Convolutional Neural Networks

Gautam Kamath

Drawbacks of fully-connected (FC) layers

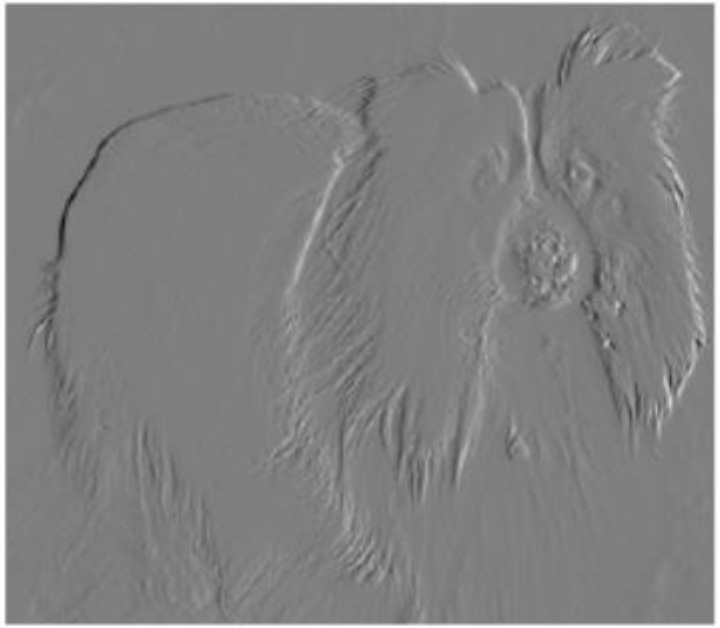
- Image classification
- Suppose 1 megapixel input (relatively small)
  - 10^6 pixels
- One hidden layer with 1000 neurons (draw)
- Total number of parameters in one layer is 1 billion
  - Compare with GPT-2, 1.5 billion parameters
- More parameters more problems
  - Huge cost to train, need lots of data, difficulties optimizing, probably overfit

Some properties of image data

- Image data has nice structure that we can take advantage of
- Translation invariance
- Locality
- Lead to following principles
  - Parameter sharing
  - Only local edges
- (Draw FC 25 params, local edges 13 params, sharing 3 params)

Convolution

- Technically, it’s the cross-correlation operator
- (Draw example: input [0 1 2; 3 4 5; 6 7 8] x kernel/map/filter [0 1; 2 3] = output [19 25; 37 43])
- Kernel [-1 1]



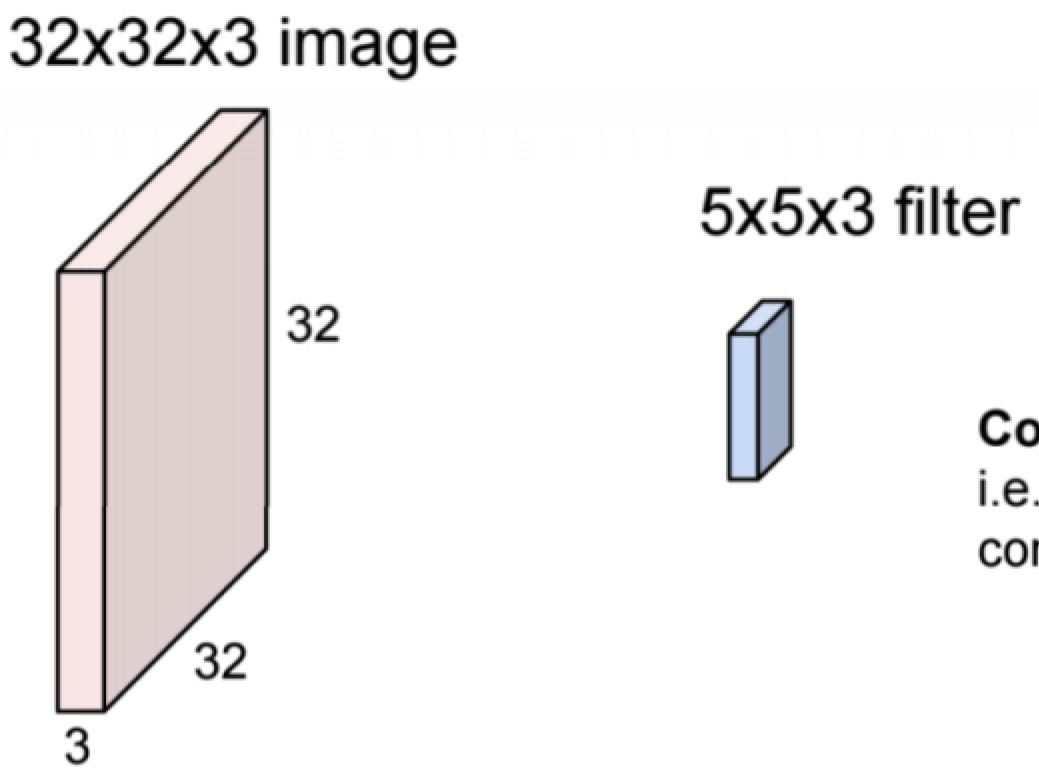
Convolutional Neural Network (CNN)

- Input: images
- 3D objects (tensor: generalization of matrix)
- Width, height, depth
  - Depth = number of channels
  - 3 for colour images (RGB), 1 for black and white
  - Will be more than 3 channels in later layers
- MNIST dataset: 28 x 28 x 1
- CIFAR-10 dataset: 32 x 32 x 3
- Output layer: 1 x 1 x c, where c is the number of classes

