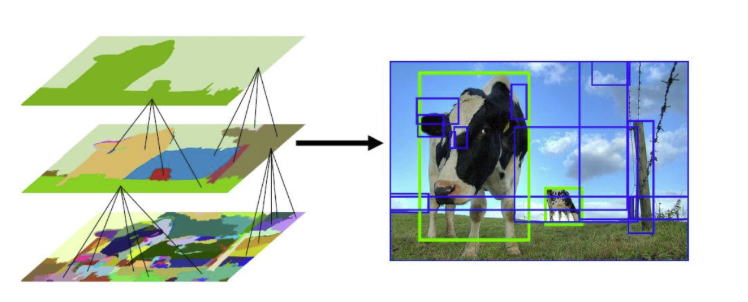
Since Convolution Neural Network (CNN) with a fully connected layer is not able to deal with the frequency of occurrence and multi objects. So, one way could be that we use a sliding window brute force search to select a region and apply the CNN model to that, but the problem with this approach is that the same object can be represented in an image with different sizes and different aspect ratios. While considering these factors we have a lot of region proposals and if we apply deep learning (CNN) to all those regions that would computationally very expensive.

*Ross Girshick et al*in 2013 proposed an architecture called R-CNN (Region-based CNN) to deal with this challenge of object detection. This R-CNN architecture uses the selective search algorithm that generates approximately *2000* region proposals. These *2000* region proposals are then provided to CNN architecture that computes CNN features. These features are then passed in an SVM model to classify the object present in the region proposal. An extra step is to perform a bounding box regressor to localize the objects present in the image more precisely.

Region proposals are simply the smaller regions of the image that possibly contains the objects we are searching for in the input image. To reduce the region proposals in the R-CNN uses a greedy algorithm called selective search.



**Selective Search**

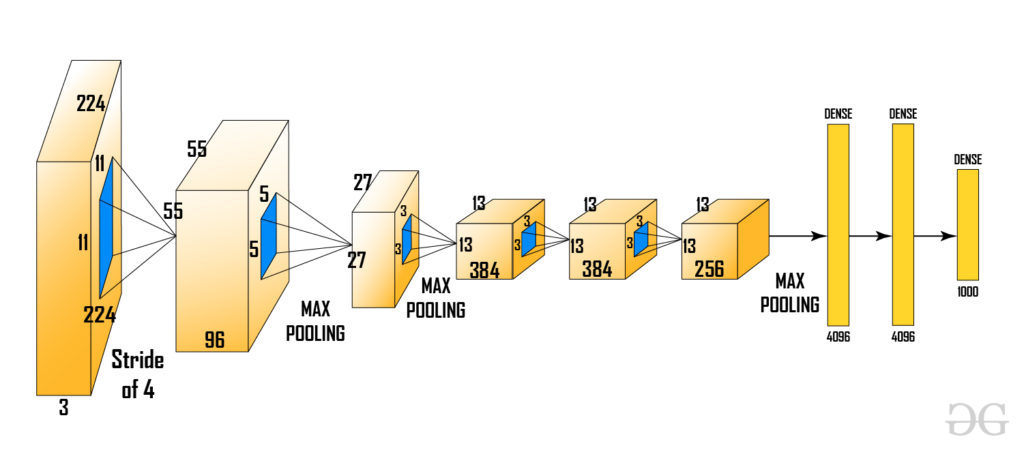
[Selective search](https://www.geeksforgeeks.org/selective-search-for-object-detection-r-cnn/) is a greedy algorithm that combines smaller segmented regions to generate region proposals. This algorithm takes an image as input and output generates region proposals on it. This algorithm has the advantage over random proposal generation in that it limits the number of proposals to approximately *2000* and these region proposals have a high recall.

### ****Algorithm****

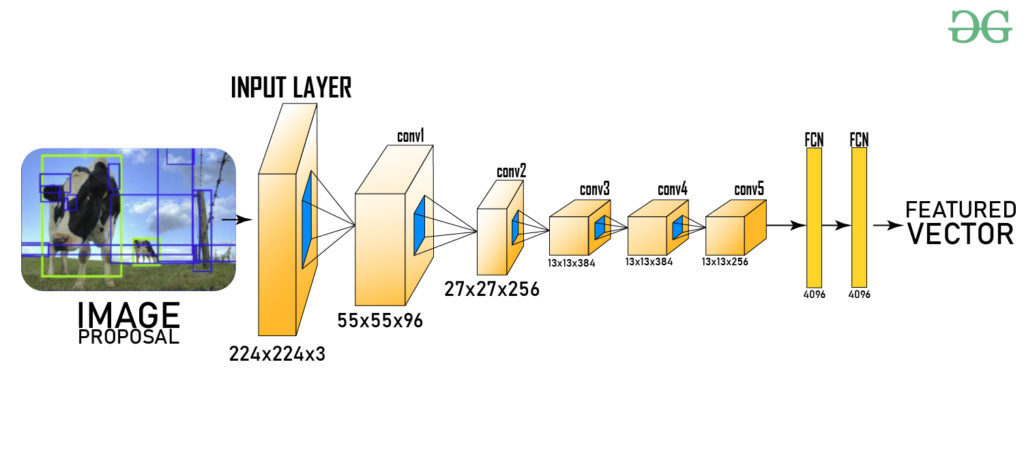
1. Generate initial sub-segmentation of the input image.
2. Combine similar bounding boxes into larger ones recursively
3. Use these larger boxes to generate region proposals for [object detection](https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/).

