Outline

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
4. Beyond ImageNet
5. Generative models, GANs
6. Segmentation
Semantic segmentation: One step further for visual scene understanding
Object Detection

Detect every instance of the category and localize it with a bounding box.
Semantic Segmentation

Label each pixel with a category label
Simultaneous Detection and Segmentation

*Detect and segment every instance of the category in the image*
Simultaneous Detection, Segmentation and Part Labeling

Detect and segment every instance of the category in the image and label its parts.
Supervised Segmentation with Deep ConvNets
Recent Deep strategies for Supervised Segmentation

1. F-CN Fully Convolutional Network
2. DeepLab approach for supervised segmentation
3. Deconvolution Networks
FCN: Fully-Convolutional Network

- CNN: state-of-the-art model for image classification

- Fully-convolutional network: classify each “pixel”

FCN: Fully-Convolutional Network

- Classify each region
FCN: Fully-Convolutional Network

- Classify each region
FCN: Fully-Convolutional Network

- Classify each region
FCN: Fully-Convolutional Network

- Classify each region
FCN: Fully-Convolutional Network

- Classify each region
FCN: Fully-Convolutional Network

- Fully-convolutional network: classify each “pixel”
- Upsampling output (bilinear interpolation + deconvolution)
- Network architecture: AlexNet, VGG16, GoogleNet
- Loss: soft-max per pixel
FCN: Fully-Convolutional Network

Learning process

1. Model pretrained on ImageNet
2. Decapitate each net by discarding the final classifier layer
3. Convert all fully-connected layers to convolutions
4. Append $n^1$ 1 × 1 convolutions
5. Fine-tuning all layers by backpropagation

$^1n=$number of classes
FCN: Fully-Convolutional Network

- Problem: max pooling and striding reduces spatial resolution
- Dense prediction: combines feature hierarchies
- Initialized with the parameters of coarse net
- Fine-tuning all layers by backpropagation
### FCN: Fully-Convolutional Network

<table>
<thead>
<tr>
<th></th>
<th>pixel acc.</th>
<th>mean acc.</th>
<th>mean IU</th>
<th>f.w. IU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN-32s-fixed</td>
<td>83.0</td>
<td>59.7</td>
<td>45.4</td>
<td>72.0</td>
</tr>
<tr>
<td>FCN-32s</td>
<td>89.1</td>
<td>73.3</td>
<td>59.4</td>
<td>81.4</td>
</tr>
<tr>
<td>FCN-16s</td>
<td>90.0</td>
<td>75.7</td>
<td>62.4</td>
<td>83.0</td>
</tr>
<tr>
<td>FCN-8s</td>
<td><strong>90.3</strong></td>
<td><strong>75.9</strong></td>
<td><strong>62.7</strong></td>
<td><strong>83.2</strong></td>
</tr>
</tbody>
</table>

![Images showing results for different models](image)

**FCN-32s-fixed**: only the last layer is fine-tuned
Recent Deep strategies for Supervised Segmentation

1. F-CN Fully Convolutional Network
2. DeepLab approach for supervised segmentation
3. Deconvolution Networks
DeepLab [ICLR 15] approach for supervised segmentation

Problem of the spatial resolution reduction

Solution of the DeepLab approach

1. Learn CNN for dense prediction tasks (Atrous)
2. Improve the localization of object boundaries with fully-connected CRF [?] (FC-CRF)
Atrous (‘Holes’) Algorithm

- Remove the down-sampling from the last pooling layers.
- Up-sample the original filter by a factor of the strides:

Atrous convolution for 1-D signal:

\[
y[i] = \sum_{k=1}^{K} x[i + r \cdot k] w[k]
\]

- Note: standard convolution is a special case for rate \( r=1 \).
DeepLab: Atrous (Dilated) Convolution

- Efficient dense feature extraction
- Upsample filters

(a) Sparse feature extraction
(b) Dense feature extraction
Atrous (‘Holes’) Algorithm

Filters field-of-view

- **Small** field-of-view → accurate localization
- **Large** field-of-view → context assimilation
- ‘Holes’: Introduce zeros between filter values.
- **Effective filter size increases** (enlarge the field-of-view of filter):
  \[ k \times k \text{ filter to } k_e = k + (k - 1)(r - 1) \]
- However, we take into account **only** the non-zero filter values:
  - ✓ Number of filter parameters is the same.
  - ✓ Number of operations per position is the same.
Atrous (‘Holes’) Algorithm

Standard convolution

Atrous convolution

Original filter

downsampling
stride=2

convolution
kernel=7

upsampling
stride=2

Padded filter

atrous convolution
kernel=7
rate=2
stride=1
DeepLab: Fully-Connected CRF

• Problem: poor object delineation (spatial and appearance consistency neglected)
• Solution: fully-connected CRF accounts for contextual information in the image

\[ E(y) = \sum_i \theta_i(y_i) + \sum_{i,j} \theta_{ij}(y_i, y_j) \]

