Advanced Convolutional Neural Networks

Reading: Chapter 7.1, 7.2, 7.5, 7.6 of the book "*Dive into Deep Learning*".

Outline

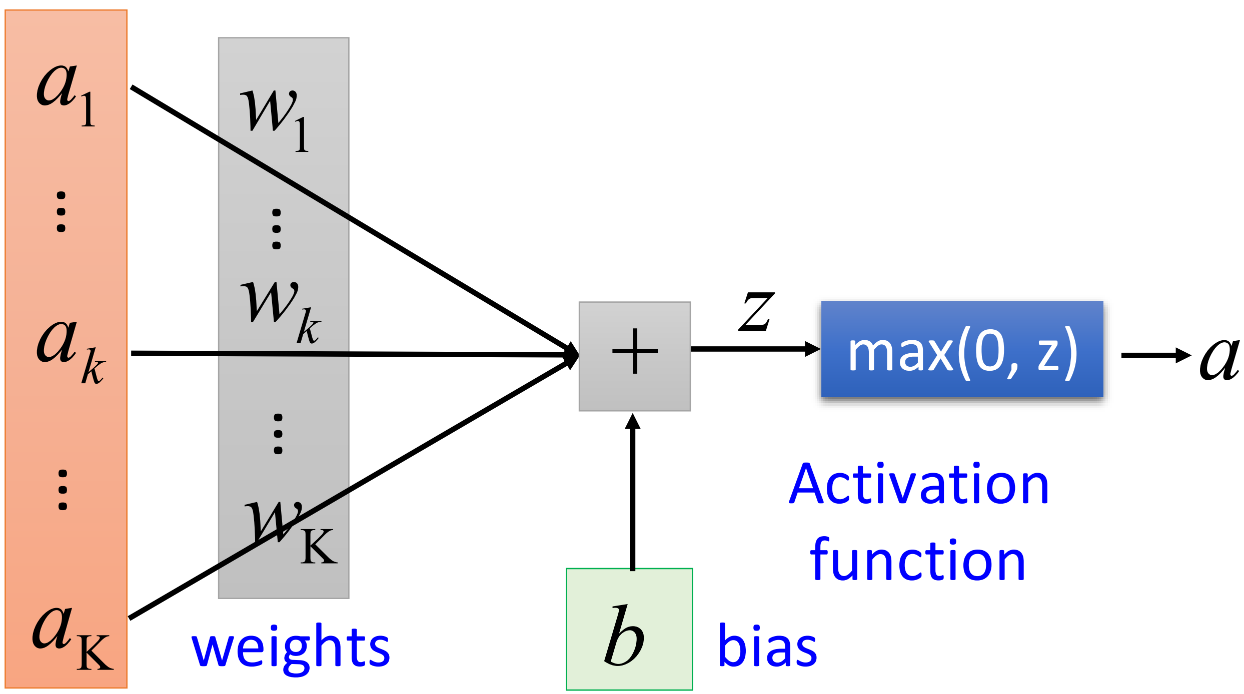
- Review of Neural Network Basics
- Evolution of Advance Neural Networks
  - Alex Net 2012
  - VGG Net 2013
  - Residual Networks 2014
- Concluding Remarks

Components of NN

- Neurons
- Fully Connected Layer
- Convolution Layer
- Activation: ReLU, Sigmoid
- Pooling: Maxout, Average

Neurons

$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$



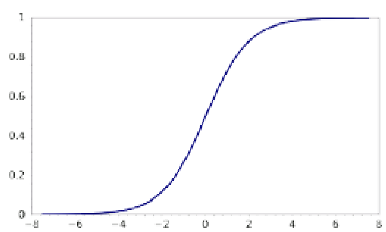
Nonlinearity

Non-Linear Activation Function

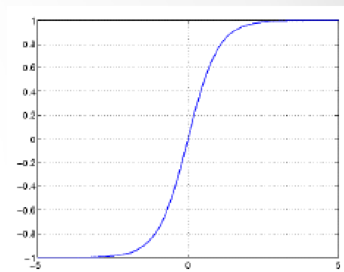
- Sigmoid: $$S(t) = \frac{1}{1 + e^{-t}}.$$
- Tanh: $$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
- Rectified Linear Unit (ReLU):

$$f(x) = \max(0, x)$$

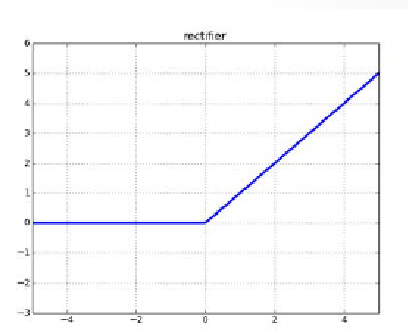
Most popular activation function for DNN as of 2015, avoids saturation issues, makes learning faster



