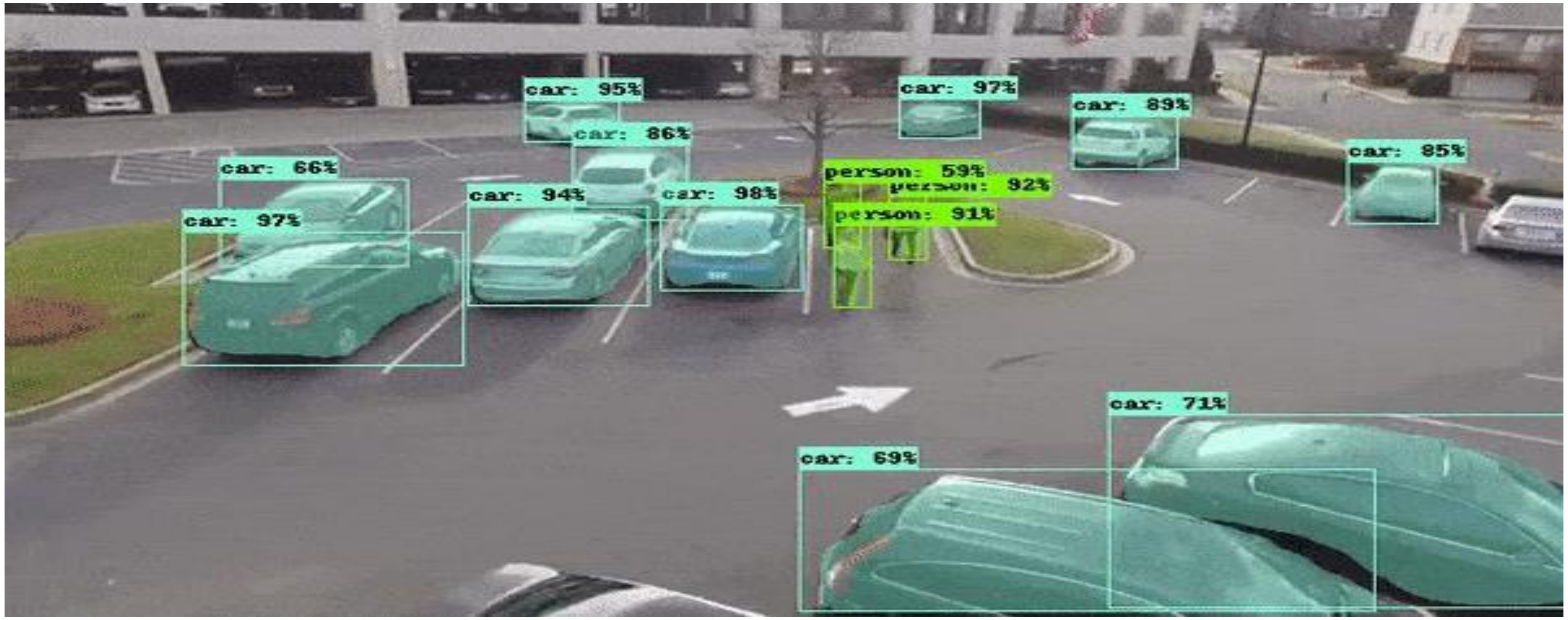
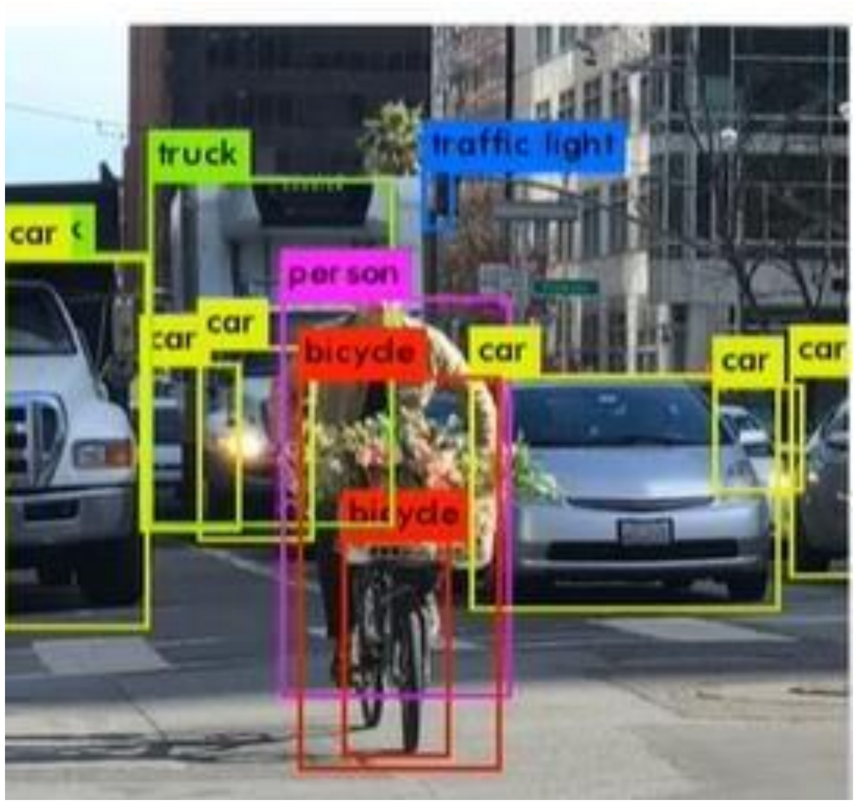
Region-Based CNNs



- CNN models are only able to tell whether an image contains an object or no.
- Suppose we want to work on models that could also tell where that object is, in an image?
- **Object Detection:** Predicting bounding boxes of multiple objects of multiple classes. May contain many objects belonging to the same class as well.



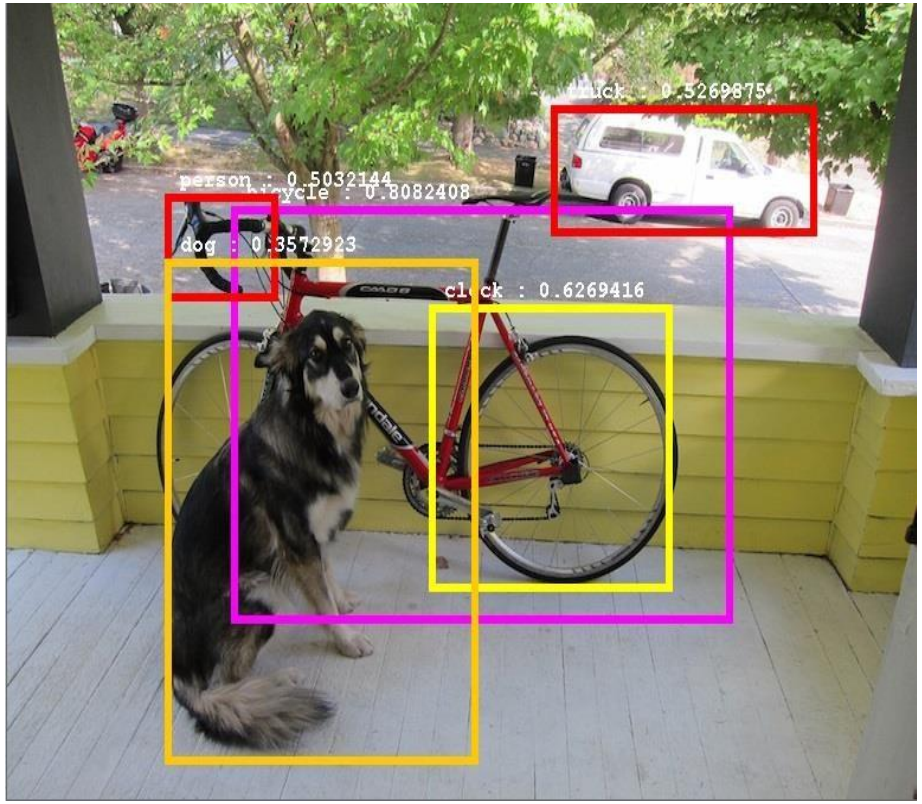


Image Localization

- Predicting bounding box of only a single object, of a single class, in an image
- In classification algorithms, the final layer gives a probability value ranging from 0 to 1.
- In contrast, localization algorithms give an output in four real numbers, as localization is a regression problem.

