AlexNet

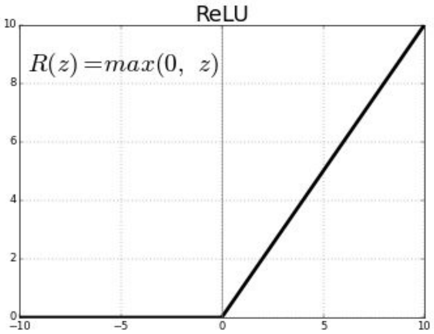
ImageNet Classification with Deep Convolutional Neural Networks

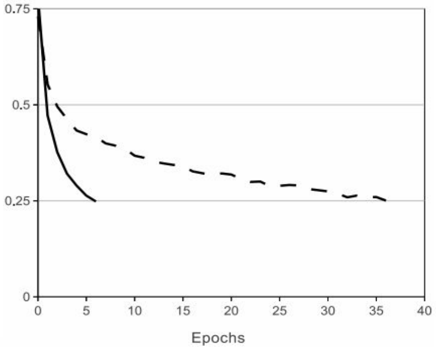
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

Lisa Fan & Jason Krone

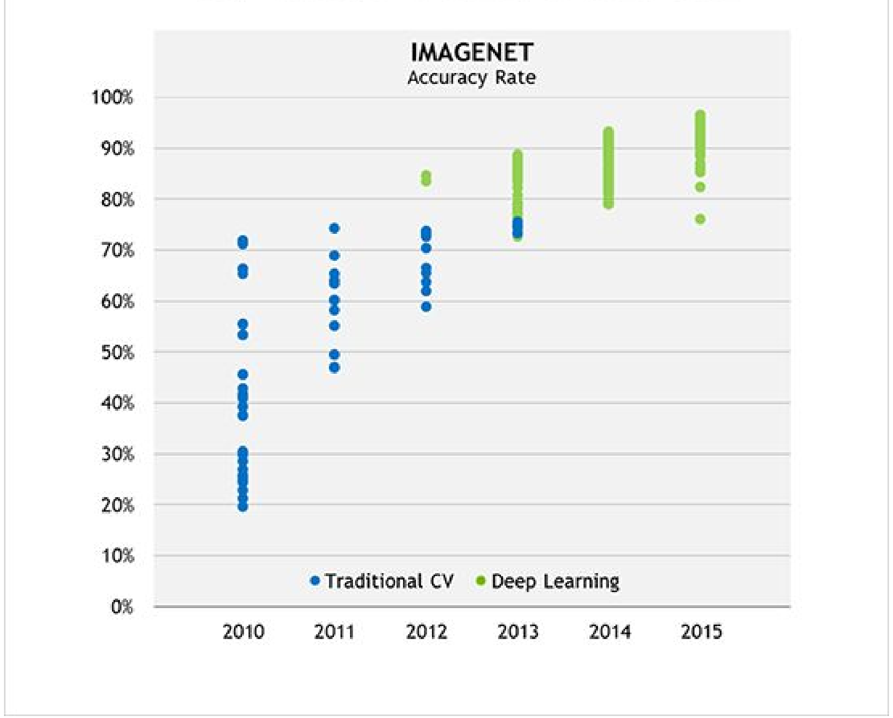
Introduction

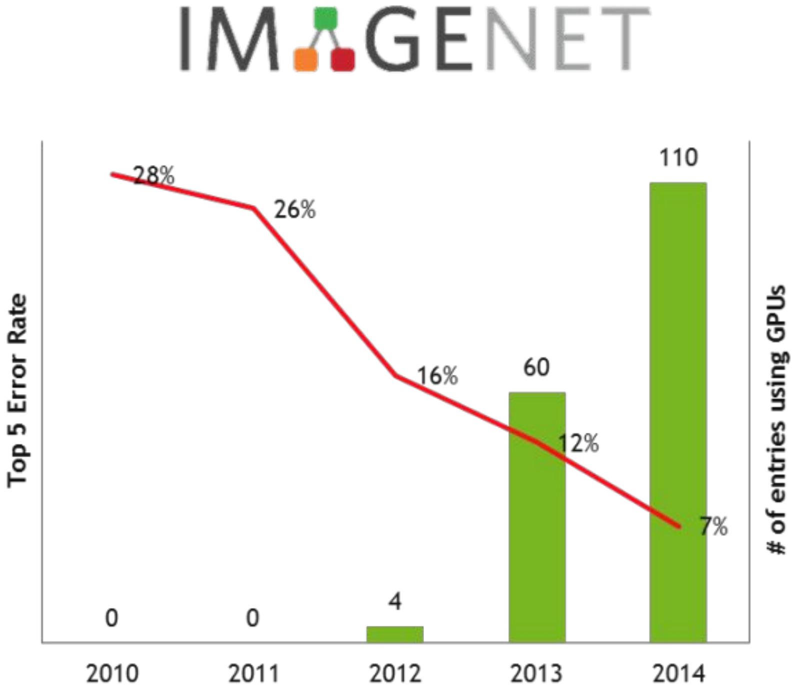
- Submitted to ImageNet Challenge in 2012
    - Won competition with 16% top 5 error. Large improvement over 2nd runner-up with 26%
- Popularized CNNs for computer vision
    - Although CNNs were previously used, e.g. LeNet (Yann LeCun), AlexNet was deeper, bigger, and included stacked Convolutional layers.
- Introduced ReLU activation function





