Object Detectors

Last Time: Transfer Learning

1. Train on ImageNet

Last Time: Localization Tasks

Classification

CAT

No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

Multiple Objects

This image is CC0 public domain

Last Time: R-CNN

Classify each region

Bounding box regression:
Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

Warped image regions (224x224)

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

Input image

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

Last Time: R-CNN

Forward each region through ConvNet

Warped image regions (224x224)

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

Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Problem: Very slow! Need to do 2000 forward passes through CNN per image

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

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

Last Time: R-CNN

Forward each region through ConvNet

Warped image regions (224x224)

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

Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Problem: Very slow! Need to do 2000 forward passes through CNN per image

Idea: Overlapping proposals cause a lot of repeated work: same pixels processed many times. Can we avoid this?

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

“Slow” R-CNN

Process each region independently

Fast R-CNN

Input image

"Slow" R-CNN

Process each region independently

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

Fast R-CNN

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

"Slow" R-CNN

Process each region independently

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

Image features

Run whole image through ConvNet

Input image

"Slow" R-CNN

Process each region independently

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

Crop + Resize features

Image features

Run whole image through ConvNet

Input image

"Slow" R-CNN

Process each region independently

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

Per-Region Network

Crop + Resize features

Image features

Run whole image through ConvNet

Input image

"Slow" R-CNN

Process each region independently

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

Fast R-CNN

"Slow" R-CNN

Process each region independently

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

Fast R-CNN

Category and box transform per region

Per-Region network is relatively lightweight

Per-Region Network

Regions of Interest (RoIs) from a proposal method

Crop + Resize features

Image features

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

Run whole image through ConvNet

Most of the computation happens in backbone network; this saves work for overlapping region proposals

Input image

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

Fast R-CNN

Example: When using AlexNet for detection, five conv layers are used for backbone and two FC layers are used for per-region network

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

Fast R-CNN

Category and box transform per region

Per-Region Network

Crop + Resize features

Image features

Run whole image through ConvNet

Input image

Regions of Interest (RoIs) from a proposal method

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

Example:
For ResNet, last stage is used as per-region network; the rest of the network is used as backbone

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

Fast R-CNN

Category and box transform per region

Per-Region Network

Regions of Interest (RoIs) from a proposal method

Crop + Resize features

Image features

How to crop features?

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

Run whole image through ConvNet

Input image

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

Recall: Receptive Fields

Input Image: 8 x 8

Every position in the output feature map depends on a 3x3 receptive field in the input

3x3 Conv
Stride 1, pad 1

Output Image: 8 x 8

Recall: Receptive Fields

Input Image: 8 x 8

Every position in the output feature map depends on a 3x3 receptive field in the input

3x3 Conv
Stride 1, pad 1

Output Image: 8 x 8

Recall: Receptive Fields

Input Image: 8 x 8

Every position in the output feature map depends on a 5x5 receptive field in the input

3x3 Conv
Stride 1, pad 1

3x3 Conv
Stride 1, pad 1

Output Image: 8 x 8

Recall: Receptive Fields

Input Image: 8 x 8

Moving one unit in the output space also moves the receptive field by one

3x3 Conv
Stride 1, pad 1

3x3 Conv
Stride 1, pad 1

Output Image: 8 x 8

Recall: Receptive Fields

Moving one unit in the output space also moves the receptive field by one

3x3 Conv
Stride 1, pad 1

3x3 Conv
Stride 1, pad 1

There is a correspondence between the coordinate system of the input and the coordinate system of the output

Projecting Points

We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

3x3 Conv
Stride 1, pad 1

There is a correspondence between the coordinate system of the input and the coordinate system of the output

Projecting Points

Same logic holds for more complicated CNNs, even if spatial resolution of input and output are different

We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

2x2 MaxPool
Stride 2

There is a correspondence between the coordinate system of the input and the coordinate system of the output

Projecting Points

Same logic holds for more complicated CNNs, even if spatial resolution of input and output are different

We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

4x4 MaxPool
Stride 4

There is a correspondence between the coordinate system of the input and the coordinate system of the output

Projecting Boxes

We can use this idea to project **bounding boxes** between an input image and a feature map

We can align arbitrary points between coordinate system of input and output

3x3 Conv
Stride 1, pad 1

4x4 MaxPool
Stride 4

There is a correspondence between the coordinate system of the input and the coordinate system of the output

Cropping Features: RoI Pool

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

Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool

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

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

Want features for the box of a fixed size (2x2 in this example, 7x7 or 14x14 in practice)

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

Cropping Features: RoI Pool

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

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

Want features for the box of a fixed size (2x2 in this example, 7x7 or 14x14 in practice)

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

Cropping Features: RoI Pool

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

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

Want features for the box of a fixed size (2x2 in this example, 7x7 or 14x14 in practice)

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

Cropping Features: RoI Pool

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

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

Divide into 2x2
grid of (roughly)
equal subregions

Want features for the
box of a fixed size
(2x2 in this example,
7x7 or 14x14 in practice)

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

Cropping Features: RoI Pool

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

Project proposal onto features

CNN

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

"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 512x7x7)

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

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

Cropping Features: RoI Pool

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

Project proposal onto features

CNN

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

"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 512x7x7)

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

Problem: Slight misalignment due to snapping; different-sized subregions is weird

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

Cropping Features: RoI Align

Divide into equal-sized subregions
(may not be aligned to grid!)

