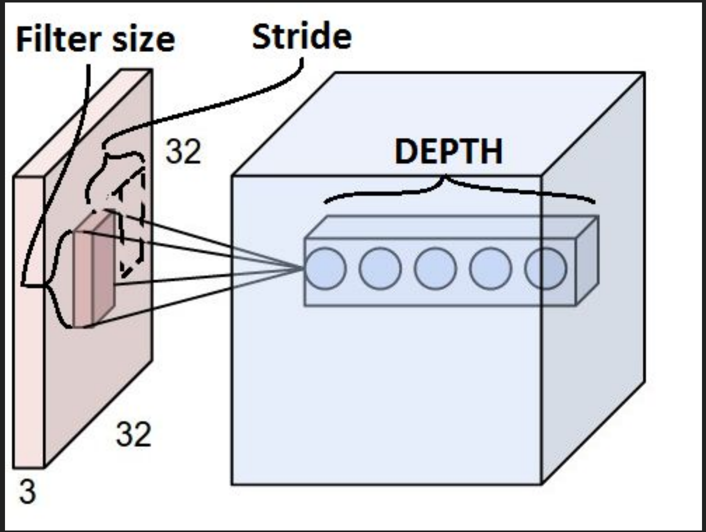
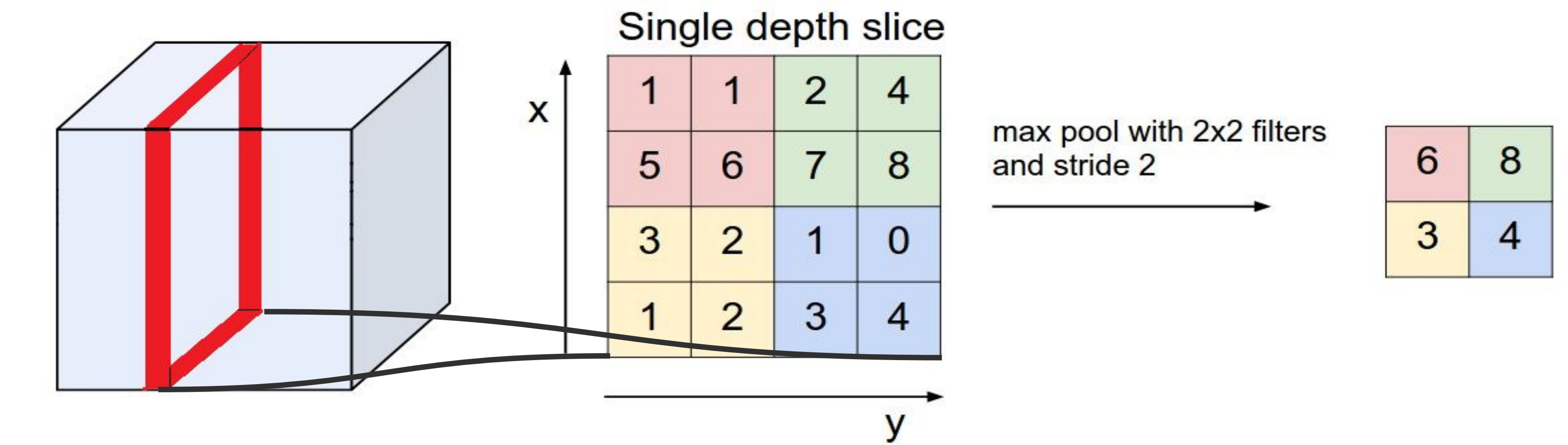
AlexNet and ZFNet

