# INTERNSHIP REPORT

# OBJECT DETECTION IN SATELLITE IMAGE USING DEEP LEARNING

A Project report submitted in partial fulfillment of the requirements for the award of the Degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

**Submitted by**

G Raghul(312420205072)

S Saraniya (3124202025086)

A Pandi (312420205068)

**Under the guidance of**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**ST. JOSEPH'S INSTITUTE OF TECHNOLOGY**

(**An Anna University Affiliated College**, **Approved by AICTE)**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**ST. JOSEPH'S INSTITUTE OF TECHNOLOGY**

(**An Anna University Affiliated College**, **Approved by AICTE**)

## CERTIFICATE

This is to certify that the project report entitled **"REAL TIME OBJECT DETECTION INSATELLITE IMAGE USING DEEP LEARNING**" submitted by **G. Raghul (312420205072) S. Saraniya (312420205086) A. Pandi (312420205068)** in partial Fulfillment of the requirements for the award of the **Degree of Bachelor of Technology in Information Technology** at **AP3 Solutions.**

**Industry Mentor**

Dr.Ajantha Devi,
Research Head, AP3 Solutions,
Chennai.

**Department Internship Coordinator**

Mrs.G.Subhashini
Assistant Professor
Specialization in Cloud Computing
and Machine Learning

**Head of the Department**

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our project guide Dr. Ajantha Devi and Mrs.G.Subhashini for her guidance with unsurpassed knowledge and immense encouragement.

We are grateful to Dr.S.Kalarani, Head of the Department, INFORMATION TECHNOLOGY, for providing us with the required facilities for the completion of the project work. We are very much thankful to the Principal and Management, St. Joseph's Institute of Technology, fortheir encouragement and cooperation to carry out this work

We express our thanks to all teaching faculty of Department of Information Technology, whose suggestionsduring reviews helped us in accomplishment of our project. We would like to thank all nonteaching staff of the Department of Information Technology, St. Joseph's Institute of Technology for providing great assistance in accomplishment of our project.

**PROJECT STUDENTS**

RAGHUL G(312420205072)

SARANIYA S(31240205086)

PANDI A(312420205068)

Table of Contents

**ABSTRACT:**

Computer Vision is the branch of the science of computers and software systems which can recognize as well as understand images and scenes. Computer Vision consists of various aspects such as image recognition, object detection, image generation, image super-resolution and many more. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars. In this project, we are using highly accurate object detection-algorithms such as CNN(Convolution Neural Network) with RPN(Regional Proposal Networks) formally known as Faster R-CNN.

**INTRODUCTION:**

A few years ago, the creation of the software and hardware image processing systems was mainly limited to the development of the user interface, which most of the programmers of each firm were engaged in. The situation has been significantly changed with the advent of the Windows operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analyzing remote and medical images, etc. Each of these "eternal" problems is solved by trial and error by the efforts of numerous groups of the engineers and scientists .As modern technical solutions are turn out to be excessively expensive, the task of automating the creation of the software tools for solving intellectual problems is formulated and intensively solved abroad. In the field of image processing, the required tool kit should be supporting the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers. Just as the Windows toolkit supports the creation of interfaces for solving various applied problems.

So, we can distinguish between these three computer vision tasks with this example:

Image Classification: This is done by Predict the type or class of an object in an image. Input: An image which consists of a single object, such as a photograph. Output: A class label (e.g. one or more integers that are mapped to class labels). Object Localization: This is done through, Locate the presence of objects in an image and indicate their location with a bounding box. Input: An image which consists of one or more objects, such as a photograph. Output: One or more bounding boxes (e.g.

defined by a point, width, and height). Object Detection: This is done through, Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

Input: An image which consists of one or more objects, such as a photograph. Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box. One of the further extension to this breakdown of computer vision tasks is object segmentation, also called "object instance segmentation" or "semantic segmentation," where instances of recognized objects are indicated by highlighting the specific pixels of the object instead of a coarse bounding box .From this breakdown, we can understand that object recognition refers to a suite of challenging computer vision tasks. For example, image classification is simply straight forward, but the differences between object localization and object detection can be confusing, especially when all three tasks may be just a sequally referred to as object recognition.

Humans can detect and identify objects present in an image. The human visual system is fast and accurate and can also perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. The availability of large sets of data, faster GPUs, and better algorithms ,we can now easily train computers to detect and classify multiple objects within an image with high accuracy. We need to understand terms such as object detection, object localization, loss function for object detection and localization, and finally explore an object detection algorithm known as " You only look once" (YOLO). Image classification also involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is always more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all these problems are referred to as object recognition. Object recognition refers to a collection of related tasks for identifying objects in digital photographs .Region-based Convolutional Neural Networks, or R-CNNs, is a family of techniques for addressing object localization and recognition tasks, designed for model performance. You Only Look Once ,or YOLO is known as the second family of techniques for object recognition designed for speed and real-time use.

**DIGITAL IMAGE PROCESSING :**

Computerized picture preparing is a range portrayed by the requirement for broad test work to build up the practicality of proposed answers for a given issue. A critical trademark hidden the plan of picture preparing frameworks is the huge level of testing and experimentation that Typically is required before touching base at a satisfactory

arrangement. This trademark infers that the capacity to plan approaches and rapidly model hopeful arrangements by and large assumes a noteworthy part in diminishing the cost and time required to land at a suitable framework execution.

**WHAT IS DIP?**

A picture might be characterized as a two-dimensional capacity f(x, y), where x, y are spatial directions, and the adequacy off at any combine of directions (x, y) is known as the power or dark level of the picture by then. Whenever x, y and the abundance estimation of are all limited discrete amounts, we call the picture a computerized picture. The field of DIP alludes to preparing advanced picture by methods for computerized PC. Advanced picture is made out of a limited number of components, each of which has a specific area and esteem. The components are called pixels. Vision is the most progressive of our sensor, so it is not amazing that picture play the absolute most imperative part in human observation. Be that as it may, dissimilar to people, who are constrained to the visual band of the EM range imaging machines cover practically the whole EM range, going from gamma to radio waves. They can work likewise on pictures produced by sources that people are not acclimated to partner with picture. There is no broad understanding among creators in regards to where picture handling stops and other related territories, for example, picture examination and PC vision begin. Now and then a qualification is made by characterizing picture handling as a teach in which both the info and yield at a procedure are pictures. This is constraining and to some degree manufactured limit. The range of picture investigation is in the middle of picture preparing and PC vision. There are no obvious limits in the continuum from picture handling toward one side to finish vision at the other. In any case, one helpful worldview is to consider three sorts of mechanized procedures in this continuum: low, mid and abnormal state forms. Low-level process includes primitive operations, for example, picture preparing to decrease commotion differentiate upgrade and picture honing. A low level process is described by the way that both its sources of info and yields are pictures. Mid-level process on pictures includes assignments, for example, division, depiction of that 11 Question diminish them to a frame reasonable for PC handling and characterization of individual articles A mid-level process is portrayed by the way that its sources of info by and large are pictures however its yields are properties removed from those pictures. At long last more clevated amount handling includes "Understanding an outlet of perceived items, as in picture examination and at the farthest end of the continuum playing out the intellectual capacities typically connected with human vision. Advanced picture handling, as effectively characterized is utilized effectively in a wide scope of regions of outstanding social and monetary esteem.

**Image Processing:**

Since the digital image is invisible, it must be prepared for viewing on one or more output device(laser printer, monitor at).The digital image can be optimized for the application by enhancing the appearance of the structures within it. There are three of image processing used. They are

1. Image to Image transformation
2. Image to Information transformations
3. Information to Image transformations

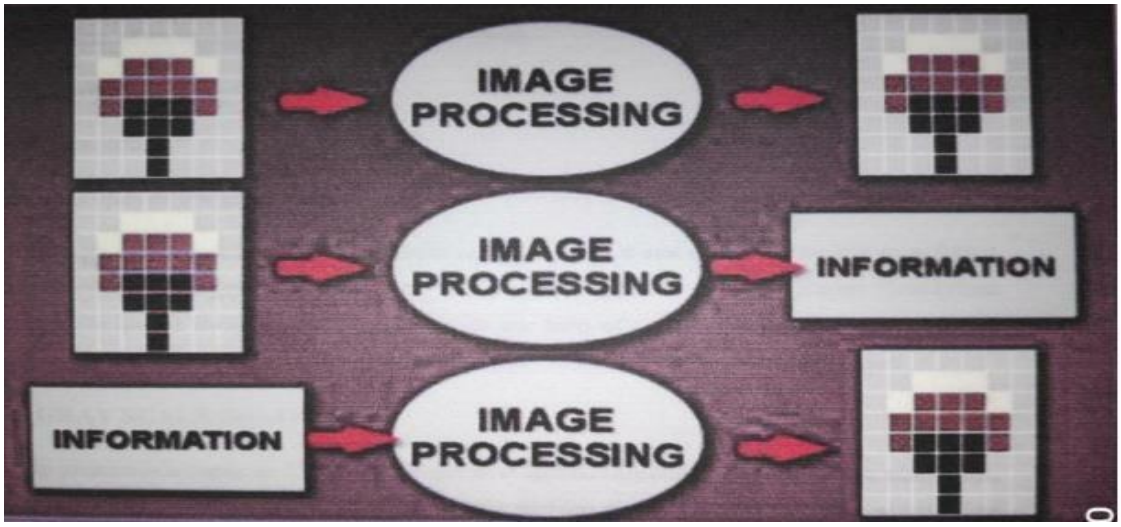



**FIGURE 1:Types of Image Processing**

**Background :**

The aim of object detection is to detect all instances of objects from a known class, such as people , cars or faces in an image. Generally, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each detection of the image is reported with some form of pose information. This is as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In some other situations, the pose information is more detailed and contains the parameters of a linear or non-linear transformation. For example for face detection in a face detector may compute the

locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of a bicycle detection in an image that specifies the locations of certain parts is shown in Figure 1. The pose can also be defined by a three-dimensional transformation specifying the location of the object relative to the camera. Object detection systems always construct a model for an object class from a set of training examples.In the case of a fixed rigid object in an image, only one example may be needed, but more generally multi training examples are necessary to capture certain aspects of class variability.

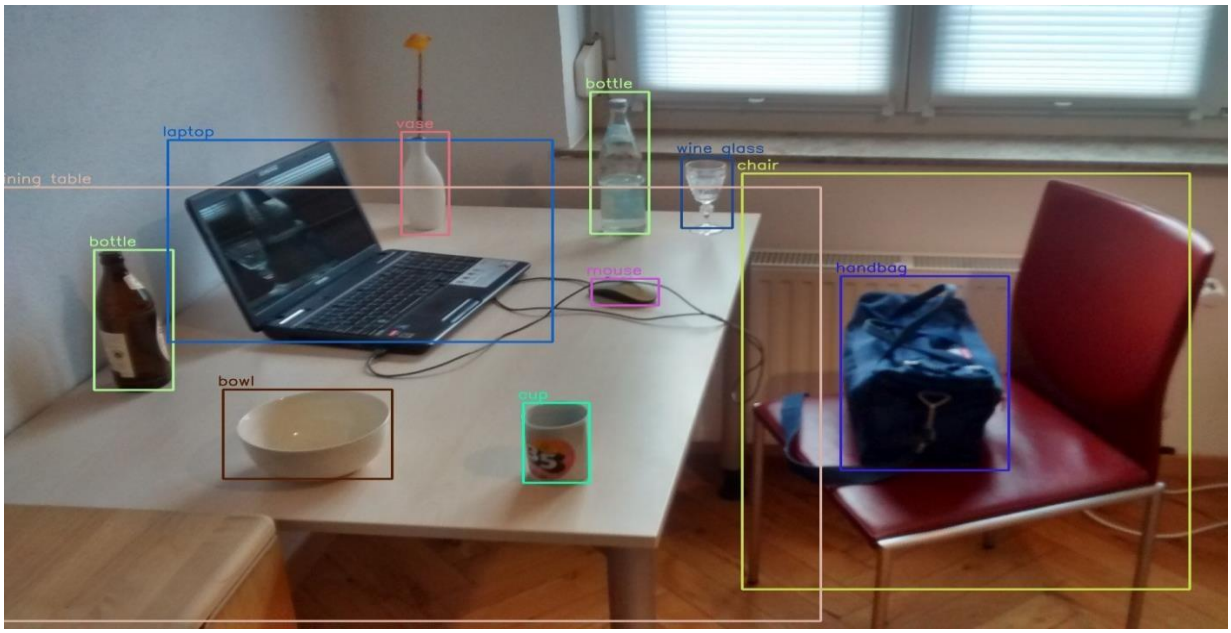



Figure 2:Object Detection

Convolutional implementation of the sliding windows Before we discuss the implementation of the sliding window using convents, let us analyze how we can convert the fully connected of the network into convolutional layer . Fig. 2 shows a simple convolutional network with two fully connected layers each of shape .

