CS7015 (Deep Learning) : Lecture 12

Object Detection: R-CNN, Fast R-CNN, Faster R-CNN, You Only Look Once (YOLO)

Mitesh M. Khapra

Department of Computer Science and Engineering
Indian Institute of Technology Madras

Acknowledgements

- Some images borrowed from Ross Girshick's original slides on RCNN, Fast RCNN, etc.
- Some ideas borrowed from the presentation of Kaustav Kundu*

* Deep Object Detection

Module 12.1 : Introduction to object detection

- So far we have looked at Image Classification
- We will now move on to another Image Processing Task - *Object Detection*

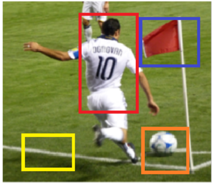
Task Image classification

Task Image classification

Output Car

Task	Image classification	Object Detection
Output	Car	

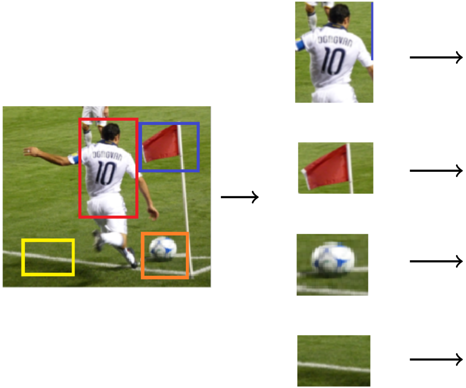
Task	Image classification	Object Detection
Output	Car	Car, exact bounding box containing car



$\longrightarrow$

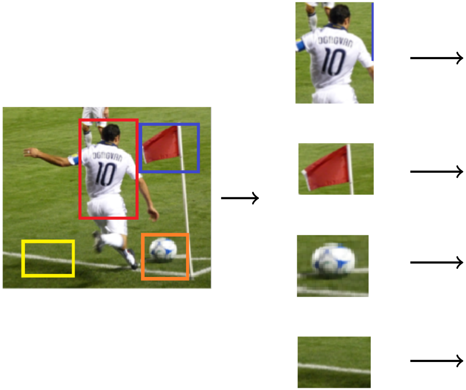
- Let us see a typical pipeline for *object detection*

Region proposals

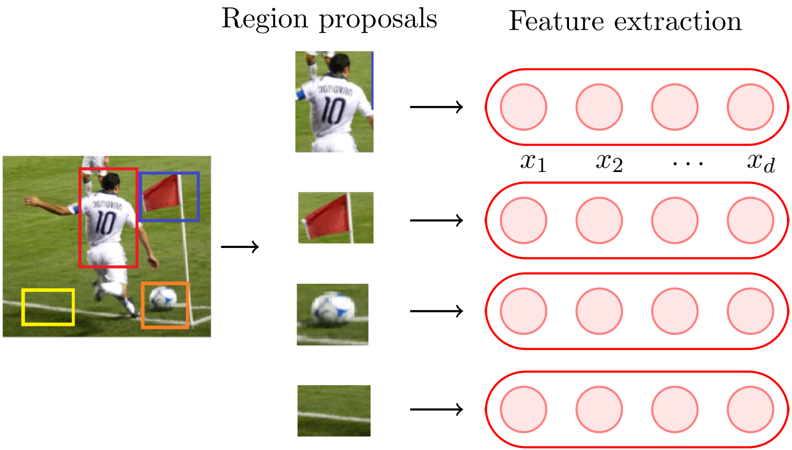


- Let us see a typical pipeline for *object detection*
- It starts with a region proposal stage where we identify potential regions which may contain objects

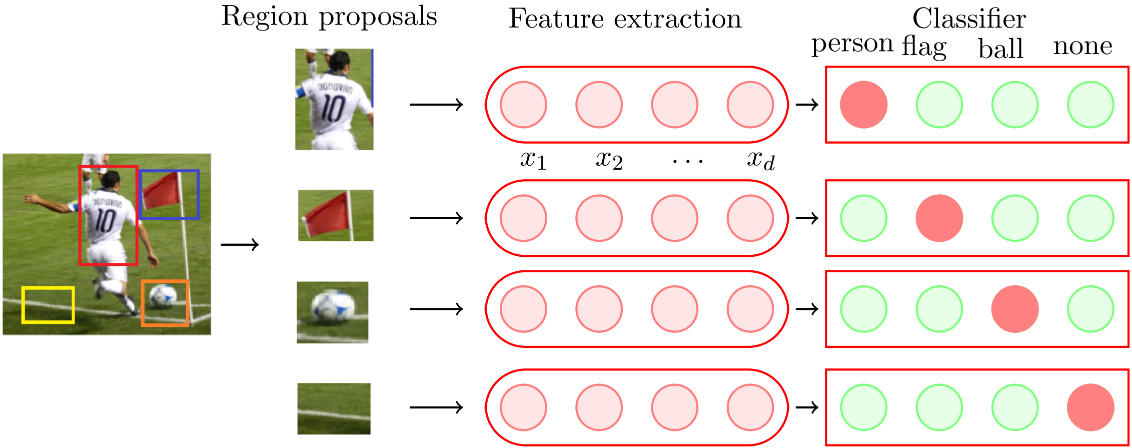
Region proposals



- We could think of these regions as mini-images

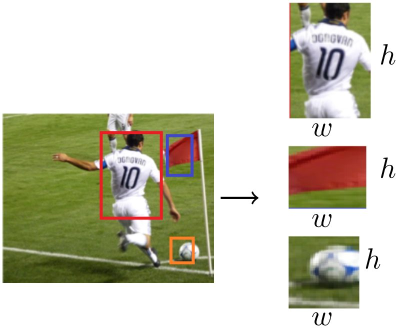


- We could think of these regions as mini-images
- We extract features(SIFT, HOG, CNNs) from these mini-images



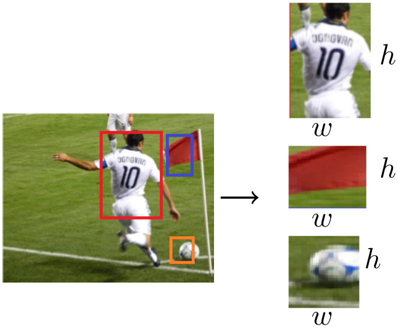
- Pass these through a standard image classifer to determine the class

Region proposals

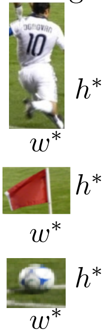


- In addition we would also like to correct the proposed bounding boxes

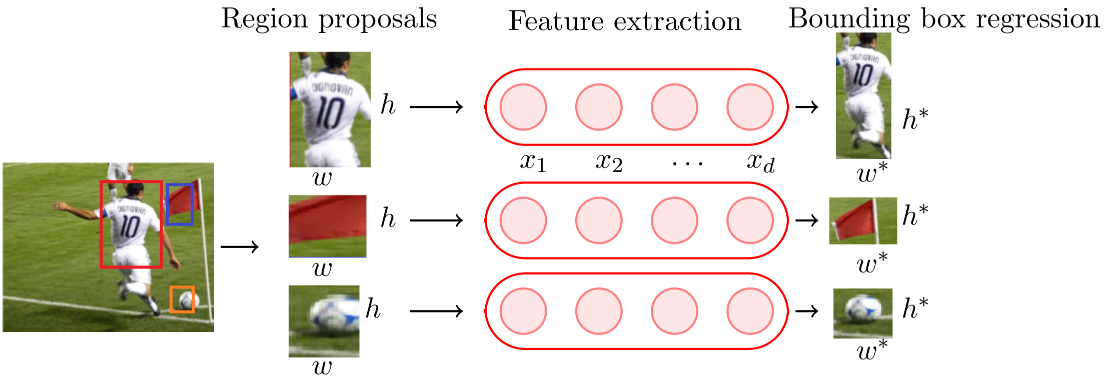
Region proposals



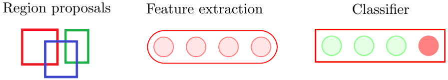
Bounding box regression



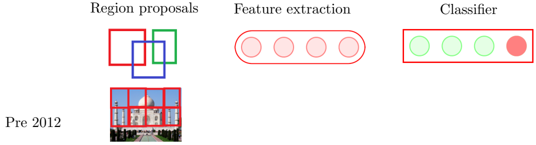
- In addition we would also like to correct the proposed bounding boxes
- This is posed as a regression problem (for example, we would like to predict w^*, h^* from the proposed w and h)



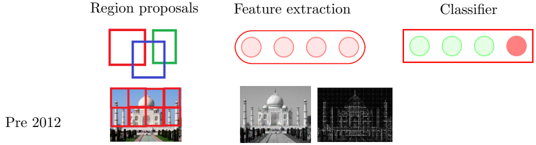
- In addition we would also like to correct the proposed bounding boxes
- This is posed as a regression problem (for example, we would like to predict w^*, h^* from the proposed w and h)



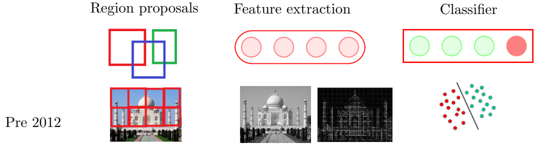
- Let us see how these three components have evolved over time



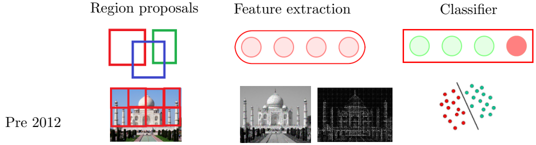
- Let us see how these three components have evolved over time
- Propose all possible regions in the image of varying sizes (almost brute force)



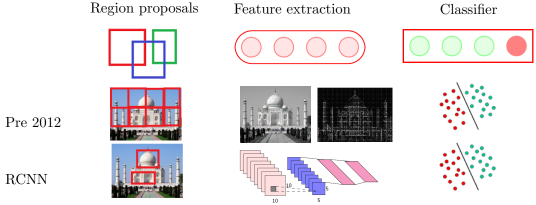
- Let us see how these three components have evolved over time
- Propose all possible regions in the image of varying sizes (almost brute force)
- Use handcrafted features (SIFT, HOG)



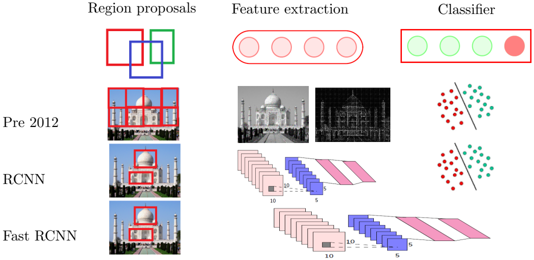
- Let us see how these three components have evolved over time
- Propose all possible regions in the image of varying sizes (almost brute force)
- Use handcrafted features (SIFT, HOG)
- Train a linear classifier using these features



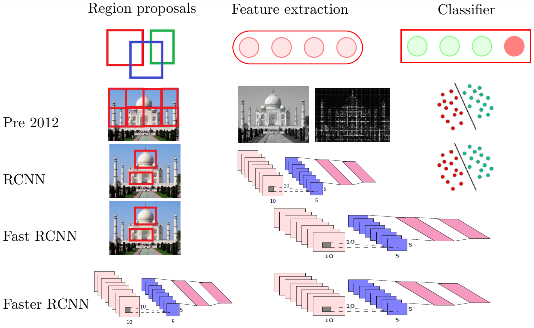
- Let us see how these three components have evolved over time
- Propose all possible regions in the image of varying sizes (almost brute force)
- Use handcrafted features (SIFT, HOG)
- Train a linear classifier using these features
- We will now see three algorithms that progressively improve these components



- Let us see how these three components have evolved over time
- Propose all possible regions in the image of varying sizes (almost brute force)
- Use handcrafted features (SIFT, HOG)
- Train a linear classifier using these features
- We will now see three algorithms that progressively improve these components

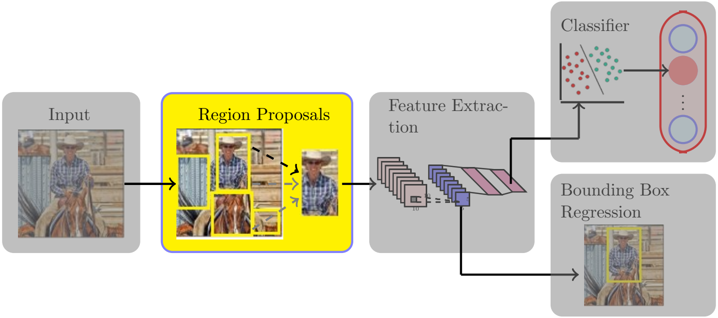


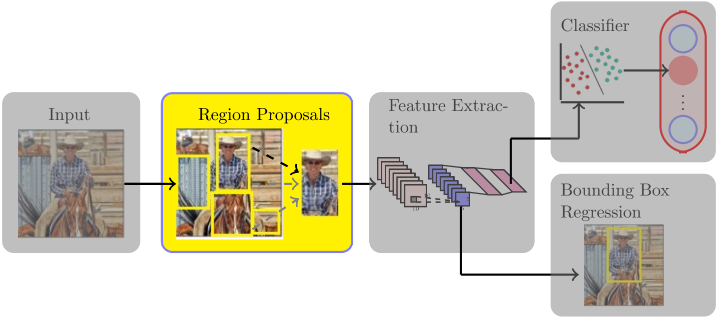
- Let us see how these three components have evolved over time
- Propose all possible regions in the image of varying sizes (almost brute force)
- Use handcrafted features (SIFT, HOG)
- Train a linear classifier using these features
- We will now see three algorithms that progressively improve these components



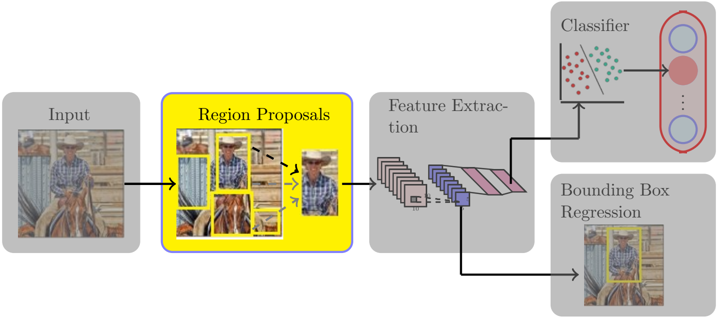
- Let us see how these three components have evolved over time
- Propose all possible regions in the image of varying sizes (almost brute force)
- Use handcrafted features (SIFT, HOG)
- Train a linear classifier using these features
- We will now see three algorithms that progressively improve these components

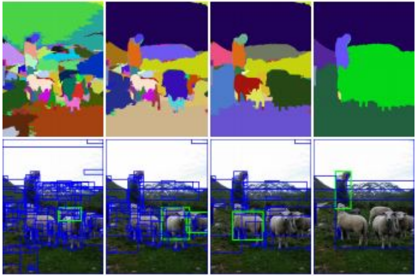
Module 12.2 : RCNN model for object detection


Input
Region Proposals
Feature Extraction
Classifier
Bounding Box Regression

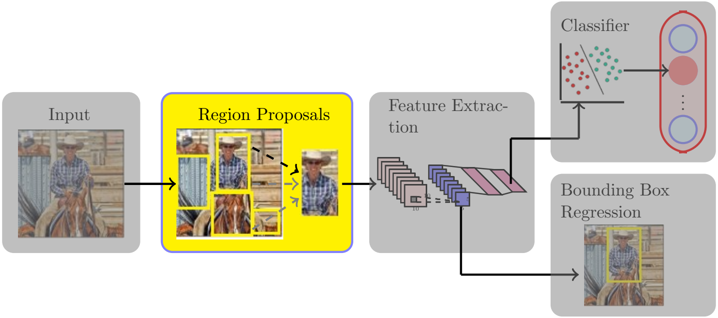


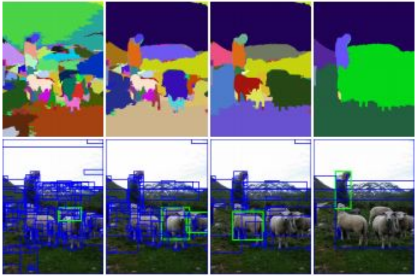
- **Selective Search** for region proposals



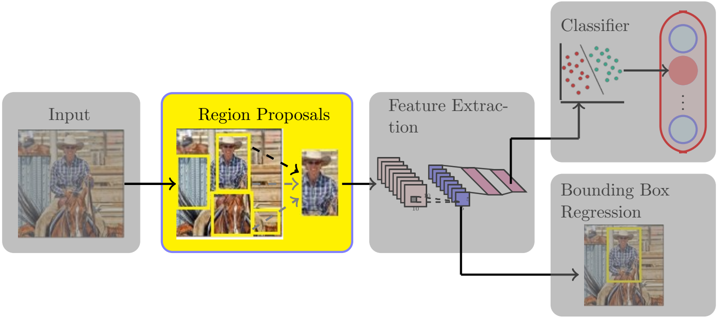


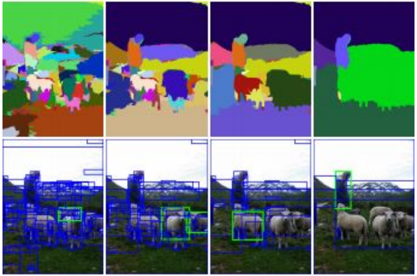
- **Selective Search** for region proposals
- Does hierarchical clustering at different scales



