filter and input

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

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

Input: 4 x 4

Output: 2 x 2

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

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

Input: 4 x 4

Dot product between filter and input

Output: 2 x 2

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

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

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

**Dot product between filter and input**
- Filter moves 2 pixels in the input for every one pixel in the output
- Stride gives ratio between movement in input and output

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2  
Output: 4 x 4

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

3 \times 3 \text{transpose} \text{ convolution, stride 2 pad 1}

Input: 2 \times 2

Output: 4 \times 4

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

Input gives weight for filter

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

Sum where output overlaps

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

Stride gives ratio between movement in output and input

Input gives weight for filter

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: Transpose Convolution

Other names:
- Deconvolution (bad)
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

Input gives weight for filter

Sum where output overlaps

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

Stride gives ratio between movement in output and input

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learnable Upsampling: 1D Example

Output contains copies of the filter weighted by the input, summing at where at overlaps in the output.

Need to crop one pixel from output to make output exactly 2x input.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \\ 0 & 0 & 0 & 0 & x & y \\ 0 & 0 & 0 & 0 & 0 & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1
Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

\[ \tilde{x} \ast \tilde{a} = X \tilde{a} \]

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

\[
\begin{bmatrix}
x & y & x & 0 & 0 & 0 \\
0 & x & y & x & 0 & 0 \\
0 & 0 & x & y & x & 0 \\
0 & 0 & 0 & x & y & x \\
0 & 0 & 0 & 0 & x & y \\
\end{bmatrix}
\begin{bmatrix}
0 \\
a \\
b \\
c \\
d \\
0 \\
\end{bmatrix}
= 
\begin{bmatrix}
ay + bz \\
ax + by + cz \\
bx + cy + dz \\
ex + dy \\
\end{bmatrix}
\]

Convolution transpose multiplies by the transpose of the same matrix:

\[ \tilde{x} \ast^T \tilde{a} = X^T \tilde{a} \]

\[
\begin{bmatrix}
x & 0 & 0 & 0 \\
y & x & 0 & 0 \\
z & y & x & 0 \\
0 & z & y & x \\
0 & 0 & z & y \\
0 & 0 & 0 & z \\
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c \\
d \\
\end{bmatrix}
= 
\begin{bmatrix}
ax \\
ay + bx \\
az + by + cx \\
bz + cy + dx \\
\end{bmatrix}
\]

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Convolutions as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

\[ \vec{x} \ast \vec{a} = X \vec{a} \]

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

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

Convolution transpose multiplies by the transpose of the same matrix:

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

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

When stride>1, convolution transpose is no longer a normal convolution!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Semantic Segmentation Idea: Fully Convolutional

**Downsampling:**
Pooling, strided convolution

**Upsampling:**
Unpooling or strided transpose convolution

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

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

2D Object Detection

Semantic Segmentation

GRASS, CAT, TREE, SKY
No objects, just pixels

2D Object Detection

DOG, DOG, CAT
Object categories + 2D bounding boxes

3D Object Detection

Car
Object categories + 3D bounding boxes

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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

Treat localization as a regression problem!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Classification + Localization

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

Box Coordinates (x, y, w, h)

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 1000

Vector: 4096

Fully Connected: 4096 to 4

L2 Loss

Correct box: (x’, y’, w’, h’)

Treat localization as a regression problem!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

This image is CC0 public domain
Classification + Localization

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

Softmax Loss

Multitask Loss

Correct label: Cat

Vector: 4096

Fully Connected: 4096 to 1000

Box Coordinates (x, y, w, h)

L2 Loss

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

Treat localization as a regression problem!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Classification + Localization

Often pretrained on ImageNet (Transfer learning)

Treat localization as a regression problem!
Object Detection as Regression?

CAT: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Object Detection as Regression?

Each image needs a different number of outputs!

