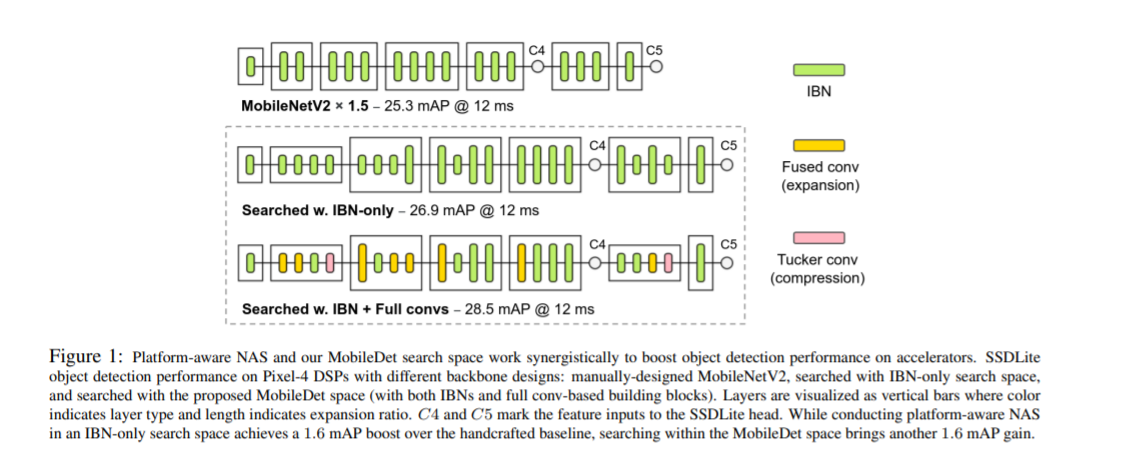
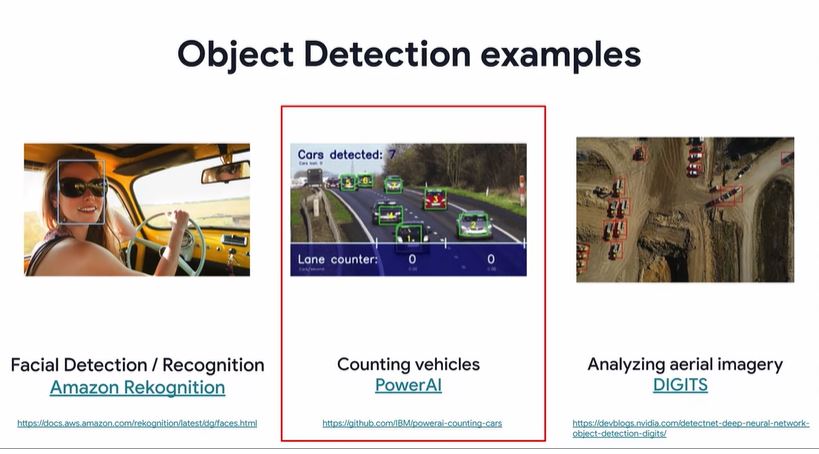
# Mobiledet





Reference: https://www.coursera.org/learn/advanced-computer-vision-with-tensorflow/supplement/3oVBO/references-amazon-rekognition-powerai-digits

# 2 approaches used by object detection:

1. **Sliding window:**

****

Sliding windows make use of rectangles that pass across the image.

At each position of the sliding window.

The model attempts to classify what it sees within the boundaries of the rectangle and ignores the rest of the image.

So in this example, when the window is over the truck the model can classify what it sees within the window as a truck.

Plus because of the position of the sliding window We can also infer something about the trucks location within the whole image.

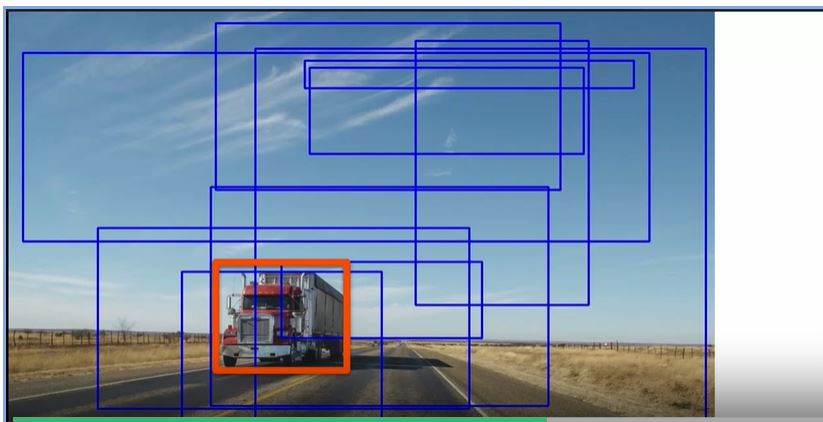
So we have the beginnings of getting a bounding box for an object of type truck.

