Fast R-CNN

1 Citation

Girshick, Ross. "Fast r-cnn." arXiv preprint arXiv:1504.08083 (2015).

https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

2 Abstract

Fast R-CNN makes many improvements to R-CNN (e.g. multi-task loss, RoI pooling) to make R-CNN 9x faster at train time and 213x faster at test-time.

3 Introduction

Existing object detection systems are multistage pipelines, which makes them slow, cumbersome to develop, and less performant. Joint training can fix this issue.

R-CNN proposes regions, extracts features with a fine-tuned pretrained CNN, makes classifications with SVMs, and does bounding box regression. It's multi-stage, slow (47 seconds per image at test time), and expensive in time (2.5 GPU-days for training on PASCAL VOC) and space (hundreds of gigabytes of storage). It's slow because you need to run the CNN for every proposed region. SPP-Net speeds this up by computing the features once and pooling over the regions. However, it's also multi-stage and takes a lot of storage space.

Fast R-CNN is better because it gets higher accuracy, is single-stage, can improve all network layers during training, and uses little disk space.

4 Fast R-CNN Architecture and Training

The input the system is an image and set of regions. We compute the features once, and pool over the regions (region of interest (RoI) pooling) to get a fixed sized vector. This is then fed into sibling layers that output a class probability distribution and bounding box prediction (x, y, width, height) for each class. The RoI pooling layer is a SPP pyramid with a single layer. That is, it takes a region and outputs a grid of numbers of fixed size. It lets us turn a variable sized region into a fixed size output vector.

We use VGG-16 (pretrained on ImageNet) for our system. However, you can use any ImageNet model - just replace the last pooling layer with RoI pooling and replace the fully connected alyers and softmax with the sibling layers.

SPP-Net can't backpropagate before the spatial pooling layer because it is expensive. To fix this, we set $N = 2$ and $R = 128$ and every minibatch will consist of N images where we pick R/N regions for each image to train on. This works well in practice. For ground truth class u, ground truth bounding box v, predicted class distribution p, and predicted bounding box for class-u $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$, our loss is:

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

Where $u = 0$ indicates the background class. $L_{cls}(p, u) = -\log p_u$.

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

$$\text{smooth}_{L_1}(x) = \mathbb{I}[x < 1] 0.5x^2 + \mathbb{I}[x \geq 1](x - 0.5)$$

The L_{loc} is designed to be outlier resistant. We normalize regression targets to have zero-mean and unit variance. We set $\lambda = 1$.

During training, we make sure 25% of our regions are positive (i.e. RoI overlaps ground-truth with intersection-over-union (IoU) of 0.5 or more) and the rest are negative (IoU is below 0.5 and at least 0.1 (for hard example mining)). We flip images with probability 0.5, but do no other data augmentation. You can derive an expression for the gradient (see paper).

We consider single-scale training and multi-scale (sample different scale each time) training. We fine-tune the pretrained CNN.

5 Fast R-CNN Detection

At test-time, we feed the image through the convolutional layers. Then, for each region proposal, we do RoI pooling and feed it through the sibling layers. We then do non-maxima suppression for each class independently.

The fully-connected layers are the bottleneck, so we can use Singular Value Decomposition where we approximate the fully connected weights ($u \times v$ size) as $W \approx U \Sigma_t V^T$ for the top t singular values. This cuts the number of parameters from uv to $t(u + v)$, which is more efficient.

6 Main Results

For our CNN, we use AlexNet, a smaller VGG, and VGG-16. We set state of the art on PASCAL VOC. We beat SPP-Net and R-CNN. We are faster and take up less disk space during training. The SVD decomposition of the fully connected layers gives a 30% speedup and tiny reduction to mean-average-precision (mAP).

For fine-tuning, we find that fine-tuning the entire CNN (instead of just the fully-connected layers), gives gains.

7 Design Evaluation

Does multi-task training help? Yes.

For scale invariance, should we use brute-force or finesse? Just use brute-force (i.e. a single scale).

Do we need more training data? If you use more training data, the model gets better. That's good because it means it is not saturated yet.

Do SVMs outperform softmax? No, they are about the same.

Are more proposals always better? Too many proposals does hurt mAP, but the drop-off is slow. Using fewer (i.e. sparse) proposals does seem to help.

8 Conclusion

Fast R-CNN is more accurate and faster than R-CNN and SPP-Net.