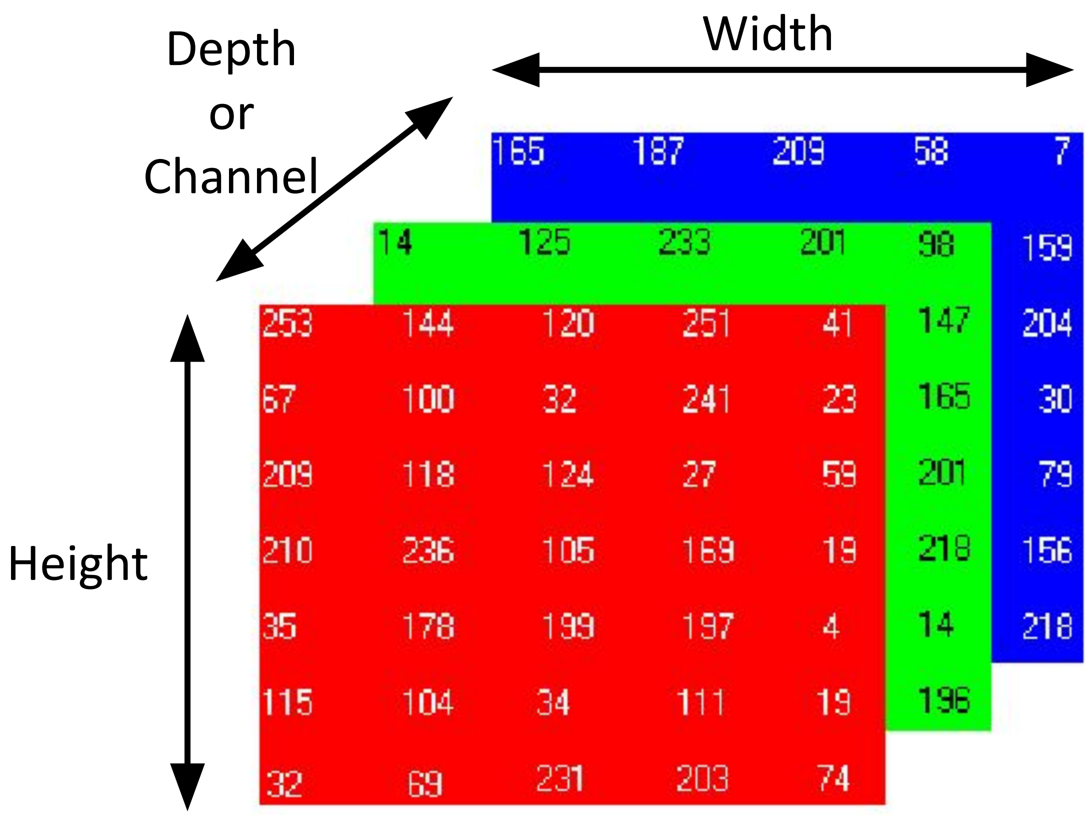
Convolutional Neural Networks (2)

Geena Kim

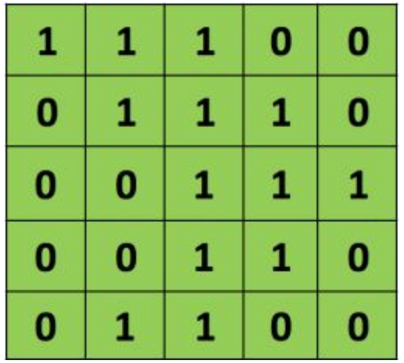
University of Colorado **Boulder**

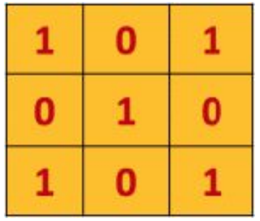
Dealing with images

An image is a multi-dimension array



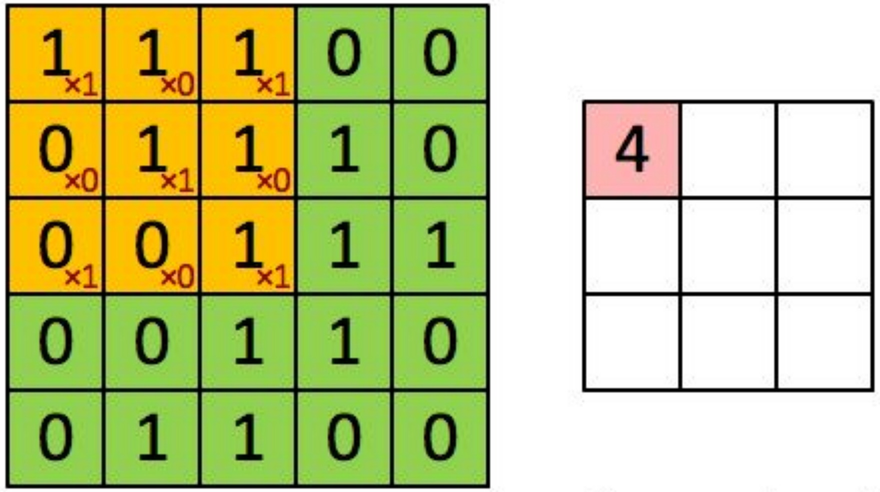
What is Convolution





filter

Image



Image

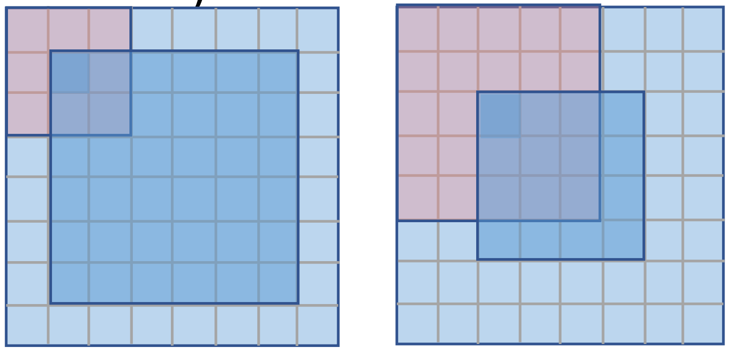
Convolved Feature

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2]$$

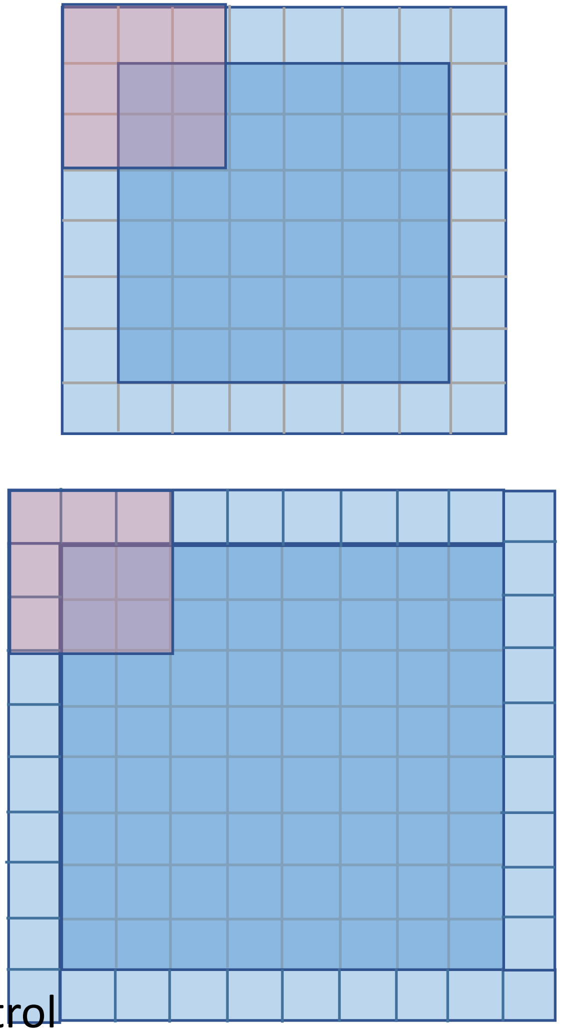
$$= (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9)$$

