Computer Science – Data Mining Master's Program

Faster R-CNN: Towards Object Detection with Region Proposal Networks[1]

+ Dataset and Implementation

Presenters:
Mahdi Rostami
Reza Pourbahreini

Instructor:
Dr. Ali Katanforoush

Artificial Neural Networks Course

Spring 2022

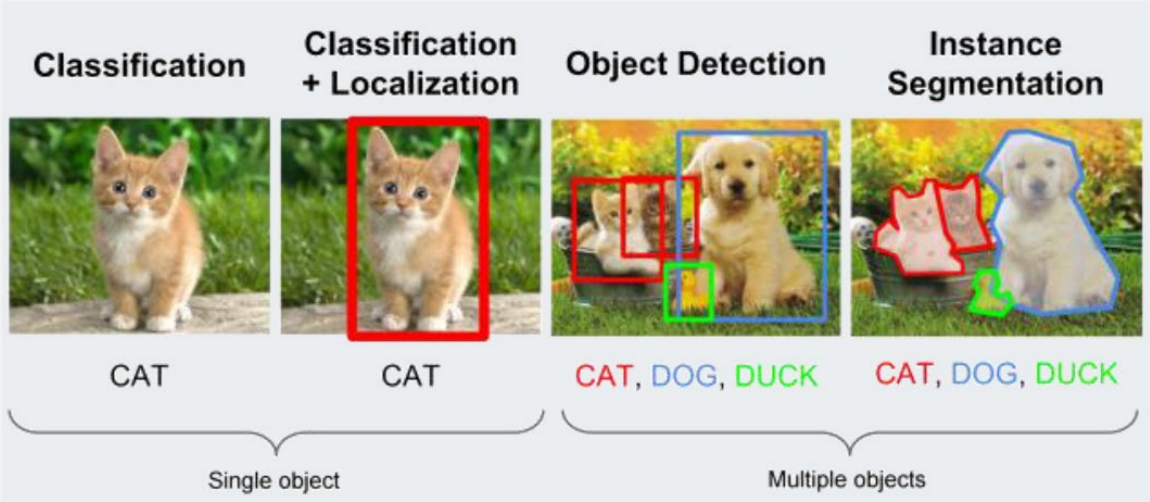
[1] Faster R-CNN: Towards Object Detection with Region Proposal Networks - Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun- NIPS' 15 - Published: Dec 2015 - https://doi.org/10.48550/arXiv.1506.01497

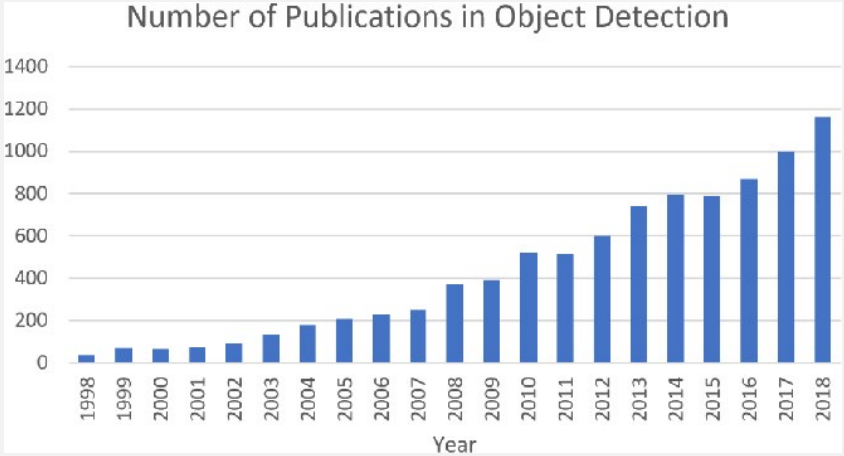
- **Introduction**
  - Task Definition
  - Applications
  - Concise Chronology of Recent Approaches Towards Object Detection
  - Region-Based Models : R-CNN , Fast R-CNN , Faster R-CNN , Mask R-CNN , Mesh R-CNN , …

- **Theory & Pipelines :** R-CNN , Fast R-CNN , Faster R-CNN
  - Model Architectures
  - RPN (Region Proposal Network)

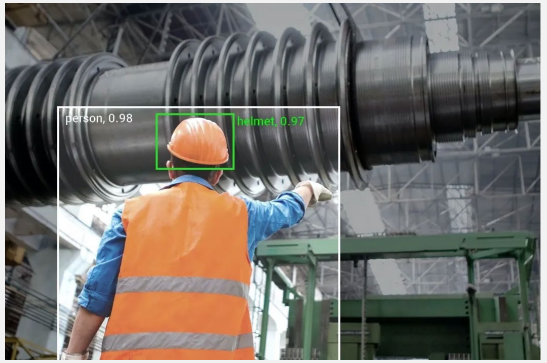
- **Practice - Faster R-CNN**
  - Dataset
  - Implementation
  - Results

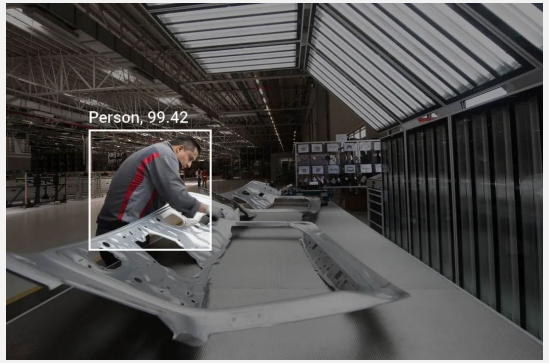
- **Task Definition**
  - **Single Object**
    - Classification
    - Classification + Localization
  - **Multiple Objects**
    - Object Detection
    - Instance Segmentation

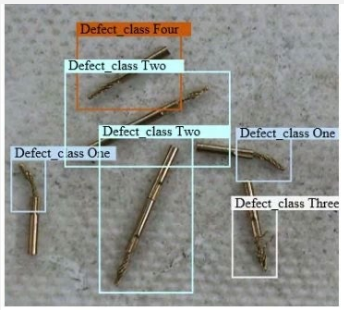


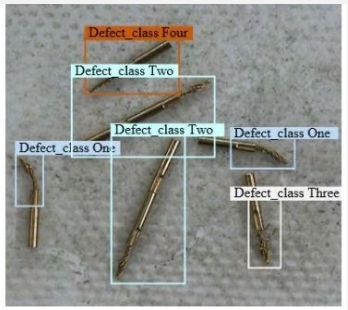


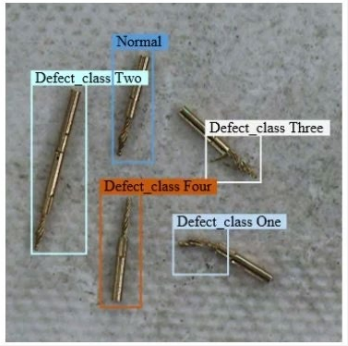
- **Applications**
  - **Manufacturing**
    - Productivity Analysis
    - Visual Inspection of Equipment
  - **Surveillance Systems**
    - Identity Verification and Authentication
    - Person Detection
    - Motion Tracking
  - **Healthcare**
    - Cancer Detection
    - Image-based Diagnosis and Prognosis
  - **Autonomous Driving**
    - Pedestrian Detection
    - Obstacle Avoidance
  - **Smartphones**
    - Face Detection
    - Smile Detection



