Faster R-CNN

Shaoqing Ren et al., 2016

곽대훈

Overview

Review & Motivation

R-CNN

1. Input
image

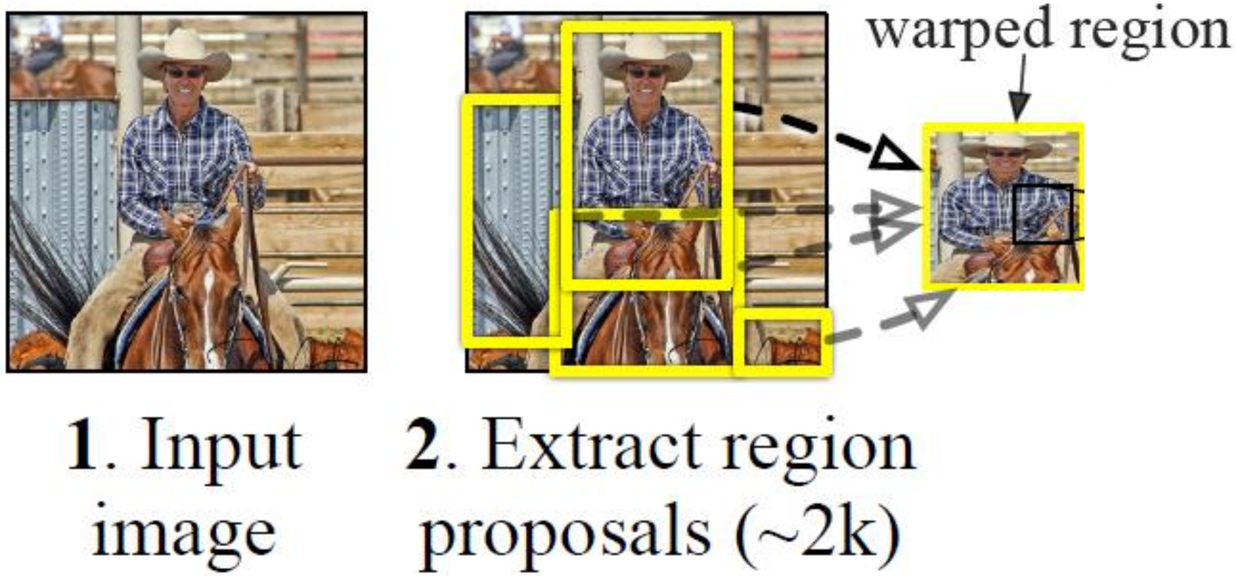
R-CNN

1. Input image

2. Extract region proposals (~2k)

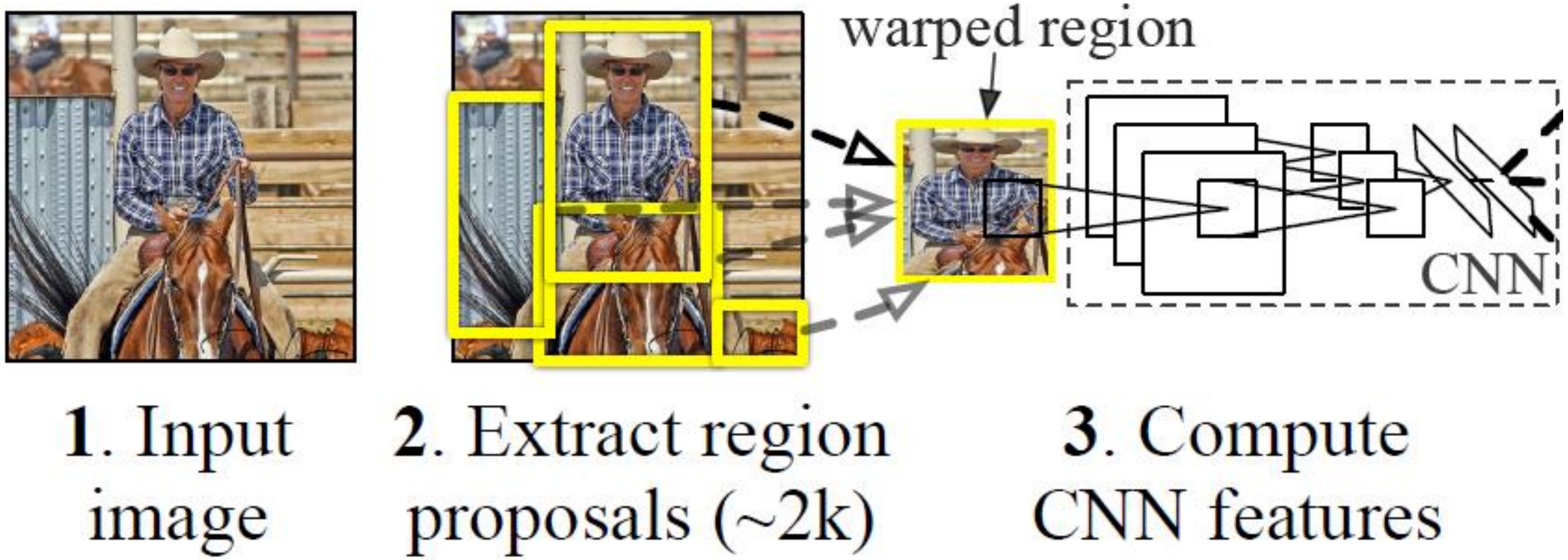
Selective Search

R-CNN



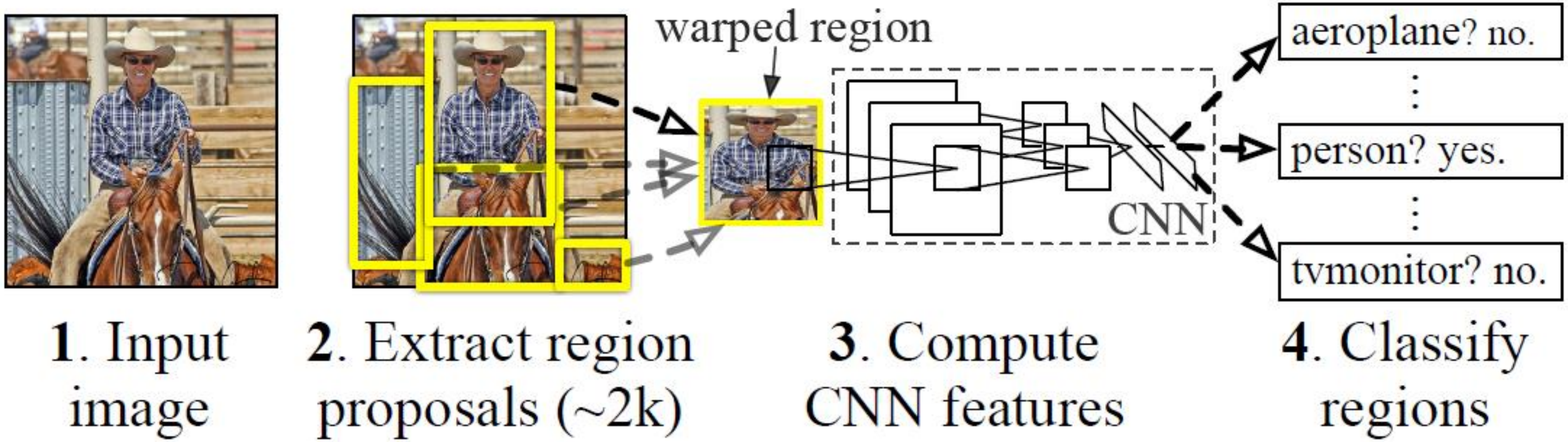
Regardless of the size or aspect ratio of the candidate region,
we warp all pixels in a tight bounding box around it to the required size.

R-CNN



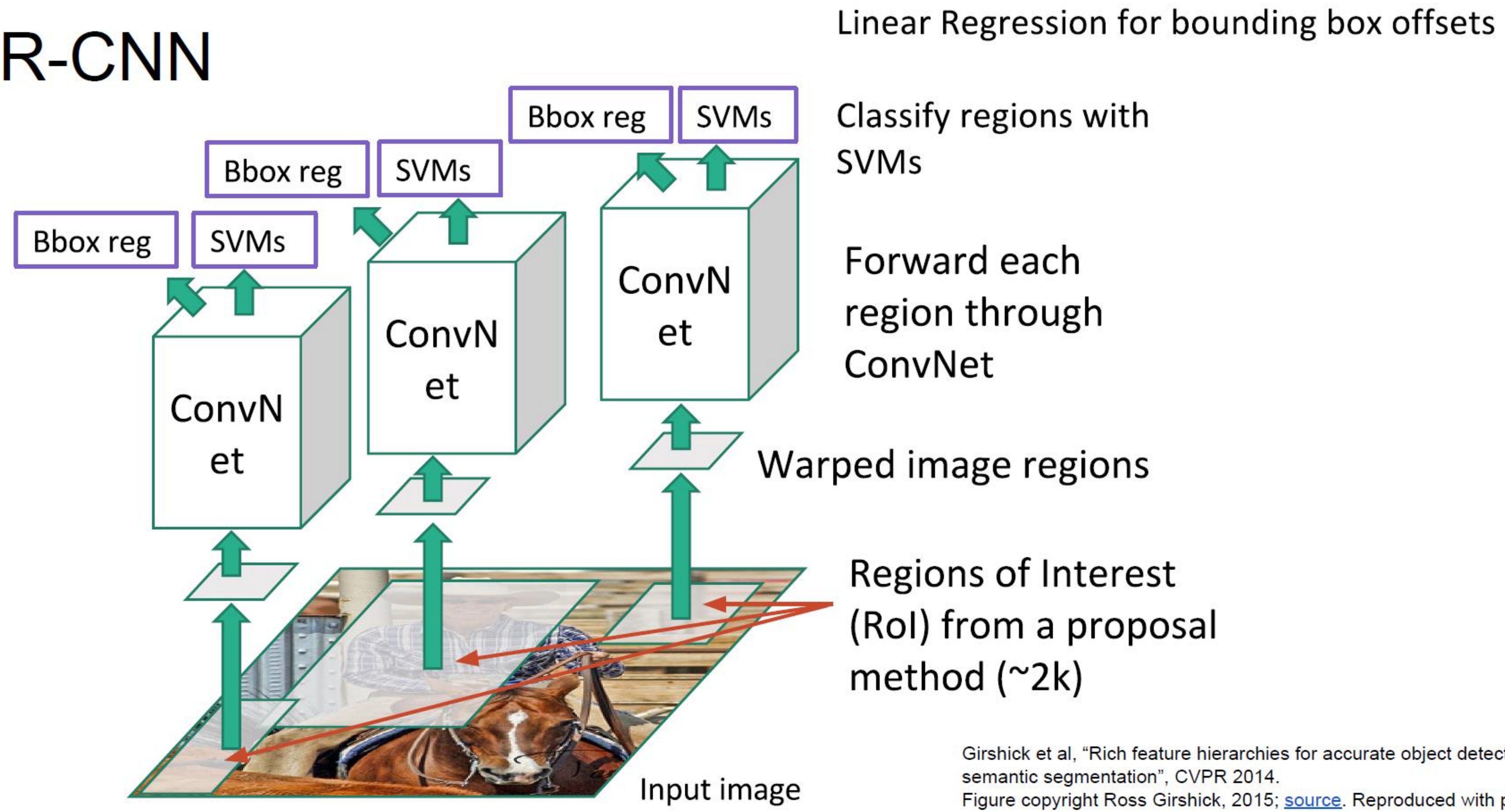
It requires a forward pass of the CNN (AlexNet) for every single region proposal for every single image (that's around 2000 forward passes per image!).

