Going deeper with convolutions

Link

arxiv

Summary

- Hebbian principle states that neurons that fire together, wire together. We can usually improve deep neural net's performance by increasing their size. But bigger nets are more prone to overfitting especially when the training set is small. They can also be very expensive. The fundamental way to solve these issues is to move from densely connected networks to sparsely connected networks. These way we can more closely mimic biological neurons. Theoretical work by Arora et al. in 'Provable bounds for learning some deep representations' states that if the probability distribution of dataset can be represented by a large, very sparse deep neural network we can construct the optimal network topology. To do so we need to analyze the correlation statistics of activations of last layer and cluster neurons with highly correlated outputs. But our current hardware are highly optimized for dense matrix operation. So in this paper they try an intermediate approach where they try to achieve data sparsity at filter level and utilize current hardware by performing dense matrix operation.

- Inception module is used to introduce local sparsity. We assume each unit in the lower layer corresponds to some regions in space. Some units will extract feature from 1x1 region, so we compute them together(wire them together). Some units should be responsible for extracting feature from 3x3 region and they are computed together. They use filter banks of 1x1, 3x3 and 5x5 kernel size(to avoid alignment issue). As higher layers capture more abstract features their spatial concentration should decrease. So the ratio of 3x3 and 5x5 filters is increased gradually(compared to 1x1 filters). Finally they concatenate features computed by all those filter banks(they also use a parallel pooling path).

- Since calculating 5x5 or even 3x3 convolutions can be too expensive they use dimension reduction before applying these convolutions. Instead of going directly from A channels to B channels by using a 5x5(or 3x3) convolutions they first use 1x1 convolutions to go to an intermediate C channels and then use the 5x5 convolution from C to B. So the operation needed is reduced by a factor of $(5^2AB)/(AC + 5^2CB)$ when C is significantly smaller than B. For example if A=480, B=128, C=32 we get a reduction of $(25 * 480 * 128)/(480 * 32 + 25 * 32 * 128) \approx 13$.

- One insight was that the relatively strong performance of the shallow network suggests that the features computed by intermediate layers are discriminative. So they connected auxiliary classifiers to some of the intermediate layers. During training their losses are added to the loss of the entire network by a weighting factor of 0.3. These auxiliary networks are discarded during inference time.