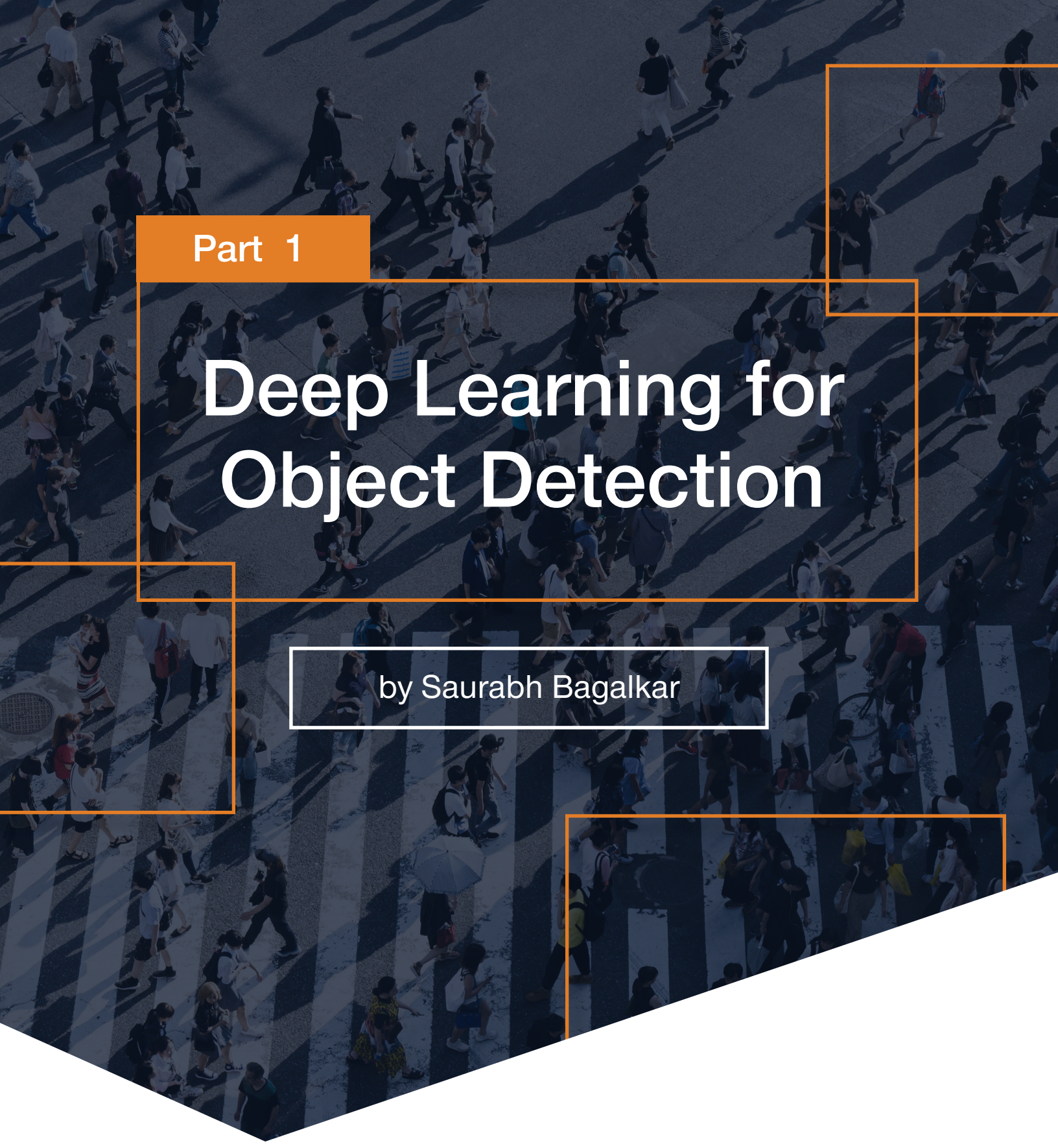


Part 1

Deep Learning for Object Detection

by Saurabh Bagalkar

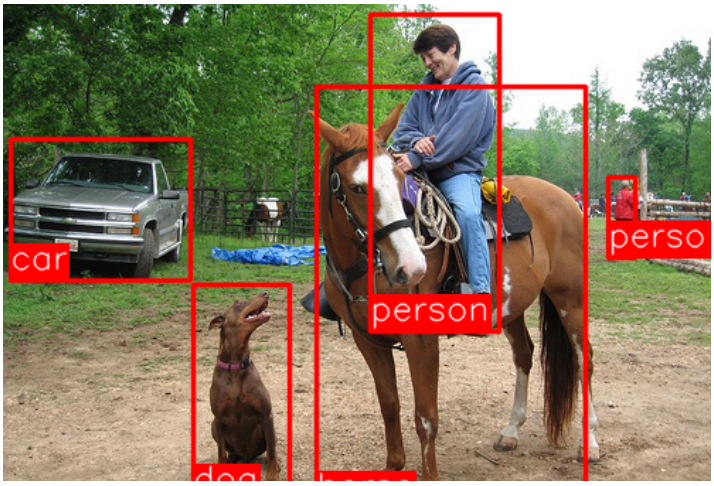
ALEGION™

Introduction

Object detection is at the forefront of computer vision research, primarily due to its plethora of practical applications in almost every field. With technology progressing so fast, it is important to develop a deep understanding of what these methods are and how they have evolved over time as we innovate beyond them. This is the first piece in a new series that will cover the concepts behind some very powerful state-of-the-art object detection techniques and localization methods that use region based object detectors. **Scene understanding**, of which object detection is a major part, helps to automate a lot of computer vision tasks, such as pedestrian detection, vehicle detection and localization, and traffic sign recognition among others. We will start with the R-CNN proposed by **Ross Girshick et al**. which is a precursor for understanding the more efficient and widely used successors like Fast R-CNN and Faster R-CNN and eventually Mask R-CNN which will be covered later in this series.