R-CNN



SVM / Bbox reg

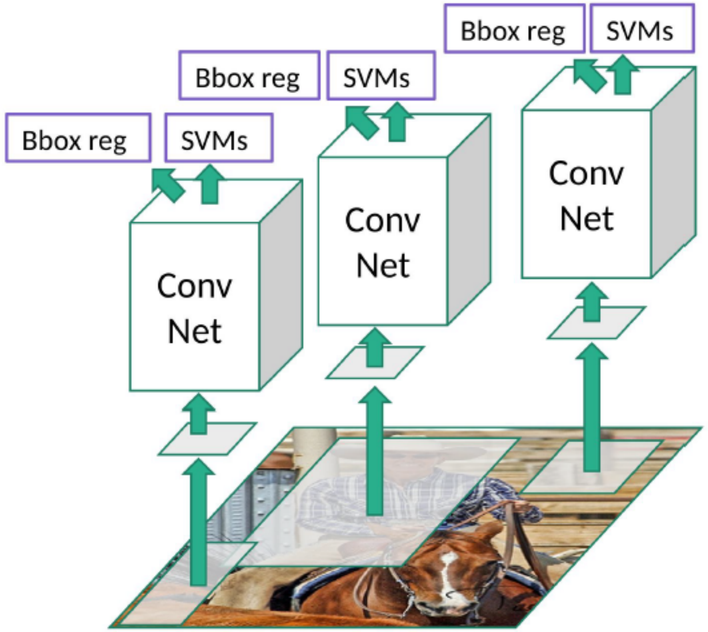
R-CNN



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

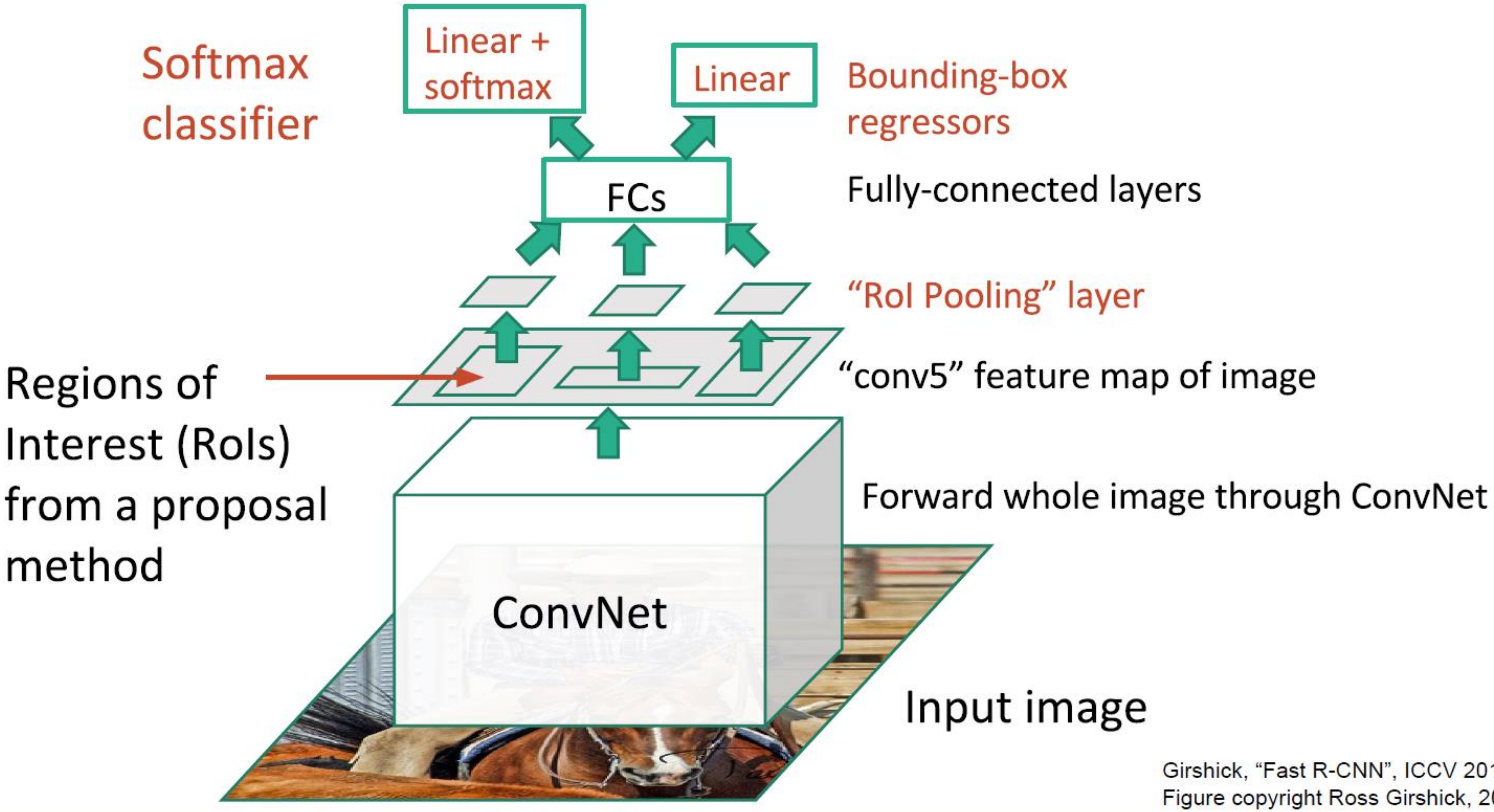
R-CNN: Problems

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



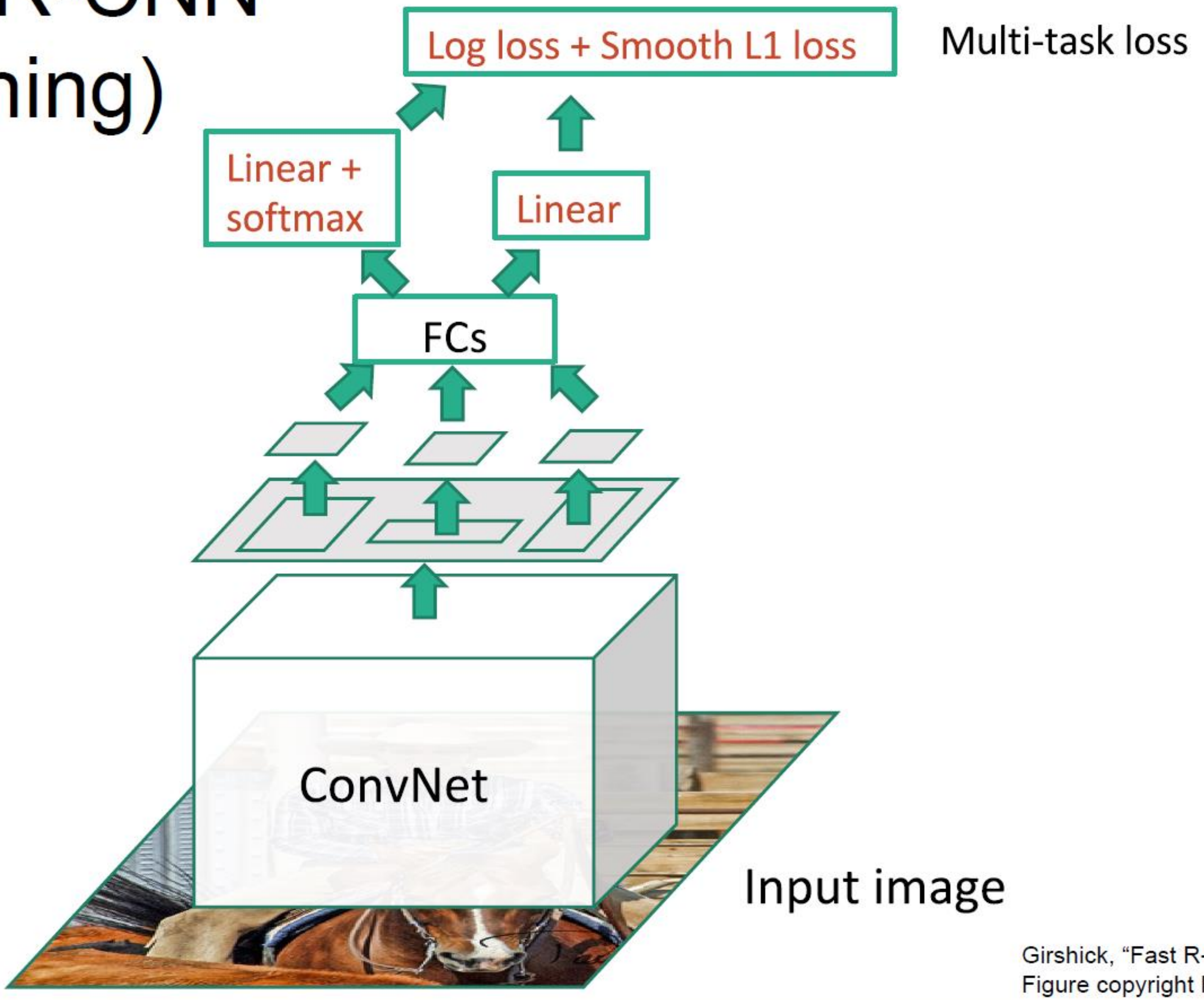
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

Fast R-CNN



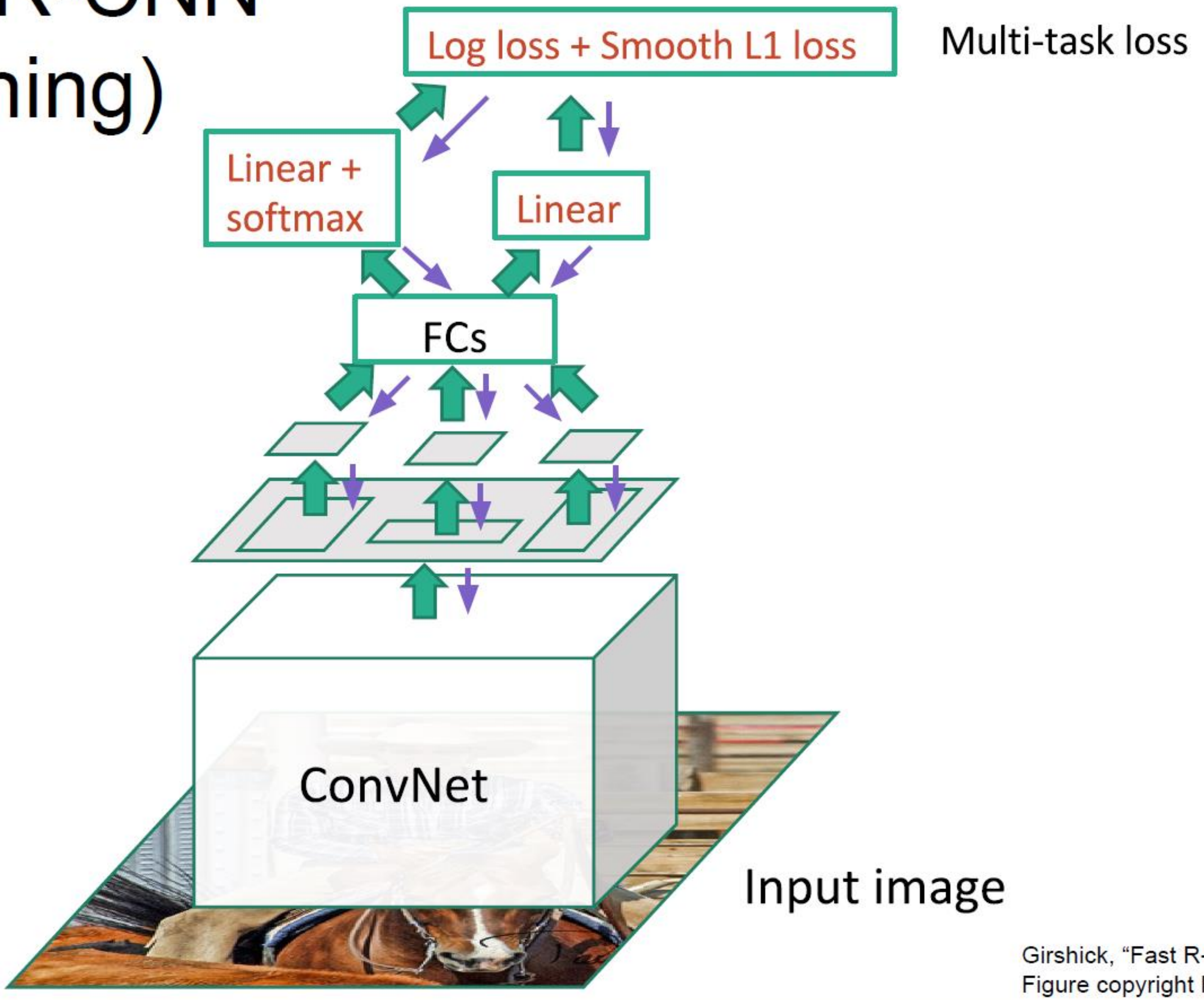
Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN (Training)



Girshick, “Fast R-CNN”, ICCV 2015.