Sigmoid

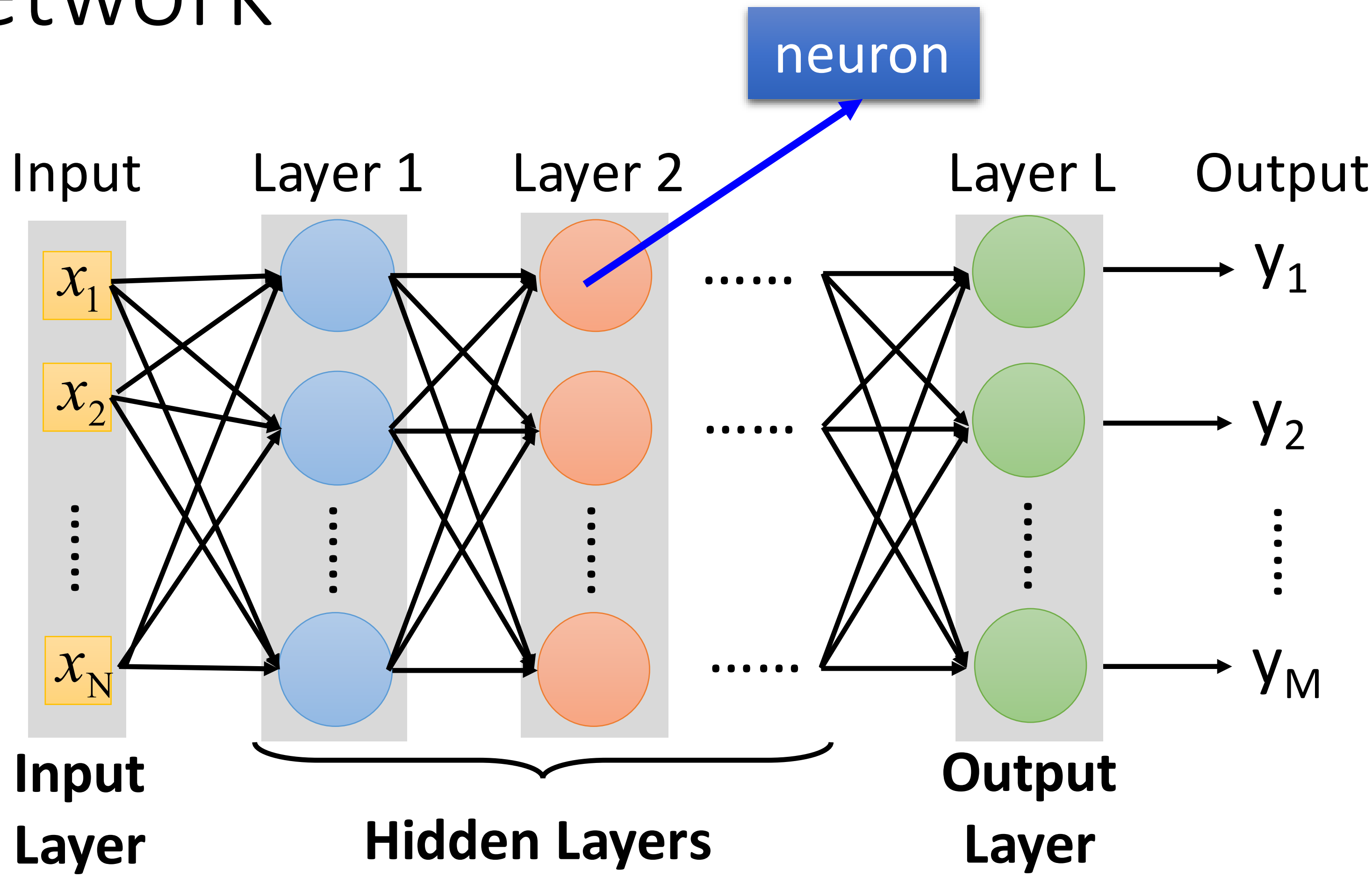


Tanh



ReLU

Fully Connect Feedforward Network



Deep means many hidden layers

- Fully connected traditional networks
  - *m* inputs in a layer and *n* outputs in next layer
  - requires to learn *m* x *n* weights

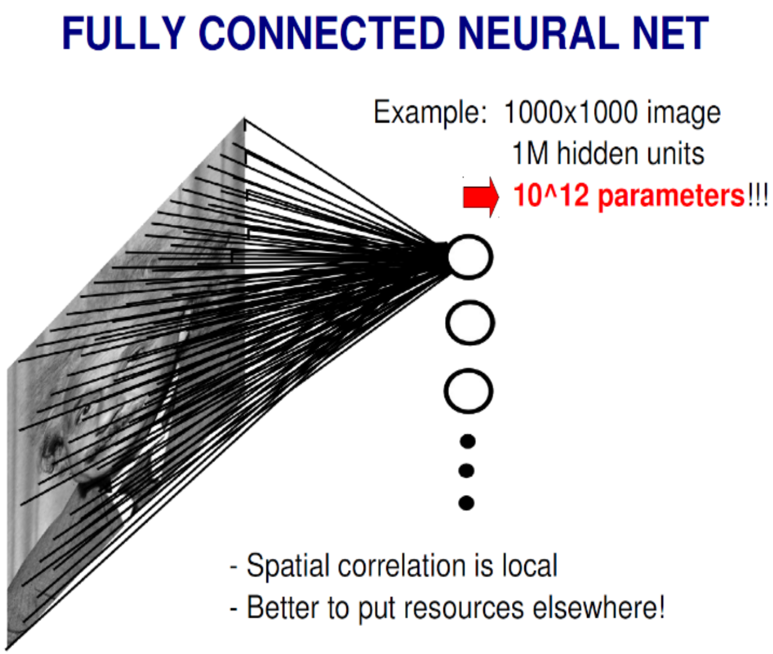


Figure from [1]

Convolution Layers

- Specially designed for data with grid-like topology
  - 1D grid – time series data
  - 2D grid –Most successful on 2D image topology
- Parameter sharing
- Sparse Interactions
  - Less number of parameters to learn
  - Local feature learning