What is R-CNN?

R-CNN is an object detection and localization method. It aims to “find” objects of interest in an image and draws bounding boxes around them while also categorizing its class. The name R-CNN stands for Regions with CNN (Convolutional Neural Networks) features.

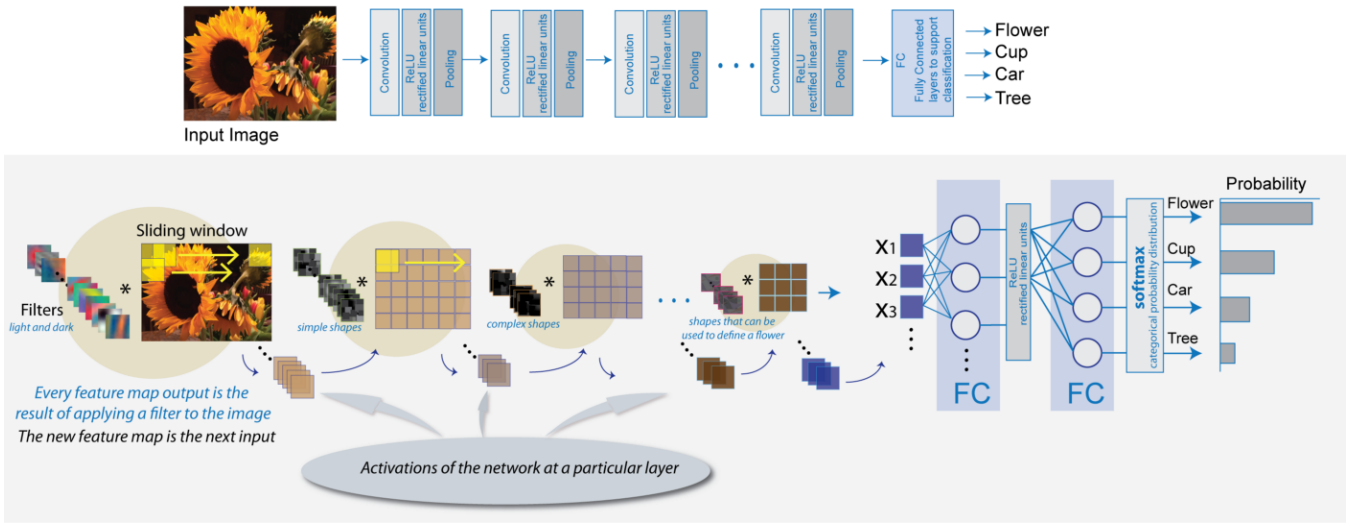


R-CNN attempts to categorize each object in an image with its associated bounding box. **Image source**

Motivation Behind R-CNN

Object Recognition has been traditionally done using **SIFT** and **HOG** features. These methods, although very successful, breakdown when there are images that have other objects, or considerable noise around them. As we know, recognition in human visual cortex system is much more complicated than just identifying gradients and clustering them, which HOG tends to do. We need a more sophisticated, hierarchical and multistage process if we are going to get to human level visual recognition.