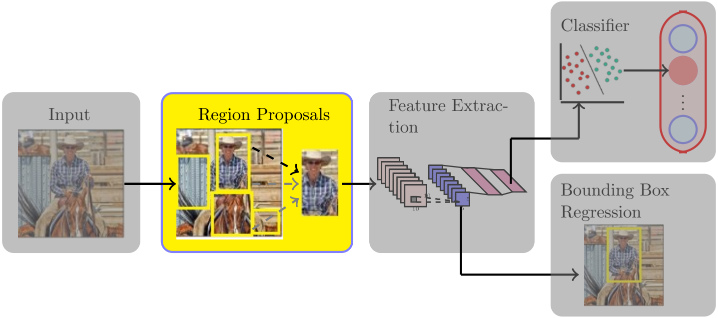


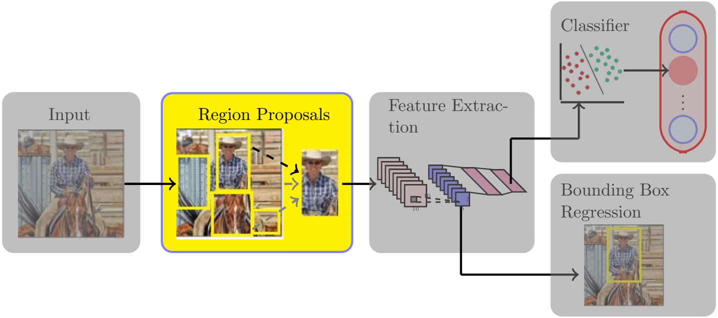
- **Selective Search** for region proposals
- Does hierarchical clustering at different scales
- For example the figures from left to right show clusters of increasing sizes

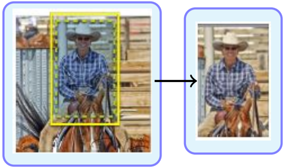




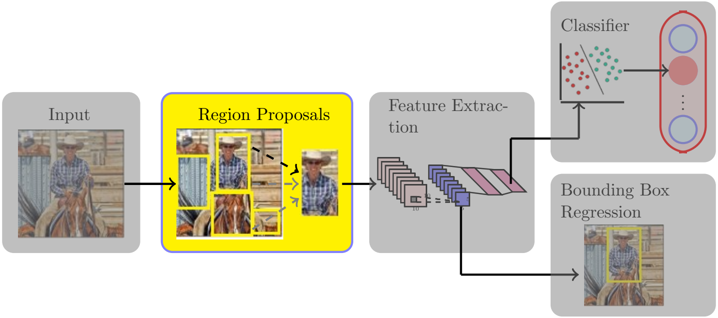
- **Selective Search** for region proposals
- Does hierarchical clustering at different scales
- For example the figures from left to right show clusters of increasing sizes
- Such a hierarchical clustering is important as we may find different objects at different scales

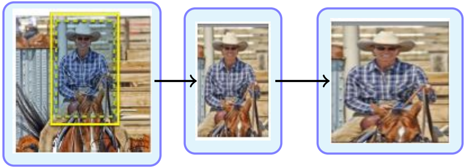

Input
Region Proposals
Feature Extraction
Classifier
Bounding Box Regression



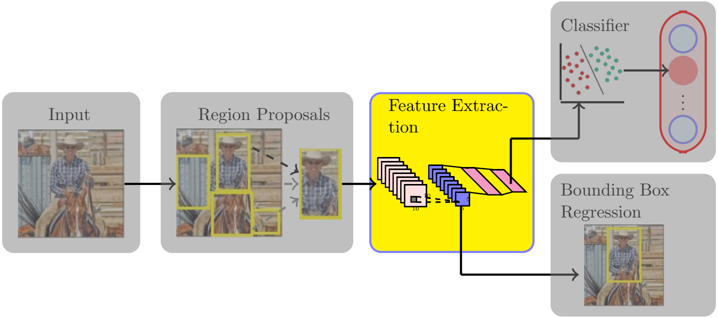


- Proposed regions are cropped to form mini images

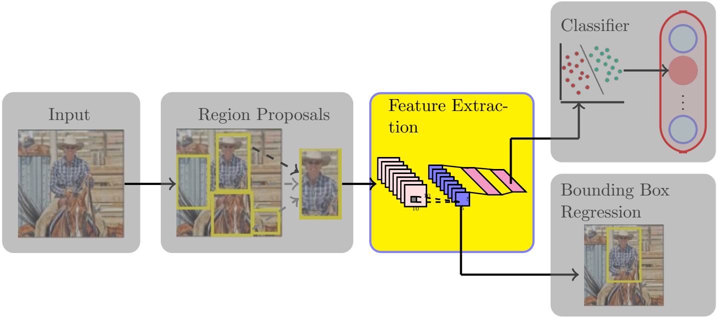


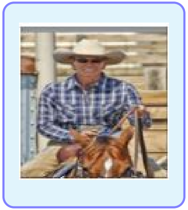


- Proposed regions are cropped to form mini images
- Each mini image is scaled to match the CNN's (feature extractor) input size

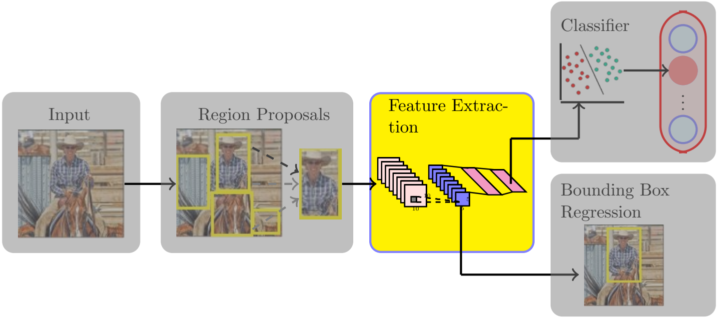


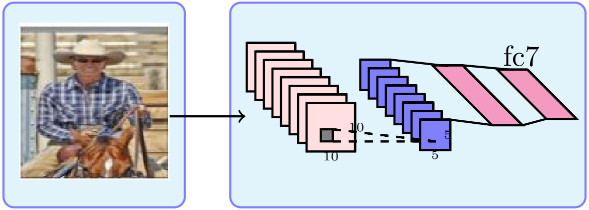
- For feature extraction any CNN trained for Image Classification can be used (AlexNet/ VGGNet etc.)



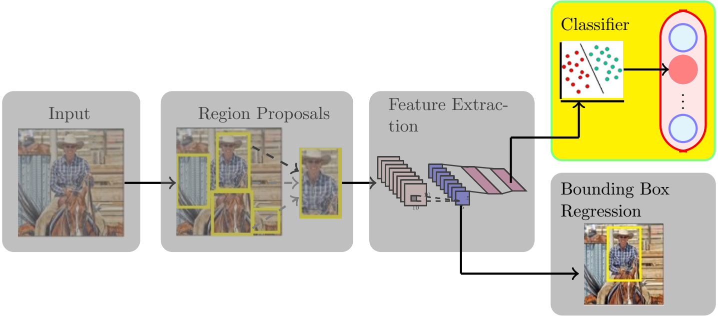


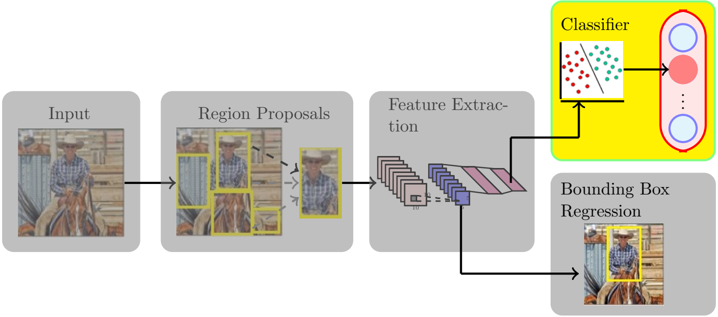
- For feature extraction any CNN trained for Image Classification can be used (AlexNet/ VGGNet etc.)
- Outputs from fc7 layer are taken as features

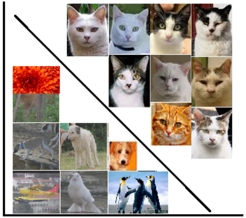




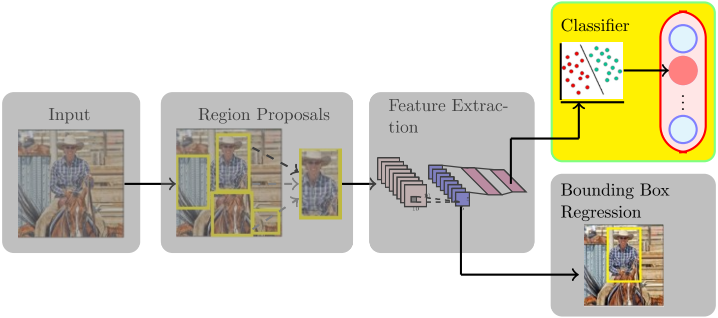
- For feature extraction any CNN trained for Image Classification can be used (AlexNet/ VGGNet etc.)
- Outputs from fc7 layer are taken as features
- CNN is fine tuned using ground truth (cropped) object images

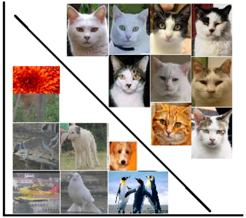

Input
Region Proposals
Feature Extraction
Classifier
Bounding Box Regression



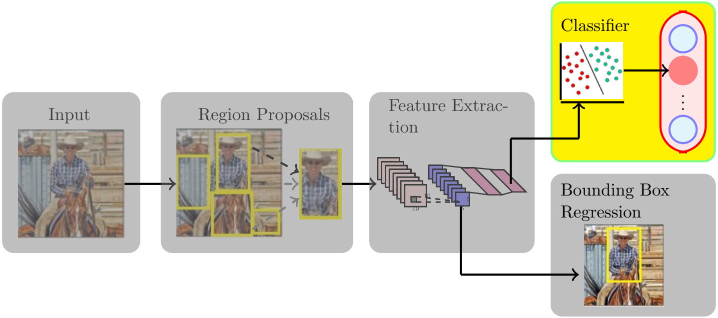


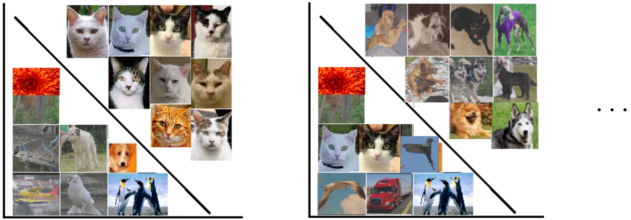
- Linear models (SVMs) are used for classification



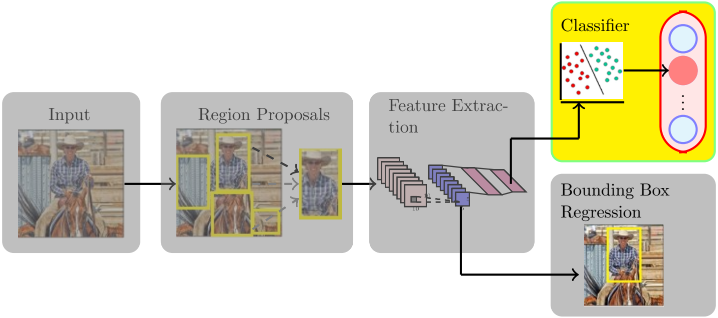


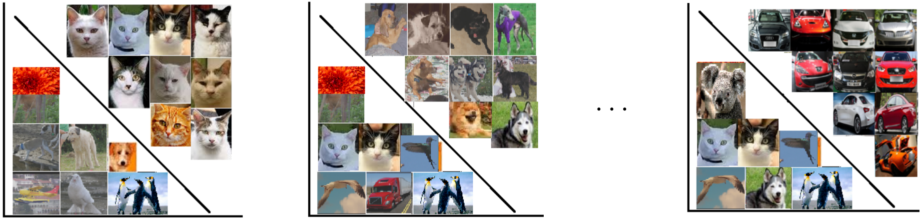
- Linear models (SVMs) are used for classification (1 model per class)



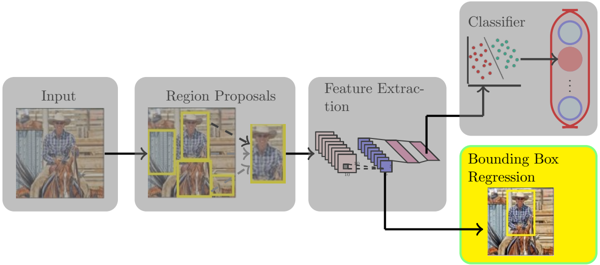


- Linear models (SVMs) are used for classification (1 model per class)

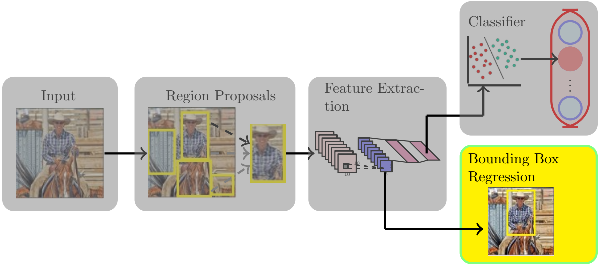




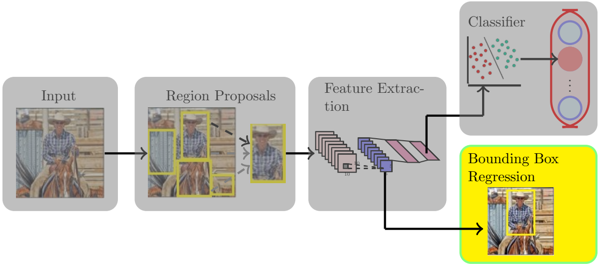
- Linear models (SVMs) are used for classification (1 model per class)



Input
Region Proposals
Feature Extraction
Classifier
Bounding Box Regression



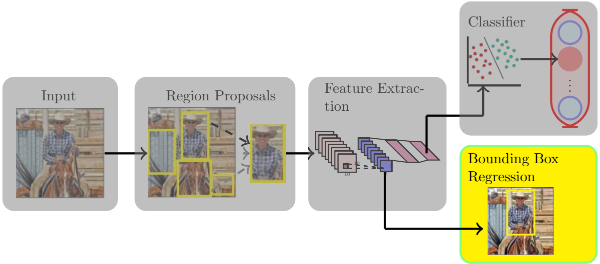
Proposed Box

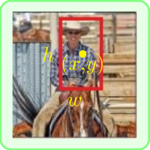


Proposed Box

True Box

- The proposed regions may not be perfect

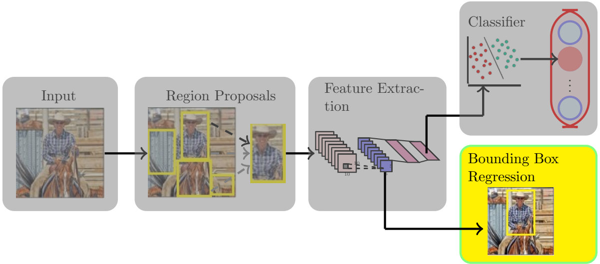




Proposed Box

True Box

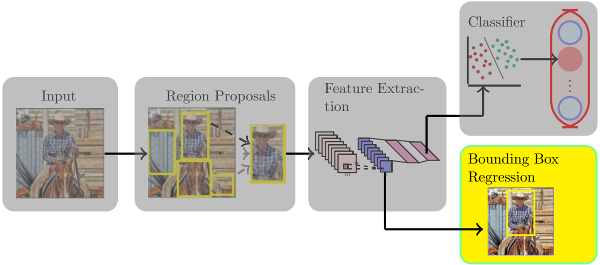
- The proposed regions may not be perfect
- We want to learn four regression models which will learn to predict x^*, y^*, w^*, h^*



Proposed Box

True Box

- The proposed regions may not be perfect
- We want to learn four regression models which will learn to predict x^*, y^*, w^*, h^*
- We will see their respective objective functions

