Very Deep Convolutional Networks for Large-Scale Image Recognition

Cheng Guan

August 11, 2018

1. Classification Framework

Today, I continue to read this paper about VGG16. In the previous section the authors presented the details of their network configurations. In this section, they describe the details of classification ConvNet training and evaluation.

1.1. Training

The ConvNet training procedure generally follows [1] (except for sampling the input crops from multi-scale training images, as explained later). Namely, the training is carried out by optimising the multinomial logistic regression objective using mini-batch gradient descent (based on back-propagation[2]) with momentum. The batch size was set to 256, momentum to 0.9. The training was regularised by weight decay (the L_2 penalty multiplier set to $5 \cdot 10^{-4}$) and dropout regularization for the first two fully-connected layers (dropout ratio set to 0.5). The learning rate was initially set to 10^{-2} , and then decreased by a factor of 10 when the validation set accuracy stopped improving. In total, the learning rate was decreased 3 times, and the learning was stopped after 370K iterations (74 epochs). The authors conjecture that in spite of the larger number of parameters and the greater depth of their nets compared to [1], the nets required less epochs to converge due to (a) implicit regularization imposed by greater depth and smaller conv. filter sizes; (b) pre-initialization of certain layers.

The initialization of the network weights is important, since bad initialization can stall learning due to the instability of gradient in deep nets. when training deeper architectures, they initialized the first four convolutional layers and the last three fully- connected layers with the layers of net A (the intermediate layers were initialized randomly). they did not decrease the learning rate for the pre-initialized layers, allowing them to change during learning. For random initialization (where applicable), they sampled the weights from a normal distribution with the zero mean and 10^{-2} variance.

Training image size. Let S be the smallest side of an isotropically-rescaled training image, from which the ConvNet input is cropped (they also refer to S as the training scale). While the crop size is fixed to 224×224, in principle S can take on any value not less than 224: for S = 224 the crop will capture whole-image statistics, completely spanning the smallest side of a training image; for $S \gg 224$ the crop will correspond to a small part of the image, containing a small object or an object part.

The authors consider two approaches for setting the training scale S. The first is to fix S, which corresponds to single-scale training (note that image content within the sampled crops can still represent multi- scale image statistics). In their experiments, they evaluated models trained at two fixed scales: S = 256 (which has been widely used in the prior art [4]

1.2. Test

At test time, given a trained ConvNet and an input image, it is classified in the following way. First, it is isotropically rescaled to a pre-defined smallest image side, denoted as Q (they also refer to it as the test scale). They note that Q is not necessarily equal to the training scale S. Then, the network is applied densely over the rescaled test image in a way similar to [3]. Namely, the fully-connected layers are first converted to convolutional layers (the first FC layer to a 7×7 conv. layer, the last two FC layers to 1×1 conv. layers).

1.3. Implementation Details

Their implementation is derived from the publicly available C++ Caffe toolbox but contains a number of significant modifications, allowing us to perform training and evaluation on multiple GPUs installed in a single system, as well as train and evaluate on full-size (uncropped) images at multiple scales (as described above). Multi-GPU training exploits data parallelism, and is carried out by splitting each batch of training images into several GPU batches, processed in parallel on each GPU.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989. 1

[3] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 1

[4] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 1