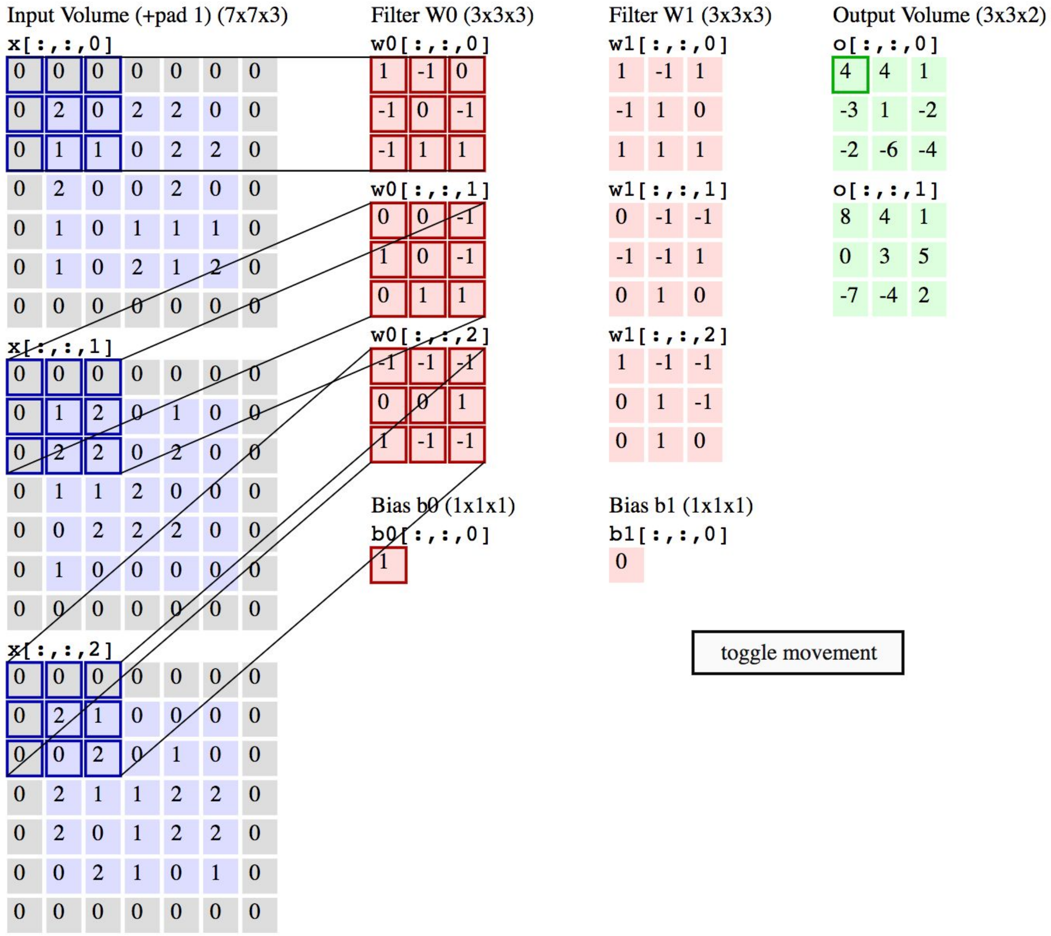
Setting future trends: GPUs + Deep Learning