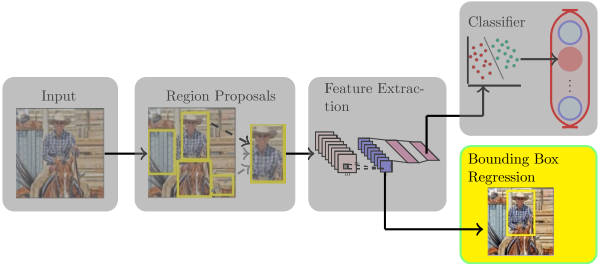


$$\min \sum_{i=1}^{N} \left(\frac{x^* - x}{w} - w_1^T z \right)^2$$

Proposed Box

True Box

z : features from pool5 layer of the network



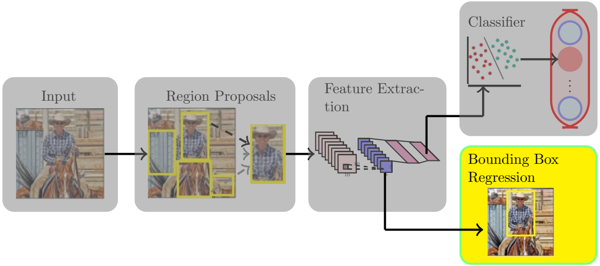
$$\min \sum_{i=1}^{N} \left(\frac{x^* - x}{w} - w_1^T z \right)^2$$

Proposed Box

True Box

z : features from pool5 layer of the network

- $\frac{x^* - x}{w}$ is the normalized difference between proposed x and true x^*



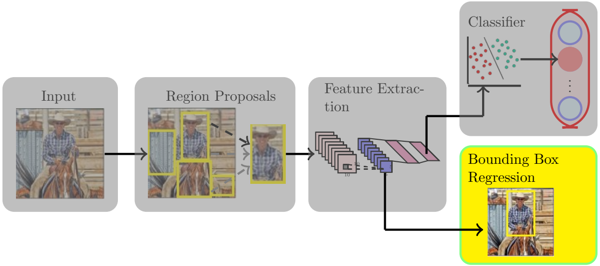
$$\min \sum_{i=1}^{N} \left(\frac{x^* - x}{w} - w_1^T z \right)^2$$

Proposed Box

True Box

z : features from pool5 layer of the network

- $\frac{x^* - x}{w}$ is the normalized difference between proposed x and true x^*
- If we can predict this difference we can compute x^*



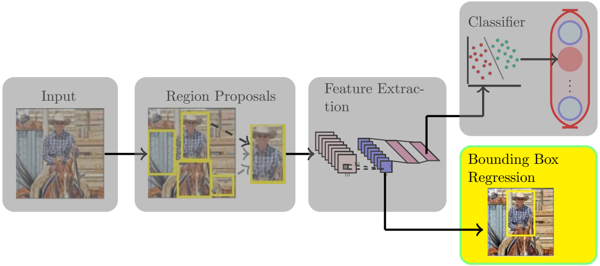
$$\min \sum_{i=1}^{N} \left(\frac{x^* - x}{w} - w_1^T z \right)^2$$

Proposed Box

True Box

z : features from pool5 layer of the network

- $\frac{x^* - x}{w}$ is the normalized difference between proposed x and true x^*
- If we can predict this difference we can compute x^*
- The model predicts $w_1^T z$ as this difference



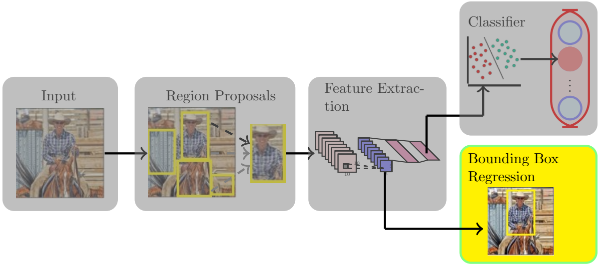
$$\min \sum_{i=1}^{N} \left(\frac{x^* - x}{w} - w_1^T z \right)^2$$

Proposed Box

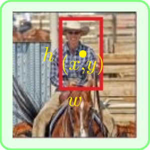
True Box

z : features from pool5 layer of the network

- $\frac{x^*-x}{w}$ is the normalized difference between proposed x and true x^*
- If we can predict this difference we can compute x^*
- The model predicts $w_1^T z$ as this difference
- The above objective function minimize the difference between the predicted and the actual value



$$\min \sum_{i=1}^{N} \left(\frac{y^* - y}{h} - w_2^T z \right)^2$$

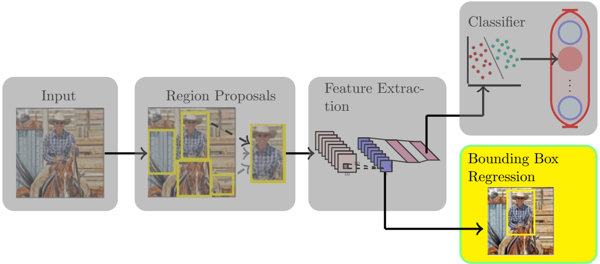


Proposed Box

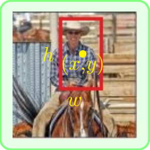
True Box

z : features from pool5 layer of the network

- Similarly for y



$$\min \sum_{i=1}^{N} \left(\ln \left(\frac{w^*}{w} \right) - w_3^T z \right)^2$$

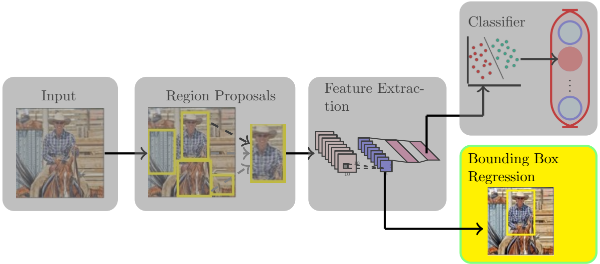


Proposed Box

True Box

z : features from pool5 layer of the network

- Similarly for w



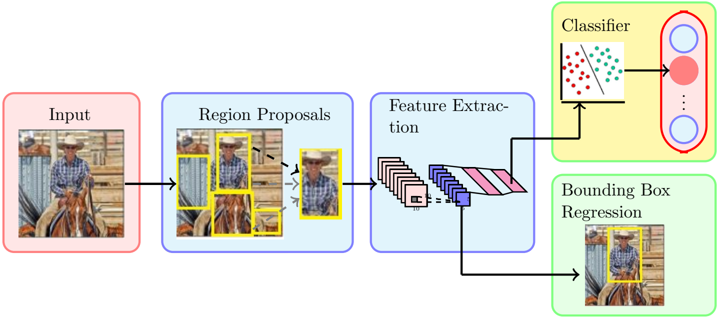
$$\min \sum_{i=1}^{N} \left(\ln\left(\frac{h^*}{h}\right) - w_4^T z \right)^2$$

Proposed Box

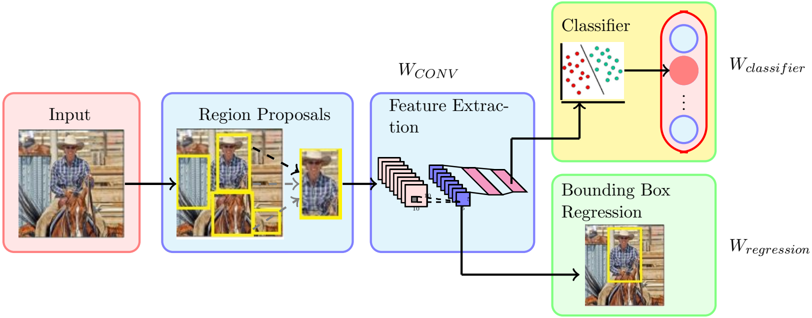
True Box

- Similarly for h

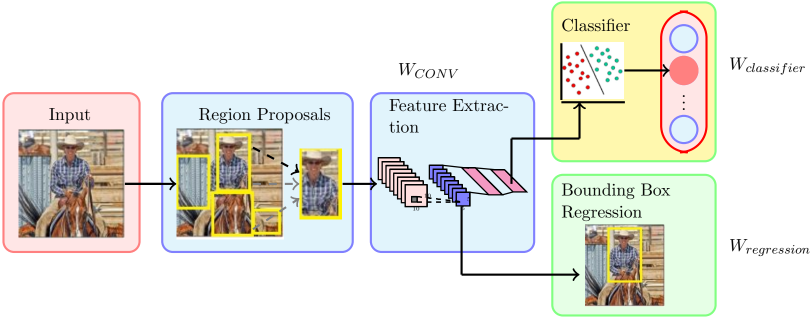
z : features from pool5 layer of the network



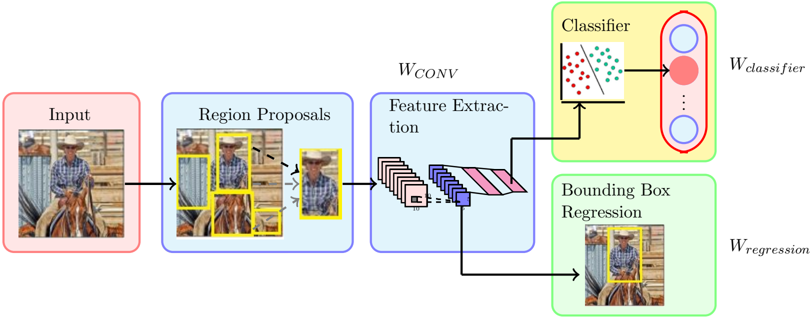
- What are the parameters of this model?



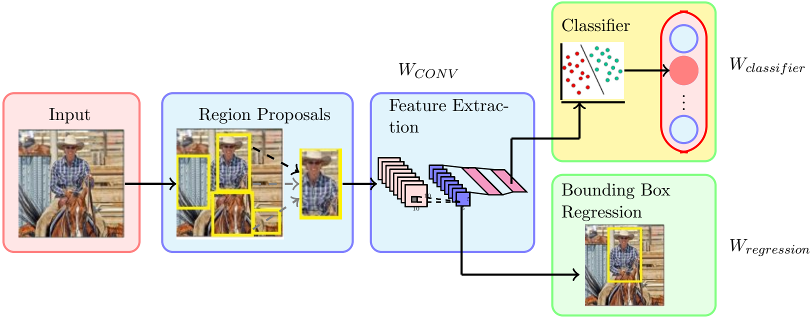
- What are the parameters of this model?



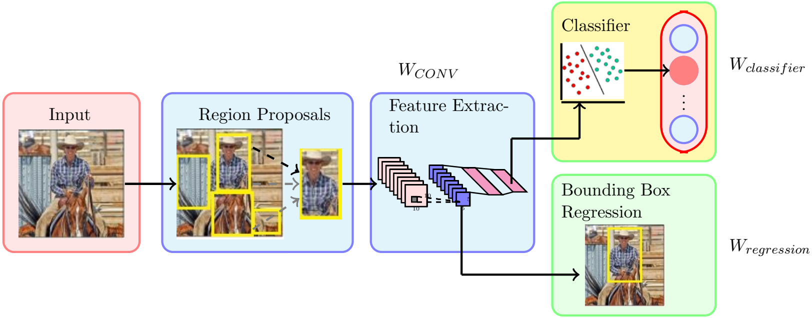
- What are the parameters of this model?
- W_{CONV} is taken as it is from a CNN trained for Image classification (say on ImageNet)



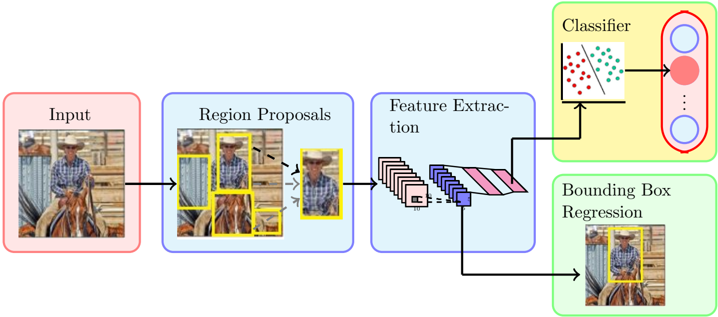
- What are the parameters of this model?
- W_{CONV} is taken as it is from a CNN trained for Image classification (say on ImageNet)
- W_{CONV} is then fine tuned using ground truth (cropped) object images



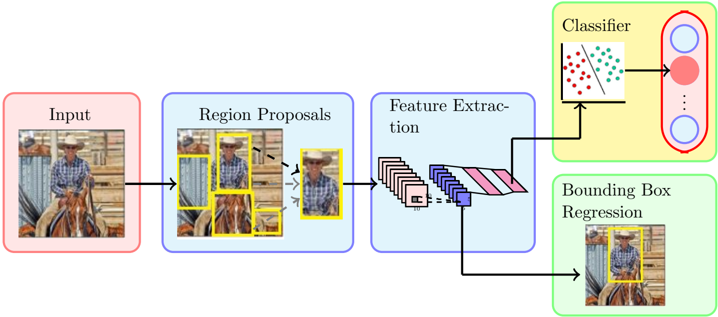
- What are the parameters of this model?
- W_{CONV} is taken as it is from a CNN trained for Image classification (say on ImageNet)
- W_{CONV} is then fine tuned using ground truth (cropped) object images
- $W_{classifier}$ is learned using ground truth (cropped) object images



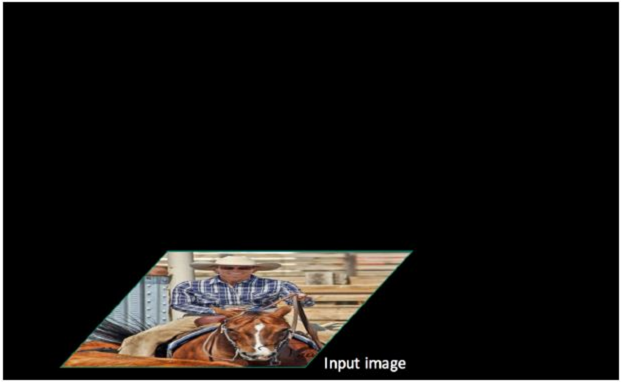
- What are the parameters of this model?
- W_{CONV} is taken as it is from a CNN trained for Image classification (say on ImageNet)
- W_{CONV} is then fine tuned using ground truth (cropped) object images
- $W_{classifier}$ is learned using ground truth (cropped) object images
- $W_{regression}$ is learned using ground truth bounding boxes



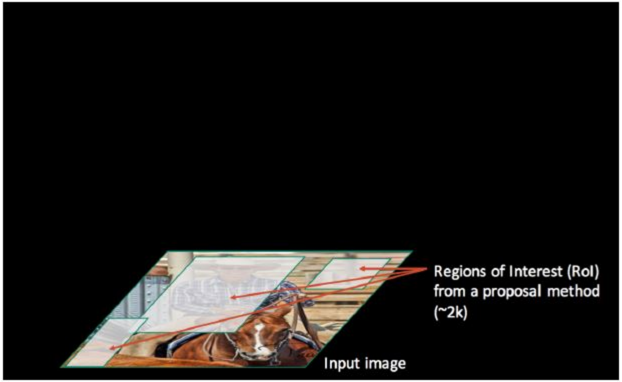
- What is the computational cost for processing one image at test time?