Recap: convolutional networks

- Filter size, depth, stride, max-pooling, parameter sharing





How this will go

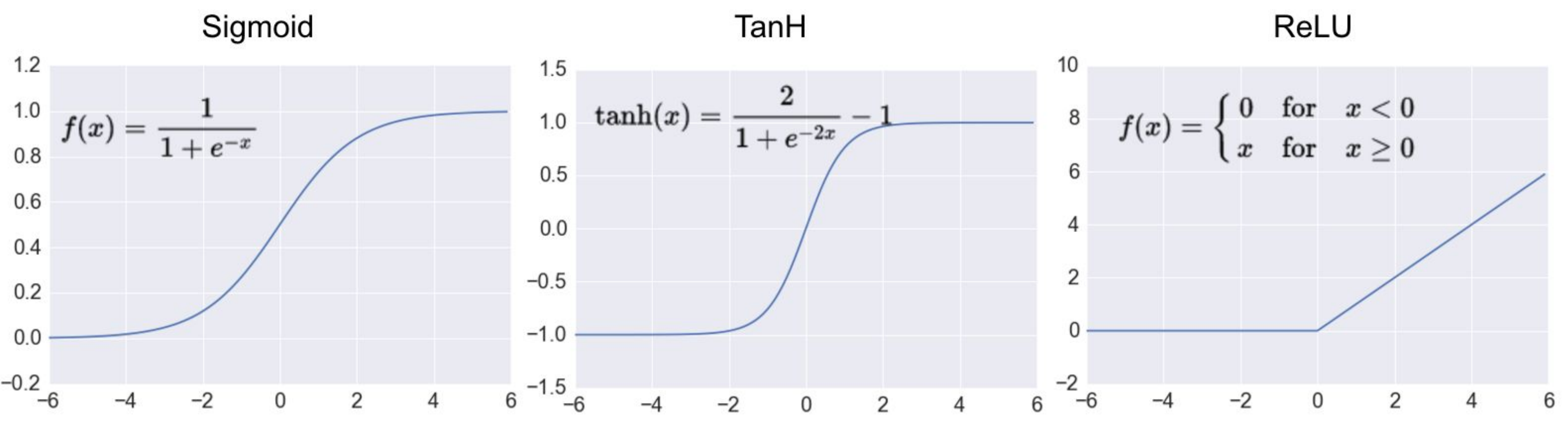
- I’m going to explain what they did, and you guys will tell me why it made sense

The dataset for AlexNet

- ImageNet has 15 million images in 22,000 categories
  - Old machine learning methods such as SVMs can’t handle this well
- New GPU technology allowed for parallelizing Convolutional Neural Network (CNN) training
- They used 1.2 million samples and 1000 categories for training
- They downsampled all images to 256x256 pixels
  - Why do we need consistent sizes?

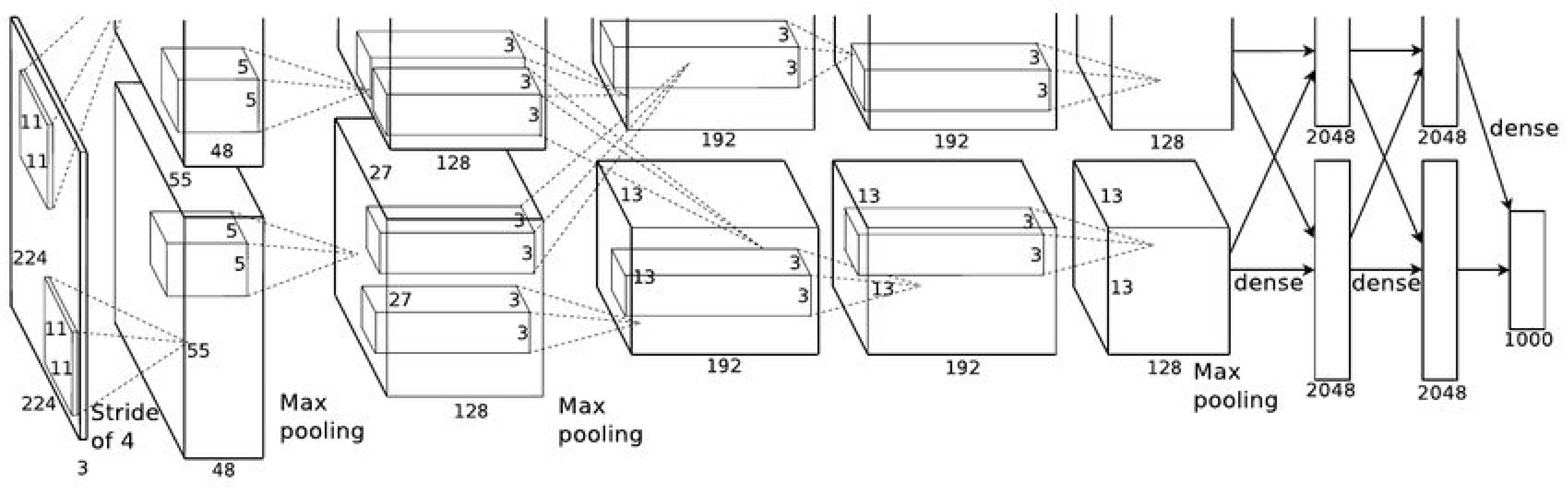
The activation function

- They used ReLU's instead of sigmoid or tanh as their activation function



Splitting across GPUs

- ~million training examples
- Used two GPUs, put half of neurons on each
  - GPUs communicated findings at third layer



