

BLUE REVOLUTION

BIODIVERSITY UNDERESTIMATION IN OUR BLUE PLANET :
ARTIFICIAL INTELLIGENCE REVOLUTION
IN BENTHIC TAXONOMY



Introduction of AI: Machine and Deep Learning

Presented By: Abdul Qayyum

Abdesslam BENZINU, Kamal NASREDDINE, khawla bengained



Introduction of Machine Learning and Deep Learning

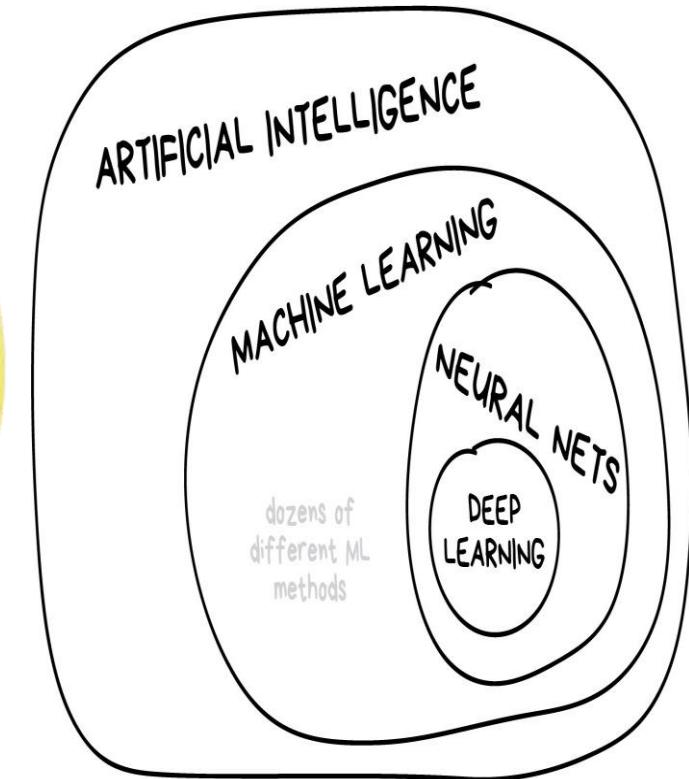
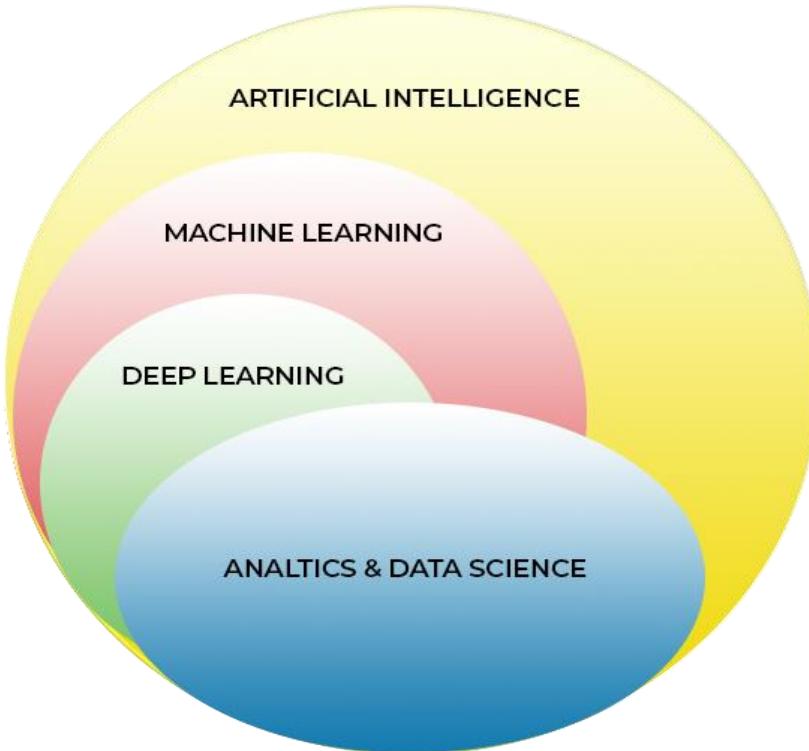
Outlines of Lecture:

In this lecture we will consider

- What is Artificial intelligence, Machine Learning ?
- Comparison of ML algorithms
- Feature Extraction and Scaling
- Applications of ML
- Introduction of Deep Learning
- Neural Network Basic
- Applications

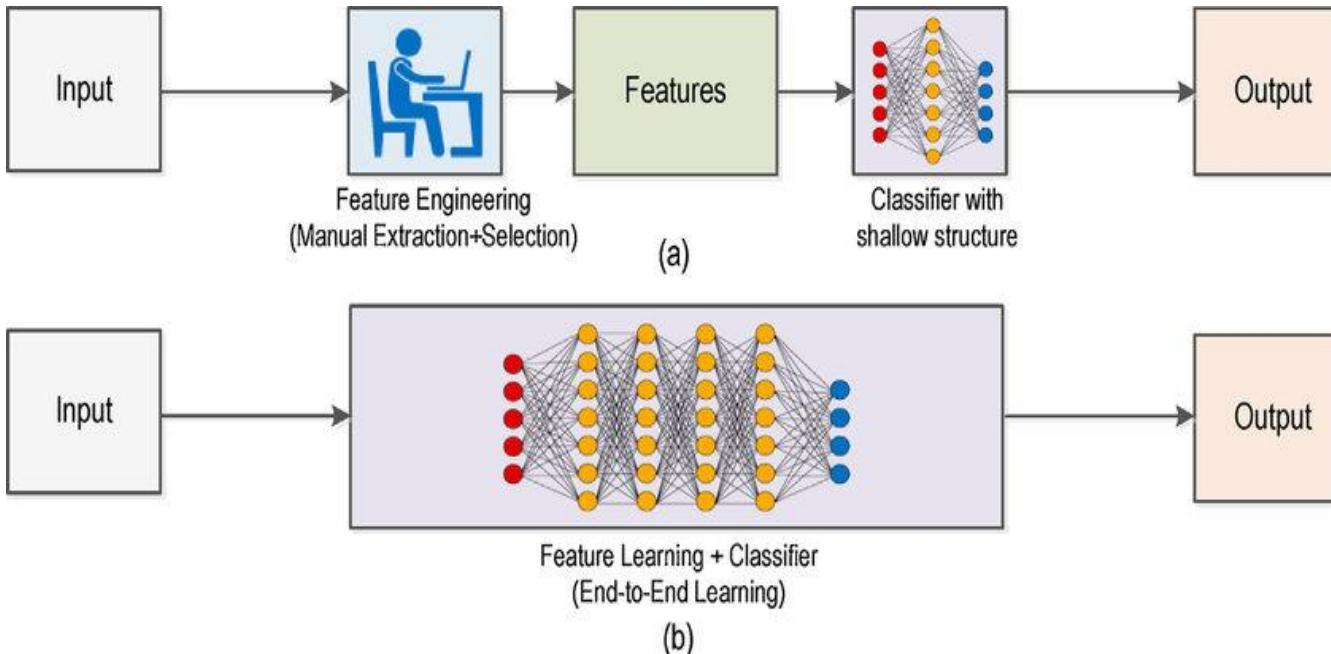
Introduction

- AI and machine learning are often used interchangeably, especially in the realm of big data.
- **Artificial intelligence** is a broader concept than machine learning, which addresses the use of computers to mimic the cognitive functions of humans.
- When machines carry out tasks based on algorithms in an “intelligent” manner, that is AI.
- **Machine learning** is a subset of AI and focuses on the ability of machines to receive a set of data and learn for themselves, changing algorithms as they learn more about the information they are processing.
- **Deep learning** goes yet another level deeper and can be considered a subset of machine learning.
The concept of deep learning is sometimes just referred to as “deep neural networks,” referring to the many layers involved. A neural network may only have a single layer of data, while a deep neural network has two or more.



Source:
<https://datascience.stackexchange.com/questions/16422/machine-learning-vs-deep-learning>

Difference ML and DL



Machine Learning

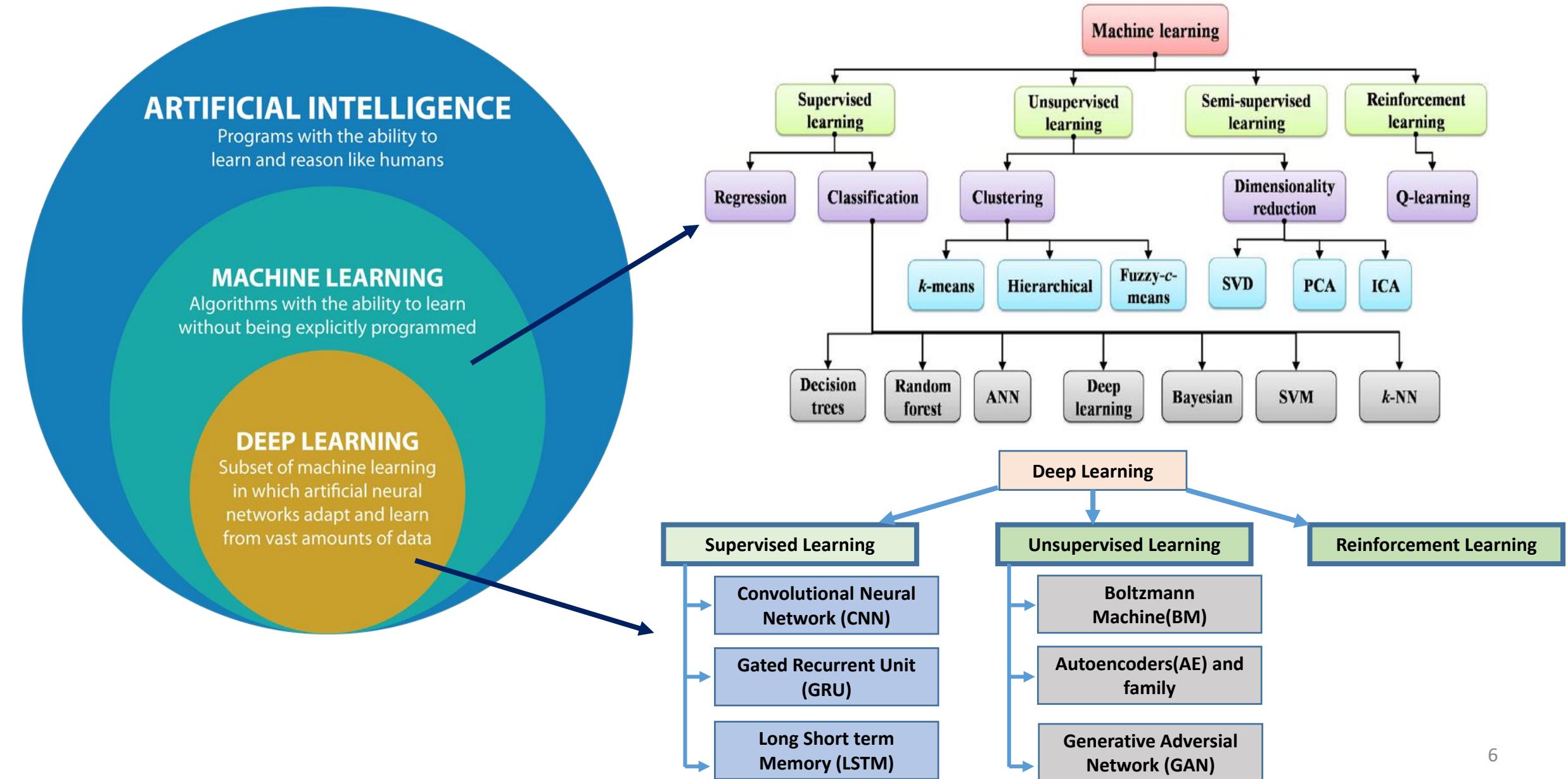
- + Good results with small data sets
- + Quick to train a model
- Need to try different features and classifiers to achieve best results
- Accuracy plateaus

Deep Learning

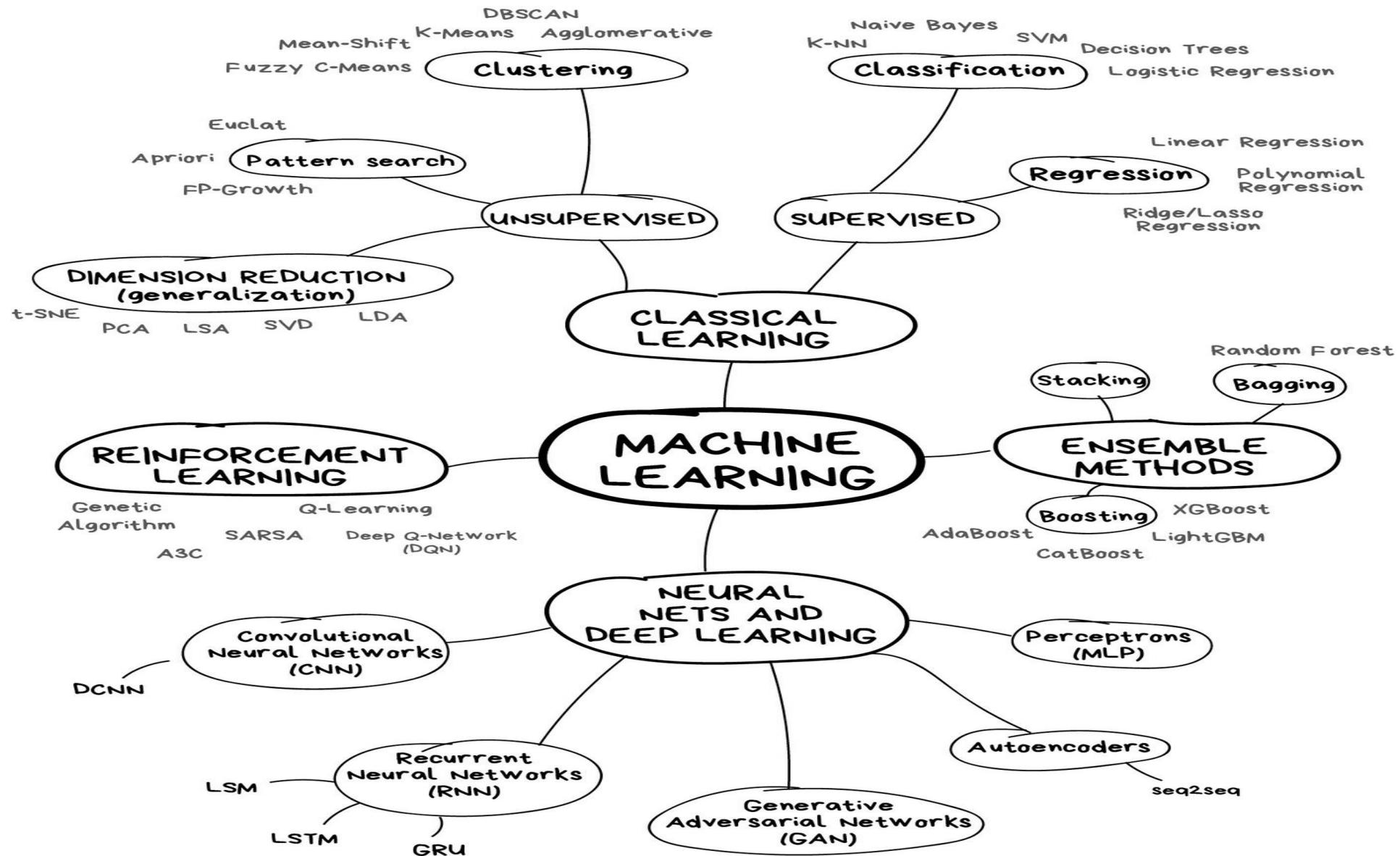
- Requires very large data sets
- Computationally intensive
- + Learns features and classifiers automatically
- + Accuracy is unlimited

Deep Learning Vs Machine Learning		
Factors	Deep Learning	Machine Learning
Data Requirement	Requires large data	Can train on lesser data
Accuracy	Provides high accuracy	Gives lesser accuracy
Training Time	Takes longer to train	Takes less time to train
Hardware Dependency	Requires GPU to train properly	Trains on CPU
Hyperparameter Tuning	Can be tuned in various different ways.	Limited tuning capabilities

Algorithms of ML and DL

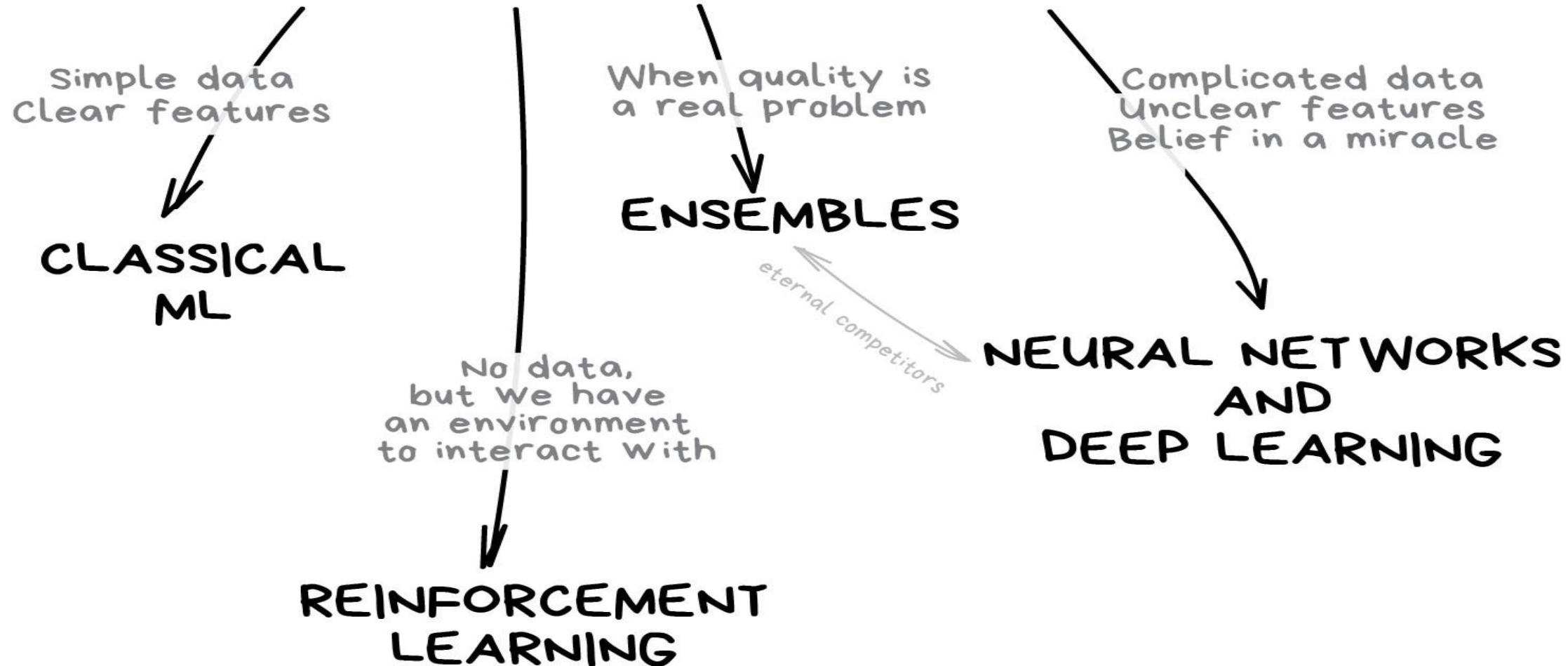


Machine Learning Tree

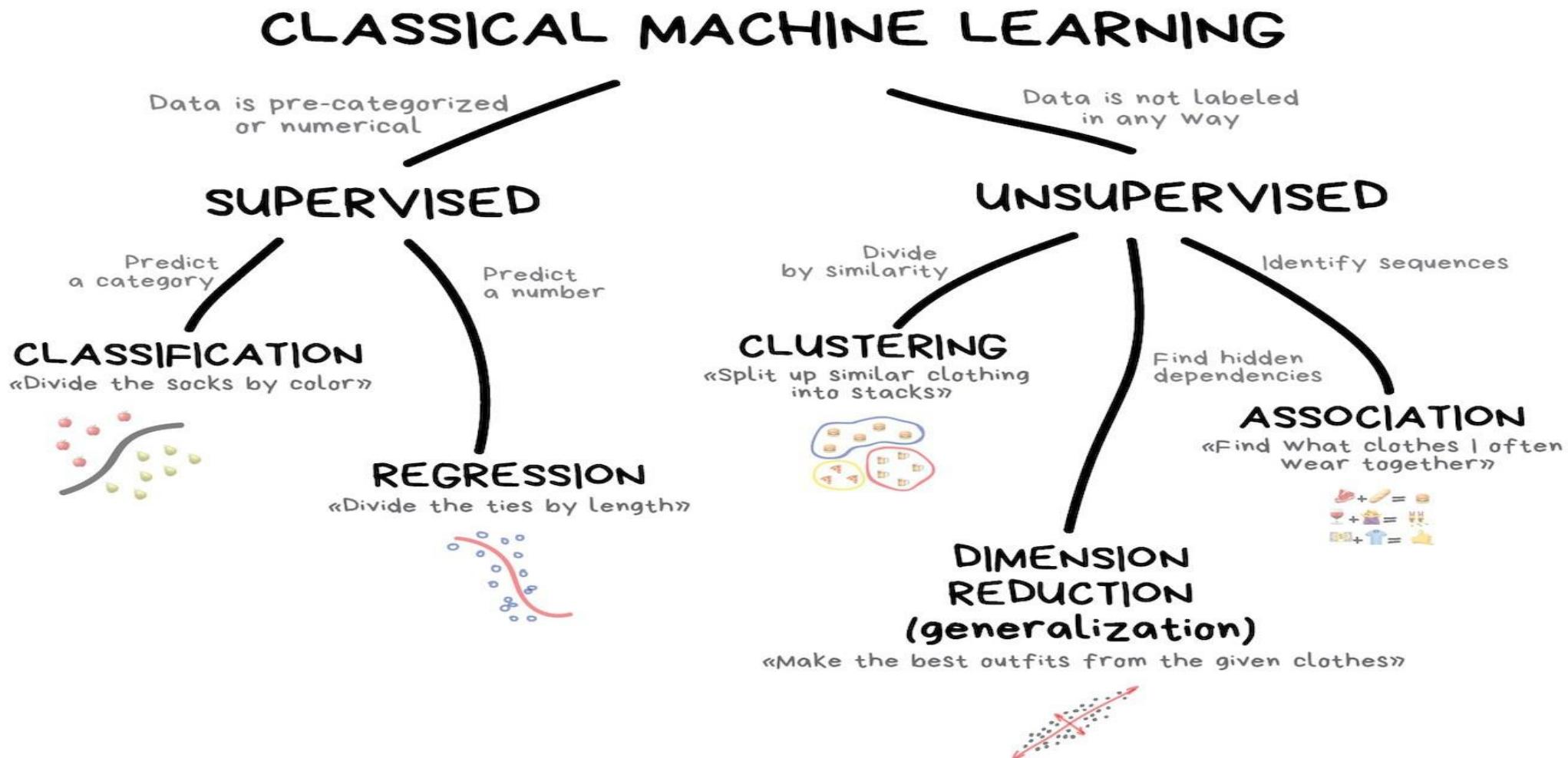


Types of Machine Learning

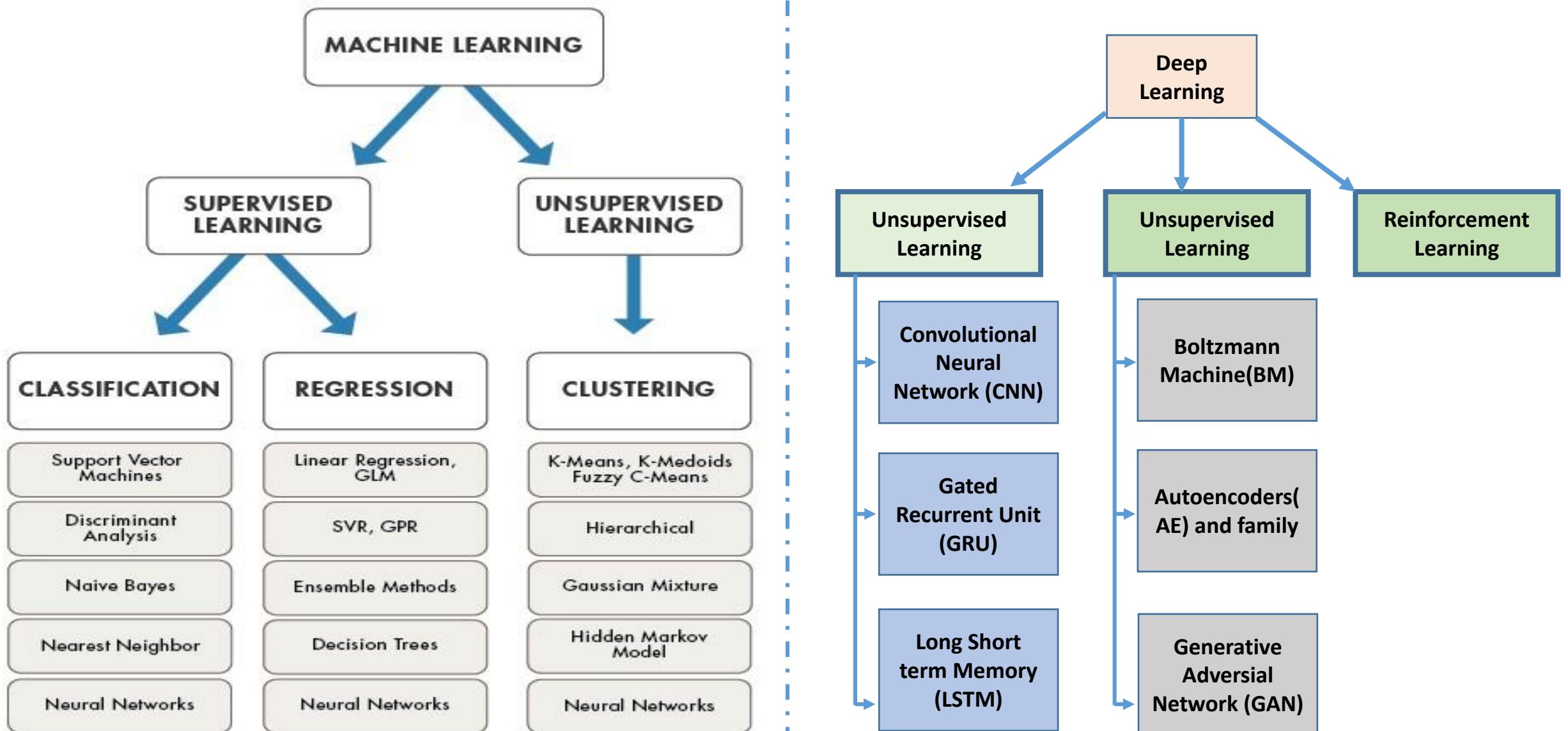
THE MAIN TYPES OF MACHINE LEARNING



Classical Machine Learning



Supervised and Unsupervised Learning



Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map $x \rightarrow y$

Examples: Classification, Regression,
Object detection, Semantic Segmentation,
Image captioning



→ Cat

Classification



DOG, DOG, CAT

Object Detection



GRASS, CAT,
TREE, SKY

Semantic Segmentation



A cat sitting on a suitcase on the floor

Image captioning

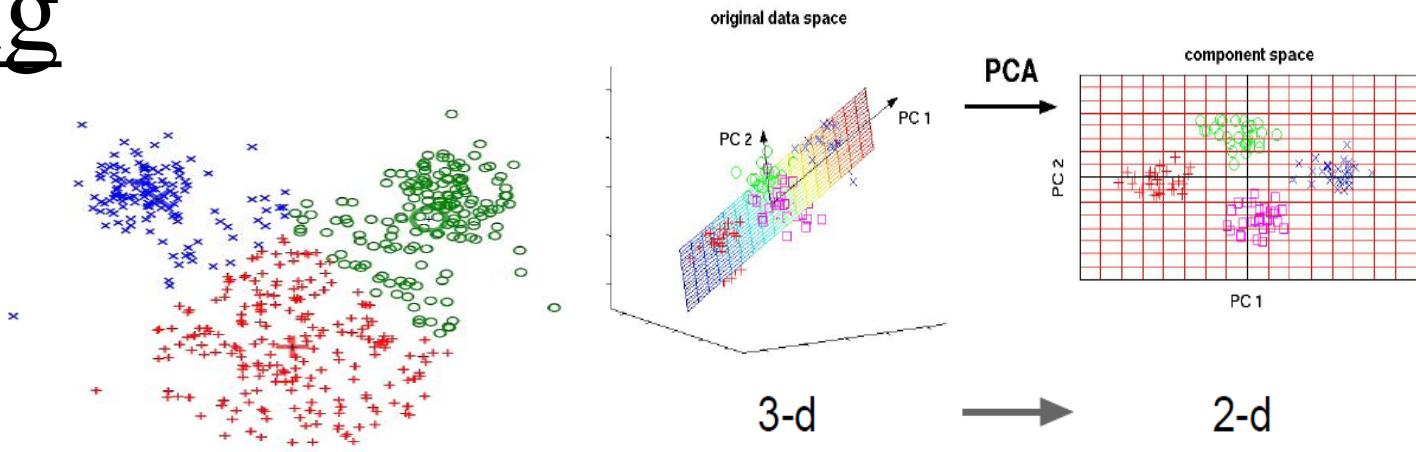
Unsupervised Learning

Data: x

x is data, no label

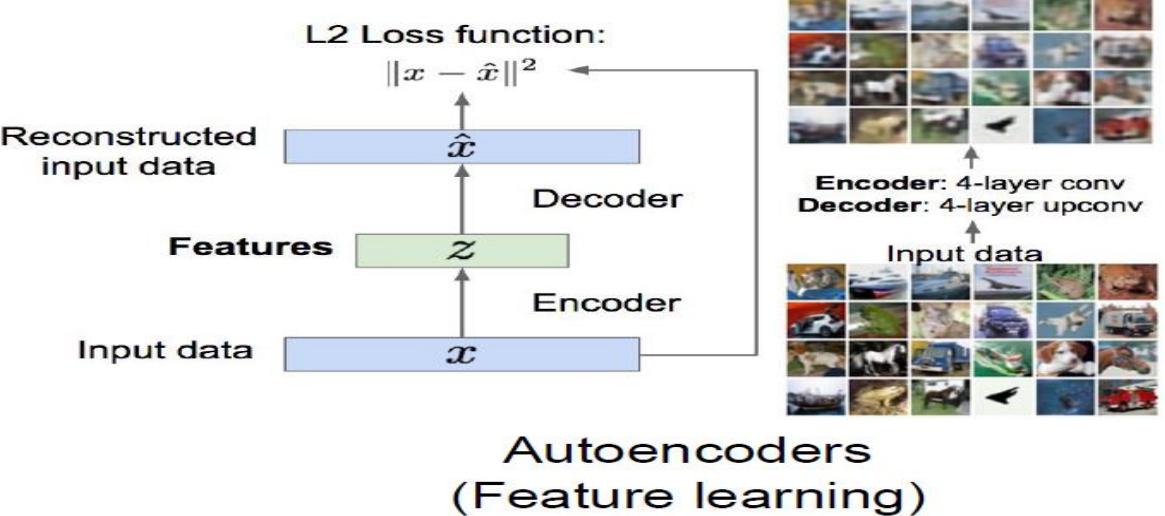
Goal: Learn some underlying
Hidden structure of the data

Examples: Clustering, Dimensionality
reduction, Feature Learning, density estimation



K-means clustering

Principal Component Analysis
(Dimensionality reduction)



Scikit-learn



scikit-learn

Machine Learning in Python

Getting Started What's New in 0.22.2 GitHub

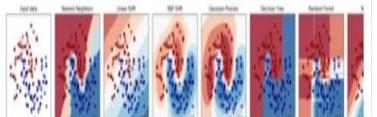
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...

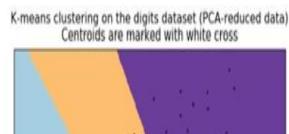


Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: K-Means, spectral clustering, mean-shift, and more...



Prev Up Next

scikit-learn 0.22.2

Other versions

Please [cite us](#) if you use the software.

1. Supervised learning

1. Supervised learning

1.1. Linear Models

- 1.1.1. Ordinary Least Squares
- 1.1.2. Ridge regression and classification
- 1.1.3. Lasso
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic-Net
- 1.1.6. Multi-task Elastic-Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
- 1.1.11. Logistic regression
- 1.1.12. Stochastic Gradient Descent - SGD
- 1.1.13. Perceptron
- 1.1.14. Passive Aggressive Algorithms
- 1.1.15. Robustness regression: outliers and modeling errors
- 1.1.16. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis

- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction

Scikit-plot

Scikit-plot
stable

Search docs

First Steps with Scikit-plot

Metrics Module

Estimators Module

Clusterer Module

Decomposition Module

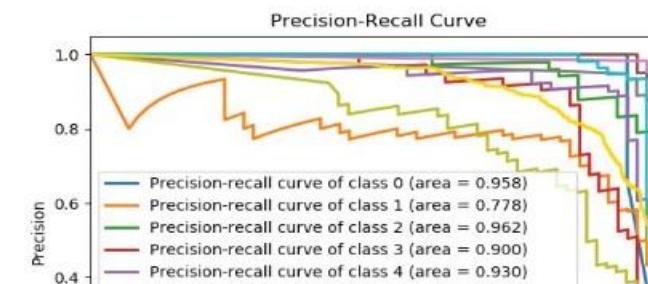
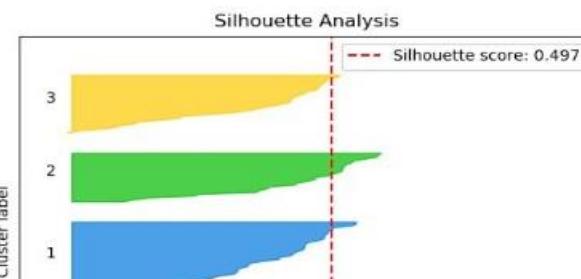
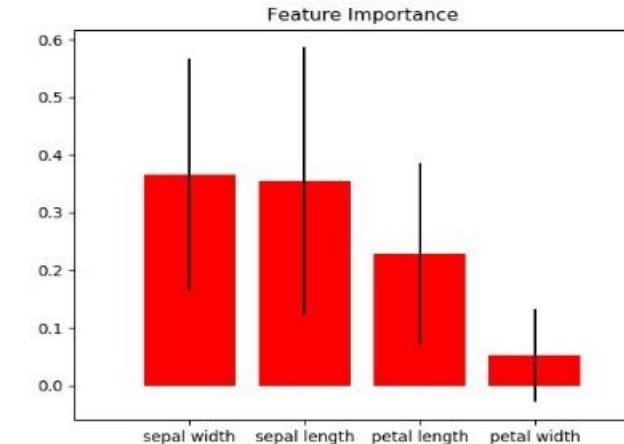
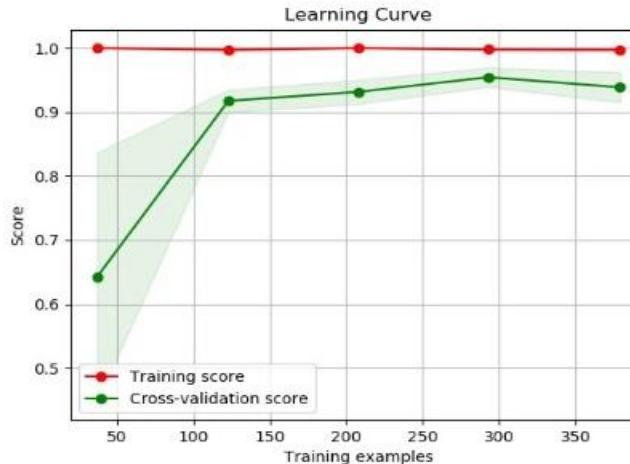
Read the Docs v: stable ▾

Docs » Welcome to Scikit-plot's documentation!

Edit on GitHub

Welcome to Scikit-plot's documentation!

The quickest and easiest way to go from analysis...