By using boxes of different sizes. We can also try to get more precise about the bounding box of the truck.

The green box that lights up here when it passes the truck doesn't encapsulate the entire truck.

So multiple boxes of varying sizes like this could be combined to figure out the overall bounding box for the truck

1. **Selective search:**

****

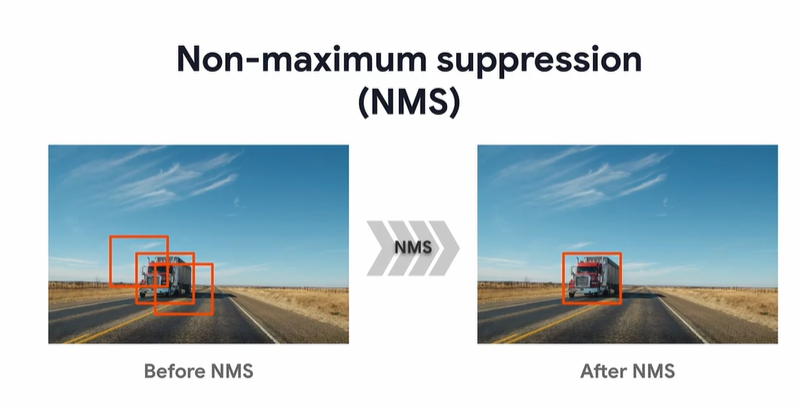
Another option is called selective search, which is abstracted and simulated here.

The algorithm makes a number of proposals about what might contain the particular object of interest.

Then when an object of interest is detected, the groups where it's detected are merged until we have a single group that should bound our object pretty well.

Now this can be quite a slow approach, but in general it does work quite well.

# Non-Maximum Supression:



Going back to the sliding window technique, you'll end up with multiple boxes that are detecting the same object. So, notice that there are three sliding windows that picked up the truck image, but there's just one truck not three

The sliding window in the middle likely has the most overlap with the true label

of the bounding box.

Remember that one way to evaluate how close the predicted bounding box is to the true label Is called intersection over union.

If we pick the sliding window that has the highest intersection of A union, we can choose that to be the bounding box and ignore the other sliding windows.

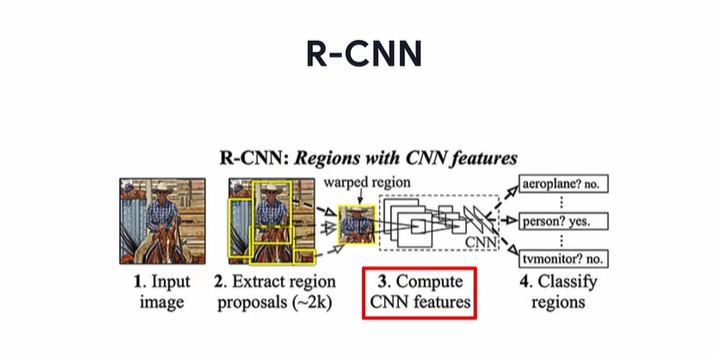
This technique of selecting the best bounding box based on the highest IOU is called non maximum suppression.

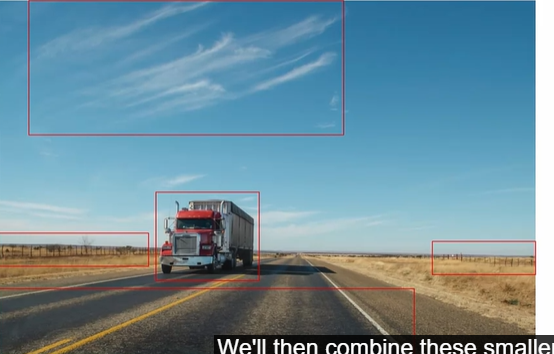
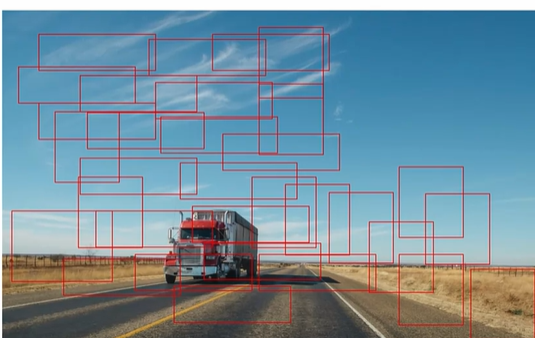
You can think of **non maximum suppression** as suppressing or ignoring the sliding windows that don't have the maximum IOU and keeping just the window with the maximum IOU, non-maximum suppression can be used during training to optimize and tweak the sliding windows that you use as the bounding box.

