Deep Learning & Tools

Asst.Prof.Rapeeporn Chamchong, PhD

7th August,2021

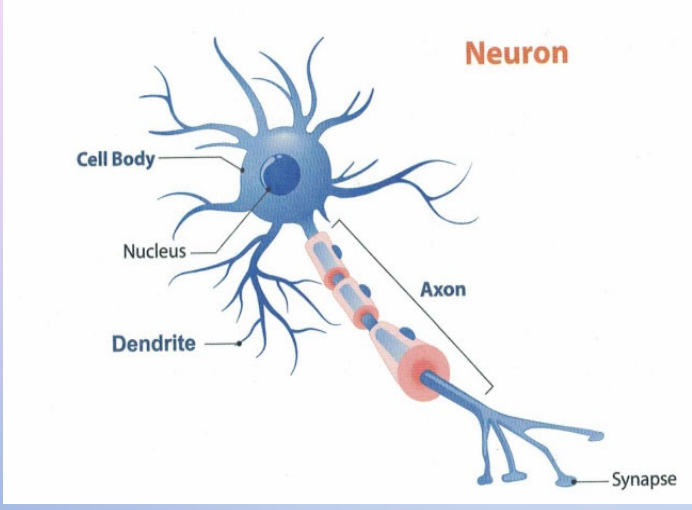
Outlines

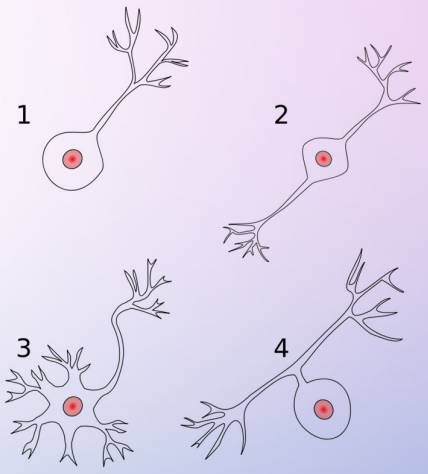
- Deep Learning
- Colaboratory, Google drive connection, read/write file on Google drive
- Read/write Zip file, Pandas package
- Tensorflow, Keras, Scikit-learn tools
- Data preparation and Convolutional Neural Networks (CNNs)
- Feature extraction from CNNs
- Recurrent Neural Networks (RNNs)
- Combination of CNNs and RNNs
- Data preparation for image detection using deep learning
- Image detection using R-CNN, Faster R-CNN, YOLO and SSD

Deep Learning

- Overview of Artificial Neural Network (ANN)
  - Actual Neurons and Perceptron
  - Multi-Layer Perceptron
- Deep Learning
  - Convolutional Neural Networks (CNNs) Based Computer Vision
  - Recurrent Neural Networks (RNNs)
- Representation of Input Image
- TensorFlow

Neurons





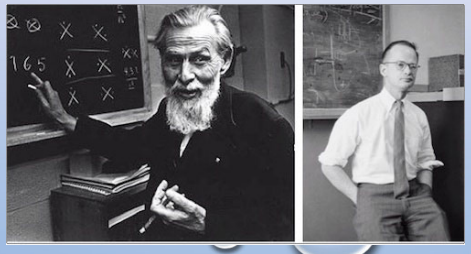
Different kinds of neurons:

1 Unipolar neuron
2 Bipolar neuron
3 Multipolar neuron
4 Pseudounipolar neuron

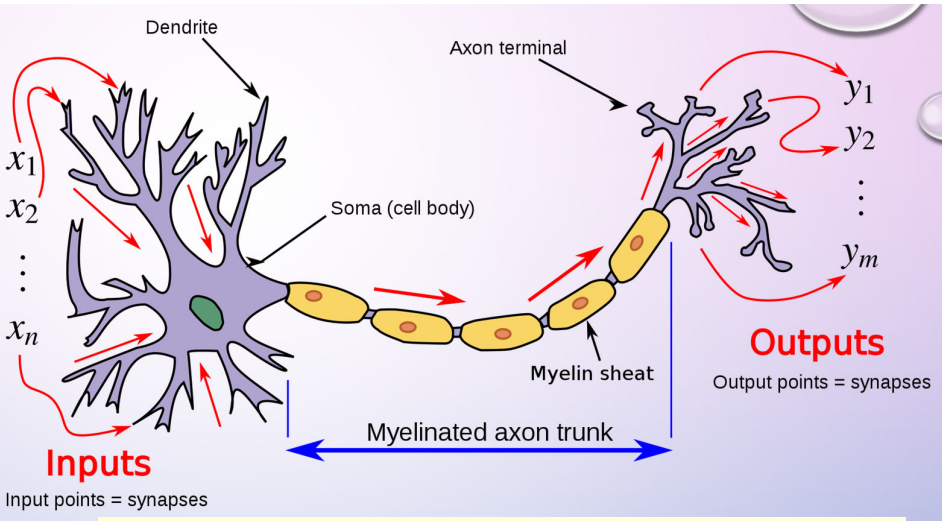
wikipedia

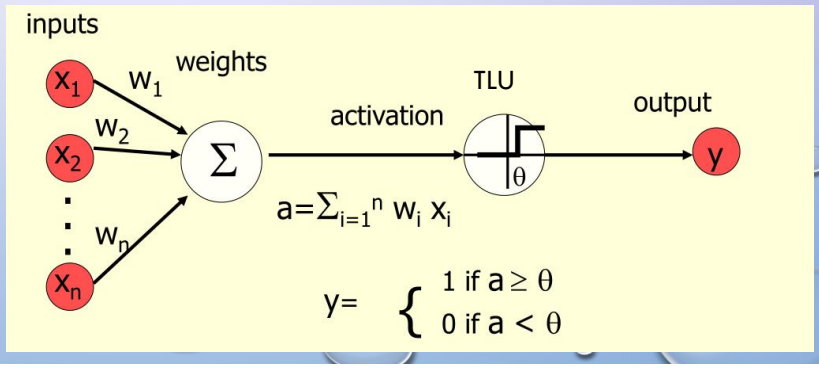
Artificial neuron

- An artificial neuron is a mathematical function conceived as a model of biological neurons, a neural network.
- Artificial neurons are elementary units in an Artificial Neural Network (ANN).
- Threshold Logic Unit (TLU)
  - The early artificial neuron was the TLU, or Linear Threshold Unit, first proposed by Warren McCulloch and Walter Pitts in 1943.



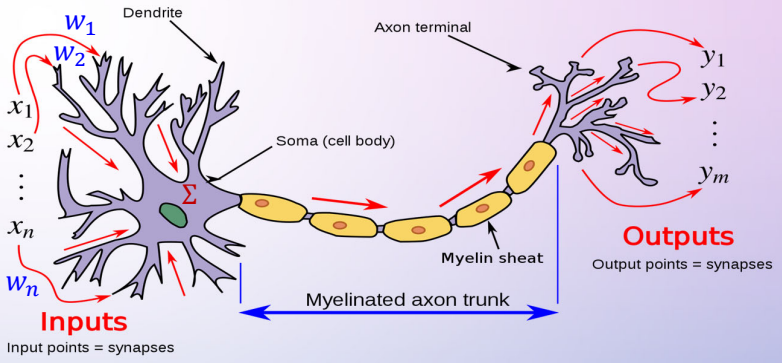
Neuron Network & Mathematical Model



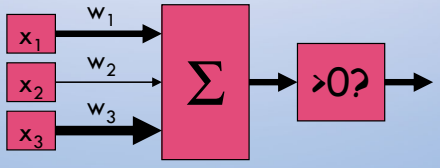


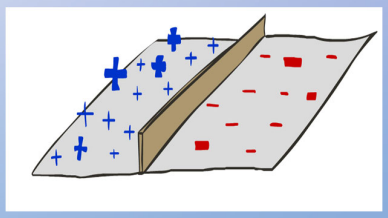
Linear Classifiers

- Inputs are feature values
- Each feature has a weight
- Sum is the activation



- If the activation is:
  - Positive, output +1
  - Negative, output -1





How to determine the values of W?

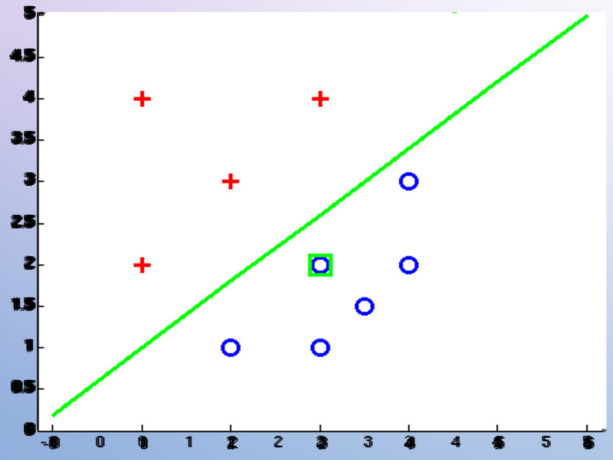
- How to solve for W

$$f(x) = \begin{cases} 1 & if \sum w_i \cdot x_i + b > 0 \\ 0 & otherwise \end{cases}$$

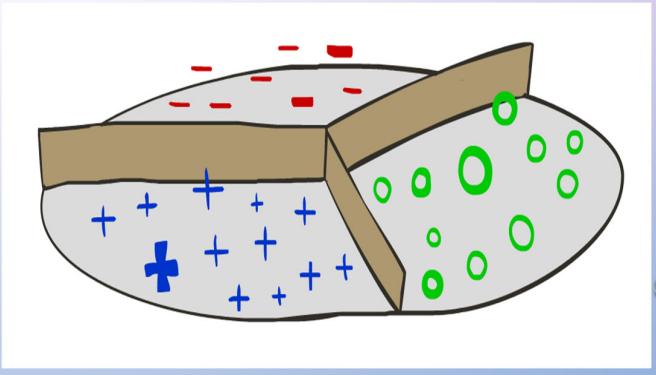
  - Randomly
  - Analytically
  - Optimization algorithm: e.g. gradient descent

Examples: Perceptron

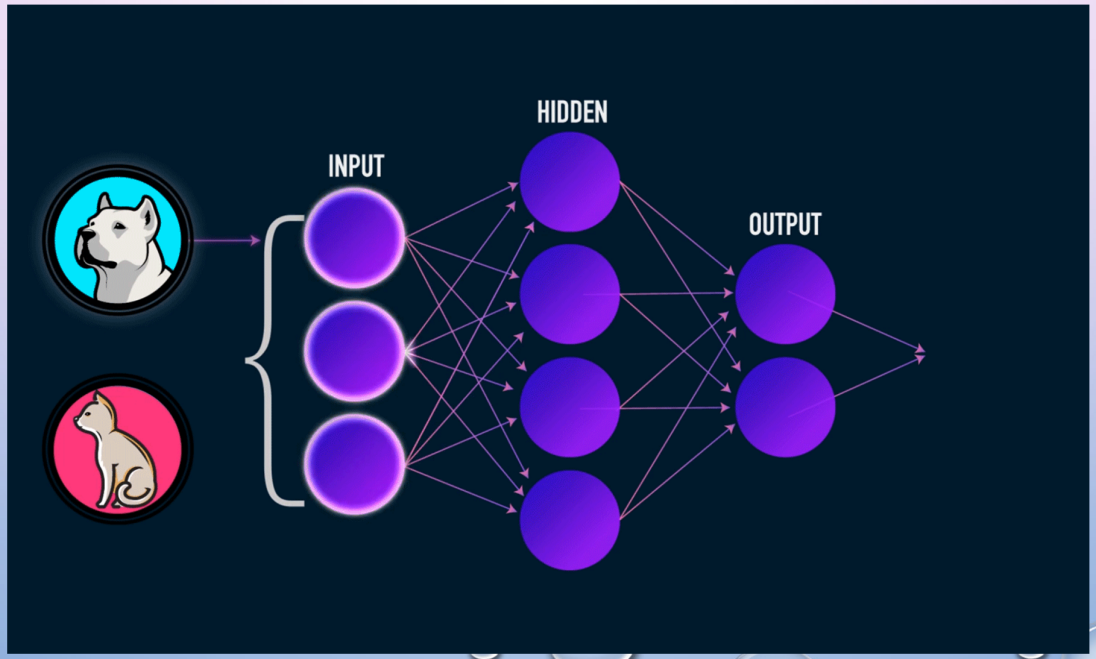
Binary class decision



Multiclass decision



Example



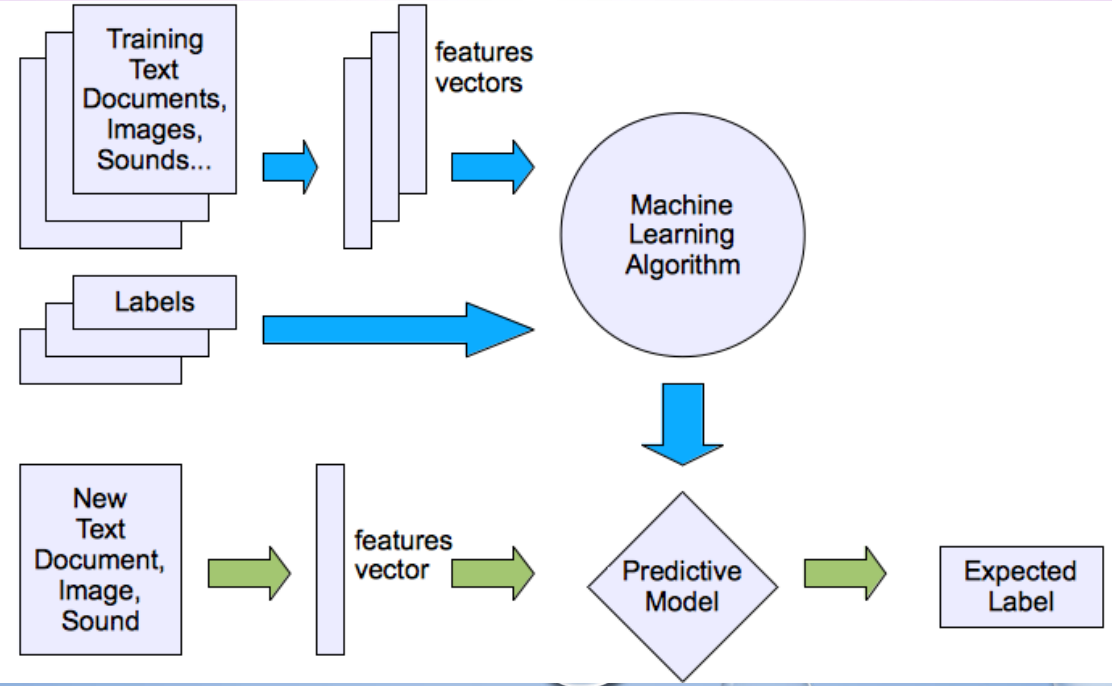
Machine Learning

- Learning paradigms
  - Supervised Learning
    - The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
  - Unsupervised Learning
    - No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
  - Reinforcement Learning
    - A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Supervised Machine Learning

- Classification-การจำแนกประเภท ที่มีลักษณะเป็น discrete data
  - Dog-Cat classification
  - Optical Character Recognition (OCR)
  - Face Recognition
- Prediction/Forecasting
  - Weather forecasting
  - Stock forecasting
  - Cost and Profit forecasting

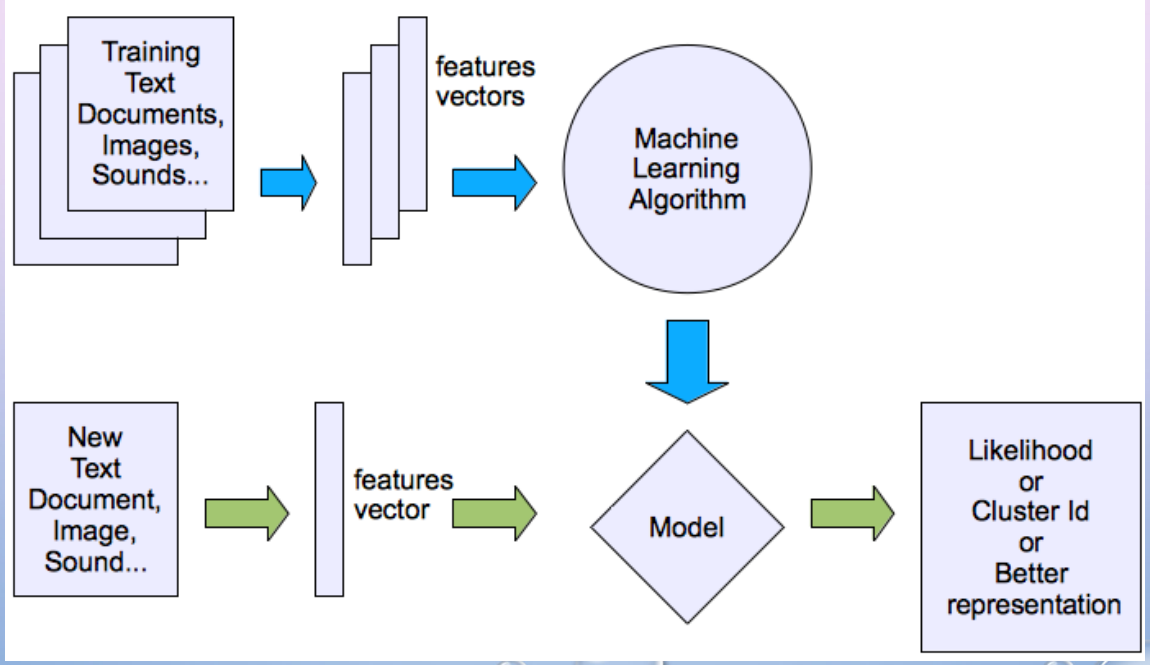
Supervised Machine Learning



Supervised Learning Techniques

- **Linear classifier** (numerical functions)
  - Perceptron, Logistic regression, Support vector machine (SVM), Multilayer perceptron (MLP)
- **Parametric** (Probabilistic functions)
  - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
- **Non-parametric** (Instance-based functions)
  - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
- **Non-metric** (Symbolic functions)
  - Classification and regression tree (CART), decision tree
- ***Aggregation***
  - Bagging (bootstrap + aggregation), Adaboost, Random forest

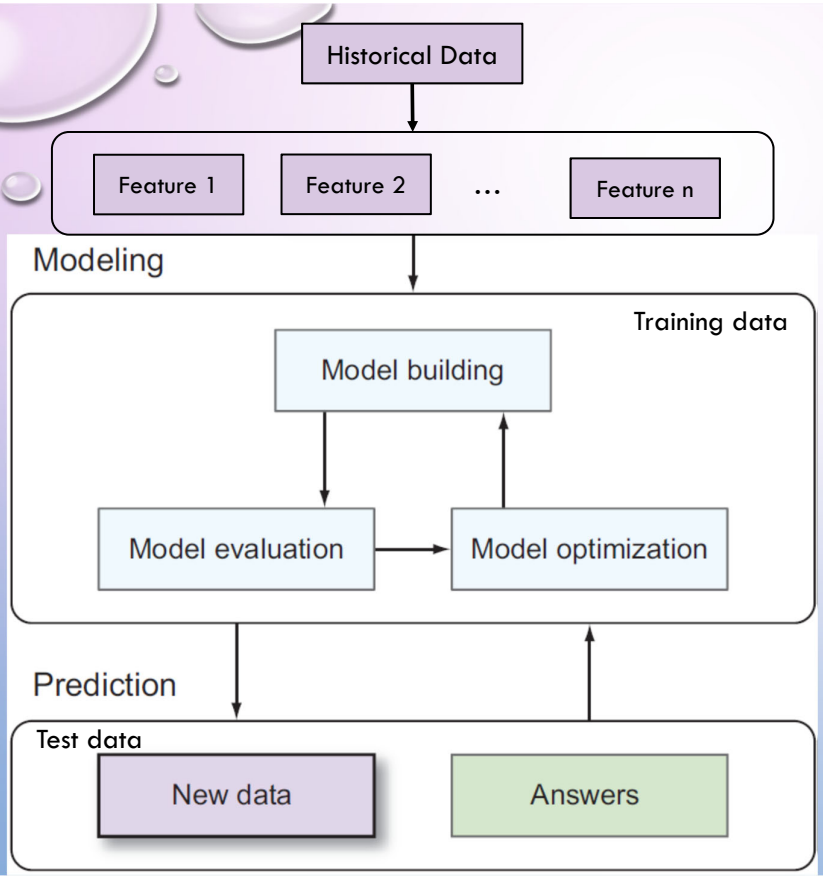
Unsupervised Learning



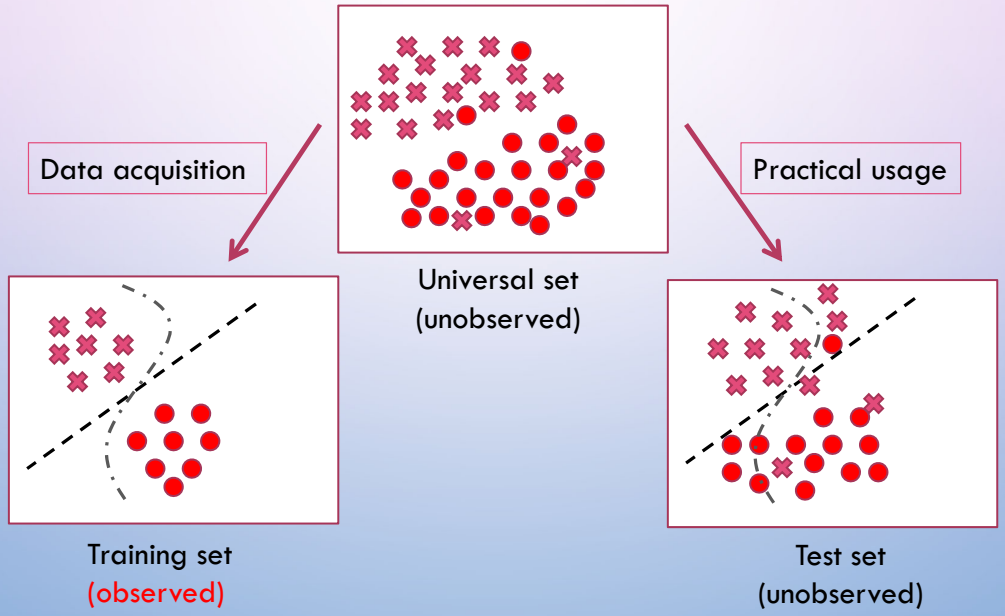
Unsupervised Learning Techniques

- **Clustering**
  - K-means clustering
  - Spectral clustering
- **Density Estimation**
  - Gaussian mixture model (GMM)
  - Graphical models
- **Dimensionality reduction**
  - Principal component analysis (PCA)
  - Factor analysis