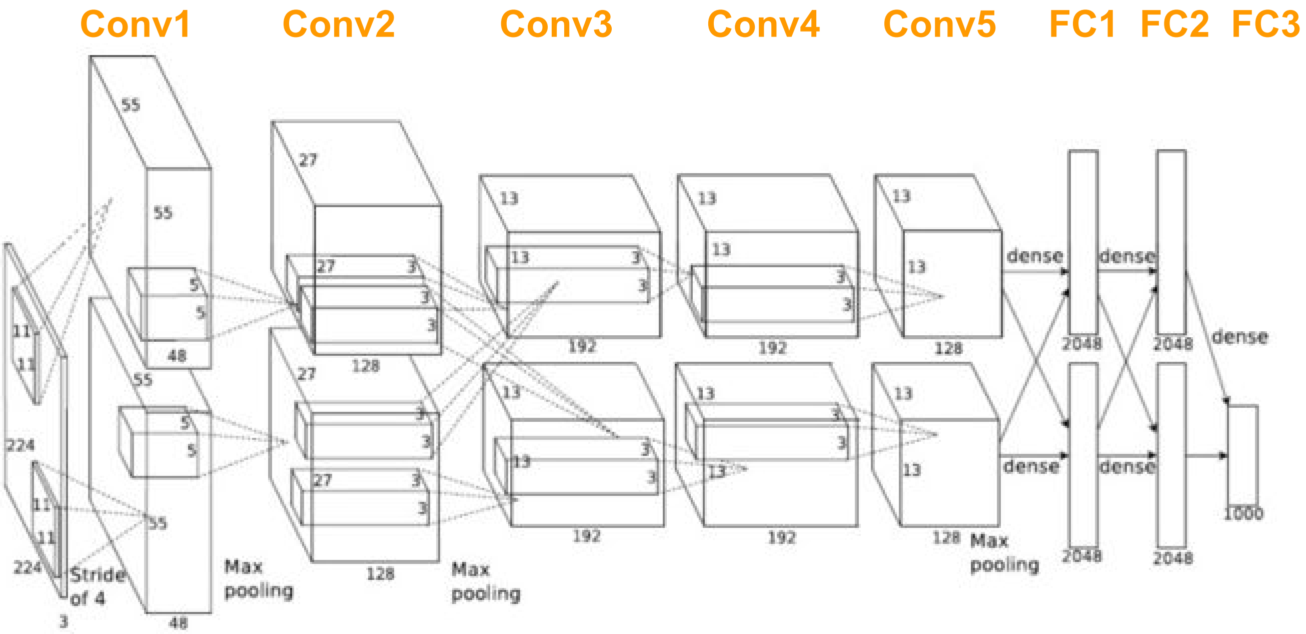
Aside:
Convolution Layers

- Element-wise multiplication
- Add all terms
- x * w [:, :, 0] =
- x * w [:, :, 1] =
- x * w [:, :, 2] =
- x*w + bias =

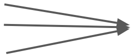


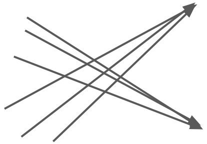
Architecture

- 8 layers: 5 Conv layers, 3 Fully connected (FC)
- Output is fed to a 1000-way softmax function



