* **Architecture Used**: Faster – RCNN
* R-CNN is the first step for Faster R-CNN. It uses search selective to find out the regions of interests (ROI) and passes them to a ConvNet.
* It tries to find out the areas that might be an object by combining similar pixels and textures into several rectangular boxes.
* The R-CNN paper uses 2,000 proposed areas (rectangular boxes) from search selective. Then, these 2,000 areas are passed to a pre-trained CNN model.
* Finally, the outputs (feature maps) are passed to SVM for classification. The regression between predicted bounding boxes (bboxes) and ground-truth bboxes are computed.

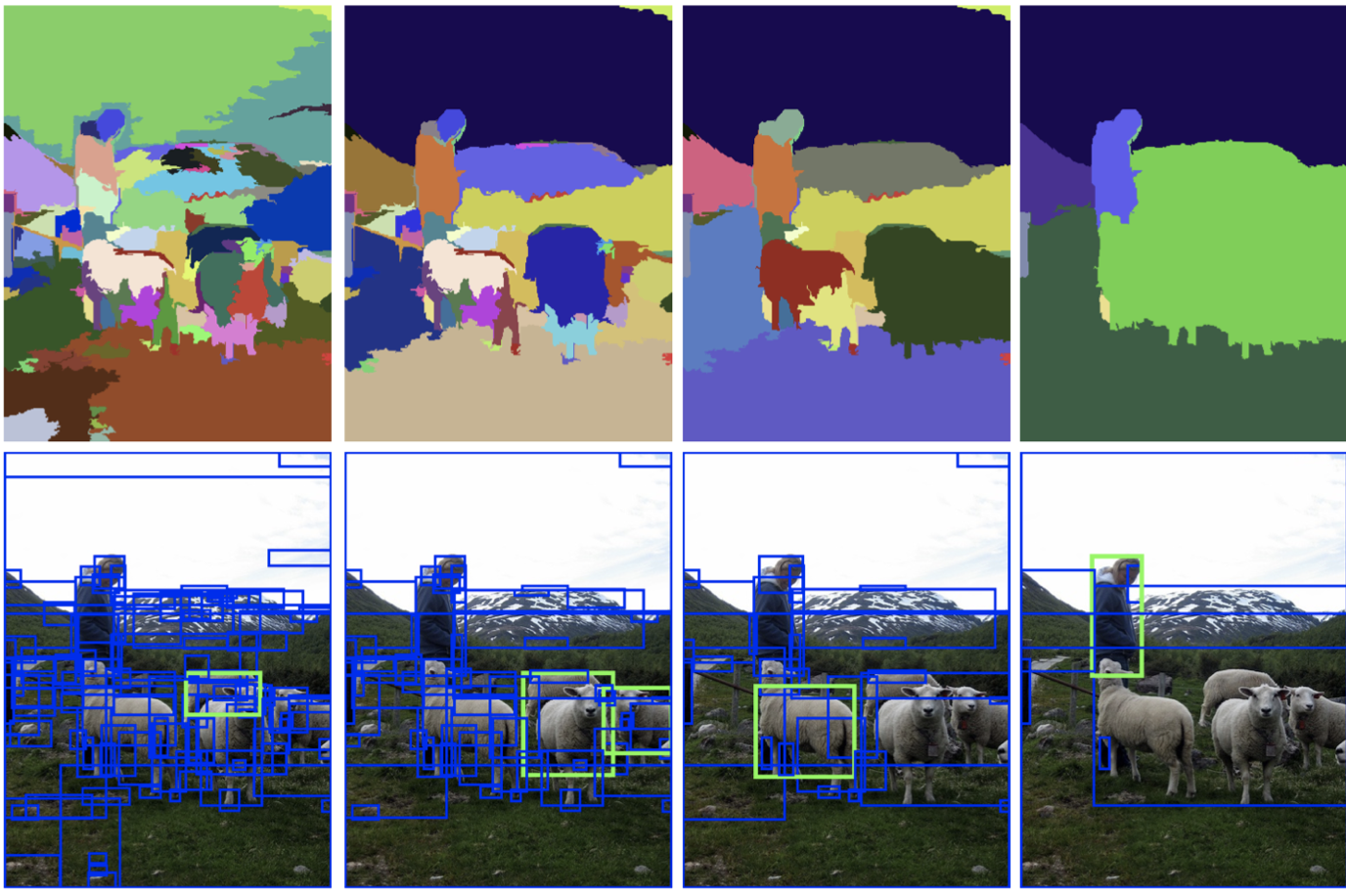


Figure: Example of Search Selective

* Faster R-CNN (Frcnn for short) makes further progress than Fast R-CNN. Search selective process is replaced by Region Proposal Network (RPN). As the name revealed, RPN is a network to propose regions. For instance, after getting the output feature map from a pre-trained model (VGG-16), if the input image has 600x800x3 dimensions, the output feature map would be 37x50x256 dimensions.

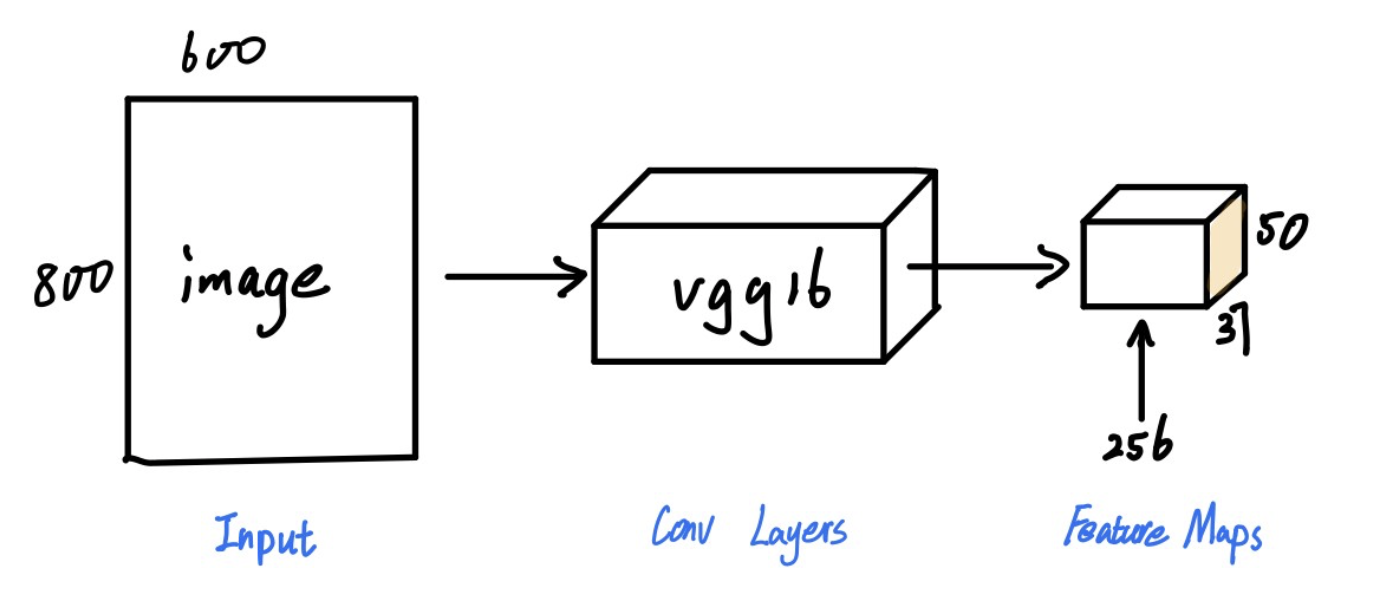
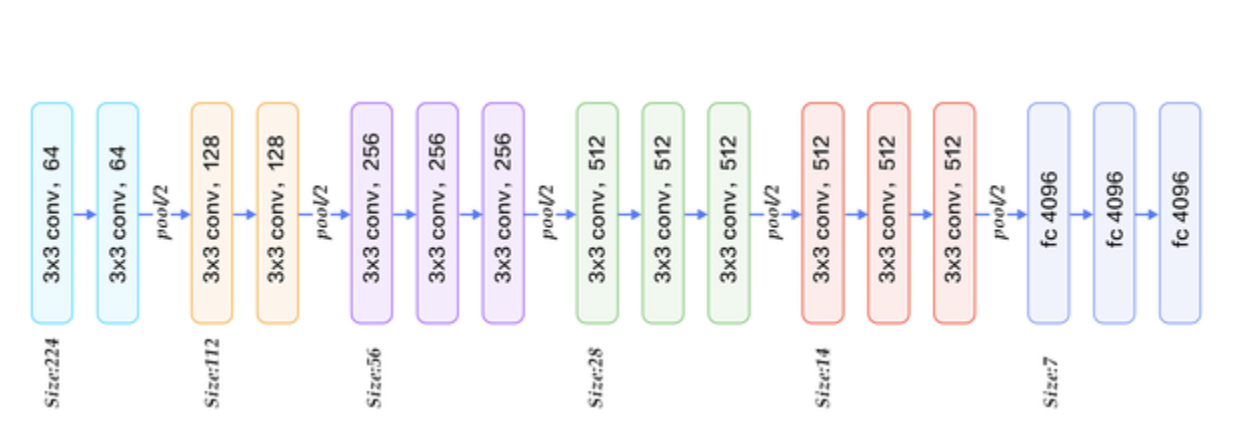
****