In Step 2 similarities are considered based on color similarity, texture similarity, region size, etc. We have discussed the selective search algorithm in great detail in [this article](https://www.geeksforgeeks.org/selective-search-for-object-detection-r-cnn/).

## ****CNN architecture of R-CNN****

After that these regions are warped into a single square of regions of dimension as required by the [CNN](https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/) model. The CNN model that we used here is a pre-trained [AlexNet](https://www.geeksforgeeks.org/ml-getting-started-with-alexnet/) model, which is the state-of-the-art CNN model at that time for image classification Let’s look at AlexNet architecture here.[](https://media.geeksforgeeks.org/wp-content/uploads/20200217183955/new6.jpg)Here the input of AlexNet is *(227, 227, 3)*. So, if the region proposals are small and large then we need to resize that region proposal to given dimensions.

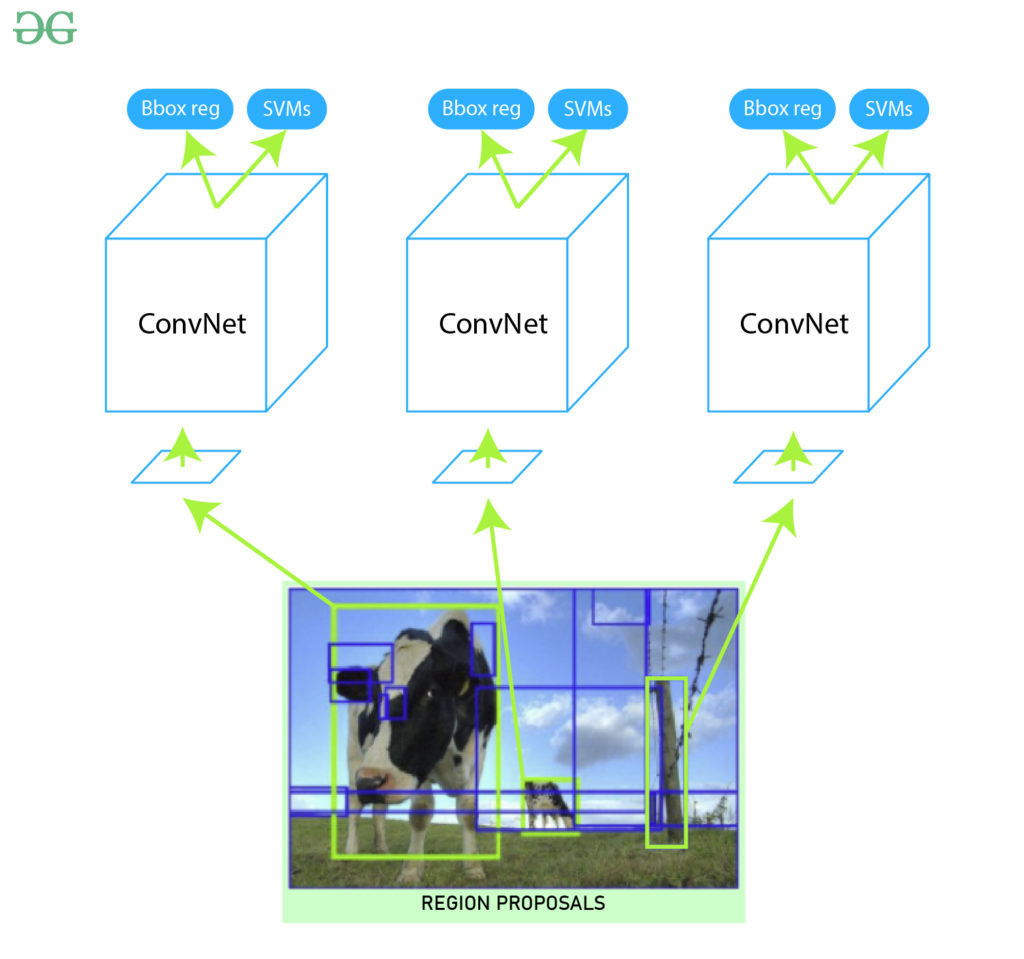
From the above architecture, we remove the last softmax layer to get the *(1, 4096)*feature vector. We pass this feature vector into [SVM](https://www.geeksforgeeks.org/support-vector-machine-algorithm/) and bounding box regressor.