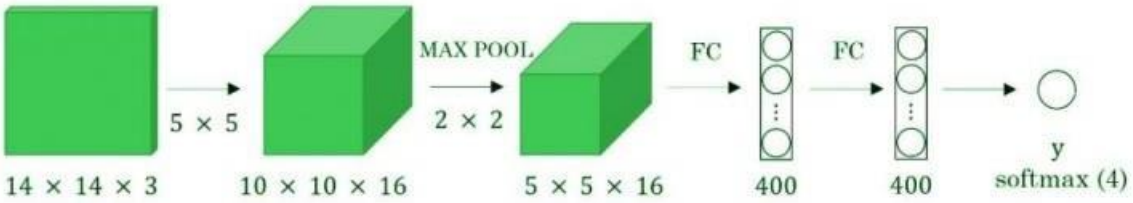



Figure 3:simple convolution network

A fully connected layer can be converted to a convolutional layer with the help of a 1Dconvolutionallayer. The width and height of this layer is equal to

one and the number of filters are equal to the shape of the fully connected layer. An example of this is shown in Fig 3.

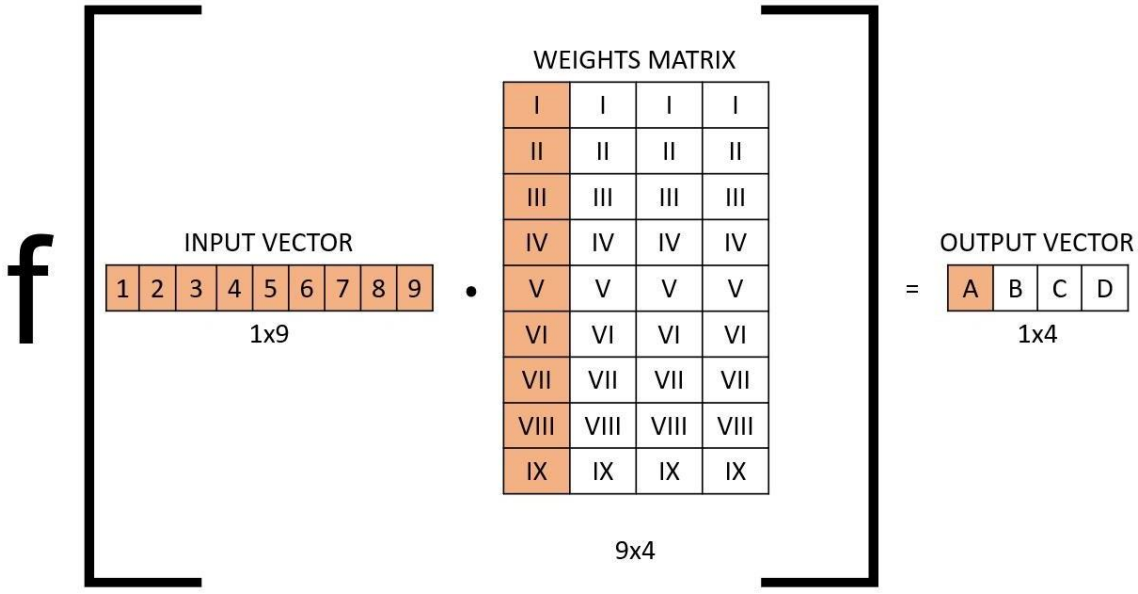



Figure 4

We can apply the concept of conversion of a fully connected layer into a convolutional layer to the model by replacing the fully connected layer with a 1-D convolutional layer. The number of filters of the 1D convolutional layer is equal to the shape of the fully connected layer. This representation is shown in Fig 4. Also, the output softmax layer is also a convolutional layer of shape (1, 1, 4), where4is the number of classes to predict.

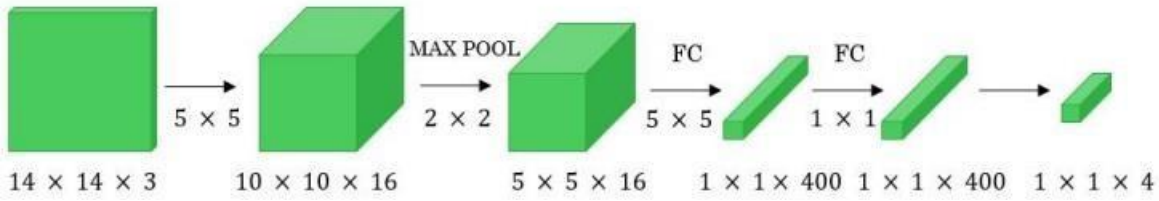



Figure 5

Now, let's extend the above approach to implement a convolutional version of the sliding window.First, let us consider the ConvNet that we have trained to be in the following representation(no fully connected layers).

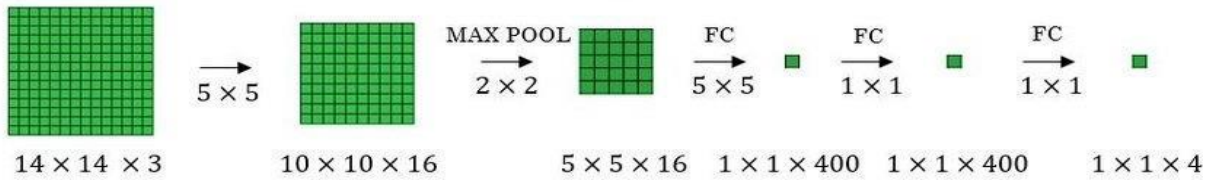



Figure 6

Let's assume the size of the input image to be 16 × 16 × 3. If we are using the sliding window approach, then we would have passed this image to the above ConvNet four times, where each time the sliding window crops the part of the input image matrix of size 14 × 14 × 3 and pass it through the ConvNet. But instead of this, we feed the full image (with shape 16 × 16 × 3) directly in to the trained ConvNet (see Fig. 6). This results will give an output matrix of shape 2 × 2 ×4. Each cell in the output matrix represents the result of the possible crop and the classified value of the cropped image. For example, the left cell of the output matrix(the green one) in Fig. 6 represents the result of the first sliding window. The other cells in the matrix represent the results of the remaining sliding window operations.

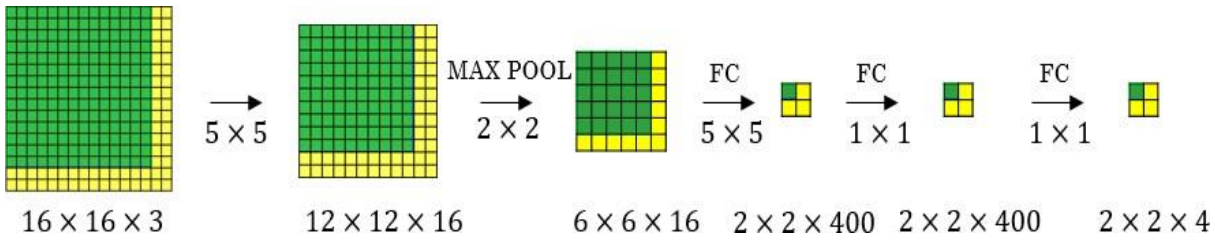



Figure 6

The stride of the sliding window is decided by the number of filters used in the Max Pool layer. In the example above, the Max Pool layer has two filters, and for the result, the sliding window moves with a stride of two resulting in four possible outputs to the given input. The main advantage of using this technique is that the sliding window runs and computes all values simultaneously. Consequently, this technique is really fast. The weakness of this technique is that the position of the bounding boxes is not very accurate.

**LITERATURE SURVEY :**

In various fields, there is a necessity to detect the target object and also track them effectively while handling occlusions and other included complexities. Many researchers (Almeida and Guting 2004, Hsiao-Ping Tsai 2011, Nicolas Papadakis and Aure lie Bugeau 2010 ) attempted for various approaches in object tracking. The nature of the techniques largely depends on the application domain. Some of the research works which made the evolution to proposed work in the field of object tracking are depicted as follows.

**OBJECT DETECTION:**

Object detection is an important task, yet challenging vision task. It is a critical part of many applications such as image search, image auto-annotation and scene understanding, object tracking. Moving object tracking of video image sequences was one of the most important subjects in computer vision. It had already been applied in many computer vision fields, such as smart video surveillance (Arun Hampapur 2005), artificial intelligence, military guidance, safety detection and robot navigation, medical and biological application. In recent years, a number of successful single-object tracking system appeared, but in the presence of several objects, object detection becomes difficult and when objects are fully or partially occluded, they are obtruded from the human vision which further increases the problem of detection. Decreasing illumination and acquisition angle. The proposed MLP based object tracking system is made robust by an optimum selection of unique features and also by implementing the Adaboost strong classification method.

### 3.1.1 Background Subtraction:

The background subtraction method by Horprasert et al (1999), was able to cope with local illumination changes, such as shadows and highlights, even globe illumination changes. In this method, the background model was statistically modelled on each pixel. Computational colour mode, include the brightness distortion and the chromaticity distortion which was used to distinguish shading background from the ordinary background or moving foreground objects. The background and foreground subtraction method used the following approach. A pixel was modelled by a 4-tuple [Ei, si, ai, bi], where Ei- a vector with expected colour value, si - a vector with the standard deviation of colour value, ai - the variation of the brightness distortion and bi was the variation of the chromaticity distortion of the ith pixel. In the next step, the difference between the background image and the current image was evaluated. Each pixel was finally classified into four categories: original background, shaded background or shadow, highlighted background and moving foreground object. Liyuan Li et al (2003), contributed a method for detecting foreground objects in non-stationary complex environments containing moving background objects. A Bayes decision rule was used for classification of background and foreground changes based on inter-frame colour co-occurrence statistics. An approach to store and fast retrieve colour cooccurrence statistics was also established. In this method, foreground objects were detected in two steps. First, both the foreground and the background changes are extracted using background subtraction and temporal differencing. The frequent background changes were then recognized using the Bayes decision rule based on the learned colour co-occurrence statistics. Both short-term and long term strategies to learn the frequent background changes were used. An algorithm focused on obtaining the stationary foreground regions as said by Álvaro Bayona et al (2010), which was useful for applications like the detection of abandoned/stolen objects and parked vehicles. This algorithm mainly used two steps. Firstly, a sub-sampling scheme based on background subtraction techniques was implemented to obtain stationary foreground regions. This detects foreground changes at different time instants in the same pixel locations. This was done by using a Gaussian distribution function. Secondly, some modifications were introduced on this base algorithm such as thresh holding the previously computed subtraction. The main purpose of this algorithm was reducing the amount of stationary foreground detected

### Template Matching:

Template Matching is the technique of finding small parts of an image which match a template image. It slides the template from the top left to the bottom right of the image

and compares for the best match with the template. The template dimension should be equal to the reference image or smaller

than the reference image. It recognizes the segment with the highest correlation as the target. Given an image S and an image T, where the dimension of S was both larger than T, output whether S contains a subset image I where I and T are suitably similar in pattern and if such I exists, output the location of I in S as in Hager and Bellhumear (1998). Schweitzer et al (2011), derived an algorithm which used both upper and lowers bound to detect 'k' best matches. Euclidean distance and Walsh transform kernels are used to calculate match measure. The positive things included the usage of priority queue improved quality of decision as to which bound-improved and when good matches exist inherent cost was dominant and it improved performance. But there were constraints like the absence of good matches that lead to queue cost and the arithmetic operation cost was higher. The proposed methods dint use queue thereby avoiding the queue cost rather used template matching. Visual tracking methods can be roughly categorized in two ways namely, the feature-based and region-based method as proposed by Ken Ito and Shigeyuki Sakane (2001). The feature-based approach estimates the 3D pose of a target object to fit the image features the edges, given a 3D geometrical model of an object. This method requires much computational cost. Region-based can be classified into two categories namely, parametric method and view-based method. The parametric method assumes a parametric model of the images in the target image and calculates optimal fitting

of the model to pixel data in a region. The view-based method was used to find the best match of a region in a search area given the reference template. This has the advantage that it does not require much computational complexity as in the feature-based approach

## Neural network:

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problem. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear 25 combination. Finally,

an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be - 1 and 1.