- Unary term: output of FCN (upscaled)
- Pairwise term: penalizes similar pixels having different labels
DeepLab V3+ [ECCV 2018]
CRF as RNN

- Unified framework: combines CNN and CRF
- Formulate mean-field approximate inference for FC-CRF as Recurrent Neural Networks (RNN)
- Learning the network end-to-end

CRF as RNN

<table>
<thead>
<tr>
<th>Model</th>
<th>[Long et al. CVPR 2015]</th>
<th>[Chen et al. ICLR 2015]</th>
<th>[Ours, ICCV 2015]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN</td>
<td>68.3</td>
<td>69.5</td>
<td>72.9</td>
</tr>
<tr>
<td>FCN → CRF</td>
<td>[Ours, ICCV 2015]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRF → FCN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRF-RNN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Recent Deep strategies for Supervised Segmentation

1. F-CN Fully Convolutional Network
2. DeepLab approach for supervised segmentation
3. Deconvolution Networks
Deconvolution Network

- Learn a multi-layer deconvolution network
- Network is composed of two parts:
  1. Convolution: feature extractor
  2. Deconvolution: shape generator that produces object segmentation from the feature extracted
- Deconvolution net is a mirrored version of the convolution net

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In ICCV, 2015. [paper]
Deconvolution Network

Unpooling

- Perform the reverse operation of pooling
- Reconstruct the original size of activations
- Useful to reconstruct the structure of input object
- Output: sparse activation map
Deconvolution Network

Deconvolution

- Connect single input activation to a multiple activations
- Learned filters correspond to bases to reconstruct shape of an input object
- Output: enlarged and dense activation map
Deconvolution Network
Fig. 3: CEN-s: The proposed architecture by Brosch et al. [11].
U-Net

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Very popular in MICCAI 2016
Works well with low data

Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

http://deeplearning.net/tutorial/unet.html
Datasets
PASCAL VOC 12

- Train 1464 images / Val 1449 images / Test 1456 images
- 21 classes: aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train, tv/monitor + background
- Evaluation: intersection-over-union metric
- Webpage: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/
COCO

- Train 80k images / Val 20k images
- 91 classes, 11 super-categories:

![Image of COCO categories]

- 3 challenges: detection, instance segmentation, captioning
- Webpage: [http://mscoco.org] [paper]
Weakly Supervised Segmentation
Supervised Image Segmentation Methods

Full supervision

- Precise annotation 😊
- Expensive and time consuming to obtain
  - "79s per label per image" [RBFL15] 😞
- Bottleneck for learning models at large scale 😞
Weakly Supervised Image Segmentation Methods

Weak supervision

- Reduce supervision: class labels (or tags)
- Cheap to obtain
  - "1s per label per image" [RBFL15]
- Scalable to large number of categories

![Image with labels]

- ✓ background
- ✗ aeroplane
- ✗ cat
- ✓ chair
- ✗ dog
- ✗ person
- ✗ sheep
- ✓ table
- ✗ tvmonitor
Weakly supervised segmentation with CNN

Standard learning algorithms

• Maximize the likelihood of the observed training data

Problem

• Require full knowledge of the ground truth labeling
  ▶ not available in the weakly supervised setting 😞

Solutions

1. Generation of segmentation mask


2. Modified loss function: CNN optimized for classification

Generation of segmentation mask


Idea: adaptive bias

- Generated segmentation mask and train fully-supervised CNN
- Adaptive bias into the multi-instance learning framework
  - Boost classes known to be present
  - Suppress all others
Segmentation Results

@Y. LeCun
Segmentation Results

@Y. LeCun
Segmentation Results

@Y. LeCun
Segmentation Results

@Y. LeCun
Segmentation Results

@Y. LeCun

=> Mask-R-CNN
Appendix
“Deconvolution” or Transposed Conv.

Want to swap the input and output dimensions
Outline

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
4. Beyond ImageNet
5. Generative models, GANs
6. Segmentation
7. Detection
Computer Vision Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation

CAT

CAT

CAT, DOG, DUCK

CAT, DOG, DUCK

Single object

Multiple objects
Computer Vision Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation

Potentially different objects per image
Bounding boxes available in train
Detection
Recap compa. With WSL for classif/localization vs. Detection

Classification
NO Box labels
ONLY global labels

Fully supervised detection
Box labels available in train
Detection

Modeling as a regression problem?

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

= 16 numbers
Detection as Regression?

DOG, (x, y, w, h)
CAT, (x, y, w, h)
= 8 numbers
Detection as Regression?