- What is the computational cost for processing one image at test time?
- Inference Time = Proposal Time + # Proposals $\times$ Convolution Time + # Proposals $\times$ classification + # Proposals $\times$ regression

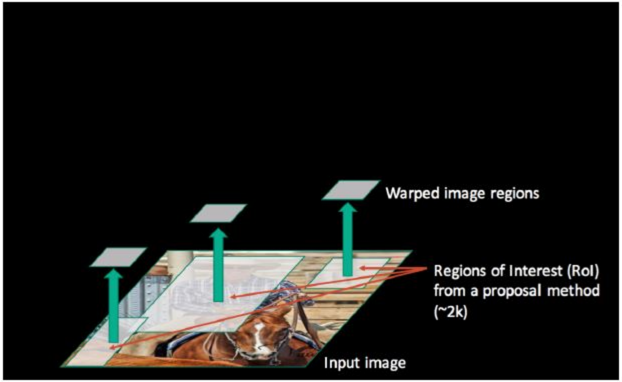


Source: *Ross Girshick*



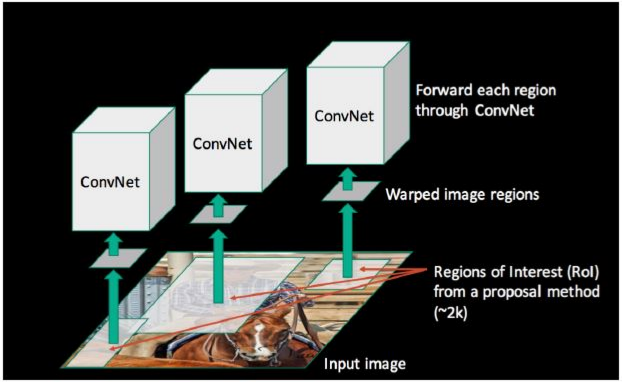
Source: *Ross Girshick*

- On average selective search gives 2K region proposal



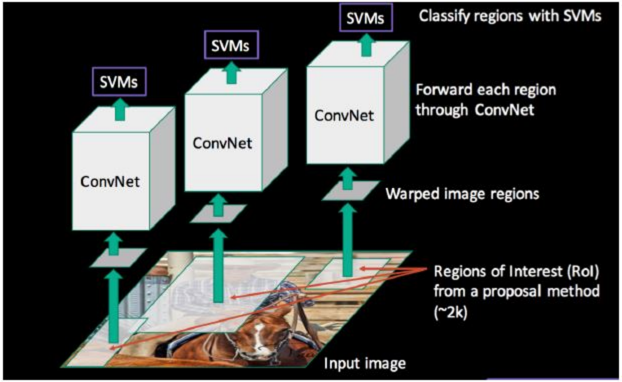
Source: *Ross Girshick*

- On average selective search gives 2K region proposal



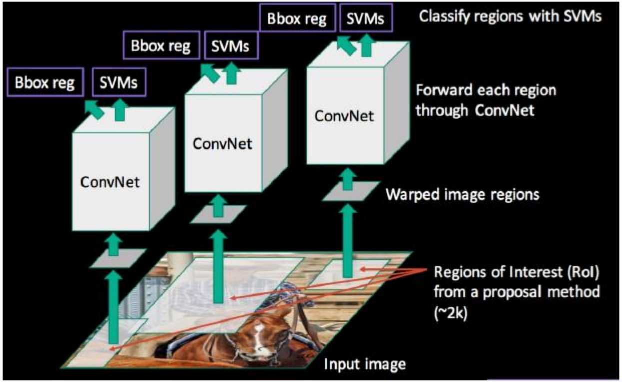
Source: *Ross Girshick*

- On average selective search gives 2K region proposal
- Each of these pass through the CNN for feature extraction



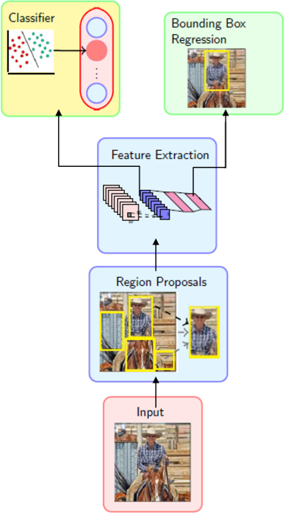
Source: *Ross Girshick*

- On average selective search gives 2K region proposal
- Each of these pass through the CNN for feature extraction
- Followed by classification and regression



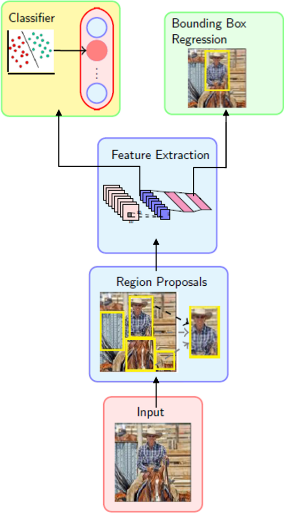
Source: *Ross Girshick*

- On average selective search gives 2K region proposal
- Each of these pass through the CNN for feature extraction
- Followed by classification and regression



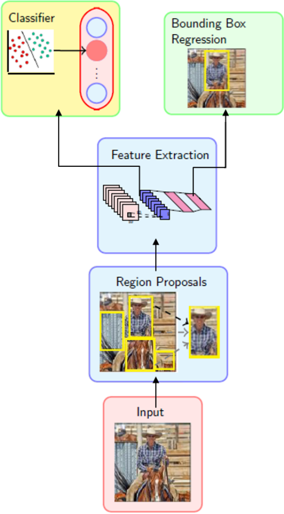
- No joint learning

[1]Source: *Ross Girshick*
[1]Using VGG-Net



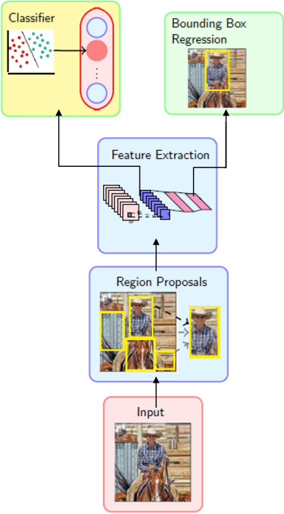
- No joint learning
- Use ad hoc training objectives

[1]Source: *Ross Girshick*
[1]Using VGG-Net



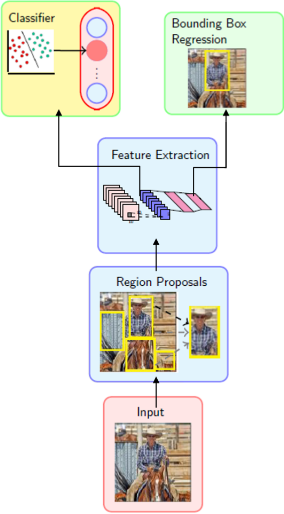
- No joint learning
- Use ad hoc training objectives
  - Fine tune network with softmax classifier (log loss)

[1] Source: *Ross Girshick*
[1] Using VGG-Net



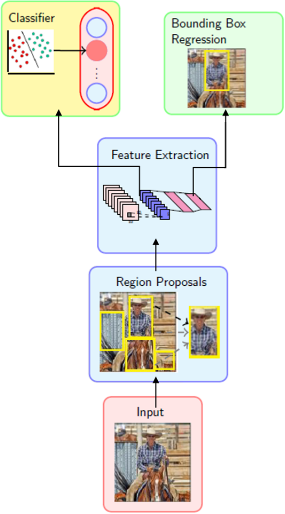
- No joint learning
- Use ad hoc training objectives
  - Fine tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)

[1]Source: *Ross Girshick*
[1]Using VGG-Net



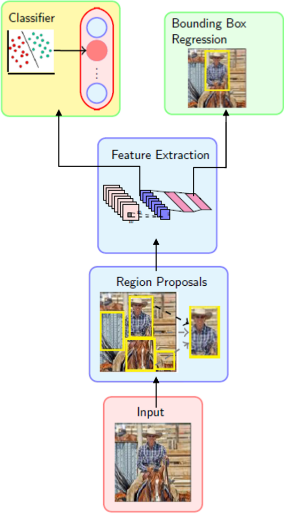
- No joint learning
- Use ad hoc training objectives
  - Fine tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressors (squared loss)

[1]Source: *Ross Girshick*
[1]Using VGG-Net



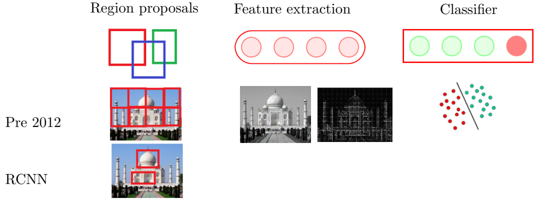
- No joint learning
- Use ad hoc training objectives
  - Fine tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressors (squared loss)
- Training ($\approx$ 3 days) and testing (47s per image) is slow[1].

[1]Source: *Ross Girshick*
[1]Using VGG-Net

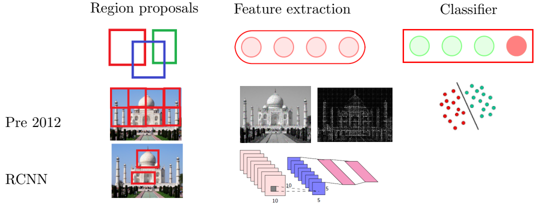


- No joint learning
- Use ad hoc training objectives
  - Fine tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressors (squared loss)
- Training ($\approx$ 3 days) and testing (47s per image) is slow[1].
- Takes a lot of disk space

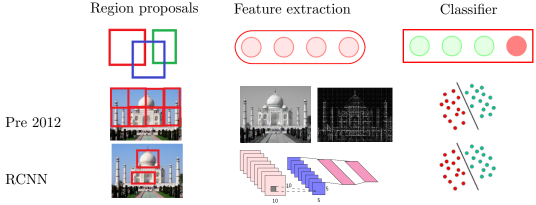
[1]Source: *Ross Girshick*
[1]Using VGG-Net



- **Region Proposals:** Selective Search

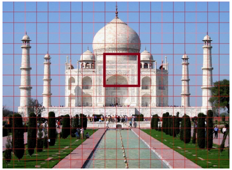


- **Region Proposals:** Selective Search
- **Feature Extraction:** CNNs

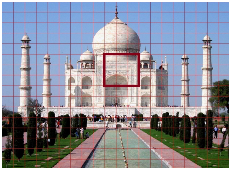


- **Region Proposals:** Selective Search
- **Feature Extraction:** CNNs
- **Classifier:** Linear

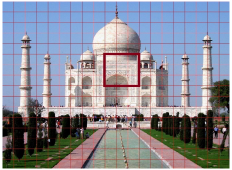
Module 12.3 : Fast RCNN model for object detection

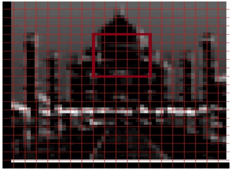


- Suppose we apply a 3×3 kernel on an image

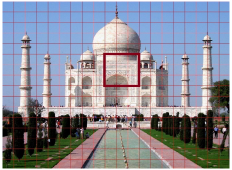


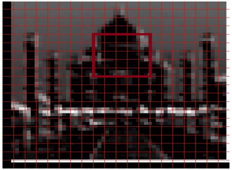
- Suppose we apply a 3×3 kernel on an image
- What is the region of influence of each pixel in the resulting output ?



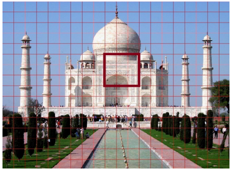


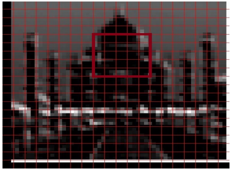
- Suppose we apply a 3×3 kernel on an image
- What is the region of influence of each pixel in the resulting output ?
- Each pixel contributes to a 5×5 region



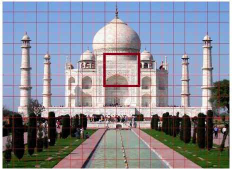


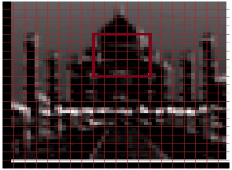
- Suppose we apply a 3×3 kernel on an image
- What is the region of influence of each pixel in the resulting output ?
- Each pixel contributes to a 5×5 region
- Suppose we again apply a 3×3 kernel on this output?

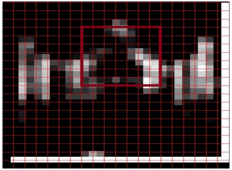




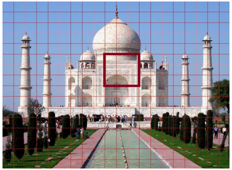
- Suppose we apply a 3×3 kernel on an image
- What is the region of influence of each pixel in the resulting output ?
- Each pixel contributes to a 5×5 region
- Suppose we again apply a 3×3 kernel on this output?
- What is the region of influence of the original pixel from the input ?

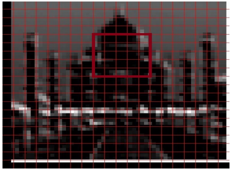


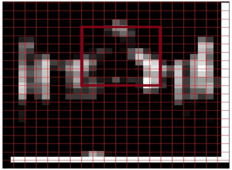




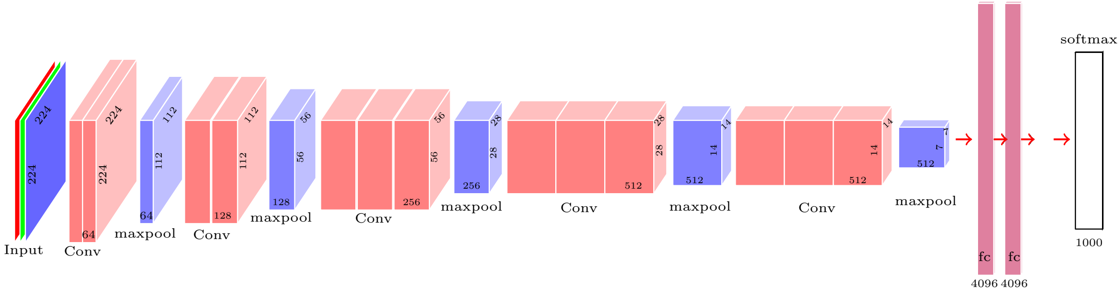
- Suppose we apply a 3×3 kernel on an image
- What is the region of influence of each pixel in the resulting output ?
- Each pixel contributes to a 5×5 region
- Suppose we again apply a 3×3 kernel on this output?
- What is the region of influence of the original pixel from the input ?

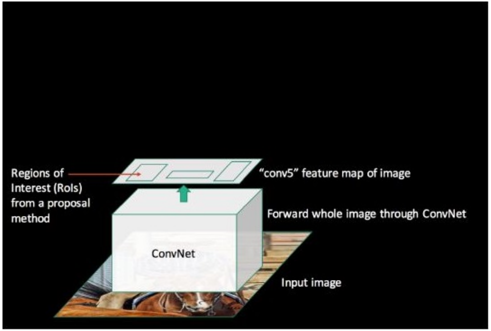






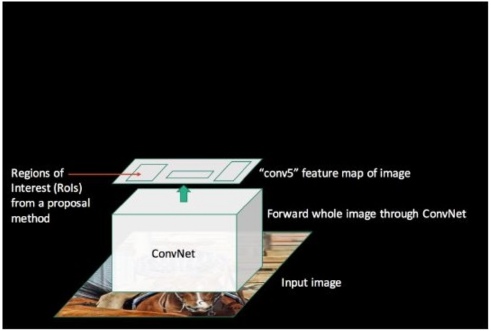
- Suppose we apply a 3×3 kernel on an image
- What is the region of influence of each pixel in the resulting output ?
- Each pixel contributes to a 5×5 region
- Suppose we again apply a 3×3 kernel on this output?
- What is the region of influence of the original pixel from the input ? (a 7×7 region)


Input
224
224
Conv
64
224
224
maxpool
64
112
112
Conv
128
112
112
maxpool
128
56
56
Conv
256
56
56
maxpool
256
28
28
Conv
512
28
28
maxpool
512
14
14
Conv
512
14
14
maxpool
512
7
7
fc
4096
fc
4096
softmax
1000



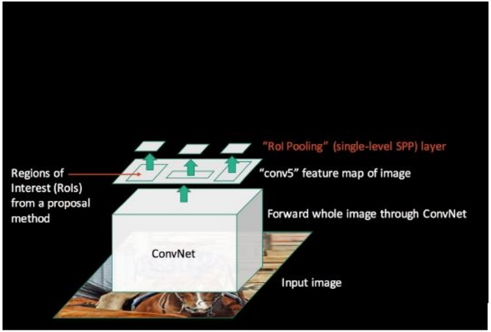
Source: *Ross Girshick*

- Using this idea we could get a bounding box's region of influence on any layer in the CNN



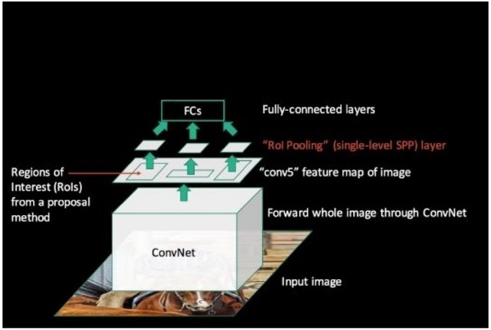
Source: *Ross Girshick*

- Using this idea we could get a bounding box's region of influence on any layer in the CNN
- The projected Region of Interest (RoI) may be of different sizes



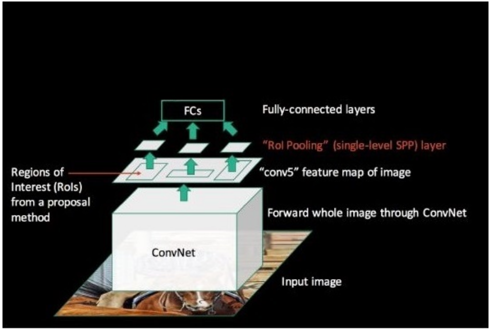
Source: *Ross Girshick*

- Using this idea we could get a bounding box's region of influence on any layer in the CNN
- The projected Region of Interest (RoI) may be of different sizes
- Divide them into k equally sized regions of dimension $H \times W$ and do max pooling in each of those regions to construct a k dimensional vector



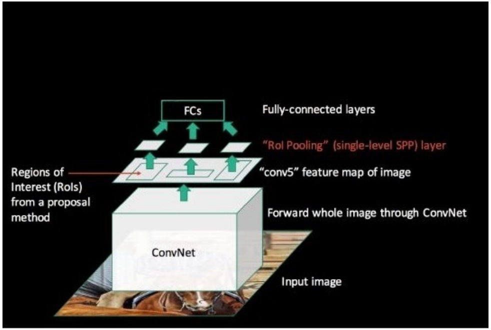
Source: *Ross Girshick*

- Using this idea we could get a bounding box's region of influence on any layer in the CNN
- The projected Region of Interest (RoI) may be of different sizes
- Divide them into k equally sized regions of dimension $H \times W$ and do max pooling in each of those regions to construct a k dimensional vector
- Connect the k dimensional vector to a fully connected layer