## ****SVM (Support Vector Machine)****

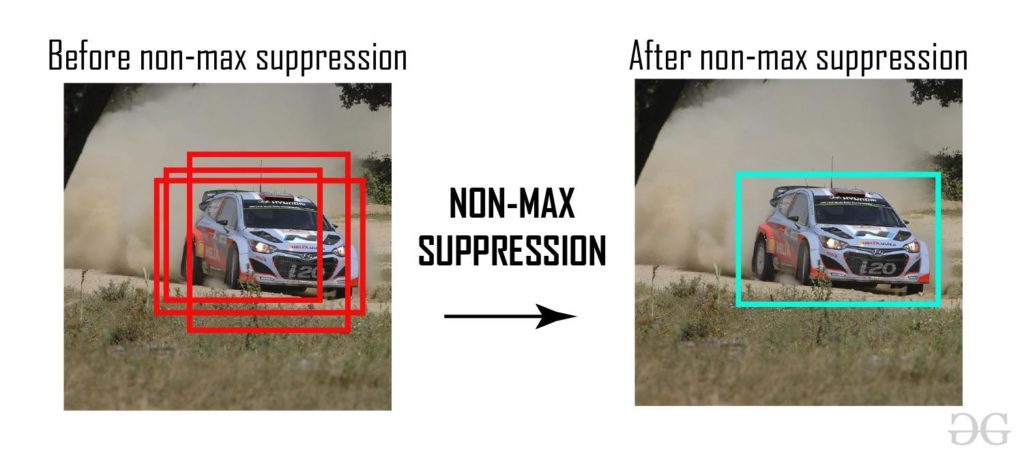
The feature vector generated by CNN is then consumed by the binary SVM which is trained on each class independently. This SVM model takes the feature vector generated in previous CNN architecture and outputs a confidence score of the presence of an object in that region. However, there is an issue with training with SVM is that we required AlexNet feature vectors for training the SVM class. So, we could not train AlexNet and SVM independently in paralleled manner. This challenge is resolved in future versions of [R-CNN](https://www.geeksforgeeks.org/r-cnn-region-based-cnns/) ([Fast R-CNN](https://www.geeksforgeeks.org/fast-r-cnn-ml/), [Faster R-CNN](https://www.geeksforgeeks.org/faster-r-cnn-ml/), etc.).

## ****Bounding Box Regressor****

In order to precisely locate the bounding box in the image., we used a scale-invariant linear regression model called bounding box regressor. For training this model we take as predicted and Ground truth pairs of four dimensions of localization. These dimensions are *(x, y, w, h)* where *x* and *y* are the pixel coordinates of the center of the bounding box respectively. w and h represent the width and height of bounding boxes. This method increases the Mean Average precision (mAP) of the result by *3-4%*.[](https://media.geeksforgeeks.org/wp-content/uploads/20200219162610/2020-02-171.jpg)**Output:**

Now we have region proposals that are classified for every class label. In order to deal with the extra bounding box generated by the above model in the image, we use an algorithm called [**Non- maximum suppression**](https://www.geeksforgeeks.org/tensorflow-js-tf-image-nonmaxsuppressionasync-function/)**.**It works in 3 steps:

* Discard those objects where the confidence score is less than a certain threshold value*( say 0.5)*.
* Select the region which has the highest probability among candidates regions for the object as the predicted region.
* In the final step, we discard those regions which have [IoU (intersection Over Union)](https://www.geeksforgeeks.org/calculation-intersection-over-union-iou-for-evaluating-an-image-segmentation-model-using-java/) with the predicted region over *0.5.*



After that, we can obtain output by plotting these bounding boxes on the input image and labeling objects that are present in bounding boxes.

#### ****Results****

The R-CNN gives a Mean Average Precision (mAPs) of *53.7%* on VOC 2010 dataset. On *200-class ILSVRC 2013* object detection dataset it gives an mAP of *31.4%* which is a large improvement from the previous best of *24.3%*. However, this architecture is very slow to train and takes *~ 49 sec* to generate test results on a single image of the VOC 2007 dataset.

## ****Challenges of R-CNN****

* The selective Search algorithm is very rigid and there is no learning happening in that. This sometimes leads to bad region proposal generation for object detection.
* Since there are approximately *2000* candidate proposals. It takes a lot of time to train the network. Also, we need to train multiple steps separately (CNN architecture, SVM model, bounding box regressor). So, This makes it very slow to implement.
* R-CNN can not be used in real-time because it takes approximately *50 sec* to test an image with a bounding box regressor.
* Since we need to save feature maps of all the region proposals. It also increases the amount of disk memory required during training.

