Deep neural architecture and applications (part II)
Week 12: Lectures 56-57

Jayanta Mukhopadhyay
Dept. of Computer Science and Engg.

Courtesy: K. Sairam

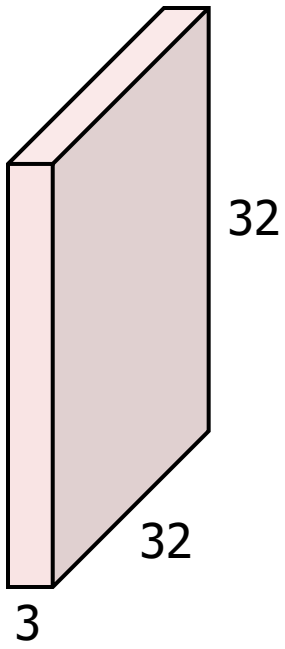
Convolutional Neural Network

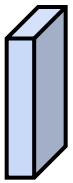
Convolution Layer

32x32x3 image

Filters always extend the full depth of the input volume

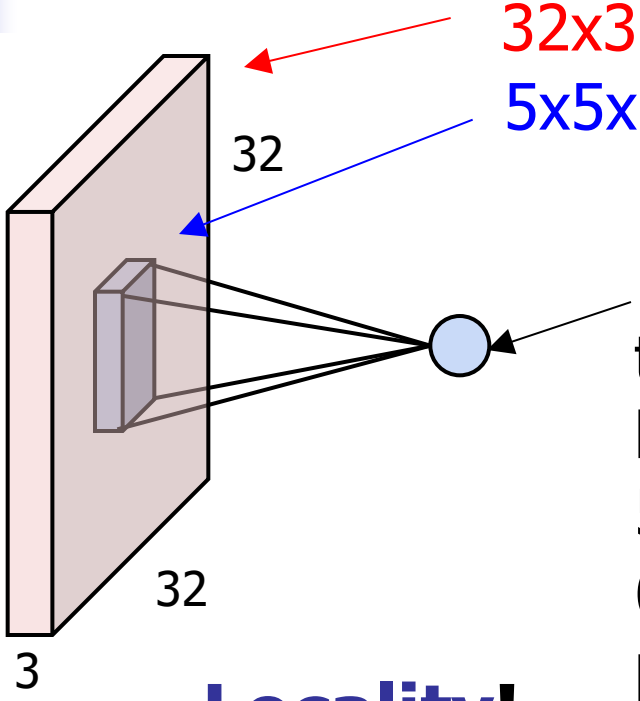
5x5x3 filter (kernel)





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

Convolution Layer



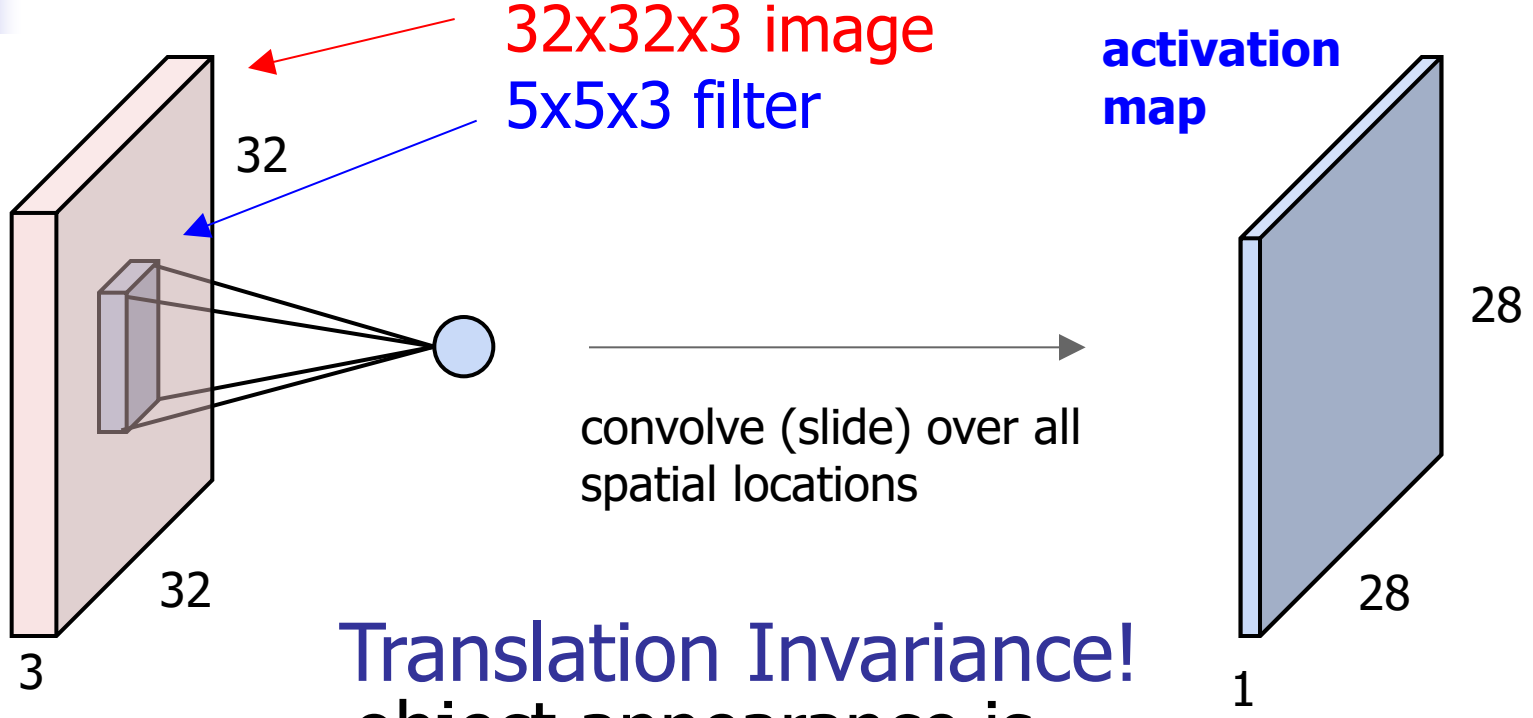
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Locality!

Objects tend to have a local spatial support.