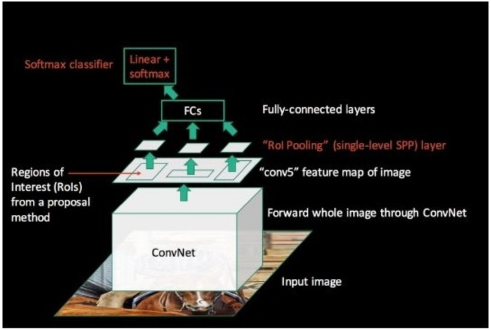
Source: *Ross Girshick*

- Using this idea we could get a bounding box's region of influence on any layer in the CNN
- The projected Region of Interest (RoI) may be of different sizes
- Divide them into k equally sized regions of dimension $H \times W$ and do max pooling in each of those regions to construct a k dimensional vector
- Connect the k dimensional vector to a fully connected layer
- This max pooling operation is call RoI pooling



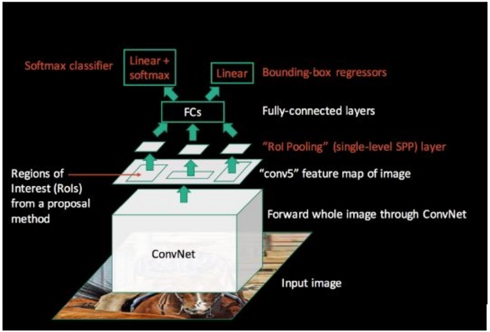
Source: *Ross Girshick*

- Once we have the FC layer it gives us the representation of this region proposal



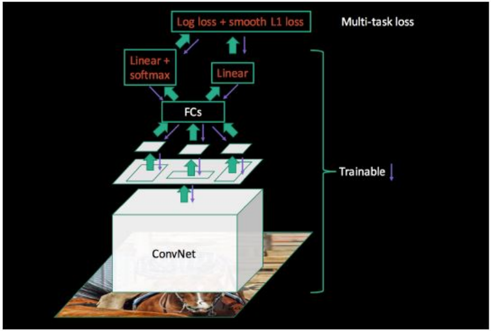
Source: *Ross Girshick*

- Once we have the FC layer it gives us the representation of this region proposal
- We can then add a softmax layer on top of it to compute a probability distribution over the possible object classes



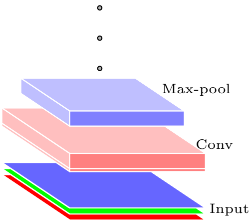
Source: *Ross Girshick*

- Once we have the FC layer it gives us the representation of this region proposal
- We can then add a softmax layer on top of it to compute a probability distribution over the possible object classes
- Similarly we can add a regression layer on top of it to predict the new bounding box (w^*, h^*, x^*, y^*)

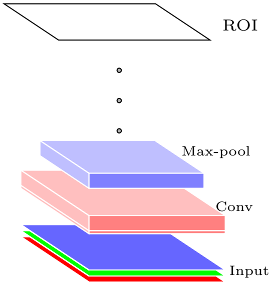


Source: *Ross Girshick*

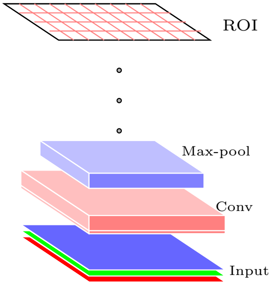
- Once we have the FC layer it gives us the representation of this region proposal
- We can then add a softmax layer on top of it to compute a probability distribution over the possible object classes
- Similarly we can add a regression layer on top of it to predict the new bounding box (w^*, h^*, x^*, y^*)



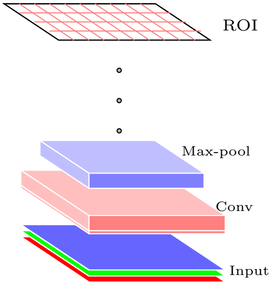
- Recall that the last pooling layer of VGGNet-16 results in an output of size $512 \times 7 \times 7$



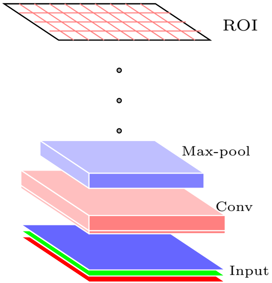
- Recall that the last pooling layer of VGGNet-16 results in an output of size $512 \times 7 \times 7$
- We replace the last max pooling layer by a RoI pooling layer



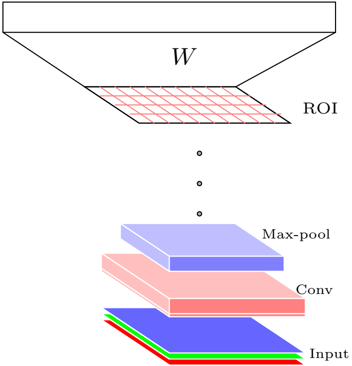
- Recall that the last pooling layer of VGGNet-16 results in an output of size $512 \times 7 \times 7$
- We replace the last max pooling layer by a RoI pooling layer
- We set $H = W = 7$ and divide each of these RoIs into $(k = 49)$ regions



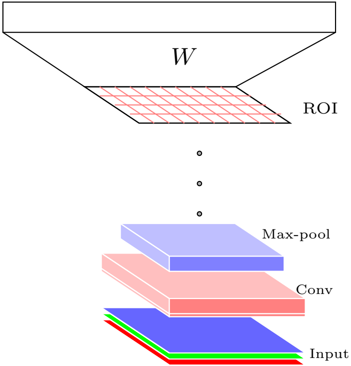
- Recall that the last pooling layer of VGGNet-16 results in an output of size $512 \times 7 \times 7$
- We replace the last max pooling layer by a RoI pooling layer
- We set $H = W = 7$ and divide each of these RoIs into $(k = 49)$ regions
- We do this for every feature map resulting in an ouput of size 512×49



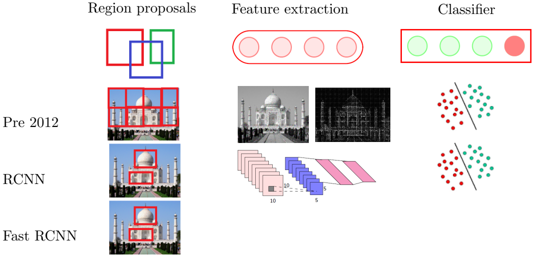
- Recall that the last pooling layer of VGGNet-16 results in an output of size $512 \times 7 \times 7$
- We replace the last max pooling layer by a RoI pooling layer
- We set $H = W = 7$ and divide each of these RoIs into $(k = 49)$ regions
- We do this for every feature map resulting in an ouput of size 512×49
- This output is of the same size as the output of the original max pooling layer



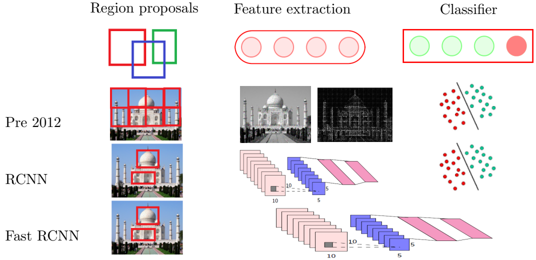
- It is thus compatible with the dimensions of the weight matrix connecting the original pooling layer to the first FC layer



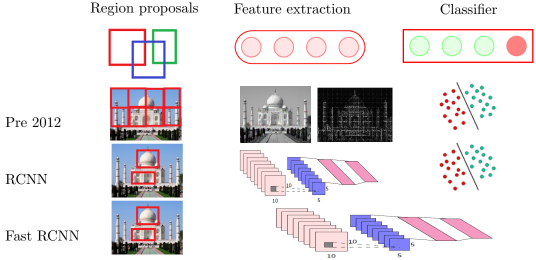
- It is thus compatible with the dimensions of the weight matrix connecting the original pooling layer to the first FC layer
- We can just retain that weight matrix and fine tune it



- **Region Proposals:** Selective Search



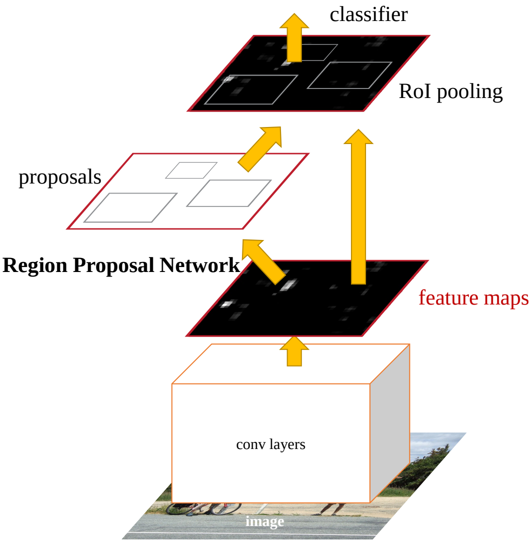
- **Region Proposals:** Selective Search
- **Feature Extraction:** CNN



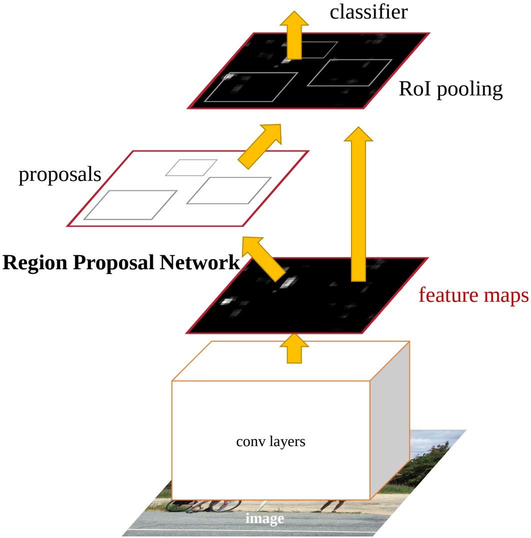
- **Region Proposals:** Selective Search
- **Feature Extraction:** CNN
- **Classifier:** CNN

Module 12.4 : Faster RCNN model for object detection

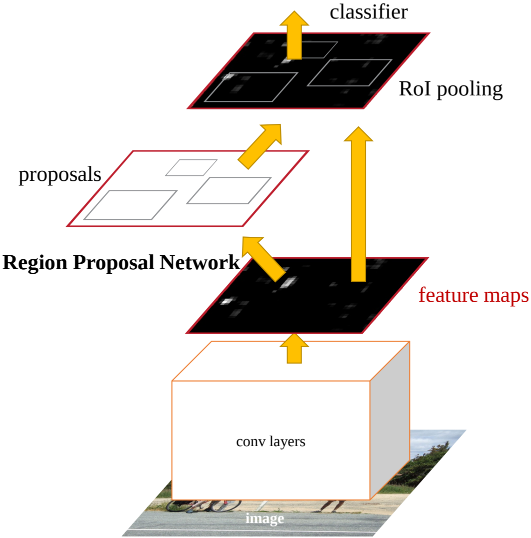
- So far the region proposals were being made using Selective Search algorithm



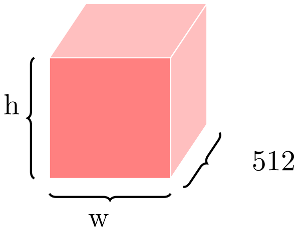
- So far the region proposals were being made using Selective Search algorithm
- **Idea:** Can we use a CNN for making region proposals also?



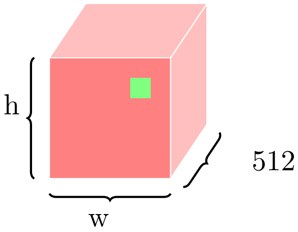
- So far the region proposals were being made using Selective Search algorithm
- **Idea:** Can we use a CNN for making region proposals also?
- How? Well it's slightly tricky



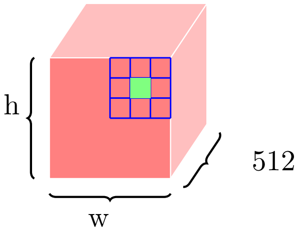
- So far the region proposals were being made using Selective Search algorithm
- **Idea:** Can we use a CNN for making region proposals also?
- How? Well it's slightly tricky
- We will illustrate this using **VGGNet**



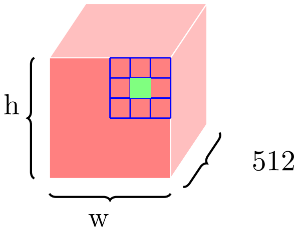
- Consider the output of the last convolutional layer of VGGNet



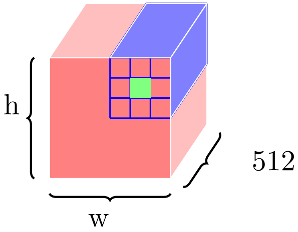
- Consider the output of the last convolutional layer of VGGNet
- Now consider one cell in one of the 512 feature maps

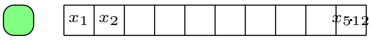


- Consider the output of the last convolutional layer of VGGNet
- Now consider one cell in one of the 512 feature maps
- If we apply a 3×3 kernel around this cell then we will get a 1D representation for this cell

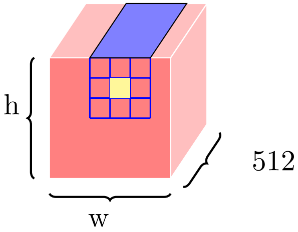


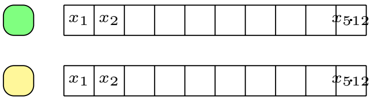
- Consider the output of the last convolutional layer of VGGNet
- Now consider one cell in one of the 512 feature maps
- If we apply a 3×3 kernel around this cell then we will get a 1D representation for this cell



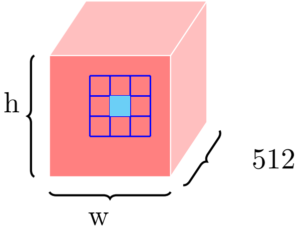


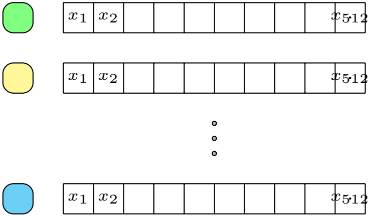
- Consider the output of the last convolutional layer of VGGNet
- Now consider one cell in one of the 512 feature maps
- If we apply a 3×3 kernel around this cell then we will get a 1D representation for this cell
- If we repeat this for all the 512 feature maps then we will get a 512 dimensional representation for this position



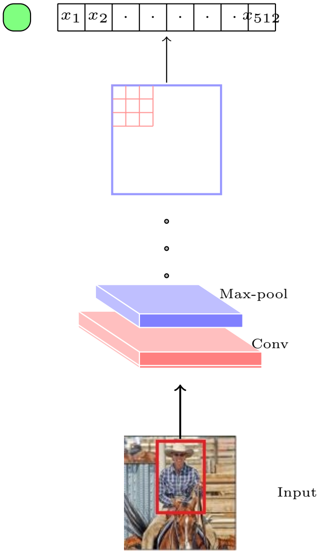


- Consider the output of the last convolutional layer of VGGNet
- Now consider one cell in one of the 512 feature maps
- If we apply a 3×3 kernel around this cell then we will get a 1D representation for this cell
- If we repeat this for all the 512 feature maps then we will get a 512 dimensional representation for this position
- We use this process to get a 512 dimensional representation for each of the $w \times h$ positions

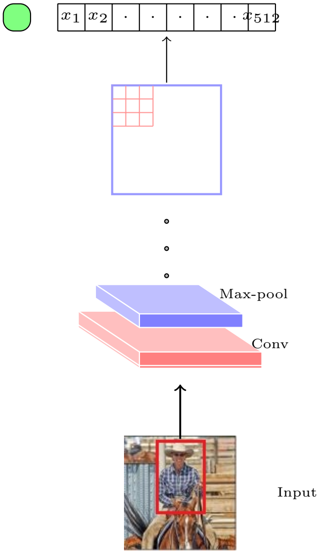


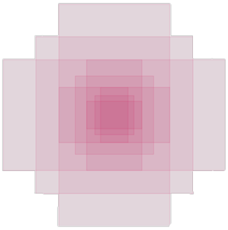


- Consider the output of the last convolutional layer of VGGNet
- Now consider one cell in one of the 512 feature maps
- If we apply a 3×3 kernel around this cell then we will get a 1D representation for this cell
- If we repeat this for all the 512 feature maps then we will get a 512 dimensional representation for this position
- We use this process to get a 512 dimensional representation for each of the $w \times h$ positions