Local Response Normalization

- The predecessor to batch-norm; it is useful when using unbounded activation functions (such as relu which goes 0-∞)
- Batch-norm sets each hidden layer to 0 mean, unit variance
  - And adds two learnable parameters
- LRN favors the most acti-vated neurons in their res-pective neighborhoods

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

where

$b^i_{x,y}$ – regularized output for kernel i at position x, y

$a^i_{x,y}$ – source ouput of kernel i applied at position x, y

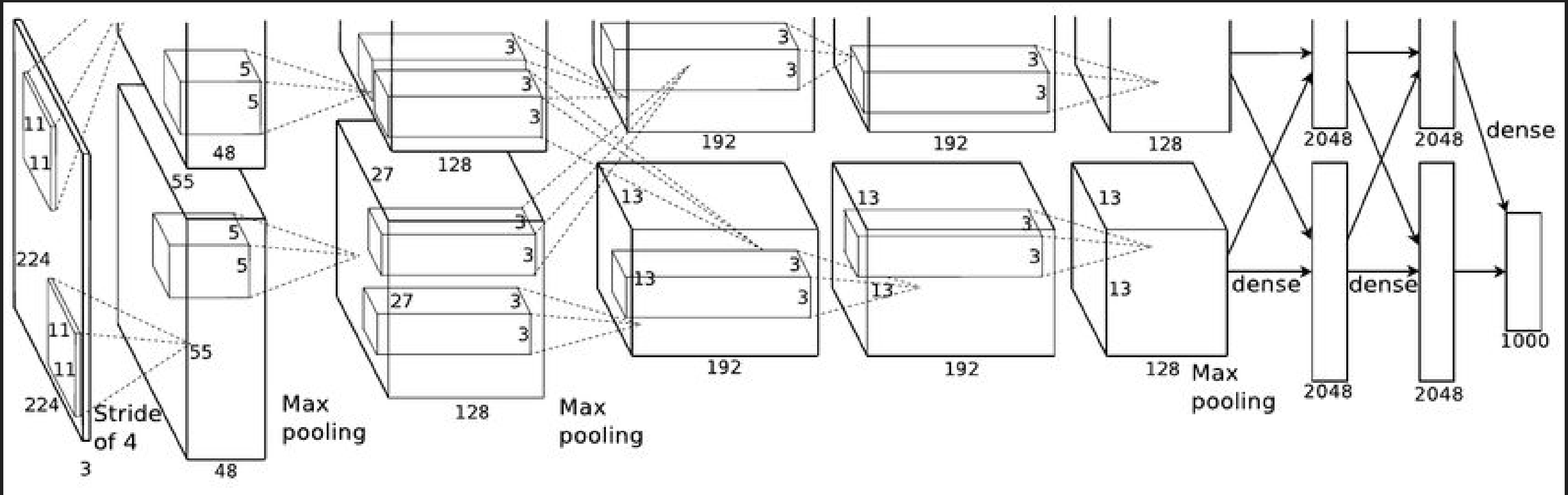
N – total number of kernels

n – size of the normalization neigbourhood

$\alpha, \beta, k, (n)$ – hyperparameters

The Architecture

- 5 convolutional layers, followed by 3 fully connected layers
- Why do convolutions get narrower/deeper further in network?
- What purpose do the three regular layers serve?
- What is the last layer's output?
- I thought the input was 256x256... Why's it 224x224?



Data Augmentation to Reduce Overfitting

- Take several random 224x224 images from each 256x256 image
  - With their horizontal flips as well
- Then, modify the RGB values linearly using PCA and eigenvalues
  - This scales the color intensities up and down randomly
  - Why is this helpful?

Dropout (p = 0.5) to reduce Overfitting

- What is dropout?
- Why does it take twice as long to train?
- Why does it reduce overfitting?

AlexNet Optimizer

- Momentum Gradient Descent
- Weight Decay

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} := w_i + v_{i+1}$$

where i is the iteration index, v is the momentum variable, ϵ is the learning rate, and $\left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$ is the average over the ith batch D_i of the derivative of the objective with respect to w, evaluated at w_i.

AlexNet Weight Initialization

- Biases all initialized to 1, so the ReLU activation function will work its magic
  - Why does >0 bias help ReLU?
- Weights initialized to mean 0 and variance 0.01
  - Why aren’t the weighs all the same values?