For all of these methodologies, we can divide object recognition down to a two stage process. The first is to propose a region, be it a sliding window or something else.

The second then identifies a classifies the object within that region. You'll see this pattern in many of the common architectures that are used by object detection, models.

# RCNN





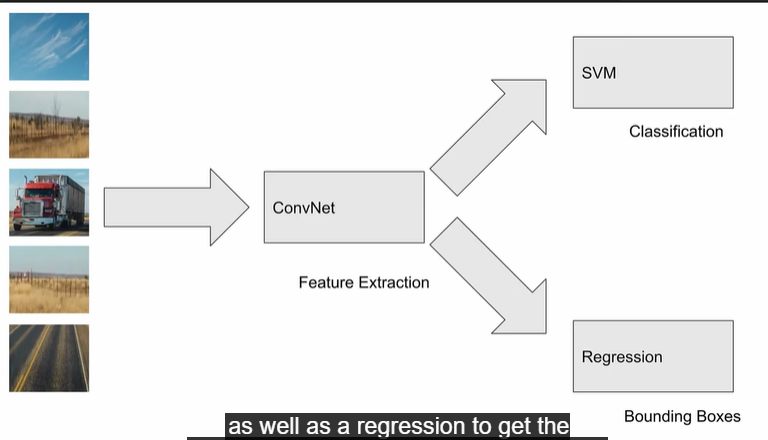
So, to use the truck image as an example, the R-CNN will start by using selective

search to identify small regions of interest in the image.

We'll then combine these smaller regions based on items within them,

color similarity, texture, sizing, shapes, all that kind of thing,

to get less regions that are larger, and these will be the final region proposals.

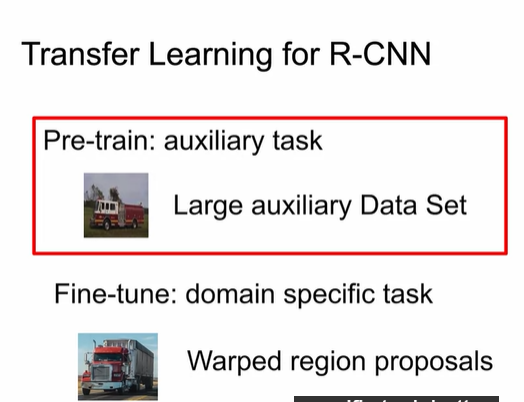


These region proposals are then extracted from the image,

they're walked into the size expected by the alex net convolutional layers.

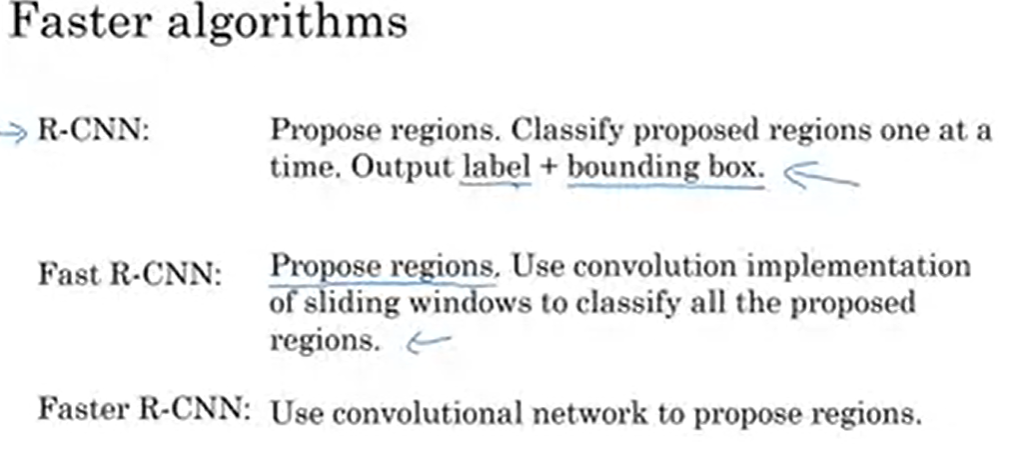
They're fed into these for Feature Extraction, and then fed into a classification layer, which in this algorithm is a scalable vector machine or SVM to get the labels for the image,

as well as a regression to get the bounding boxes for the areas of interest.