Convolution Layer

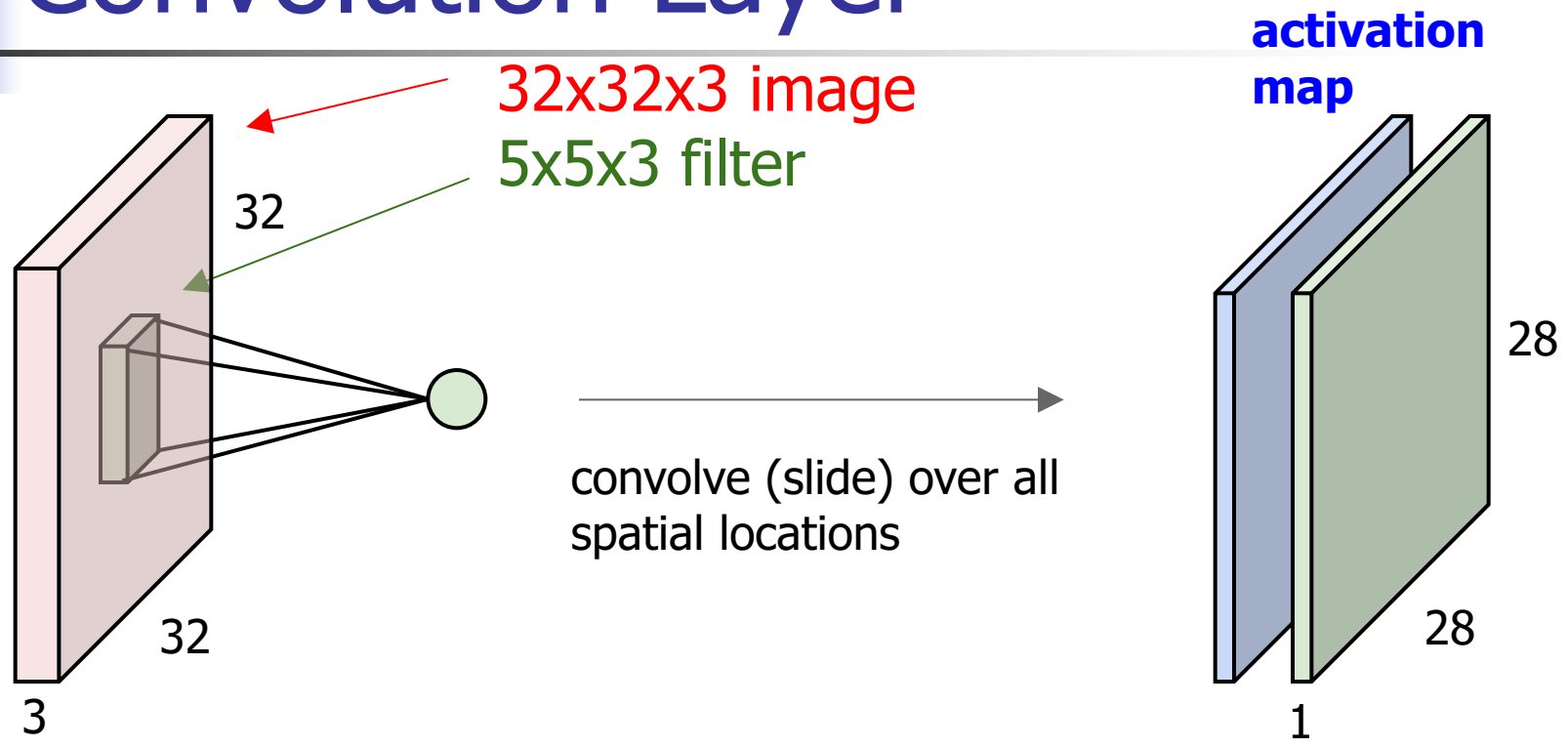


Translation Invariance!

object appearance is independent of location

Weight sharing!

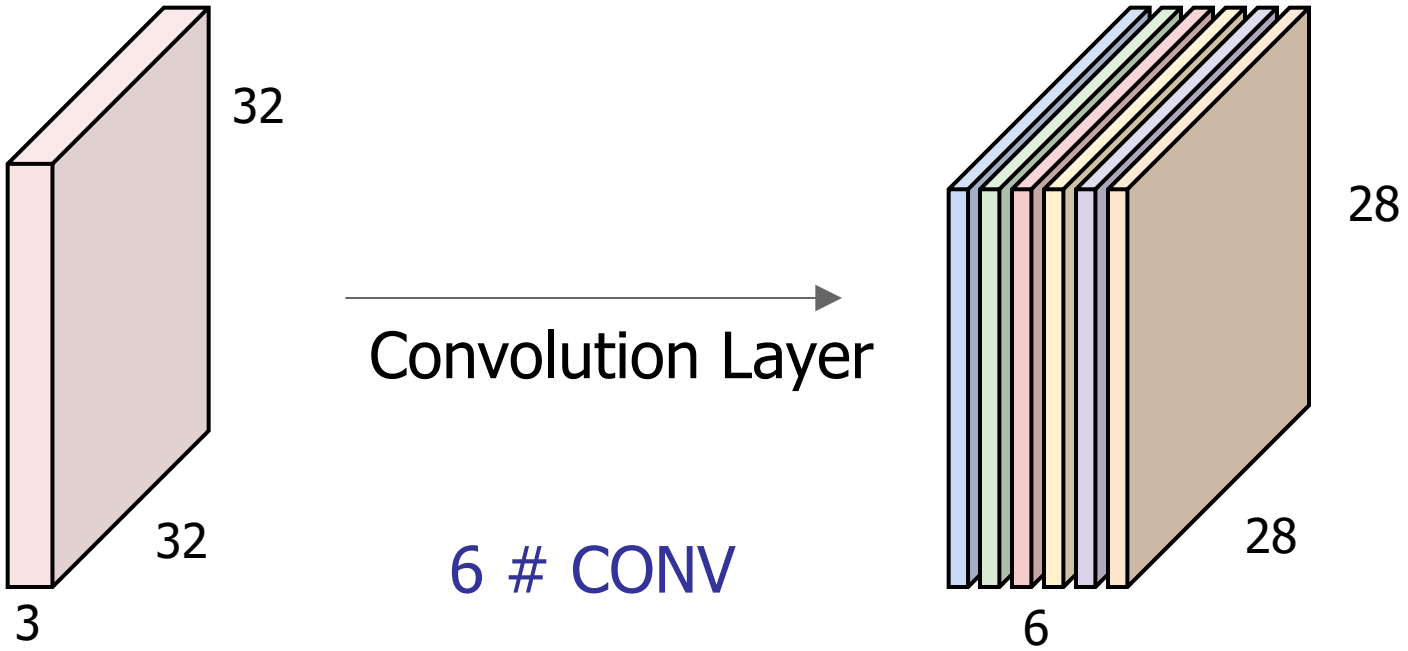
Convolution Layer



Consider a second, green filter.

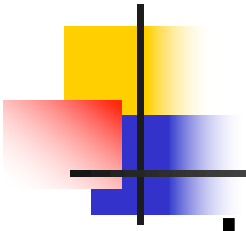
Convolution Layer (CONV)

activation maps



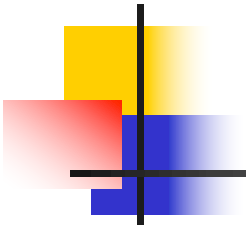
For example, if we had 6 5x5x3 filters, we'll get 6 separate activation maps:

We stack these up to get a "new image" of size 28x28x6!