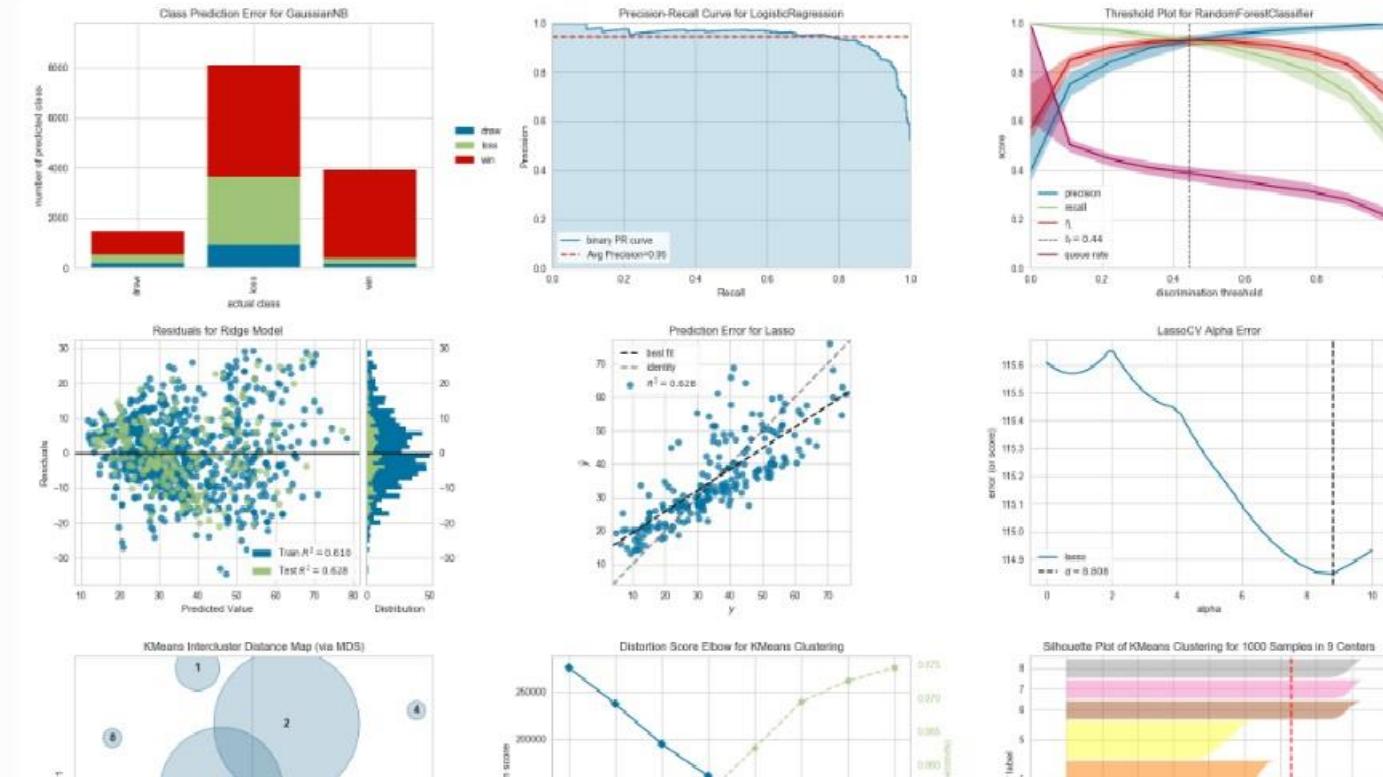
Yellowbrick

The screenshot shows the official documentation website for Yellowbrick. The top navigation bar includes the Yellowbrick logo and a "latest" link. Below the header is a search bar labeled "Search docs". The main sidebar contains a list of links: Quick Start, Model Selection Tutorial, Visualizers and API, Oneliners, Contributing, Effective Matplotlib, Yellowbrick for Teachers, Gallery, About, Frequently Asked Questions, User Testing Instructions, Code of Conduct, Changelog, and Governance.

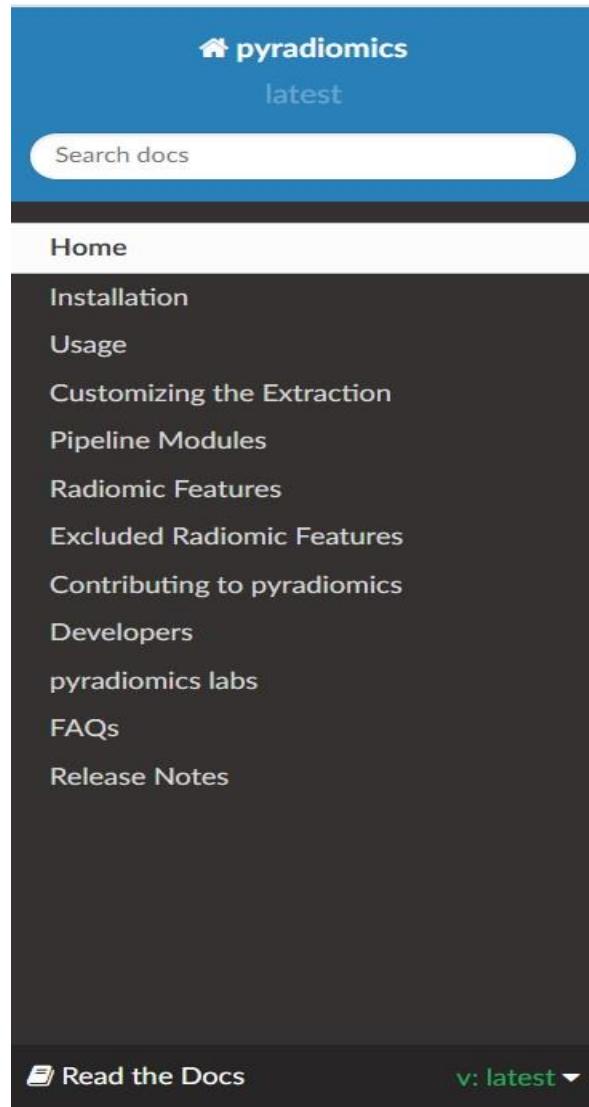
Docs » Yellowbrick: Machine Learning Visualization

Edit on GitHub

Yellowbrick: Machine Learning Visualization



Pyradiomics



The sidebar on the left contains the following navigation links:

- pyradiomics latest
- Search docs
- Home
- Installation
- Usage
- Customizing the Extraction
- Pipeline Modules
- Radiomic Features
- Excluded Radiomic Features
- Contributing to pyradiomics
- Developers
- pyradiomics labs
- FAQs
- Release Notes

At the bottom, there are two buttons: "Read the Docs" and a dropdown menu showing "v: latest ▾".

Docs » Welcome to pyradiomics documentation!

 Edit on GitHub

Welcome to pyradiomics documentation!

This is an open-source python package for the extraction of Radiomics features from medical imaging. With this package we aim to establish a reference standard for Radiomic Analysis, and provide a tested and maintained open-source platform for easy and reproducible Radiomic Feature extraction. By doing so, we hope to increase awareness of radiomic capabilities and expand the community. The platform supports both the feature extraction in 2D and 3D and can be used to calculate single values per feature for a region of interest ("segment-based") or to generate feature maps ("voxel-based").

If you publish any work which uses this package, please cite the following publication: van Griethuysen, J. J. M., Fedorov, A., Parmar, C., Hosny, A., Aucoin, N., Narayan, V., Beets-Tan, R. G. H., Fillion-Robin, J. C., Pieper, S., Aerts, H. J. W. L. (2017). Computational Radiomics System to Decode the Radiographic Phenotype. *Cancer Research*, 77(21), e104–e107. <https://doi.org/10.1158/0008-5472.CAN-17-0339> <<https://doi.org/10.1158/0008-5472.CAN-17-0339>>_

Note

This work was supported in part by the US National Cancer Institute grant 5U24CA194354,
QUANTITATIVE RADIOMICS SYSTEM DECODING THE TUMOR PHENOTYPE.

Warning

Pyradiomics is still under development.

MATLAB



Products Solutions The academic world Support Community Events

Get MATLAB



AQ

Machine Learning

Search MathWorks.com



Overview | New Arrivals | Resources | Customer testimonials

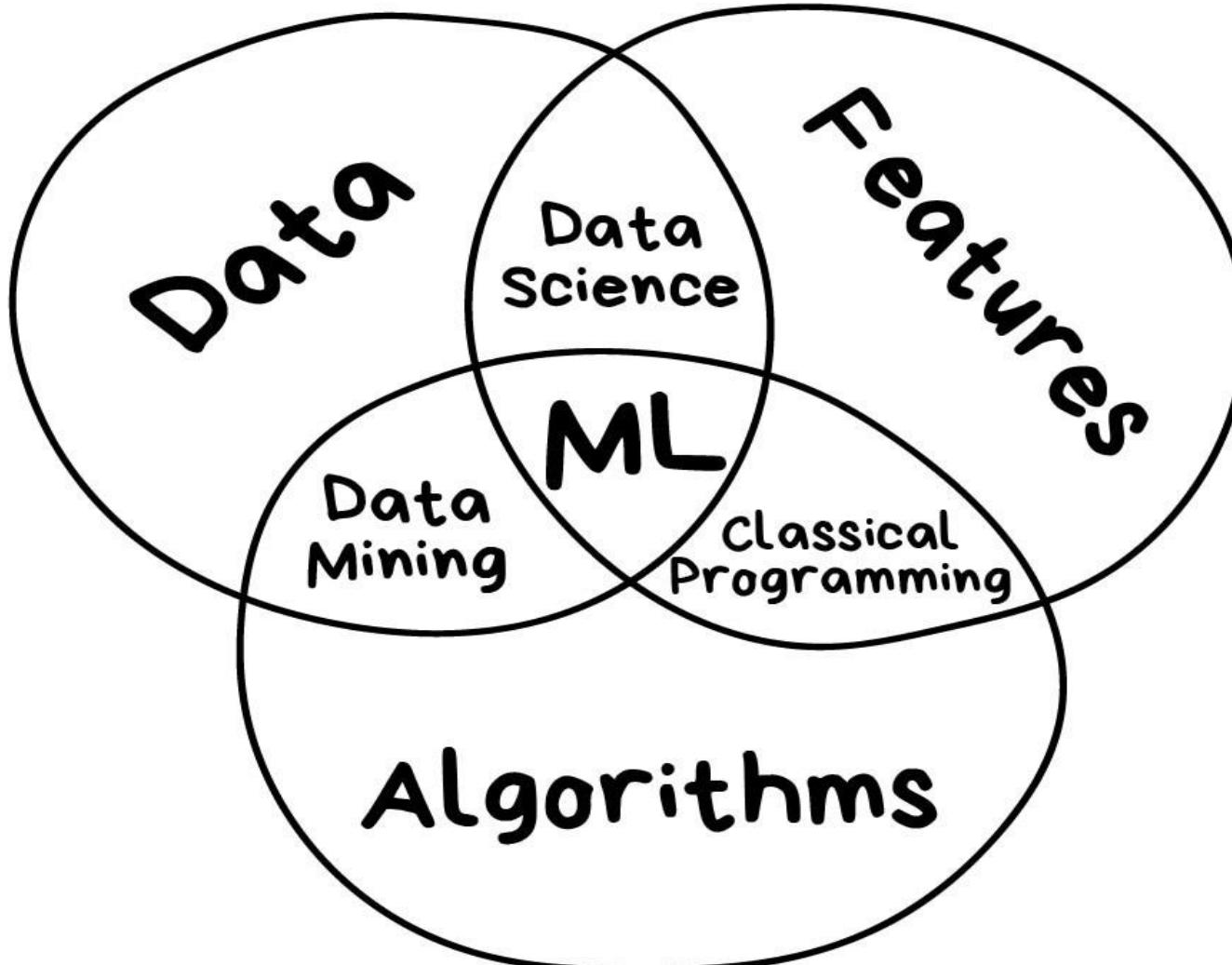
MATLAB for Machine Learning

1164.843
1199.103
Train models, adjust parameters and deploy your
models in production

Evaluation version

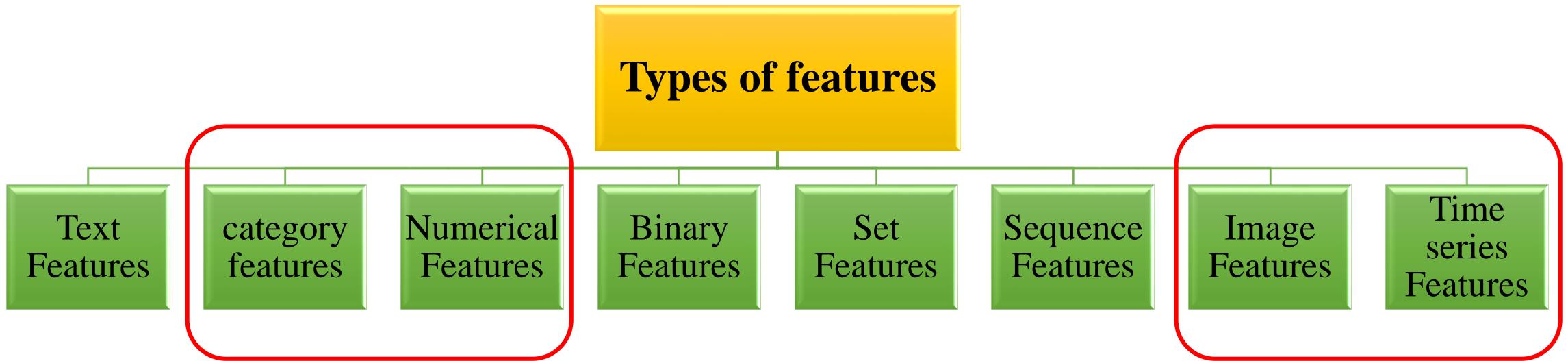
-28064.143

Components of ML

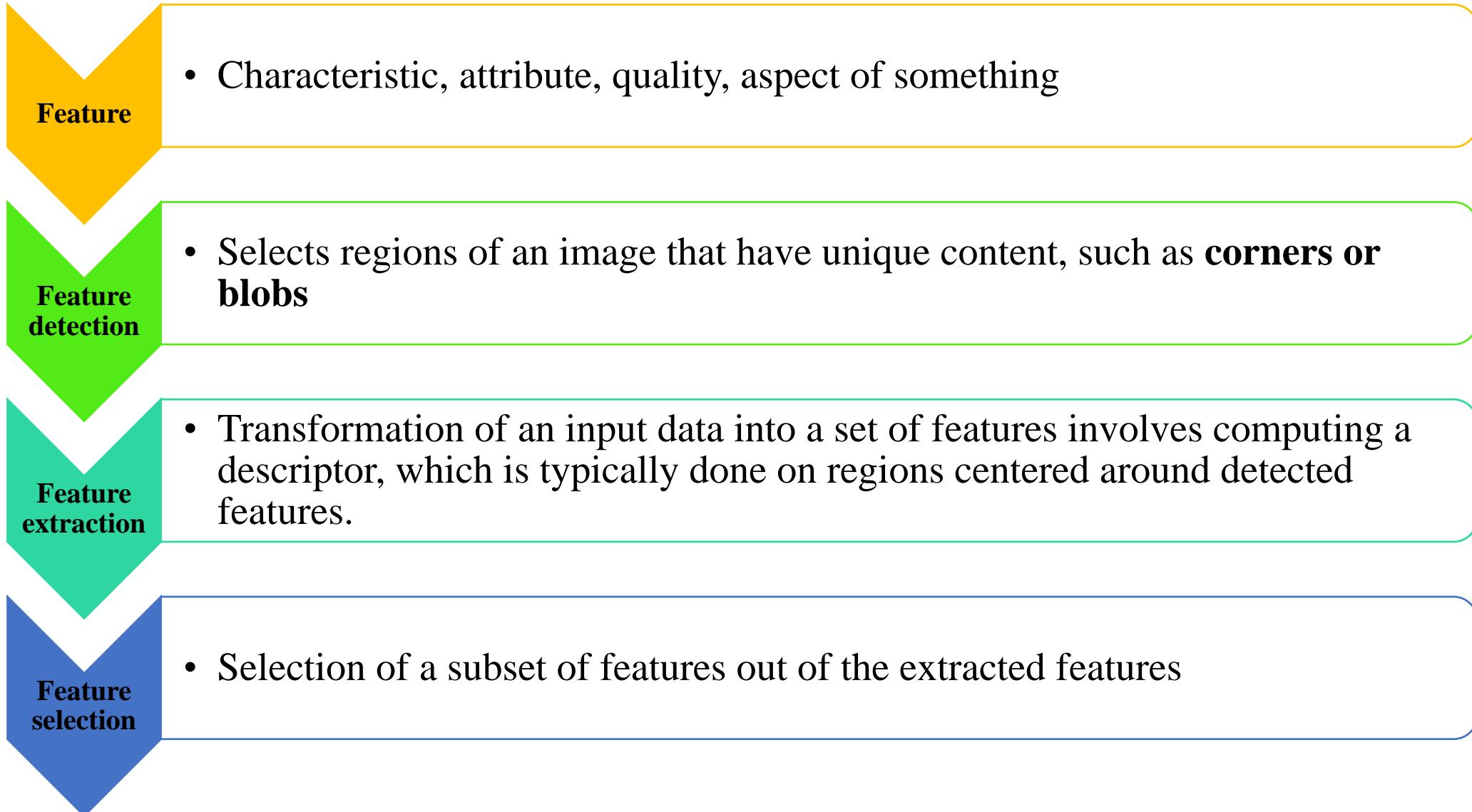


<https://noeliagorod.com/2019/05/21/machine-learning-for-everyone-in-simple-words-with-real-world-examples-yes-again/>

Types of Features



Feature Methods



Feature Selection/Extraction

Feature Selection / Extraction

Curse of Dimensionality:

samples data needed to train classifier

function grows exponentially with dimensions

Overfitting and Generalization performance

What features best characterize class?

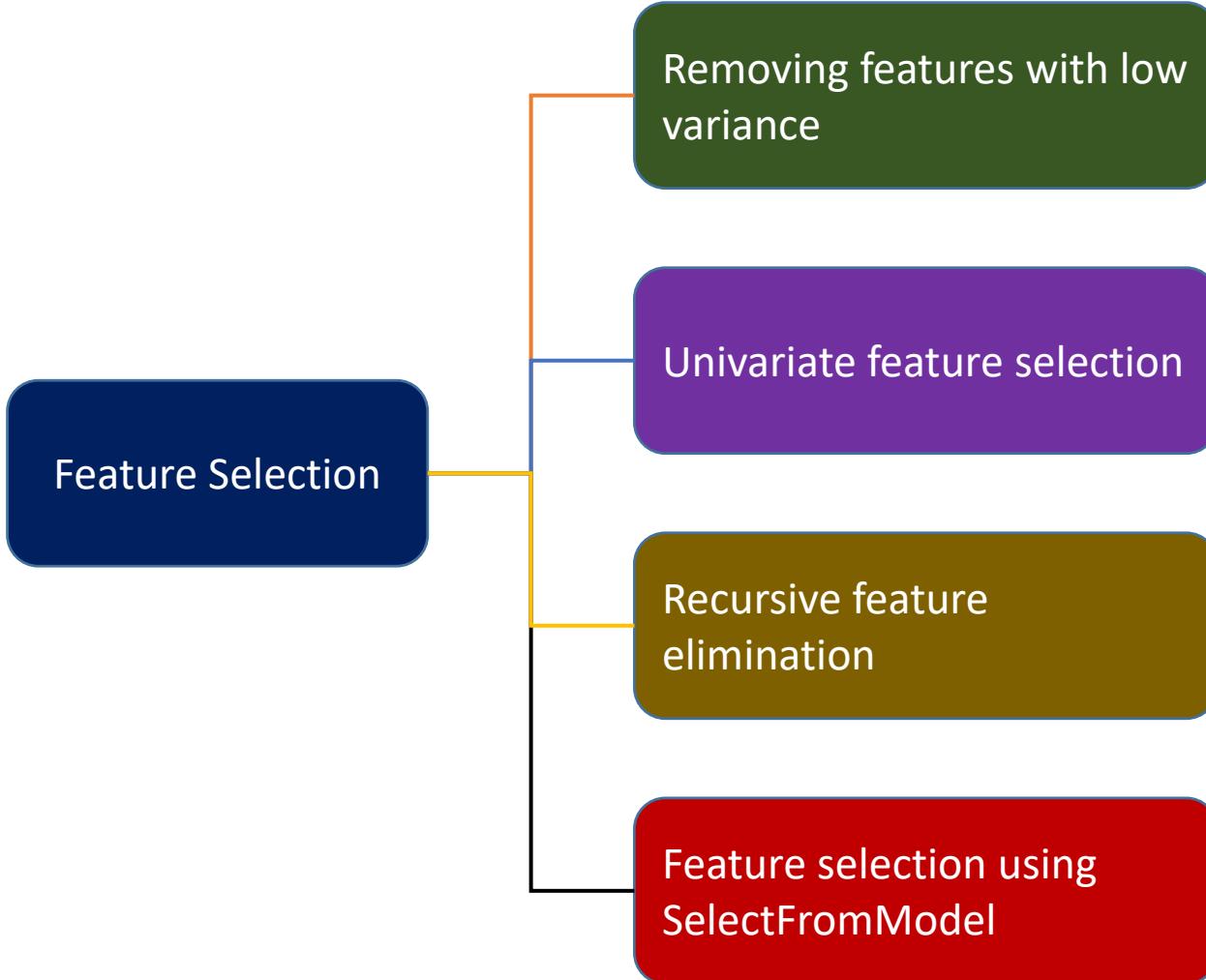
What words best characterize a document class

Shape / texture / color

What features critical for performance?

Subregions characterize protein function?

Feature Selection methods



First Order Statistical Features

First Order Statistical Features

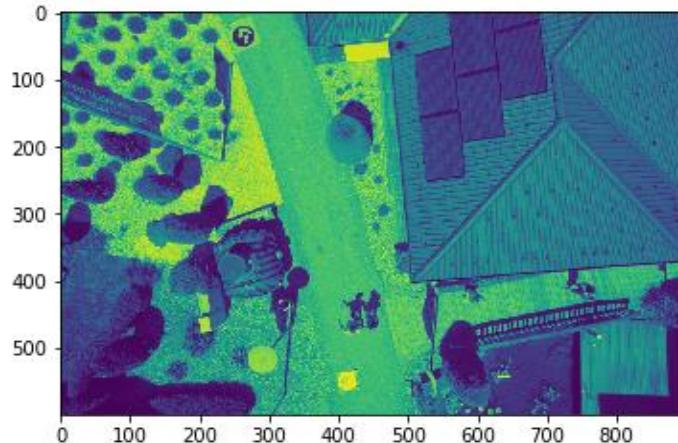
Mean(mean is the average value within the dataset)

Variance(is a measure of the histogram width.it represents the deviation of gray levels from the means)

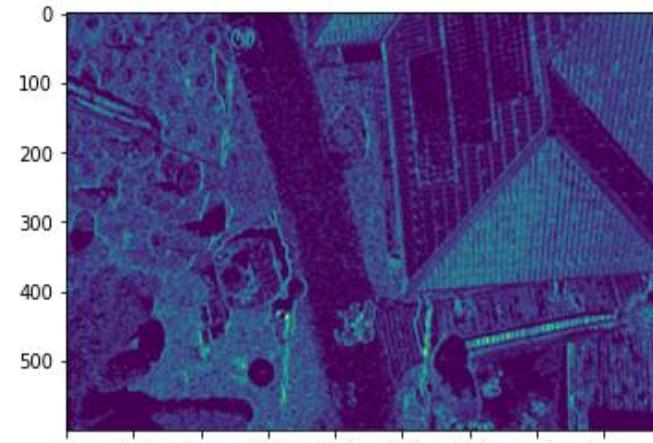
Skewness(is a measure of the degree of histogram asymmetry around the mean)

Kurtosis(is the measure of the histogram sharpness)

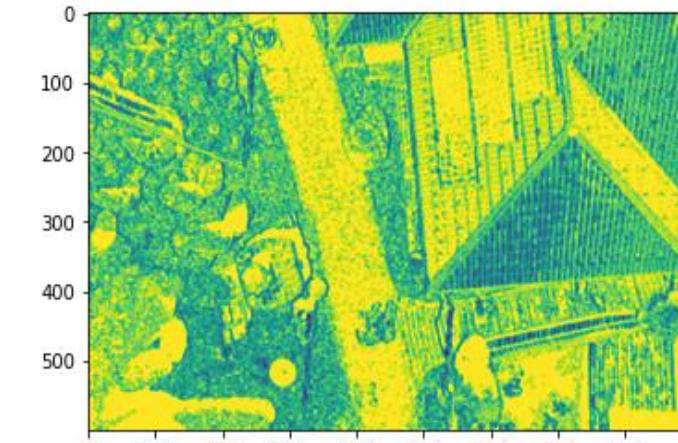
GLCM(grey level co-occurrence matrix)



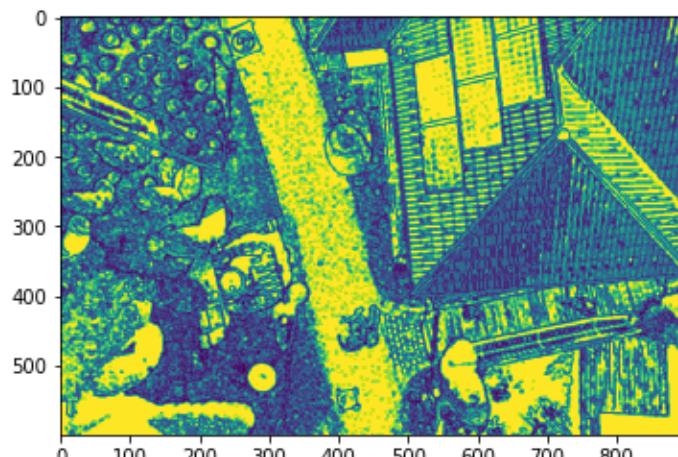
Main image



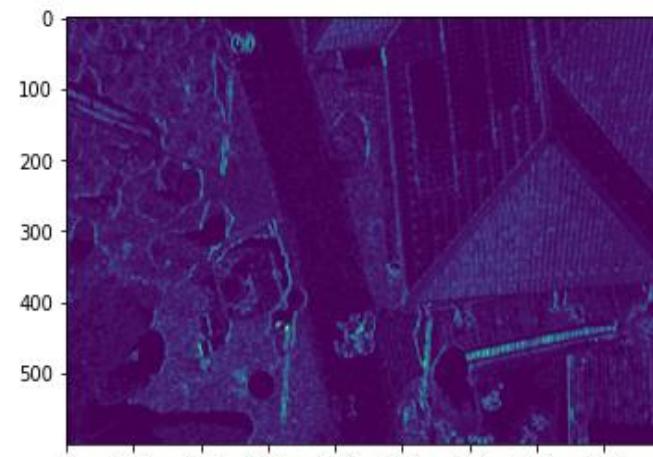
glcm_dissimilarity



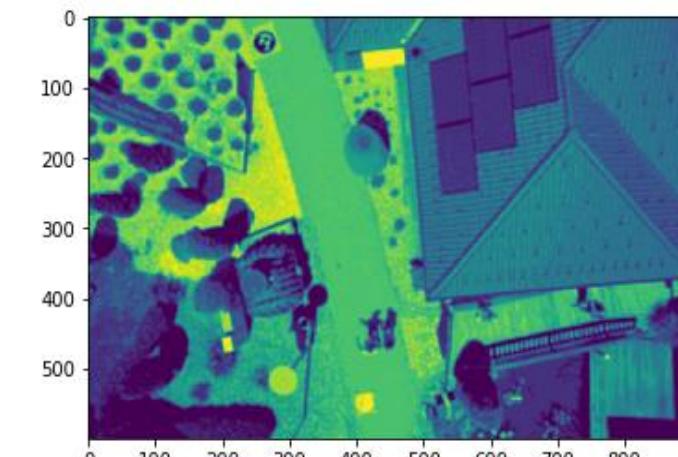
glcm_homogeneity



glcm_max



glcm_contrast

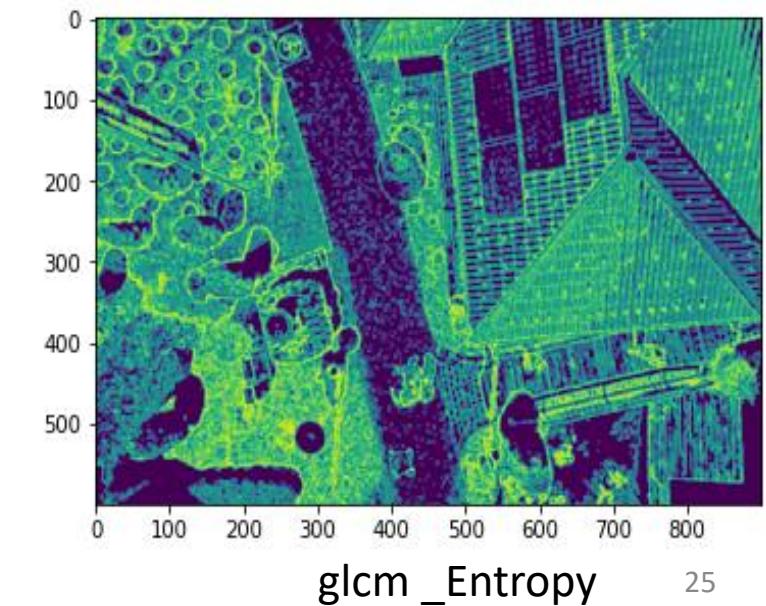
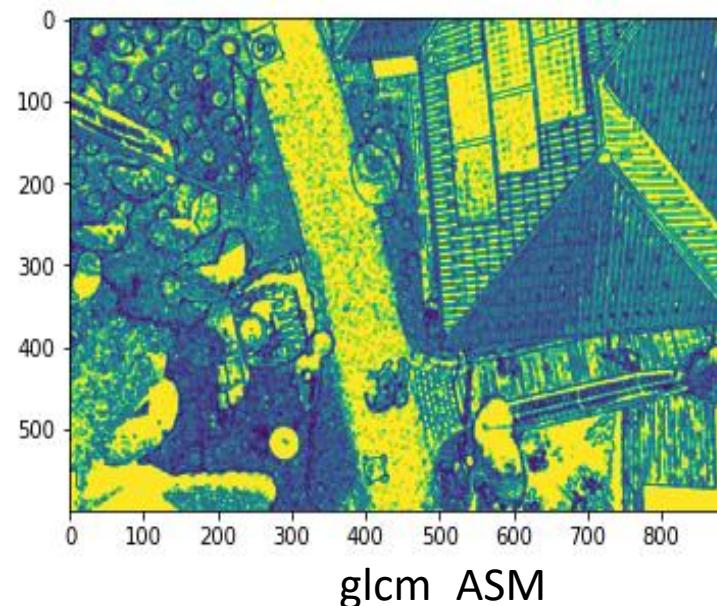
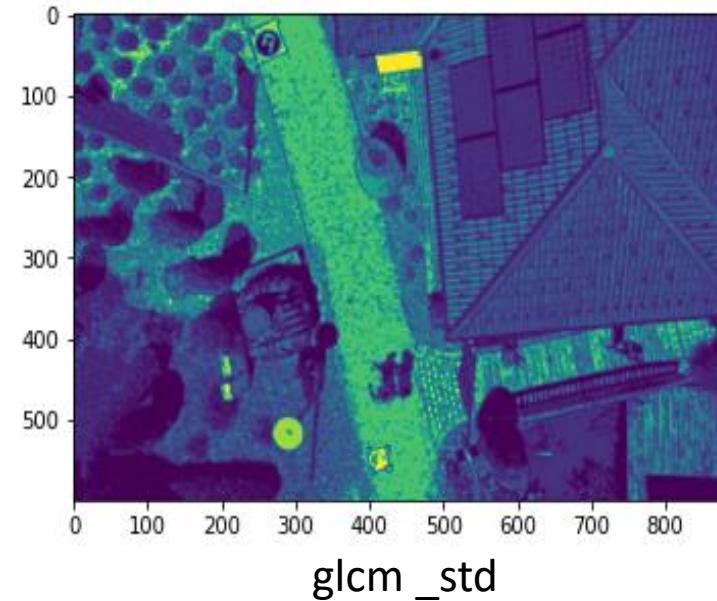


glcm_mean

GLCM(grey level co-occurrence matrix)

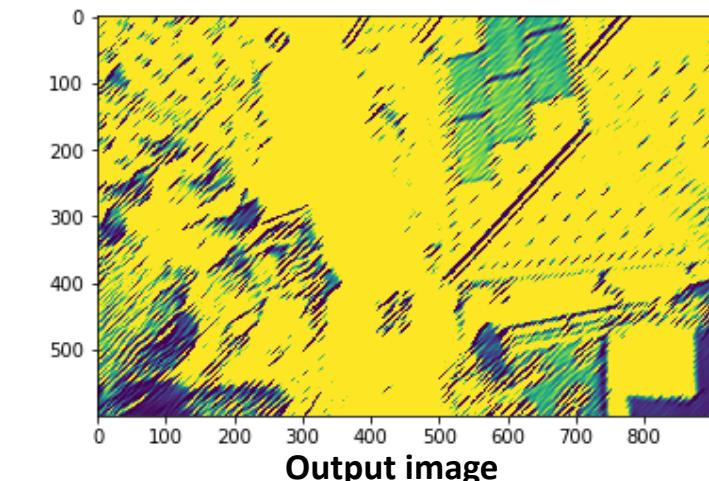
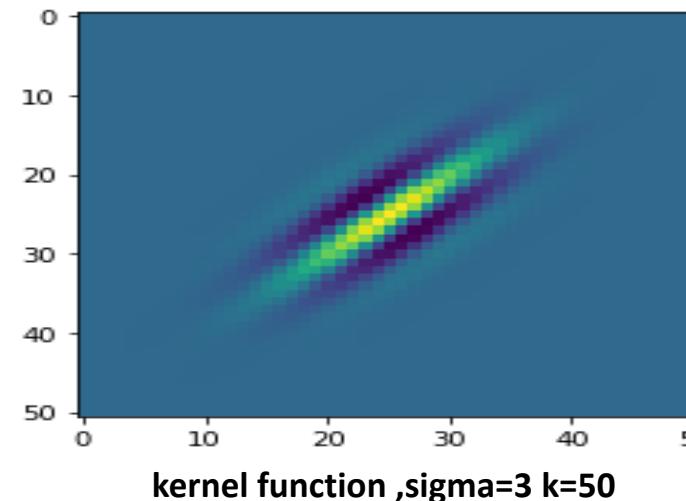
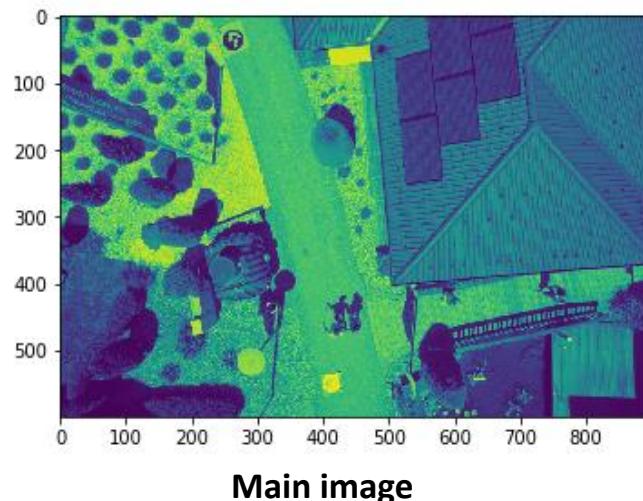
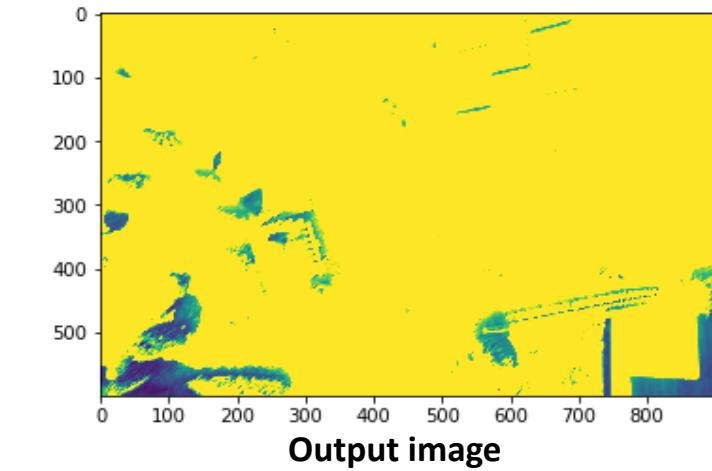
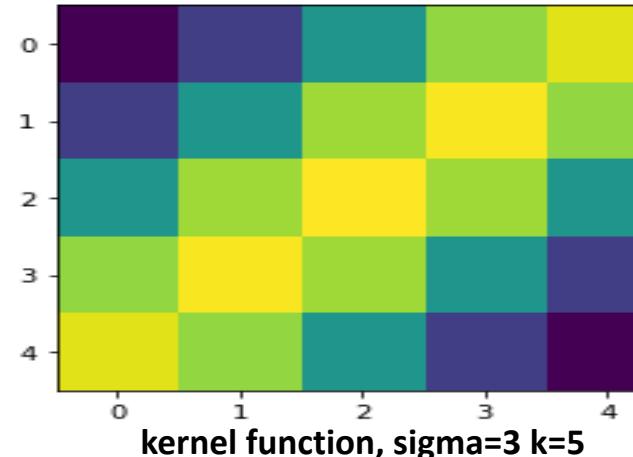
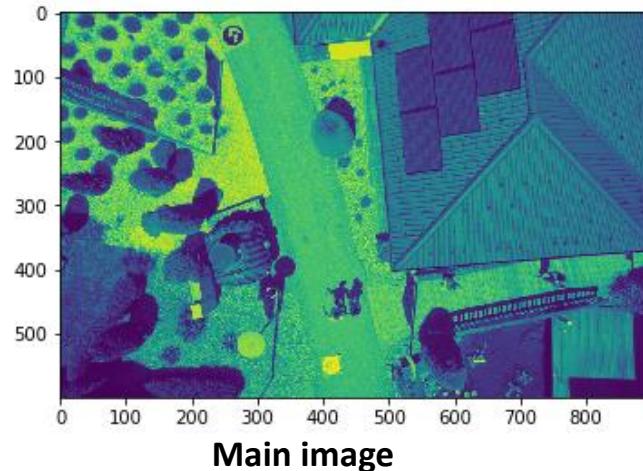
GLCM features