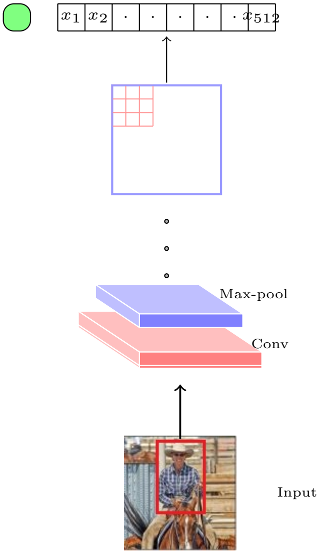


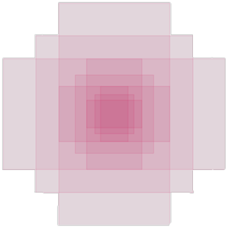
x_1 x_2 · · · · · x_{512}
Max-pool
Conv
Input



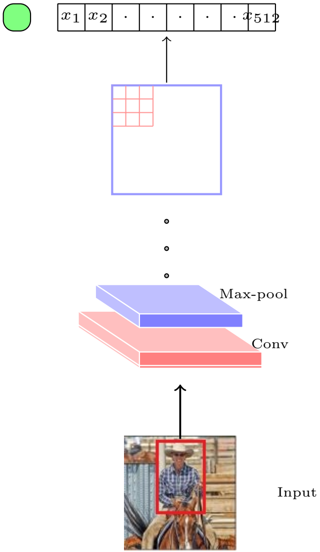


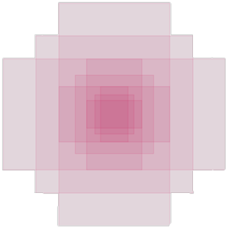
- We now consider k bounding boxes (called anchor boxes) of different sizes & aspect ratio



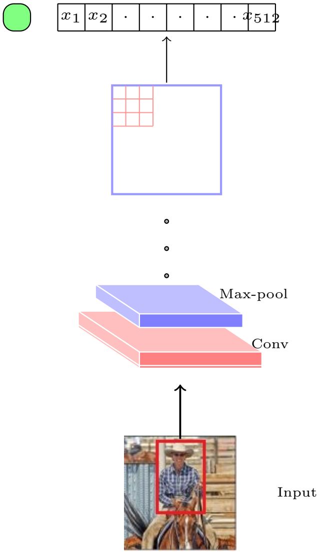


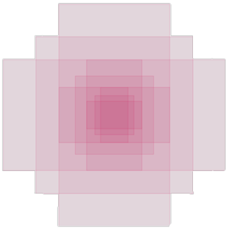
- We now consider k bounding boxes (called anchor boxes) of different sizes & aspect ratio
- We are interested in the following two questions:



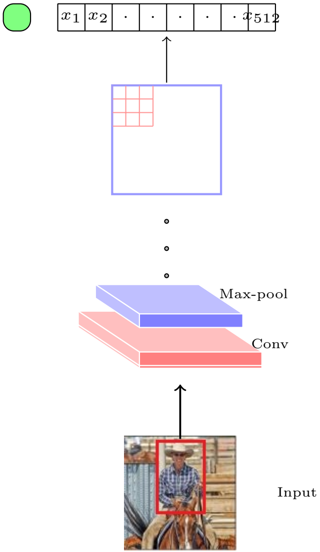


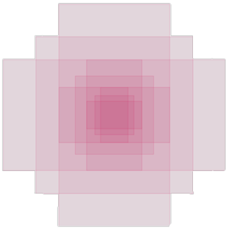
- We now consider k bounding boxes (called anchor boxes) of different sizes & aspect ratio
- We are interested in the following two questions:
- Given the $512d$ representation of a position, what is the probability that a given anchor box centered at this position contains an object?



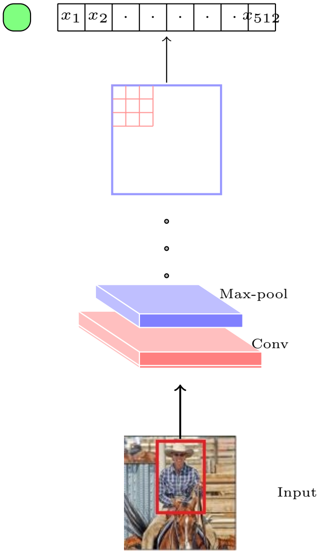


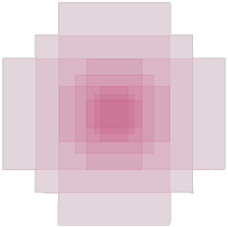
- We now consider k bounding boxes (called anchor boxes) of different sizes & aspect ratio
- We are interested in the following two questions:
- Given the $512d$ representation of a position, what is the probability that a given anchor box centered at this position contains an object? (Classification)



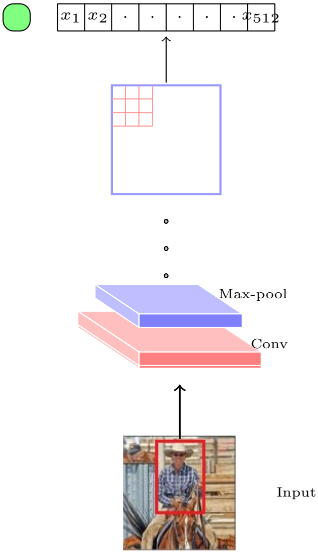


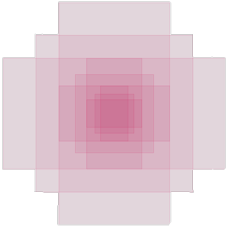
- We now consider k bounding boxes (called anchor boxes) of different sizes & aspect ratio
- We are interested in the following two questions:
- Given the $512d$ representation of a position, what is the probability that a given anchor box centered at this position contains an object? (Classification)
- How do you predict the true bounding box from this anchor box?



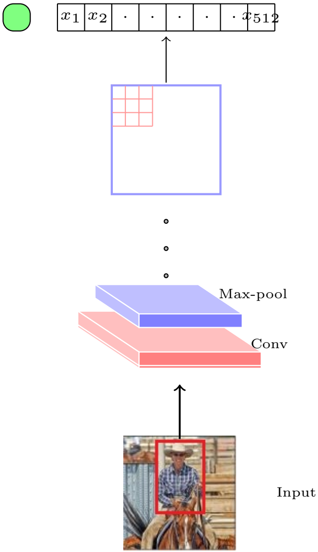


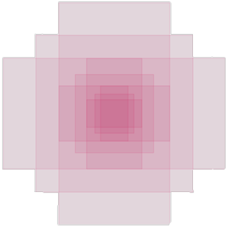
- We now consider k bounding boxes (called anchor boxes) of different sizes & aspect ratio
- We are interested in the following two questions:
- Given the $512d$ representation of a position, what is the probability that a given anchor box centered at this position contains an object? (Classification)
- How do you predict the true bounding box from this anchor box? (Regression)



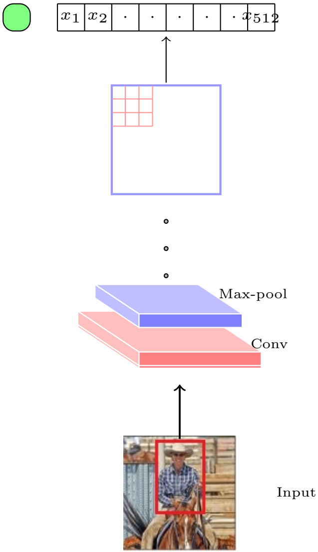


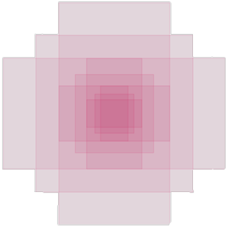
- We train a classification model and a regression model to address these two questions



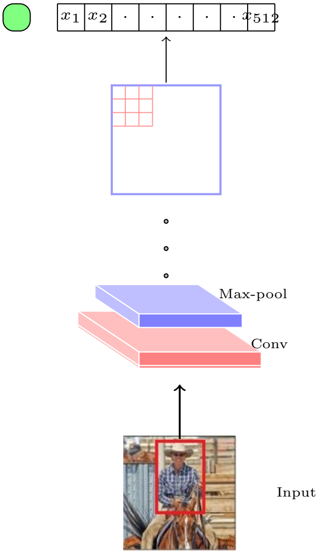


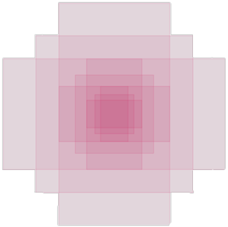
- We train a classification model and a regression model to address these two questions
- How do we get the ground truth data?





- We train a classification model and a regression model to address these two questions
- How do we get the ground truth data?
- What is the objective function used for training?

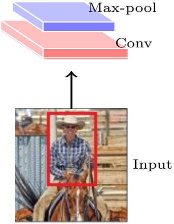




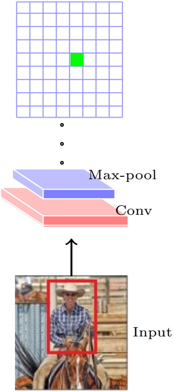
- We train a classification model and a regression model to address these two questions
- How do we get the ground truth data?
- What is the objective function used for training?

- Consider a ground truth object and its corresponding bounding box

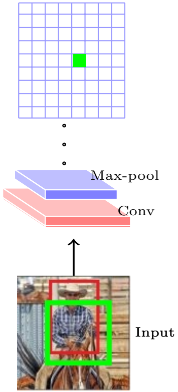
Input



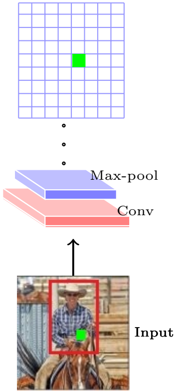
- Consider a ground truth object and its corresponding bounding box
- Consider the projection of this image onto the conv5 layer



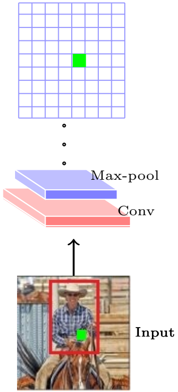
- Consider a ground truth object and its corresponding bounding box
- Consider the projection of this image onto the conv5 layer
- Consider one such cell in the output

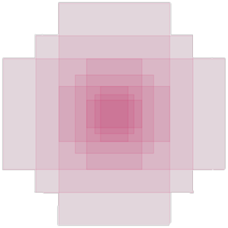


- Consider a ground truth object and its corresponding bounding box
- Consider the projection of this image onto the conv5 layer
- Consider one such cell in the output
- This cell corresponds to a patch in the original image

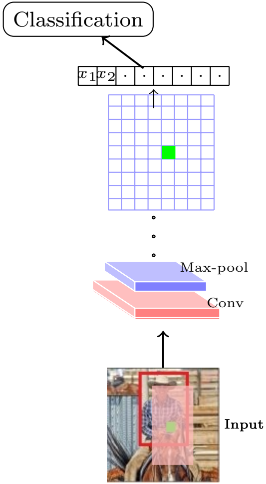


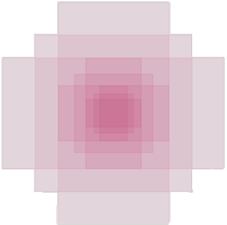
- Consider a ground truth object and its corresponding bounding box
- Consider the projection of this image onto the conv5 layer
- Consider one such cell in the output
- This cell corresponds to a patch in the original image
- Consider the center of this patch



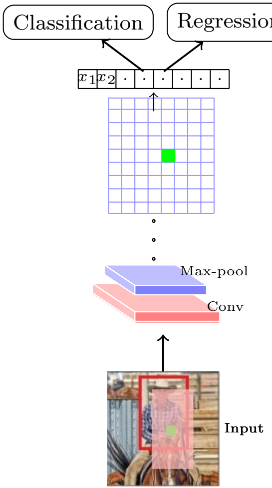


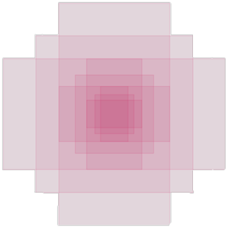
- Consider a ground truth object and its corresponding bounding box
- Consider the projection of this image onto the conv5 layer
- Consider one such cell in the output
- This cell corresponds to a patch in the original image
- Consider the center of this patch
- We consider anchor boxes of different sizes



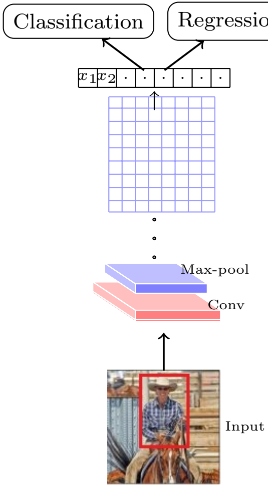


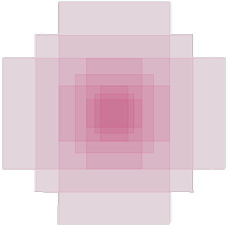
- For each of these anchor boxes, we would want the classifier to predict 1 if this anchor box has a reasonable overlap (IoU > 0.7) with the true grounding box





- For each of these anchor boxes, we would want the classifier to predict 1 if this anchor box has a reasonable overlap (IoU > 0.7) with the true grounding box
- Similarly we would want the regression model to predict the true box (red) from the anchor box (pink)





- We train a classification model and a regression model to address these two questions
- How do we get the ground truth data?
- What is the objective function used for training?

- The full network is trained using the following objective.

- The full network is trained using the following objective.

$$\mathscr{L}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i \mathscr{L}_{cls}(p_i, p_i^*)$$

- The full network is trained using the following objective.

$$\mathscr{L}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i \mathscr{L}_{cls}(p_i, p_i^*)$$

$$\begin{aligned} p_i^* &= 1 \quad \text{if anchor box contains ground truth object} \\ &= 0 \quad \text{otherwise} \\ p_i &= \text{predicted probability of anchor box containing an object} \\ N_{cls} &= \text{batch-size} \end{aligned}$$

- The full network is trained using the following objective.

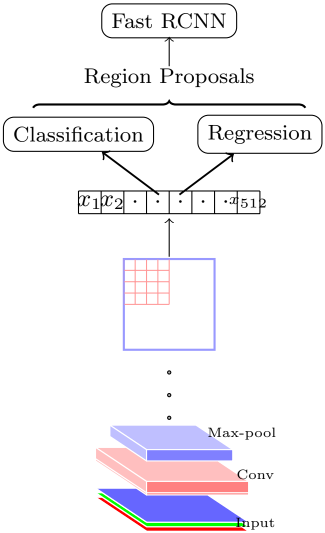
$$\mathscr{L}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i \mathscr{L}_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{reg}} \sum_i p_i^* \mathscr{L}_{reg}(t_i, t_i^*)$$

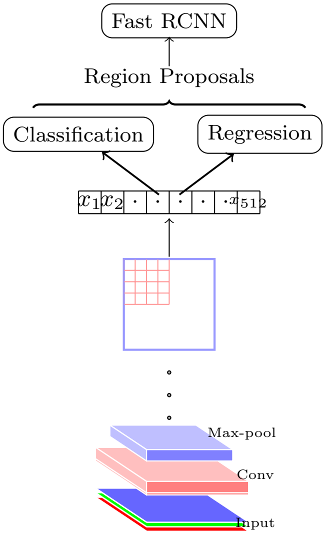
$$\begin{aligned} p_i^* &= 1 \quad \text{if anchor box contains ground truth object} \\ &= 0 \quad \text{otherwise} \\ p_i &= \text{predicted probability of anchor box containing an object} \\ N_{cls} &= \text{batch-size} \end{aligned}$$

- The full network is trained using the following objective.