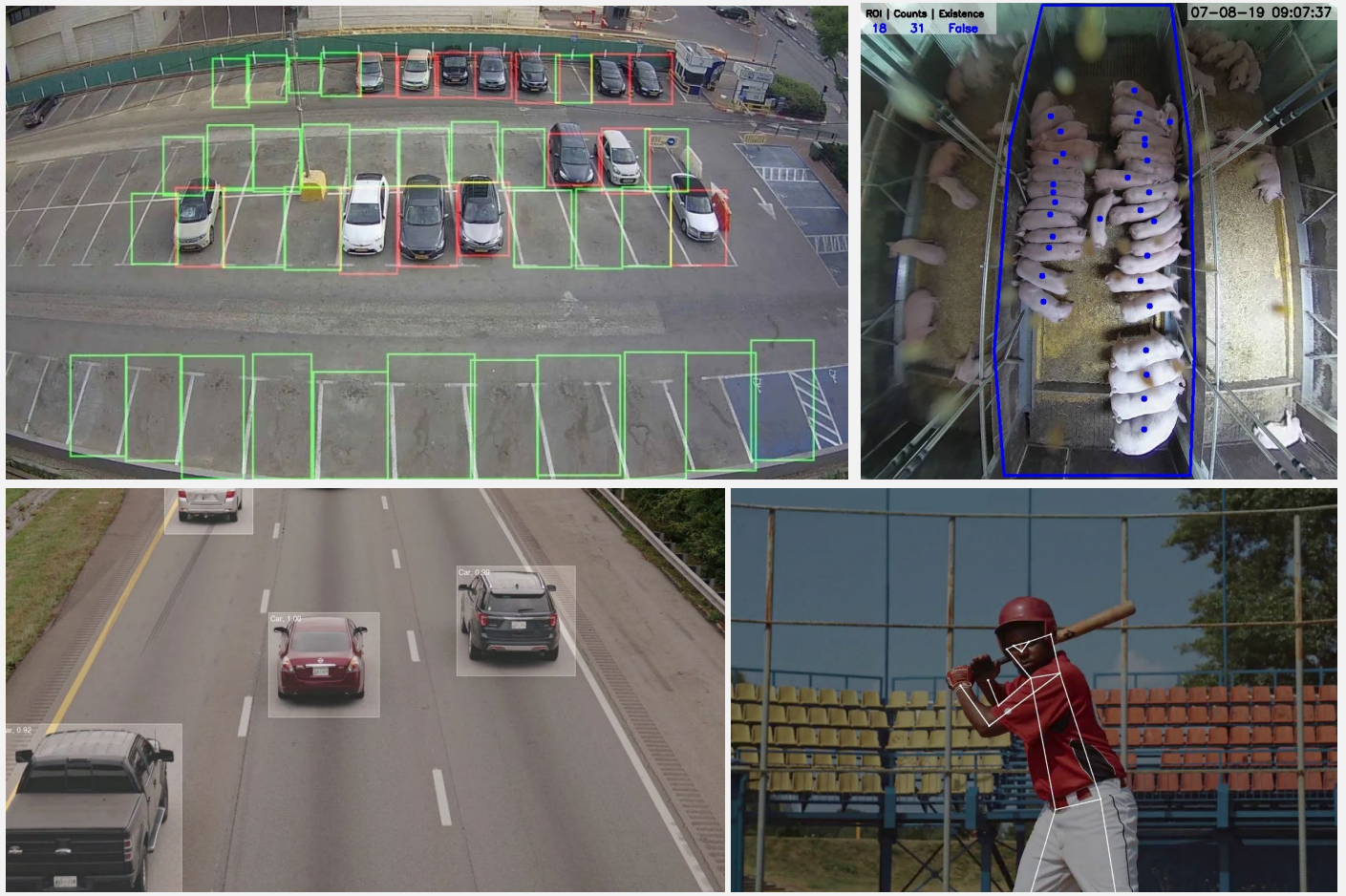




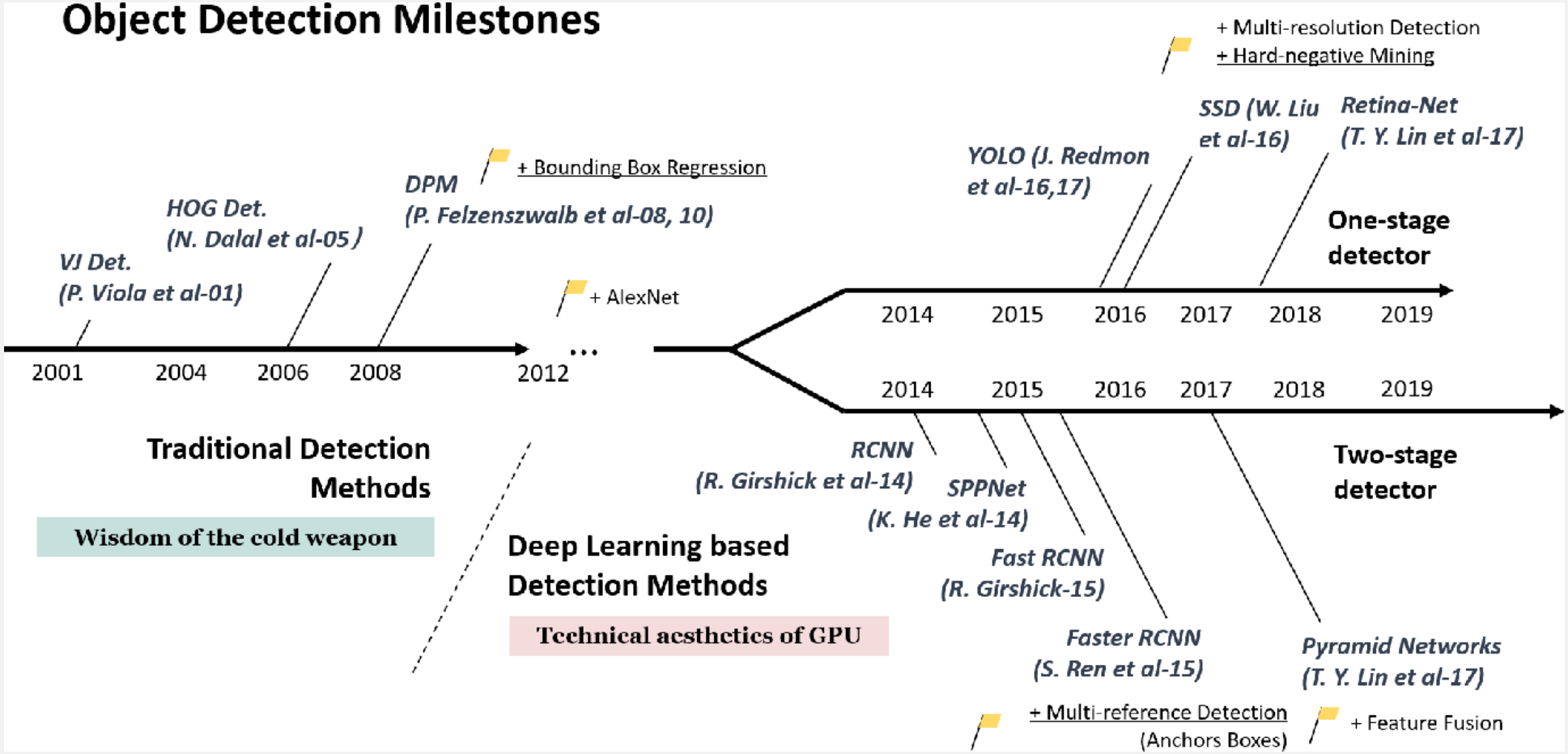




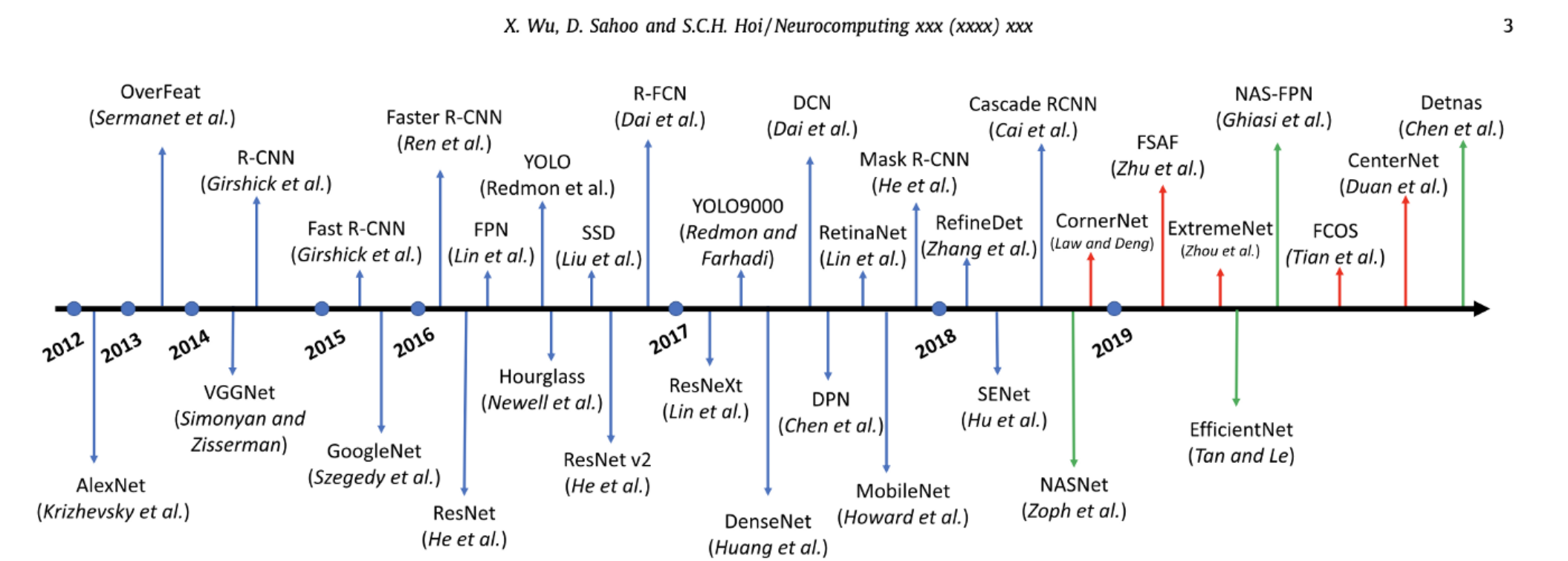
- **Applications**
  - **Agriculture**
    - Animal Monitoring
    - Farm Automation
    - Insect Detection
  - **Transportation**
    - Vehicle Classification
    - Moving Violations Detection
    - Driver Attentiveness Detection
  - **Retail**
    - Customer Tracking
    - Theft Detection
  - **Sports**
    - Player Pose Tracking
    - Ball Tracking
    - Goal-Line Technology



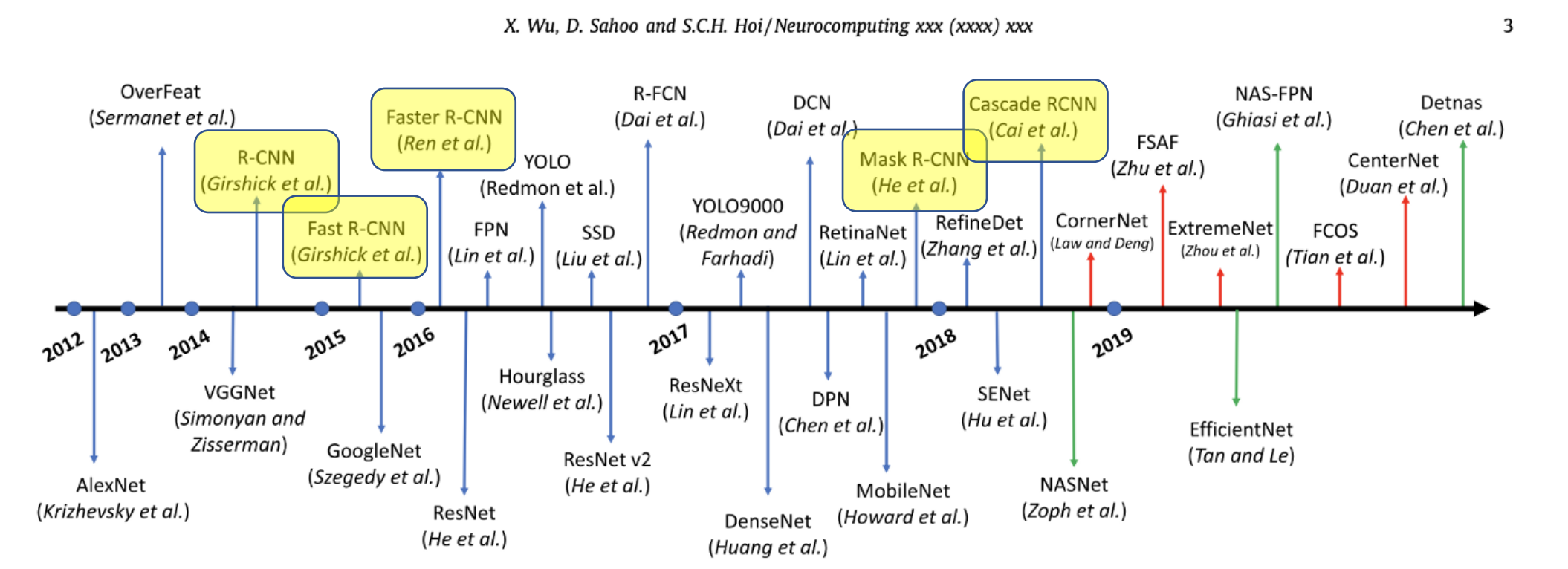
- **Concise Chronology of Recent Approaches Towards Object Detection**



- **Concise Chronology of Recent Approaches Towards Object Detection**



- **Region-Based Models**
  - Produce a set of bounding boxes as output, where each bounding box contains an object and also the category



- **R-CNNs**
  - **R-CNN**
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN

Rich feature hierarchies for accurate object detection and semantic segmentation
Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

Abstract

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012—achieving a mAP of 53.3%. Our approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. Since we combine region proposals with CNNs, we call our method R-CNN: *Regions with CNN features. We also compare R-CNN to OverFeat, a recently proposed sliding-window detector based on a similar CNN architecture. We find that R-CNN outperforms OverFeat by a large margin on the 200-class ILSVRC2013 detection dataset. Source code for the complete system is available at* http://www.cs.berkeley.edu/~rbg/rcnn.

R-CNN: *Regions with CNN features*

1. Input image 2. Extract region proposals (~2k) 3. Compute CNN features 4. Classify regions

Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset, R-CNN's mAP is 31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

1. Introduction

Features matter. The last decade of progress on various visual recognition tasks has been based considerably on the use of SIFT [29] and HOG [7]. But if we look at performance on the canonical visual recognition task, PASCAL VOC object detection [15], it is generally acknowledged that progress has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods.

SIFT and HOG are blockwise orientation histograms, a representation we could associate roughly with complex cells in V1, the first cortical area in the primate visual pathway. But we also know that recognition occurs several stages downstream, which suggests that there might be hierarchical, multi-stage processes for computing features that are even more informative for visual recognition.

Fukushima's "neocognitron" [19], a biologically-inspired hierarchical and shift-invariant model for pattern recognition, was an early attempt at just such a process. The neocognitron, however, lacked a supervised training algorithm. Building on Rumelhart et al. [33], LeCun et al. [26] showed that stochastic gradient descent via back-propagation was effective for training convolutional neural networks (CNNs), a class of models that extend the neocognitron.

CNNs saw heavy use in the 1990s (e.g., [27]), but then fell out of fashion with the rise of support vector machines. In 2012, Krizhevsky et al. [25] rekindled interest in CNNs by showing substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9, 10]. Their success resulted from training a large CNN on 1.2 million labeled images, together with a few twists on LeCun's CNN (e.g., $\max(x, 0)$ rectifying non-linearities and "dropout" regularization).

The significance of the ImageNet result was vigorously

- **R-CNNs :**
  - **R-CNN**
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN

Rich feature hierarchies for accurate object detection and semantic segmentation
Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

arXiv:1311.2524v5 [cs.CV] 22 Oct 2014

Abstract

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012—achieving a mAP of 53.3%. Our approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. Since we combine region proposals with CNNs, we call our method **R-CNN**: *Regions with CNN features. We also compare R-CNN to OverFeat, a recently proposed sliding-window detector based on a similar CNN architecture. We find that R-CNN outperforms OverFeat by a large margin on the 200-class ILSVRC2013 detection dataset. Source code for the complete system is available at* http://www.cs.berkeley.edu/~rbg/rcnn.

1. Introduction

Features matter. The last decade of progress on various visual recognition tasks has been based considerably on the use of SIFT [29] and HOG [7]. But if we look at performance on the canonical visual recognition task, PASCAL VOC object detection [15], it is generally acknowledged that progress has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods.

SIFT and HOG are blockwise orientation histograms, a representation we could associate roughly with complex cells in V1, the first cortical area in the primate visual pathway. But we also know that recognition occurs several stages downstream, which suggests that there might be hier-

R-CNN: *Regions with CNN features*

Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset, R-CNN's mAP is 31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

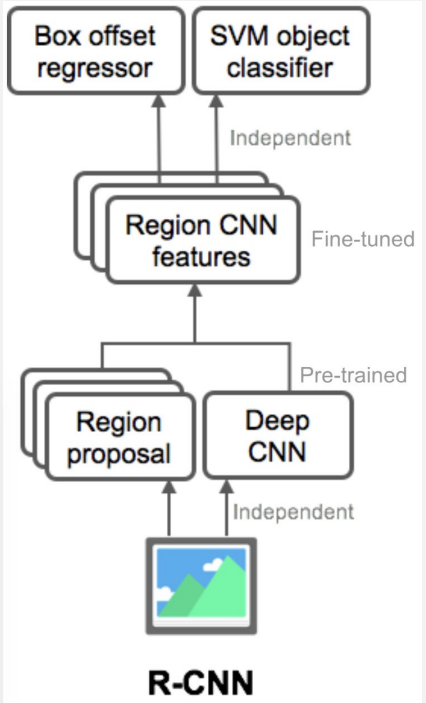
archical, multi-stage processes for computing features that are even more informative for visual recognition.

Fukushima's "neocognitron" [19], a biologically-inspired hierarchical and shift-invariant model for pattern recognition, was an early attempt at just such a process. The neocognitron, however, lacked a supervised training algorithm. Building on Rumelhart et al. [33], LeCun et al. [26] showed that stochastic gradient descent via back-propagation was effective for training convolutional neural networks (CNNs), a class of models that extend the neocognitron.

CNNs saw heavy use in the 1990s (e.g., [27]), but then fell out of fashion with the rise of support vector machines. In 2012, Krizhevsky et al. [25] rekindled interest in CNNs by showing substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9, 10]. Their success resulted from training a large CNN on 1.2 million labeled images, together with a few twists on LeCun's CNN (e.g., $\max(x, 0)$ rectifying non-linearities and "dropout" regularization).

The significance of the ImageNet result was vigorously

1



Object Detection (2013)

1. Scan the input image for possible objects using an algorithm called Selective Search, generating ~2000 **region proposals**
2. Run a **CNN** on top of each of these region proposals
3. Take the output of each **CNN** and feed it into:
   a. an **SVM** to classify the region
   b. a **linear regressor** to tighten the bounding box of the object, if such an object exists.

- **R-CNNs :**
  - **R-CNN**
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN

Rich feature hierarchies for accurate object detection and semantic segmentation
Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

arXiv:1311.2524v5 [cs.CV] 22 Oct 2014

Abstract

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012—achieving a mAP of 53.3%. Our approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. Since we combine region proposals with CNNs, we call our method **R-CNN**: *Regions with CNN features. We also compare R-CNN to OverFeat, a recently proposed sliding-window detector based on a similar CNN architecture. We find that R-CNN outperforms OverFeat by a large margin on the 200-class ILSVRC2013 detection dataset. Source code for the complete system is available at* http://www.cs.berkeley.edu/~rbg/rcnn.

1. Introduction

Features matter. The last decade of progress on various visual recognition tasks has been based considerably on the use of SIFT [29] and HOG [7]. But if we look at performance on the canonical visual recognition task, PASCAL VOC object detection [15], it is generally acknowledged that progress has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods.

SIFT and HOG are blockwise orientation histograms, a representation we could associate roughly with complex cells in V1, the first cortical area in the primate visual pathway. But we also know that recognition occurs several stages downstream, which suggests that there might be hier-

R-CNN: *Regions with CNN features*

1. Input image 2. Extract region proposals (~2k) 3. Compute CNN features 4. Classify regions

Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset, R-CNN's mAP is 31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

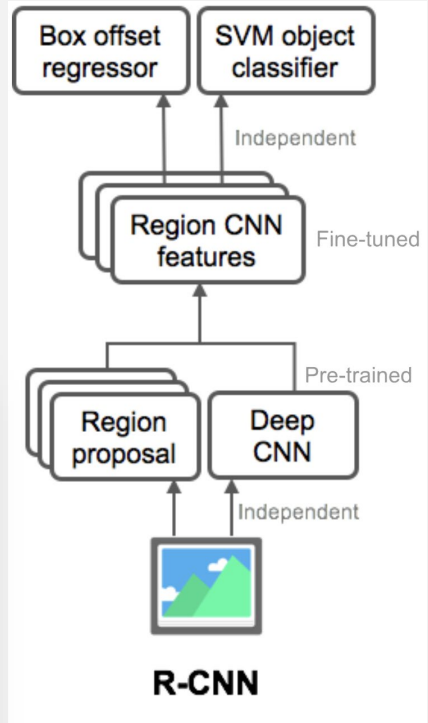
archical, multi-stage processes for computing features that are even more informative for visual recognition.

Fukushima's "neocognitron" [19], a biologically-inspired hierarchical and shift-invariant model for pattern recognition, was an early attempt at just such a process. The neocognitron, however, lacked a supervised training algorithm. Building on Rumelhart et al. [33], LeCun et al. [26] showed that stochastic gradient descent via back-propagation was effective for training convolutional neural networks (CNNs), a class of models that extend the neocognitron.