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

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

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

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Object Detection as Classification: Sliding Window

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

**Dog? NO**
**Cat? NO**
**Background? YES**

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Object Detection as Classification: Sliding Window

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

Dog? YES
Cat? NO
Background? NO

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Object Detection as Classification: Sliding Window

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

Dog? YES
Cat? NO
Background? NO

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Object Detection as Classification: Sliding Window

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

Dog? NO
Cat? YES
Background? NO

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Object Detection as Classification: Sliding Window

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Region Proposals / Selective Search

- Find “blobby” 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
Uijlings et al., “Selective Search for Object Recognition”, IJCV 2013
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.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
R-CNN

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


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
R-CNN

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
R-CNN

R-CNN is an object detection method that uses a Region Proposal Network (RPN) to propose regions of interest (RoI) within an image. These RoIs are then fed into a ConvNet to extract features. The forward pass of the ConvNet is applied to each RoI to process the image regions. This approach allows for more efficient and accurate object detection compared to previous methods.

Girshick et al., “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014. Figure copyright Ross Girshick, 2015; source: Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
R-CNN

- Regions of Interest (RoI) from a proposal method (~2k)
- Forward each region through ConvNet
- Classify regions with SVMs

Input image

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
R-CNN

- Linear Regression for bounding box offsets
- Classify regions with SVMs
- Forward each region through ConvNet
- Warped image regions
- Regions of Interest (RoI) from a proposal method (~2k)


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
R-CNN: Problems

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]

Slide copyright Ross Girshick, 2015; source. Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Segmentation as Selective Search for Object Recognition

K. van de Sande\textsuperscript{1}, J. Uijlings\textsuperscript{2}, T. Gevers\textsuperscript{1}, and A. Smeulders\textsuperscript{1}
University of Amsterdam\textsuperscript{1} and University of Trento\textsuperscript{2}

Reading Group presentation by Esa Rahtu

(material taken from van de Sande’s ICCV paper and PASCAL presentations)
Motivation

• Most current approaches use exhaustive search
  – Visit every location in an image
  – Imposes computational constraints on
    • Number of possible locations $\rightarrow$ grid/fixed aspect ratio
    • Evaluation cost per location $\rightarrow$ simple features/classifiers
  – To go beyond this, we need something more sophisticated

Viola IJCV 2004
Dalal CVPR 2005
Felzenszwalb TPAMI 2010
Vedaldi ICCV 2009
Main design criteria

• **High recall**
  – We do not want to lose any objects, since they cannot be recovered later.

• **Coarse locations are sufficient**
  – Accurate delineation is not necessary for recognition
  – In contrary, nearby context might be useful
    -> use bounding boxes

• **Fast to compute**
  – Necessary when operating with large datasets
    -> <10s/image
How to obtain high recall?

• Images are intrinsically hierarchical

• Segmentation at single scale are not enough
  -> hypotheses based on hierarchical grouping
Proposed method

- Start by oversegmenting the input image

“Efficient graph-based image segmentation”
Felzenszwalb and Huttenlocher, IJCV 2004
Method

• compute similarity measure between all adjacent region pairs a and b (e.g.) as:

$$S(a, b) = \alpha S_{size}(a, b) + \beta S_{color}(a, b)$$

- with

$$S_{size}(a, b) = 1 - \frac{\text{size}(a) + \text{size}(b)}{\text{size(image)}}$$

encourages small regions to merge early

- and

$$S_{color}(a, b) = \sum_{k=1}^{n} \min(a^k, b^k)$$

$a^k, b^k$ are color histograms, encouraging “similar” regions to merge

- for slightly more elaborated similarities see their IJCV-paper
Proposed method

1. Merge two most similar regions based on $S$.
2. Update similarities between the new region and its neighbors.
3. Go back to step 1. until the whole image is a single region.
Proposed method

• Take bounding boxes of all generated regions and treat them as possible object locations.
Proposed method
High recall revisited

- No single segmentation works for all cases
  -> diversify the set of segmentations
- Use different color spaces
  - RGB, Opponent color, normalized RGB, and hue
- Use different parameters in Felzenswalb method
  - $k = [100, 150, 200, 250]$ ($k =$ threshold parameter)
Evaluation of object hypotheses