Types of layers

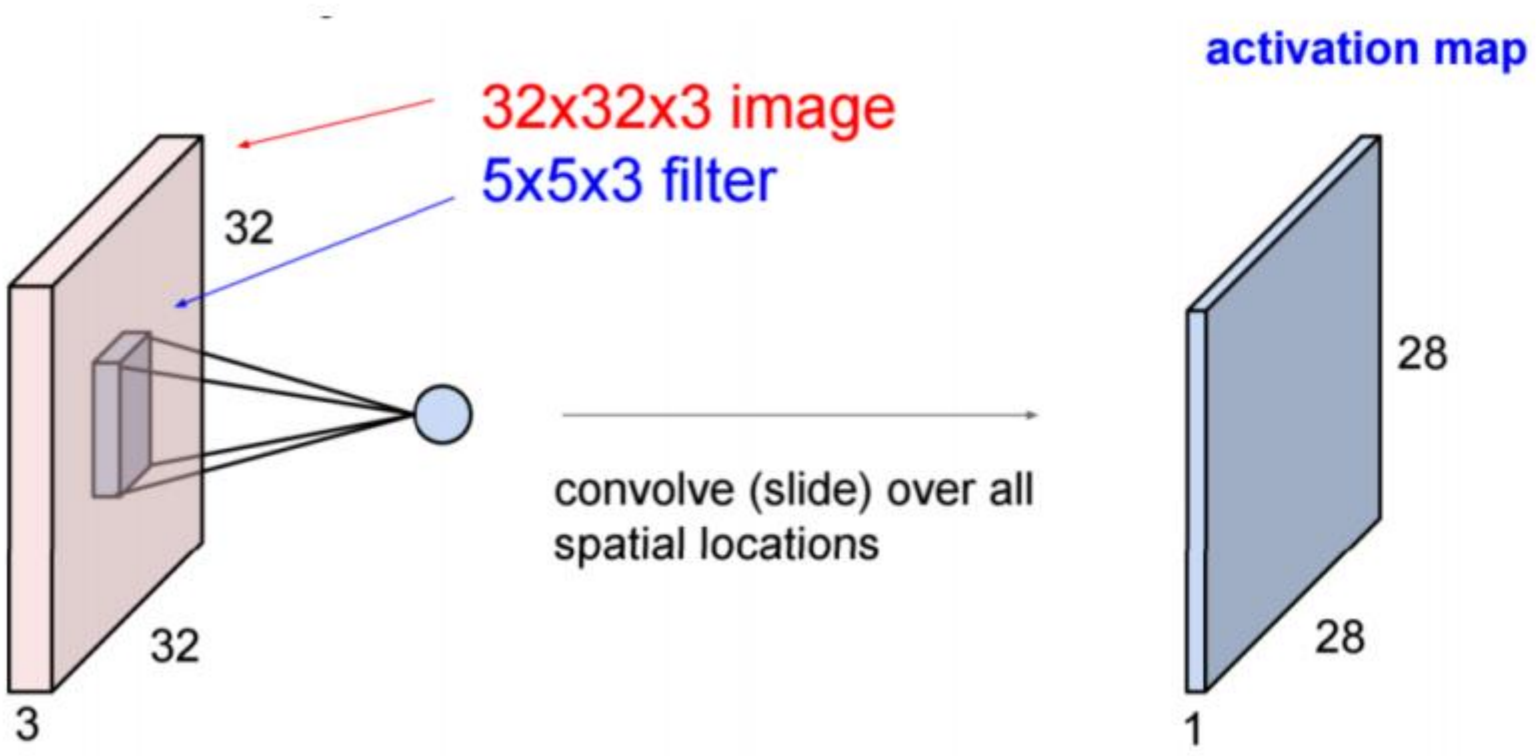
- Fully connected
- Convolutional
- Pooling

Convolutions



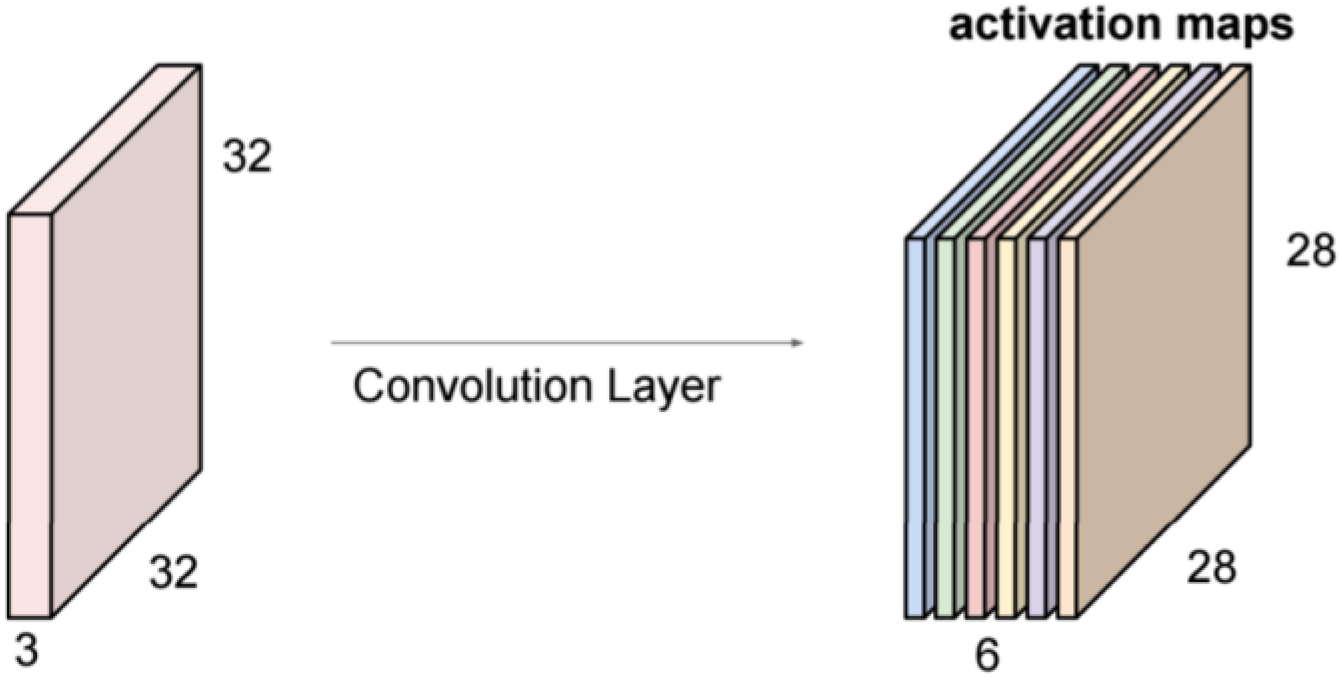
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolutions

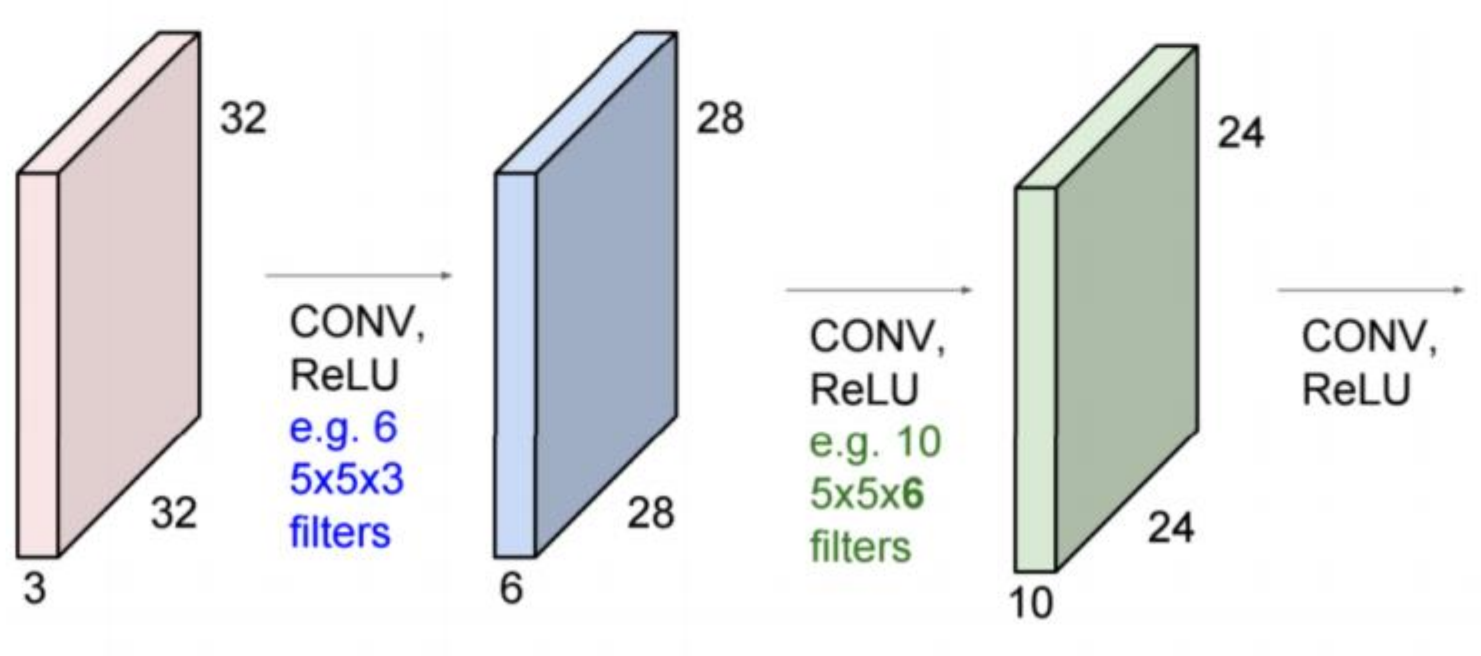


Convolutions

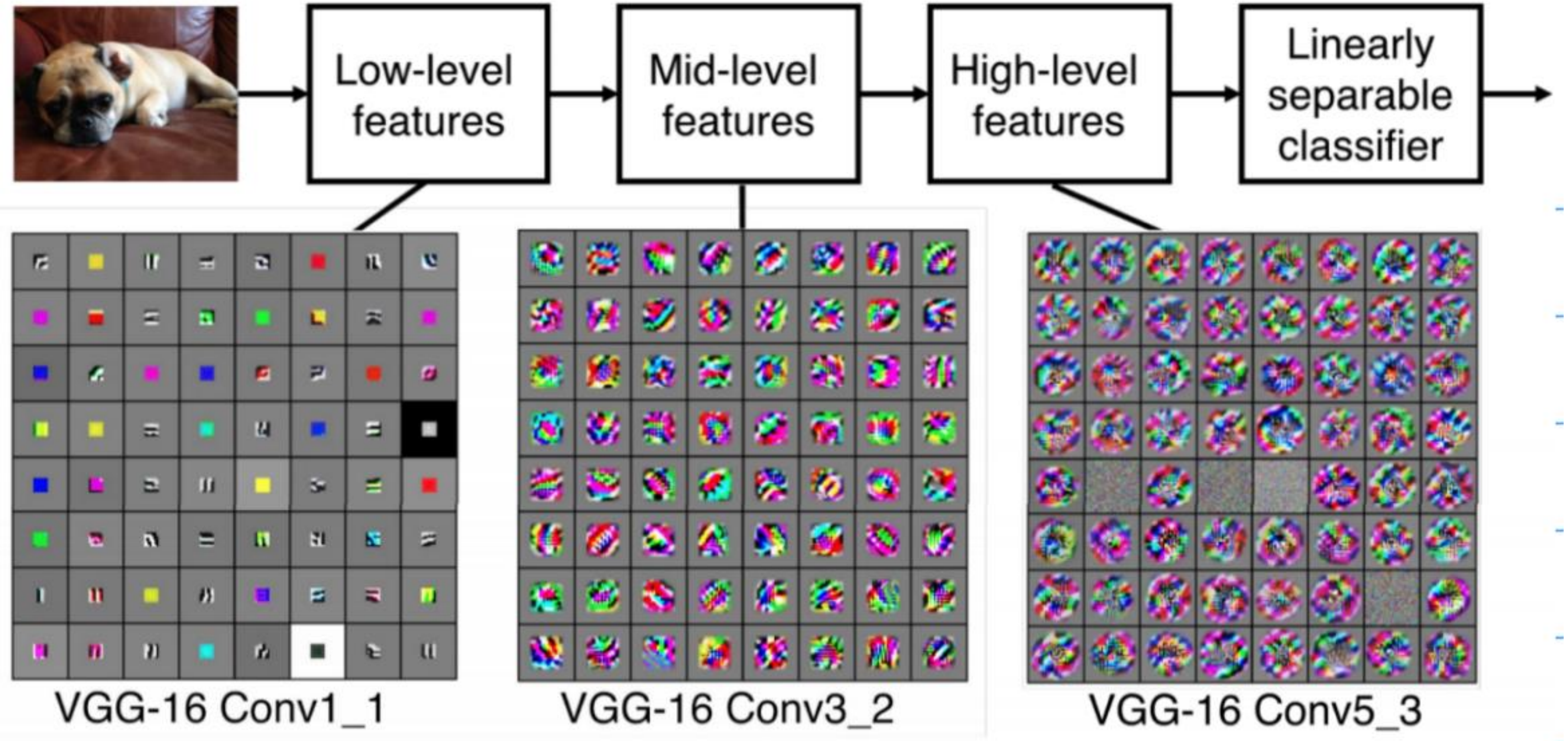
Convolutions



Convolutions



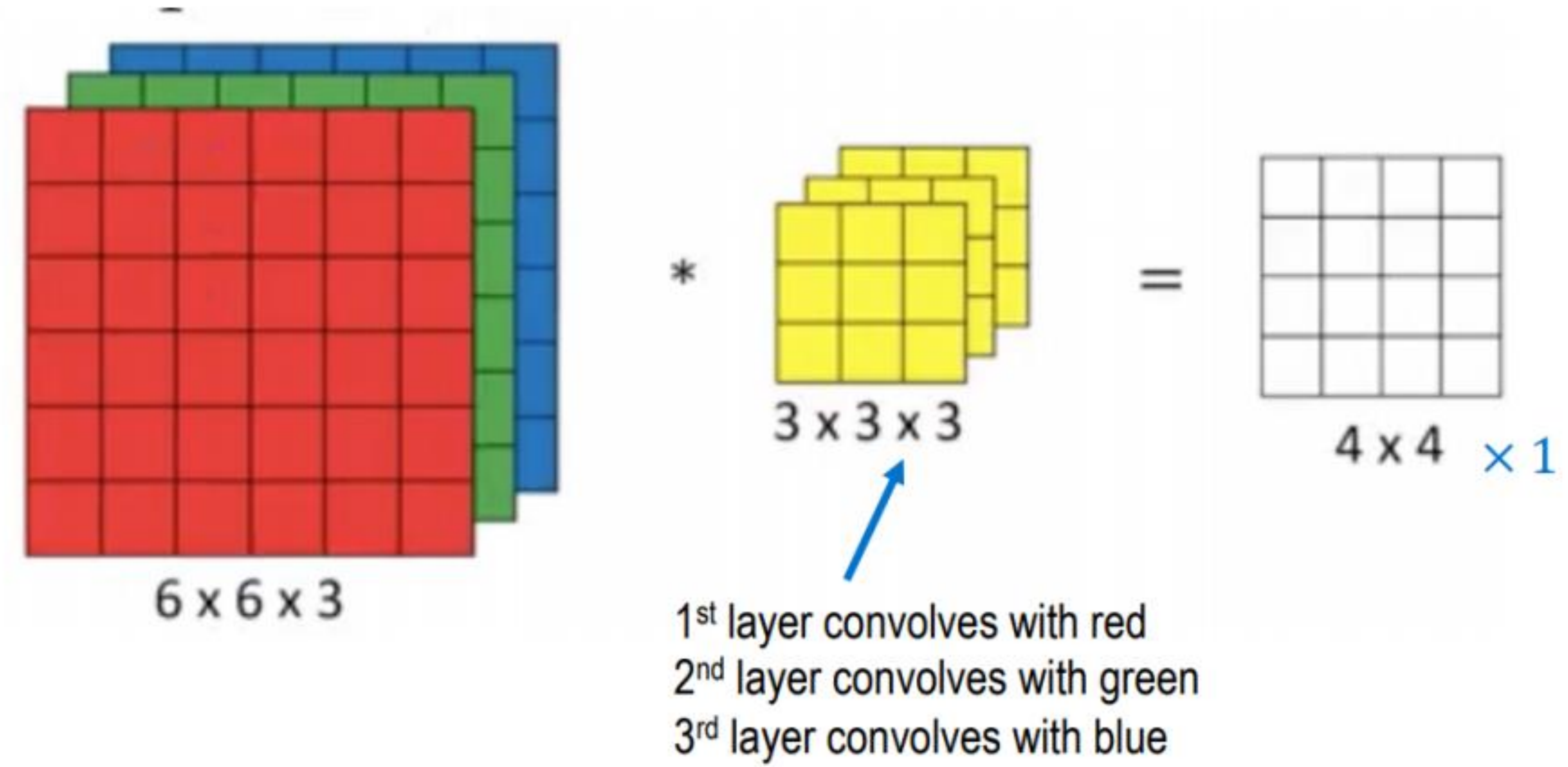
Visualizing a CNN



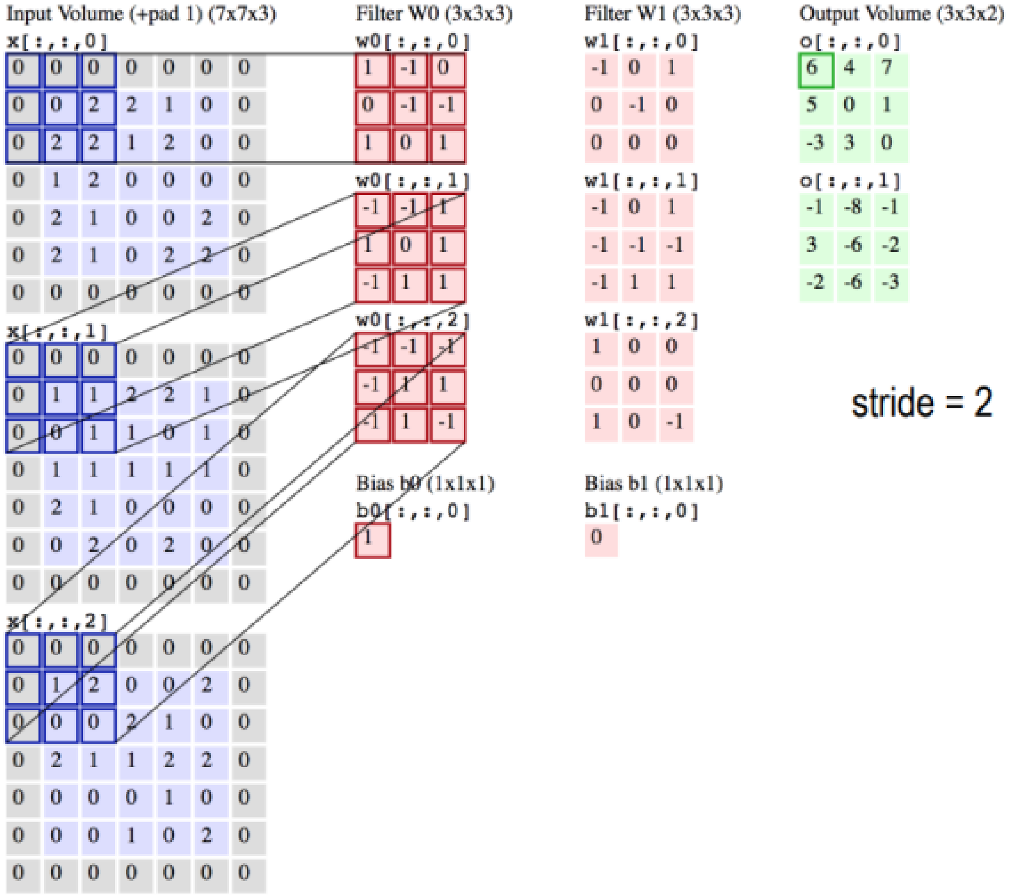
Convolutional Layer Hyperparameters

- Filter size (usually square, draw)
- Output depth (= # of filters)
- Stride (draw stride 1 vs stride 2)
- Zero padding (draw 3x3 with 3x3 filter, versus same with 1 pad)

of channels = depth of each filter



Illustrating CNNs



stride = 2

Conv Layer Summary

- Input: $W_1 \times H_1 \times D_1$
- Four hyperparameters:
  - # of filters K
  - Width/height F ($F \times F \times D_1$)
  - Stride S
  - Zero padding P
- Output: $W_2 \times H_2 \times D_2$
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$
  - $D_2 = K$
- # of parameters:
  - K filters with $F \times F \times D_1$ params
  - $F^2 D_1 K$ weights
  - K biases

Example Computation

- Input: $227 \times 227 \times 3$
- Use 96 filters, $11 \times 11 \times 3$ in size, stride 4, no padding
- Output size:
  - $W_2 = H_2 = (227 - 11 + 2 \cdot 0)/4 + 1 = 216/4 + 1 = 55$
  - $D_2 = 96$
- $(227,227,3) \rightarrow (55,55,96)$
- Number of parameters: $11^2 \cdot 3 \cdot 96 + 96 = 34944$
- If it was FC layer: (input size + 1)(output size) ≈ 45 billion
- No weight sharing, only local connections: $(11^2 \cdot 3 + 1)(55^2 \cdot 96) \approx 106\text{m}$ parameters

Pooling Layers

- (Draw max pool example, 2x2 filter w/ stride 2, [1 1 2 4; 5 6 7 8; 3 2 1 0; 1 2 3 4] -> [6 8; 3 4])
- Several types of pooling layer: max pool, average pool, L2-norm pool
- No learnable parameters
- Only 1 filter (applied independently to different channels)

Pool Layer Summary

- Input: $W_1 \times H_1 \times D_1$
- Two hyperparameters:
  - Width/height F ($F \times F$)
  - Stride S
- Output: $W_2 \times H_2 \times D_2$
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- # of parameters:
  - 0

Structure of a CNN (draw as we go)

