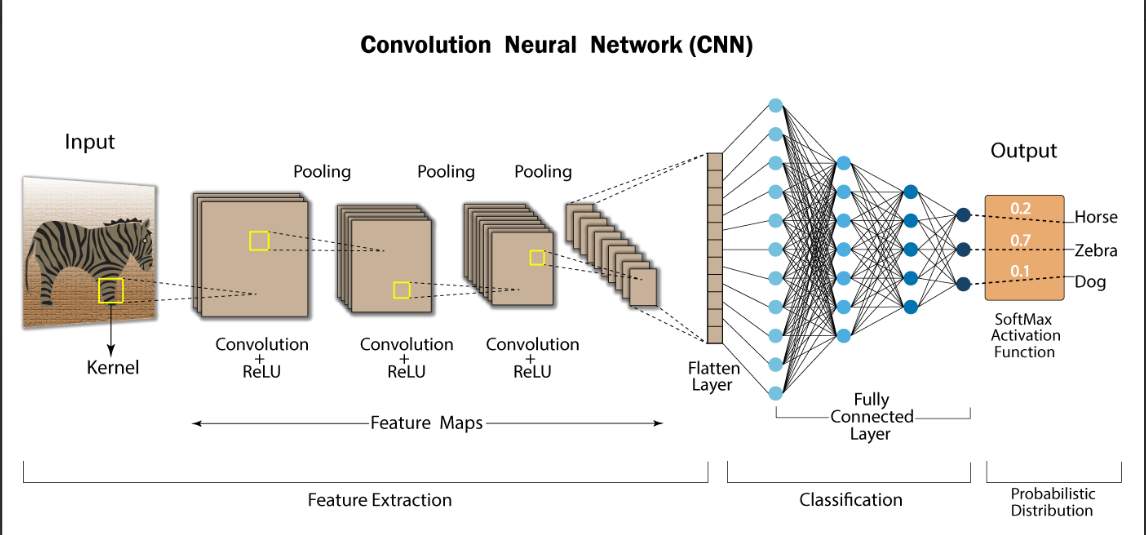
Faster R-CNN with Region Proposal Refinement

1. **Introduction**

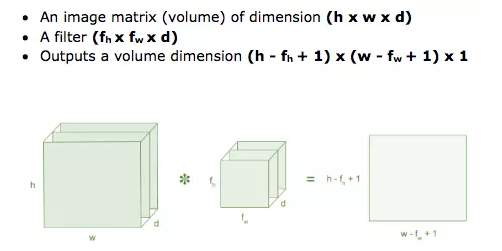
We are probably too familiar with the term Computer Vision, it was emerged recently on a few years. Some problems that you may hear or know about are MNIST classification, face mask detection, hand written detection, … or the latest idea about a model tracking student to detect cheating. So almost Computer Vision tasks is help to detection something to make a decision. Before we dive in our topic research, I will introduce a little bit about our main algorithm that we use in this topic.

1. **Relevent Knowledge**

**CNN**



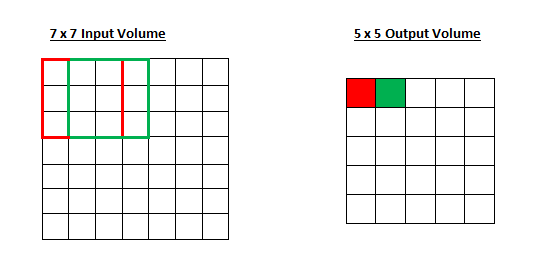
Each input of each image is an array of pixels and depends on the image resolution. Based on the image resolution, the computer will see H x W x D (height x width x depth)