- Contrast
- Homogeneity
- Dissimilarity
- Std
- Mean
- Max
- ASM
- Entropy

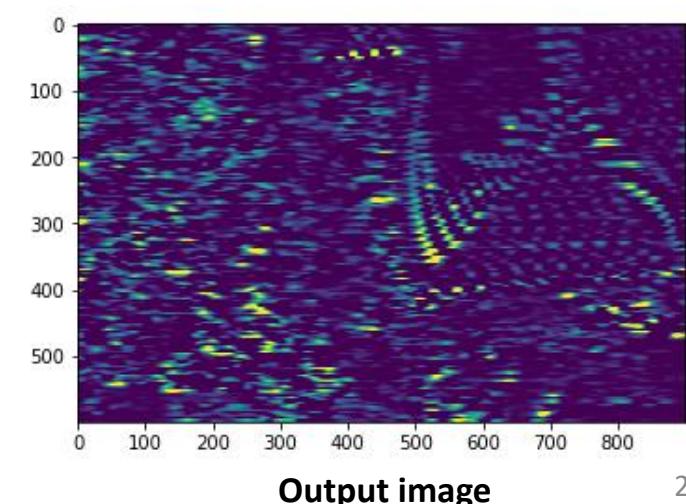
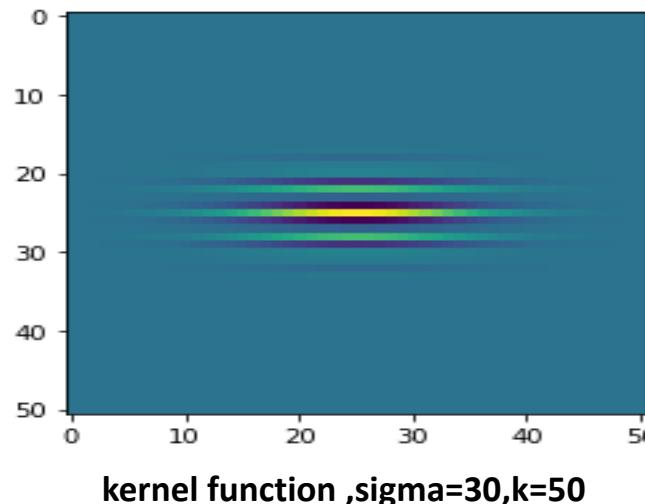
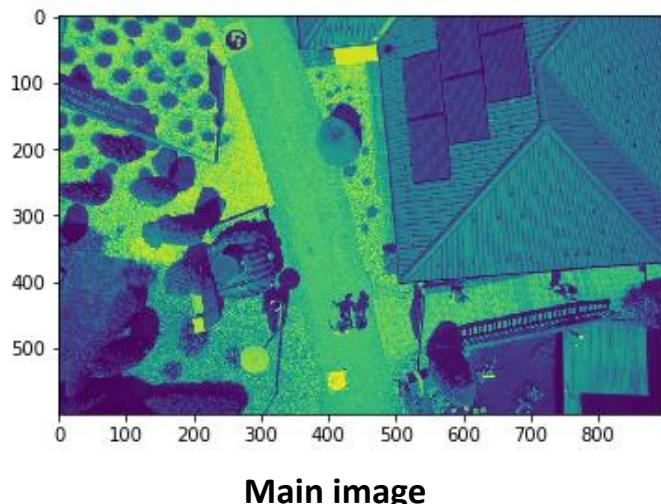
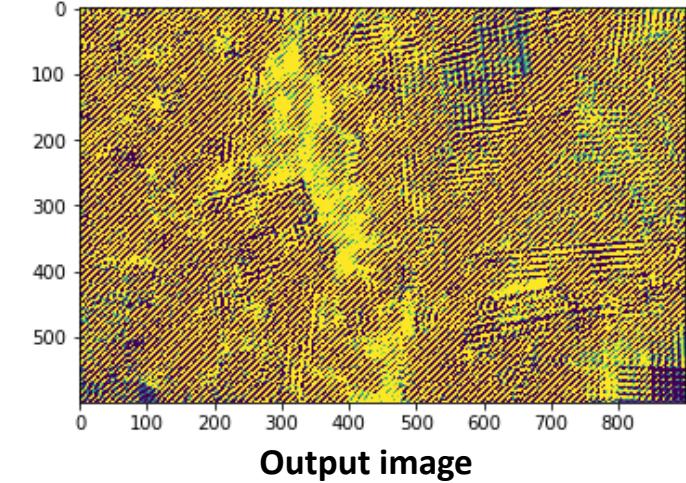
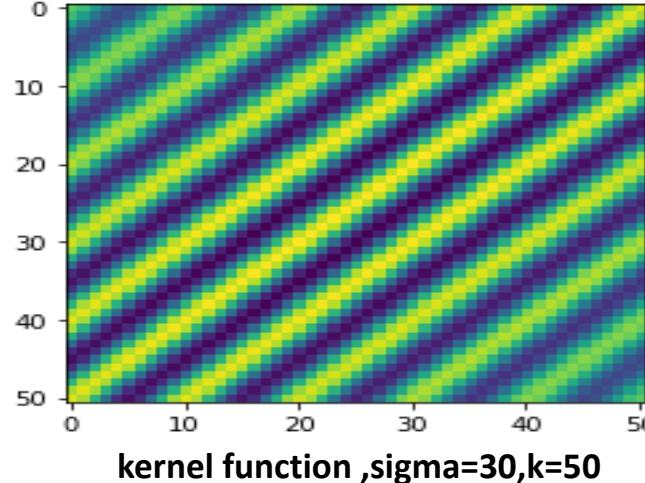
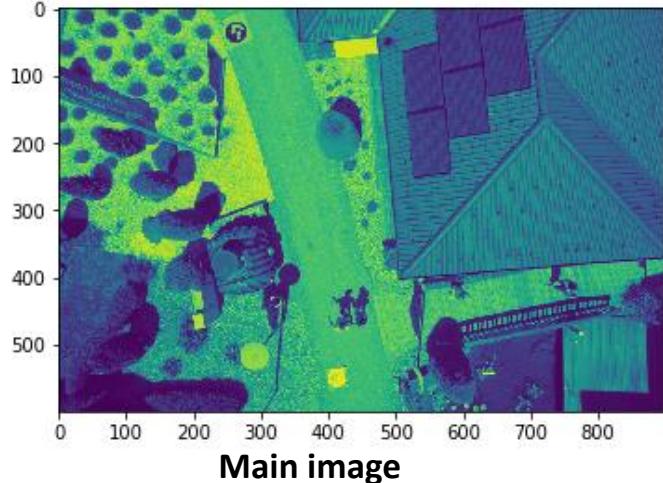


Gabor Filter (Feature Extraction)

For image processing and computer vision, Gabor filters are generally used in texture analysis, edge detection, feature extraction, etc. Gabor filters are special classes of bandpass filters, i.e., they allow a certain ‘band’ of frequencies and reject the others.



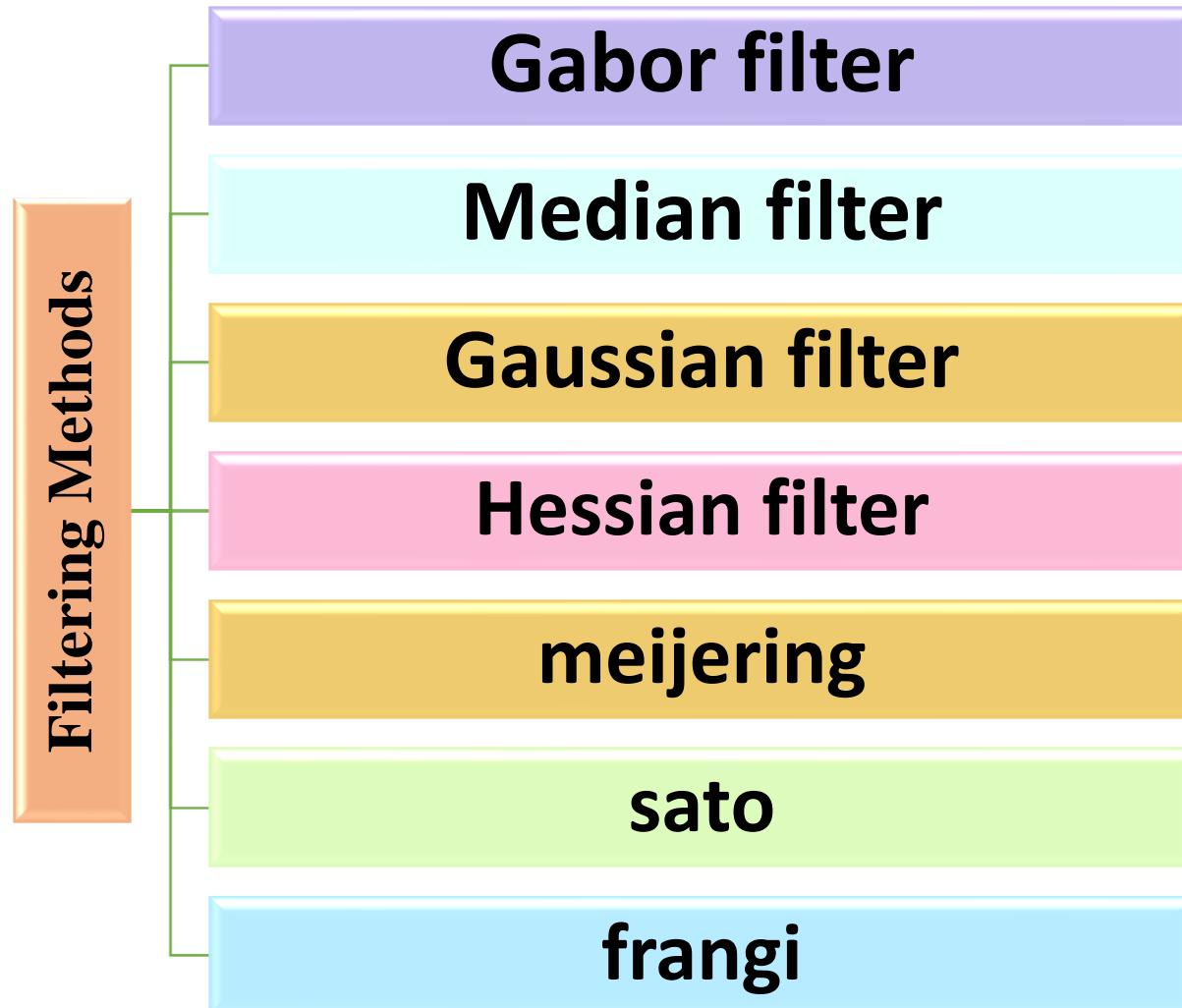
Gabor Filter (Feature Extraction)



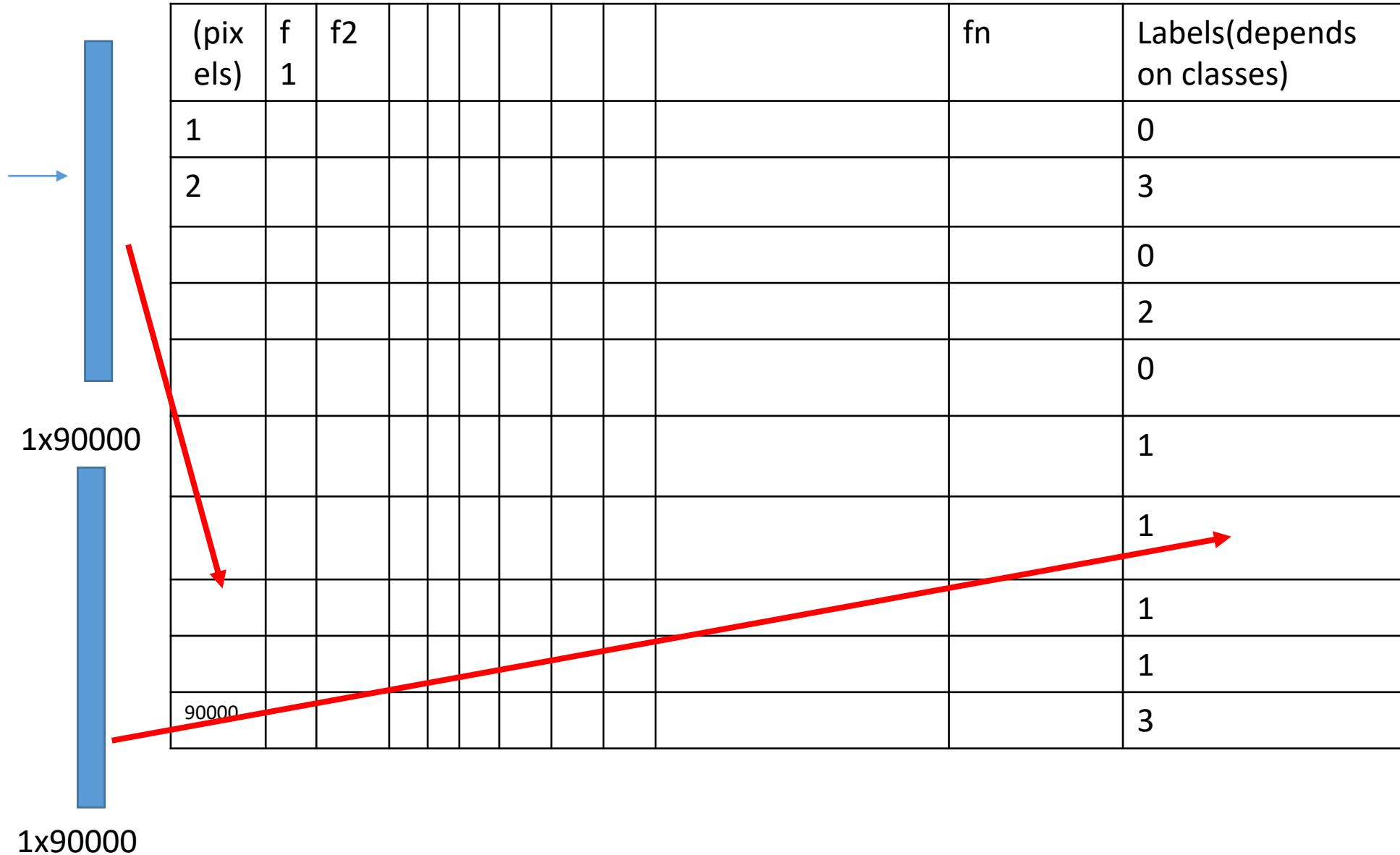
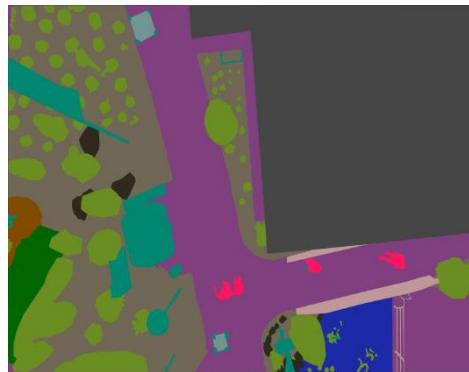
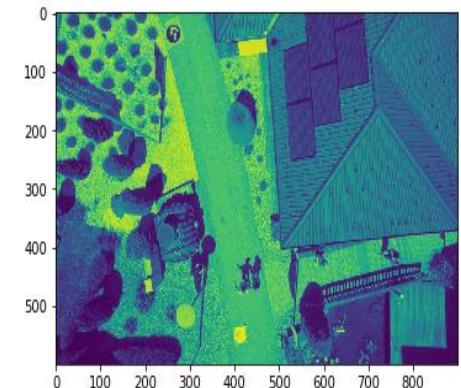
Detector and Descriptor (Feature Extraction)



Filtering Method (Feature Extraction)



Segmentation based on Image Features



Feature Scaling or Normalization Methods

- **Feature scaling** is a method used to standardize the range of independent variables or **features** of data. In data processing, it is also known as data **normalization** and is generally performed during the data preprocessing step.

▪ **Why Scaling**

- Most of the times, your dataset will contain features highly varying in magnitudes, units, and range. But since most of the machine learning algorithms use Euclidian distance between two data points in their computations, this is a problem.
- If left alone, these algorithms only take in the magnitude of features neglecting the units. The results would vary greatly between different units, 5kg, and 5000gms. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes. To suppress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling.

Feature Scaling or Normalization Methods

- In this lecture we explore 3 methods of feature scaling that are implemented in scikit-learn:

1
StandardScaler

2
MinMaxScaler

3
RobustScaler

4
Normalizer

1. Standard Scaler

- The StandardScaler assumes your data is normally distributed within each feature and will scale them such that the distribution is now centered around 0, with a standard deviation of 1.
- The mean and standard deviation are calculated for the feature and then the feature is scaled based on:

$$z = \frac{x - \mu}{\sigma}$$

μ =Mean
 σ = standard deviation

Feature Scaling or Normalization Methods

2. MinMaxScaler

- The MinMaxScaler is probably the most famous scaling algorithm, and it essentially shrinks the range such that the range is now between 0 and 1 (or -1 to 1 if there are negative values).
- This scaler works better for cases in which the standard scaler might not work so well. If the distribution is not Gaussian or the standard deviation is very small, the min-max scaler works better.
- However, it is sensitive to outliers, so if there are outliers in the data, you might want to consider the Robust Scaler below. follows the following formula for each feature:

$$X_{new} = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

Syntax :
from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()
scaled_df = scaler.fit_transform(df)

Feature Scaling or Normalization Methods

3. RobustScaler

- The RobustScaler uses a similar method to the Min-Max scaler but it instead uses the interquartile range, rather than the min-max, so that it is robust to outliers. Therefore it follows the formula:
- $\frac{x_i - Q1(x)}{Q3(x) - Q1(x)}$
- For each feature. Of course, this means it is using the less of the data for scaling so it's more suitable for when there are outliers in the data.

Syntax:

```
from sklearn import preprocessing  
scaler = preprocessing.RobustScaler()  
robust_scaled_df = scaler.fit_transform(x)
```

Feature Scaling or Normalization Methods

4. Normalizer

- The normalizer scales each value by dividing each value by its magnitude in n-dimensional space for n number of features. Say your features were x, y, and z Cartesian co-ordinates your scaled value for x would be:
- Each point is now within 1 unit of the origin on this Cartesian coordinate system.
- $X = xi / np.sqrt(xi**2 + yi**2 + zi**2)$

Syntax :

```
from sklearn import preprocessing  
scaler = preprocessing.Normalizer()  
scaled_df = scaler.fit_transform(df)
```

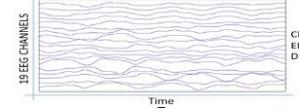
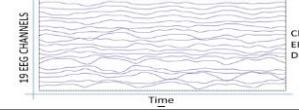
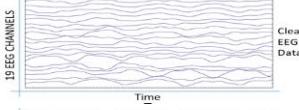
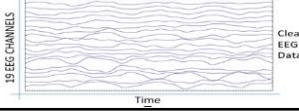
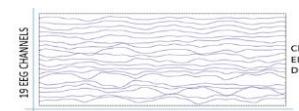
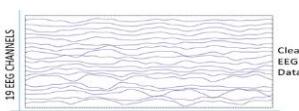
Feature Scaling or Normalization Methods

- **Where and when to Scale**
- **Rule of thumb I follow here is an algorithm that computes distance or assumes normality, scales your features!!!**

Some examples of algorithms where feature scaling matters are:

1. k-nearest neighbors with a Euclidean distance measure is sensitive to magnitudes and hence should be scaled for all features to weigh in equally.
2. Scaling is critical while performing Principal Component Analysis(PCA). PCA tries to get the features with maximum variance and the variance is high for high magnitude features. This skews the PCA towards high magnitude features.
3. We can speed up gradient descent by scaling. This is because θ will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to the optimum when the variables are very uneven.
4. Tree-based models are not distance-based models and can handle varying ranges of features. Hence, Scaling is not required while modeling trees.
5. Algorithms like Linear Discriminant Analysis(LDA), Naive Bayes is by design equipped to handle this and gives weights to the features accordingly. Performing a feature scaling in these algorithms may not have much effect.

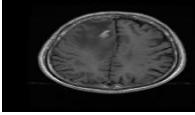
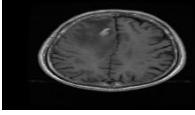
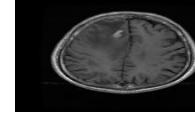
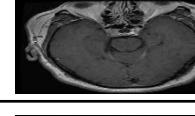
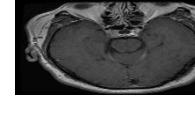
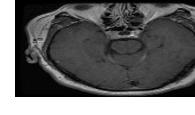
Basic Dimensions in 1D-CNN for Training and Testing

No of samples	Classes	Dimension	Labels
1	Patient1	19x256	
2	Patient1	19x256	
	Patient1	---	0
	Patient1	---	0
50	Patient1	19x256	
51	Patient2	19x256	
	Patient2	19x256	
	Patient2	---	1
99	Patient2	---	1
100	Patient2	19x256	

Total samples x timestep x features
100 x 256 x 19

Training samples
80 x 256 x 19
Testing samples
20 x 256 x 19

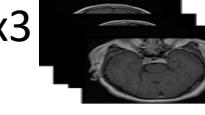
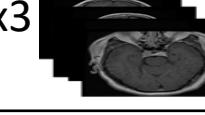
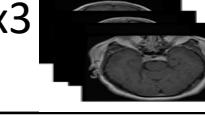
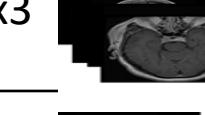
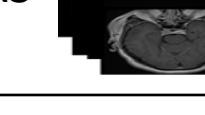
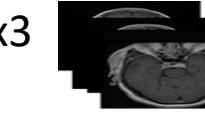
Basic Dimensions in 2D-CNN for Training and Testing

No of samples	Classes	Dimension	Labels
1	Patient1	224x224x3 	0
2	Patient1	224x224x3 	0
	Patient1	---	0
	Patient1	---	0
50	Patient1	224x224x3 	0
51	Patient2	224x224x3 	1
	Patient2	224x224x3 	1
	Patient2	---	1
99	Patient2	---	1
100	Patient2	224x224x3 	1

Total samples x row x cols x channels
100 x 224x224x3

Training samples
80x224x224x3
Testing samples
20 x 224x224x3

Basic Dimensions in 3D-CNN for Training and Testing

No of samples	Classes	Dimension	Labels
1	Patient1	30x224x224x3	 0
2	Patient1	30x224x224x3	 0
	Patient1	---	0
	Patient1	---	0
50	Patient1	30x224x224x3	 0
51	Patient2	30x224x224x3	 1
	Patient2	30x224x224x3	 1
	Patient2	---	1
99	Patient2	---	1
100	Patient2	30x224x224x3	 1

Total samples x slices rows x cols x channels
100 x30x 224x224x3

Training samples
80x30x224x224x3

Testing samples
20 x30x 224x224x3

Machine Learning Applications

Information Fusion 52 (2019) 1–12



ELSEVIER

Contents lists available at [ScienceDirect](#)

Information Fusion

journal homepage: www.elsevier.com/locate/inffus



Full Length Article

Ensembles for feature selection: A review and future trends



Verónica Bolón-Canedo, Amparo Alonso-Betanzos*

CITIC Research Centre on Information and Communication Technology, University of A Coruña, Campus de Elviña s/n 15071, A Coruña, Spain

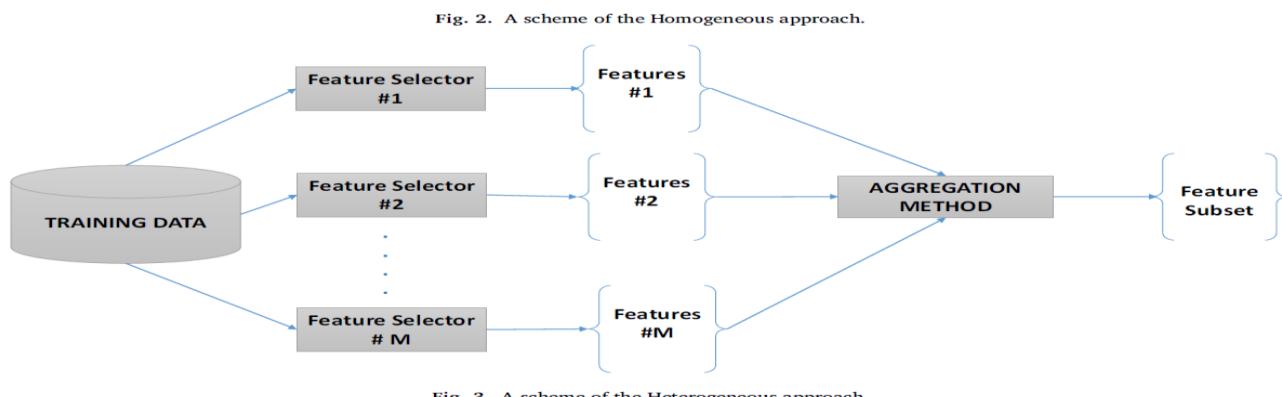
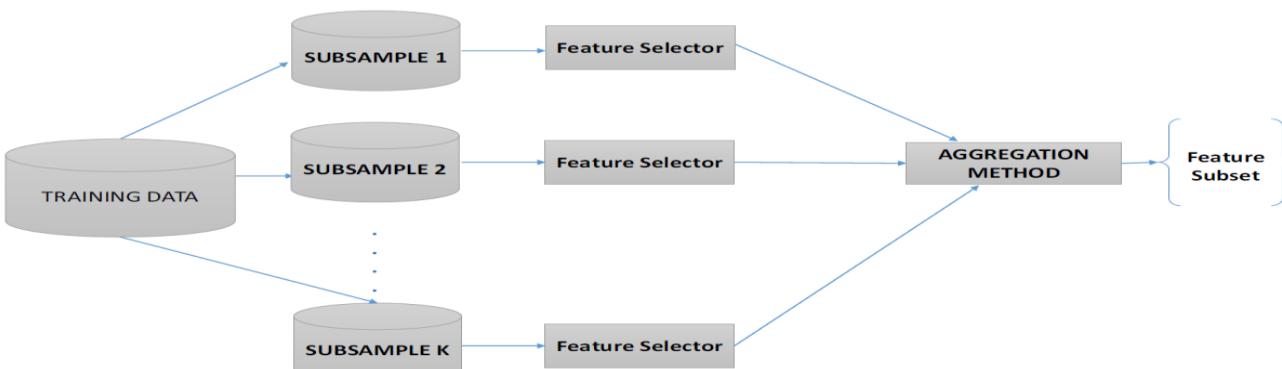
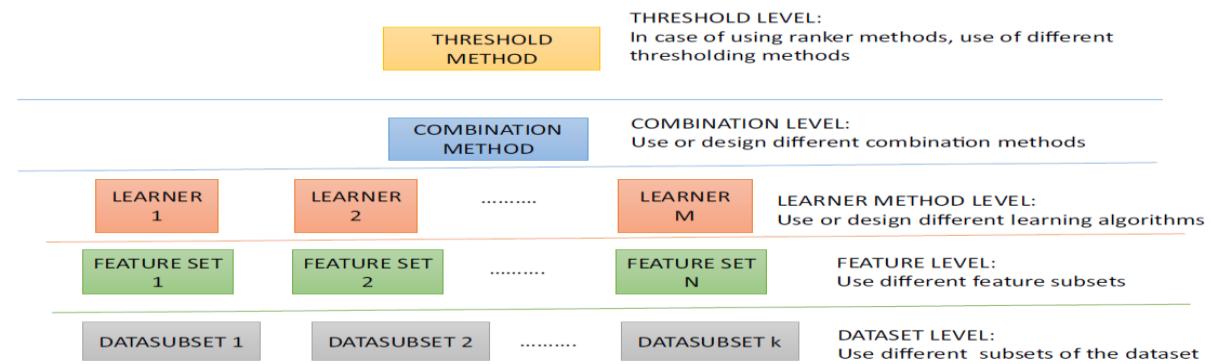
ARTICLE INFO

Keywords:
Ensemble learning
Feature selection

ABSTRACT

Ensemble learning is a prolific field in Machine Learning since it is based on the assumption that combining the output of multiple models is better than using a single model, and it usually provides good results. Normally, it has been commonly employed for classification, but it can be used to improve other disciplines such as feature selection. Feature selection consists of selecting the relevant features for a problem and discard those irrelevant or redundant, with the main goal of improving classification accuracy. In this work, we provide the reader with the basic concepts necessary to build an ensemble for feature selection, as well as reviewing the up-to-date advances and commenting on the future trends that are still to be faced.

Machine Learning Applications



Information Fusion 52 (2019) 1–12



Full Length Article

Ensembles for feature selection: A review and future trends

Verónica Bolón-Canedo, Amparo Alonso-Betanzos*

CITIC Research Centre on Information and Communication Technology, University of A Coruña, Campus de Elviña s/n 15071, A Coruña, Spain



ARTICLE INFO

Keywords:
Ensemble learning
Feature selection

ABSTRACT

Ensemble learning is a prolific field in Machine Learning since it is based on the assumption that combining the output of multiple models is better than using a single model, and it usually provides good results. Normally, it has been commonly employed for classification, but it can be used to improve other disciplines such as feature selection. Feature selection consists of selecting the relevant features for a problem and discard those irrelevant or redundant, with the main goal of improving classification accuracy. In this work, we provide the reader with the basic concepts necessary to build an ensemble for feature selection, as well as reviewing the up-to-date advances and commenting on the future trends that are still to be faced.

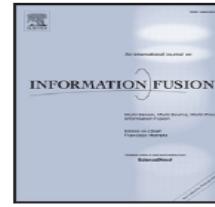
Machine Learning Applications



Contents lists available at [ScienceDirect](#)

Information Fusion

journal homepage: www.elsevier.com/locate/inffus



Full Length Article

Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities



Marinka Zitnik^{a,*}, Francis Nguyen^{b,c}, Bo Wang^d, Jure Leskovec^{a,e,*}, Anna Goldenberg^{f,g,h,*}, Michael M. Hoffman^{b,c,g,h,*}

^a Department of Computer Science, Stanford University, Stanford, CA, USA

^b Department of Medical Biophysics, University of Toronto, Toronto, ON, Canada

^c Princess Margaret Cancer Centre, Toronto, ON, Canada

^d Hikvision Research Institute, Santa Clara, CA, USA

^e Chan Zuckerberg Biohub, San Francisco, CA, USA

^f Genetics & Genome Biology, SickKids Research Institute, Toronto, ON, Canada

^g Department of Computer Science, University of Toronto, Toronto, ON, Canada

^h Vector Institute, Toronto, ON, Canada

ARTICLE INFO

Keywords:

Computational biology

Personalized medicine

Systems biology

Heterogeneous data

Machine learning

ABSTRACT

New technologies have enabled the investigation of biology and human health at an unprecedented scale and in multiple dimensions. These dimensions include a myriad of properties describing genome, epigenome, transcriptome, microbiome, phenotype, and lifestyle. No single data type, however, can capture the complexity of all the factors relevant to understanding a phenomenon such as a disease. Integrative methods that combine data from multiple technologies have thus emerged as critical statistical and computational approaches. The key challenge in developing such approaches is the identification of effective models to provide a comprehensive and relevant systems view. An ideal method can answer a biological or medical question, identifying important features and predicting outcomes, by harnessing heterogeneous data across several dimensions of biological variation. In this Review, we describe the principles of data integration and discuss current methods and available implementations. We provide examples of successful data integration in biology and medicine. Finally, we discuss current challenges in biomedical integrative methods and our perspective on the future development of the field.

Machine Learning Applications

M. Zitnik et al.

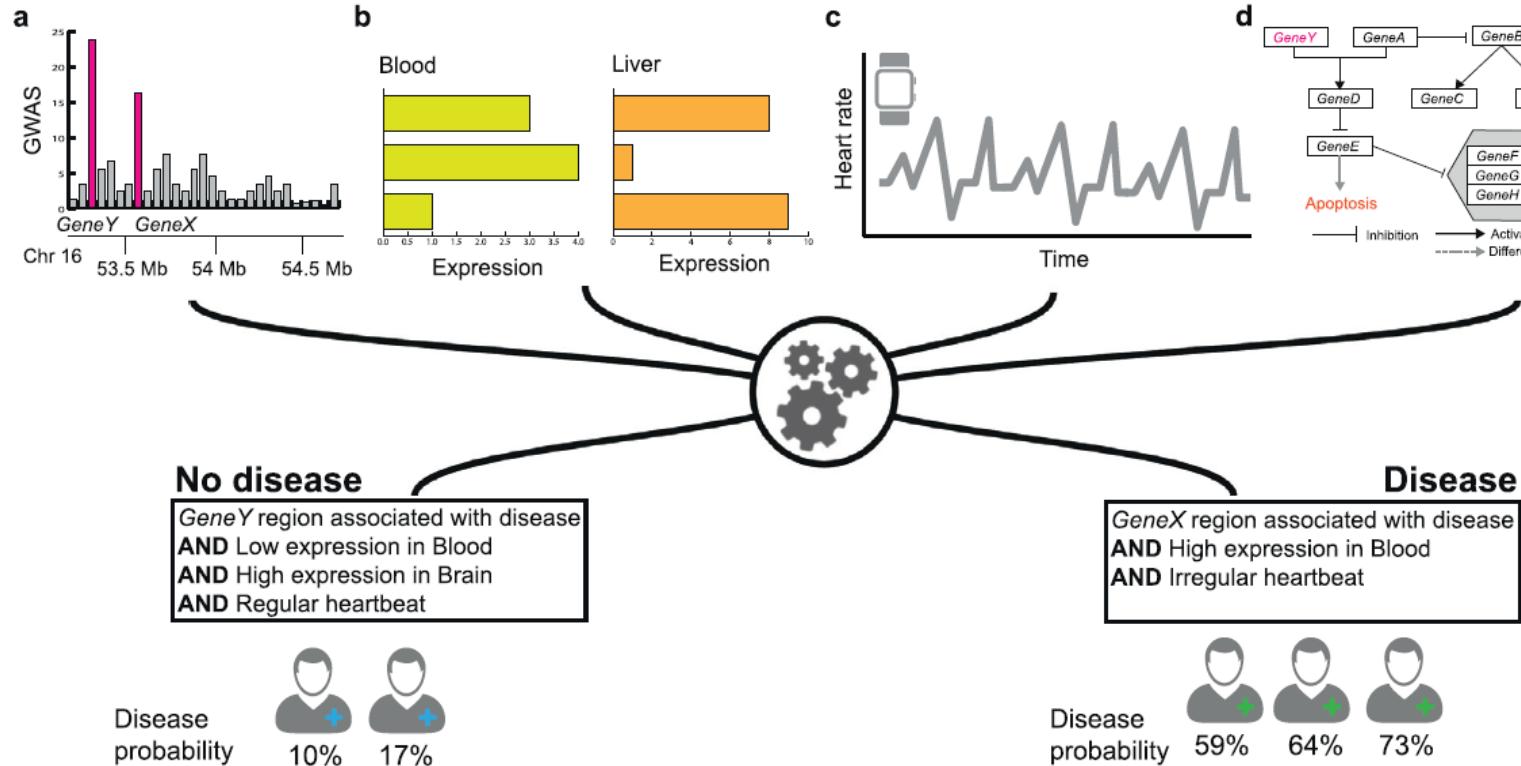


Fig. 1. The importance of data integration in biomedicine. Considering variation in only a single data type can miss many important patterns that can only be observed by considering multiple levels of biomedical data. Shown is a hypothetical example using disease diagnostics as a point of interest. When a new patient arrives to the clinic, (a) domain experts sequence the patient's genome and compare it with a database to identify mutations and disease-causing genes, (b) perform laboratory tests using tissue samples, and (c) process information about the patient's behavior and lifestyle. (d) The patient's genomic, transcriptomic, and lifestyle information is combined with curated databases of biomedical knowledge (e.g., disease and metabolic pathways). Finally, a machine learning algorithm predicts probability that the patient will develop a particular disease in near future. To make accurate prediction, the machine learning model needs to use many different types of data. This example illustrates that accurate prediction can only be made by analyzing multiple types of patient's data.