Convolutional Layer- hyper parameters

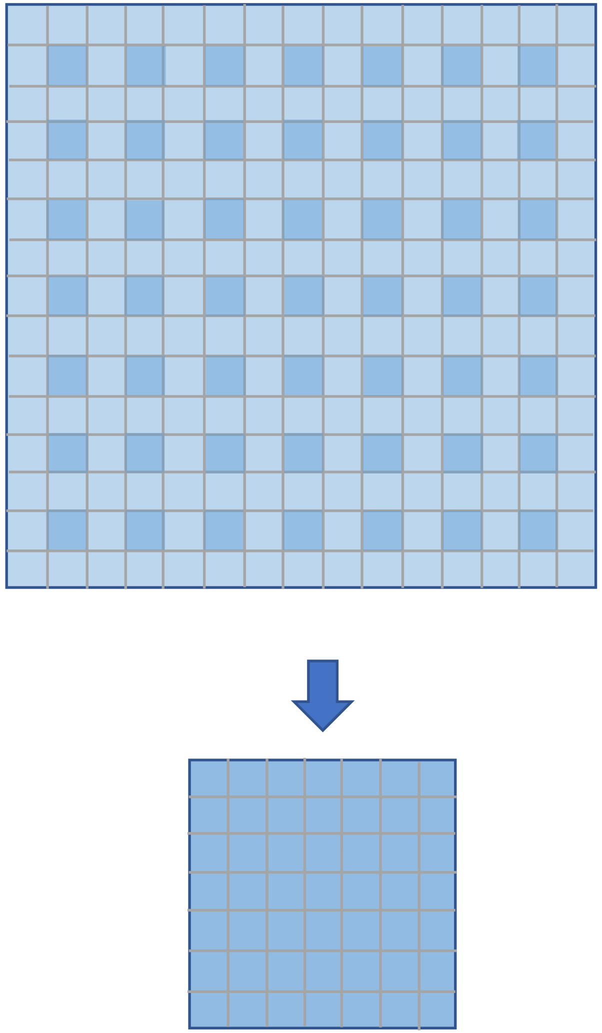
- Filter size (3x3, 5x5, 1x1….)



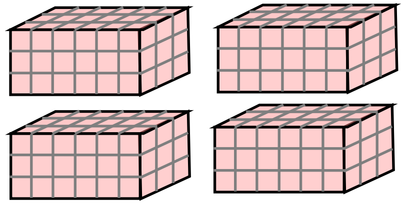
- Padding



- Stride



- Number of filters



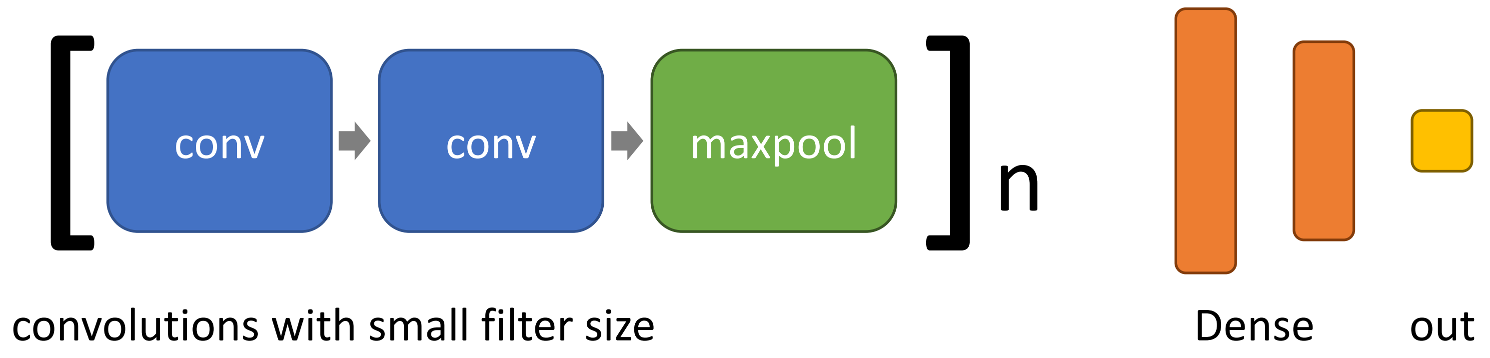
** Parameters and Hyper parameters are different!
Parameters = Weights to be optimized.
Hyperparameters = design parameters you can control

Why CNN?

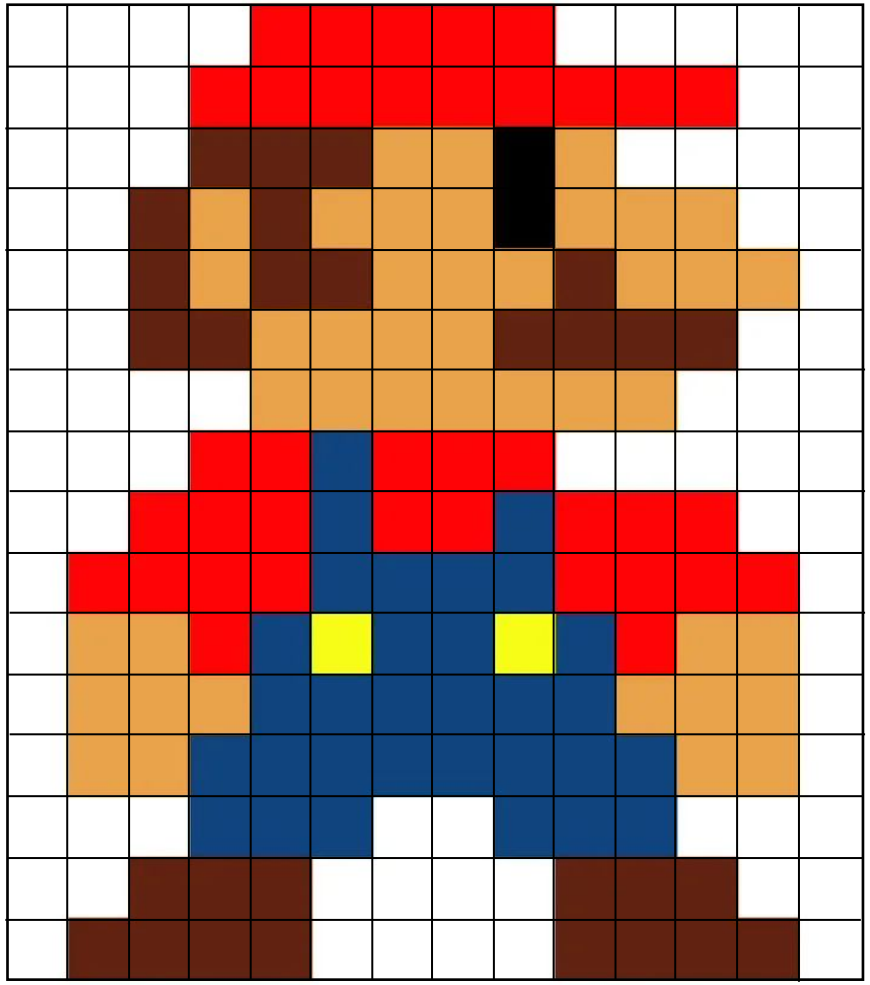
- Images have big pixels!
- Fully-connected neural network would have too many parameters!
- Translational invariance in images

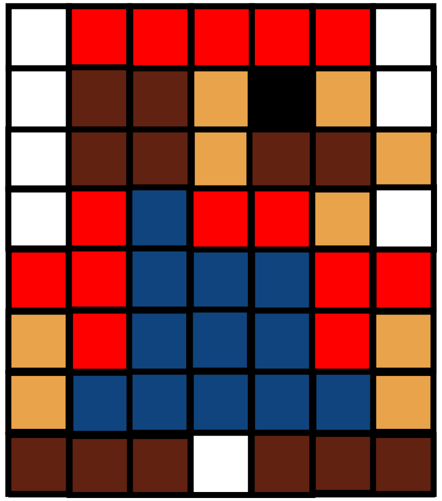
Deep convolutional neural network Architectures