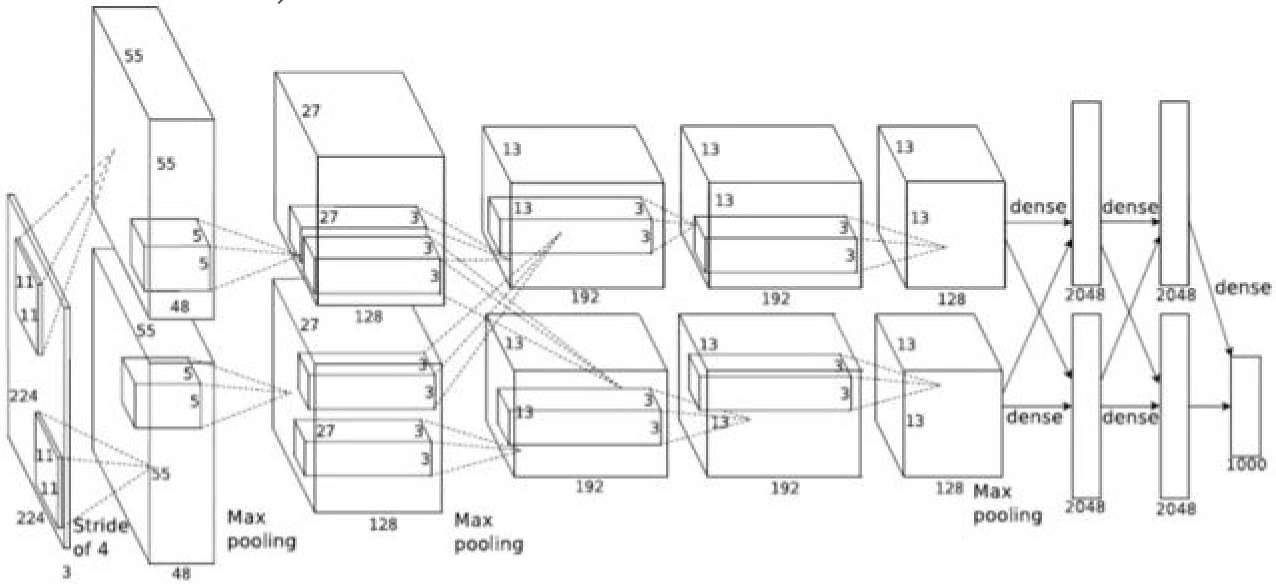
Training on GPUs

Intra GPU connection:  2nd, 4th, and 5th layers

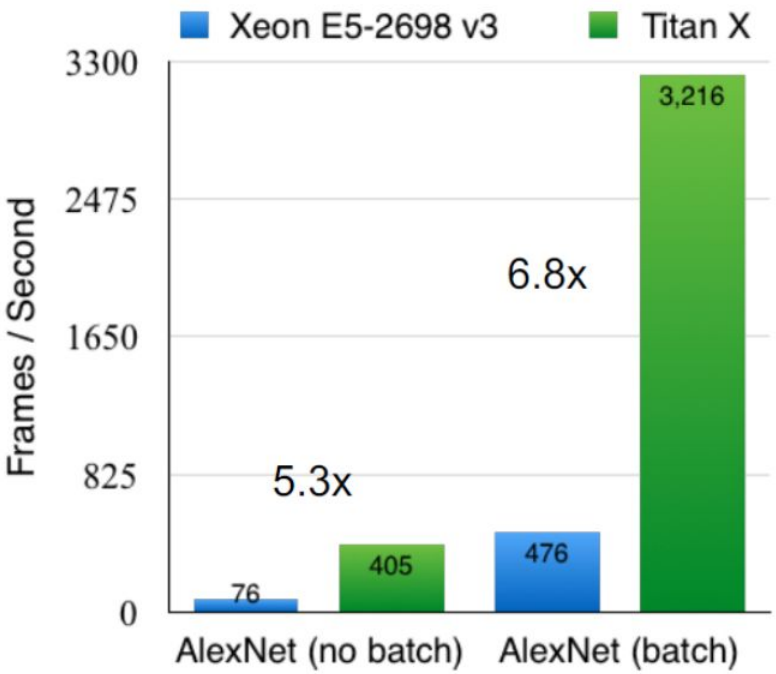
Inter GPU connection:  3rd, 6th, 7th, 8th layers