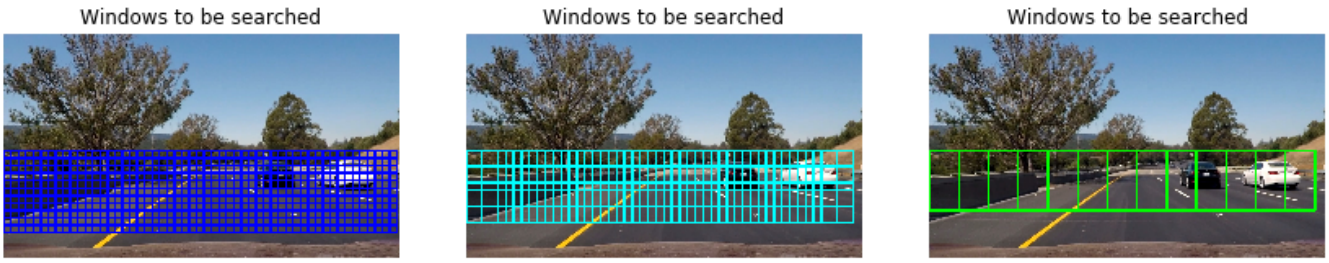
With the advent of Deep-Learning and specifically CNN's, we have a good starting point to extract features from an image using a similar hierarchical architecture. As **AlexNet** outperformed every other image recognition model out there in 2012, because of its clever use of dropouts and ReLU activations, it quickly became the de facto standard for image recognition. Then the burning question became, *can we somehow extend CNN classification results from ImageNet challenge to object detection results on PASCAL VOC challenge?* This idea gave rise to R-CNN or region proposal using CNN, which bridges the gap between image classification and object detection.



CNN's classify images by extracting their high and low level features. **Image source**

Why is Object Detection and Localization a non-trivial task?

Localizing an object can be framed as a regression problem by itself but that is not terribly effective because it yields poor results and is very slow. Another approach is to use a **sliding-window detector**, traditional for CNN's, but as the layers and subsequent strides increase, resolution is lost. This makes precise localization challenging, not to mention the fact that it is too slow for many real world applications. Another issue with using the sliding-window approach is that if you need to capture different aspect ratios of different regions within an image, spatial location will vary greatly. This will also blow up the computational power.



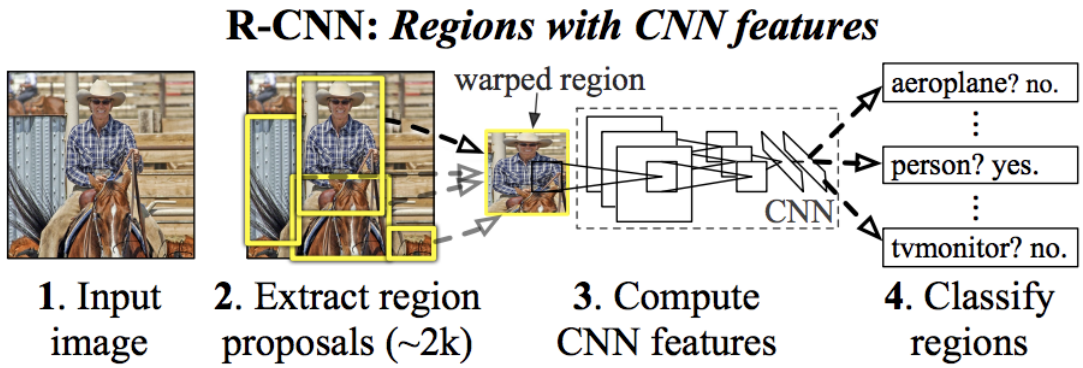
The sliding window approach is very cumbersome, when used to localize cars in an image. **Image source**

R-CNN building blocks and architecture

To deal with the aforementioned localization issues that slow down not just the detection speed but also the accuracy, R-CNN's were introduced which use a very clever approach to detection and localization.