Typical CNN architecture

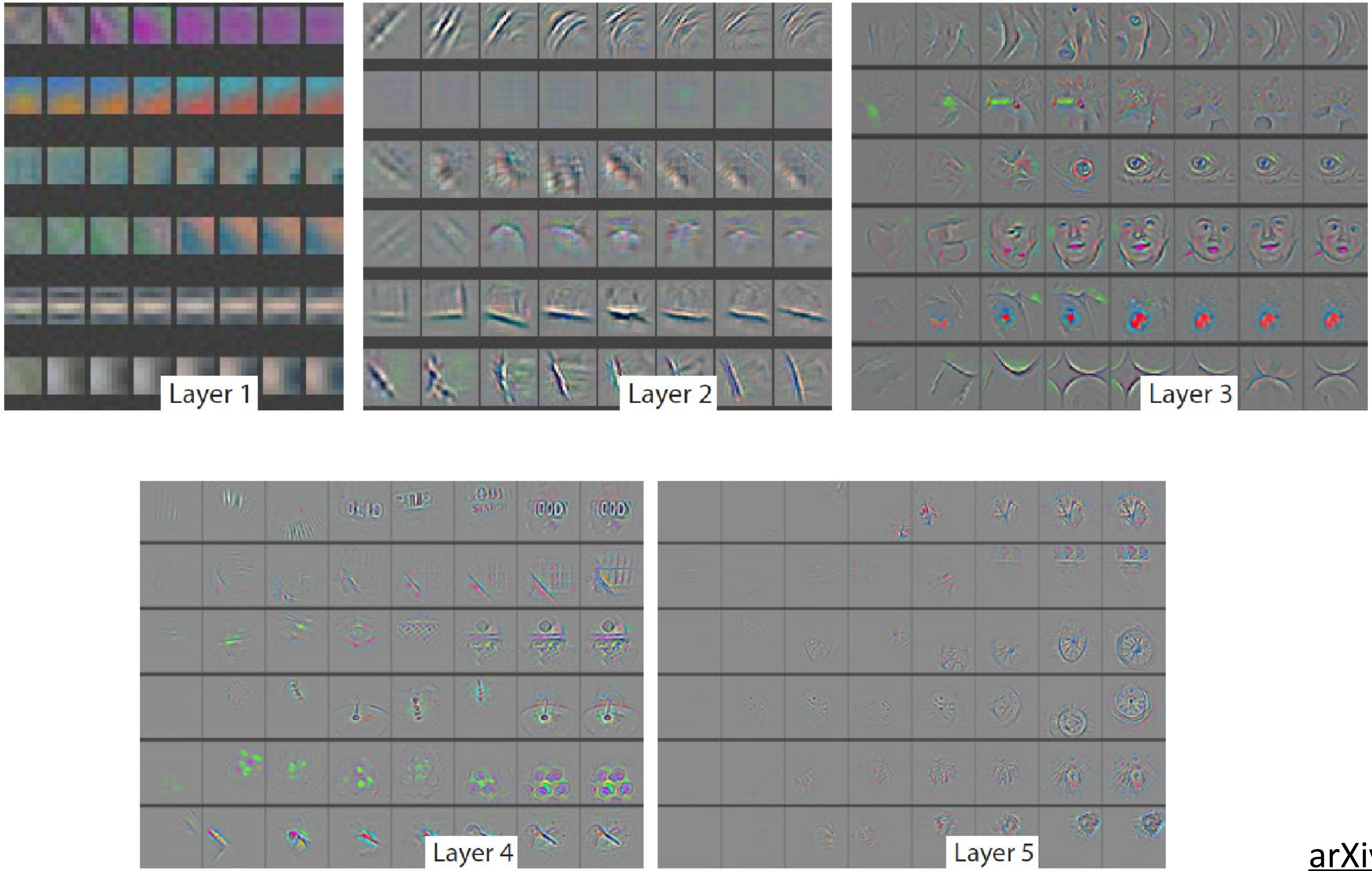


Max pooling operation



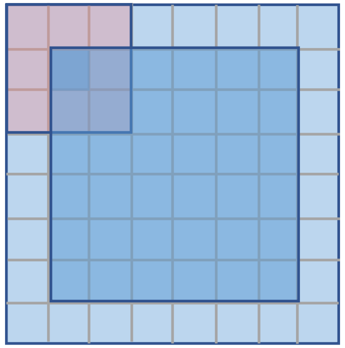


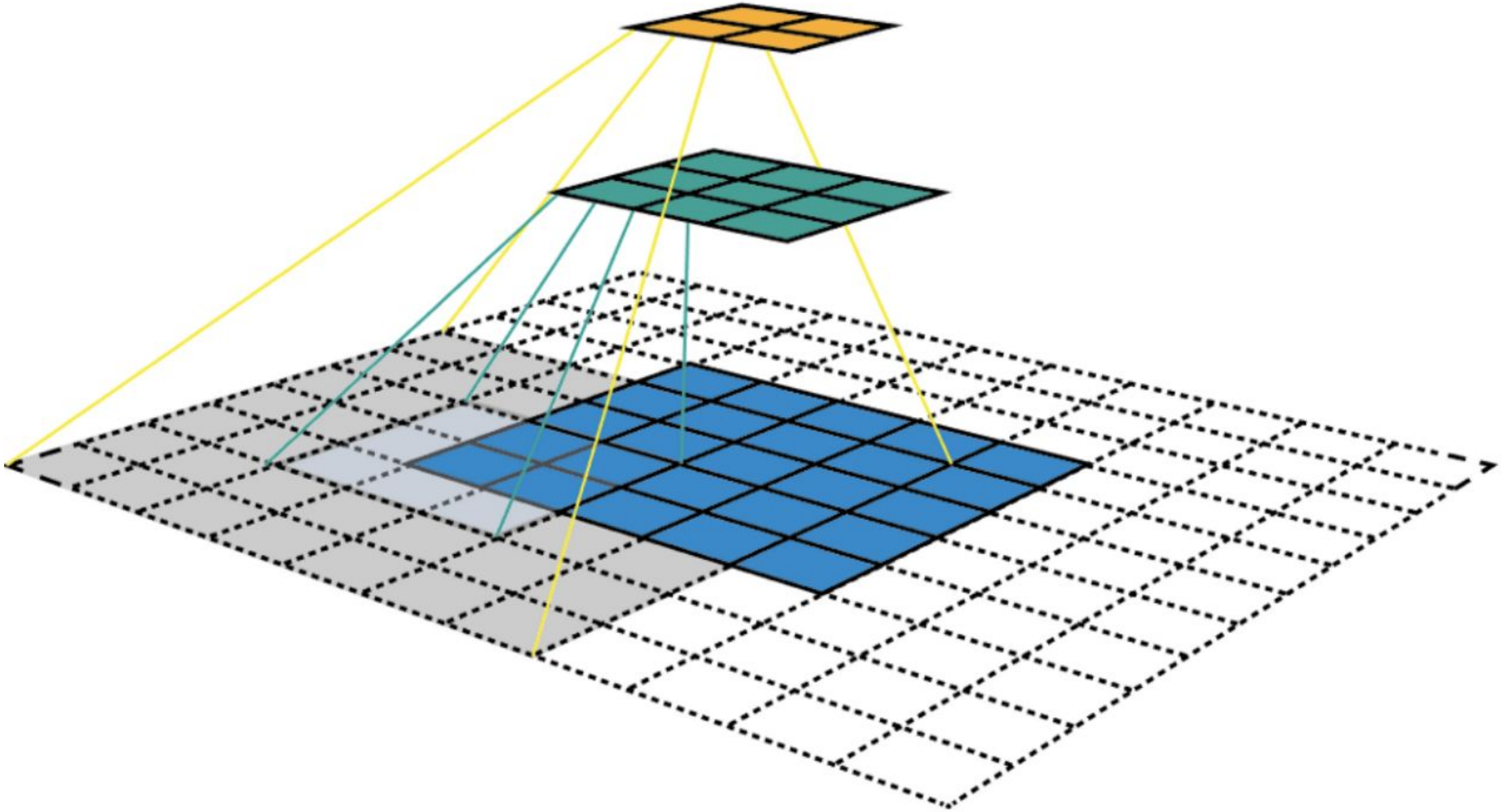
What does the deep convolutional layers do?