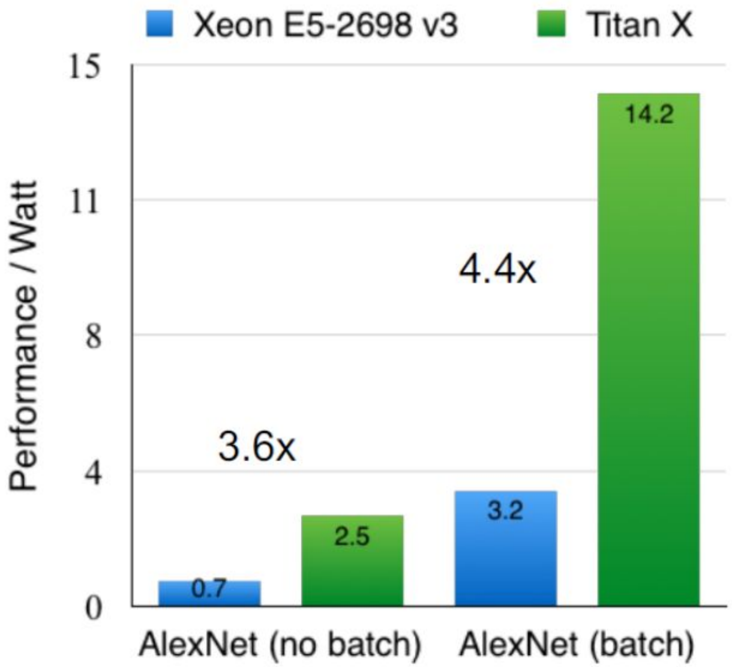
GPU #1

GPU #2



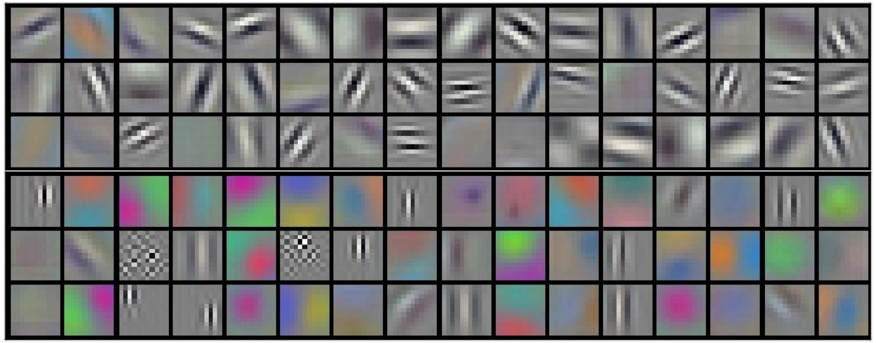
Performance: CPUs vs GPUs

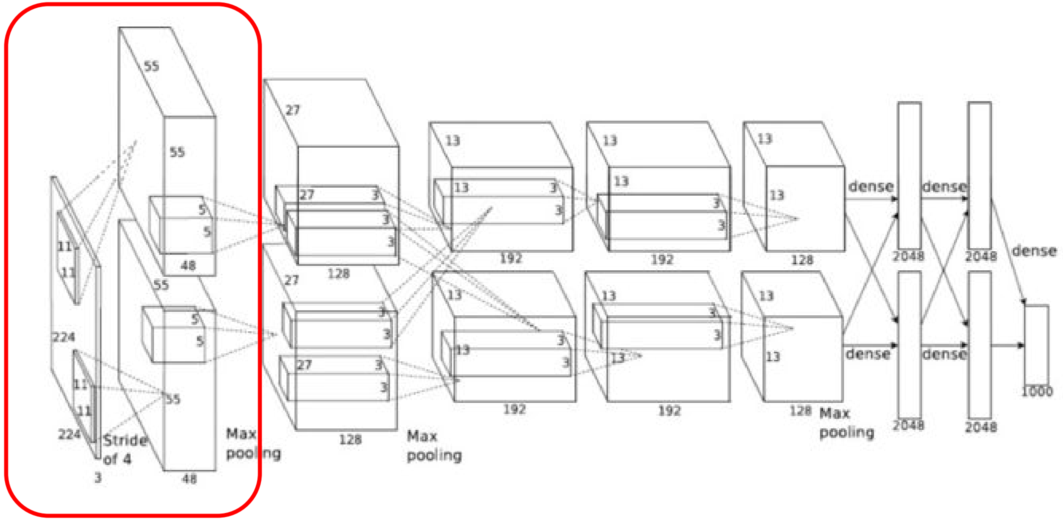




First Convolution Layer

- Input: 224x224x3
- 96 11x11x3 filters with stride of 4
- Output? (W - F) / S + 1



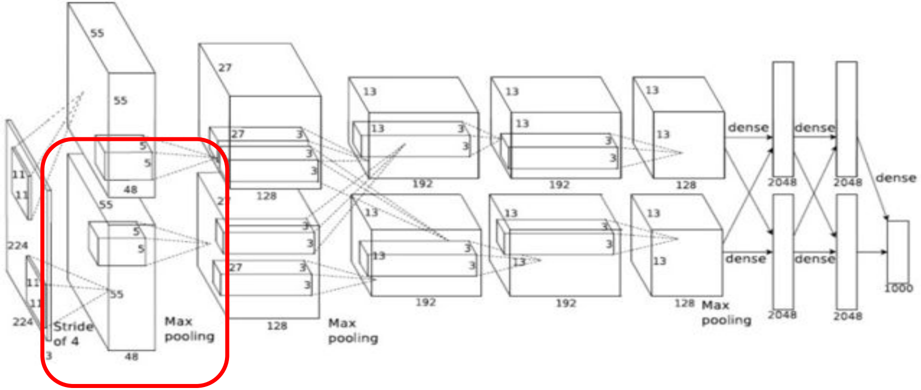


Overlapping Pooling

- AlexNet uses **overlapping max pooling** after the first and second convolution layer
- The “pooling filter” is 3x3 with a stride of 2
- Input: 55x55x48
- Output? (W - F + 2P) / S + 1

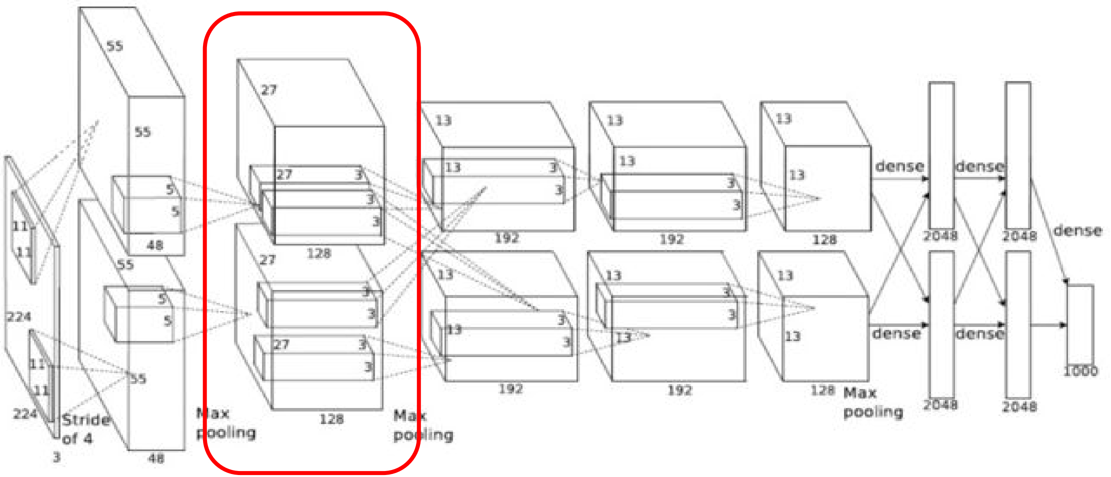
1	2	2	3	4	5	8
7	6	1	2	0	7	9
2	3	4	5	6	1	2