Fast R-CNN (Training)



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

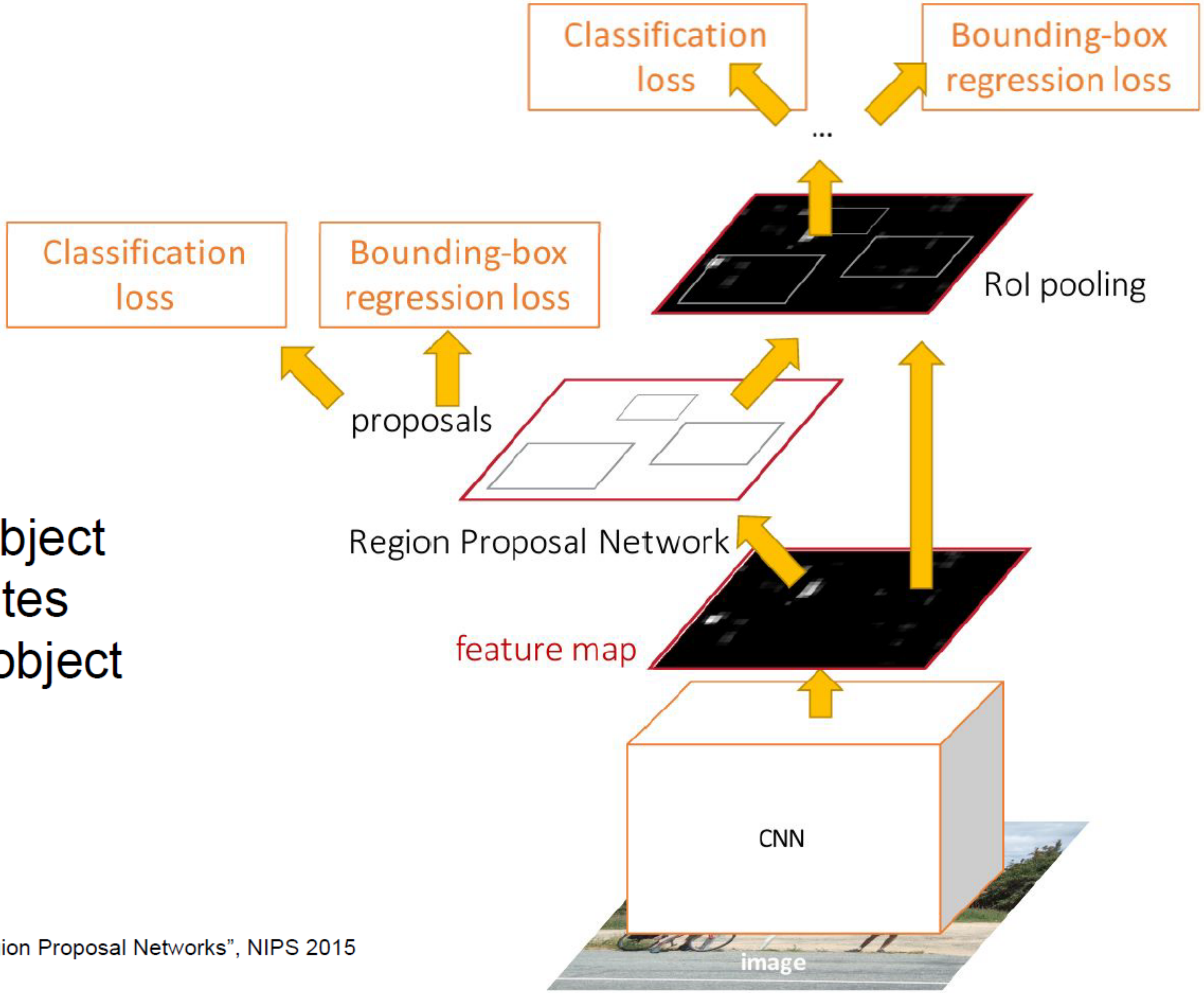
Faster R-CNN:

Make CNN do proposals!

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

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



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

Faster R-CNN

" We introduce a Region Proposal Network (RPN) that shares full-image convolutional features "
with the detection network, thus enabling nearly cost-free region proposals.

Faster R-cnn

1.Introduction

- Fast R-CNN achieves near real-time rates using very deep networks when ignoring the time spent on region proposals.

Faster R-cnn

1.Introduction

- Fast R-CNN achieves near real-time rates using very deep networks when ignoring the time spent on region proposals.
- Now, proposals are the test-time computational bottleneck in state-of-the-art detection systems.

Faster R-cnn

1.Introduction

- 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.

Faster R-cnn

1.Introduction

- 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.

Faster R-cnn

1.Introduction

- 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.

Faster R-cnn

1.Introduction

- 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.

Faster R-cnn

1.Introduction

- We introduce novel Region Proposal Networks (RPNs) that share convolutional layers with state-of-the-art object detection networks [1], [2].

Faster R-cnn

1.Introduction

- 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).

Faster R-cnn

1.Introduction

- 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.

Faster R-cnn

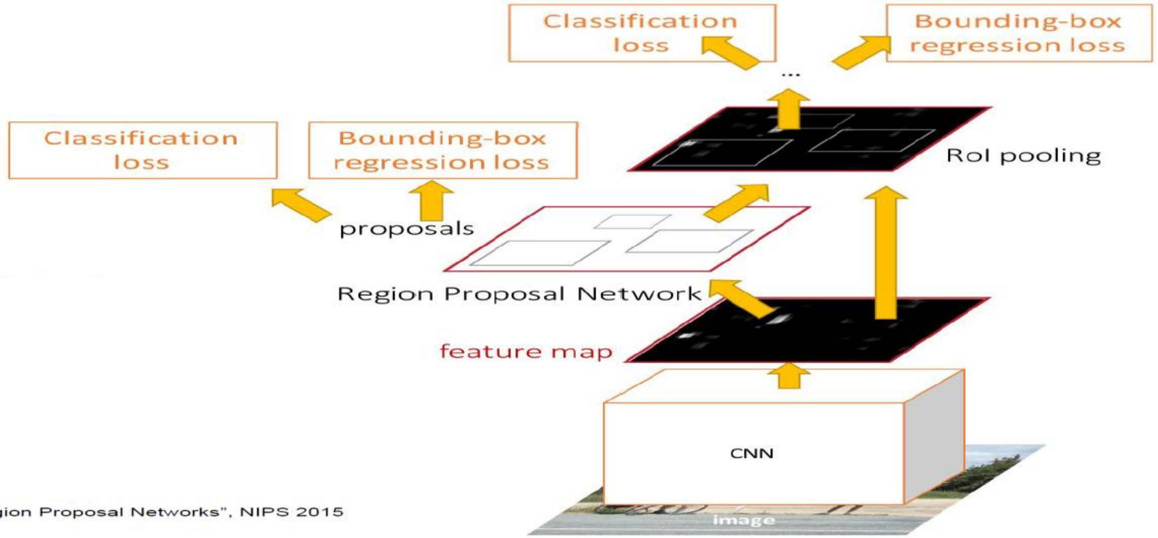
1.Introduction

- 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.

Faster R-cnn

1.Introduction

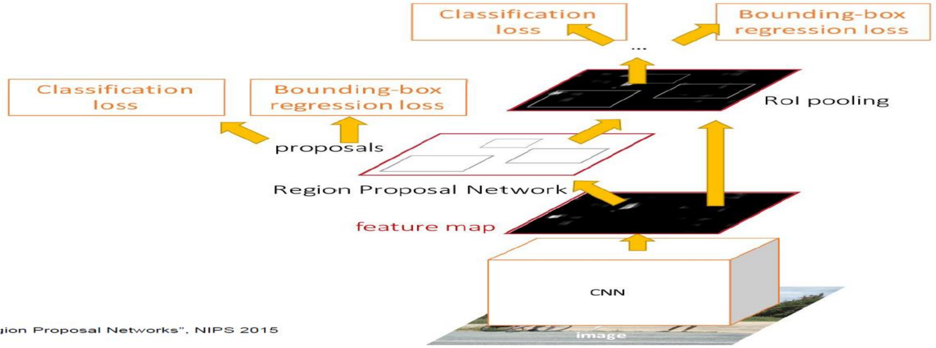
- Our observation is that the convolutional feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals.
- On top of these convolutional features, we construct an RPN by adding a few additional convolutional layers.



jion Proposal Networks", NIPS 2015

Faster R-cnn

1.Introduction



gion Proposal Networks", NIPS 2015

- 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.
- The RPN is thus a kind of fully convolutional network (FCN) and can be trained end-to-end specifically for the task for generating detection proposals.

Faster R-cnn
1.Introduction

- RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios.

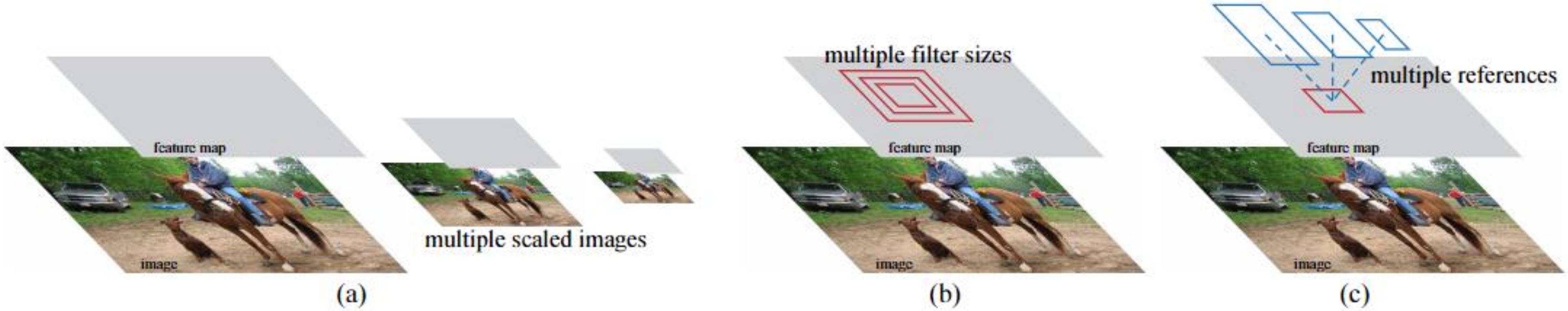


Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

Updated Jun 15, 2017 By Cedric Demers , Mehdi Azzabi

What is the Aspect Ratio?

(4:3, 16:9, 21:9)

The aspect ratio refers to the proportions of the height and width of an image. It defines its overall shape, and it is usually shown as W:H (W is the width and H is the height). The most common aspect ratio today is 16:9, which means that if the width is divided into 16 equal parts, the height of the TV or picture should be 9 parts.

16:9 works great for TVs since that is the format modern TV shows are delivered on, but most movies are made using the cinema standard, which is 21:9. 21:9 is much wider, so parts of the screen need to be filled with black bars above and below the image in order to fit most TVs. These horizontal bars a called "letterboxes". Similar to movies, TV shows used to be made using a 4:3 aspect ratio, which is a lot more square than current TVs (this is why 16:9 is often called a widescreen aspect ratio). To fit modern TVs, vertical black bars or "pillarboxing" is used. We've listed the most common aspect ratios in this table, but every TV sold today uses 16:9.

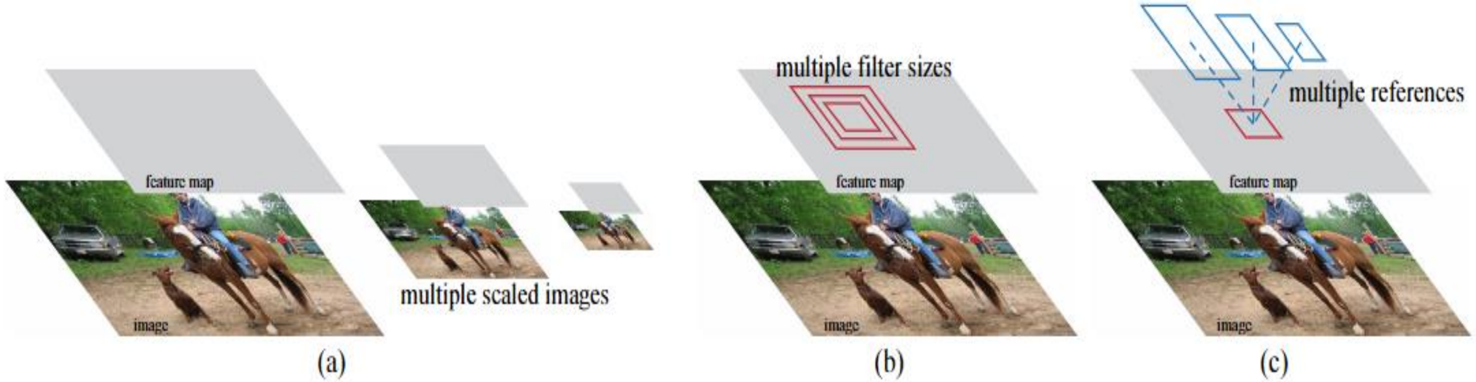
> In theaters, this is why the screen grows wider at the beginning of a movie. Ads shown before the movie follow the TV ratio of 16:9, while the movie itself is 21:9.

Aspect Ratio		Uses	TVs
4:3	1.33:1	Standard Channels	Old TVs
16:9	1.77:1	HD Channels	The majority of HDTVs
21:9	2.35:1	Most movies	Most theaters
14:10	1.4:1	IMAX Film	Very few theaters
19:10	1.9:1	IMAX Digital	Most IMAX theaters

The most common aspect ratios in the video industry.

Faster R-cnn

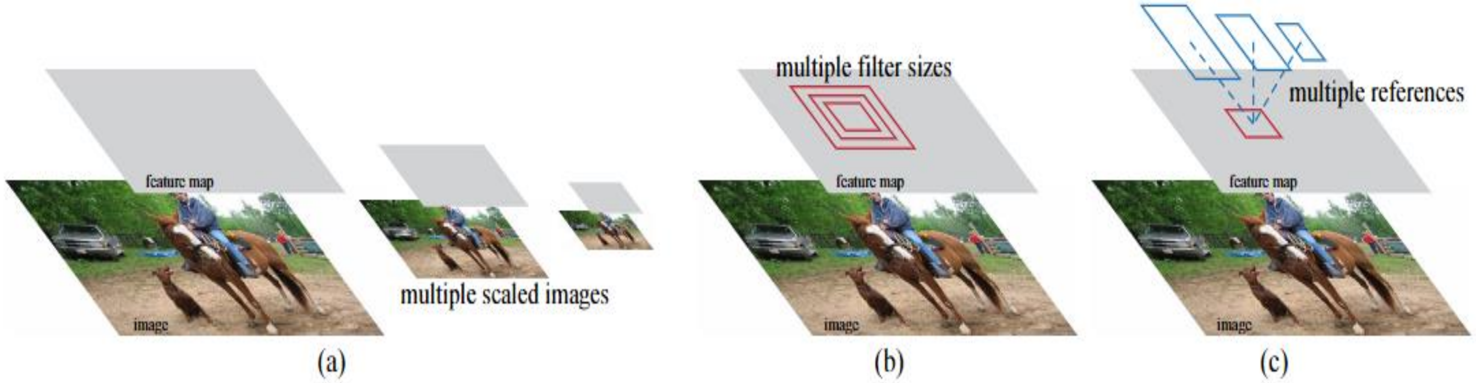
1.Introduction



- RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios.