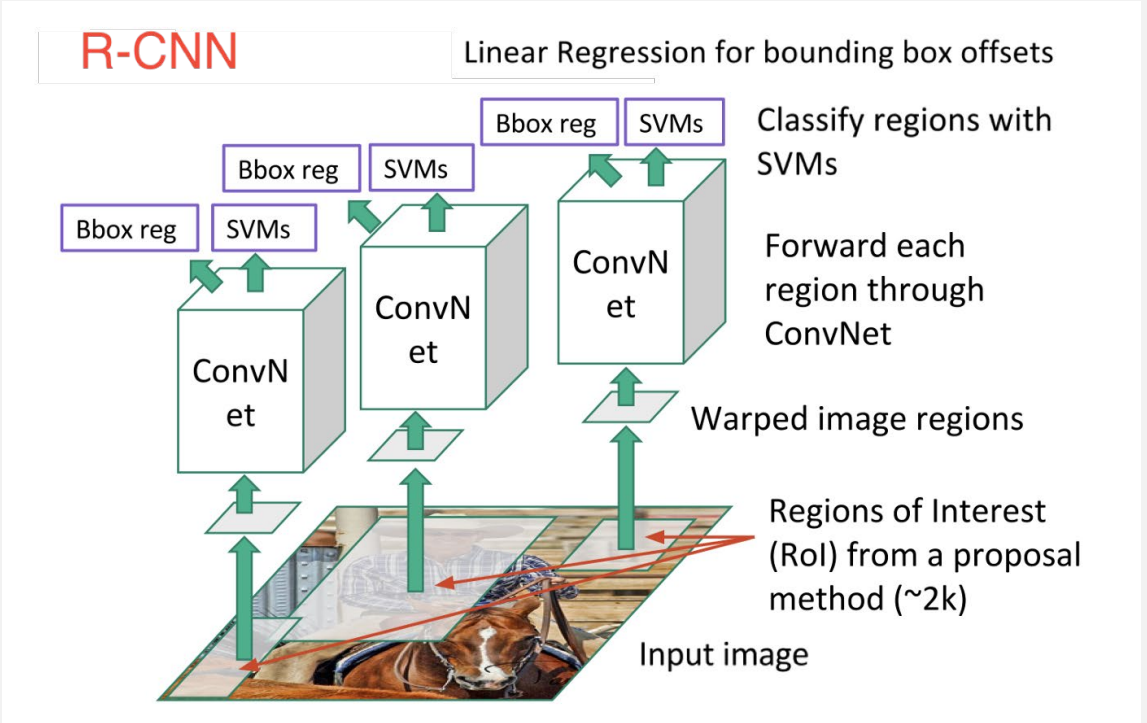
CNNs saw heavy use in the 1990s (e.g., [27]), but then fell out of fashion with the rise of support vector machines. In 2012, Krizhevsky et al. [25] rekindled interest in CNNs by showing substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9, 10]. Their success resulted from training a large CNN on 1.2 million labeled images, together with a few twists on LeCun's CNN (e.g., $\max(x,0)$ rectifying non-linearities and "dropout" regularization).

The significance of the ImageNet result was vigorously

1



Object Detection (2013)



- **R-CNNs :**
  - **R-CNN**
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN

Rich feature hierarchies for accurate object detection and semantic segmentation
Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

arXiv:1311.2524v5 [cs.CV] 22 Oct 2014

Abstract

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012—achieving a mAP of 53.3%. Our approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. Since we combine region proposals with CNNs, we call our method **R-CNN**: *Regions with CNN features. We also compare R-CNN to OverFeat, a recently proposed sliding-window detector based on a similar CNN architecture. We find that R-CNN outperforms OverFeat by a large margin on the 200-class ILSVRC2013 detection dataset. Source code for the complete system is available at* http://www.cs.berkeley.edu/~rbg/rcnn.

1. Introduction

Features matter. The last decade of progress on various visual recognition tasks has been based considerably on the use of SIFT [29] and HOG [7]. But if we look at performance on the canonical visual recognition task, PASCAL VOC object detection [15], it is generally acknowledged that progress has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods.

SIFT and HOG are blockwise orientation histograms, a representation we could associate roughly with complex cells in V1, the first cortical area in the primate visual pathway. But we also know that recognition occurs several stages downstreams, which suggests that there might be hier-

R-CNN: *Regions with CNN features*

1. Input image 2. Extract region proposals (~2k) 3. Compute CNN features 4. Classify regions

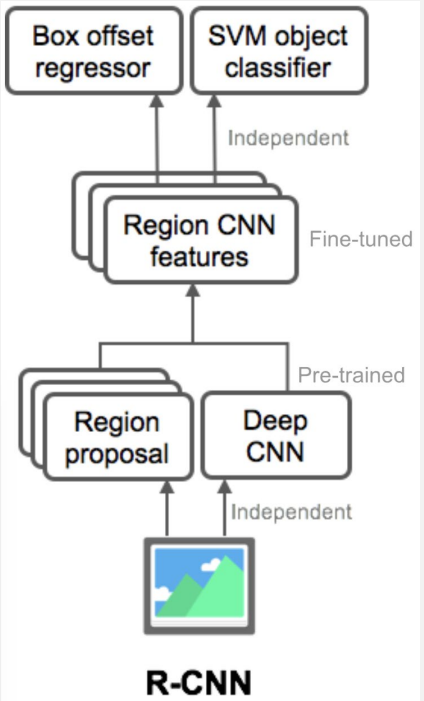
Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset, R-CNN's mAP is 31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

Fukushima's "neocognitron" [19], a biologically-inspired hierarchical and shift-invariant model for pattern recognition, was an early attempt at just such a process. The neocognitron, however, lacked a supervised training algorithm. Building on Rumelhart et al. [33], LeCun et al. [26] showed that stochastic gradient descent via back-propagation was effective for training convolutional neural networks (CNNs), a class of models that extend the neocognitron.

CNNs saw heavy use in the 1990s (e.g., [27]), but then fell out of fashion with the rise of support vector machines. In 2012, Krizhevsky et al. [25] rekindled interest in CNNs by showing substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9, 10]. Their success resulted from training a large CNN on 1.2 million labeled images, together with a few twists on LeCun's CNN (e.g., $\max(x, 0)$ rectifying non-linearities and "dropout" regularization).

The significance of the ImageNet result was vigorously

1



Object Detection (2013)

Problems with R-CNN

- Huge amount of time to train the network
  - (you would have to classify 2000 region proposals per image)
- Cannot be implemented real time
  - (47 seconds for each test image)
- The selective search algorithm is a fixed algorithm
  - (generation of bad candidate region proposals)

- **R-CNNs :**
  - R-CNN
  - **Fast R-CNN**
  - Faster R-CNN
  - Mask R-CNN

Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

Abstract

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) *for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License at* https://github.com/rbgirshick/fast-rcnn.

1. Introduction

Recently, deep ConvNets [14, 16] have significantly improved image classification [14] and object detection [9, 19] accuracy. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches (*e.g.*, [9, 11, 19, 25]) train models in multi-stage pipelines that are slow and inelegant.

Complexity arises because detection requires the accurate localization of objects, creating two primary challenges. First, numerous candidate object locations (often called "proposals") must be processed. Second, these candidates provide only rough localization that must be refined to achieve precise localization. Solutions to these problems often compromise speed, accuracy, or simplicity.

In this paper, we streamline the training process for state-of-the-art ConvNet-based object detectors [9, 11]. We propose a single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations.

The resulting method can train a very deep detection network (VGG16 [20]) 9× faster than R-CNN [9] and 3× faster than SPPnet [11]. At runtime, the detection network processes images in 0.3s (excluding object proposal time) while achieving top accuracy on PASCAL VOC 2012 [7] with a mAP of 66% (vs. 62% for R-CNN).[1]

1.1. R-CNN and SPPnet

The Region-based Convolutional Network method (R-CNN) [9] achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN, however, has notable drawbacks:

1. **Training is a multi-stage pipeline.** R-CNN first fine-tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.
2. **Training is expensive in space and time.** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 trainval set. These features require hundreds of gigabytes of storage.
3. **Object detection is slow.** At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

R-CNN is slow because it performs a ConvNet forward pass for each object proposal, without sharing computation. Spatial pyramid pooling networks (SPPnets) [11] were proposed to speed up R-CNN by sharing computation. The SPPnet method computes a convolutional feature map for the entire input image and then classifies each object proposal using a feature vector extracted from the shared feature map. Features are extracted for a proposal by max-pooling the portion of the feature map inside the proposal into a fixed-size output (*e.g.*, 6 × 6). Multiple output sizes are pooled and then concatenated as in spatial pyramid pooling [15]. SPPnet accelerates R-CNN by 10 to 100× at test time. Training time is also reduced by 3× due to faster proposal feature extraction.

[1] All timings use one Nvidia K40 GPU overclocked to 875 MHz.

- **R-CNNs :**
  - R-CNN
  - **Fast R-CNN**
  - Faster R-CNN
  - Mask R-CNN

Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

Abstract

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) *for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License at* https://github.com/rbgirshick/fast-rcnn.

1. Introduction

Recently, deep ConvNets [14, 16] have significantly improved image classification [14] and object detection [9, 19] accuracy. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches (e.g., [9, 11, 19, 25]) train models in multi-stage pipelines that are slow and inelegant.

Complexity arises because detection requires the accurate localization of objects, creating two primary challenges. First, numerous candidate object locations (often called "proposals") must be processed. Second, these candidates provide only rough localization that must be refined to achieve precise localization. Solutions to these problems often compromise speed, accuracy, or simplicity.

In this paper, we streamline the training process for state-of-the-art ConvNet-based object detectors [9, 11]. We propose a single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations.

The resulting method can train a very deep detection network (VGG16 [20]) 9× faster than R-CNN [9] and 3× faster than SPPnet [11]. At runtime, the detection network processes images in 0.3s (excluding object proposal time) while achieving top accuracy on PASCAL VOC 2012 [7] with a mAP of 66% (vs. 62% for R-CNN).[1]

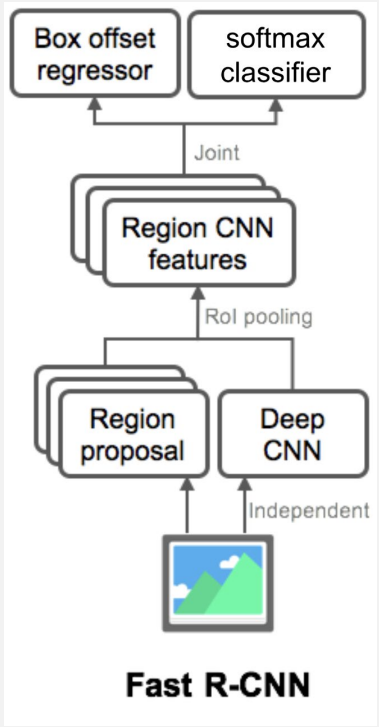
1.1. R-CNN and SPPnet

The Region-based Convolutional Network method (R-CNN) [9] achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN, however, has notable drawbacks:

1. **Training is a multi-stage pipeline.** R-CNN first fine-tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.
2. **Training is expensive in space and time.** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 trainval set. These features require hundreds of gigabytes of storage.
3. **Object detection is slow.** At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

R-CNN is slow because it performs a ConvNet forward pass for each object proposal, without sharing computation. Spatial pyramid pooling networks (SPPnets) [11] were proposed to speed up R-CNN by sharing computation. The SPPnet method computes a convolutional feature map for the entire input image and then classifies each object proposal using a feature vector extracted from the shared feature map. Features are extracted for a proposal by max-pooling the portion of the feature map inside the proposal into a fixed-size output (e.g., 6×6). Multiple output sizes are pooled and then concatenated as in spatial pyramid pooling [15]. SPPnet accelerates R-CNN by 10 to 100× at test time. Training time is also reduced by 3× due to faster proposal feature extraction.

[1] All timings use one Nvidia K40 GPU overclocked to 875 MHz.



Object Detection (2015)

1. Feature extraction over the image before proposing regions
2. Replacing the SVM with a SoftMax layer

- **R-CNNs :**
  - R-CNN
  - **Fast R-CNN**
  - Faster R-CNN
  - Mask R-CNN

Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

Abstract

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License at https://github.com/rbgirshick/fast-rcnn.

1. Introduction

Recently, deep ConvNets [14, 16] have significantly improved image classification [14] and object detection [9, 19] accuracy. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches (e.g., [9, 11, 19, 25]) train models in multi-stage pipelines that are slow and inelegant.

Complexity arises because detection requires the accurate localization of objects, creating two primary challenges. First, numerous candidate object locations (often called "proposals") must be processed. Second, these candidates provide only rough localization that must be refined to achieve precise localization. Solutions to these problems often compromise speed, accuracy, or simplicity.

In this paper, we streamline the training process for state-of-the-art ConvNet-based object detectors [9, 11]. We propose a single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations.