Features of CONV

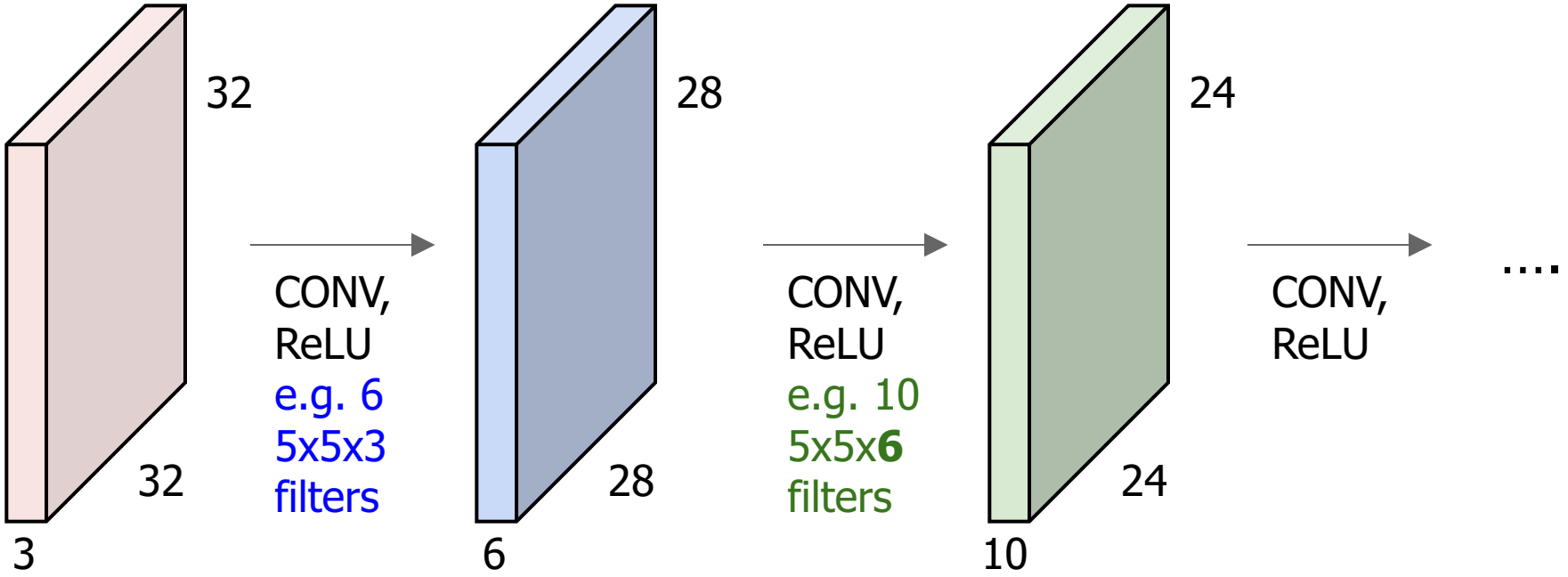
- **Locality**:
  - objects tend to have a local spatial support
- **Translation invariance**:
  - object appearance is independent of location
- **Weight sharing**
  - units connected to different locations have the same weights
  - equivalently, each unit is applied to all locations
  - weights of filters are invariant.
- Each unit output of filter is connected to a local rectangular area in the input.
  - – Receptive Field



Non-Linear Layer

- Increase the nonlinearity of the entire architecture without affecting the receptive fields of the convolution layer.
  - Commonly used in CNN is ReLU.

Convolutional Neural Networks (CNN)

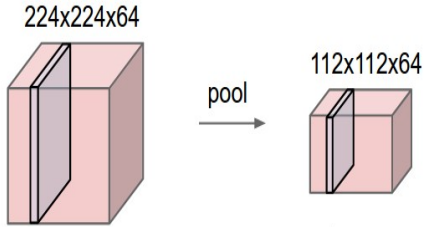


A CNN is a sequence of convolution layers and nonlinearities.

Parameters involved in convolution layer

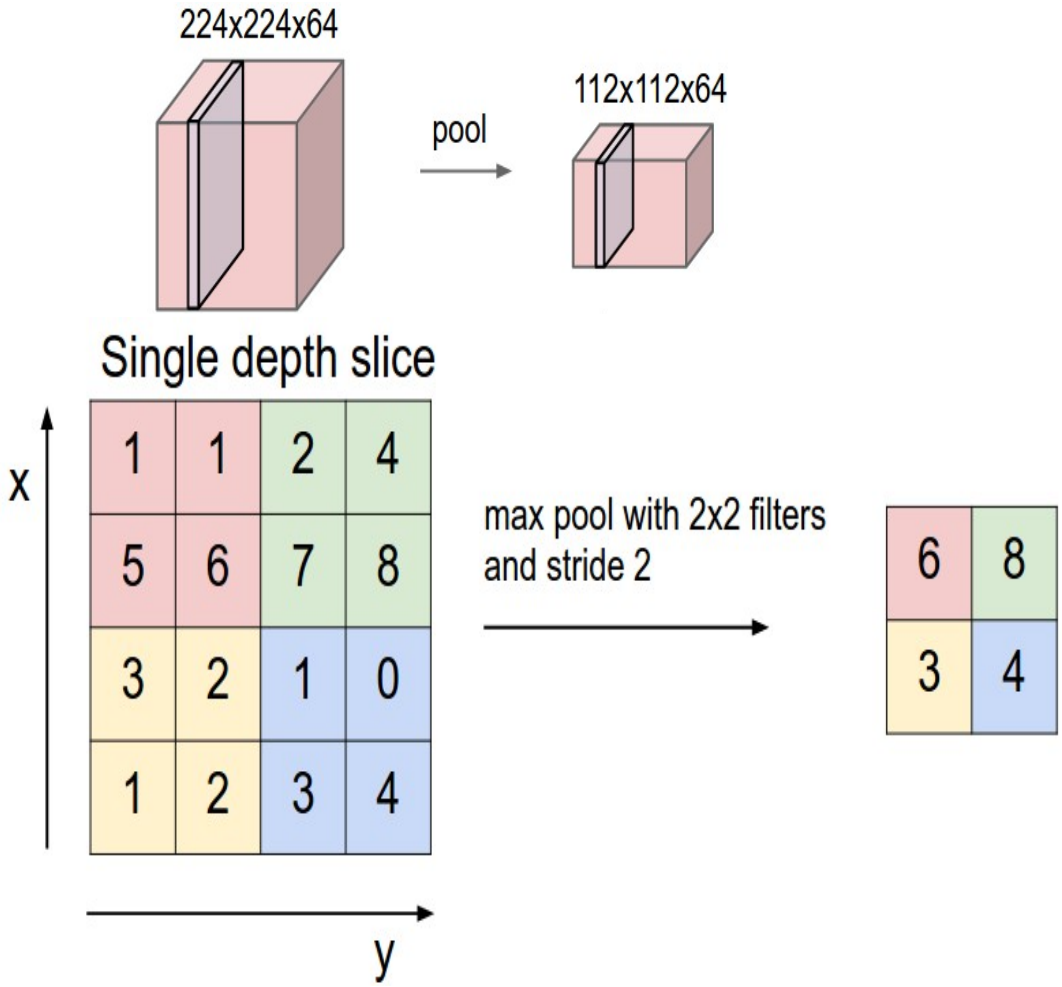
- Input Volume size W_1 x H_1 x D_1
- No. of filters K with size F_w x F_h x D_1 convolved with stride (S_w,S_h).
- Input zero padded by (P_w, P_h) on both sides.
- Output volume size W_2 x H_2 x D_2?
  - $W_2 = (W_1 - F_w + 2P_w)/S_w + 1$
  - $H_2 = (H_1 - F_h + 2P_h)/S_h + 1$
  - $D_2 = D_1$
- Parameters ?
  - $(F_w * F_h * D_1)$ * K weights + K biases
- **d-th depth slice of output is the result of convolution of d-th filter over the padded input volume with a stride, then offset by d-th bias**