No “snapping”!

Project proposal
onto features

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

CNN

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

Want features for the
box of a fixed size
(2x2 in this example,
7x7 or 14x14 in practice)

He et al, “Mask R-CNN”, ICCV 2017.

Cropping Features: RoI Align

Divide into equal-sized subregions (may not be aligned to grid!)

No "snapping"!

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

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

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

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

Cropping Features: RoI Align

Divide into equal-sized subregions (may not be aligned to grid!)

No "snapping"!

Project proposal onto features

CNN

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

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

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

Cropping Features: RoI Align

Divide into equal-sized subregions (may not be aligned to grid!)

No "snapping"!

Project proposal onto features

CNN

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

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

Cropping Features: RoI Align

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

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

Cropping Features: RoI Align

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

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

Cropping Features: RoI Align

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

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

Cropping Features: RoI Align

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2) + (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$$

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

Cropping Features: RoI Align

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

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

Sample features at regularly-spaced points in each subregion using **bilinear interpolation**

After sampling, max-pool in each subregion

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

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

Fast R-CNN vs “Slow” R-CNN

Fast R-CNN: Apply differentiable cropping to shared image features

“Slow” R-CNN: Apply differentiable cropping to shared image features

Fast R-CNN vs “Slow” R-CNN

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Fast R-CNN vs “Slow” R-CNN

Problem: Runtime dominated by region proposals!

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Fast R-CNN vs “Slow” R-CNN

Problem: Runtime dominated by region proposals!

Recall: Region proposals computed by heuristic ”Selective Search” algorithm on CPU -- let’s learn them with a CNN instead!

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

Fast<u>er</u> R-CNN: Learnable Region 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

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Region Proposal Network (RPN)

Run backbone CNN to get
features aligned to input image

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

Image features
(e.g. 512 x 5 x 6)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

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

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

Predict object vs not object scores for all anchors with a conv layer (512 input filters, 2 output filters)

Anchor is object?
2 x 5 x 6

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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

For positive anchors, also predict a transform that converting the anchor to the GT box (like R-CNN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

Anchor is object?
2 x 5 x 6

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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

For positive anchors, also predict a transform that converting the anchor to the GT box (like R-CNN)
Predict transforms with conv

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Classify each anchor as positive (object) or negative (no object)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

CNN

Image features
(e.g. 512 x 5 x 6)

Conv

Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

During training, supervised positive / negative anchors and box transforms like R-CNN

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

CNN

Image features
(e.g. 512 x 5 x 6)

Conv

Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

Positive anchors: >= 0.7 IoU with some GT box (plus highest IoU to each GT)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Negative anchors: < 0.3 IoU with all GT boxes. Don't supervised transforms for negative boxes.

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

CNN

Image features
(e.g. 512 x 5 x 6)

Conv

Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

Neutral anchors: between 0.3 and 0.7 IoU with all GT boxes; ignored during training

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

CNN

Image features
(e.g. 512 x 5 x 6)

Conv

Anchor is object?
2K x 5 x 6

Anchor transforms
4K x 5 x 6

At test-time, sort all K*5*6 boxes by their positive score, take top 300 as our region proposals

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Faster R-CNN: Learnable Region Proposals

Jointly train with 4 losses:

1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Fast<u>er</u> R-CNN: Learnable Region Proposals

Faster R-CNN: Learnable Region 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

Dealing with Scale

We need to detect objects of many different scales.
How to improve *scale invariance* of the detector?

This image is free for commercial use under the Pixabay license

Dealing with Scale: Image Pyramid

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.

Problem: Expensive! Don't share any computation between scales

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Multiscale Features

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Multiscale Features

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level

Problem: detector on early features doesn't make use of the entire backbone; doesn't get access to high-level features

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice

Faster R-CNN with RPN: Detector at each level gets its own RPN to produce proposals; proposals from all levels route to a shared second stage

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Faster R-CNN: Learnable Region Proposals

Question: Do we really need the second stage?