Results

- This paper is the reason that deep learning is such a hot discipline right now
- They got 16.4% error compared to 2nd place's 26.1%
  - Using an unconventional strategy!
- Subsequent years saw results go as low as 5% using spinoffs of AlexNet
- Drawbacks:
  - Only recognize from 1000 categories
  - No explanation of context

Cool! Let’s move on to the second paper, ZF-Net

- ZF stands for Matthew Zeiler and Rob Fergus, the NYU researchers who made it
- Won the ImageNet challenge the year after AlexNet using a similar architecture
- Paper’s official name is Visualizing and Understanding Deep Neural Networks
  - Made methods to see what a convolutional net ‘knows’

Post-AlexNet

- Improvement of neural networks little more than trial-and-error
  - But trials take a week or two to train!
- ZF-Net introduced visualization techniques that show what neurons are stimulated by specific inputs
  - “Deconvnet”

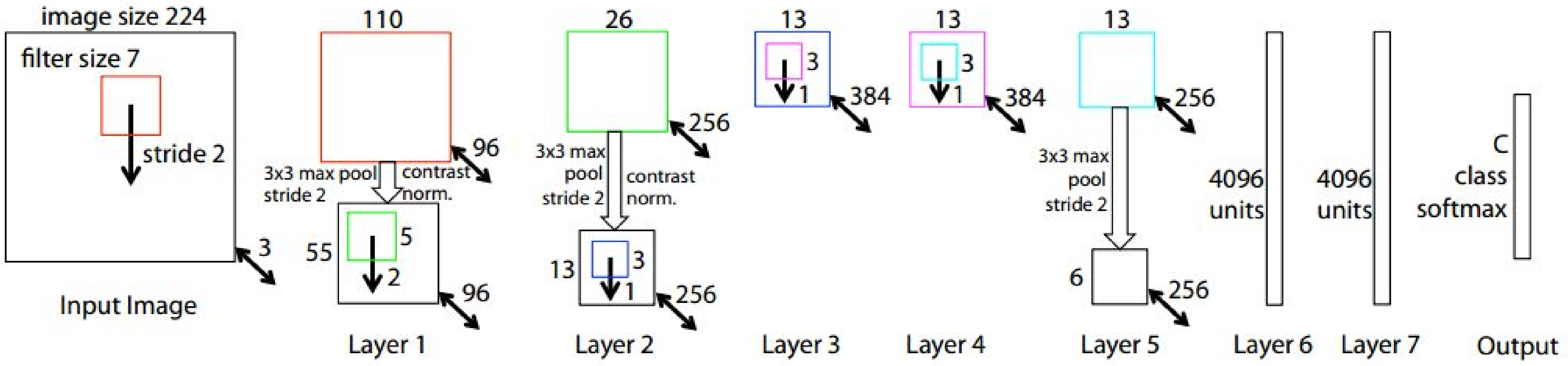
Deconvnet

- At every layer, we map the activated neurons back to the pixels
- **Unpooling**: How do we reverse the max pooling layer?
  - Loss of information?
- **Rectification**: ReLU keeps everything positive
- **Filtering**: Which neurons from the previous layer contributed to neuron_i in the current layer

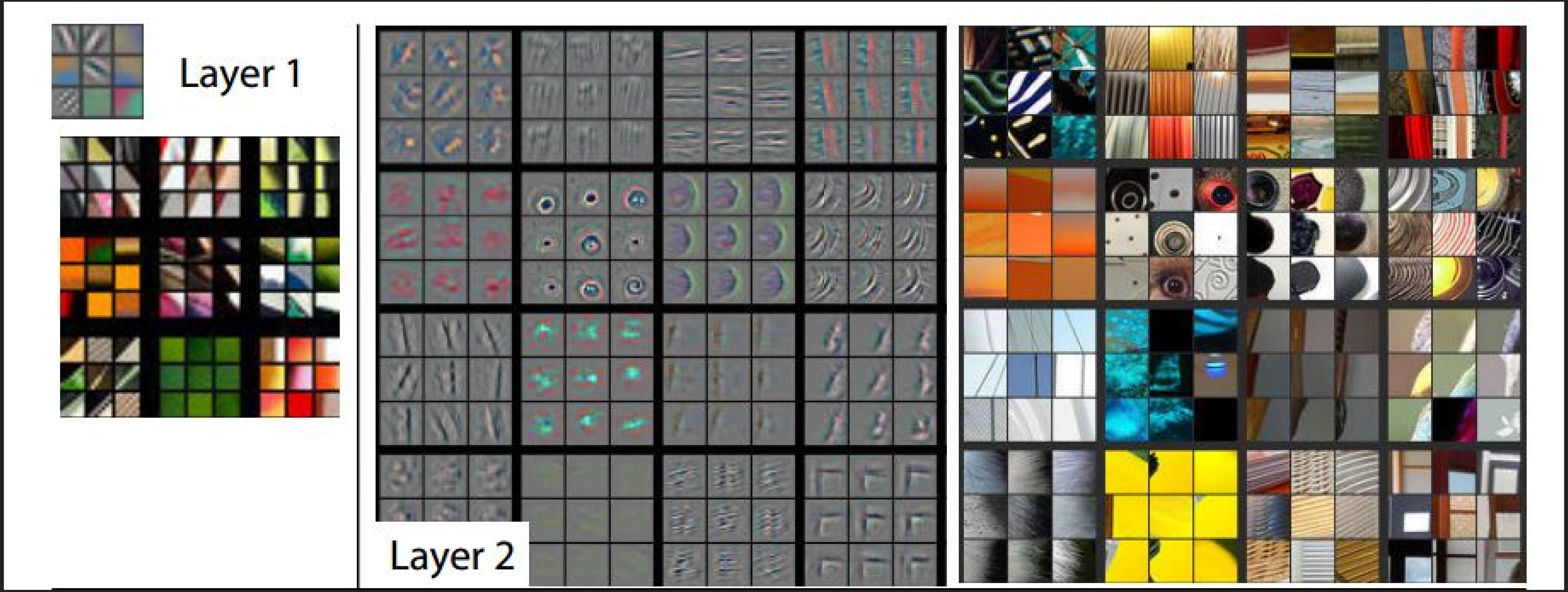
The reconstruction we obtain of the pixels is not complete! The pixels are instead shown weighted by their contribution

