Summary of AlexNet

Anowarul Kabir

20th February 2020

The contribution of this paper is two-fold. First, the authors introduce one of the largest convolutional neural networks which achieved the best results by far over ILSVRC-2010 and ILSVRC-2012 datasets. And second, it is implemented on multiple GPU which reduces its training time as well as shows some explainable learning characteristics. Beside that, this emphasizes on several features along with its architecture which aid to improve its performance. To model a neuron's output f as a function of its input x, they use a non-saturating nonlinearity called rectified linear units (ReLUs), expressed as $f(x) = max(0, x)$. Previously, the standard way to model this, *sigmoid* or *tanh* functions were proved effective. Their empirical evaluation shows that ReLUs over *sigmoid* or *tanh* function converges faster. Another critical feature of the architecture is local response normalization which aids in generalization and reduces top-1 and top-5 error rates by 1.4% and 1.2% respectively. Although ReLUs do not require input normalization, however the authors show its empirical effectiveness over CIFAR-10 dataset. The idea behind this is basically brightness normalization. This normalization is applied after ReLU nonlinearity in first and second convolutional layers. They also employ overlapping max-pooling layers to summarize the outputs of neighboring groups of neurons in the same kernel map. The pooling layers reduce top-1 and top-5 error rates by 0.4% and 0.3% respectively in comparison with non-overlapping scheme. Max-pooling layers are followed by response normalization layers as well as fifth convolutional layer.

The input images are 256x256x3 dimensional. Using data augmentation, they cut ten patches from a single image where each patch is 224x224x3 dimensional. The architecture takes input images after data augmentation, so each input images are 224x224x3 dimensional. It has five convolutional layers followed by three fully-connected layers. The full architecture is as follows.

1. The first convolutional layer has 96 kernels (48 kernels in each GPU) of size 11x11x3 with a stride of 4 pixels. The outputs are fed into response normalization layer and max pooling layer.

2. The second convolutional layer takes input after max-pooling layer. It has 256 kernels (128 kernels for each GPU) of size 5x5x48. Then the outputs are fed into response normalization layer and max pooling layer.

3. The third convolutional layer has 384 kernels of size 3x3x256.

4. The forth convolutional layer has 384 kernels of size 3x3x192.
5. The fifth convolutional layer has 256 kernels of size 3x3x192. Then max-pooling layer follows the fifth layer.
6. Then three fully-connected layers follow, each of them has 4096 neurons.
7. At the end there is a softmax layer which outputs the probability distribution for each example image over 10 classes.
8. Besides the general architecture, here are some nuances, the kernels of the second, forth and fifth convolutional layers are only connected to those layer which reside in the same GPU. The response-normalization layers follow only first and second convolutional layers. Max pooling layers follow only response-normalization layers after first and second layers as well as after the fifth convolutional layer. The ReLU is applied to the output of every convolutional and fully-connected layers.

This architecture has 60 million parameters. So it requires lots of examples to learn all these parameters. As a result it is very sensitive to overfitting. To combat overfitting the paper describes two techniques. First, augmenting the dataset which increases the training set by a factor of 2048 and this scheme reduces top-1 error rate by over 1%. The second technique is dropout which sets to zero the output of each hidden neuron with probability of 0.5. They use dropout in the first two fully-connected layers only. Although dropout requires about double the number of iterations, it handles overfitting effectively. This architecture achieves top-1 and top-5 error rate of 37.5% and 17.0% in comparison with other state of the arts on the ILSVRC-2010 test dataset.

This architecture consume lots of memory and requires lot of time to converge at train time. This is somewhat explainable about how each convolutional kernels learn different parts of the image. For instance, the kernels on GPU-1 are color agnostic while the kernels on GPU-2 are color specific. It has several hyperparameters which requires tuning for converging and as it takes lots of time for a training loop tuning hyperparameters is a huge headache.