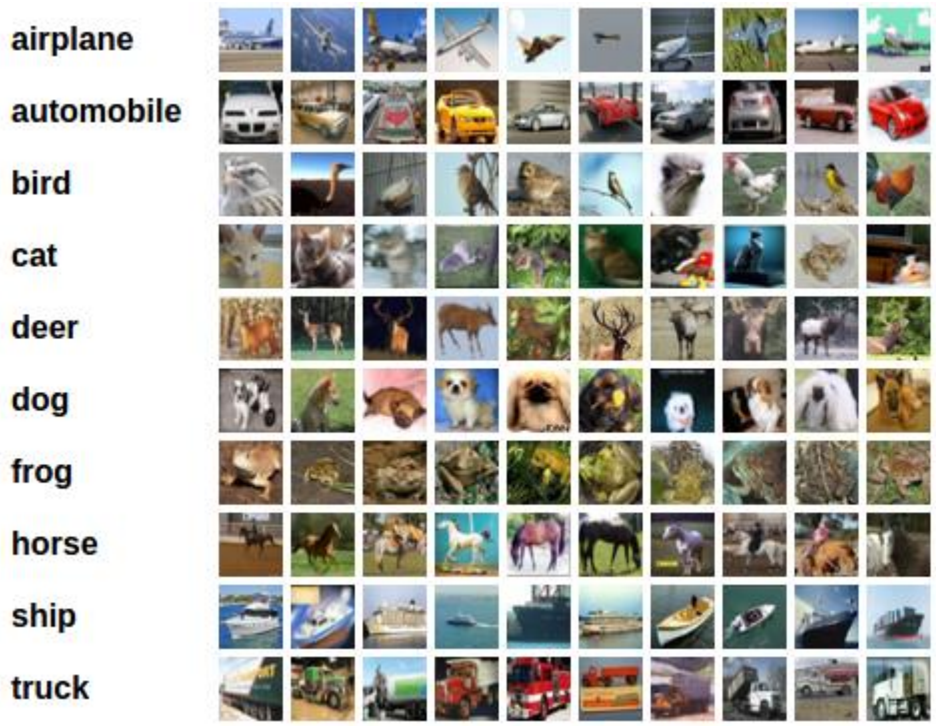
- Input
- Conv+ReLU(+pool) (repeat)
- FC+ReLU (repeat)
- FC+SoftMax

MNIST Dataset

- Digit recognition
- 60,000 training images
- 10,000 test images
- 10 classes
- 28 x 28 x 1 dimensions
- State of the art: 99.9%
  - Very easy task – most things can get 95%+

CIFAR-10 Dataset

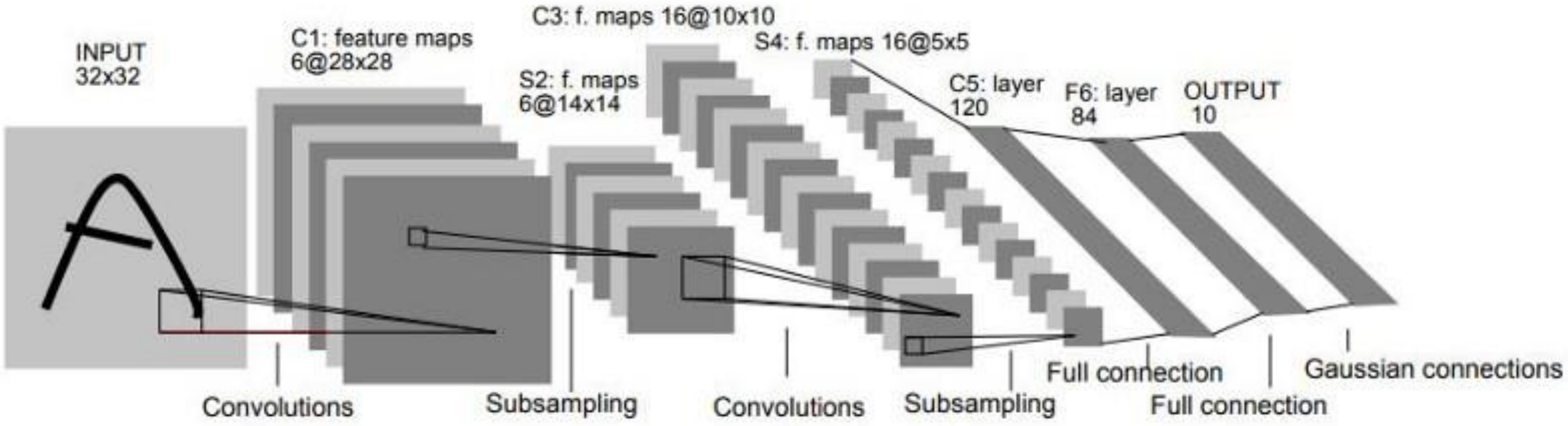
- 50,000 training images
- 10,000 test images
- 10 classes
- 32 x 32 x 3 dimensions
- Harder than MNIST: bigger images, RGB, inherently more difficult

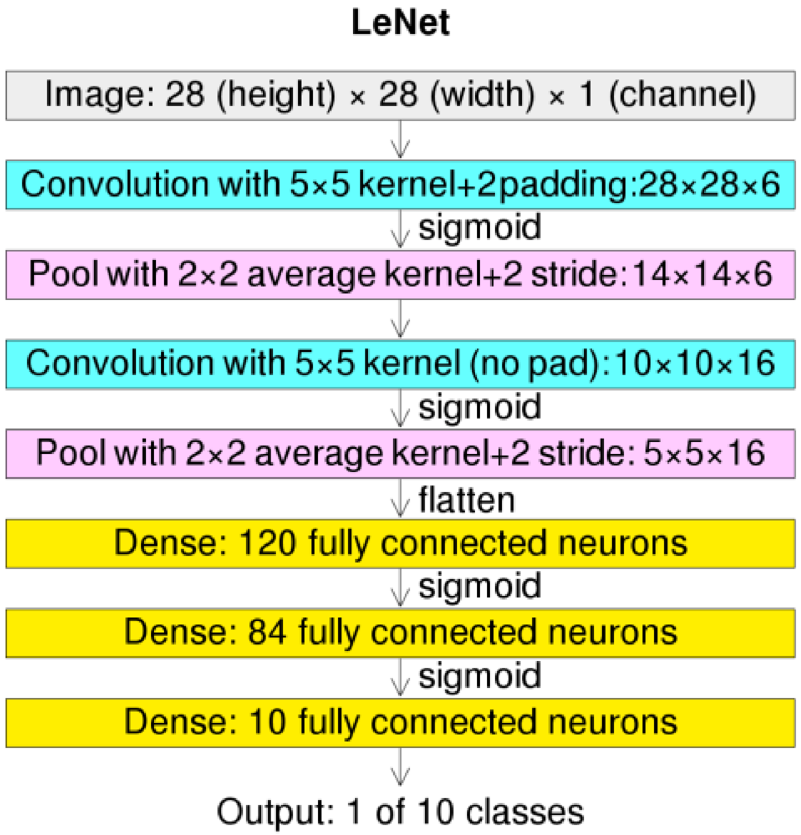


ImageNet

- 14 million images
- ~400 x 500 x 3 dimensions
- 10,000+ classes
- More commonly used: ILSVRC2012
- 1.2 million training images
- 50,000 test images
- 1000 classes