ARTICLE INFO

Keywords:
Computational biology
Personalized medicine
Systems biology
Heterogeneous data
Machine learning

ABSTRACT

New technologies have enabled the investigation of biology and human health at an unprecedented scale and in multiple dimensions. These dimensions include a myriad of properties describing genome, epigenome, transcriptome, microbiome, phenotype, and lifestyle. No single data type, however, can capture the complexity of all the factors relevant to a particular disease or trait. As a result, the need for methods that can integrate multiple data modalities and link them to a particular disease or trait has increased. Machine learning approaches have thus emerged as critical ones as they can handle large amounts of heterogeneous data and predict outcomes. In this Review, we describe the principles of data integration and discuss current methods and available implementations. We provide examples of successful data integration in biology and medicine. Finally, we discuss current challenges in biomedical integrative methods and our perspective on the future development of the field.

Machine Learning Applications

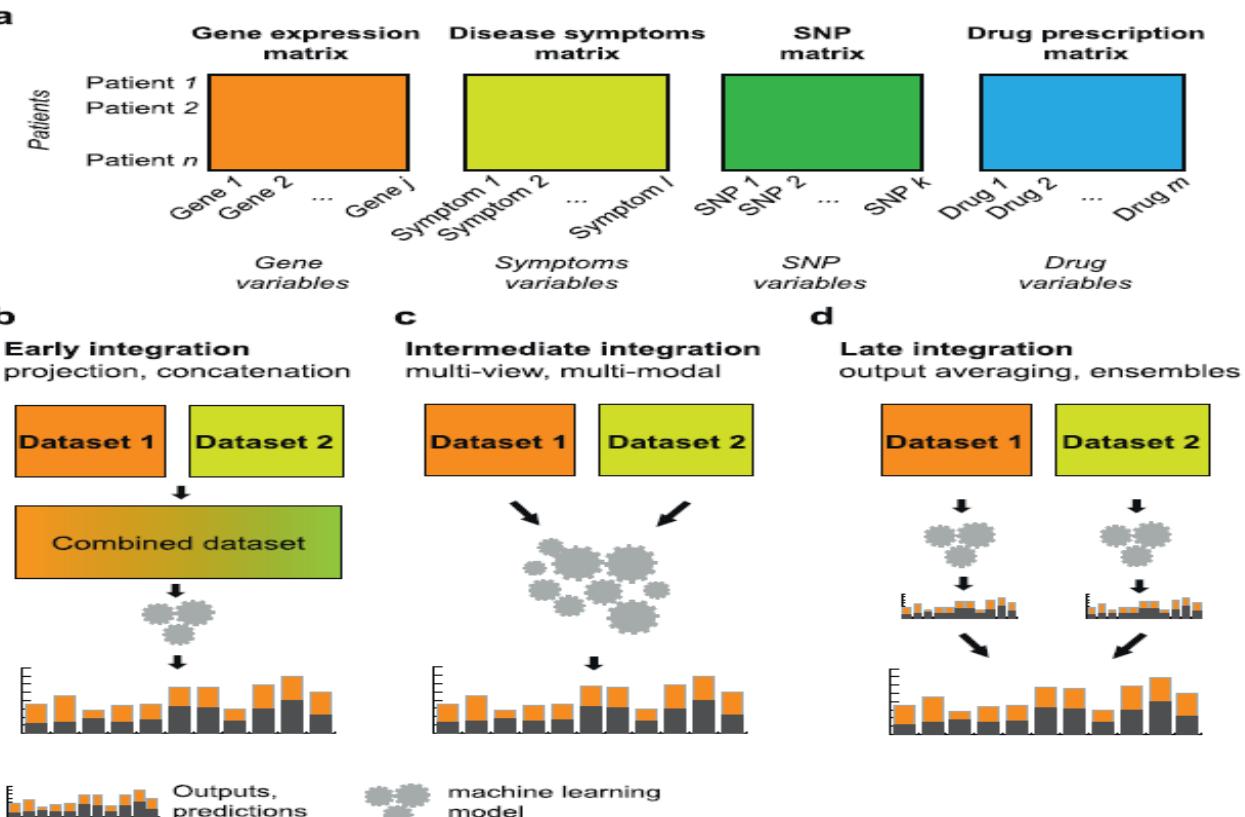


Fig. 2. Categorization of approaches for data integration. (a) Examples of multi-omics data about patients. (b-d) Data integration approaches can be divided into three categories. (b) *Early integration approaches* involve combining datasets from different data types at the raw or processed level before analysis and prediction. (c) *Intermediate integration approaches* transform or map the underlying datasets at the same time as they estimate model parameters. (d) *Late integration approaches* perform analysis on each dataset independently, which is followed by integration of the resulting models to generate predictions, e.g., prognosis for a particular patient. SNP, single-nucleotide polymorphism.

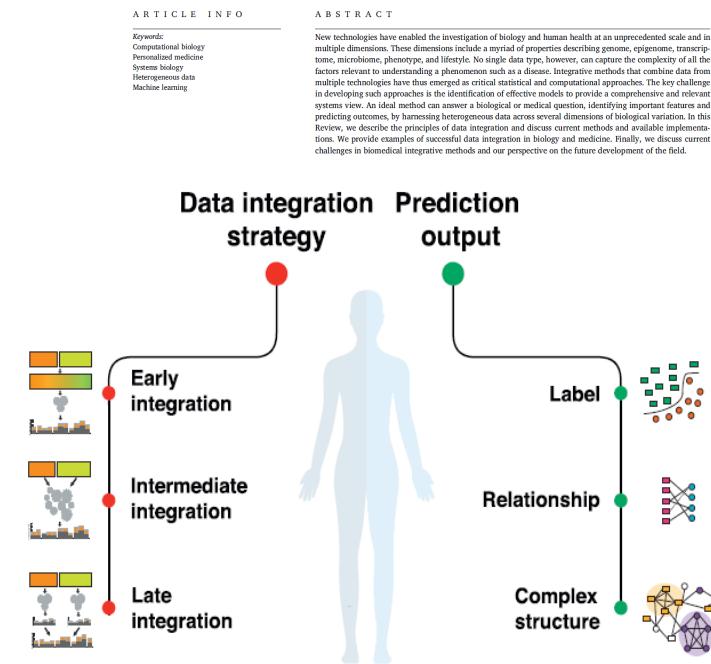


Fig. 3. Data integration. Data integration approaches combine multiples sources of information in a statistically meaningful way to provide a comprehensive analysis of biomedical data. Broadly, existing approaches use three distinct strategies (i.e., early, intermediate, and late integration; see also Fig. 2) and produce three types of prediction outputs (i.e., a label representing probability of an entity belonging to a given class; a relationship representing probability of an association between two entities; and a complex structure, such as an inferred network or a partitioning of entities into groups).

Machine Learning Applications

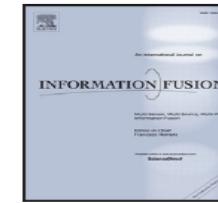
Information Fusion 50 (2019) 92–111



Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus



Full Length Article

Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0



Alberto Diez-Olivan ^a, Javier Del Ser ^{a,b,c,*}, Diego Galar ^{a,d}, Basilio Sierra ^e

^a TECNALIA, Donostia-San Sebastián 20009, Spain

^b Department of Communications Engineering, University of the Basque Country (UPV/EHU), Bilbao 48013, Spain

^c Basque Center for Applied Mathematics (BCAM), Bilbao, Bizkaia 48009, Spain

^d Department of Civil, Environmental and Natural Resources Engineering, Operation, Maintenance and Acoustics, Luleå University of Technology, Luleå, Sweden

^e Department of Computer Sciences and Artificial Intelligence, University of the Basque Country (UPV/EHU), Donostia-San Sebastián 20018, Spain

ARTICLE INFO

Keywords:

Data-driven prognosis
Data fusion
Machine learning
Industry 4.0

ABSTRACT

The so-called “smartization” of manufacturing industries has been conceived as the fourth industrial revolution or Industry 4.0, a paradigm shift propelled by the upsurge and progressive maturity of new Information and Communication Technologies (ICT) applied to industrial processes and products. From a data science perspective, this paradigm shift allows extracting relevant knowledge from monitored assets through the adoption of intelligent monitoring and data fusion strategies, as well as by the application of machine learning and optimization methods. One of the main goals of data science in this context is to effectively predict abnormal behaviors in industrial machinery, tools and processes so as to anticipate critical events and damage, eventually causing important economical losses and safety issues. In this context, data-driven prognosis is gradually gaining attention in different industrial sectors. This paper provides a comprehensive survey of the recent developments in data fusion and machine learning for industrial prognosis, placing an emphasis on the identification of research trends, niches of opportunity and unexplored challenges. To this end, a principled categorization of the utilized feature extraction techniques and machine learning methods will be provided on the basis of its intended purpose: analyze what caused the failure (descriptive), determine when the monitored asset will fail (predictive) or decide what to do so as to minimize its impact on the industry at hand (prescriptive). This threefold analysis, along with a discussion on its hardware and software implications, intends to serve as a stepping stone for future researchers and practitioners to join the community investigating on this vibrant field.

Machine Learning Applications

Information Fusion 50 (2019) 92–111



Contents lists available at ScienceDirect



Information Fusion

journal homepage: www.elsevier.com/locate/inffus



Full Length Article

Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0

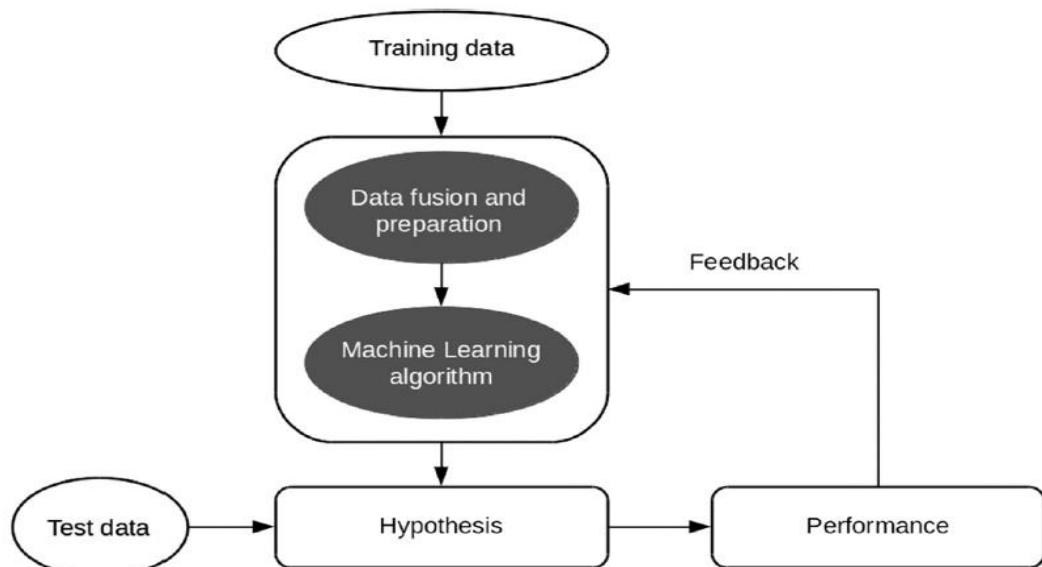


Fig. 1. Generic schematic diagram of a machine learning process.

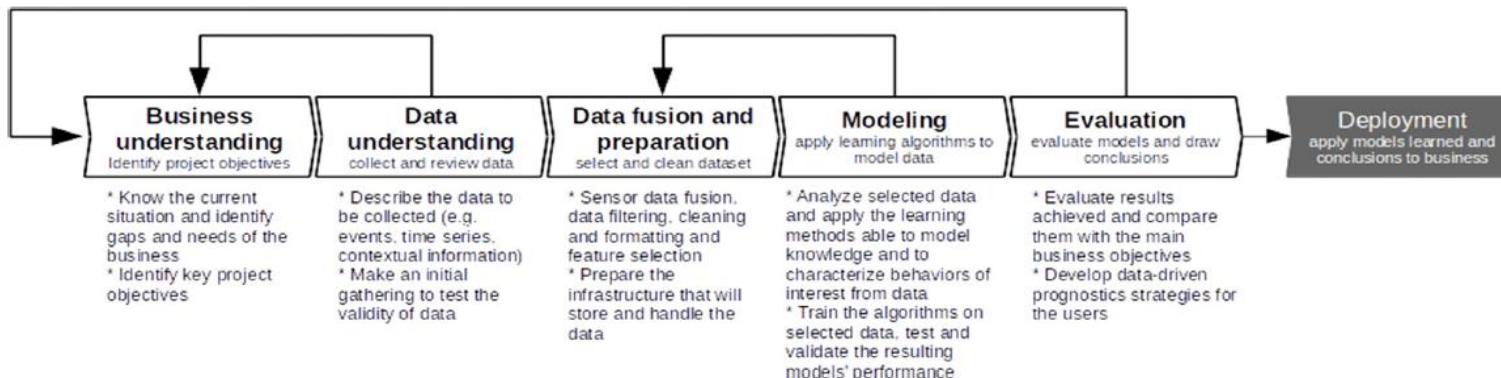


Fig. 2. The CRISP methodology for data-driven industrial prognosis.

ARTICLE INFO

Keywords:
Data-driven prognosis
Data fusion
Machine learning
Industry 4.0

^aTECNALIA, Donostia-San Sebastián 20009, Spain

^bDepartment of Communications Engineering, University of the Basque Country (UPV/EHU), Bilbao 48013, Spain

^cBasque Center for Applied Mathematics (BCAM), Bilbao, Bizkaia 48009, Spain

^dDepartment of Civil, Environmental and Natural Resources Engineering, Operation, Maintenance and Acoustics, Luleå University of Technology, Luleå, Sweden

^eDepartment of Computer Sciences and Artificial Intelligence, University of the Basque Country (UPV/EHU), Donostia-San Sebastián 20018, Spain

ABSTRACT

The so-called “smartization” of manufacturing industries has been conceived as the fourth industrial revolution or Industry 4.0, a paradigm shift propelled by the upsurge and progressive maturity of new Information and Communication Technologies (ICT) applied to industrial processes and products. From a data science perspective, this paradigm shift allows extracting relevant knowledge from monitored assets through the adoption of intelligent monitoring and data fusion strategies, as well as by the application of machine learning and optimization methods. One of the main goals of data science in this context is to effectively predict abnormal behaviors in industrial machinery, tools and processes so as to anticipate critical events and damage, eventually causing important economical losses and safety issues. In this context, data-driven prognosis is gradually gaining attention in different industrial sectors. This paper provides a comprehensive survey of the recent developments in data fusion and machine learning for industrial prognosis, placing an emphasis on the identification of research trends, niches of opportunity and unexplored challenges. To this end, a principled categorization of the utilized feature extraction techniques and machine learning methods will be provided on the basis of its intended purpose: analyze what caused the failure (descriptive), determine when the monitored asset will fail (predictive) or decide what to do so as to minimize its impact on the industry at hand (prescriptive). This threefold analysis, along with a discussion on its hardware and software implications, intends to serve as a stepping stone for future researchers and practitioners to join the community investigating on this vibrant field.

Machine Learning Applications

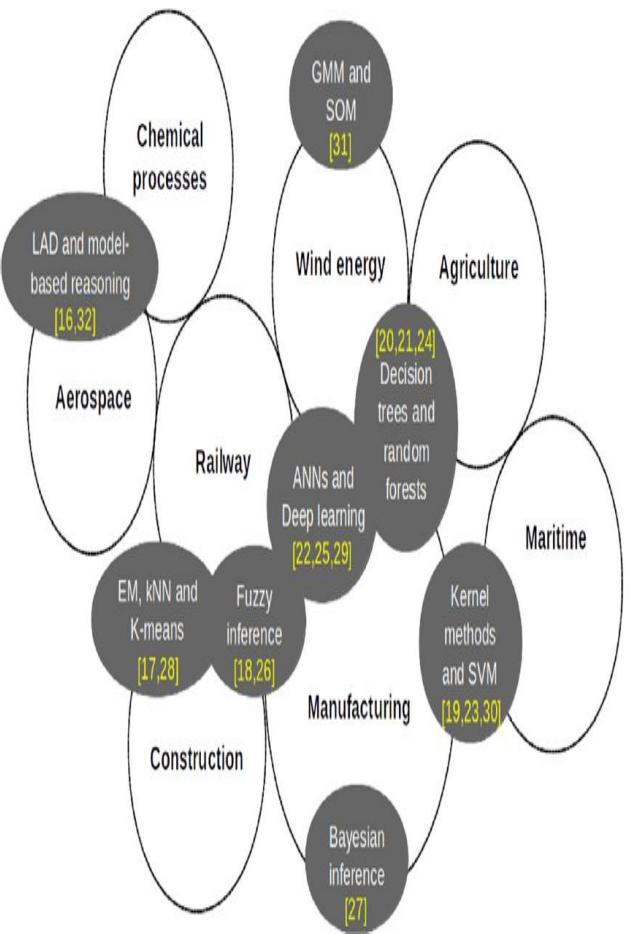


Fig. 4. Solutions and industrial sectors addressed for descriptive prognosis.

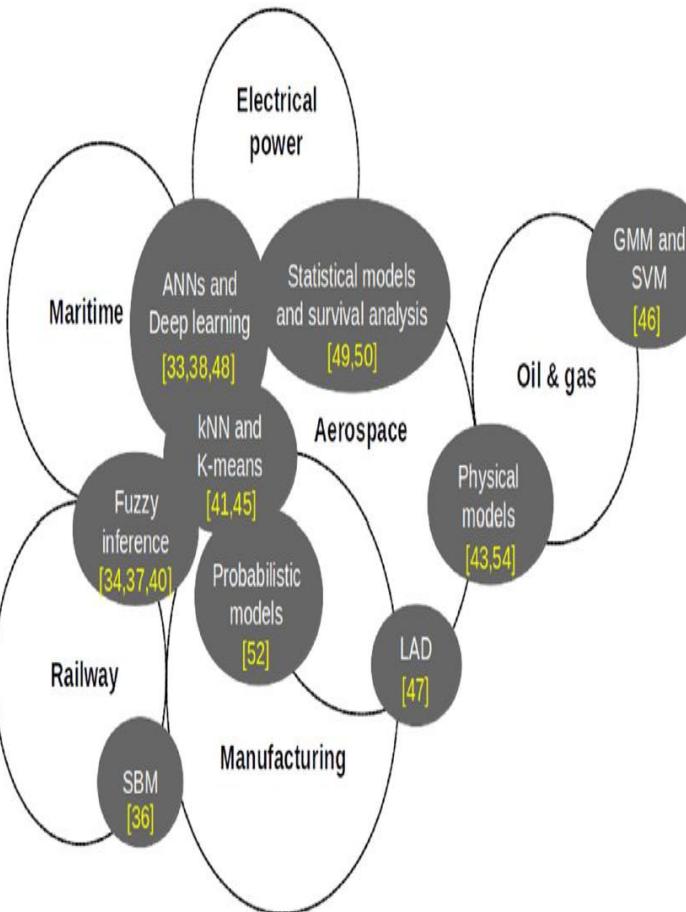


Fig. 5. Solutions and industrial sectors addressed for predictive prognostics.



Full Length Article

Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0



Alberto Diez-Oliván ^a, Javier Del Ser ^{a,b,c,*}, Diego Galar ^{a,d}, Basilio Sierra ^e

^a TECNALIA, Donostia-San Sebastián 20009, Spain

^b Department of Communications Engineering, University of the Basque Country (UPV/EHU), Bilbao 48013, Spain

^c Basque Center for Applied Mathematics (BCAM), Bilbao, Biscay 48009, Spain

^d Department of Civil, Environmental and Natural Resources Engineering, Operation, Maintenance and Acoustics, Luleå University of Technology, Luleå, Sweden

^e Department of Computer Sciences and Artificial Intelligence, University of the Basque Country (UPV/EHU), Donostia-San Sebastián 20018, Spain

ARTICLE INFO

Keywords:
Data-driven prognosis
Data fusion
Machine learning
Industry 4.0

ABSTRACT

The so-called “smartization” of manufacturing industries has been conceived as the fourth industrial revolution or Industry 4.0, a paradigm shift propelled by the upsurge and progressive maturity of new Information and Communication Technologies (ICT) applied to industrial processes and products. From a data science perspective, this paradigm shift allows extracting relevant knowledge from monitored assets through the adoption of intelligent monitoring and data fusion strategies, as well as by the application of machine learning and optimization methods. One of the main goals of data science in this context is to effectively predict abnormal behaviors in industrial machinery, tools and processes so as to anticipate critical events and damage, eventually causing important economical losses and safety issues. In this context, data-driven prognosis is gradually gaining attention in different industrial sectors. This paper provides a comprehensive survey of the recent developments in data fusion and machine learning for industrial prognosis, placing an emphasis on the identification of research trends, niches of opportunity and unexplored challenges. To this end, a principled categorization of the utilized feature extraction techniques and machine learning methods will be provided on the basis of its intended purpose: analyze what caused the failure (descriptive), determine when the monitored asset will fail (predictive) or decide what to do so as to minimize its impact on the industry at hand (prescriptive). This threefold analysis, along with a discussion on its hardware and software implications, intends to serve as a stepping stone for future researchers and practitioners to join the community investigating on this vibrant field.

Machine Learning Applications

Information Fusion 53 (2020) 222–239



Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus



Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook



Marcus A.G. Santos^a, Roberto Munoz^{b,c}, Rodrigo Olivares^d, Pedro P. Rebouças Filho^e, Javier Del Ser^f, Victor Hugo C. de Albuquerque^{a,*}

^a University of Fortaleza, Fortaleza, CE, Brazil

^b School of Informatics Engineering, Universidad de Valparaíso, Valparaíso, Chile

^c Centro de Investigación y Desarrollo en Ingeniería en Salud, Universidad de Valparaíso, Valparaíso, Chile

^d Pontificia Universidad Católica de Valparaíso, Valparaíso, 2362807, Chile

^e Federal Institute of Education, Science and Technology of Ceará, Fortaleza, CE, Brazil

^f TECNALIA, Donostia-San Sebastián, Spain. Department of Communications Engineering, University of the Basque Country, Bilbao, Spain. Basque Center for Applied Mathematics, Bilbao, Bizkaia, Spain

ARTICLE INFO

Keywords:

Internet of health things
Heart
Bio sensors
Online monitoring
Reference model

ABSTRACT

The Internet of Health Things promotes personalized and higher standards of care. Its application is diverse and attracts the attention of a substantial section of the scientific community. This approach has also been applied by people looking to enhance quality of life by using this technology. In this paper, we perform a survey that aims to present and analyze the advances of the latest studies based on medical care and assisted environment. We focus on articles for online monitoring, detection, and support of the diagnosis of cardiovascular diseases. Our research covers published manuscripts in scientific journals and recognized conferences since the year 2015. Also, we present a reference model based on the evaluation of the resources used from the selected studies. Finally, our proposal aims to help future enthusiasts to discover and enumerate the required factors for the development of a prototype for online heart monitoring purposes.

Machine Learning Applications

Information Fusion 53 (2020) 222–239

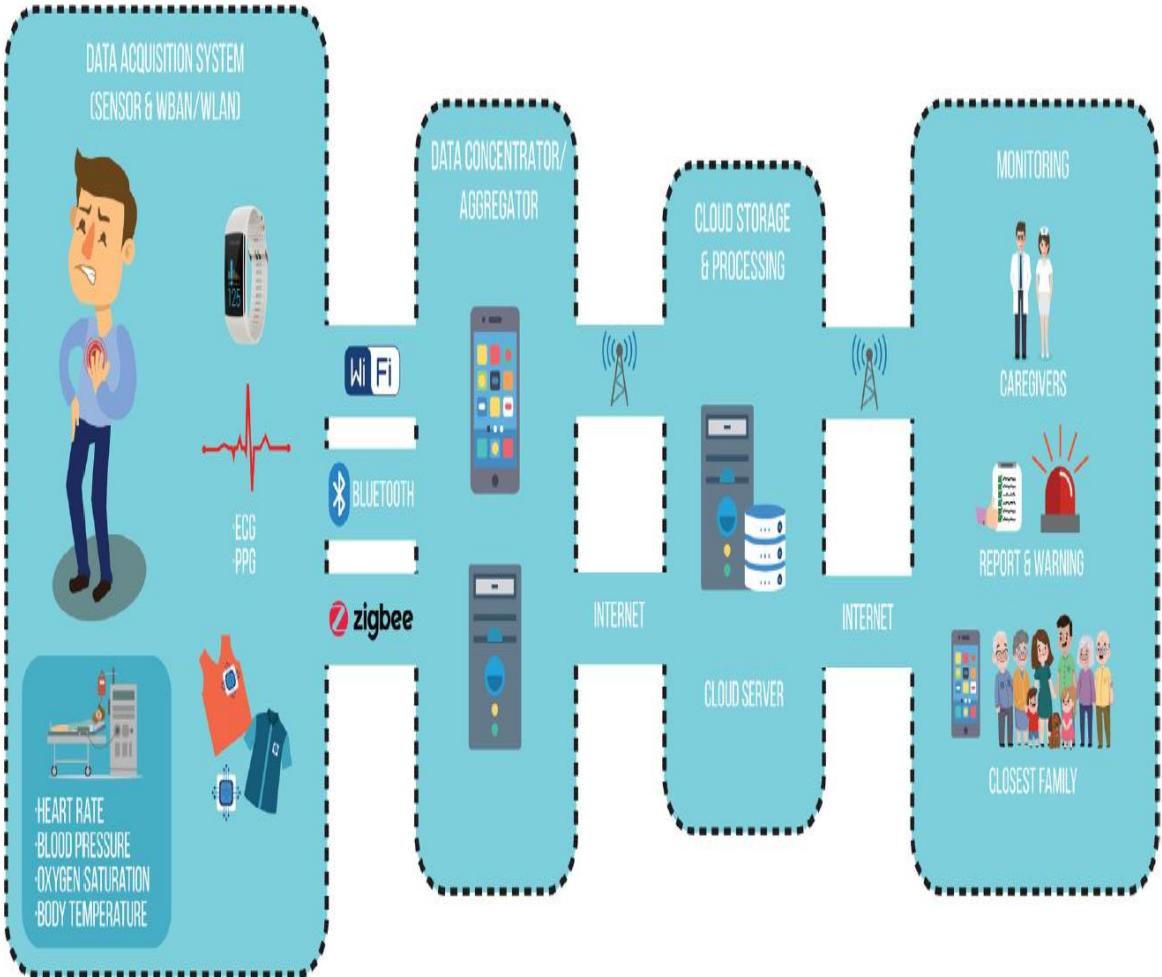


Fig. 1. IoT of the Heart.



Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus



Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook



Marcus A.G. Santos^a, Roberto Munoz^{b,c}, Rodrigo Olivares^d, Pedro P. Rebouças Filho^e, Javier Del Ser^f, Victor Hugo C. de Albuquerque^{a,*}

^aUniversity of Fortaleza, Fortaleza, CE, Brazil

^bSchool of Informatics Engineering, Universidad de Valparaíso, Valparaíso, Chile

^cCentro de Investigación y Desarrollo en Ingeniería en Salud, Universidad de Valparaíso, Valparaíso, Chile

^dPontificia Universidad Católica de Valparaíso, Valparaíso, 2362807, Chile

^eFederal Institute of Education, Science and Technology of Ceará, Fortaleza, CE, Brazil

^fTECNALIA, Donostia-San Sebastián, Spain. Department of Communications Engineering, University of the Basque Country, Bilbao, Spain. Basque Center for Applied Mathematics, Bilbao, Bizkaia, Spain

ARTICLE INFO

Keywords:

Internet of health things
Heart
Bio sensors
Online monitoring
Reference model

ABSTRACT

The Internet of Health Things promotes personalized and higher standards of care. Its application is diverse and attracts the attention of a substantial section of the scientific community. This approach has also been applied by people looking to enhance quality of life by using this technology. In this paper, we perform a survey that aims to present and analyze the advances of the latest studies based on medical care and assisted environment. We focus on articles for online monitoring, detection, and support of the diagnosis of cardiovascular diseases. Our research covers published manuscripts in scientific journals and recognized conferences since the year 2015. Also, we present a reference model based on the evaluation of the resources used from the selected studies. Finally, our proposal aims to help future enthusiasts to discover and enumerate the required factors for the development of a prototype for online heart monitoring purposes.

Machine Learning Applications

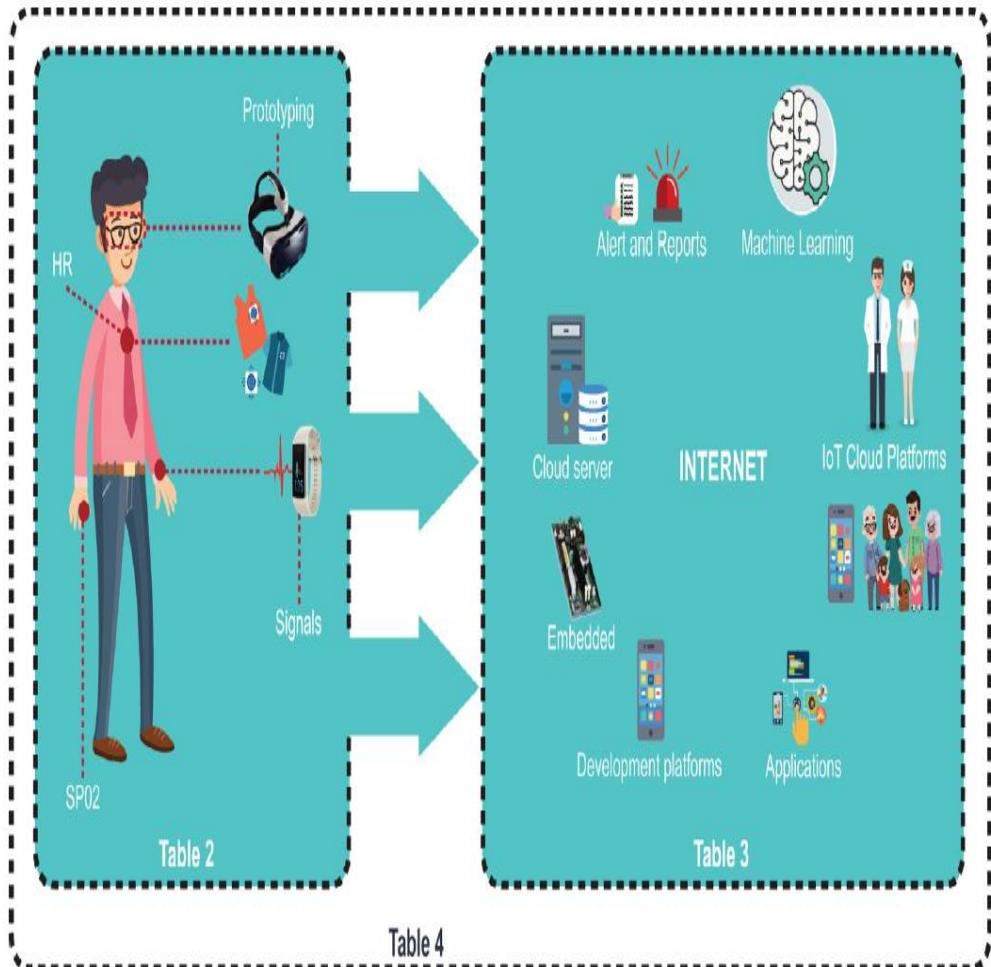


Fig. 2. Development framework.

Information Fusion 53 (2020) 222–239



Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus



Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook

Marcus A.G. Santos^a, Roberto Munoz^{b,c}, Rodrigo Olivares^d, Pedro P. Rebouças Filho^e, Javier Del Ser^f, Victor Hugo C. de Albuquerque^{a,*}

^a University of Fortaleza, Fortaleza, CE, Brazil

^b School of Informatics Engineering, Universidad de Valparaíso, Valparaíso, Chile

^c Centro de Investigación y Desarrollo en Ingeniería en Salud, Universidad de Valparaíso, Valparaíso, Chile

^d Pontificia Universidad Católica de Valparaíso, Valparaíso, 2362807, Chile

^e Federal Institute of Education, Science and Technology of Ceará, Fortaleza, CE, Brazil

^f TECNALIA, Donostia-San Sebastián, Spain. Department of Communications Engineering, University of the Basque Country, Bilbao, Spain. Basque Center for Applied Mathematics, Bilbao, Biscay, Spain

ARTICLE INFO

Keywords:

Internet of health things
Heart
Bio sensors
Online monitoring
Reference model

ABSTRACT

The Internet of Health Things promotes personalized and higher standards of care. Its application is diverse and attracts the attention of a substantial section of the scientific community. This approach has also been applied by people looking to enhance quality of life by using this technology. In this paper, we perform a survey that aims to present and analyze the advances of the latest studies based on medical care and assisted environment. We focus on articles for online monitoring, detection, and support of the diagnosis of cardiovascular diseases. Our research covers published manuscripts in scientific journals and recognized conferences since the year 2015. Also, we present a reference model based on the evaluation of the resources used from the selected studies. Finally, our proposal aims to help future enthusiasts to discover and enumerate the required factors for the development of a prototype for online heart monitoring purposes.