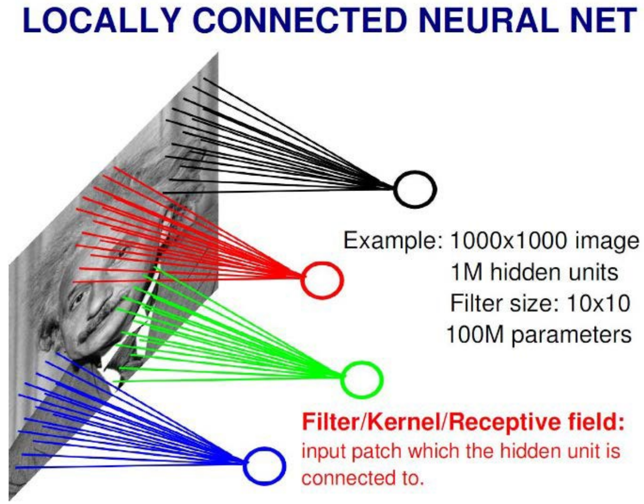
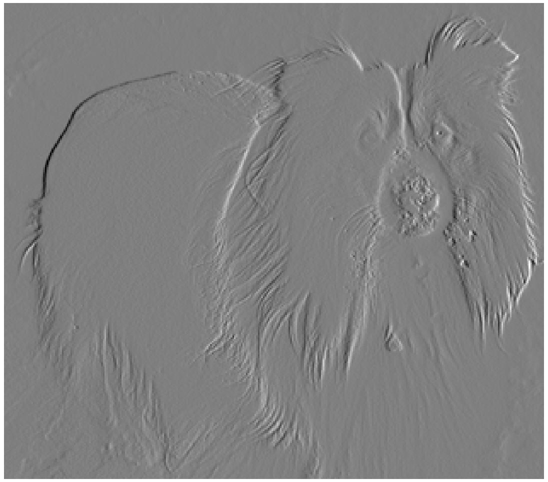


Figure from [1]

An Example: Edge Detection

-1	1

Kernel



Input

Output

Right image = each orig pixel – left pixel detects edges

Figure from [1]

Convolution with multiple channels

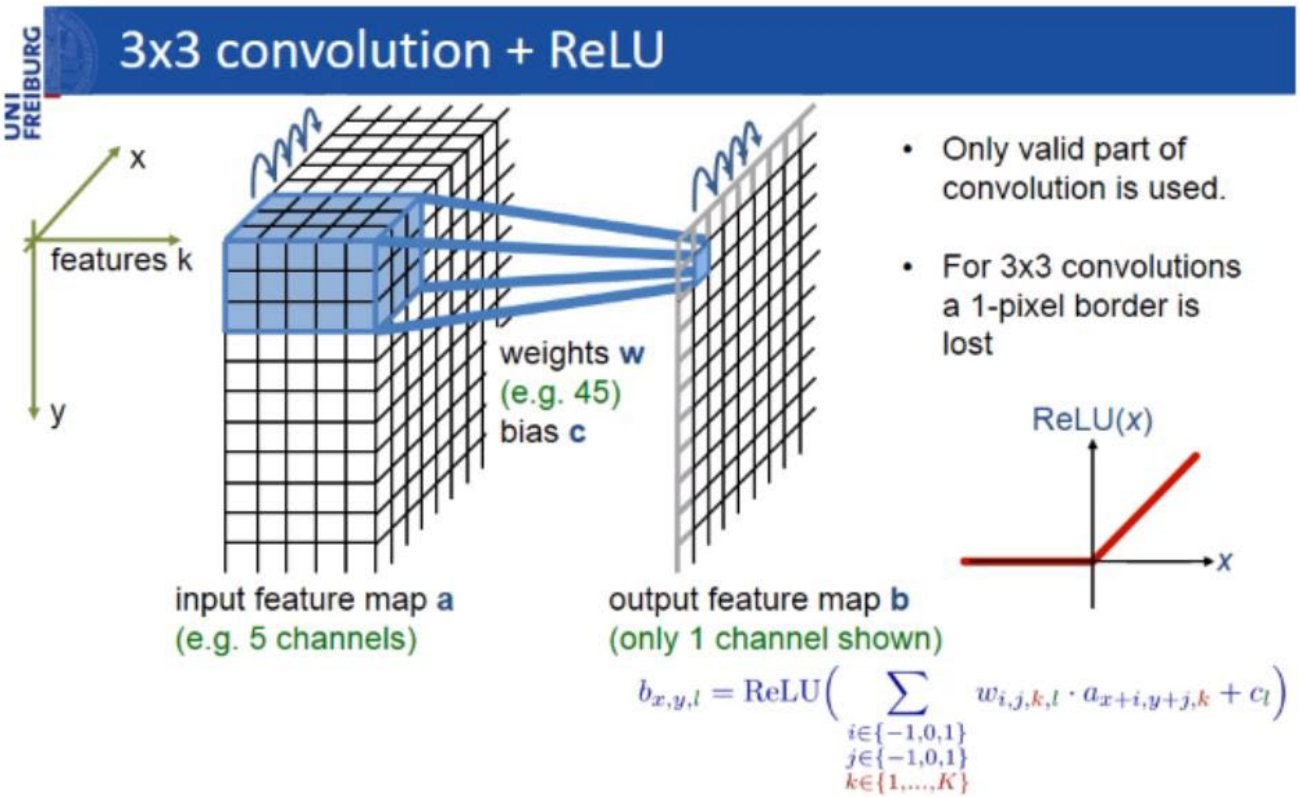
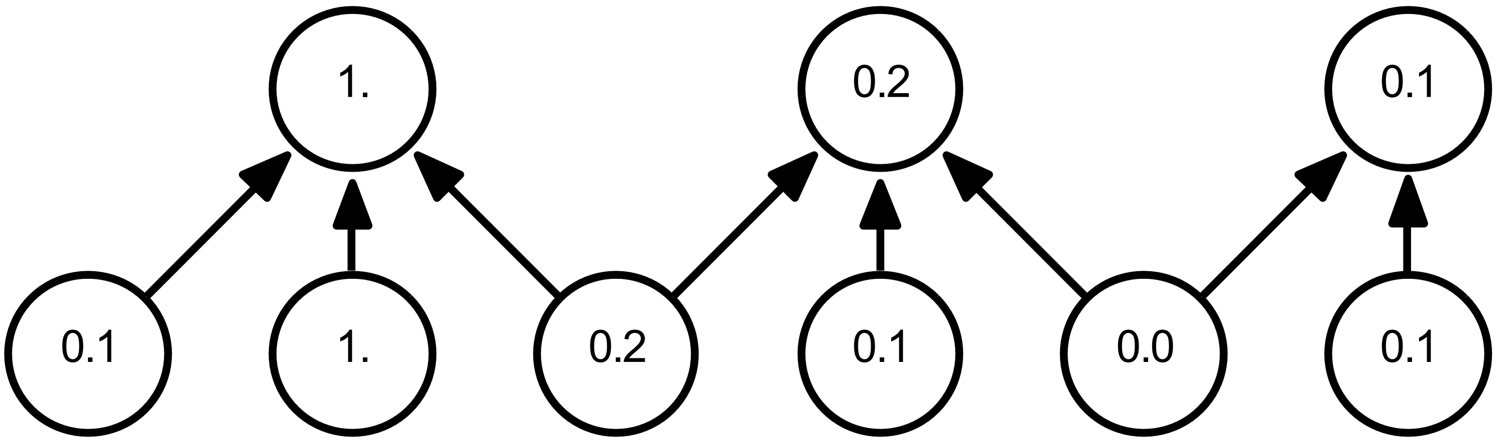