R-CNN has the following main modules:

1. **Region Proposals** - For generating 2000 proposed bounding boxes
2. **Feature extraction using CNN** - For image classification of each region
3. **Classification and Localization**
    a. Class recognition using **Linear SVM**
    b. Bounding-box regression for localization



R-CNN modules. **Image source**

Let's look at each module in detail to understand how the architectural components interact with each other to give us the desired end result.

Region Proposals

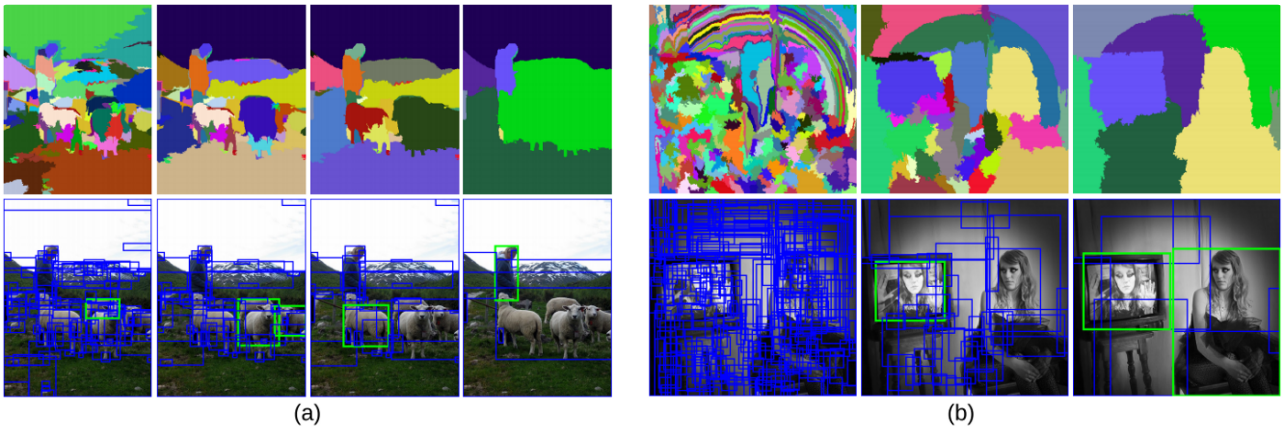


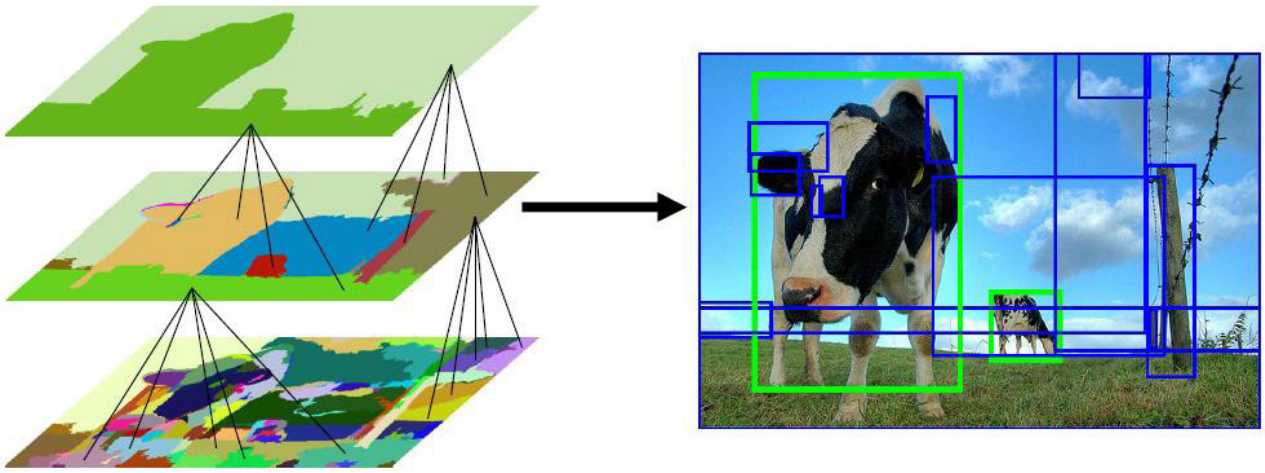
Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.