Pooling Layer (POOL)



- To progressively reduce the spatial size of the representation.
  - to reduce the amount of parameters and computation in the network.
  - to control overfitting.
- Pooling partitions the input image into a set of non-overlapping rectangles.
- For each such sub-region, outputs an aggregated value of the features in that region.
  - Maximum value (Max pooling)
  - Average value (Average pooling)
- Operates over each activation map independently

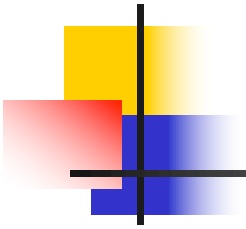
Pooling Layer (POOL)



Parameters involved in pooling

- Input Volume size W_1 x H_1 x D_1
- Pool size F_w x F_h with stride (S_w,S_h).
- Output volume size W_2 x H_2 x D_2?
  - $W_2 = (W_1 - F_w)/S + 1$
  - $H_2 = (H_1 - F_h)/S + 1$
  - $D_2 = D_1$
- Parameters ?
  - 0!
- **Uncommon to use zero-padding in Pooling layers.**

Fully Connected Layer (FC)



- Contains neurons that connect to the entire input volume
  - as in ordinary Neural Networks.
- Input volume to FC layer can also be treated as Deep Features.
- If the FC layer is a classifier, the input to FC can also be treated as feature vector representation for the sample.

Batch Normalization

- Normalizes input activation map to a layer by considering its distribution over a batch of training samples.
- To make Gaussian activation maps.
- Improves gradient flow through the network.
- Allows higher learning rates.
- Reduces the strong dependence on initialization.
- Acts as a form of regularization.
- Usually inserted after FC / CONV layers, and before non-linearity.

Batch Normalization (BN)

- Normalizes activation responses of a channel of previous layer
  - by subtracting mean of a responses of batch and dividing it by their standard deviation.
- Transforms the resultant output operation by scaling and translation by parameters a and b.
  - learnt by the gradient descent algorithm.
- During test time running averages and s.d.'s of activation maps used along with learnt parameter a and b for each channel at a layer.

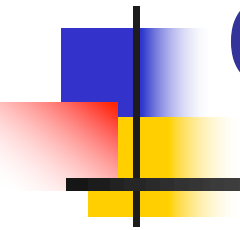
Drop out

- Randomly dropping out nodes of network (at hidden / visible layers) during training.
  - Temporarily removing it from the network, along with all its incoming and outgoing connections.
  - To regulate overfitting, more effective for smaller dataset.
  - Simulates learning sparse representation in hidden layers.
- Implementation
  - Retain output of a node with a probability p.
    - Typically within [0.5,1] at hidden layers and [0.8,1] in visible layers.

Learning weights with drop out

- Weights become larger due to drop out.
  - Needs to be scaled at the end training.
  - A simple heuristic.
    - Outgoing weights of a unit retained with probability p during training, multiplied by p at test time.
  - Scaling may be carried out during training time at each weight update.
    - No need to rescale weight for the test network.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting ,JMLR, 15(Jun):1929−1958, 2014.



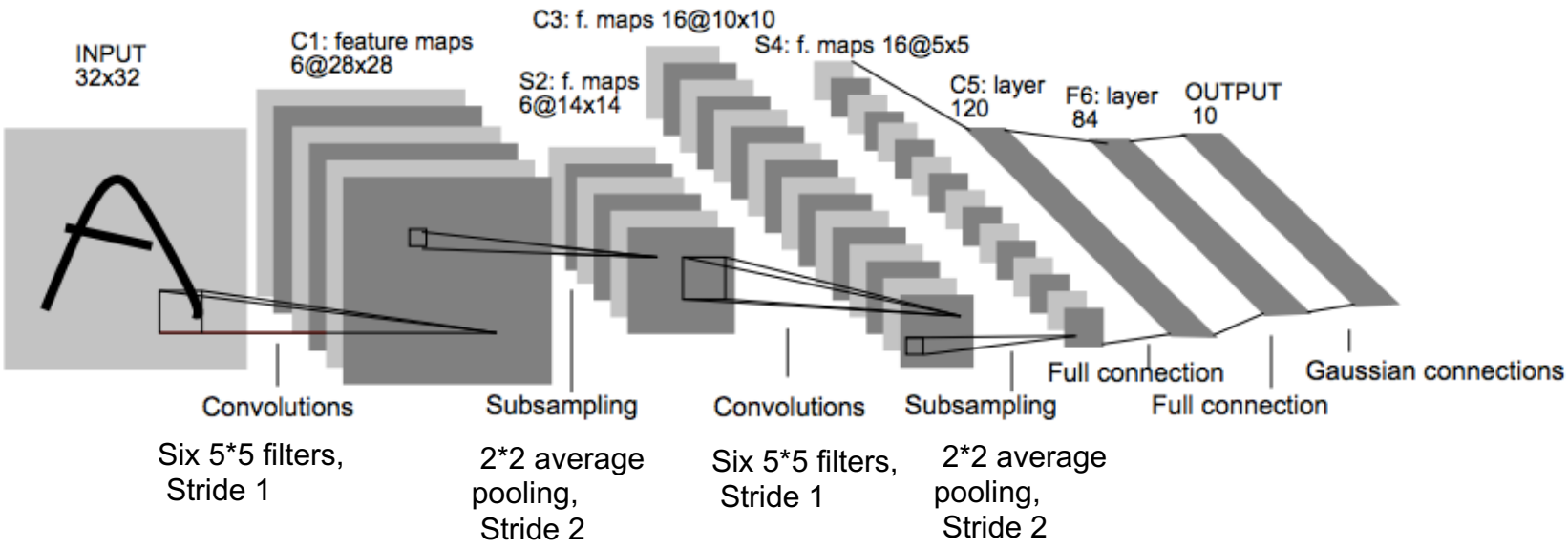
CNN Architectures

LeNet

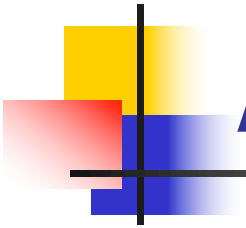
- Gradient-based learning applied to document recognition.
- Architecture: Input→CONV→POOL→CONV→POOL→FC→FC →Output
- Number of parameters: 60k
- Number of floating point operations per inference: 341k
- Sigmoid used for non-linearity.