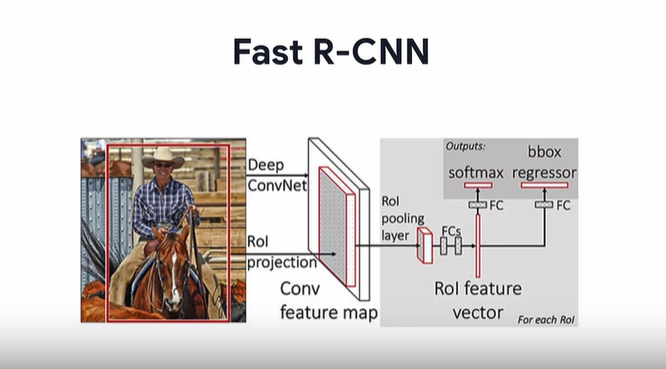
Transfer learning is used to pre train the CNN section of the R-CNN model, and then fine tune that model to this specific task. First, the researchers identified a large auxilary area set for pre training. And the word auxiliary here means to provide additional help. So the pre training auxiliary task is helping the model to perform its domain specific task better. An interesting thing to note is that the images in the auxiliary Data Set are not warped, like the region proposals used in this object detection task.These images also don't have bounding box labels, but the auxiliary data can still be used for pre-training, even when it's different than the data that's used for the desired domain specific task. The auxiliary data can help the model learn generally useful feature extraction,if it's a large data set. So pre-training is normally performed on very large data sets,even if there are different classes or different formats than the actual final task that we want to perform.

After pre-training, the CNN layers are fine tuned on the domain specific task, which is the object detection task that we're trying to solve. The researchers fine tune the model on warped region proposals. And this is a smaller data set, but specific to the task that we're trying to perform with object detection.



# Fast RCNN

Ross Girshick proposed an updated architecture called Fast R-CNN to improve this. The speed and memory issues were resolved to some extent in Fast R-CNN with the removal of the expensive Selective Search Algorithm and some exciting architectural changes.



First an image is fed to the network along with a set of region proposals. This right box on this image is a visual example of a region proposal for this image. Note the difference between Fast R-CNN and the original R-CNN. Instead of using Selective Search to generate several region proposals for each image, Fast R-CNN expects these region proposals as inputs and it doesn't generate them itself.

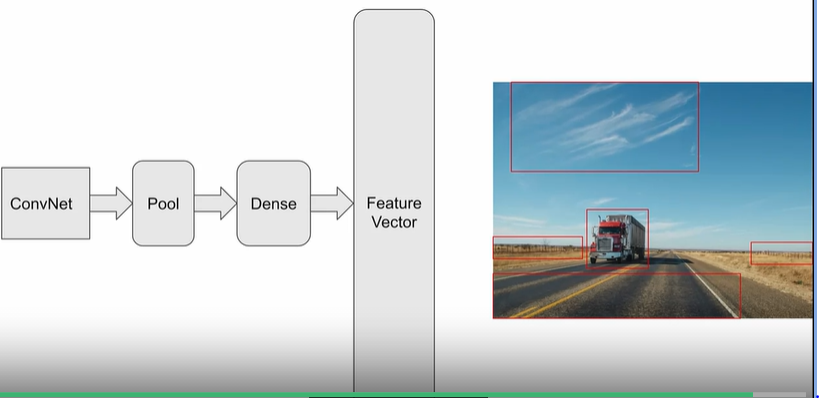
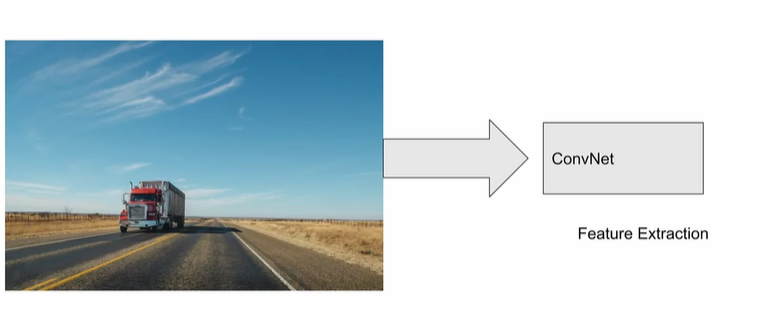
The CNN processes the image and outputs a set of features which we call the feature map. It's called the feature map because the detected features are stored relative to where they're detected in the original image. For instance, if it's a heart and a horse's mouth that are detected, the heart feature are stored near the top of the map and the horse's mouth is stored near the bottom, similar to where they were in the original image.

The next step is to use the input region proposals and extract the region of interest from this feature map and create what we'll call a region proposal feature map, one for each proposed region. This process is called the region of interest projection.