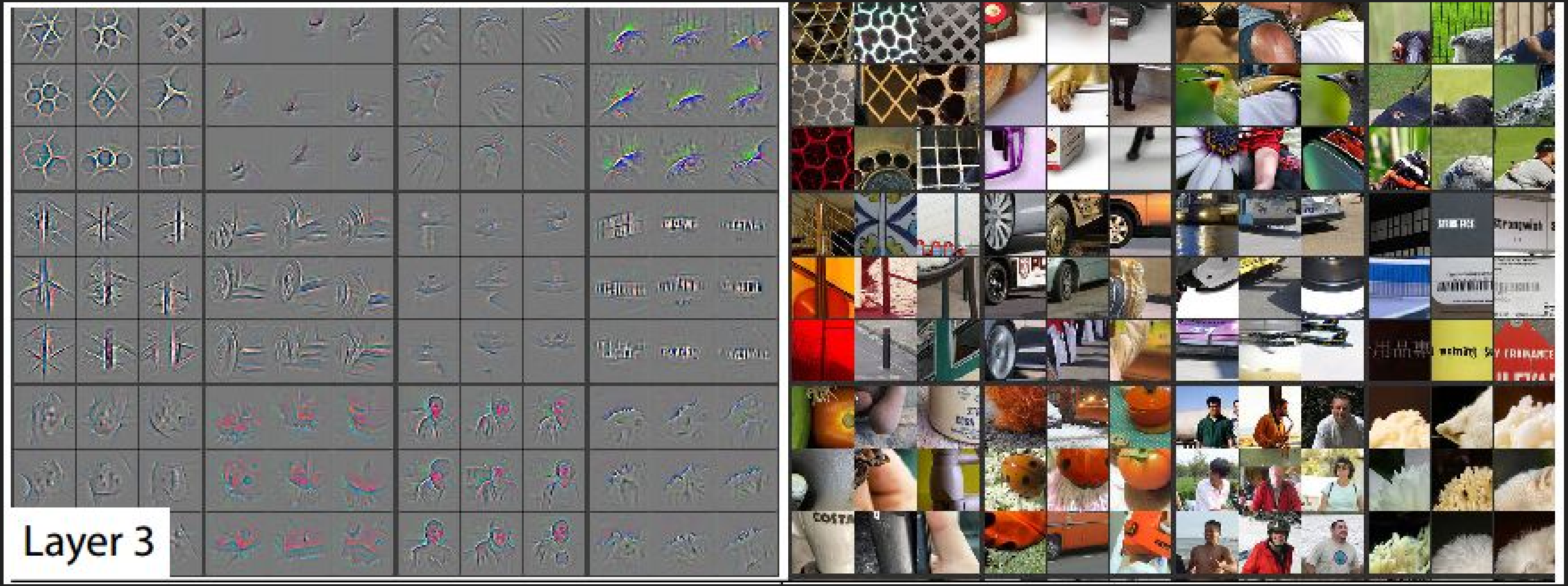
ZF-Net Architecture

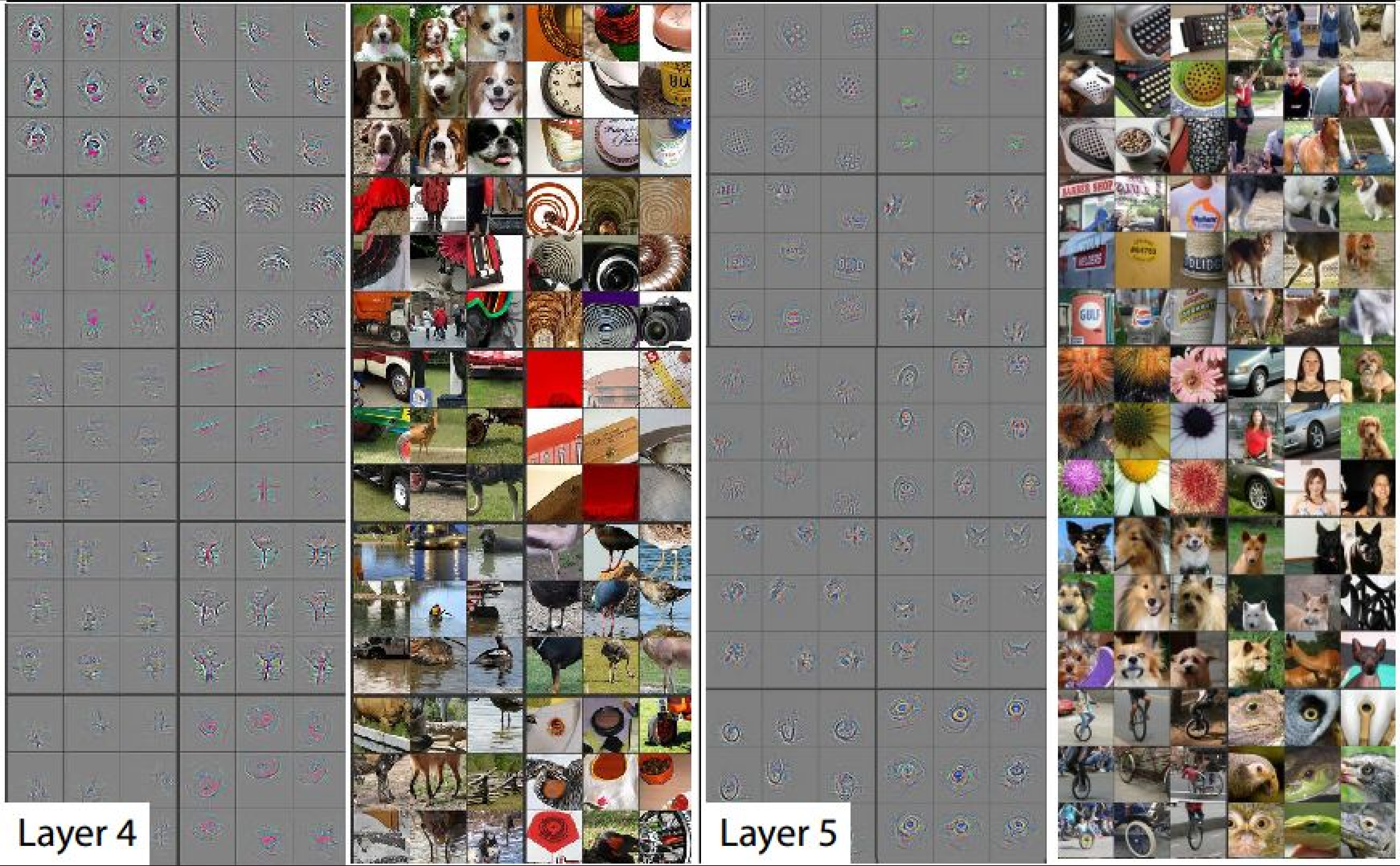
- Their architecture started as the AlexNet architecture
- Looked at deconvnet results on AlexNet
- Changed architecture as necessary



