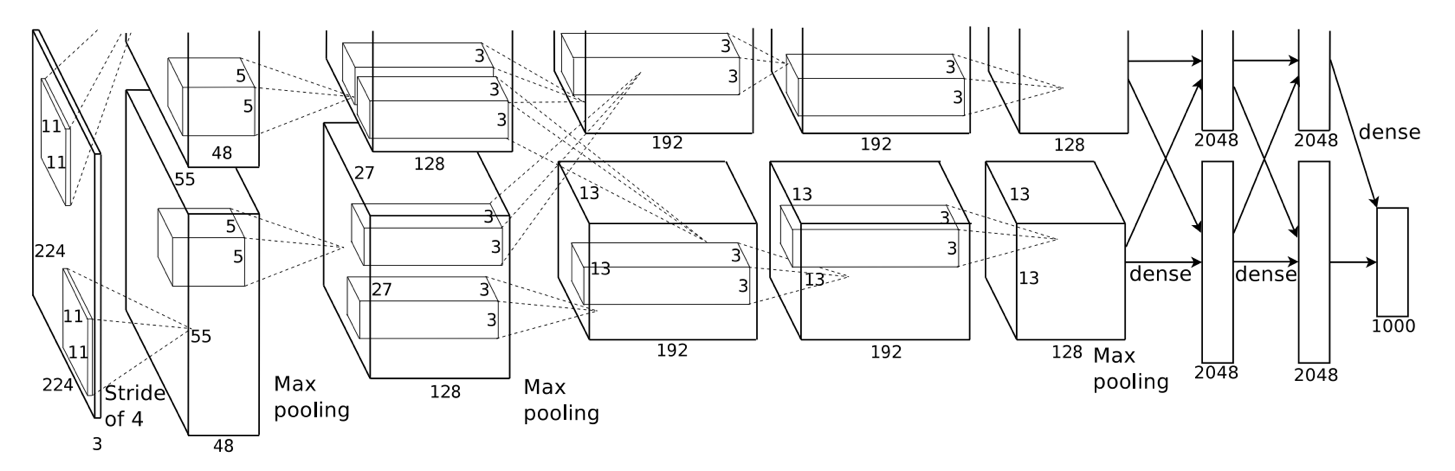
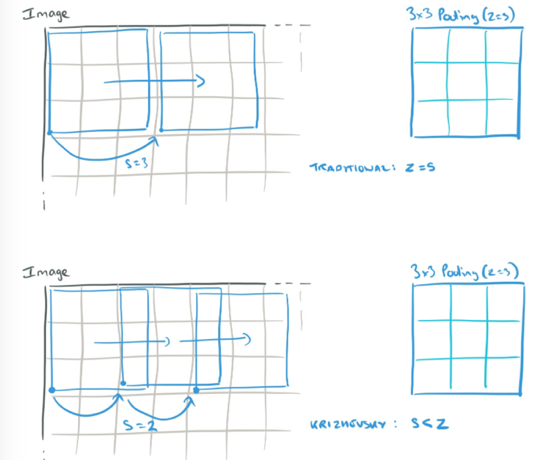
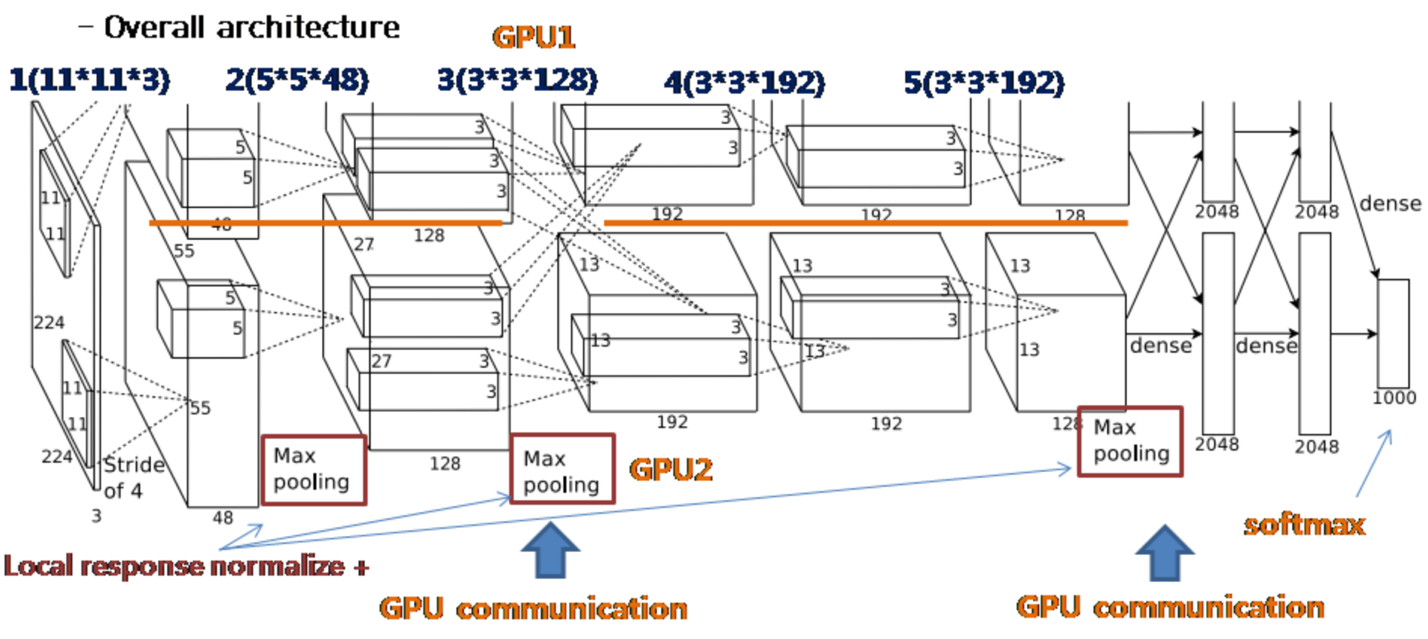
ImageNet Classification with Deep CNN (AlexNet)