7		



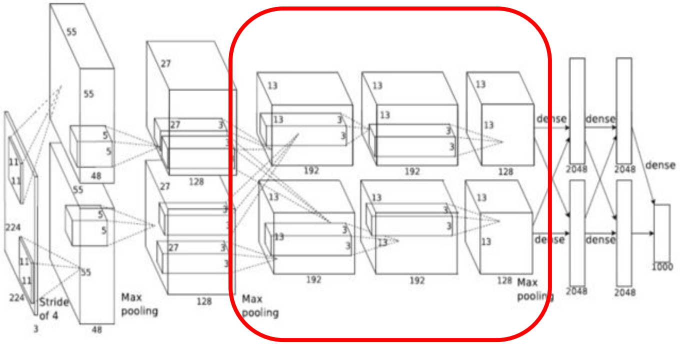
Second Convolution Layer

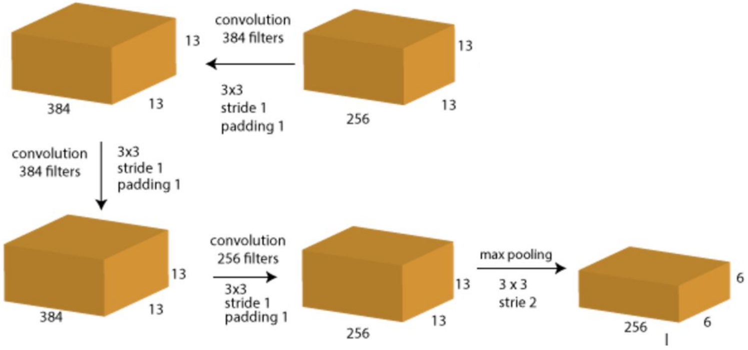
- Layer 2 Input: 27x27x48 (on each GPU)
- 256 5x5x48 filters
- Padding = 2, stride = 1
- Output? (W - F + 2P) / S + 1



Third to Fifth Convolution Layers

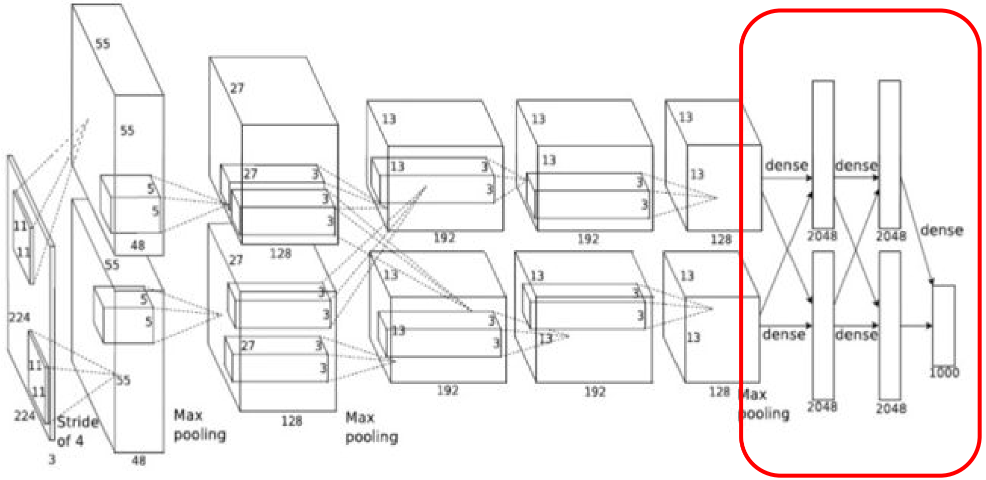
- Stride = 1, Padding = 1
- Layer 3:
  - Input: 13x13x128 (on each GPU)
  - 384 3x3x256 filters
  - Output: 13x13x384 (or 13x13x192 on each GPU)
  - Cross-GPU Layer: All output goes through all filters
- Layer 4:
  - 384 3x3x192 filters
  - Output: 13x13x384 (or 13x13x192 on each GPU)
- Layer 5:
  - 256 3x3x192 filters
  - Output: 13x13x256 (or 13x13x128 on each GPU)
  - After pooling: 6x6x256





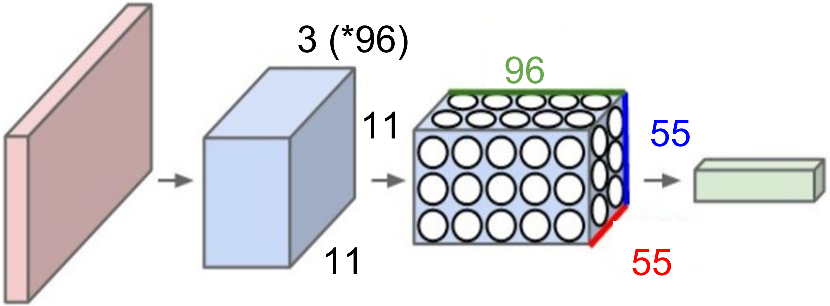
Layers 6, 7, 8: Fully Connected Layers

- All FC Layers are cross-GPU: all inputs go through all neurons
- Layer 6
  - Input: 6x6x256
  - 4096 neurons
  - Output: 4096
- Layer 7
  - 4096 neurons
- Layer 8
  - 1000 neurons
  - Softmax to get final scores
  - Final output: 1000x1 vector
  - ImageNet has 1000 classes