Faster R-cnn

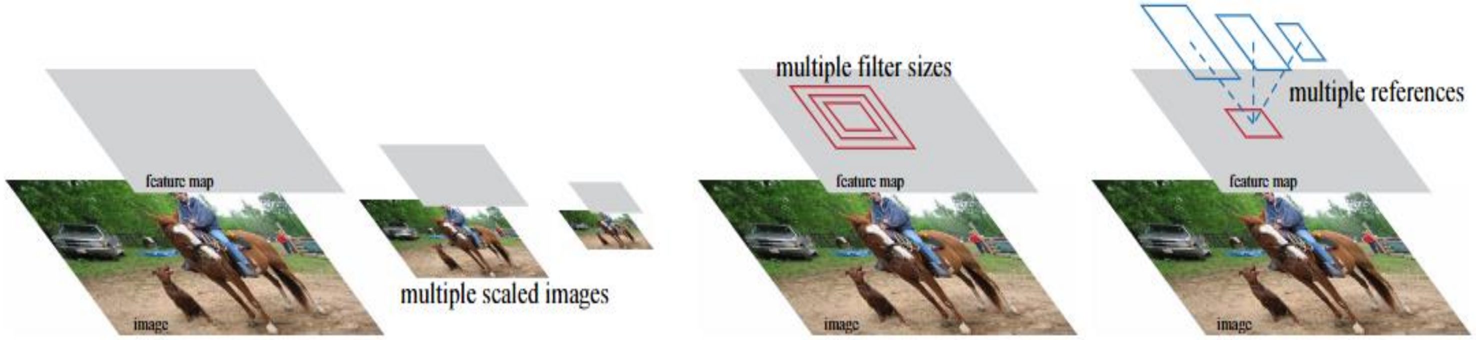
1.Introduction



- RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios.
- We introduce novel “anchor” boxes that serve as references at multiple scales and aspect ratios. Our scheme can be thought of as a pyramid of regression references (Figure 1, c), which avoids enumerating images or filters of multiple scales or aspect ratios.

Faster R-cnn

1.Introduction



(a) (b) (c)

- RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios.
- We introduce novel "anchor" boxes that serve as references at multiple scales and aspect ratios. Our scheme can be thought of as a pyramid of regression references (Figure 1, c), which avoids enumerating images or filters of multiple scales or aspect ratios.
- This model performs well when trained and tested using single-scale images and thus benefits running speed.

Faster R-cnn

1.Introduction

- To unify RPNs with Fast R-CNN object detection networks, we propose a training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection, while keeping the proposals fixed.

Faster R-cnn

1.Introduction

- To unify RPNs with Fast R-CNN object detection networks, we propose a training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection, while keeping the proposals fixed.
- This scheme converges quickly and produces a unified network with convolutional features that are shared between both tasks.

Faster R-cnn

1.Introduction

- To unify RPNs with Fast R-CNN object detection networks, we propose a training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection, while keeping the proposals fixed.
- This scheme converges quickly and produces a unified network with convolutional features that are shared between both tasks.
- We have also found that RPNs can be trained jointly with Fast R-CNN networks leading to less training time.

Faster R-cnn

1.Introduction

- RPNs completely learn to propose regions from data, and thus can easily benefit from deeper and more expressive features

Faster R-cnn

1.Introduction

- RPNs completely learn to propose regions from data, and thus can easily benefit from deeper and more expressive features
- Our method is not only a cost-efficient solution for practical usage, but also an effective way of improving object detection accuracy.

Faster R-cnn

2. Related Work

- Object Proposals.

- Object proposal methods were adopted as external modules independent of the detectors.

Faster R-cnn

2. Related Work

- Deep Networks for Object Detection.

- R-CNN mainly plays as a classifier, and it does not predict object bounds. (except for refining by bounding box regression)

2. Related Work

- Deep Networks for Object Detection.

- R-CNN mainly plays as a classifier, and it does not predict object bounds. (except for refining by bounding box regression)
- Its accuracy depends on the performance of the region proposal module.

Faster R-cnn

2. Related Work

- Deep Networks for Object Detection.

- Several papers have proposed ways of using deep networks for predicting object bounding boxes.

2. Related Work

- Deep Networks for Object Detection.

- Several papers have proposed ways of using deep networks for predicting object bounding boxes.
- In the OverFeat method, a fully-connected layer is trained to predict the box coordinates for the localization task that assumes a single object.

2. Related Work

- Deep Networks for Object Detection.

- Several papers have proposed ways of using deep networks for predicting object bounding boxes.
- In the OverFeat method, a fully-connected layer is trained to predict the box coordinates for the localization task that assumes a single object.
- The MultiBox methods generate region proposals from a network whose last fully-connected layer simultaneously predicts multiple class-agnostic boxes, generalizing the “single box” fashion of OverFeat.

Faster R-cnn

2. Related Work

- Deep Networks for Object Detection.

- Several papers have proposed ways of using deep networks for predicting object bounding boxes.
- In the OverFeat method, a fully-connected layer is trained to predict the box coordinates for the localization task that assumes a single object.
- The MultiBox methods generate region proposals from a network whose last fully-connected layer simultaneously predicts multiple class-agnostic boxes, generalizing the “single box” fashion of OverFeat.
- We discuss OverFeat and MultiBox in more depth later in context with our method.

Faster R-cnn

2. Related Work

- Deep Networks for Object Detection.

- Several papers have proposed ways of using deep networks for predicting object bounding boxes.
- In the OverFeat method, a fully-connected layer is trained to predict the box coordinates for the localization task that assumes a single object.
- The MultiBox methods generate region proposals from a network whose last fully-connected layer simultaneously predicts multiple class-agnostic boxes, generalizing the “single box” fashion of OverFeat.
- We discuss OverFeat and MultiBox in more depth later in context with our method.
- Concurrent with our work, the DeepMask method is developed for learning segmentation proposals.

Faster R-cnn

2. Related Work

- Deep Networks for Object Detection.

- Shared computation of convolutions has been attracting increasing attention for efficient, yet accurate, visual recognition.

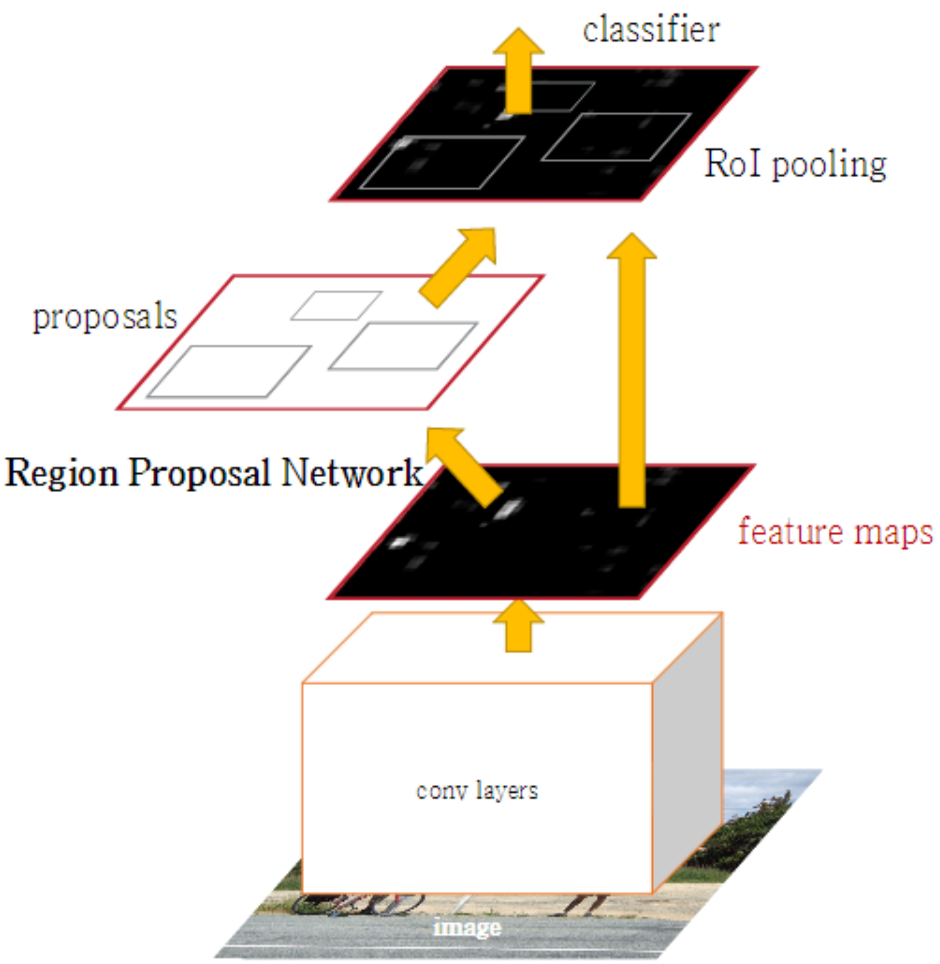
Faster R-cnn

2. Related Work

- Deep Networks for Object Detection.

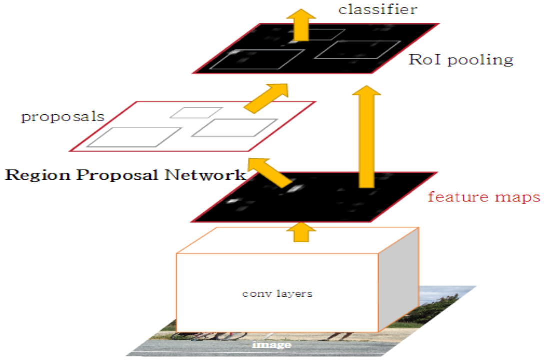
- Shared computation of convolutions has been attracting increasing attention for efficient, yet accurate, visual recognition.
- OverFeat
- SPPnet
- Fast R-CNN

3. Faster R-cnn



Faster R-cnn

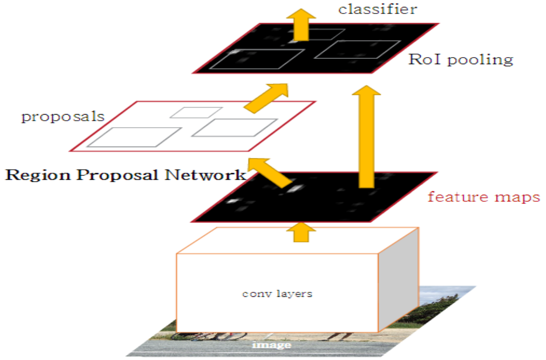
3. Faster R-cnn



- Our object detection system, called Faster R-CNN, is composed of two modules.

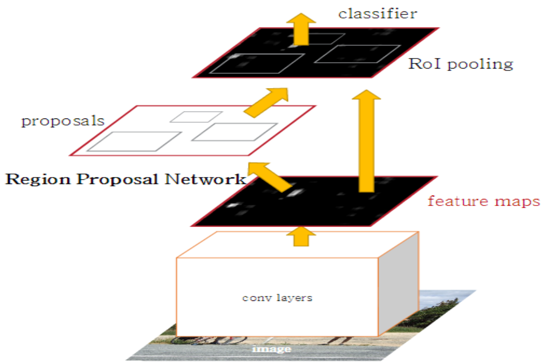
Faster R-cnn

3. Faster R-cnn



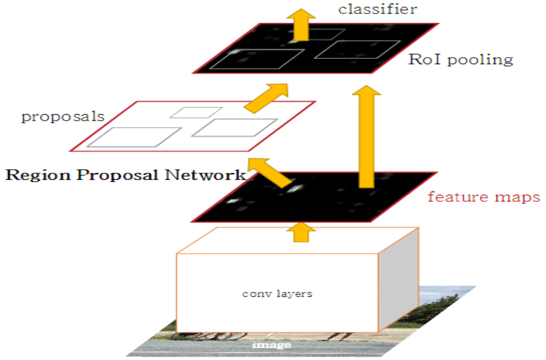
- Our object detection system, called Faster R-CNN, is composed of two modules.
- The first module is a deep fully convolutional network that proposes regions.

3. Faster R-cnn



- Our object detection system, called Faster R-CNN, is composed of two modules.
- The first module is a deep fully convolutional network that proposes regions.
- The second module is the Fast R-CNN detector that uses the proposed regions.