The resulting method can train a very deep detection network (VGG16 [20]) 9× faster than R-CNN [9] and 3× faster than SPPnet [11]. At runtime, the detection network processes images in 0.3s (excluding object proposal time) while achieving top accuracy on PASCAL VOC 2012 [7] with a mAP of 66% (vs. 62% for R-CNN).[1]

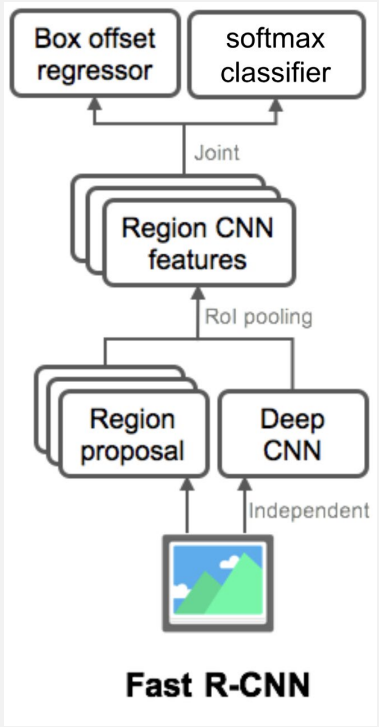
1.1. R-CNN and SPPnet

The Region-based Convolutional Network method (R-CNN) [9] achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN, however, has notable drawbacks:

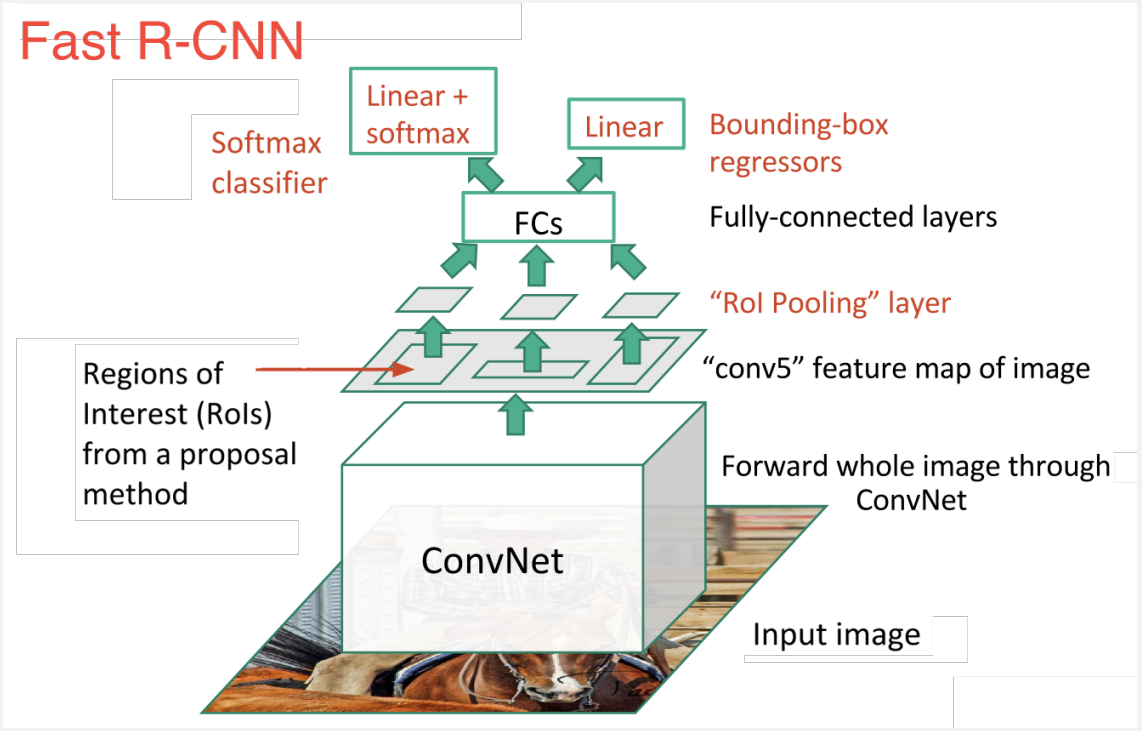
1. **Training is a multi-stage pipeline.** R-CNN first fine-tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.
2. **Training is expensive in space and time.** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 trainval set. These features require hundreds of gigabytes of storage.
3. **Object detection is slow.** At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

R-CNN is slow because it performs a ConvNet forward pass for each object proposal, without sharing computation. Spatial pyramid pooling networks (SPPnets) [11] were proposed to speed up R-CNN by sharing computation. The SPPnet method computes a convolutional feature map for the entire input image and then classifies each object proposal using a feature vector extracted from the shared feature map. Features are extracted for a proposal by max-pooling the portion of the feature map inside the proposal into a fixed-size output (e.g., 6×6). Multiple output sizes are pooled and then concatenated as in spatial pyramid pooling [15]. SPPnet accelerates R-CNN by 10 to 100× at test time. Training time is also reduced by 3× due to faster proposal feature extraction.

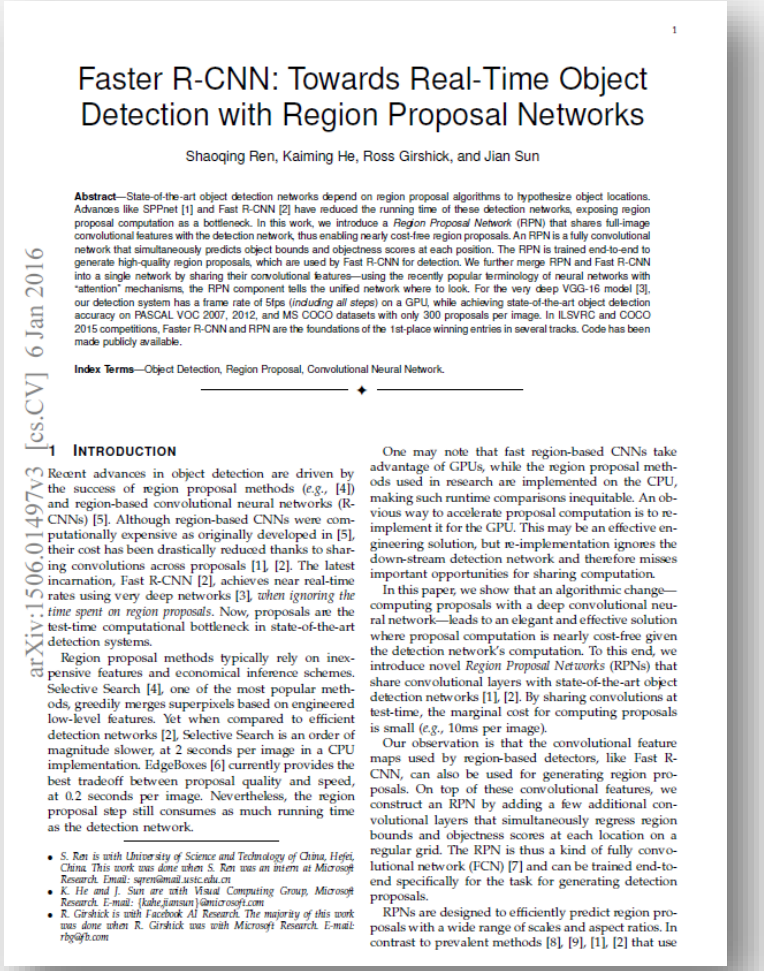
[1] All timings use one Nvidia K40 GPU overclocked to 875 MHz.



Object Detection (2015)



- **R-CNNs :**
  - R-CNN
  - Fast R-CNN
  - **Faster R-CNN**
  - Mask R-CNN



Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with "attention" mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (*including all steps*) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

Index Terms—Object Detection, Region Proposal, Convolutional Neural Network.

1 Introduction

Recent advances in object detection are driven by the success of region proposal methods (*e.g.*, [4]) and region-based convolutional neural networks (R-CNNs) [5]. Although region-based CNNs were computationally expensive as originally developed in [5], their cost has been drastically reduced thanks to sharing convolutions across proposals [1], [2]. The latest incarnation, Fast R-CNN [2], achieves near real-time rates using very deep networks [3], *when ignoring the time spent on region proposals*. Now, proposals are the test-time computational bottleneck in state-of-the-art detection systems.

Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search [4], one of the most popular methods, greedily merges superpixels based on engineered low-level features. Yet when compared to efficient detection networks [2], Selective Search is an order of magnitude slower, at 2 seconds per image in a CPU implementation. EdgeBoxes [6] currently provides the best tradeoff between proposal quality and speed, at 0.2 seconds per image. Nevertheless, the region proposal step still consumes as much running time as the detection network.

- *S. Ren is with University of Science and Technology of China, Hefei, China. This work was done when S. Ren was an intern at Microsoft Research. Email: sqren@mail.ustc.edu.cn*
- *K. He and J. Sun are with Visual Computing Group, Microsoft Research. E-mail: {kahe,jiansun}@microsoft.com*
- *R. Girshick is with Facebook AI Research. The majority of this work was done when R. Girshick was with Microsoft Research. E-mail: rbg@fb.com*

One may note that fast region-based CNNs take advantage of GPUs, while the region proposal methods used in research are implemented on the CPU, making such runtime comparisons inequitable. An obvious way to accelerate proposal computation is to re-implement it for the GPU. This may be an effective engineering solution, but re-implementation ignores the down-stream detection network and therefore misses important opportunities for sharing computation.

In this paper, we show that an algorithmic change—computing proposals with a deep convolutional neural network—leads to an elegant and effective solution where proposal computation is nearly cost-free given the detection network's computation. To this end, we introduce novel *Region Proposal Networks* (RPNs) that share convolutional layers with state-of-the-art object detection networks [1], [2]. By sharing convolutions at test-time, the marginal cost for computing proposals is small (*e.g.*, 10ms per image).

Our observation is that the convolutional feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals. On top of these convolutional features, we construct an RPN by adding a few additional convolutional layers that simultaneously regress region bounds and objectness scores at each location on a regular grid. The RPN is thus a kind of fully convolutional network (FCN) [7] and can be trained end-to-end specifically for the task for generating detection proposals.

RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios. In contrast to prevalent methods [8], [9], [1], [2] that use

- **R-CNNs :**
  - R-CNN
  - Fast R-CNN
  - **Faster R-CNN**
  - Mask R-CNN

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with "attention" mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (*including all steps*) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

Index Terms—Object Detection, Region Proposal, Convolutional Neural Network.

1 Introduction

Recent advances in object detection are driven by the success of region proposal methods (*e.g.*, [4]) and region-based convolutional neural networks (R-CNNs) [5]. Although region-based CNNs were computationally expensive as originally developed in [5], their cost has been dramatically reduced thanks to sharing convolutions across proposals [1], [2]. The latest incarnation, Fast R-CNN [2], achieves near real-time rates using very deep networks [3], when (*ignoring the time spent on region proposals*). Now, proposals are the test-time computational bottleneck in state-of-the-art detection systems.

Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search [4], one of the most popular methods, greedily merges superpixels based on engineered low-level features. Yet when compared to efficient detection networks [2], Selective Search is an order of magnitude slower, at 2 seconds per image in a CPU implementation. EdgeBoxes [6] currently provides the best tradeoff between proposal quality and speed, at 0.2 seconds per image. Nevertheless, the region proposal step still consumes as much running time as the detection network.

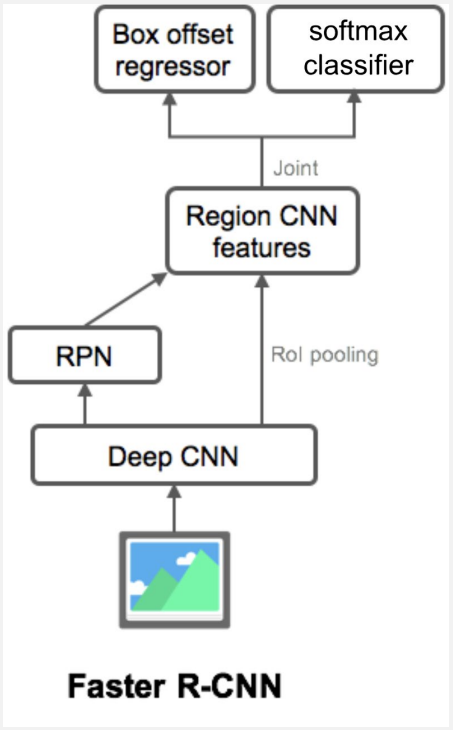
One may note that fast region-based CNNs take advantage of GPUs, while the region proposal methods used in research are implemented on the CPU, making such runtime comparisons inequitable. An obvious way to accelerate proposal computation is to re-implement it for the GPU. This may be an effective engineering solution, but re-implementation ignores the down-stream detection network and therefore misses important opportunities for sharing computation.

In this paper, we show that an algorithmic change—computing proposals with a deep convolutional neural network—leads to an elegant and effective solution where proposal computation is nearly cost-free given the detection network's computation. To this end, we introduce novel Region Proposal Networks (RPNs) that share convolutional layers with state-of-the-art object detection networks [1], [2]. By sharing convolutions at test-time, the marginal cost for computing proposals is small (*e.g.*, 10ms per image).

Our observation is that the convolutional feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals. On top of these convolutional features, we construct an RPN by adding a few additional convolutional layers that simultaneously regress region bounds and objectness scores at each location on a regular grid. The RPN is thus a kind of fully convolutional network (FCN) [7] and can be trained end-to-end specifically for the task for generating detection proposals.

RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios. In contrast to prevalent methods [8], [9], [1], [2] that use

• Y. Ren is with University of Science and Technology of China, Hefei, China. S. Ren and J. Sun are with Visual Computing Group, Microsoft Research. Email: sqren@mail.ustc.edu.cn.
• K. He and J. Sun are with Visual Computing Group, Microsoft Research. E-mail: {kahe,jiansun}@microsoft.com
• R. Girshick is with Facebook AI Research. The majority of this work was done when R. Girshick was with Microsoft Research. E-mail: rbg@fb.com



Object Detection (2015)

- One of the drawbacks of R-CNN and Fast R-CNN:
  - Slow selective search algorithm for region proposal
- Faster R-CNN:
  - introducing something called Region Proposal Network (RPN)

Faster R-CNN = RPN + Fast R-CNN

- **R-CNNs :**
  - R-CNN
  - Fast R-CNN
  - **Faster R-CNN**
  - Mask R-CNN

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with "attention" mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (*including all steps*) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

Index Terms—Object Detection, Region Proposal, Convolutional Neural Network.

1 Introduction

Recent advances in object detection are driven by the success of region proposal methods (*e.g.*, [4]) and region-based convolutional neural networks (R-CNNs) [5]. Although region-based CNNs were computationally expensive as originally developed in [5], their cost has been dramatically reduced thanks to sharing convolutions across proposals [1], [2]. The latest incarnation, Fast R-CNN [2], achieves near real-time rates using very deep networks [3], when (*ignoring the time spent on region proposals*). Now, proposals are the test-time computational bottleneck in state-of-the-art detection systems.

Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search [4], one of the most popular methods, greedily merges superpixels based on engineered low-level features. Yet when compared to efficient detection networks [2], Selective Search is an order of magnitude slower, at 2 seconds per image in a CPU implementation. EdgeBoxes [6] currently provides the best tradeoff between proposal quality and speed, at 0.2 seconds per image. Nevertheless, the region proposal step still consumes as much running time as the detection network.

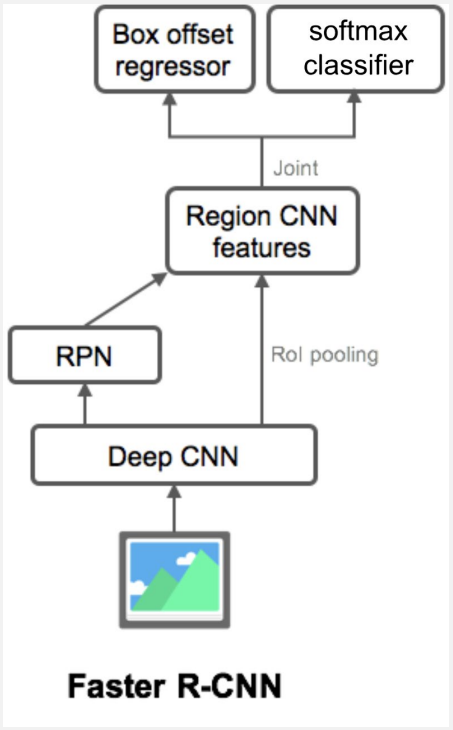
One may note that fast region-based CNNs take advantage of GPUs, while the region proposal methods used in research are implemented on the CPU, making such runtime comparisons inequitable. An obvious way to accelerate proposal computation is to re-implement it for the GPU. This may be an effective engineering solution, but re-implementation ignores the down-stream detection network and therefore misses important opportunities for sharing computation.