These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks,

which can derive conclusions from a complex and seemingly unrelated set of information.

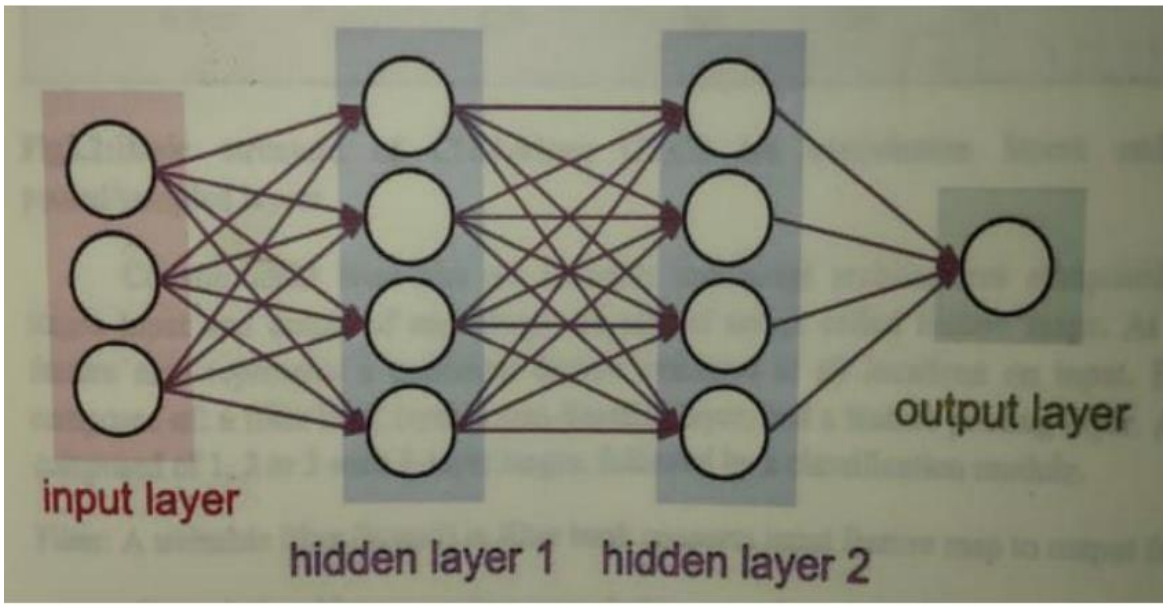



**Figure 7:**A simple neural network

**Object detection with CNN:**

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single

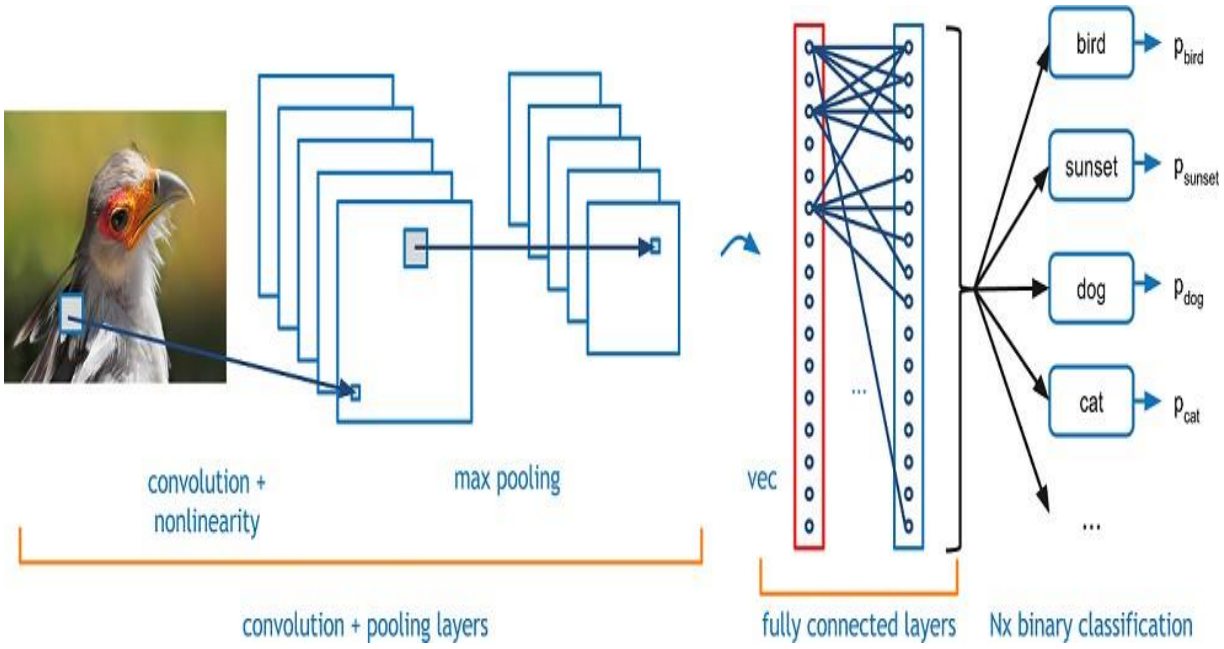



**FIGURE 8:CONVOLUTIONAL NEURAL NETWORKS**

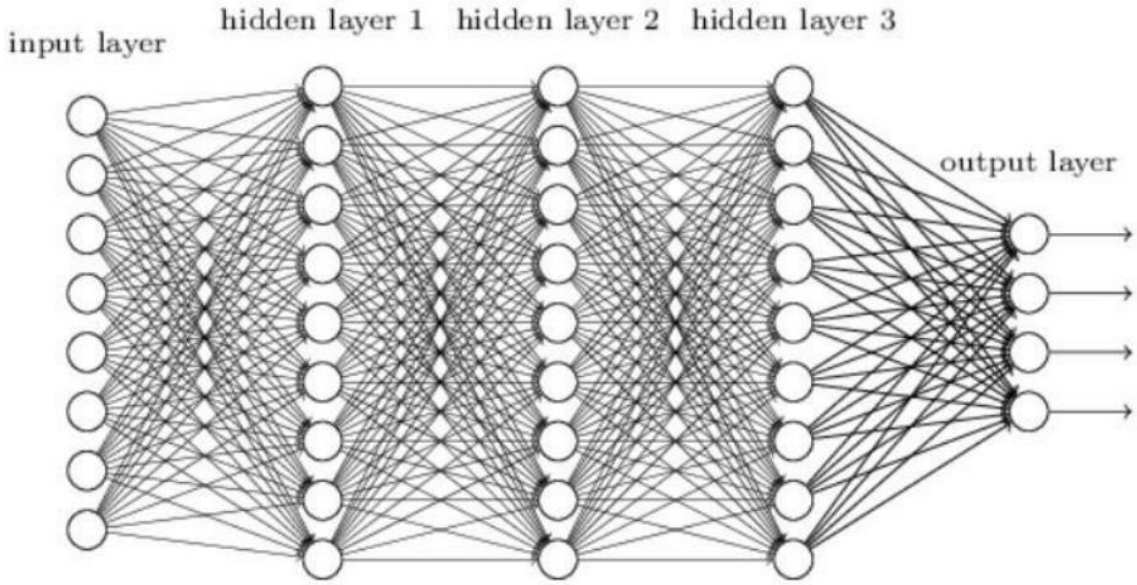
## CNN ARCHITECTURE:

CNNs are feed forward networks in that information flow takes place in one direction only, from their inputs to their outputs. Just as artificial neural networks (ANN) are

biologically inspired, so are CNNs. The visual cortex in the brain, which consists of alternating layers of simple and complex cells (Hubel & Wiesel, 1959, 1962), motivates their architecture.

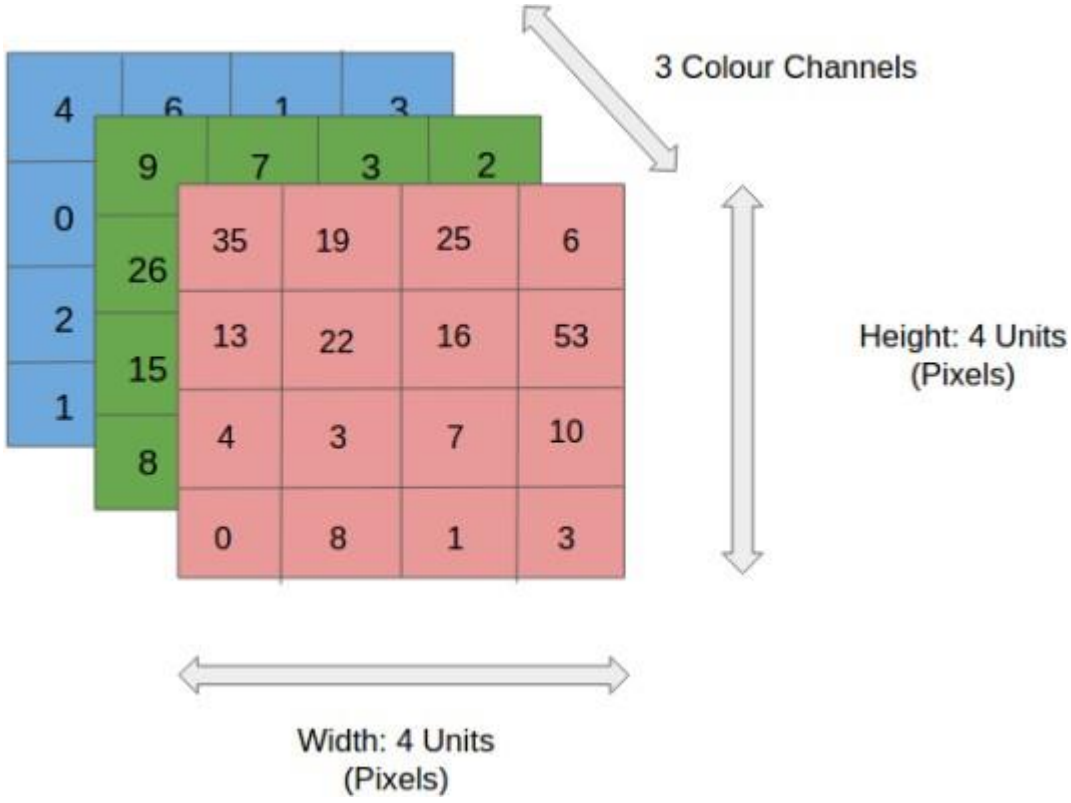
CNN architectures come in several variations; however, in general, they consist of convolutional and pooling (or subsampling) layers, which are grouped into modules. Either one or more fully connected layers, as in a standard feedforward neural network, follow these modules. Modules are often stacked on top of each other to form a deep model. It illustrates typical CNN architecture for a toy image classification task. An image is input directly to the network, and this is followed by several stages of convolution and pooling. Thereafter, representations from these operations feed one or more fully connected layers.

Finally, the last fully connected layer outputs the class label. Despite this being the most popular base architecture found in the literature, several architecture changes have been proposed in recent years with the objective of improving image classification accuracy or reducing computation costs. Although for the remainder of this section, we merely fleetingly introduce standard CNN architecture.




***Working Method of CNN:***

All the Images which are in the used in the CNN Algorithm Should be in the form of RGB image. An RGB image is nothing but a matrix of pixel values having three planes whereas a grayscale image is the same but it has a single plane. Take a look at this image to understand more.



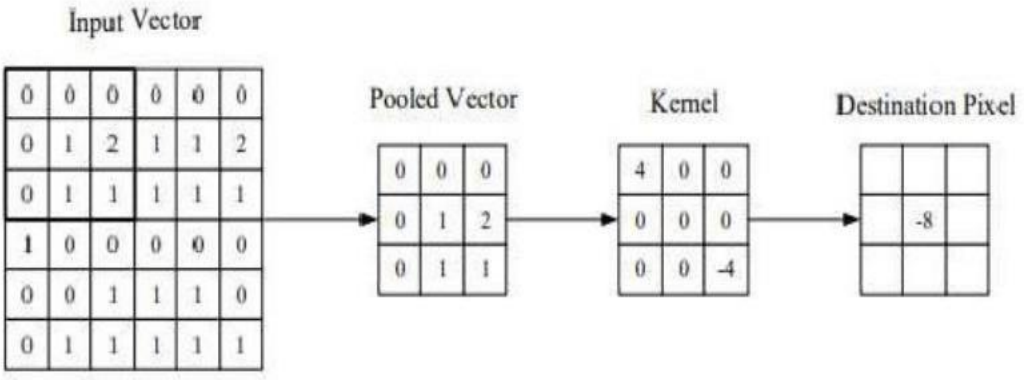

The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights, sometimes referred to as a filter bank. Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function.