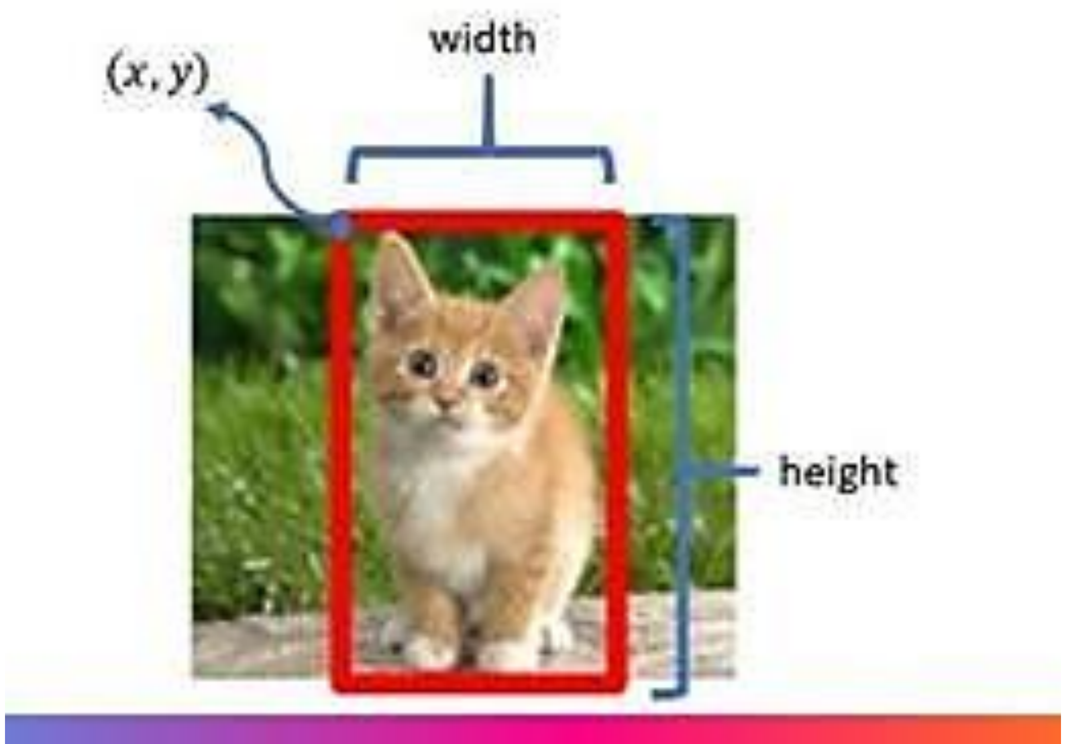


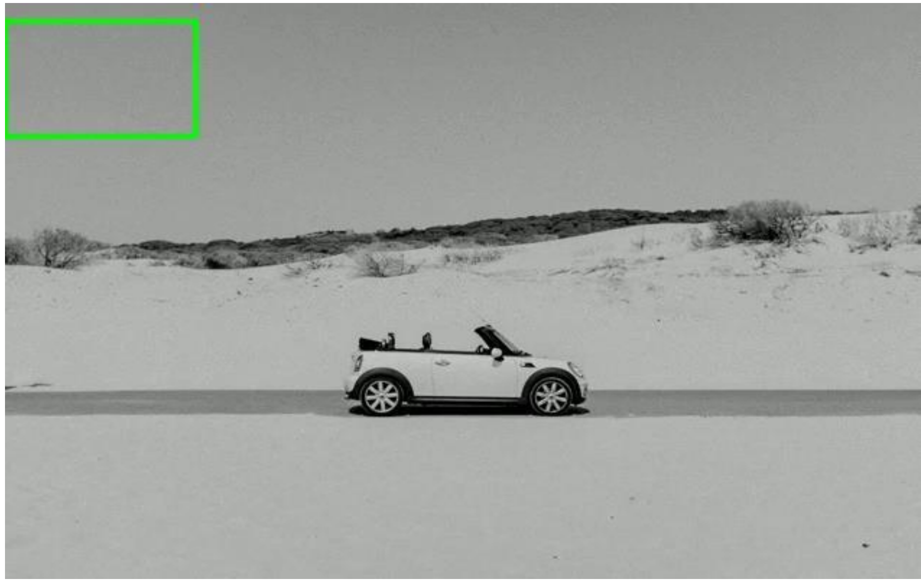
Image Segmentation

- To create a pixel-wise mask for each of the objects in an image.

Sliding window approach

- Many of the sub-section of images selected by the window won't be detecting any objects
- This will not work for a class of objects.
- Unknown position/scale/aspects of objects could not be traced.

- Can a hint be given on the places where there are objects and tell that whether the object is there in that specific location?

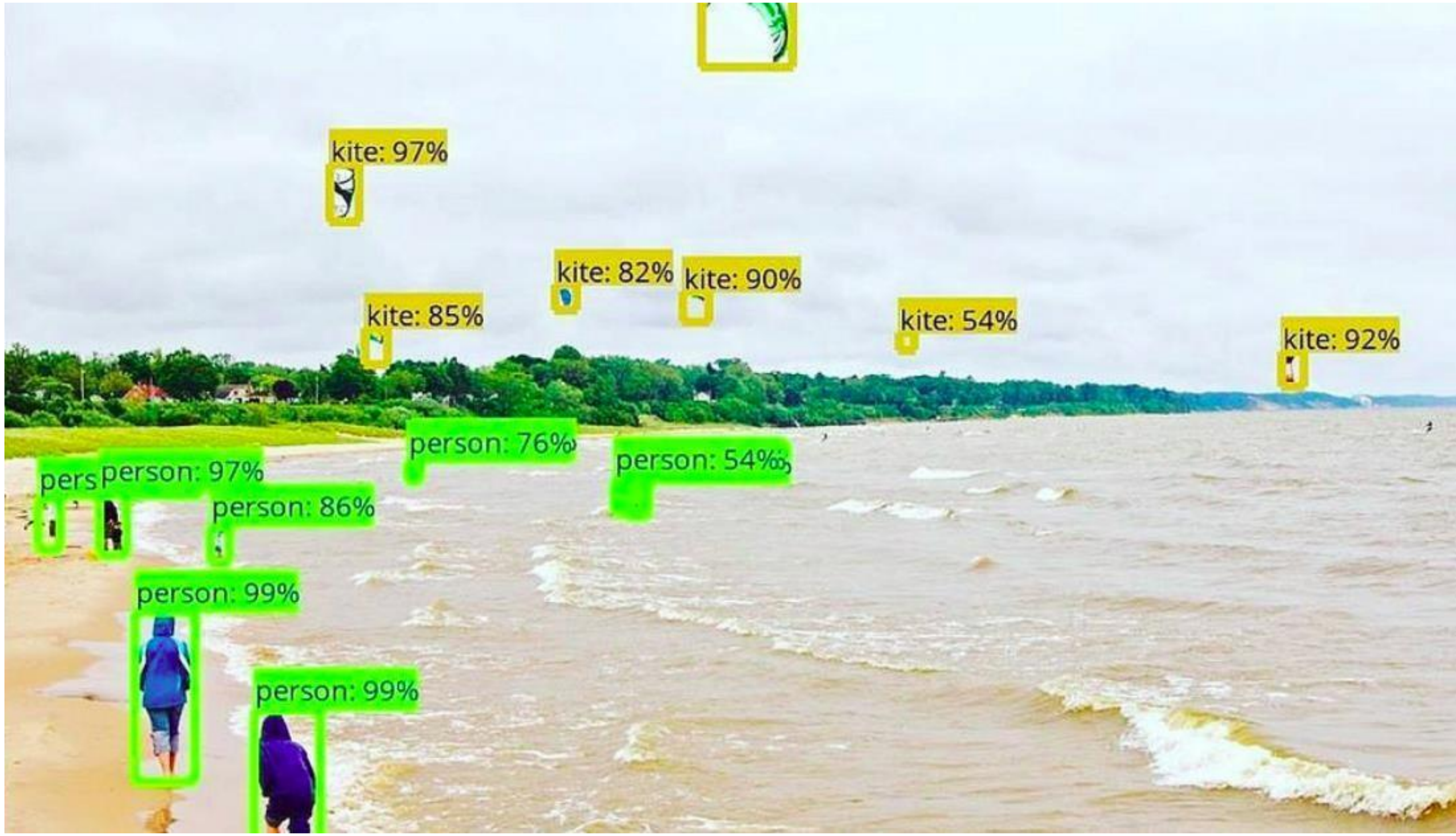


Solution:

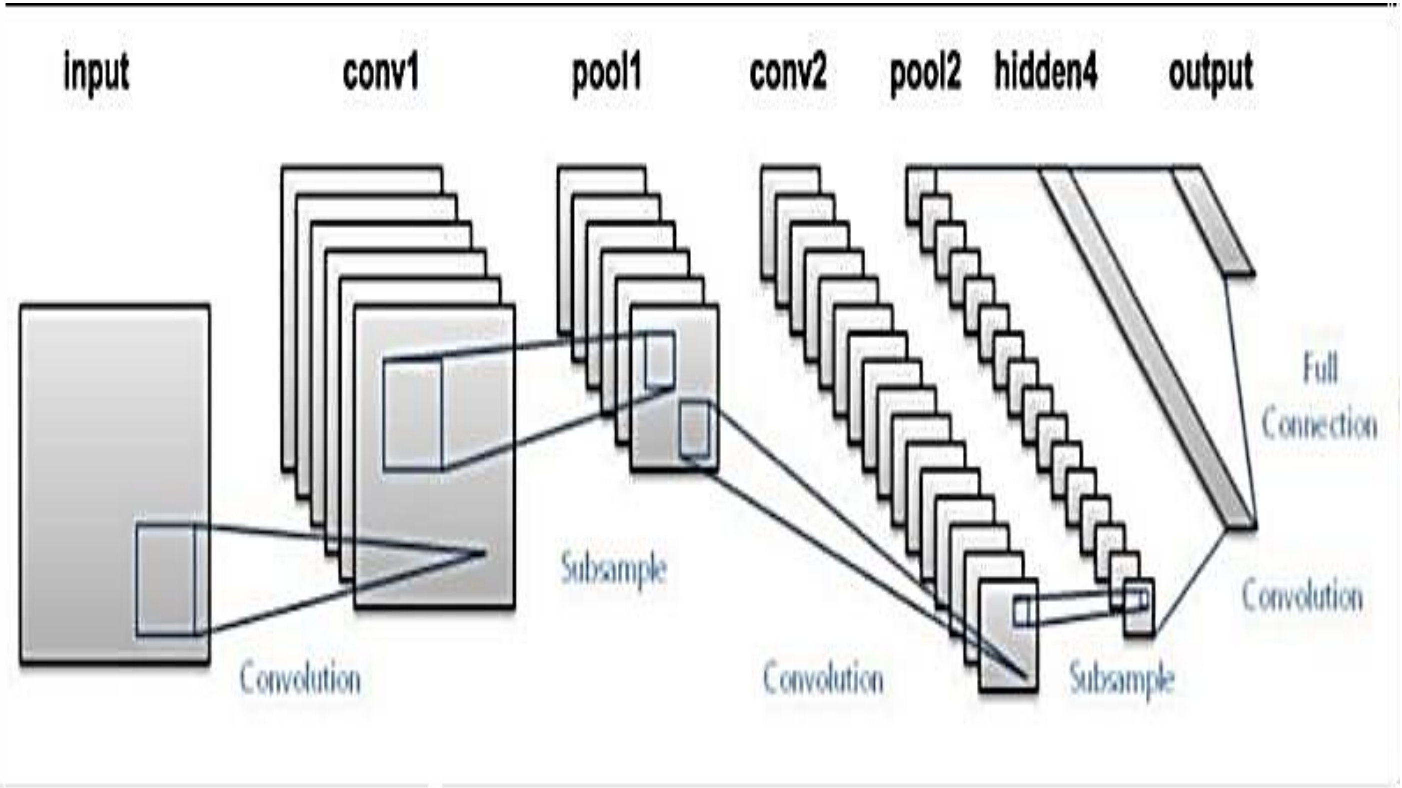
1. Selective Search (used in R-CNN)
2. Region Proposal Networks (used in Faster R-CNN)

Object Detection Task

- Each object in the image, from a person to a kite, have been located and identified with a certain level of precision.



