1. **R-CNN**

R-CNN is short for "Region-based Convolutional Neural Networks" [1]. The main idea is composed of two steps. First, using selective search, it identifies a manageable number of bounding-box object region candidates ("region of interest" or "RoI"). And then it extracts CNN features from each region independently for classification. The architecture of R-CNN is shown in Figure 1.

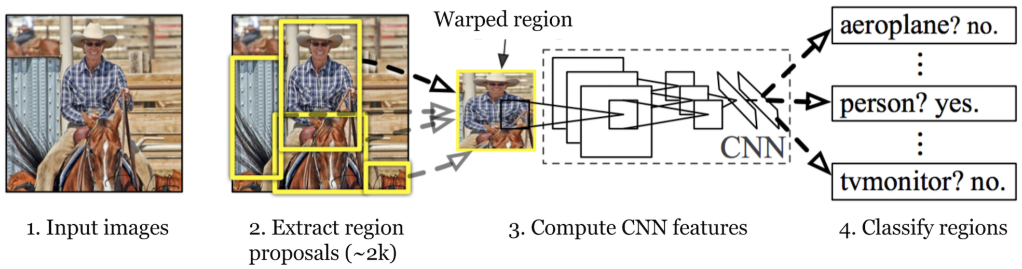

Figure 1. The architecture of R-CNN.

How R-CNN works can be summarized as follows:

- Pre-train a CNN network on image classification tasks; for example, VGG or ResNet trained on ImageNet dataset. The classification task involves N classes.
- Propose category-independent regions of interest by selective search (2k candidates per image). Those regions may contain target objects and they are of different sizes.
- Region candidates are warped to have a fixed size as required by CNN.
- Continue fine-tuning the CNN on warped proposal regions for $K + 1$ classes; The additional one class refers to the background (no object of interest). In the fine-tuning stage, we should use a much smaller learning rate and the mini-batch oversamples the positive cases because most proposed regions are just background.
- Given every image region, one forward propagation through the CNN generates a feature vector. This feature vector is then consumed by a binary SVM trained for each class independently. The positive samples are proposed regions with IoU (intersection over union) overlap threshold $>= 0.3$, and negative samples are irrelevant others.
- To reduce the localization errors, a regression model is trained to correct the predicted detection window on bounding box correction offset using CNN features.

2. **Fast R-CNN**

To make R-CNN faster, [2] improved the training procedure by unifying three independent models into one jointly trained framework and increasing shared computation results, named Fast R-CNN. Instead of extracting CNN feature vectors independently for each region proposal, this model aggregates them into one CNN forward pass over the entire image and the region proposals share this feature matrix. Then the same feature matrix is branched out to be used for learning the object classifier and the bounding-box regressor. In conclusion, computation sharing speeds up R-CNN. The architecture of Fast R-CNN is shown in Figure 2.
How Fast R-CNN works is summarized as follows; many steps are same as in R-CNN:

- First, pre-train a convolutional neural network on image classification tasks.
- Propose regions by selective search (2k candidates per image).
- Alter the pre-trained CNN:

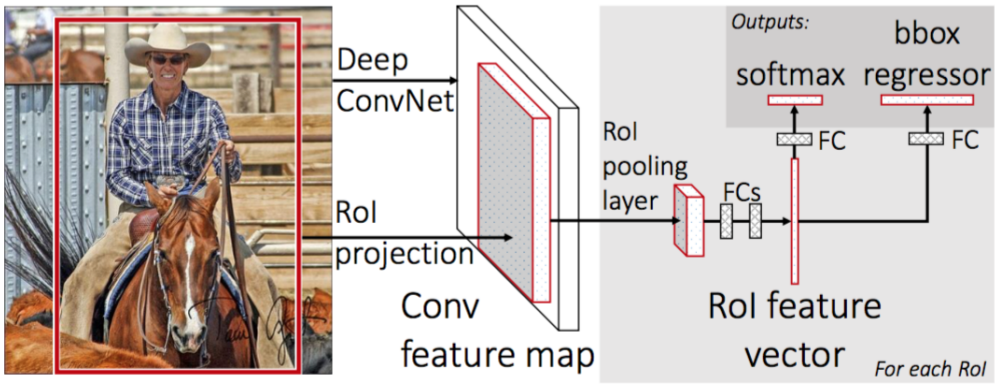

FIGURE 2. The architecture of Fast R-CNN

  - Replace the last max pooling layer of the pre-trained CNN with a RoI pooling layer. The RoI pooling layer outputs fixed-length feature vectors of region proposals. Sharing the CNN computation makes a lot of sense, as many region proposals of the same images are highly overlapped.
  - Replace the last fully connected layer and the last softmax layer (K classes) with a fully connected layer and softmax over K + 1 classes.
- Finally the model branches into two output layers:
  - A softmax estimator of K + 1 classes (same as in R-CNN, +1 is the "background" class), outputting a discrete probability distribution per RoI.
  - A bounding-box regression model which predicts offsets relative to the original RoI for each of K classes.

3. **Faster R-CNN**

An intuitive speedup solution is to integrate the region proposal algorithm into the CNN model. Faster R-CNN [3] is doing exactly this: construct a single, unified model composed of RPN (region proposal network) and fast R-CNN with shared convolutional feature layers. The architecture of Faster R-CNN is shown in Figure 3.

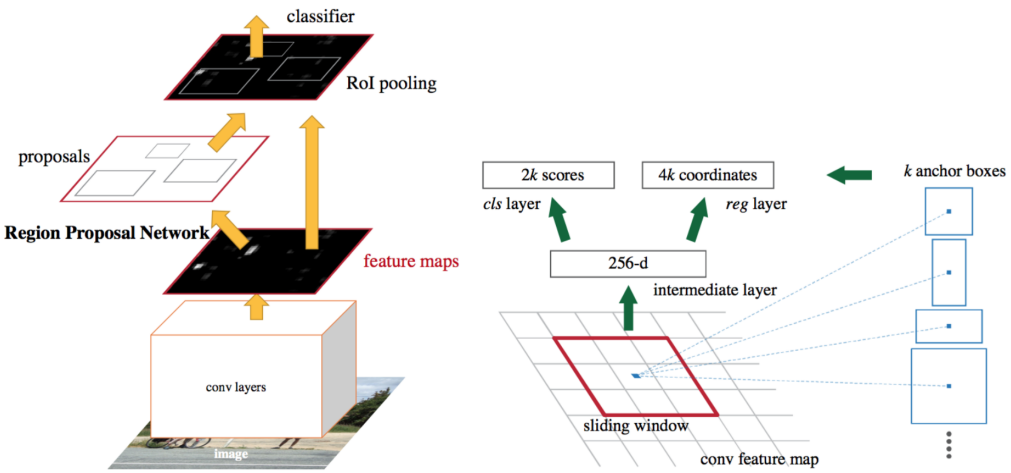

FIGURE 3. The architecture of Faster R-CNN

How Faster R-CNN works can be summarized as follows:

- Pre-train a CNN network on image classification tasks.

- Fine-tune the RPN (region proposal network) end-to-end for the region proposal task, which is initialized by the pre-train image classifier. Positive samples have IoU (intersection-over-union) > 0.7, while negative samples have $IoU < 0.3$.
  - Slide a small $n \times n$ spatial window over the conv feature map of the entire image.
  - At the center of each sliding window, we predict multiple regions of various scales and ratios simultaneously. An anchor is a combination of (sliding window center, scale, ratio). For example, 3 scales + 3 ratios $=>$ $k = 9$ anchors at each sliding position.
- Train a Fast R-CNN object detection model using the proposals generated by the current RPN
- Then use the Fast R-CNN network to initialize RPN training. While keeping the shared convolutional layers, only fine-tune the RPN-specific layers. At this stage, RPN and the detection network have shared convolutional layers!
- Finally fine-tune the unique layers of Fast R-CNN
- We can train RPN and Fast R-CNN alternatively if needed.

4. **Mask R-CNN**

Mask R-CNN [4] extends Faster R-CNN to pixel-level image segmentation. The key point is to decouple the classification and the pixel-level mask prediction tasks. Based on the framework of Faster R-CNN, it added a third branch for predicting an object mask in parallel with the existing branches for classification and localization, as shown in Figure 4. The mask branch is a small fully-connected network applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.