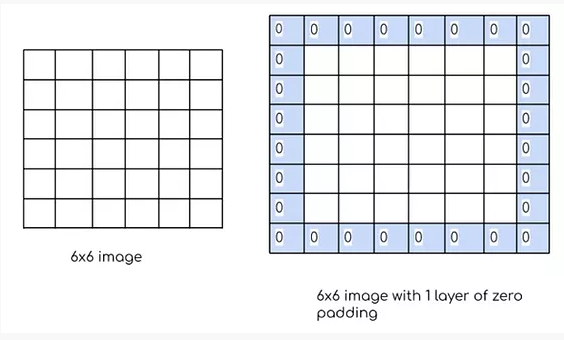
- Which is important when multiplying convolution in CNN:

+ Filter size: when the model is finished training, if the accuracy rate is still low, increase the value instead of 3x3 to 5x5 or 7x7

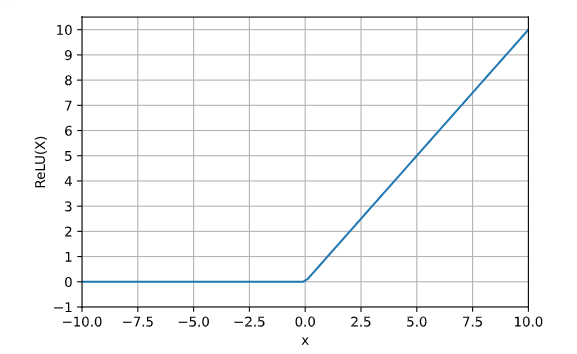
+ Stride



+ Padding



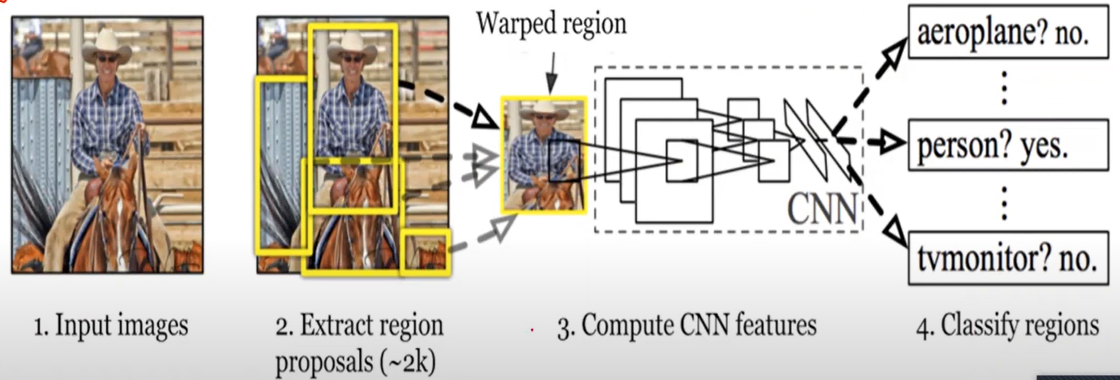
* Output is “feature map”
  + ReLU: Lọc các giá trị < 0





* Helps increase accuracy after each iteration Conv + ReLU and Max Pooling player
* Flatten layer: the set of pixels or points from the input of the image to after the last iteration của Conv + ReLU and Max Pooling player
* FL combined with FCL to create weights or bias
* Finally, it use SoftMax function to predict labels of features in image
* The disadvantage of using CNN for image recognition is that it can only recognize one object, but in fact, there is not only one object in an image, so the CNN problem cannot be used.

**R-CNN**

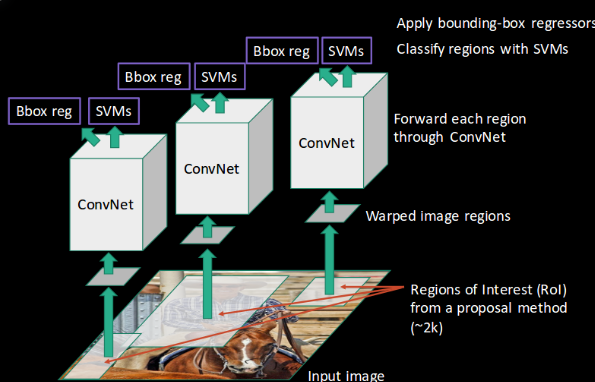


Selective Search algorithm:

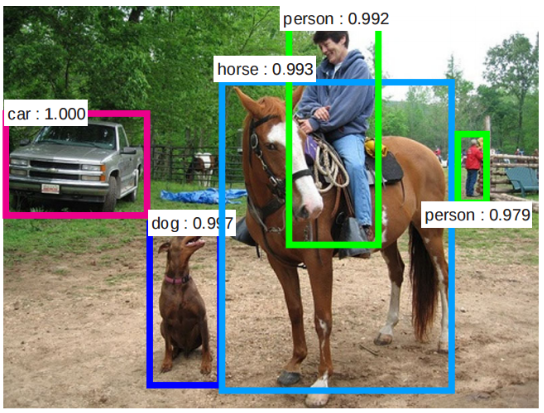
* + This is widely used in R-CNN and Fast R-CNN. Selective Search aim to solve the problem object location
  + This use sliding window of different size to locate objects in the image. This approach is required very much computation because we need to search a thousands of window even on a small image (about 2000 regions). Selective Search choose some regions proposal base on these characteristics such as color similarity, size and shape, …

Region- Convolutional Neural Network:

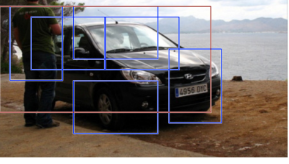
* + Firstly, extracts a bouding box of regions from the image using selective search, and then checks if any of these boxes contain an object



* + When, extracted a bounding box on image



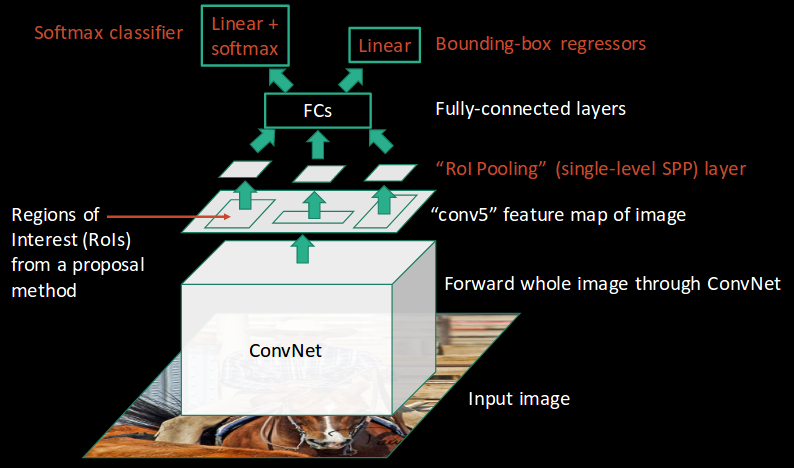
* + Seconds, warp image regions to resize image and put into CNN model to extract specific features and object detection.
  + Finally, these features are then used to detect objects



* + Weakness: R-CNN is slow because of these multiple steps involved in the process

Image 🡪 selective search -> regions proposal + wrap image 🡪 CNN model 🡪 SVM model -> classify regions + bouding-box

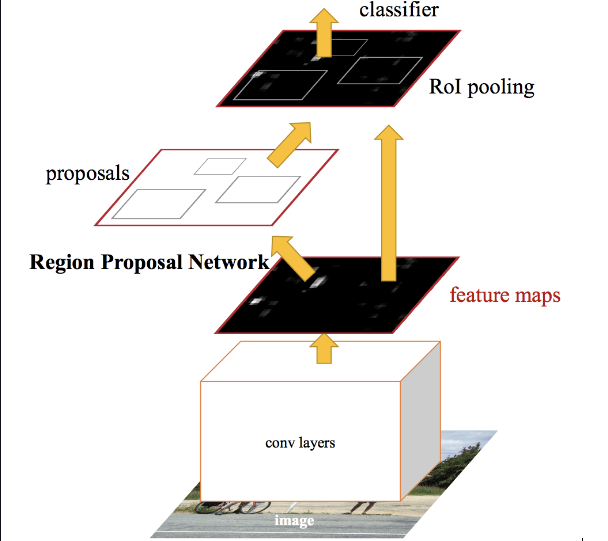
**Fast R-CNN**

****

* + It is a derivative idea of R-CNN. It still use selective search to extract regions proposal but in other way: It will passes the entire image to ConvNet which generates regions of Interes (instead of passing the extracted region from the image).
  + Then it use a single model which extracts features from the regions, classifies them into different classes and return the bouding box