Object Detection Task in CNN



- We pass an image to the network, and it is then sent through various convolutions and pooling layers. Finally, we get the output in the form of the object's class. Fairly straightforward, isn't it?

General steps in Object Detection Using a CNN

1. First, we take an image as input

2. Then we divide the image into various regions

3. We will then consider each region as a separate image.

4. Pass all these regions (images) to the CNN and classify them into various classes.

5. Once we have divided each region into its corresponding class, we can combine all these regions to get the original image with the detected object

General Object Detection Problem Using a CNN

- The problem with using this approach is that the objects in the image can have different aspect ratios and spatial locations.
- For instance, in some cases, the object might be covering most of the image, while in others the object might only be covering a small percentage of the image.
- The shapes of the objects might also be different (which happens a lot in real-life use cases).

As a result of these factors, we would require a very large number of regions resulting in a huge amount of computational time.

So to solve this problem and reduce the number of regions, we can use region-based CNN.

R-CNN

- The goal of R-CNN is to take in an image, and correctly identify where the main objects (via a bounding box) in the image.
- **Inputs**: Image
- **Outputs**: Bounding boxes + labels for each object in the image.
- R-CNN creates these bounding boxes, or region proposals, using a process called Selective Search.

Understanding Region-based CNNs

- Instead of working on a massive number of regions, the RCNN algorithm proposes a bunch of boxes in the image and checks if any of these boxes contain any object.
- RCNN uses selective search to extract these boxes from an image (these boxes are called regions).

Let's first understand what selective search is and how it identifies the different regions

- There are basically four regions that form an object: varying scales, colors, textures, and enclosure

Understanding Region-based CNNs

- Input image

Initial sub-segmentations so that we have multiple regions from this image.

Then combines similar regions to form a larger region (based on color similarity, texture similarity, size similarity, and shape compatibility).

- Finally, these regions then produce the final object locations (Region of Interest).

Steps Involved in RCNN to Detect Objects

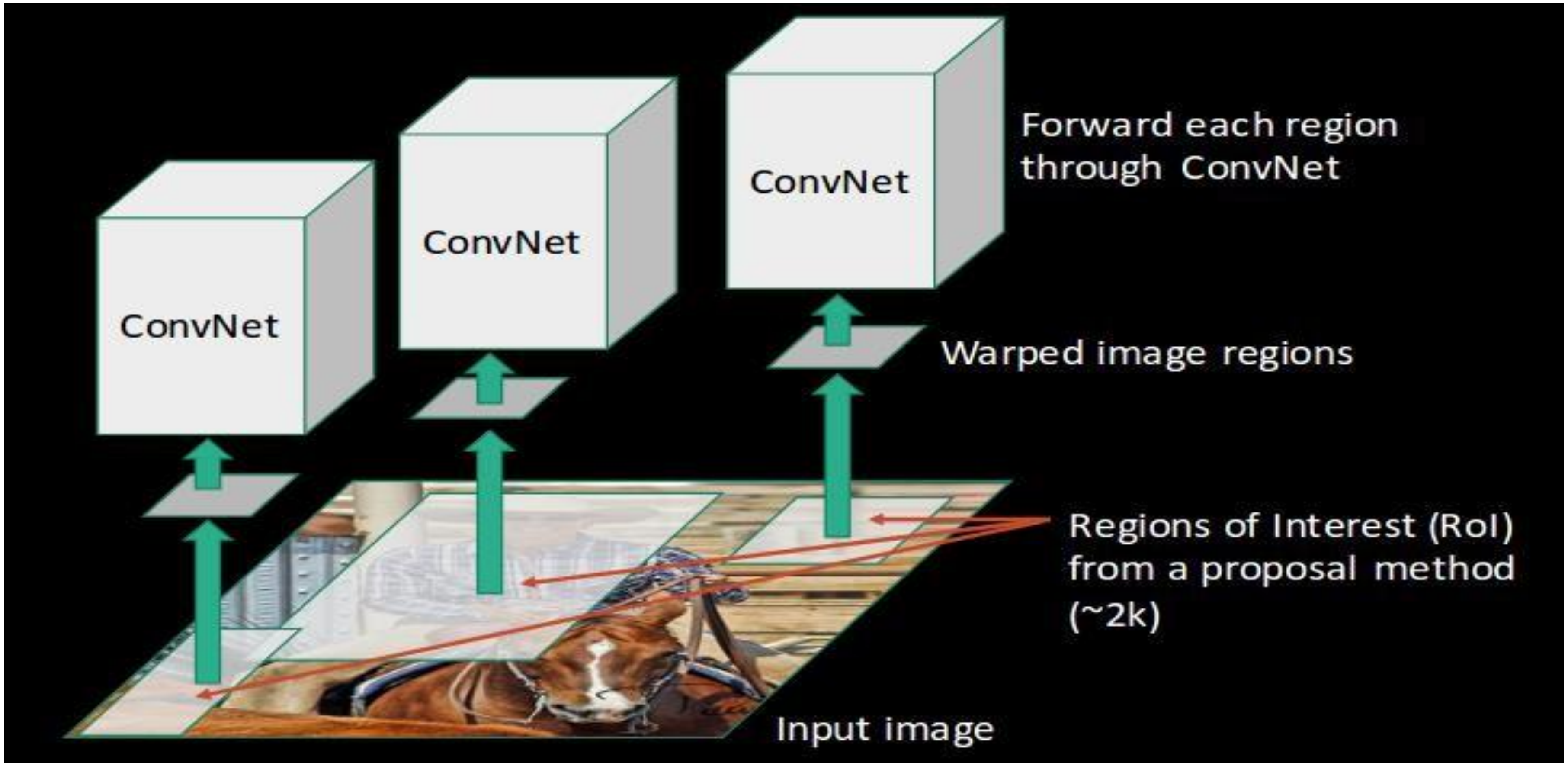
1. Use **Selective Search** to generate around **2000 region proposals** (Region of Interests – ROIs) from the input image. These are candidate bounding boxes where an object might be present
2. Each proposed **region is warped to a fixed size** (e.g., 224x224) and **passed through a pre-trained CNN** (like AlexNet or VGG) **to extract feature** vectors.
3. These feature vectors are **then passed into class-specific SVMs** (Support Vector Machines) **to classify the object present in the region**.
4. A separate **regression model is trained to refine the bounding box** coordinates to better fit the object.