All neurons within a feature map have weights that are constrained to be equal; however, different feature maps within the same convolutional layer have different weights so that several features can be extracted at each location.

As the name implies, the convolutional layer plays a vital role in how CNNs operate. The layers parameters focus around the use of learnable kernels.

These kernels are usually small in spatial dimensionality, but spreads along the entirety of the depth of the input. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation map. These activation maps can be visualised.

As we glide through the input, the scalar product is calculated for each value in that kernel. From this the network will learn kernels that 'fire' when they see a specific feature at a given spatial position ofthe input. These are commonly known as activations.




**Visual representation of a convolutional layerz**

The centre element of the kernel is placed over the input vector, of which is then calculated and replaced with a weighted sum of itself and any nearby pixels.

Every kernel will have a corresponding activation map, of which will be stacked along the depth dimension to form the full output volume from the convolutional layer

Pooling layers aim to gradually reduce the dimensionality of the representation, and thus further reduce the number of parameters and the computational complexity of the model.

The pooling layer operates over each activation map in the input, and scales its dimensionality using the "MAX" function. In most CNNs, these come in the form of max-pooling layers with kernels of 31 a dimensionality of 2 × 2 applied with a stride of 2 along the spatial dimensions of the input. This scales the activation map down to 25% of the original size - whilst maintaining the depth volume to its standard size

## *Object Detection Using R-CNN and Faster R-CNN:*

*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.

These 2000 candidate regions which are proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The CNN plays a role of feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM for the classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values for increasing the precision of the bounding box. For example, given the region proposal, the algorithm might have predicted the presence of a person but the face of

that person within that region proposal could have been cut in half. Therefore, the offset values which is given help in adjusting the bounding box of the region proposal.

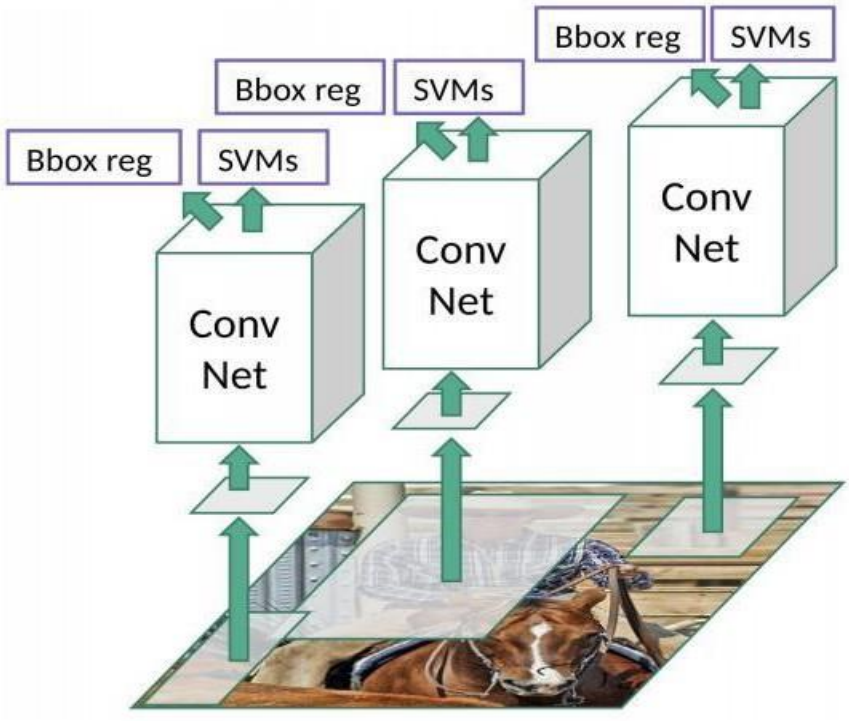
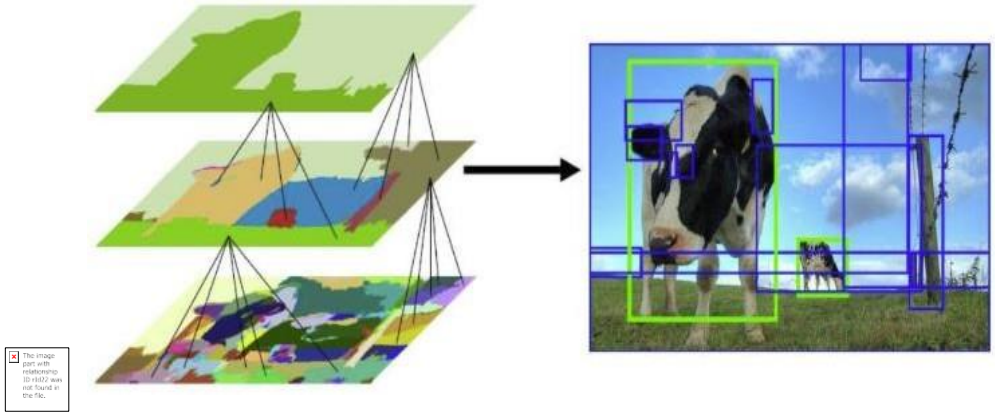



**Figure:R-CNN**

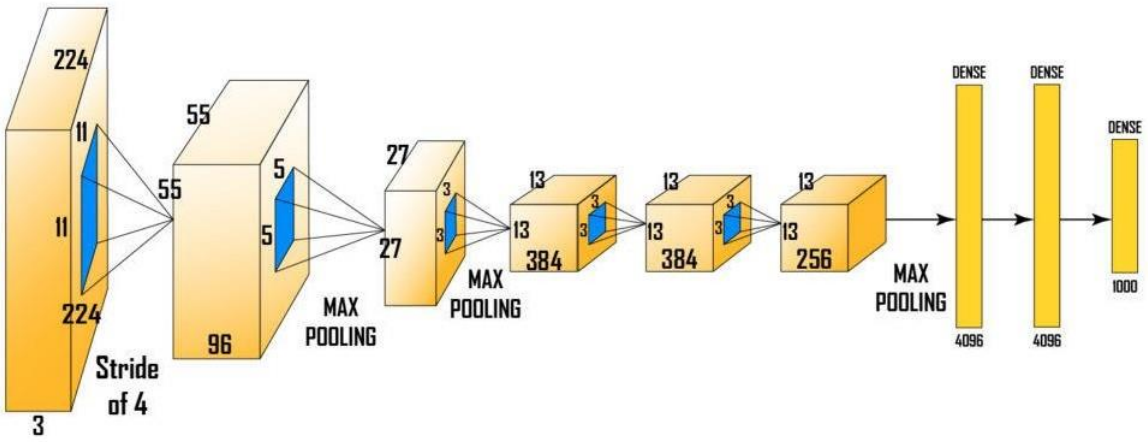
## Region Proposals:

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.



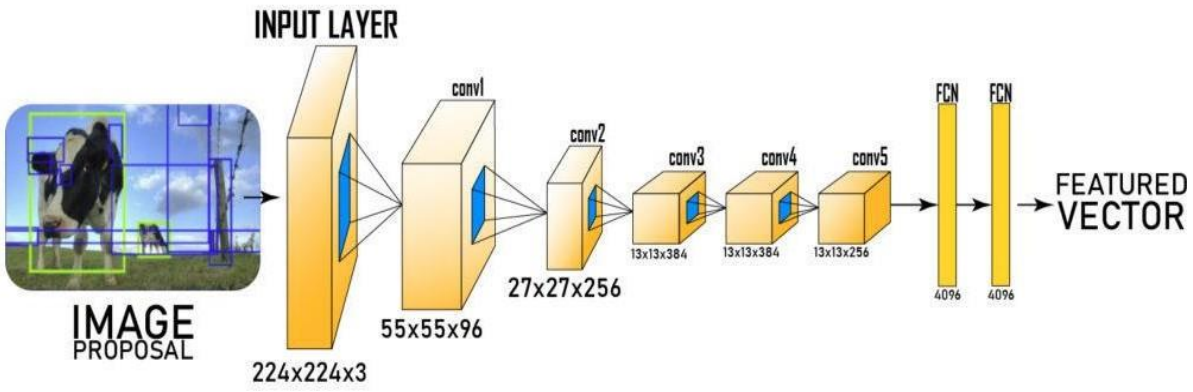
## CNN architecture of R-CNN :

After that these regions are warped into the single square of regions of dimension as required by the CNN model. The CNN model that we used here is a pre-trained AlexNet model, which is the state of the art CNN model at that time for image classification Let's look at AlexNet architecture here.




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 (1, 4096) feature vector. We pass this feature vector into SVM and bounding box regressor.




## Faster R-CNN for object detection

In the R-CNN family of Algorithms, the evolution between versions was usually in terms of computational efficiency (integrating the different training stages), reduction in test time, and improvement in performance (mAP). These networks usually consist of

a) A region proposal algorithm to generate "bounding boxes" or locations of possible objects in the image

b) A feature generation stage to obtain features of these objects, usually using a CNN

c) A classification layer to predict which class this object belongs to; and d) A regression layer to make the coordinates of the object bounding box more precise.

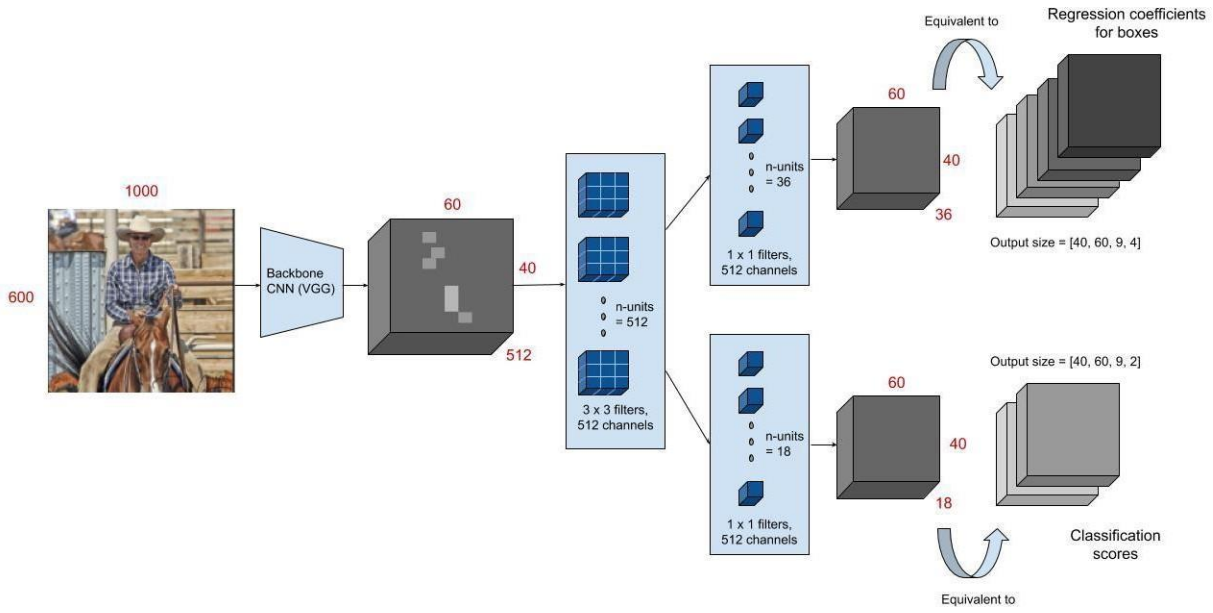
The only stand-alone portion of the network left in Fast R-CNN was the region proposal algorithm. Both R-CNN and Fast R-CNN use CPU based region proposal algorithms.