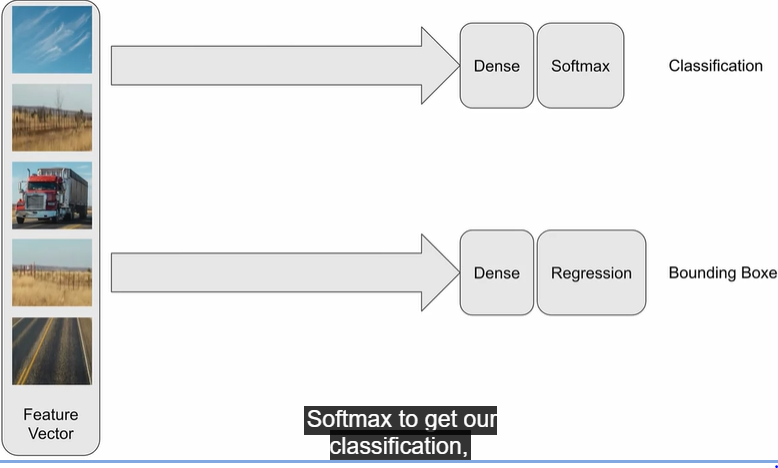
The difference between the first feature map that we mentioned and this region proposal feature map is that the original feature map is for the entire image, whereas the region proposal feature map is for the specific subsections of the proposed region of the image.

We'll then down-sample this feature map with the help of a region of interest pooling layer to get a fixed length feature map of a consistent height and width. This means that regardless of the dimensions of each proposed region, which may vary in size, the fixed length feature map is consistently the same size.

To make this map usable, we can then flatten the fixed size feature map into a one-dimensional vector, which we'll call the region of interest feature vector, and we'll create this region of interest feature vector using a few fully connected layers. We can then use this region of interest feature vector to generate two outputs. The first output uses a fully-connected layer, followed by Softmax in order to classify the image. The second output uses a fully-connected layer and irregular regression outputs to define the size and location of the bounding box for that classified objects.

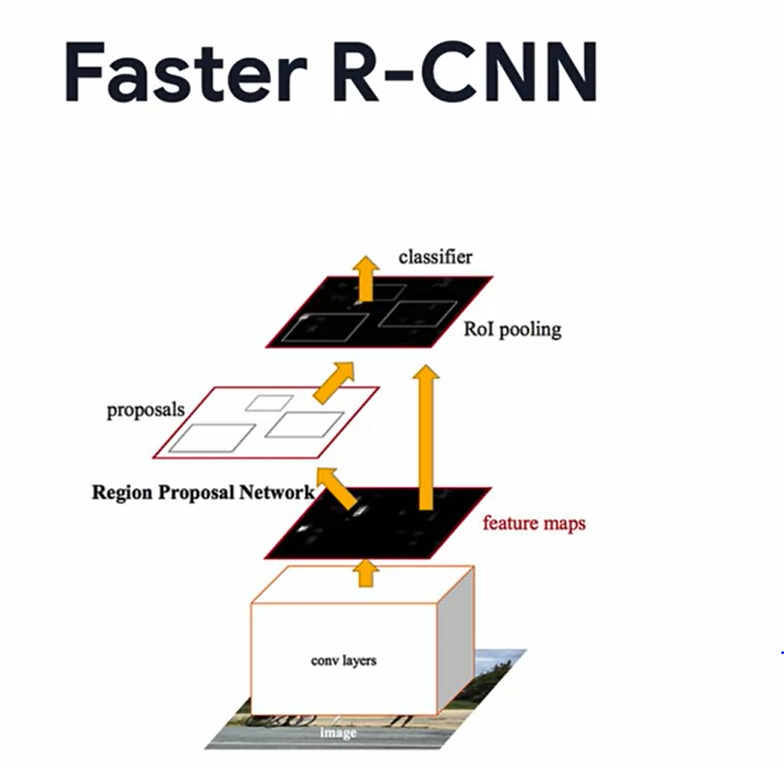


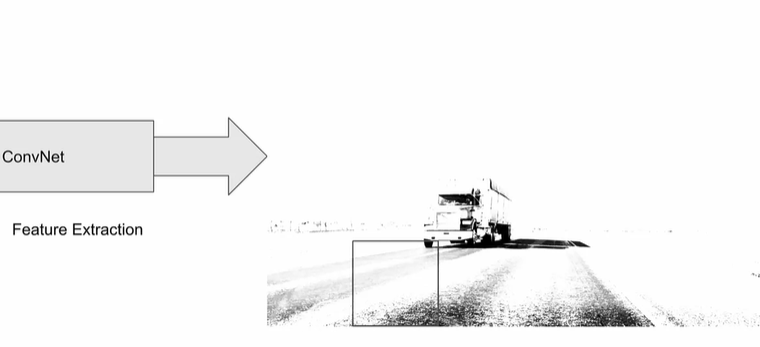
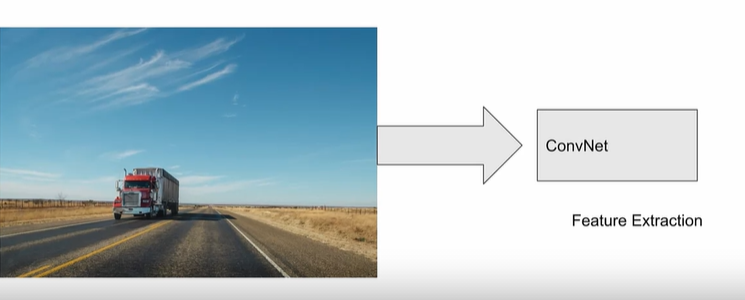
In this case, no selective search is done to find the regions of interest. Instead of ConvNet is used and its filters determined those regions of interests. The entire image, and not subsets, is passed into the This saves the expensive Selective Search Stamp. The ConvNet trained on finding features can then give us a feature map of the image. The feature map outputs can then be pooled and fed through a fully-connected dense layer to get a feature vector representing our regions of interest within the image.