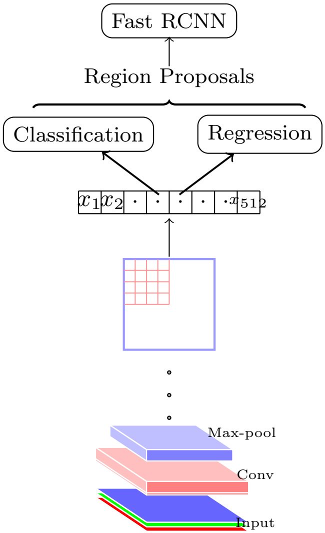
$$\mathscr{L}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i \mathscr{L}_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{reg}} \sum_i p_i^* \mathscr{L}_{reg}(t_i, t_i^*)$$

$$\begin{aligned}
p_i^* &= 1 \quad \text{if anchor box contains ground truth object} \\
&= 0 \quad \text{otherwise} \\
p_i &= \text{predicted probability of anchor box containing an object} \\
N_{cls} &= \text{batch-size} \\
N_{reg} &= \text{batch-size} \times k \\
k &= \text{anchor boxes}
\end{aligned}$$

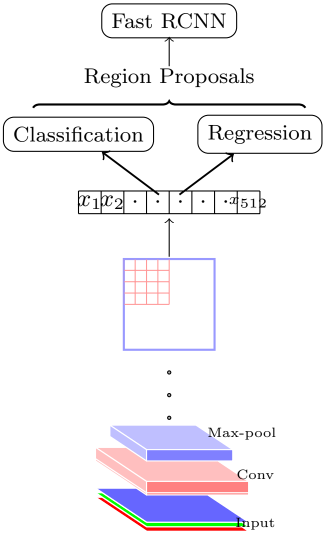

Fast RCNN
Region Proposals
Classification
Regression
Max-pool
Conv
Input



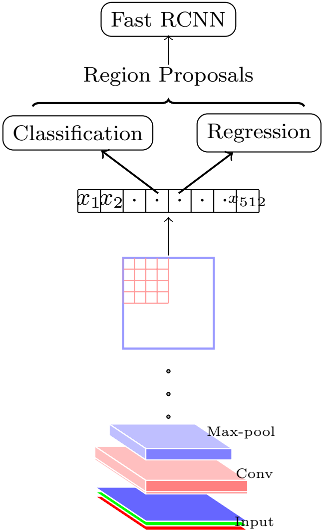
- So far we have seen a CNN based approach for region proposals instead of using selective search



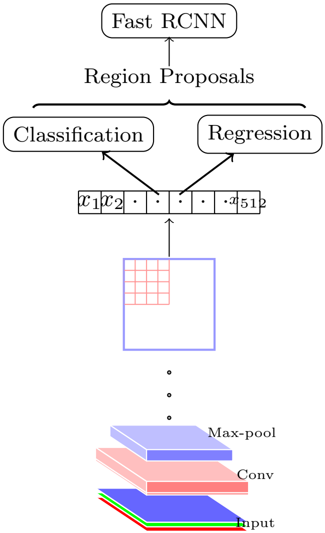
- So far we have seen a CNN based approach for region proposals instead of using selective search
- We can now take these region proposals and then add fast RCNN on top of it to predict the class of the object



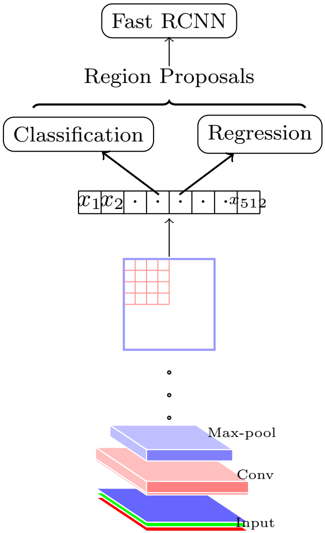
- So far we have seen a CNN based approach for region proposals instead of using selective search
- We can now take these region proposals and then add fast RCNN on top of it to predict the class of the object
- And regress the proposed bounding box



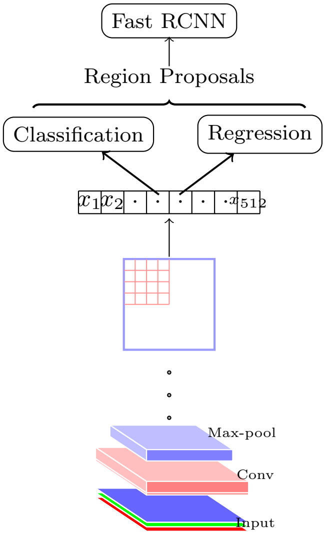
Fast RCNN
Region Proposals
Classification
Regression
x_1 x_2 · · · · · x_{512}
Max-pool
Conv
Input



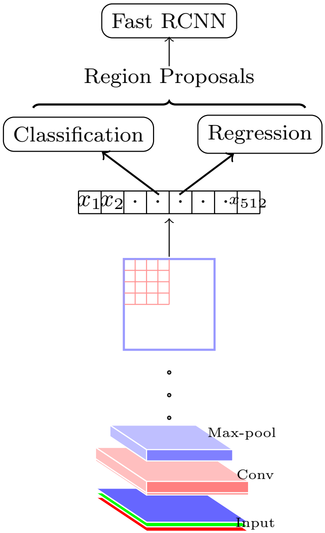
- But the fast RCNN would again use a VGG Net



- But the fast RCNN would again use a VGG Net
- Can't we use a single VGG Net and share the parameters of RPN and RCNN

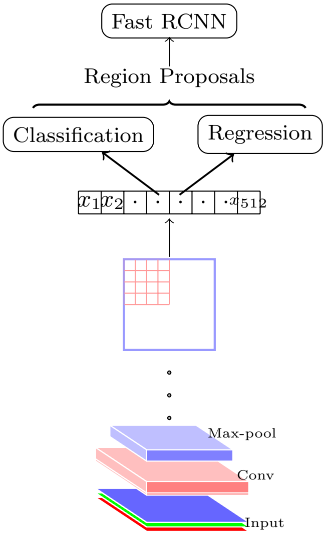


- But the fast RCNN would again use a VGG Net
- Can't we use a single VGG Net and share the parameters of RPN and RCNN
- Yes, we can

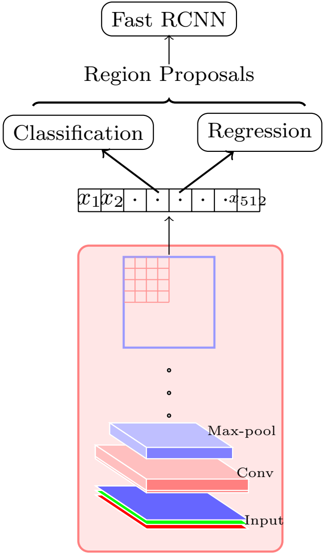


- But the fast RCNN would again use a VGG Net
- Can't we use a single VGG Net and share the parameters of RPN and RCNN
- Yes, we can
- In practice, we use a 4 step alternating training process

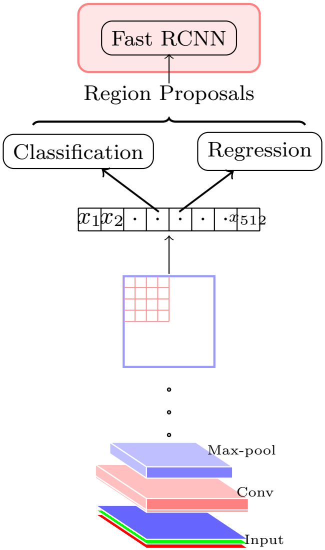
Faster RCNN:Training



Faster RCNN:Training

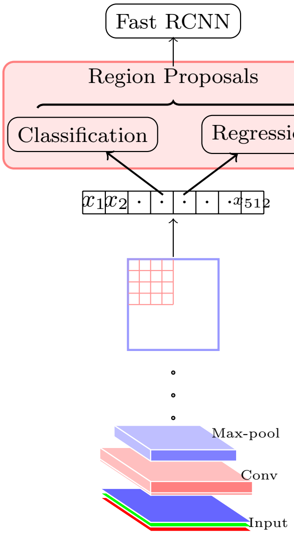


- Fine-tune RPN using a pre-trained ImageNet network



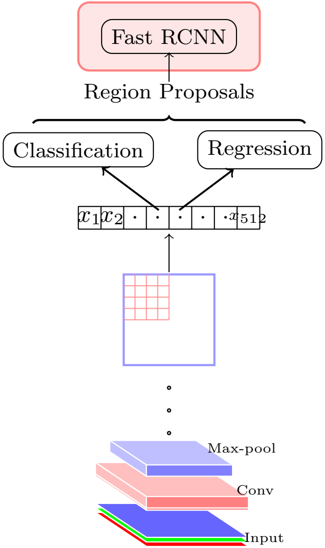
Faster RCNN:Training

- Fine-tune RPN using a pre-trained ImageNet network
- Fine-tune fast RCNN from a pre-trained ImageNet network using bounding boxes from step 1



Faster RCNN:Training

- Fine-tune RPN using a pre-trained ImageNet network
- Fine-tune fast RCNN from a pre-trained ImageNet network using bounding boxes from step 1
- Keeping common convolutional layer parameters fixed from step 2, fine-tune RPN (post conv5 layers)



Faster RCNN:Training

- Fine-tune RPN using a pre-trained ImageNet network
- Fine-tune fast RCNN from a pre-trained ImageNet network using bounding boxes from step 1
- Keeping common convolutional layer parameters fixed from step 2, fine-tune RPN (post conv5 layers)
- Keeping common convolution layer parameters fixed from step 3, fine-tune fc layers of fast RCNN

Faster RCNN and RPN are the basis of several 1st place entries in the ILSVRC and COCO tracks on :

Faster RCNN and RPN are the basis of several 1st place entries in the ILSVRC and COCO tracks on :

- Imagenet detection

Faster RCNN and RPN are the basis of several 1st place entries in the ILSVRC and COCO tracks on :

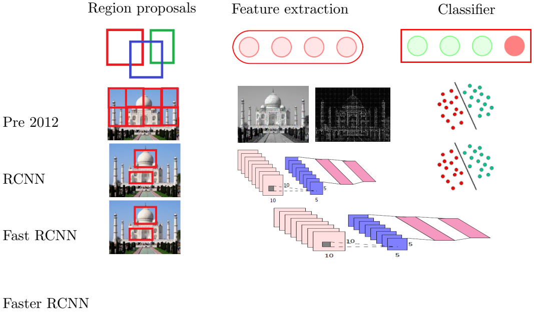
- Imagenet detection
- COCO Segmentation

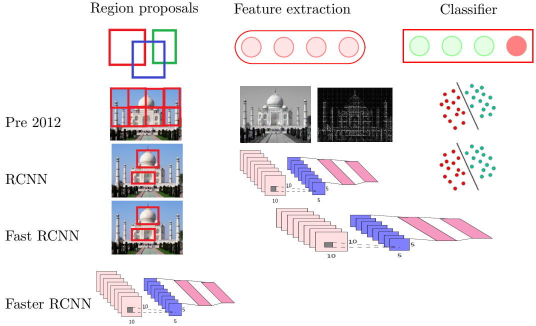
Faster RCNN and RPN are the basis of several 1st place entries in the ILSVRC and COCO tracks on :

- Imagenet detection
- COCO Segmentation
- Imagenet localization

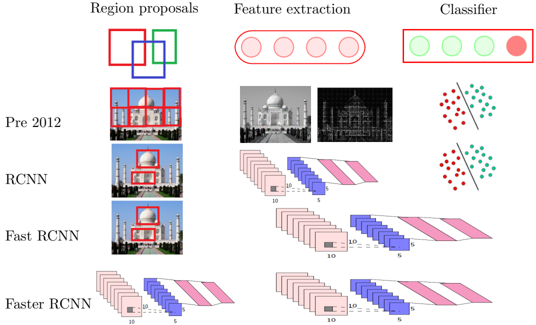
Faster RCNN and RPN are the basis of several 1st place entries in the ILSVRC and COCO tracks on :

- Imagenet detection
- COCO Segmentation
- Imagenet localization
- COCO detection

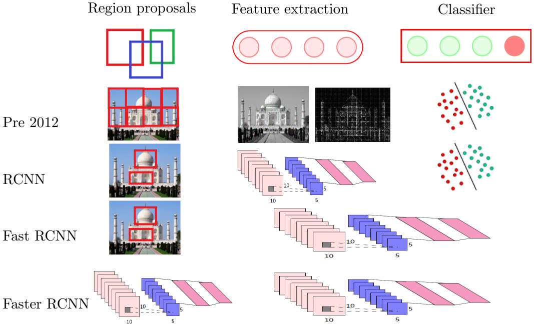

Region proposals
Feature extraction
Classifier
Pre 2012
RCNN
Fast RCNN
Faster RCNN
10
10
5
5



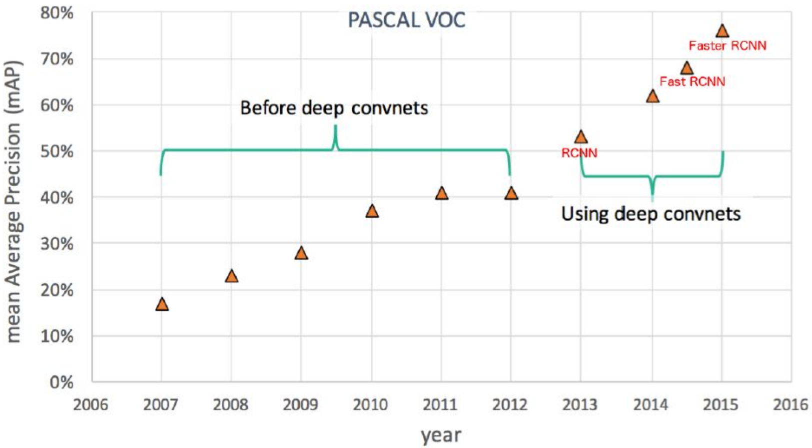
- **Region Proposals:** CNN



- **Region Proposals:** CNN
- **Feature Extraction:** CNN



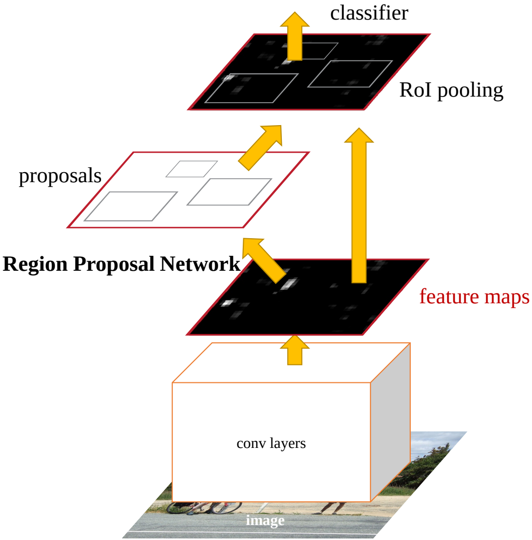
- **Region Proposals:** CNN
- **Feature Extraction:** CNN
- **Classifier:** CNN



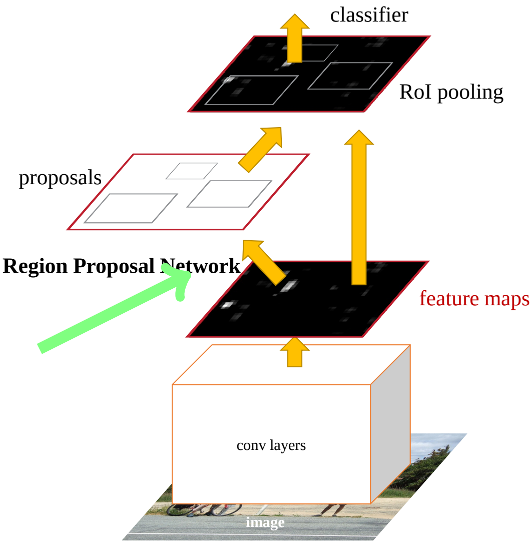
Object Detection Performance

Source: Ross Girshick

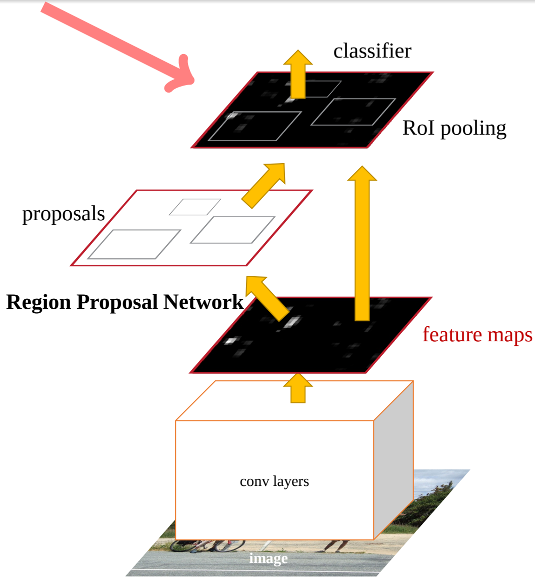
Module 12.5 : YOLO model for object detection



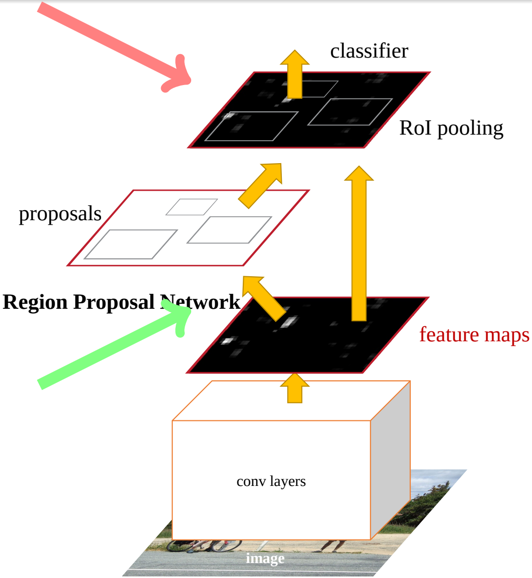
- The approaches that we have seen so far are two stage approaches



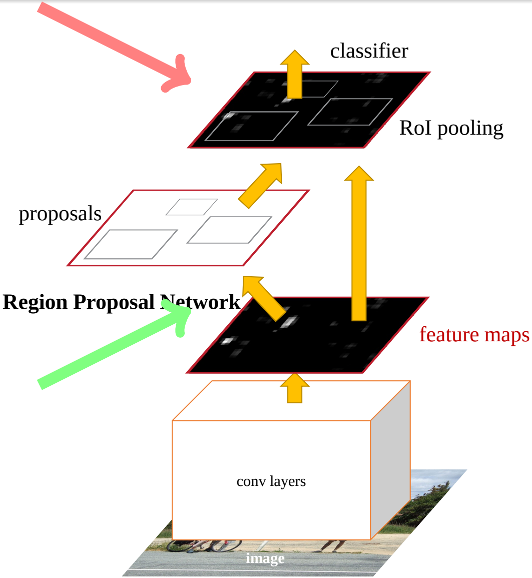
- The approaches that we have seen so far are two stage approaches
- They involve a region proposal stage and then a classification stage



- The approaches that we have seen so far are two stage approaches
- They involve a region proposal stage and then a classification stage



- The approaches that we have seen so far are two stage approaches
- They involve a region proposal stage and then a classification stage
- Can we have an end-to-end architecture which does both proposal and classification simultaneously ?