Jo MinKi

1. Intro.   
   Let’s do this   
   (This is legendary research in machine learning field and the moment when this paper was published is great starting point of machine learning boom. Before this paper, every research about the NN was automatically rejected from the academic community and NN researcher had a rough time.)
2. The Dataset  
   ImageNet(15millon images / 22,000 categories) with variable resolution.   
   Because Alexnet requires a constant input dimensionality, they manipulate the size of images to 256256. (I had been through this process when I used VGG16 before and I think it’s because the size of filtering window. CNN is not fully connected NN and it should have resizing plan because of padding size. But I don’t know why it should be a rectangular.)
3. The Architecture  
     
   It has total 8 layers, 5 of CNN and 3 of fully connected.   
     
   This has several important features and following sections are sorted according to estimation of importance.

(!@@! Section 3.2 and GPU separation techniques in 3.5 are not needed now. They did that just because at that time, GPU performance was not good enough so they could not put the whole data set at once. But now the memory of GPU expended four times so we don’t need to do that trick anymore.)

* 1. ReLU  
     Standard activation functions like sigmoid or tanh are much slower than ReLU when we calculate the gradient. (And it seems saturation issue is also important, but I don’t know why.) According to the experiment we can see that ReLU is several times faster than tanh.
  2. Training on Multiple GPUs  
     They used 1.2 million training data and this is too much for one GPU (they used GTX 580 which has 3GB memory). They explained about systemic issues and tricks like GPU parallel computing or communication control, but I can’t understand this at all. As far as I understand, GPU is well suited to parallelization computing, and this is able to read and write the data to each other directly, without communicating host memory. (usually later way make it slower) And GPUs only communicate at the particular layer.
  3. Local Response Normalization   
     In case of sigmoid and tanh, it need input normalization to prevent the gradient saturating. If it take raw input without normalization process, the gradient of activation function would be 0 with high probability so the value will not change even after several iterations.   
     But in case of ReLU, it does not need to be considered about the gradient saturation. However, they are thinking that the Normalization still make it help for generalization. This normalization emphasizes big response from many other responses, so other weak responses are relatively ignored. This gives same effect with lateral inhibition in biology.
  4. Overlapping Pooling   
     Traditionally pooling process performed without over lapping, (stride s is bigger than the size of pooling cell z) but they did overlapping pooling. (s = 2 < z = 3) They claim that in this way they could avoid the overfitting and achieved better accuracy.
  5. Overall Architecture   
     As we mentioned above, first 5 layer is convolutional layer and rest of them are fully connected layer. [1, 2, 3, 4, 5] Layer has [96, 156, 384, 384, 256] filter and two of GPU are computing half of each. In layer 1,2 and 3,4,5, GPUs are not communicating each other side, but in layer 2,3 are share their data.   
     Response-normalization layers follow the first and second layer. The last fully connected layer has 1000 way of softmax so it distributes the data over 1000 class of labels.  
       
     GPU2 process information about color while GPU1 process information which is irrelevant to the colors such like edges.   
     But these information is shared at the 3rd layer.

1. Reducing Overfitting  
   Because the size of network is too large, (60millon parameter and 1000 classes) the number of data was not enough to prevent the overfitting. So they did some process to prevent overfitting.
   1. Data augment  
      Randomly crop the cross of the image (extract 224X224 image from 256X256 image) and make the horizontal reflection image of it. So make 10 image from one original image with this way. Or in traditional way, randomly enlarged image also could be used. There was one the other way which using PCA process on the set of RGB pixel value.   
      One tricky part is that these augment process performed by simple calculation, so processed image does not need to be stored in the disk and but just can be used for the later process directly.
   2. Drop out  
      Do a drop out. To prevent the synchronization between neurons, set the drop-out rate to 0.5.

Later part (5 Detail, 6 Result) seems too detail so I think we don’t need to memorize

Question.

1. According to 3.1, it seems Relu is always better than sigmoid or tanh. Is it? Even if we don’t consider about the computation speed and do a normalization every time to prevent the saturation issue?
2. What is the ‘kernel’ in the CNN?