Eg- the Selective search algorithm which takes around 2 seconds per image and runs on CPU computation. The Faster R-CNN [3] paper fixes this by using another convolutional network (the RPN) to generate the region proposals. This not only brings down the region proposal time from 2s to 10ms per image but also allows the region proposal stage to share layers with the following detection stages, causing an overall improvement in feature representation. In the rest of the article, "Faster R-CNN" usually refers to a detection pipeline that uses the RPN as a region proposal algorithm, and Fast R-CNN as a detector network.

## Region Proposal Network (RPN):

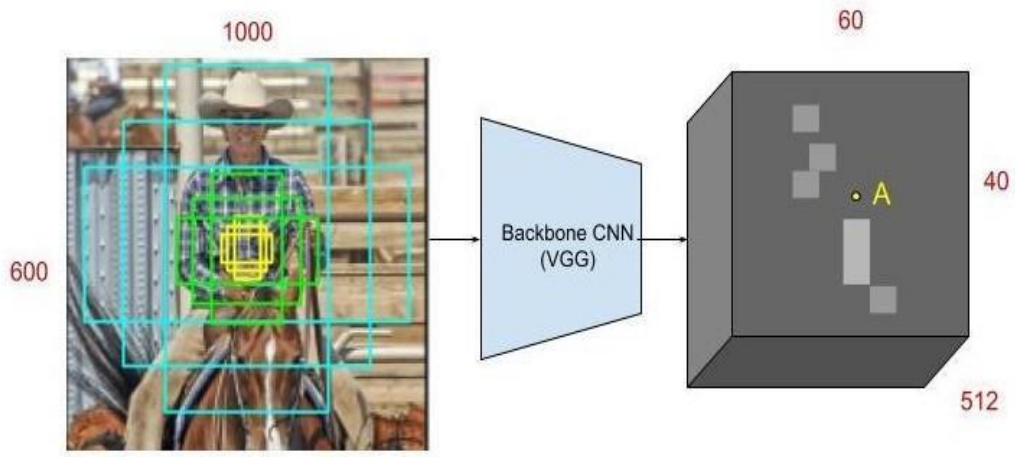
The region proposal network (RPN) starts with the input image being fed into the backbone convolutional neural network. The input image is first re sized such that it's shortest side is 600px with the longer side not exceeding 1000px.

The output features of the backbone network (indicated by H x W) are usually much smaller than the input image depending on the stride of the backbone network. For both the possible backbone networks used in the paper (VGG, ZF-Net) the network stride is 16. This means that two consecutive pixels in the backbone output features correspond to two points 16 pixels apart in the input image.




For every point in the output feature map, the network has to learn whether an object is present in the input image at its corresponding location and estimate its size. This is done by placing a set of "Anchors" on the input image for each location on the output feature map from the backbone network. These anchors indicate possible objects in various sizes and aspect ratios at this location. The figure below shows 9 possible anchors in 3 different aspect ratios and 3 different sizes placed on the input image for a

point A on the output feature map. For the PASCAL challenge, the anchors used have 3 scales of box area $128^2$, $256^2$, $512^2$ and 3 aspect ratios of 1:1, 1:2 and 2:1.




As the network moves through each pixel in the output feature map, it has to check whether these *k* corresponding anchors spanning the input image actually contain objects, and refine these anchors' coordinates to give bounding boxes as "Object proposals" or regions of interest.

First, a 3 x 3 convolution with 512 units is applied to the backbone feature map as shown in Figure 1, to give a 512-d feature map for every location. This is followed by two sibling layers: a 1 x 1 convolution layer with 18 units for object classification, and a 1 x 1 convolution with 36 units for bounding box regression.

The 18 units in the classification branch give an output of size (H, W, 18). This output is used to give probabilities of whether or not each point in the backbone feature map (size: H x W) **contains an object** within all 9 of the anchors at that point.

The 36 units in the regression branch give an output of size (H, W, 36). This output is used to give the 4 regression coefficients of each of the 9 anchors for every point in the backbone feature map (size: H x W). These regression coefficients are used to improve the coordinates of the anchors that contain objects.

**Training and Loss functions:**

**The output feature map consists of about 40 x 60 locations, corresponding to 40*60*9 20k anchors in total. At train time, all the anchors that cross the boundary are ignored so that**

**they do not contribute to the loss. This leaves about 6k anchors per image.**

An anchor is considered to be a "positive" sample if it satisfies either of the two conditions — a) The anchor has the highest IoU (Intersection over Union, a measure of overlap) with a groundtruth box; b) The anchor has an IoU greater than 0.7 with any groundtruth box. The same groundtruth box can cause multiple anchors to be assigned positive labels.

An anchor is labeled "negative" if its IoU with all groundtruth boxes is less than 0.3. The remaining anchors (neither positive nor negative) are disregarded for RPN training.

Each mini-batch for training the RPN comes from a single image. Sampling all the anchors from this image would bias the learning process toward negative samples, and so 128 positive and 128 negative samples are randomly selected to form the batch, padding with additional negative samples if there are an insufficient number of positives.

The training loss for the RPN is also a multi-task loss, given by:

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

Here *i* is the index of the anchor in the mini-batch. The classification loss *L* is the log loss over two classes (object vs not object). $p_i$ is the output score from the classification branch for anchor *i*, and $p_i^*$ is the groundtruth label (1 or 0).

The regression loss $L_{re}(t_i, t_i^*)$ is activated only if the anchor actually contains an object i.e., the groundtruth $p_i^*$ is 1. The term $t_i$ is the output prediction of the regression layer and consists of 4 variables [$t_x$, $t_y$, $t_w$, $t_h$]. The regression target $t_i^*$ is calculated as

$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \quad t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a)$$

Here x, y, w, and h correspond to the *(x, y)* coordinates of the box centre and the height *h* and width *w* of the box. $x_a$, $x^*$ stand for the coordinates of the anchor box and its corresponding groundtruth bounding box.

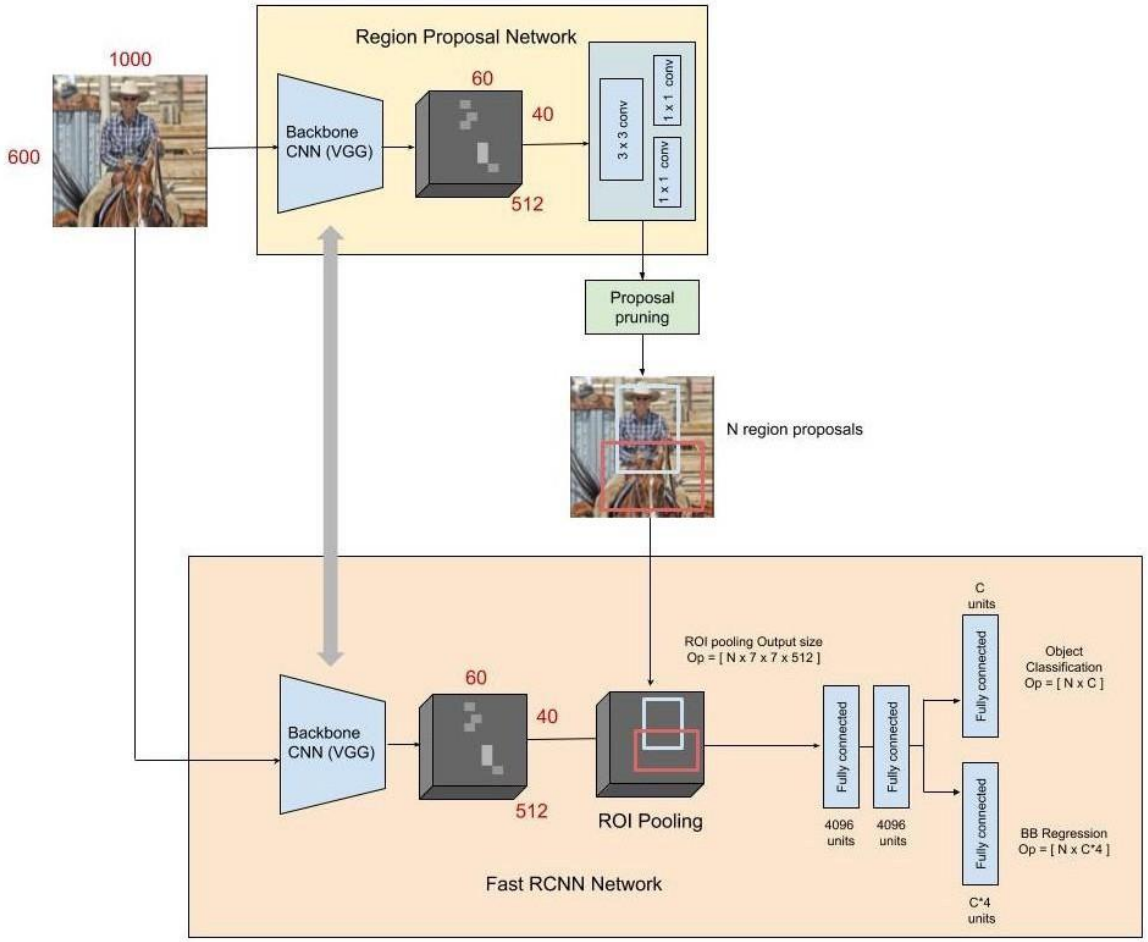
Remember that all *k* (= 9) of the anchor boxes have different regressors that do not share weights. So the regression loss for an anchor *i* is applied to its corresponding regressor (if it is a positive sample).

At test time, the learned regression output $t_i$ can be applied to its corresponding anchor box (that is predicted positive), and the *x, y, w, h* parameters for the predicted object proposal bounding box can be back-calculated from

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = \log(w/w_a), \quad t_h = \log(h/h_a)$$

## Object detection: Faster R-CNN (RPN + Fast R-CNN)

The Faster R-CNN architecture consists of the RPN as a region proposal algorithm and the Fast R-CNN as a detector network.



**Fast R-CNN as a detector for Faster R-CNN:**

The Fast R-CNN detector also consists of a CNN backbone, an ROI pooling layer and fully connected layers followed by two sibling branches for classification and bounding box regression as shown in Figure 3.

The input image is first passed through the backbone CNN to get the feature map (Feature size: 60, 40, 512). Besides test time efficiency, another key reason using an RPN as a proposal generator makes sense is the advantages of **weight sharing between the RPN backbone and the Fast R-CNN detector backbone**.

Next, the bounding box proposals from the RPN are used to pool features from the backbone feature map. This is done by the ROI pooling layer. The ROI pooling layer, in essence, works by a) Taking the region corresponding to a proposal from the backbone feature map; b) Dividing this region into a fixed number of sub-windows; c) Performing max-pooling over these sub-windows to give a fixed size output. To understand the details of the ROI pooling layer and it's advantages, read Fast R-CNN

The output from the ROI pooling layer has a size of (N, 7, 7, 512) where N is the number of proposals from the region proposal algorithm. After passing them through two fully connected layers, the features are fed into the sibling classification and regression branches.

Note that these classification and detection branches are different from those of the RPN. Here the classification layer has C units for each of the classes in the detection task (including a catch-all background class). The features are passed through a softmax layer to get the classification scores — the probability of a proposal belonging to each class. The regression layer coefficients are used to improve the predicted bounding boxes. Here the regressor is size agnostic, (unlike the RPN) but is specific to each class. That is, all the classes have individual regressors with 4 parameters each corresponding to C*4 output units in the regression layer.