Figure from https://heartbeat.fritz.ai/deep-learning-for-image-segmentation-u-net-architecture-ff17f6e4c1cf

Pooling

- The pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs.
- Non-linear down-sampling to simplify the information in output from convolutional layer.
- Variants:
  - Max pooling (popular): reports the maximum output within a rectangular neighborhood
  - Average pooling: reports the average output

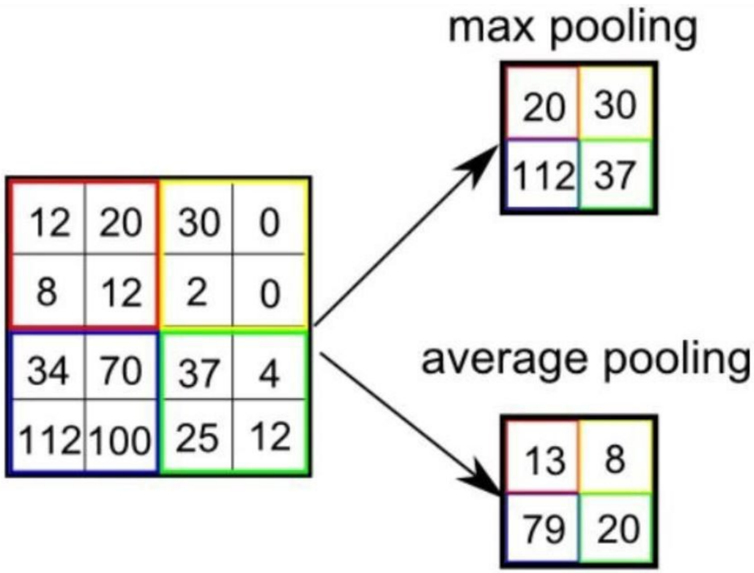
Max Pooling



Max pooling
downsized in next layer

Figure from [1]