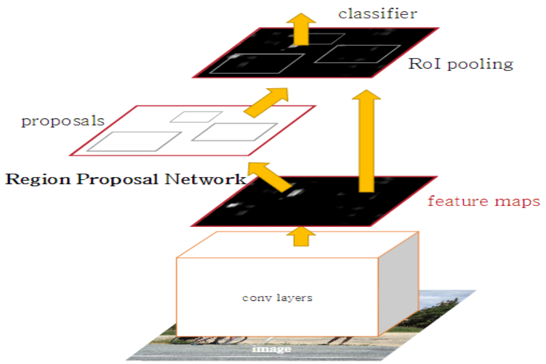
3. Faster R-cnn



- Our object detection system, called Faster R-CNN, is composed of two modules.
- The first module is a deep fully convolutional network that proposes regions.
- The second module is the Fast R-CNN detector that uses the proposed regions.
- The entire system is a single, unified network for object detection.

Faster R-cnn

3. Faster R-cnn



- Our object detection system, called Faster R-CNN, is composed of two modules.
- The first module is a deep fully convolutional network that proposes regions.
- The second module is the Fast R-CNN detector that uses the proposed regions.
- The entire system is a single, unified network for object detection.
- Using the recently popular terminology of neural networks with ‘attention’ mechanisms, the RPN module tells the Fast R-CNN module where to look.

Faster R-cnn

3.1 Region Proposal Network

- A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.

Faster R-cnn

3.1 Region Proposal Network

- A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.
- “Region” is a generic term and in this paper we only consider rectangular regions, as is common for many methods.

Faster R-cnn

3.1 Region Proposal Network

- A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.
- “Region” is a generic term and in this paper we only consider rectangular regions, as is common for many methods.
- “Objectness” measures membership to a set of object classes vs. background.

Faster R-cnn

3.1 Region Proposal Network

- A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.
- "Region" is a generic term and in this paper we only consider rectangular regions, as is common for many methods.
- "Objectness" measures membership to a set of object classes vs. background.
- We model this process with a fully convolutional network.

Faster R-cnn

3.1 Region Proposal Network

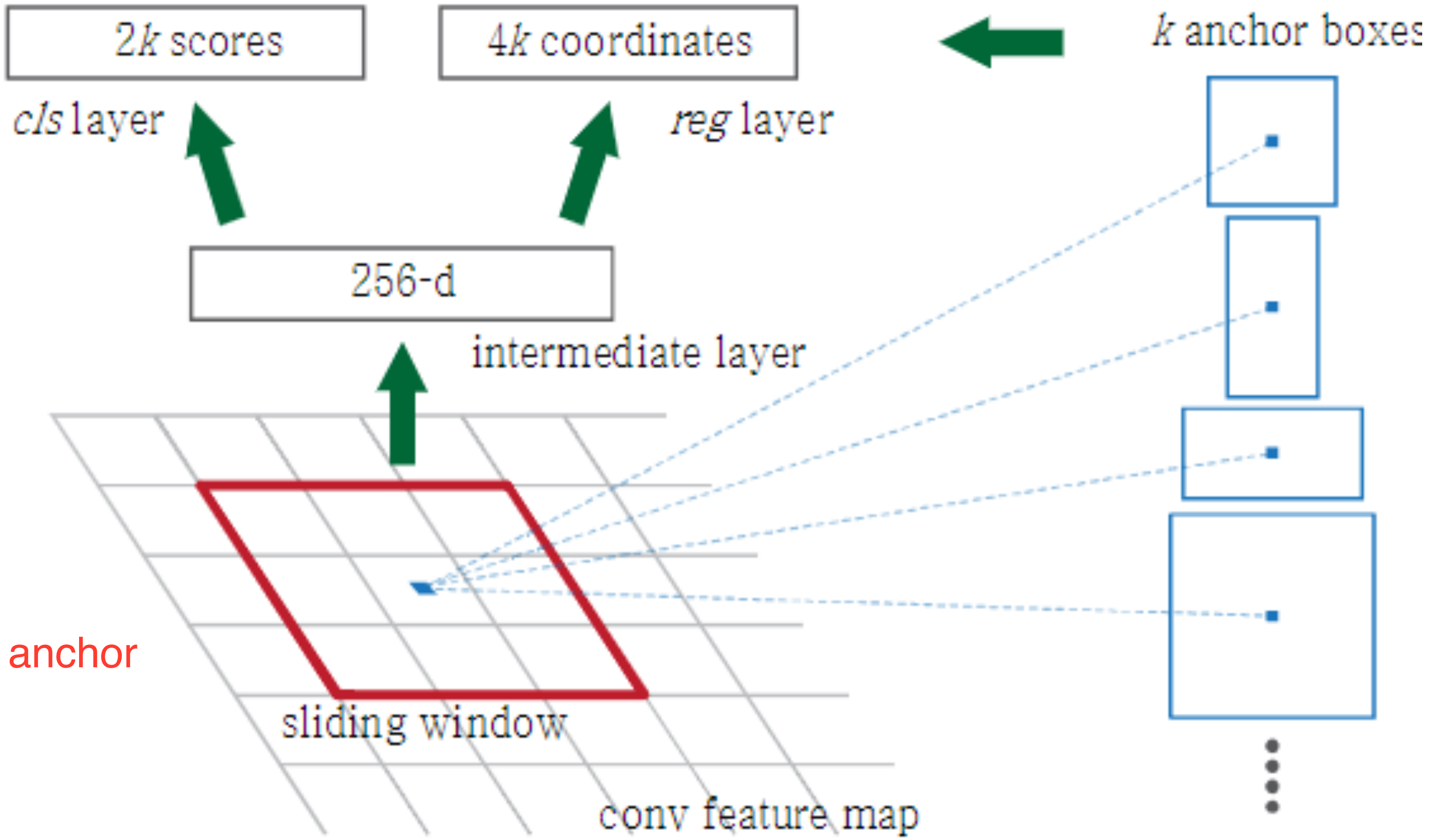
- Because our ultimate goal is to share computation with a Fast R-CNN object detection network, we assume that both nets share a common set of convolutional layers.

3.1 Region Proposal Network

- Because our ultimate goal is to share computation with a Fast R-CNN object detection network, we assume that both nets share a common set of convolutional layers.
- ZF & VGG16

Faster R-cnn

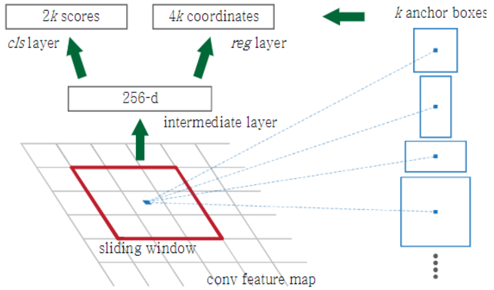
3.1 Region Proposal Network



중심을 anchor

Faster R-cnn

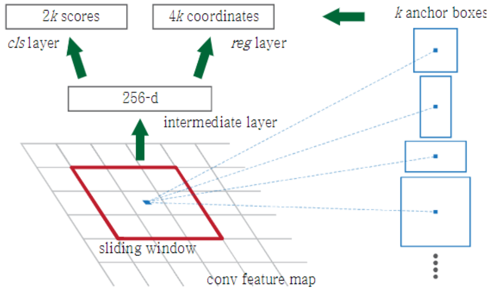
3.1 Region Proposal Network



- To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer.

Faster R-cnn

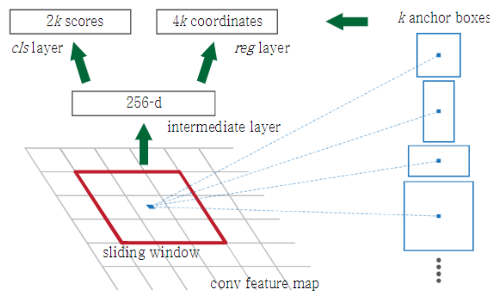
3.1 Region Proposal Network



- To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer.
- This small network takes as input an n x n spatial window of the input convolutional feature map.

Faster R-cnn

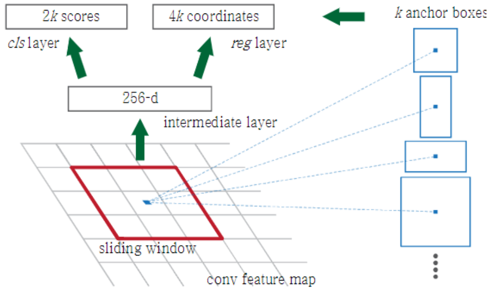
3.1 Region Proposal Network



- To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer.
- This small network takes as input an n x n spatial window of the input convolutional feature map.
- Each sliding window is mapped to a lower-dimensional feature (256-d for ZF and 512-d for VGG, with ReLU following).

Faster R-cnn

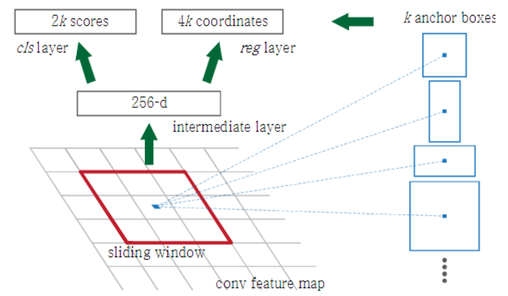
3.1 Region Proposal Network



- To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer.
- This small network takes as input an n x n spatial window of the input convolutional feature map.
- Each sliding window is mapped to a lower-dimensional feature (256-d for ZF and 512-d for VGG, with ReLU following).
- This feature is fed into two sibling fully-connected layers—a box-regression layer and a box-classification layer.

Faster R-cnn

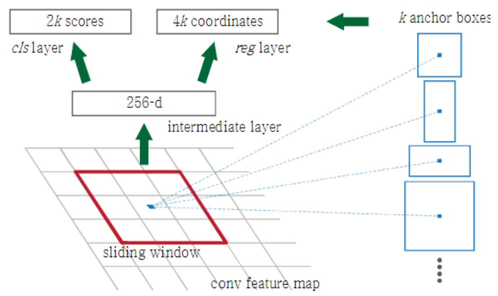
3.1 Region Proposal Network



- To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer.
- This small network takes as input an n x n spatial window of the input convolutional feature map.
- Each sliding window is mapped to a lower-dimensional feature (256-d for ZF and 512-d for VGG, with ReLU following).
- This feature is fed into two sibling fully-connected layers—a box-regression layer and a box-classification layer.
- We use n = 3 in this paper, noting that the effective receptive field on the input image is large.

Faster R-cnn

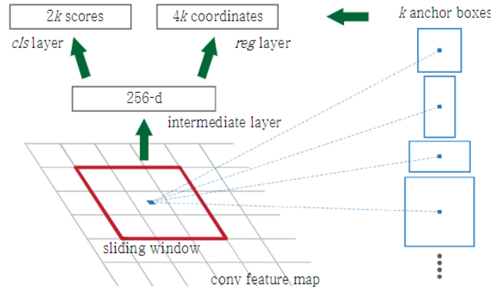
3.1 Region Proposal Network



- Note that because the mini-network operates in a sliding-window fashion, the fully-connected layers are shared across all spatial locations.

Faster R-cnn

3.1 Region Proposal Network



- Note that because the mini-network operates in a sliding-window fashion, the fully-connected layers are shared across all spatial locations.
- This architecture is naturally implemented with an n x n convolutional layer followed by two sibling 1x1 convolutional layers (for reg and cls, respectively).

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors

- At each sliding-window location, we simultaneously predict multiple (denoted as k) region proposals

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors

- At each sliding-window location, we simultaneously predict multiple (denoted as k) region proposals
- So the reg layer has 4k outputs encoding the coordinates of k boxes, and the cls layer outputs 2k scores that estimate probability of object or not object for each proposal.