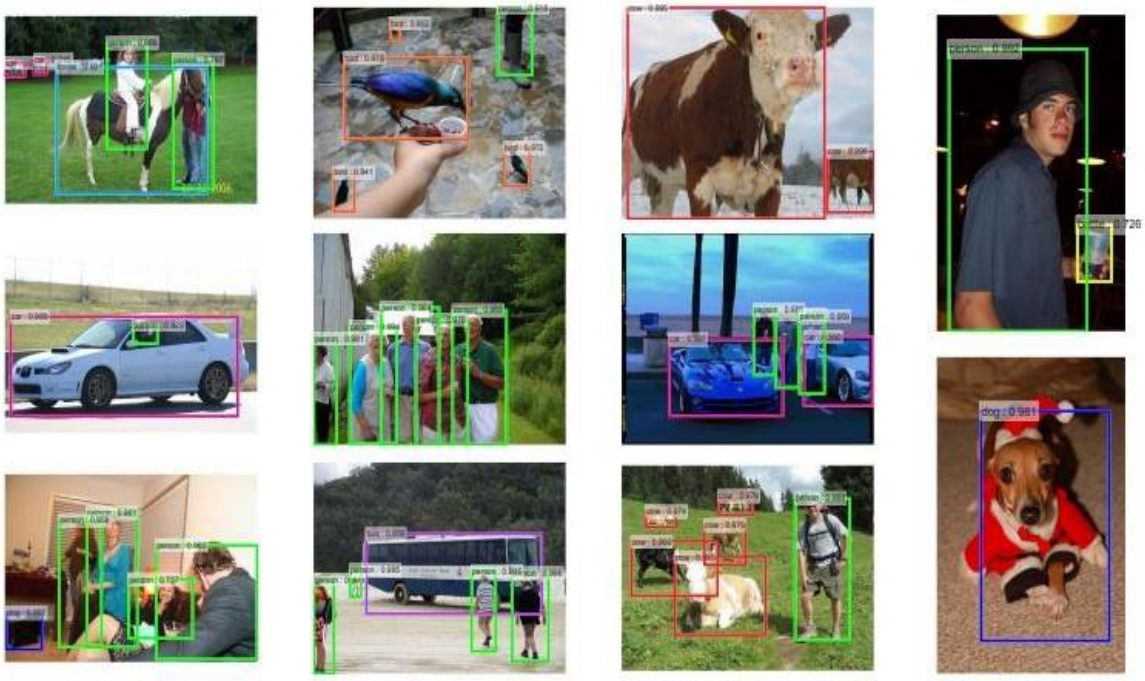
For more details on how the Faster R-CNN is trained and its loss functions refer to Fast R-CNN

In order to force the network to share the weights of the CNN backbone between the RPN and the detector, the authors use a 4 step training method:

1. The RPN is trained independently as described above. The backbone CNN for this task is initialized with weights from a network trained for an ImageNet classification task, and is then fine-tuned for the region proposal task.
2. The Fast R-CNN detector network is also trained independently. The backbone CNN for this task is initialized with weights from a network trained for an ImageNet

classification task, and is then fine-tuned for the object detection task. The RPN weights are fixed and the proposals from the RPN are used to train the Faster R-CNN.
3. The RPN is now initialized with weights from this Faster R-CNN, and fine-tuned for the region proposal task. This time, weights in the common layers between the RPN and detector remain fixed, and only the layers unique to the RPN are fine-tuned. This is the final RPN.
4. Once again using the new RPN, the Fast R-CNN detector is fine-tuned. Again, only the layers unique to the detector network are fine-tuned and the common layer weights are fixed.

This gives a Faster R-CNN detection framework that has shared convolutional layers



## Region Proposal Network (RPN)

The R-CNN and Fast R-CNN models depend on the Selective Search algorithm for generating region proposals. Each proposal is fed to a pre-trained CNN for classification. This paper [3][3] proposed a network called region proposal network (RPN) that can produce the region proposals. This has some advantages:

1. The region proposals are now generated using a network that could be trained and customized according to the detection task.

2. Because the proposals are generated using a network, this can be trained end-to-end to be customized on the detection task. Thus, it produces better region proposals compared to generic methods like **Selective Search** and **EdgeBoxes**.
3. The RPN processes the image using the same convolutional layers used in the Fast R-CNN detection network. Thus, the RPN does not take extra time to produce the proposals compared to the algorithms like Selective Search.
4. Due to sharing the same convolutional layers, the RPN and the Fast R-CNN can be merged/unified into a single network. Thus, training is done only once.

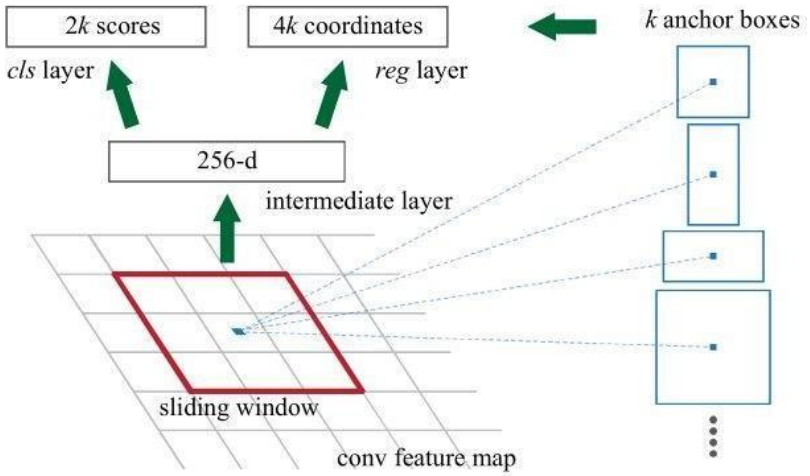
The RPN works on the output feature map returned from the last convolutional layer shared with the Fast R-CNN. This is shown in the next figure. Based on a rectangular window of size nxn, a sliding window passes through the feature map. For each window, several candidate region proposals are generated. These proposals are not the final proposals as they will be filtered based on their "objectness score" (explained below).

**Anchor**

According to the next figure, the feature map of the last shared convolution layer is passed through a rectangular sliding window of size nxn, where n=3for the VGG-16 net. For each window, K region proposals are generated. Each proposal is parametrized according to a reference box which is called an **anchor box**. The 2 parameters of the anchor boxes are:

1. Scale
2. Aspect Ratio

Generally, there are 3 scales and 3 aspect ratios and thus there is a total of K=9 anchor boxes. But K may be different than 9. In other words, K regions are produced from each region proposal, where each of the Kregions varies in either the scale or the aspect ratio. Some of the anchor variations are shown in the next figure.



Using reference anchors (i.e. anchor boxes), a single image at a single scale is used while being able to offer scale-invariant object detectors, as the anchors exist at different scales. This avoids using multiple images or filters. The multi-scale anchors are key to share features across the RPN and the Fast R-CNN detection network.

For each nxn region proposal, a feature vector (of length 256 for ZF net and 512 for the VGG-16 net) is extracted. This vector is then fed to 2 sibling fully-connected layers:

1. The first FC layer is named cls, and represents a binary classifier that generates the **objectness score** for each region proposal (i.e. whether the region contains an object, or is part of the background).
2. The second FC layer is named regwhich returns a 4-D vector defining the bounding box of the region.

The first FC layer (i.e. binary classifier) has 2 outputs. The first is for classifying the region as a background, and the second is for classifying the region as an object. The next section discusses how the objectness score is assigned to each anchor and how it is used to produce the classification label.

### Objectness Score

The cls layer outputs a vector of 2 elements for each region proposal. If the first element is 1 and the second element is 0, then the region proposal is classified asbackground. If the second element is 1 and the first element is 0, then the region represents an object.

For training the RPN, each anchor is given a positive or negative objectness score based on the **Intersection-over-Union (IoU)**.

The IoU is the ratio between the **area of intersection** between the anchor box and the ground-truth box to the **area of union** of the 2 boxes. The IoU ranges from 0.0 to 1.0. When there is no intersection, the IoU is 0.0. As the 2 boxes get closer to each other, the IoU increases until reaching 1.0 (when the 2 boxes are 100% identical).

The next 4 conditions use the IoU to determine whether a positive or a negative **objectness score** is assigned to an anchor:

1. An anchor that has an IoU overlap higher than **0.7** with any ground-truth box is given a positive objectness label.
2. If there is no anchor with an IoU overlap higher than **0.7**, then assign a positive label to the anchor(s) with the highest IoU overlap with a ground-truth box.
3. A negative **objectness score** is assigned to a **non-positive** anchor when the IoU overlap for all ground-truth boxes is less than **0.3**. A negative objectness score means the anchor is classified as background.
4. Anchors that are neither positive nor negative do not contribute to the training objective.

I was confused by the second and third conditions when I was first reading the paper. So, let's give more clarification.

Assume there are 3 region proposals associated with 3 anchors, where their IoU scores with 3 ground-truth boxes are listed below. Because there is an anchor with an IoU score of 0.9, which is higher than 0.7, it is assigned a positive objectness score with that ground-truth box, and negative to all other boxes.

0.9, 0.55, 0.1

Here is the result of classifying the anchors:

positive, negative, negative

The second condition means that when no anchor has an IoU overlap score higher than **0.7**, then search for the anchor with the highest IoU and assign it a positive objectness score. It is expected that the maximum IoU score is less than or equal to 0.7, but the confusing part is that the paper did not mention a minimum value of theIoU score.

It is expected that the minimum value should be **0.5**. So, if an anchor box has an IoU score that is greater than **0.5** but less than or equal to **0.7**, then assign it a positive objectness score.

Assume that the IoU scores of an anchor are listed below. Because the highest IoU score is the second one with a value of 0.55, it falls under the second condition. Thus, it is assigned a positive objectness score.

0.2, 0.55, 0.1

Here is the result of classifying the anchors:

negative, positive, negative

The third condition specifies that when the IoU scores of an anchor with all ground-truth boxes are less than 0.3, then this anchor is assigned a negative objectness score. For the next IoU scores, the anchor is given a negative score for the 3 cases because all of the IoU scores are less than 0.3.

0.2, 0.25, 0.1

Here is the result of classifying the anchors:

negative, negative, negative

According to the fourth condition, when an anchor has an IoU score that is greater than or equal to 0.3 but less than or equal to 0.5, it is neither classified as positive nor negative. This anchor is not used in training the classifier.

For the following IoU scores the anchor is not assigned any label, as all of the scores are between 0.3 and 0.5 (inclusive).

0.4, 0.43, 0.45

The next equation summarizes the 4 conditions.

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = \log(w/w_a), \quad t_h = \log(h/h_a)$$

Note that the first condition (0.7 < IoU) is usually sufficient to label an anchor as positive (i.e. contains an object) but the authors preferred to mention the second condition (0.5 < IoU <= 0.7) for the rare cases where there is no region with an IoUof 0.7.

## Feature Sharing between RPN and Fast R-CNN

The 2 modules in the Fast R-CNN architecture, namely the RPN and Fast R-CNN, are independent networks. Each of them can be trained separately. In contrast, for Faster R-CNN it is possible to build a unified network in which the RPN and Fast R-CNN are trained at once.

The core idea is that both the RPN and Fast R-CNN share the same convolutional layers. These layers exist only once but are used in the 2 networks. It is possible to call it **layer sharing** or **feature sharing**. Remember that the anchors [3][3] are what makes it possible to share the features/layers between the 2 modules in the Faster R-CNN.

## Training Faster R-CNN

The Faster R-CNN paper [3][3] mentioned 3 different ways to train both the RPN andFast R-CNN while sharing the convolutional layers:

1. Alternating Training
2. Approximate Joint Training

3. Non-Approximate Joint Training

## Alternating Training

The first method is called **alternating training**, in which the RPN is first trained to generate region proposals. The weights of the shared convolutional layers are initialized based on a pre-trained model on ImageNet. The other weights of the RPN are initialized randomly.

After the RPN produces the boxes of the region proposals, the weights of both the RPN and the shared convolutional layers are tuned.

The generated proposals by the RPN are used to train the Fast R-CNN module. In this case, the weights of the shared convolutional layers are initialized with the tuned weights by the RPN. The other Fast R-CNN weights are initialized randomly. While the Fast R-CNN is trained, both the weights of Fast R-CNN and the shared layers are tuned. The tuned weights in the shared layers are again used to train the RPN, and the process repeats.

**Alternating training** is the preferred way to train the 2 modules and is applied in all experiments.

## Approximate Joint Training

The second method is called **approximate joint training**, in which both the RPN and Fast R-CNN are regarded as a single network, not 2 separate modules. In this case, the region proposals are produced by the RPN.

Without updating the weights of neither the RPN nor the shared layers, the proposals are fed directly to the Fast R-CNN which detects the objects' locations. Only after the Fast R-CNN produces its outputs are the weights in the Faster R-CNN tuned.

Because the weights of the RPN and the shared layers are not updated after the region proposals are generated, the weights' gradients with respect to the region proposals are ignored. This reduces the accuracy of this method compared to the first method (even if the results are close). On the other hand, the training time is reduced by about 25-50%.

## Non-Approximate Joint Training

In the **approximate joint training** method, an **RoI Warping** layer is used to allow the weights' gradients with respect to the proposed bounding boxes to be calculated..