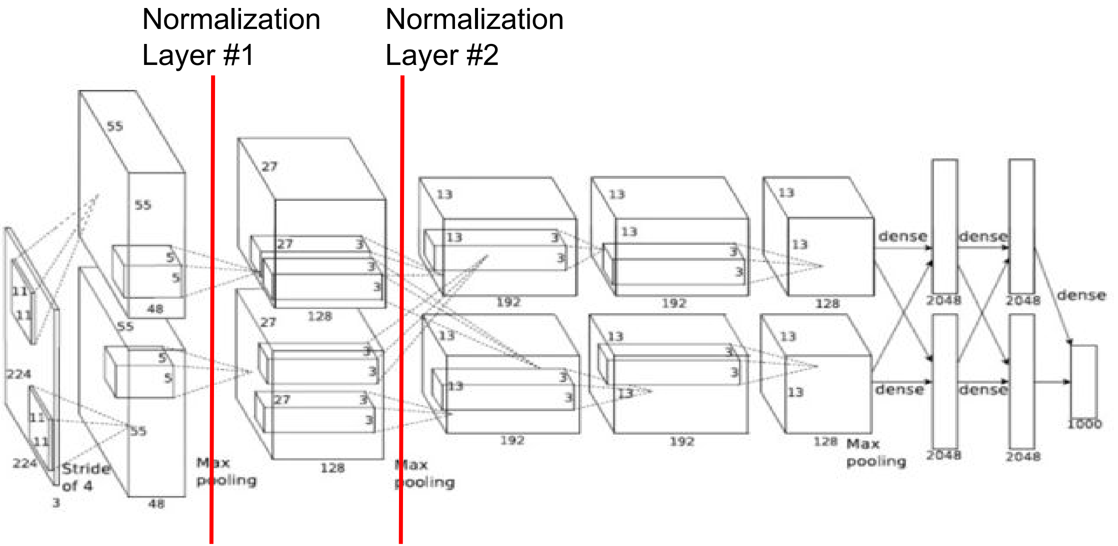
Parameters in AlexNet: First Conv Layer

- 96 11x11x3 filters
- Output: 55x55x96
- How many parameters?
  - 11*11*3 = 363 weights per filter
  - 55*55*96 = 290,400 neurons
  - 290,400 * 364 = 105,705,600 parameters!
- Parameter sharing
  - Neurons of each depth slice share the same weights and bias
  - We now have 96*11*11*3 = 34,848 parameters instead of ~100 million
- AlexNet ends up having 60 million parameters and 650,000 neurons!



Response normalization

- Normalize after ReLU before pooling
- “Brightness normalization”
- Difference from batch normalization
  - Internal covariate shift



Response normalization

b - response normalized activity

a - activity of a neuron by applying kernel (or filter) i at position (x, y)

N - number of kernels in a given layer

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0, i-n/2)}^{\min(N-1, i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

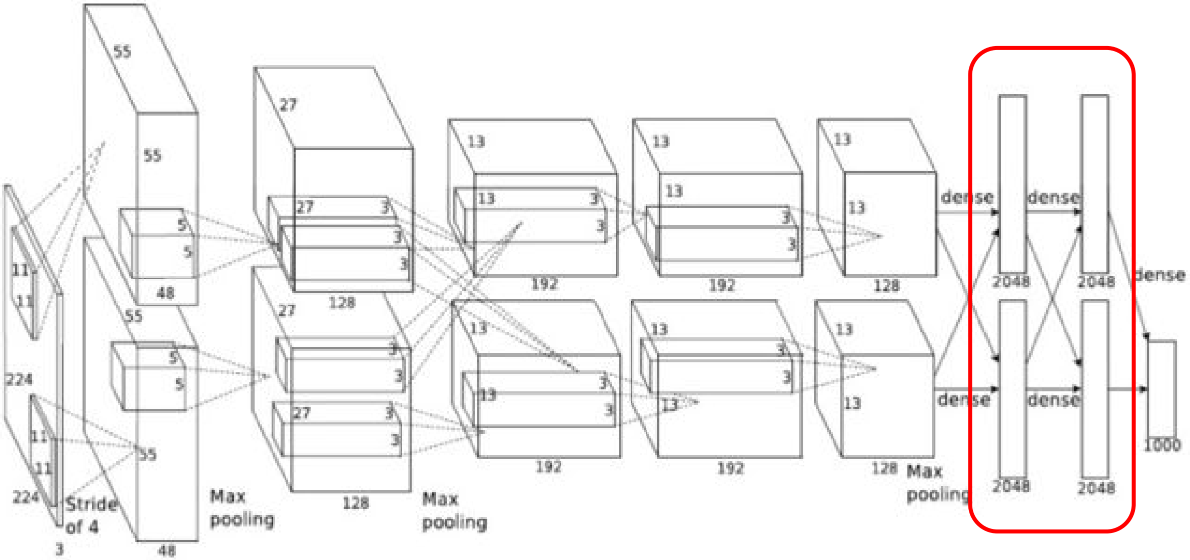
Hyper-parameters: k = 2, alpha = 10^-4, beta = 0.75, n = 5