Faster R-cnn

3.1 Region Proposal Network

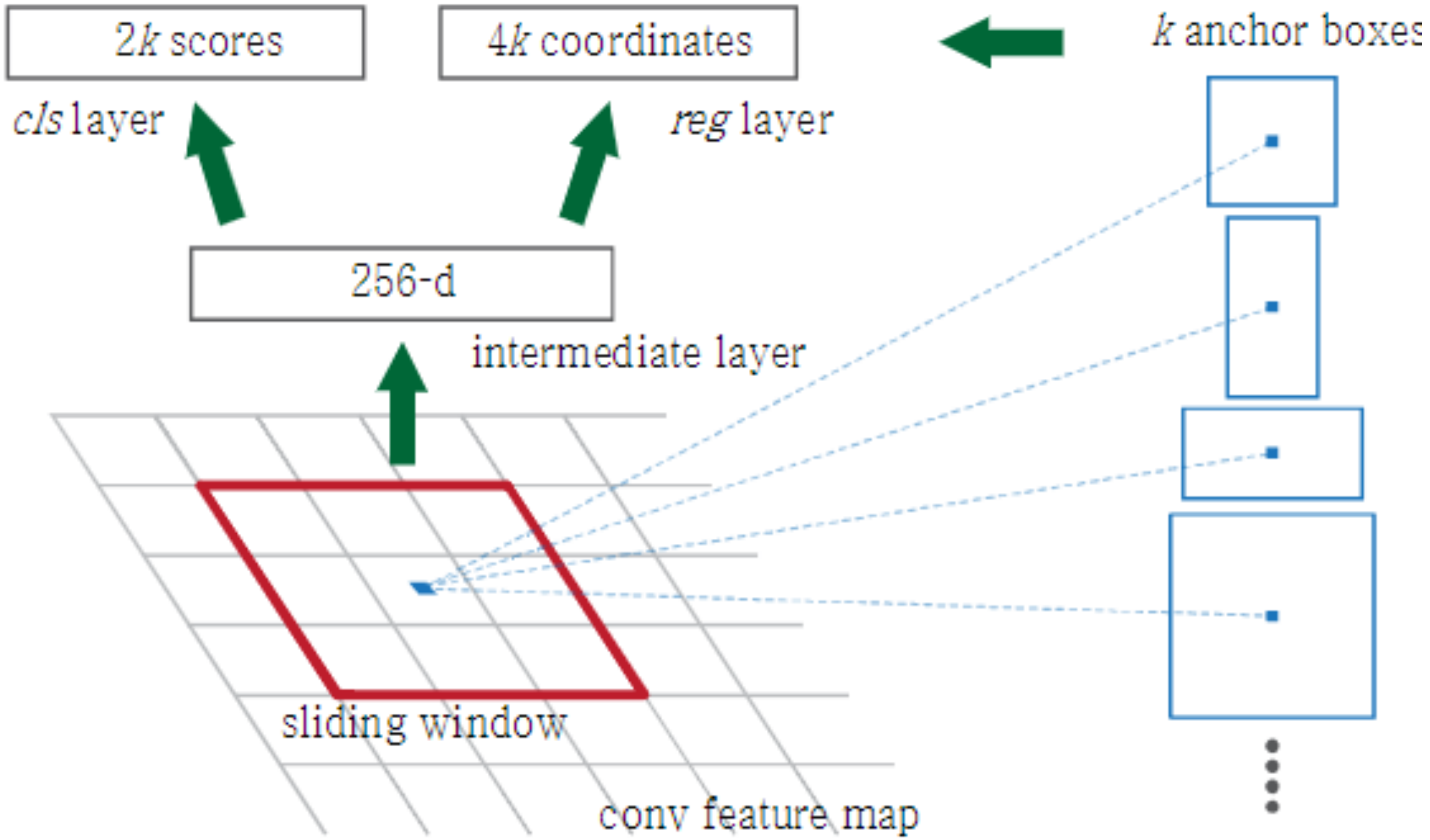
3.1.1 Anchors

- At each sliding-window location, we simultaneously predict multiple (denoted as k) region proposals
- So the reg layer has 4k outputs encoding the coordinates of k boxes, and the cls layer outputs 2k scores that estimate probability of object or not object for each proposal.
- For simplicity we implement the cls layer as a two-class softmax layer.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors



Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors

- The k proposals are parameterized relative to k reference boxes, which we call anchors.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors

- The k proposals are parameterized relative to k reference boxes, which we call anchors.
- An anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors

- The k proposals are parameterized relative to k reference boxes, which we call anchors.
- An anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio.
- By default we use 3 scales and 3 aspect ratios, yielding k = 9 anchors at each sliding position.

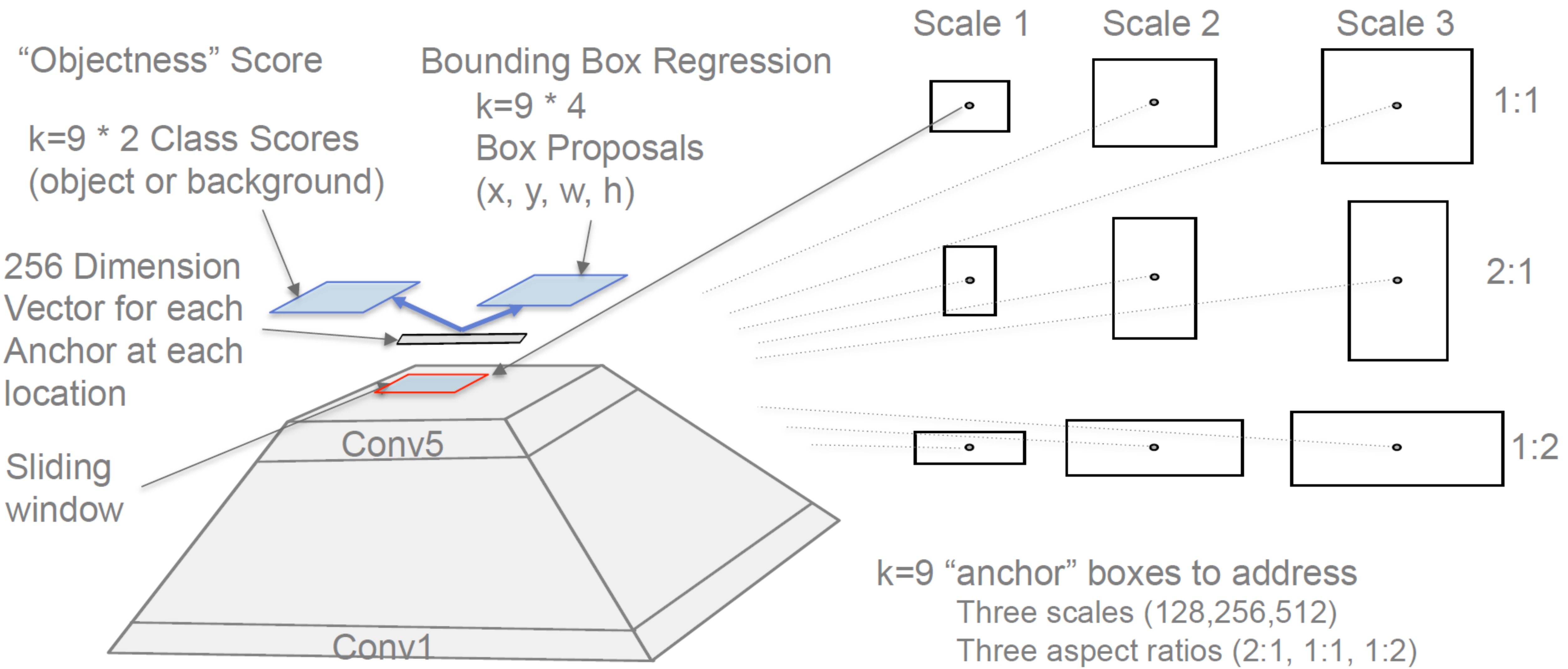
Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors

- The k proposals are parameterized relative to k reference boxes, which we call anchors.
- An anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio.
- By default we use 3 scales and 3 aspect ratios, yielding k = 9 anchors at each sliding position.
- For a convolutional feature map of a size WxH (typically ~2,400), there are WHk anchors in total.

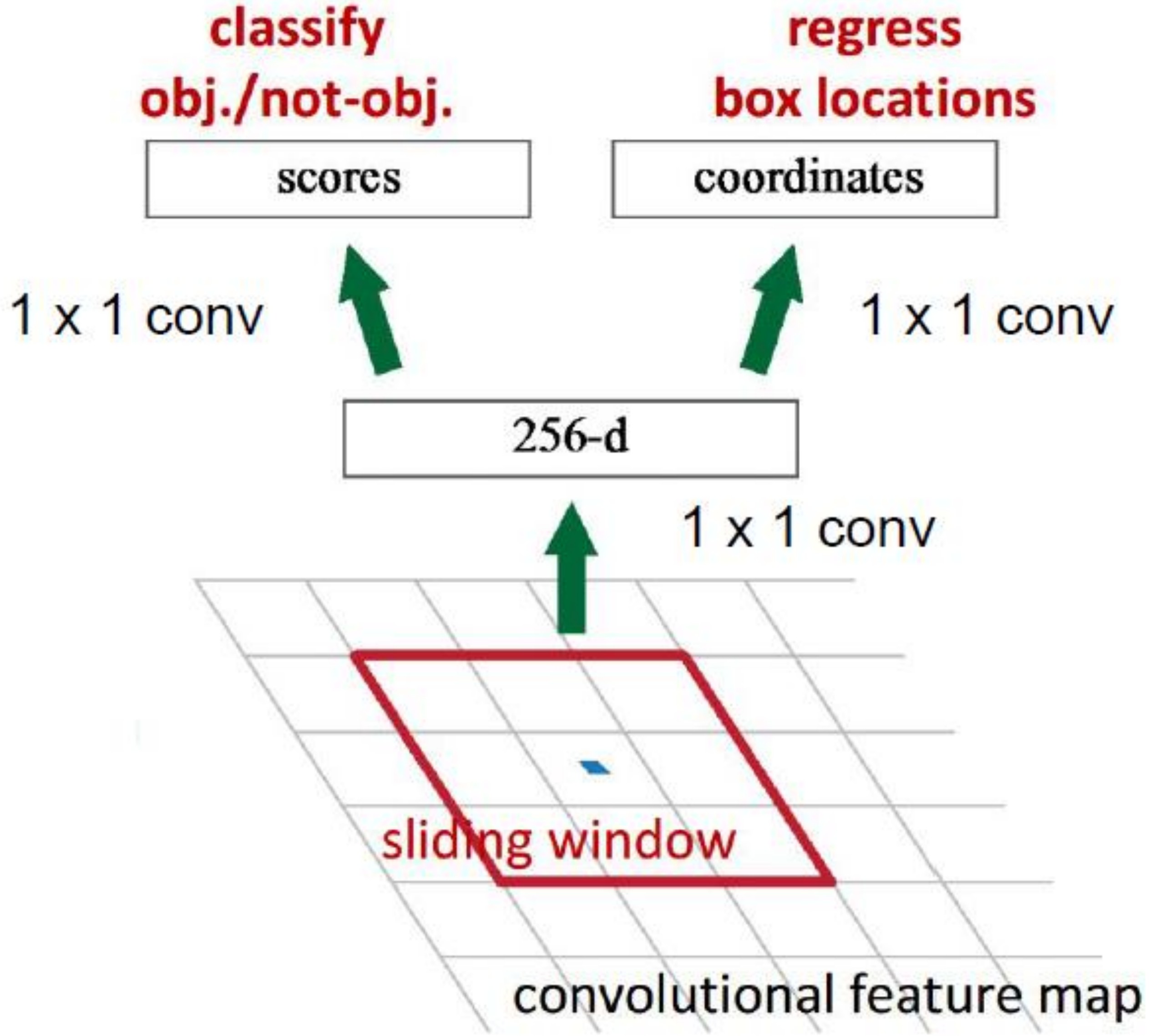
Region Proposal Network
Training Classifies Objectness & Regresses Bounding Boxes



k=9 "anchor" boxes to address

Three scales (128,256,512)

Three aspect ratios (2:1, 1:1, 1:2)


classify
obj./not-obj.
regress
box locations
scores
coordinates
1 x 1 conv
1 x 1 conv
256-d
1 x 1 conv
sliding window
convolutional feature map

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Translation-Invariant Anchors

- An important property of our approach is that it is translation invariant, both in terms of the anchors and the functions that compute proposals relative to the anchors.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Translation-Invariant Anchors

- An important property of our approach is that it is translation invariant, both in terms of the anchors and the functions that compute proposals relative to the anchors.
- If one translates an object in an image, the proposal should translate and the same function should be able to predict the proposal in either location.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Translation-Invariant Anchors

- An important property of our approach is that it is translation invariant, both in terms of the anchors and the functions that compute proposals relative to the anchors.
- If one translates an object in an image, the proposal should translate and the same function should be able to predict the proposal in either location.
- This translation-invariant property is guaranteed by our method.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Translation-Invariant Anchors

- The translation-invariant property also reduces the model size.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Translation-Invariant Anchors

- The translation-invariant property also reduces the model size.
- MultiBox has a (4+1)x800-dimensional fully-connected output layer, whereas our method has a (4+2)x9-dimensional convolutional output layer in the case of k = 9 anchors.

3.1 Region Proposal Network

3.1.1 Anchors : Translation-Invariant Anchors

- The translation-invariant property also reduces the model size.
- MultiBox has a (4+1)x800-dimensional fully-connected output layer, whereas our method has a (4+2)x9-dimensional convolutional output layer in the case of k = 9 anchors.
- We expect our method to have less risk of overfitting on small datasets, like PASCAL VOC.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Multi-scale Anchors as Regression References

- Our design of anchors presents a novel scheme for addressing multiple scales (and aspect ratios).