arXiv:1311.2901v3

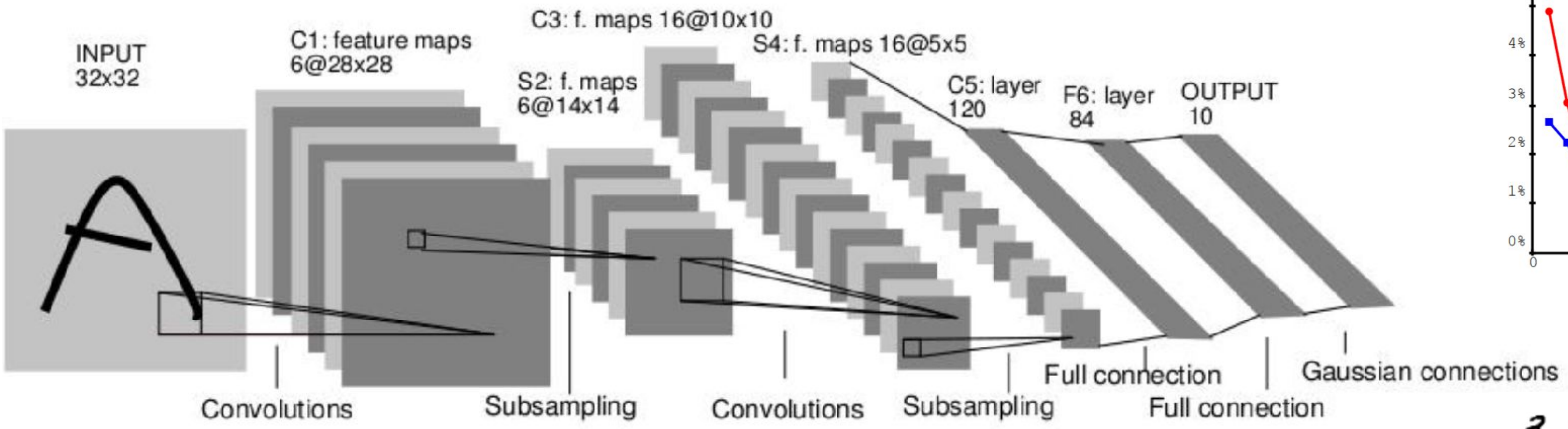
What do the multiple convolution layers do?

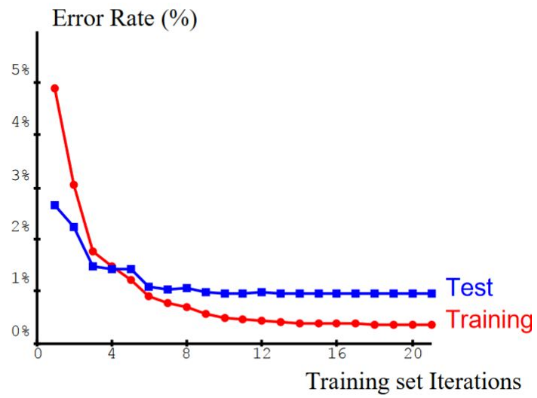




Modern CNN

LeNet 5 (1998)





MNIST dataset

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Natural Images Dataset

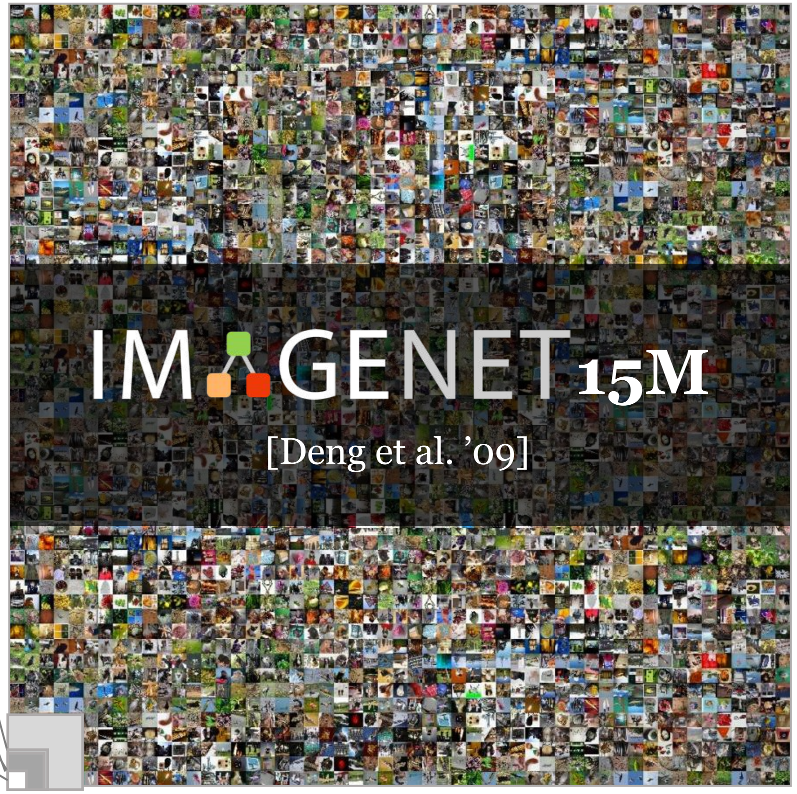
IMAGENET

SUN, 131K
[Xiao et al. '10]

LabelMe, 37K
[Russell et al. '07]

PASCAL VOC, 30K
[Everingham et al. '06-'12]

Caltech101, 9K
[Fei-Fei, Fergus, Perona, '03]



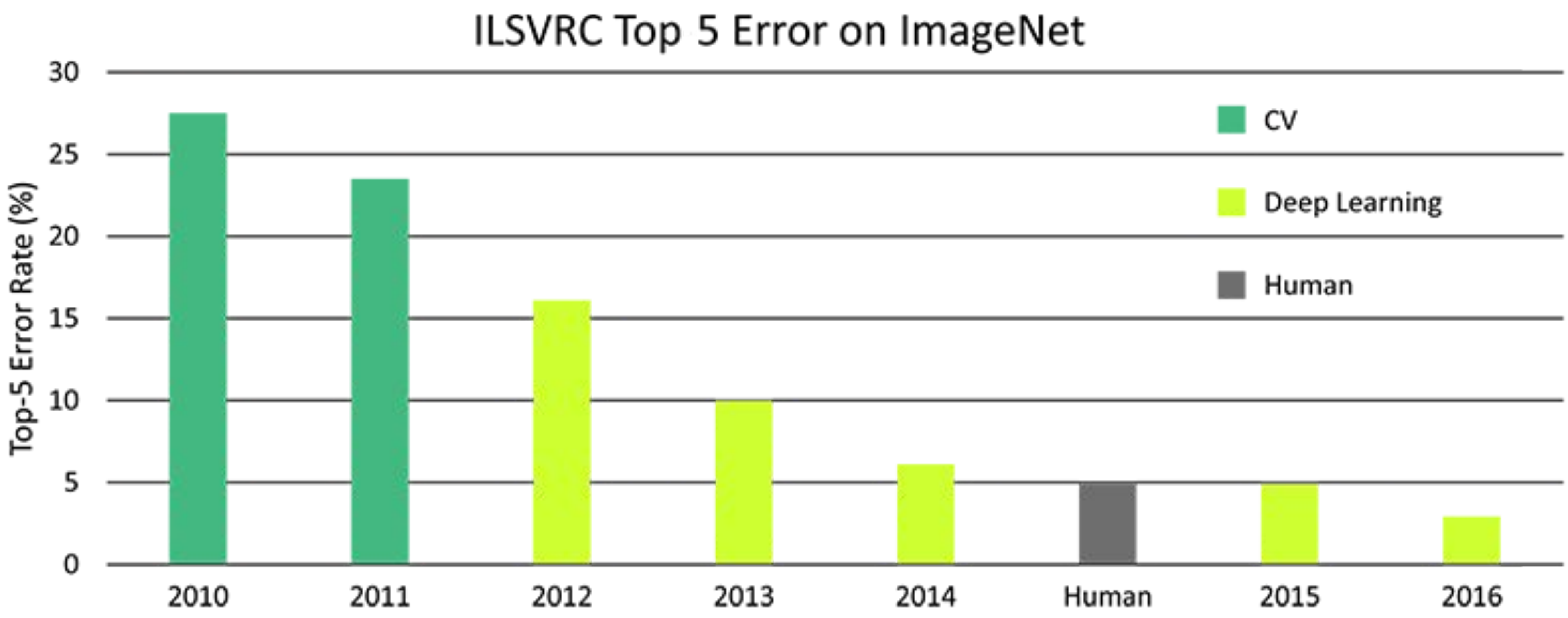
* Massive size
* High resolution
* High quality annotation
* Ontology

```
Carnivore
 - Canine
   - Dog
     - Working Dog
       - Husky
```