*Y. Lecun et al, Proceedings of the IEEE, 1998

LeNet



*Y. Lecun et al, Proceedings of the IEEE, 1998

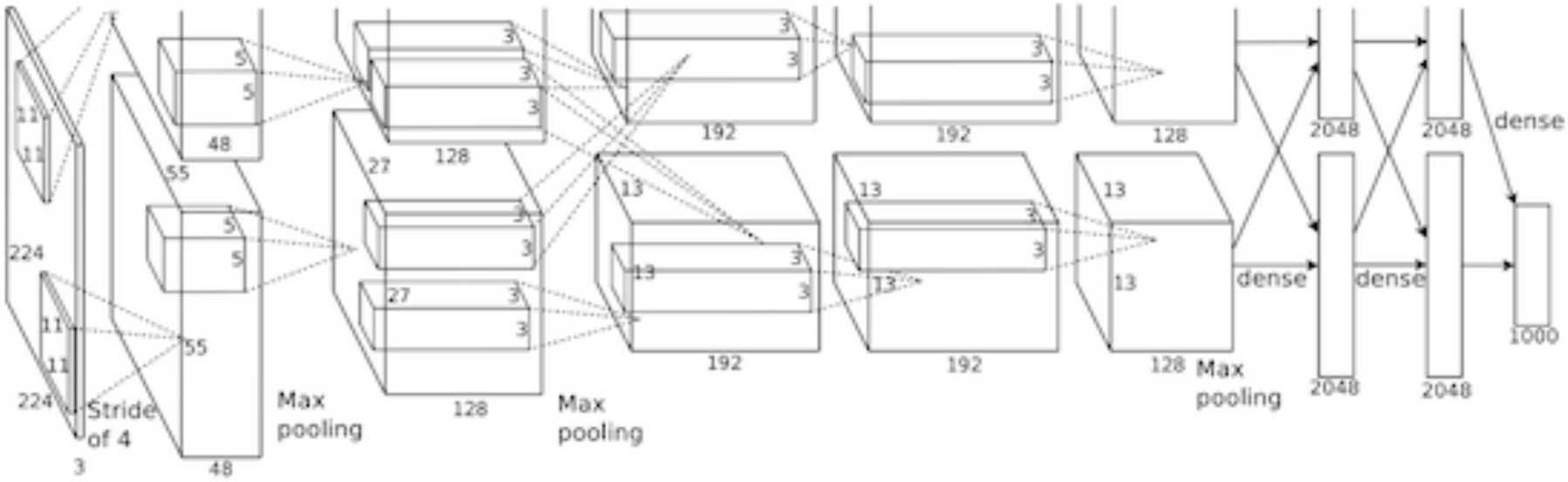


AlexNet

- Uses Local Response Normalization (LRN)
  Architecture:
  Input→CONV1→MAXPOOL1→NORM1→CONV2→MAXPOOL2→NORM2
  →CONV3→CONV4→CONV5→MAXPOOL3
  →FC6→FC7→FC8→Output
- # of Weights: 61M
- # of floating point oeprations: 724M
- ReLU used for non-linearity

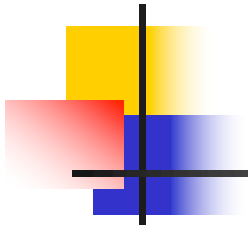
*Krizhevsky et al., ImageNet Classification with Deep Convolutional Neural Networks NIPS, 2012

AlexNet



*Krizhevsky et al., ImageNet Classification with Deep Convolutional Neural Networks NIPS, 2012

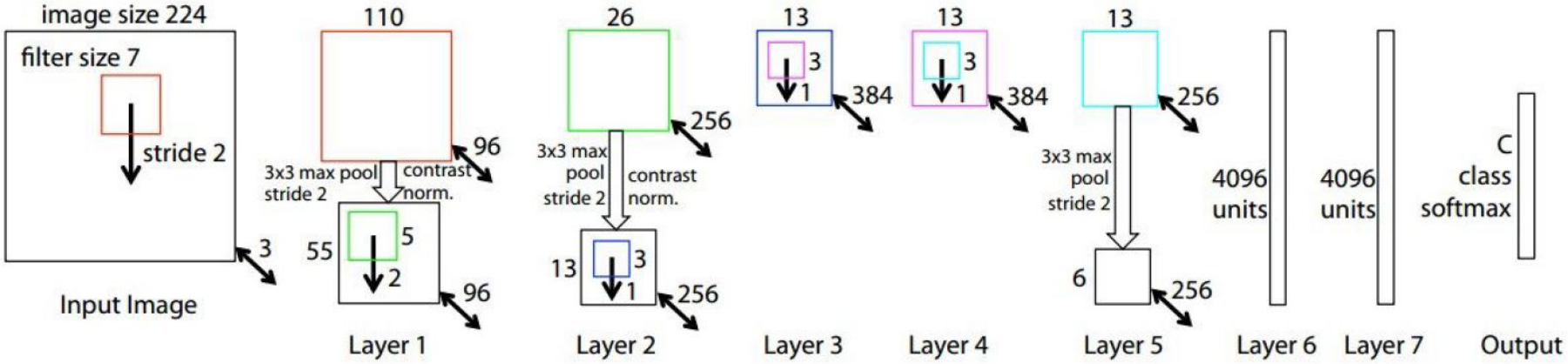
Parameters Count: AlexNet



- Input: 227x227x3 images
- First layer (CONV1): 96 11x11 filters applied at stride 4
- Q: what is the output volume size?
  - Along length and breadth: (227-11)/4+1 = 55
  - Output volume [55x55x96]
- Q: What is the total number of parameters in this layer?
  - Parameters: (11*11*3)*96 = 35K (Without bias)
  - Parameters: (11*11*3)*96 + 96 (With bias)
- Second layer (POOL1): 3x3 filters applied at stride 2
- Q: what is the output volume size?
  - Along length and breadth: (55-3)/2+1 = 27
  - Output volume: 27x27x96 (Input to POOL1 is output of CONV1)
- Q: what is the number of parameters in this layer?
  - Parameters: 0