Example

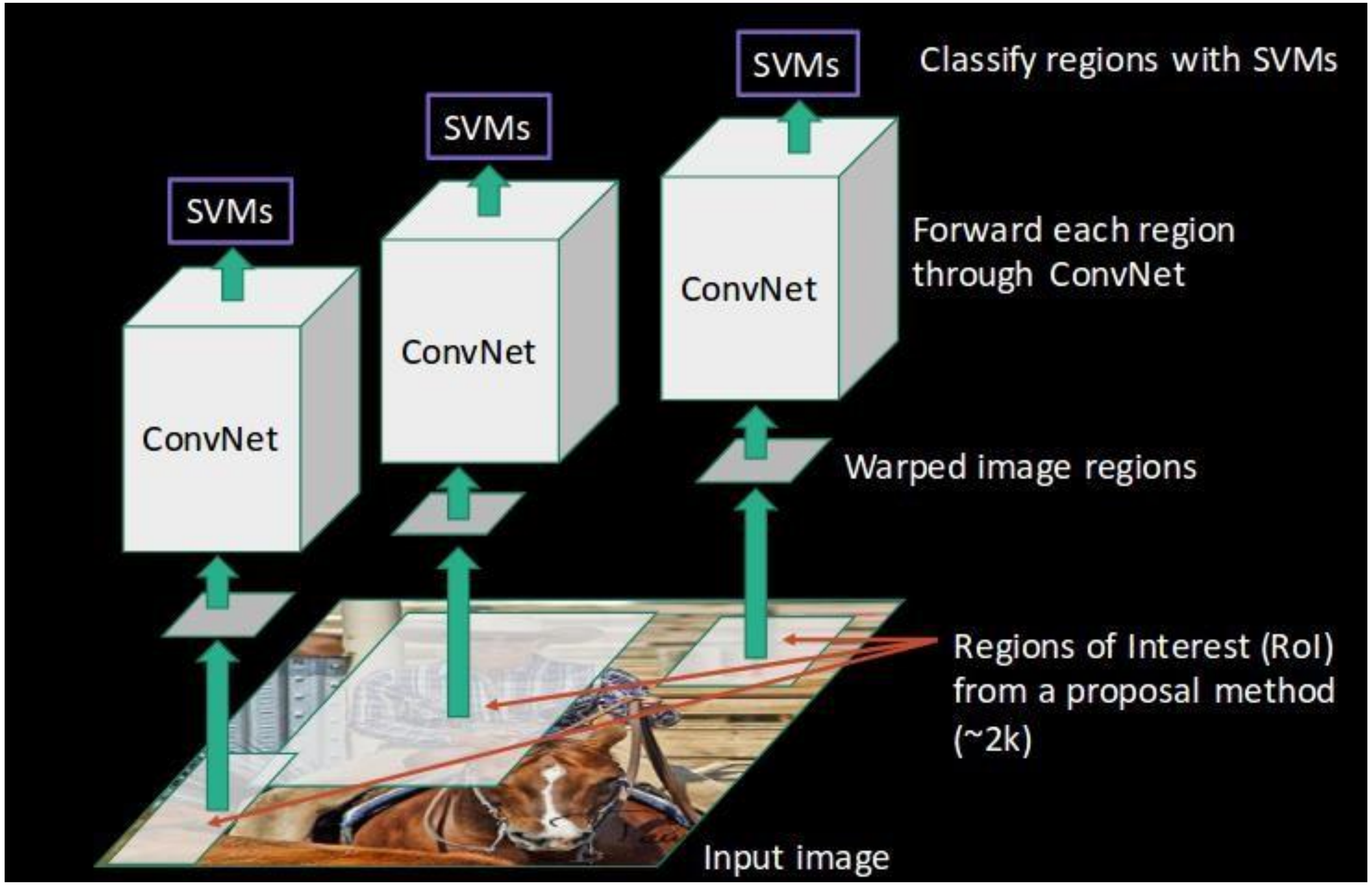
- Input image

1. Get the Regions of Interest (ROI) By using selective search

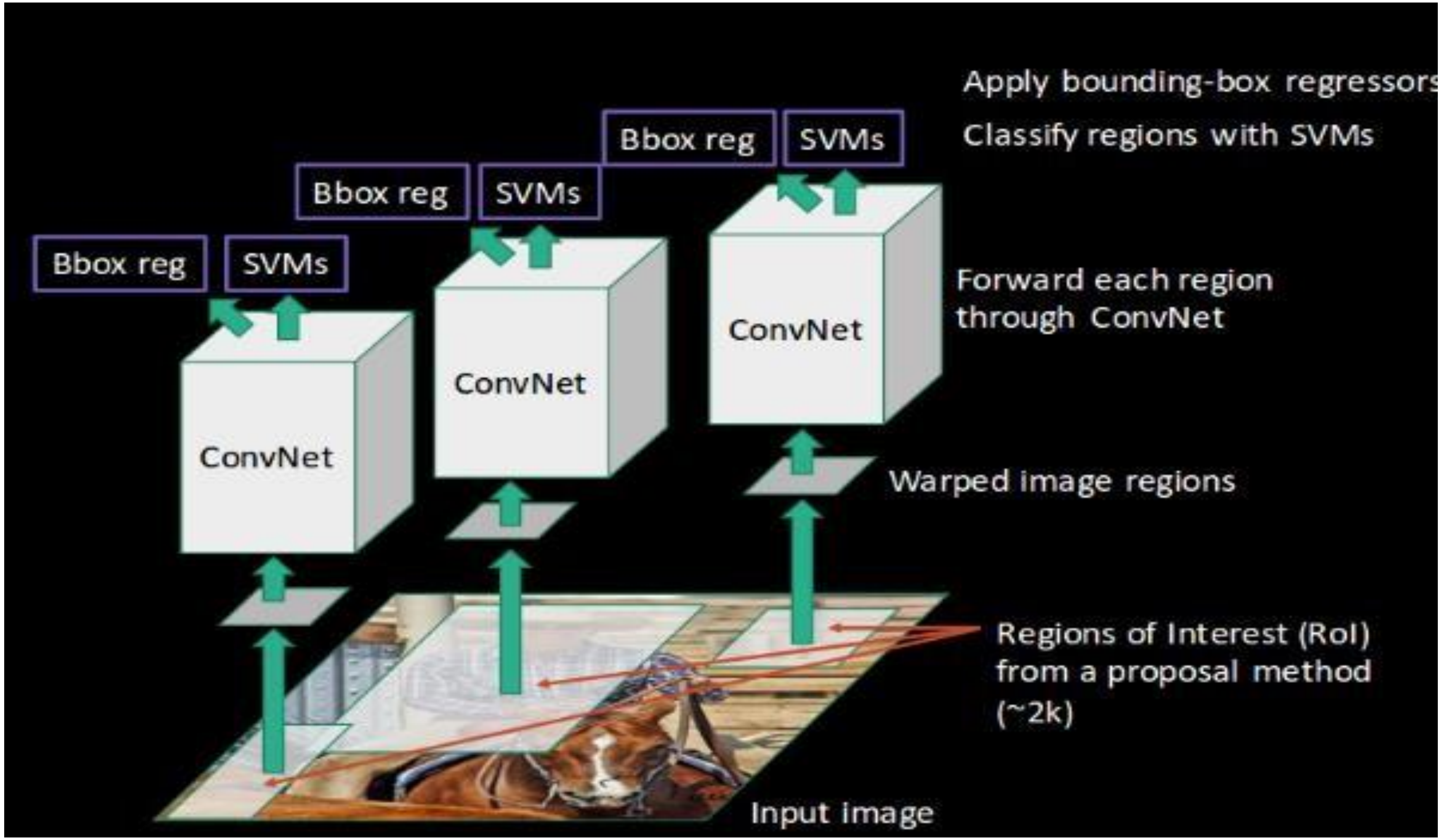
2.All these regions are then reshaped as per the input of the CNN, and each region is passed to the ConvNet:



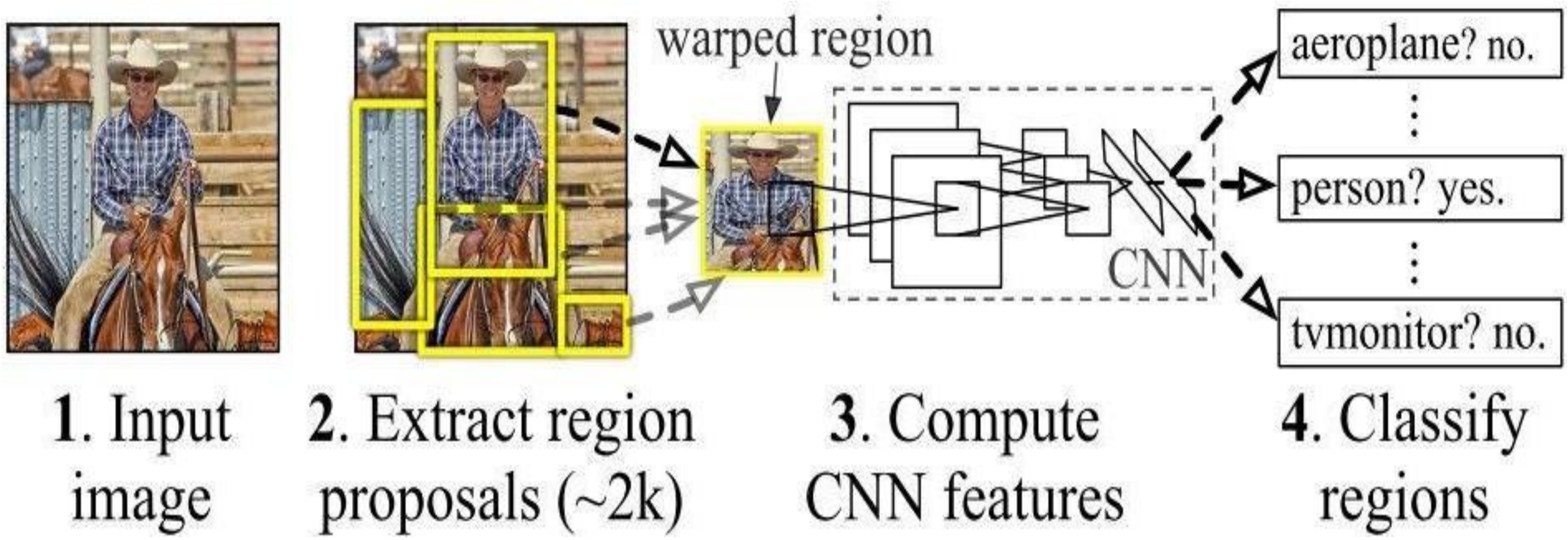
3. CNN then extracts features for each region and SVMs are used to divide these regions into different classes



4. Finally, a bounding box regression (Bbox reg) is used to predict the bounding boxes for each identified region.



Over All Representation of RCNN



Advantages and Applications of RCNN

- It is used in autonomous vehicles for perceiving objects in their surroundings to ensure a safe driving experience.
- In the field of construction, RCNN can be used for maintenance work like analyzing high-resolution pictures of rust.
- In the manufacturing industry, RCNN can be used for defective product identification and automated inspections.

Problems with RCNN

- Extracting 2,000 regions for each image based on selective search.
- Extracting features using CNN for every image region. Suppose we have N images, then the number of CNN features will be N*2,000.
- The entire process of object detection using RCNN has three models:

  CNN for feature extraction

  Linear SVM classifier for identifying objects

  Regression model for tightening the bounding boxes.
- All these processes combine to make RCNN very slow. It takes around **40-50 seconds** to make predictions for each new image, which essentially makes the model cumbersome and practically impossible to build when faced with a gigantic dataset.

Fast R-CNN

- Fast R-CNN overcomes several issues in R-CNN. As its name suggests, one advantage of the Fast R-CNN over R-CNN is its speed.
- Instead of running a CNN 2,000 times per image, we can run it just once per image and get all the regions of interest (regions containing some object).
- In Fast RCNN, we feed the input image to the CNN, which in turn generates the convolutional feature maps.
- Using these maps, the regions of proposals are extracted.
- We then use a RoI pooling layer to reshape all the proposed regions into a fixed size, so that it can be fed into a fully connected network.

Steps Involved in Fast R-CNN to Detect Objects

1. The image is passed to a ConvNet which in turn generates the Regions of Interest.

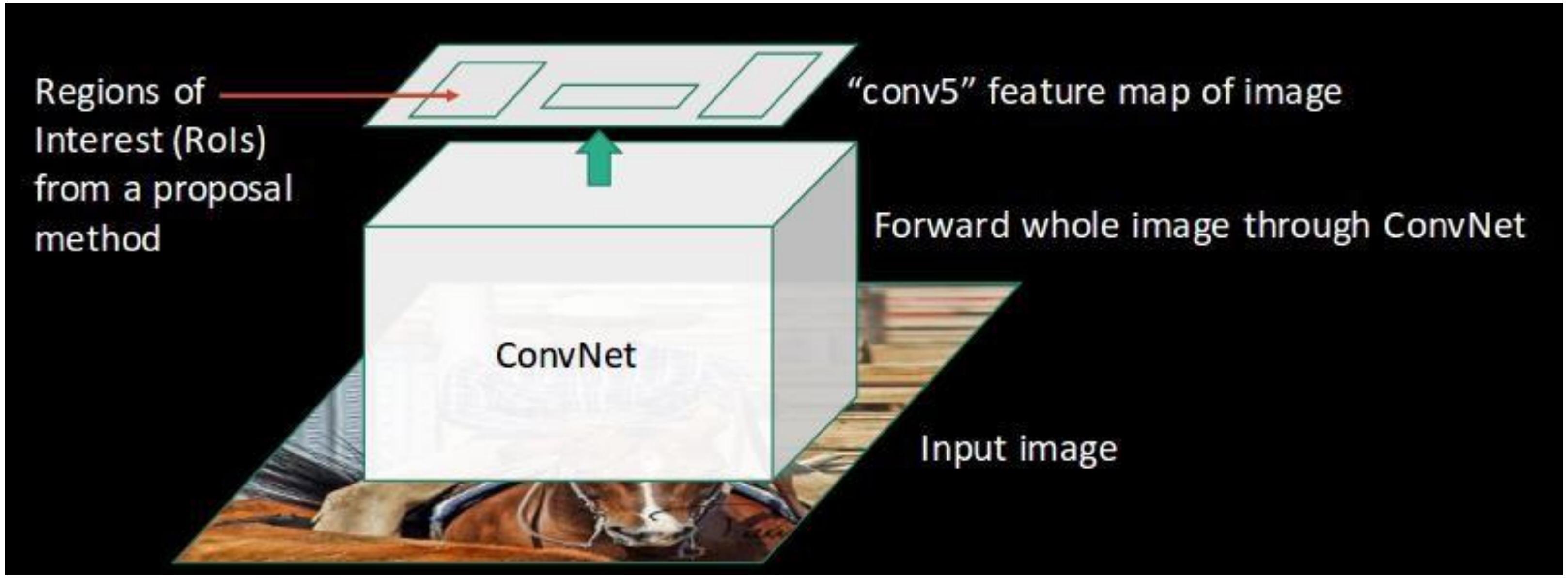
3. A RoI pooling layer is applied on all of these regions to reshape them as per the input of the ConvNet. Then, each region is passed on to a fully connected network.