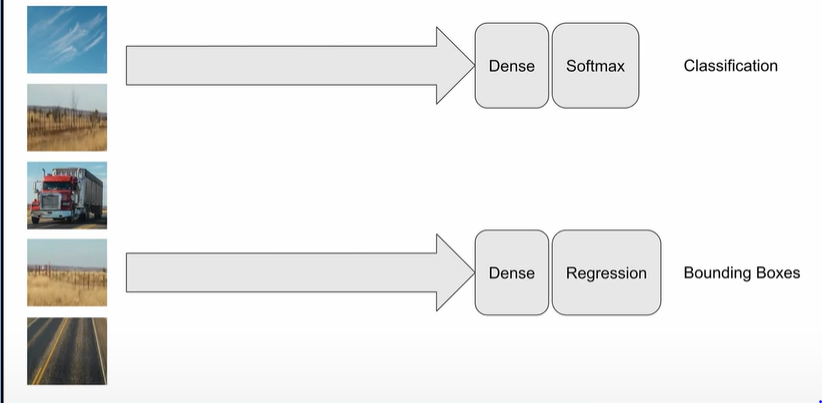
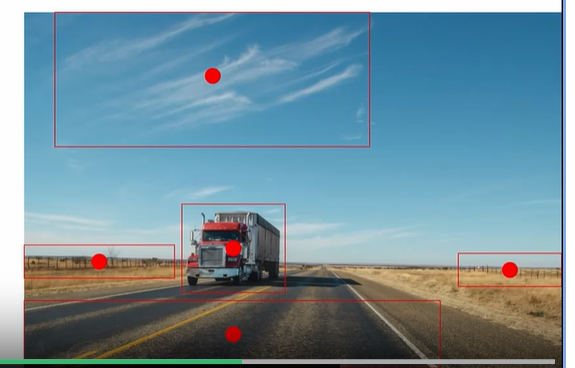


The feature vector can then be classified through a fully connected layer with Softmax to get our classification, and another with Regression to get our bounding boxes.

