OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks

1 Citation

Sermanet, Pierre, et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." arXiv preprint arXiv:1312.6229 (2013).

https://arxiv.org/pdf/1312.6229.pdf

2 Abstract

We win localization in ILSVRC 2013 and do well on detection + classification with a single shared network. We solve localization by predicting boundaries and accumulating bounding boxes. We release a feature extractor you can use.

3 Introduction

Convnets are great because they are trained end-to-end, so you don't need to write your own feature extraction. The problem is they need a lot of training data.

We solve classification, localization, and detection in a single network in one network and also accumulate bounding boxes (avoids training on background samples).

Basically, we slide a Convnet over the image at multiple scales, predict the category and bounding box, and then we accumulate these bounding boxes.

4 Vision Tasks

Classification = identify dominant category in image. Localization = identify dominant category and its location(s). Detection = Identify all categories and their locations. Classification and localization each get 5 guesses. For localization, the intersection over union of the predicted and true bounding boxes must be at least 50%. For detection, false positives are penalized by mean average precision. Classification + localization use the same dataset.

5 Classification

Scale image so shortest side has length 256. Extract 5 random crops of size 221×221. Use batch size of 128. Initialize weights with zero-mean Gaussian with $\sigma = 0.001$. Use stochastic gradient descent with momentum of 0.6 and weight decay of 1×10^{-5}. Learning rate begins at 5×10^{-2} and drops by half at $(30, 50, 60, 70, 80)$ iterations. Use dropout on the first and second fully connected layers.

We use 5 convolutional layers and 3 fully connected layers, like in AlexNet. However, we don't use contrast normalization, pooling regions don't overlap, and our model has larger 1st and 2nd layer feature maps (our stride is smaller).

We release two feature extractors - a fast one and an accurate one.

Let's discuss how to do multiscale classification. First, we consider 6 different image scales.

1. We apply the convolutional layers over each scaled image to get the unpooled layer 5 feature maps.
2. We apply a 3×3 max pooling operation where we apply an x and y offset taken from $\Delta_x \in 0, 1, 2$ and $\Delta_y \in 0, 1, 2$. This is $3 \times 3 = 9$ pooled feature maps for each scaled image.
3. We apply the classification layers in a sliding window fashion to these pooled feature maps.
4. We assemble each (Δ_x, Δ_y)'s feature maps into a 3D tensor with two spatial dimensions and C channels.
5. Do the above steps for horizontally flipped images too.
6. Take spatial max for each class at each scale and flip
7. Average C-dimensional vectors from different scales and flips
8. Take top-1 or top-5 elements.

Note that you can turn a fully connected layer into a convolutional layer, so the entire network is a combination of convolution, thresholding, and max pooling.

6 Localization

Replace classification layers with regression layers. So, first train the whole network on classification. Then fix the feature extraction layers and just train the regression layers. Then run both the classification network and regression network in a sliding window over the images at different scales.

Regression layers have two fully-connected hidden layers with 4096 and 1024 channels, respectively. The output of the regression layers is the set of 4 numbers representing the bounding box edges. We found sharing one regressor over all classes was better than training one regressor per class (probably because there were too few training examples per class).

We optimize the L2 loss between predicted and true bounding box. We don't train when more than 50% of the true bounding box is out of frame. We train on multiple scales and use the same (Δ_x, Δ_y) from before.

Here's how we merge bounding boxes:

1. $C_s :=$ classes in top-k for each scale $s \in 1, ..., 6$ found by maximum detection output across spatial locations for that scale.
2. $B_s :=$ bounding boxes for for each class in C_s across all spatial locations at scale s.
3. $B := \cup_s B_s$
4. Repeat until done
   (a) $(b_1^*, b_2^*) := \text{argmin}_{b_1 \neq b_2 \in B} \text{matchscore}(b_1, b_2)$
   (b) if $\text{matchscore}(b_1, b_2) > t$, stop
   (c) $B := (B - \{b_1^*, b_2^*\}) \cup \text{merge}(b_1^*, b_2^*)$

The matchscore is the sum of distance between centers and intersection. We then pick the bounding box with maximum score (each box gets a score equal to the class detection output from its input window - sum this when we merge boxes).

7 Detection

It's like classification, but done spatially. It's like localization, but you need to predict a background class when you don't find anything. We get the negative (i.e. background) example on the fly for each image, rather than doing multiple bootstrapping iterations, which is complicated.

8 Discussion

We get 1st in localization, 1st in detection, and 4th in classification. To improve our work, you could:

1. Backprop through the whole regression network, rather than just through regression layers.
2. Optimize intersection-over-union loss rather than L2 loss.
3. Parametrize the bounding box in different ways to avoid correlated predictions.