ML Workflow



Training and Test Set



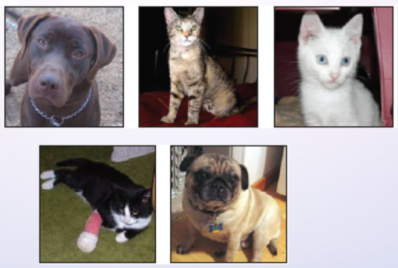
Dog-Cat Classification

Labeled training data

Machine Learning Process

Model

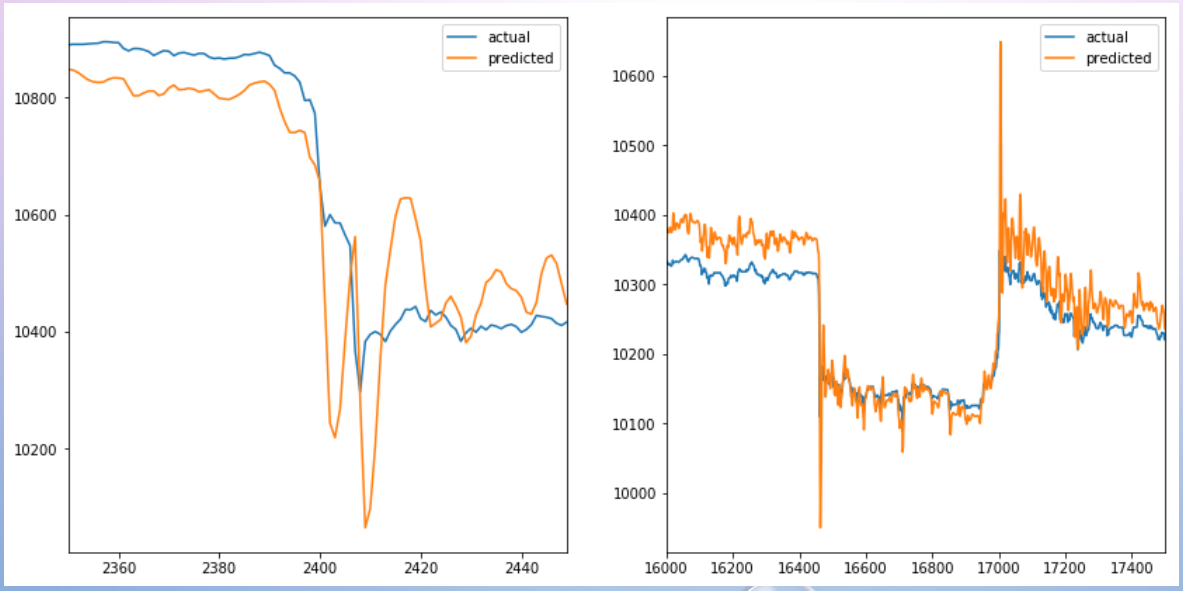
Unlabeled test data



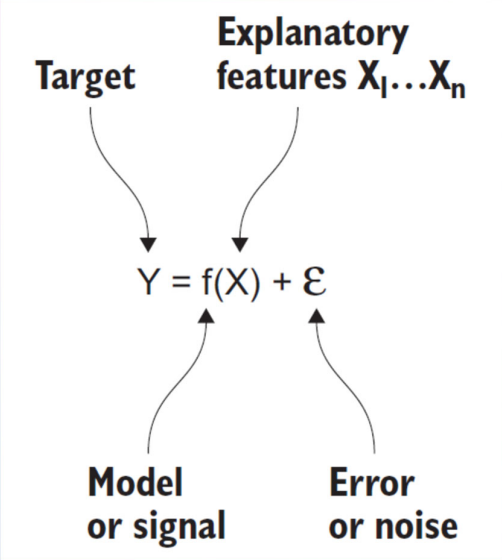
Prediction Process

Results

Regression Example



Model Generation and Prediction

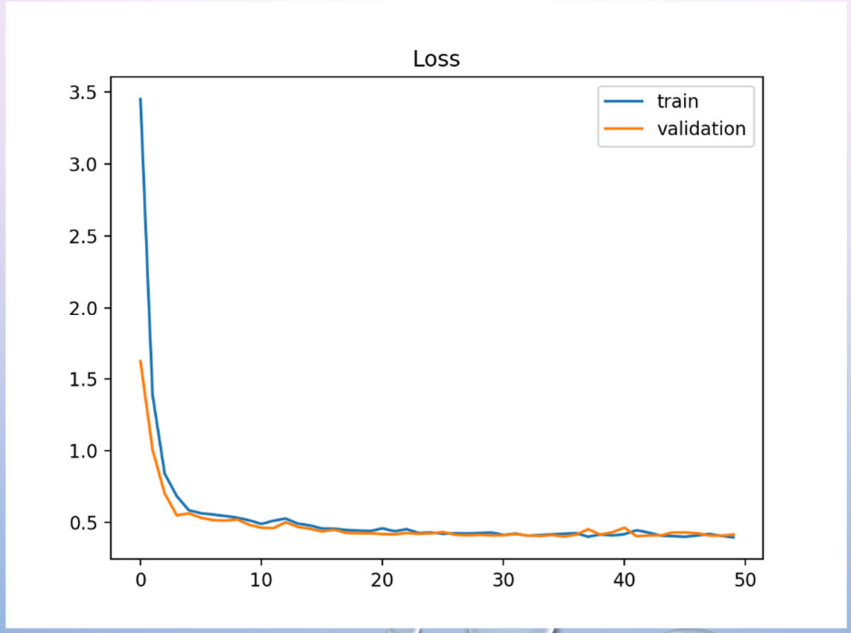


Model Generation

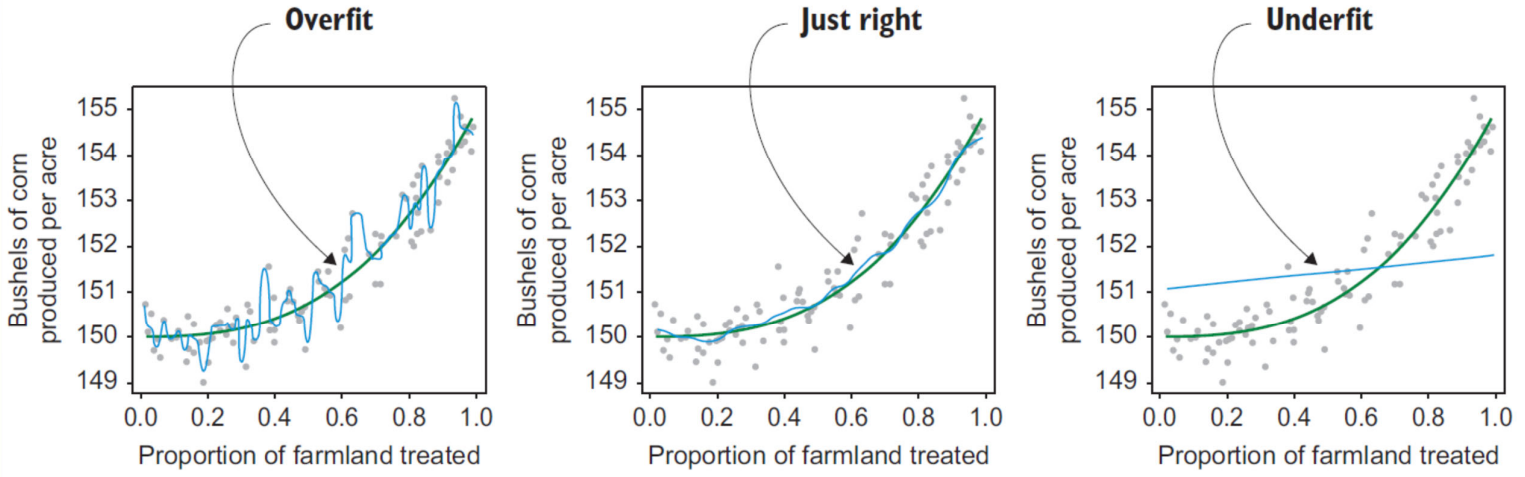
$$Y_{pred} = f_{est}(X_{new})$$

Prediction

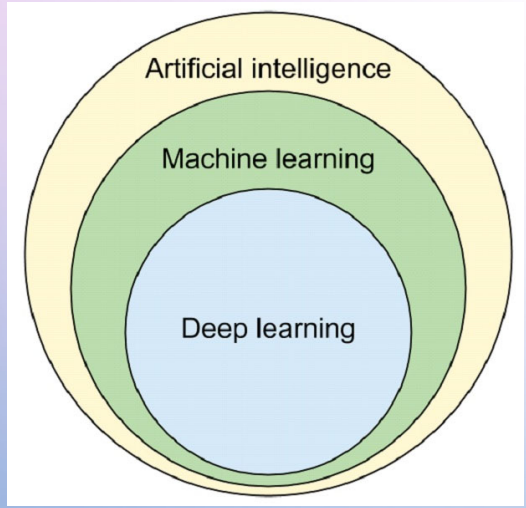
Model Evaluation and Optimization

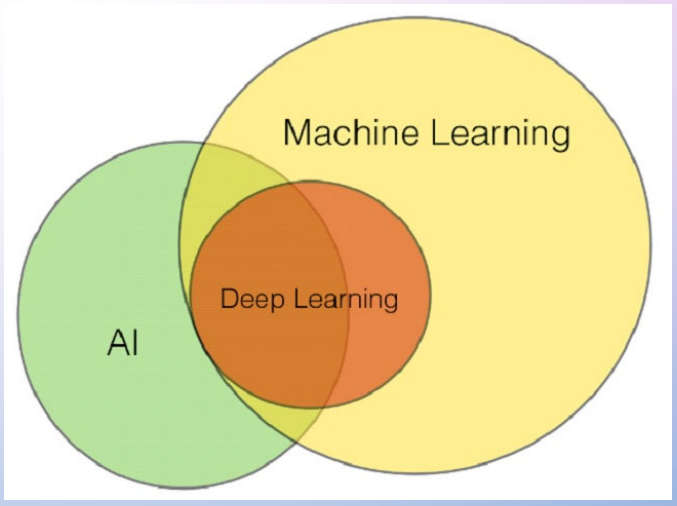


Model Evaluation and Optimization

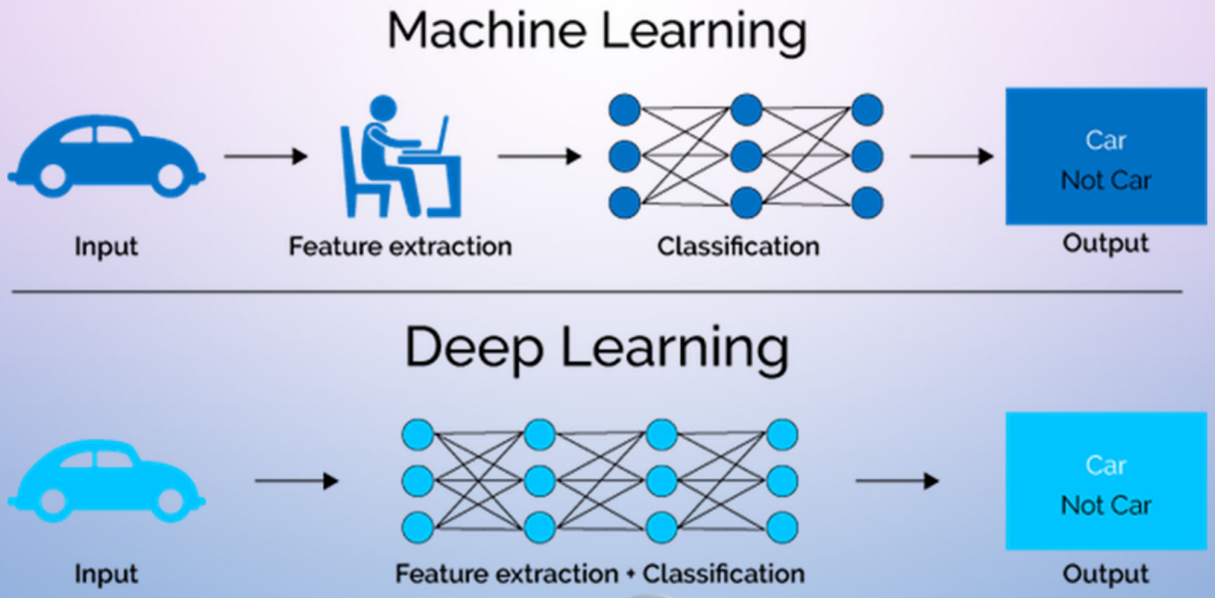


Deep Learning



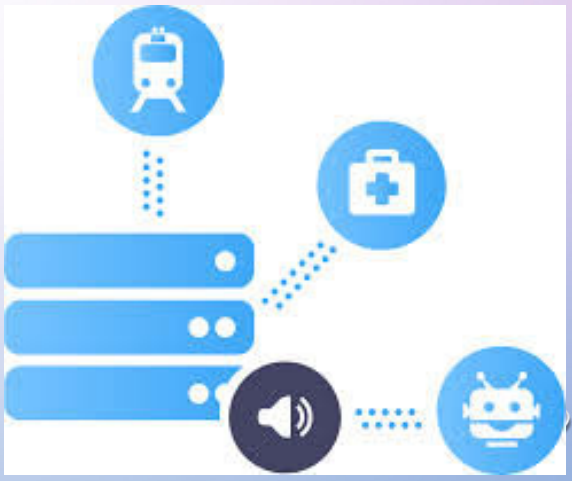


ML vs DL

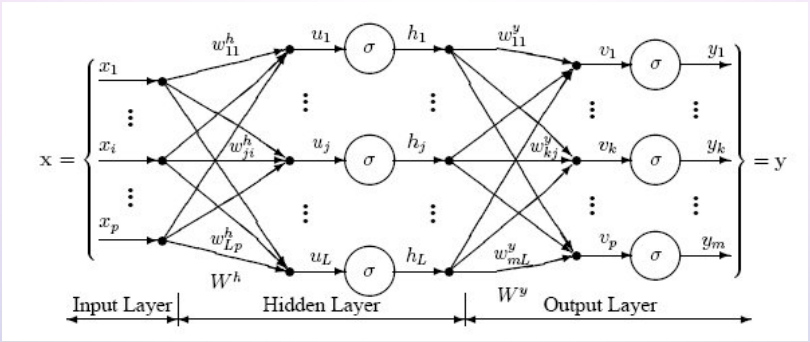


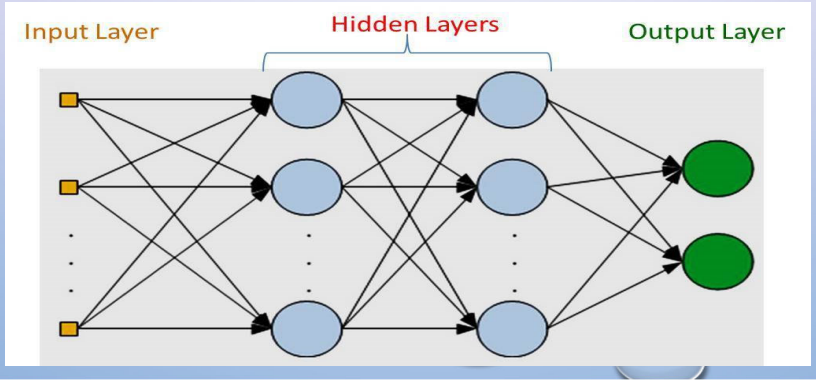
So, What do you think, what is Deep Learning

Deep learning is a machine learning technique that learns features and task directly from the data, where data may by images, text or sound!

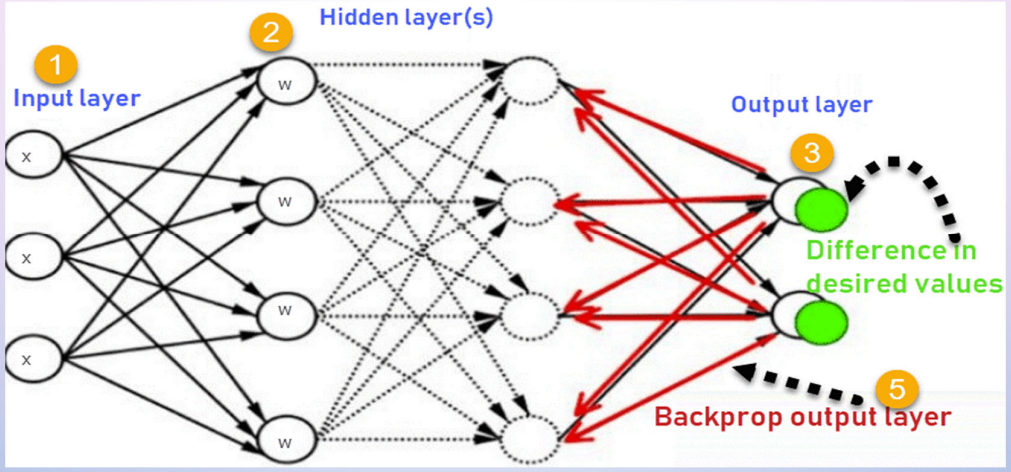


Multi-layer Perceptron (MLP)





MLP - Backpropagation



Mean square error loss

$$E = (t - y)^2$$

Learning Approaches

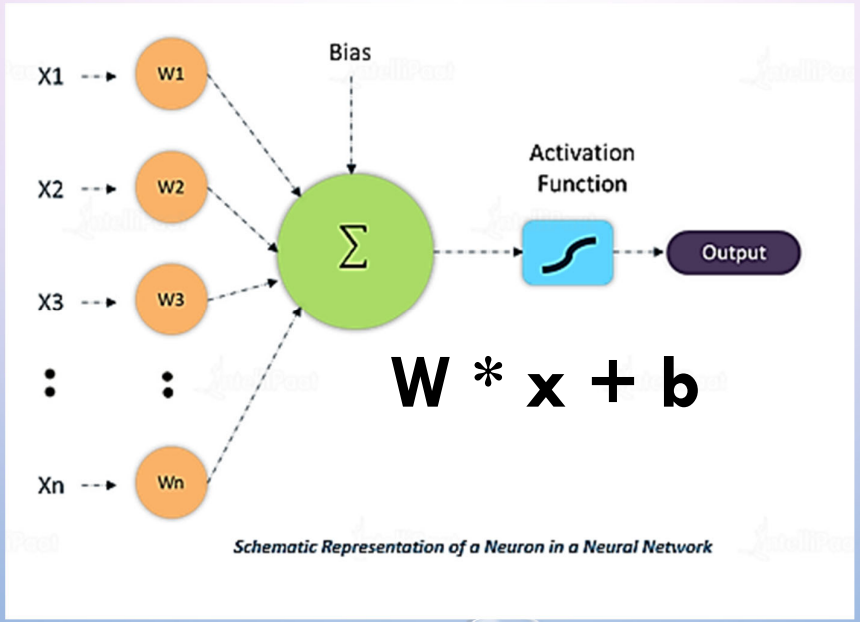
- Hebbian rules: cells that fire together wire together
- Competitive learning: increasing specialization of cells
- Hopfield networks: associative memory
- Error correction: backpropagation that can reduce cost function

output = sum (weights * inputs) + bias

- The output is calculated by multiplying the inputs with their weights and then passing it through an activation function like the Sigmoid function, etc. Here, bias acts like a constant which helps the model to fit the given data. The steepness of the Sigmoid depends on the weight of the inputs.
- A simpler way to understand bias is through a constant c of a linear function

$$y = mx + c$$

What is Bias

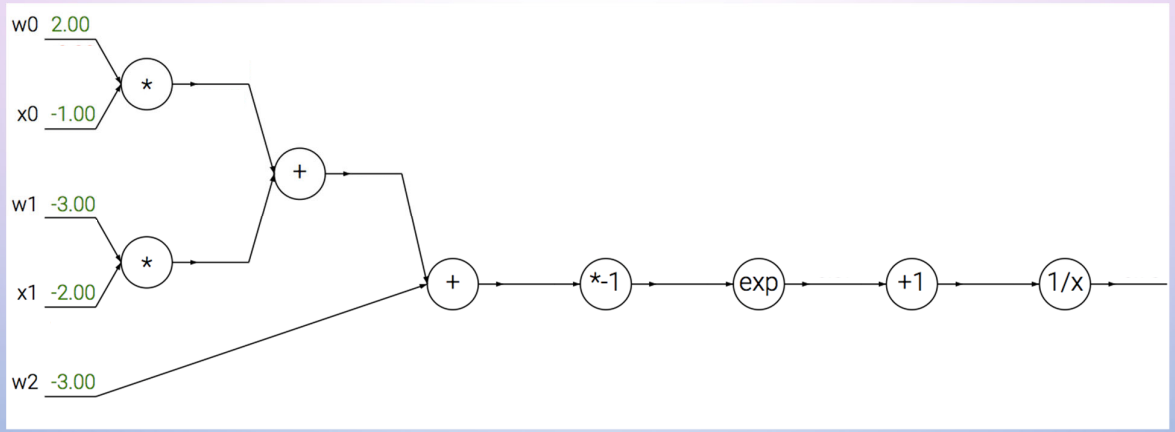


What is Gradient Descent (BGD)

- Gradient Descent is an optimization technique that is used to improve deep learning and neural network-based models by minimizing the cost function.
- Gradient Descent is a process that occurs in the backpropagation phase where the goal is to continuously resample the gradient of the model's parameter in the opposite direction based on the weight w, updating consistently until we reach the global **minimum of function** J(w).
- More Precisely,
  - Gradient descent is an algorithm, which is used to iterate through different combinations of weights in an optimal way.....to find the best combination of weights which has a minimum error.

A detailed example

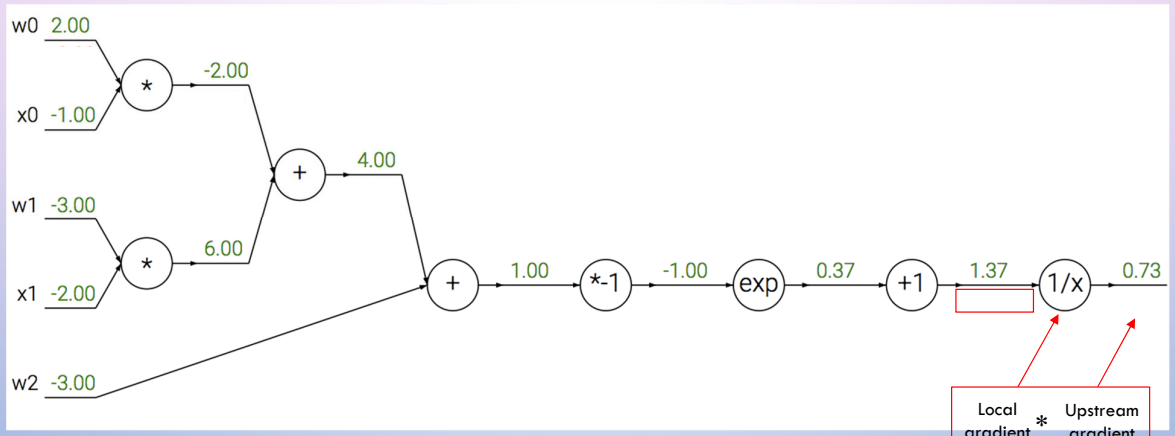
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



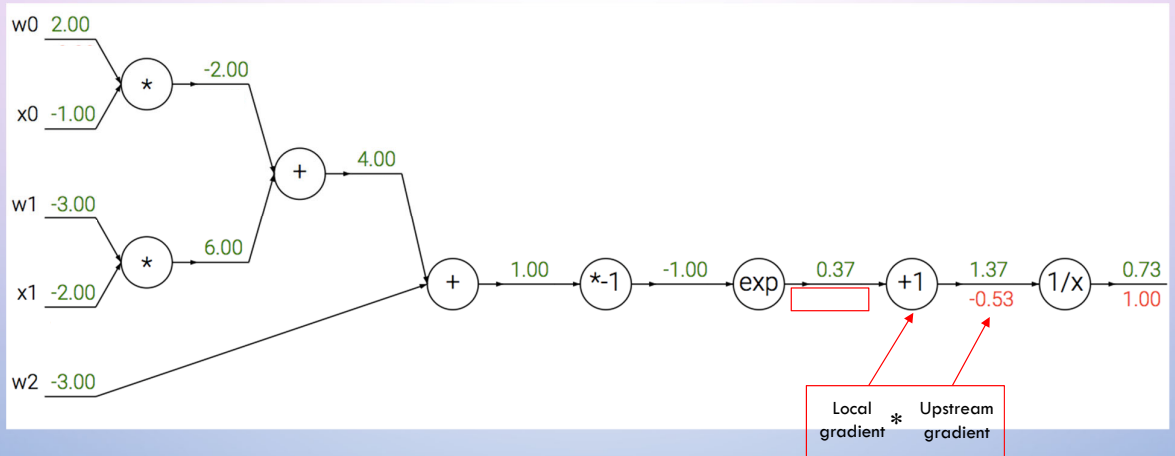
$$(1/x)' = -1/x^2$$

$$-\frac{1}{1.37^2} * 1 = -0.53$$

Source: Stanford 231n

A detailed example

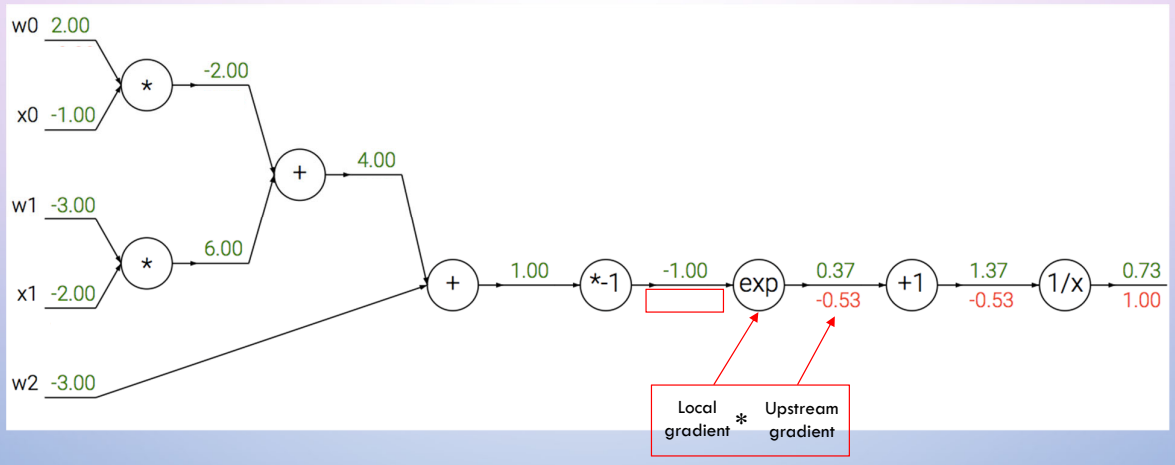
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$