Need variable sized outputs
=> Not easy to model

CAT, (x, y, w, h)
CAT, (x, y, w, h)
...
CAT (x, y, w, h)
= many numbers
Detection as Classification

Modeling detection as classification with a sliding window approach?

CAT? NO
DOG? NO
Detection as Classification

Modeling detection as classification with a sliding window approach?

CAT? YES!

DOG? NO
Detection as Classification

Modeling detection as classification with a sliding window approach?

CAT? NO

DOG? NO
Detection as Classification

**Problem:** Need to test many positions and scales, and use a computationally demanding classifier (CNN)

**Solution:** Only look at a tiny subset of possible positions
Detection scheme

1. RoI detection, region proposal

2. Region classifier

Dog, cat, background

Bounding boxes in train
Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions
Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes

Detection scheme

Region selection

Region classifier

warped region
Detection: R-CNN approach [CVPR 2014]

Region CNN (R-CNN) approach
Putting it together: R-CNN

Apply bounding-box regressors
Classify regions with SVMs

Forward each region through ConvNet

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

Warped image regions


Slide credit: Ross Girshick
R-CNN Training

Step 1: Train (or download) a classification model for ImageNet (AlexNet)
R-CNN Training

**Step 2:** Fine-tune model for detection
- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images
R-CNN Training

**Step 3: Extract features**
- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!
R-CNN Training

**Step 4:** Train one binary SVM per class to classify region features

**Training image regions**:

- Positive samples for cat SVM
- Negative samples for cat SVM

**Cached region features**: 

- Positive samples for cat SVM
- Negative samples for cat SVM
R-CNN Training

**Step 4**: Train one binary SVM per class to classify region features
R-CNN Training

Step 5 (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals.

Training image regions

Cached region features

Regression targets (dx, dy, dw, dh)

Normalized coordinates

Proposal is good

(0, 0, 0, 0)

Proposal too far to left

(.25, 0, 0, 0)

Proposal too wide

(0, 0, -0.125, 0)
Object Detection: Datasets

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of classes</td>
<td>20</td>
<td>200</td>
<td>80</td>
</tr>
<tr>
<td>Number of images (train + val)</td>
<td>~20k</td>
<td>~470k</td>
<td>~120k</td>
</tr>
<tr>
<td>Mean objects per image</td>
<td>2.4</td>
<td>1.1</td>
<td>7.2</td>
</tr>
</tbody>
</table>
Background: Evaluating object detectors

- Algorithm outputs ranked list of boxes with category labels
- Compute overlap between detection and ground truth box

\[
\text{Overlap IoU} = \frac{\text{Area}(\cap)}{\text{Area}(\cup)}
\]
Background: Evaluating object detectors

- Algorithm outputs ranked list of boxes with category labels
- Compute overlap between detection and ground truth box

Overlap $\text{IoU} = \frac{|\text{intersection}|}{|\text{union}|}$
Background: Evaluating object detectors

- Algorithm outputs ranked list of boxes with category labels
- Compute overlap between detection and ground truth box

\[
\text{Overlap IoU} = \frac{\text{\# of overlapping pixels}}{\text{Total number of pixels in both boxes}}
\]
Object Detection: Evaluation

Evaluation metric: mean average precision (mAP)

Compute average precision (AP) separately for each class, then average over classes

A detection is a true positive if it has IoU with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)

Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve

mAP is a number from 0 to 100; high is good
R-CNN Results

Big improvement compared to pre-CNN methods

![Bar chart showing R-CNN Results with improvements compared to pre-CNN methods](chart.png)
R-CNN Results

Bounding box regression helps a bit

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP VOC 2007</th>
<th>mAP VOC 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM (2011)</td>
<td>33.7</td>
<td>29.6</td>
</tr>
<tr>
<td>Regionlets (2013)</td>
<td>41.7</td>
<td>39.7</td>
</tr>
<tr>
<td>R-CNN (2014, AlexNet)</td>
<td>54.2</td>
<td>50.2</td>
</tr>
<tr>
<td>R-CNN + bbox reg (AlexNet)</td>
<td>58.5</td>
<td>53.7</td>
</tr>
<tr>
<td>R-CNN (VGG-16)</td>
<td>66</td>
<td>62.9</td>
</tr>
</tbody>
</table>
R-CNN Results

Features from a deeper network help a lot

![Bar Chart]

- DPM (2011)
- Regionlets (2013)
- R-CNN (2014, AlexNet)
- R-CNN + bbox reg (AlexNet)
- R-CNN (VGG-16)

VOC 2007
VOC 2010
R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal

2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors

3. Complex multistage training pipeline
Fast R-CNN (test time)

- Softmax classifier
- Linear + softmax
- Linear
- Bounding-box regressors
- Fully-connected layers
- "Rol Pooling" (single-level SPP) layer
- "conv5" feature map of image
- Forward whole image through ConvNet

Regions of Interest (Rols) from a proposal method

ConvNet

Input image

Slide credit: Ross Girschick
R-CNN Problem #1: Slow at test-time due to independent forward passes of the CNN

Solution: Share computation of convolutional layers between proposals for an image
R-CNN Problem #2:
Post-hoc training: CNN not updated in response to final classifiers and regressors.