* Free

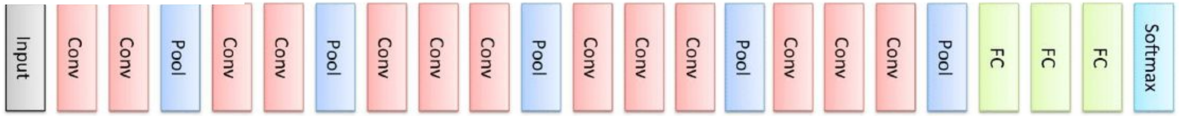
Credit: Fei-Fei Li

Deep Learning’s performance

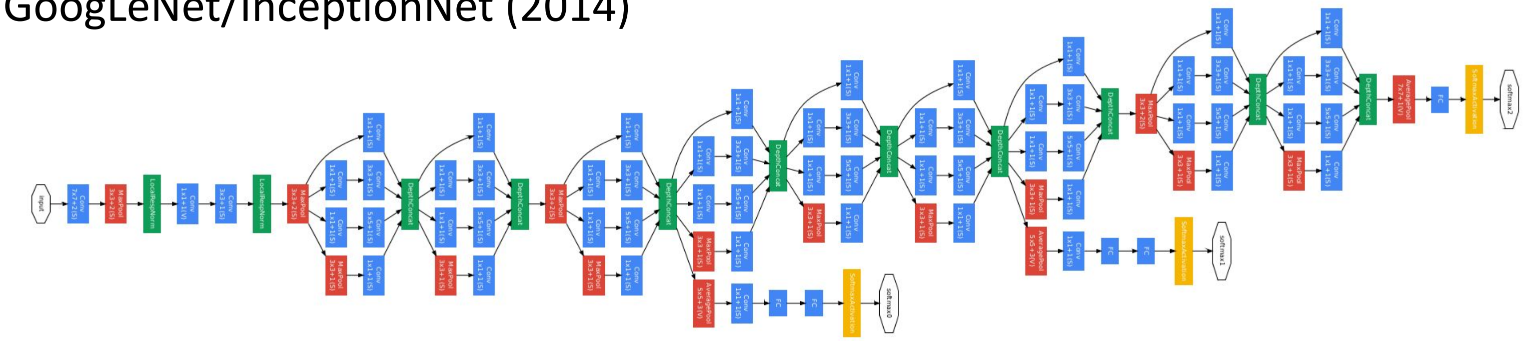


Most successful CNN models from ImageNet

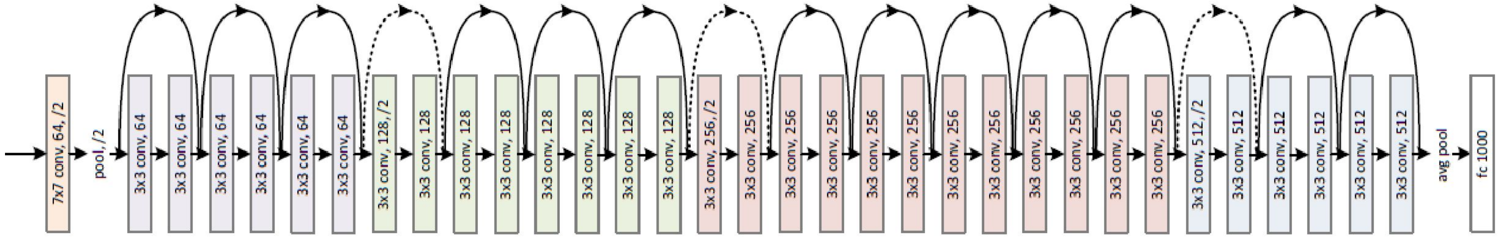
VGGNet (2014)



GoogLeNet/InceptionNet (2014)

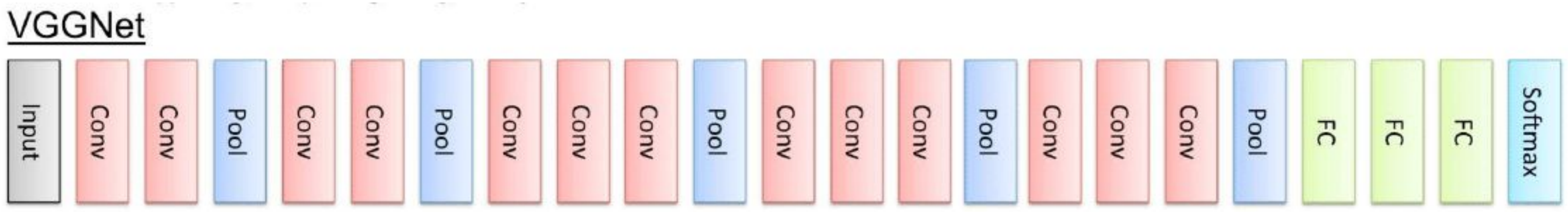


ResNet (2015)