Introduction of Deep Learning

Unsupervised Learning Algorithms Or Generative Algorithms

- Autoencoders
 - Stack Autoencoders
 - Denoising Autoencoders
 - Sparse Autoencoders
 - Deep Autoencoders
- Boltzmann Machine (BM)
 - Restricted Boltzmann Machine (RBM)
 - Deep Belief Network (DBN)
- Variational Autoencoders (VEs)
- Generative adversarial networks(GAN)

Could be

- Recurrent Neural Network (RNN)
- Long Short-Term Memory Network (LSTM)

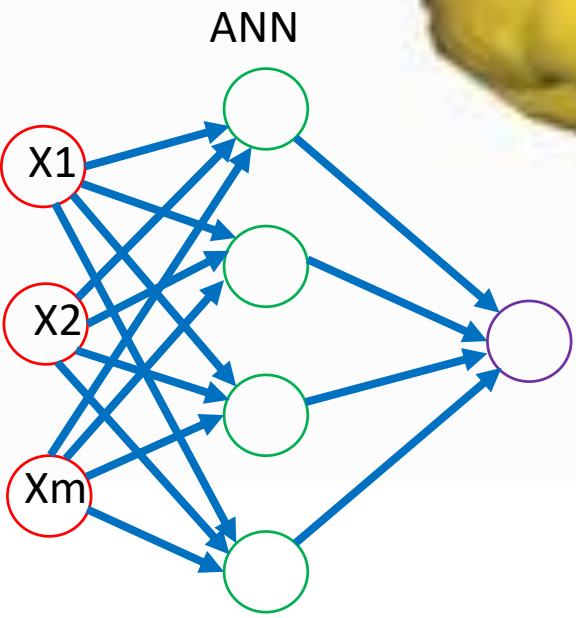
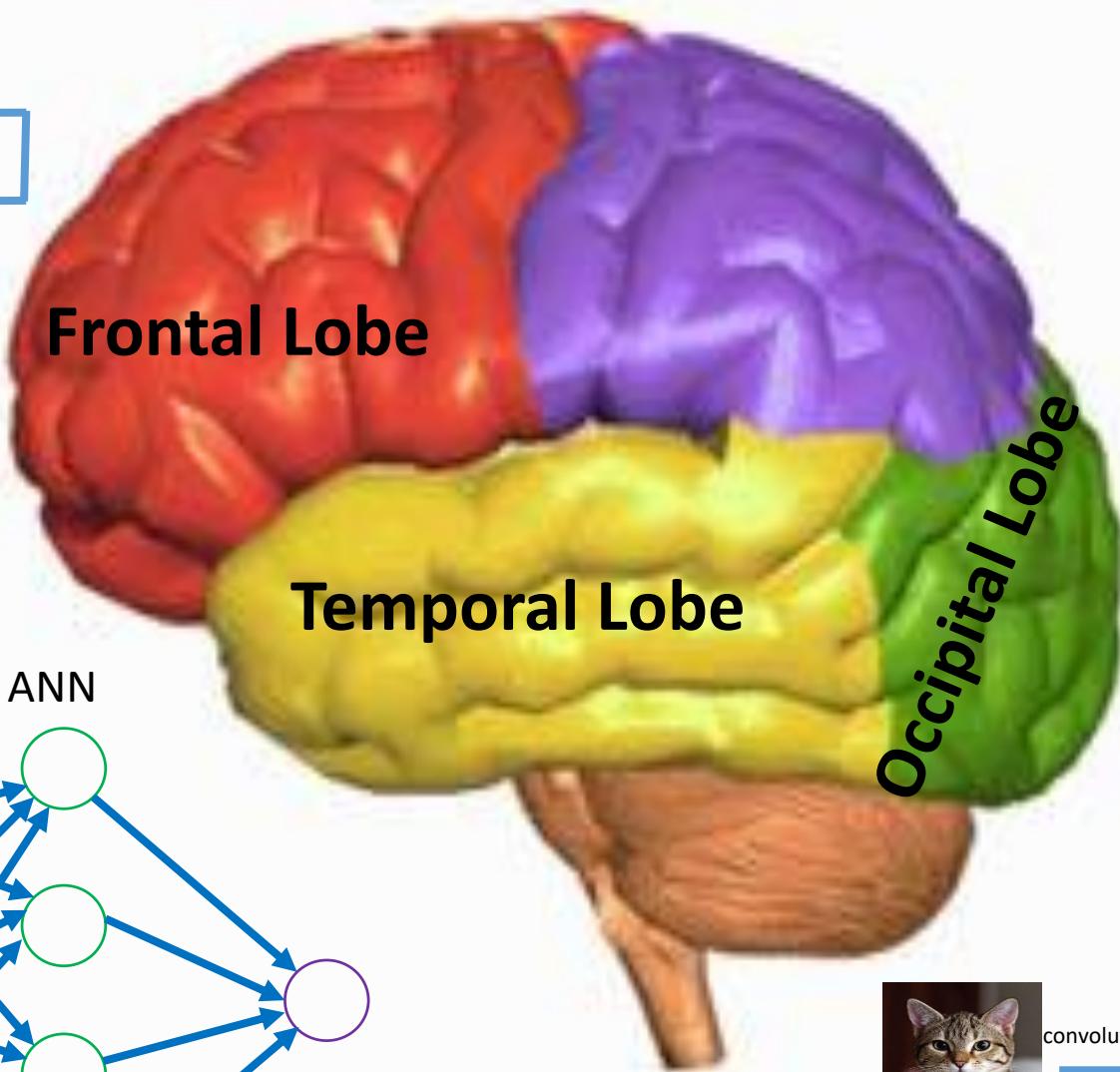
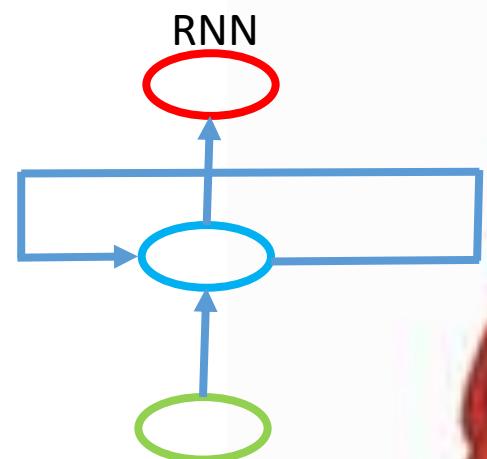
Supervised Learning Algorithms

- Artificial Neural Network (ANN)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Long Short-Term Memory Network (LSTM)
- Gated Recurrent Unit (GRU)

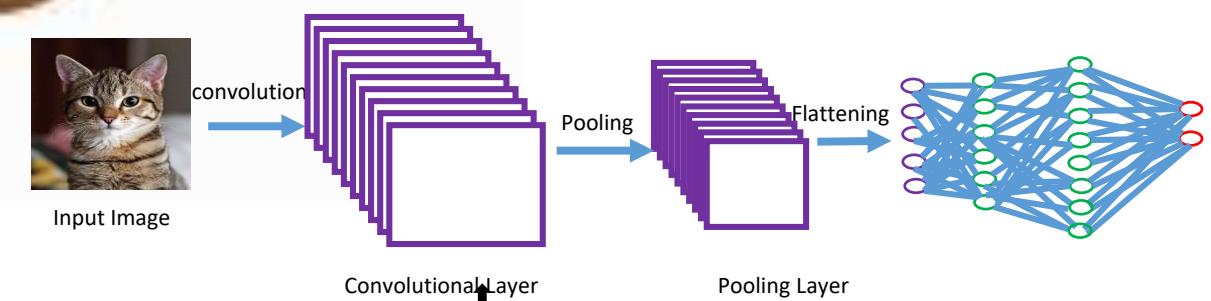
Could be

- Autoencoders and their variants

Brain Analogy with Deep Neural Networks



Input Image



Convolutional Layer

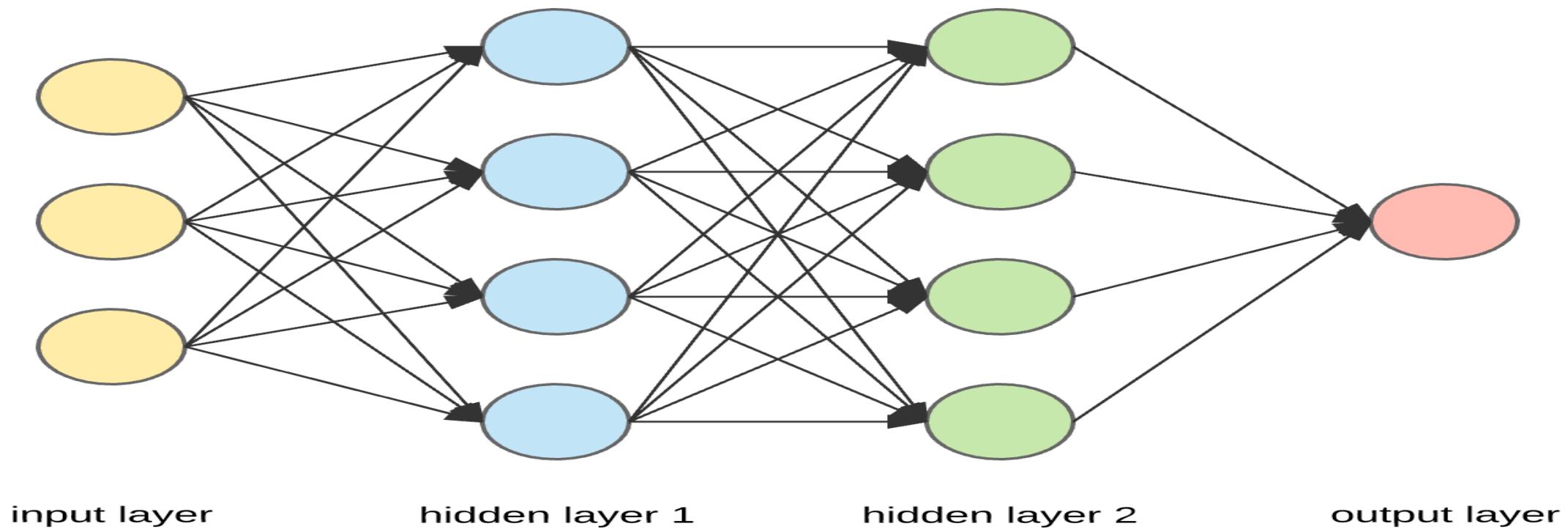
Pooling Layer

Rectifier Function
 $\varphi(x) = \max(x, 0)$

Artificial Neural Network (ANN)

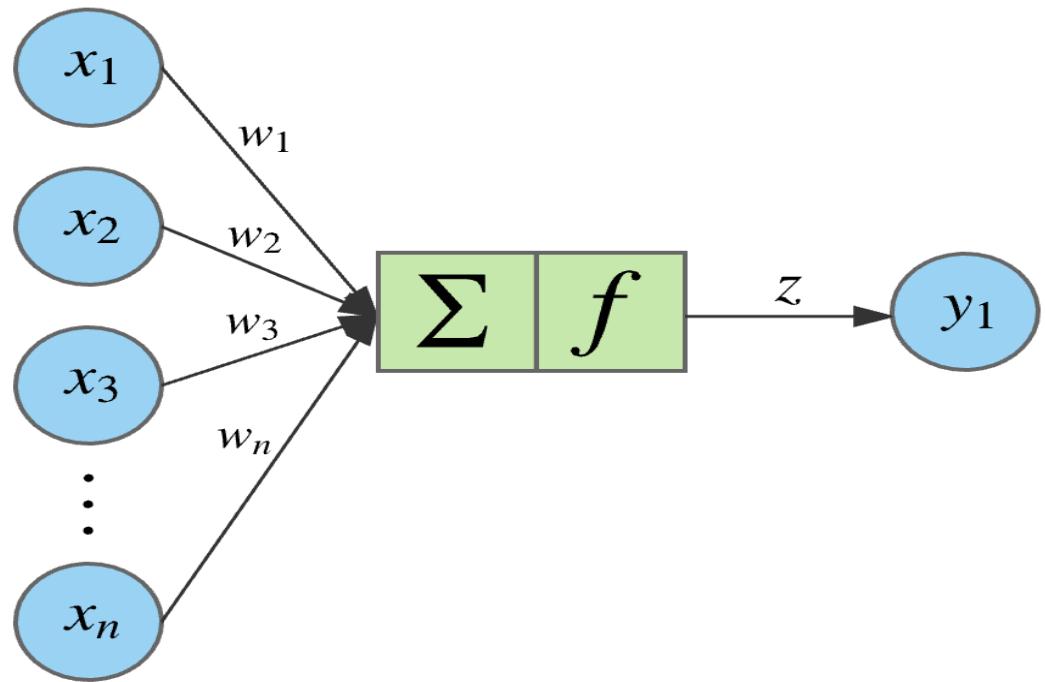
How NN works?

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets that look like the figure below. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers.



How NN works?

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets that look like the figure below. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers.



$$z = f(x \cdot w) = f \left(\sum_{i=1}^n x_i w_i \right)$$

$$x \in d_{1 \times n}, w \in d_{n \times 1}, z \in d_{1 \times 1}$$

$$z = f(b + x \cdot w) = f \left(b + \sum_{i=1}^n x_i w_i \right)$$

$$x \in d_{1 \times n}, w \in d_{n \times 1}, b \in d_{1 \times 1}, z \in d_{1 \times 1}$$

How NN works?

Training procedure works as follows:

Randomly initialize the weights for all the nodes

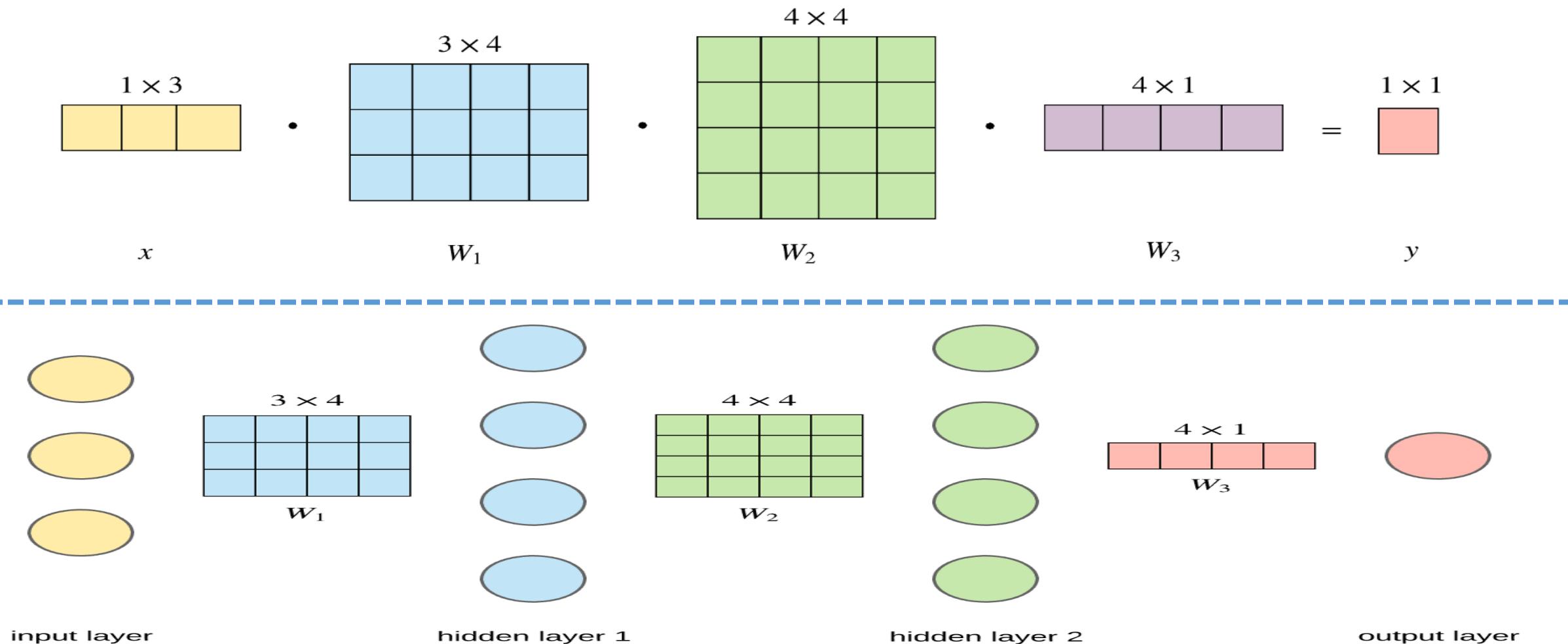
For every training example, perform a forward pass using the current weights, and calculate the output of each node going from left to right. The final output is the value of the last node.

Compare the final output with the actual target in the training data, and measure the error using a loss function.

Perform a backwards pass from right to left and propagate the error to every individual node using backpropagation. Calculate each weight's contribution to the error, and adjust the weights accordingly using gradient descent. Propagate the error gradients back starting from the last layer.

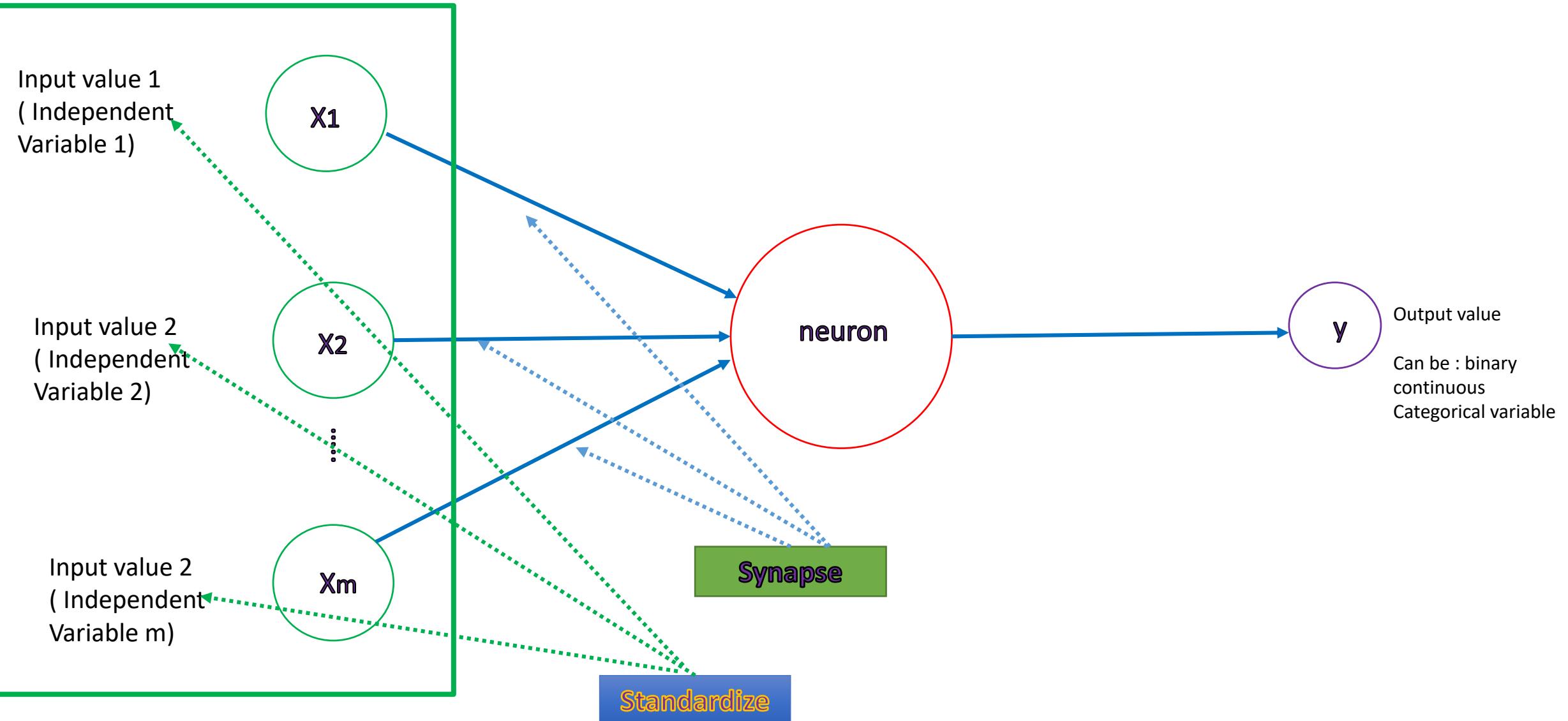
How NN works?

Training procedure works as follows:



The Neuron

All independent variable is called one observation or one example or one sample



The Neuron

All independent variable is called one observation or one example or one sample

Input value 1
(Independent Variable 1)

Input value 2
(Independent Variable 2)

Input value 2
(Independent Variable m)

x_1

x_2

x_m

neuron

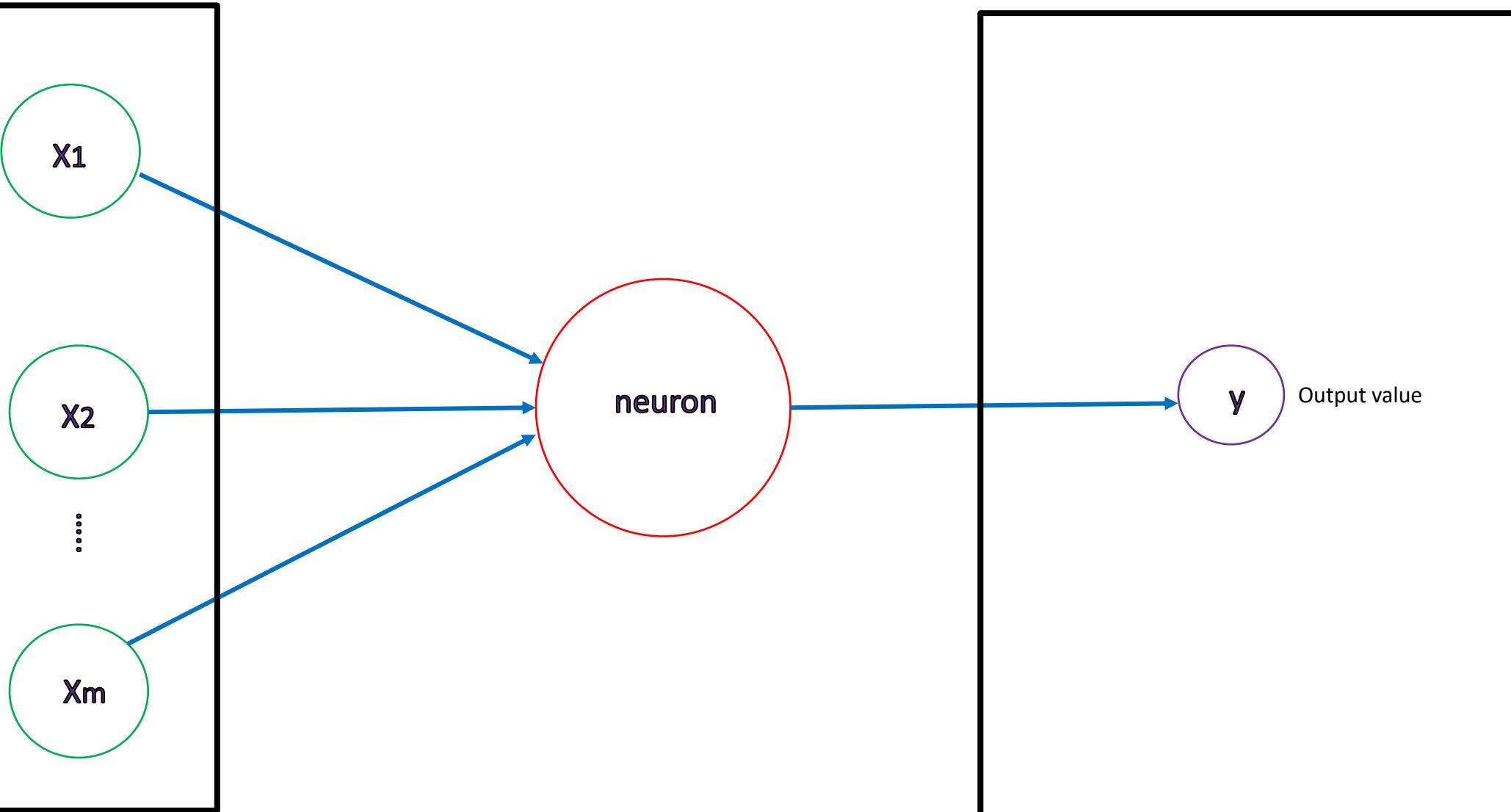
y

Output value

Single Observation

Same observation

Single Observation

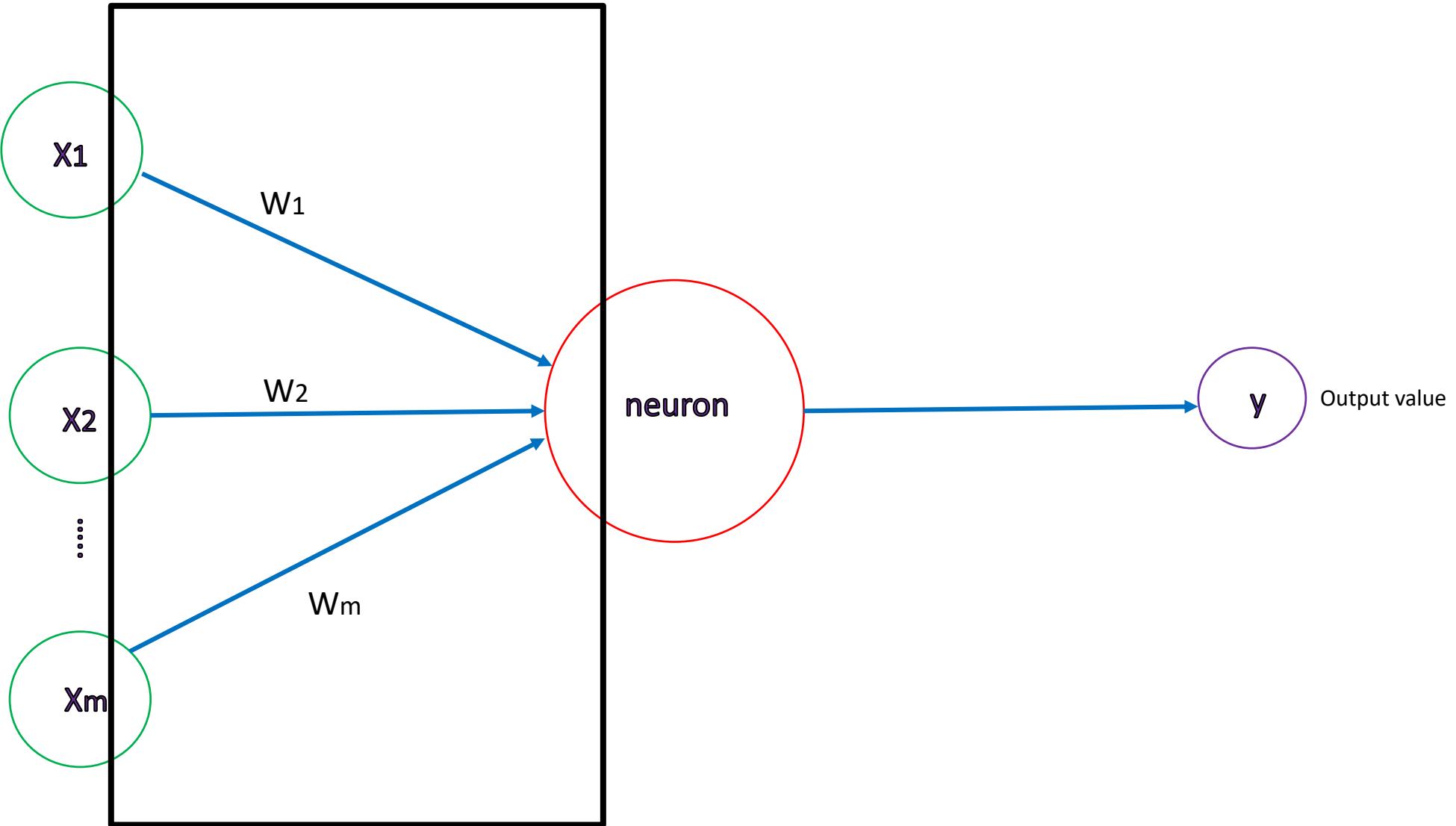


The Neuron

Input value 1
(Independent Variable 1)

Input value 2
(Independent Variable 2)

Input value 2
(Independent Variable m)



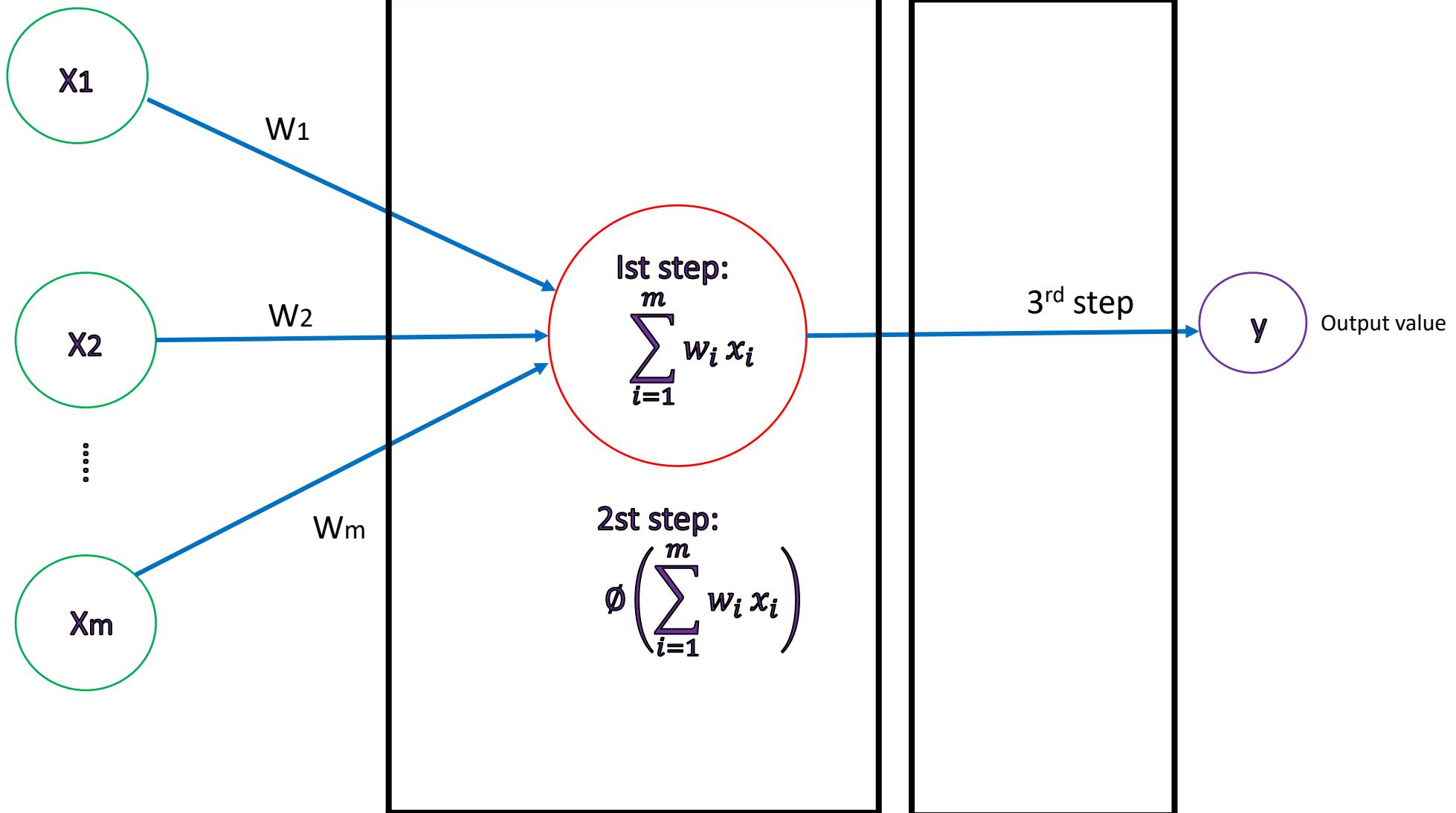
The Neuron

The weighted sum of the input is passed to the activation

Input value 1
(Independent Variable 1)

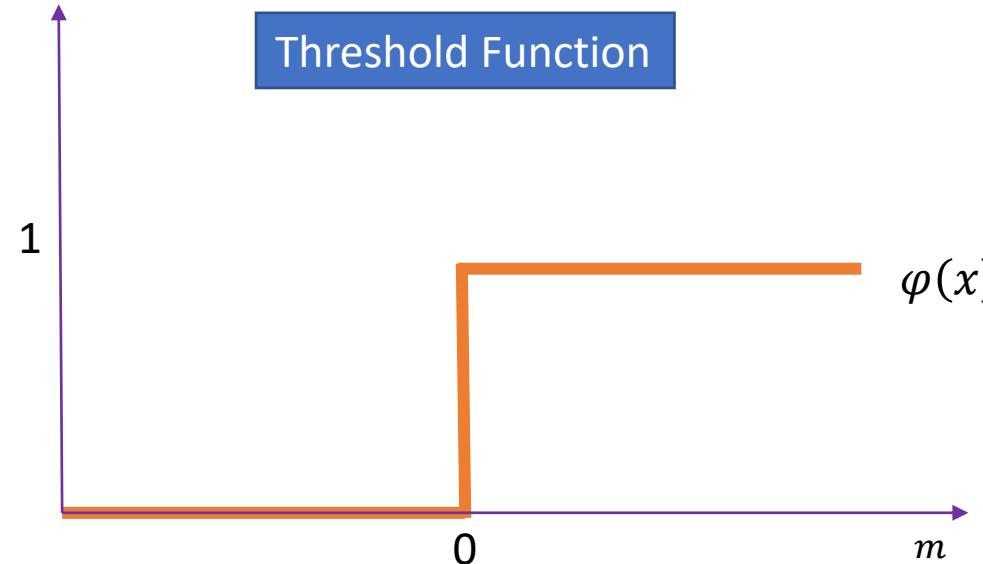
Input value 2
(Independent Variable 2)

Input value 2
(Independent Variable m)



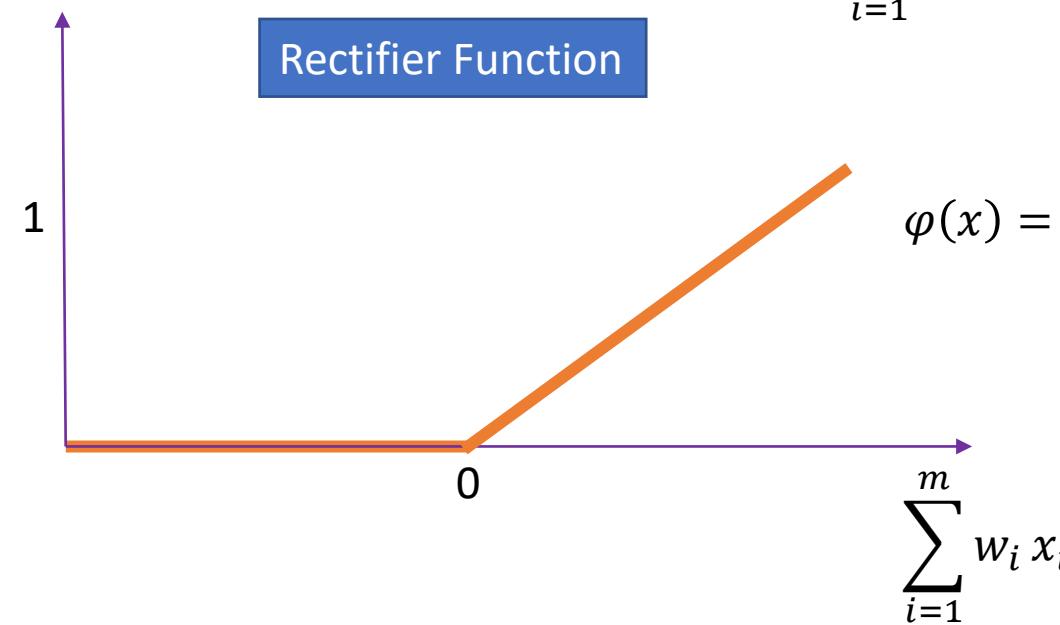
Activation Functions

Threshold Function



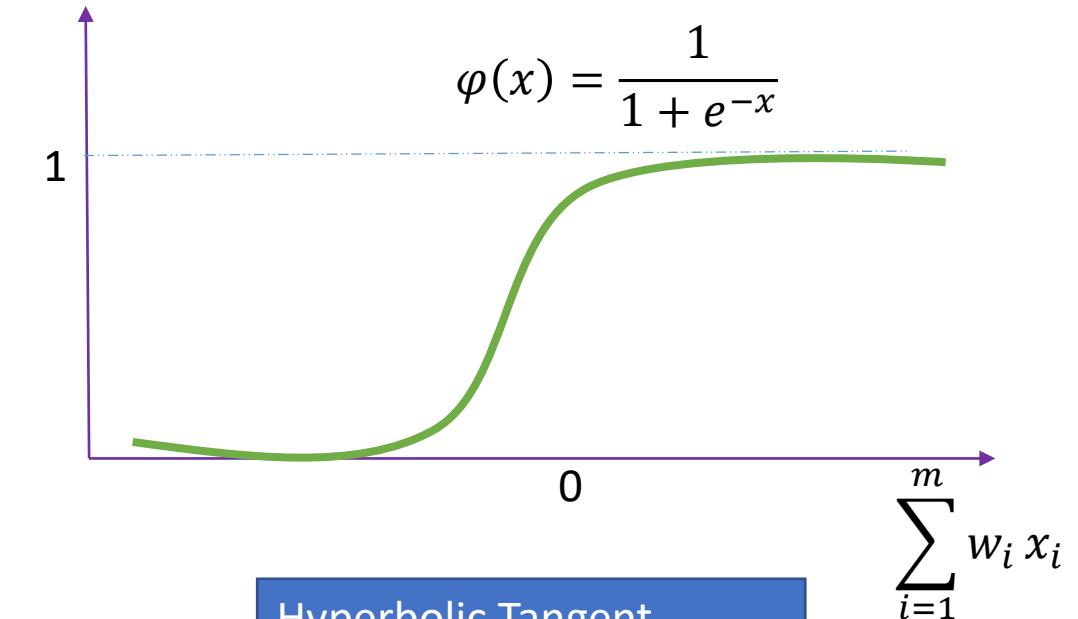
$$\varphi(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Rectifier Function



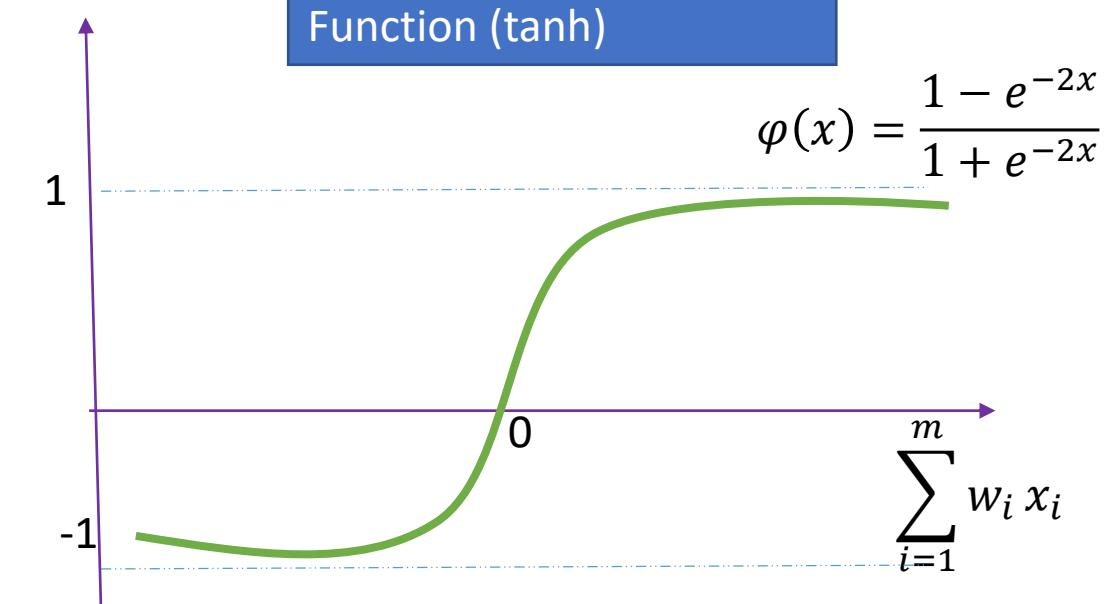
$$\varphi(x) = \max(x, 0)$$

Sigmoid Function



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

Hyperbolic Tangent Function (tanh)