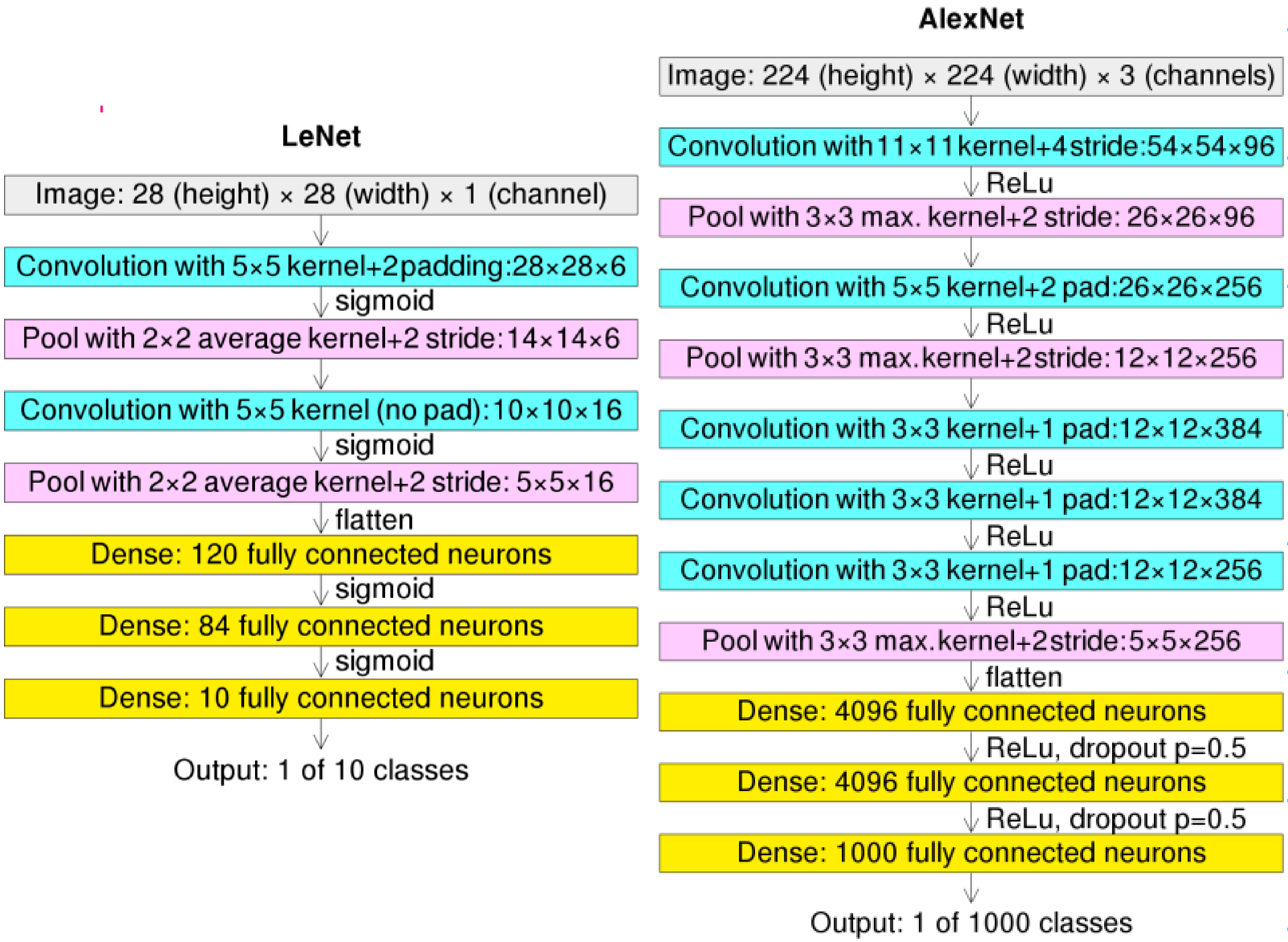
LeNet (1998)


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections


LeNet
Image: 28 (height) × 28 (width) × 1 (channel)
Convolution with 5×5 kernel+2padding:28×28×6
sigmoid
Pool with 2×2 average kernel+2 stride:14×14×6
Convolution with 5×5 kernel (no pad):10×10×16
sigmoid
Pool with 2×2 average kernel+2 stride: 5×5×16
flatten
Dense: 120 fully connected neurons
sigmoid
Dense: 84 fully connected neurons
sigmoid
Dense: 10 fully connected neurons
Output: 1 of 10 classes

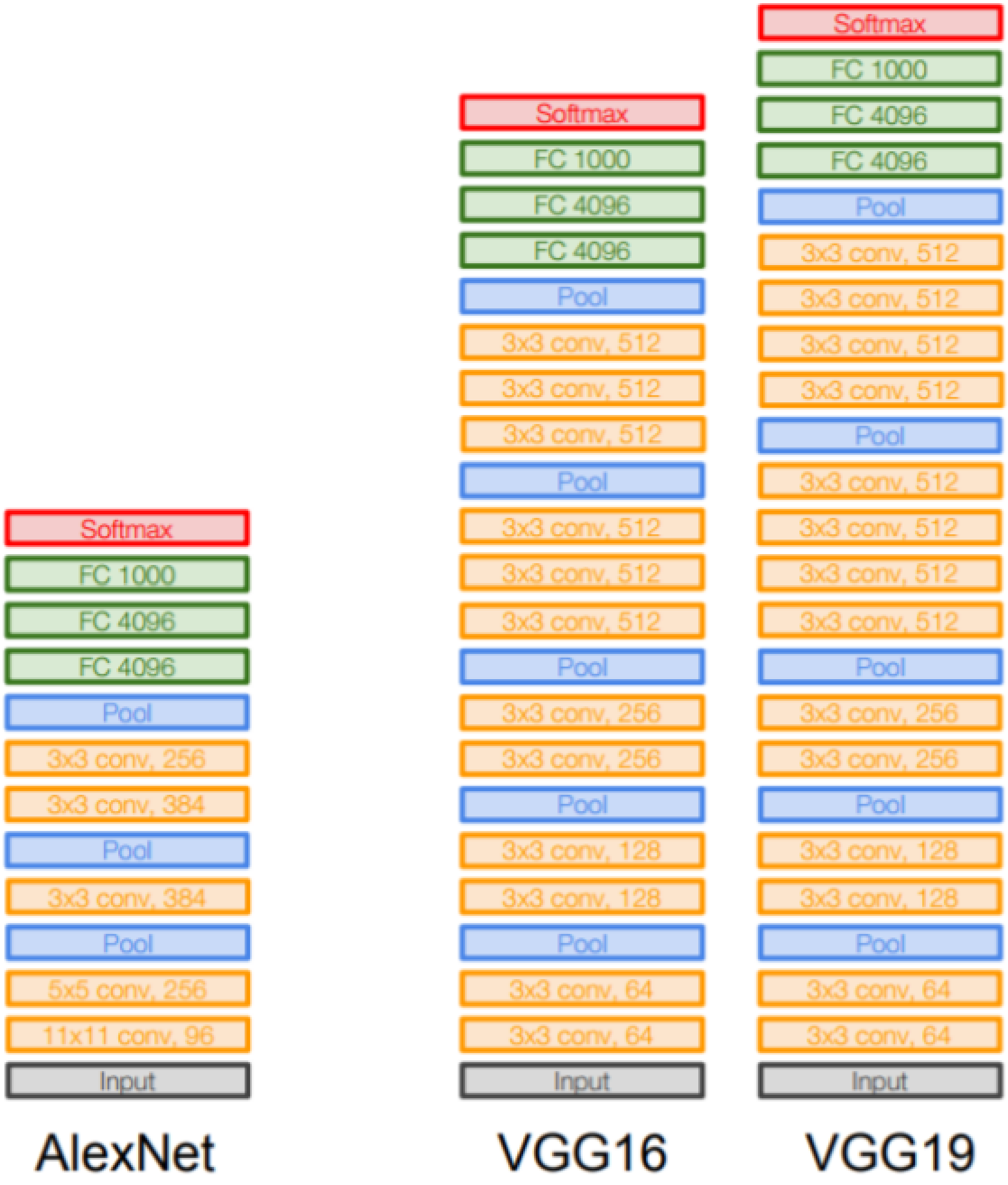
AlexNet (2012)