Convolutional Neural Network

VGGNet

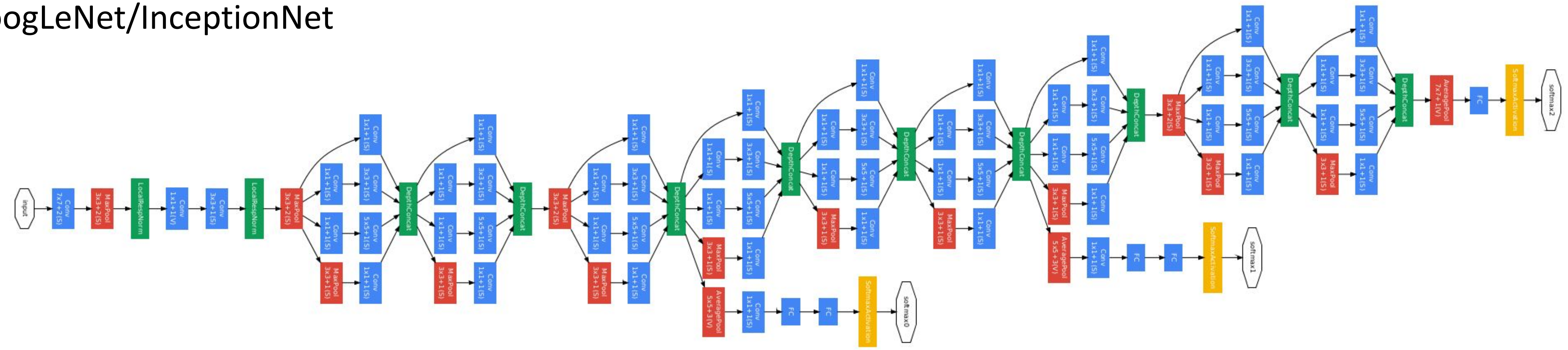


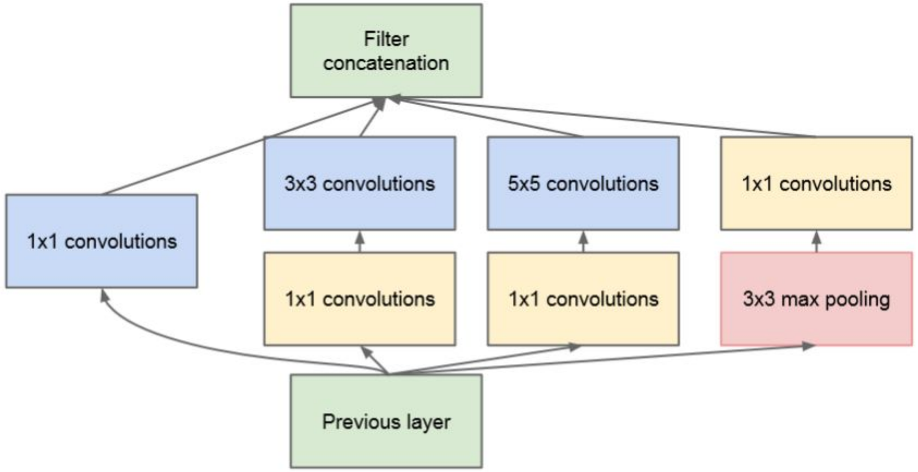
https://arxiv.org/abs/1409.1556

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

GoogLeNet

GoogLeNet/InceptionNet



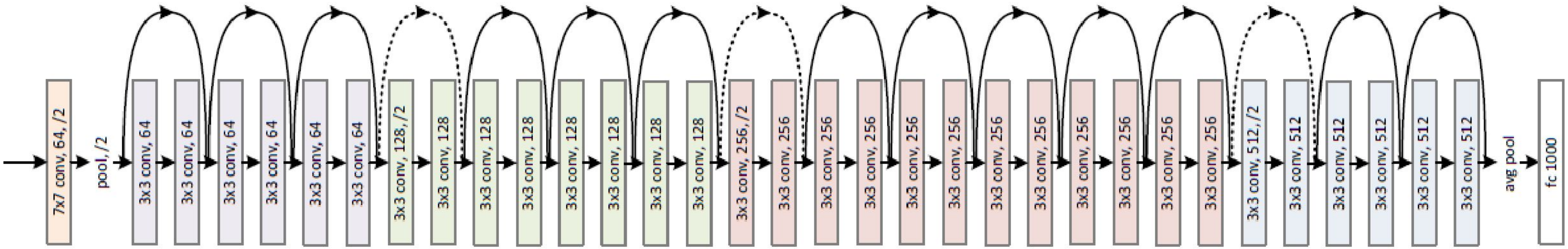


(b) Inception module with dimensionality reduction

https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

ResNet

ResNet



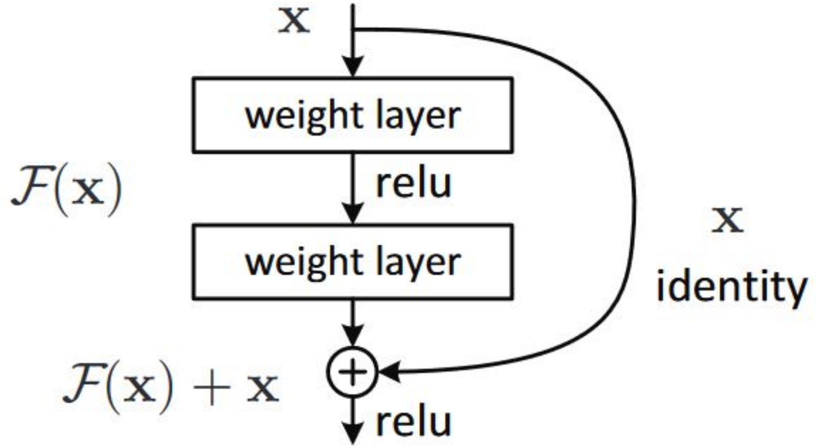
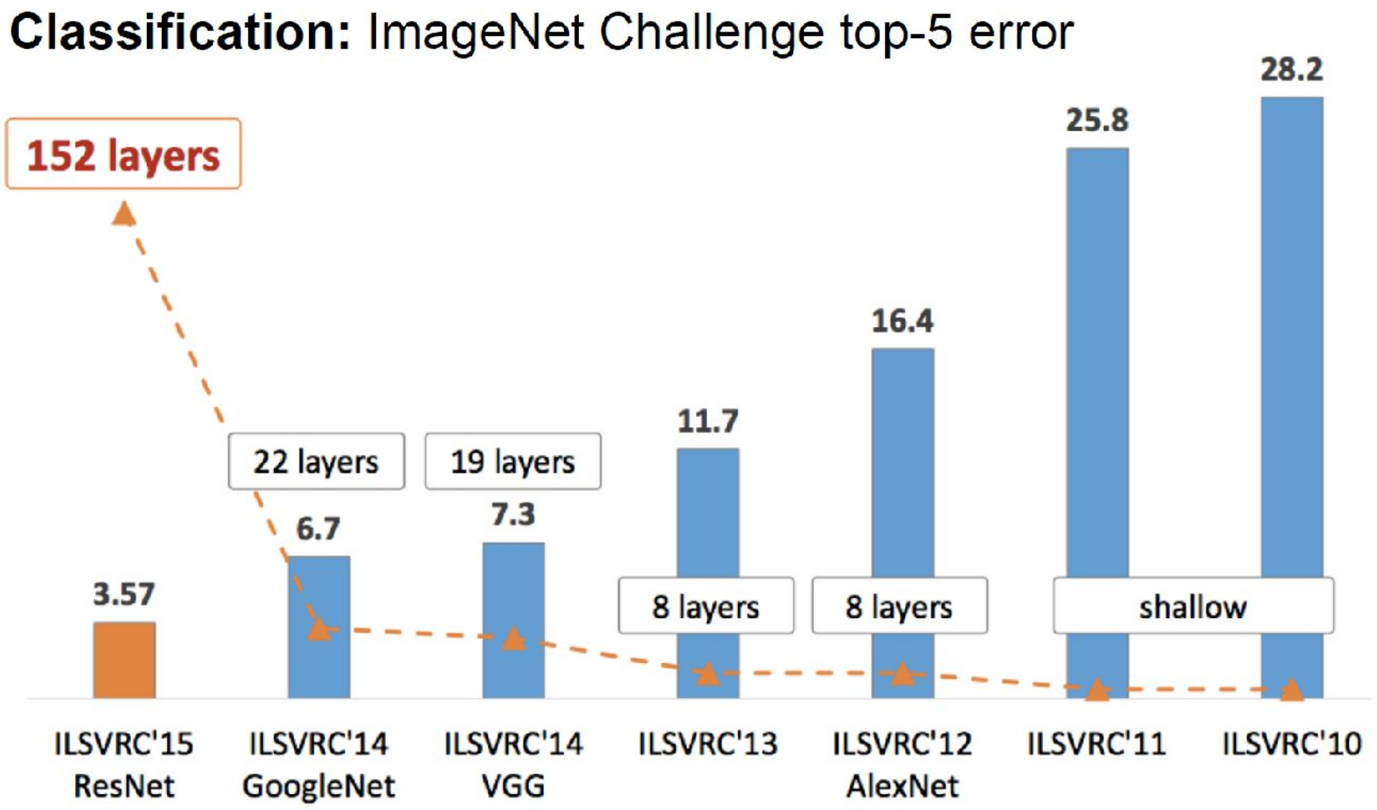


Figure 2. Residual learning: a building block.

http://arxiv.org/abs/1512.03385

ResNet



Credit: Kaiming He

Architecture summary

- If the data is bigger and complex, deep network is beneficial

- Improve computational efficiency while keeping the same depth
- e.g. small filter size, 1x1 filter compression

- Designs to have a better gradient propagation (avoid vanishing or exploding gradients)

Design Choices or Training

Key Tuning Parameters in Training

- Optimization method
- Learning rate, momentum, etc
- Number of epochs
- Regularization