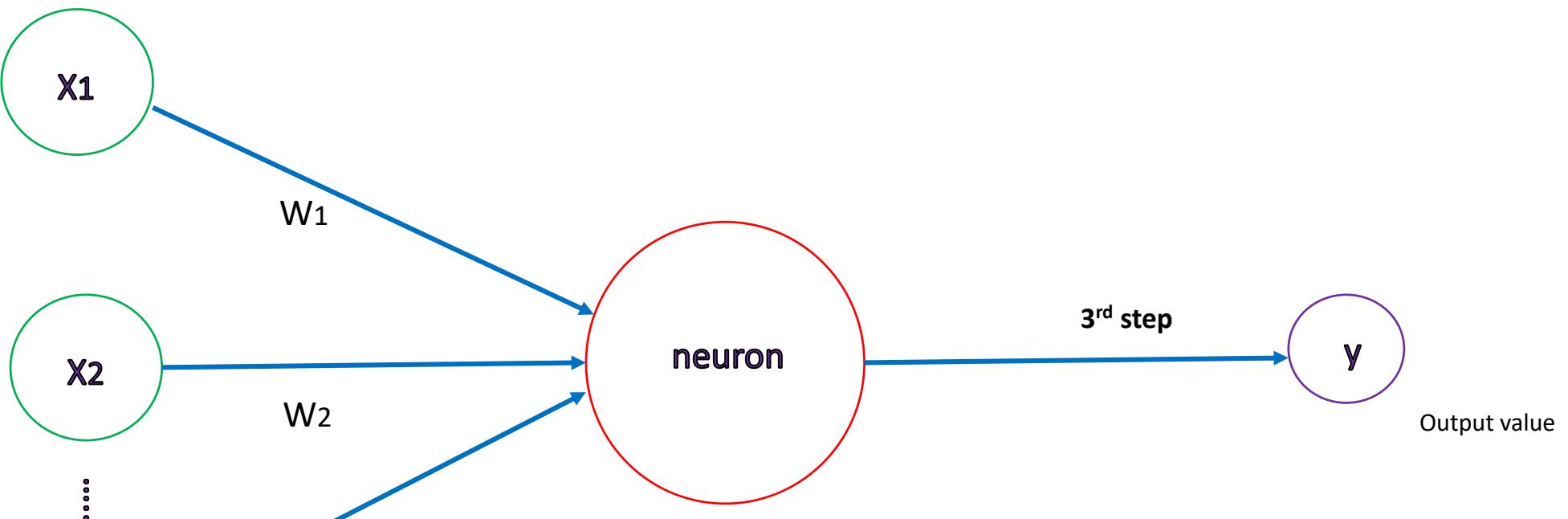
$$\varphi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

Neural Network

Input value 1
(Independent Variable 1)

Input value 2
(Independent Variable 2)

Input value 2
(Independent Variable m)

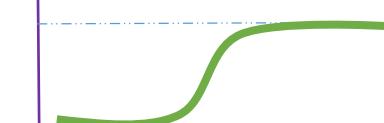


Assuming the dependent variable is binary ($y=0$ or 1)

If threshold activation function: $y = \emptyset\left(\sum_{i=1}^m w_i x_i\right)$



If sigmoid activation function: $y = P(y = 1) = \emptyset\left(\sum_{i=1}^m w_i x_i\right)$



Neural Network

Input value 1

x_1

Input value 2

x_2

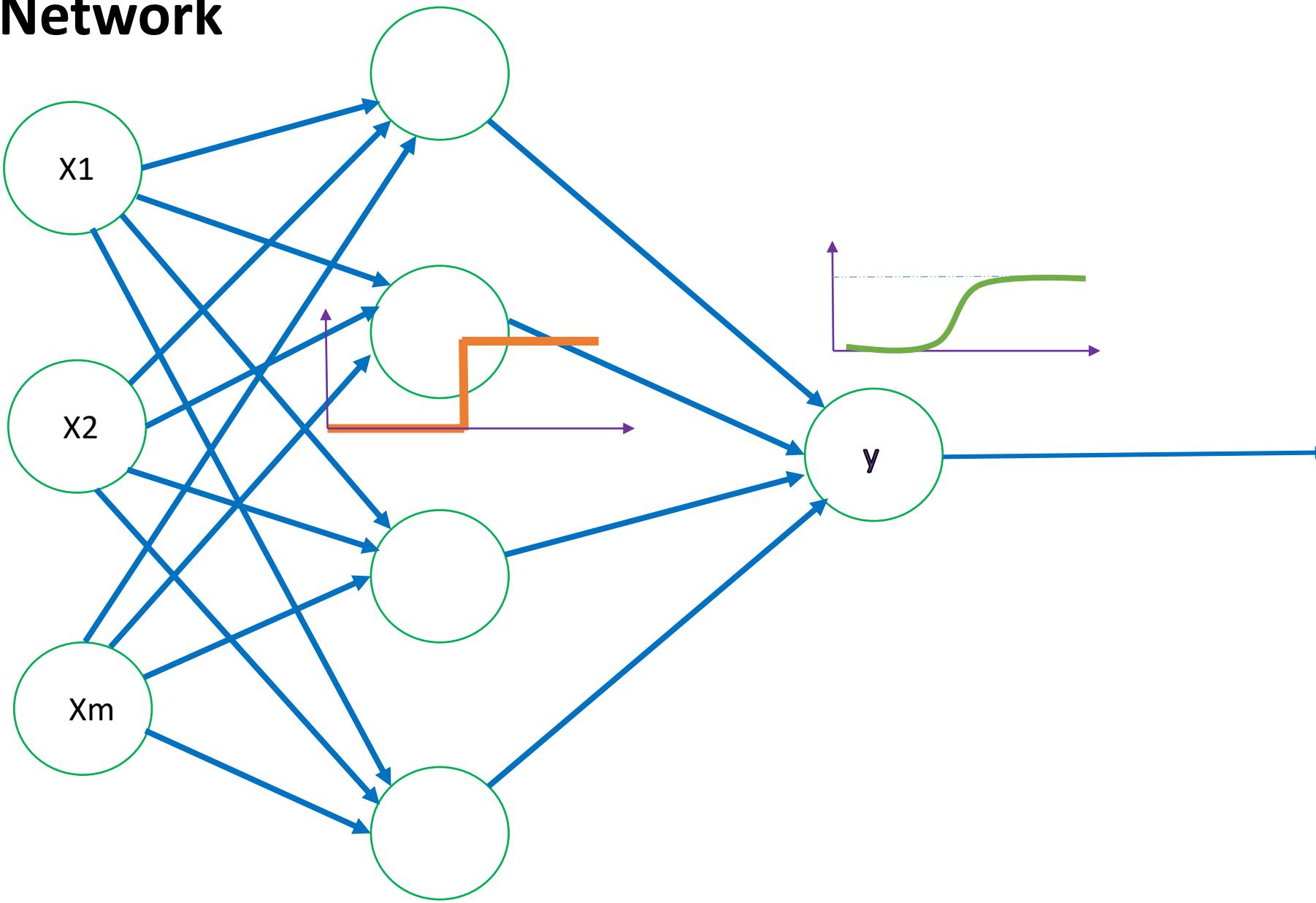
Input value m

x_m

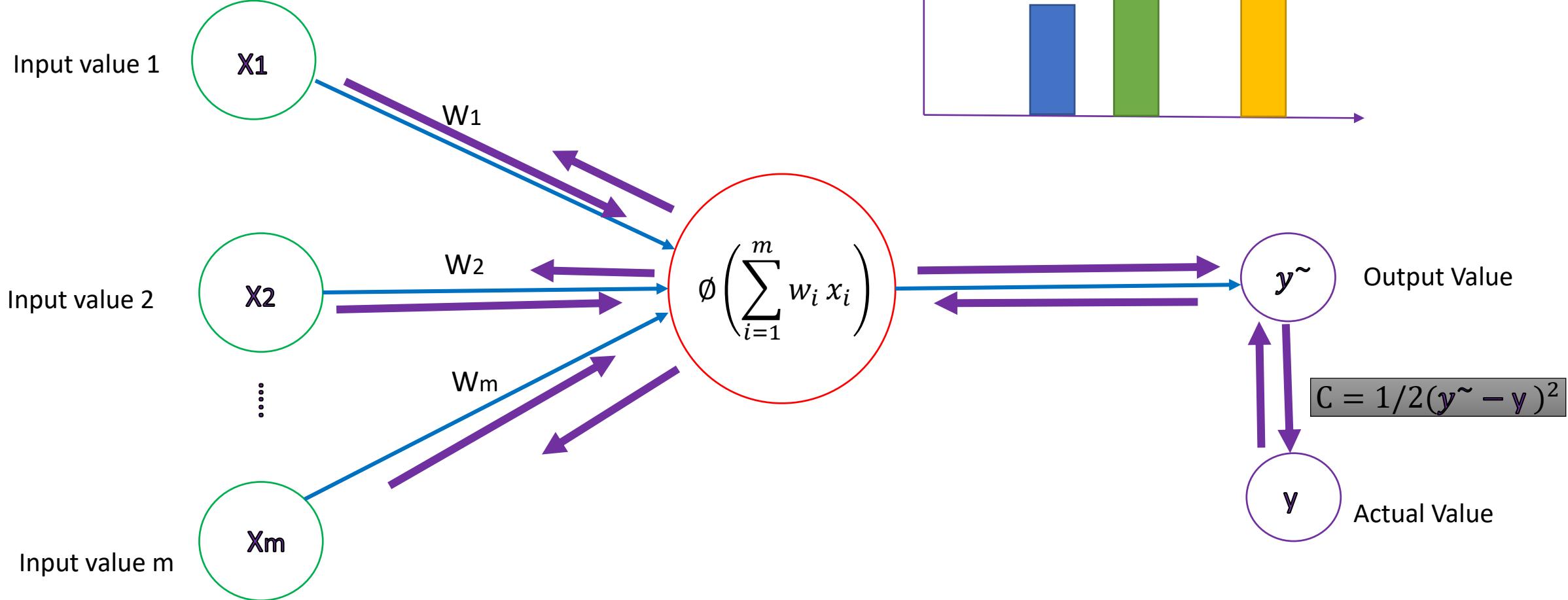
Input layer

Hidden Layer

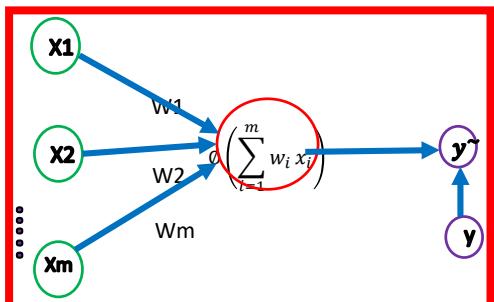
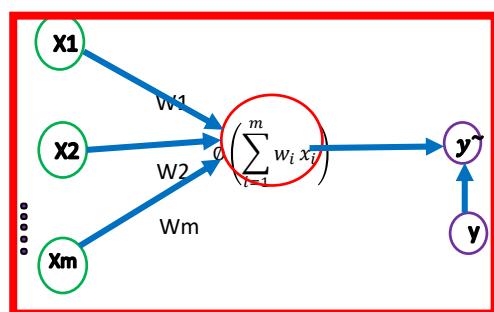
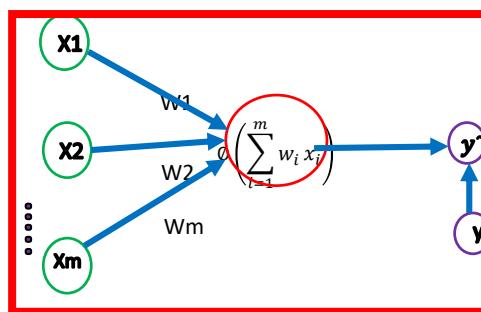
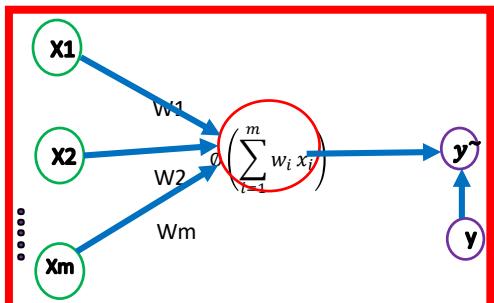
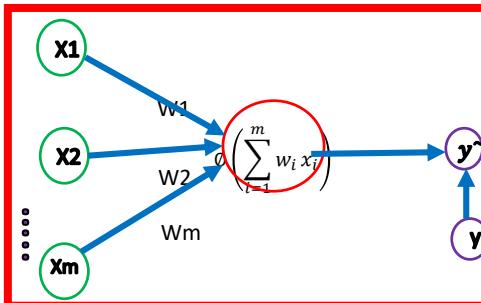
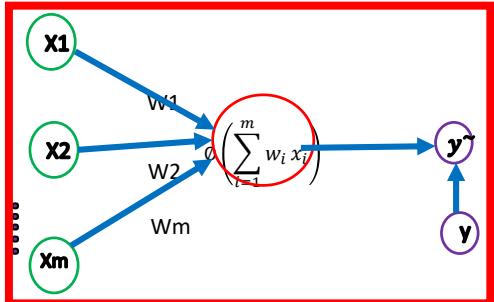
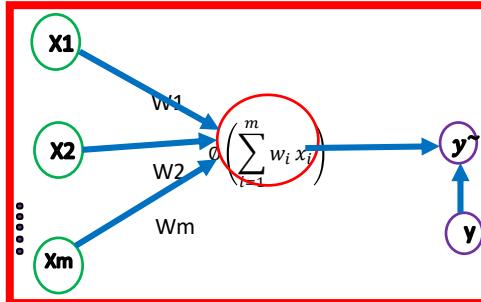
Output Layer



How NN works?



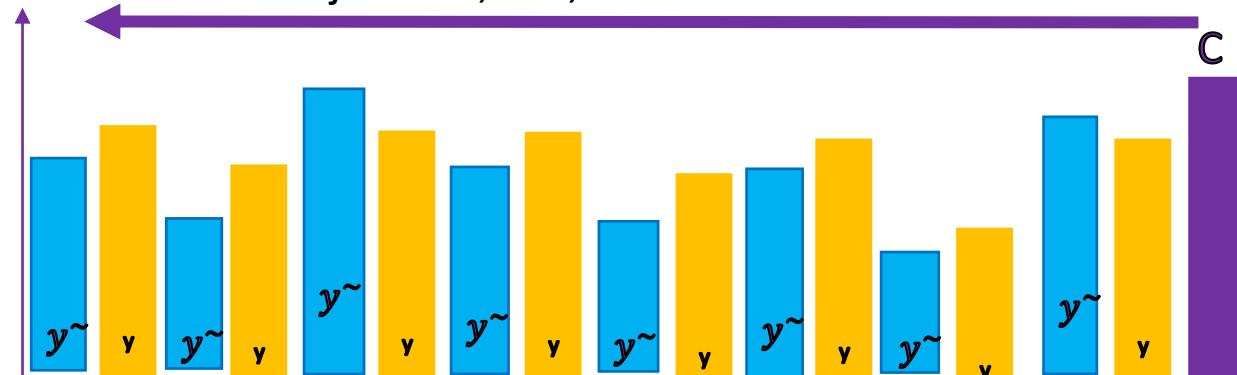
How NN works? An Example:



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

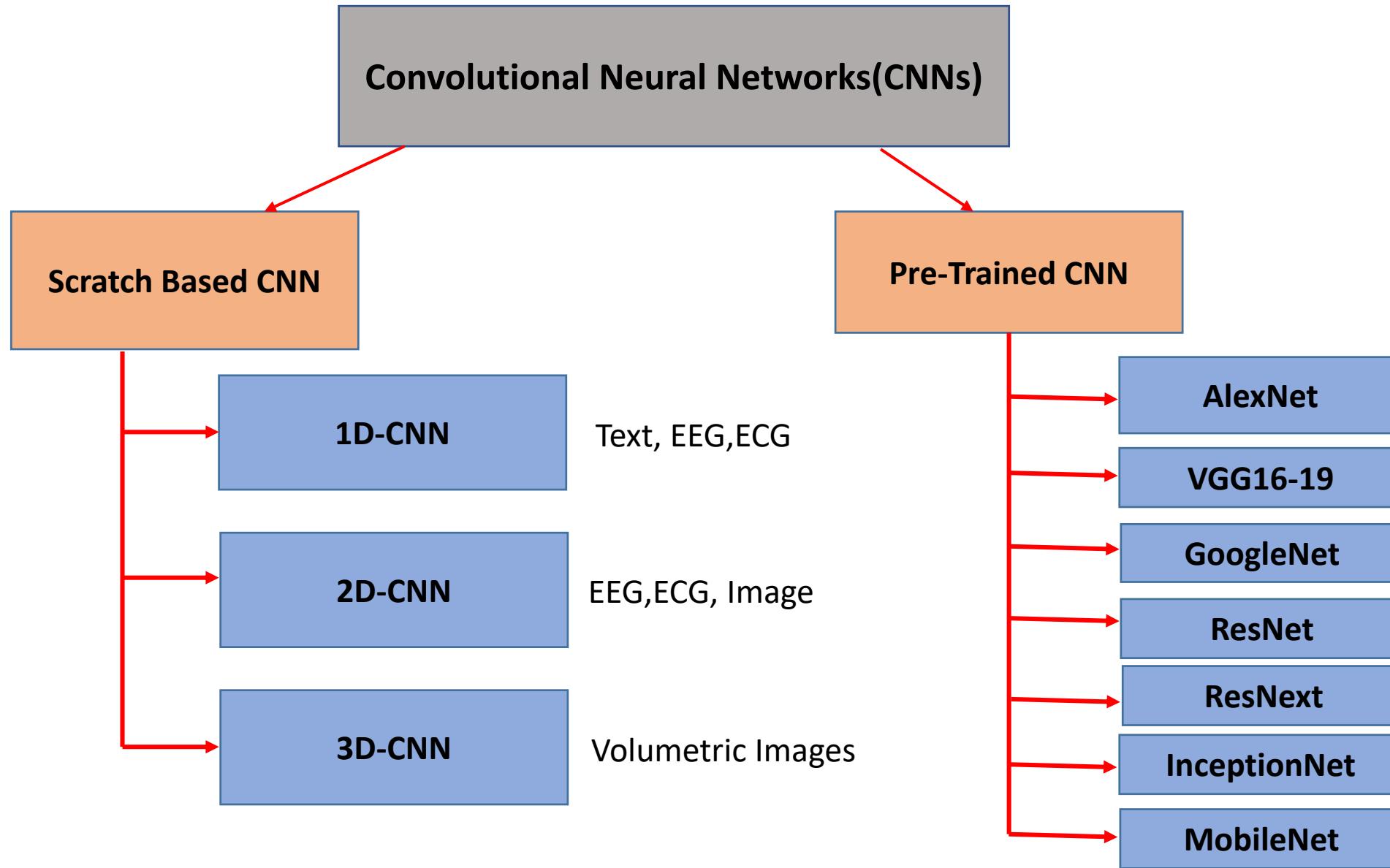
$$C = \sum 1/2(\hat{y} - y)^2$$

Adjust W_1, W_2, W_3



Convolutional Neural Network (CNN)

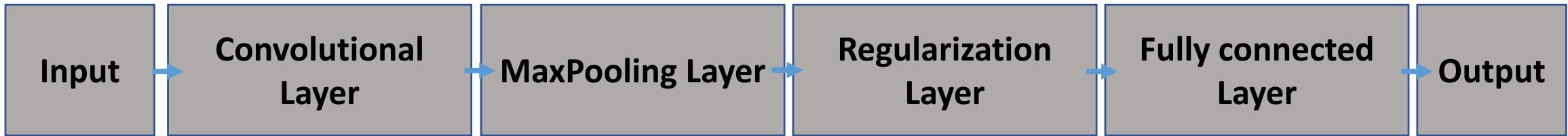
Convolutional Neural Networks



Convolutional Neural Networks

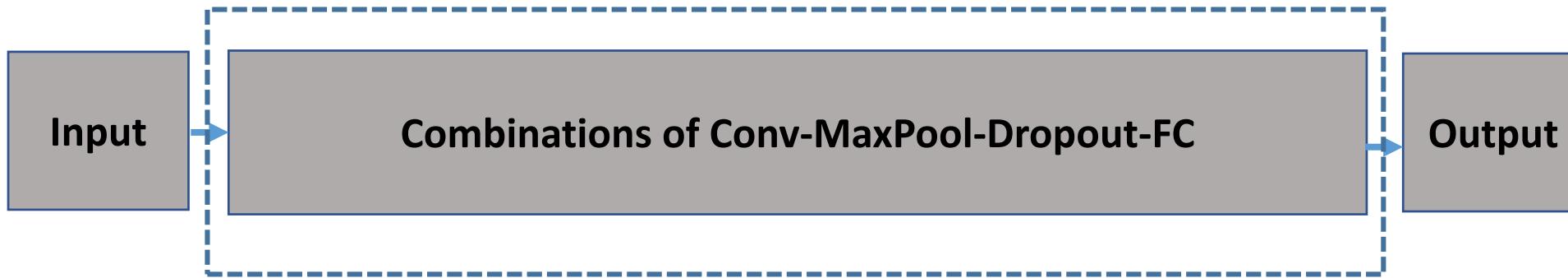
Scratch Based Network

Trained layers from scratch based on your own dataset



Pre-trained Networks

Trained layers on ImageNet dataset



2D-CNN

Step 1- Convolution

Conv Layer:

Accepts a volume of size =**W1×H1×D1**

Requires four hyperparameters:

Number of filters K, their spatial extent F or size of filter or kernel, the stride S, the amount of zero padding P.

Produces a volume of size $W2 \times H2 \times D2$ where:

$$W2 = (W1 - F + 2P) / S + 1$$

$$H2 = (H1 - F + 2P) / S + 1 \text{ (i.e. width and height are computed equally by symmetry)}$$

$$D2 = K \text{ (depth equal to length of number of filters)}$$

Example:

Input_volume=Image or 2D matrix (height width x channels)=**W1×H1×D1** =28x28x3

2D Kernel or 2D filter= $F \times F$ (spatial filter or kernel)=**3x3xk(k is number of filters)**

In this example

k=3

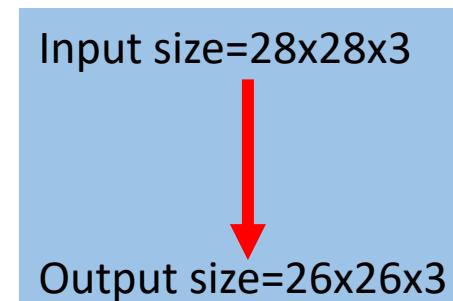
Stride=1

P=0

$$W2 = (W1 - F + 2P) / S + 1 = (28 - 3 + 2(0)) / 1 + 1 = 26$$

$$H2 = (H1 - F + 2P) / S + 1 = (28 - 3 + 2(0)) / 1 + 1 = 26$$

$$D2 = 3$$



Step 1- Convolution

- **One channel and 3 channel convolution**

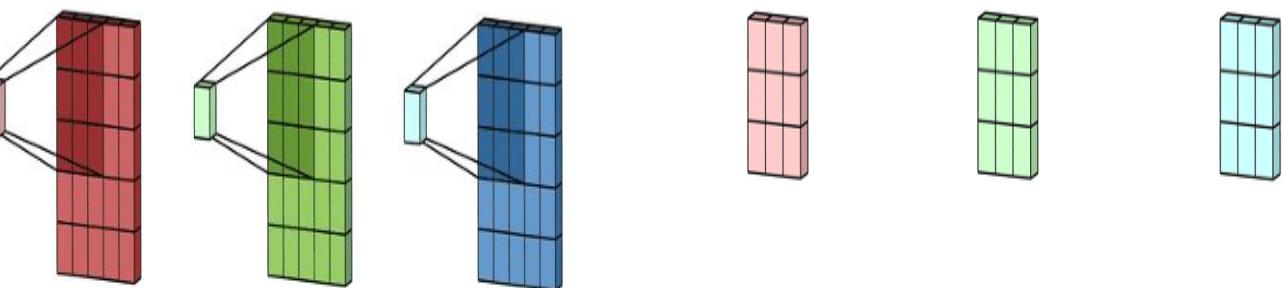
1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Convolutional (1 channel)

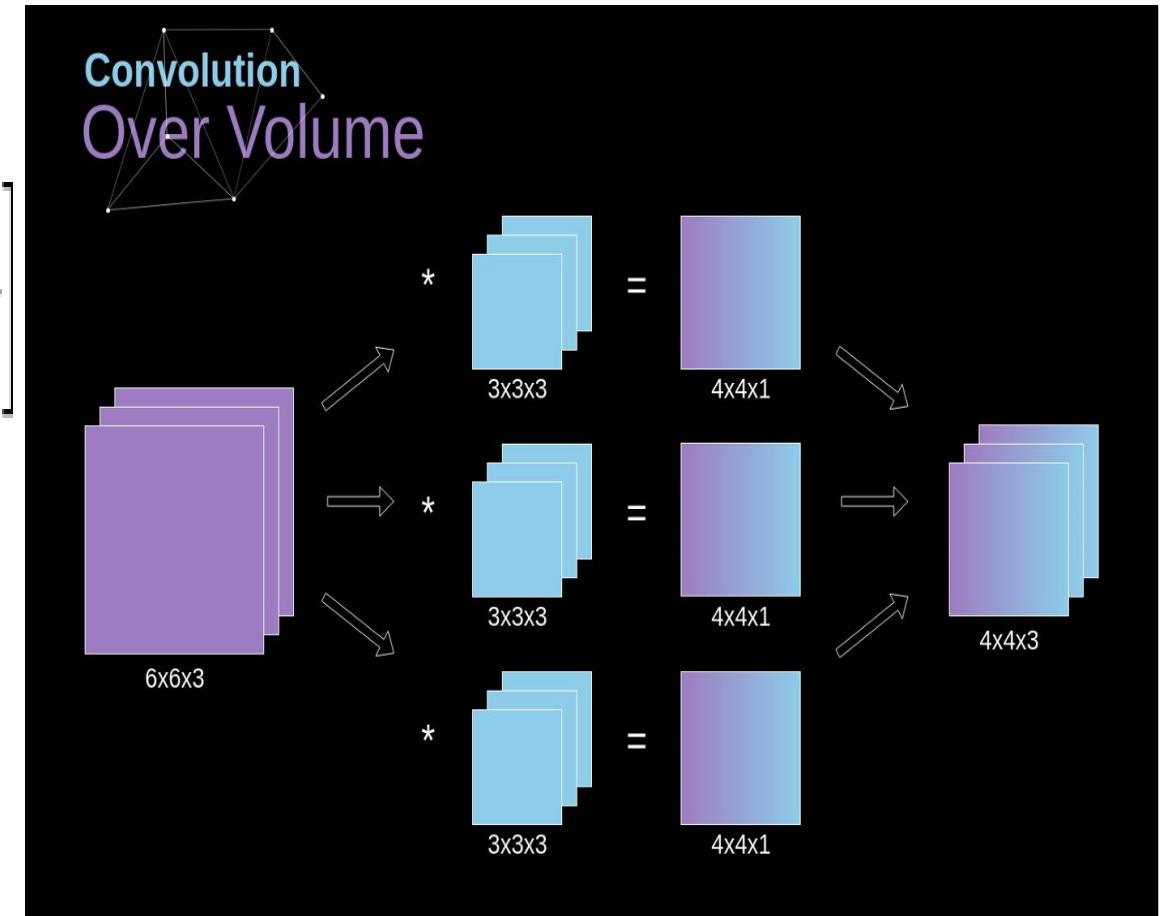


Convolutional (3 channel)

Step 1 - Convolution

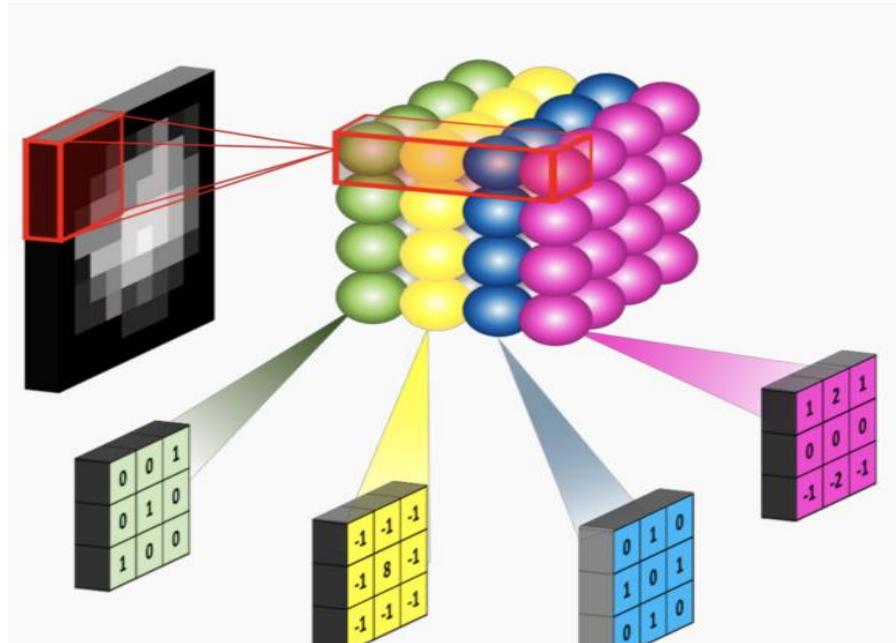
▪ Convolution

$$[n, n, n_c] * [f, f, n_c] = \left[\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor, \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor, n_f \right]$$

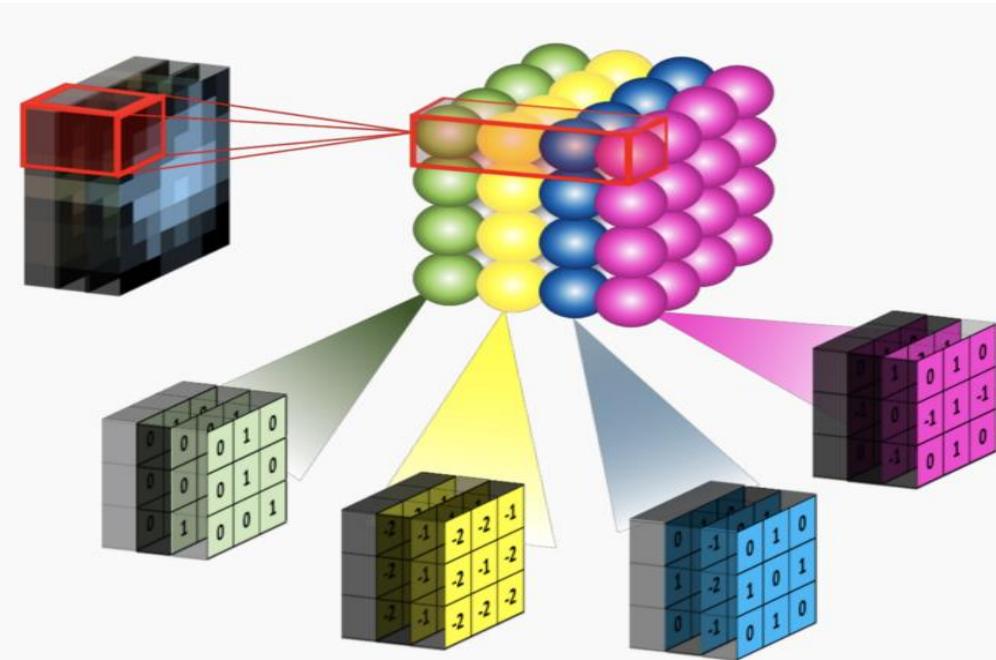


Optimization Algorithms

▪ Convolution



Convolutional layer with four 3x3 filters on a **black and white image** (just one channel)

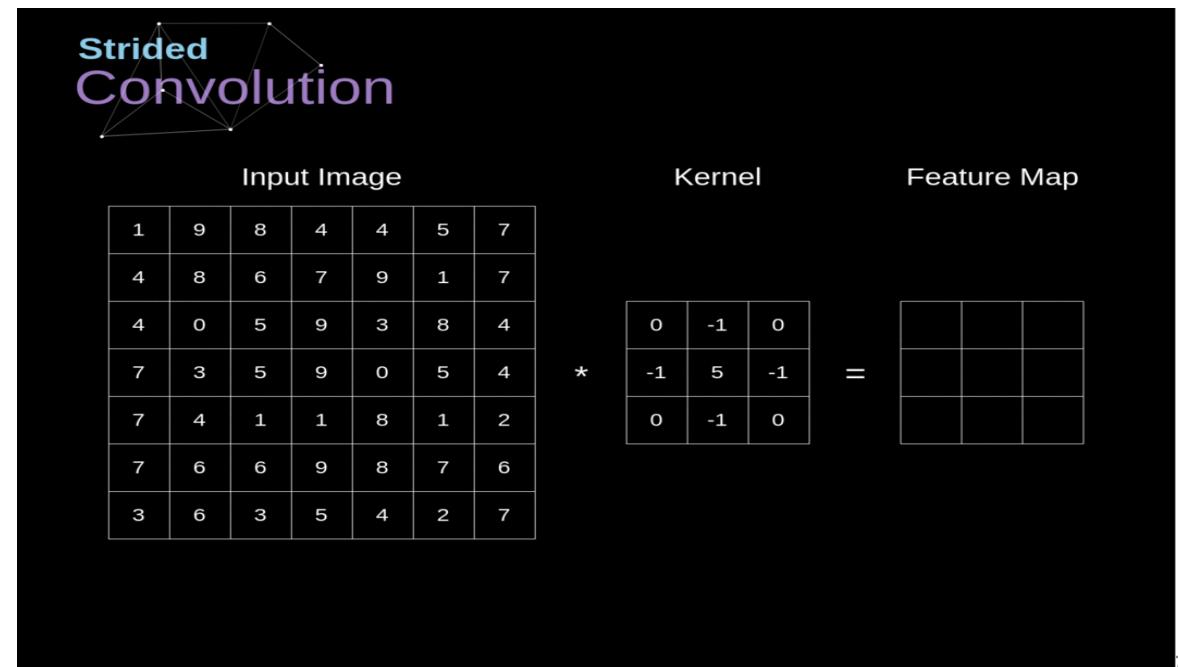


Convolutional layer with four 3x3 filters on an **RGB image**. As you can see, the filters are now cubes, and they are applied on the full depth of the image..