In this paper, we show that an algorithmic change—computing proposals with a deep convolutional neural network—leads to an elegant and effective solution where proposal computation is nearly cost-free given the detection network's computation. To this end, we introduce novel Region Proposal Networks (RPNs) that share convolutional layers with state-of-the-art object detection networks [1], [2]. By sharing convolutions at test-time, the marginal cost for computing proposals is small (*e.g.*, 10ms per image).

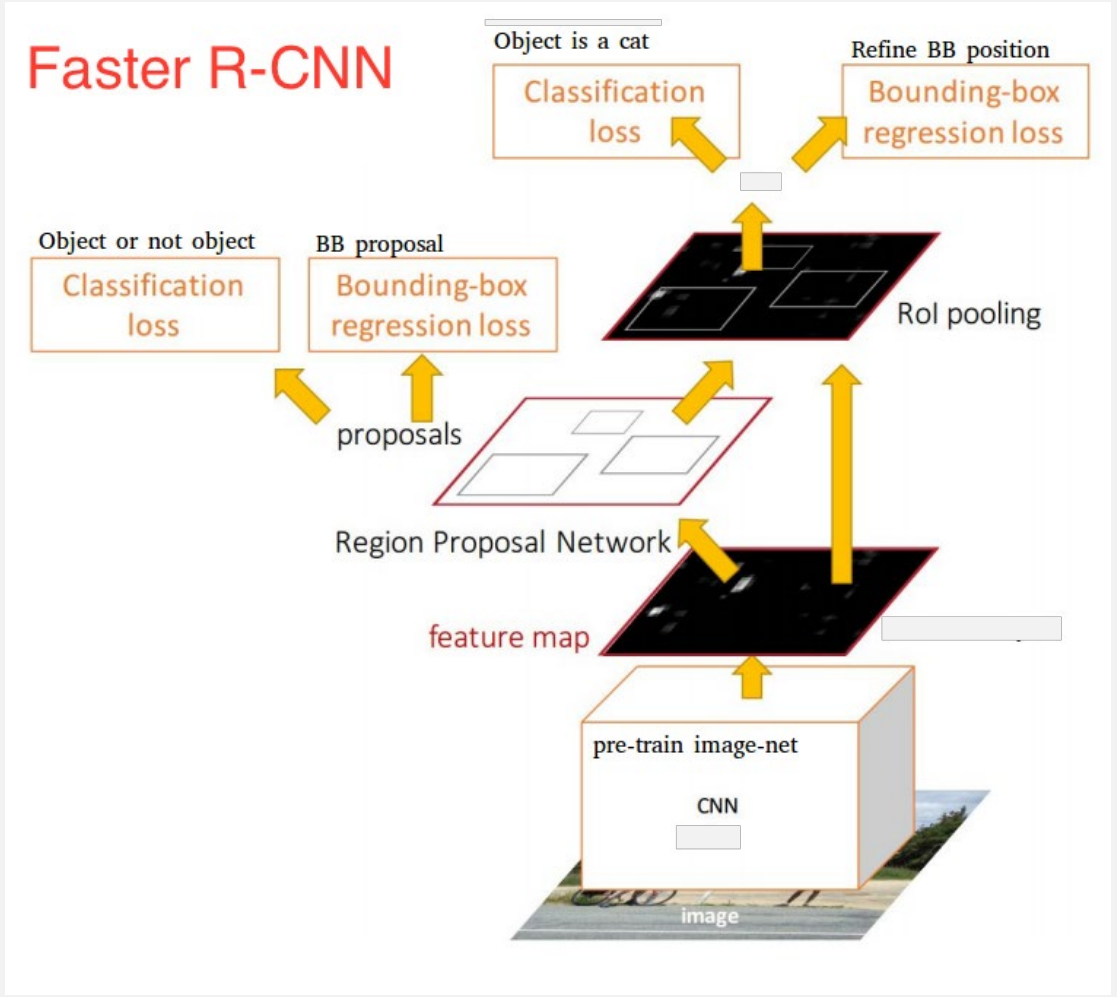
Our observation is that the convolutional feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals. On top of these convolutional features, we construct an RPN by adding a few additional convolutional layers that simultaneously regress region bounds and objectness scores at each location on a regular grid. The RPN is thus a kind of fully convolutional network (FCN) [7] and can be trained end-to-end specifically for the task for generating detection proposals.

RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios. In contrast to prevalent methods [8], [9], [1], [2] that use

• S. Ren is with University of Science and Technology of China, Hefei, China. This work was done when he was an intern at Microsoft Research. E-mail: sqren@mail.ustc.edu.cn.
• K. He and J. Sun are with Visual Computing Group, Microsoft Research. E-mail: {kahe,jiansun}@microsoft.com
• R. Girshick is with Facebook AI Research. The majority of this work was done when R. Girshick was with Microsoft Research. E-mail: rbg@fb.com



Object Detection (2015)



- **R-CNNs :**
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - **Mask R-CNN**

Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick
Facebook AI Research (FAIR)

Abstract

We present a conceptually simple, flexible, and general framework for object instance segmentation. Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework. We show top results in all three tracks of the COCO suite of challenges, including instance segmentation, bounding-box object detection, and person keypoint detection. Without bells and whistles, Mask R-CNN outperforms all existing, single-model entries on every task, including the COCO 2016 challenge winners. We hope our simple and effective approach will serve as a solid baseline and help ease future research in instance-level recognition. Code has been made available at: https://github.com/facebookresearch/Detectron.

1. Introduction

The vision community has rapidly improved object detection and semantic segmentation results over a short period of time. In large part, these advances have been driven by powerful baseline systems, such as the Fast/Faster R-CNN [12, 36] and Fully Convolutional Network (FCN) [30] frameworks for object detection and semantic segmentation, respectively. These methods are conceptually intuitive and offer flexibility and robustness, together with fast training and inference time. Our goal in this work is to develop a comparably enabling framework for *instance segmentation*.

Instance segmentation is challenging because it requires the correct detection of all objects in an image while also precisely segmenting each instance. It therefore combines elements from the classical computer vision tasks of *object detection*, where the goal is to classify individual objects and localize each using a bounding box, and *semantic segmentation*, where the goal is to classify each pixel into a fixed set of categories without differentiating object instances.[1] Given this, one might expect a complex method is required to achieve good results. However, we show that a surprisingly simple, flexible, and fast system can surpass prior state-of-the-art instance segmentation results.

Figure 1. The **Mask R-CNN** framework for instance segmentation.

Our method, called *Mask R-CNN*, extends Faster R-CNN [36] by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in *parallel* with the existing branch for classification and bounding box regression (Figure 1). The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner. Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

In principle Mask R-CNN is an intuitive extension of Faster R-CNN, yet constructing the mask branch properly is critical for good results. Most importantly, Faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is most evident in how *RoIPool* [18, 12], the *de facto* core operation for attending to instances, performs coarse spatial quantization for feature extraction. To fix the misalignment, we propose a simple, quantization-free layer, called *RoIAlign*, that faithfully preserves exact spatial locations. Despite being

[1] Following common terminology, we use *object detection* to denote detection via *bounding boxes*, not masks, and *semantic segmentation* to denote per-pixel classification without differentiating instances. Yet we note that *instance segmentation* is both semantic and a form of detection.

- **R-CNNs :**
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - **Mask R-CNN**

Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick

Facebook AI Research (FAIR)

Abstract

We present a conceptually simple, flexible, and general framework for object instance segmentation. Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework. We show top results in all three tracks of the COCO suite of challenges, including instance segmentation, bounding-box object detection, and person keypoint detection. Without bells and whistles, Mask R-CNN outperforms all existing, single-model entries on every task, including the COCO 2016 challenge winners. We hope our simple and effective approach will serve as a solid baseline and help ease future research in instance-level recognition. Code has been made available at: https://github.com/facebookresearch/Detectron.

1. Introduction

The vision community has rapidly improved object detection and semantic segmentation results over a short period of time. In large part, these advances have been driven by powerful baseline systems, such as the Fast/Faster R-CNN [12, 36] and Fully Convolutional Network (FCN) [30] frameworks for object detection and semantic segmentation, respectively. These methods are conceptually intuitive and offer flexibility and robustness, together with fast training and inference time. Our goal in this work is to develop a comparably enabling framework for instance segmentation.

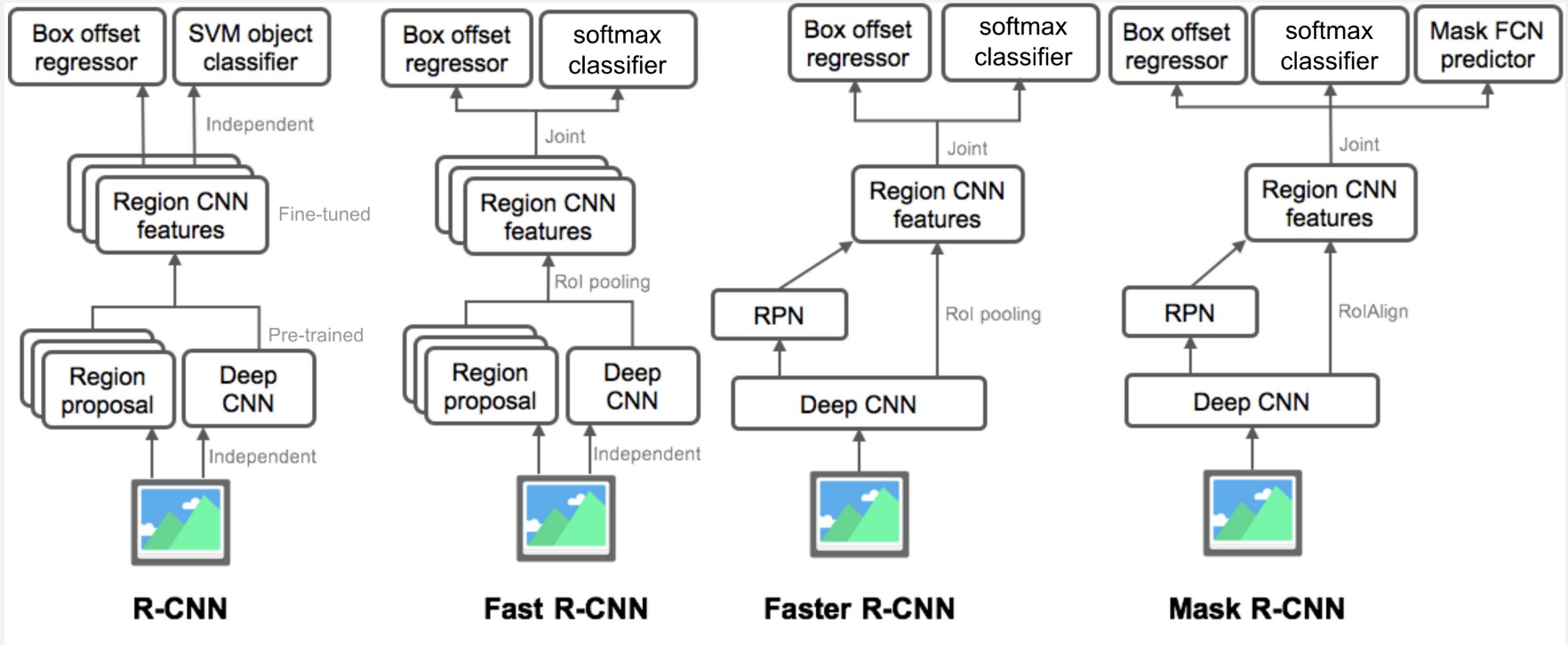
Instance segmentation is challenging because it requires the correct detection of all objects in an image while also precisely segmenting each instance. It therefore combines elements from the classical computer vision tasks of object detection, where the goal is to classify individual objects and localize each using a bounding box, and semantic segmentation, where the goal is to classify each pixel into a fixed set of categories without differentiating object instances.[1] Given this, one might expect a complex method is required to achieve good results. However, we show that a surprisingly simple, flexible, and fast system can surpass prior state-of-the-art instance segmentation results.

Our method, called Mask R-CNN, extends Faster R-CNN [36] by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression (Figure 1). The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner. Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

In principle Mask R-CNN is an intuitive extension of Faster R-CNN, yet constructing the mask branch properly is critical for good results. Most importantly, Faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is most evident in how RoIPool [18, 12], the de facto core operation for attending to instances, performs coarse spatial quantization for feature extraction. To fix the misalignment, we propose a simple, quantization-free layer, called RoIAlign, that faithfully preserves exact spatial locations. Despite being

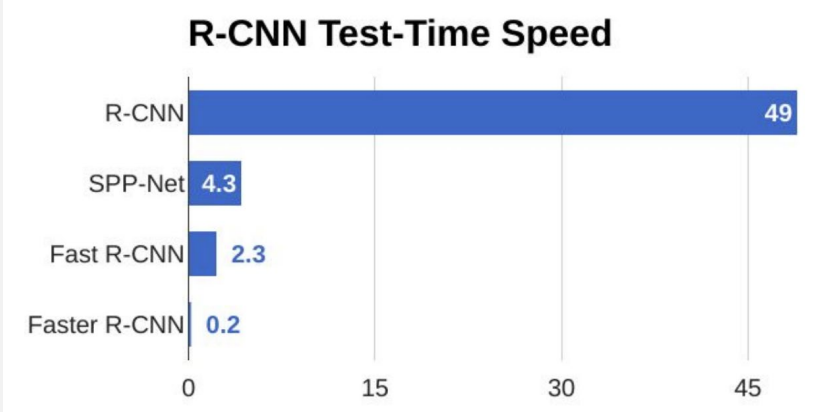
[1] Following common terminology, we use *object detection* to denote detection via bounding boxes, not masks, and *semantic segmentation* to denote per-pixel classification without differentiating instances. Yet we note that instance segmentation is both semantic and a form of detection.