# Faster RCNN:



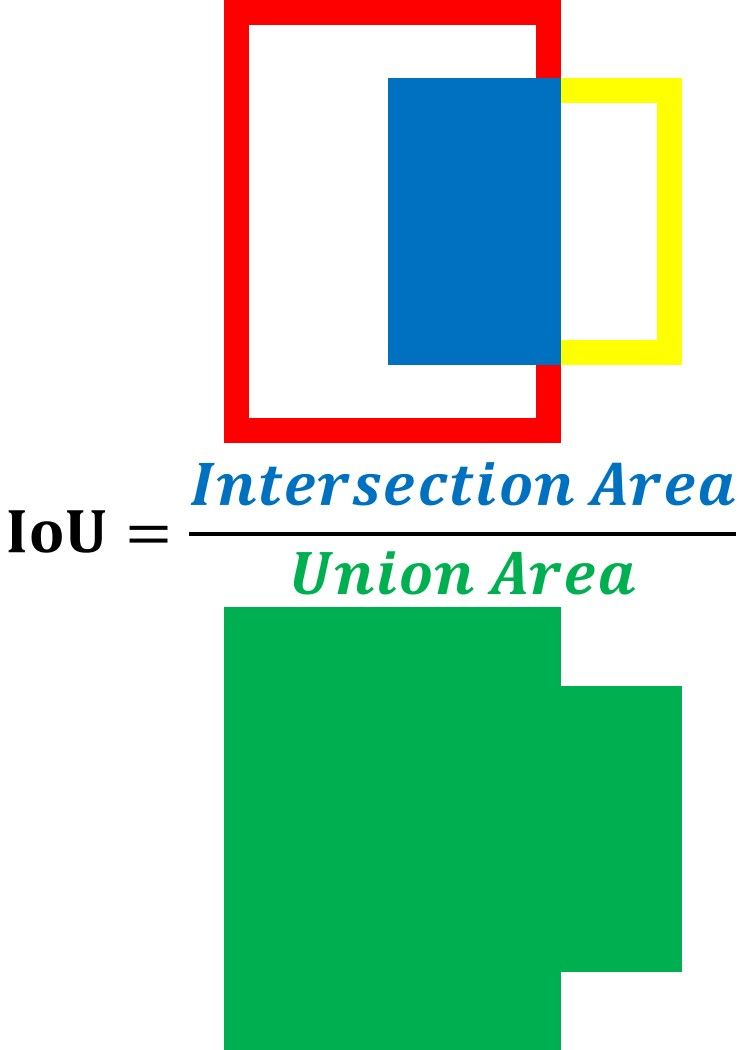


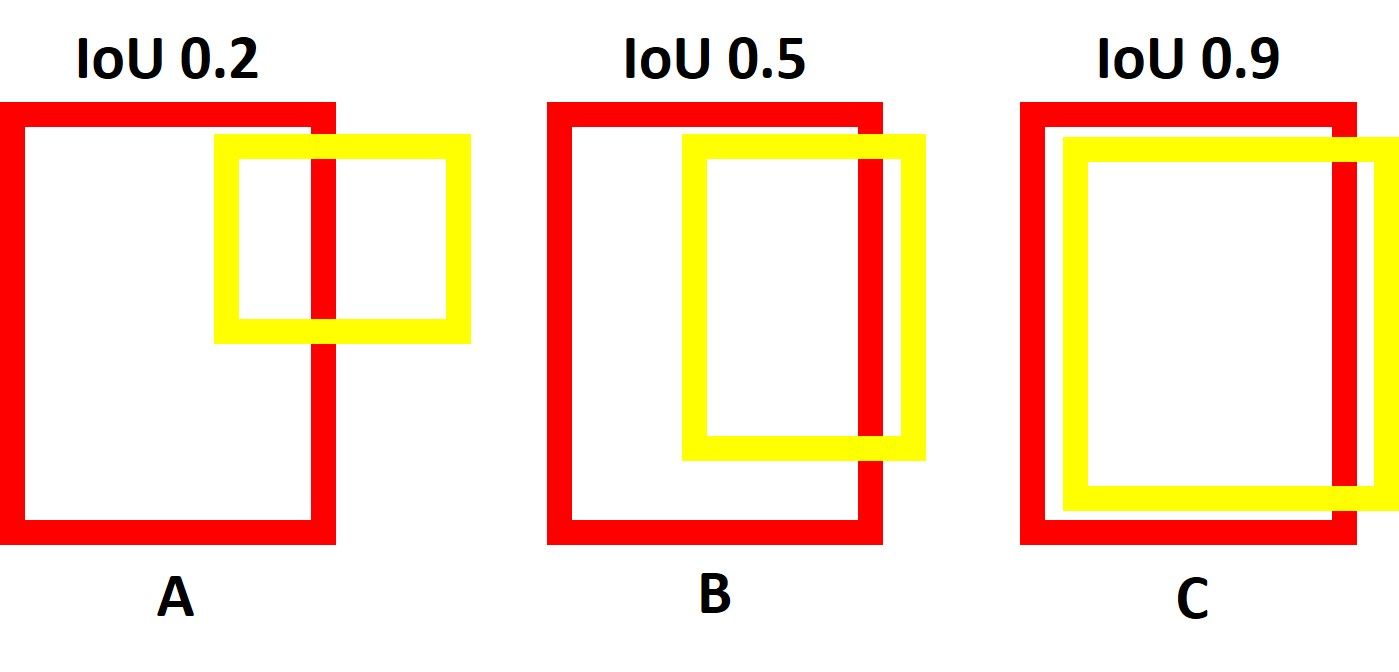


Faster R-CNN is an extension to fast our CNN with an addition of a region proposal network to propose regions of interest in the region proposal feature map. A region proposal network RPN for short, is a fully convolutional network. And this is a network that just uses convolutions are not dense layers. So we can simultaneously predict object bounds and object scores for each pixel. The RPN is designed to be trained end to end to generate high quality region proposals, which are used by faster R-CNN for detection. The RPN makes object proposals possible employing anchors or priors, we'll dive into that next. So as before, with faster R-CNN the entire image is passed into a ConvNet. This feature map then has a sliding window go across it to find areas of interest and a new entity called a region proposal network is used with the data established here to find it create anchor boxes on the image. The center of the anchor box comes from the coordinates of the sliding window and the boundaries of the box come from the RPN, giving us a score that the boundaries of the box better fit the objects. The details of how the RPN works are a little bit beyond the scope of this course. But the bottom line is that it's a fast way of finding the areas of interest in the image.