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Multi-scale Anchors as Regression References

- Our design of anchors presents a novel scheme for addressing multiple scales (and aspect ratios).
- Our anchor-based method is built on a pyramid of anchors, which is more cost-efficient.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Multi-scale Anchors as Regression References

- Our design of anchors presents a novel scheme for addressing multiple scales (and aspect ratios).
- Our anchor-based method is built on a pyramid of anchors, which is more cost-efficient.
- Our method classifies and regresses bounding boxes with reference to anchor boxes of multiple scales and aspect ratios.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Multi-scale Anchors as Regression References

- Our design of anchors presents a novel scheme for addressing multiple scales (and aspect ratios).
- Our anchor-based method is built on a pyramid of anchors, which is more cost-efficient.
- Our method classifies and regresses bounding boxes with reference to anchor boxes of multiple scales and aspect ratios.
- It only relies on images and feature maps of a single scale, and uses filters (sliding windows on the feature map) of a single size

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Multi-scale Anchors as Regression References

- Our design of anchors presents a novel scheme for addressing multiple scales (and aspect ratios).
- Our anchor-based method is built on a pyramid of anchors, which is more cost-efficient.
- Our method classifies and regresses bounding boxes with reference to anchor boxes of multiple scales and aspect ratios.
- It only relies on images and feature maps of a single scale, and uses filters (sliding windows on the feature map) of a single size
- We show by experiments the effects of this scheme for addressing multiple scales and sizes.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Multi-scale Anchors as Regression References

- Because of this multi-scale design based on anchors, we can simply use the convolutional features computed on a single-scale image, as is also done by the Fast R-CNN detector.

Faster R-cnn

3.1 Region Proposal Network

3.1.1 Anchors : Multi-scale Anchors as Regression References

- Because of this multi-scale design based on anchors, we can simply use the convolutional features computed on a single-scale image, as is also done by the Fast R-CNN detector.
- The design of multi-scale anchors is a key component for sharing features without extra cost for addressing scales.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For training RPNs, we assign a binary class label (of being an object or not) to each anchor.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For training RPNs, we assign a binary class label (of being an object or not) to each anchor.
- We assign a positive label to two kinds of anchors:

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For training RPNs, we assign a binary class label (of being an object or not) to each anchor.
- We assign a positive label to two kinds of anchors:

(i) the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For training RPNs, we assign a binary class label (of being an object or not) to each anchor.
- We assign a positive label to two kinds of anchors:

(i) the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box

(ii) an anchor that has an IoU overlap higher than 0.7 with any ground-truth box.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For training RPNs, we assign a binary class label (of being an object or not) to each anchor.
- We assign a positive label to two kinds of anchors:

(i) the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box

(ii) an anchor that has an IoU overlap higher than 0.7 with any ground-truth box.

- Note that a single ground-truth box may assign positive labels to multiple anchors

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For training RPNs, we assign a binary class label (of being an object or not) to each anchor.
- We assign a positive label to two kinds of anchors:

(i) the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box

(ii) an anchor that has an IoU overlap higher than 0.7 with any ground-truth box.

- Note that a single ground-truth box may assign positive labels to multiple anchors
- Usually the second condition is sufficient to determine the positive samples; but we still adopt the first condition for the reason that in some rare cases the second condition may find no positive sample.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- We assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- We assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes.
- Anchors that are neither positive nor negative do not contribute to the training objective.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- With these definitions, we minimize an objective function following the multi-task loss in Fast R-CNN

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- With these definitions, we minimize an objective function following the multi-task loss in Fast R-CNN

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- i is the index of an anchor in a mini-batch and pi is the predicted probability of anchor i being an object.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- With these definitions, we minimize an objective function following the multi-task loss in Fast R-CNN

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- i is the index of an anchor in a mini-batch and pi is the predicted probability of anchor i being an object.
- The ground-truth label p*i is 1 if the anchor is positive, and is 0 if the anchor is negative.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*).$$

- ti is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t*i is that of the ground-truth box associated with a positive anchor.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- ti is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t*i is that of the ground-truth box associated with a positive anchor.
- L_{cls} is log loss over two classes (object vs. not object).

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- ti is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t*i is that of the ground-truth box associated with a positive anchor.
- L_{cls} is log loss over two classes (object vs. not object).
- $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smooth L1) defined in [2].

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- ti is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t∗i is that of the ground-truth box associated with a positive anchor.
- L_{cls} is log loss over two classes (object vs. not object).
- $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smooth L1) defined in [2].
- regression loss is activated only for positive anchors (p∗i= 1) and is disabled otherwise (p∗i= 0).

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- ti is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t∗i is that of the ground-truth box associated with a positive anchor.
- Lcls is log loss over two classes (object vs. not object).
- $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smooth L1) defined in [2].
- regression loss is activated only for positive anchors (p∗i= 1) and is disabled otherwise (p∗i= 0).
- The outputs of the cls and reg layers consist of {pi} and {ti} respectively.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- The two terms are normalized by Ncls and Nreg and weighted by a balancing parameter lambda.

$N_{cls} = 256$ $N_{reg} \sim 2,400$ By default we set $\lambda = 10$

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- The two terms are normalized by Ncls and Nreg and weighted by a balancing parameter lambda.

$N_{cls} = 256$ $N_{reg} \sim 2,400$ By default we set $\lambda = 10$

- We show by experiments that the results are insensitive to the values of lambda in a wide range.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- The two terms are normalized by Ncls and Nreg and weighted by a balancing parameter lambda.

$N_{cls} = 256$ $N_{reg} \sim 2,400$ By default we set $\lambda = 10$

- We show by experiments that the results are insensitive to the values of lambda in a wide range.
- We also note that the normalization as above is not required and could be simplified.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For bounding box regression, we adopt the parameterizations of the 4 coordinates following:

$$
\begin{aligned}
t_{\mathrm{x}} &= (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t_{\mathrm{w}} &= \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}), \\
t_{\mathrm{x}}^{*} &= (x^{*} - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}}^{*} = (y^{*} - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t_{\mathrm{w}}^{*} &= \log(w^{*}/w_{\mathrm{a}}), \quad t_{\mathrm{h}}^{*} = \log(h^{*}/h_{\mathrm{a}}),
\end{aligned}
$$

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For bounding box regression, we adopt the parameterizations of the 4 coordinates following:

$$
\begin{aligned}
t_{\mathrm{x}} &= (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t_{\mathrm{w}} &= \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}), \\
t^{*}_{\mathrm{x}} &= (x^{*} - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t^{*}_{\mathrm{y}} = (y^{*} - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t^{*}_{\mathrm{w}} &= \log(w^{*}/w_{\mathrm{a}}), \quad t^{*}_{\mathrm{h}} = \log(h^{*}/h_{\mathrm{a}}),
\end{aligned}
$$

- where x, y, w, and h denote the box's center coordinates and its width and height.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For bounding box regression, we adopt the parameterizations of the 4 coordinates following:

$$
\begin{aligned}
t_{\mathrm{x}} &= (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t_{\mathrm{w}} &= \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}), \\
t_{\mathrm{x}}^{*} &= (x^{*} - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}}^{*} = (y^{*} - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t_{\mathrm{w}}^{*} &= \log(w^{*}/w_{\mathrm{a}}), \quad t_{\mathrm{h}}^{*} = \log(h^{*}/h_{\mathrm{a}}),
\end{aligned}
$$

- where x, y, w, and h denote the box's center coordinates and its width and height.
- Variables x, xa, and x* are for the predicted box, anchor box, and ground-truth box respectively.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- For bounding box regression, we adopt the parameterizations of the 4 coordinates following:

$$
\begin{aligned}
t_{\mathrm{x}} &= (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t_{\mathrm{w}} &= \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}), \\
t^*_{\mathrm{x}} &= (x^* - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t^*_{\mathrm{y}} = (y^* - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t^*_{\mathrm{w}} &= \log(w^*/w_{\mathrm{a}}), \quad t^*_{\mathrm{h}} = \log(h^*/h_{\mathrm{a}}),
\end{aligned}
$$

- where x, y, w, and h denote the box's center coordinates and its width and height.
- Variables x, xa, and x* are for the predicted box, anchor box, and ground-truth box respectively.
- This can be thought of as bounding-box regression from an anchor box to a nearby ground-truth box.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- our method achieves bounding-box regression by a different manner from previous RoI-based (Region of Interest) methods [1], [2].

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- our method achieves bounding-box regression by a different manner from previous RoI-based (Region of Interest) methods [1], [2].
- In [1], [2], bounding-box regression is performed on features pooled from arbitrarily sized RoIs, and the regression weights are shared by all region sizes.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- our method achieves bounding-box regression by a different manner from previous RoI-based (Region of Interest) methods [1], [2].
- In [1], [2], bounding-box regression is performed on features pooled from arbitrarily sized RoIs, and the regression weights are shared by all region sizes.
- In our formulation, the features used for regression are of the same spatial size (3 x 3) on the feature maps.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- our method achieves bounding-box regression by a different manner from previous RoI-based (Region of Interest) methods [1], [2].
- In [1], [2], bounding-box regression is performed on features pooled from arbitrarily sized RoIs, and the regression weights are shared by all region sizes.
- In our formulation, the features used for regression are of the same spatial size (3 x 3) on the feature maps.
- To account for varying sizes, a set of k bounding-box regressors are learned.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- our method achieves bounding-box regression by a different manner from previous RoI-based (Region of Interest) methods [1], [2].
- In [1], [2], bounding-box regression is performed on features pooled from arbitrarily sized RoIs, and the regression weights are shared by all region sizes.
- In our formulation, the features used for regression are of the same spatial size (3 x 3) on the feature maps.
- To account for varying sizes, a set of k bounding-box regressors are learned.
- Each regressor is responsible for one scale and one aspect ratio, and the k regressors do not share weights.

Faster R-cnn

3.1 Region Proposal Network

3.1.2 Loss Function

- our method achieves bounding-box regression by a different manner from previous RoI-based (Region of Interest) methods [1], [2].
- In [1], [2], bounding-box regression is performed on features pooled from arbitrarily sized RoIs, and the regression weights are shared by all region sizes.
- In our formulation, the features used for regression are of the same spatial size (3 x 3) on the feature maps.
- To account for varying sizes, a set of k bounding-box regressors are learned.
- Each regressor is responsible for one scale and one aspect ratio, and the k regressors do not share weights.
- As such, it is still possible to predict boxes of various sizes even though the features are of a fixed size/scale, thanks to the design of anchors.

Faster R-cnn

3.1 Region Proposal Network

3.1.3 Training RPNs

- The RPN can be trained end-to-end by back-propagation and stochastic gradient descent (SGD)

Faster R-cnn

3.1 Region Proposal Network

3.1.3 Training RPNs

- The RPN can be trained end-to-end by back-propagation and stochastic gradient descent (SGD)
- We follow the "image-centric" sampling strategy from [2] to train this network. Each mini-batch arises from a single image that contains many positive and negative example anchors.

Faster R-cnn

3.1 Region Proposal Network

3.1.3 Training RPNs

- The RPN can be trained end-to-end by back-propagation and stochastic gradient descent (SGD)
- We follow the “image-centric” sampling strategy from [2] to train this network. Each mini-batch arises from a single image that contains many positive and negative example anchors.
- It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominate.

Faster R-cnn

3.1 Region Proposal Network

3.1.3 Training RPNs

- The RPN can be trained end-to-end by back-propagation and stochastic gradient descent (SGD)
- We follow the “image-centric” sampling strategy from [2] to train this network. Each mini-batch arises from a single image that contains many positive and negative example anchors.
- It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominate.
- Instead, we randomly sample 256 anchors in an image to compute the loss function of a mini-batch, where the sampled positive and negative anchors have a ratio of up to 1:1.

Faster R-cnn

3.1 Region Proposal Network

3.1.3 Training RPNs

- The RPN can be trained end-to-end by back-propagation and stochastic gradient descent (SGD)
- We follow the “image-centric” sampling strategy from [2] to train this network. Each mini-batch arises from a single image that contains many positive and negative example anchors.
- It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominate.
- Instead, we randomly sample 256 anchors in an image to compute the loss function of a mini-batch, where the sampled positive and negative anchors have a ratio of up to 1:1.
- If there are fewer than 128 positive samples in an image, we pad the mini-batch with negative ones.

Faster R-cnn

3.1 Region Proposal Network

3.1.3 Training RPNs

We randomly initialize all new layers by drawing weights from a zero-mean Gaussian distribution with standard deviation 0.01. All other layers (*i.e.*, the shared convolutional layers) are initialized by pre-training a model for ImageNet classification [36], as is standard practice [5]. We tune all layers of the ZF net, and conv3_1 and up for the VGG net to conserve memory [2]. We use a learning rate of 0.001 for 60k mini-batches, and 0.0001 for the next 20k mini-batches on the PASCAL VOC dataset. We use a momentum of 0.9 and a weight decay of 0.0005 [37]. Our implementation uses Caffe [38].

3.2 Sharing Features for RPN and Fast R-CNN

- Thus far we have described how to train a network for region proposal generation, without considering the region-based object detection CNN that will utilize these proposals.

3.2 Sharing Features for RPN and Fast R-CNN

- Thus far we have described how to train a network for region proposal generation, without considering the region-based object detection CNN that will utilize these proposals.
- For the detection network, we adopt Fast R-CNN.

3.2 Sharing Features for RPN and Fast R-CNN

- Thus far we have described how to train a network for region proposal generation, without considering the region-based object detection CNN that will utilize these proposals.
- For the detection network, we adopt Fast R-CNN.
- We describe algorithms that learn a unified network composed of RPN and Fast R-CNN with shared convolutional layers.

3.2 Sharing Features for RPN and Fast R-CNN

- Both RPN and Fast R-CNN, trained independently, will modify their convolutional layers in different ways.