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

Single-Stage Detectors: RetinaNet

Similar to RPN – but rather than classify anchors as object/no object, directly predict object category (among C categories) or background

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

CNN

Image features
(e.g. 512 x 5 x 6)

Conv

→ Anchor classification
2K*(C+1) x 5 x 6

→ Anchor transforms
4K x 5 x 6

Lin et al, “Focal Loss for Dense Object Detection”, ICCV 2017

Single-Stage Detectors: RetinaNet

Problem: class imbalance – many more background anchors vs non-background

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

CNN

Image features
(e.g. 512 x 5 x 6)

Conv

→ Anchor classification
2K*(C+1) x 5 x 6

→ Anchor transforms
4K x 5 x 6

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Single-Stage Detectors: RetinaNet

Problem: class imbalance – many more background anchors vs non-background

Solution: new loss function (Focal Loss); see paper

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

CNN

Image features
(e.g. 512 x 5 x 6)

Conv

→ Anchor classification
2K*(C+1) x 5 x 6

→ Anchor transforms
4K x 5 x 6

$$\mathrm{CE}(p_t) = -\log(p_t)$$

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t)$$

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Single-Stage Detectors: RetinaNet

In practice, RetinaNet also uses Feature Pyramid Network to handle multiscale

Lin et al, “Focal Loss for Dense Object Detection”, ICCV 2017

Figure credit: Lin et al, ICCV 2017

Single-Stage Detectors: RetinaNet

Single-Stage detectors can be much faster than two-stage detectors

	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

Lin et al, “Focal Loss for Dense Object Detection”, ICCV 2017

Figure credit: Lin et al, ICCV 2017

Single-Stage Detectors: RetinaNet

Single-Stage detectors can be much faster than two-stage detectors

	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

Faster R-CNN with Feature Pyramid Network

Lin et al, “Focal Loss for Dense Object Detection”, ICCV 2017

Figure credit: Lin et al, ICCV 2017

Single-Stage Detectors: FCOS

"Anchor-free" detector

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Single-Stage Detectors: FCOS

"Anchor-free" detector

Classify points as positive if they fall into a GT box, or negative if they don't

Train independent per-category logistic regressors

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

→ Class scores
C x 5 x 6

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Single-Stage Detectors: FCOS

"Anchor-free" detector

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

For positive points, also regress distance to left, right, top, and bottom of ground-truth box (with L2 loss)

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

Image features
(e.g. 512 x 5 x 6)

Class scores
C x 5 x 6

Box edges
4 x 5 x 6

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Single-Stage Detectors: FCOS

"Anchor-free" detector

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

For positive points, also regress distance to left, right, top, and bottom of ground-truth box (with L2 loss)

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

CNN

Image features
(e.g. 512 x 5 x 6)

CNN

Class scores
C x 5 x 6

Box edges
4 x 5 x 6

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Single-Stage Detectors: FCOS

"Anchor-free" detector

Finally, predict "centerness" for all positive points (using logistic regression loss)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

CNN

Image features
(e.g. 512 x 5 x 6)

CNN

Class scores
C x 5 x 6

Box edges
4 x 5 x 6

Centerness
1 x 5 x 6

$$centerness = \sqrt{\frac{\min(L,R)}{\max(L,R)} \cdot \frac{\min(T,B)}{\max(T,B)}}$$

Ranges from 1 at box center to 0 at box edge

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Single-Stage Detectors: FCOS

"Anchor-free" detector

Test-time: predicted "confidence" for the box from each point is product of its class score and centerness

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

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

Image features
(e.g. 512 x 5 x 6)

Class scores
C x 5 x 6

Box edges
4 x 5 x 6

Centerness
1 x 5 x 6

$$centerness = \sqrt{\frac{\min(L,R)}{\max(L,R)} \cdot \frac{\min(T,B)}{\max(T,B)}}$$

Ranges from 1 at box center to 0 at box edge

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Single-Stage Detectors: FCOS

"Anchor-free" detector

FCOS also uses a Feature Pyramid Network with heads shared across stages

Tian et al, "FCOS: Fully Convolutional One-Stage Object Detection", ICCV 2019

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve

Precision = 1/1 = 1.0

Recall = 1/3 = 0.33

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve

Precision = 2/2 = 1.0

Recall = 2/3 = 0.67

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve

Precision = 2/3 = 0.67
Recall = 2/3 = 0.67

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve

Precision = 2/4 = 0.5

Recall = 2/3 = 0.67

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve

Precision = 3/5 = 0.6

Recall = 3/3 = 1.0

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
   2. Average Precision (AP) = area under PR curve

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
   2. Average Precision (AP) = area under PR curve

How to get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
   2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
   2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category
4. For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

…

mAP@0.95 = 0.2

COCO mAP = 0.4

Summary: Beyond Image Classification

Classification

CAT

No spatial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

Multiple Objects

This image is CC0 public domain

Summary

"Slow" R-CNN: Run CNN independently for each region

Fast R-CNN: Apply differentiable cropping to shared image features

Faster R-CNN: Compute proposals with CNN

Single-Stage: Fully convolutional detector

With anchors: RetinaNet
Anchor-Free: FCOS