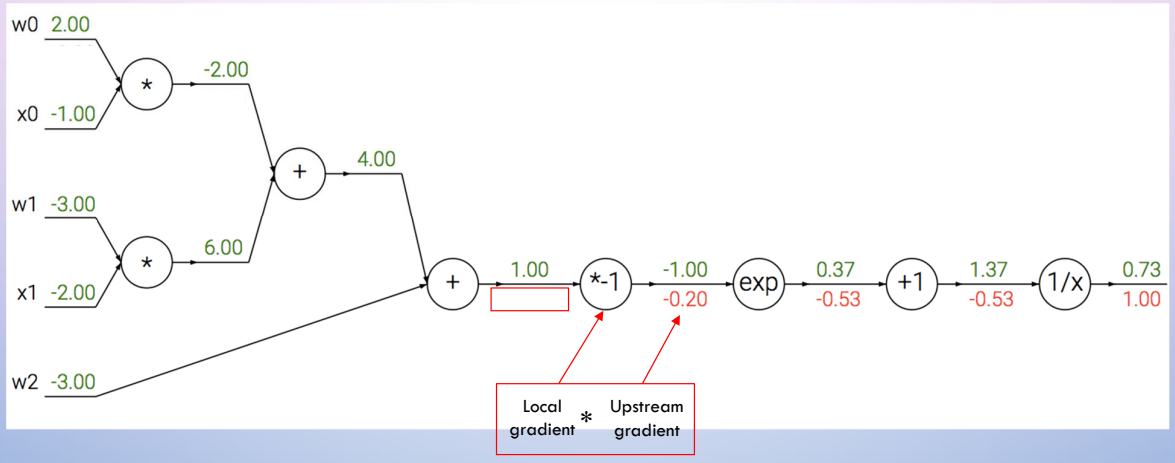


$$\exp(-1) * (-0.53) = -0.20$$

Source: Stanford 231n

A detailed example

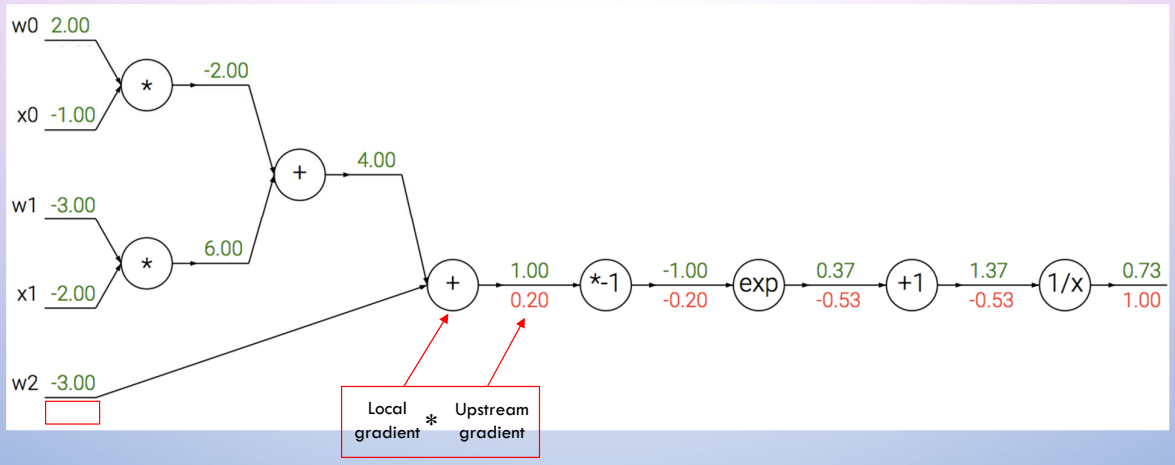
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

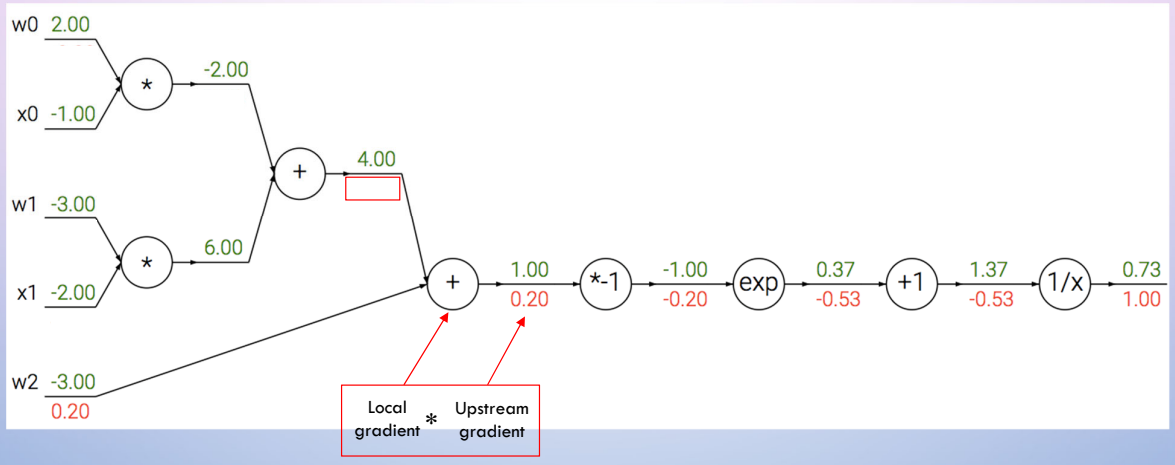
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

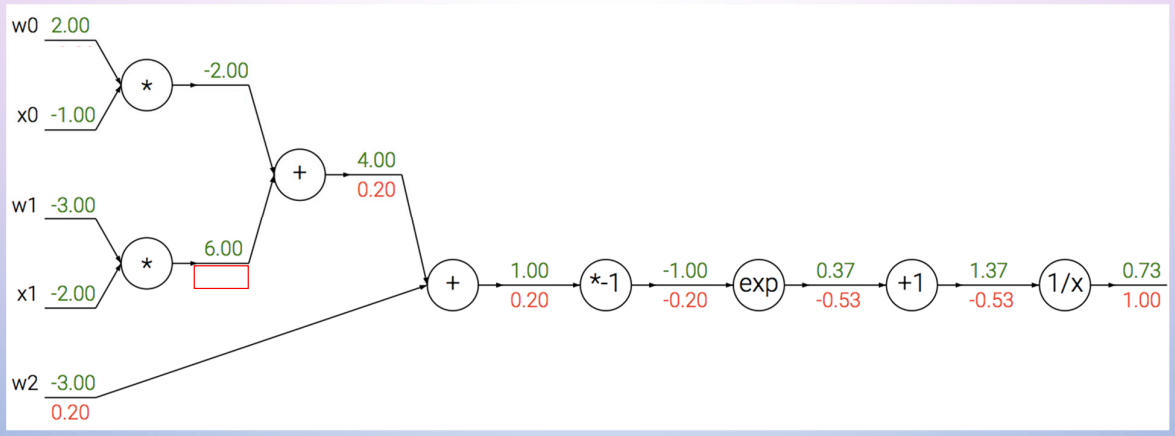
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

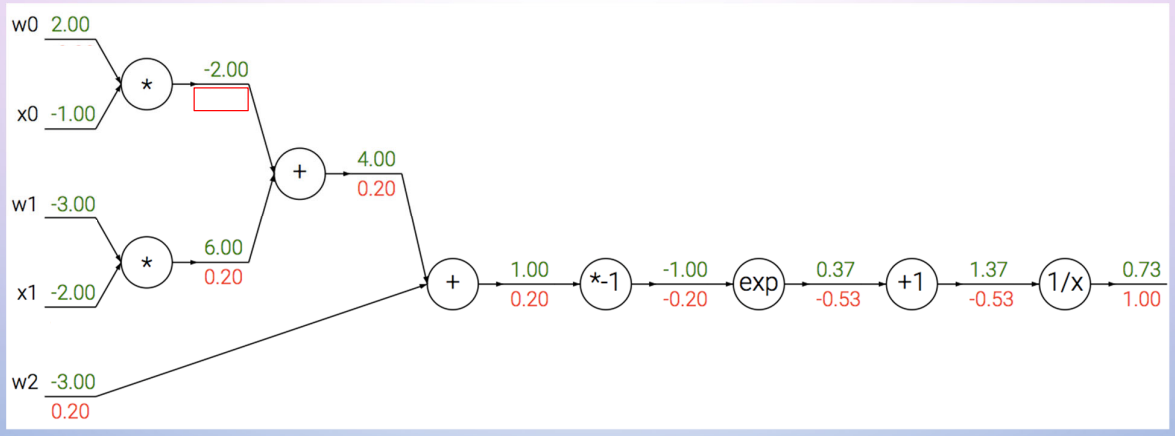
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

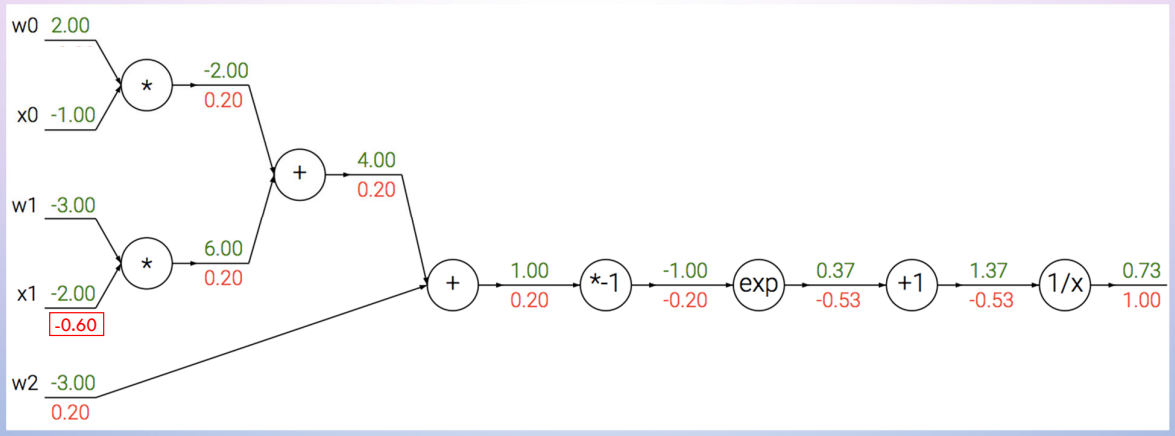
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

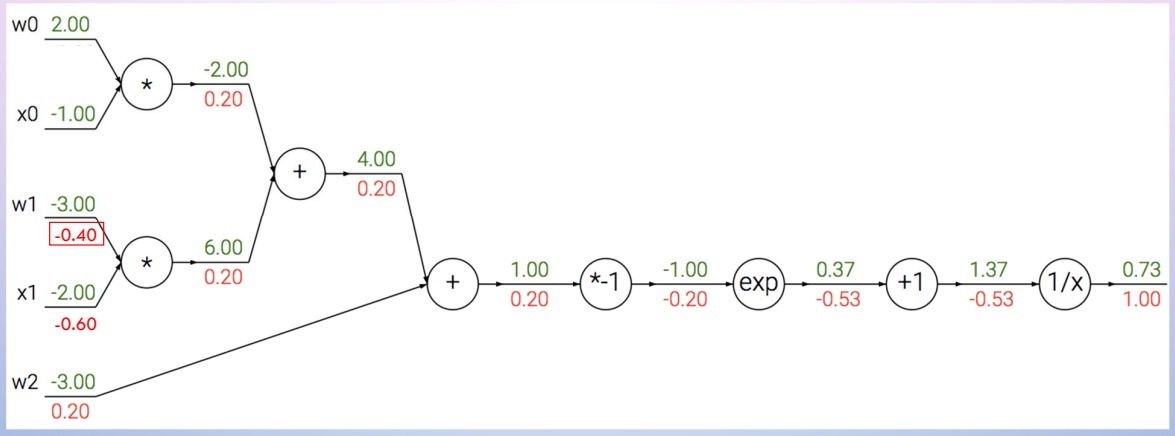
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

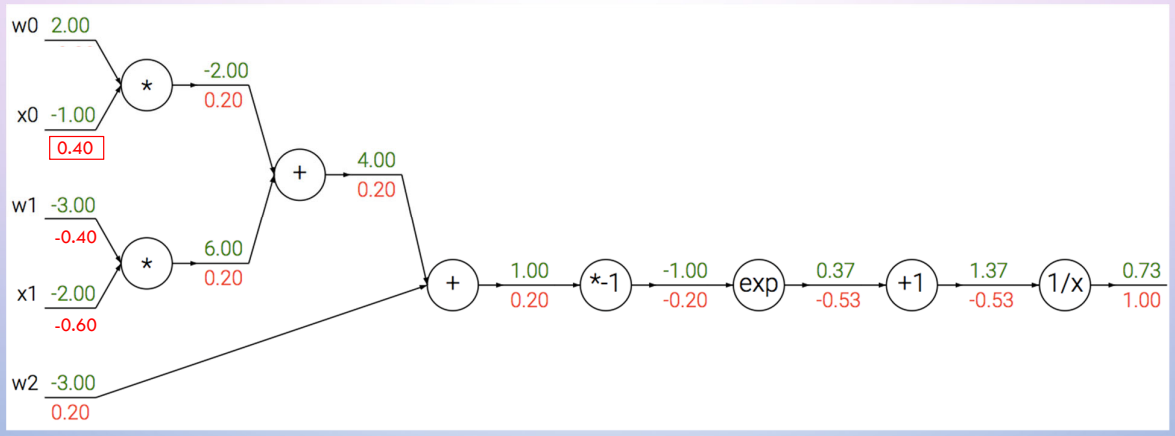
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

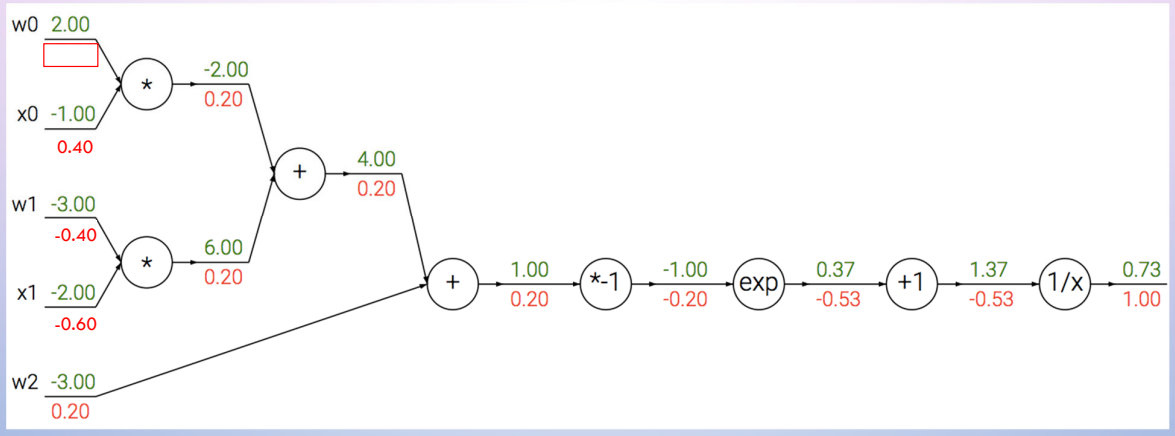
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

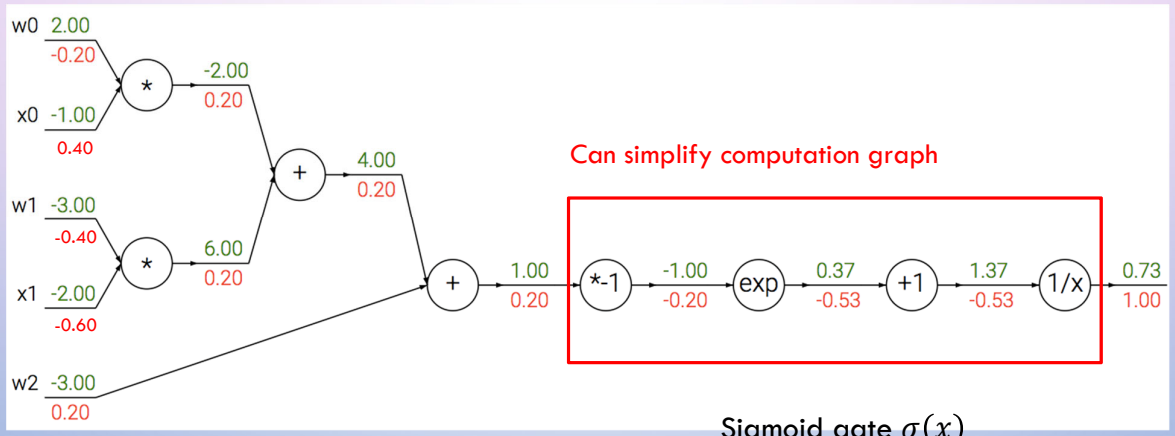
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



Source: Stanford 231n

A detailed example

$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$



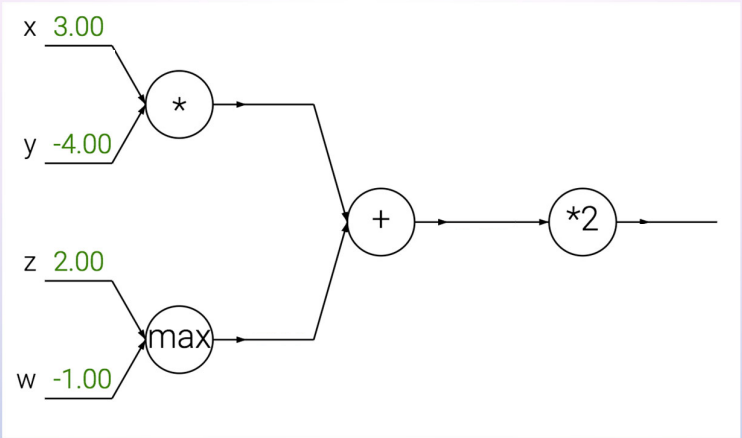
Sigmoid gate $\sigma(x)$

$$\sigma'(x) = \sigma(x)\big(1-\sigma(x)\big)$$

$$\sigma(1)\big(1-\sigma(1)\big) = 0.73 * (1-0.73) = 0.20$$

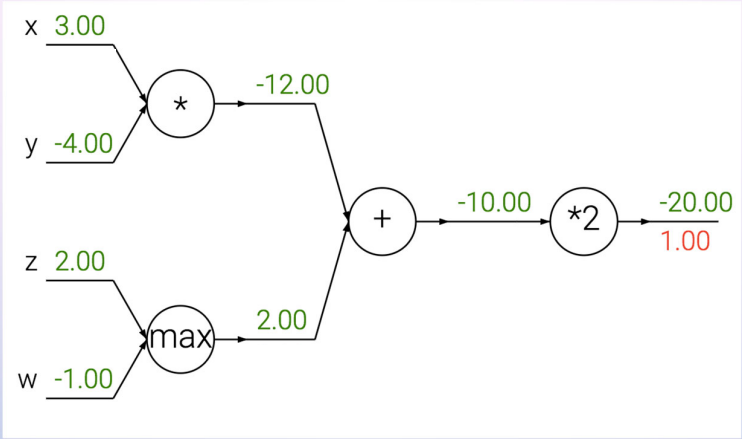
Source: Stanford 231n

Patterns in gradient flow



Source: Stanford 231n

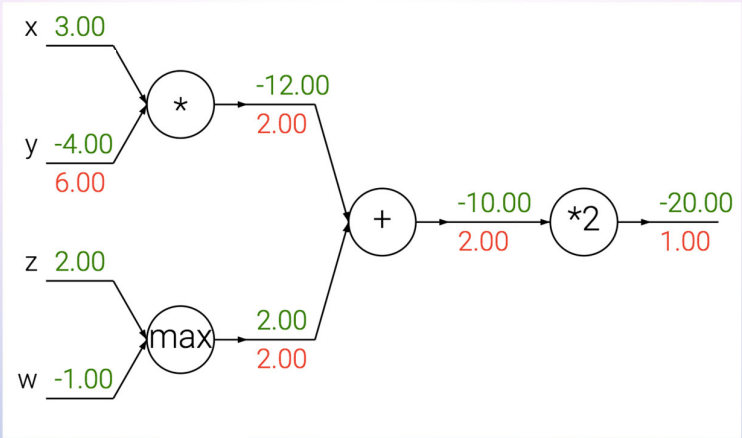
Patterns in gradient flow



Add gate: "gradient distributor"

Source: Stanford 231n

Patterns in gradient flow

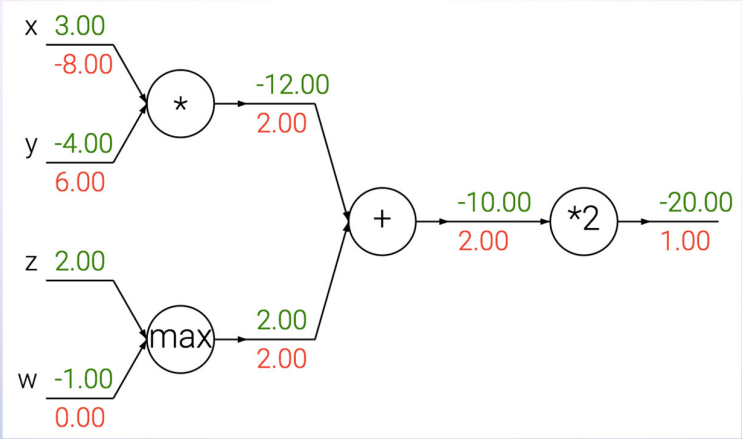


Add gate: "gradient distributor"

Multiply gate: "gradient switcher"

Source: Stanford 231n

Patterns in gradient flow



Add gate: "gradient distributor"

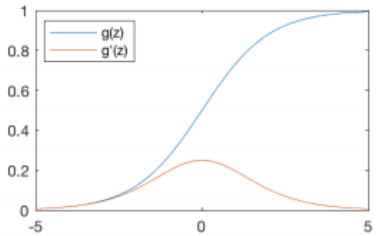
Multiply gate: "gradient switcher"

Max gate: "gradient router"

Source: Stanford 231n

Common Activation Functions

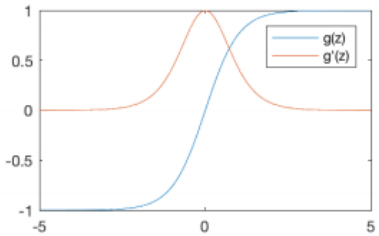
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

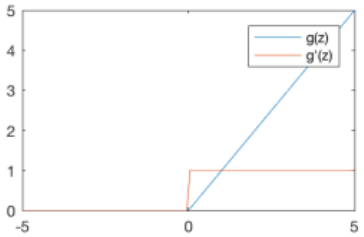
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)

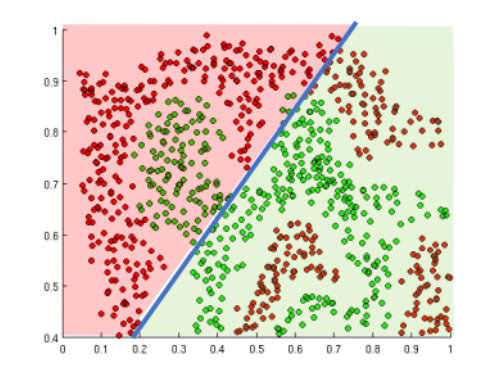


$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Importance of Activation Functions

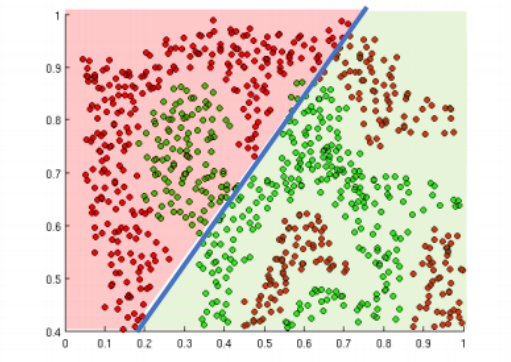
*The purpose of activation functions is to **introduce non-linearities** into the network*



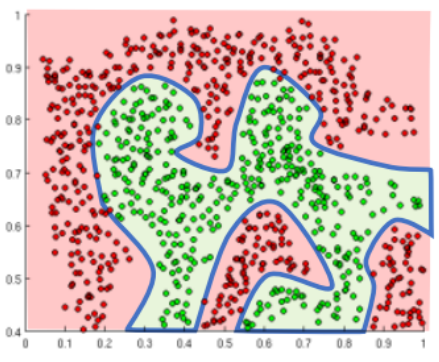
Linear Activation functions produce linear decisions no matter the network size

Importance of Activation Functions

*The purpose of activation functions is to **introduce non-linearities** into the network*

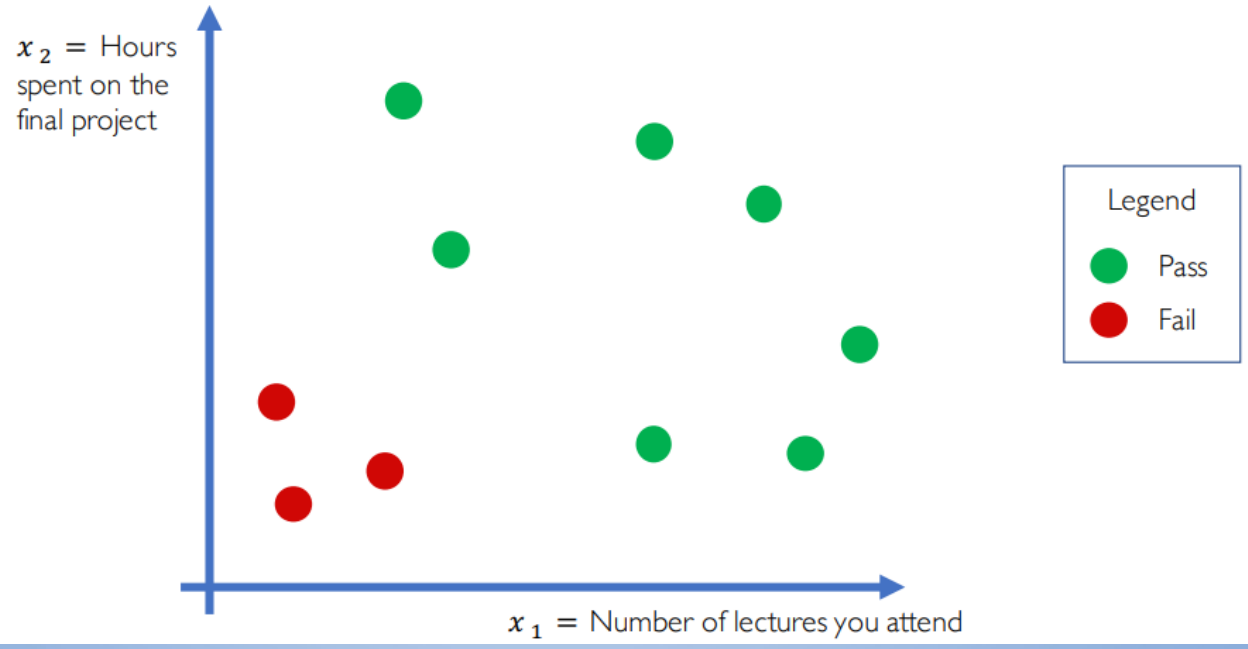


Linear Activation functions produce linear decisions no matter the network size

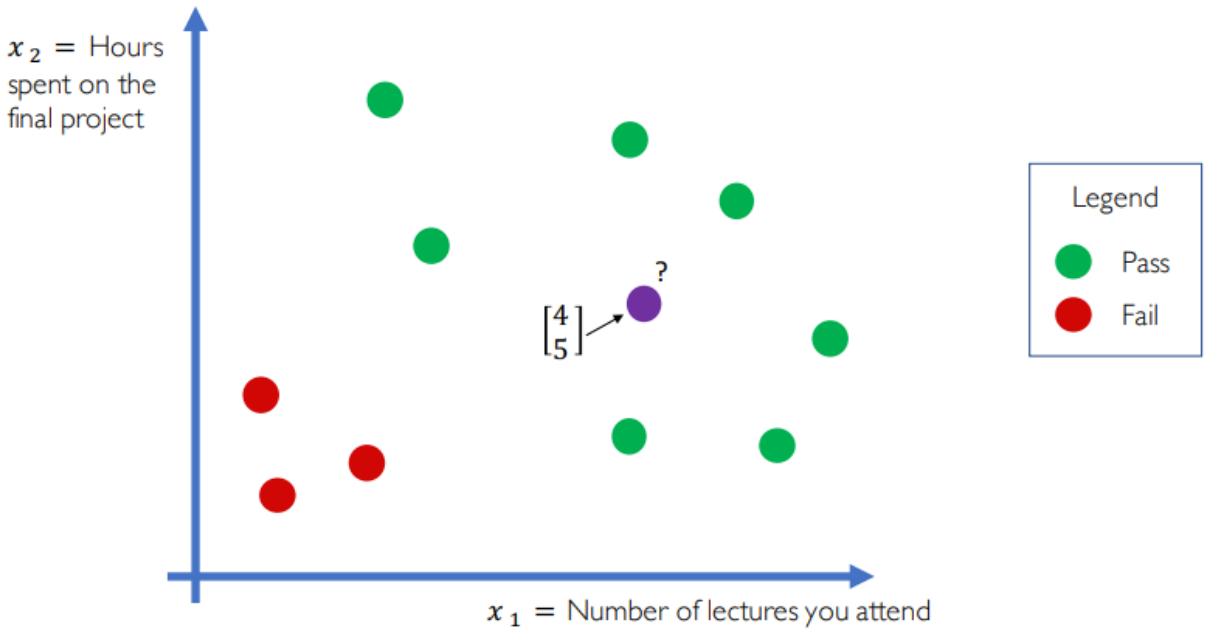


Non-linearities allow us to approximate arbitrarily complex functions

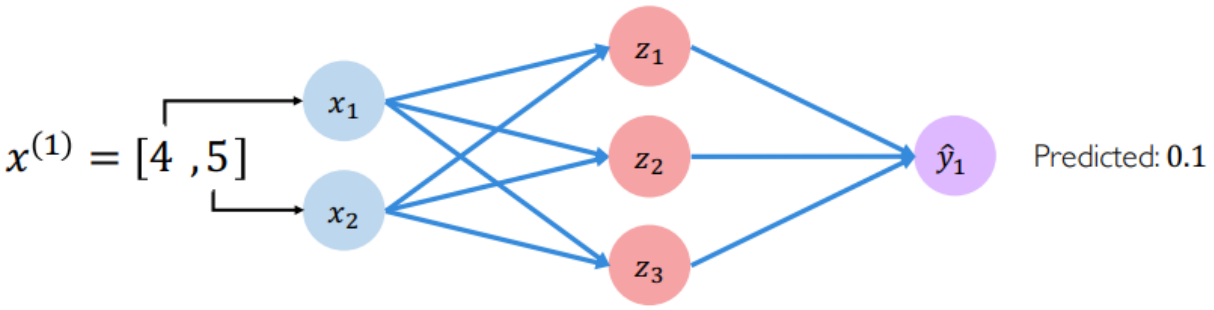
Example Problem: Will I pass this class?



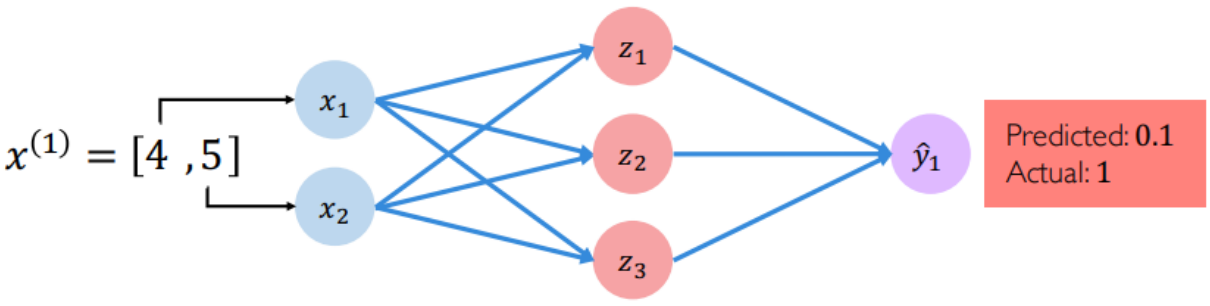
Example Problem: Will I pass this class?



Example Problem: Will I pass this class?

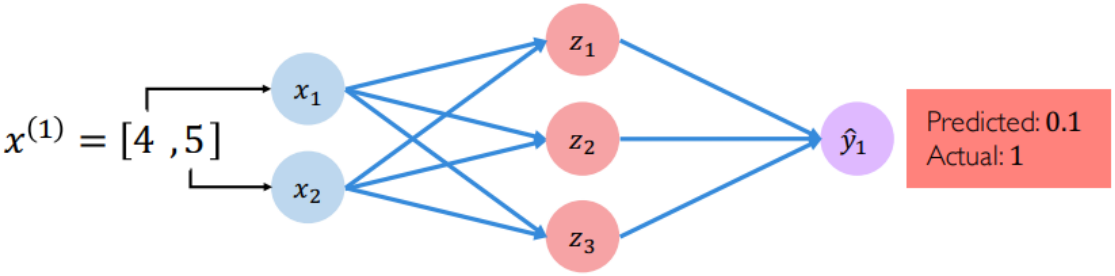


Example Problem: Will I pass this class?



Quantifying Loss

*The **loss** of our network measures the cost incurred from incorrect predictions*

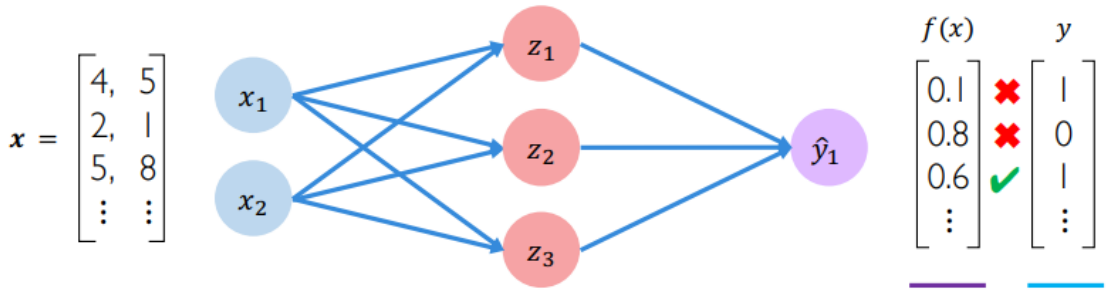


$$\mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{\theta}\right), y^{(i)}\right)$$

Predicted Actual

Empirical Loss

*The **empirical loss** measures the total loss over our entire dataset*



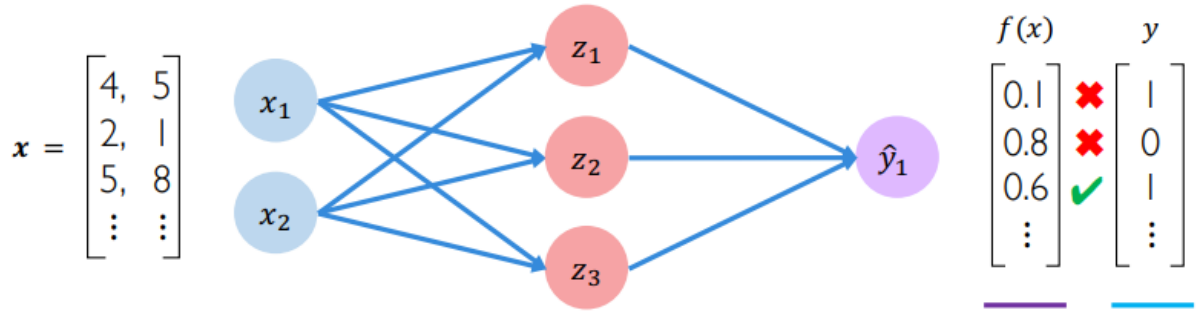
Also known as:
- Objective function
- Cost function
- Empirical Risk

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right)$$

Predicted Actual

Binary Cross Entropy Loss

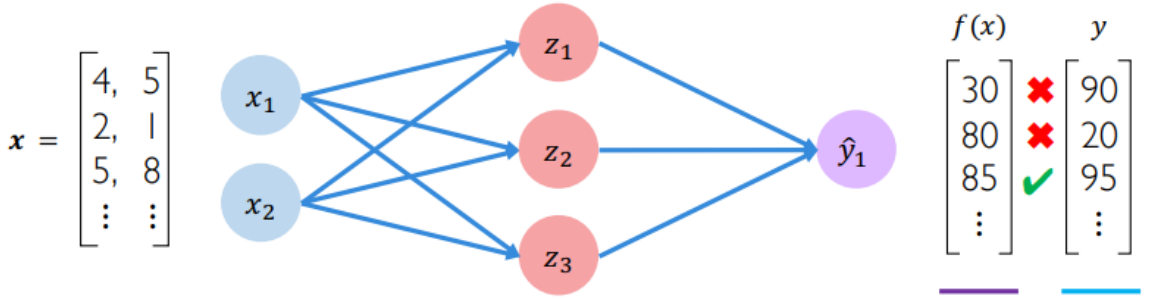
Cross entropy loss *can be used with models that output a probability between 0 and 1*



$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)};\theta\big)}_{\text{Predicted}}\Big) + (1 - \underbrace{y^{(i)}}_{\text{Actual}}) \log\Big(1 - \underbrace{f\big(x^{(i)};\theta\big)}_{\text{Predicted}}\Big)$$

Mean Squared Error Loss

Mean squared error loss *can be used with regression models that output continuous real numbers*



$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \Big(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\big(x^{(i)};\theta\big)}_{\text{Predicted}}\Big)^2$$

Training Neural Networks

Loss Optimization

We want to find the network weights that ***achieve the lowest loss***

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{\theta}\right), y^{(i)}\right)$$

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta})$$

Loss Functions Can Be Difficult to Optimize

Remember:

Optimization through gradient descent

$$\theta \leftarrow \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

How can we set the learning rate?

Why Now ?

1952 Stochastic Gradient Descent

1958 Perceptron
- Learnable Weights

1986 Backpropagation
- Multi-Layer Perceptron

1995 Deep Convolutional NN
- Digit Recognition

Neural Networks date back decades, so why the resurgence?

1. Big Data
- Larger Datasets
- Easier Collection & Storage

IMAGENET

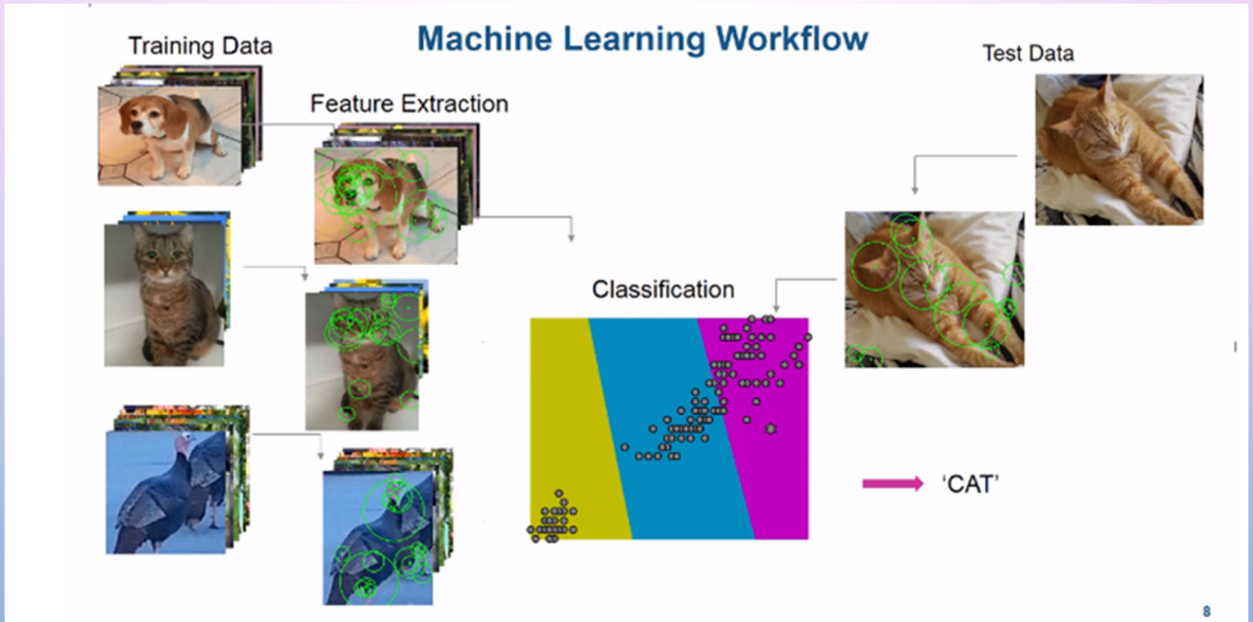
WIKIPEDIA The Free Encyclopedia

2. Hardware
- Graphics Processing Units (GPUs)
- Massively Parallelizable

3. Software
- Improved Techniques
- New Models
- Toolboxes

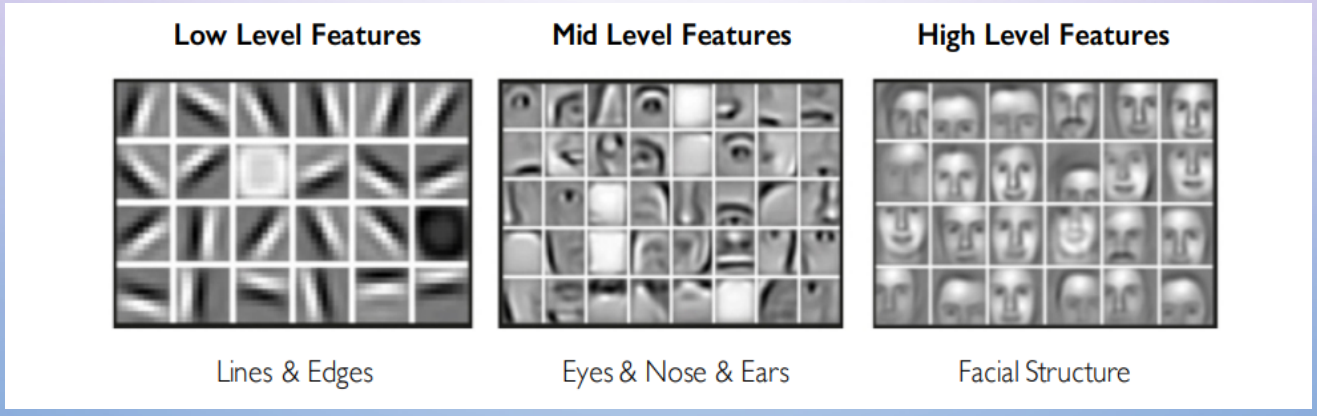
TensorFlow

Until Now…

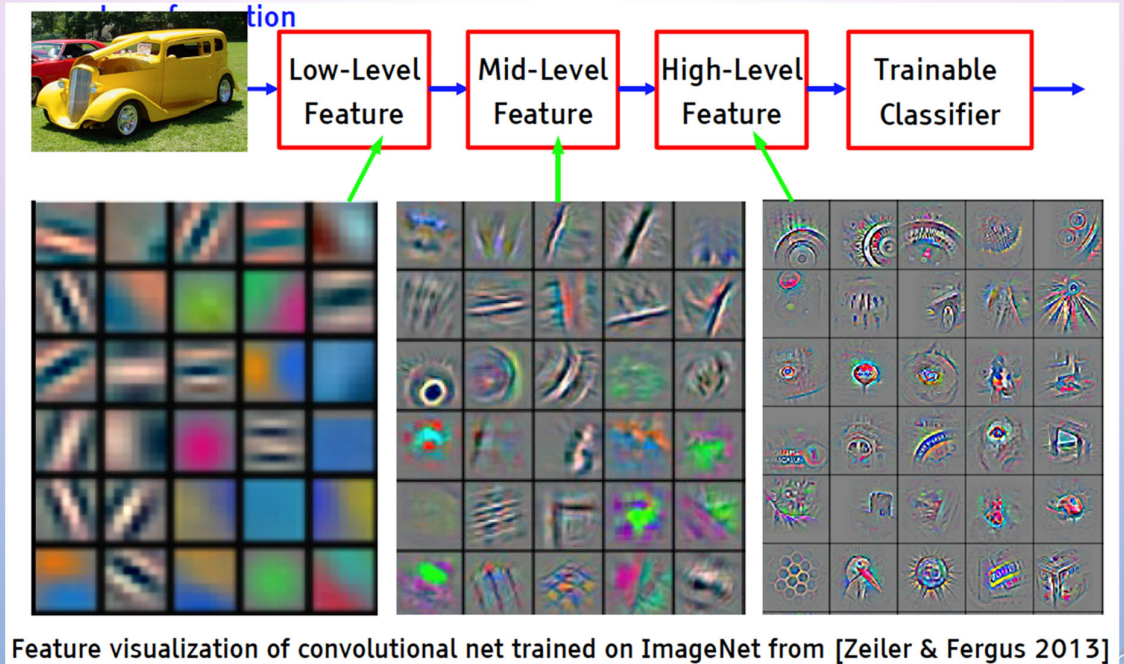


Deep Learning = Learning Hierarchical Representations

- Hand engineered feature are time consuming, brittle and not scalable in practice.
- Can we learn the underlying features directly from data?

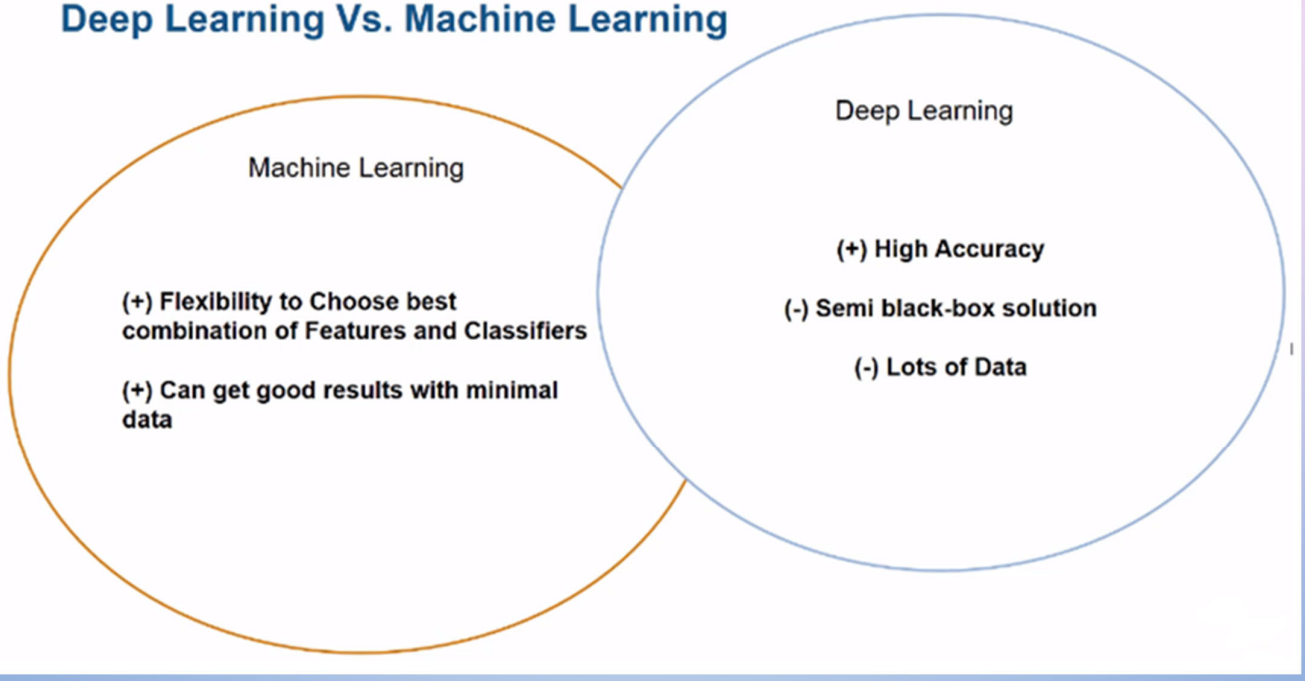


Deep Learning = Learning Hierarchical Representations



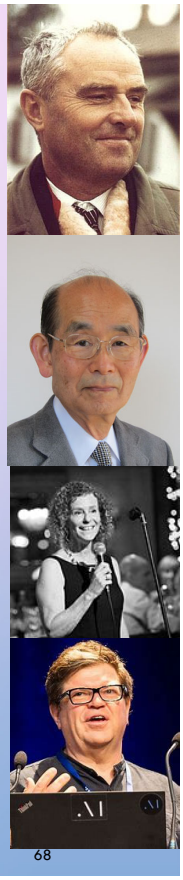
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Deep Learning Vs. Machine Learning



History of Deep Learning

- In 1971, Alexey Ivakhnenko (Father of Deep Learning) was a Soviet and Ukrainian mathematician most famous for developing the Group Method of Data Handling (GMDH) with eight layers of a deep network.
- In 1980, other deep learning working architectures, specifically those built for computer vision, began with the Neocognitron introduced by Kunihiko Fukushima.
- In 1986, the term Deep Learning was introduced to the machine learning community by Rina Dechter.
- In 1989, Yann LeCun et al. applied the standard backpropagation algorithm, which had been around as the reverse mode of automatic differentiation since 1970, to a deep neural network with the purpose of recognizing handwritten ZIP codes on mail. While the algorithm worked, training required 3 days.

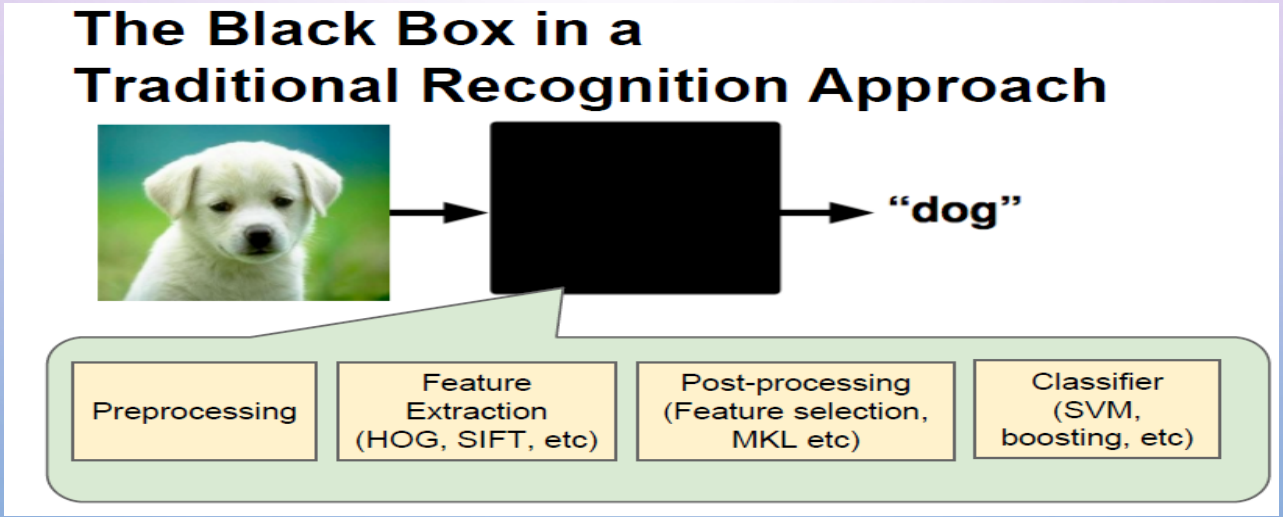


Types of Deep Learning

- Unsupervised learning
  - Deep Belief Networks (DBN), which are based on Restricted Boltzmann Machines (RBM)
  - Deep Autoencoders, which are based on Autoencoders.
- Supervised learning
  - Convolutional Neural Networks (CNNs), which are suitable for computer vision
  - Recurrent Neural Networks (RNNs), which are suitable for sequence data.

Convolutional Neural Networks (CNNs)

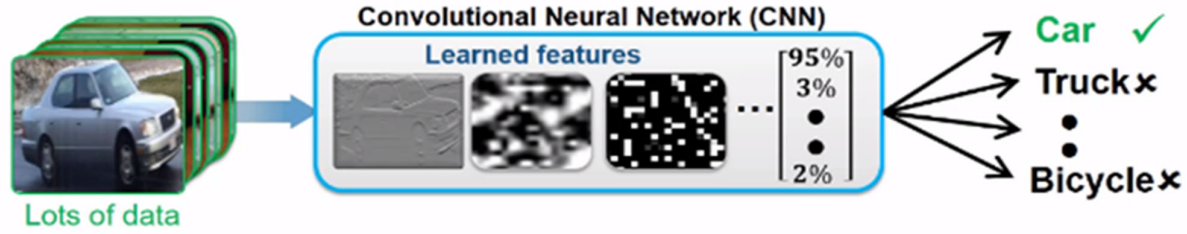
- A supervised deep learning method.
- ConvNets



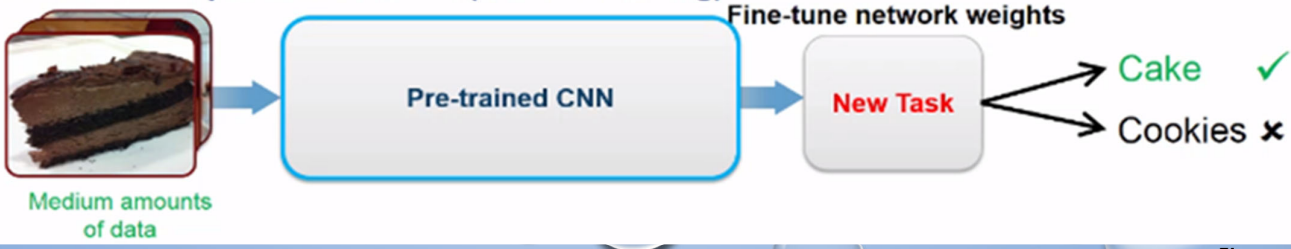
ConvNets

Approaches for Deep Learning

1. Train a Deep Neural Network from Scratch



2. Fine-tune a pre-trained model (transfer learning)



ConvNets

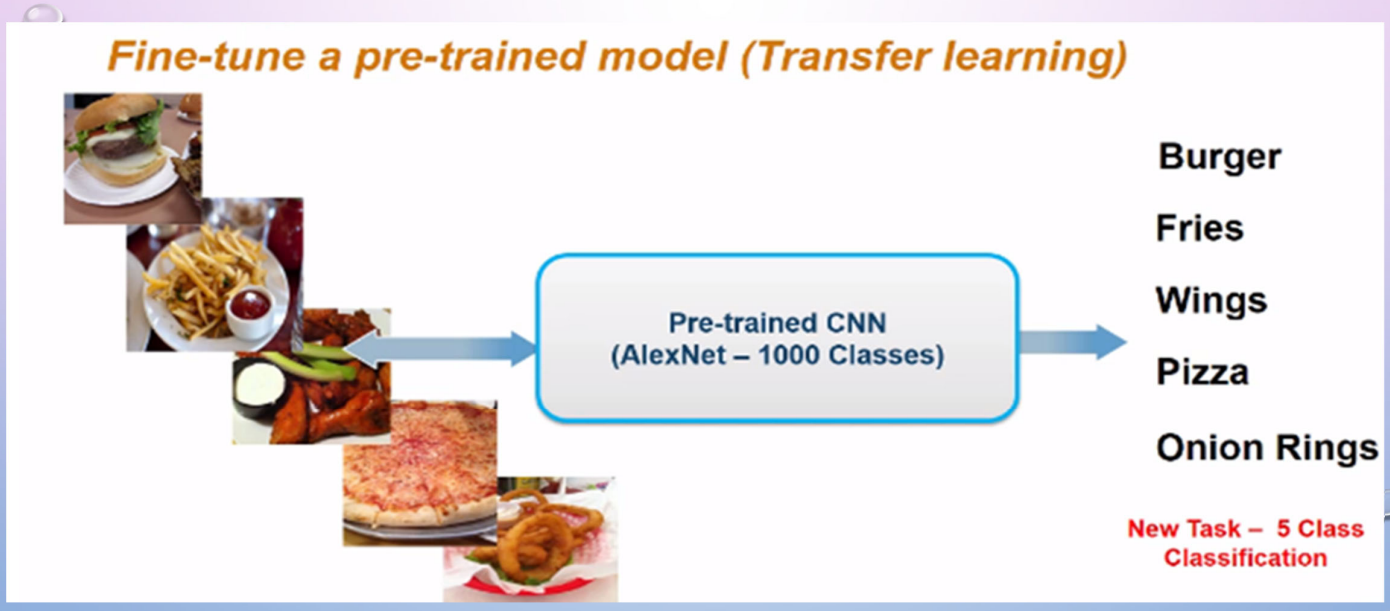
Deep Learning Approaches

Approach 1: Train a Deep Neural Network from Scratch

Recommended only when:

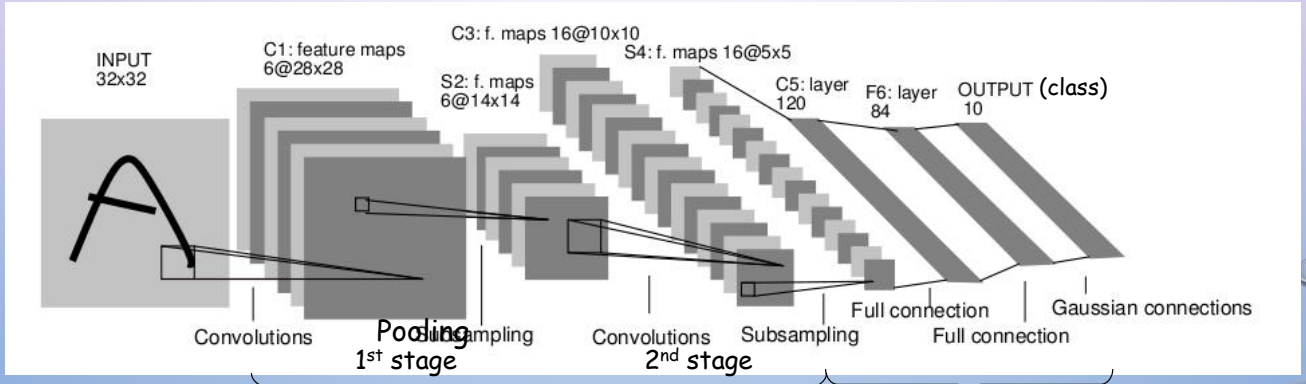
Training data	1000s to millions of labeled images
Computation	Compute intensive (requires GPU)
Training Time	Days to Weeks for real problems
Model accuracy	High (can over fit to small datasets)

ConvNets

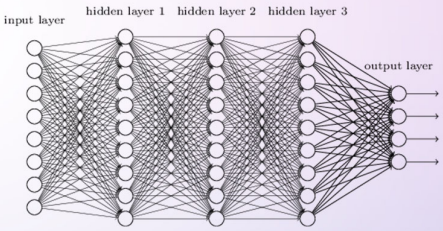


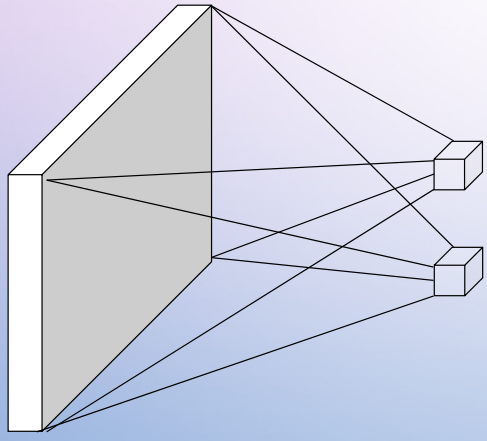
ConvNets

- LeNet-5 proposed by Lecun (1998).
- CNNs
  - Convolution layer
  - Pooling layer
  - Fully connected layer



Convolutional layer anatomy

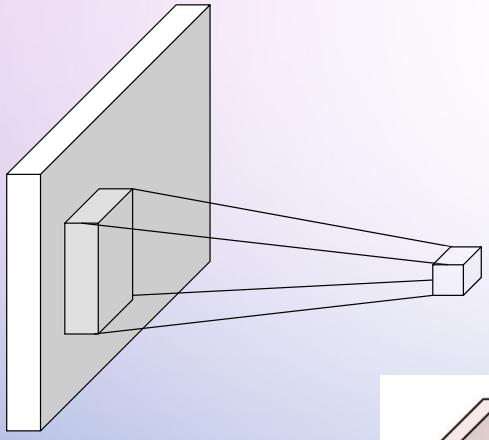




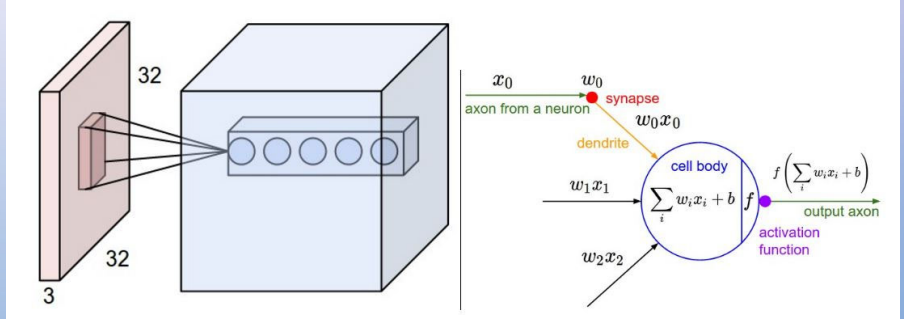
image

Fully connected layer

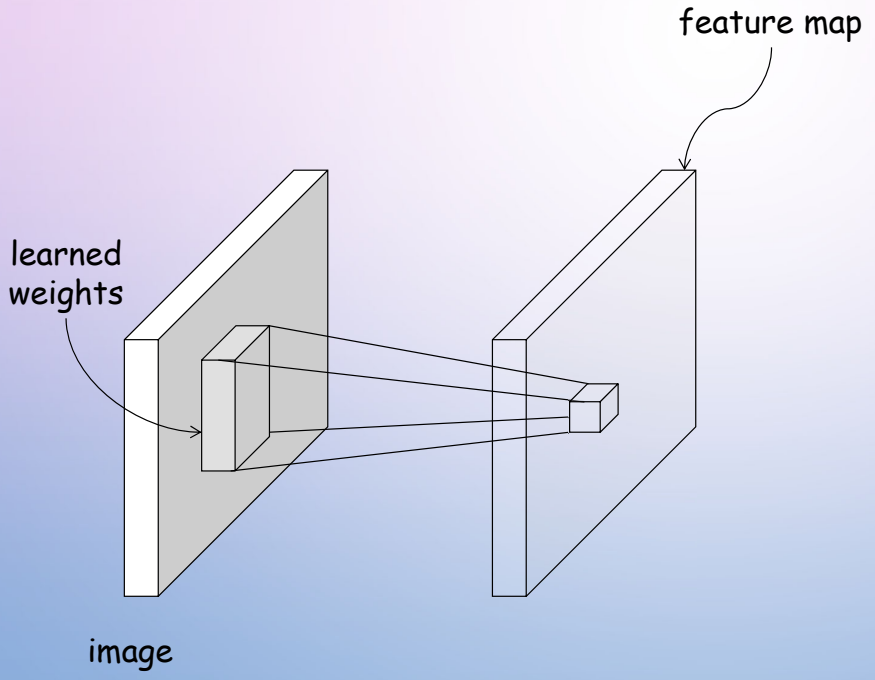
Convolutional layer anatomy



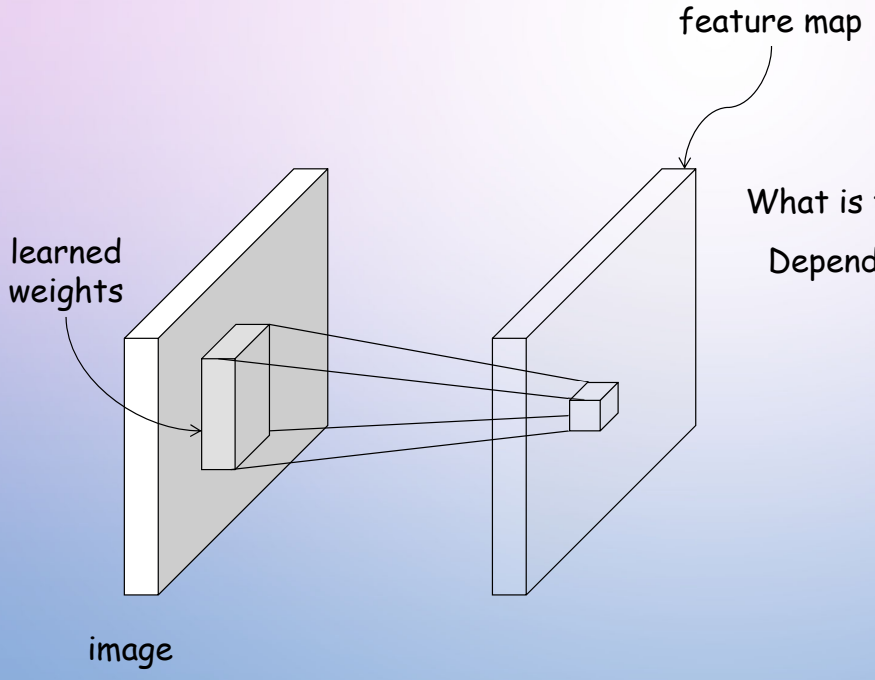
image



Convolutional layer anatomy

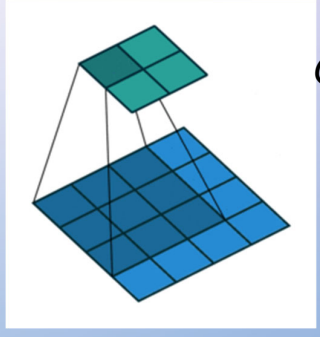


Convolutional layer anatomy



What is the feature map resolution?

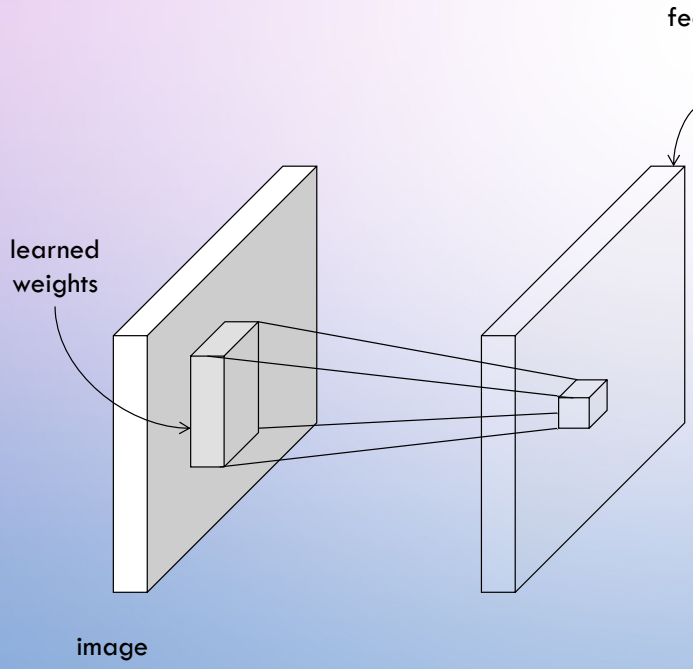
Depends on *padding* and *stride*:



No padding, stride 1

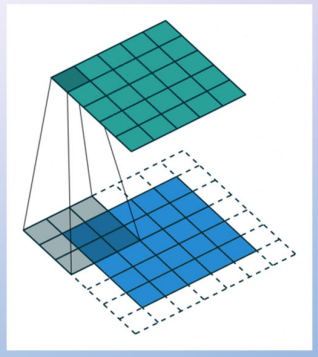
Animation source

Convolutional layer anatomy



What is the feature map resolution?

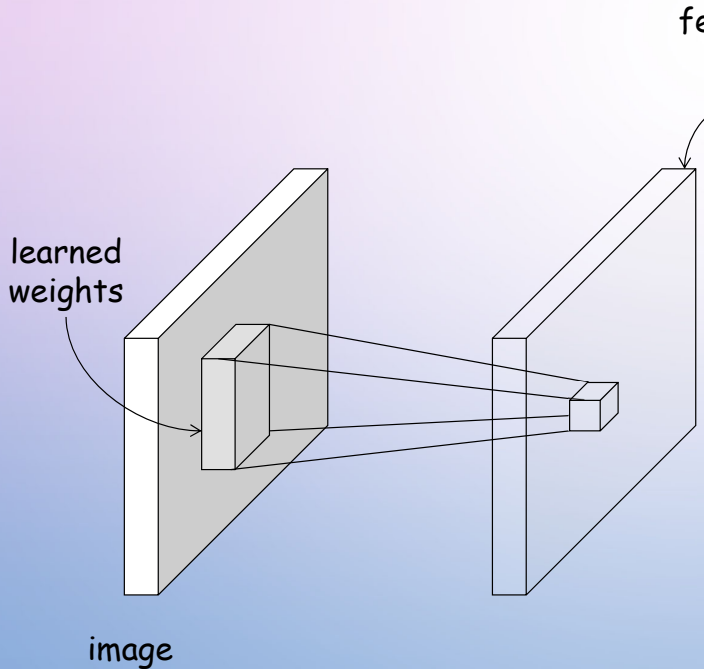
Depends on *padding* and *stride*:



With padding, stride 1

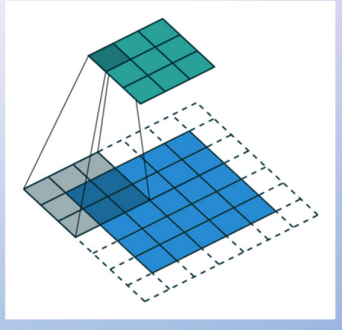
Animation source

Convolutional layer anatomy



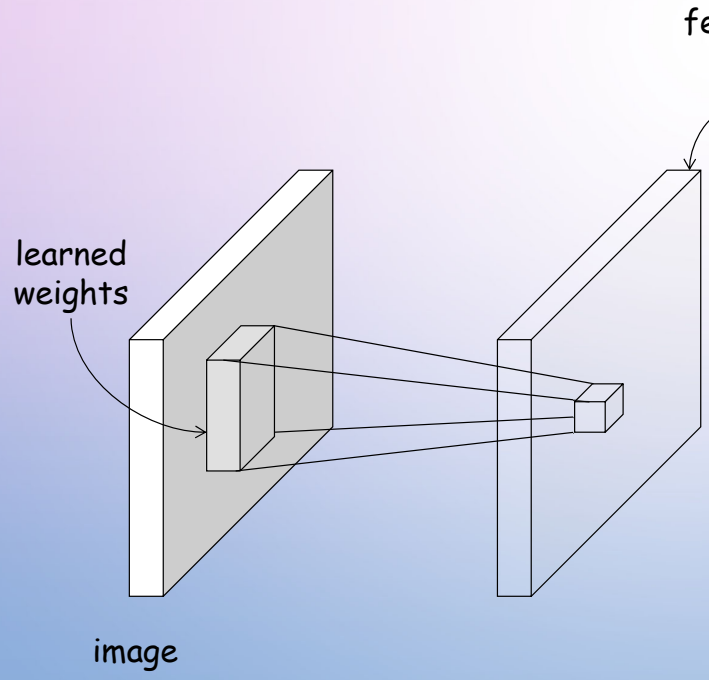
What is the feature map resolution?

Depends on *padding* and *stride*:

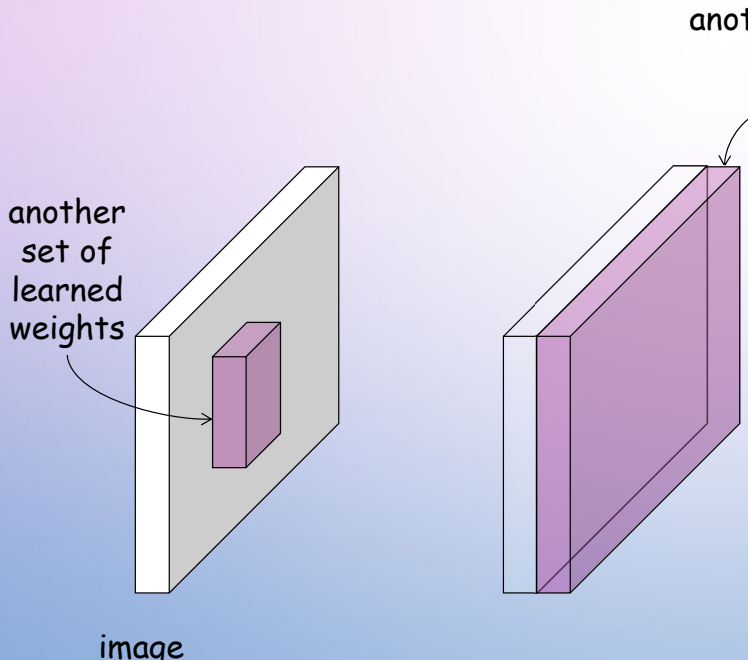


With padding, stride 2

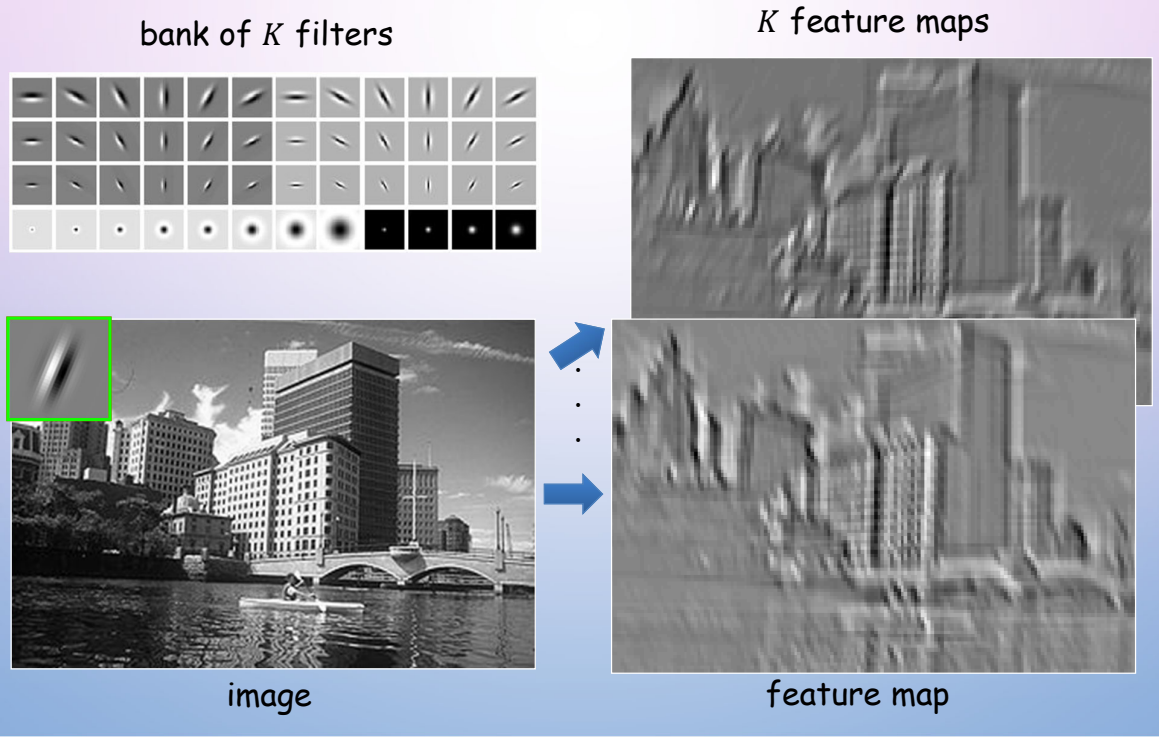
Convolutional layer anatomy



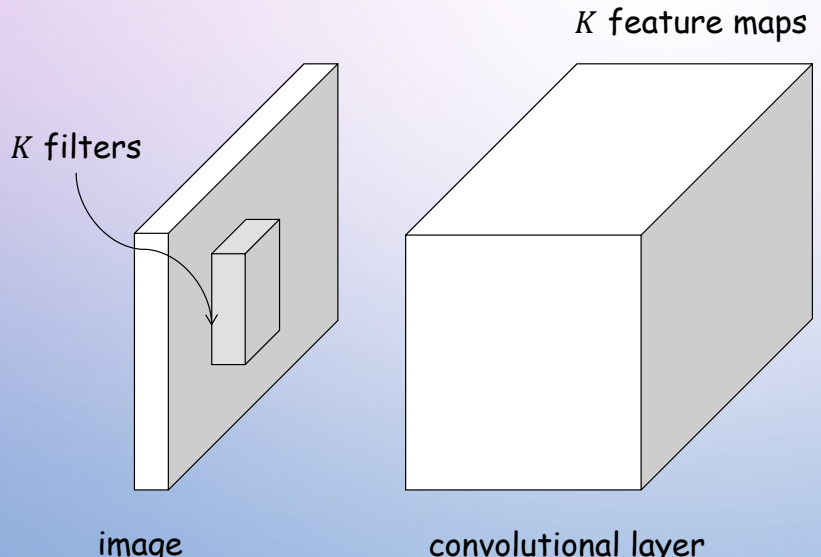
Convolutional layer anatomy



Convolution and traditional feature extraction

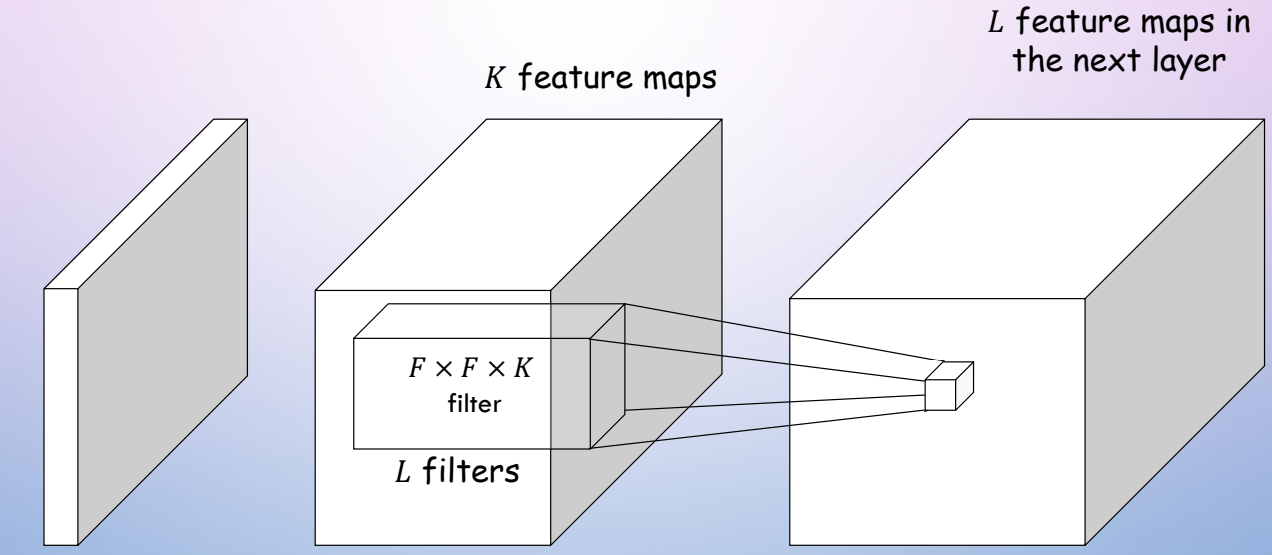


Convolutional layer anatomy



Spatial resolution:
(roughly) the same if stride of 1 is used, reduced by $1/S$ if stride of S is used

Convolutional layer anatomy



Convolutional layer example

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	1	2	2	2	0
0	2	0	1	0	2	0
0	0	0	1	1	2	0
0	1	0	2	2	0	0
0	2	1	2	1	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	0	1	0	2	2	0
0	0	1	0	2	0	0
0	2	1	1	2	0	0
0	1	0	0	0	0	0
0	0	2	1	0	0	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	1	2	1	2	0
0	2	0	1	0	1	0
0	1	1	2	1	0	0
0	1	0	2	1	0	0
0	2	0	0	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

0	0	1
1	-1	-1
1	1	0

w0[:,:,1]

0	1	-1
0	1	0
0	0	0

w0[:,:,2]

1	0	-1
0	-1	1
-1	1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	1	0
1	1	-1
0	-1	-1

w1[:,:,1]

1	1	-1
1	1	1
0	-1	1

w1[:,:,2]

1	1	-1
-1	0	1
-1	1	-1

Bias b1 (1x1x1)

b1[:,:,0]

0

Stride = 2

Animation: http://cs231n.github.io/convolutional-networks/#conv

Convolutional layer example

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	1	2	2	2	0
0	2	0	1	0	2	0
0	0	0	1	1	2	0
0	1	0	2	2	0	0
0	2	1	2	1	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	0	1	0	2	2	0
0	0	1	0	2	0	0
0	2	1	1	2	0	0
0	1	0	0	0	0	0
0	0	2	1	0	0	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	1	2	1	2	0
0	2	0	1	0	1	0
0	1	1	2	1	0	0
0	1	0	2	1	0	0
0	2	0	0	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

0	0	1
1	-1	-1
1	1	0

w0[:,:,1]

0	1	-1
0	1	0
0	0	0

w0[:,:,2]

1	0	-1
0	-1	1
-1	1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	1	0
1	1	-1
0	-1	-1

w1[:,:,1]

1	1	-1
1	1	1
0	-1	1

w1[:,:,2]

1	1	-1
-1	0	1
-1	1	-1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

?	-1	4
4	1	1
-3	3	1

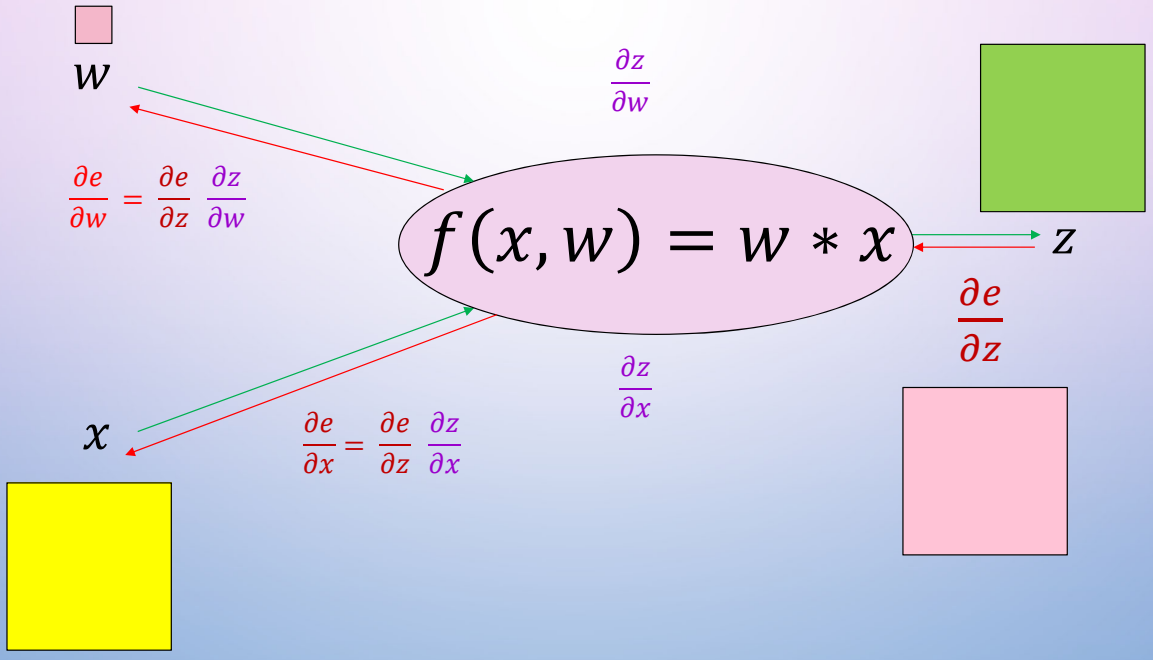
o[:,:,1]

2	6	6
6	2	8
6	10	-2

Stride = 2

Animation: http://cs231n.github.io/convolutional-networks/#conv

Backpropagation for convolutional layer



Size of Feature Map

Output width=((W-F_w+2*P)/S)+1

Output height=((H-F_h+2*P)/S)+1

Example

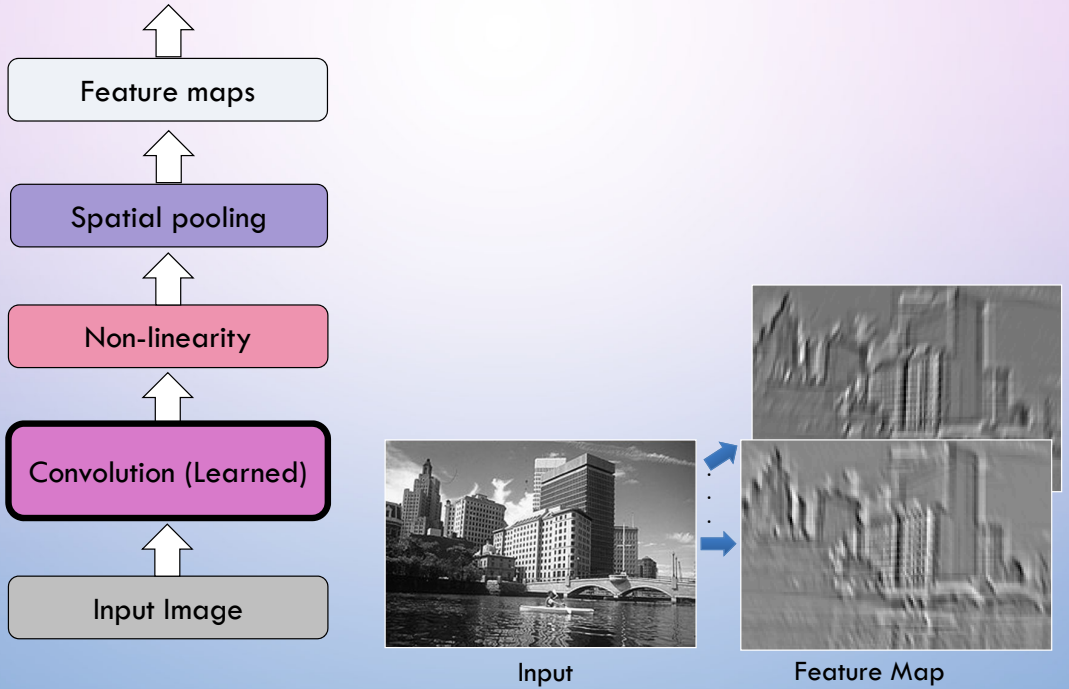
- Tensor size or shape: (width = 28, height = 28)
- Convolution filter size (F): (F_width = 5, F_height = 5)
- Padding (P): 0
- Stride (S): 1

Output width=((28-5+2*0)/1)+1=24

Output height=((28-5+2*0)/1)+1=24

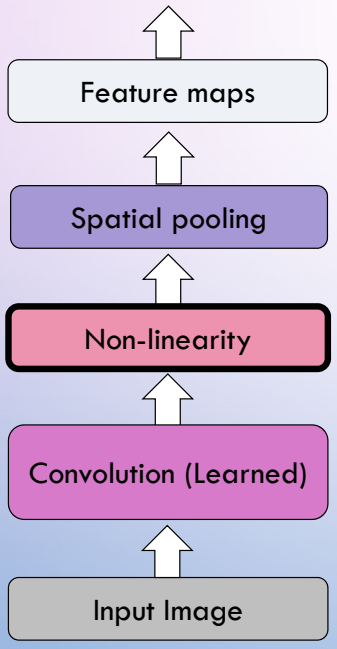
The output dimension will be (24,24)

Summary: CNN pipeline



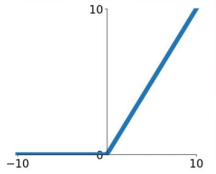
Source: R. Fergus, Y. LeCun

Summary: CNN pipeline



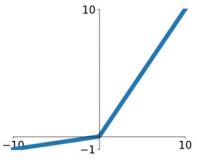
ReLU

$$\max(0, x)$$



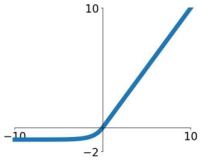
Leaky ReLU

$$\max(0.1x, x)$$



ELU

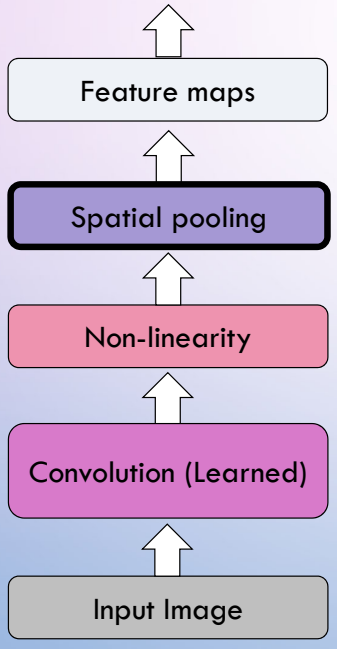
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

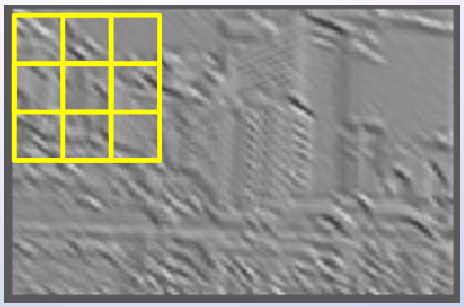


Source: R. Fergus, Y. LeCun

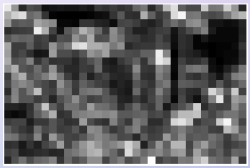
Source: Stanford 231n

Summary: CNN pipeline





Max
(or Avg)

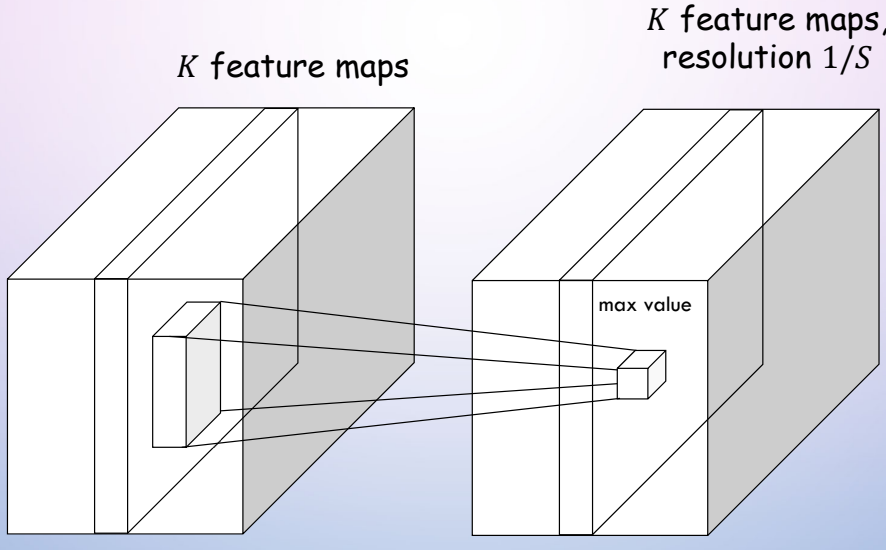


Source: R. Fergus, Y. LeCun

Pooling

- Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information, therefore, controlling overfitting.
- Spatial Pooling can be of different types: Max, Average, Sum etc.
- In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

Max pooling layer

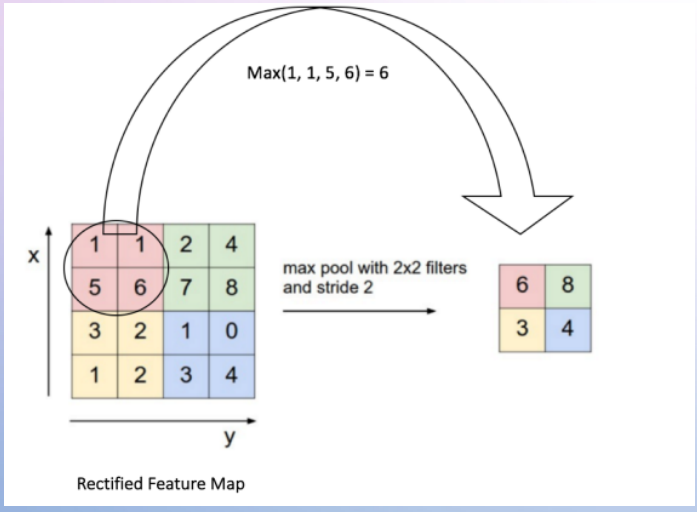


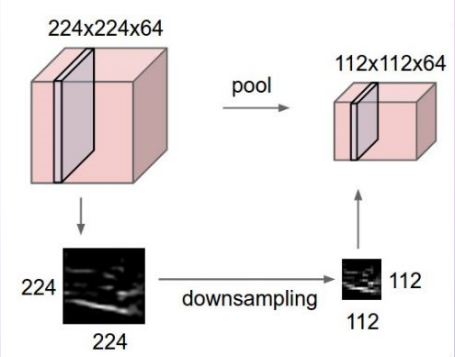
$F \times F$ pooling window, stride S

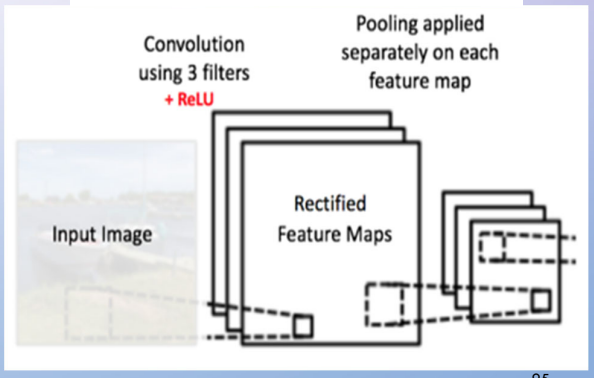
Usually: $F = 2$ or 3, $S = 2$

Backward pass: upstream gradient is passed back only to the unit with max value

Pooling

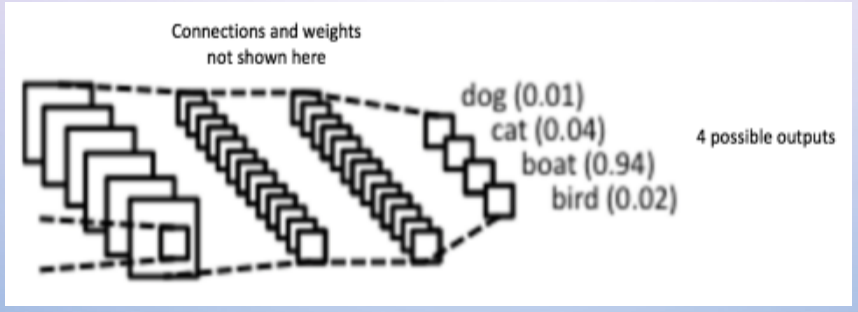






Fully Connected Layer

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax* activation function in the output layer (other classifiers like SVM can also be used). The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer.



***Softmax classifiers give you *probabilities* for each class label.** Softmax classifier is a generalization of the binary form of Logistic Regression

Training the network

The overall training process of the Convolution Network may be summarized as below:
Step1: We initialize all filters and parameters / weights with random values
Step2: The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.
Step3: Calculate the total error at the output layer
Step4: Use Backpropagation to calculate the *gradients* of the error with respect to all weights in the network and use *gradient descent* to update all filter values / weights and parameter values to minimize the output error.
Step5: Repeat steps 2-4 with all images in the training set.

Testing the network

When a new (unseen) image is input into the ConvNet, the network would go through the forward propagation step and output a probability for each class (for a new image, the output probabilities are calculated using the weights which have been optimized to correctly classify all the previous training examples). If our training set is large enough, the network will (hopefully) generalize well to new images and classify them into correct categories.

Some Network Architectures from ImageNet

- Architectures:
  - LeNet-5 (1990s)
  - 1st generation (2012-2013): AlexNet by Alex Krizhevsky improved from LeNet
  - 2nd generation (2014): VGGNet, GoogLeNet
  - 3rd generation (2015): ResNet
  - 4th generation (2016): ResNeXt, DenseNet
  - 5th generation (2017): SENet

Some Network Architectures

- **LeNet (1990s)**
- **1990s to 2012:** In the years from late 1990s to early 2010s convolutional neural network were in incubation. As more and more data and computing power became available, tasks that convolutional neural networks could tackle became more and more interesting.
- **AlexNet (2012)** – In 2012, Alex Krizhevsky (and others) released AlexNet which was a deeper and much wider version of the LeNet and won by a large margin the difficult ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It was a significant breakthrough with respect to the previous approaches and the current widespread application of CNNs can be attributed to this work.
- **GoogLeNet (2014)** – The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google. Its main contribution was the development of an *Inception Module* that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M).

Some Network Architectures

- **VGGNet (2014)** – The runner-up in ILSVRC 2014 was the network that became known as the VGGNet. Its main contribution was in showing that the depth of the network (number of layers) is a critical component for good performance.
- **ResNets (2015)** – Residual Network developed by Kaiming He (and others) was the winner of ILSVRC 2015. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice (as of May 2016).
- **DenseNet (August 2016)** – Recently published by Gao Huang (and others), the Densely Connected Convolutional Network has each layer directly connected to every other layer in a feed-forward fashion. The DenseNet has been shown to obtain significant improvements over previous state-of-the-art architectures on five highly competitive object recognition benchmark tasks
- **SeNet (2017)**
- **MobileNet**

Some Network Architectures - Parameters

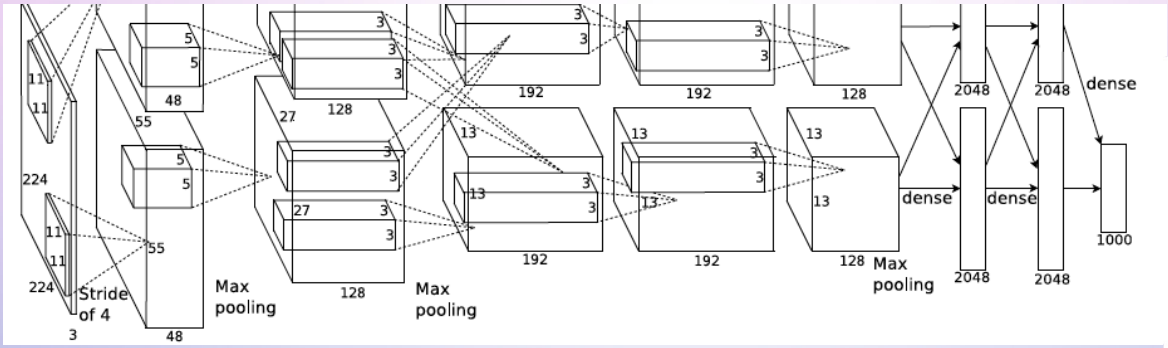
Network	Number of Parameters
LeNet-5	60000
AlexNet	60 million
VGG-16	138 million
GoogleNet	5 million (V1) - 23 million (V3)
ResNet50	25 million
DenseNet-190	40 million
KarNet	16 million

ImageNet Challenge

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/

AlexNet: ILSVRC 2012 winner



- Changes from LeNet-5:
  - Max pooling, ReLU nonlinearity
  - More data and bigger model (7 hidden layers, 650K units, 60M params)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Dropout regularization

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Clarifai: ILSVRC 2013 winner

- Refinement of AlexNet

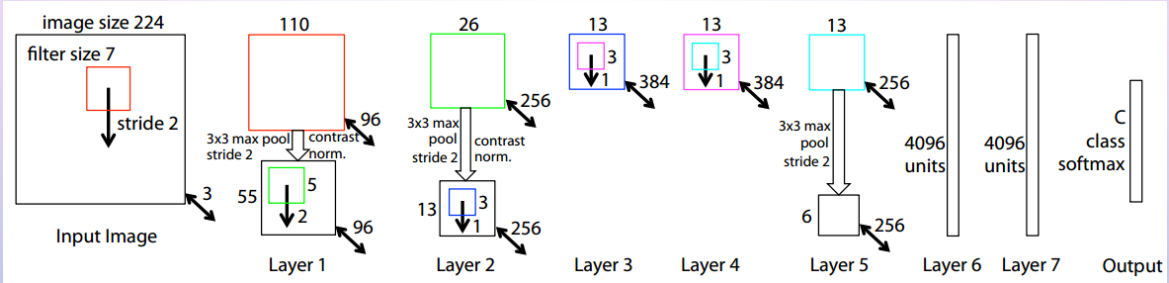


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014 (Best Paper Award winner)

VGGNet: ILSVRC 2014 2nd place

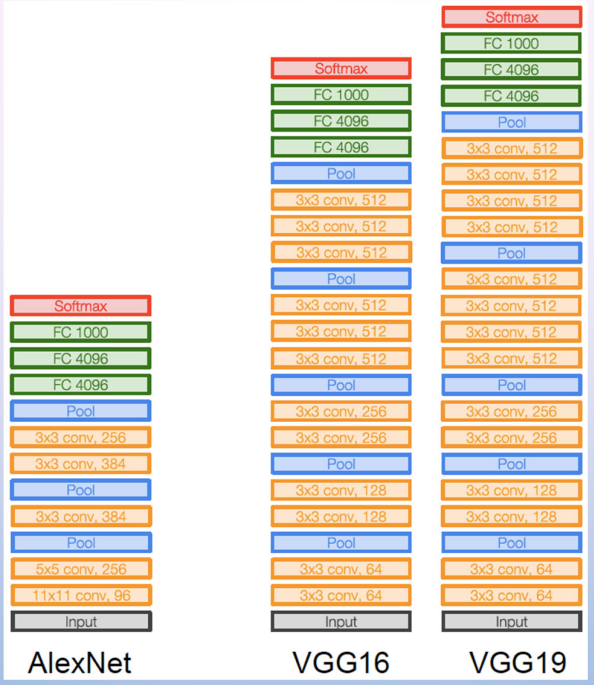


Image source

K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

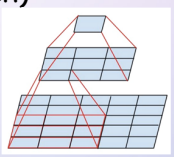
VGGNet: ILSVRC 2014 2nd place

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					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

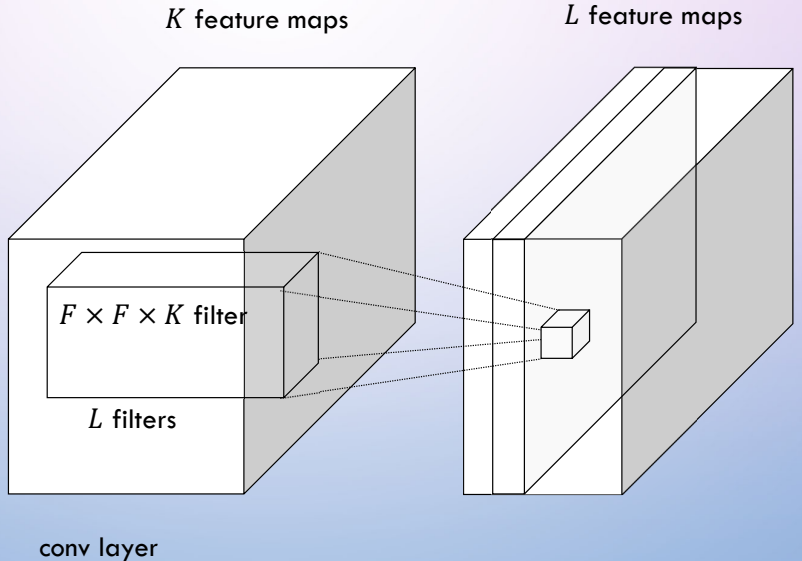
- Sequence of deeper networks trained progressively
- Large receptive fields replaced by successive layers of 3x3 convolutions (with ReLU in between)



- One 7x7 conv layer with K feature maps needs $49K^2$ weights, three 3x3 conv layers need only $27K^2$ weights
- Experimented with 1x1 convolutions

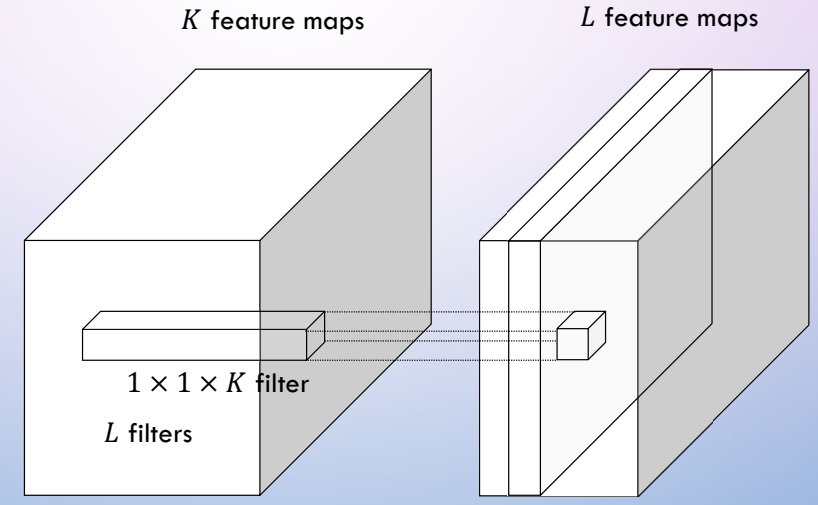
K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

1x1 convolutions



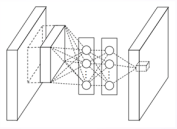
conv layer

1x1 convolutions

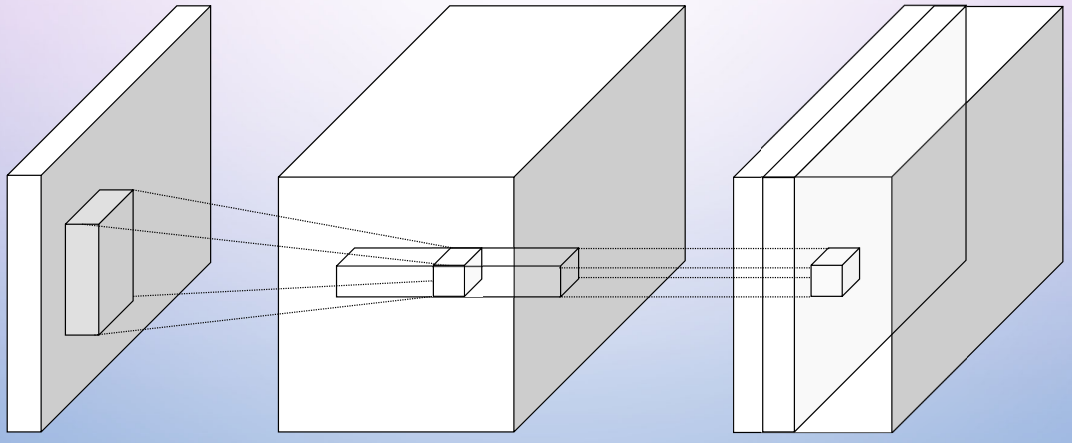


1×1 conv layer

1x1 convolutions

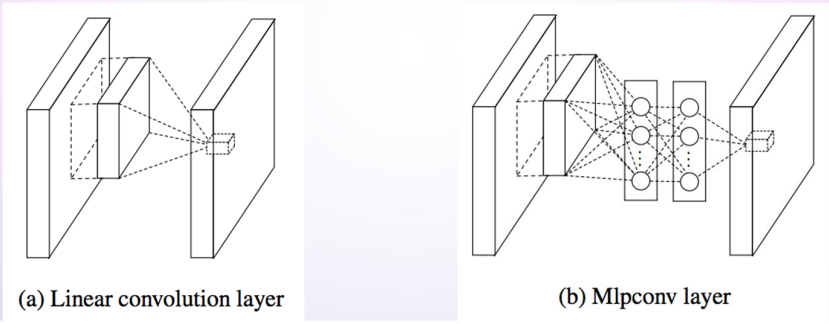


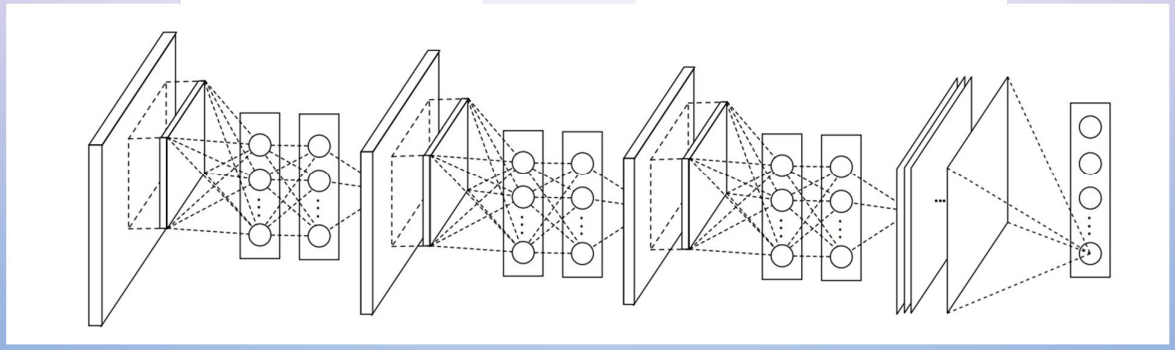
"network in network"



M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014

Network in network





M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014

GoogLeNet: ILSVRC 2014 winner

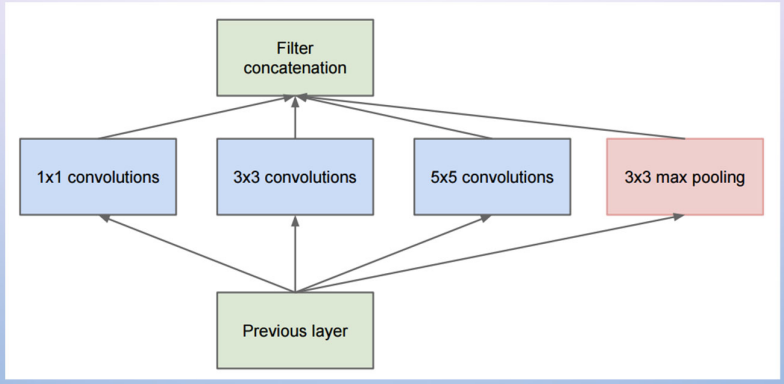
- The Inception Module

http://knowyourmeme.com/memes/we-need-to-go-deeper

C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet

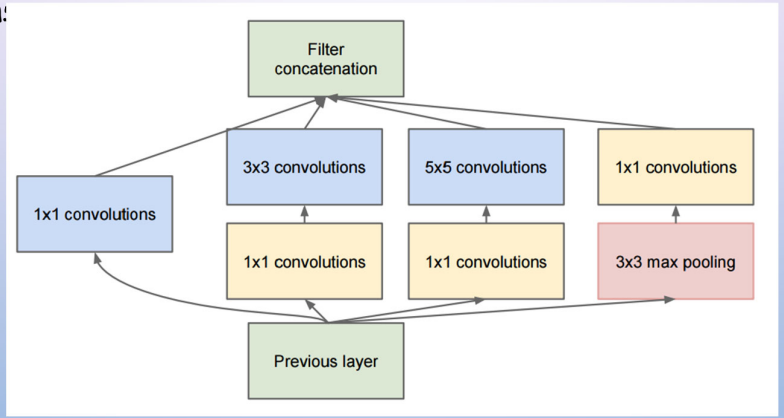
- The Inception Module
  - Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

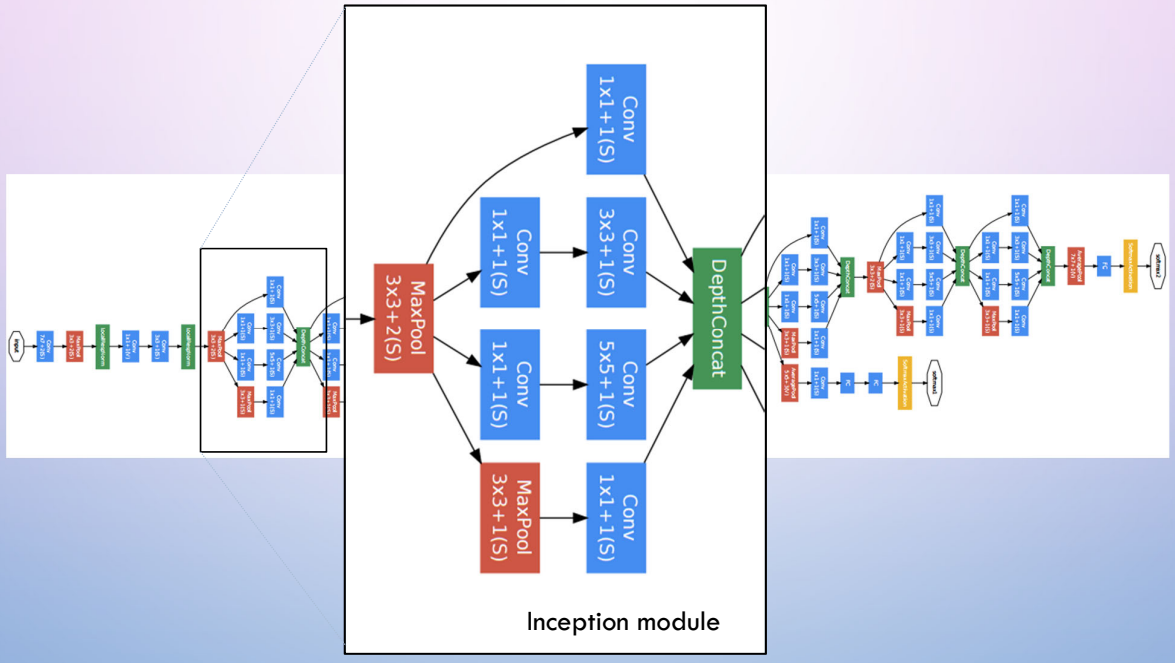
GoogLeNet

- The Inception Module
  - Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps
  - Use 1x1 convolutions for dimensionality reduction before expensive convolutions



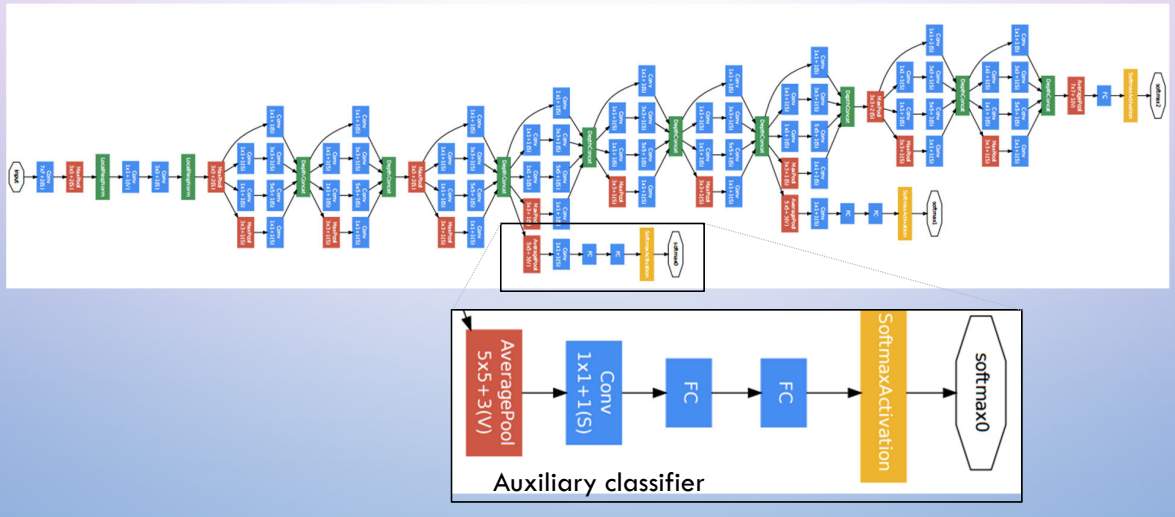
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet

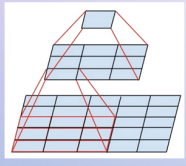
- An alternative view:

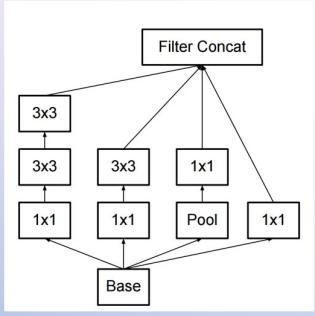
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

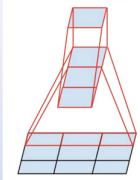
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

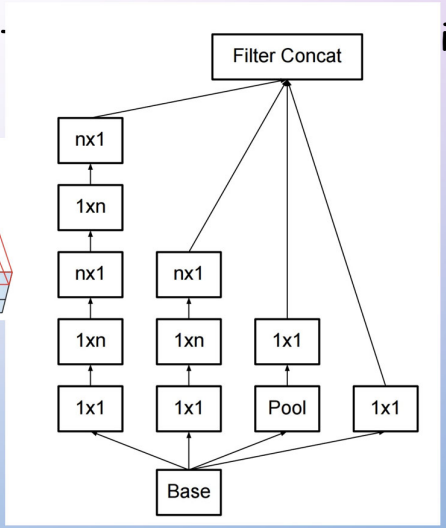
Inception v2, v3

- Improve training with batch normalization, reducing importance of auxiliary classifiers
- More variants of inception modules wi ion of filters





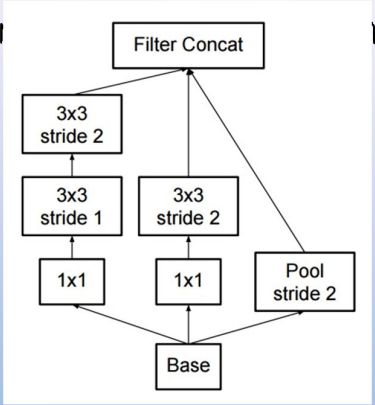




C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016

Inception v2, v3

- Improve training with batch normalization, reducing importance of auxiliary classifiers
- More variants of inception modules with aggressive factorization of filters
- Increase the numbe[illegible]ile decreasing spatial resolution (pooling)



C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016

ImageNet Challenge 2012-2014

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

ResNet: ILSVRC 2015 winner

K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

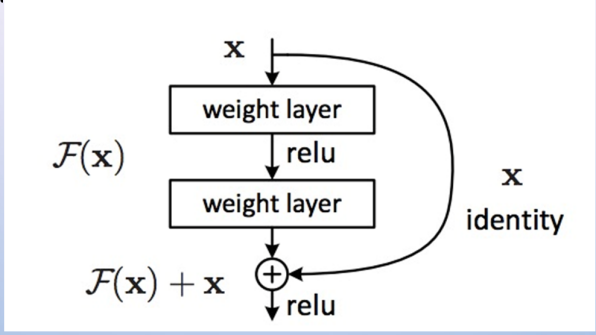
ResNet: ILSVRC 2015 winner

K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

Source (?)

ResNet

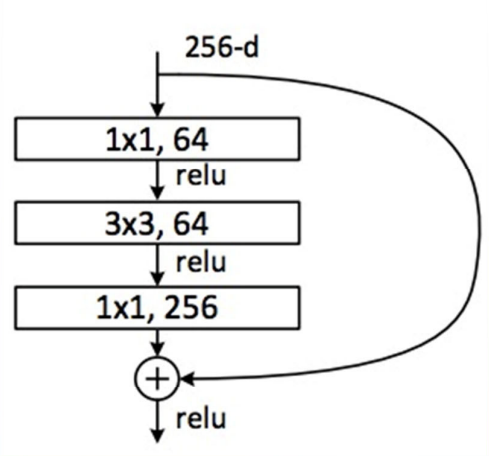
- The residual module
  - Introduce *skip* or *shortcut* connections (existing before in various forms in literature)
  - Make it easy for network layers to represent the identity mapping



K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

ResNet

Deeper residual module (bottleneck)



- Directly performing 3x3 convolutions with 256 feature maps at input and output:
  256 x 256 x 3 x 3 ~ 600K operations
- Using 1x1 convolutions to reduce 256 to 64 feature maps, followed by 3x3 convolutions, followed by 1x1 convolutions to expand back to 256 maps:
  256 x 64 x 1 x 1 ~ 16K
  64 x 64 x 3 x 3 ~ 36K
  64 x 256 x 1 x 1 ~ 16K
  Total: ~70K

K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

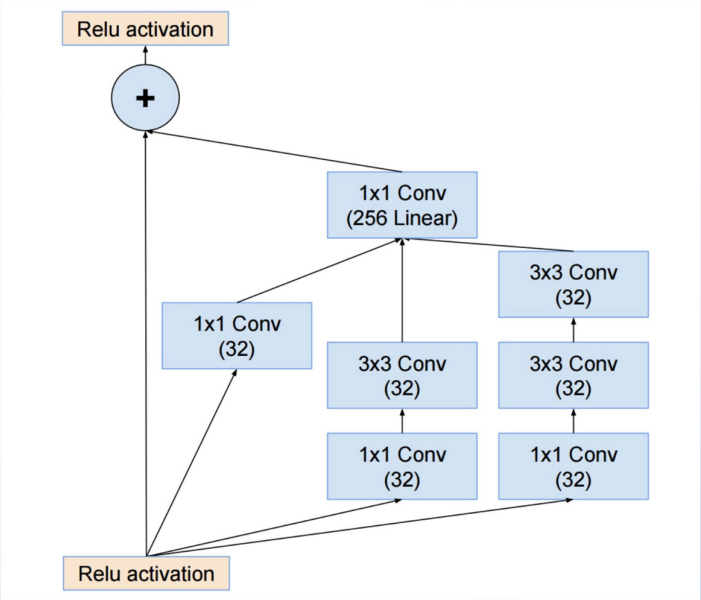
ResNet

- Architectures for ImageNet:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

Inception v4



C. Szegedy et al., Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv 2016

Summary: ILSVRC 2012-2015

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (AlexNet, 7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st*	3.57%	

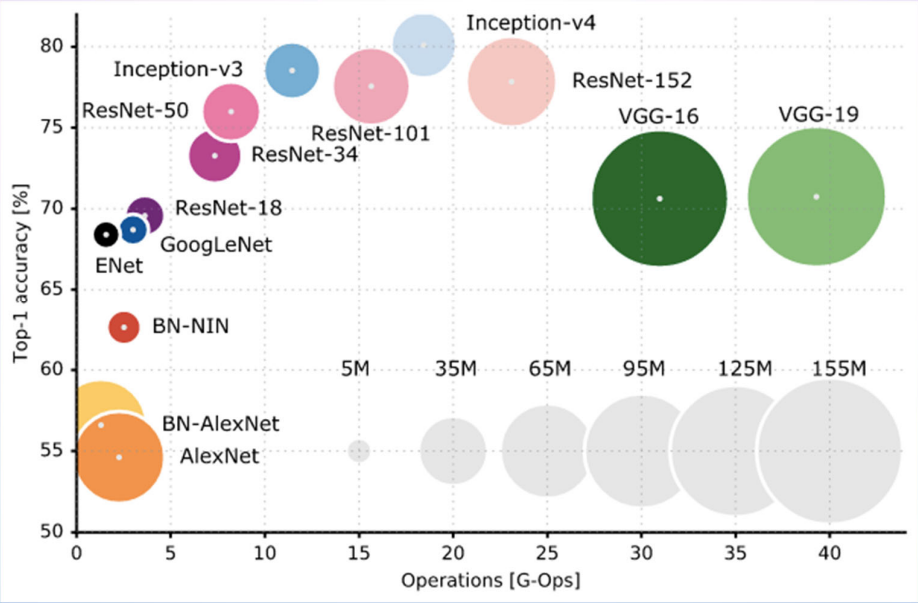
*Officially, there was no longer a classification competition and ResNet-based systems won in localization and detection

Techniques necessary for state-of-the-art performance

- Training tricks and details: initialization, regularization, normalization
- Training data augmentation
- Averaging classifier outputs over multiple crops/flips
- Ensembles of networks

•(to be covered next time)

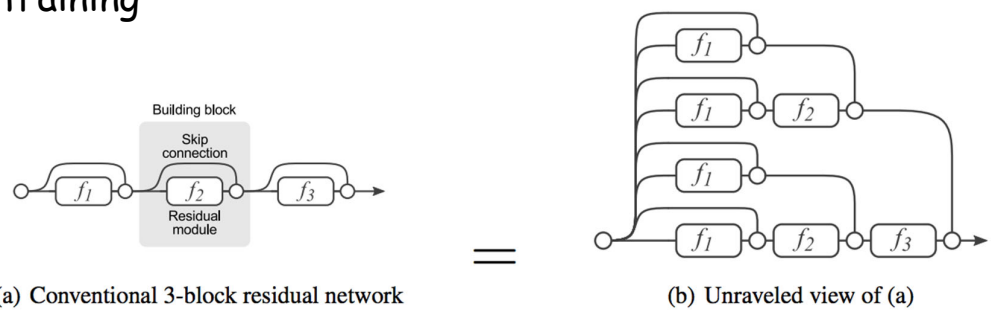
Comparing architectures



https://culurciello.github.io/tech/2016/06/04/nets.html

Beyond ResNets: Why do they work?

- ResNets are collections of many paths of different length, and shorter paths predominantly contribute to training



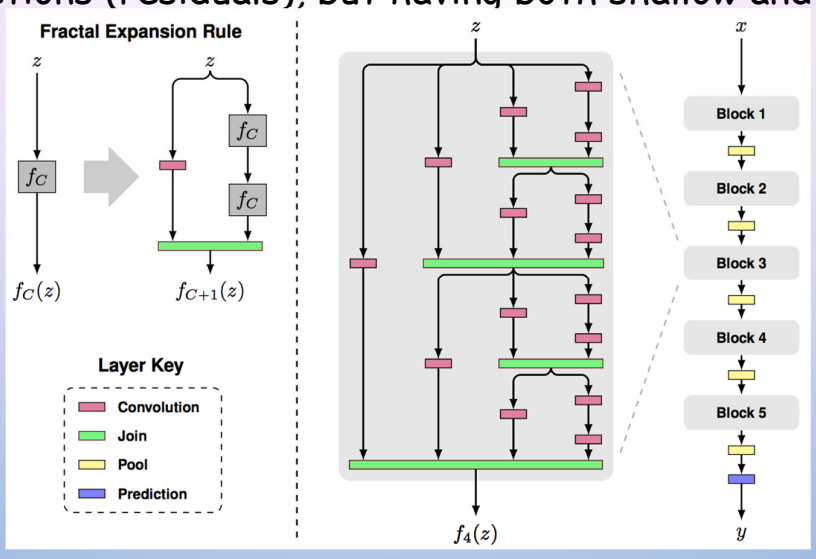
(a) Conventional 3-block residual network (b) Unraveled view of (a)

Figure 1: Residual Networks are conventionally shown as (a), which is a natural representation of Equation (1). When we expand this formulation to Equation (6), we obtain an *unraveled view* of a 3-block residual network (b). Circular nodes represent additions. From this view, it is apparent that residual networks have $O(2^n)$ implicit paths connecting input and output and that adding a block doubles the number of paths.

A. Veit, M. Wilber, S. Belongie, Residual Networks Behave Like Ensembles of Relatively Shallow Networks, NIPS 2016

Beyond ResNet: FractalNet

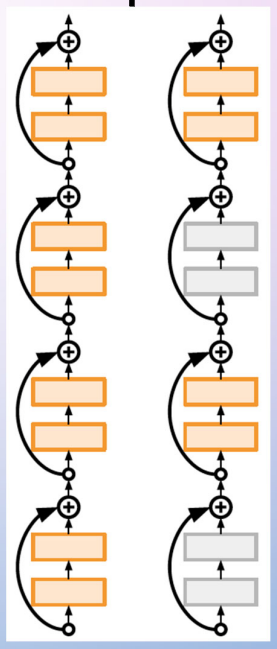
- Claim: key to good performance is not having skip connections (residuals), but having both shallow and deep paths



S. Larsson, M. Maire and G. Shakhnarovich, FractalNet: Ultra-Deep Neural Networks without Residuals, ICLR 2017

Beyond ResNet: Stochastic depth

- At training time, in each mini-batch, randomly drop a subset of layers (bypass with identity function). At test time, use full network
  - Forces the network to learn better, speeds up convergence, improves accuracy



G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, Deep Networks with Stochastic Depth, ECCV 2016

Beyond ResNet: Wide ResNet

- Reduce number of residual blocks, but increase number of feature maps in each block
  - More parallelizable, better feature reuse
  - 16-layer WRN outperforms 1000-layer ResNets, though with much larger # of parameters

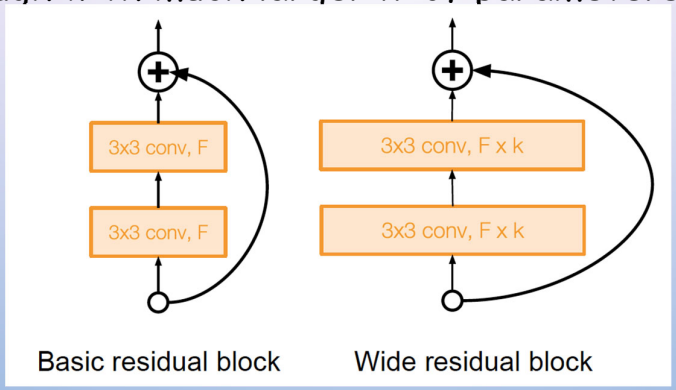


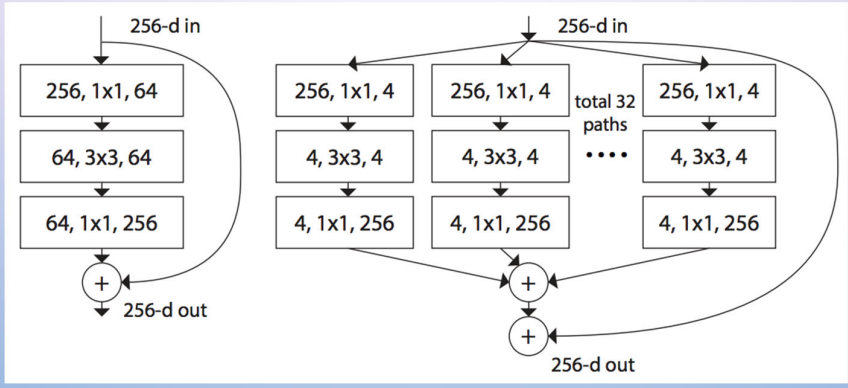
Image source

S. Zagoryuko and N. Komodakis, Wide Residual Networks, BMVC 2016

Beyond ResNet: ResNeXt

- Propose "cardinality" as a new factor in network design, apart from depth and width
- Claim that increasing cardinality is a better way to increase capacity than increasing depth or width

ResNet ResNeXt, roughly same complexity



S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, Aggregated Residual Transformations for Deep Neural Networks, CVPR 2017

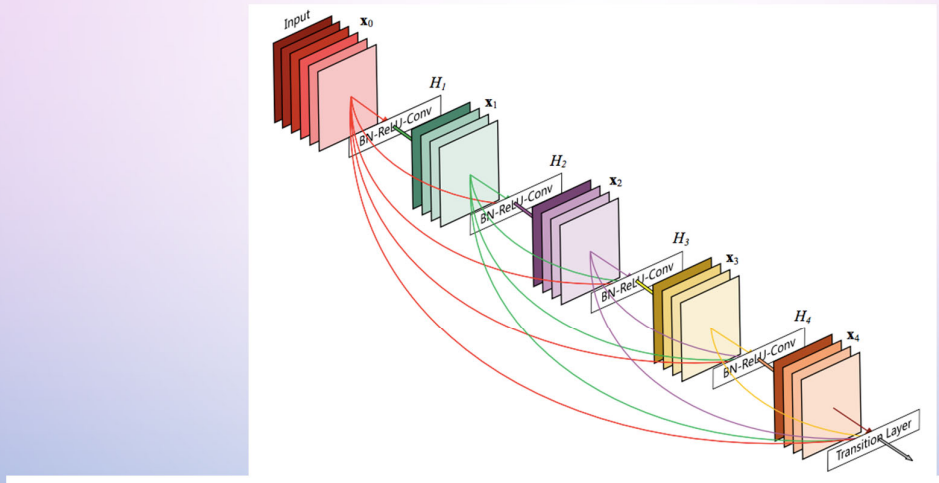
Beyond ResNet: ResNeXt

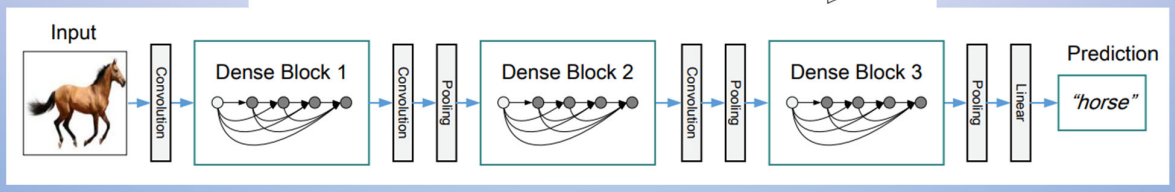
	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

Table 3. Ablation experiments on ImageNet-1K. (**Top**): ResNet-50 with preserved complexity (∼4.1 billion FLOPs); (**Bottom**): ResNet-101 with preserved complexity (∼7.8 billion FLOPs). The error rate is evaluated on the single crop of 224×224 pixels.

S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, Aggregated Residual Transformations for Deep Neural Networks, CVPR 2017

Beyond ResNet: DenseNets





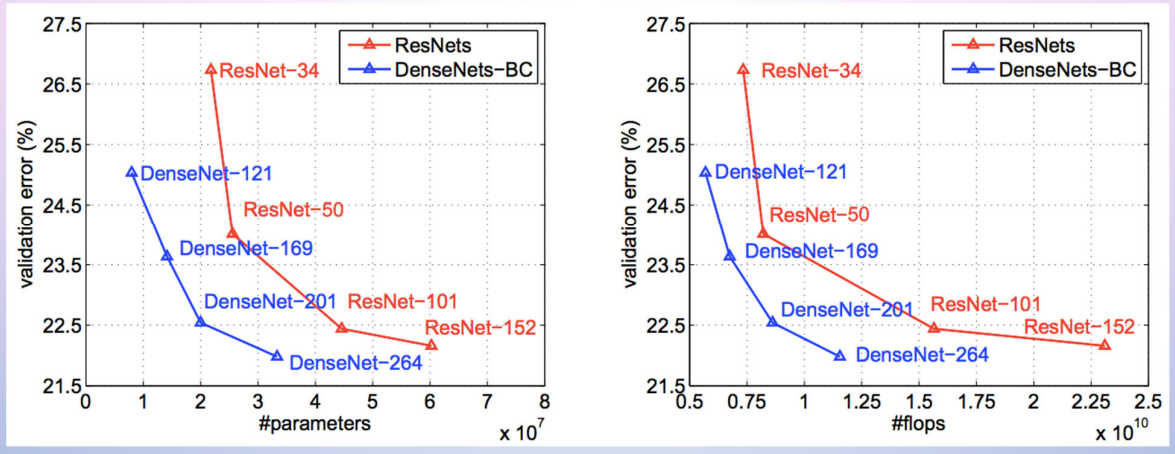
G. Huang, Z. Liu, and L. van der Maaten, Densely Connected Convolutional Networks, CVPR 2017 (Best Paper Award)

DenseNets

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

G. Huang, Z. Liu, and L. van der Maaten, Densely Connected Convolutional Networks, CVPR 2017 (Best Paper Award)

DenseNets



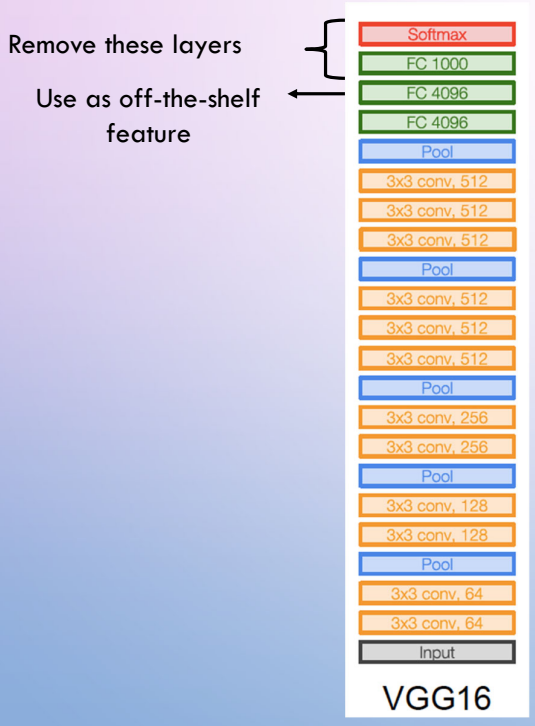
ImageNet validation error vs. number of parameters and test-time operations

G. Huang, Z. Liu, and L. van der Maaten, Densely Connected Convolutional Networks, CVPR 2017 (Best Paper Award)

Design principles

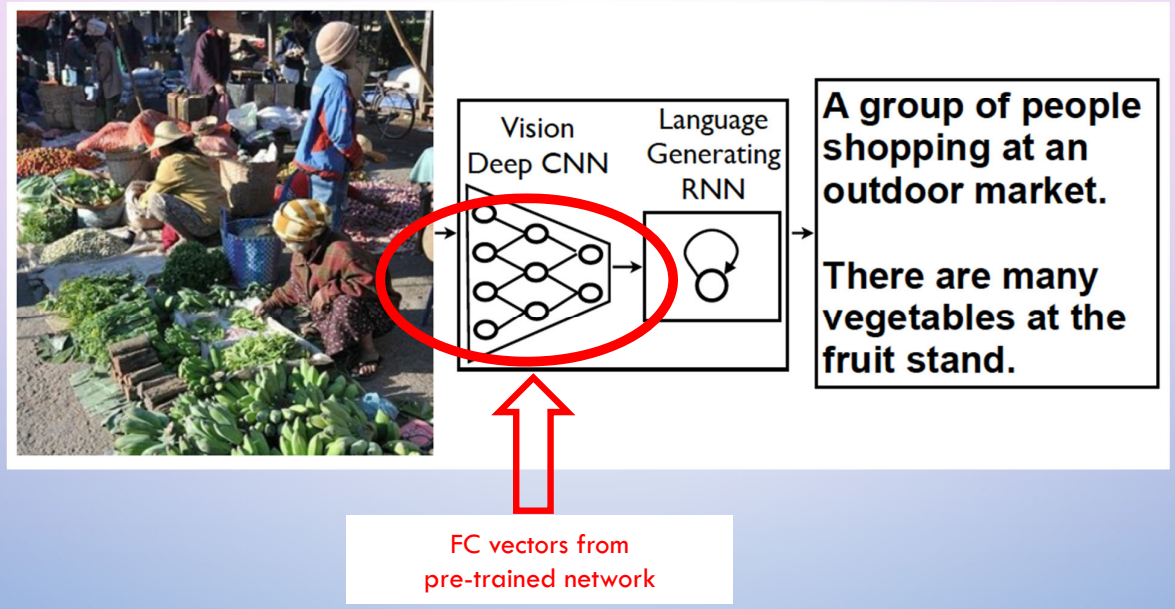
- Make networks parameter-efficient
  - Reduce filter sizes, factorize filters
  - Use 1x1 convolutions to reduce number of feature maps before more expensive operations
  - Minimize reliance on FC layers
- Reduce spatial resolution gradually, within each level of resolution replicate a given "block" multiple times
- Use skip connections and/or create multiple redundant paths through the network
- Play around with depth vs. width vs. "cardinality"

How to use a pre-trained network for a new task?



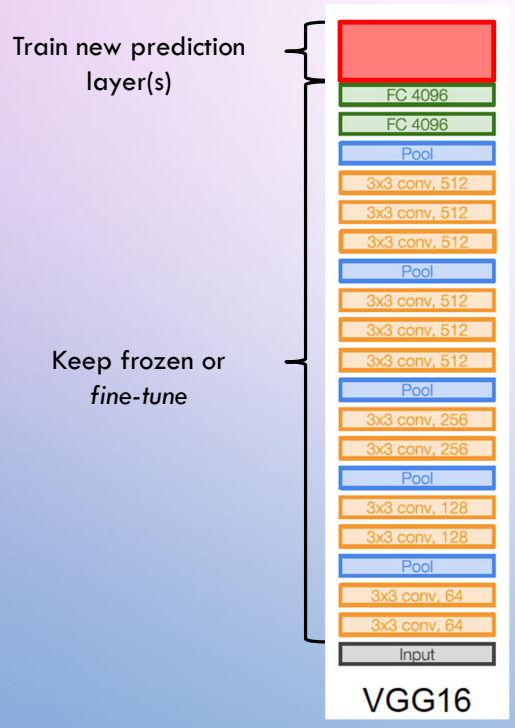
- Strategy 1: Use as feature extractor

Example: CNNs for image captioning



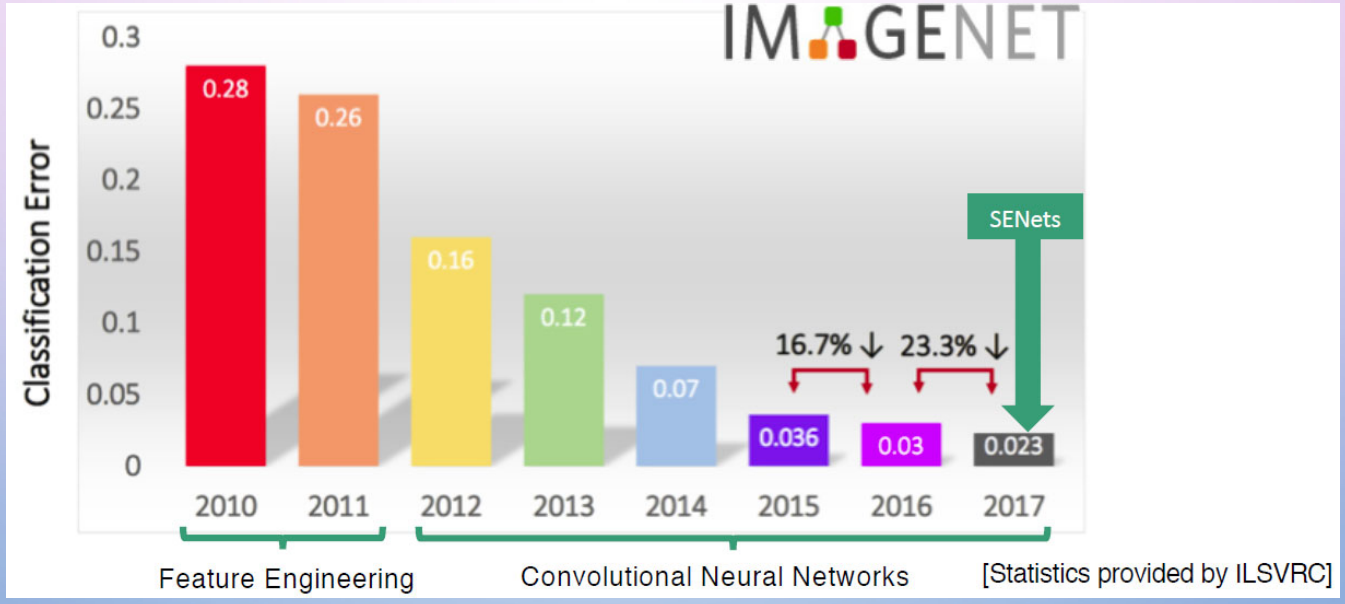
O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. CVPR 2015

How to use a pre-trained network for a new task?



- Strategy 2: Transfer learning

SeNets



Installation

Packages	Installation	Link documents
Python 3.6-3.8.x	Installer 64 bits	https://www.python.org/
numpy	pip install --upgrade numpy	http://www.numpy.org/
sciPy	pip install --upgrade scipy	https://www.scipy.org/
Matplotlib	pip install --upgrade matplotlib	https://matplotlib.org/
Pandas	pip install --upgrade pandas	http://pandas.pydata.org/
scikit-learn	pip install --upgrade scikit-learn	http://scikit-learn.org/stable
TensorFlow 2.X	pip install --upgrade tensorflow	https://www.tensorflow.org
Keras	pip install --upgrade keras	https://keras.io/

Wheel link: http://www.lfd.uci.edu/~gohlke/pythonlibs/
***Set path of python in environment variable,
C:\Users\...\AppData\Local\Programs\Python\Python3X\Scripts\
C:\Users\...\AppData\Local\Programs\Python\Python3X\

Installation

Packages	Installation	Link documents
Jupyter	pip install --upgrade jupyterlab	http://jupyter.org
OpenCV	pip install --upgrade opencv-python	https://pypi.org/project/opencv-python/ https://pythonexamples.org/python-opencv/
Pillow	pip install --upgrade pil	https://pypi.org/project/Pillow/ Image Module — Pillow (PIL Fork) 8.0.1 documentation
Pytorch		https://pytorch.org/

GPU Tools	Link documents
NVIDIA® GPU drivers	CUDA® architectures 3.5, 5.0, 6.0, 7.0, 7.5, 8.0 and higher than 8.0 https://www.nvidia.com/download/index.aspx?lang=en-us
CUDA® Toolkit (64bits)	TensorFlow supports CUDA® 11.2 (TensorFlow >= 2.5.0) https://developer.nvidia.com/cuda-toolkit-archive
cuDNN SDK 8.1.0	https://developer.nvidia.com/rdp/cudnn-archive

References

- http://introtodeeplearning.com/
- https://www.deeplearningbook.org/
- https://cs231n.github.io/
- http://slazebni.cs.illinois.edu/fall20/
- (Tutorial) KERAS Tutorial: DEEP LEARNING in PYTHON - DataCamp