- Won ILSVRC2012
- Top 5 error of ~15%
- Best alternative 25%



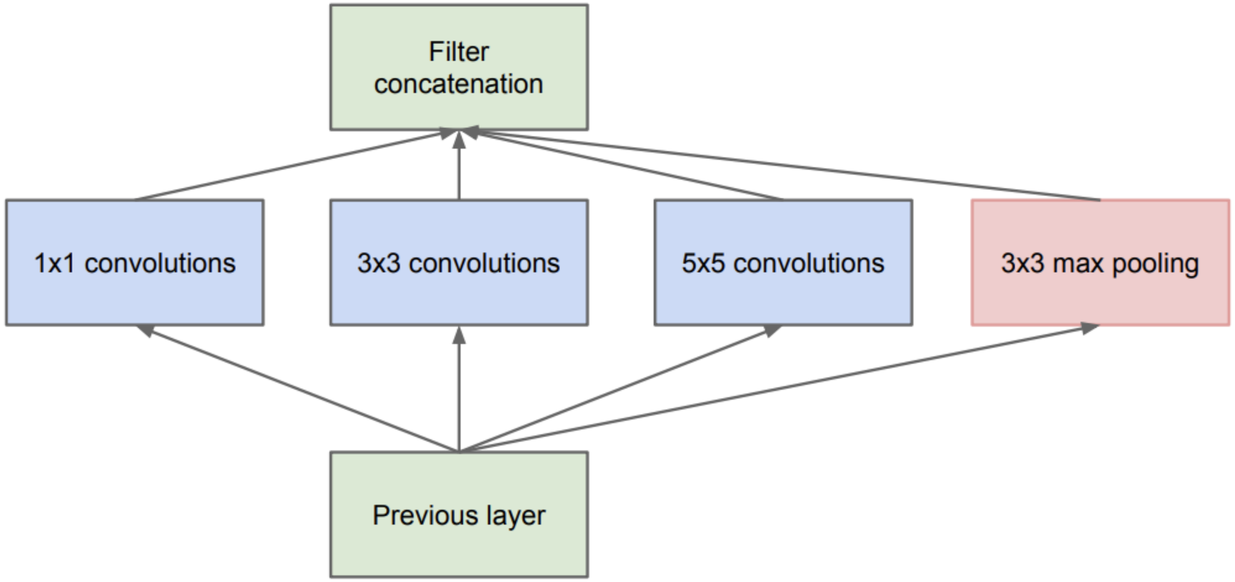
VGGNet (2014)

- Deeper than AlexNet
- Smaller convolutions
  - But more layers of them
- 2 3x3 and 1 5x5 can “see” same
- First has fewer parameters
- More nonlinearities though

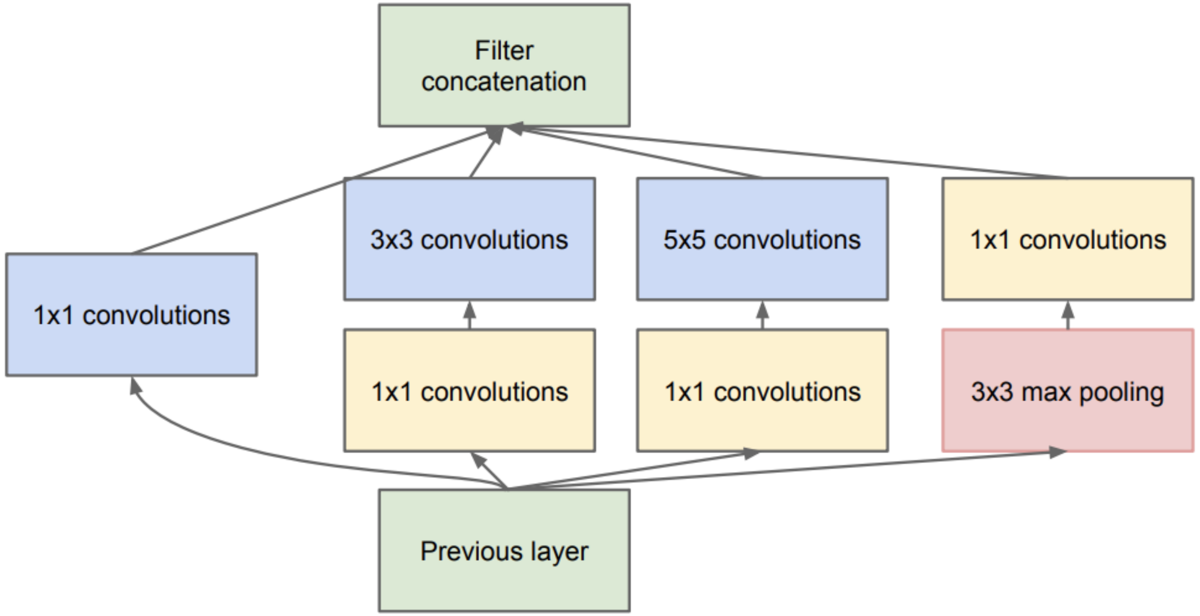


GoogLeNet (2014)

- Inception module
- Dimension reduction: K 1x1 filters takes $W \times H \times D$ to $W \times H \times K$