4. A softmax layer is used on top of the fully connected network to output classes. Along with the softmax layer, a linear regression layer is also used parallelly to output bounding box coordinates for predicted classes.

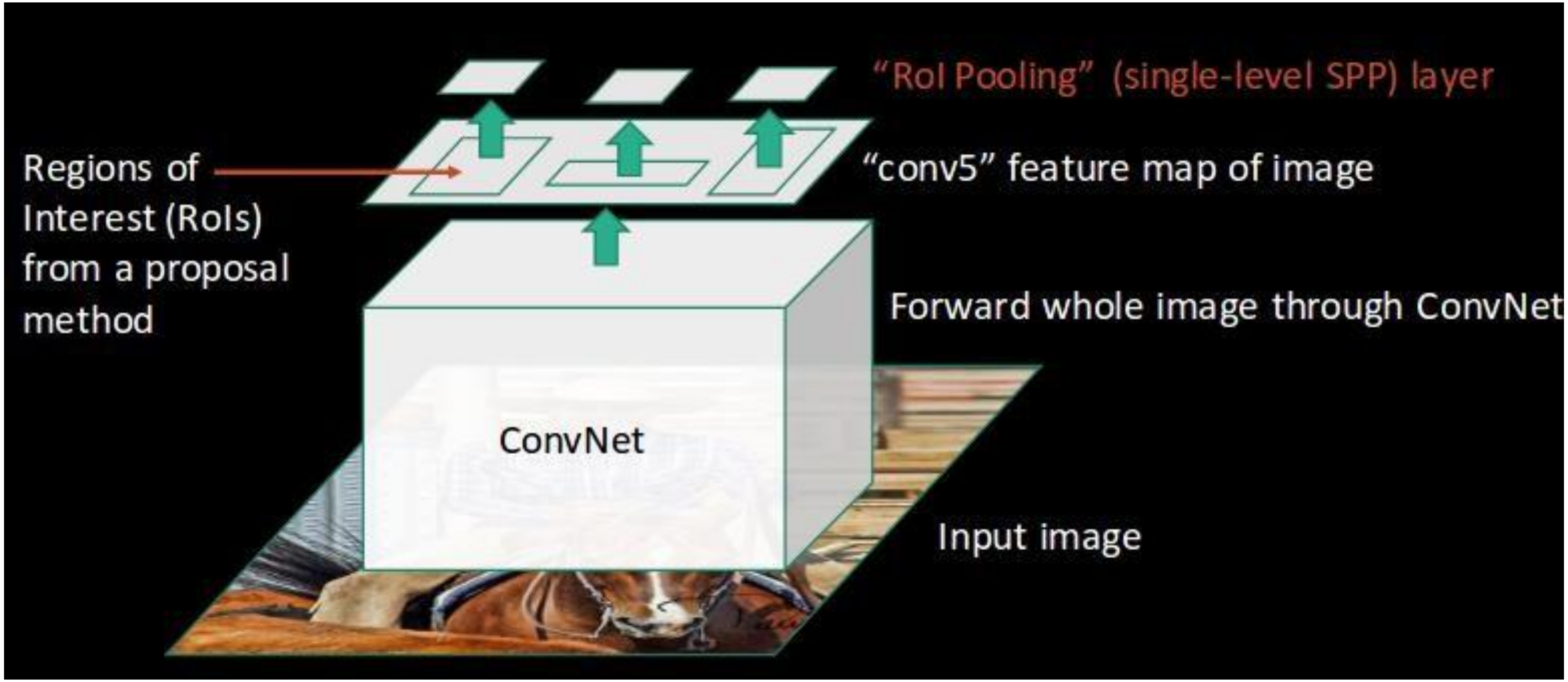
Example

- Input image

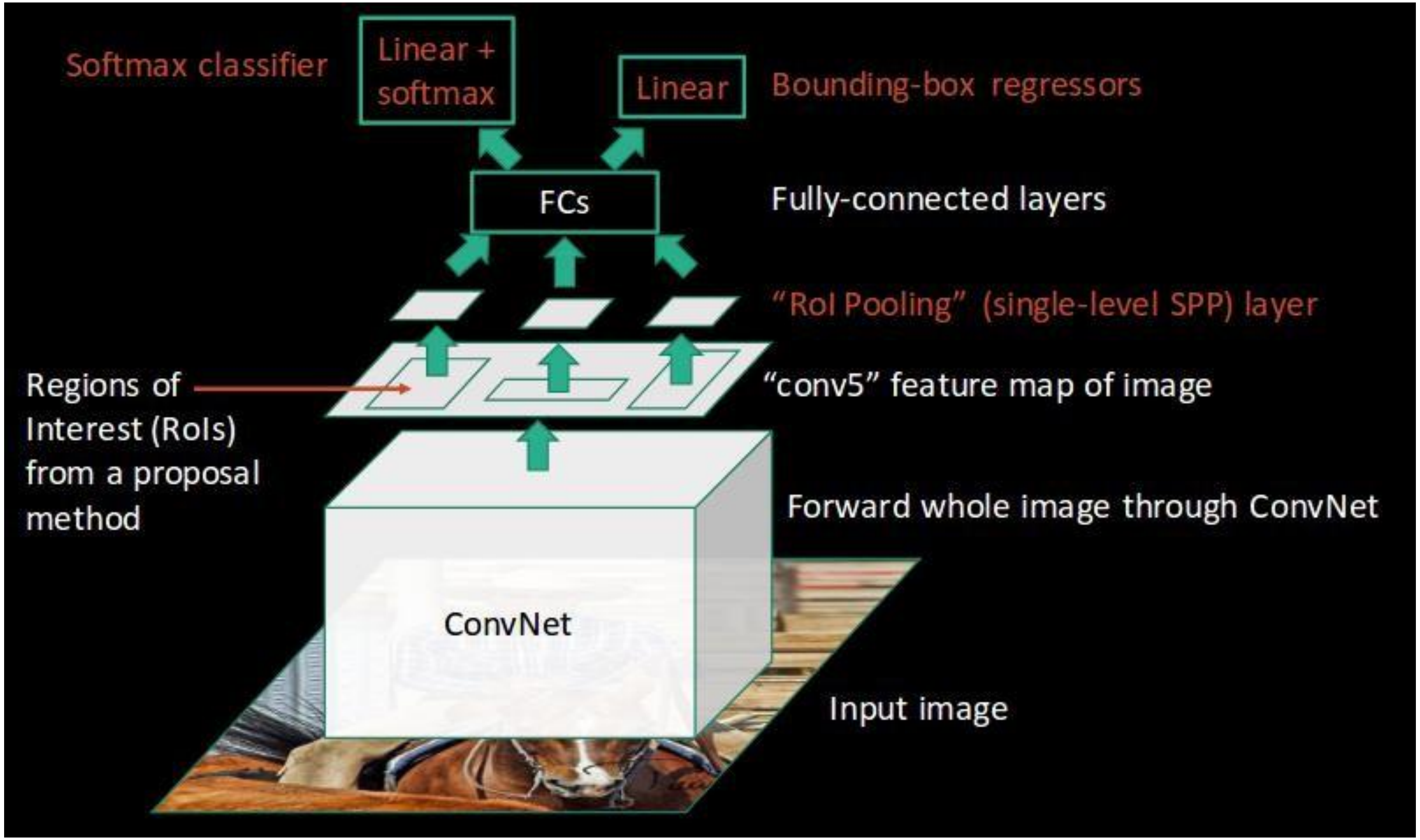
1. This image is passed to a ConvNet which returns the region of interest accordingly



