Very Deep Convolutional Networks For Large Scale Image Recognition

1 Citation

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

https://arxiv.org/pdf/1409.1556.pdf

TODO: Put citation here.

2 Abstract

We won ImageNet 2014 Classification + Localization with a deep (16-19 layer) CNN with small (3×3) filters.

3 Introduction

Ever since the AlexNet CNN work ImageNet 2012, people have been improving CNNs. We build a very deep CNN (enabled by having filters with small receptive fields) that gets state of the art performance on ImageNet classification and localization.

4 ConvNet Configurations

Input is 224×224 image where mean RGB value is subtracted from each pixel. Each of our CNNs uses 3×3 filters (we pick this so we can look at the upper, lower, left, and right pixel). Some use 1×1 filters, which is just a linear transformation of input channels. We use 2×2 spatial pooling with a stride of 2. Our CNNs end with three fully connected layers and a softmax - the first and second FCs have 4096 units and the last FC has 1000 units. We use ReLU activation. We tried Local Response Normalization (like in AlexNet), but it didn't help.

Our smallest CNN (called network A) has 11 layers and the largest (network E) has 19.

Notice that three stacked 3×3 filters has an effective receptive field of 7×7, but only uses $3(3 \times 3)C^2 = 27C^2$ parameters as opposed to the $(7 \times 7)C^2 = 49C^2$ parameters of a 7×7 filter. The three small filters have a regularizing effect. The 1×1 filters (+ ReLU) add more nonlinearity to our decision function.

5 Classification Framework

We optimize softmax regression loss using minibatch ($B = 256$) gradient descent with momentum. We regularizing with L2 weight decay and dropout. Learning rate is cut by a factor of 10 each time validation

error stops improving. We train for 74 epochs which is less than AlexNet because we have greater regularization and we pre-initialize certain layers. We trained network A with random initialization. Then, for the deeper networks, we initialize the first 4 layers and FC layers with the weights with network A.

We randomly sample crops from different scales, do horizontal flips, and do RGB color shift to augment the dataset.

When taking a crop, we first scale the image so that shortest size has length S. How do you pick S? We tried fixing it at $S = 256$ and $S = 384$, and we also tried sampling it uniformly from $S \in [S_{min}, S_{max}] = [256, 512]$. For this multiscale model, we just fine-tuned the $S = 384$ model.

At test-time, we scale the image so the shortest side has length Q. We apply our CNN densely (also tried multi-scale/crops) over the image and sum-pool the resulting spatial map. We do the same thing for the flipped image and take the average between the flipped and non-flipped images as our prediction.

We built our CNNs using a modified Caffe library and takes 2-3 weeks to train on 4 NVIDIA Titan GPUs.

6 Classification Experiments

ImageNet classification has 1.3M/50K/100K training/validation/test images.

We set $Q = S$ for fixed-S and $Q = 0.5(S_{min} + S_{max})$ for jittered S.

Local Response Normalization does not help.

Deep network with small filters > Shallow network with large filters.

Scale jittering > single scale. Scale jittering at test time also helps.

At test time, multi-crop is slightly better than dense application of the CNN, and combining them both is the best.

Averaging all the CNNs we made does well. Averaging the two best CNNs does even better.

We did not win ILSVRC classification (GoogLeNet did), but we won localization.

7 Conclusion

Deep networks with small filters work well for classification and other tasks.

8 Appendix A - Localization

We use network D and modify it to predict a bounding box (a 4D vector with the center coordinate, height, and width). We actually found that having a per-class bounding box (so the vector would be $1000 \times 4 = 4000D$), did better than having one bounding box for all classes.

We use fixed-S and train on Euclidean loss.

We considered two testing protocols - apply to central crop and densely apply network. For the latter, use the greedy bounding-box merging strategy from OverFeat. The latter worked better.

In ImageNet, a localization is correct if the intersection over union of the bounding boxes is at least 50%.

9 Generalization of Very Deep Features

We remove the last fully connected layer for network D and network E - this gives a feature extractor that you can use for other tasks. All you need to do scale your image to size Q, densely apply the CNN to get feature vectors, L2 normalize the feature descriptors, global average pool them (or stack them), and feed into an SVM.

We do pretty well on PASCAL VOC and Caltech datasets.