Faster R-cnn

3.2 Sharing Features for RPN and Fast R-CNN

- Both RPN and Fast R-CNN, trained independently, will modify their convolutional layers in different ways.
- We therefore need to develop a technique that allows for sharing convolutional layers between the two networks, rather than learning two separate networks.

Faster R-cnn

3.2 Sharing Features for RPN and Fast R-CNN

- Both RPN and Fast R-CNN, trained independently, will modify their convolutional layers in different ways.
- We therefore need to develop a technique that allows for sharing convolutional layers between the two networks, rather than learning two separate networks.
- We discuss three ways for training networks with features shared.

3.2 Sharing Features for RPN and Fast R-CNN

- (i) Alternating training. In this solution, we first train RPN, and use the proposals to train Fast R-CNN. The network tuned by Fast R-CNN is then used to initialize RPN, and this process is iterated. This is the solution that is used in all experiments in this paper.

3.2 Sharing Features for RPN and Fast R-CNN

- (i) Alternating training. In this solution, we first train RPN, and use the proposals to train Fast R-CNN. The network tuned by Fast R-CNN is then used to initialize RPN, and this process is iterated. This is the solution that is used in all experiments in this paper.
- (ii) Approximate joint training

3.2 Sharing Features for RPN and Fast R-CNN

- (i) Alternating training. In this solution, we first train RPN, and use the proposals to train Fast R-CNN. The network tuned by Fast R-CNN is then used to initialize RPN, and this process is iterated. This is the solution that is used in all experiments in this paper.
- (ii) Approximate joint training
- (iii) Non-approximate joint training

Faster R-cnn

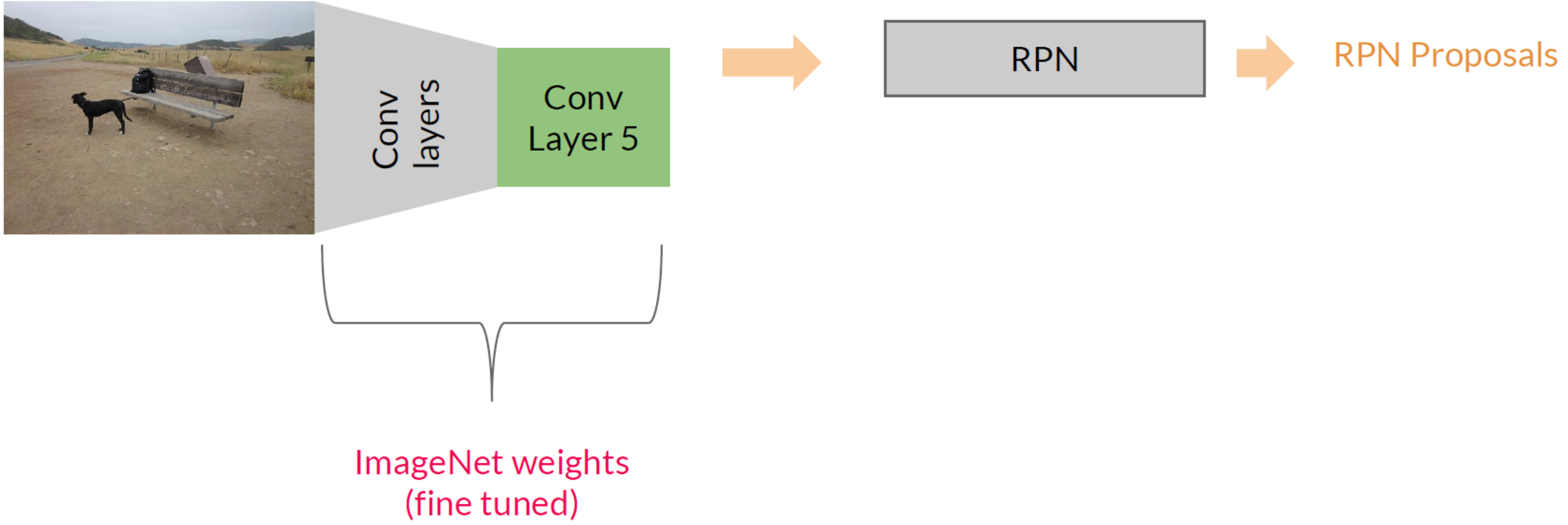
3.2 Sharing Features for RPN and Fast R-CNN

4-Step Alternating Training

- In this paper, we adopt a pragmatic 4-step training algorithm to learn shared features via alternating optimization.

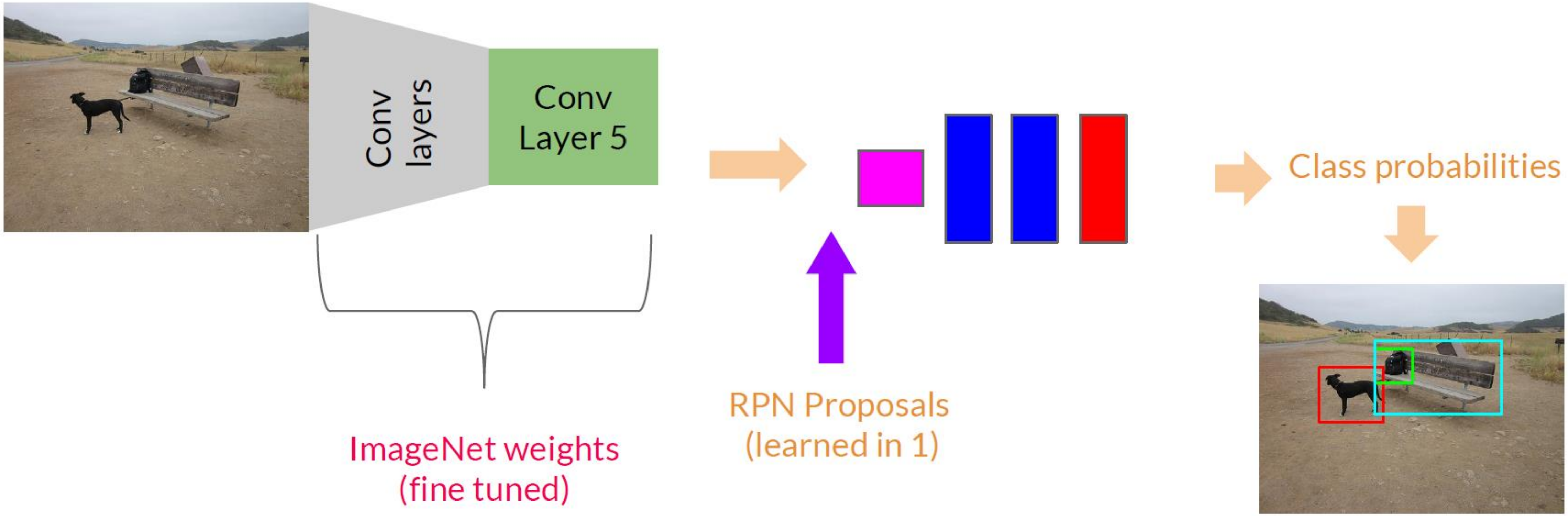
Faster R-CNN: 4-step training

Step 1: Train RPN initialized with an ImageNet pre-trained model.



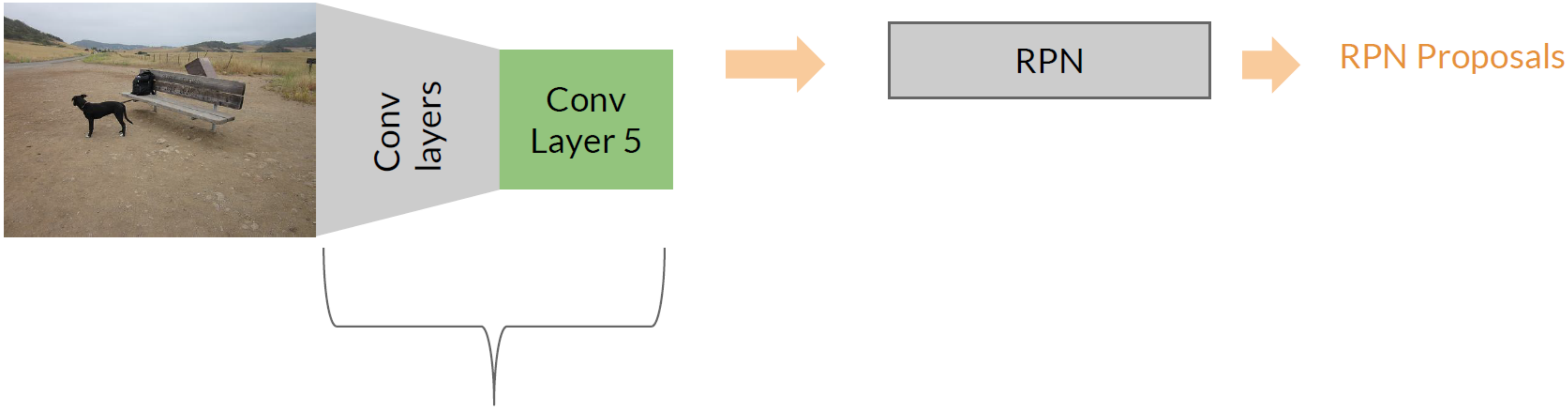
Faster R-CNN: 4-step training

Step 2: Train Fast R-CNN with learned RPN proposals.



Faster R-CNN: 4-step training

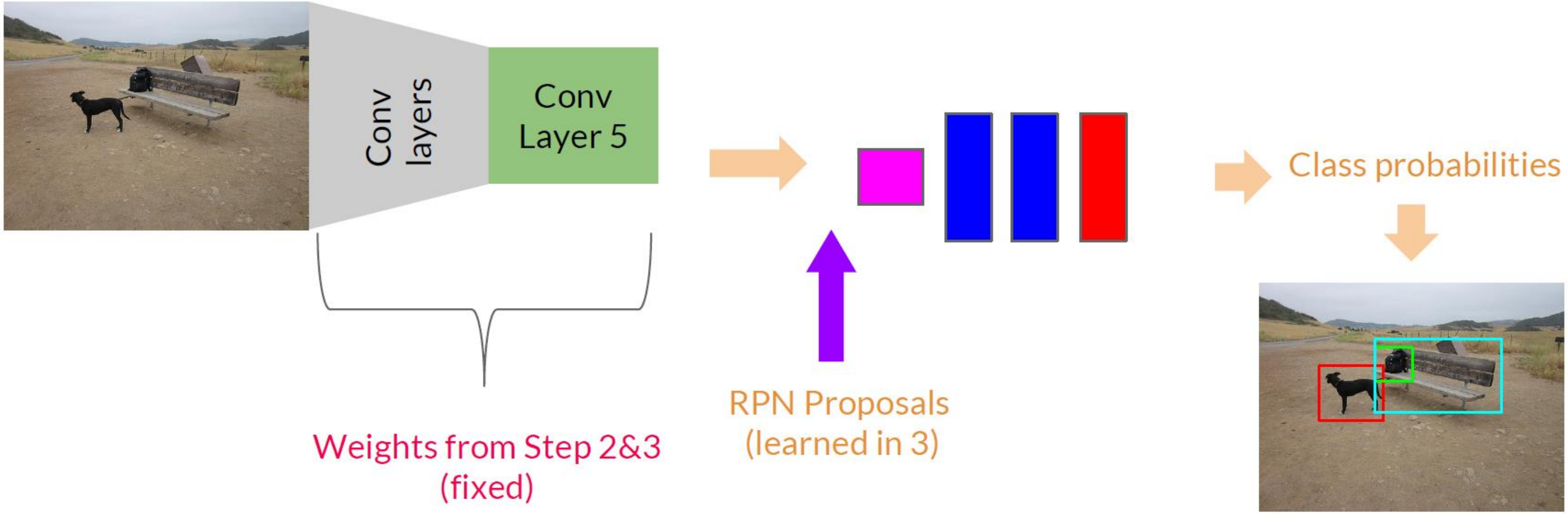
Step 3: The model trained in 2 is used to initialize RPN and train again.



Weights from Step 2
(fixed)

Faster R-CNN: 4-step training

Step 4: Fine tune FC layers of Fast R-CNN using same shared convolutional layers as in 3.



Faster R-cnn

3.2 Sharing Features for RPN and Fast R-CNN

4-Step Alternating Training

- In this paper, we adopt a pragmatic 4-step training algorithm to learn shared features via alternating optimization.
- As such, both networks share the same convolutional layers and form a unified network.

Faster R-cnn

3.2 Sharing Features for RPN and Fast R-CNN

4-Step Alternating Training

- In this paper, we adopt a pragmatic 4-step training algorithm to learn shared features via alternating optimization.
- As such, both networks share the same convolutional layers and form a unified network.
- A similar alternating training can be run for more iterations, but we have observed negligible improvements.

Faster R-cnn

5 CONCLUSION

- We have presented RPNs for efficient and accurate region proposal generation.
- By sharing convolute features with the down-stream detection network, the region proposal step is nearly cost-free.
- Our method enables a unified, deep-learning-based object detection system to run at near real-time frame rates.
- The learned RPN also improves region proposal quality and thus the overall object detection accuracy.

Thank you!

Faster R-CNN