- Recall is a proportion of objects that are covered by some box with >0.5 overlap

VOC2007 test

1,536 windows/image
96.7% recall

Selected settings
Fast R-CNN

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Fast R-CNN

“conv5” feature map of image

Forward whole image through ConvNet

ConvNet

Input image

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

“conv5” feature map of image

Forward whole image through ConvNet

ConvNet

Input image

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

“RoI Pooling” layer

“conv5” feature map of image

Forward whole image through ConvNet

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Fast R-CNN: RoI Pooling

Hi-res input image: 3 x 640 x 480 with region proposal

Hi-res conv features: 512 x 20 x 15;
Projected region proposal is e.g. 512 x 18 x 8 (varies per proposal)

RoI conv features: 512 x 7 x 7 for region proposal

Fully-connected layers expect low-res conv features: 512 x 7 x 7


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Fast R-CNN

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Fast R-CNN

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Fast R-CNN (Training)

Log loss + Smooth L1 loss

softmax

Linear

FCs

Multi-task loss

ConvNet

Input image

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Fast R-CNN (Training)

Log loss + Smooth L1 loss

Multi-task loss

softmax
Linear

FCs

ConvNet

Input image

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
R-CNN vs SPP vs Fast R-CNN

Training time (Hours)

<table>
<thead>
<tr>
<th>Method</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN</td>
<td>84</td>
</tr>
<tr>
<td>SPP-Net</td>
<td>25.5</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>8.75</td>
</tr>
</tbody>
</table>

Test time (seconds)

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN</td>
<td>49</td>
</tr>
<tr>
<td>SPP-Net</td>
<td>2.3</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>0.32</td>
</tr>
</tbody>
</table>


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
R-CNN vs SPP vs Fast R-CNN

Training time (Hours)

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN</td>
<td>84</td>
</tr>
<tr>
<td>SPP-Net</td>
<td>25.5</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>8.75</td>
</tr>
</tbody>
</table>

Test time (seconds)

<table>
<thead>
<tr>
<th>Method</th>
<th>Including Region proposals</th>
<th>Excluding Region proposals</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN</td>
<td>49</td>
<td>47</td>
</tr>
<tr>
<td>SPP-Net</td>
<td>4.3</td>
<td>2.3</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>2.3</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Problem: Runtime dominated by region proposals!


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Faster R-CNN:
Make CNN do proposals!

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
**Faster R-CNN:**
Make CNN do proposals!

![R-CNN Test-Time Speed](chart)

- R-CNN: 49
- SPP-Net: 4.3
- Fast R-CNN: 2.3
- Faster R-CNN: 0.2

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
### Mask R-CNN

- **CNN**
  - **RoI Align**
  - **Conv**
    - Classification Scores: $C$
    - Box coordinates (per class): $4 \times C$
    - Predict a mask for each of $C$ classes
    - $C \times 14 \times 14$

---

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Mask R-CNN: Very Good Results!

He et al, “Mask R-CNN”, arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Mask R-CNN
Also does pose

He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Detection without Proposals: YOLO / SSD

Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Input image
3 x H x W

Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Output:
7 x 7 x (5 * B + C)


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutional network!

Input image
3 x H x W

Divide image into grid
7 x 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, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
## Object Detection: Lots of variables …

<table>
<thead>
<tr>
<th>Base Network</th>
<th>Object Detection architecture</th>
<th>Takeaways</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>Faster R-CNN</td>
<td>Faster R-CNN is slower but more accurate</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>R-FCN</td>
<td>SSD is much faster but not as accurate</td>
</tr>
<tr>
<td>Inception V2</td>
<td>SSD</td>
<td></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**
- ...

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


slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Object Detection: Impact of Deep Learning

Figure copyright Ross Girshick, 2015. Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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

Caffe2 Detectron:
https://github.com/facebookresearch/Detectron
Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN

Finetune on your own dataset with pre-trained models

slide credit: Fei-Fei, Justin Johnson, Serena Yeung