Step 1- Convolution

- **Convolution**
- we are dealing with two types of convolution — **Valid and Same**.
- **Valid** — means that we use the original image,
- **Same** — we use the border around it, so that the images at the input and output are the same size.
- In the second case, the padding width, should meet the following equation, where p is padding and f is the filter dimension (usually odd).

$$p = \frac{f - 1}{2}$$



Step 1- Convolution

Valid Convolutional layer

Example:

Input_volume=Image or 2D matrix (height width x channels)=**W1×H1×D1** =28x28x3

2D Kernel or 2D filter= FxF(spatial filter or kernel)=**3x3xk(k is number of filters)**

In this example

k=3

Stride=1

P=0

W2=(W1-F+2P)/S+1=(28-3+2(0)/1+1)=26

H2=(H1-F+2P)/S+1 =(28-3+2(0)/1+1)=26

D2=3

Input size=28x28x3



Output size=26x26x3

Same Convolutional layer

Example:

Input_volume=Image or 2D matrix (height width x channels)=**W1×H1×D1** =28x28x3

2D Kernel or 2D filter= FxF(spatial filter or kernel)=**3x3xk(k is number of filters)**

In this example

k=3

Stride=1 $p = \frac{f - 1}{2}$
P=1

W2=(W1-F+2P)/S+1=(28-3+2(1)/1+1)=28

H2=(H1-F+2P)/S+1 =(28-3+2(1)/1+1)=28

D2=3

Input size=28x28x3

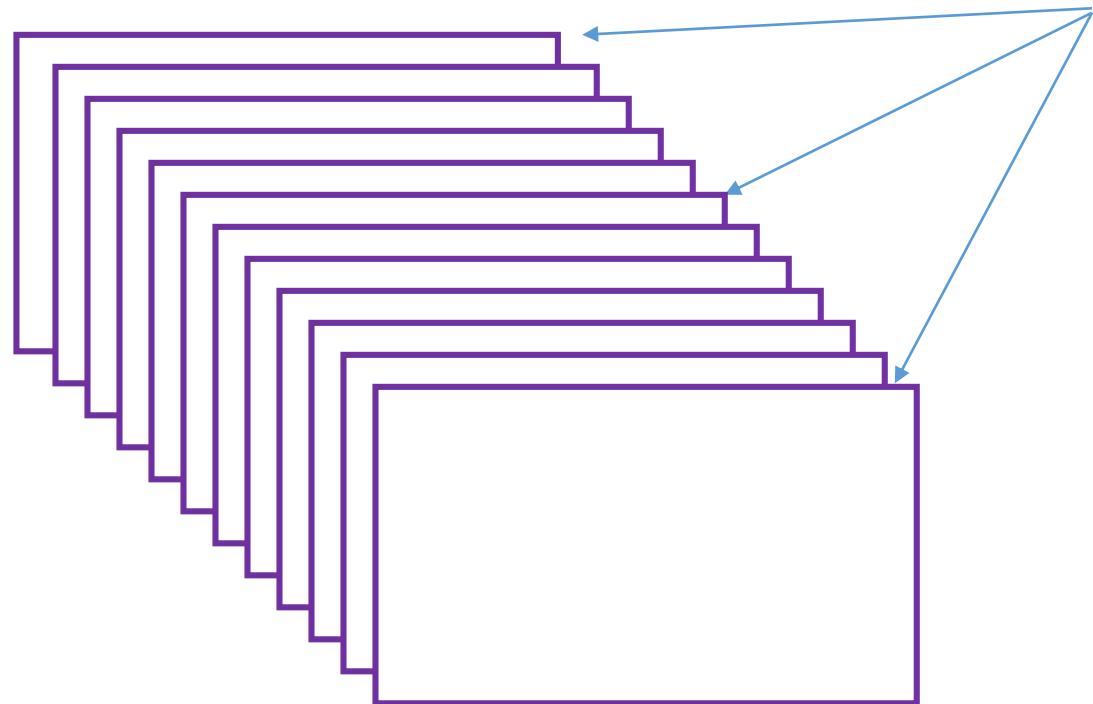


Output size=28x28x3

Step 1- Convolution

0	0	1	3	1	1	1
3	4	2	4	1	1	1
1	3	3	5	1	3	1
2	4	4	5	0	0	0
1	5	6	6	0	1	1
1	6	5	7	9	1	3
1	7	3	9	4	1	4

Input Image



Convolutional Layer

Feature Maps

Step 1- Convolution



f_1
 f_2



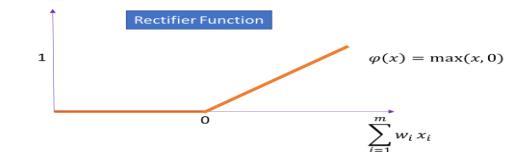
Step 1a- ReLU(Non-Linear)



Black = negative; white = positive values



Only non-negative values



Step 2- Max Pooling

Accepts a volume of size $W1 \times H1 \times D1$

Requires two hyperparameters:

- their spatial extent F ,
- the stride S ,

Produces a volume of size $W2 \times H2 \times D2$

where:

$$W2 = (W1 - F) / S + 1$$

$$H2 = (H1 - F) / S + 1$$

$$D2 = D1$$

Introduces zero parameters since it computes a fixed function of the input. For Pooling layers, it is not common to pad the input using zero-padding.

Example:

Input_volume=Image or 2D matrix (height width x channels)= $W1 \times H1 \times D1 = 28 \times 28 \times 3$

(pool_size)= $F \times F = 2 \times 2$

In this example

Stride=2

P=0

$$W2 = (W1 - F + 2P) / S + 1 = (28 - 3 + 2(0)) / 2 + 1 = 14$$

$$H2 = (H1 - F + 2P) / S + 1 = (28 - 3 + 2(0)) / 2 + 1 = 14$$

$$D2 = 3$$

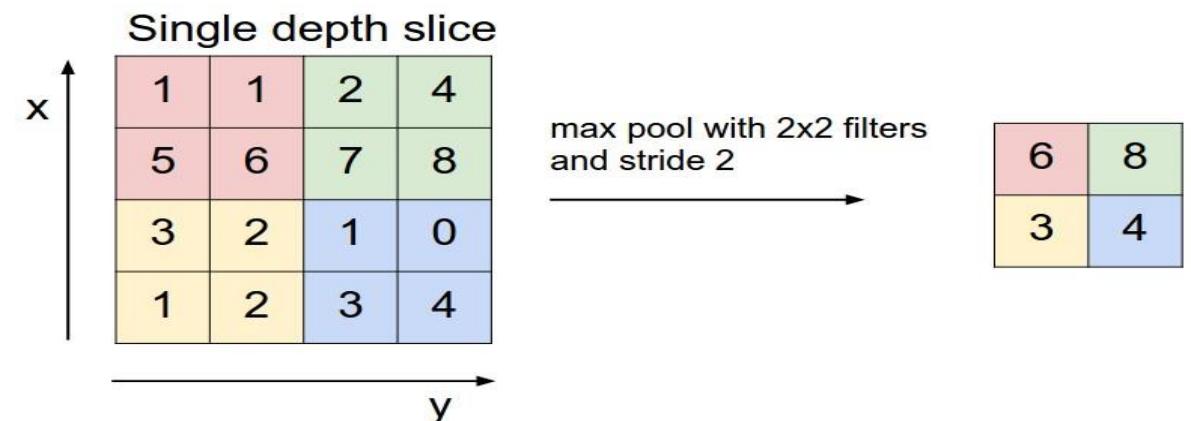
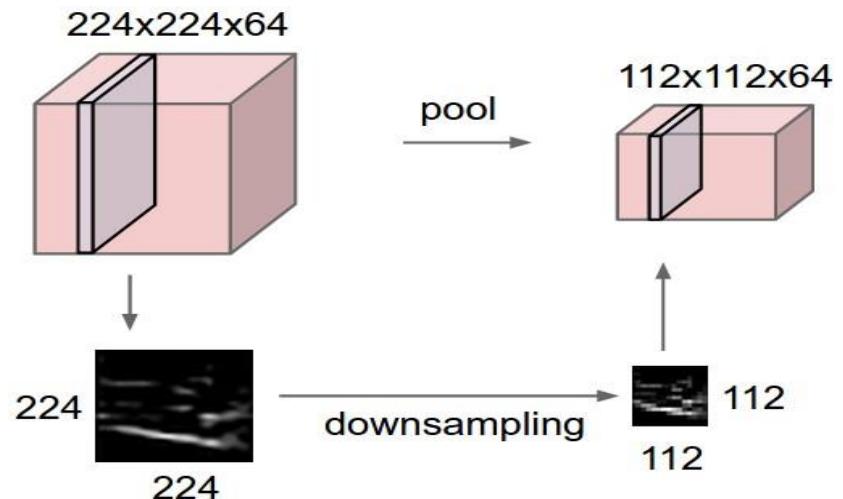
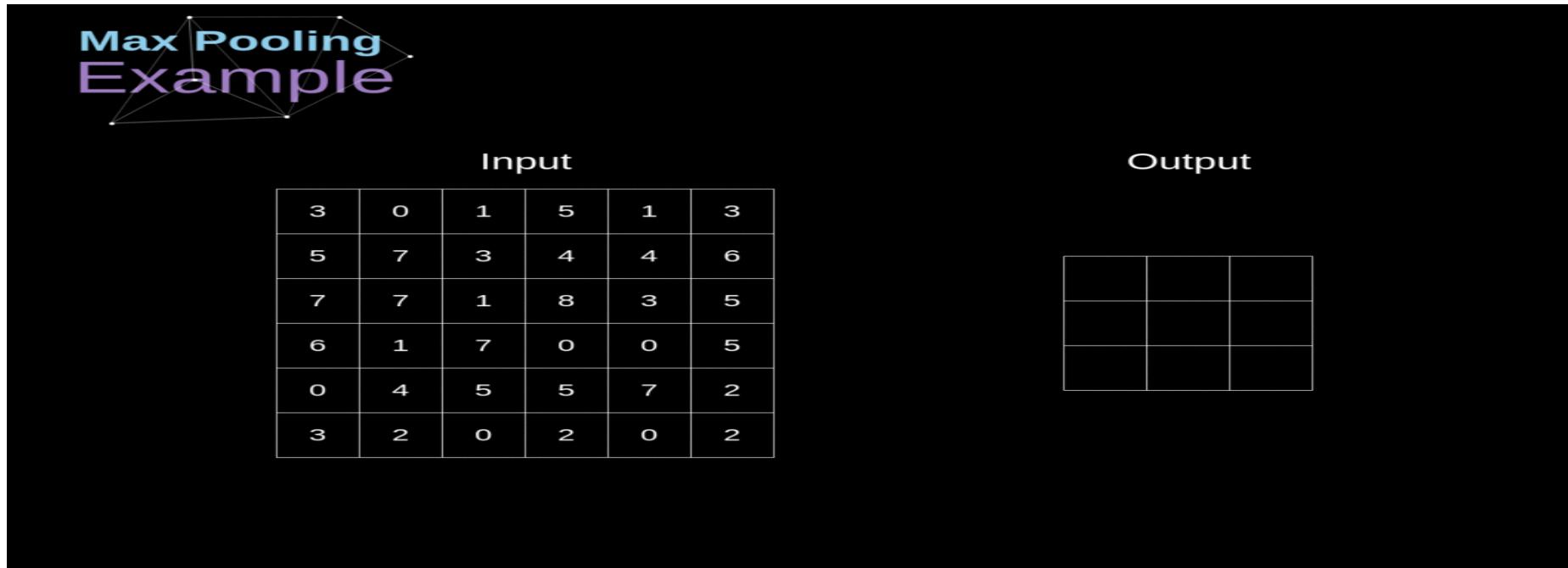
Input size=28x28x3

$\text{maxp}(2,2)$



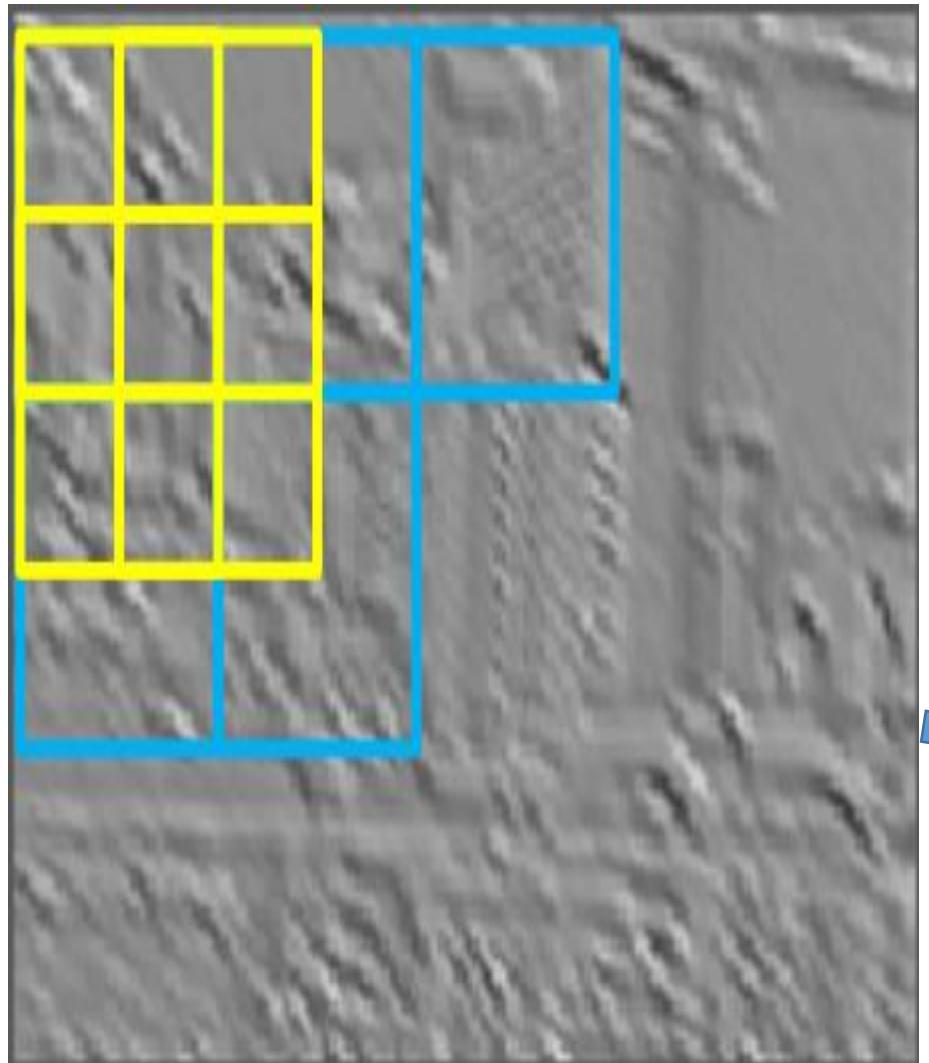
Output size=14x14x3

Step 2- Max Pooling

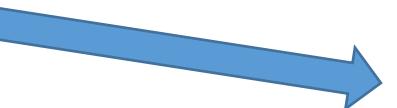
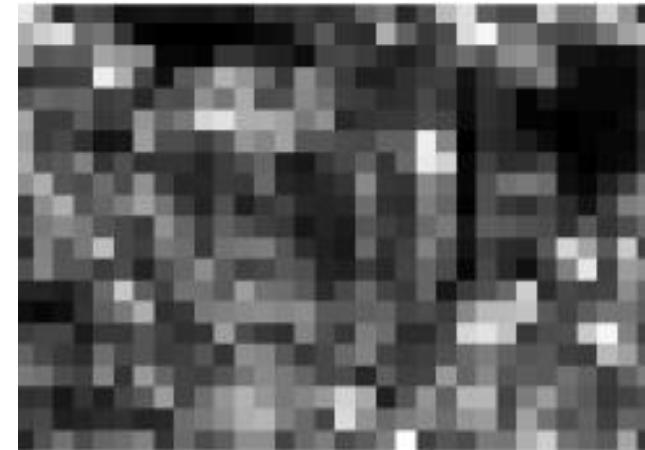


<https://cs231n.github.io/convolutional-networks/>

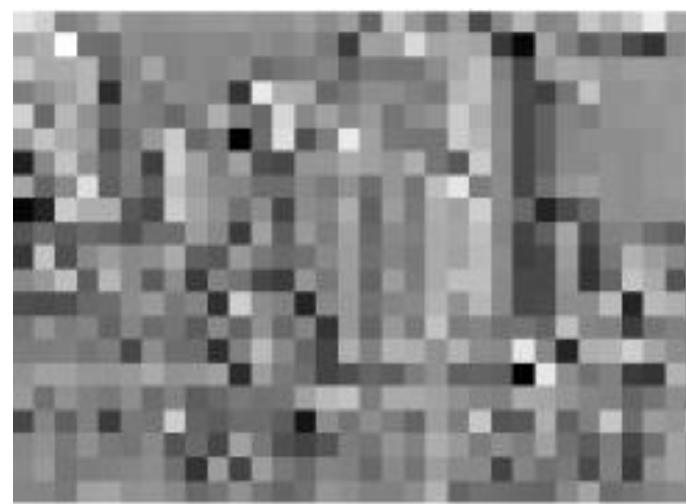
Step 2- Pooling Layers



Max



Sum



Step 3- Flattening

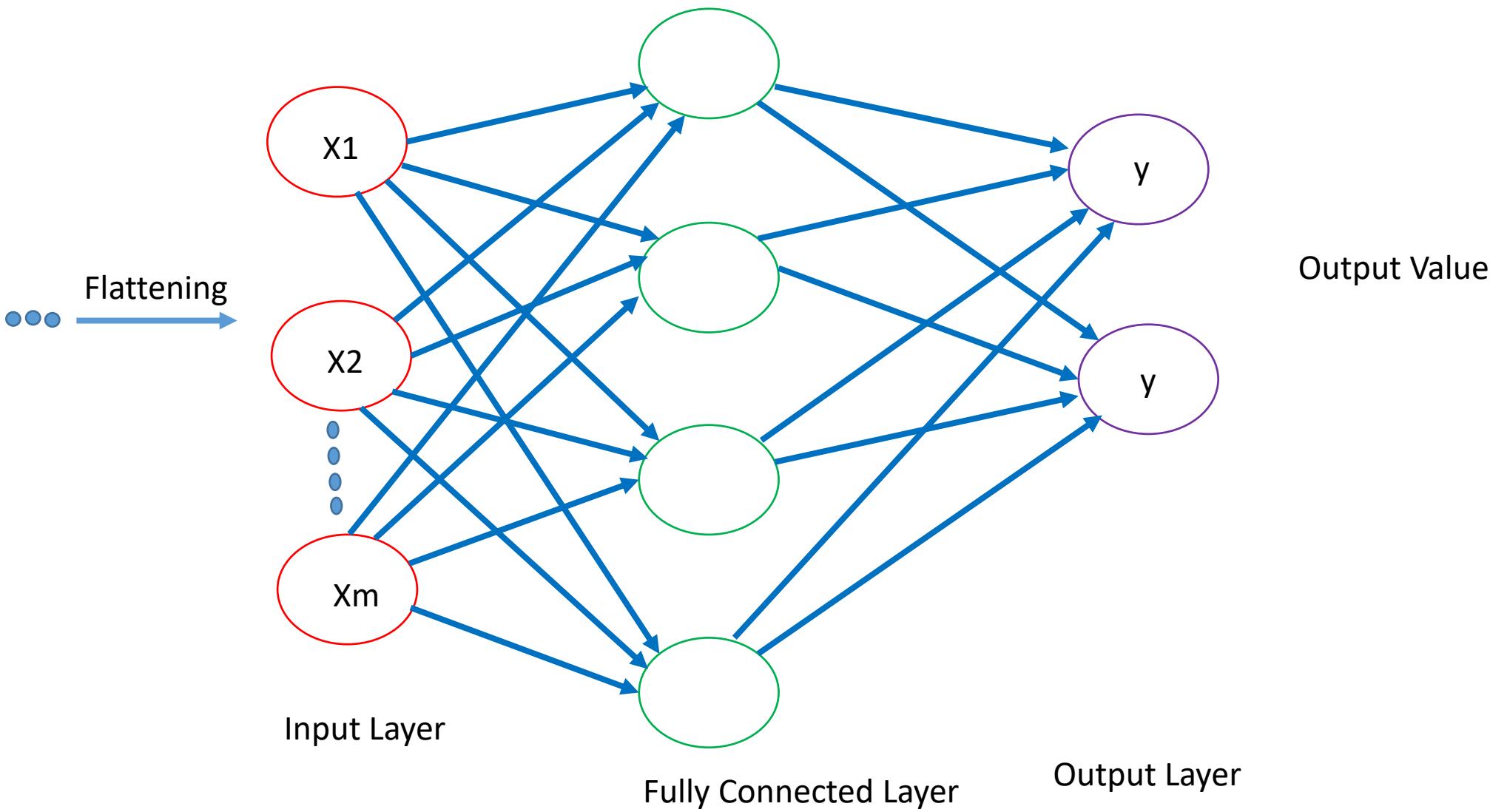
1	1	0
4	2	1
0	2	1

Pooled Feature Map

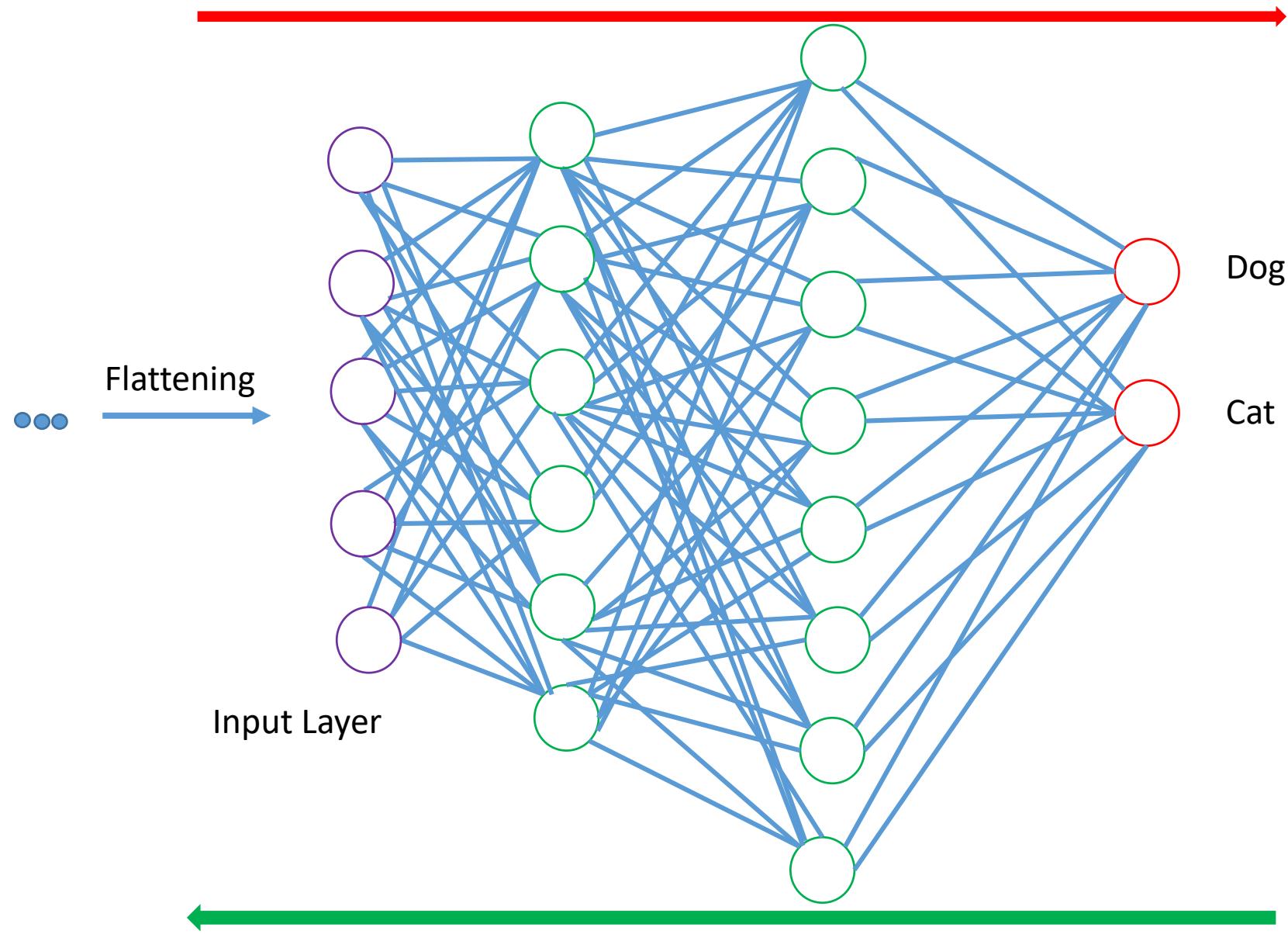
Flattening

1
1
0
4
2
1
0
2
1

Step 4- Full Connection



Step 4- Full Connection

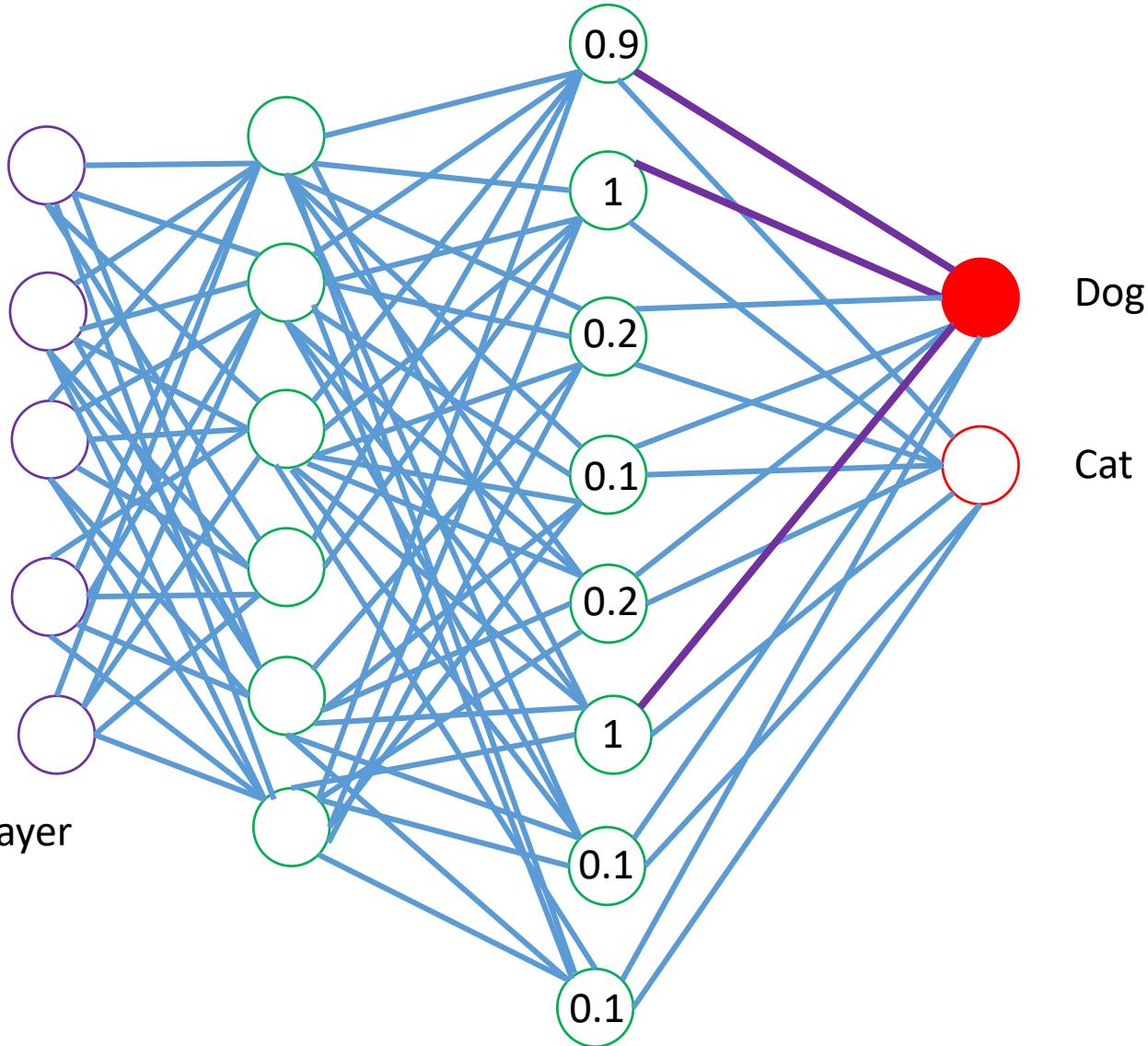


Step 4- Full Connection



Flattening
...

Input Layer



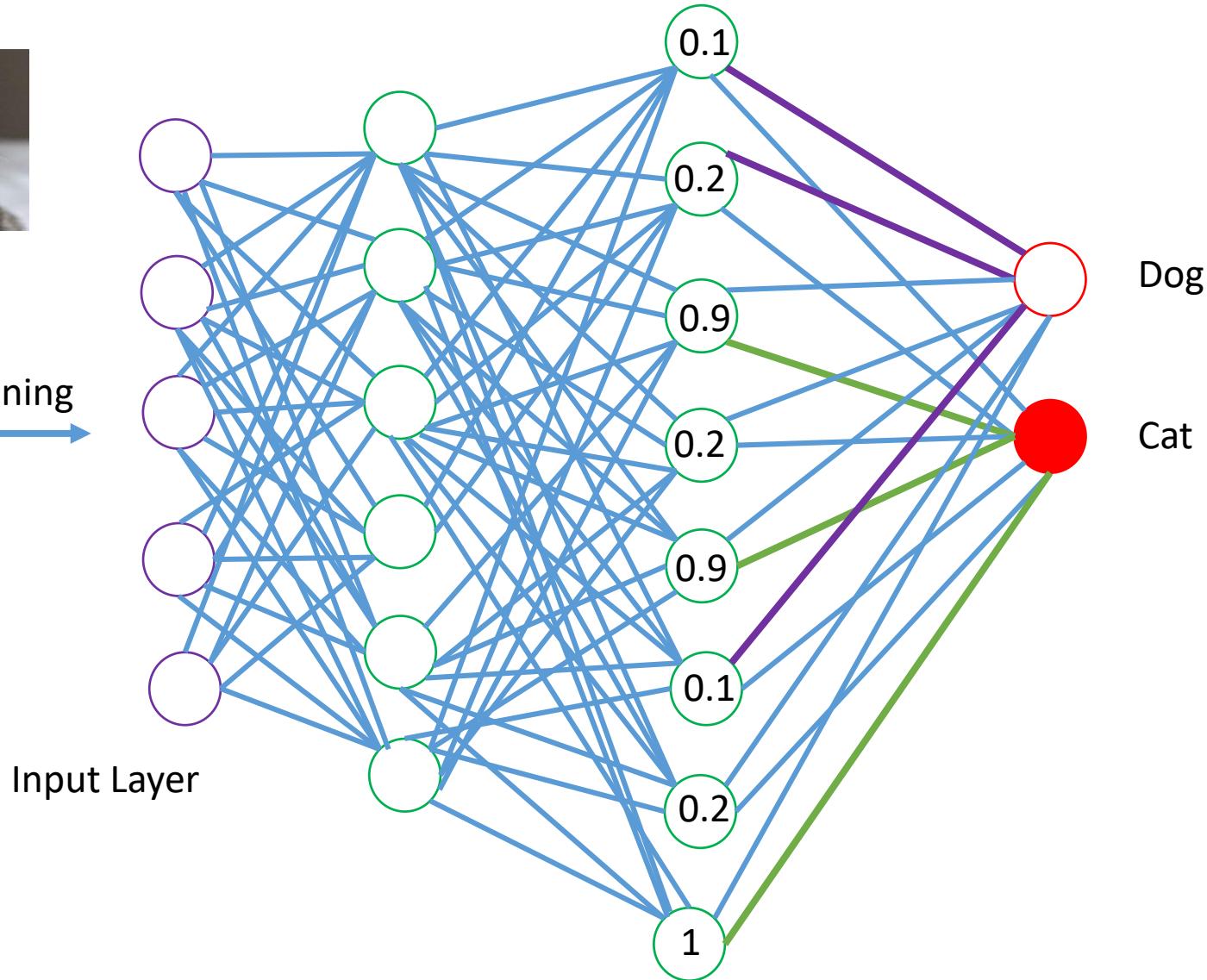
Dog

Cat

Step 4- Full Connection



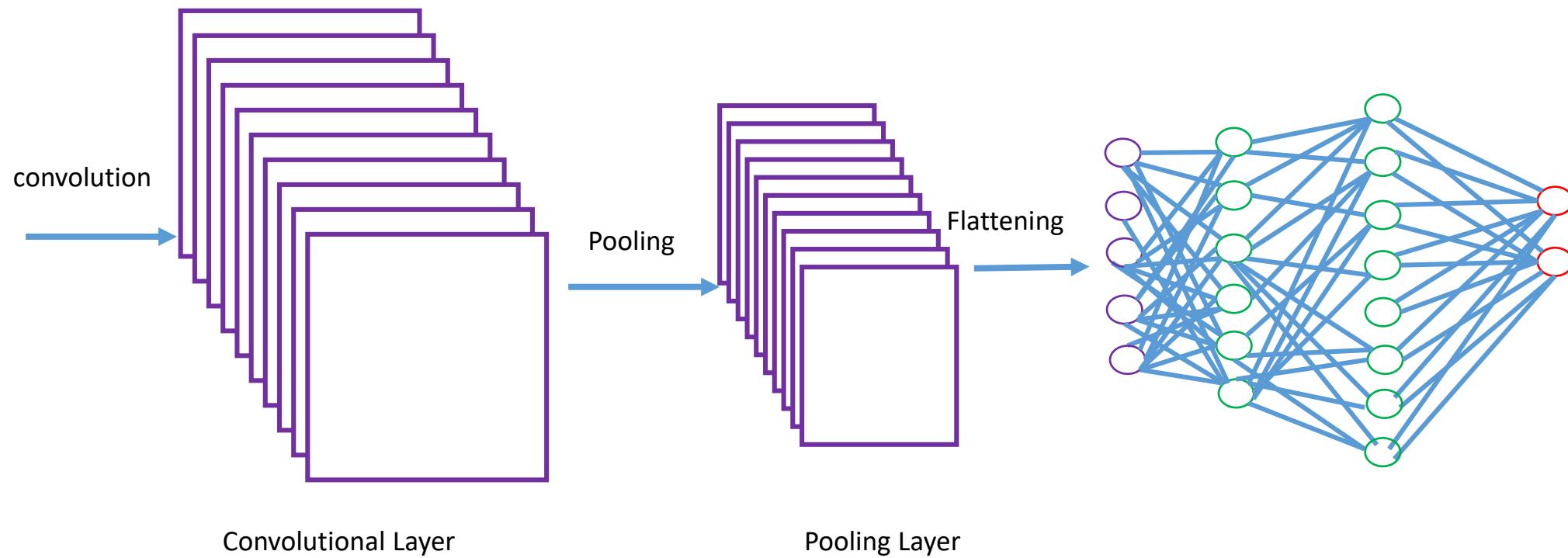
... Flattening



2D-Convolutional Neural Network

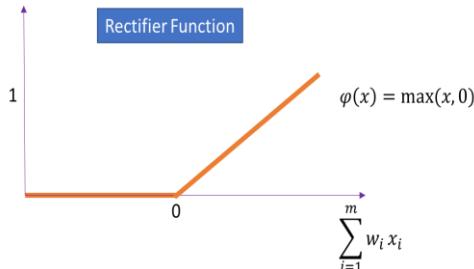
0	0	1	3	1	1	1	1
3	4	2	4	1	1	1	1
1	3	3	5	1	3	1	1
2	4	4	5	0	0	0	0
1	5	6	6	0	1	1	1
1	6	5	7	9	1	3	3
1	7	3	9	4	1	4	4

Input Image



Convolutional Layer

Pooling Layer



Batch Normalization Layer

To increase the stability of a neural network, batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

Also, batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

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

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

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

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

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

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

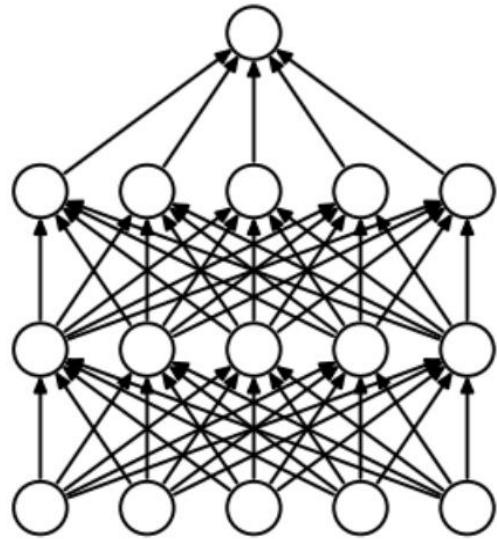
Dropout Layer

In machine learning, regularization is way to prevent over-fitting. Regularization reduces over-fitting by adding a penalty to the loss function. By adding this penalty, the model is trained such that it does not learn interdependent set of features weights. **Those of you who know Logistic Regression might be familiar with L1 (Laplacian) and L2 (Gaussian) penalties.**

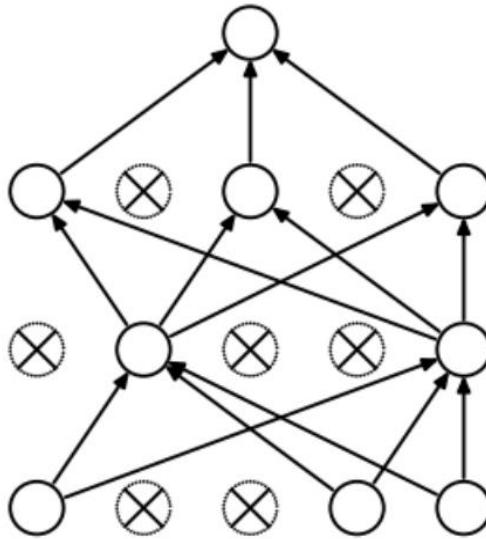
Dropout is an approach to regularization in neural networks which helps reducing interdependent learning amongst the neurons.