R-CNN is an algorithm used for object detection in images. It stands for Region-based Convolutional Neural Network. The main goal of R-CNN is to identify and locate objects within an image.

### How R-CNN Works

1. **Input Image**:
   * You start with an image where you want to detect objects, like a picture of a street with cars and pedestrians.
2. **Region Proposals**:
   * The algorithm uses a method called Selective Search to generate many potential regions (proposals) where objects might be located. Think of it as making several guesses about where objects could be in the image.
3. **Feature Extraction**:
   * Each of these proposed regions is cropped and resized to a fixed size.
   * These regions are then passed through a Convolutional Neural Network (CNN), which converts the cropped images into feature vectors. Feature vectors are just lists of numbers that represent important characteristics of the regions.
4. **Classification and Localization**:
   * The feature vectors are fed into a classifier to determine what object is in each region (e.g., car, pedestrian, etc.).
   * At the same time, a regression model refines the exact position and size of the bounding box around the detected object.
5. **Output**:
   * The algorithm outputs the locations and categories of the objects detected in the image.

### Example

Let's say you have an image of a street, and you want to detect cars and pedestrians.

1. **Input Image**:
   * You start with a photo of a street scene.
2. **Region Proposals**:
   * The algorithm generates many region proposals (e.g., 2000 regions) suggesting where objects might be. These could be boxes around areas that might contain cars or pedestrians.
3. **Feature Extraction**:
   * Each proposed region is resized and passed through a CNN. The CNN converts each region into a feature vector that represents the region's important details.
4. **Classification and Localization**:
   * The feature vectors are analyzed to classify the regions (e.g., this region is a car, that region is a pedestrian).
   * The position and size of the bounding box around each detected object are refined to be more accurate.
5. **Output**:
   * The algorithm outputs the detected objects with their bounding boxes, like "car detected at this position" and "pedestrian detected at that position."

### Visual Summary

1. **Input Image**:
2. **Region Proposals**:
3. **Feature Extraction**:
4. **Classification and Localization**:
5. **Output**:

### Key Points

* **Region Proposals**: R-CNN generates many potential regions where objects might be.
* **Feature Extraction**: Each region is analyzed to understand its characteristics.
* **Classification**: The model determines what object is in each region.
* **Localization**: The exact position and size of each object are refined.

### Limitations of R-CNN

* **Slow**: The process of generating region proposals and running CNNs on each proposal is time-consuming.
* **Resource-Intensive**: Requires a lot of computational power and memory.

### Improvements (Fast R-CNN and Faster R-CNN)

* **Fast R-CNN**: Integrates region proposals and feature extraction, making the process faster.
* **Faster R-CNN**: Introduces the Region Proposal Network (RPN) to generate region proposals more efficiently within the CNN itself.

In summary, R-CNN is a powerful but initially slow method for detecting objects in images. It works by proposing regions, extracting features, classifying objects, and refining their positions.

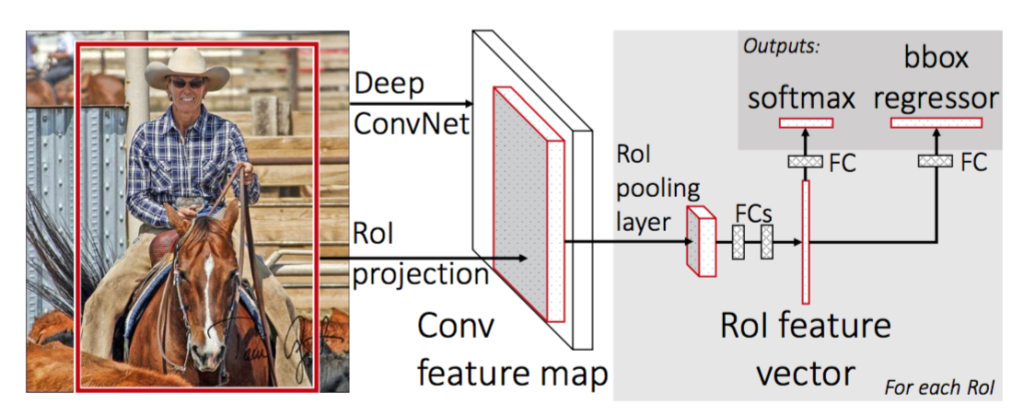
**FAST\_RCNN**

Fast R-CNN (Region-based Convolutional Neural Networks) is a method used for object detection, which aims to identify and localize objects within an image. It improves upon the original R-CNN by making the process more efficient and faster.

Before discussing Fast R-CNN, let’s look at the challenges faced by R-CNN.