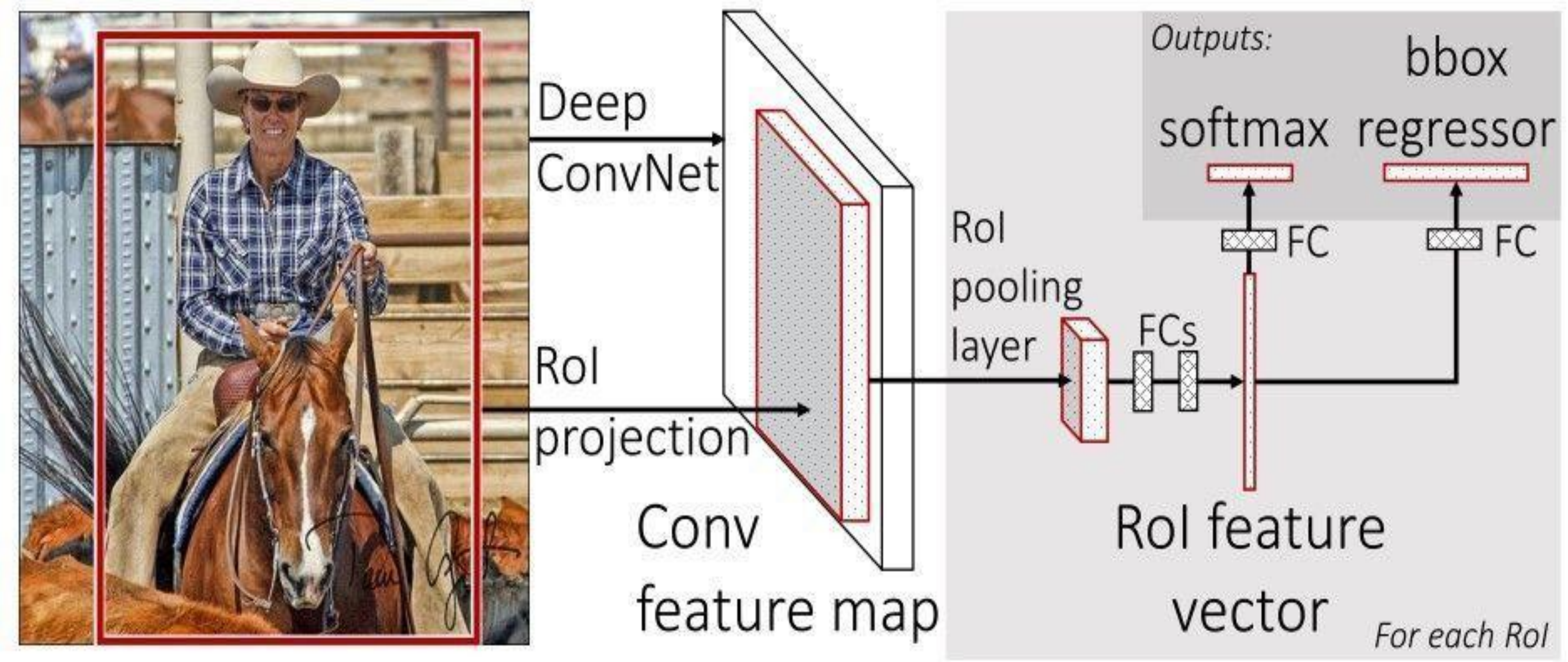
2. Then we apply the RoI pooling layer on the extracted regions of interest to make sure all the regions are of the same size



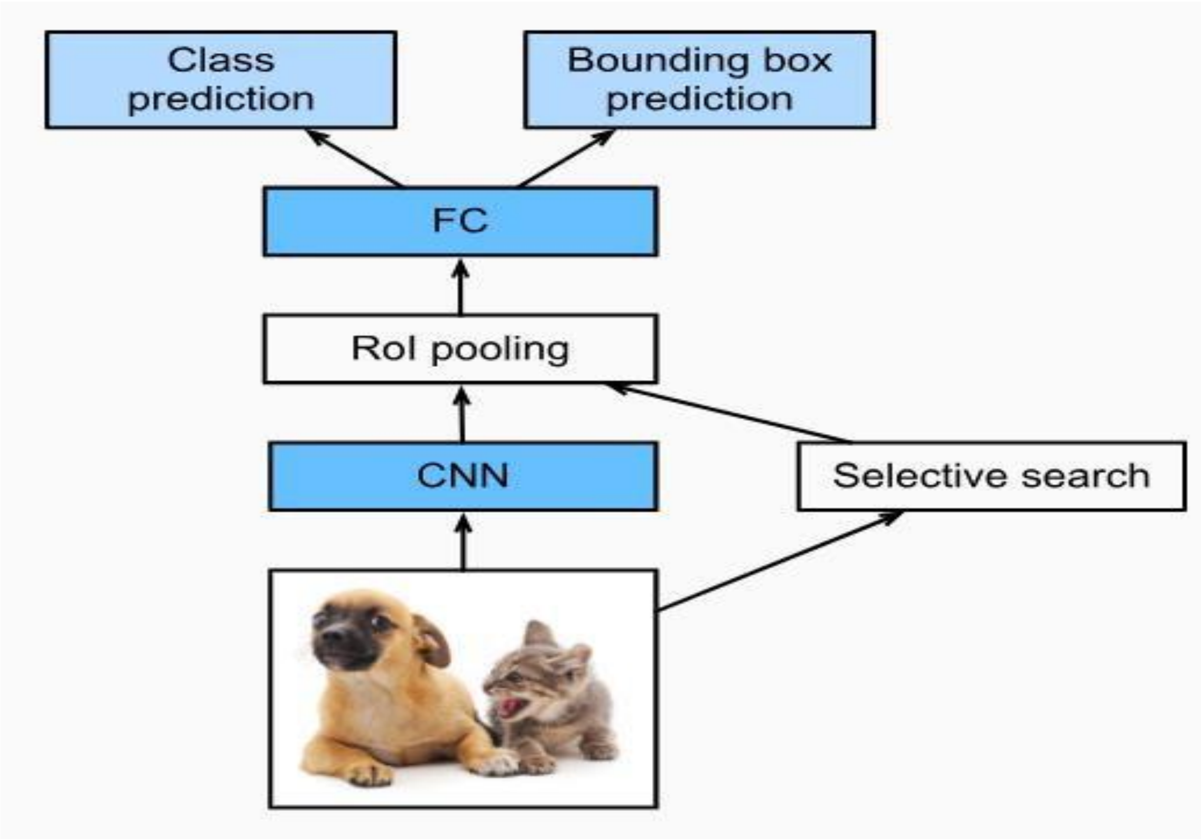
3. Finally, these regions are passed on to a fully connected network which classifies them, as well as returns the bounding boxes using softmax and linear regression layers simultaneously



Example


Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
RoI feature
vector
Outputs:
softmax
bbox
regressor
FC
FC
For each RoI

Over All Representation of Fast-RCNN



This is how Fast RCNN resolves two major issues of RCNN, i.e., **passing one instead of 2,000 regions per image to the ConvNet,** and using one instead of **three different models for extracting features, classification and generating bounding boxes**.

Faster RCNN

- Faster RCNN is the modified version of Fast RCNN.
- The major difference between them is that Fast RCNN uses the selective search for generating Regions of Interest, while Faster RCNN uses "Region Proposal Network", aka RPN.
- RPN takes image feature maps as input and generates a set of object proposals, each with an objectness score as output.
- To reduce region proposals without loss of accuracy.

Steps Involved in Faster RCNN to Detect Objects

1. We take an image as input and pass it to the ConvNet which returns the feature map for that image.

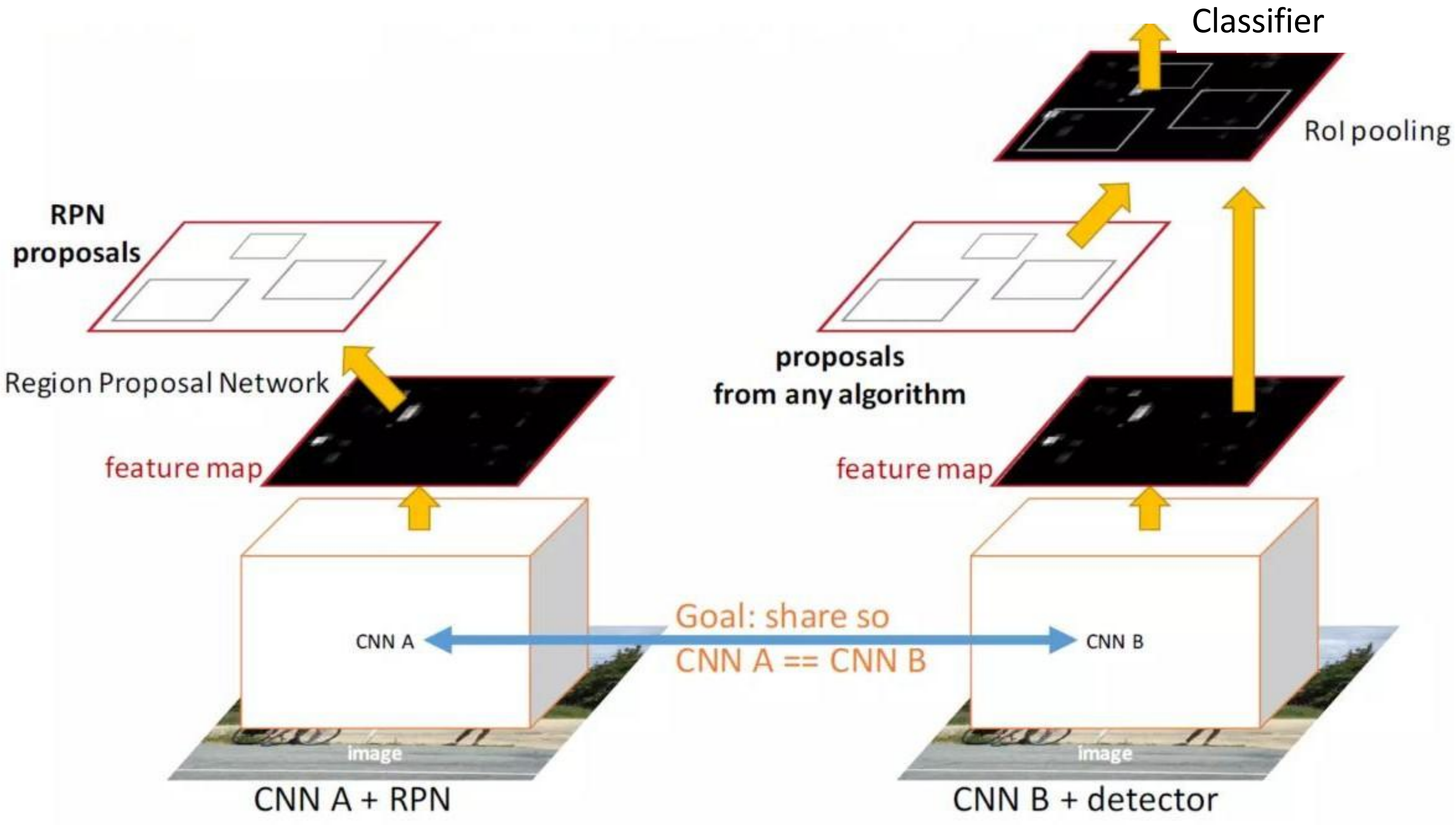
2. Region proposal network is applied on these feature maps. This returns the object proposals along with their objectness score.

3. A RoI pooling layer is applied to these proposals to bring down all the proposals to the same size.

4. Finally, the proposals are passed to a fully connected layer which has a softmax layer and a linear regression layer at its top, to classify and output the bounding boxes for objects.

Faster RCNN

- Share the Features



Region Proposal Network (RPN)

1. So in the very first step, our input image goes through the Convolutional Neural Network and its last layer gives the features maps as output.