R-CNN Problem #3:
Complex training pipeline

Solution:
Just train the whole system end-to-end all at once!

Slide credit: Ross Girshick
Fast R-CNN: Region of Interest Pooling

Hi-res input image: 3 x 800 x 600 with region proposal

Convolution and Pooling

Hi-res conv features: C x H x W with region proposal

Fully-connected layers

Problem: Fully-connected layers expect low-res conv features: C x h x w
Fast R-CNN: Region of Interest Pooling

- **Hi-res input image:** 3 x 800 x 600 with region proposal
- **Hi-res conv features:** C x H x W with region proposal
- **Problem:** Fully-connected layers expect low-res conv features: C x h x w
Fast R-CNN: Region of Interest Pooling

Hi-res input image:
3 x 800 x 600
with region proposal

Hi-res conv features:
C x H x W
with region proposal

Divide projected region into h x w grid

Fully-connected layers

Problem: Fully-connected layers expect low-res conv features: C x h x w
Fast R-CNN: Region of Interest Pooling

Hi-res input image: 3 x 800 x 600 with region proposal

Hi-res conv features: C x H x W with region proposal

Max-pool within each grid cell

RoI conv features: C x h x w for region proposal

Fully-connected layers expect low-res conv features: C x h x w
Fast R-CNN: Region of Interest Pooling

Hi-res input image: 3 x 800 x 600 with region proposal

Convolution and Pooling

Hi-res conv features: \( C \times H \times W \) with region proposal

Can back propagate similar to max pooling

RoI conv features: \( C \times h \times w \) for region proposal

Fully-connected layers expect low-res conv features: \( C \times h \times w \)
**Fast R-CNN Results**

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Time:</td>
<td>84 hours</td>
<td>9.5 hours</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>8.8x</td>
</tr>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
</tr>
</tbody>
</table>

Using VGG-16 CNN on Pascal VOC 2007 dataset
Fast R-CNN Problem Solution:

Test-time speeds don’t include region proposals
Just make the CNN do region proposals too!

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</td>
</tr>
<tr>
<td>Test time per image with Selective Search</td>
<td>50 seconds</td>
<td>2 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>25x</td>
</tr>
</tbody>
</table>
Faster R-CNN:

Insert a Region Proposal Network (RPN) after the last convolutional layer

RPN trained to produce region proposals directly; no need for external region proposals!

After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN


Slide credit: Ross Girshick
Faster R-CNN: Training

In the paper: Ugly pipeline
- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

Since publication: Joint training!
One network, four losses
- RPN classification (anchor good / bad)
- RPN regression (anchor -> proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal -> box)
Faster R-CNN: Results

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
<th>Faster R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image (with proposals)</td>
<td>50 seconds</td>
<td>2 seconds</td>
<td>0.2 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>25x</td>
<td>250x</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
<td>66.9</td>
</tr>
</tbody>
</table>
Object Detection State-of-the-art: 
ResNet 101 + Faster R-CNN + some extras

<table>
<thead>
<tr>
<th>training data</th>
<th>COCO train</th>
<th>COCO trainval</th>
</tr>
</thead>
<tbody>
<tr>
<td>test data</td>
<td>COCO val</td>
<td>COCO test-dev</td>
</tr>
<tr>
<td>mAP</td>
<td>@.5</td>
<td>@ [.5, .95]</td>
</tr>
<tr>
<td>baseline Faster R-CNN (VGG-16)</td>
<td>41.5</td>
<td>21.2</td>
</tr>
<tr>
<td>baseline Faster R-CNN (ResNet-101) + box refinement</td>
<td>48.4</td>
<td>27.2</td>
</tr>
<tr>
<td>+context</td>
<td>51.1</td>
<td>30.0</td>
</tr>
<tr>
<td>+multi-scale testing</td>
<td>53.8</td>
<td>32.5</td>
</tr>
<tr>
<td>ensemble</td>
<td>53.3</td>
<td>32.2</td>
</tr>
<tr>
<td></td>
<td>55.7</td>
<td>34.9</td>
</tr>
<tr>
<td></td>
<td>59.0</td>
<td>37.4</td>
</tr>
</tbody>
</table>

To sum up: Detection

Region CNN (R-CNN) approach

Fast(er) R-CNN approach
Other approaches: Yolo (You Only Look Once), SSD (Single Shot Detector), RetinaNet, …