* The training of R-CNN is very slow because each part of the model such as (CNN, SVM classifier, and bounding box) requires training separately and cannot be paralleled.
* Also, in R-CNN we need to forward and pass every region proposal through the Deep Convolution architecture (that’s up to *~2000* region proposals per image). That explains the amount of time taken to train this model
* The testing time of inference is also very high. It takes *49* seconds to test an image in R-CNN (along with selective search region proposal generation).

Fast R-CNN works to solve these problems. Let’s look at the architecture of Fast R-CNN.

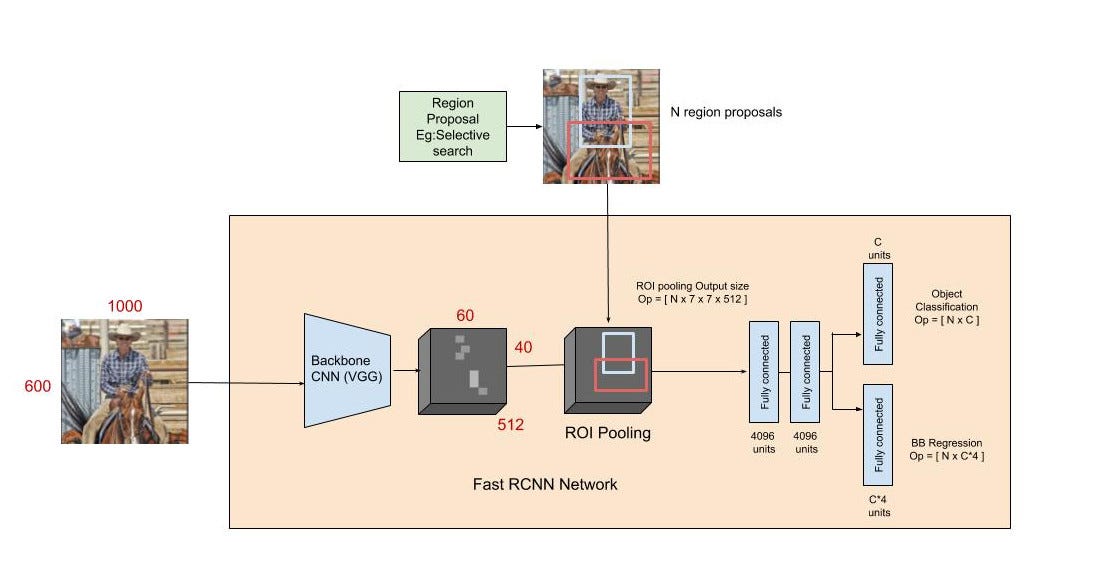
[](https://media.geeksforgeeks.org/wp-content/uploads/20200219160147/fast-RCNN1.png)

Here's a detailed explanation of how Fast R-CNN works:

### Overview

Fast R-CNN integrates region proposal generation and object detection into a single, unified network. It consists of four main steps:

1. **Input Image and Region Proposals**: The input image and a set of region proposals (usually generated by an external algorithm like Selective Search) are provided.
2. **Feature Extraction**: The entire image is passed through a state-of-art convolutional neural network (CNN) to produce a feature map.
3. **Region of Interest (RoI) Pooling**: The region proposals are mapped onto the feature map, and a fixed-size feature vector is extracted from each region using the RoI pooling layer.
4. **Classification and Regression**: The feature vectors are passed through fully connected layers to classify the objects and refine their bounding boxes.



### Detailed Steps

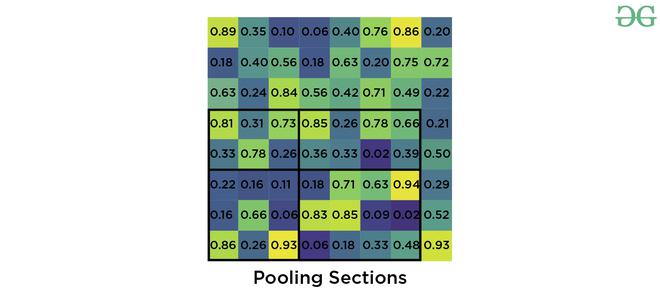
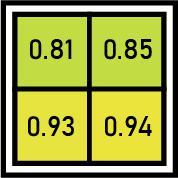
#### 1. Input Image and Region Proposals

* The input is an image along with several region proposals that indicate potential object locations.
* These proposals are typically rectangles that are generated by an external method (e.g., Selective Search).
* **Input Image**: We have an image of a street scene.
* **Region Proposals**: An algorithm like Selective Search generates region proposals, suggesting areas where objects might be located. For simplicity, let's say it generates four regions.