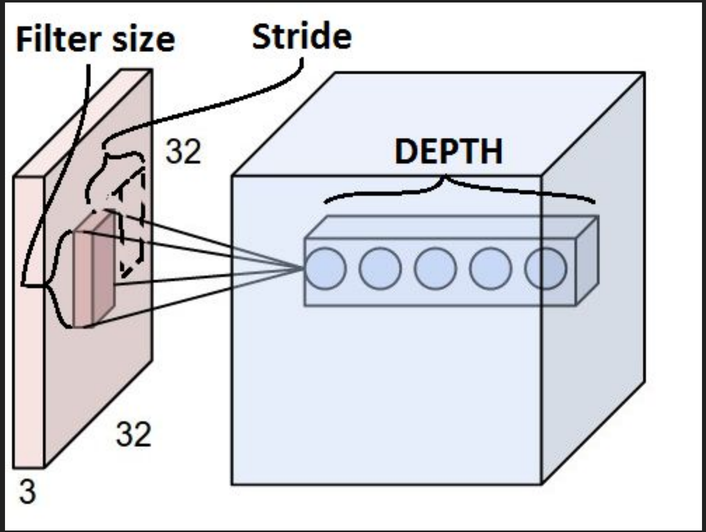
AlexNet architecture deconvnet results

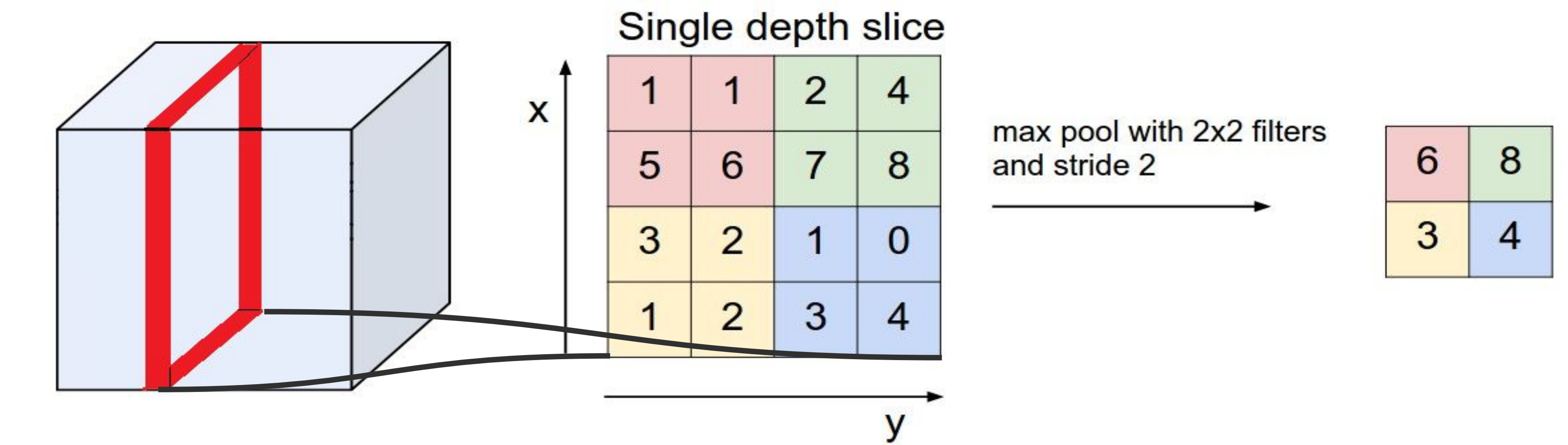



Layer 3


Layer 4
Layer 5

Reminder of CNN method



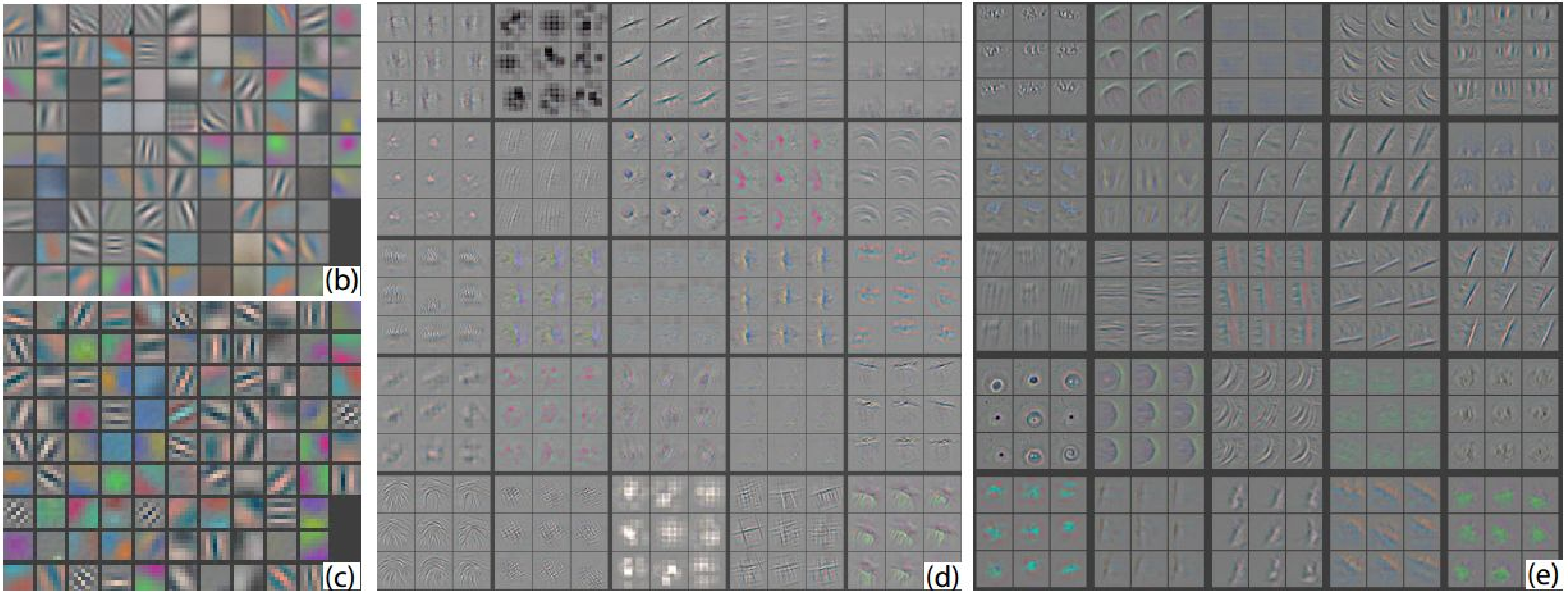


Changes based on deconvolutions

- First Layer responds only to stark differences in intensity
  - Little information passed on about medium intensity
  - Second Layer then has trouble putting together complicated patterns
- Old architecture did 11x11 squares and stride of 4 for first layer
- New architecture does 7x7 squares and stride of 2
  - Why does this fix the problem?

c, e are 1st, 2nd layers of ZF-Net; b, d are same for AlexNet