arXiv:1703.06870v3 [cs.CV] 24 Jan 2018



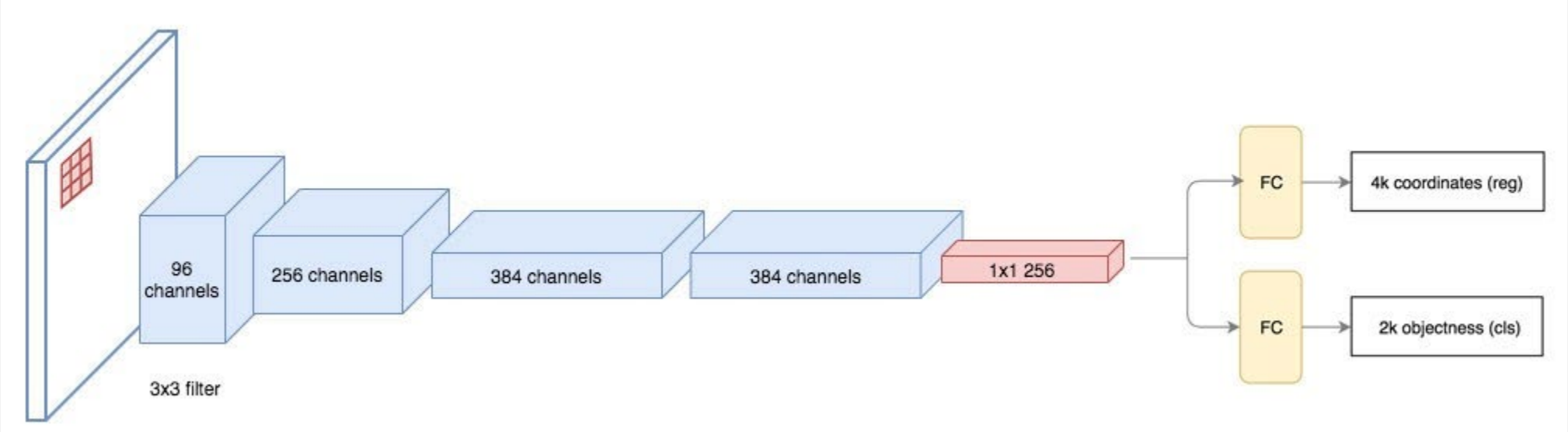
Object Detection (2013) | Object Detection (2015) | Object Detection (2015) | Instance Segmentation (2017)

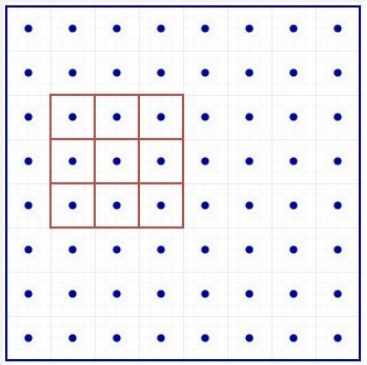
- **R-CNNs :**
  - **R-CNN**
  - **Fast R-CNN**
  - **Faster R-CNN**
  - **Mask R-CNN**



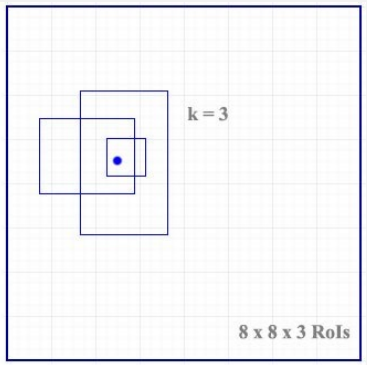
	R-CNN	Fast R-CNN	Faster R-CNN
Region Proposals Method	Selective search	Selective search	Region proposal network
Prediction Timing	40-50 sec	2 seconds	0.2 seconds
computation	High computation time	High computation time	Low computation time
The mAP on Pascal VOC 2007 Test Dataset(%)	58.5	66.9 (when trained with VOC 2007 only)	69.9(when trained with VOC 2007 only)
		70.0 (when trained with VOC 2007 and 2012 both)	
The mAP on Pascal VOC 2012 Test Dataset (%)	53.3	65.7 (when trained with VOC 2012 only)	67.0(when trained with VOC 2012 only)
		68.4 (when trained with VOC 2007 and 2012 both)	70.4 (when trained with VOC 2007 and 2012 both)
			75.9(when trained with VOC 2007 and 2012 and COCO)

- **RPN** (Region Proposal Network)

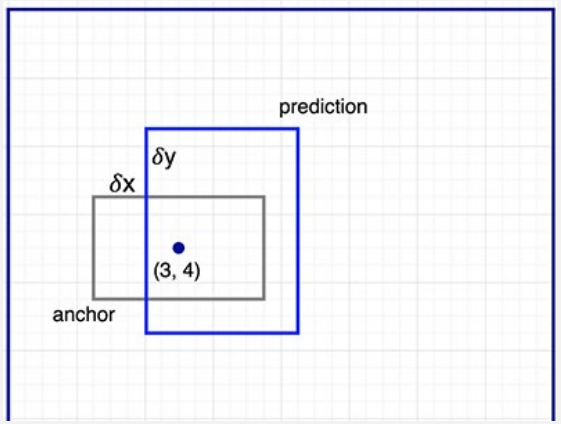




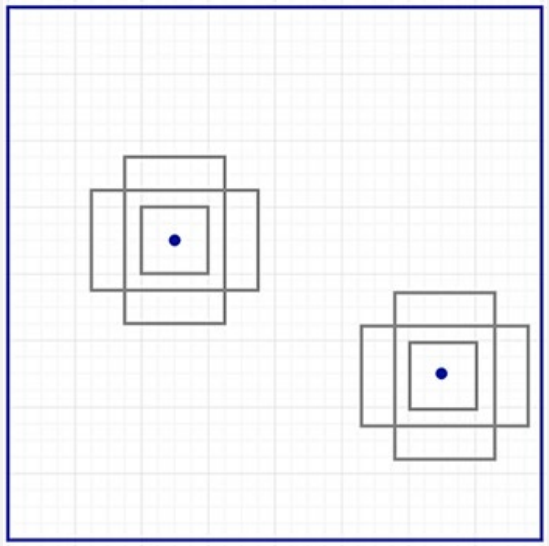
8 x 8 feature maps

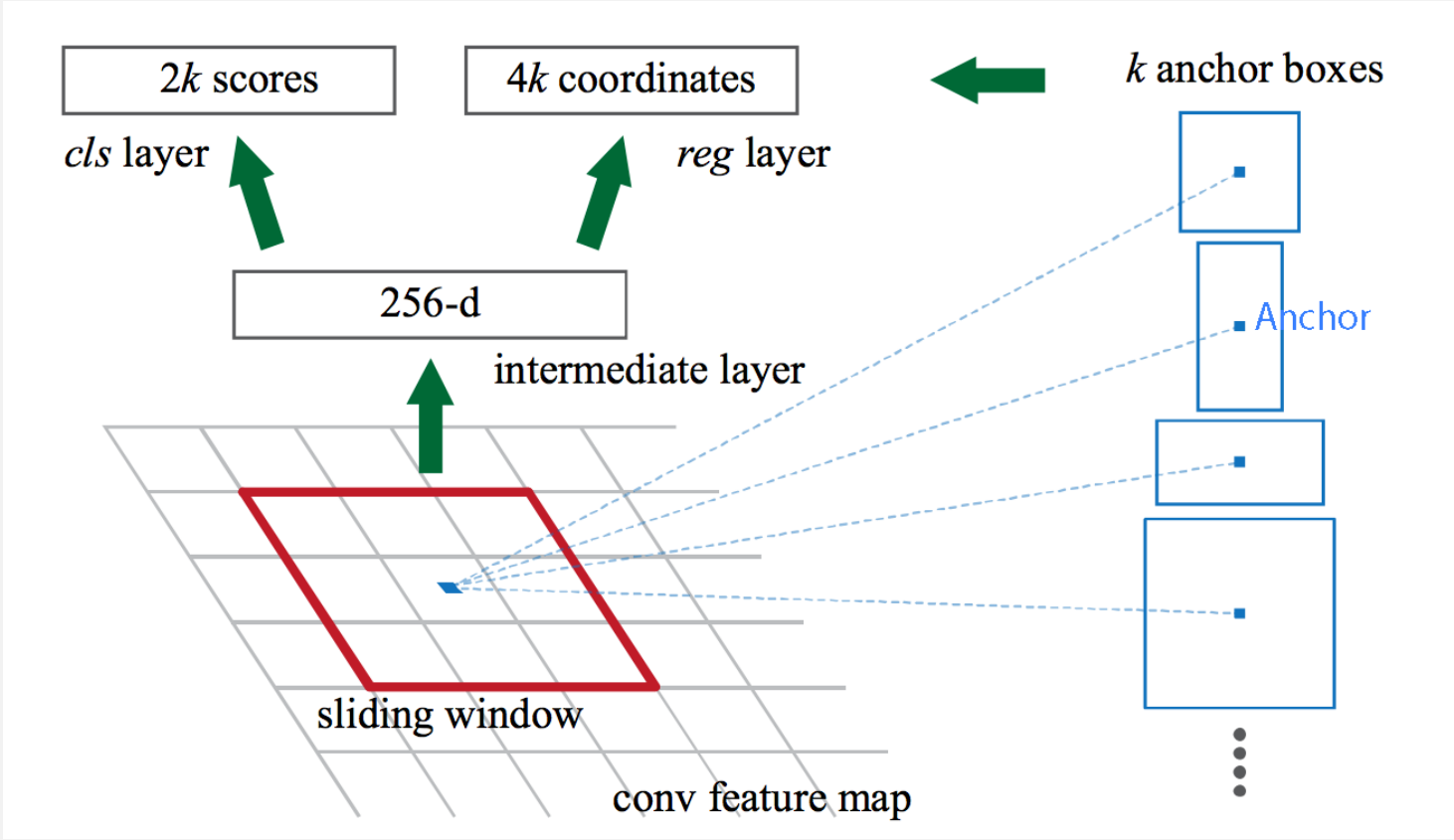


3 proposals for each location

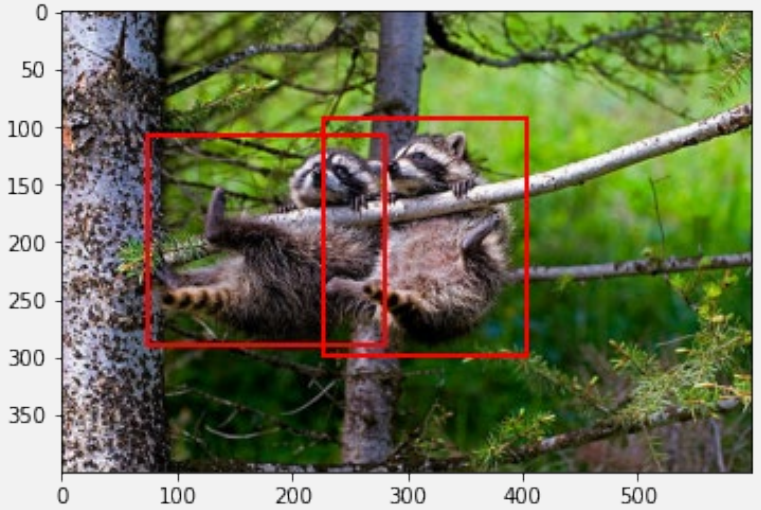


- **RPN** (Region Proposal Network)
  - Number of guesses : *k*
    - Objectiveness score
    - Bounding-box coordinates
    - Anchors (Default boundary boxes)