#### 2. Feature Extraction

* The entire image is fed into a deep CNN (e.g., VGG16 or ResNet).
* The CNN processes the image and outputs a convolutional feature map.
* The entire image is fed into a pre-trained Convolutional Neural Network (CNN) like VGG16 or ResNet.
* The CNN processes the image and outputs a convolutional feature map. This feature map captures essential features (e.g., edges, textures) from the image.

#### 3. Region of Interest (RoI) Pooling

* Each region proposal is mapped onto the feature map.
* RoI pooling layer extracts a fixed-size feature vector (e.g., 7x7) from each proposal. This is done by dividing the proposed region into a grid and then applying max pooling within each grid cell.
* This allows for handling proposals of different sizes and aspect ratios.
* The region proposals are projected onto the feature map.
* For each region proposal, RoI pooling extracts a fixed-size feature map (e.g., 7x7 grid). This involves dividing each region proposal into a grid and applying max pooling within each grid cell.
* As a result, all proposals are transformed into a fixed-size feature vector, regardless of their original size.
* Now if we need to convert this region proposal into a *2 x 2* output block and we know that the dimensions of the pooling section do not perfectly divisible by output dimension. We take pooling such that it is fixed into *2 x 2* dimensions.
* 
* Now we apply the max pooling operator to select the maximum value from each of the regions that we divided into.
* [](https://media.geeksforgeeks.org/wp-content/uploads/20200219155919/output19.jpg)
* *Max pooling output*

#### 4. Classification and Regression

* The fixed-size feature vectors are flattened and fed into fully connected layers.
* The network then performs two tasks:
  + **Classification**: Assigns a probability distribution over K+1K+1K+1 classes (including the background class).
  + **Bounding Box Regression**: Predicts refined bounding box coordinates for each of the object classes.
* Each vector goes through two branches:
  + **Classification Branch**: Outputs class probabilities for each region (e.g., car, pedestrian, background).
  + **Bounding Box Regression Branch**: Outputs refined bounding box coordinates for the detected objects.

### Loss Function

Fast R-CNN uses a multi-task loss function that combines classification loss and bounding box regression loss:

* **Classification Loss**: Typically a softmax loss that evaluates how well the predicted class probabilities match the true class labels.
* **Regression Loss**: Usually a smooth L1 loss that measures the difference between the predicted and true bounding box coordinates.

### Advantages Over R-CNN

* **Speed**: By sharing computation for the entire image instead of processing each region proposal independently, Fast R-CNN is much faster than the original R-CNN.
* **Memory Efficiency**: It reduces the need for redundant storage of feature maps for each proposal, making it more memory-efficient.

### Summarized Example

Given the input image of a street, the Fast R-CNN process can be summarized as follows:

1. **Region Proposals**: Generate several proposals, e.g., four regions where objects might be present.
2. **Feature Map Extraction**: The entire image passes through a CNN, generating a feature map.
3. **RoI Pooling**: Extract fixed-size feature maps (e.g., 7x7) for each region proposal from the feature map.
4. **Classification and Bounding Box Regression**:
   * For each proposal, classify the object (e.g., car, pedestrian, or background).
   * Refine the bounding box coordinates for a better fit around the object.

### Example Outputs

* **Region 1**: Classified as a car, bounding box adjusted to fit the car more precisely.
* **Region 2**: Classified as a pedestrian, bounding box adjusted.
* **Region 3**: Classified as background (no object).
* **Region 4**: Classified as a car, bounding box adjusted.

Fast R-CNN improves the efficiency and speed of object detection by sharing convolutional computations and using RoI pooling to handle proposals of varying sizes, making it a significant improvement over the original R-CNN approach.

### Summary

Fast R-CNN significantly speeds up the object detection process by integrating region proposal handling into the deep learning network, sharing convolutional computations, and using efficient RoI pooling. It represents a significant step forward from R-CNN by streamlining the detection pipeline and improving computational efficiency.

**ERROR Detection**

In object detection models like R-CNN (Region-based Convolutional Neural Network) and Fast R-CNN, evaluating the performance and accuracy is crucial. Several metrics and matrices are commonly used to measure their effectiveness. Here are the key evaluation metrics:

### Common Evaluation Metrics for Object Detection

1. **Intersection over Union (IoU)**
2. **Precision and Recall**
3. **Average Precision (AP)**
4. **Mean Average Precision (mAP)**
5. **Confusion Matrix (specific to classification within detected regions)**

### Detailed Explanation of Each Metric

#### 1. Intersection over Union (IoU)

**Definition**: IoU measures the overlap between the predicted bounding box and the ground truth bounding box.

**Formula**: IoU=Area of OverlapArea of Union\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}IoU=Area of UnionArea of Overlap​

**Usage**:

* IoU is used to determine whether a predicted bounding box is a true positive. Typically, a threshold (e.g., IoU > 0.5) is set to decide this.