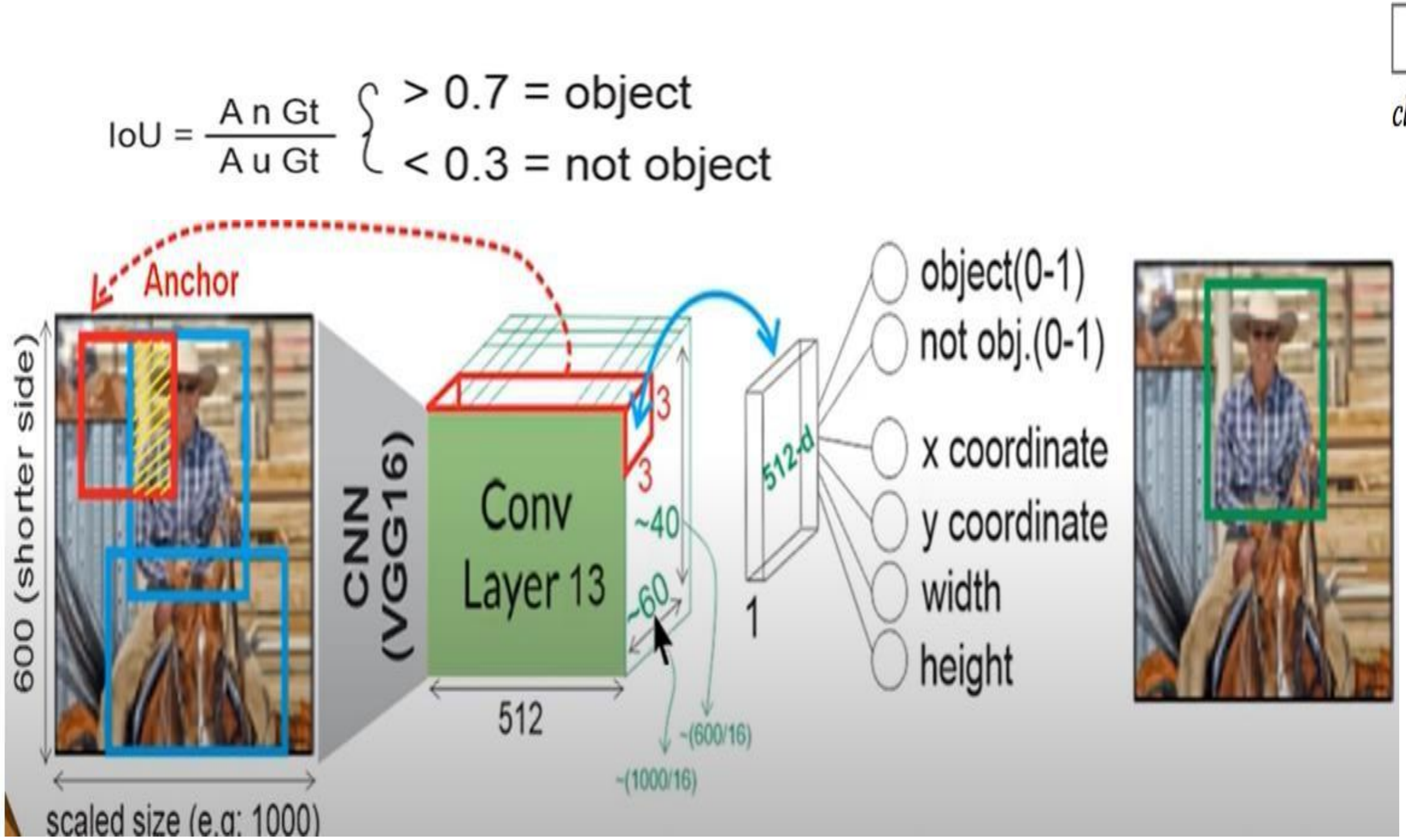
2. In this step, **a sliding window** is run through the feature maps obtained from the last step. The size of the sliding window is n*n. For each sliding window, a particular **set of anchors are generated** but with 3 different **aspect ratios** (1:1, 1:2, 2:1) and 3 different **scales** (128, 256 and

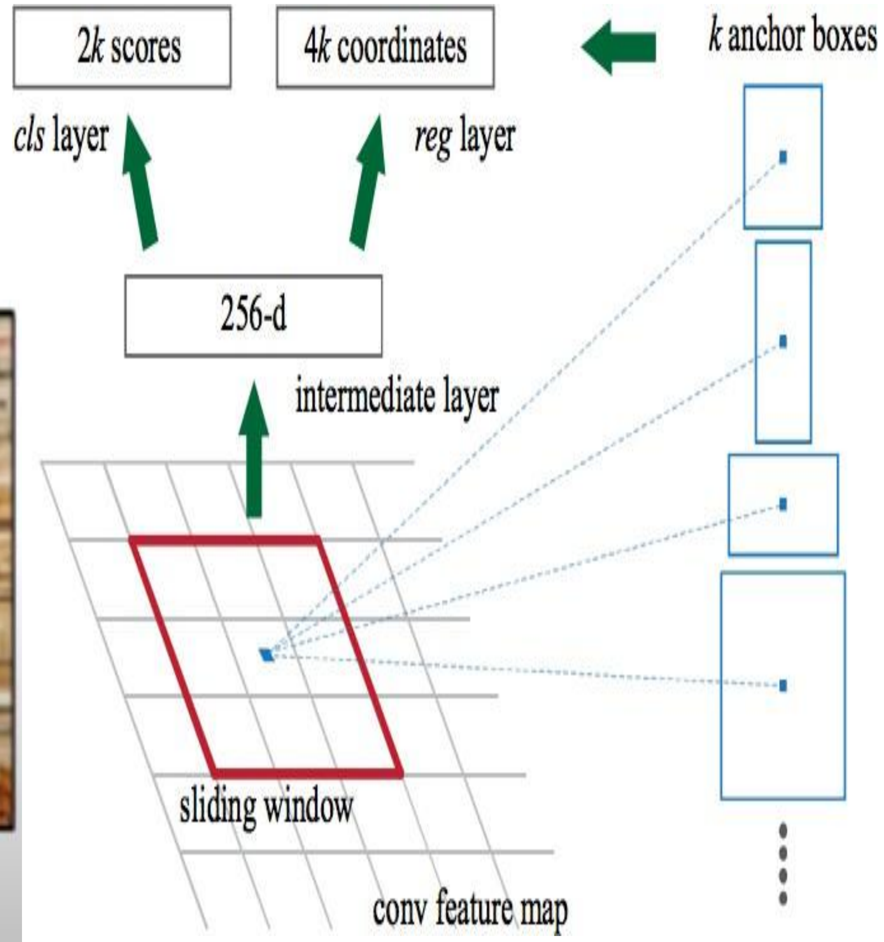
512).

3. In step 3, Localizing and classifying the anchor box is done by **Bounding box Regressor layer** and **Bounding box Classifier layer**.

- The bounding Box Classifier calculates the IoU score of the Ground Truth Box with anchor boxes and classifies the Anchor box in either Foreground or Background with a certain probability aka objectness score.

Region Proposal Network





40*60=2400 possible sliding window locations/anchors

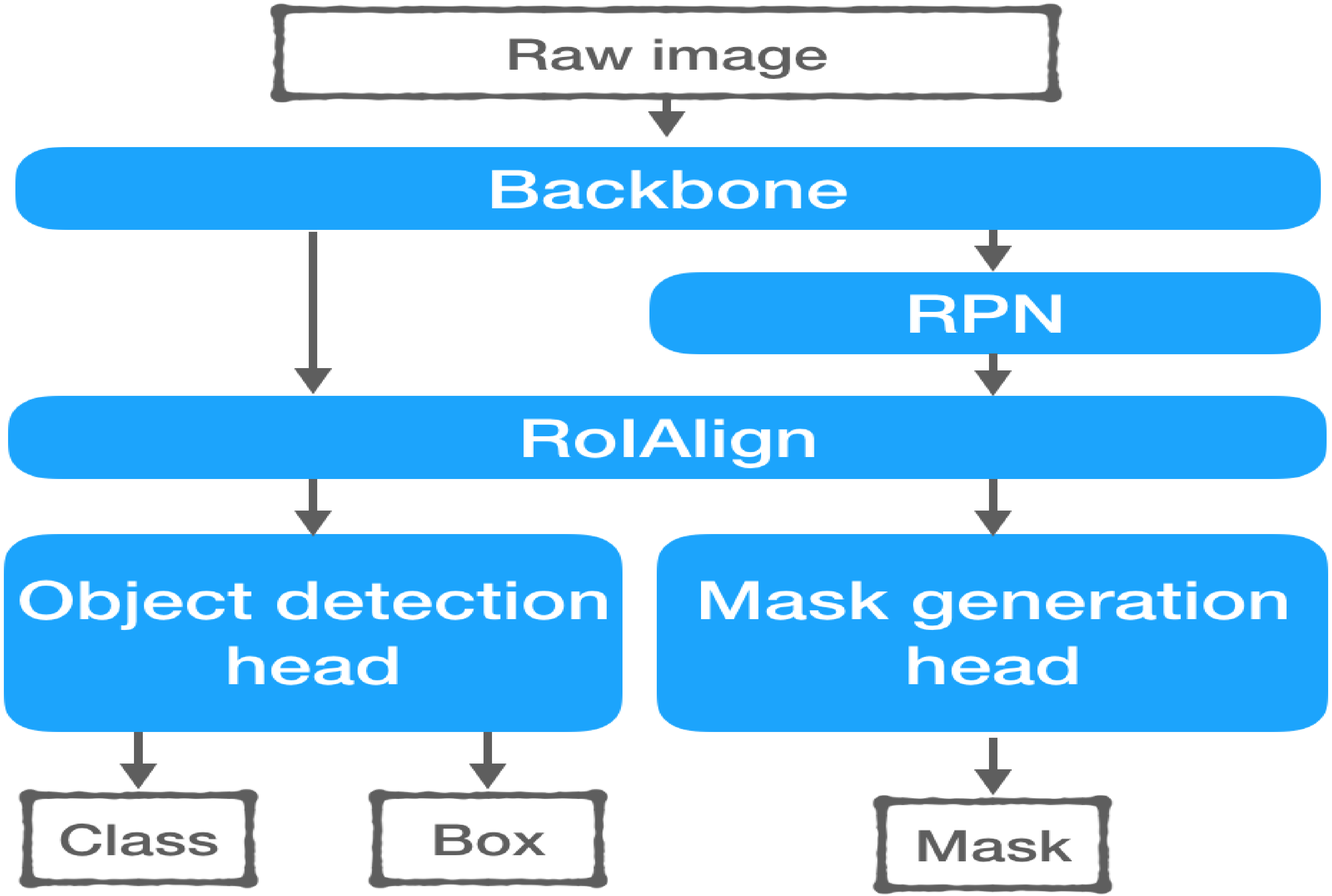
Mask R-CNN

- Mask R-CNN is basically an extension of Faster R-CNN.
- Mask R-CNN has an extra branch for outputting the Segmentation masks on each Region of Interest (RoI) in a per-pixel way.
- Therefore, it has three outputs
  - Class Label,
  - Bounding-Box Offset, and
  - Object Mask for each detected object.