Reducing Overfitting: Data Augmentation

- Take 5 224x224 patches from original 256x256 images
  - Each corner + center
  - Take horizontal reflections of each of these
  - Dataset is now 10x bigger
  - Average scores of 10 patches at test time
- PCA performed on RGB pixel values

Reducing Overfitting: Dropout

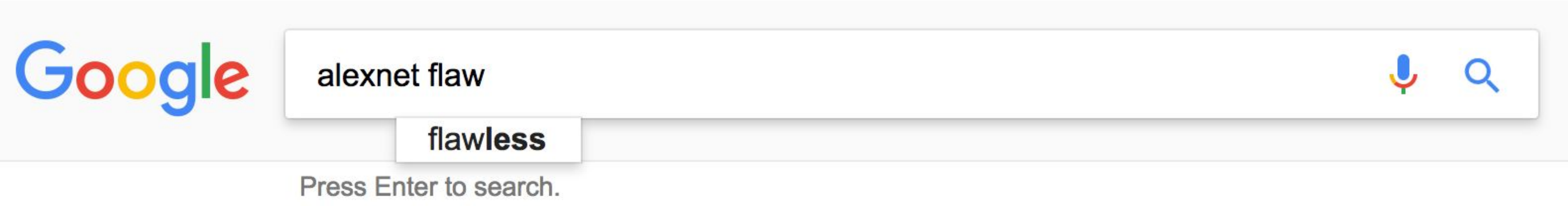
- Dropout used on first 2 fully-connected layers
- p = 0.5
- Overfitting observed without dropout



More Details

- Update rule: Stochastic Gradient Descent + Momentum
- Batch size: 128
- Weight Decay: 0.0005
- Learning rate: 0.01
- Weights initialized from Gaussian distribution (mean=0, std dev=0.01)
- Neuron biases initialized to 1 to provide ReLU with positive inputs
- Training set size: 1.2 million images
- Run for ~90 cycles
- Took 5~6 days to run

Weaknesses

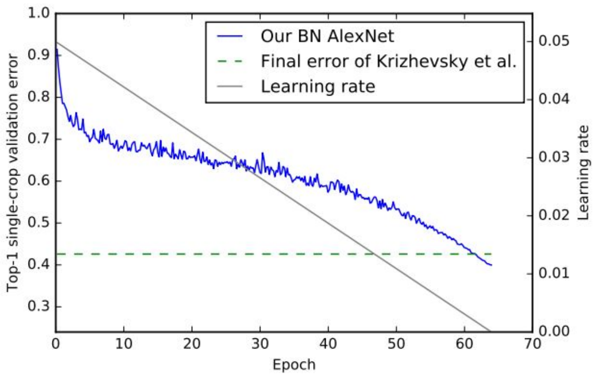


Room for Improvement

- Batch Normalization
  - Simon et. al. added a batch normalization layer between each convolutional and activation unit layer, and removed the local response normalization and dropout layers.
- Not great for object detection (bounding box)
- Shallow network by today's standards

Table 1. **Single-crop** top-1 and top-5 error of our models on the validation set of ILSVRC 2012.

Model	Top-1 error Ours	Top-1 error Original	Top-5 error Ours	Top-5 error Original
AlexNet	**39.9%**	42.6%	**18.1%**	19.6%
VGG19	**26.9%**	28.7%	**8.8%**	9.9%
ResNet-10	**36.1%**	–	**14.8%**	–
ResNet-50	**24.6%**	24.7%	**7.6%**	7.8%



Questions

- How does each filter in a layer learn different weights?
  - Random Initialization

Works Cited

- http://cs231n.github.io/convolutional-networks/
- http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghee.pdf
- https://www.analyticsvidhya.com/blog/2016/04/deep-learning-computer-vision-introduction-convolution-neural-networks/
- Simon, Marcel, Erik Rodner, and Joachim Denzler. "ImageNet pre-trained models with batch normalization." arXiv preprint arXiv:1612.01452 (2016).
- Demo: http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/