Pooling



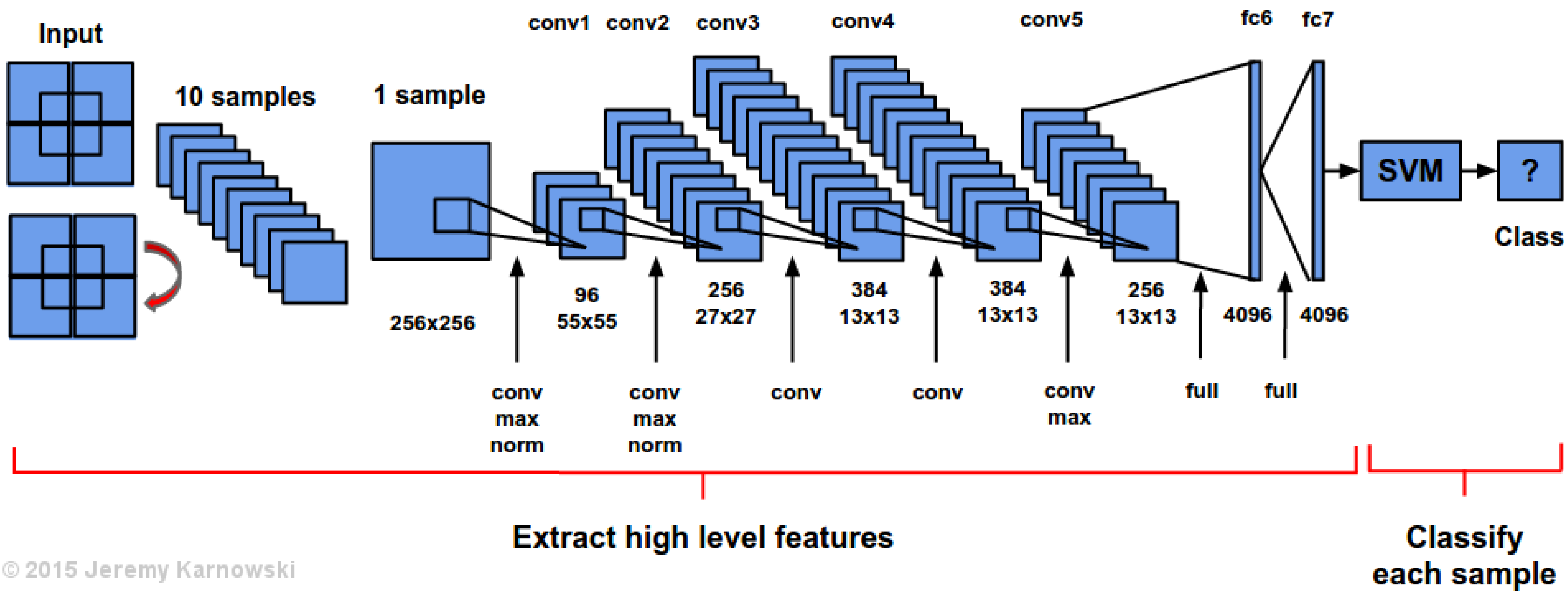
Representation: Max and Avg. Pooling

Figure from https://heartbeat.fritz.ai/deep-learning-for-image-segmentation-u-net-architecture-ff17f6e4c1cf

Evolution of Neural Network Architectures

- Alex Net: 8 layers
  - A Krizhevsky et al. NIPS 2012
- VGG Net: 19 layers
  - K. Simonyan et al., ICLR 2015
- Residual Net: 152 layers
  - K. He et al., CVPR 2016

Alex Net



Input
10 samples
1 sample
256x256
conv1
conv2
conv3
conv4
conv5
fc6
fc7
96
55x55
256
27x27
384
13x13
384
13x13
256
13x13
4096
4096
conv
max
norm
conv
max
norm
conv
conv
conv
max
full
full
SVM
?
Class
Extract high level features
Classify
each sample

Alex Net *cont*

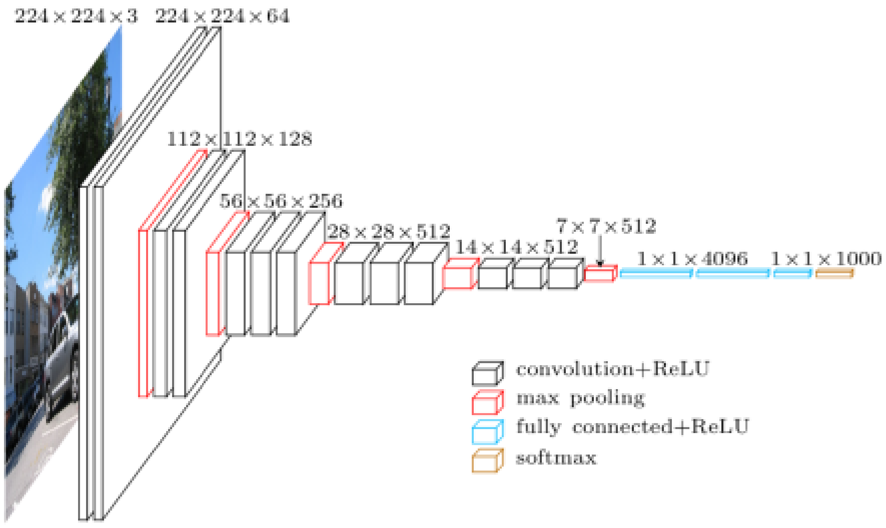
- Five CNN layers, 2 fully connected layers
- Surprisingly, local minima does not appear an issue in training
  - ReLu,Max pooling, Drop out
  - Data Augment
  - Large training data: 1.3 m images, 1000 classes

Alex Net *Cont*

- Winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012
- Reducing the top-5 error from 26% to 16.4%.
- The second place: 26.2%

VGG Net

- Small filter size
- More layers

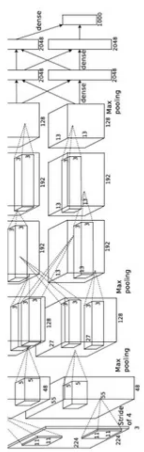


Deep = Many layers

http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

8 layers

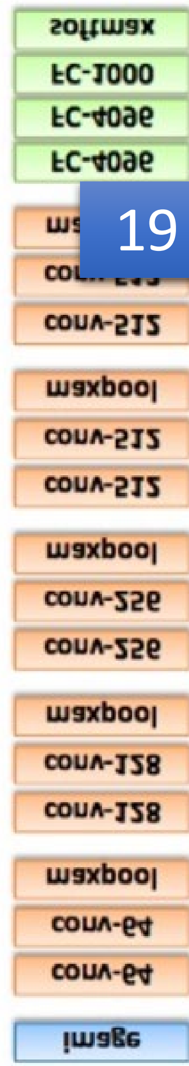
16.4%



AlexNet (2012)