Mask R-CNN

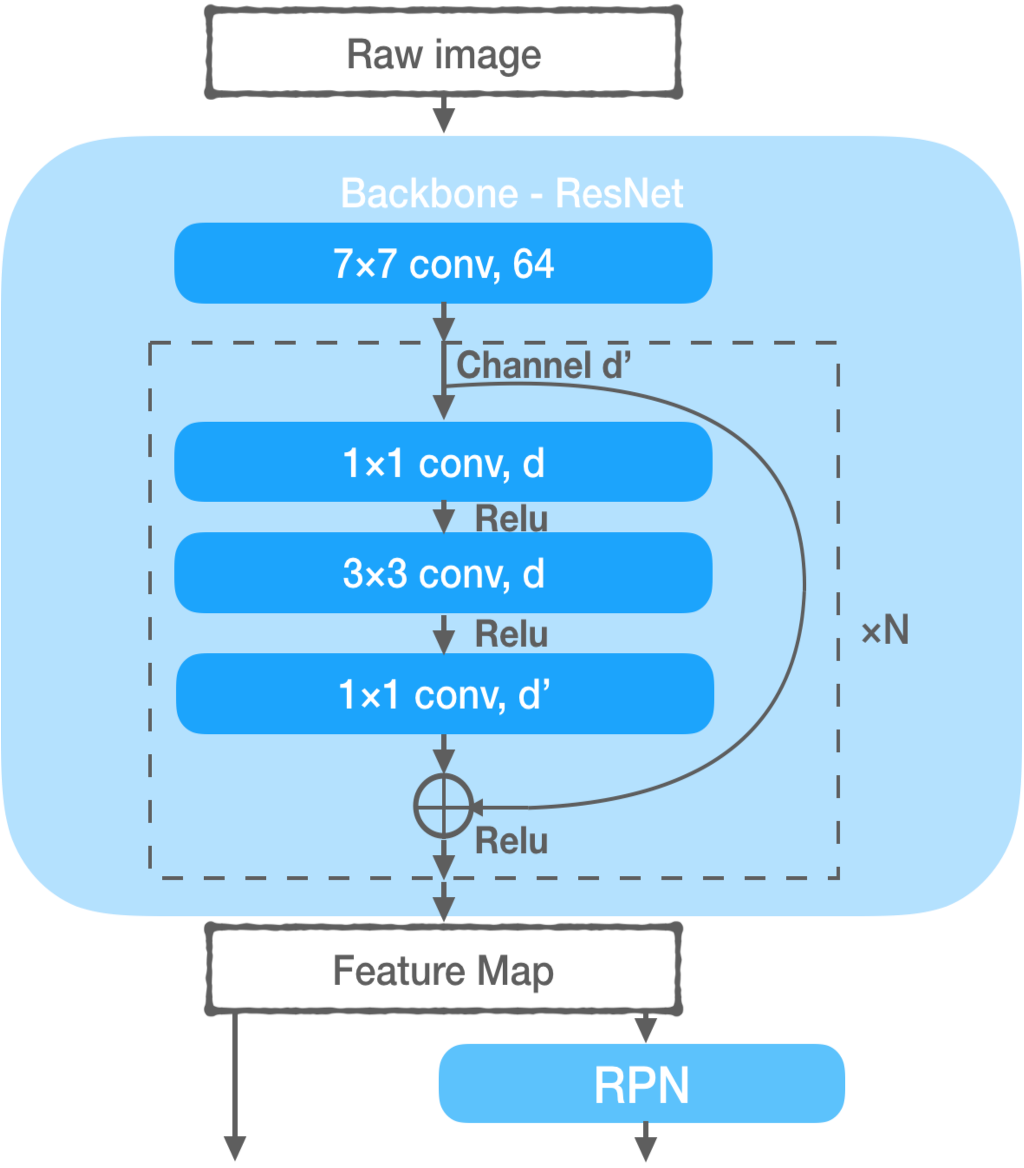
- Mask R-CNN is a popular deep learning framework, for instance segmentation tasks in the computer vision field.
- Automatically segment and construct pixel-wise masks for every object in an image.
- It adds fully convolutional networks (FCN) to Faster R-CNN to generate a mask for each object, while Faster R-CNN, Fast R-CNN, R-CNN is for bounding-box object detection.
- There are two stages of Mask RCNN. First, it generates proposals about the regions where there might be an object based on the input image.
- Second, it predicts the class of the object, refines the bounding box, and generates a mask at pixel level of the object based on the first stage proposal. Both stages are connected to the backbone structure.

Mask R-CNN


Raw image
Backbone
RPN
RoIAlign
Object detection head
Mask generation head
Class
Box
Mask

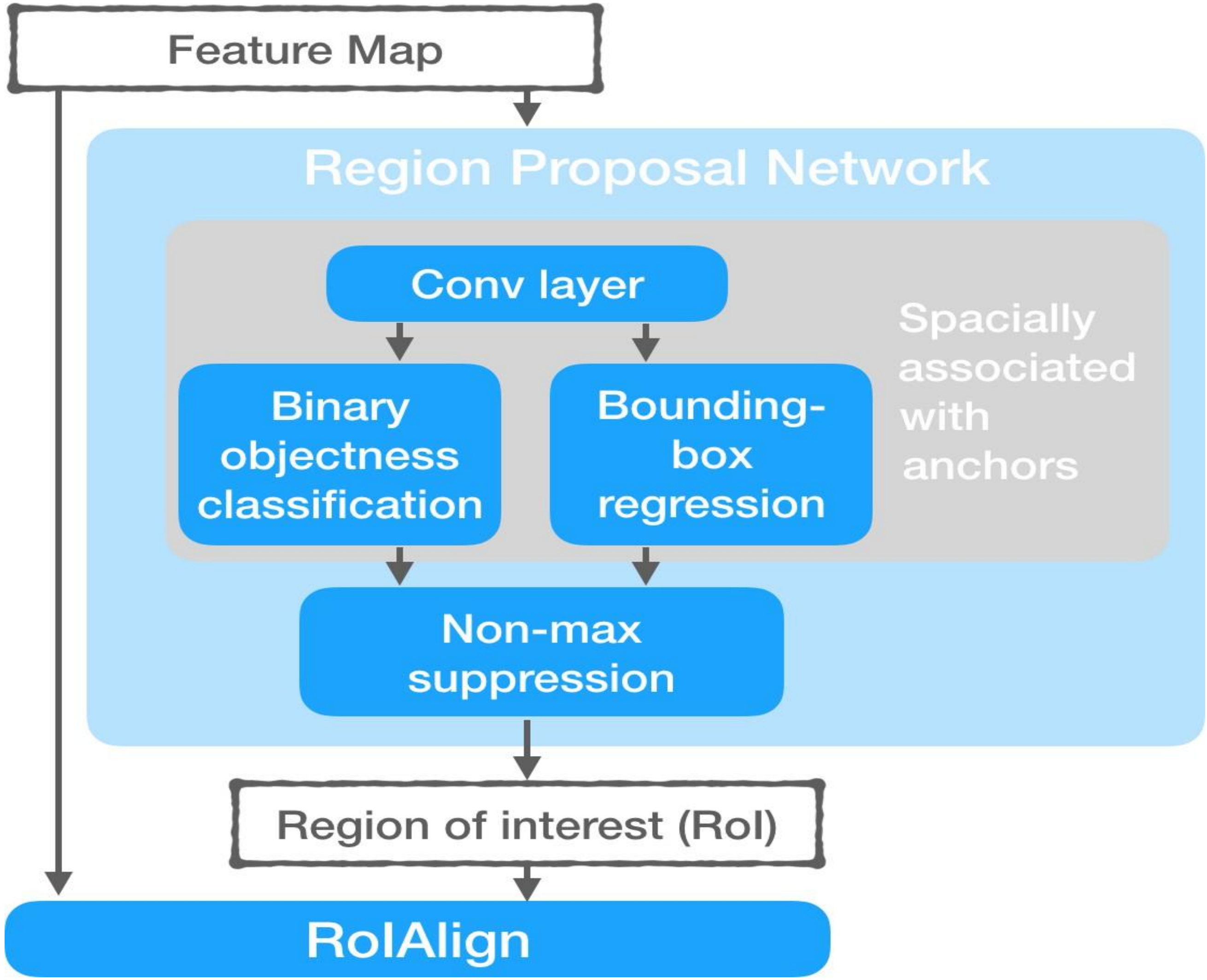
Mask R-CNN

- A backbone is the main feature extractor of Mask R-CNN.
- [Common choices of this part are residual networks (ResNets) with or without FPN.]()
- ResNet without FPN as a backbone is used.
- A raw image is fed into a ResNet backbone, data goes through multiple residual bottleneck blocks, and turns into a feature map.
- Feature map from the final convolutional layer of the backbone contains abstract information of an image, e.g., different object instances, their classes and spatial properties. It is then fed to the RPN.
- Example: 1024x1024x3 image into a 32x32x2048 feature map that is input for subsequent layers.



Mask R-CNN

Scanning the feature map and proposing regions that may have objects in them (Region of Interest or RoI).



Mask Generation Branch

- we feed RoI feature map to a transposed convolutional layer and a convolutional layer successively.
- This branch is a fully convolutional network.
- One binary segmentation mask is generated for one class.
- Then we pick the output mask according to the class prediction in object detection branch.