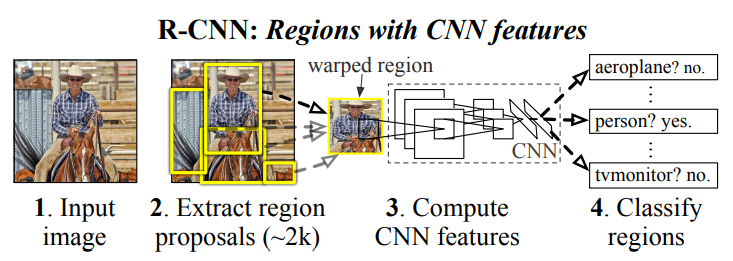
**Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)**

**(Regions with CNN, R-CNN)**

*Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik*

* Source: <https://arxiv.org/pdf/1311.2524.pdf>
* Improves mAP (Mean Average Precision) by more than 30% relative to previous best result on VOC 2012
* Earlier approaches: SIFT and HOG
* This paper is the first to propose CNN for object detection and shows that CNN can be very effective for object detection
* Detection requires localization of objects within the image.
  + One approach is to treat frame localization as a regression problem, but this doesn’t work well in practice.
  + Second approach is to use a sliding-window detector. If we use large stride, the localization is not that accurate. But if we use a small stride, no. of computations blows up tremendously and becomes infeasible.
  + The paper proposes another approach:

It uses a region selection algorithm (Selective Search) that finds 2000 regions in each input image that may contain objects. These regions could be of various shapes, so they need to be rescaled so that they can be fed into a CNN. The paper uses Affine Image Warping to convert all regions to same shape. Each such region is then fed to a CNN to get a feature map, and this feature map is classified using class specific linear SVMs.



* Since this method combines region proposal with CNN, it is called R-CNN (Regions with CNN).
* The second contribution of this paper is that it uses a pre-trained model on an auxiliary dataset, and then fine-tunes it for object detection. This shows that transfer learning is effective.
* Since R-CNN operates on regions, it can be used for semantic segmentation as well.
* The overall solution to object detection has 3 modules:
  + The first module is the Selective Search algo that generates 2000 category-independent regions per image. These regions are then warped to the required size. Since we are using AlexNet, required shape is 227\*227.

Note: <https://towardsdatascience.com/r-cnn-for-object-detection-a-technical-summary-9e7bfa8a557c>

**Selection Search algorithm proposes regions containing only one object.**

* + The second module is AlexNet that extracts 4096-length feature vector from each region (4096-length because **the last FC layer** in AlexNet contains 4096 neurons).
  + The third module consists of class-specific linear SVMs which take each extracted 4096-length feature vector and produce a probability/score for each class.

Given all scored regions, we then apply non-max suppression method for each class independently that rejects a region if it has an IoU overlap with another high scoring region.

* Two properties make this solution more efficient. First, all the CNN parameters are shared across all categories. Second, the feature vectors computed by CNN are low-dimensional compared to other common approaches.
* Training:
  + The CNN was pre-trained on a large auxiliary dataset using image-level annotations only (it didn’t have class labels).
  + For fine-tuning the network for our task, the output layer of AlexNet having 1000 units was replaced with a layer having N+1 units, where N is the no. of classes and extra 1 is for background. To adapt this pre-trained model for detection, the model was trained on warped region proposals using SGD.  
    (Note, this output layer having K+1 neurons is used only during pre-training CNN; actual class prediction happens using SVMs)
  + All region proposals having IoU >= 0.5 with the ground-truth box are treated as positive.
  + One linear SVM per class.
* The authors also used pre-trained VGG16 model in place of AlexNet. mAP increased from 58.5% to 66%, but forward pass using VGG16 took 7 times longer than forward pass using AlexNet.

(The paper uses the name O-Net to refer to VGG16 and T-Net to refer to AlexNet)

* The regions proposed by Selection Algorithm can be used as the bounding boxes. However, to improve accuracy further, a linear regression model is also used in the end.
* So, SVMs are used to classify objects in proposed regions, but linear regression is used to get the bounding boxes for the objects detected.
* So, the output from **the last FC layer** (just before the softmax layer) in the AlexNet is fed to class-specific SVMs and the output of **the last pooling layer** is passed to the linear regression model for bounding box prediction.
* Appendix C – shows bounding-box linear regressors’ details

We have class specific regressors. Once the SVMs give the output class for a proposed region, we use the class-specific regressor to predict the bounding box.

The input is a set of N pairs (Pi, Gi), where Pi = (Pi­­x, Piy, Piw, Pih), Pix and Piy specify the centre of the proposed region Pi and Piw and Pih specify its width and height.

Similarly, Gi = (Gi­­x, Giy, Giw, Gih) specifies the ground-truth bounding box for the proposed region Pi.

We want to learn functions and that specify the translations of the centre of the proposed region and and that specify log-space translations of the width and height of the proposed region.

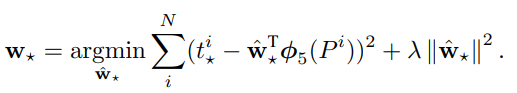
Once these functions are learnt, we can transform an input proposal P into a predicted ground-truth box using functions:

The output of the last pooling layer is fed to bounding-box regressors.

The functions , , , and in the above equations represent linear functions of the form

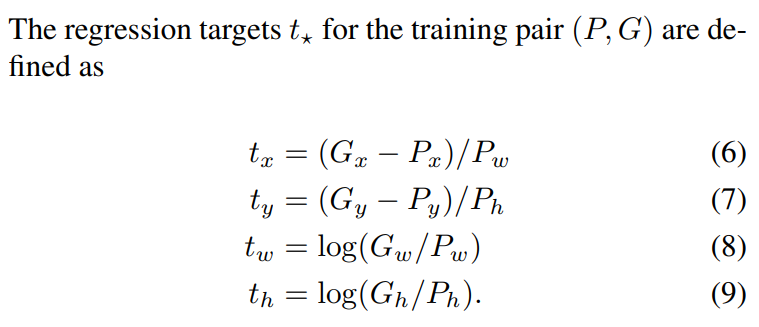
where WTx, WTy,WTh, and WTw are learnable parameters.

The cost function used for training:



(\* is used to refer to x, y, w, and h)

The target for a regressor is defined as: (below functions are obtained by rearranging terms in the above four equations)



Basically, we want regressors to predict bounding box relative to ground-truth box. So, we convert Gx, Gy, Gw, and Gh to tx, ty, tw, and th for training the linear regressors.

Similarly, during testing, in order to get the actual coordinates of the bounding box, we need to convert the regressor output, which is relative to ground-truth box.

<https://medium.com/@selfouly/r-cnn-3a9beddfd55a>

* Selection Search is not a trainable algorithm. We use the algorithm as it is to propose regions
* CNN, SVMs, and bounding box regressors are all trained individually.
* First a pre-trained CNN model (e.g. AlexNet) is taken. Then, its output softmax layer is replaced with another softmax layer having units as per the no. of classes in our dataset. Now, we train this model on the regions proposed by Selection Search algorithm.
* After training (i.e. fine-tuning) CNN, remove the softmax layer.
* Now, we train the SVMs. We pass the output of the CNN to the SVMs and train them.
* Finally, we train bounding box regressors.

To understand Selective Search refer below: (SKIPPED)

<https://learnopencv.com/selective-search-for-object-detection-cpp-python/>