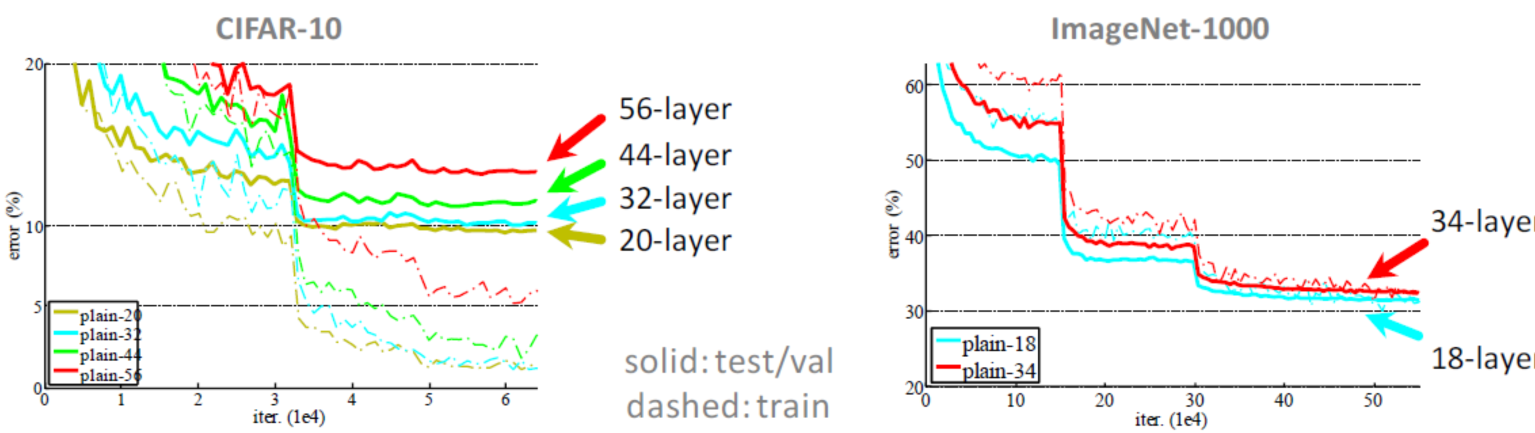
19 layers

7.3%



VGG (2014)

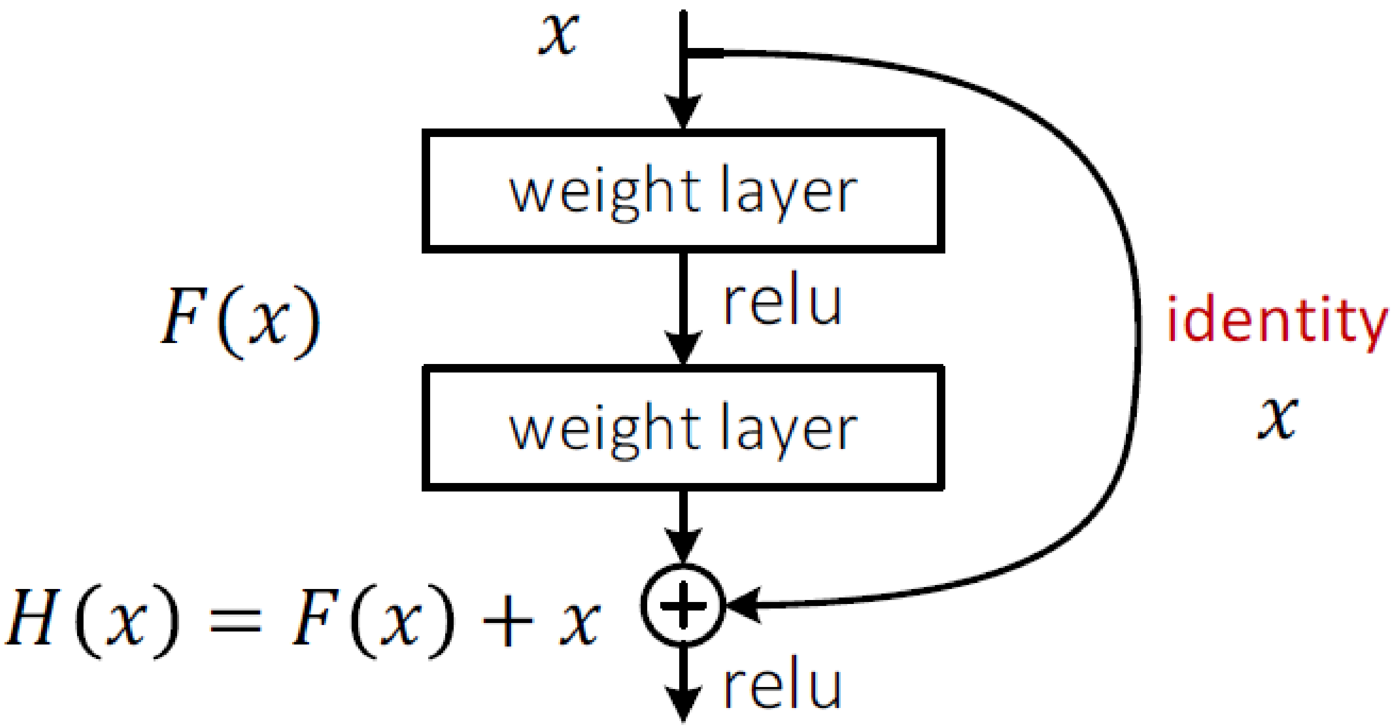
Stacking for Deep Nets?