Monitoring Overfitting in Training

Dataset split
Train / Validation / Test

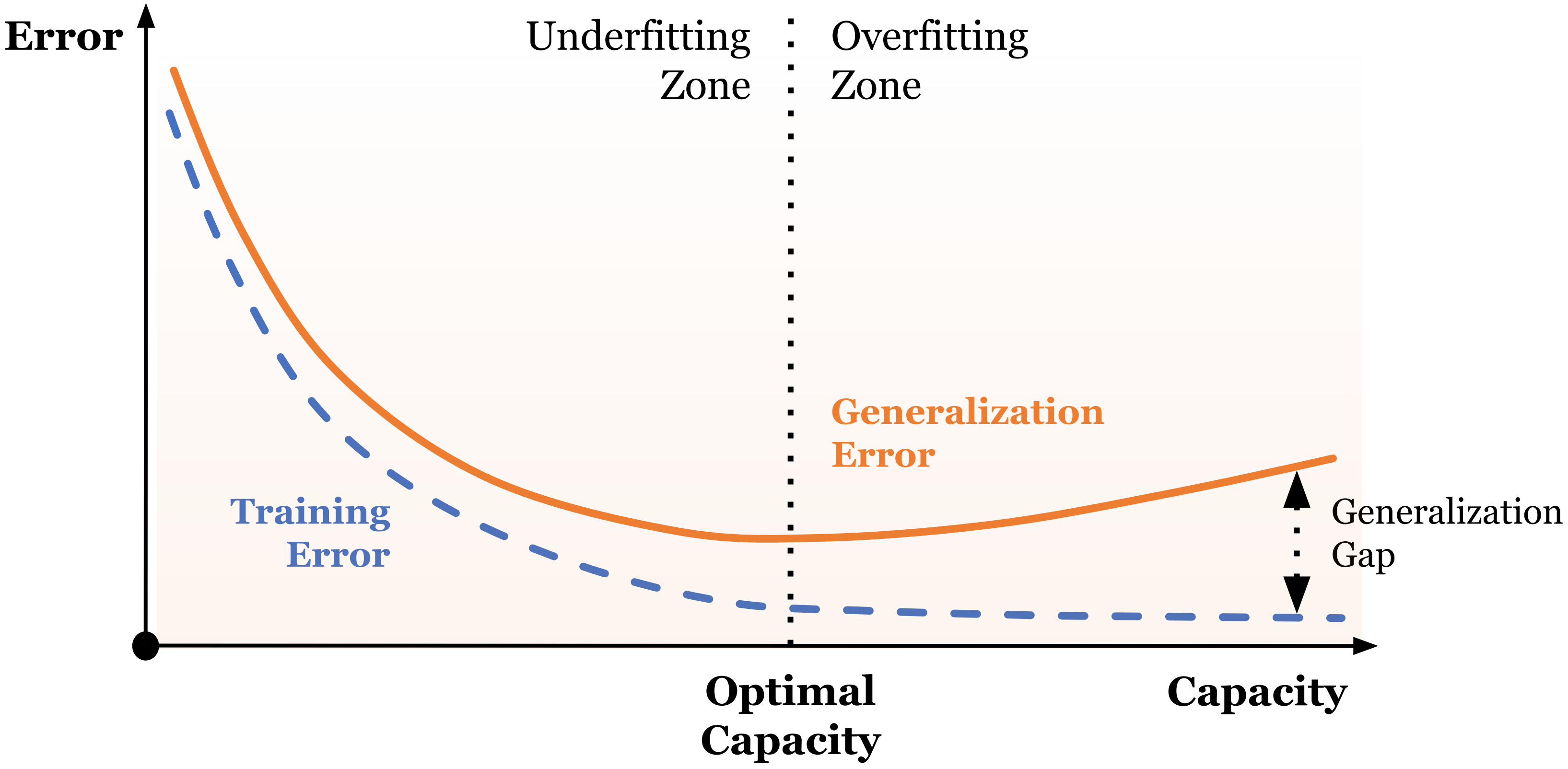
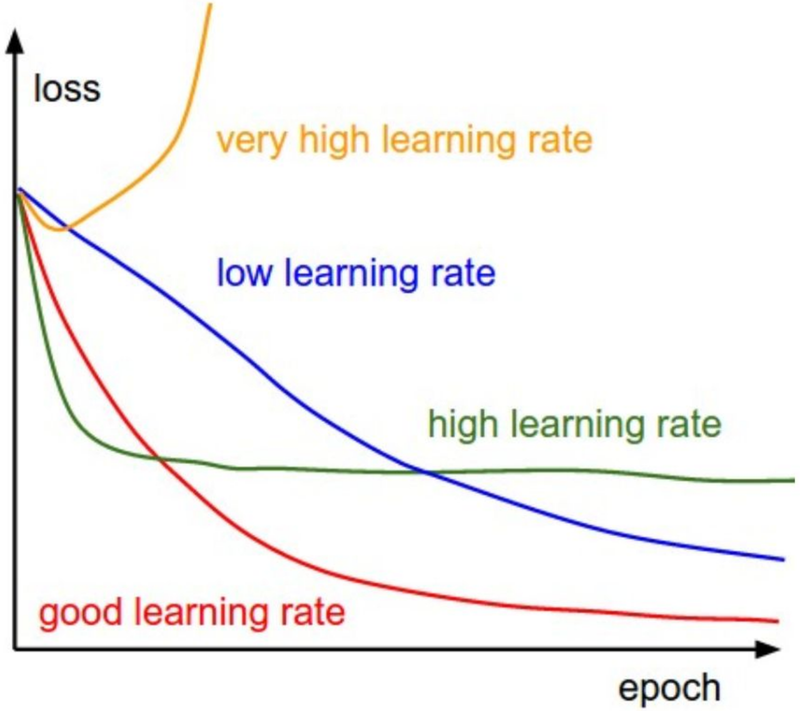
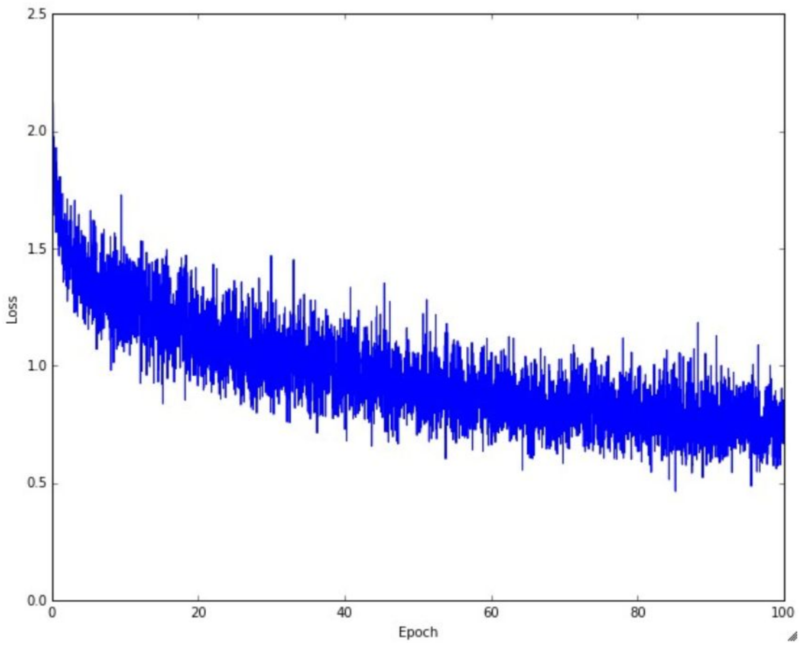
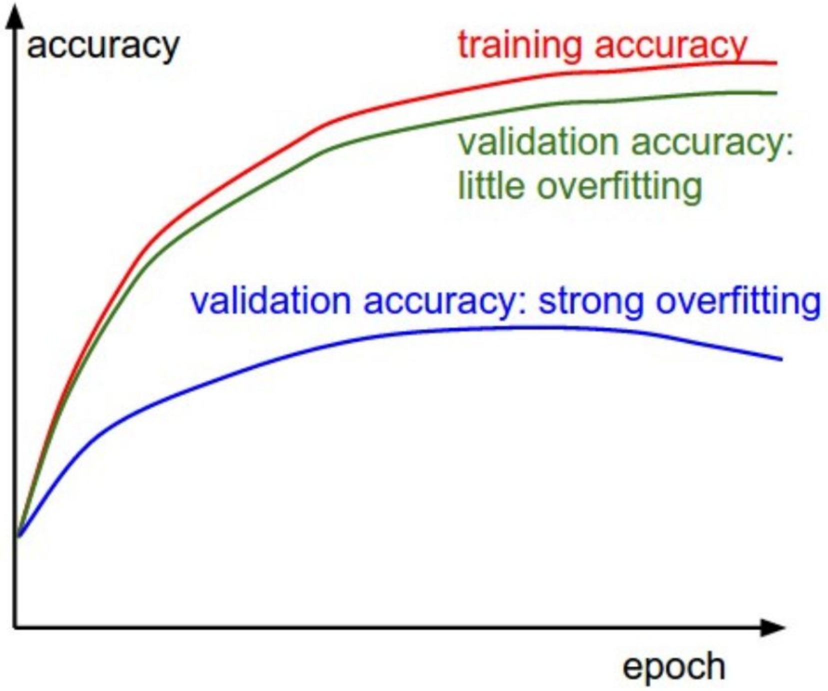


Diagram credit: Fei-Fei Li

Monitoring Overfitting in Training







http://cs231n.github.io/neural-networks-3/

SGD tuning parameters


```
tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.0, nesterov=False, name='SGD', **kwargs
)
```

Popular options to tweak

- learning_rate: the base learning rate
- momentum
- decay
- nestrov
- (advanced) callback

Stochastic Gradient Descent with momentum

SGD with learning rate alone is slow to converge

Adding a momentum (moving average of a weight) can make it faster

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right),$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}.$$