Figure: First Step of FRCNN

* **Code Explanation:**

****

* Rebuild the structure of VGG-16 and load pre-trained model (nn\_base)
* Prepared training data and training labels (get\_anchor\_gt) : The input data is from annotation.txt file which contains a bunch of images with their bounding boxes information. We need to use RPN method to create proposed bboxes.
* Calculated rpn for each image (calc\_rpn)
* Calculate region of interest from RPN (rpn\_to\_roi)
* **Parameters**
* The number of anchors is 9.
* Max number of non-max-suppression is 300.
* Number of RoI to process in the model is 4 (I haven’t tried larger size which might speed up the calculation, but more memory needed)
* Adam is used for optimisation and the learning rate is 1e-5. It might work different if we applied the original paper’s solution. They used a learning rate of 0.001 for 60k mini-batches, and 0.0001 for the next 20k mini-batches on the PASCAL VOC dataset.
* For images augmentation, I turn on the horizontal\_flips, vertical\_flips and 90-degree rotations.
* **Environment**
* Floyd Hub with Tesla K80 GPU acceleration for training.
* **Training time**
* The length of each epoch that I choose is 1000. Note that every batch only processes one image in here. The total number of epochs I trained is 114. Every epoch spends around 700 seconds under this environment which means that the total time for training is around 26 hours.
* **Result**
* There are two loss functions we applied to both the RPN model and Classifier model.
* RPN model has two output. One is for classifying whether it’s an object and the other one is for bounding boxes’ coordinates regression.
* From the figure below, we can see that it learned very fast at the first 20 epochs. Then, it became slower for classifier layer while the regression layer still keeps going down.