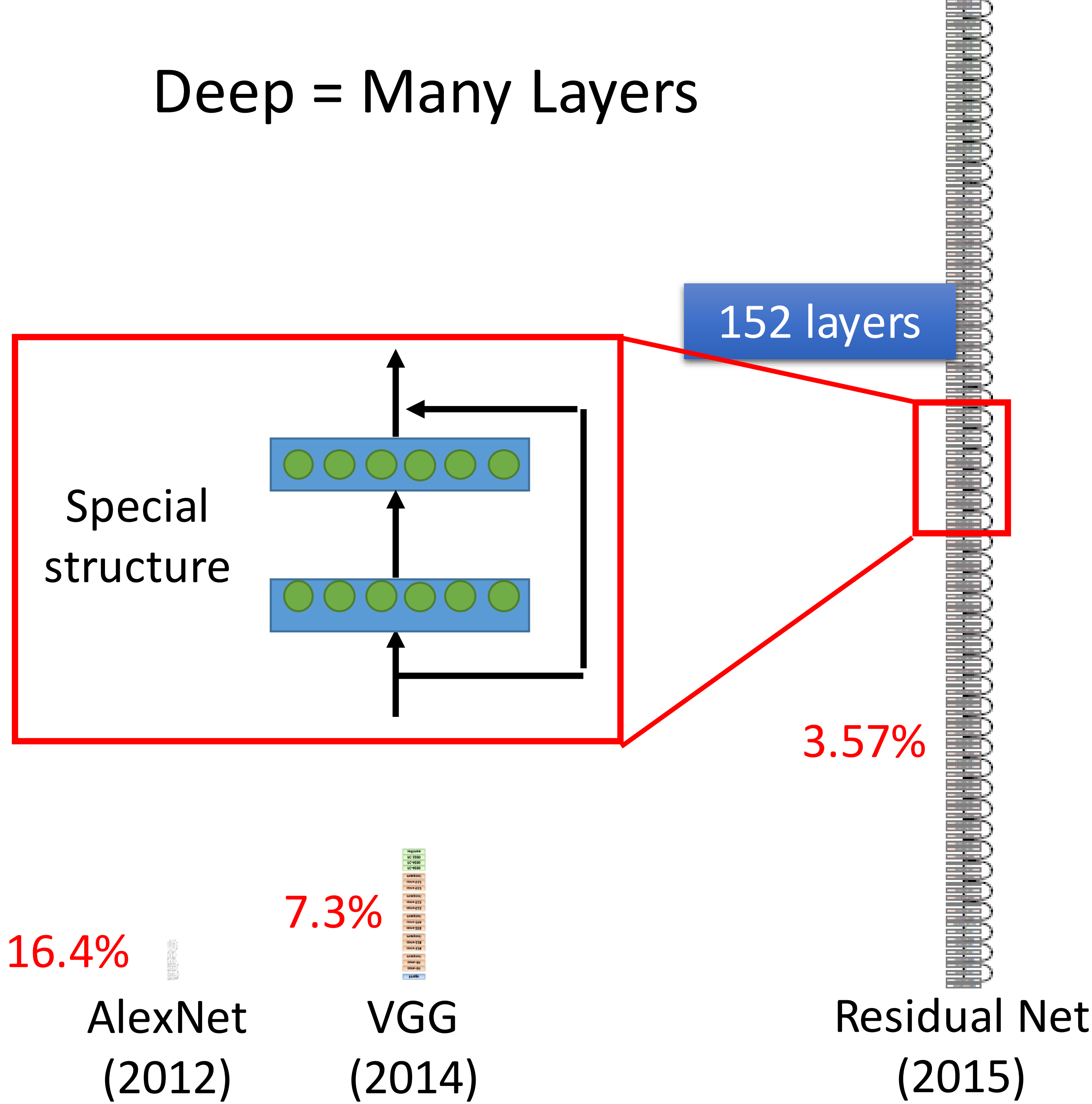
Residual Network

- 152 layers
- lower complexity than VGGNet.
- Top-5 error rate: 3.57% which beats human-level performance.
- Key techniques:
  - Unity skip connections to overcome diminishing gradient problem, any odd layer has a direct path to the output layer
  - Batch Normalization

Residual Unit



Deep = Many Layers



ResNet

- Core units for many other learning architectures including
  - Reinforcement learning for Alpha Go
  - Generative adversarial networks (GAN)
- Key components:
  - Convolution layers
  - Relu
  - Max pooling
  - batch normalization
  - *skip connection*

What Makes Deep Learning So Successful?

- Depth of Neural Nets
  - Hypothesis: Deeper net generalize better
    - Verified in many practical applications
  - Skip connection enables the training of deep networks
- Successful Training: surprisingly, local minima does not appear a big issue
    - Big Data, GPU, Smart Algorithms
- The three requirements for ML are satisfied

GPU Processing

- GPU processing for big data
  - Cost function: additive across training instances

$$\mathbb{E}_{\boldsymbol{x},\mathrm{y}\sim\hat{p}_{\text{data}}(\boldsymbol{x},y)}[L(f(\boldsymbol{x};\boldsymbol{\theta}),y)] = \frac{1}{m}\sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),y^{(i)}),$$

  - Gradient: additive across training instances
  - Can be processed efficiently with GPU

Training: learning rates

- Small: slow, stuck in poor local minima with high training error

- Large: oscillating performance; may not converge; training error may increase.

- Manual selection: first large then small

- Adaptive: Adam (2015), LookAhead (2019), Rectified Adam (RAdam, 2019)

Regularisation

- Aim: Reduce overfitting and improve generalization performance
- Data Augmentation
- Early Stopping
- Batch normalization
- Drop out
  - Randomly dropping out nodes during training.
  - Computationally cheap and remarkably effective.
  - Why? Ensembles of neural networks are known to reduce overfitting