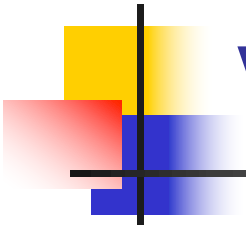
ZFNet

- AlexNet but:
- CONV1: changed from (11x11 stride 4) to (7x7 stride 2)
- CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512
- ImageNet top 5 error: 16.4% → 11.7%



Smaller filter size, More filters in layer

* Zeiler and Fergus, 2013



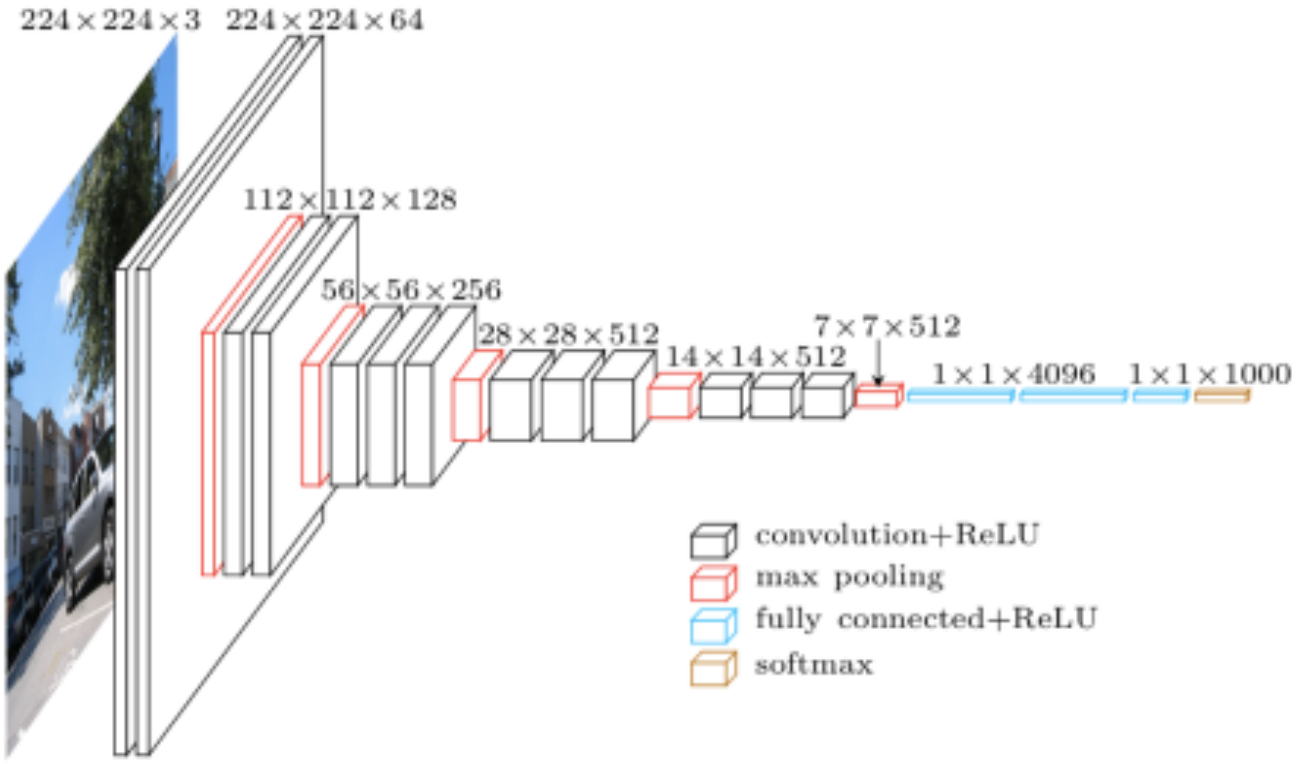
VGG

- Smaller filters, deeper layers
- 8 layers (AlexNet) → 13 layers (VGG13) / 16 layers (VGG16Net) / 19 layers (VGG19Net)
- Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2
- 7.3% top 5 error in ILSVRC'14
- Weights: 138M & FLOPS: 15.5G

Deeper the layer, better accuracy.

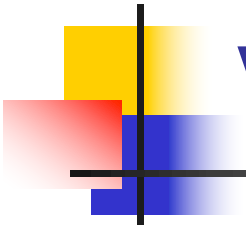
*Simonyan et al., arXiv 2014, ICLR 2015

VGG



Deeper the layer, better accuracy.

*Simonyan et al., arXiv 2014, ICLR 2015



VGG

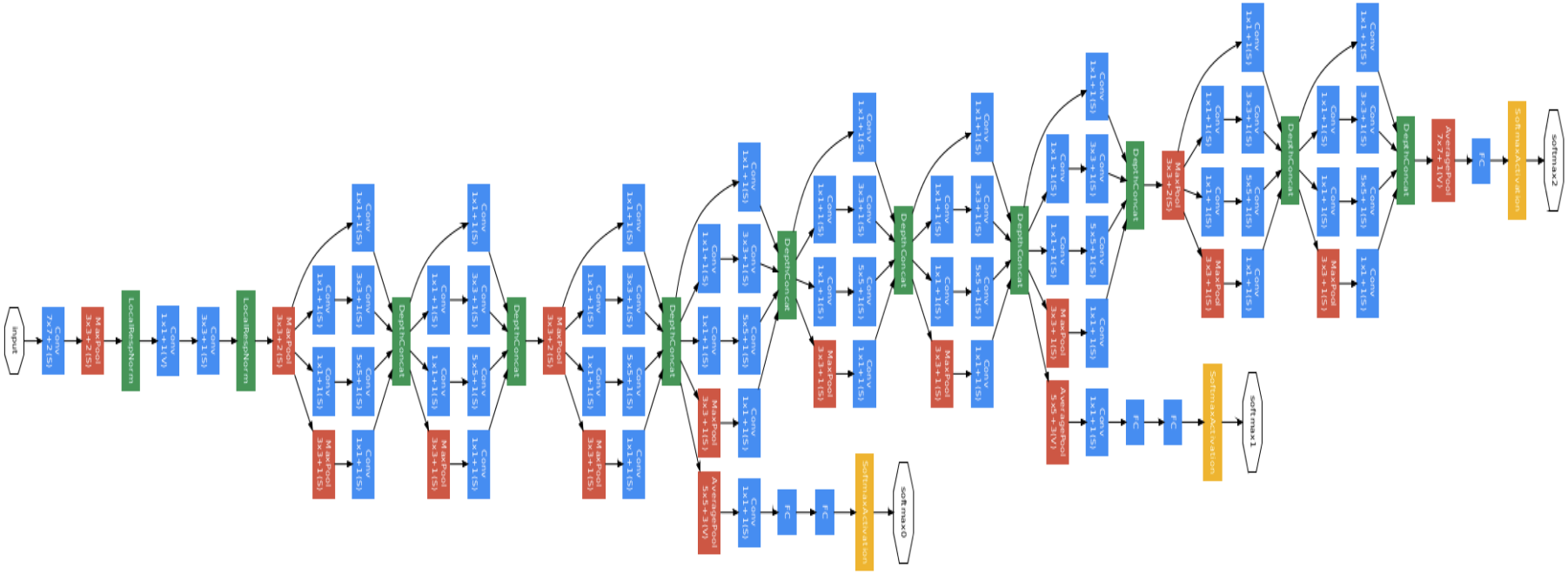
- Stack of three 3x3 conv (stride 1) layers has **same effective field** as one 7x7 conv layer
  - deeper with more non-linearities
  - Fewer parameters: How?
    - $3*(3^2\ C^2)$ vs. $(7^2\ C^2)$ for C channels per layer