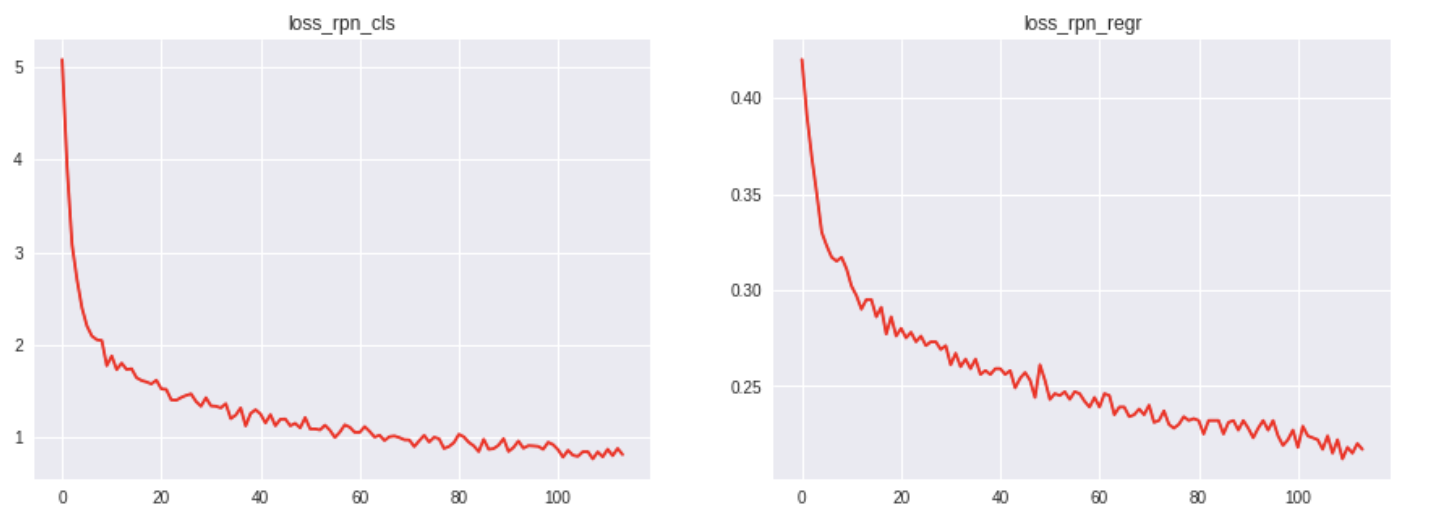


Figure: Epochs vs. Loss value for RPN model’s classifying output and bboxes regression output

* The reason for this might be that the accuracy is already high for the early stage of our training, but at the same time, the accuracy of bounding boxes’ coordinates is still low and needs more time to learn.
* The similar learning process is shown in Classifier model. Compared with the two plots for bbox’s regression, they show a similar tendency and even similar loss value.