A clever way to select coherent regions within an image. **Image source**

A smarter way to localize an object is to generate region proposals using the *concept of recognition using regions.* There are many ways of generating region proposals, but we will do it using the Selective Search method which, in a nutshell, clusters or groups similar regions and produces demarcated regions within an image. It does so using an iterative approach somewhat similar to **region growing** until no further change is possible.

Selective Search uses these main steps to propose regions:

1. **From bottom up, finding similarities between pixels in an image based on**
   - Color
   - Texture
   - Size
   - Shape Compatibility
2. **Groups the pixels together, which give a high similarity metric using a weighted sum of aforementioned similarities**
3. **Repeats until there are no more changes**



Selective Search uses a bottom up approach to propose similar regions. **Image source**

At test time, R-CNN's region proposal method using selective search generates 2000 category-independent region proposals. These region proposals will be filtered along the way to only account for the "best" region proposals. So the main task of this module is to propose a set of candidate regions for the next module i.e CNN feature extractor. The output of this module, which is a list of coordinates {x-coord, y-coord, width, height} is then fed to the feature extraction module. For a more in-depth understanding of how Selective Search works, please refer to this **link**.

Feature Extraction using CNN

Since the popularity of AlexNet proposed by **Krizhevsky** et al, CNN's have become hugely popular for feature extraction from images. This module extracts a 4096-dimensional feature vector for each region proposed by region proposals module.

The overarching steps for feature extraction are as follows:

Resize and warp the RGB image to 227 x 227 and perform mean subtraction as part of preprocessing. Before warping, there is some dilation of bounding box to add some context padding, which has shown better **mAP** results.

Pass, i.e. forward propagate the individual resized regions through pre-trained AlexNet to get a 4096-dimensional feature vector of each region with each image. Finally a 1000 length classification layer is replaced by (N+1) length layer where N is number of classes and an extra one is for the background.

All region proposals with IOU ≥ 0.5 overlap with a ground-truth box as it is considered positive for that box's class, while the rest are considered negatives. Essentially, the output for this module will be a 2000 x 4096 vector for each image.

Classification and Localization

Object classification is performed using a linear SVM model

Once we have a 4096 dimensional vector for each candidate object in each image, we can run a trained Linear SVM model to categorize that object in one of the many classes.

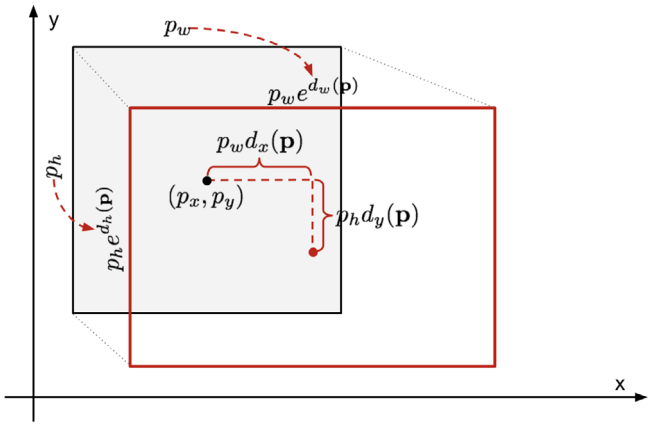
During training of SVM, a hard negative mining approach is used to compensate for training data that is too large to fit in memory. Hard negative mining makes sure:

We have an approximately equal number of positive and negative examples. How? We only look for “hard” negatives and not all negatives. All negatives i.e the background, are in abundance in an image, whereas “hard” negatives comprise of objects which are occluded or difficult to classify.

The classifier learns during training which objects to focus on, as we explicitly capture a false positive and tell the classifier to consider it as a negative example for the next epoch of the training cycle, thus increasing the robustness of the classifier.

In contrast to fine-tuning using CNN’S, where we use **IOU** of at least 0.5, for training SVM’s we only take ground truth boxes as positive examples, and label proposals with less than 0.3 IOU as negatives, ignoring everything else. This approach while training has shown to improve the mAP score.