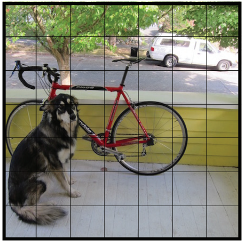


- The approaches that we have seen so far are two stage approaches
- They involve a region proposal stage and then a classification stage
- Can we have an end-to-end architecture which does both proposal and classification simultaneously ?
- This is the idea behind **YOLO**-You Only Look Once.

$P(cow)$ $P(truck)$

c	w	h	x	y			$\cdot$	$\cdot$	

$P(dog)$



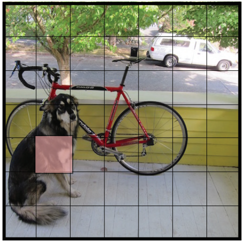
S × S grid on input

- Divide an image into $S \times S$ grids (S=7)

$P(cow)$ $P(truck)$

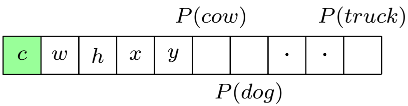
c	w	h	x	y			$\cdot$	$\cdot$	

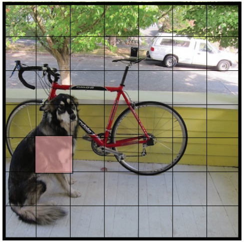
$P(dog)$



S × S grid on input

- Divide an image into $S \times S$ grids (S=7)
- For each such cell we are interested in predicting $5 + k$ quantities





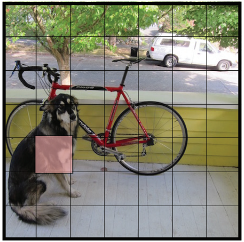
S × S grid on input

- Divide an image into $S \times S$ grids (S=7)
- For each such cell we are interested in predicting $5 + k$ quantities
- Probability (confidence) that this cell is indeed contained in a true bounding box

$P(cow)$ $P(truck)$

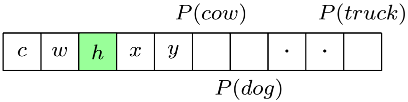
c	w	h	x	y			$\cdot$	$\cdot$	

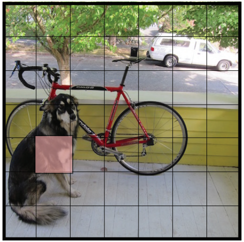
$P(dog)$



S × S grid on input

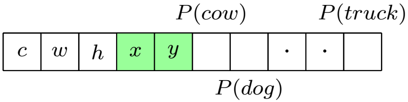
- Divide an image into $S \times S$ grids (S=7)
- For each such cell we are interested in predicting $5 + k$ quantities
- Probability (confidence) that this cell is indeed contained in a true bounding box
- Width of the bounding box

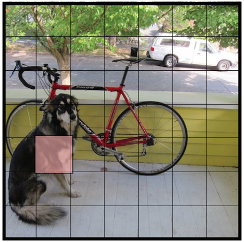




S × S grid on input

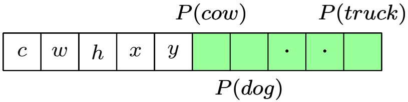
- Divide an image into $S \times S$ grids (S=7)
- For each such cell we are interested in predicting $5 + k$ quantities
- Probability (confidence) that this cell is indeed contained in a true bounding box
- Width of the bounding box
- Height of the bounding box

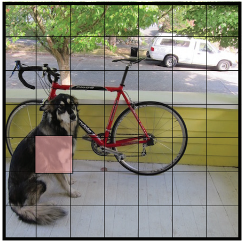




S × S grid on input

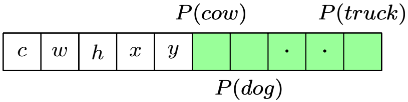
- Divide an image into $S \times S$ grids (S=7)
- For each such cell we are interested in predicting $5 + k$ quantities
- Probability (confidence) that this cell is indeed contained in a true bounding box
- Width of the bounding box
- Height of the bounding box
- Center (x,y) of the bounding box

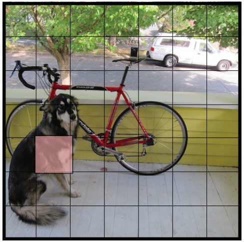




S × S grid on input

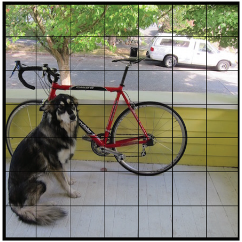
- Divide an image into $S \times S$ grids (S=7)
- For each such cell we are interested in predicting $5 + k$ quantities
- Probability (confidence) that this cell is indeed contained in a true bounding box
- Width of the bounding box
- Height of the bounding box
- Center (x,y) of the bounding box
- Probability of the object in the bounding box belonging to the k^{th} class (k - values)





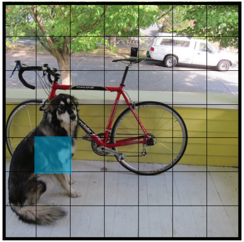
S × S grid on input

- Divide an image into $S \times S$ grids (S=7)
- For each such cell we are interested in predicting $5 + k$ quantities
- Probability (confidence) that this cell is indeed contained in a true bounding box
- Width of the bounding box
- Height of the bounding box
- Center (x,y) of the bounding box
- Probability of the object in the bounding box belonging to the k^{th} class (k - values)
- The output layer thus contains $S \times S \times (5 + k)$ elements



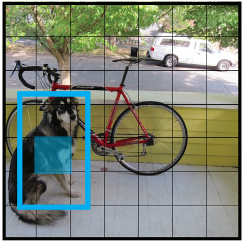
Input Image

- How do we interpret this $S \times S \times (5+k)$ dimensional output?



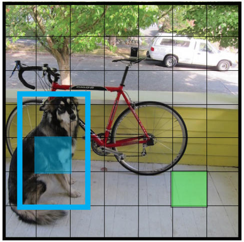
S × S grid on input

- How do we interpret this $S \times S \times (5+k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it



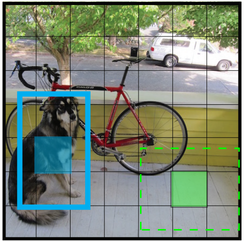
S × S grid on input

- How do we interpret this $S \times S \times (5+k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it



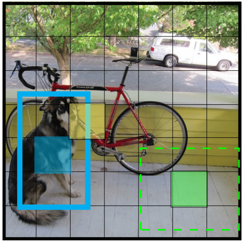
S × S grid on input

- How do we interpret this $S \times S \times (5+k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it

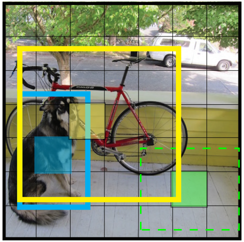


S × S grid on input

- How do we interpret this $S \times S \times (5+k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it

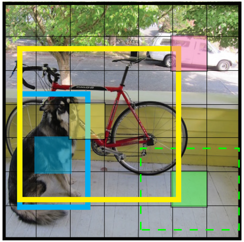

S × S grid on input

- How do we interpret this $S \times S \times (5+k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it



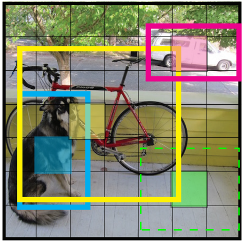
S × S grid on input

- How do we interpret this $S \times S \times (5+k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it



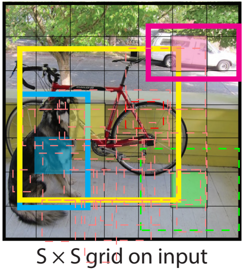
S × S grid on input

- How do we interpret this $S \times S \times (5 + k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it



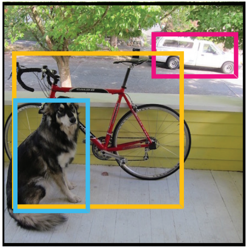
S × S grid on input

- How do we interpret this $S \times S \times (5+k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it



Bounding Boxes & Confidence

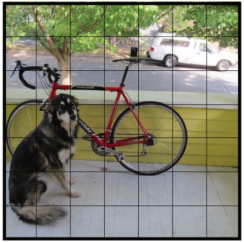
- How do we interpret this $S \times S \times (5+k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it



- How do we interpret this $S \times S \times (5 + k)$ dimensional output?
- For each cell, we are computing a bounding box, its confidence and the object in it
- We then retain the most confident bounding boxes and the corresponding object label

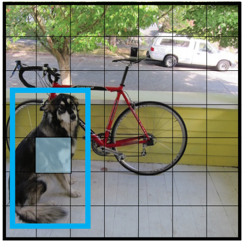
- How do we train this network ?

$\hat{c}$	$\hat{w}$	$\hat{h}$	$\hat{x}$	$\hat{y}$	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$



S × S grid on input

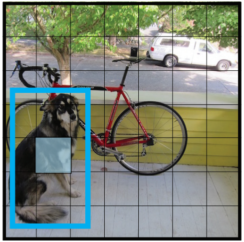
$\hat{c}$	$\hat{w}$	$\hat{h}$	$\hat{x}$	$\hat{y}$	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$



S × S grid on input

- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it

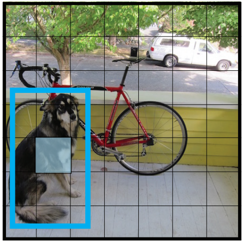
$\hat{c}$	$\hat{w}$	$\hat{h}$	$\hat{x}$	$\hat{y}$	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$



S × S grid on input

- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ

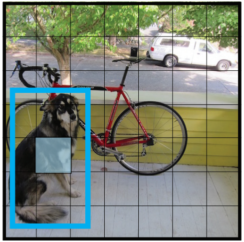
$\hat{c}$	$\hat{w}$	$\hat{h}$	$\hat{x}$	$\hat{y}$	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$



S × S grid on input

- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ
- We can then compute the following losses

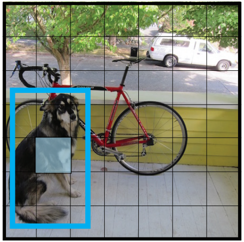
c	$\hat{w}$	$\hat{h}$	$\hat{x}$	$\hat{y}$	$\hat{\ell}_1$	$\hat{\ell}_2$	$\cdot$	$\cdot$	$\hat{\ell}_k$



S × S grid on input

- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ
- We can then compute the following losses
- $(1 - \hat{c})^2$

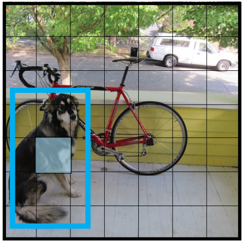
$\hat{c}$	w	$\hat{h}$	$\hat{x}$	$\hat{y}$	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$



S × S grid on input

- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ
- We can then compute the following losses
- $(\sqrt{w} - \sqrt{\hat{w}})^2$

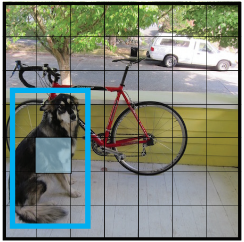
$\hat{c}$	$\hat{w}$	h	$\hat{x}$	$\hat{y}$	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$



S × S grid on input

- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ
- We can then compute the following losses
- $(\sqrt{h} - \sqrt{\hat{h}})^2$

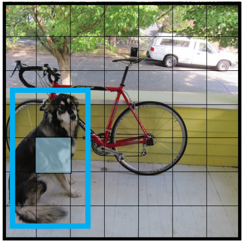
$\hat{c}$	$\hat{w}$	$\hat{h}$	x	y	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$



S × S grid on input

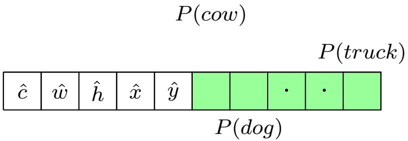
- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ
- We can then compute the following losses
- $(x - \hat{x})^2$

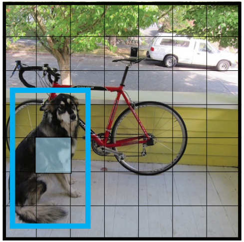
$\hat{c}$	$\hat{w}$	$\hat{h}$	x	y	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$



S × S grid on input

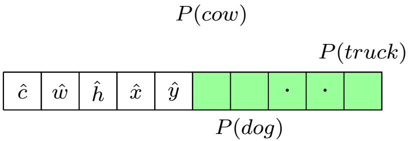
- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ
- We can then compute the following losses
- $(y - \hat{y})^2$

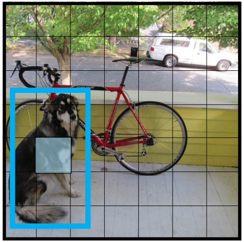




S × S grid on input

- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ
- We can then compute the following losses
- $\sum_{i=1}^{k}(\ell_i - \hat{\ell}_i)^2$



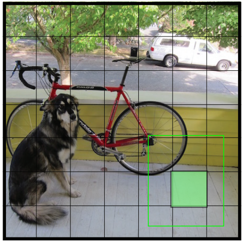


S × S grid on input

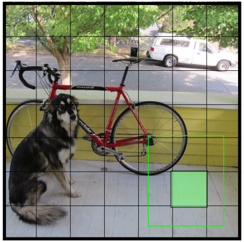
- How do we train this network ?
- Consider a cell such that the center of the true bonding box lies in it
- The network is initialized randomly and it will predict some values for c, w, h, x, y & ℓ
- We can then compute the following losses
- $\sum_{i=1}^{k} (\ell_i - \hat{\ell}_i)^2$
- And train the network to minimize the sum of these losses

$\hat{c}$	$\hat{w}$	$\hat{h}$	$\hat{x}$	$\hat{y}$	$\hat{\ell_1}$	$\hat{\ell_2}$	$\cdot$	$\cdot$	$\hat{\ell_k}$

- Now consider a grid which does not contain any object

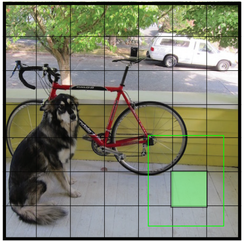


S × S grid on input



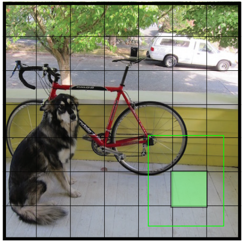
S × S grid on input

- Now consider a grid which does not contain any object
- For this grid we do not care about the predictions w, h, x, y & ℓ



S × S grid on input

- Now consider a grid which does not contain any object
- For this grid we do not care about the predictions w, h, x, y & ℓ
- But we want the confidence to be low



S × S grid on input

- Now consider a grid which does not contain any object
- For this grid we do not care about the predictions w, h, x, y & ℓ
- But we want the confidence to be low
- So we minimize only the following loss

$$(0 - \hat{c})^2$$

Method	Pascal 2007 mAP	Speed
DPM v5	33.7	0.07 FPS — 14 sec/ image

Method	**Pascal 2007 mAP**	**Speed**
DPM v5	33.7	0.07 FPS — 14 sec/ image
RCNN	66.0	0.05 FPS — 20 sec/ image

Method	**Pascal 2007 mAP**	**Speed**
DPM v5	33.7	0.07 FPS — 14 sec/ image
RCNN	66.0	0.05 FPS — 20 sec/ image
Fast RCNN	70.0	0.5 FPS — 2 sec/ image

Method	**Pascal 2007 mAP**	**Speed**
DPM v5	33.7	0.07 FPS — 14 sec/ image
RCNN	66.0	0.05 FPS — 20 sec/ image
Fast RCNN	70.0	0.5 FPS — 2 sec/ image
Faster RCNN	73.2	7 FPS — 140 msec/ image

Method	**Pascal 2007 mAP**	**Speed**
DPM v5	33.7	0.07 FPS — 14 sec/ image
RCNN	66.0	0.05 FPS — 20 sec/ image
Fast RCNN	70.0	0.5 FPS — 2 sec/ image
Faster RCNN	73.2	7 FPS — 140 msec/ image
YOLO	69.0	45 FPS — 22 msec/ image