#### 2. Precision and Recall

**Definitions**:

* **Precision**: The fraction of true positive detections among all positive detections (i.e., how many detected objects are actually relevant).
* **Recall**: The fraction of true positive detections among all ground truth objects (i.e., how many relevant objects are detected).

**Formulas**: Precision=True PositivesTrue Positives+False Positives\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}Precision=True Positives+False PositivesTrue Positives​ Recall=True PositivesTrue Positives+False Negatives\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}Recall=True Positives+False NegativesTrue Positives​

**Usage**:

* Precision and recall help evaluate the quality of the object detection by assessing the accuracy and completeness of the detections.

#### 3. Average Precision (AP)

**Definition**: AP is the average of the precision values at different recall levels. It provides a single metric that summarizes the precision-recall curve.

**Formula**: AP=∫01p(r) dr\text{AP} = \int\_0^1 p(r) \, drAP=∫01​p(r)dr where p(r)p(r)p(r) is the precision as a function of recall rrr.

**Usage**:

* AP is used to measure the precision-recall trade-off for a particular object class.

#### 4. Mean Average Precision (mAP)

**Definition**: mAP is the mean of the Average Precision values across all classes.

**Formula**: mAP=1N∑i=1NAPi\text{mAP} = \frac{1}{N} \sum\_{i=1}^{N} AP\_imAP=N1​∑i=1N​APi​ where NNN is the number of classes and APiAP\_iAPi​ is the Average Precision for class iii.

**Usage**:

* mAP is a comprehensive metric that evaluates the overall performance of the object detection model across all object classes.

#### 5. Confusion Matrix

For the classification component within the detected regions, a confusion matrix can be used to assess how well the model classifies the detected objects into correct categories.

**Components of Confusion Matrix**:

* **True Positives (TP)**: Correctly classified objects.
* **True Negatives (TN)**: Correctly classified as not the object.
* **False Positives (FP)**: Incorrectly classified as the object.
* **False Negatives (FN)**: Correctly classified objects missed by the model.

**Matrix Representation**:

mathematica

Copy code

Predicted Positive Predicted Negative

Actual Positive TP FN

Actual Negative FP TN

**Derived Metrics**:

* **Accuracy**: TP+TNTP+TN+FP+FN\frac{TP + TN}{TP + TN + FP + FN}TP+TN+FP+FNTP+TN​
* **Precision**: TPTP+FP\frac{TP}{TP + FP}TP+FPTP​
* **Recall**: TPTP+FN\frac{TP}{TP + FN}TP+FNTP​
* **F1 Score**: 2×Precision×RecallPrecision+Recall2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}2×Precision+RecallPrecision×Recall​

### Summary

* **IoU**: Evaluates the overlap between predicted and ground truth bounding boxes.
* **Precision and Recall**: Measure the accuracy and completeness of the detections.
* **AP and mAP**: Provide single-value metrics summarizing the precision-recall trade-off for individual classes and overall model performance.
* **Confusion Matrix**: Assesses the classification accuracy within the detected regions.

These metrics collectively ensure a thorough evaluation of object detection models like R-CNN and Fast R-CNN, covering both detection accuracy and classification performance.

FASTER RCNN

### Faster R-CNN in Simple Terms

**Objective**: To detect objects in an image quickly and accurately.

### How Faster R-CNN Works

1. **Input Image**:
   * You start with an image that you want to analyze, like a picture of a street with cars and pedestrians.
2. **Feature Extraction**:
   * The entire image is passed through a Convolutional Neural Network (CNN), which converts the image into a feature map. This feature map highlights important features like edges, textures, and shapes.
3. **Region Proposal Network (RPN)**:
   * Instead of using an external algorithm to generate region proposals (like in Fast R-CNN), Faster R-CNN has an internal Region Proposal Network (RPN).
   * The RPN looks at the feature map and suggests regions where objects might be located. For example, it might suggest 300 regions instead of the 2000 generated by Fast R-CNN's external method.
4. **RoI Pooling**:
   * Each region proposal from the RPN is mapped onto the feature map.
   * Using RoI (Region of Interest) Pooling, the regions are resized to a fixed size (e.g., 7x7 grid), regardless of their original size.
5. **Classification and Bounding Box Regression**:
   * Each resized region is analyzed to classify what object it contains (e.g., car, pedestrian) and to refine the exact position of the bounding box around the object.

### Summary

1. **Input Image**: An image is processed.
2. **Feature Extraction**: CNN processes the image into a feature map.
3. **Region Proposal Network (RPN)**: Proposes regions directly from the feature map.
4. **RoI Pooling**: Regions are resized and mapped onto the feature map.
5. **Classification and Bounding Box Regression**: Regions are classified and refined.