The next step, of course, is just the same. Where the areas of interest are cropped and passed to a classification and regression layer to get our labels in our bounding boxes.

# Evaluation

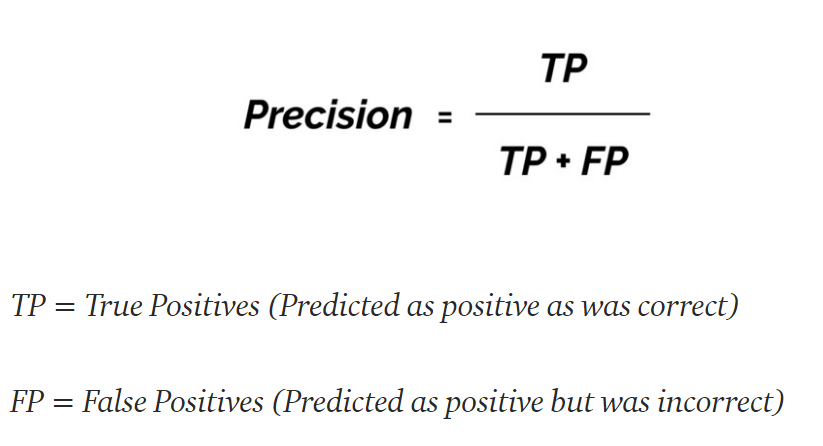




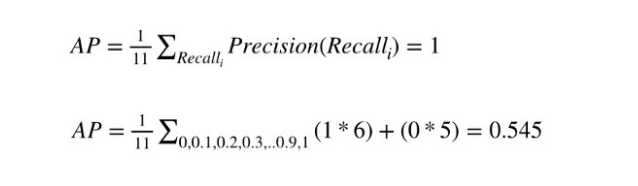
**Recall = (True Positive)/ (True Positive + False Negative)**

False Negative: A result that appears negative when it should not.

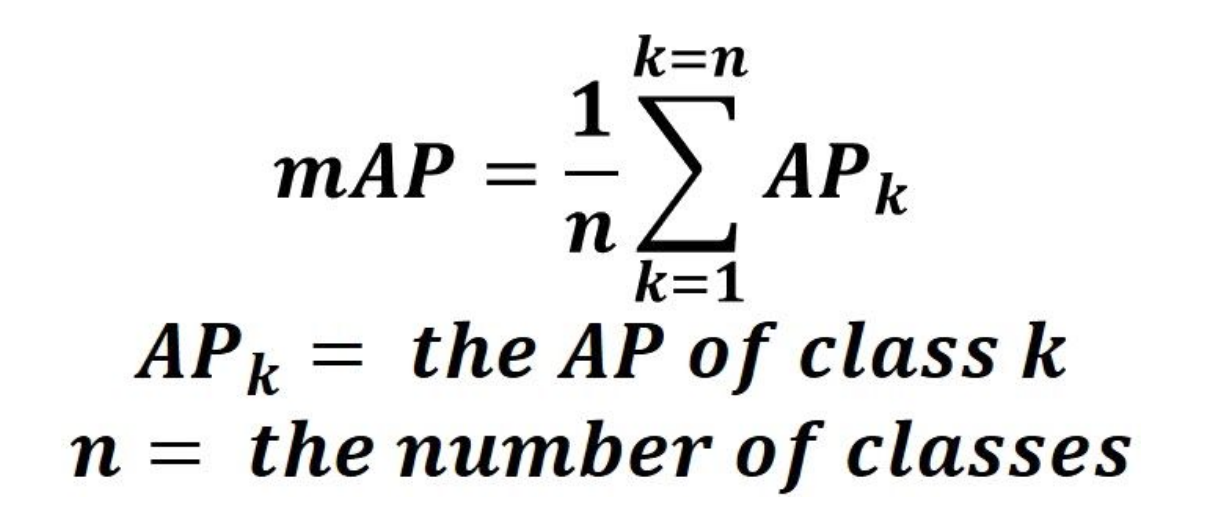
True Positive: A result is detected and it is present



If you have a precision score of close to 1.0 then there is a high likelihood that whatever the classifier predicts as a positive detection is in fact a correct prediction. Recall measures the “false negative rate” or the ratio of true object detections to the total number of objects in the data set. If you have a recall score close to 1.0 then almost all objects that are in your dataset will be positively detected by the model.



If we divide it by number of classes we will get mAP.



# References:

https://www.coursera.org/learn/advanced-computer-vision-with-tensorflow/supplement/tzRMX/reference-r-cnn-fast-r-cnn

# Object detection API

<https://colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/tf2_object_detection.ipynb#scrollTo=h9GtBUD3nlTV>

or refer to object\_detection\_inference\_on\_tf2\_and\_tfhub notebook

Must see the videos and read retina net research paper