# RESULTS AND DISCUSSIONS

**INTRODUCTION TO IMPLEMENTATION OF PROBLEM:**

Deep learning is a popular technique used in computer vision. We chose **Faster Regions with convolutional neural networks** layers as building blocks to create our model architecture.

The Faster R-CNN detector adds a region proposal network (RPN) to generate region proposals directly in the network instead of using an external algorithm like Edge Boxes. The RPN uses Anchor Boxes for Object Detection. Generating region proposals in the network is faster and better tuned to your data.

Use the trainFasterRCNNObjectDetector function to train a Faster R-CNN object detector. The function returns a fasterRCNNObjectDetector that detects objects from an image

## Input Layer:

The input layer has pre-determined, fixed dimensions, so the image must be pre-processed before it can be fed into the layer. We used OpenCV, a computer vision library, for object detection Using Satellite Image.

The OpenCV contains pre-trained filters and uses Adaboost to quickly find and crop the object. The cropped object is then converted into gray scale using cv2.cvtColor and resized to 48-by-48 pixels with cv2.resize. This step greatly reduces the dimensions compared to the original RGB format with three colour dimensions (3, 48, 48). The pipeline ensures every image can be fed into the input layer as a (1, 48, 48) numpy array.

**Convolutional Layers:**

The numpy array gets passed into the Convolution2D layer where we specify the number of filters as one of the hyper parameters. The set of filters are unique with randomly generated weights. Each filter, (3, 3) receptive field, slides across the original image with shared weights to create a feature map.

Convolution generates feature maps that represent how pixel values are enhanced, for example, edge and pattern detection. A feature map is created by applying filter 1 across the entire image. Other filters are applied one after another creating a set of feature maps.

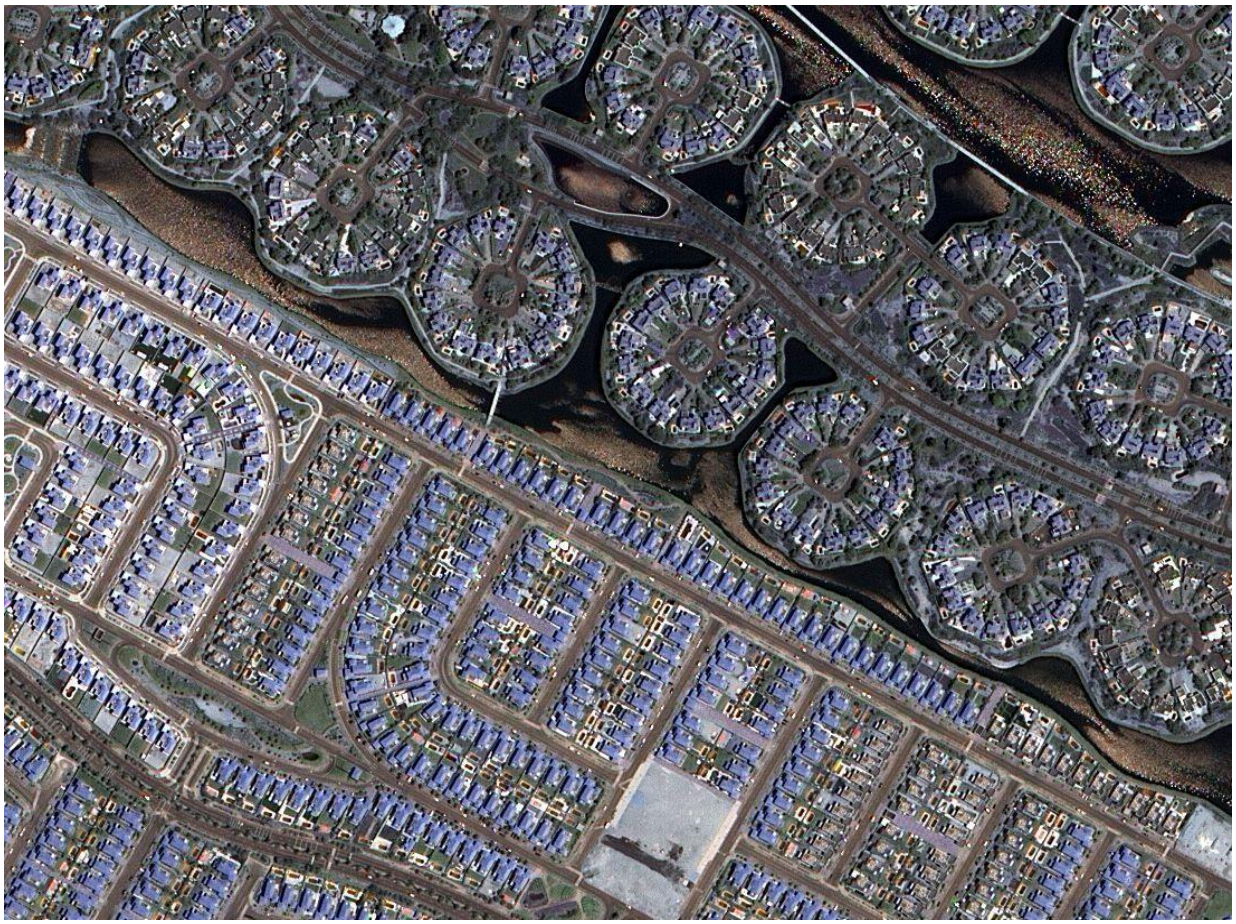
Pooling is a dimension reduction technique usually applied after one or several convolutional layers. It is an important step when building CNNs as adding more convolutional layers can greatly affect computational time. We used a popular pooling method called MaxPooling2D that uses (2, 2) windows across the feature map only keeping the maximum pixel value. The pooled pixels form an image with dimentions reduced by 4.

## RPN Layer:

The RPN works on the output feature map returned from the last convolutional layer shared with the Fast R-CNN. This is shown in the next figure. Based on a rectangular window of size nxn, a sliding window passes through the feature map. For each window, several candidate region proposals are generated. These proposals are not the final proposals as they will be filtered based on their "objectness score"

## Results:

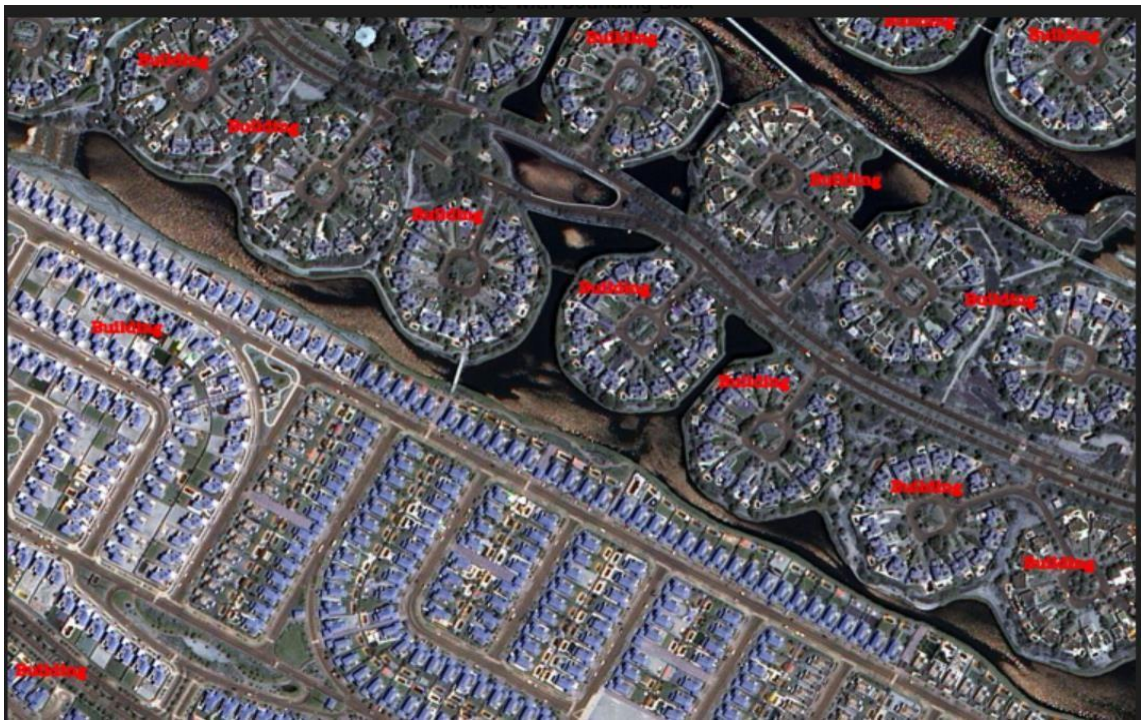
### INPUT SATELLITE IMAGE 1:



## OUTPUT CO-ORDINATE 1:

```
Image:(846, 1099, 3)
Building
Coordinates:(117, 83),(188, 108)
Building
Coordinates:(343, 203),(127, 114)
Building
Coordinates:(506, 321),(89, 76)
Building
Coordinates:(489, 509),(115, 102)
Building
Coordinates:(436, 676),(102, 100)
Building
Coordinates:(258, 743),(103, 73)
Building
Coordinates:(55, 629),(106, 102)
Building
Coordinates:(63, 341),(112, 123)
Building
Coordinates:(466, 35),(126, 106)
Building
Coordinates:(622, 149),(129, 127)
```

## OUTPUT IMAGE 1:



## INPUT SATELLITE IMAGE 2:

## OUTPUT CO-ORDINATES 2:

```
Building
Coordinates:(199, 244),(78, 62)
Building
Coordinates:(215, 460),(62, 31)
Building
Coordinates:(123, 452),(48, 41)
Building
Coordinates:(78, 413),(26, 30)
Building
Coordinates:(78, 350),(26, 41)
Building
Coordinates:(131, 287),(27, 50)
Building
Coordinates:(253, 165),(48, 31)
Building
Coordinates:(236, 127),(24, 26)
Building
Coordinates:(105, 175),(70, 51)
Building
Coordinates:(81, 119),(25, 32)
```

## OUTPUT IMAGE 2:

## INPUT SATELLITE IMAGE 3:

## OUTPUT CO-ORDINATES 3:

```
Image:(544, 510, 3)
Building
Coordinates:(20, 144),(67, 41)
Building
Coordinates:(56, 278),(37, 32)
Building
Coordinates:(76, 330),(15, 28)
Building
Coordinates:(88, 368),(39, 20)
Building
Coordinates:(2, 234),(41, 45)
Building
Coordinates:(24, 379),(23, 25)
Building
Coordinates:(31, 423),(32, 26)
Building
Coordinates:(10, 495),(48, 37)
Building
Coordinates:(185, 402),(51, 26)
Building
Coordinates:(146, 353),(31, 24)
```

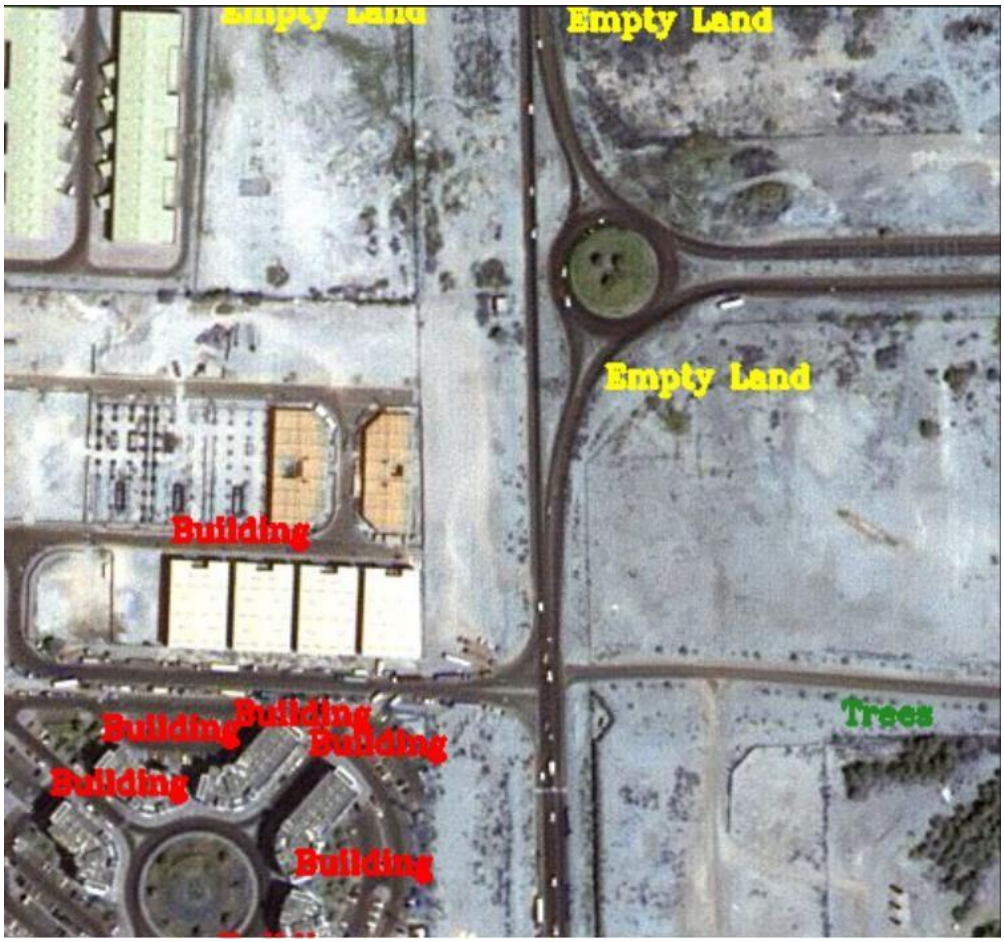
## OUTPUT IMAGE 3:

## INPUT SATELLITE IMAGE 4:

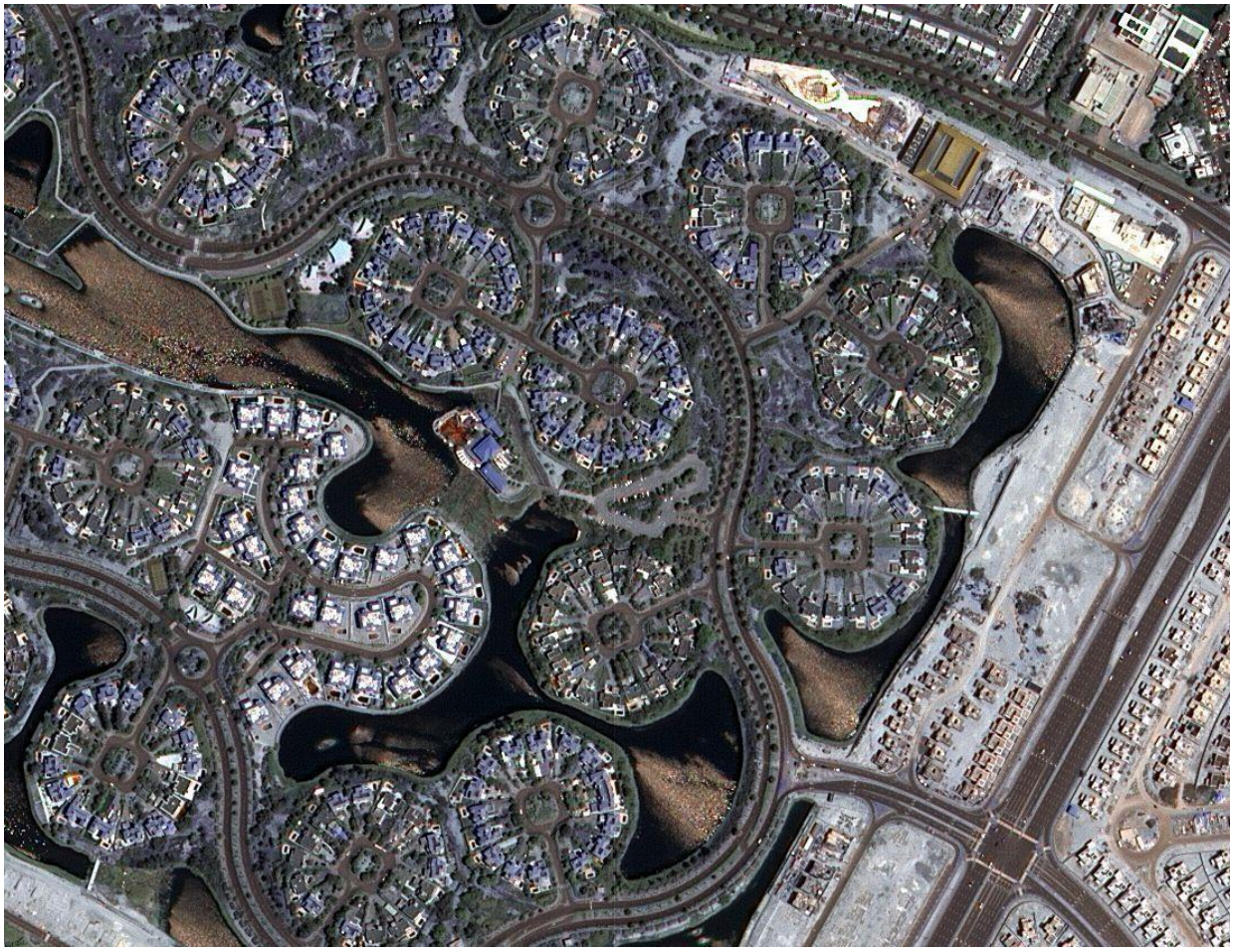
## IMAGE CO-ORDINATE 4:

```
Building
Coordinates:(147, 516),(31, 12)
Building
Coordinates:(116, 461),(34, 18)
Building
Coordinates:(154, 421),(13, 30)
Building
Coordinates:(161, 364),(15, 29)
Building
Coordinates:(123, 350),(35, 13)
Building
Coordinates:(194, 535),(12, 9)
Building
Coordinates:(19, 530),(37, 14)
Building
Coordinates:(92, 263),(119, 46)
...
Empty Land
Coordinates:(117, 6),(120, 161)
Trees
Coordinates:(429, 350),(80, 102)
```

## OUTPUT IMAGE 4:



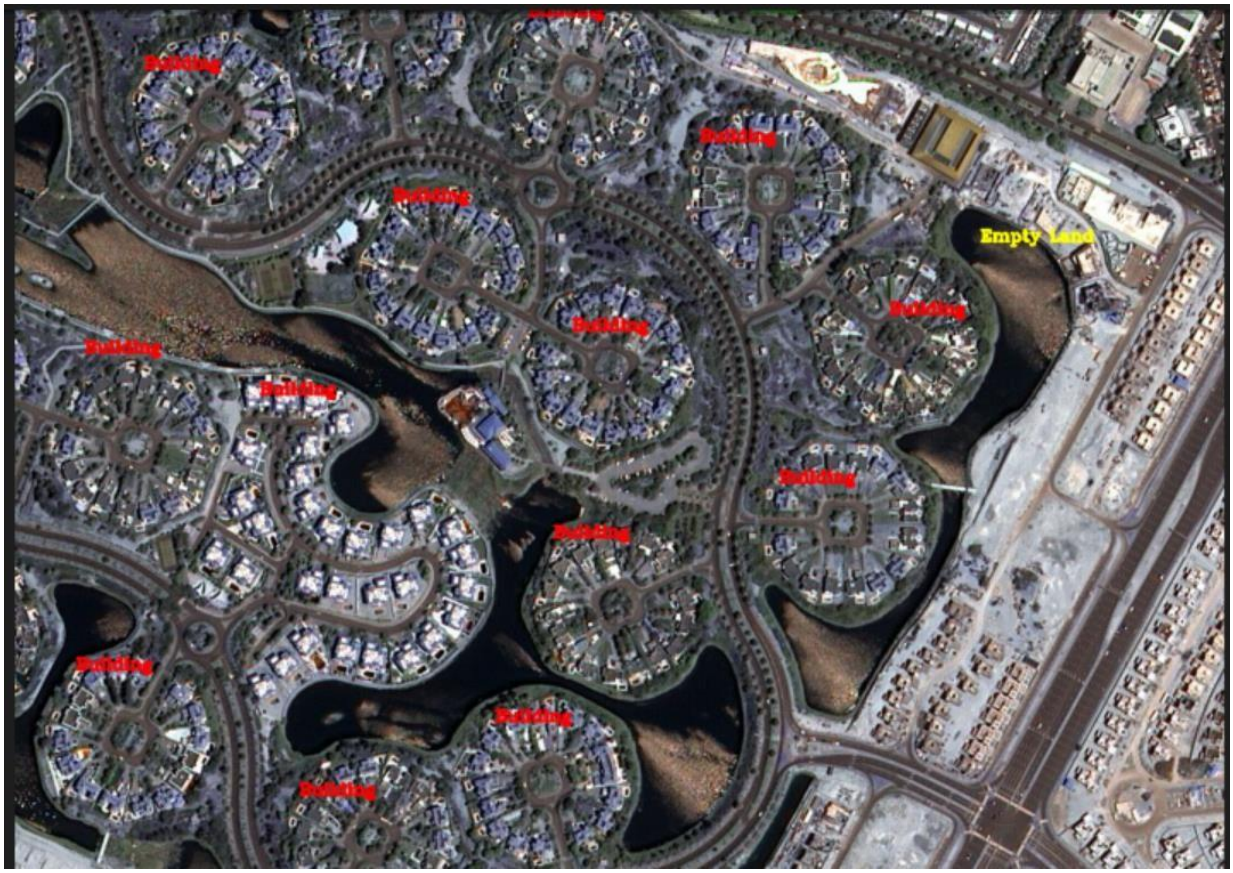
## INPUT SATELLITE IMAGE 5:



## IMAGE CO-ORDINATE 5:

```
Image:(1058, 1126, 3)
Water
Coordinates:(17, 25),(1109, 273)
Water
Coordinates:(593, 387),(404, 252)
Empty Land
Coordinates:(134, 497),(453, 554)
```

## OUTPUT IMAGE 5:




## APPLICATION:

**Here are a some of the future implementation of object detection.**

1. **Face detections and recognition:**

Face detection perhaps be a separate class of object detection. We wonder how some applications like Facebook, Faceapp, etc., detect and recognize our faces. this is often a sample example of object detection in our day to day life. Face detection is already in use in our lifestyle to unlock our mobile phones and for other security systems to scale back rate .

**2. Object tracking:**

Object detection is additionally utilized in tracking objects like tracking an individual and his actions, continuously monitoring a ball within the game of Football or Cricket. As there's an enormous interest for people in these games, these tracking techniques enables them to know it during a better way and obtain some additional information.

Tracking of the ball is of maximal importance in any ball-based games to automatically record the movement of the ball and adjust the video frame accordingly.

**3. Self-driving cars:**

This is often one among the main evolutions of the planet and is that the best example why we'd like object detection. so as for a car to travel to the specified destination automatically with none human interference or to form decisions whether to accelerate or to use brakes and to spot the objects around it. this needs object detection.

4. **Emotions detection**:

This permits the system to spot the type of emotion the person puts on his face. the corporate Apple has already tried to use this by detecting the emotion of the user and converting it into a respective emoji within the smart phone.

5. **Biometric identification through retina scan**:

Retina scan through iris code is one among the techniques utilized in high security systems because it is one among the foremost accurate and unique biometric.

6. **Smart text search and text selection (Google lens)** :

In recent times, we've encountered an application in smart phones called google lens. this will recognize the text and also images and search the relevant information within the browser without much effort

## CONCLUSION:

Deep-learning based object detection has been a search hotspot in recent years. This project starts on generic object detection pipelines which give base architectures for other related tasks. With the assistance of this the 3 other common tasks, namely object detection, face detection and pedestrian detection, are often accomplished. Authors accomplished this by combing 2 things: Object detection with deep learning and OpenCV and Efficient, threaded video streams with OpenCV. The camera sensor noise and lightening condition can change the result because it can create problem in recognizing the objects. generally, this whole process requires GPU's rather than CPU's. But we've done using CPU's and executes in much less time, making it efficient. Object Detection algorithms act as a mixture of both image classification and object localization. It takes the given image as input and produces the output having the bounding boxes adequate to the amount of objects present within the image with the category label attached to every bounding box at the highest. It projects the scenario of the bounding box up the shape of position, height and width.

**REFERENCE:**

1. Abdul Vahab, Maruti S Naik, Prasanna G Raikar an Prasad S R4, "Applications of Object Detection System", International Research Journal of Engineering and Technology (IRJET)

2. Astha Gautam, Anjana Kumari, Pankaj Singh: "The Concept of Object Recognition", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 3, March 2015

3. https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/