### Key Differences from Previous Models

1. **Integrated Region Proposal**:
   * Faster R-CNN combines the region proposal step into the neural network itself, making it faster and more efficient than models like Fast R-CNN, which relied on external algorithms for region proposals.
2. **Speed**:
   * By generating region proposals internally, Faster R-CNN significantly reduces the time needed to analyze an image.

### Visualizing the Process

Imagine you have a picture of a street scene:

* **Step 1**: The CNN processes the entire image and creates a feature map, highlighting various important features.
* **Step 2**: The RPN scans this feature map and suggests regions where objects might be (e.g., a few rectangles around potential cars and pedestrians).
* **Step 3**: These regions are resized to a uniform size and then further analyzed.
* **Step 4**: The model classifies each region (e.g., "car", "pedestrian") and adjusts the bounding box to fit the object more precisely.

By integrating the region proposal network into the same system that classifies and refines these proposals, Faster R-CNN achieves faster and more accurate object detection.

### Faster R-CNN in Simple Terms with Example

**Objective**: To detect objects in an image quickly and accurately.

### How Faster R-CNN Works

1. **Input Image**:
   * Imagine you have a picture of a street with cars and pedestrians.
2. **Feature Extraction**:
   * The entire image is passed through a Convolutional Neural Network (CNN), which converts the image into a feature map. This feature map highlights important features like edges, textures, and shapes.
   * **Example**: The CNN might detect the shapes of the car roofs, the outlines of the people, and the edges of the buildings.
3. **Region Proposal Network (RPN)**:
   * Instead of using an external algorithm to generate region proposals (like in Fast R-CNN), Faster R-CNN has an internal Region Proposal Network (RPN).
   * The RPN looks at the feature map and suggests regions where objects might be located.
   * **Example**: The RPN might suggest 300 regions, highlighting areas that likely contain cars and pedestrians, such as a rectangle around a car on the left and a rectangle around a pedestrian on the right.
4. **RoI Pooling**:
   * Each region proposal from the RPN is mapped onto the feature map.
   * Using RoI (Region of Interest) Pooling, the regions are resized to a fixed size (e.g., 7x7 grid), regardless of their original size.
   * **Example**: If the original region is a rectangle around a car that’s 20x30 pixels, RoI Pooling will resize this to a standard 7x7 grid for uniform processing.
5. **Classification and Bounding Box Regression**:
   * Each resized region is analyzed to classify what object it contains (e.g., car, pedestrian) and to refine the exact position of the bounding box around the object.
   * **Example**: The region around the car is classified as "car" and the bounding box is adjusted to fit the car more precisely. Similarly, the region around the pedestrian is classified as "pedestrian" and the bounding box is refined.

### Summary

1. **Input Image**: An image of a street scene with cars and pedestrians.
2. **Feature Extraction**: CNN processes the image into a feature map, highlighting edges and shapes of objects.
3. **Region Proposal Network (RPN)**: Proposes regions directly from the feature map, such as rectangles around cars and pedestrians.
4. **RoI Pooling**: Regions are resized to a uniform size (e.g., 7x7 grid) for further processing.
5. **Classification and Bounding Box Regression**: Each region is classified (e.g., "car", "pedestrian") and the bounding boxes are refined for accuracy.

### Key Differences from Previous Models

1. **Integrated Region Proposal**:
   * Faster R-CNN combines the region proposal step into the neural network itself, making it faster and more efficient than models like Fast R-CNN, which relied on external algorithms for region proposals.
2. **Speed**:
   * By generating region proposals internally, Faster R-CNN significantly reduces the time needed to analyze an image.

### Visualizing the Process

Imagine you have a picture of a busy street:

* **Step 1**: The CNN processes the entire image and creates a feature map, highlighting important features such as the edges of cars and people.
* **Step 2**: The RPN scans this feature map and suggests regions where objects might be, like rectangles around potential cars and pedestrians.
* **Step 3**: These regions are resized to a uniform size and then further analyzed by the network.
* **Step 4**: The model classifies each region (e.g., "car", "pedestrian") and adjusts the bounding box to fit the object more precisely.

By integrating the region proposal network into the same system that classifies and refines these proposals, Faster R-CNN achieves faster and more accurate object detection.

### Example Scenario

Let's say you have an image like this:

* **Input**: This image is fed into the Faster R-CNN model.
* **Feature Extraction**: The CNN detects features like the car outlines, pedestrian shapes, and building edges.
* **RPN**: The network suggests possible regions for objects, such as a rectangle around a car and another around a pedestrian.
* **RoI Pooling**: These regions are resized uniformly.
* **Classification and Bounding Box Regression**: The model identifies the regions correctly as "car" and "pedestrian," and refines the bounding boxes to fit the objects accurately.

This step-by-step approach ensures that Faster R-CNN can quickly and accurately detect objects within images, making it highly effective for real-world applications like autonomous driving, security surveillance, and more.

YOLO