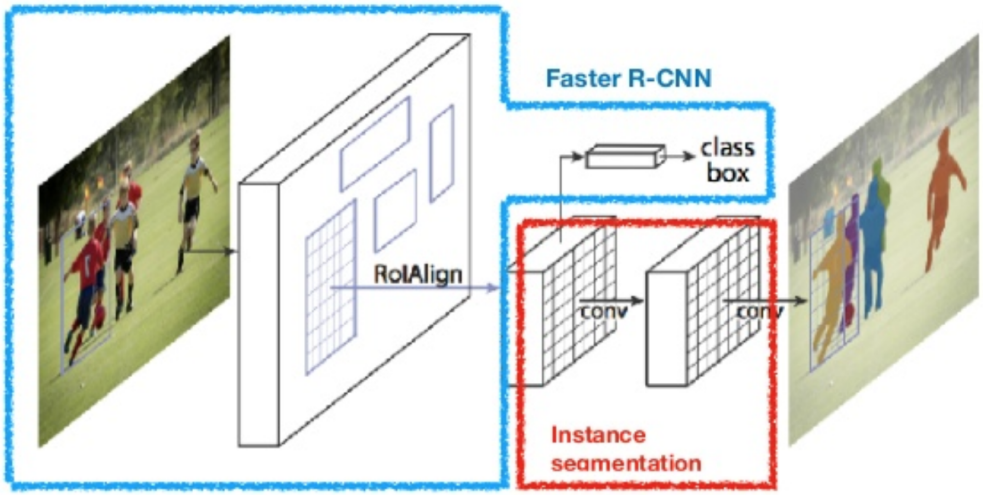

FIGURE 4. Mask R-CNN is Faster R-CNN model with image segmentation

Because pixel-level segmentation requires much more fine-grained alignment than bounding boxes, mask R-CNN improves the RoI pooling layer (named "RoIAlign layer") so that RoI can be better and more precisely mapped to the regions of the original image.
The RoIAlign layer is designed to fix the location misalignment caused by quantization in the RoI pooling. RoIAlign removes the hash quantization, for example, by using x/16 instead of [x/16], so that the extracted features can be properly aligned with the input pixels. Bilinear interpolation is used for computing the floating-point location values in the input.

5. An overview of models in in the R-CNN family

Figure 5 is an overview of model designs of R-CNN, Fast R-CNN, Faster R-CNN and Mask R-CNN. You can track how one model evolves to the next version by comparing the small differences [5].

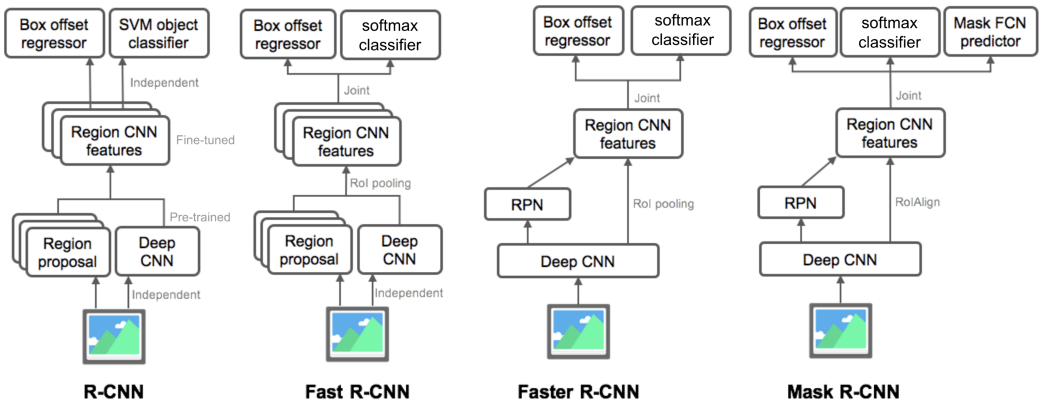

FIGURE 5. R-CNN family.

References

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In Proc. IEEE Conf. on computer vision and pattern recognition (CVPR), pp. 580-587. 2014.

[2] Ross Girshick. "Fast R-CNN." In Proc. IEEE Intl. Conf. on computer vision, pp. 1440-1448. 2015.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks." In Advances in neural information processing systems (NIPS), pp. 91-99. 2015.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN." arXiv preprint arXiv:1703.06870, 2017.

[5] Object Detection for Dummies Part 3: R-CNN Family.