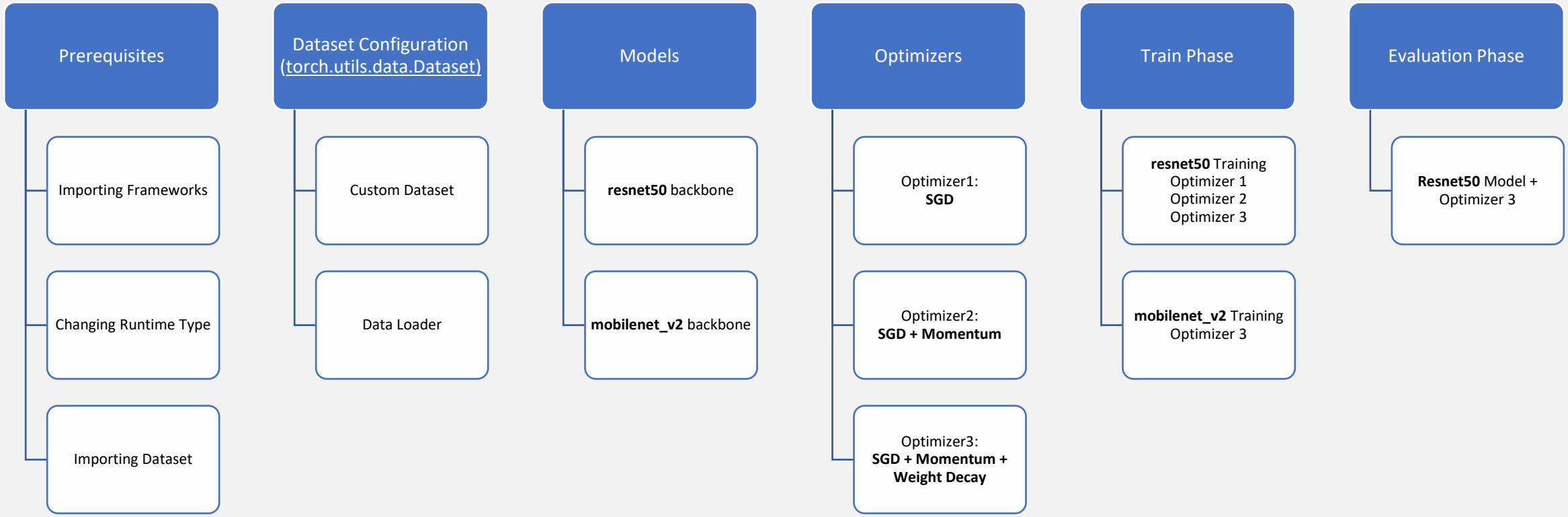


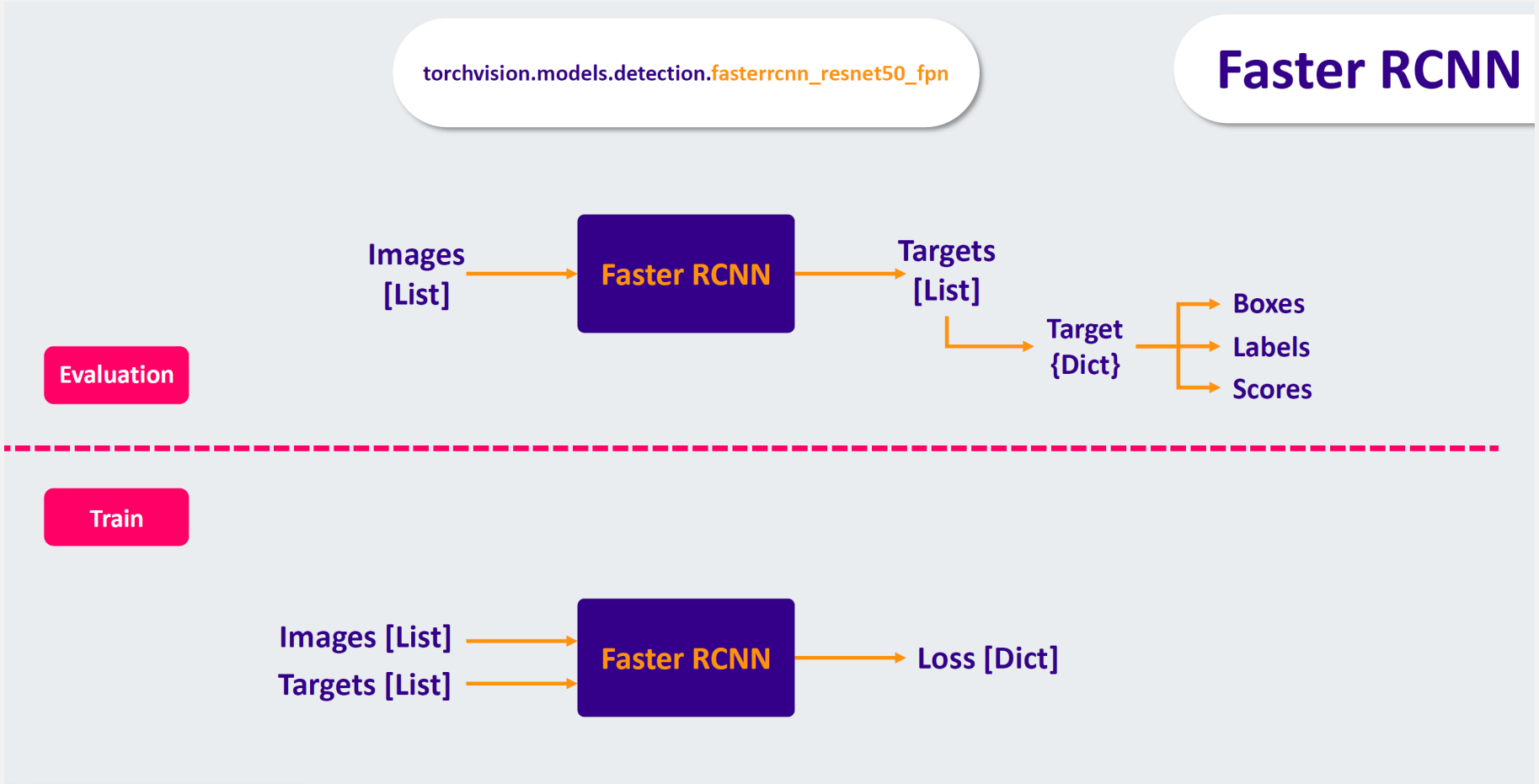
- **Code Summary**
  - Programming language : Python
  - IDE : Google Colab / Jupyter Notebook Format (ipynb) [Click Here]
  - ML Framework : Pytorch
  - Runtime Type : GPU
  - **Dataset** : Raccooon dataset
    - Size of Dataset : 200 (160 : Train Phase / 40 : Validation Phase)
    - Based on Google Images and Pixabay
    - Self-Annotated
    - Resolution : Width : 600 / Height : 400
    - Bounding Box info : (xmin , ymin , xmax , ymax)



	filename	width	height	class	xmin	ymin	xmax	ymax
2	raccoon-63.jpg	600	400	raccoon	74	107	280	290
3	raccoon-63.jpg	600	400	raccoon	227	93	403	298

- **Implementation Workflow**




torchvision.models.detection.fasterrcnn_resnet50_fpn
Faster RCNN
Images
[List]
Faster RCNN
Targets
[List]
Target
{Dict}
Boxes
Labels
Scores
Evaluation
Train
Images [List]
Targets [List]
Faster RCNN
Loss [Dict]

- **Training Error Profiles (resnet50 Backbone)**

```
epoch [0]:      lr: [0.005]     loss: 0.23906825482845306
epoch [1]:      lr: [0.005]     loss: 0.14593052864074707
epoch [2]:      lr: [0.0005]    loss: 0.11503149569034576
epoch [3]:      lr: [0.0005]    loss: 0.09726031124591827
epoch [4]:      lr: [0.0005]    loss: 0.09543923288583755
epoch [5]:      lr: [5e-05]     loss: 0.09296727180480957
epoch [6]:      lr: [5e-05]     loss: 0.09172710031270981
epoch [7]:      lr: [5e-05]     loss: 0.0920213833451271
epoch [8]:      lr: [5e-06]     loss: 0.09002673625946045
epoch [9]:      lr: [5e-06]     loss: 0.09163390845060349
backbone: resnet50, optimizer: SGD
```

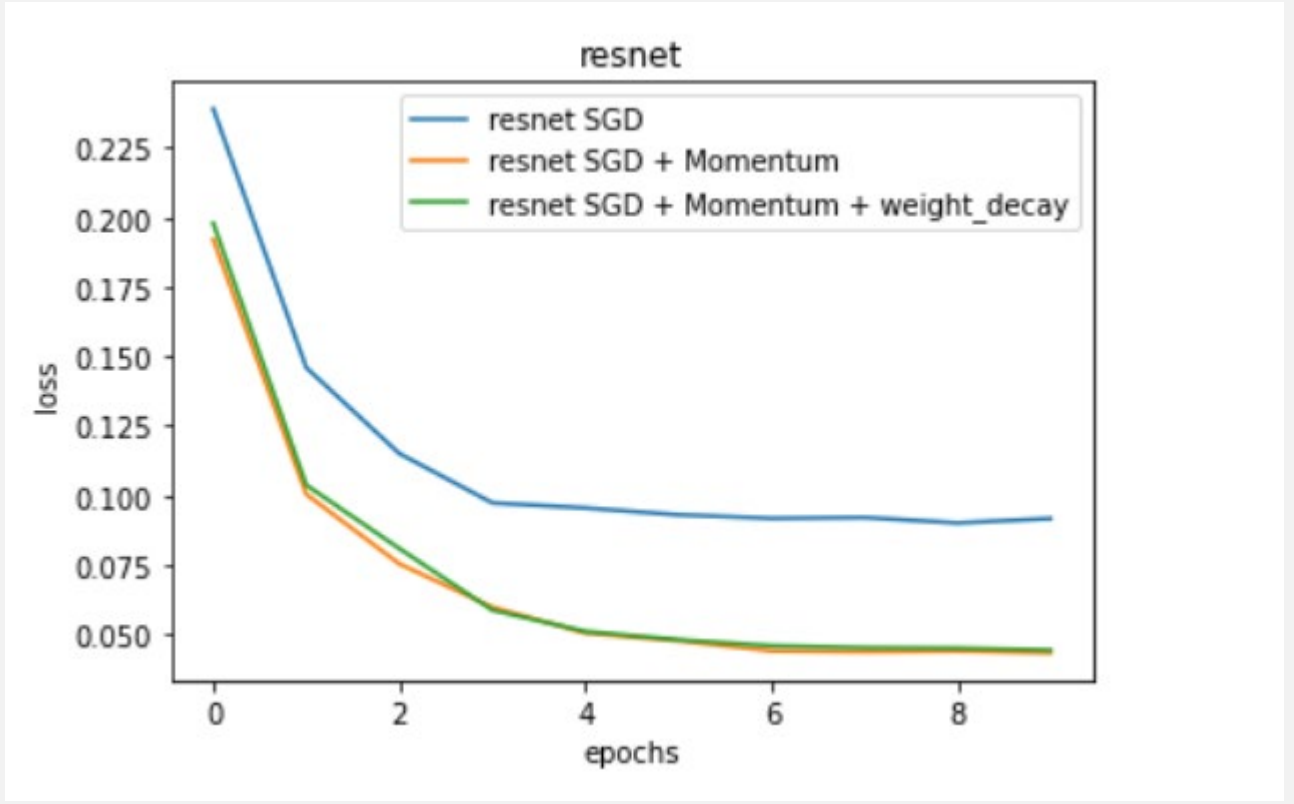
```
epoch [0]:      lr: [0.005]     loss: 0.1978999674320221
epoch [1]:      lr: [0.005]     loss: 0.1036696583032608
epoch [2]:      lr: [0.0005]    loss: 0.08084581792354584
epoch [3]:      lr: [0.0005]    loss: 0.05865839868783951
epoch [4]:      lr: [0.0005]    loss: 0.051022160798311234
epoch [5]:      lr: [5e-05]     loss: 0.04801340028643608
epoch [6]:      lr: [5e-05]     loss: 0.04583305865526199
epoch [7]:      lr: [5e-05]     loss: 0.045158885419368744
epoch [8]:      lr: [5e-06]     loss: 0.04497438296675682
epoch [9]:      lr: [5e-06]     loss: 0.044314462691545486
backbone: resnet50, optimizer: SGD + Momentum + weight_decay
```

```
epoch [0]:      lr: [0.005]     loss: 0.19206878542900085
epoch [1]:      lr: [0.005]     loss: 0.10037096589803696
epoch [2]:      lr: [0.0005]    loss: 0.07534044235944748
epoch [3]:      lr: [0.0005]    loss: 0.05975959077477455
epoch [4]:      lr: [0.0005]    loss: 0.05037936940789223
epoch [5]:      lr: [5e-05]     loss: 0.04760122299194336
epoch [6]:      lr: [5e-05]     loss: 0.04401402547955513
epoch [7]:      lr: [5e-05]     loss: 0.04373014718294144
epoch [8]:      lr: [5e-06]     loss: 0.043966878205537796
epoch [9]:      lr: [5e-06]     loss: 0.04333403334021568
backbone: resnet50, optimizer: SGD + Momentum
```

- **Training Error Profiles (resnet50 Backbone)**

```
epoch [0]:        lr: [0.005]
epoch [1]:        lr: [0.005]
epoch [2]:        lr: [0.0005]
epoch [3]:        lr: [0.0005]
epoch [4]:        lr: [0.0005]
epoch [5]:        lr: [5e-05]
epoch [6]:        lr: [5e-05]
epoch [7]:        lr: [5e-05]
epoch [8]:        lr: [5e-06]
epoch [9]:        lr: [5e-06]
backbone: resnet50, optimizer
```

```
epoch [0]:        lr: [0.005]
epoch [1]:        lr: [0.005]
epoch [2]:        lr: [0.0005]
epoch [3]:        lr: [0.0005]
epoch [4]:        lr: [0.0005]
epoch [5]:        lr: [5e-05]
epoch [6]:        lr: [5e-05]
epoch [7]:        lr: [5e-05]
epoch [8]:        lr: [5e-06]
epoch [9]:        lr: [5e-06]
backbone: resnet50, optimizer
```



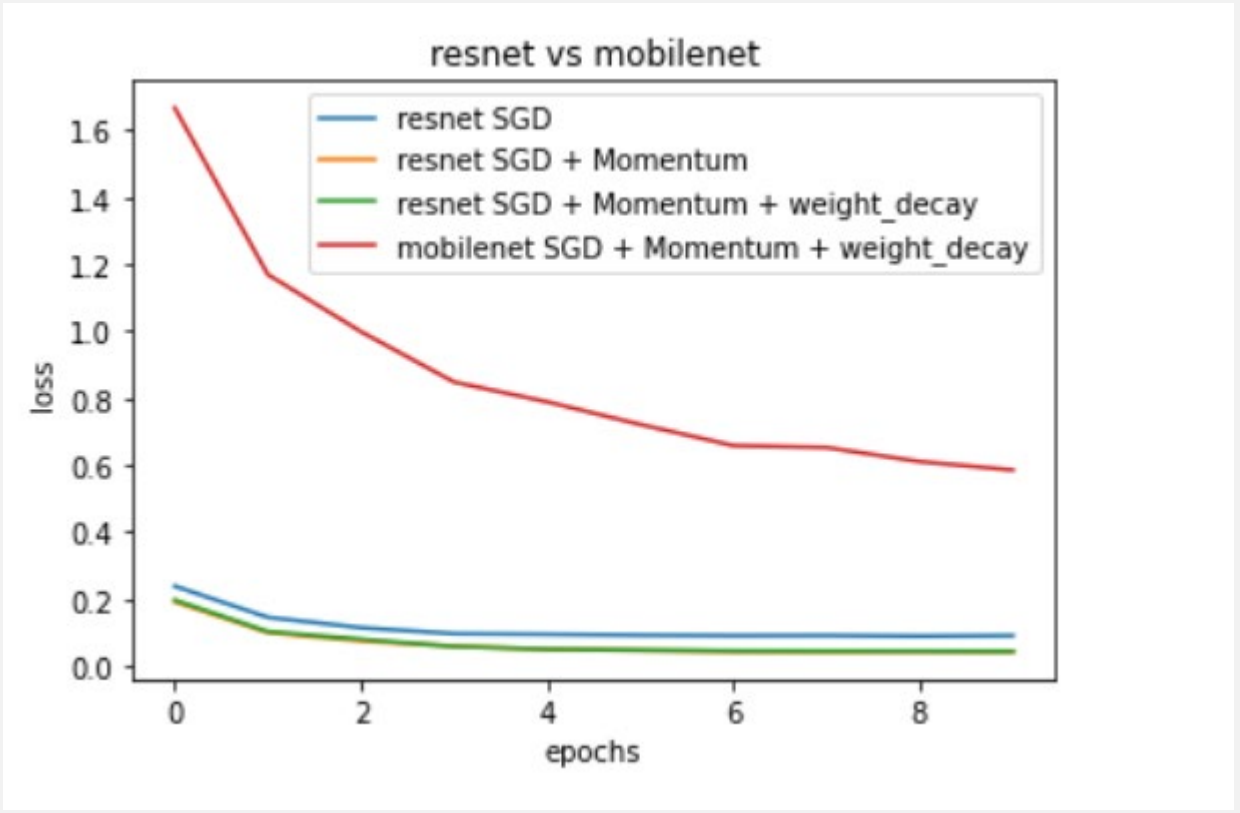

```
19206878542900085
10037096589803696
07534044235944748
05975959077477455
05037936940789223
04760122299194336
04401402547955513
04373014718294144
043966878205537796
04333403334021568
ntum
```

- **Training Error Profiles (mobilenet_v2 Backbone + Optimizer 3 {SGD + Momentum + Weight Decay})**