GoogleNet

- CONV Layers: 21 (depth), 57 (total)
- Introduces inception modules.
  - Concatenates output of filters of different sizes.
- Fully Connected Layers: 1
- Weights: 7.0M & FLOPS: 1.43G
- Architecture: (9 Inception Modules)
  INPUT→CONV1→POOL1→CONV2→CONV3→POOL2→INCEPTION1
  →INCEPTION2→POOL3→INCEPTION3→INCEPTION4→INCEPTION5
  →INCEPTION6→INCEPTION7→POOL4→INCEPTION8→INCEPTION9
  →POOL5→FC1→OUTPUT
- ILSVRC'14 classification winner (6.7% top 5 error)

*Szegedy et al., arXiv 2014, CVPR 2015

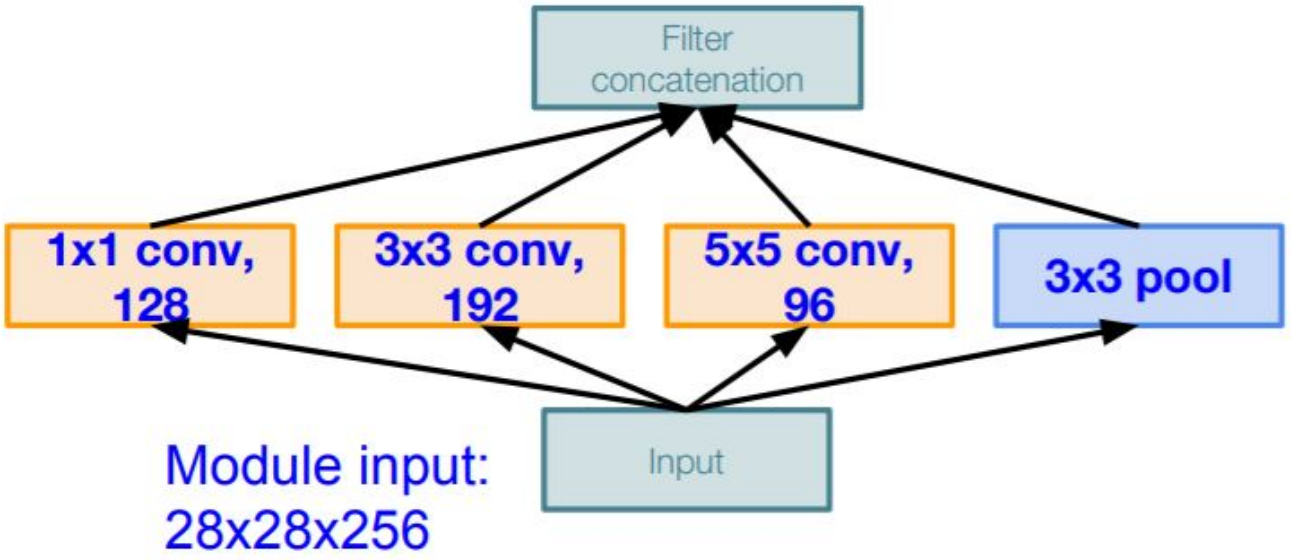
GoogleNet



*Szegedy et al., arXiv 2014, CVPR 2015

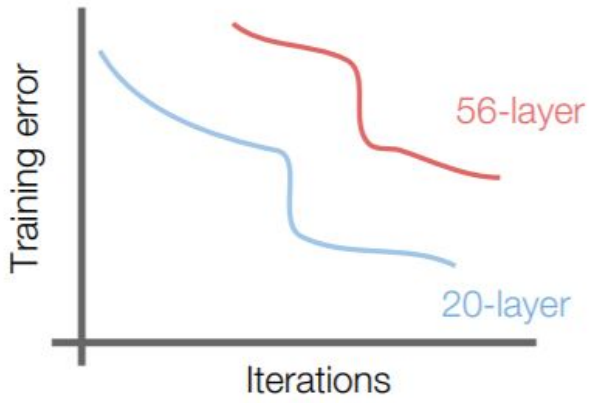
Naïve Inception Module

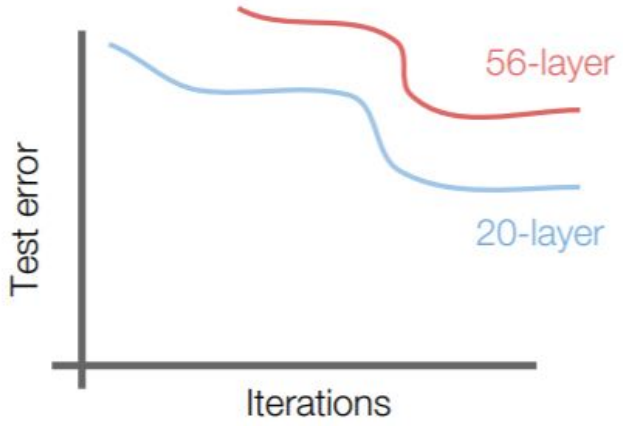
28x28x(128+192+96+256) = 28x28x672



ResNet

- Problems with deeper model
  - causes overfitting
  - harder to optimize, because of vanishing gradients.
    - gradients die as we go deeper.