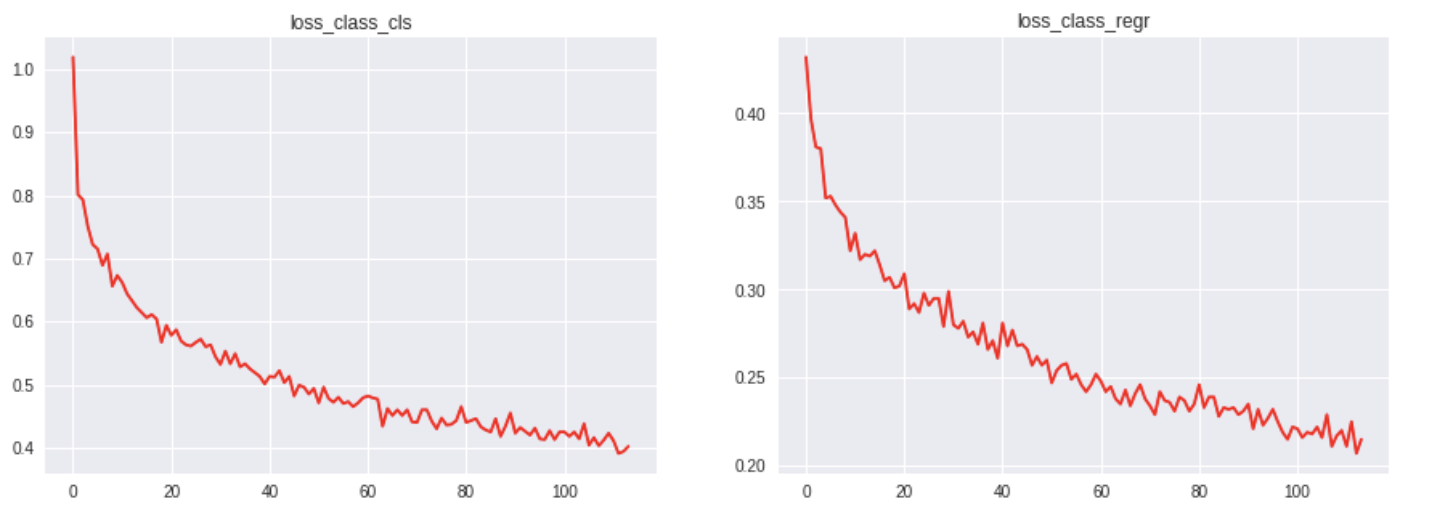


Figure: Epochs vs. Loss value for Classifier model’s classifying output and bbox’s regression output

* I think it’s because they are predicting the quite similar value with a little difference of their layer structure. Compared with two plots for classifying, we can see that predicting object is easier than predicting the class name of a bbox.
* **Total Loss Calculation:**

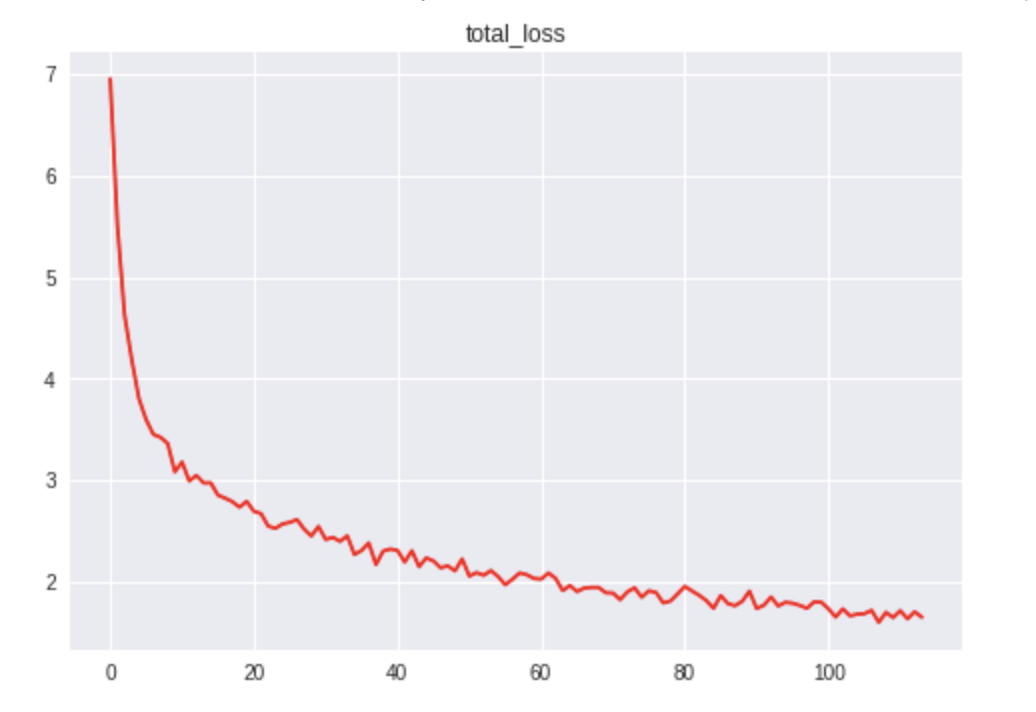


Figure: Epochs vs. Total loss for two models

This total loss is the sum of four losses above. It has a decreasing tendency. However, the mAP (mean average precision) doesn’t increase as the loss decreases. The mAP is 0.61 when the number of epochs is 60. The mAP is 0.58 when the number of epochs is 87. The mAP is 0.49 when the number of epochs is 114.

**I think this is because of the small number of training images which leads to overfitting of the model.**