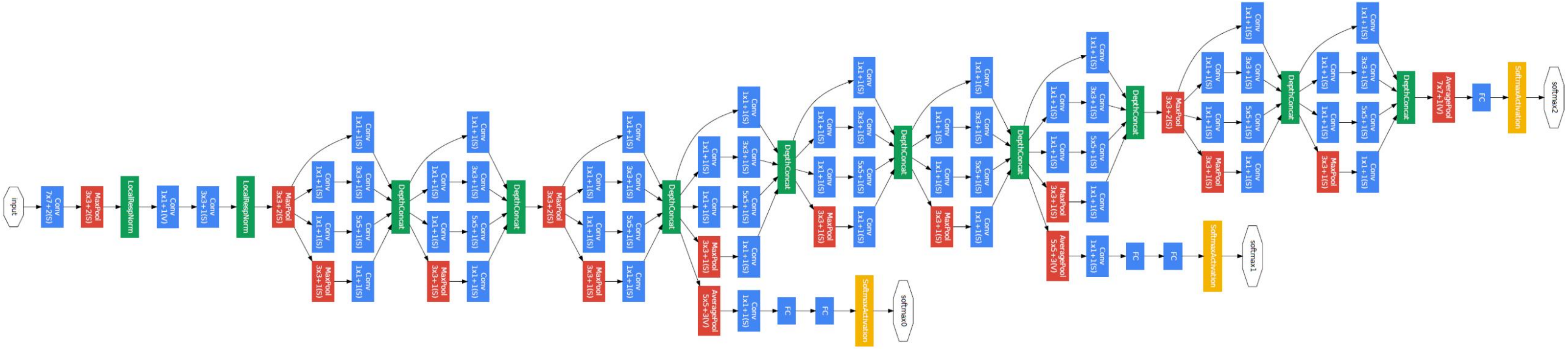


(a) Inception module, naïve version



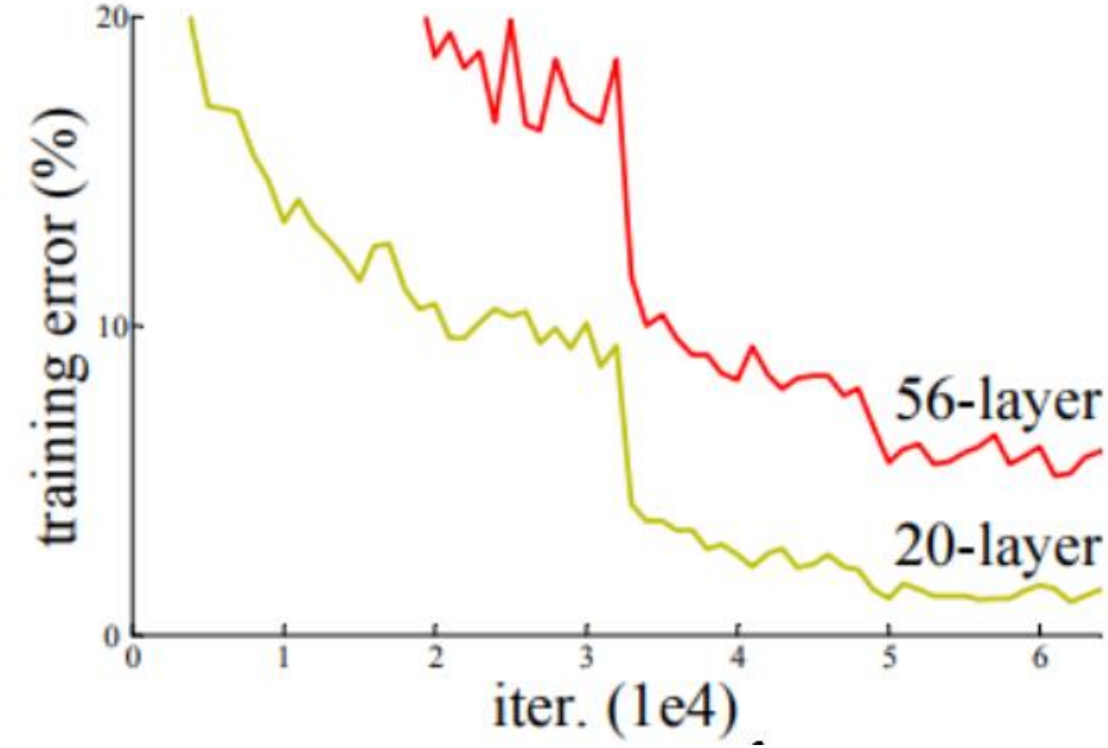
(b) Inception module with dimension reductions

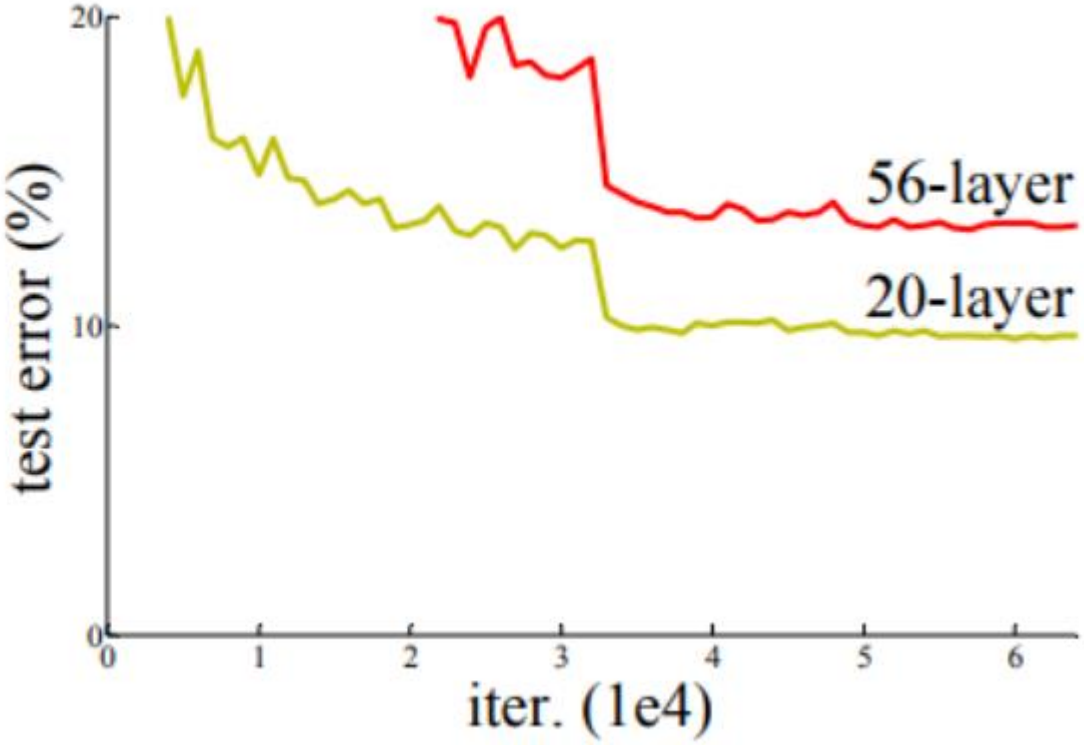
GoogLeNet (2014)



ResNet (2015)

- Deeper is not always better…





ResNet (2015)

- Challenges with bigger networks: optimization is harder
- Can we get the “signals” to where they need to be faster/directly?
- (Draw residual connection: 3x3, ReLU, 3x3, ReLU, residual connection from start to after second 3x3)