- ZF-Net’s features more precise by second layer, due to first layer having more detail

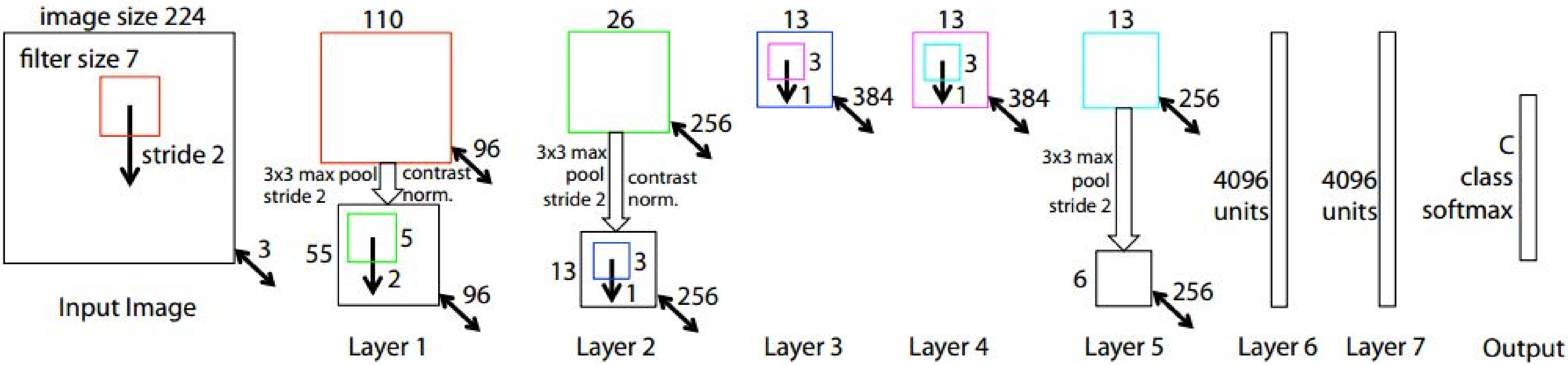


Occlusion Sensitivity

- Is the model actually learning on the important section of the image or the environment?
- Randomly covered parts of the input with a gray square
- What if we cover the part that the convnet is sensitive to?

Generalization

- Use a brand new dataset, but *keep the old neural network*
- Only allow learning on the last layer's parameters
- How does this make sense?
- Achieved 15% error using this method on brand new datasets!



Next week

- Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection
- Their code is available here (but you don't need to look at it for the paper)