Training Phase:

Training Phase: For each hidden layer, for each training sample, for each iteration, ignore (zero out) a random fraction, p , of nodes (and corresponding activations).



(a) Standard Neural Net



(b) After applying dropout.

Some Observations:

- Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.
- Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.
- With H hidden units, each of which can be dropped, we have 2^H possible models. In testing phase, the entire network is considered and each activation is reduced by a factor p .

2D-Convolutional Neural Network

Number of Parameters of a Conv Layer

In a CNN, each layer has two kinds of parameters : weights and biases. The total number of parameters is just the sum of all weights and biases.

Parameters or trainable parameters are optimized during the training of deep learning layers such as convolutional layers and fully connected layers. There are no parameters associated with a MaxPool layer. The pool size, stride, and padding are hyperparameters.

No of parameters or trainable parameters are computed for Convolutional layer

$$C1 \text{ layer(parameters)} = (F1 \times F2 \times K + 1) * M$$

Where F1 and F2 are height and width of filter or kernel, K total number of filters or channels used in previous layers and M are the number of filters used in that layer

No of parameters or trainable parameters are computed for Fully connected layer 1

$$C1 \text{ layer(parameters)} = (K + 1) * M$$

K No of neurons for previous layers say convolutional or Maxpooling and M are the neurons or units used in that FC1 layer

2D-Convolutional Neural Network

Input_size
(100x100x3)

C1 layer
(kernek_size=3x3,
No.filters=7)

P1 layer
(Pool_size=2x2,
strides=2x2)

C2 layer
(kernek_size=5x5,
No.filters=20)

P2 layer
(Pool_size=3x3,
strides=3x3)

Input_size
100x100x3

98x98x7

49x49x7

45x45x20

15x15x20

Flatten layer=15x15x20=4500

4500x50

FC1 layer
No.Units=
50

50x60

FC1 layer
No.Units=
60

60x3

Output
layer(class
es=3)

No.of trainable parameters=Input_layer=0

C1 layer(parameters)= $(F_1 \times F_2 \times K + 1) \times M = (3 \times 3 \times 3 + 1) \times 7 = 196$

P1=0

C2 layer(parameters)= $(F_1 \times F_2 \times K + 1) \times M = (5 \times 5 \times 7 + 1) \times 20 = 3520$

P2=0

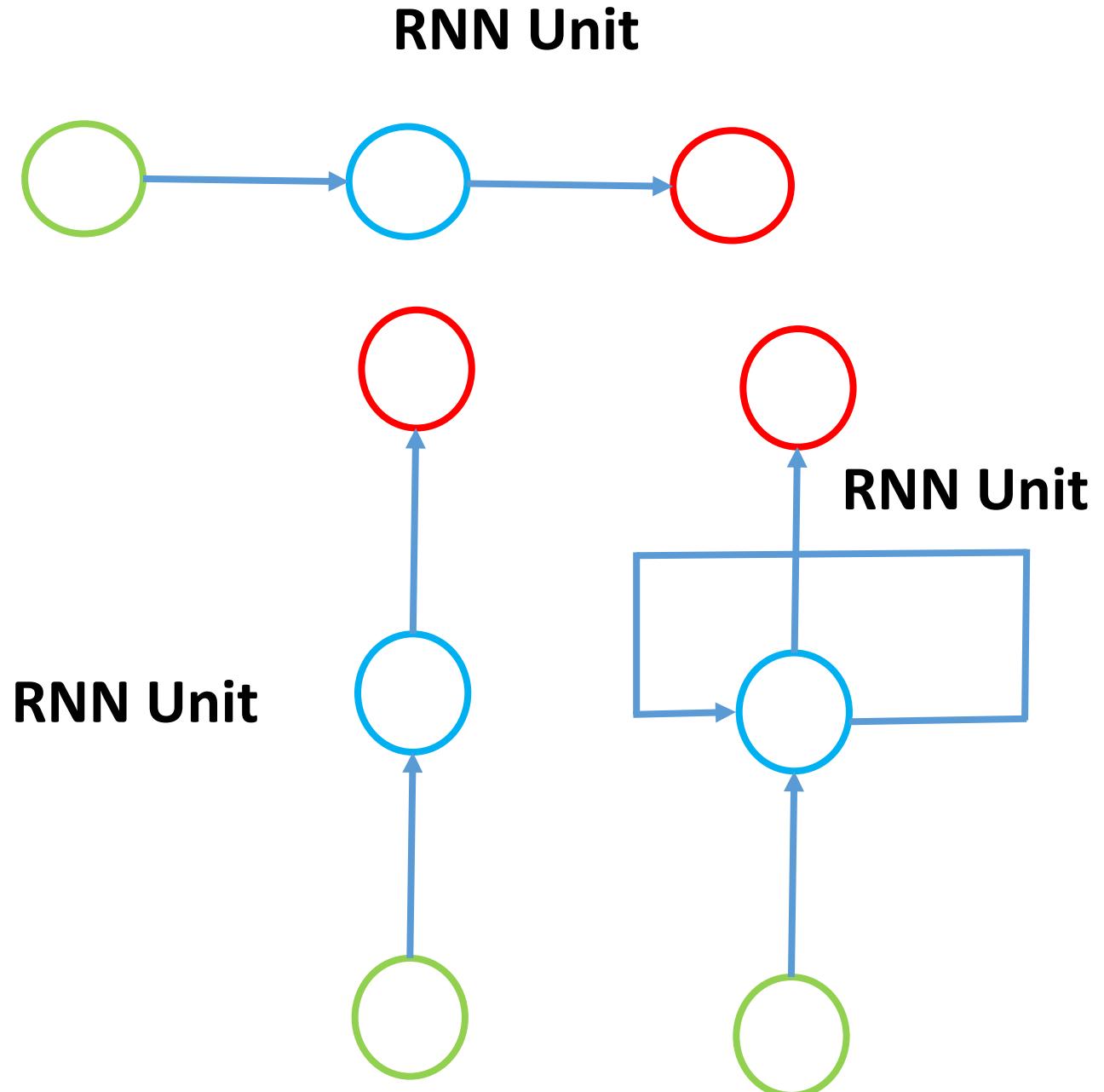
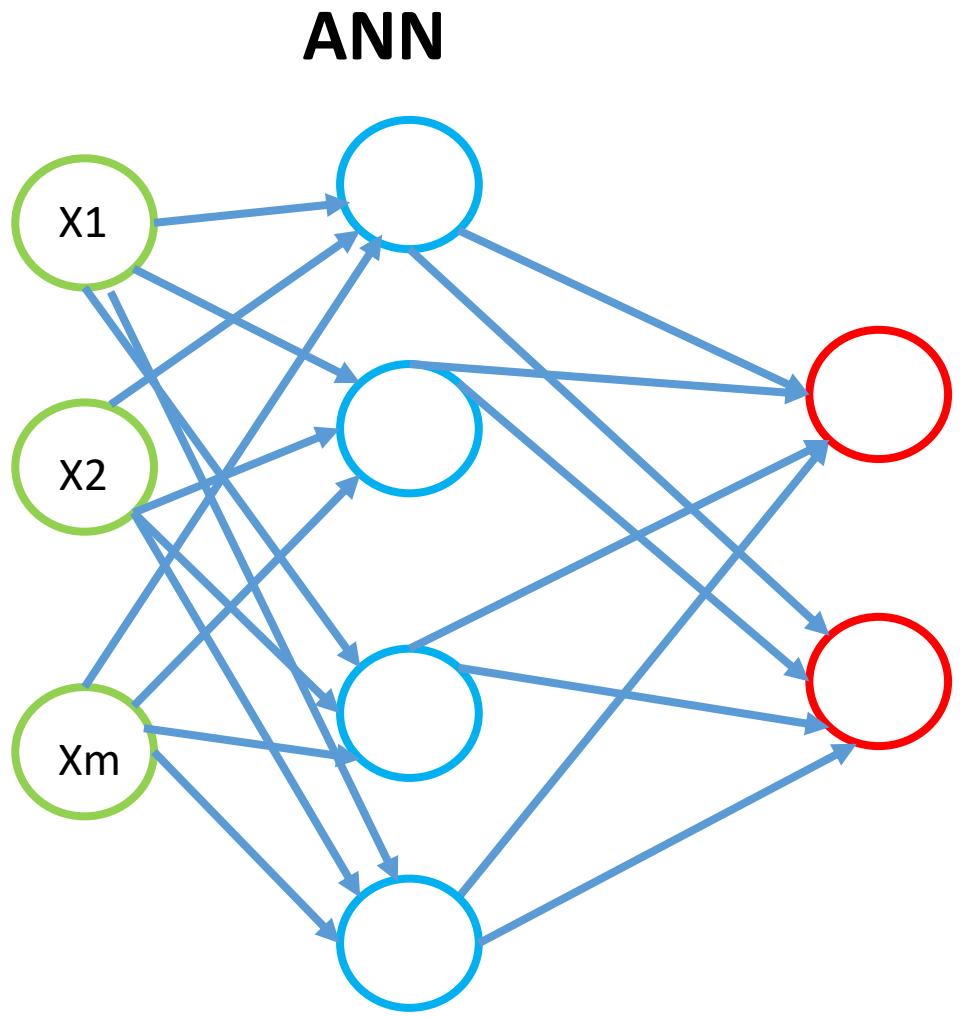
FC1(parameters)=(4500+1)*50=225,050

FC2(parameters)=(50+1)*60=3060

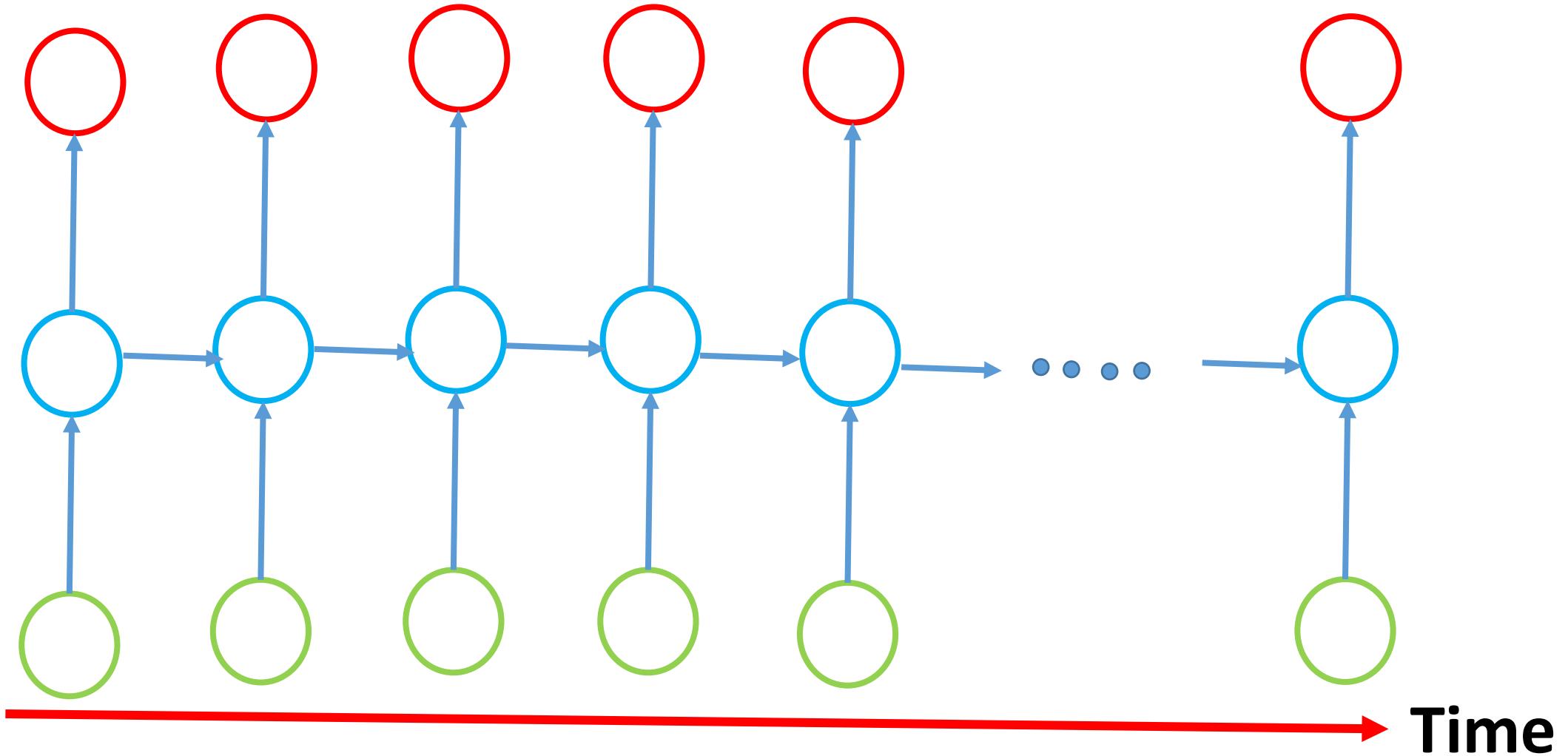
Output_layer(parameters)=(60+1)*3=183

Recurrent Neural Network (RNN)

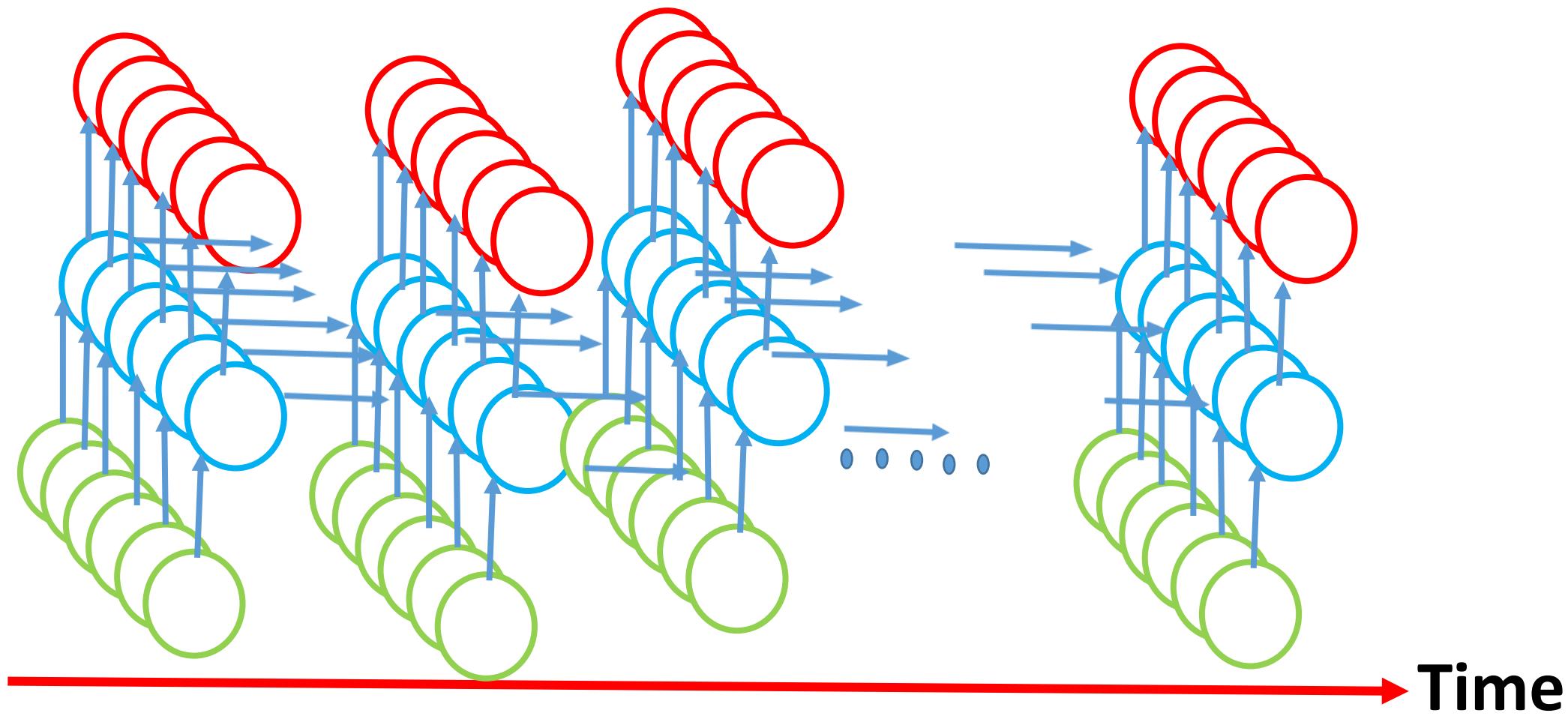
Recurrent Neural Network (RNN)



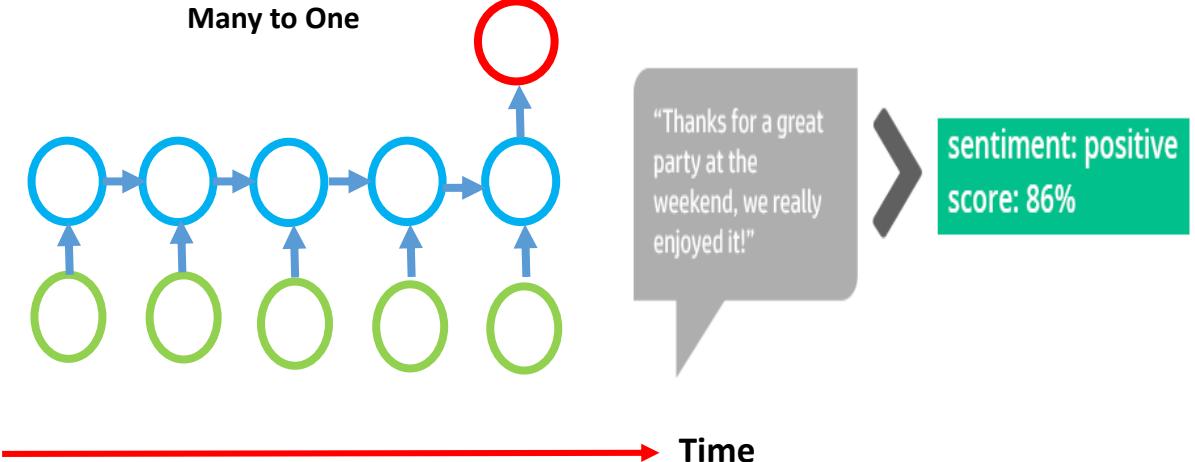
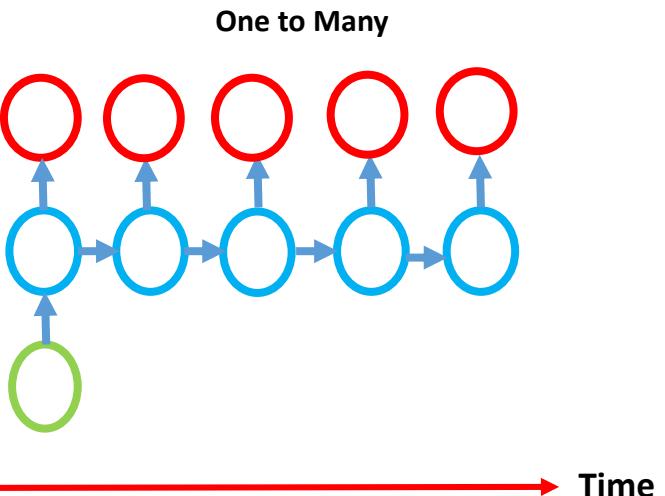
RNN



RNN



RNN Models



english to french

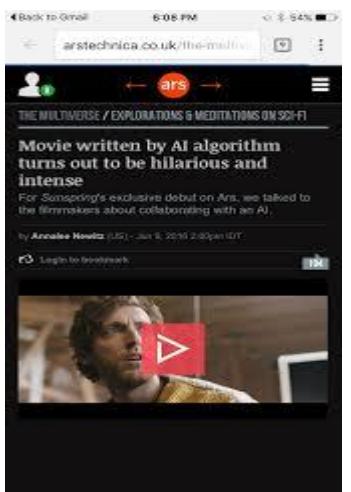
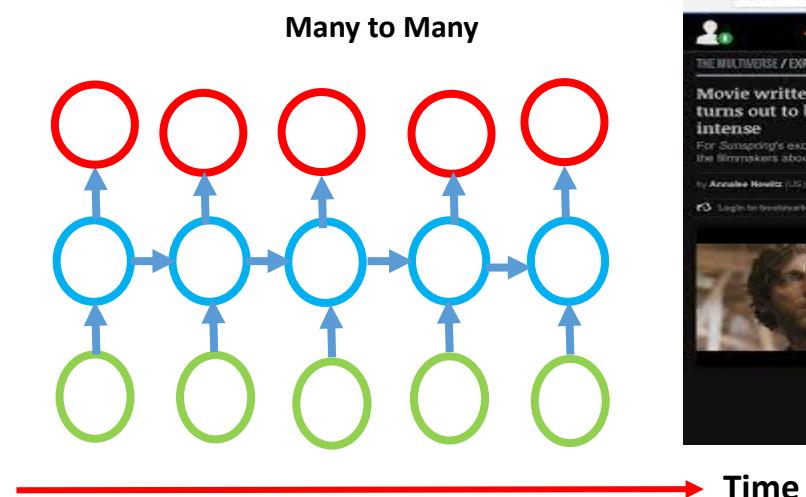
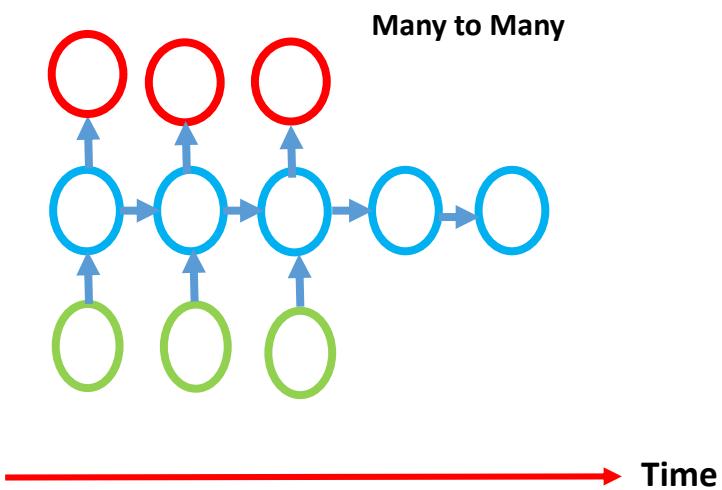
All Images News Videos Maps More Settings Tools

About 1,710,000,000 results (0.48 seconds)

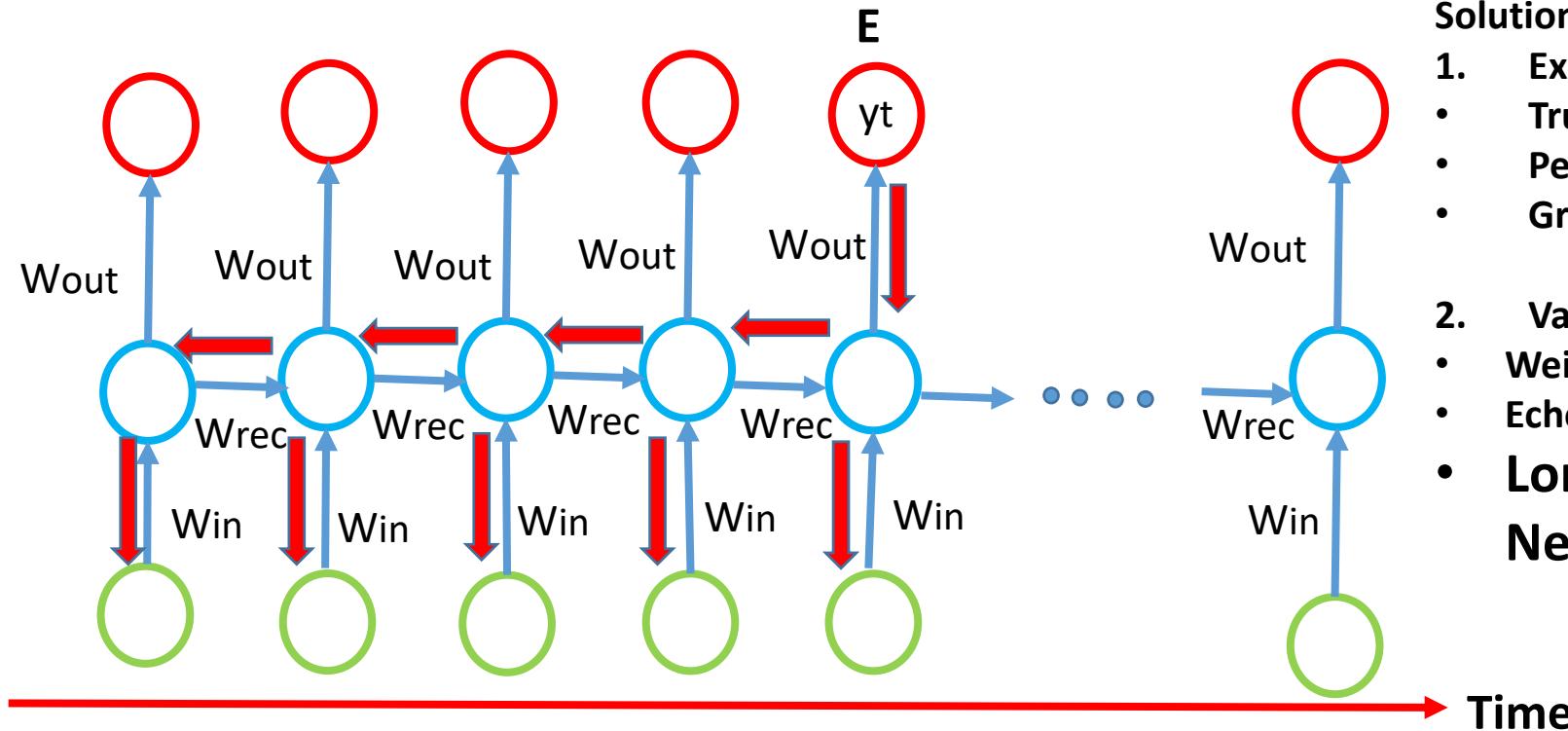
English ▾ French ▾

I am a girl who likes to learn Je suis une fille qui aime apprendre

Open in Google Translate Feedback



The Vanishing Gradient Problem



The Vanishing Gradient Problem Solution:

1. Exploding
 - Truncated Backpropagation
 - Penalties
 - Grading Clipping
2. Vanishing Gradient
 - Weight initialization
 - Echo State Networks
 - Long Short-Term Memory Networks (LSTM)

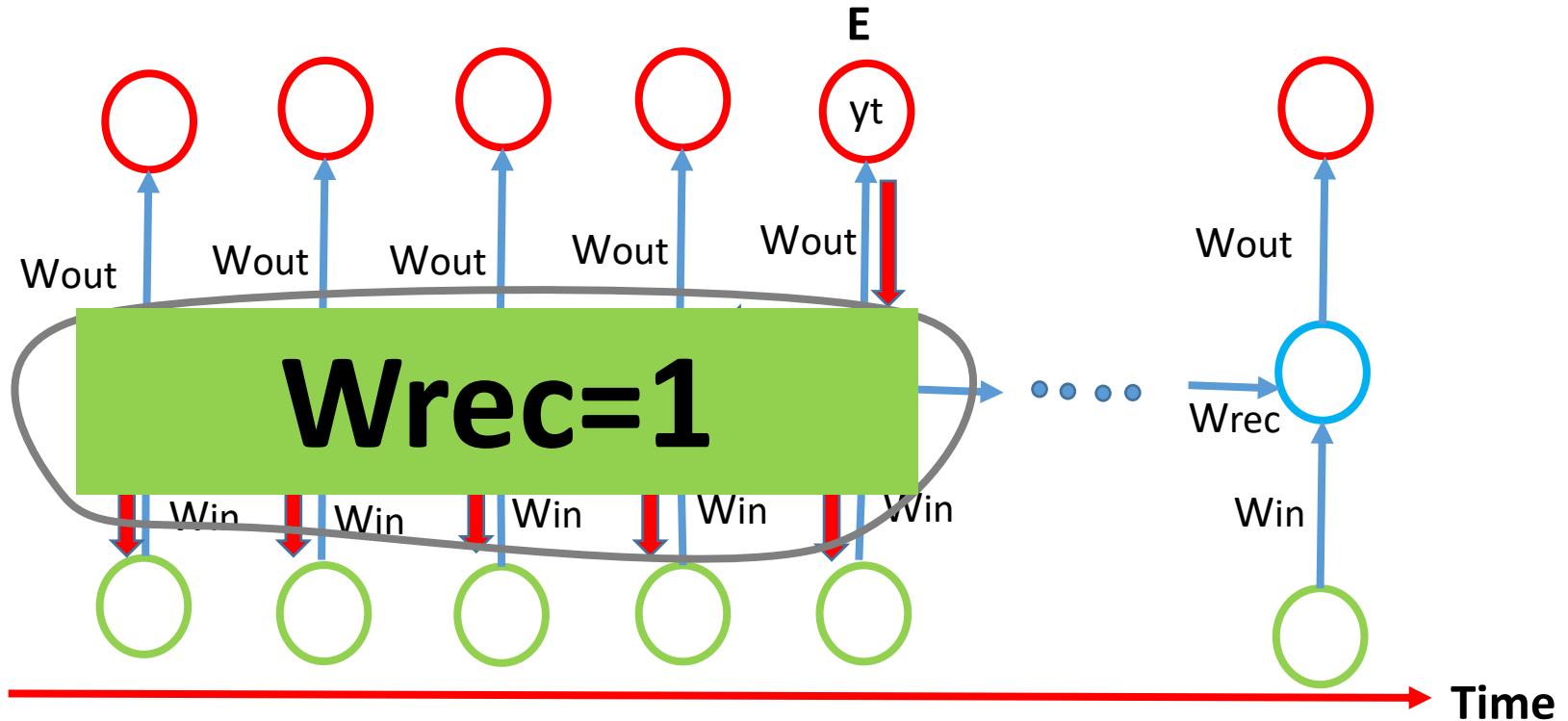
Time



$W_{rec} \sim \text{Small} \xrightarrow{\text{Blue Arrow}} \text{Vanishing}$

$W_{rec} \sim \text{Large} \xrightarrow{\text{Blue Arrow}} \text{Exploding}$

LSTMs



A horizontal sequence of five green arrows pointing left, arranged from smallest on the left to largest on the right.

Wrec ~ Small  **Vanishing**

Wrec ~ Large  **Exploding**

LSTMs Dimension Computation

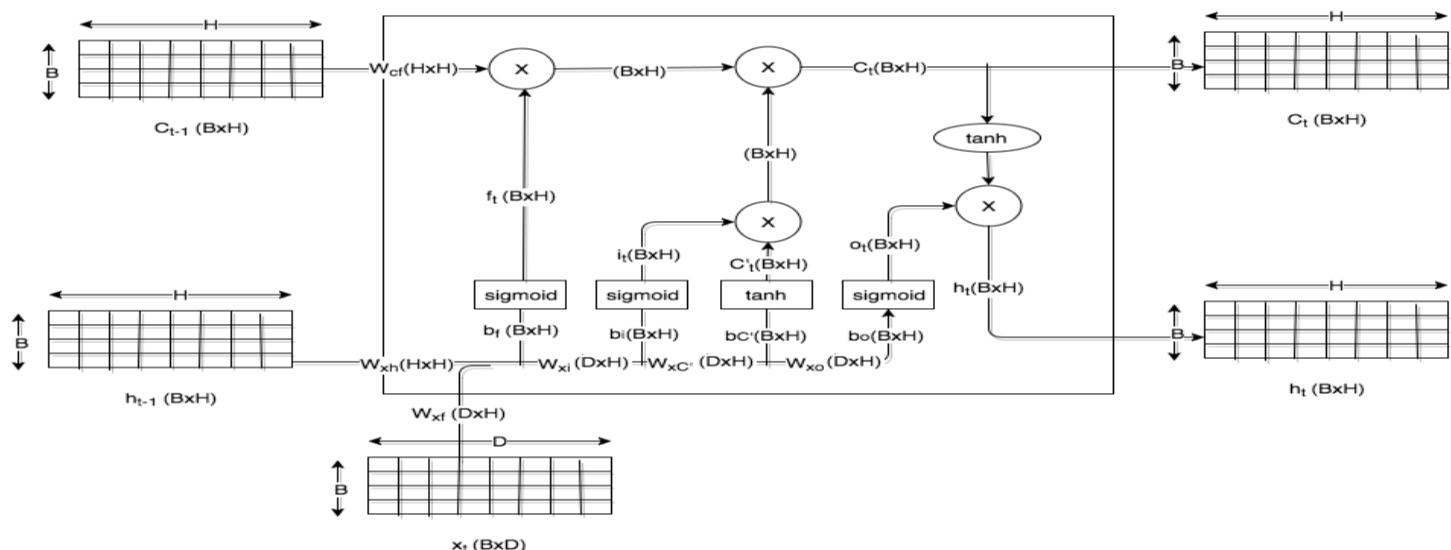
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$

$$h_t = o_t \tanh(c_t)$$



Input Dimensions

Input_size=xt=D=244x1

Hidden_size=ct-1=ht-1=H=1x128

Weights Dimensions

$W_{xi}=D \times H = 244 \times 128$

$W_{xf}=D \times H = 244 \times 128$

$W_{xc}=D \times H = 244 \times 128$

$W_{xo}=D \times H = 244 \times 128$

Weights Dimensions hidden states

$W_{hi}=H \times H = 128 \times 128$

$W_{hf}=H \times H = 128 \times 128$

$W_{hc}=H \times H = 128 \times 128$

$W_{ho}=H \times H = 128 \times 128$

Biases Dimensions

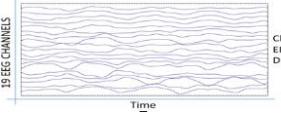
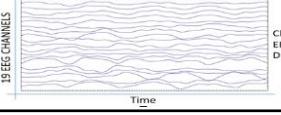
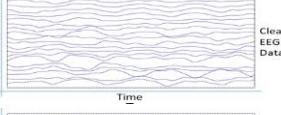
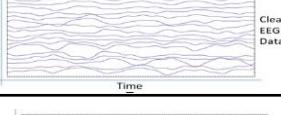
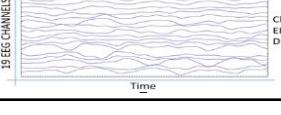
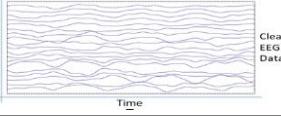
$b_i=H=1 \times 128$

$b_f=H=1 \times 128$

$b_c=H=1 \times 128$

$b_o=H=1 \times 128$

Basic dimensions in LSTM models for Training and Testing

No of samples	Classes	Dimension	Labels
1	Patient1	19x256	
2	Patient1	19x256	
	Patient1	---	0
	Patient1	---	0
50	Patient1	19x256	
51	Patient2	19x256	
	Patient2	19x256	
	Patient2	---	1
99	Patient2	---	1
100	Patient2	19x256	

Total samples x timestep x features
100 x 256 x 19

Training samples
80 x 256 x 19
Testing samples
20 x 256 x 19

Data pre-processing and Acquisitions Libraries

Python

https://www.w3schools.com/python/python_intro.asp

NumPy

<https://numpy.org/>

Pandas

<https://pandas.pydata.org/>

Matplotlib

<https://matplotlib.org/>

ITK(segmentation and registration toolkit)

<https://itk.org/>

ITK-snap for 3d Modeling and visualization

<http://www.itksnap.org/pmwiki/pmwiki.php>

NiBabel processing nifty (Neuroimaging Informatics Technology Initiative) image

<https://nipy.org/nibabel/gettingstarted.html>

Libraries used for Deep Learning and Machine Learning Implementation

Machine Learning library

Scikit-Learn

<https://scikit-learn.org/stable/>

Scikit-learn plot

<https://scikit-plot.readthedocs.io/en/stable/Quickstart.html>

Yellowbrick (visualization)

<https://www.scikit-yb.org/en/latest/>

MATLAB

<https://fr.mathworks.com/products/matlab.html>

Deep Learning libraries

Tensorflow

<https://www.tensorflow.org/tutorials>

Pytorch

<https://pytorch.org/docs/stable/index.html>

Keras

<https://keras.io/>

MATLAB

<https://fr.mathworks.com/products/matlab.html>



Q&A