Fast R-CNN Summary

17 December 2023

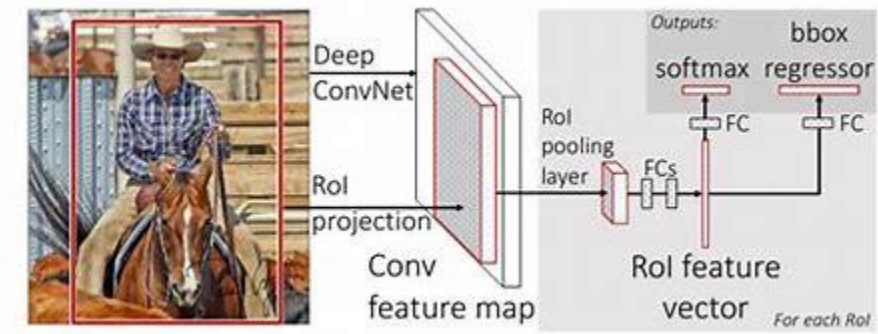
Introduction

- The authors of the fast R-CNN paper aim to solve the notable drawbacks of the Region-based Convolutional Network method (R-CNN) which achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. The drawbacks include:-
    1. **Training is a multi-stage pipeline**:- R-CNN first fine-tunes a ConvNet on object proposals using log loss.Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.
    2. **Training is expensive in space and time:-** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 trainval set. These features require hundreds of gigabytes of storage.
    3. **Object detection is slow:-** At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

- SPP-Nets or "Spatial Pyramid Pooling Networks" solved one part of the problem. Its main contribution was
    - The use of multi-scale pooling of image features to improve the network performance for classification and detection and also
    - The use of arbitrary sized input images to train CNN's despite the use of fully connected (FC) layers. In its detection pipeline, it introduced an important concept— extracting a single feature map from the entire image only once, and pooling the features of arbitrary sized "region proposals" for that image, from this single feature map.

- The advantages of the proposed Fast R-CNN architecture are:-
    - Higher detection quality (mAP) than R-CNN, SPPnet
    - Training is single-stage, using a multi-task loss
    - Training can update all network layers
    - No disk storage is required for feature caching

Fast R-CNN architecture and training



- The network processes the image using convolutional and max pooling layers to create a conv feature map. For each object proposal, a region of interest pooling layer extracts a fixed-length feature vector. These vectors are fed into fully connected layers, branching into two output layers - a (K + 1) category softmax layer branch and a category-specific bounding box regression branch.
- The RoI layer is simply the special-case of the spatial pyramid pooling layer used in SPPnets in which there is only one pyramid level.The layer basically divides the features from the selected proposal windows (that come from the region proposal algorithm) into sub-windows of size *h/H* by *w/W* and performs a pooling operation in each of these sub-windows. This gives rise to fixed-size output features of size (H x W) irrespective of the input size. H and W are chosen such that the output is compatible with the network's first fully-connected layer. The chosen values of H and W in the Fast R-CNN paper is 7. Like regular pooling, ROI pooling is carried out in every channel individually.
- In Fast RCNN training, stochastic gradient descent (SGD) mini batches are sampled hierarchically, first by sampling N images and then by sampling R/N RoIs from each image. Critically, RoIs from the same image share computation and memory in the forward and backward passes. Making N small decreases mini-batch computation.

Multi-task Loss:-

- Fast R-CNN network has two sibling output layers. The first outputs a discrete probability distribution (per RoI), p = (p0, . . . , pK), over K + 1 categories. . The second sibling layer outputs bounding-box regression offsets, t^k = (t^k_x, t^k_y, t^k_w, t^k_h), for each of the K object classes, indexed by K.
- Multi-task loss L on each labeled RoI to jointly train for classification and bounding-box regression:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v),$$

- The regression branch produces 4 bounding box regression offsets t^k_i where *i = x, y, w,* and *h. (x, y)* stands for the top-left corner and *w* and *h* denote the height and width of the bounding box. The true bounding box regression targets for a class u are indicated by v_i where *i = x, y, w,* and *h* when u >= 1. The case where *u = 0* is ignored because the background classes have no ground truth boxes.
- For bounding box regression the loss used is:

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Training:-

- During training each mini-batch is constructed from N = 2 images. The mini-batch consists of 64 ROIs from each image.
- Like R-CNN, 25% of the ROIs are object proposals that have at least 0.5 IoU with a ground-truth bounding box of a foreground class. These would be positive for that particular class and would be labelled with the appropriate u = 1…K.
- The remaining RoIs are sampled from object proposals that have a maximum IoU with ground truth in the interval [0.1, 0.5). These are the background examples and are labeled with u = 0.
- During training, images are horizontally flipped with probability 0.5. No other data augmentation is used
- The batch size of the network is very small for the CNN until the ROI pooling layer (batch size = 2), but is much larger (batch size = 128) for the following softmax and regression layers.
- For each mini-batch ROI *r,* let the ROI pooling output unit *y_(rj)* be the output of max-pooling in it's sub-window *R(r, j).* Then, the gradient is accumulated in an input unit (*x_i*) in *R(r, j)* if this position ***i*** is the argmax selected for *y_(rj).*
- At test-time, an image pyramid is used to approximately scale-normalize each object proposal. During multi-scale training, we randomly sample a pyramid scale each time an image is sampled, as a form of data augmentation.
- The large fully connected layers can be compressed with truncated SVD to make the network more efficient. Here a layer parameterized by W as its weight matrix can be factorized to reduce the parameter count by splitting it into two layers (ΣV^t and U with biases) without a nonlinearity between them, where W ~ U ΣV^t.

Results:-

Three main results support this paper's contributions:

1. State-of-the-art mAP on VOC07, 2010, and 2012
2. Fast training and testing compared to R-CNN, SPPnet
3. Fine-tuning conv layers in VGG16 improves mAP