** see what happens when the gradient is 0 (on plateau)

warning- different notations used (from deeplearningbook.org)

Stochastic Gradient Descent with decay

Learning rate scheduling using decay

For iteration k (epoch)

$$\epsilon_k = (1-\alpha)\epsilon_0 + \alpha\epsilon_\tau \qquad \alpha = \frac{k}{\tau}$$

** In the algorithm pseudocode k is for step (each mini batch),
and decay learning rate by step,
but normally we decrease learning rate each epoch

warning- different notations used (from deeplearningbook.org)

Advanced optimization

RMSprop

Variant of Adadelta
RMSprop takes a moving average when it calculate the RMS of the gradient

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

An overview of gradient descent optimization algorithms
https://arxiv.org/pdf/1609.04747.pdf

warning- different notations used

Advanced optimization

Adaptive Moment Estimation (Adam)

Mimics momentum for gradient and gradient-squared

m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

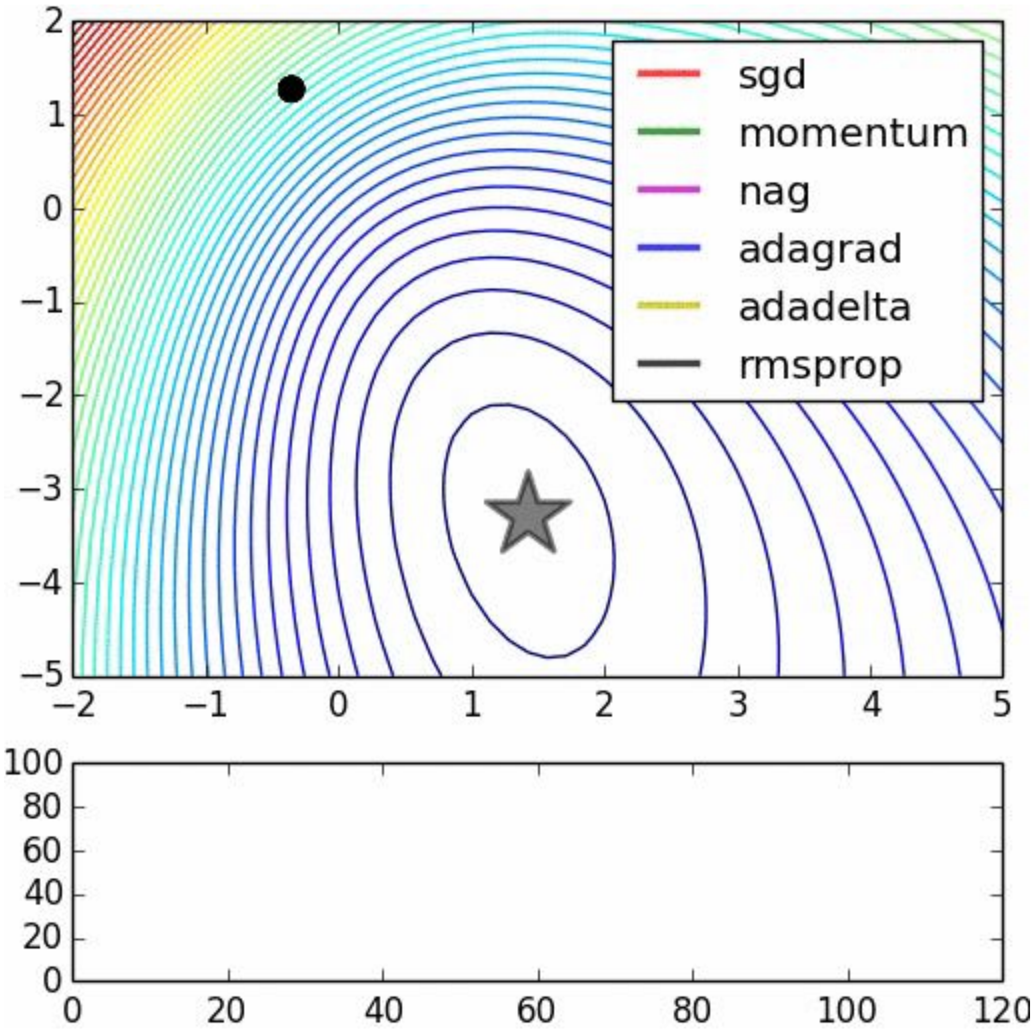
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

An overview of gradient descent optimization algorithms

https://arxiv.org/pdf/1609.04747.pdf

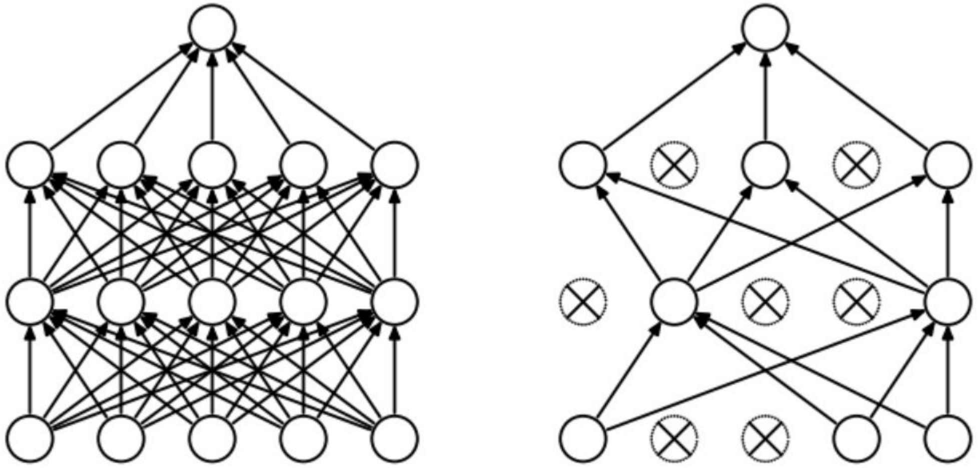
warning- different notations used

Optimization: Momentum



Ways to reduce overfitting

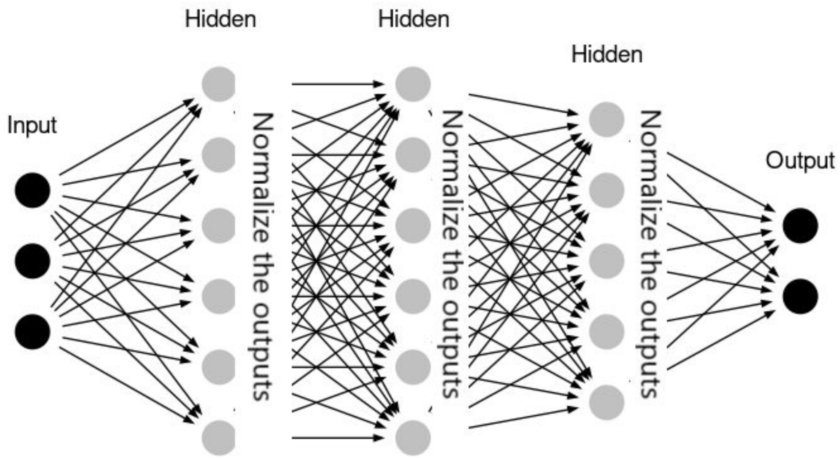
Dropout



(a) Standard Neural Net

(b) After applying dropout.

Batch normalization



Practical Tips for CNN

Optimization

- Learning rate (0.01~0.0001)
- Optimization method: Adam or RMSProp

Architecture

- ReLU/PReLU for hidden layers, Sigmoid/Softmax/Tanh/PReLU for the output layer
- 3x3 filters
- $[\text{Conv-Conv-MaxPool}]_n$ structure

Regularization

- L2 regularization
- Dropouts
- Batch Normalization

Transfer Learning

How long it takes to train on a large dataset?

Multi GPU + Deep CNN + Large Dataset = 2-3 Weeks Training time

Transfer learning

- As fixed feature extractor: remove the output layer, weights frozen
- Fine-tuning the CNN: also let weights updated
- Use part of layers

Transfer Learning

When should I use a pre-trained network?

- New dataset is small and similar to original dataset (X)
- New dataset is large and similar to the original dataset (O)
- New dataset is small but very different from the original dataset (X)
- New dataset is large and very different from the original dataset (O)

Various models pre-trained on ImageNet

Keras https://keras.io/applications/