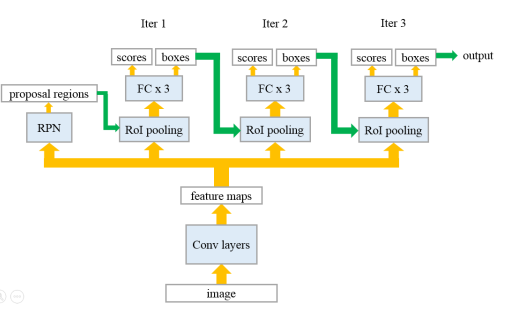
Image -> ConvNet 🡪 RoIs -> selective search 🡪 regions proposal -> RoI pooling player 🡪 Regions with same size -> FCs 🡪 bouding-box + softmax classifier

**Faster R-CNN**



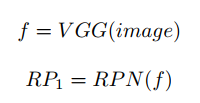
* + Firstly, we employ a pre-trained deep convolutional neural network such as VGG to extract feature maps. Then, Faster R-CNN replace selective search with a Region Proposal Network to detect the regions that might contain object in the feature maps
  + Seconds, the network employ a RoI pooling layer to crop and resize the the feature maps according to these region proposals
  + Finally, these maps are classified and the bouding-boxes are predicted

**Iterative Region Proposal Refinement Model**

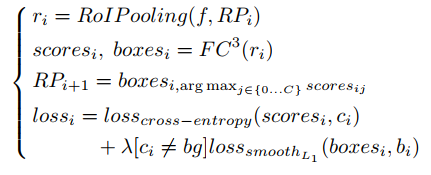
****

* + Refine region extraction through loop to get output including best scores and boxes
  + Iteract of RoI pooling layer to crop and resize new feature maps for each proposal, and use a three-layer fully connected network to get the final class scores and bounding box regression

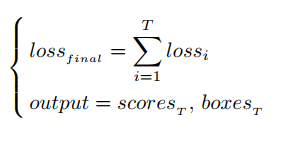
We have:



Then suppose we set iteration number = *T*, then for each iteration *i*, the model can be represented as for *i* =1*;* 2*; …T*

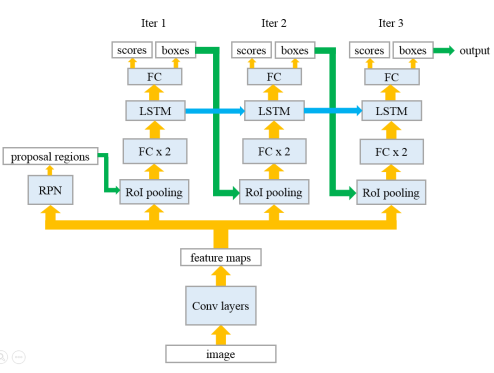


We will get a *lossi* in each iteration



* + Issue of the iterative refinement model is the gradient of loss can’t not be backpropagation from the later iteration to the earlier iterations
  + The above image results show that the gradient of each loss in iterative  
    model can only be propagated downward (yellow arrows) but cannot be propagated to the left (green arrows)

**LSTM Region Proposal Refinement Model**



* + To be can pass useful information to later time steps and also propagate the gradient backward to the previous time steps
  + One benefit of adding a LSTM layer is that the hidden state of previous iteration can contain information that is useful to improve classification and bounding box regression results in current iteration.
  + Another benefit is that can now backpropagate the gradient of loss from the later iterations to the earlier ones.

We have:

