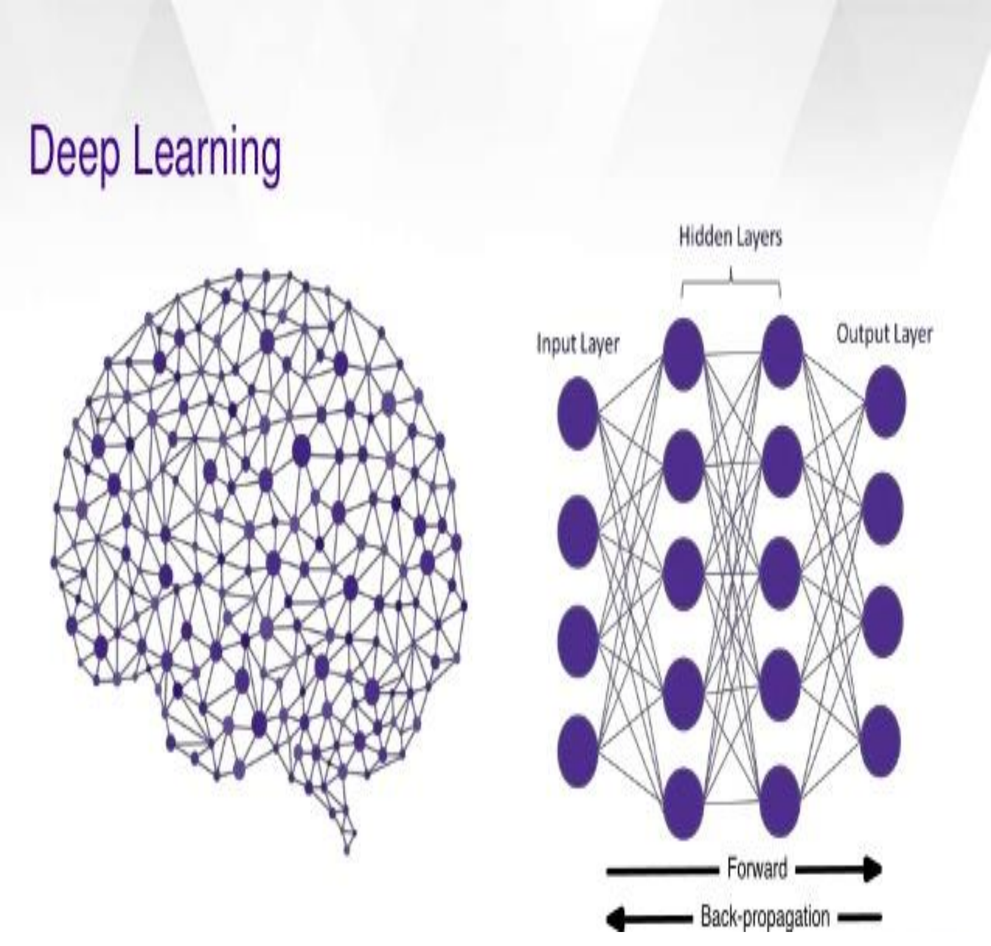


Machine Learning VS Deep Learning

Presented by : Dr. Hanaa Bayomi

h.mobarz@fci-cu.edu.eg

Agenda

1- Machine learning

- Definition and types
- machine Learning road map
- feature selection
  - filter, wrapper and embedded
- Model selection
  - cross validation(K-fold)

2- Deep learning

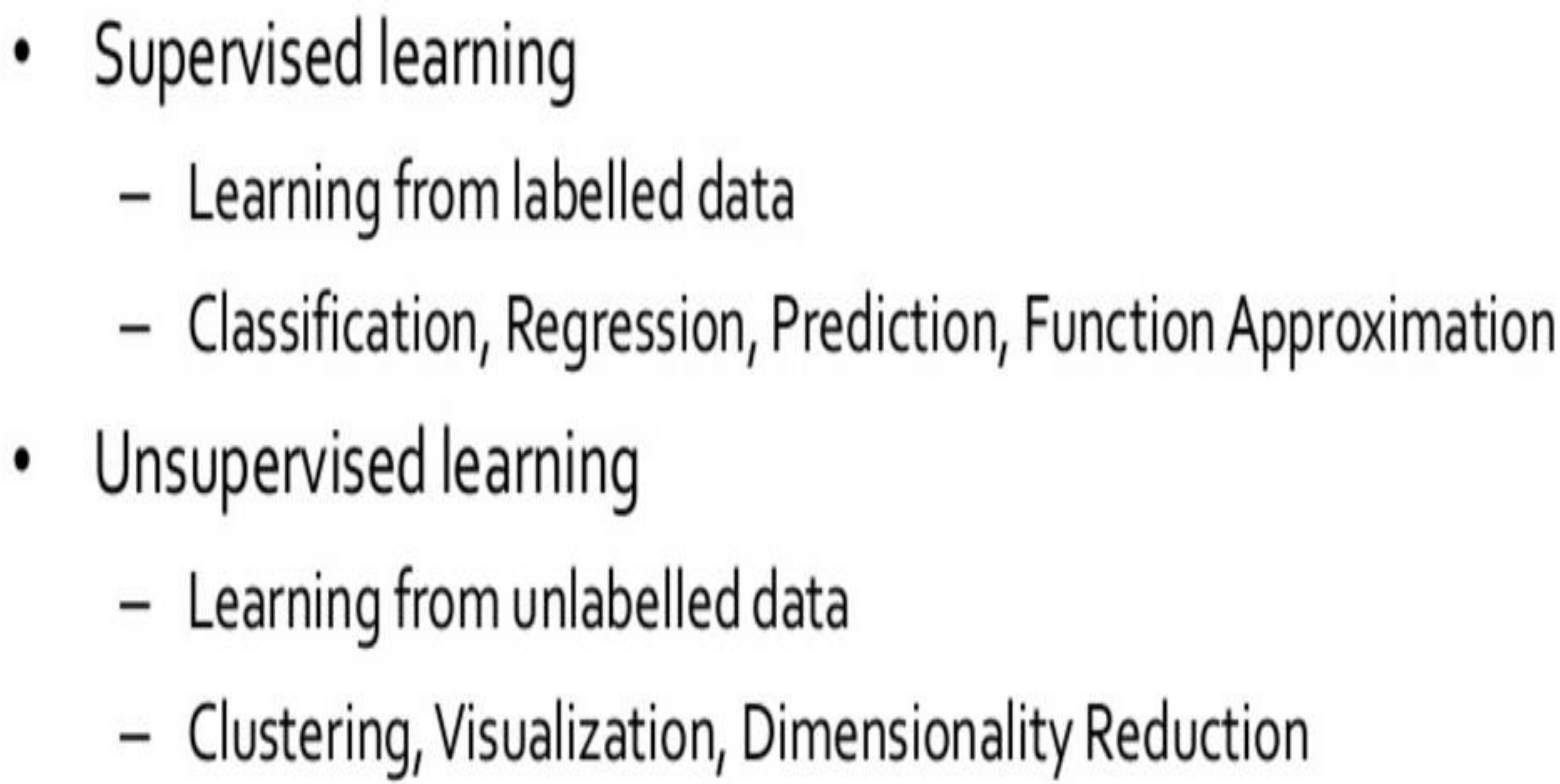
- Definition
- ML VS DL
- DL architecture
  - fully connected NN
  - convolution NN
  - Recurrent NN (LSTM)

3- NLP Tasks

Machine Learning definition

- One definition of machine learning: A computer program improves its performance on a given task with experience (i.e. examples, data).
- Task: What is the problem that the program is solving?
- Experience: What is the data (examples) that the program is using to improve its performance?
- Performance measure: How is the performance of the program (when solving the given task) evaluated?

Machine Learning types

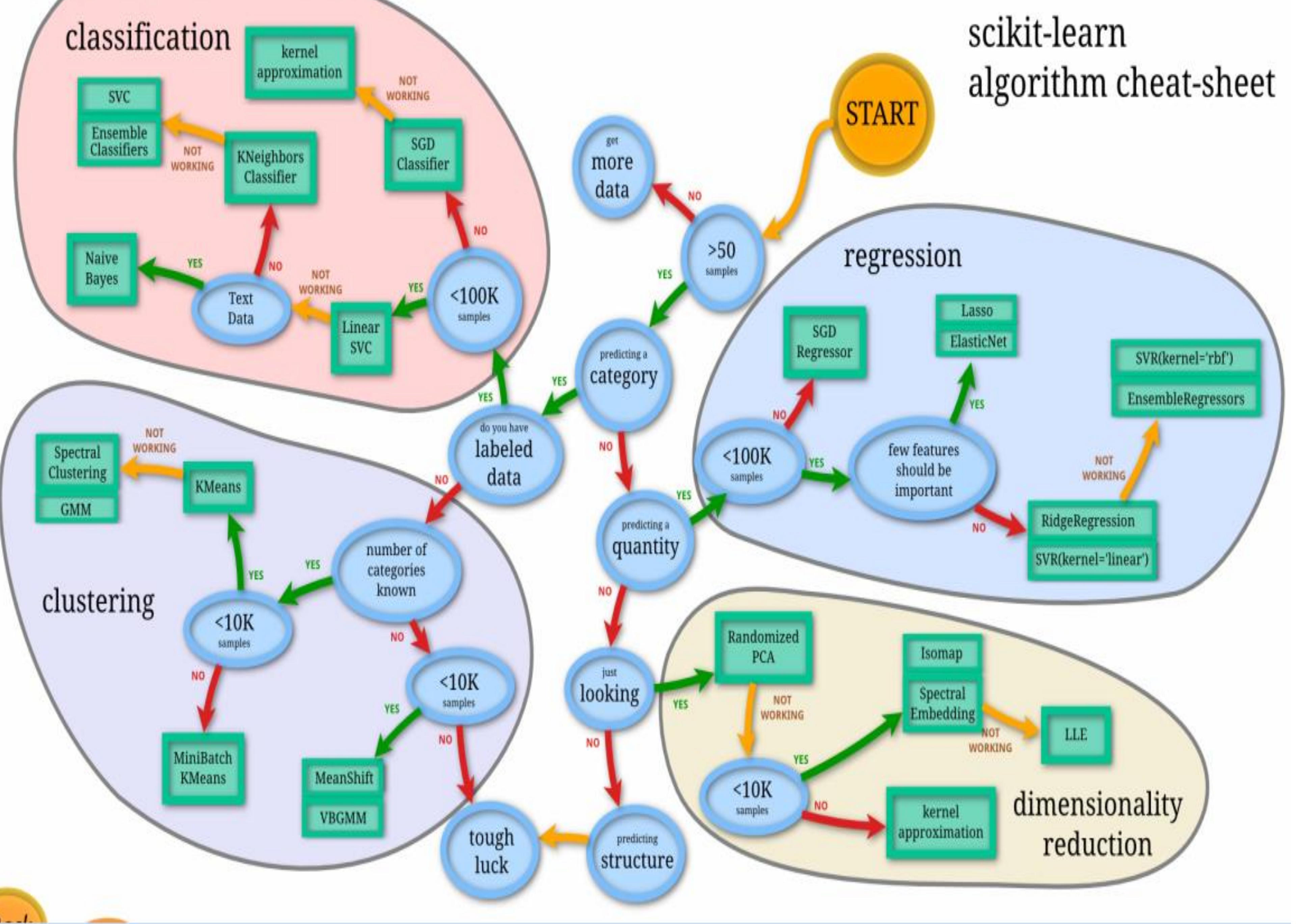


- Supervised learning
  - Learning from labelled data
  - Classification, Regression, Prediction, Function Approximation
- Unsupervised learning
  - Learning from unlabelled data
  - Clustering, Visualization, Dimensionality Reduction

Machine Learning types

- Semi-supervised learning
  - mix of Supervised and Unsupervised learning
  - usually small part of data is labelled
- Reinforcement learning
  - Model learns from a series of actions by maximizing a reward function
  - The reward function can either be maximized by penalizing bad actions and/or rewarding good actions
  - Example - training of self-driving car using feedback from the environment

http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html



Learning Types

	Supervised	Unsupervised
Discrete	Classification	Clustering
Continuous	Regression	Dimensionality reduction

Feature selection

Performance of Machine Learning model depend on

- Choice of algorithm
- Feature selection
- Feature creation
- Model selection

https://archive.ics.uci.edu/ml/datasets.html

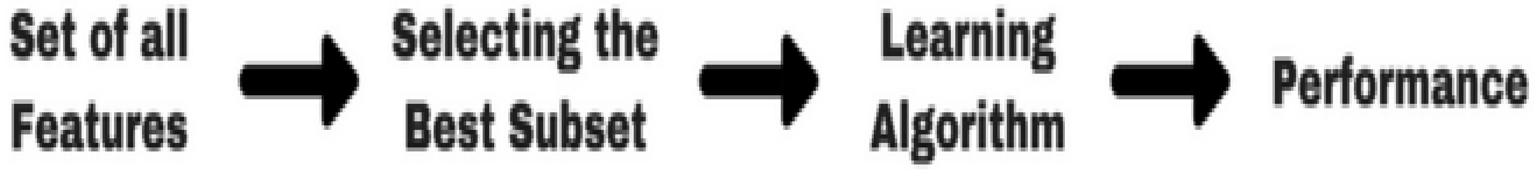
UCI Machine Learning Repository: Data Sets

Classification of FS methods

- Filter (single factor analysis)
  - Assess the relevance of features only by looking at the essential properties of the data.
  - Usually, calculate the feature relevance score and remove low-scoring features.
- Wrapper
  - Bundle the search for best model with the FS.
  - Generate and evaluate various subsets of features. The evaluation is obtained by training and testing a specific ML model.
- Embedded
  - Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are **regularization methods**.

Filter methods

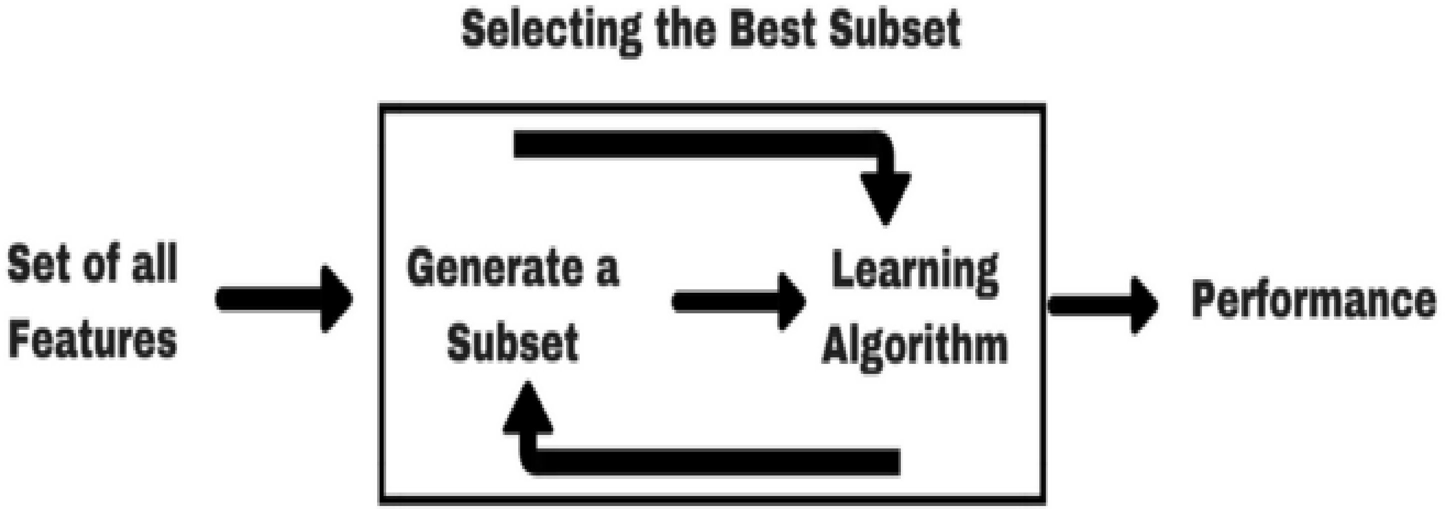
- Filter methods are generally used as a *preprocessing step*. The selection of features is independent of any machine learning algorithms.
- Two steps (score-and-filter approach)
  1. assess each feature individually for its potential in discriminating among classes in the data
  2. features falling beyond threshold are eliminated



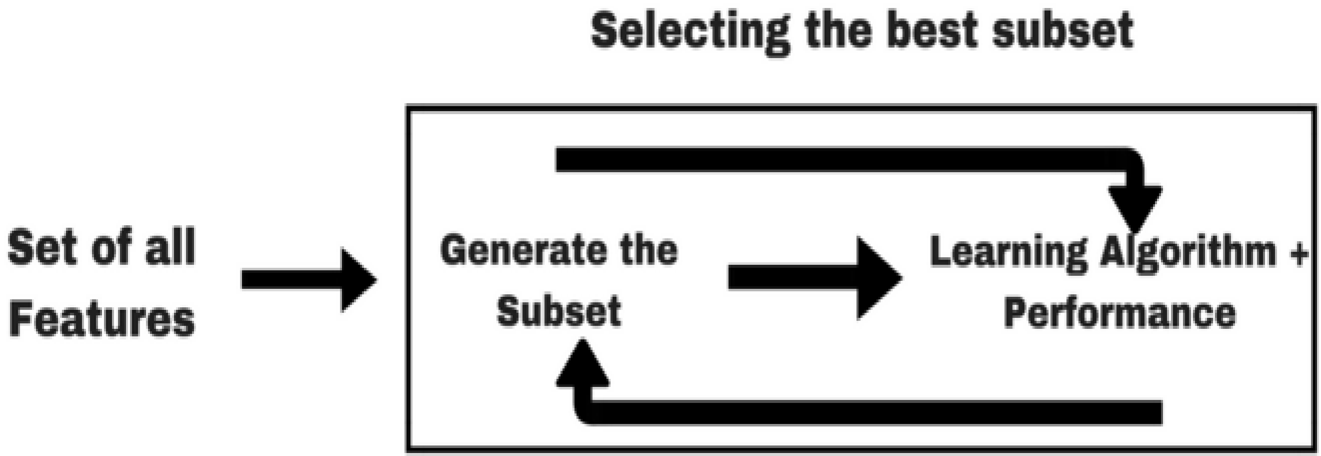
Wrappers

- Search for the best feature subset in combination with a fixed classification method.
- The goodness of a feature subset is determined

-one-out



Embedded



Some of the most popular examples of these methods are LASSO and RIDGE regression which have inbuilt penalization functions to reduce over fitting.

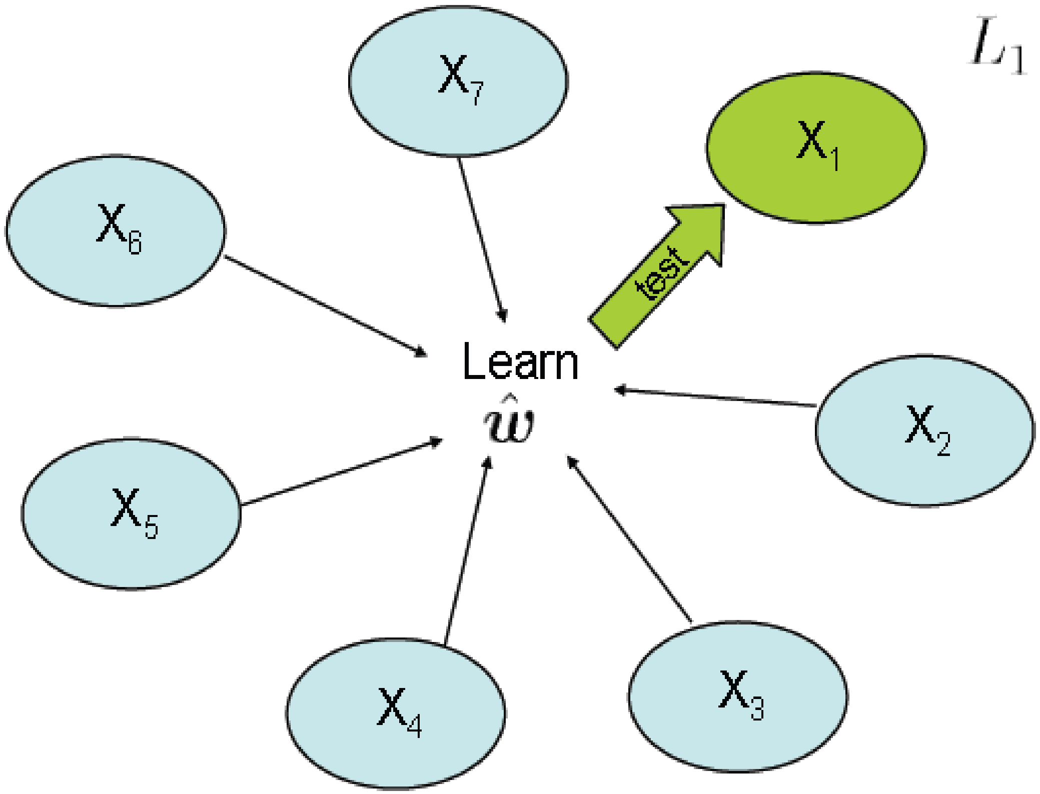
Lasso regression performs L1 regularization which adds penalty equivalent to absolute value of the magnitude of coefficients.

Ridge regression performs L2 regularization which adds penalty equivalent to square of the magnitude of coefficients.

Choosing the best model

K-fold cross validation

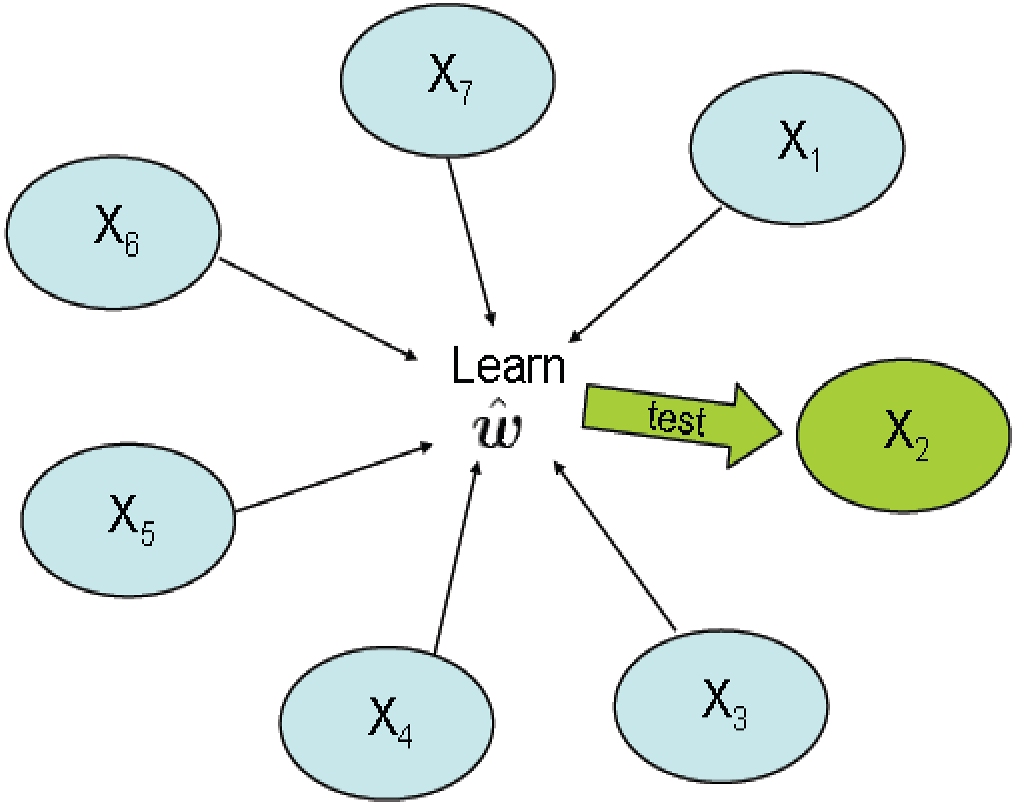
- A technique for estimating test error
- Uses *all* of the data to validate
- Divide data into K groups $\{X_1, X_2, \ldots, X_K\}$.
- Use each group as a validation set, then average all validation errors



$$L_1 = \sum_{(\boldsymbol{x},y)\in X_1} (y - \hat{\boldsymbol{w}}^\top \boldsymbol{x})$$

K-fold cross validation

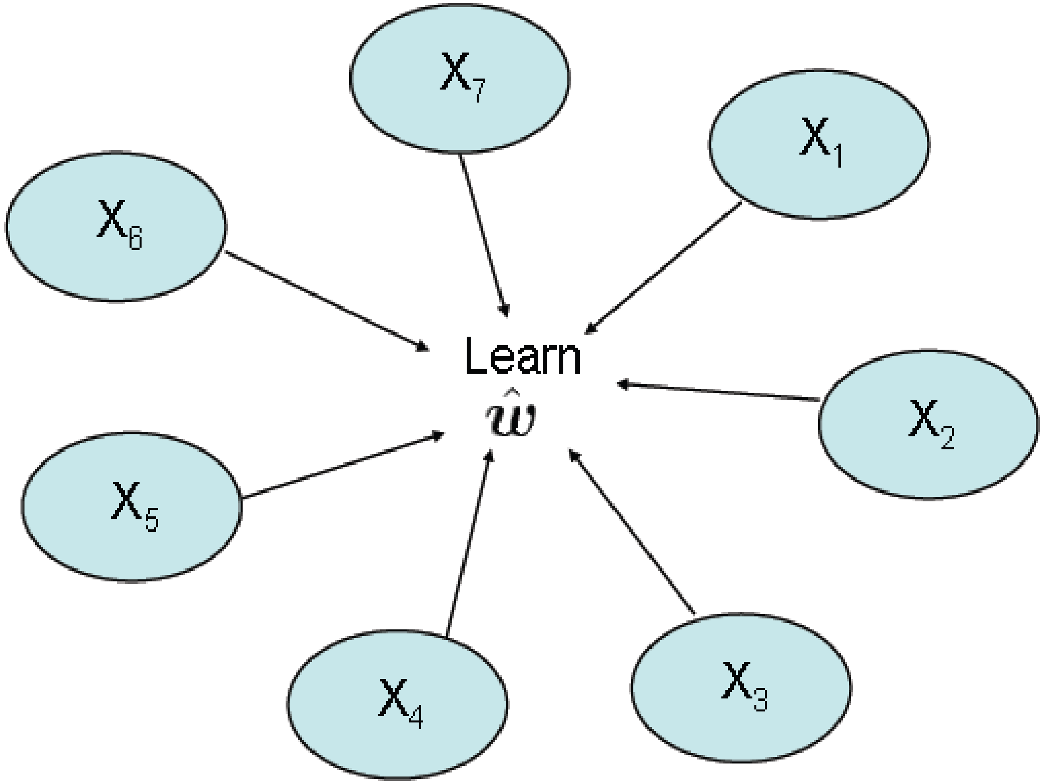
- A technique for estimating test error
- Uses *all* of the data to validate
- Divide data into K groups $\{X_1, X_2, \ldots, X_K\}$.
- Use each group as a validation set, then average all validation errors



$$L_2 = \sum_{(\boldsymbol{x},y)\in X_2} (y - \hat{\boldsymbol{w}}^\top \boldsymbol{x})$$

K-fold cross validation

- A technique for estimating test error
- Uses *all* of the data to validate
- Divide data into K groups $\{X_1, X_2, \ldots, X_K\}$.
- Use each group as a validation set, then average all validation errors



$$CV(s) = \frac{1}{K}\sum_{i=1}^{K} L_i$$

Deep Learning definition

- *Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.*

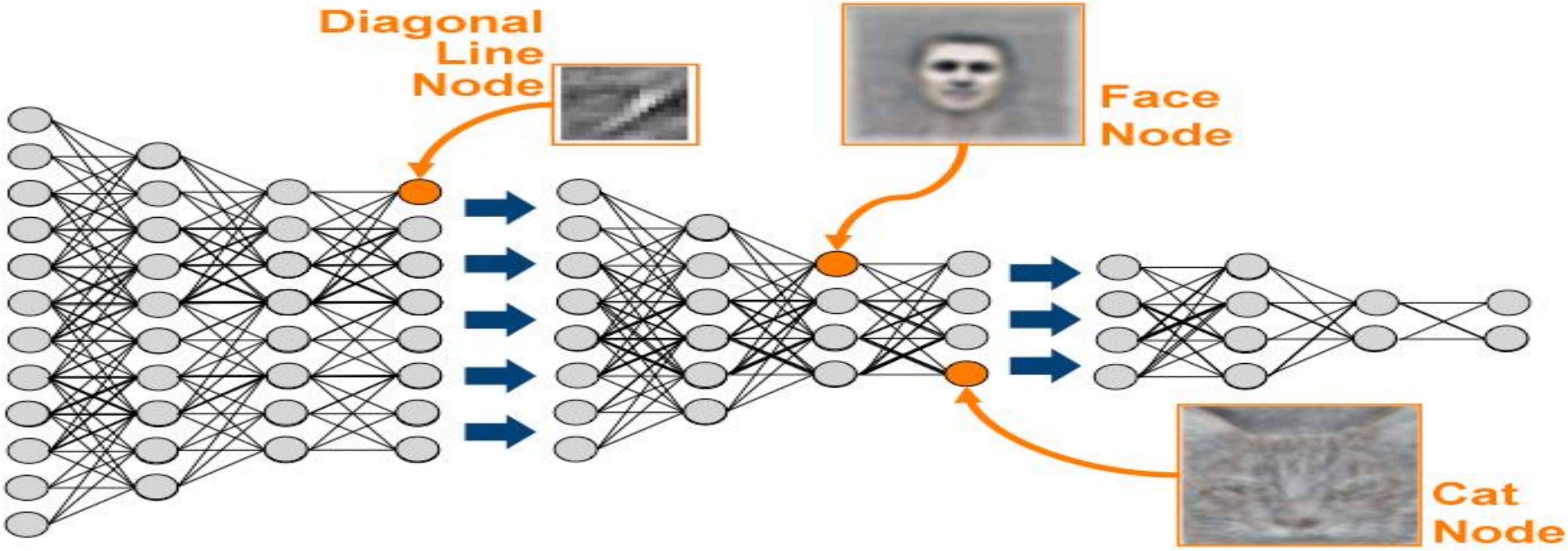
- Learning deep (many layered) neural networks

- The more layers in a Neural Network, the more abstract features can be represented

Deep Learning definition

E.g. Classify a cat:

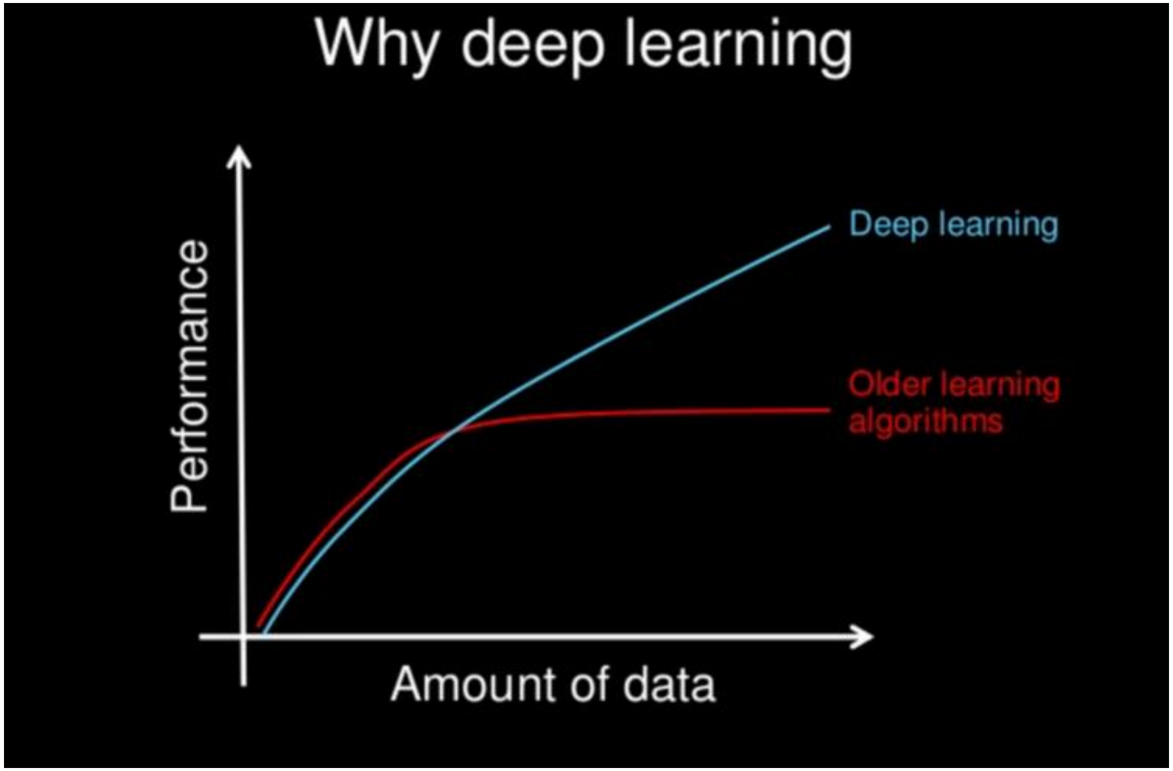
- – Bottom Layers: Edge detectors, curves, corners straight lines
- – Middle Layers: Fur patterns, eyes, ears
- – Higher Layers: Body, head, legs
- – Top Layer: Cat or Dog



Machine Learning VS Deep Learning

1- Data Dependency

- Deep learning need large amount of data to understand it perfectly



Machine Learning VS Deep Learning

2- Hardware Dependency

- Deep learning algorithms heavily depend on high-end machines This is because the requirements of deep learning algorithm include GPUs which are an integral part of its working.
- Machine Learning which can work on low-end machines.

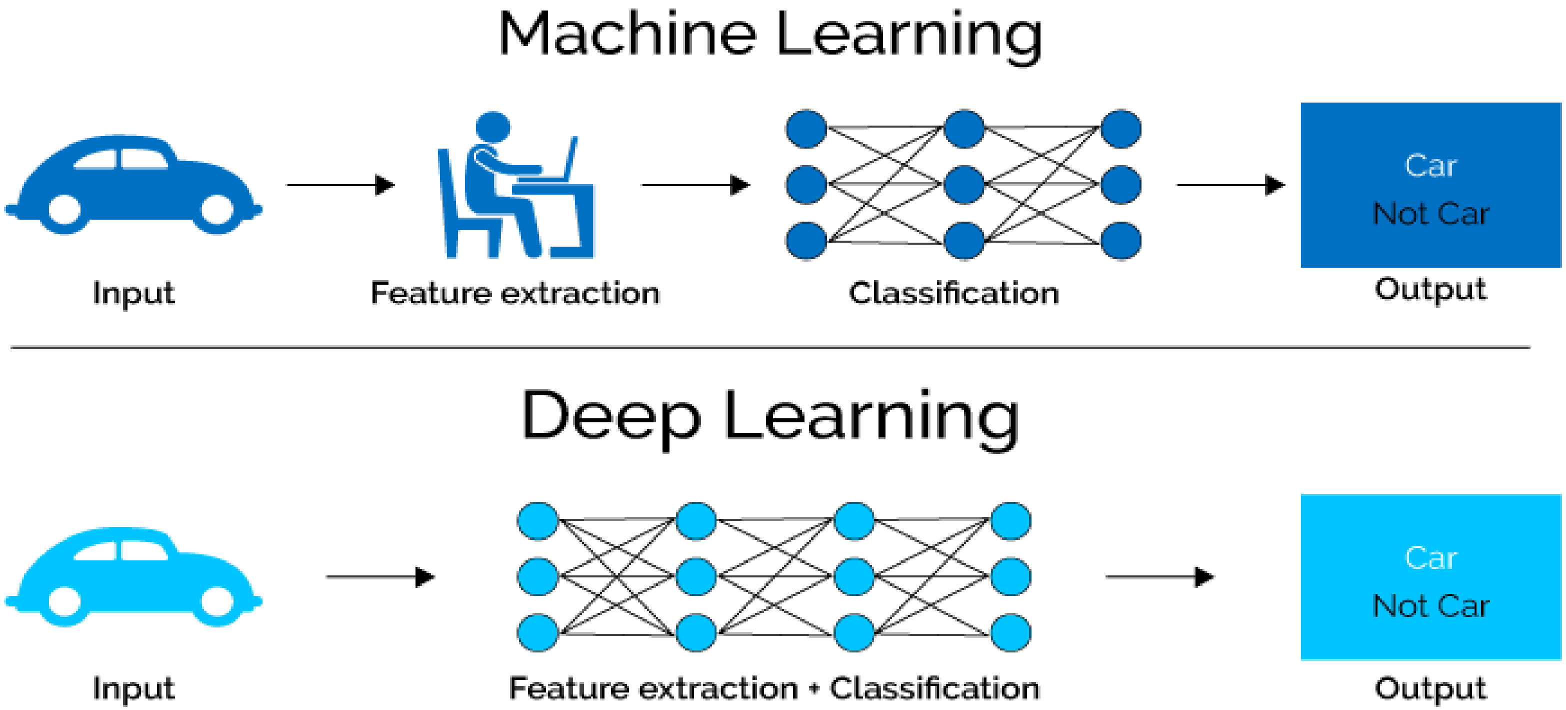
3- Execution time

- deep learning algorithm takes a long time to train. This is because there are so many parameters in a deep learning algorithm that training them takes longer than usual.

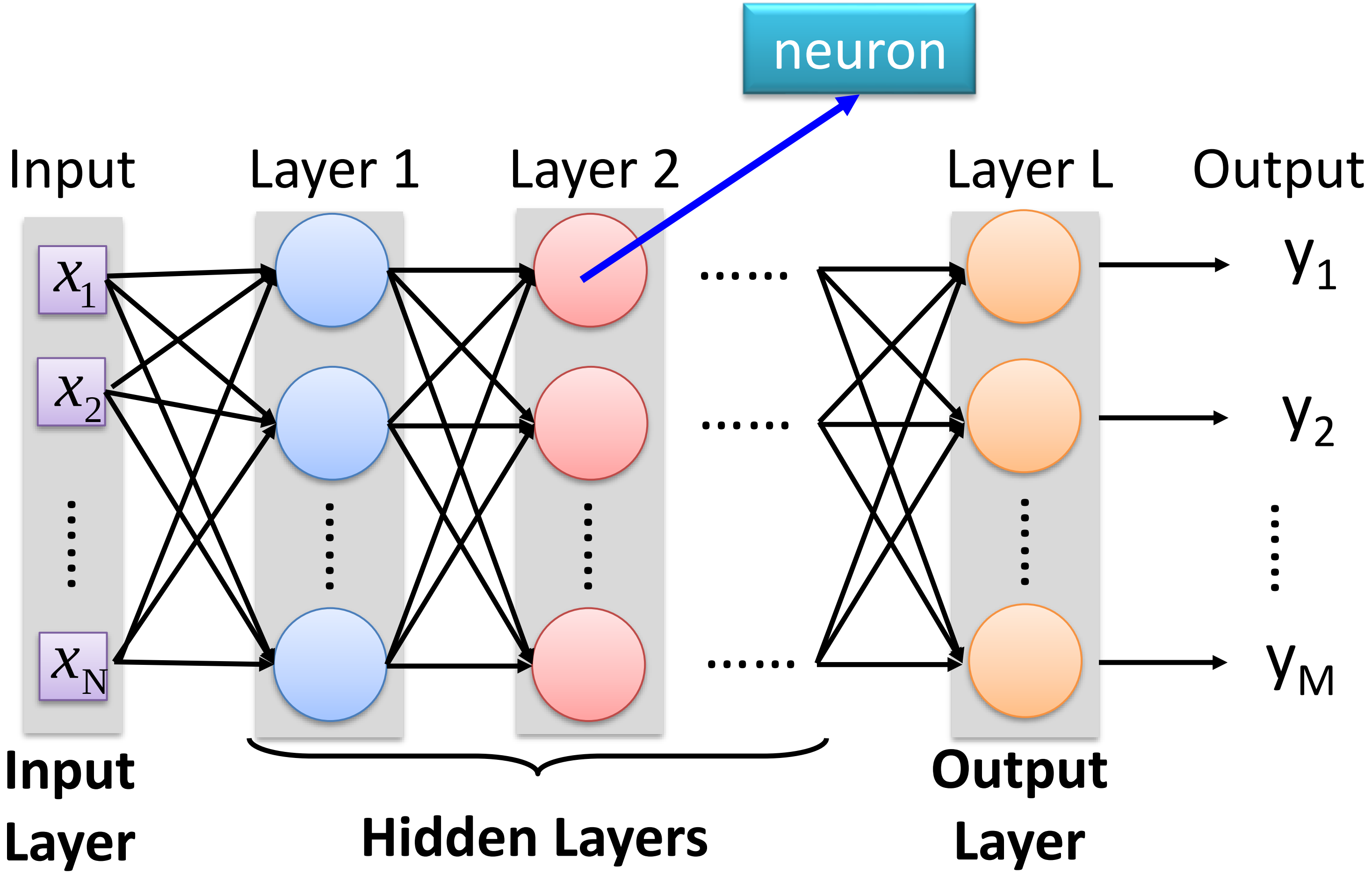
Machine Learning VS Deep Learning

4- Feature engineering

- Deep learning algorithms try to learn high-level features from data.
- Machine Learning which can work on low-end machines.



Element of Neural Network



neuron

Input

Layer 1

Layer 2

Layer L

Output

x_1

x_2

x_N

y_1

y_2

y_M

……

Input Layer

Hidden Layers

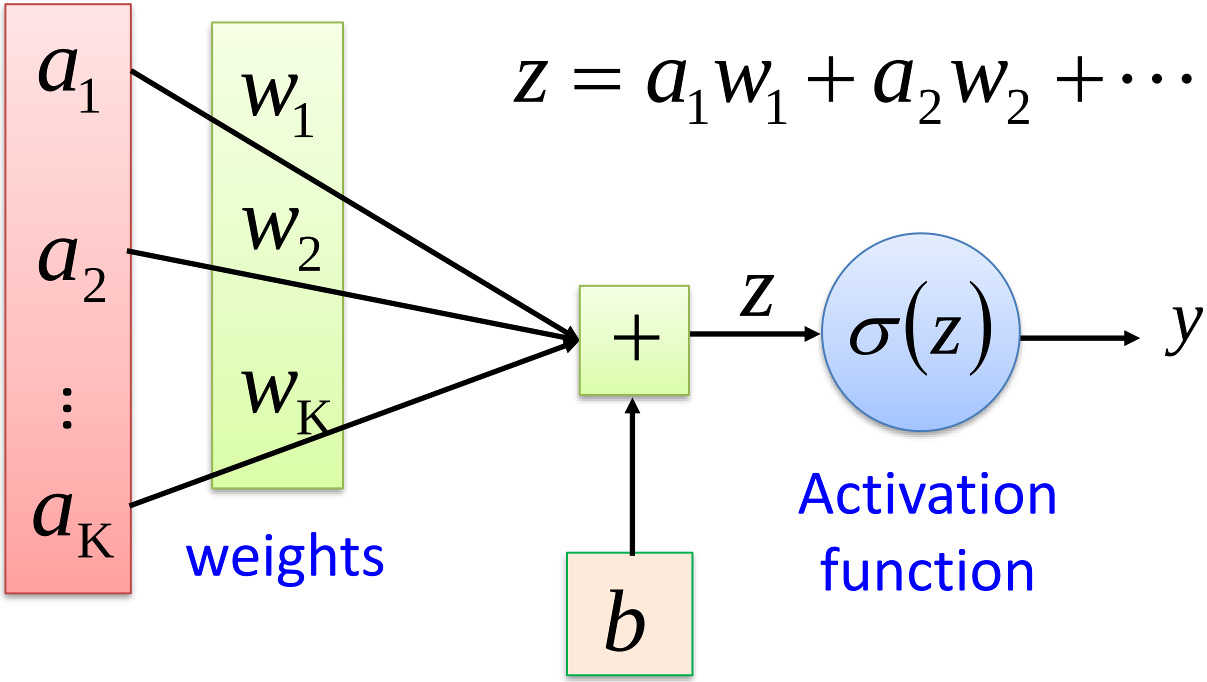
Output Layer

Deep means many hidden layers

Neural Network

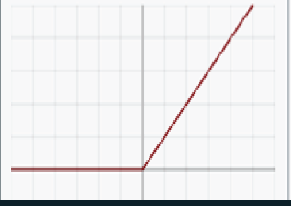
Neuron $f: R^K \rightarrow R$

$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$



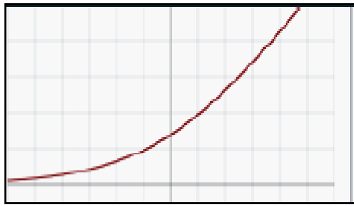
bias

Activation Function types



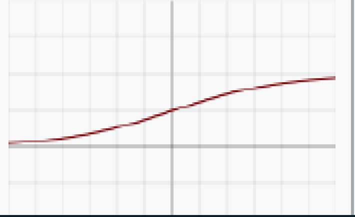
$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

ReLU



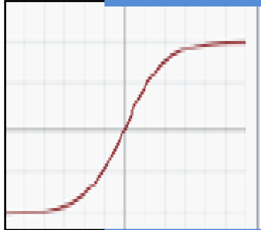
$$f(x) = \ln(1 + e^x)$$

Softplus



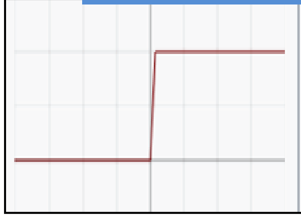
$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid/logistic



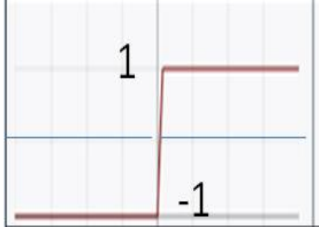
$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Tanh



$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

Binary



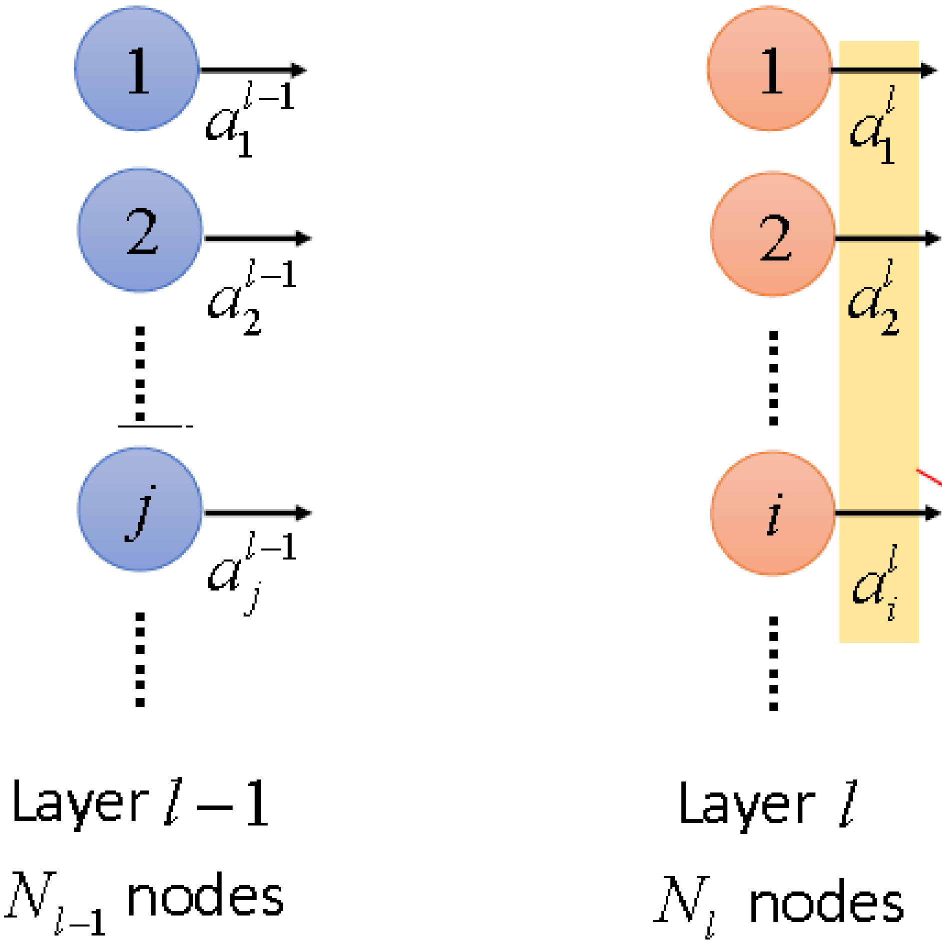
$$f(x) = \begin{cases} -1 & \text{for } x < 0 \\ 1 & \text{for } x > 0 \end{cases}$$

Signum

$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad \text{for } i = 1, \ldots, J$$

Softmax

Fully Connected Layer Vanilla



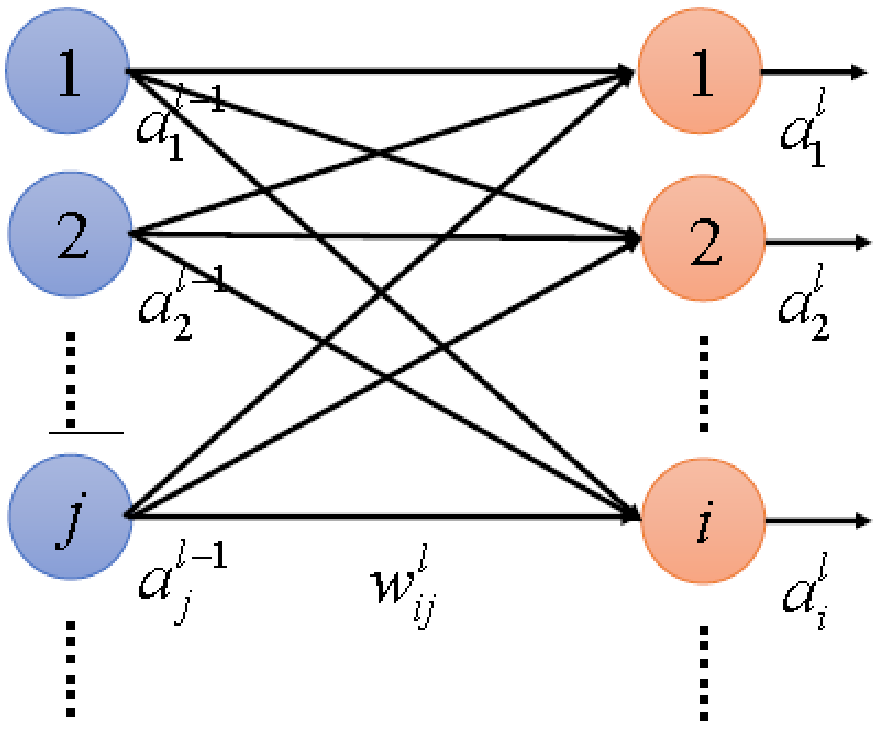
Output of a neuron:

a_i^l → Layer l, → Neuron i

Output of one layer:

a^l : a vector

Fully Connected Layer



Layer $l-1$
N_{l-1} nodes

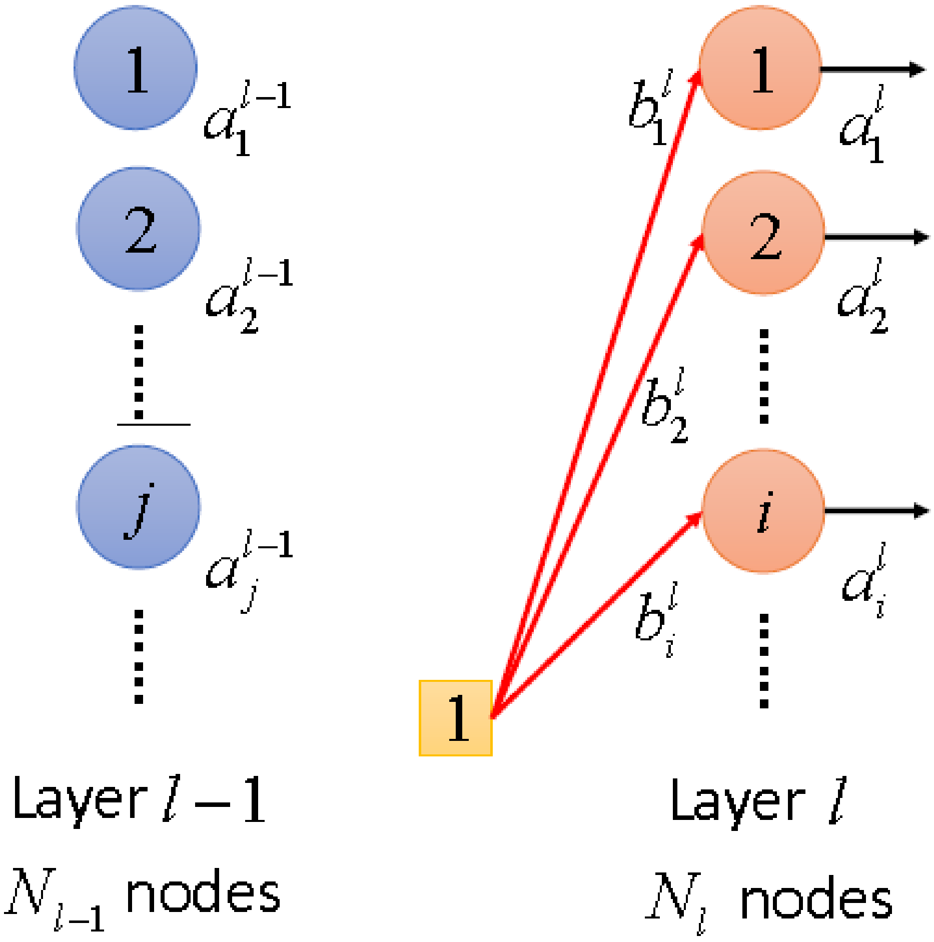
Layer l
N_l nodes

w_{ij}^{l} → Layer $l-1$ to Layer l

↓ from neuron j (Layer $l-1$)
to neuron i (Layer l)

$$W^l = \overbrace{\begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \\ & & \end{bmatrix}}^{N_{l-1}} \Bigg\} N_l$$

Fully Connected Layer

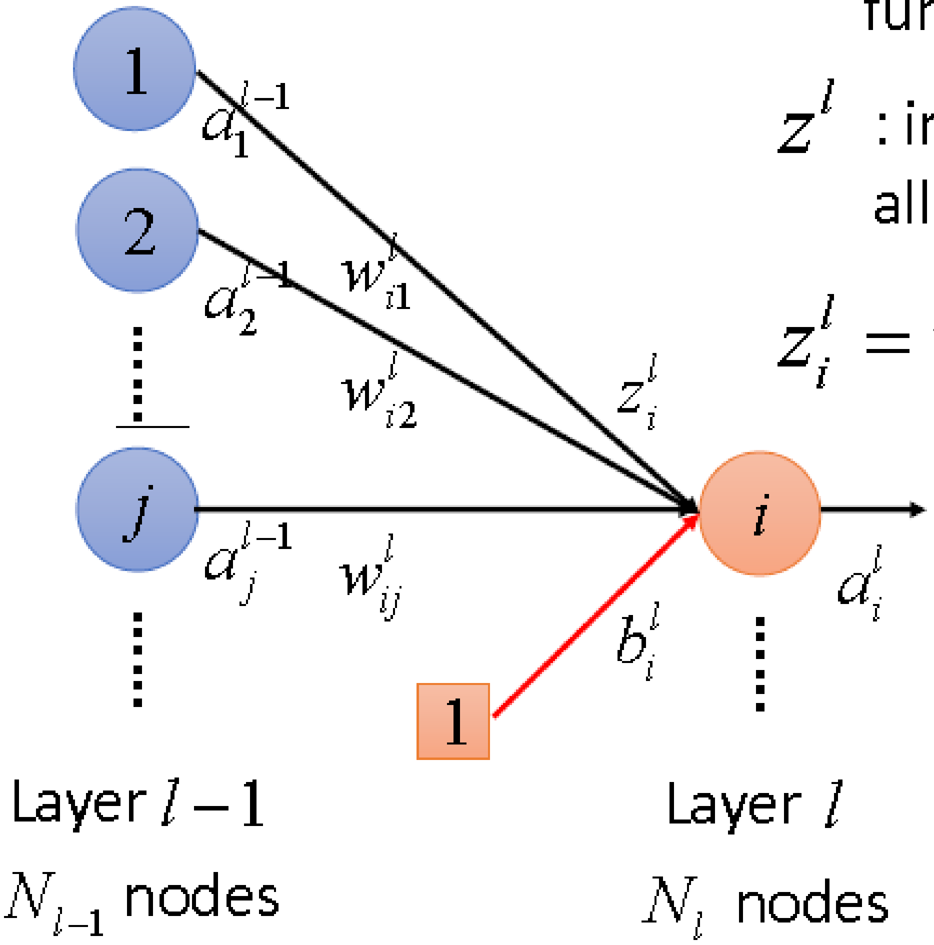


b_i^l : bias for neuron i at layer l

$$b^l = \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

bias for all neurons in layer l

Fully Connected Layer



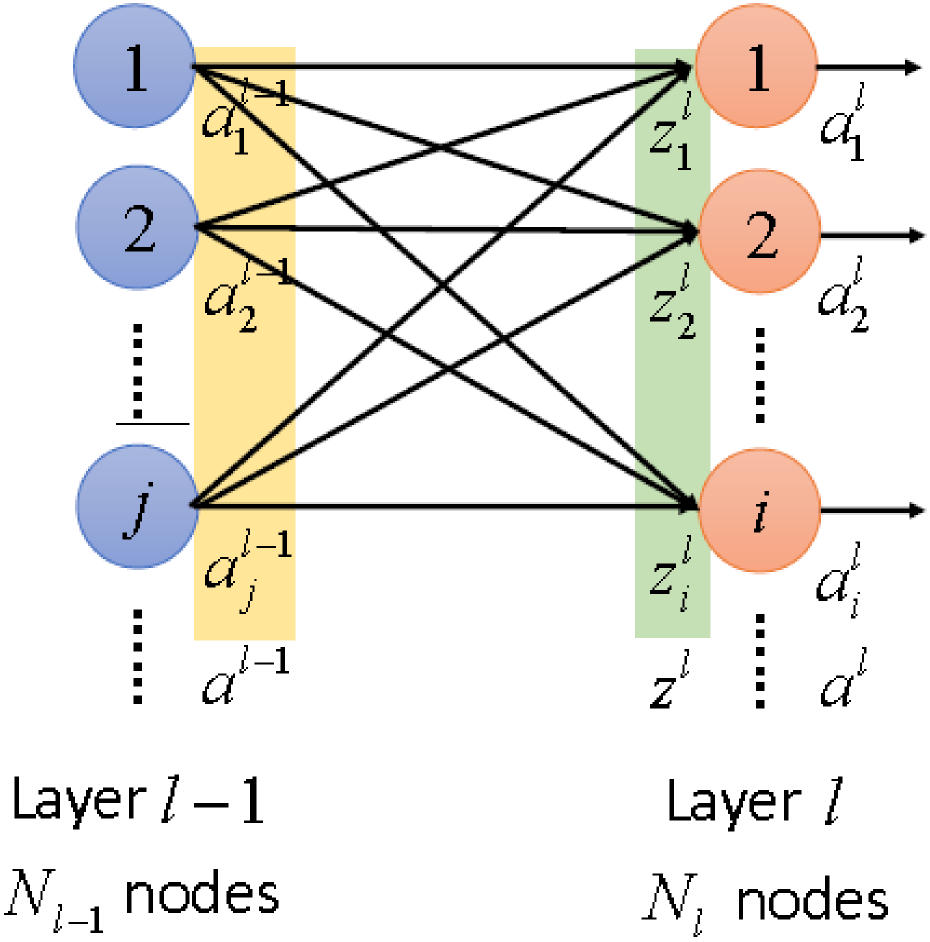
z_i^l : input of the activation function for neuron i at layer l

z^l : input of the activation function all the neurons in layer l

$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} \ldots + b_i^l$$

$$z_i^l = \sum_{j=1}^{N_{l-1}} w_{ij}^l a_j^{l-1} + b_i^l$$

Relations between Layer Outputs



$$z_1^l = w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \cdots + b_1^l$$

$$z_2^l = w_{21}^l a_1^{l-1} + w_{22}^l a_2^{l-1} + \cdots + b_2^l$$

$$\vdots$$

$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \cdots + b_i^l$$

$$\vdots$$

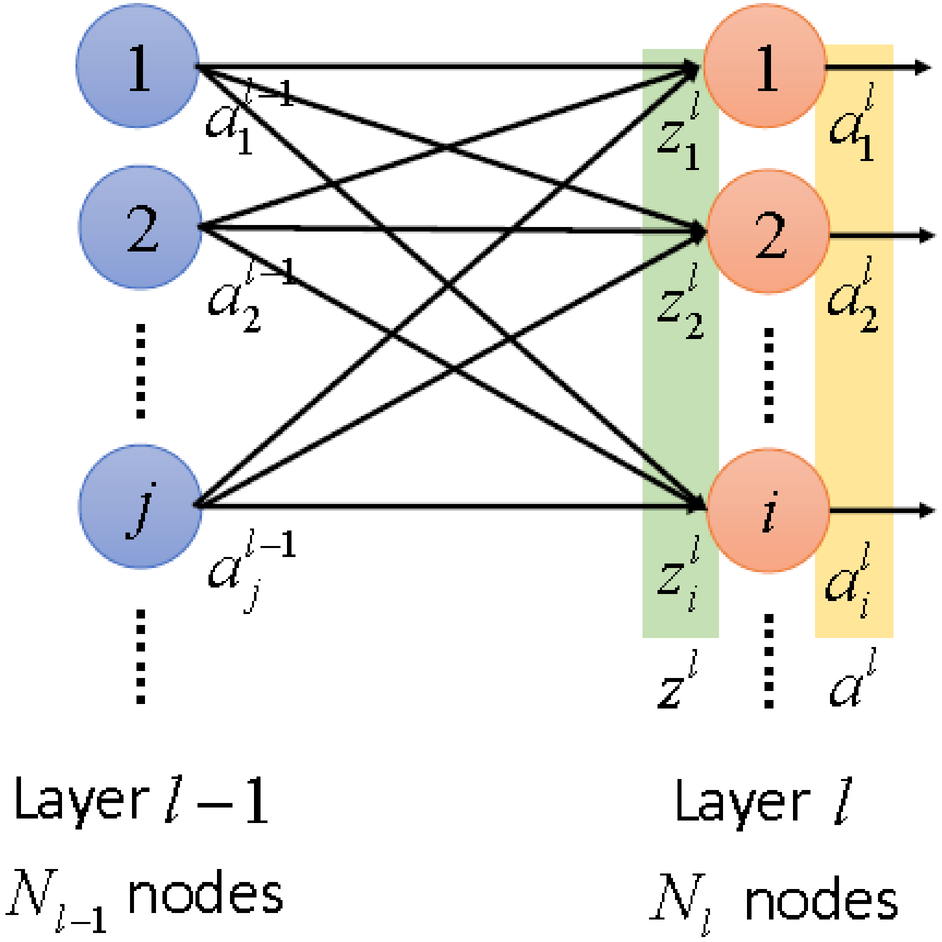
$$\Downarrow$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$\Downarrow$$

$$z^l = W^l a^{l-1} + b^l$$

Relations between Layer Outputs

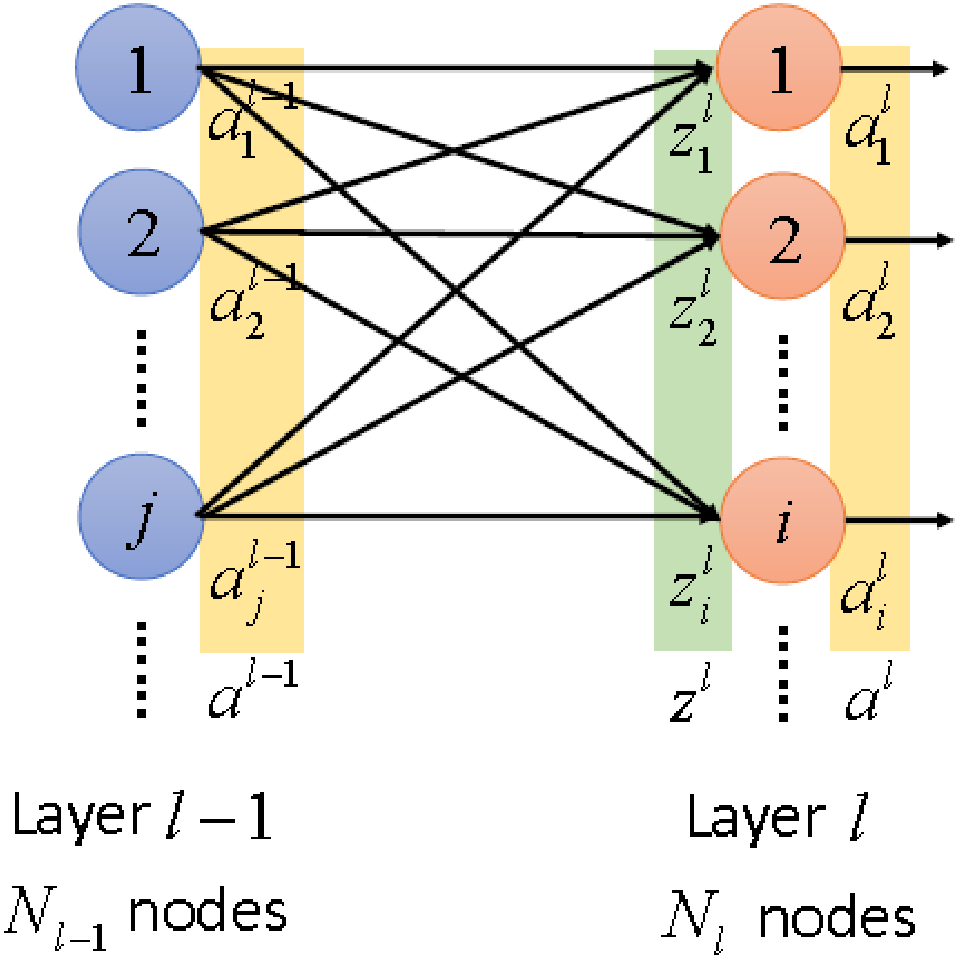


$$a_i^l = \sigma\left(z_i^l\right)$$

$$\begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma\left(z_1^l\right) \\ \sigma\left(z_2^l\right) \\ \vdots \\ \sigma\left(z_i^l\right) \\ \vdots \end{bmatrix}$$

$$a^l = \sigma\left(z^l\right)$$

Relations between Layer Outputs

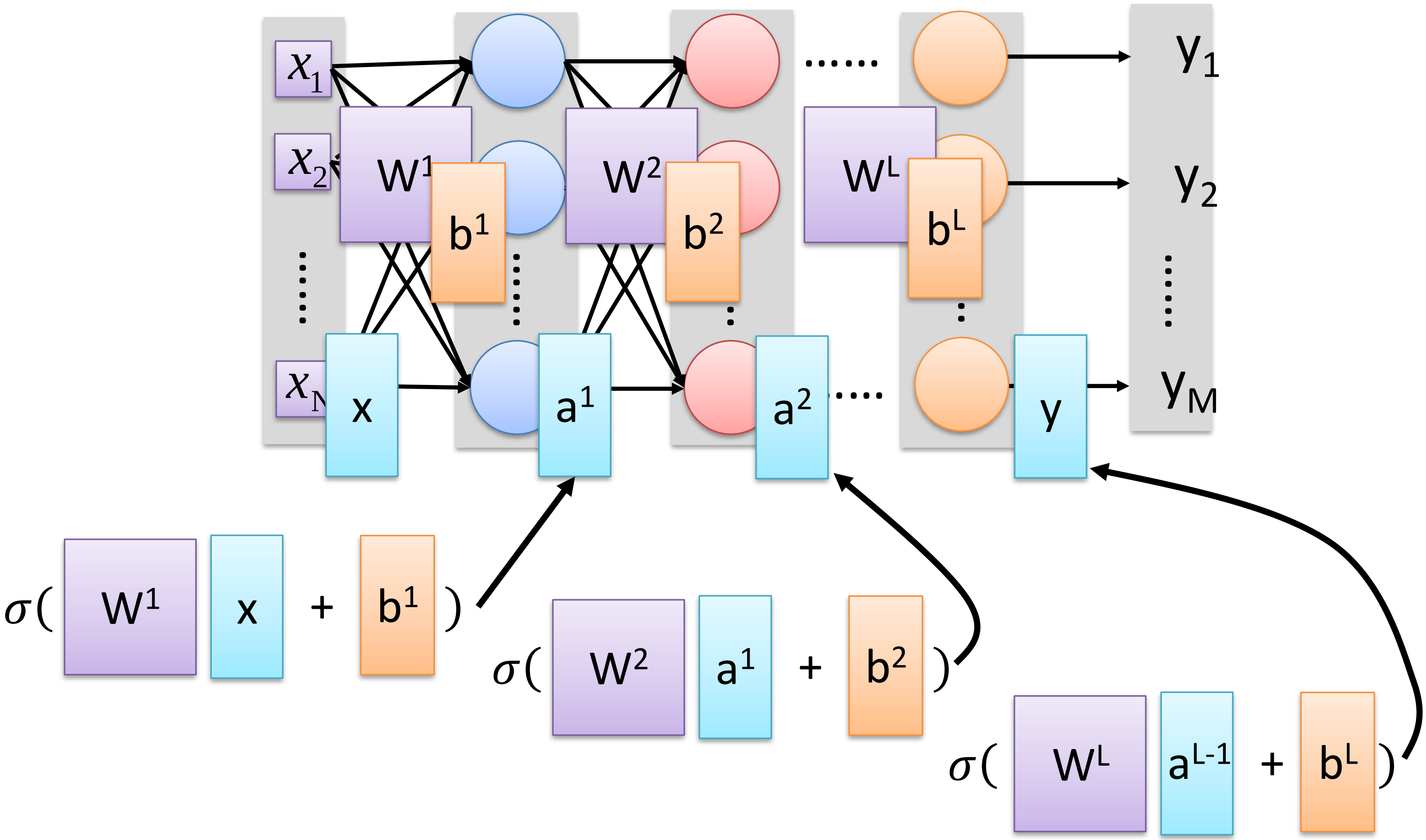


$$z^l = W^l a^{l-1} + b^l$$

$$a^l = \sigma\left(z^l\right)$$

$$a^l = \sigma\left(W^l a^{l-1} + b^l\right)$$

Neural Network

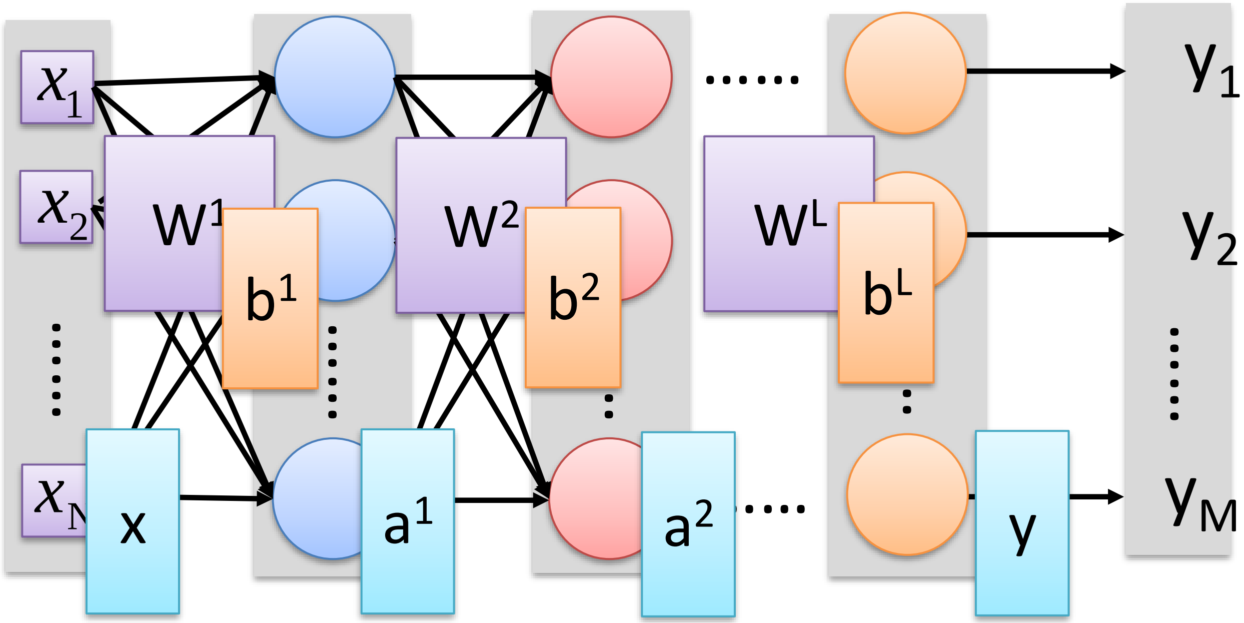


$\sigma(W^1 \; x + b^1)$

$\sigma(W^2 \; a^1 + b^2)$

$\sigma(W^L \; a^{L-1} + b^L)$

Neural Network



$$y = f(x)$$

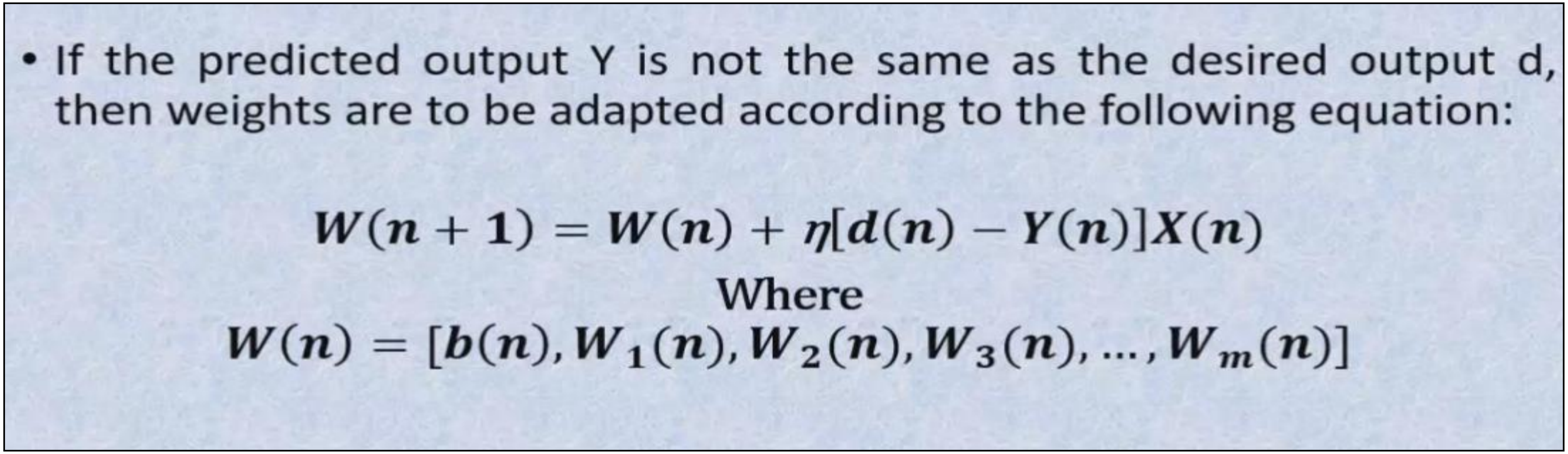
$$= \sigma(W^L \cdots \sigma(W^2 \, \sigma(W^1 \, x + b^1) + b^2) \cdots + b^L)$$

Neural Network training steps

1. Weight Initialization
2. Inputs Application
3. Sum of inputs - Weights product
4. Activation functions
5. Weights Adaptations
6. Back to step 2

Regarding 5th step: Weights Adaptation

First method:



- If the predicted output Y is not the same as the desired output d, then weights are to be adapted according to the following equation:

$$W(n+1) = W(n) + \eta[d(n) - Y(n)]X(n)$$

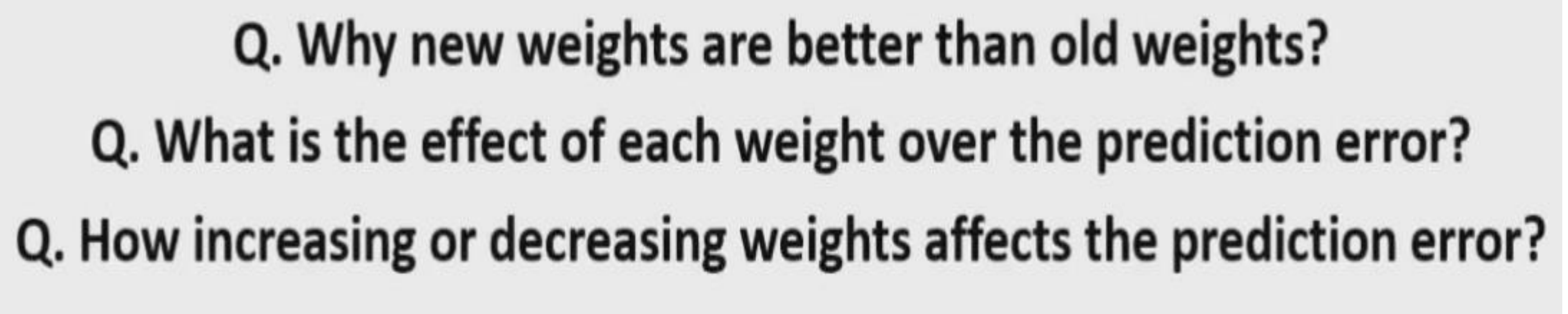
Where

$$W(n) = [b(n), W_1(n), W_2(n), W_3(n), \dots, W_m(n)]$$

Learning Rate η

$0 \leq \eta \leq 1$

$0 \leq \alpha \leq 1$



Q. Why new weights are better than old weights?

Q. What is the effect of each weight over the prediction error?

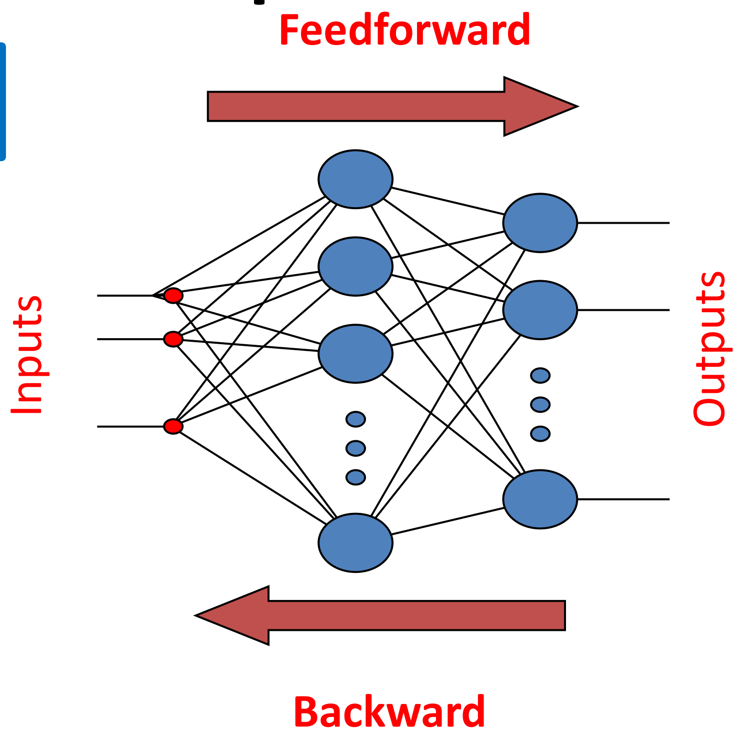
Q. How increasing or decreasing weights affects the prediction error?

Regarding 5th step: Weights Adaptation

second method: Back propagation

- **Fowrward VS Backword passes**

The Backpropagation algorithm is a sensible approach for dividing the contribution of each weight.



Fowrward	Input weights	→	SOP	→	Prediction Output	→	Prediction Error
backward	Prediction Error	→	Prediction Output	→	SOP	→	Input weights

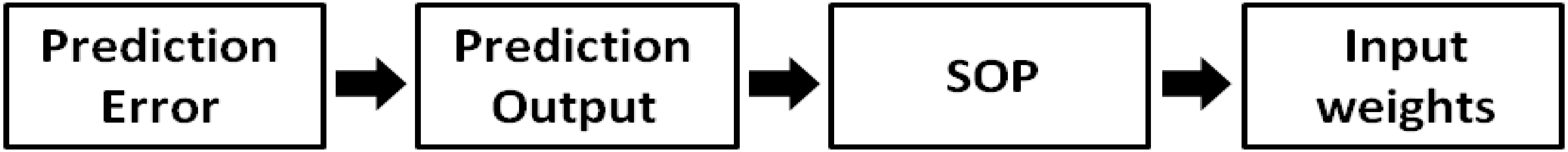
Regarding 5th step: Weights Adaptation

second method: Back propagation

- **Backword pass**

What is the change in prediction Error (E) given the change in weight (W) ?

Get partial derivative of E W.R.T W $\frac{\partial E}{\partial W}$



Prediction Error → Prediction Output → SOP → Input weights

$E = \frac{1}{2}(d - y)^2$ $\qquad f(s) = \frac{1}{1 + \mathrm{e}^{-s}}$ $\qquad s = \sum_{j}^{m} x_i \, w_{ji} + b_i$ $\qquad w_1, w_2$

d (desired output) Const

y (predicted output)

s (Sum Of Product SOP)

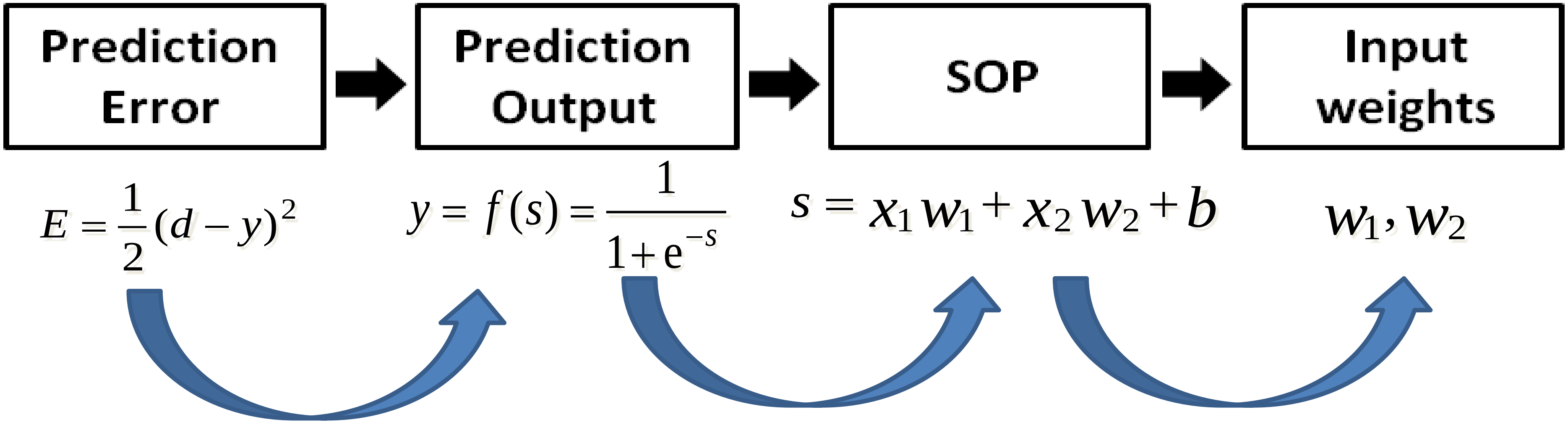
$$E = \frac{1}{2}\left(d - \frac{1}{e^{-\sum_{j}^{n} x_i w_{ij} + b_i}}\right)^2$$

Regarding 5th step: Weights Adaptation

second method: Back propagation

Chain Rule

- **Weight derivative**



Prediction Error → Prediction Output → SOP → Input weights

$$E = \frac{1}{2}(d-y)^2 \qquad y = f(s) = \frac{1}{1+e^{-s}} \qquad s = x_1 w_1 + x_2 w_2 + b \qquad w_1, w_2$$

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial s} \times \frac{\partial s}{\partial w_1}, \frac{\partial s}{\partial w_2}$$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y} x \frac{\partial y}{\partial s} x \frac{\partial s}{\partial w_1}$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y} x \frac{\partial y}{\partial s} x \frac{\partial s}{\partial w_2}$$

Regarding 5th step: Weights Adaptation

second method: Back propagation

- **Weight derivative**

$$\frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(d-y)^2 = y-d$$

$$\frac{\partial y}{\partial s} = \frac{\partial}{\partial s}\frac{1}{1+\mathrm{e}^{-s}} = \frac{1}{1+\mathrm{e}^{-s}}(1-\frac{1}{1+\mathrm{e}^{-s}})$$

$$\frac{\partial s}{\partial w_1} = \frac{\partial}{\partial w_1} x_1 w_1 + x_2 w_2 + b \; = x_1$$

$$\frac{\partial s}{\partial w_2} = \frac{\partial}{\partial w_2} x_1 w_1 + x_2 w_2 + b \; = x_2$$

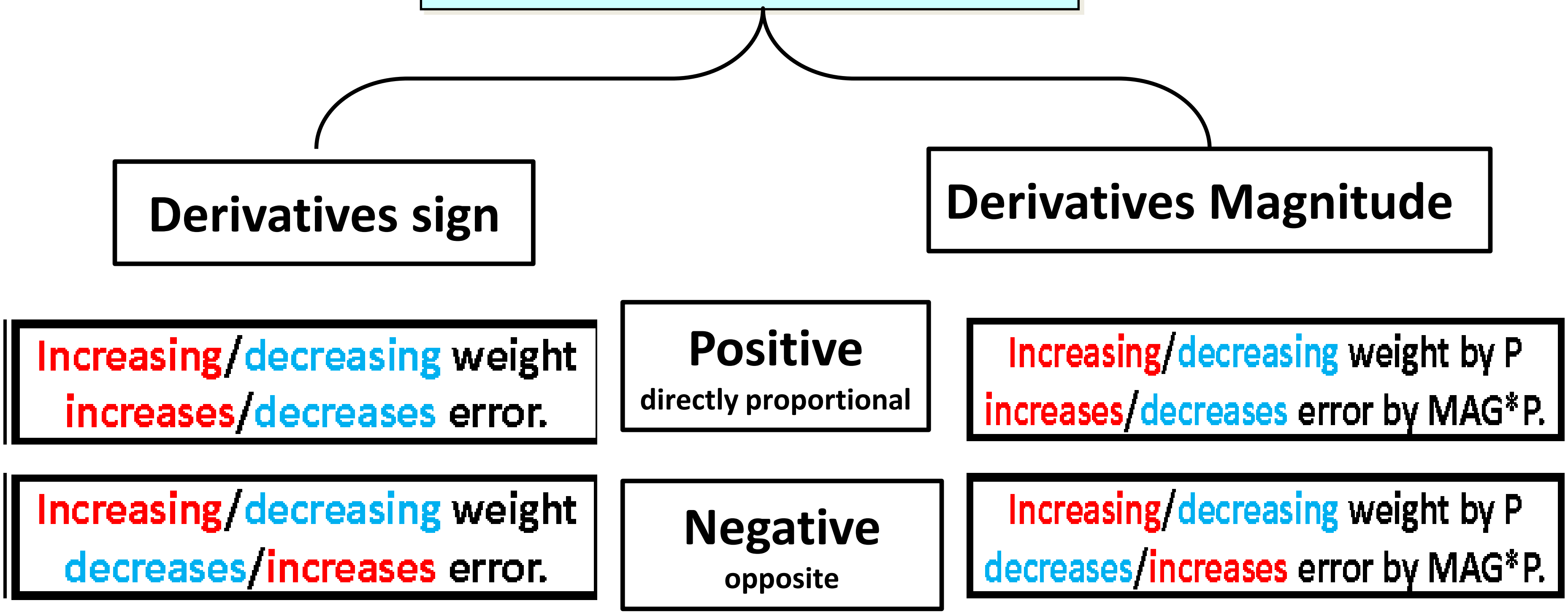
$$\frac{\partial E}{\partial w_i} = (y-d)\frac{1}{1+\mathrm{e}^{-s}}(1-\frac{1}{1+\mathrm{e}^{-s}})\, x_i$$

Regarding 5th step: Weights Adaptation

second method: Back propagation

- interpreting derivatives ∇W

$$\frac{\partial E}{\partial w_i} = (y-d)\,\frac{\partial f(s)}{\partial s}\;x_i$$



Derivatives sign

Derivatives Magnitude

Increasing/decreasing weight increases/decreases error.

Positive
directly proportional

Increasing/decreasing weight by P increases/decreases error by MAG*P.

Increasing/decreasing weight decreases/increases error.

Negative
opposite

Increasing/decreasing weight by P decreases/increases error by MAG*P.

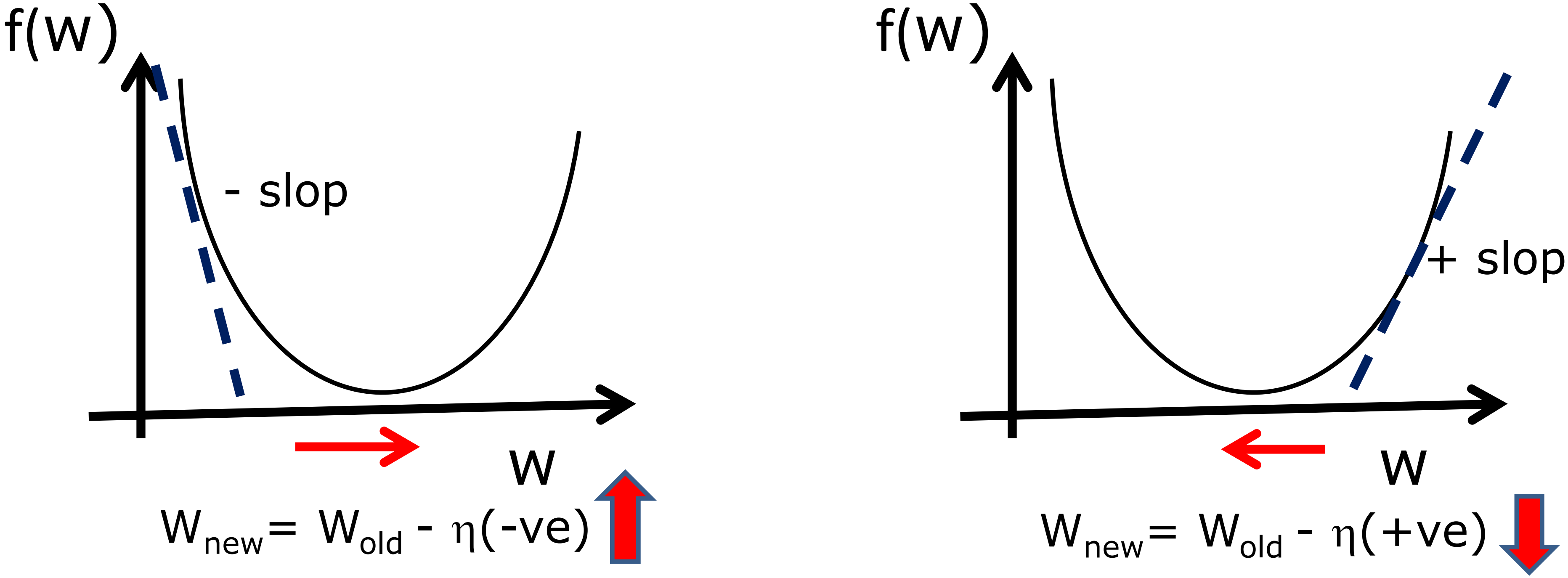
Regarding 5th step: Weights Adaptation

second method: Back propagation

- **Update the Weights**

In order to update the weights , use the Gradient Descent

$$W_{inew} = W_{iold} - \eta * \frac{\partial E}{\partial W_i}$$



f(w)
- slop
w
$W_{new} = W_{old} - \eta(-ve)$
f(w)
+ slop
w
$W_{new} = W_{old} - \eta(+ve)$

Convolution Neural Network
CNN

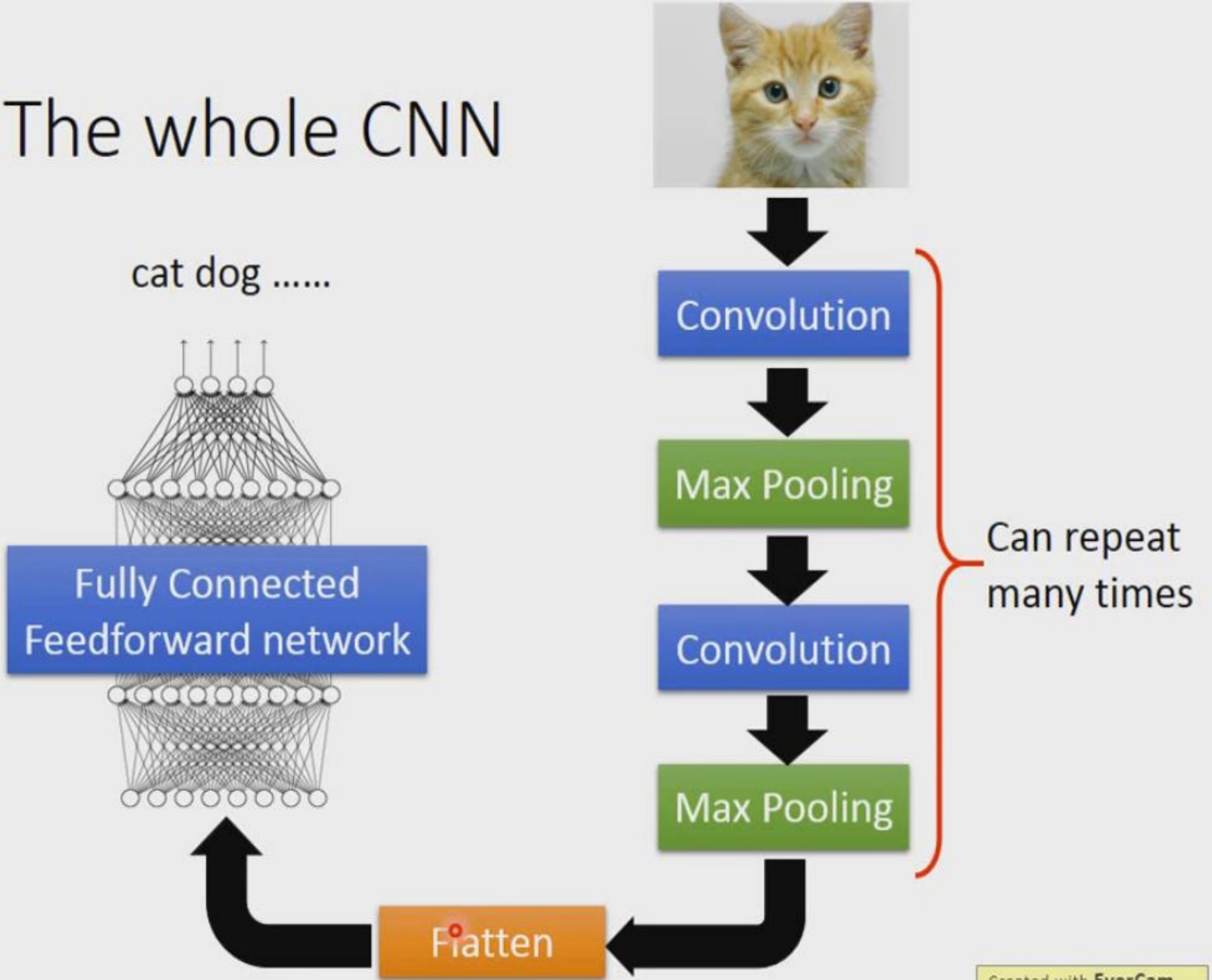
introduction

➢ Convolutional neural networks (or convnets for short) are used in situations where data can be expressed as a "map" wherein the proximity between two data points indicates how related they are.

➢ Convnets contain one or more of each of the following layers:

1. convolution layer
2. ReLU (rectified linear units) layer (element wise threshold)
3. pooling layer
4. fully connected layer
5. loss layer (during the training process)

The whole CNN



cat dog ……

Fully Connected Feedforward network

Convolution

Max Pooling

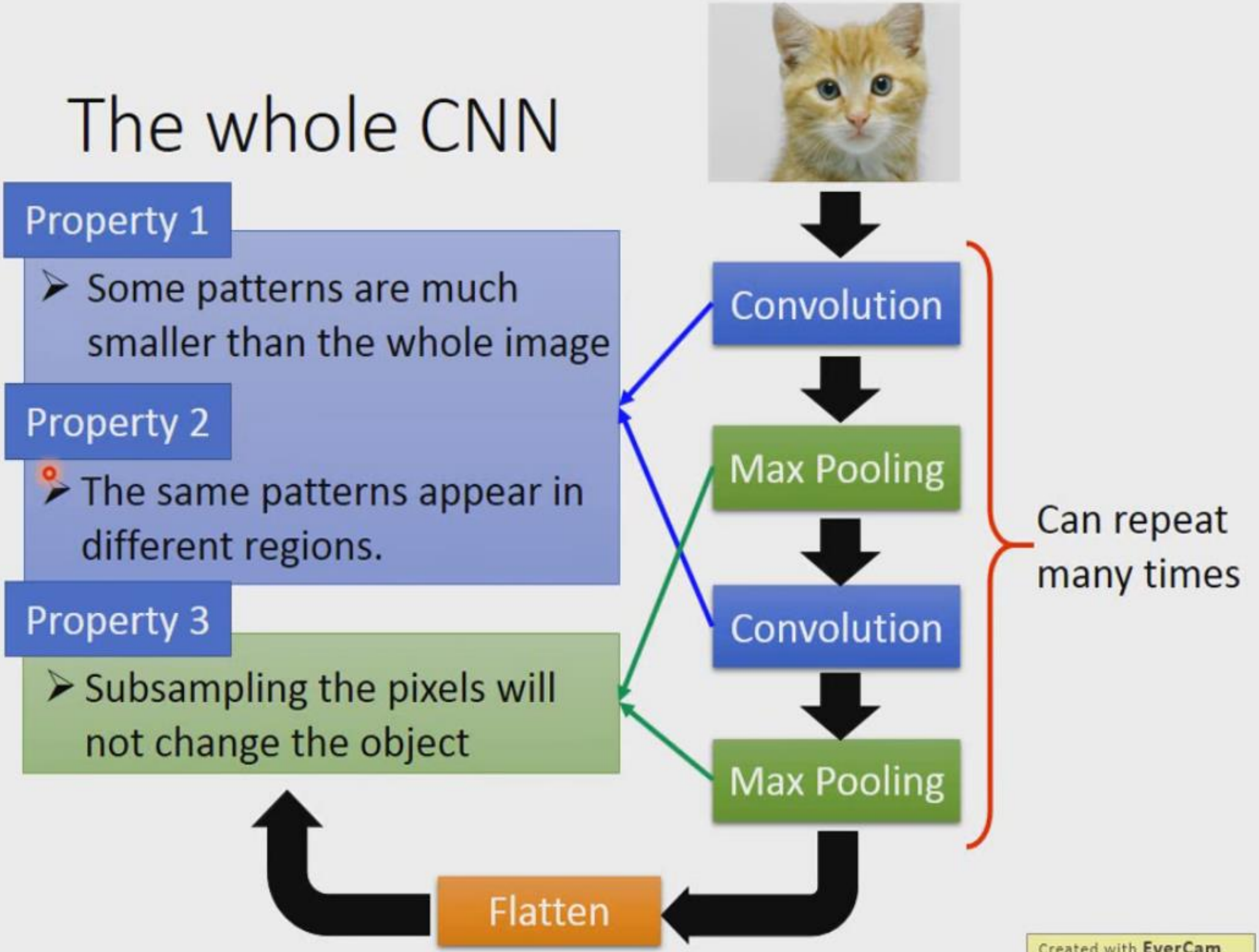
Convolution

Max Pooling

Can repeat many times

Flatten

The whole CNN



Property 1

- Some patterns are much smaller than the whole image

Property 2

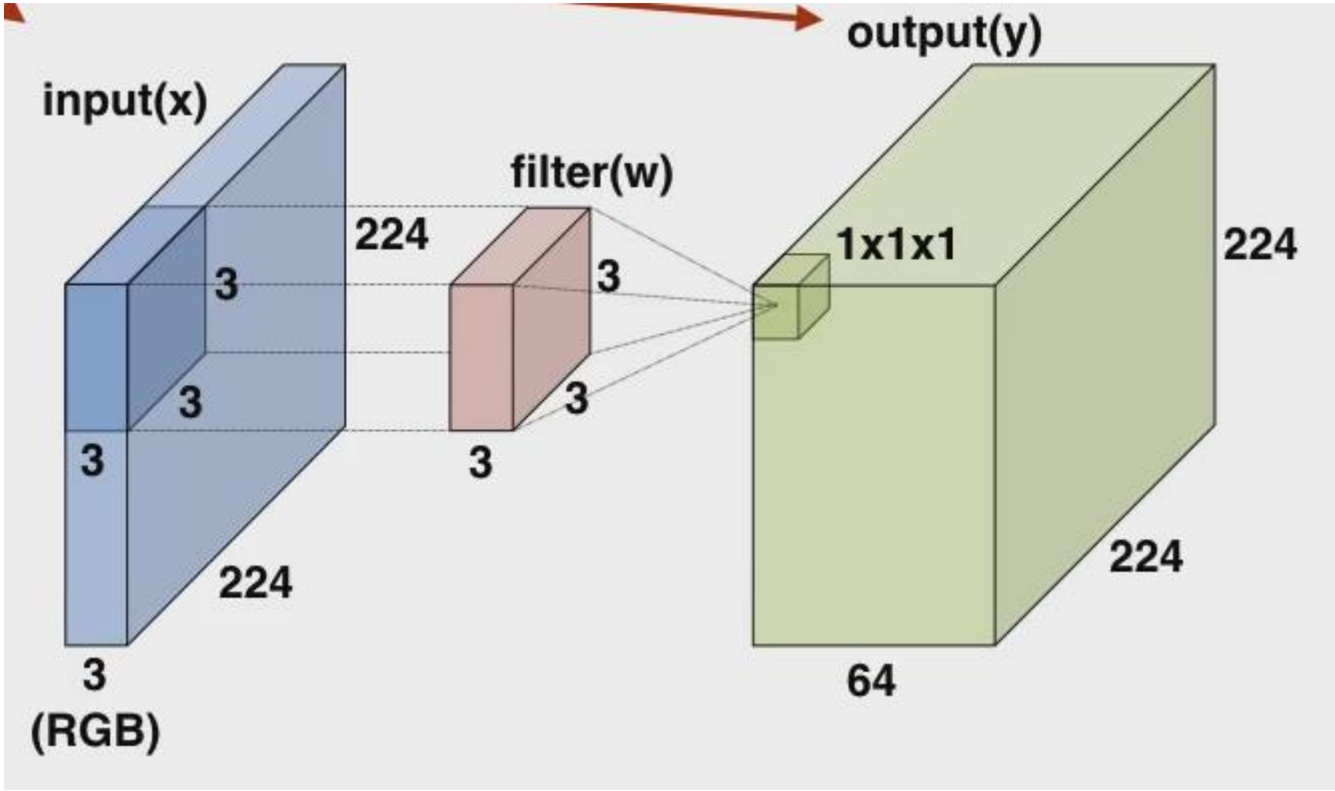
- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

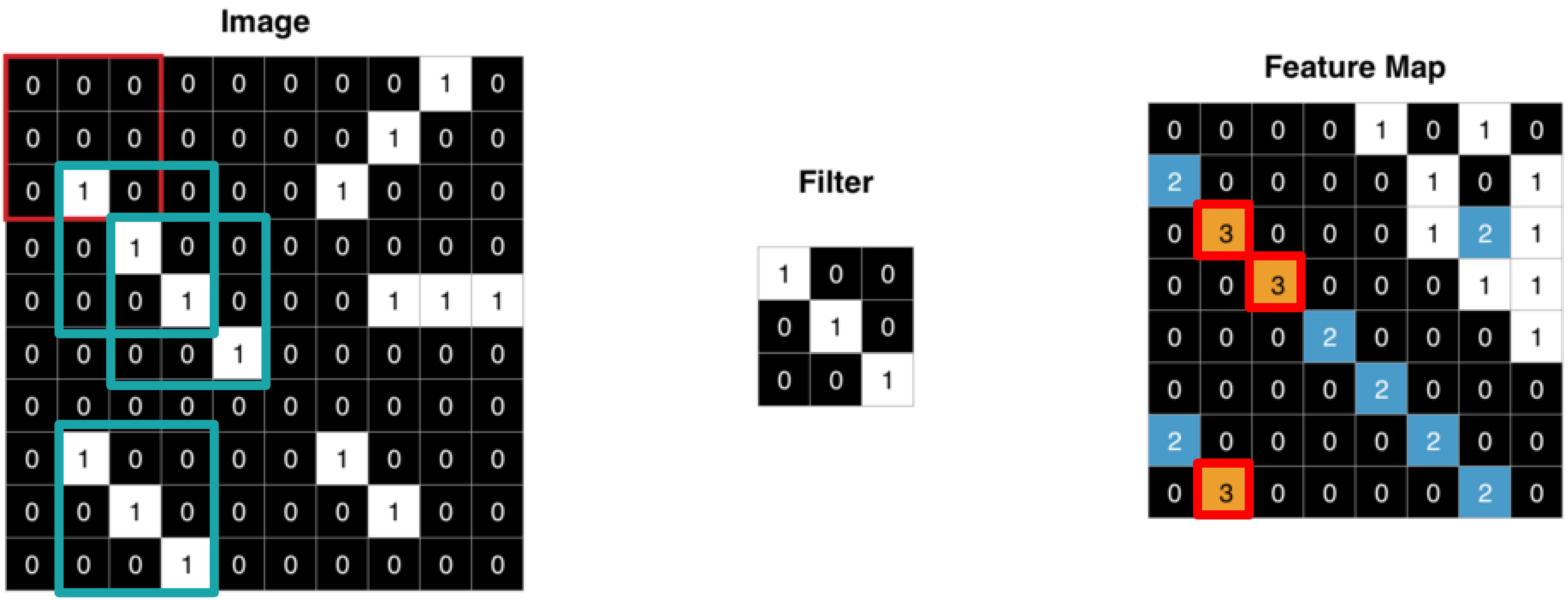
1- Convolution layer

a convnet processes an image using a matrix of weights called filters (or features) that detect specific attributes such as diagonal edges, vertical edges, etc. Moreover, as the image progresses through each layer, the filters are able to recognize more complex attributes.



Convolution layer

The convolution layer is always the first step in a convnet. Let's say we have a 10 x 10 pixel image, here represented by a 10 x 10 x 1 matrix of numbers:



Image

0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1	1
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0

Filter

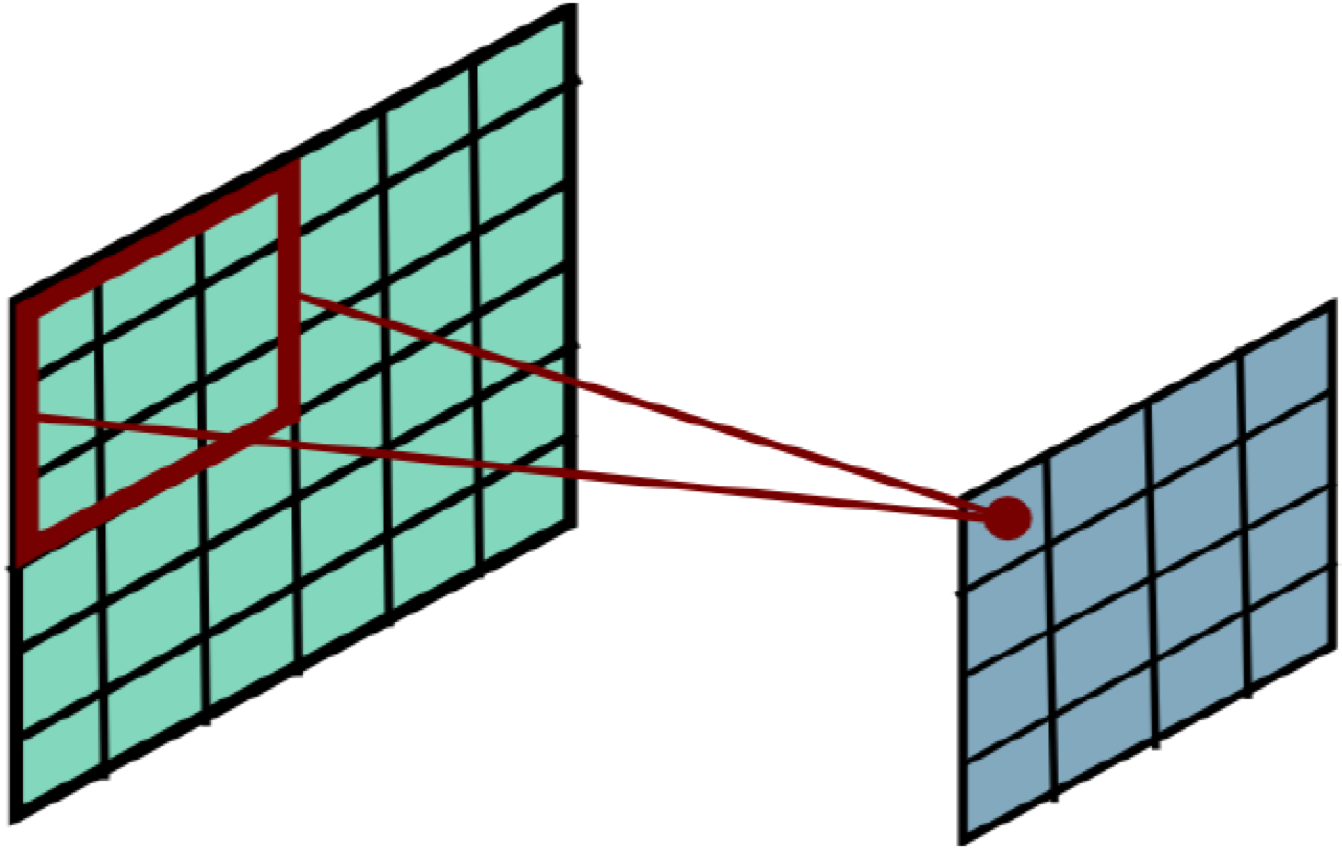
1	0	0
0	1	0
0	0	1

Feature Map

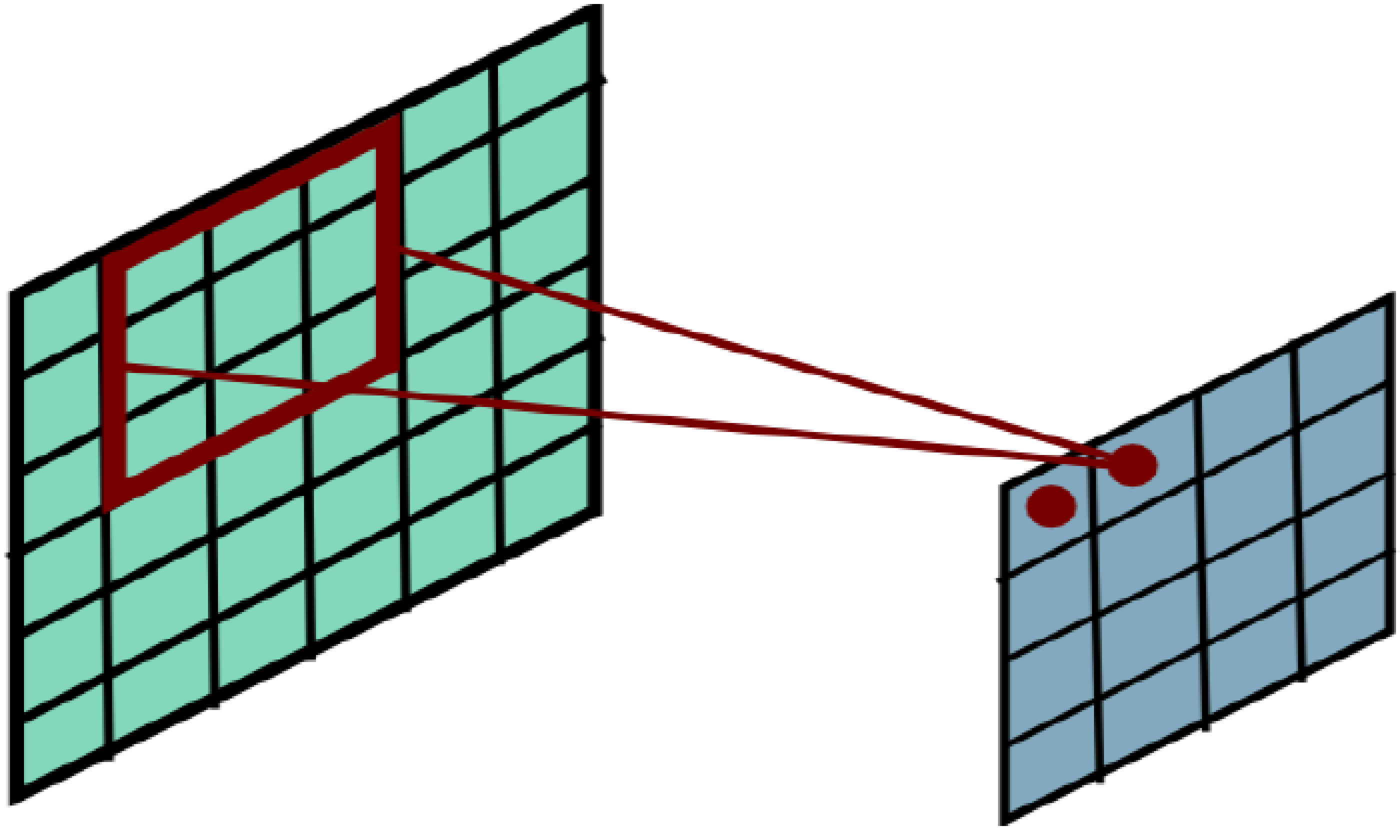
0	0	0	0	1	0	1	0
2	0	0	0	0	1	0	1
0	3	0	0	0	1	2	1
0	0	3	0	0	0	1	1
0	0	0	2	0	0	0	1
0	0	0	0	2	0	0	0
2	0	0	0	0	2	0	0
0	3	0	0	0	0	2	0

0 x 1 + 0 x 0 + 0 x 0 + 0 x 0 + 0 x 1 + 0 x 0 + 0 x 0 + 1 x 0 + 0 x 1 = 0

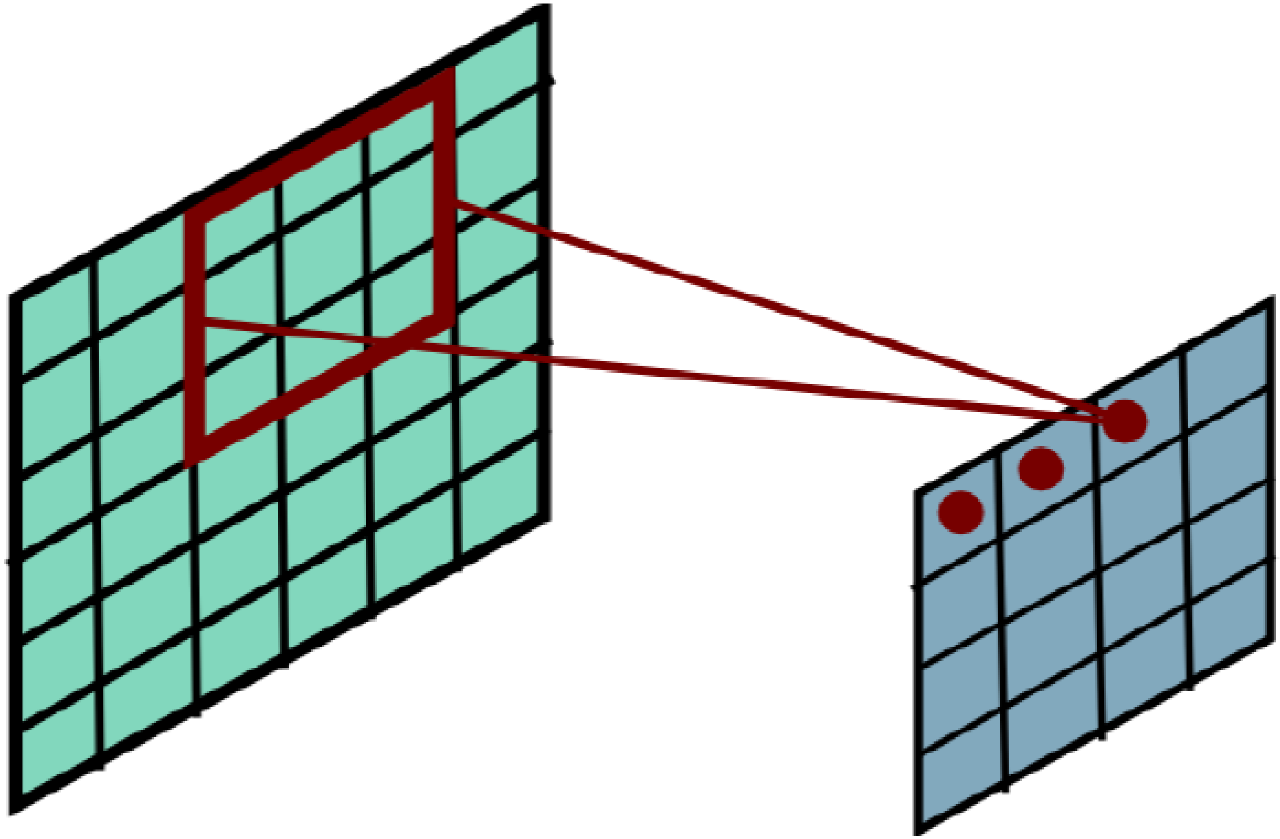
Convolutional Layer



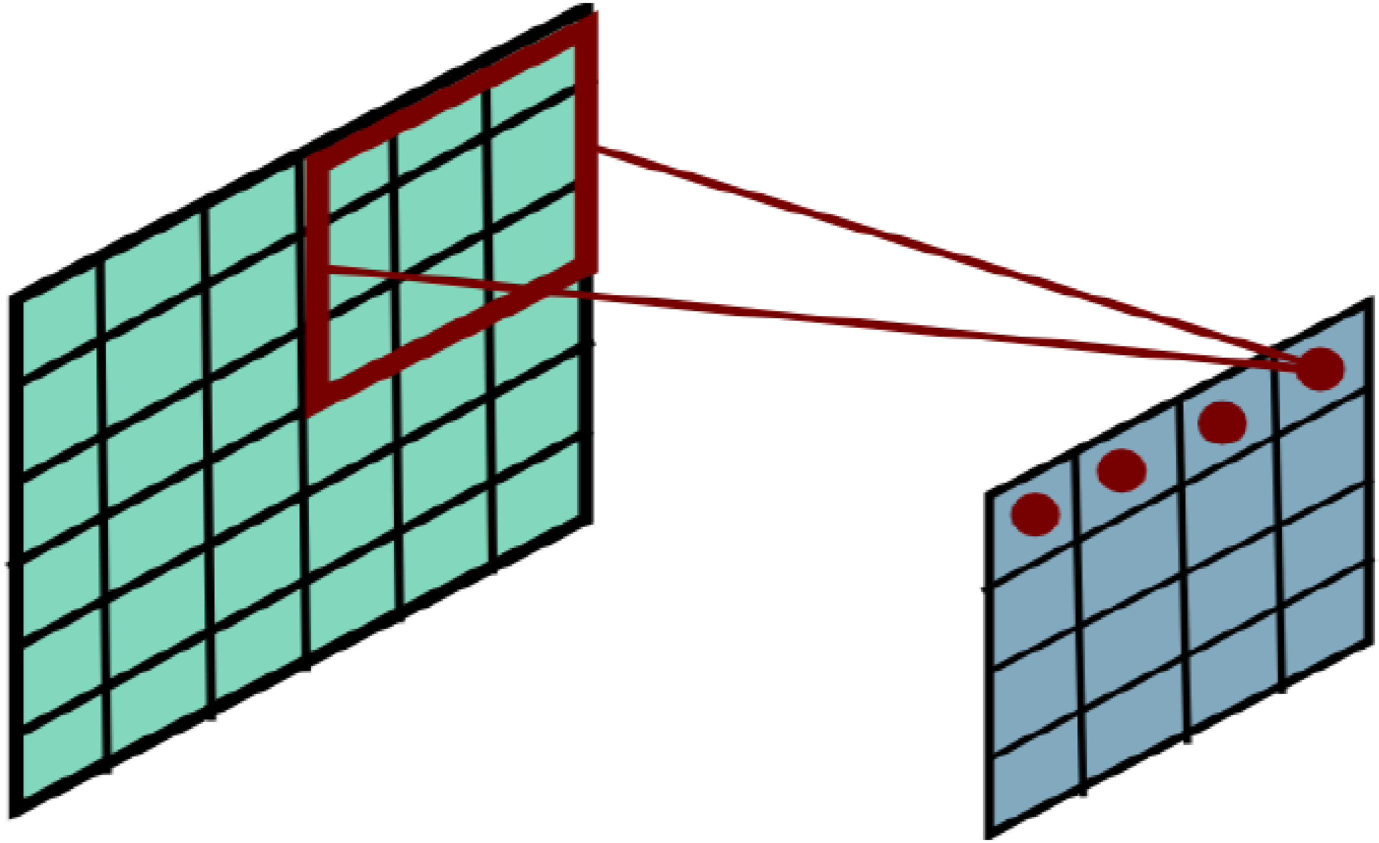
Convolutional Layer



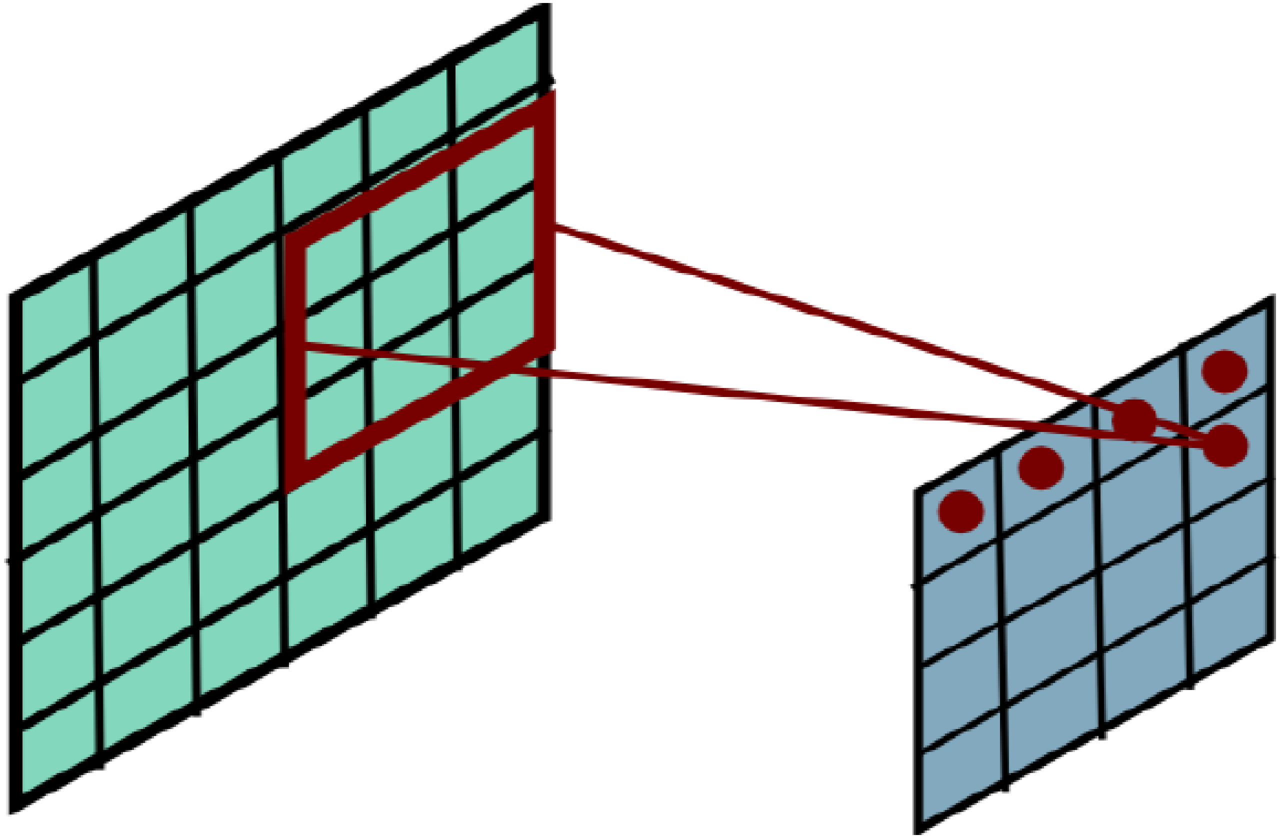
Convolutional Layer



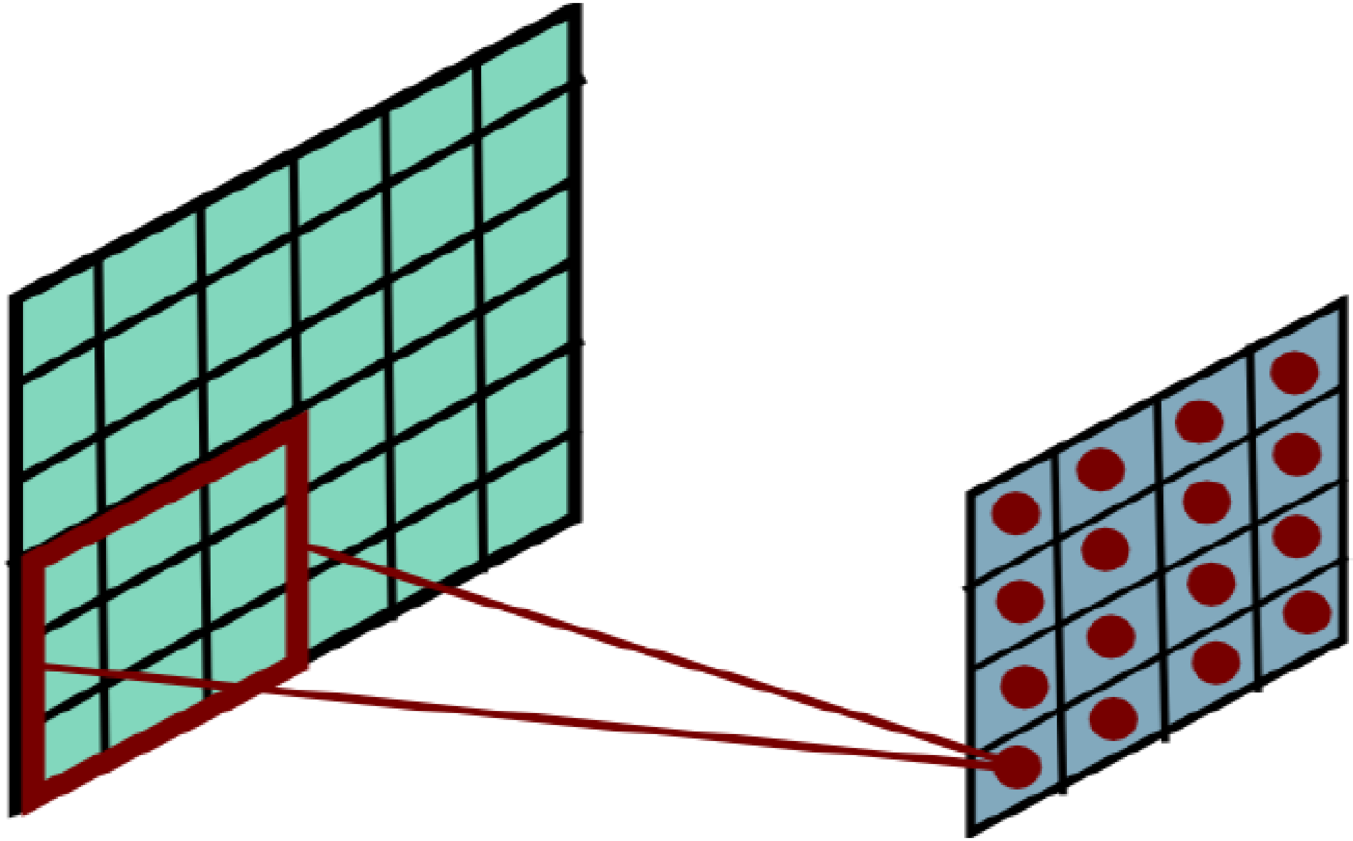
Convolutional Layer



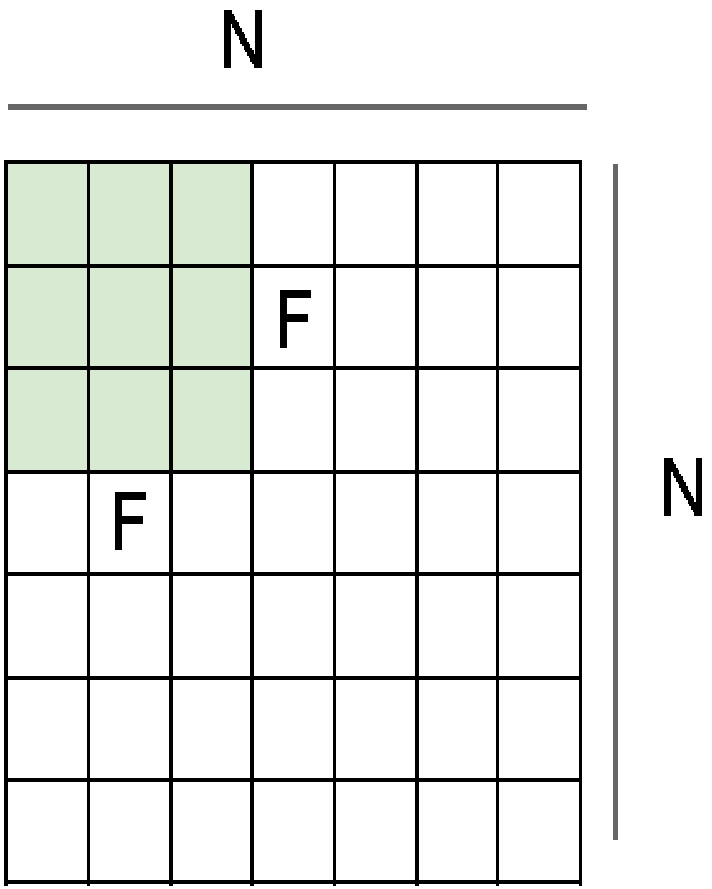
Convolutional Layer



Convolutional Layer



stride



Output size:

(N - F) / stride + 1

e.g. N = 7, F = 3:

stride 1 => (7 - 3)/1 + 1 = 5

stride 2 => (7 - 3)/2 + 1 = 3

stride 3 => (7 - 3)/3 + 1 = 2.33 :\

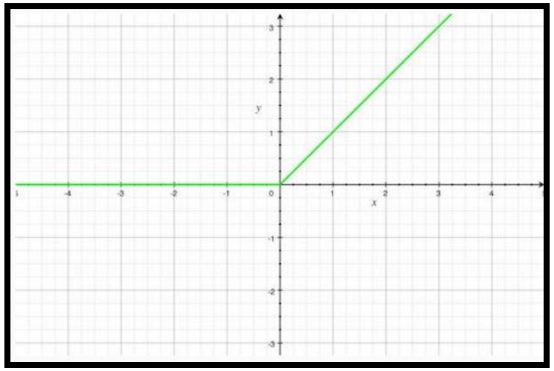
2- ReLU Layer $f(x) = \max(0,x)$

- The ReLU (short for rectified linear units) layer commonly follows the convolution layer.

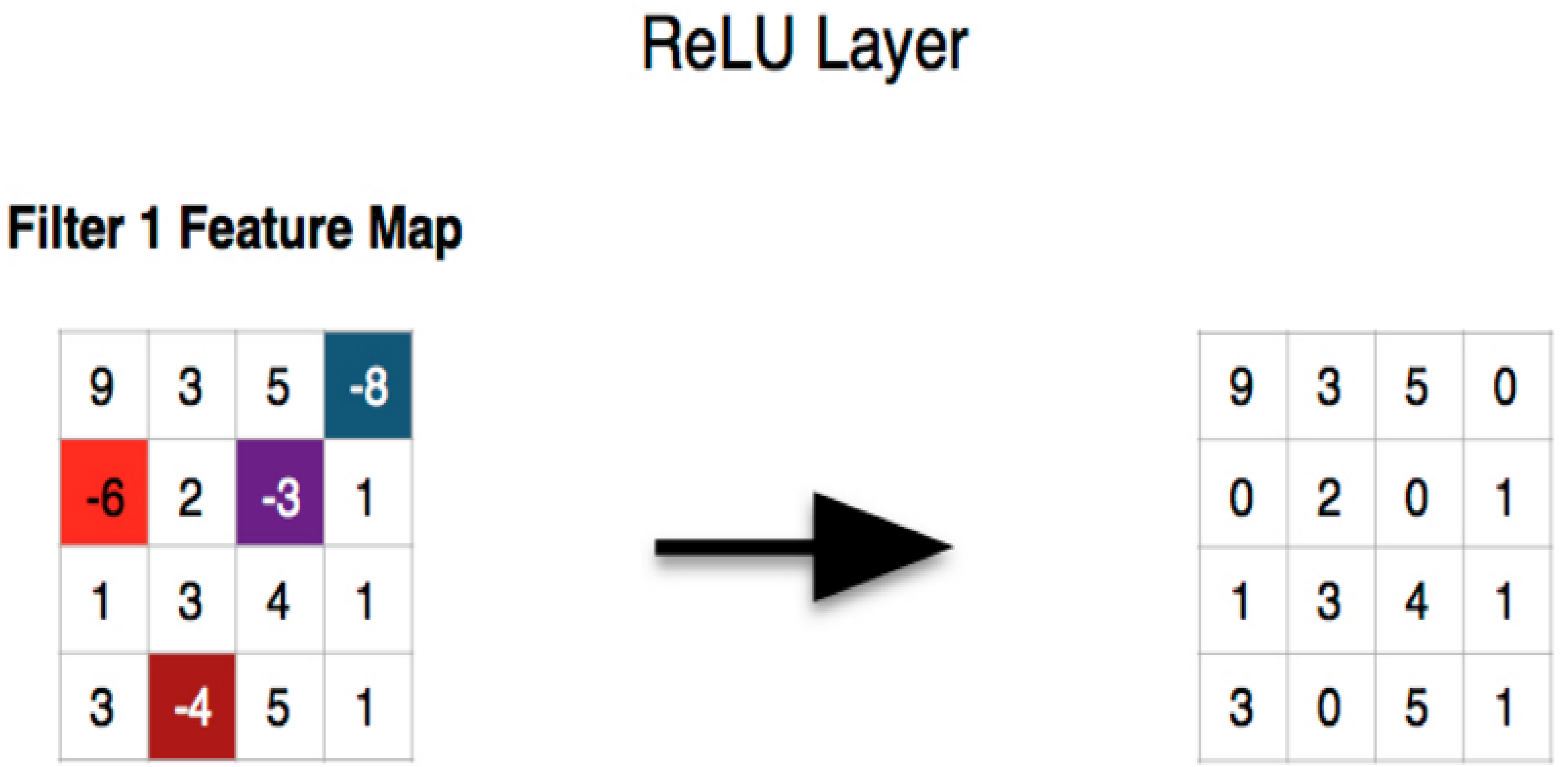
- The addition of the ReLU layer allows the neural network to account for non-linear relationships, i.e. the ReLU layer allows the convnet to account for situations in which the relationship between the pixel value inputs and the convnet output is not linear.

- the convolution operation is a linear one. $\mathbf{y = w_1x_1 + w_2x_2 + w_3x_3 + \ldots}$

- The ReLU function takes a value **x** and returns 0 if **x** is negative and **x** if **x** is positive.



2- ReLU Layer $f(x) = \max(0,x)$



ReLU Layer

Filter 1 Feature Map

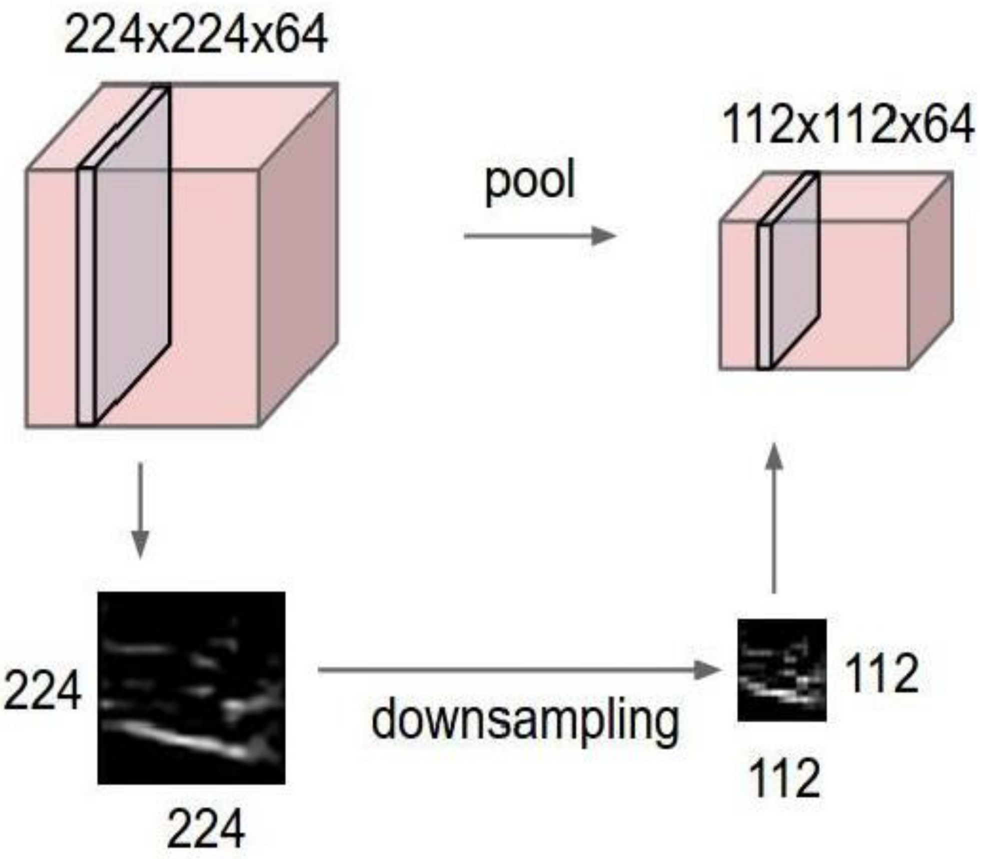
9	3	5	-8
-6	2	-3	1
1	3	4	1
3	-4	5	1

9	3	5	0
0	2	0	1
1	3	4	1
3	0	5	1

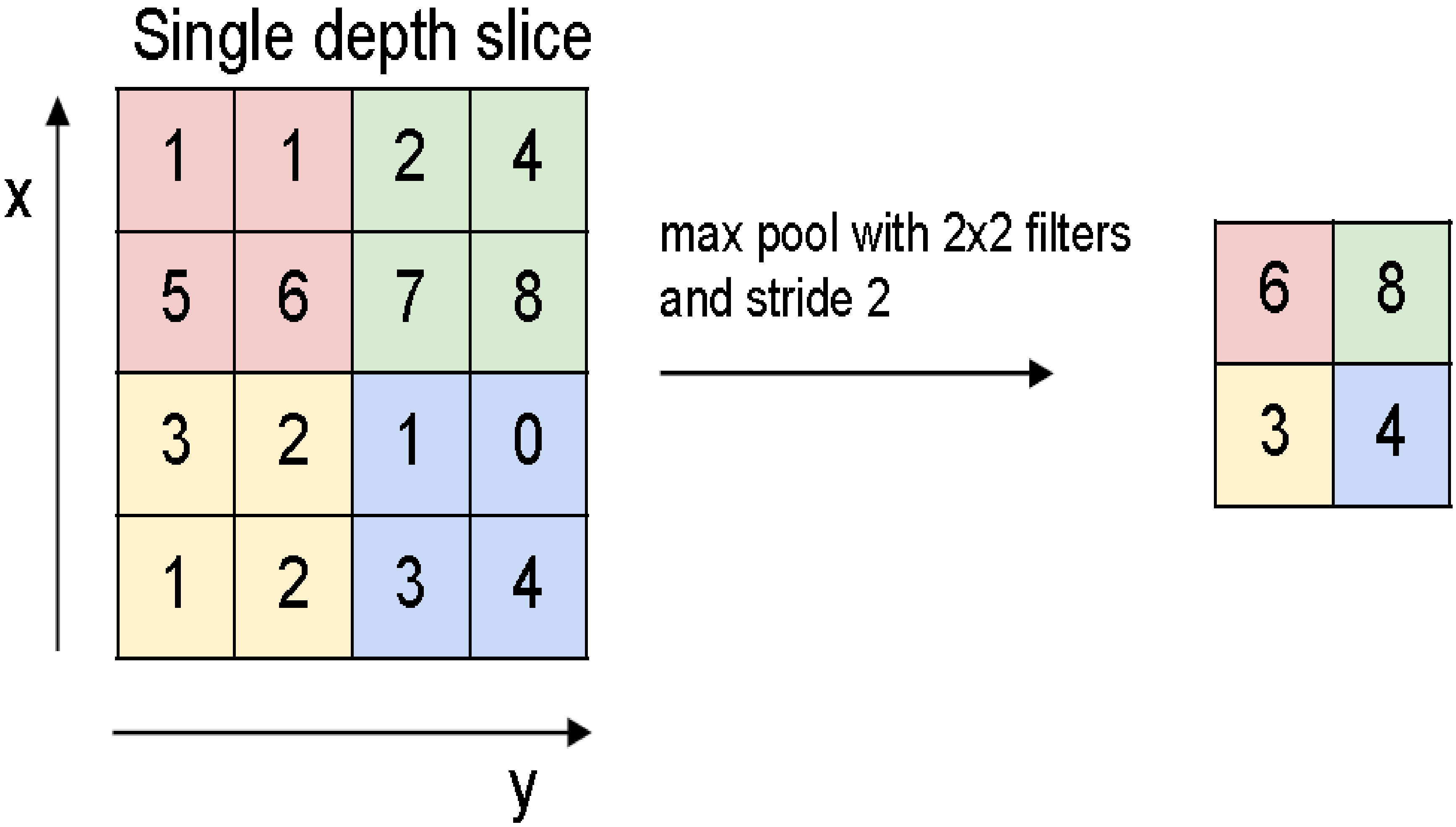
Other functions such as tanh or the sigmoid function can be used to add non-linearity to the network, but ReLU generally works better in practice.

3- Pooling layer

- the pooling layer makes the convnet less sensitive to small changes in the location of a feature
- Pooling also reduces the size of the feature map, thus simplifying computation in later layers.



MAX POOLING



Single depth slice

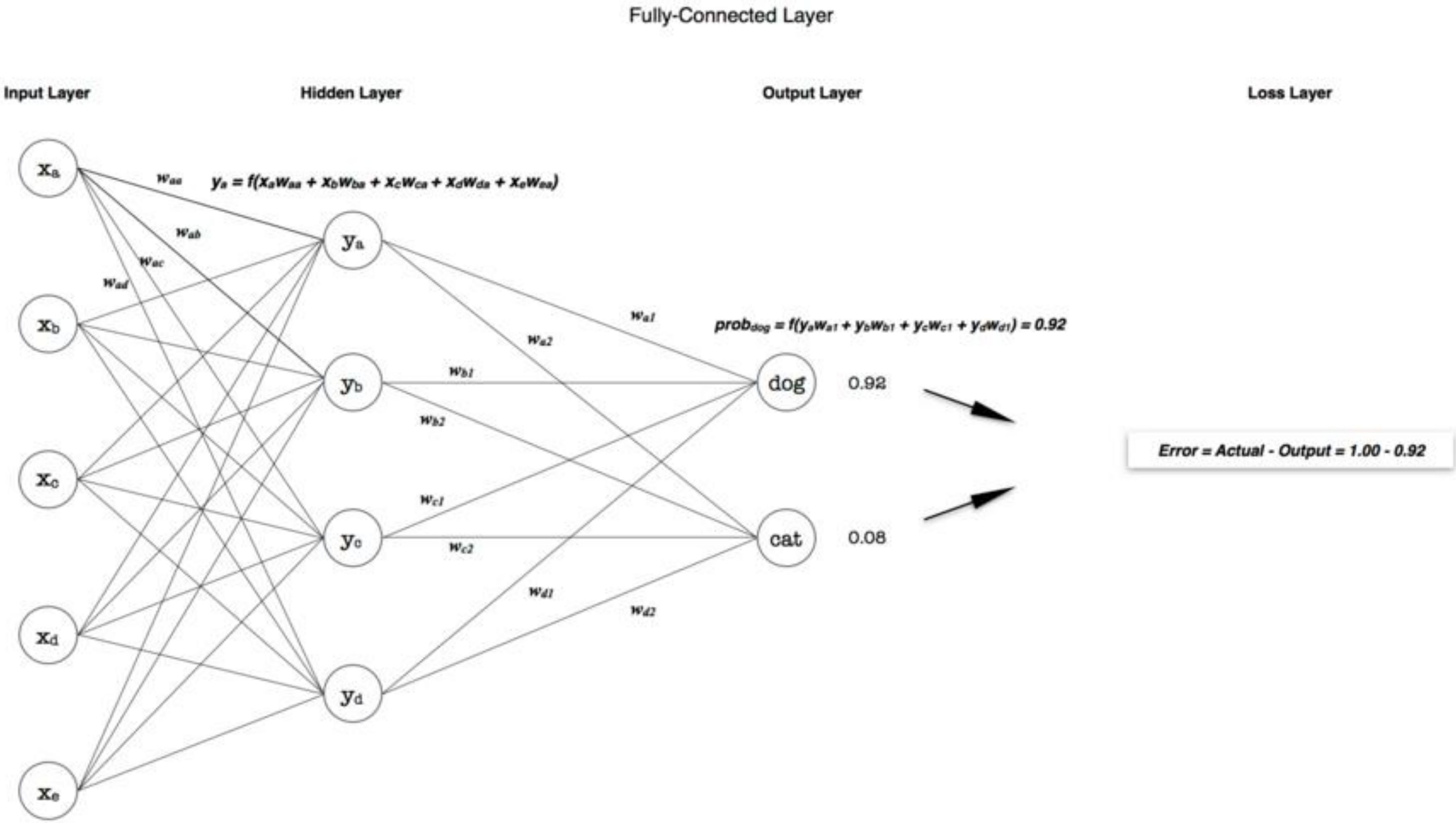
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters and stride 2

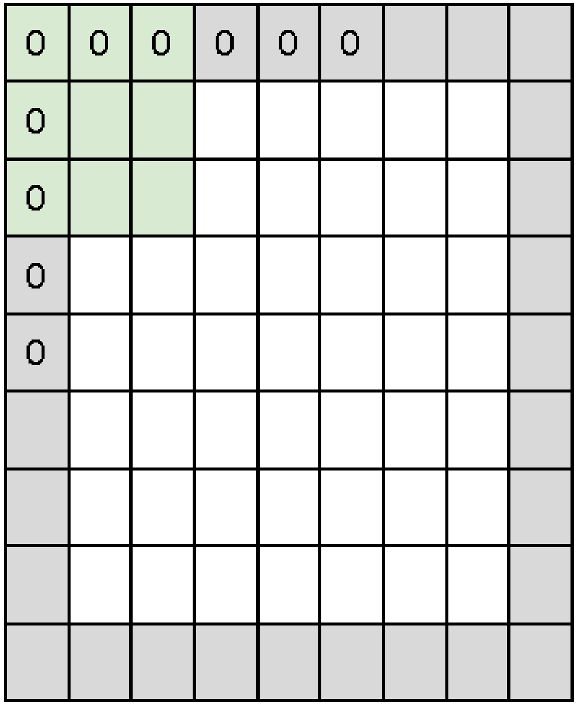
6	8
3	4

4- fully connected NN + loss layers

The fully-connected layer is where the final "decision" is made.



In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

The formula for calculating the output size for any given conv layer is

$$O = \frac{(W - K + 2P)}{S} + 1$$

where O is the output height/length, W is the input height/length, K is the filter size, P is the padding, and S is the stride.

CNN

what do they learn?

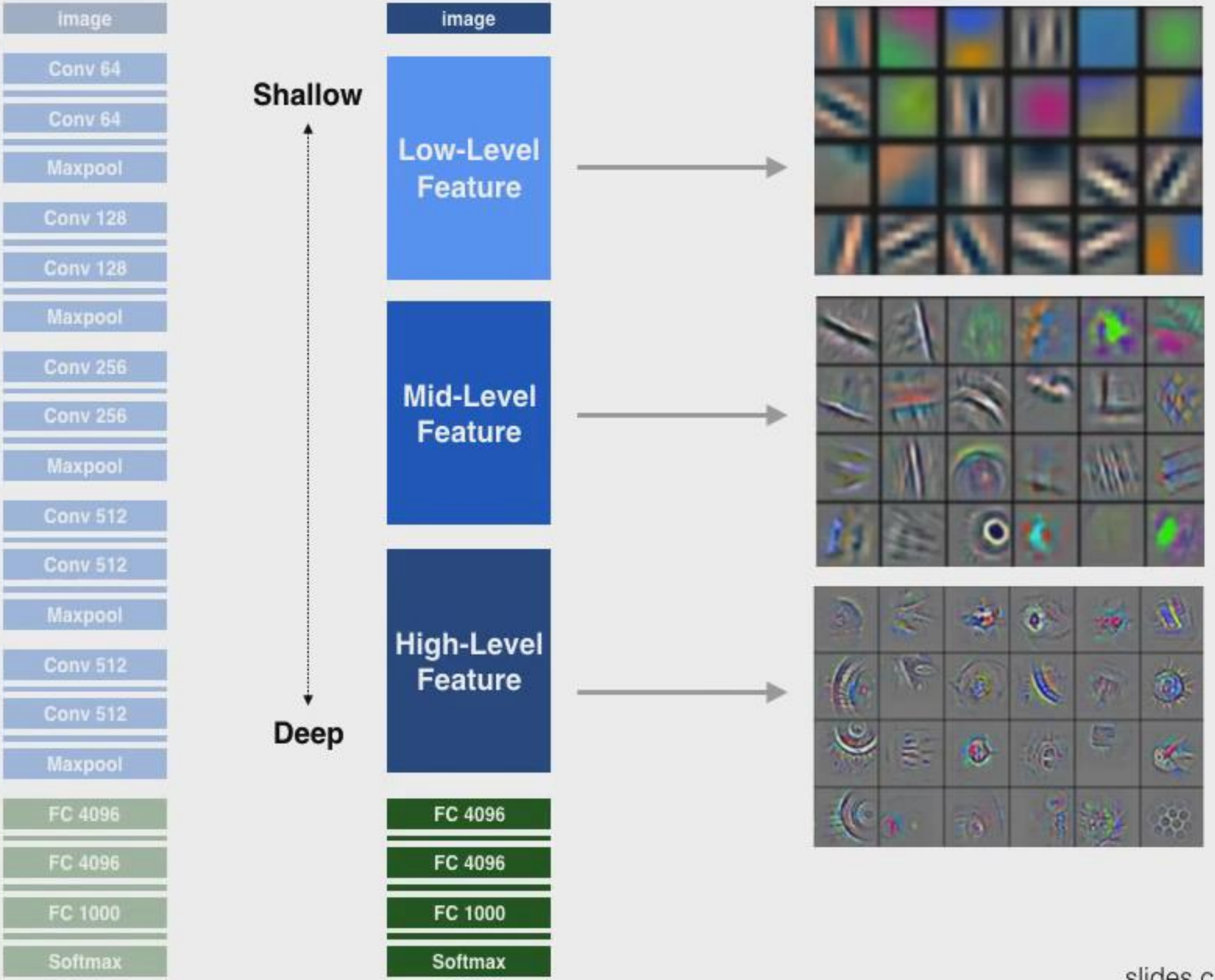


image
Conv 64
Conv 64
Maxpool
Conv 128
Conv 128
Maxpool
Conv 256
Conv 256
Maxpool
Conv 512
Conv 512
Maxpool
Conv 512
Conv 512
Maxpool
FC 4096
FC 4096
FC 1000
Softmax

Shallow

Deep

image

Low-Level Feature

Mid-Level Feature

High-Level Feature

FC 4096
FC 4096
FC 1000
Softmax

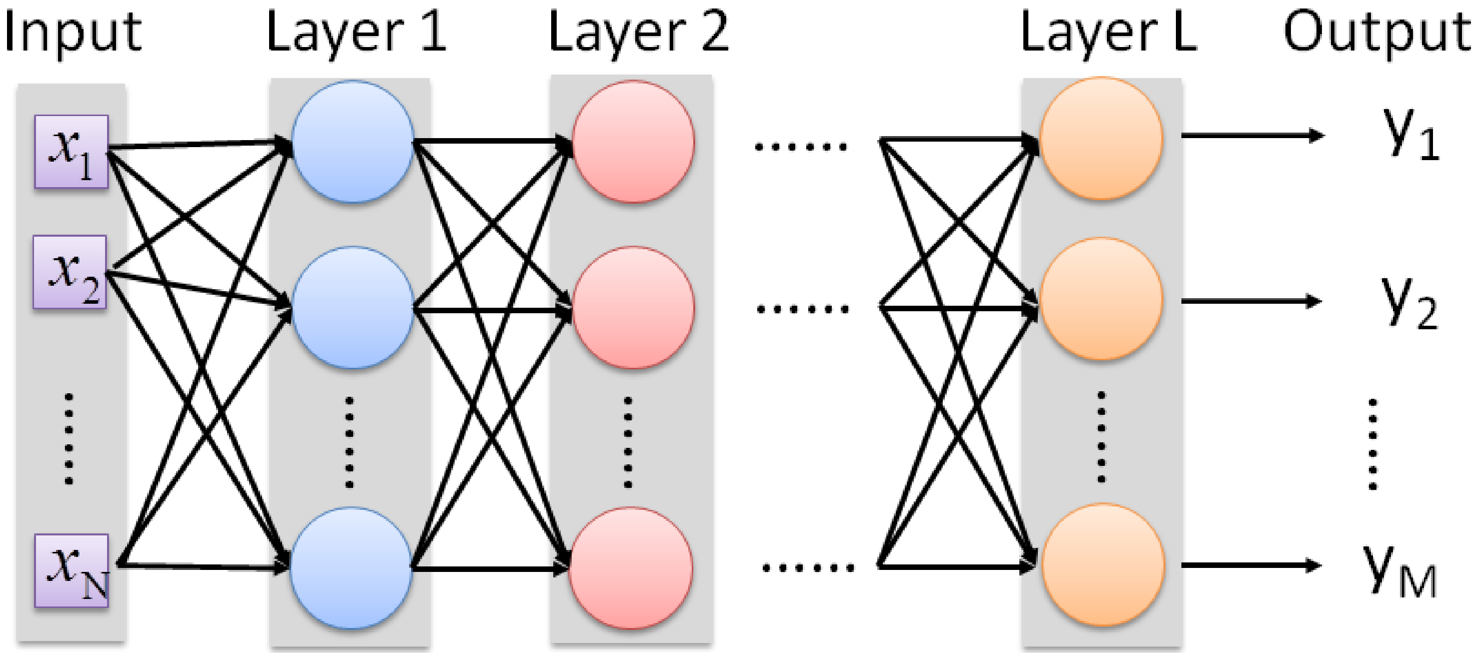
slides.co

Recurrent Neural Network RNN

Learning sequences

RNN VS Vanilla

Vanilla



- pass all input in the same time
- inputs are independent in each other
- fixed input and fixed output
- using different parameters with different layers in the network

Motivation

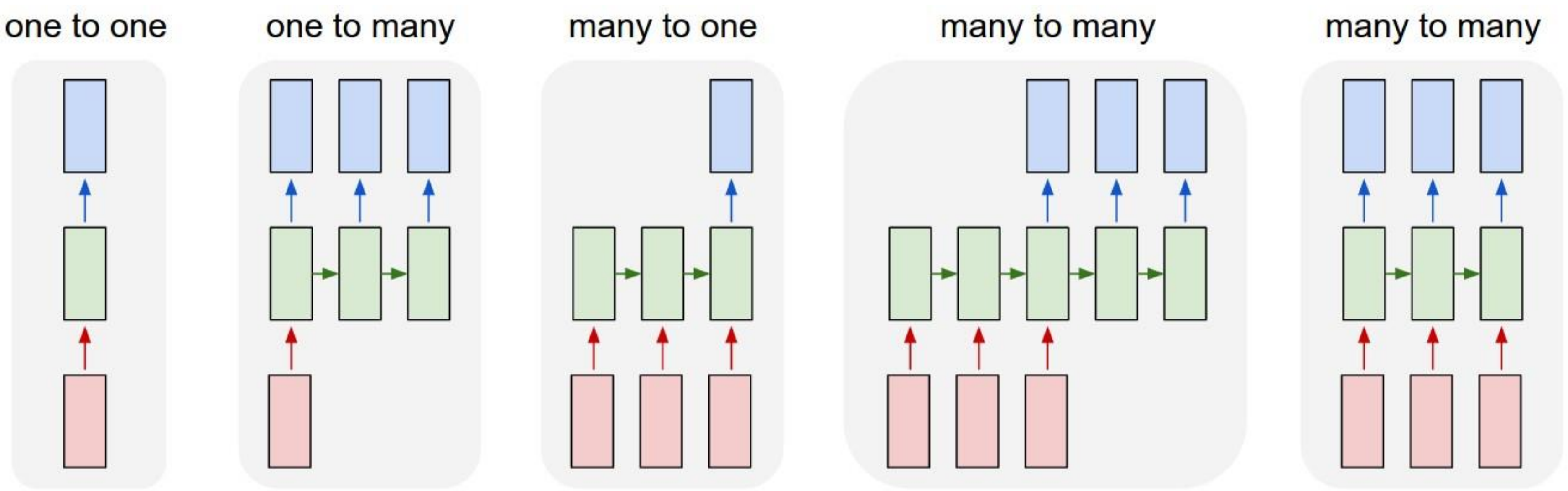


Image classification

Image captioning

Sentiment analysis

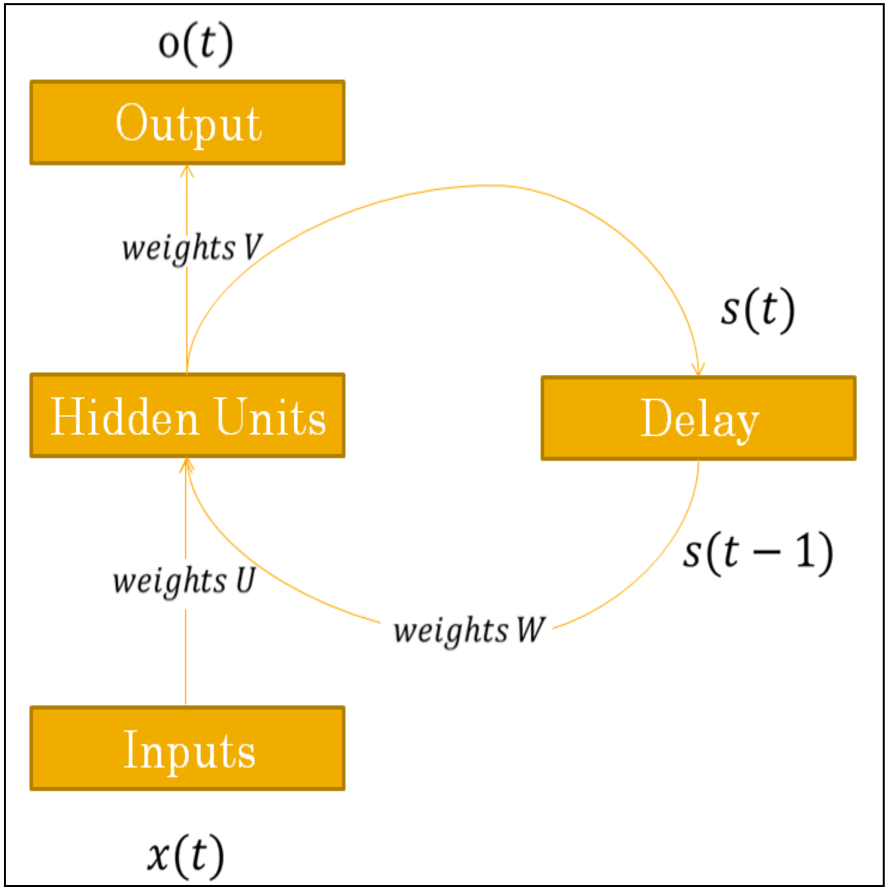
Machine translation

Synced sequence(video classification)

RNN architecture

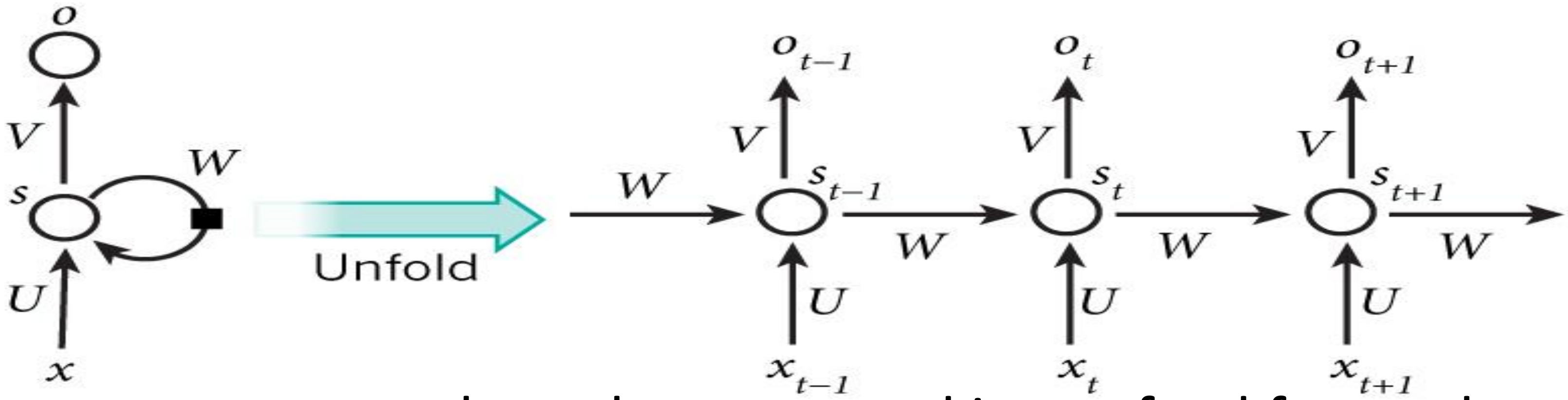
- RNNs are called *recurrent* because they perform the same task for every element of a sequence, with the output being depended on the previous computations (memory).

- Inputs ***x(t)*** outputs ***y(t)*** hidden state ***s(t)*** the memory of the network
A delay unit is introduced to hold activation until they are processed at the next step.



- The decision a recurrent net reached at time step ***t-1*** affects the decision it will reach one moment later at time step ***t***. So recurrent networks *have two sources of input*, **the present and the recent past**, which combine to determine how they respond to new data

RNN Architecture



- The recurrent network can be converted into a feed forward network by **unfolding over time**
  - The network input at time t:

    $$a_h(t) = Ux(t) + Ws(t - 1)$$

  - The activation of the input at time t:

    $$s(t) = f_h(a_h(t))$$

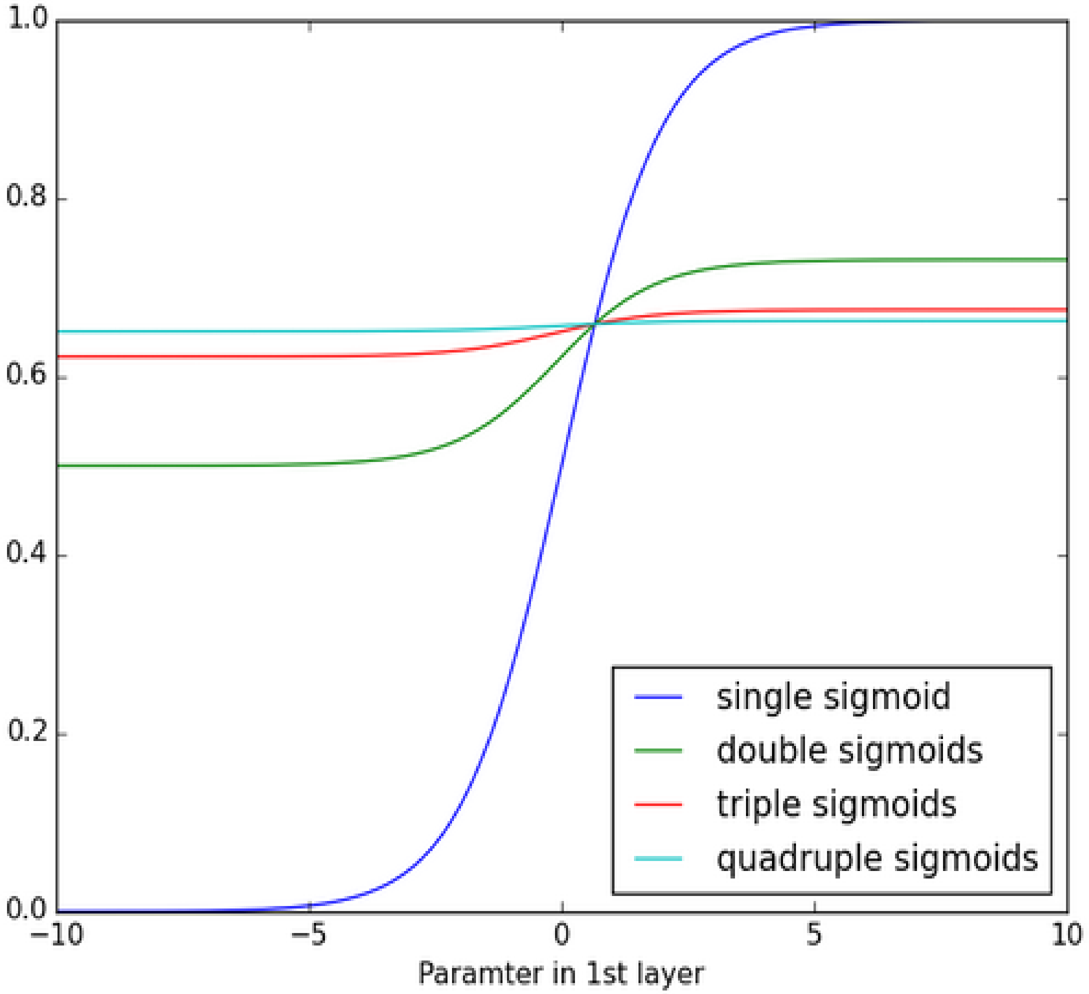
  - The network input to the output unit at time t:

    $$a_o(t) = Vs(t)$$

  - The output of the network at time t is:

    $$o(t) = f_o(a_o(t))$$

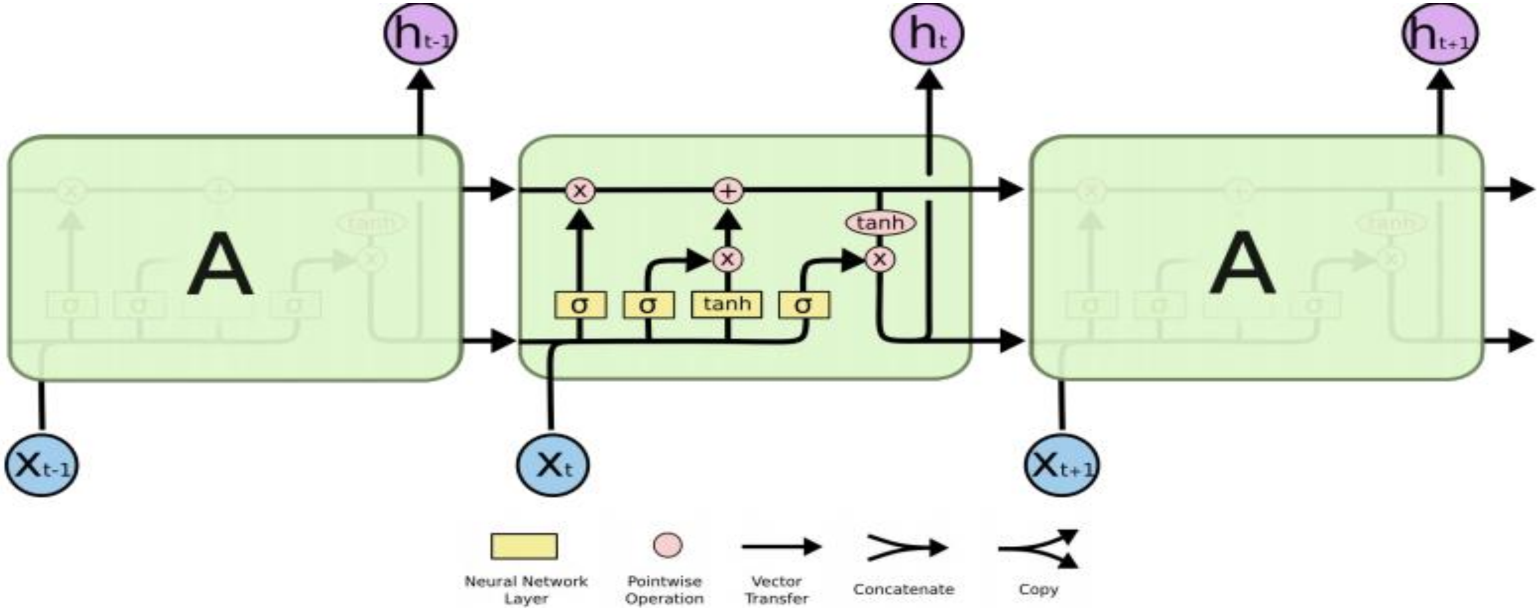
Vanishing Gradients



long-term dependencies

The key difference from regular networks is that we sum up the gradients for W at each time step

Recurrent NN - LSTM



The basic unit in the hidden layer of an LSTM network is a memory block, it replaces the hidden unit in a traditional RNN. A memory block contains one or more memory cell and a pair of adaptive multiplicative gating units which gates input and output to all cells in the block. Memory blocks allow cells to share the same gates thus reducing the number of parameters. Each cell has in its core a recurrently self connected linear unit called the “Constant error carousel” whose activation we call the cell state.

Natural Language Processing Tasks

1- Automatic Summarization

the process of shortening a text document with software, in order to create a summary with the major points of the original document.

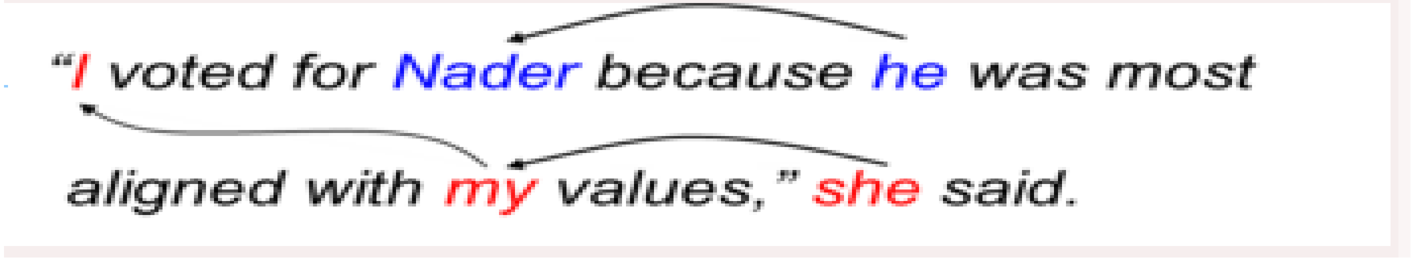
There are two methods

1-extracting sentences or parts thereof from the original text
2- generating abstract summaries.

Tools- The Python library sumy,

2- Co reference resolution

Coreference resolution is the task of finding all expressions that refer to the same entity in a text.



"I voted for Nader because he was most aligned with my values," she said.

Tools- The Apache OpenNLP

tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co reference resolution.

3- Named Entity Recognition

Named-entity recognition (**NER**) (also known as **entity identification**, **entity chunking** and **entity extraction**) is a subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as

- person names
- company/organization names
- locations
- dates & time
- percentages
- monetary amounts (Currency)

- number
- Device
- Jop
- Car
- Cell Phone

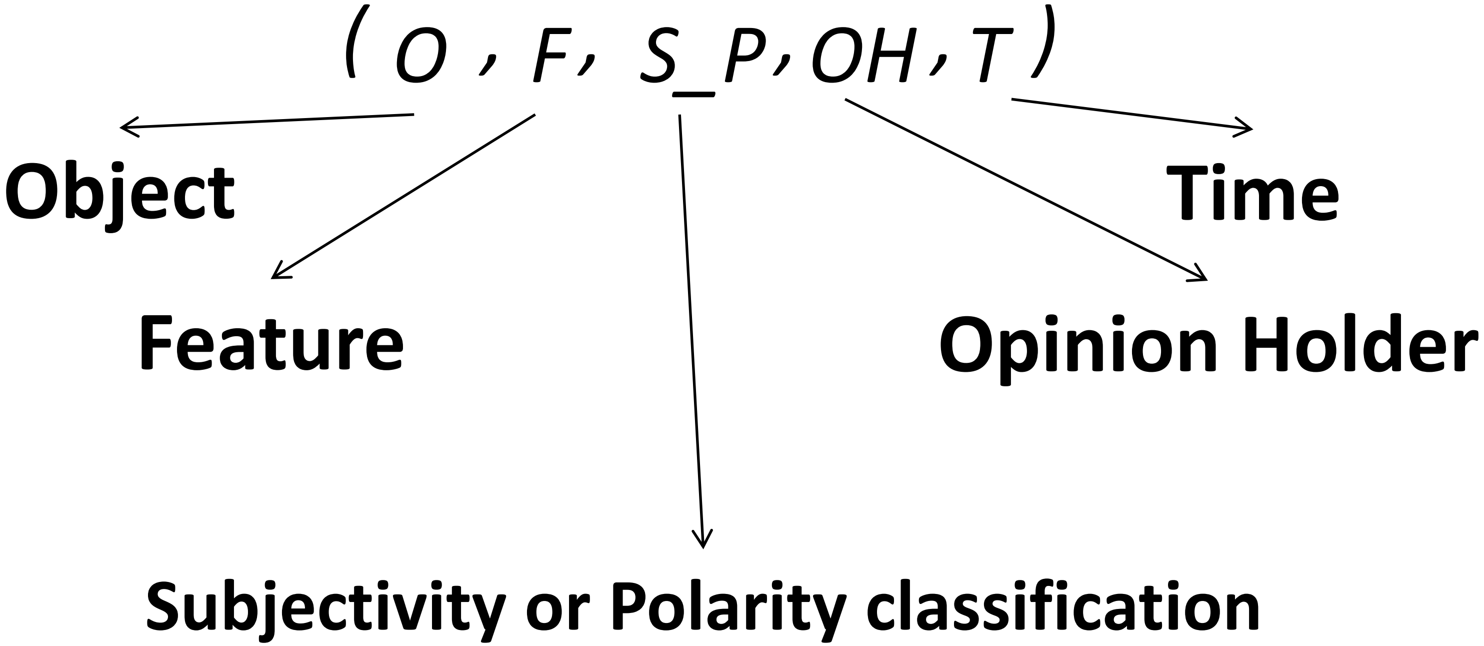
Tools- The Apache OpenNLP

4- Sentiment analysis

The task of finding the opinions of authors about specific entities.

Sentiment Analysis Problem

An *opinion* is a quintuple



(O , F, S_P, OH, T)

Object

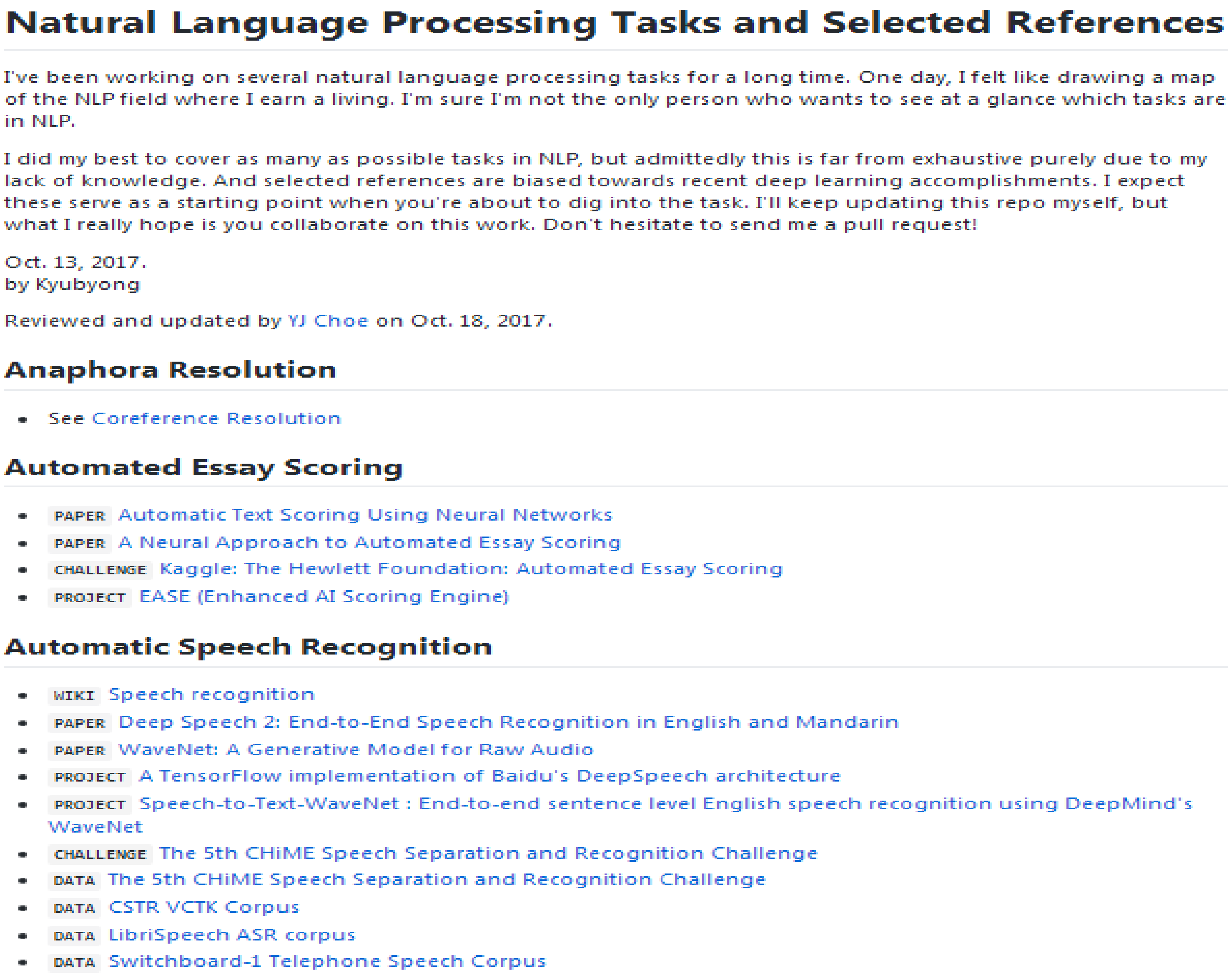
Feature

Subjectivity or Polarity classification

Opinion Holder

Time

https://github.com/Kyubyong/nlp_tasks#coreference-resolution



Natural Language Processing Tasks and Selected References

I've been working on several natural language processing tasks for a long time. One day, I felt like drawing a map of the NLP field where I earn a living. I'm sure I'm not the only person who wants to see at a glance which tasks are in NLP.

I did my best to cover as many as possible tasks in NLP, but admittedly this is far from exhaustive purely due to my lack of knowledge. And selected references are biased towards recent deep learning accomplishments. I expect these serve as a starting point when you're about to dig into the task. I'll keep updating this repo myself, but what I really hope is you collaborate on this work. Don't hesitate to send me a pull request!

Oct. 13, 2017.
by Kyubyong

Reviewed and updated by YJ Choe on Oct. 18, 2017.

Anaphora Resolution

- See Coreference Resolution

Automated Essay Scoring

- PAPER Automatic Text Scoring Using Neural Networks
- PAPER A Neural Approach to Automated Essay Scoring
- CHALLENGE Kaggle: The Hewlett Foundation: Automated Essay Scoring
- PROJECT EASE (Enhanced AI Scoring Engine)

Automatic Speech Recognition

- WIKI Speech recognition
- PAPER Deep Speech 2: End-to-End Speech Recognition in English and Mandarin
- PAPER WaveNet: A Generative Model for Raw Audio
- PROJECT A TensorFlow implementation of Baidu's DeepSpeech architecture
- PROJECT Speech-to-Text-WaveNet : End-to-end sentence level English speech recognition using DeepMind's WaveNet
- CHALLENGE The 5th CHiME Speech Separation and Recognition Challenge
- DATA The 5th CHiME Speech Separation and Recognition Challenge
- DATA CSTR VCTK Corpus
- DATA LibriSpeech ASR corpus
- DATA Switchboard-1 Telephone Speech Corpus