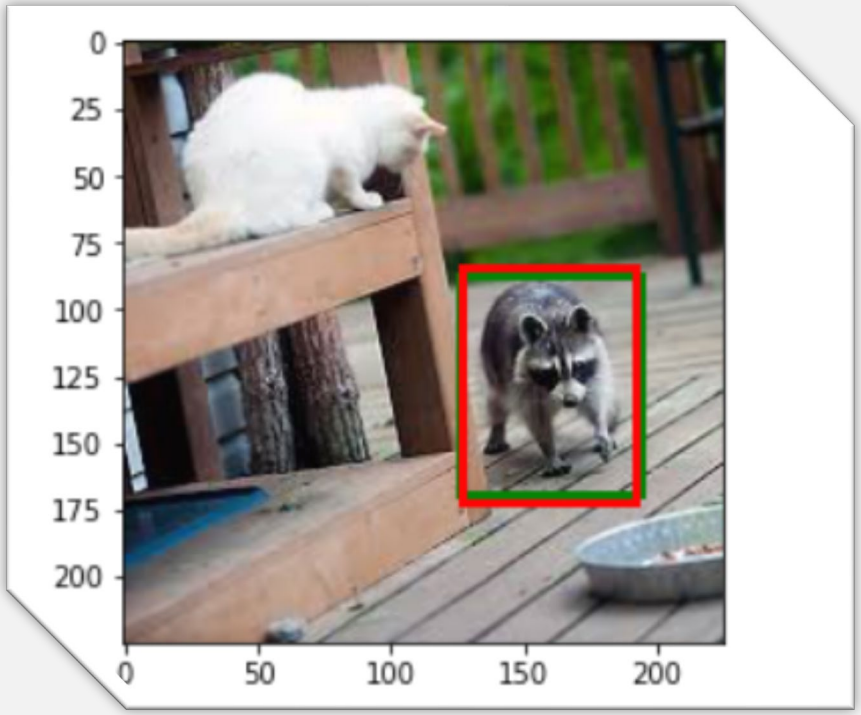
```
epoch [0]:       lr: [0.001]     loss: 1.6662325859069824
epoch [1]:       lr: [0.001]     loss: 1.168940782546997
epoch [2]:       lr: [0.0007]    loss: 0.9984908699989319
epoch [3]:       lr: [0.0007]    loss: 0.8479897379875183
epoch [4]:       lr: [0.0007]    loss: 0.7889596223831177
epoch [5]:       lr: [0.00049]           loss: 0.7214080095291138
epoch [6]:       lr: [0.00049]           loss: 0.6583783626556396
epoch [7]:       lr: [0.00049]           loss: 0.6526705026626587
epoch [8]:       lr: [0.000343]          loss: 0.6106132864952087
epoch [9]:       lr: [0.000343]          loss: 0.585443377494812
backbone: mobilenet, optimizer: SGD + Momentum + weight_decay
```

- **Training Error Profiles (mobilenet_v2 Backbone + Optimizer 3 {SGD + Momentum + Weight Decay})**

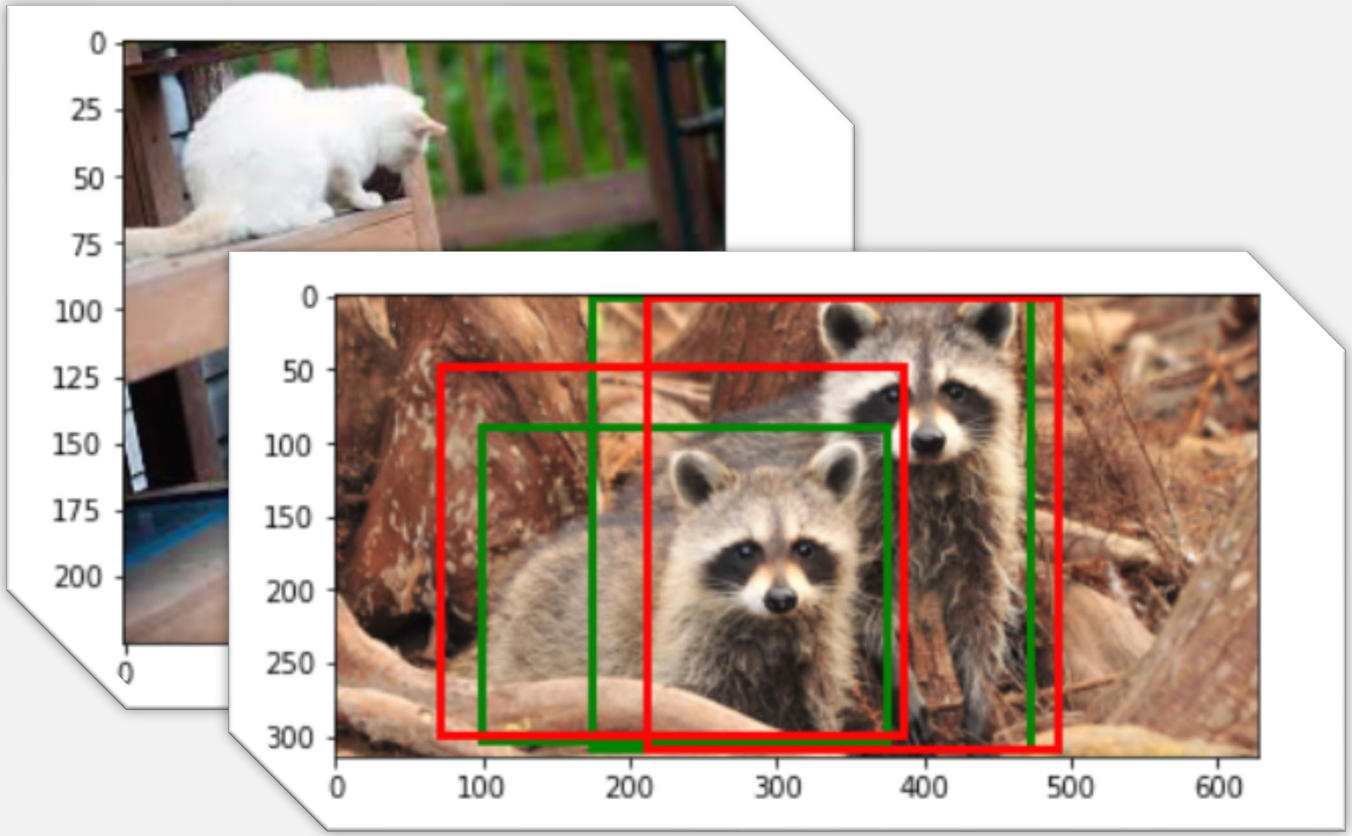
```
epoch [0]:        lr: [0.001]
epoch [1]:        lr: [0.001]
epoch [2]:        lr: [0.0007]
epoch [3]:        lr: [0.0007]
epoch [4]:        lr: [0.0007]
epoch [5]:        lr: [0.00049]
epoch [6]:        lr: [0.00049]
epoch [7]:        lr: [0.00049]
epoch [8]:        lr: [0.000343]
epoch [9]:        lr: [0.000343]
backbone: mobilenet, optimizer:
```



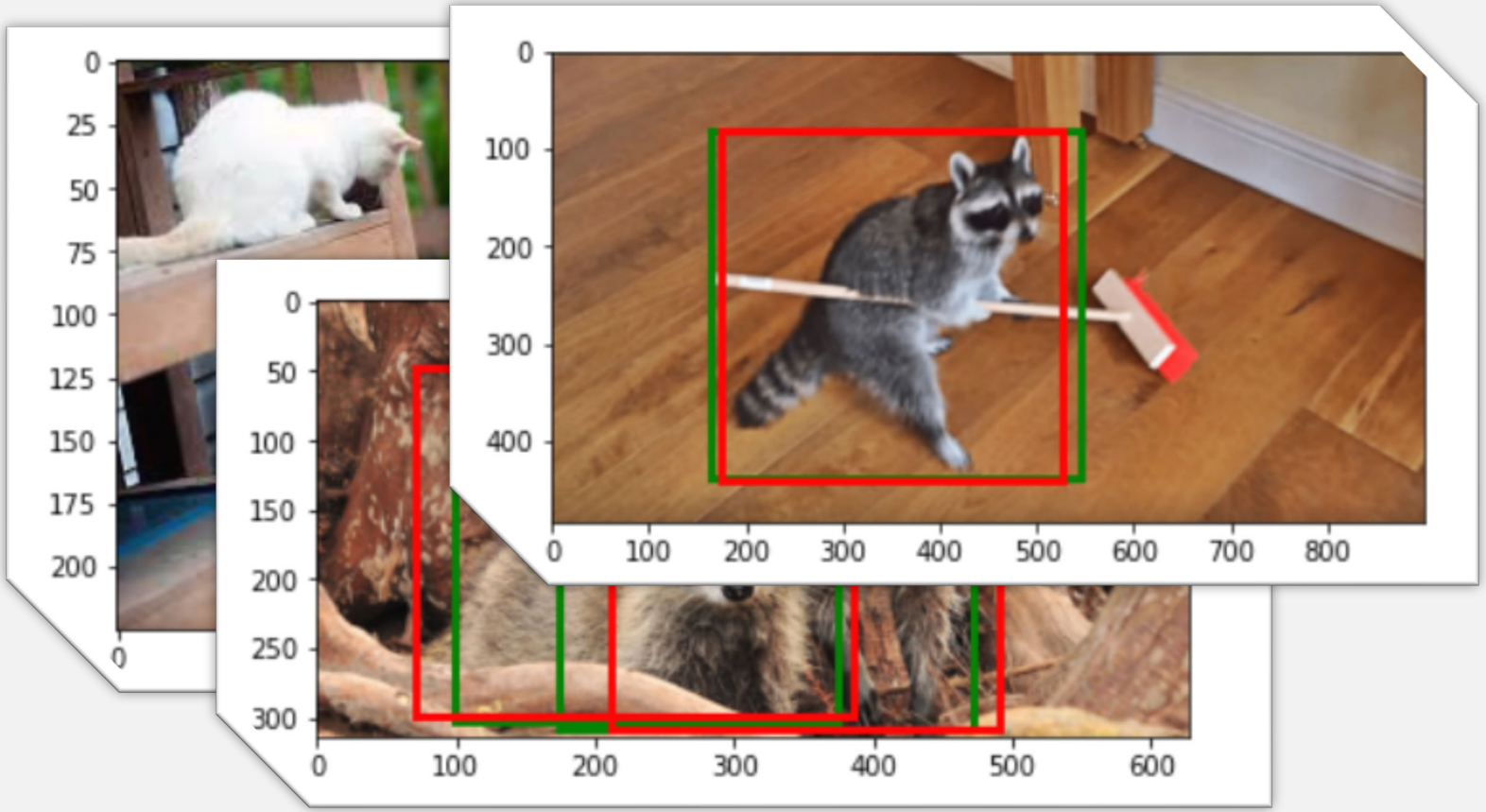
- **Evaluation Results (resnet Backbone + Optimizer 3 {SGD + Momentum + Weight Decay})**
  - **In evaluation, the model gives us scores with boxes (boxes with score > 0.7)**
  - **Red : Model suggestions / Green : Targets**



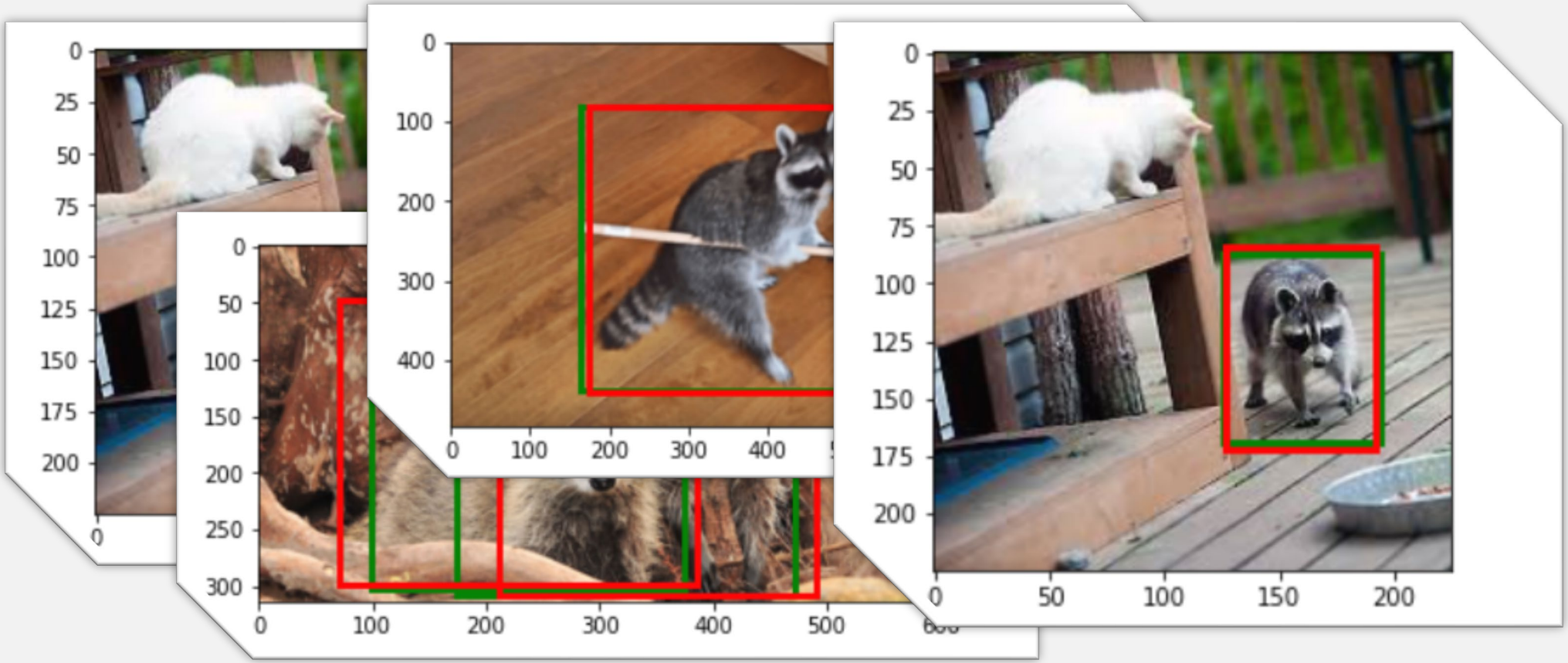
- **Evaluation Results (resnet Backbone + Optimizer 3 {SGD + Momentum + weight_decay)**
  - **In evaluation, the model gives us scores with boxes (boxes with score > 0.7)**
  - **Red : Model suggestions / Green : Targets**



- **Evaluation Results (resnet Backbone + Optimizer 3 {SGD + Momentum + weight_decay)**
  - **In evaluation, the model gives us scores with boxes (boxes with score > 0.7)**
  - **Red : Model suggestions / Green : Targets**



- **Evaluation Results (resnet Backbone + Optimizer 3 {SGD + Momentum + weight_decay)**
  - **In evaluation, the model gives us scores with boxes (boxes with score > 0.7)**
  - **Red : Model suggestions / Green : Targets**



- **References**
  - Main Paper:
    - Faster R-CNN: Towards Object Detection with Region Proposal Networks - Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun- NIPS' 15 - Published: Dec 2015
  - Dataset:
    - Raccooon dataset
  - Other References:
    - Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3)
    - R-CNN vs Fast R-CNN vs Faster R-CNN – A Comparative Guide
    - R-CNN for object detection : A technical paper summary
    - Fast R-CNN for object detection : A technical paper summary
    - Faster R-CNN for object detection : A technical paper summary
    - Distilled Notes for Stanford CS231n: Convolutional Neural Networks for Visual Recognition
    - Deep Learning for Object Detection: A Comprehensive Review
    - CS231n Winter 2016: Lecture 8: Localization and Detection (Stanford Winter Quarter 2016 Class)
    - MaskRCNN with MobileNet backbone