# fast-rcnn

shhs

September 27, 2016

## Contents



# 1 Fast R-CNN –Ross Girshick

Paper: Fast R-CNN Code: Fast R-CNN's code

# 2 Contents

1. Fast R-CNN architecture

### 2.1 Fast R-CNN architecture

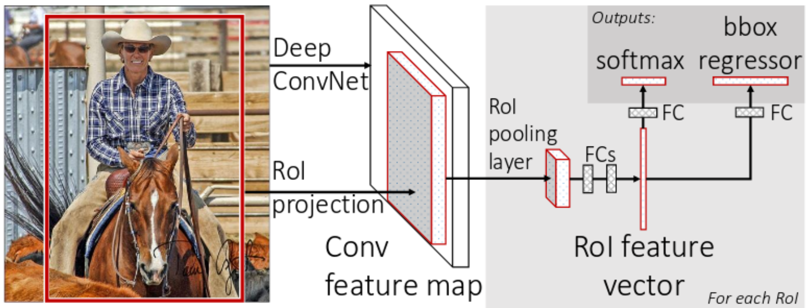


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

- inputs: **an entire image**, **a set of object proposals**
- several convolutional and max pooling layers -> produce a conv feature map
- for each object proposal: a RoI(region of interest) pooling layer extracts a fixed-length feature vector from the feature map
- each feature vector is fed into a sequence of fully connected(fc) layers that finally branch into two sibling() output layers: **one** that produces softmax probability estimates over K object classes plus a catch-all "background" class and **another layer** that outputs four real-valued numbers for each of the K object classes.
- ☐ ? 2. *Each set of 4 values encodes refined bounding-box positions for one of the K classes.*

#### 2.1.1 The RoI pooling layer

- The RoI pooling layer uses max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of HxW, wherer H and W are layer hyper-parameters that are independent of any particular RoI.

- RoI: (r,c,h,w) specifies its top-left corner(r,c) and its height and width(h,w).
- RoI max pooling layer divides the hxw RoI window into an HxW grid of sub-windows of approximate size h/H x w/W and then max-pooling the values in each sub-window into the corresponding output grid cell.

#### 2.1.2 Initializing from pre-trained networks

- When a pre-trained network initializes a Fast R-CNN network, it undergoes three transformations:
  1. The last max pooling layer is replaced by a RoI pooling layer that is configured by setting H and W to be compatible with the net's first fully connected layer (e.g., H = W = 7 for VGG16).
  2. The network's last fully connected layer and softmax are replaced with the two sibling layers described earlier: a fully connected layer and softmax over K + 1 categories, category-specific bounding-box regressors.
  3. The network is modified to take two data inputs: a list of images and a list of RoIs in those images.

#### 2.1.3 Fine-tuning for detection

- In Fast R-CNN training, stochastic gradient descent(SGD) mini-batches are sampled hierarchically.
  1. First sampling N images
  2. Second sampling R/N RoIs from each image.
- RoIs from the same image share computation and memory in the forward and backward passes.
- One concern over this strategy is it may cause slow training convergence because RoIs from the same image are correlated. This concern does not appear to be a practical issue and we achieve good results with N = 2 and R = 128 using fewer SGD iterations than R-CNN.

- Multi-task loss

  - A Fast R-CNN network has two sibling output layers.

    1. The first outputs a discrete probability distribution(per RoI), $a_1$

$$\sum_{i=1}^{n}(a_i * w_i) \tag{1}$$

$$\frac{1^p + 2^p + \cdots + n^p}{n^{1+p}} \tag{2}$$

$$\xrightarrow{abc} \tag{3}$$

I am $op_1 \xrightarrow{abc} op_2$