Drop out

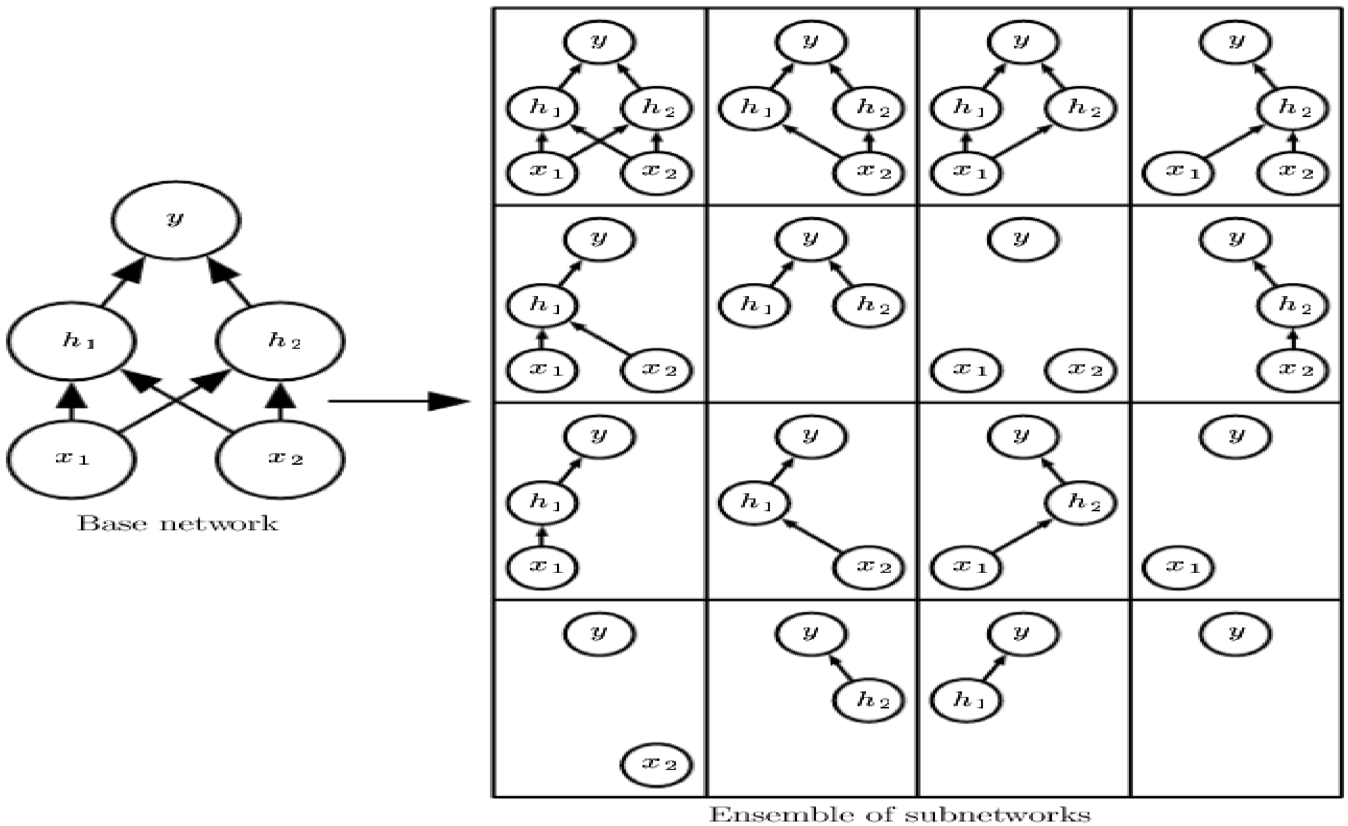


Figure from [4]

Early stopping

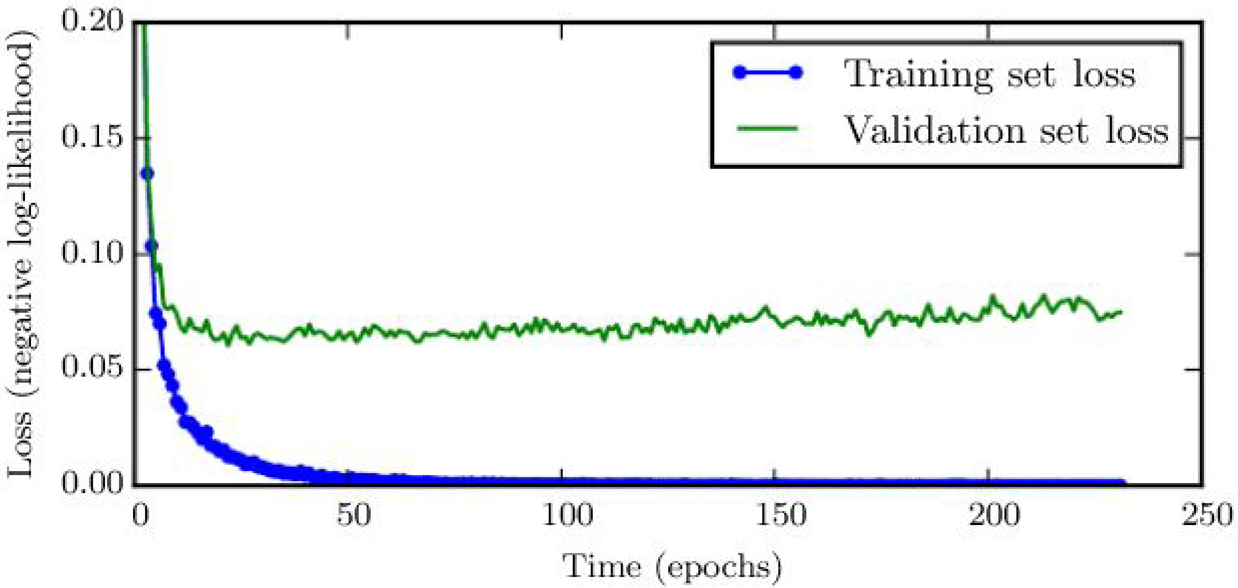


Figure from [4]

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Figure from [4]

Transfer Learning: CNN features off the shelf

- CNN feature extraction from pre-trained models on ImageNet
  - Trained feature from datasets
  - Adaptive (not in fixed position)
  - Hierarchical feature learning: low-level, high-level, abstract
- Train classifiers based on CNN features for small data sets
- VGG16 works remarkable good in many applications [2]
- The performances of the CNN features are highly correlated to the performance of the models on ImageNet

Concluding Remarks

- Deep learning solves a complex non-convex optimization problem with the help of big data, GPU
- Deep learning fulfils the three essential requirements of machine learning: capacity, compactness and learnability
- Deep learning has achieved unprecedented success in machine learning and computer vision
- The expansion of deep learning to wide range of applications is tremendously fast