Bounding-box regression is used to precisely localize an object in an image



Transformation between predicted and ground truth coordinates. **Image source**

Selective search gives us candidate blobs around each object of interest along with its 4 coordinates. These bounding boxes might not be as tight as desired. So to resolve this issue, a function with learnable parameters P is used to map and transform the proposed box coordinates

$$P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$$

to the ground truth coordinates G

$$G = (G_x, G_y, G_w, G_h)$$

by imposing scale-invariant, between two centers, and log-space, on the width and height, transformations as shown here

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w e^{d_w(P)}$$
$$\hat{G}_h = P_h e^{d_h(P)}$$

The regression targets t is defined as:

$$t_x = \frac{(G_x - P_x)}{P_w}$$
$$t_y = \frac{(G_y - P_y)}{P_h}$$
$$t_w = log(\frac{G_w}{P_w})$$
$$t_h = log(\frac{G_h}{P_h})$$

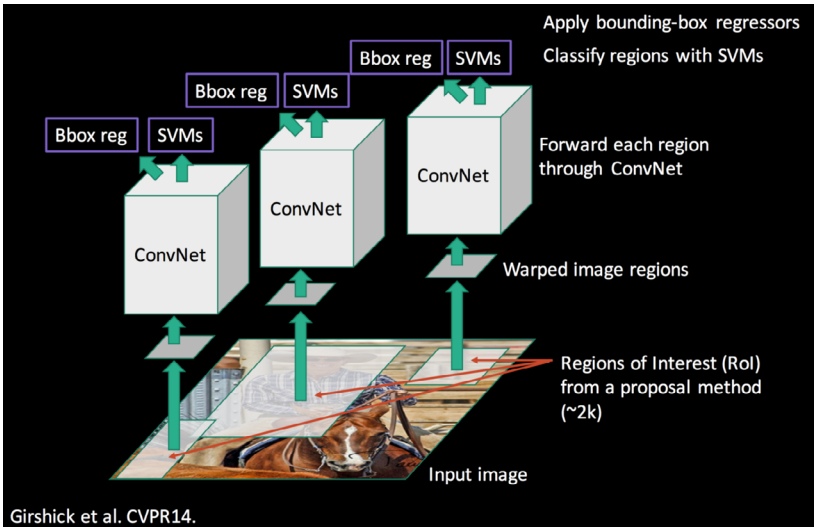
The standard regression model can solve this optimization problem using least squares objective

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t^i_\star - \hat{\mathbf{w}}^{\mathrm{T}}_\star \boldsymbol{\phi}_5(P^i))^2 + \lambda \left\| \hat{\mathbf{w}}_\star \right\|^2$$

where w is a vector of learnable model parameters, which needs to be optimized. Please refer to the **original paper** to understand the in-depth specifics of these transformations and mapping.

Recap of R-CNN process

In a nutshell, R-CNN does the following:



The R-CNN process

1. Generates around 2000 class-independent region proposals for each image using Selective Search approach. These object proposals are different sizes, so they are warped into a standard 227 x 227 image, which is then fed to a pre-trained CNN architecture like AlexNet.

2. The CNN extracts a 4096 dimensional vector of each object in an image thus creating a 2000 x 4096 matrix.

3. This feature vector is passed onto a binary SVM, which is trained for each class separately, thus giving the classification of the object.

4. Precise localization is achieved using a least squares regression model, which minimizes the differences between the proposed coordinates and the ground truth coordinates by correcting the offset values.

Why R-CNN needs improvement?

Although, R-CNN was a breakthrough paper in its time, it faced several drawbacks:

1. **It is computationally very costly to implement**
   - *For every region in each image, we have to calculate features using a forward pass in CNN architecture, which means there will be 2000 forward passes for each image. It takes 47 secs per image on a GPU.*

2. **Selective search is not a learnable model and hence can generate bad initial candidates.**

3. **There are 3 different models to use, which adds architectural and time complexity.**
   - *CNN architecture for feature extraction, the SVM to recognize the object class, and learnable parameters for bounding box regressor, all of which are comprised of trainable parameters.*

To ameliorate these pitfalls of R-CNN, the authors improved the design of R-CNN with a new revamped approach called Fast R-CNN. We will take a look at it in our next piece.