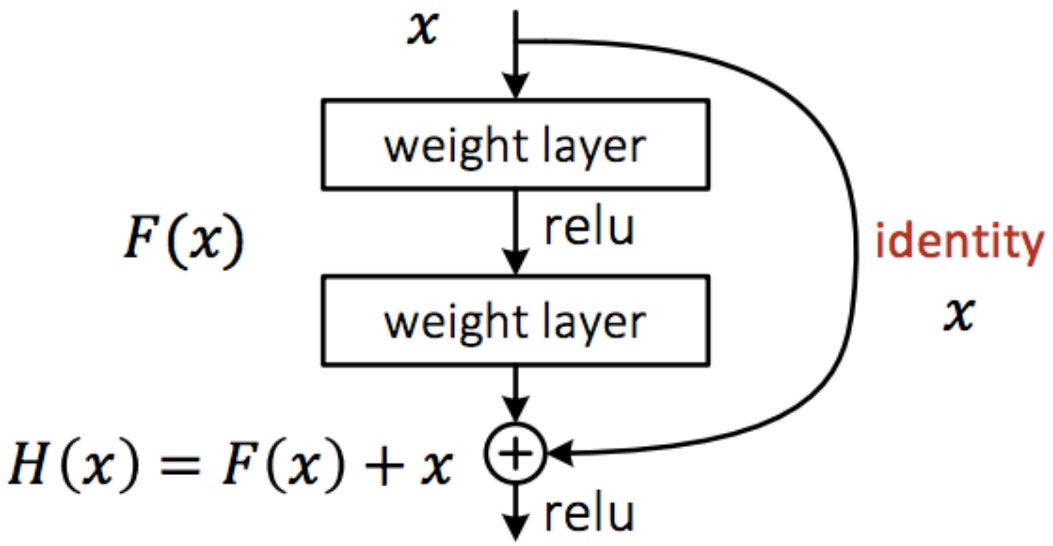




*He et al., ResNet arxiv 2015

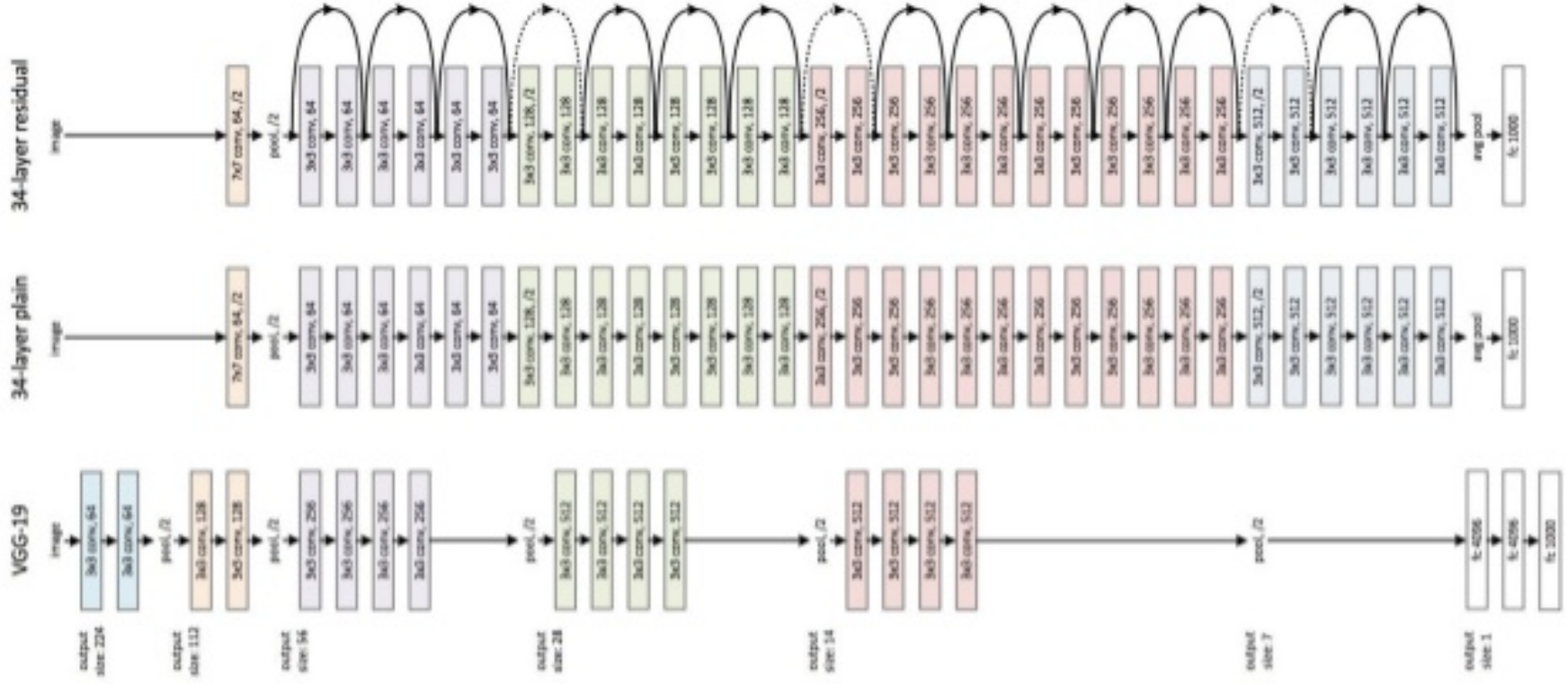
ResNet

- Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping.

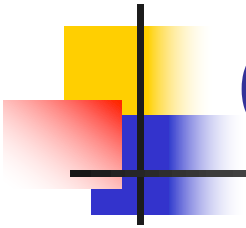


ResNet

Total depths of 34, 50, 101, or 152 layers architectures are also available

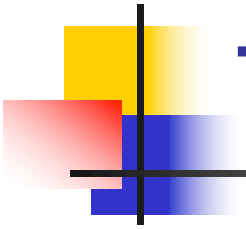


*https://www.slideshare.net/SomnathBanerjee17/lenet-to-resnet



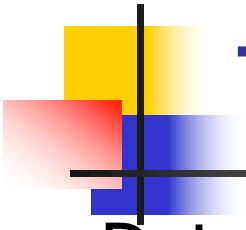
Other Networks

- Network in Network (NIN)
- Wide Residual Networks
- Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)
- DenseNets
- SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size
- MobileNet (Depthwise Separable Convolutions)
- ShuffleNet (Grouped Convolutions)
- FractalNet: Ultra-Deep Neural Networks without Residuals



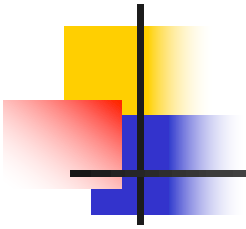
Training steps:

- Preprocessing of training dataset.
  - Normalize data.
  - Decorrelate data (Diagonal Covariance Matrix).
  - Whitening of data (Identity Covariance Matrix).
  - Subtract Mean.



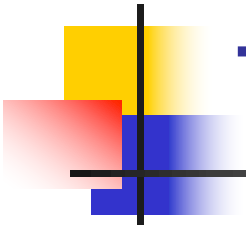
Training steps:

- Data augmentation.
  - Horizontal Flips
  - Random Crops on scaled input
  - Color jitter
  - Distortions
  - Transformations
- Weight initialization
- Train the network by update of the weight parameters.



Few Training Tips

- Start with small regularization and find learning rate that makes the loss go down.
- Can overfit very small portion of the training data.
- Train first few epochs with few samples to initiate the hyper-parameters.
- If big gap between training accuracy and validation accuracy, then it is overfitting.
  - Try increase regularization.
- If no gap, then may increase model capacity.



Transfer Learning

- No need of a lot of a data to train a CNN.
- Pre-trained models can be initialized for CNNs at the early stage of training.

*Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

Transfer Learning