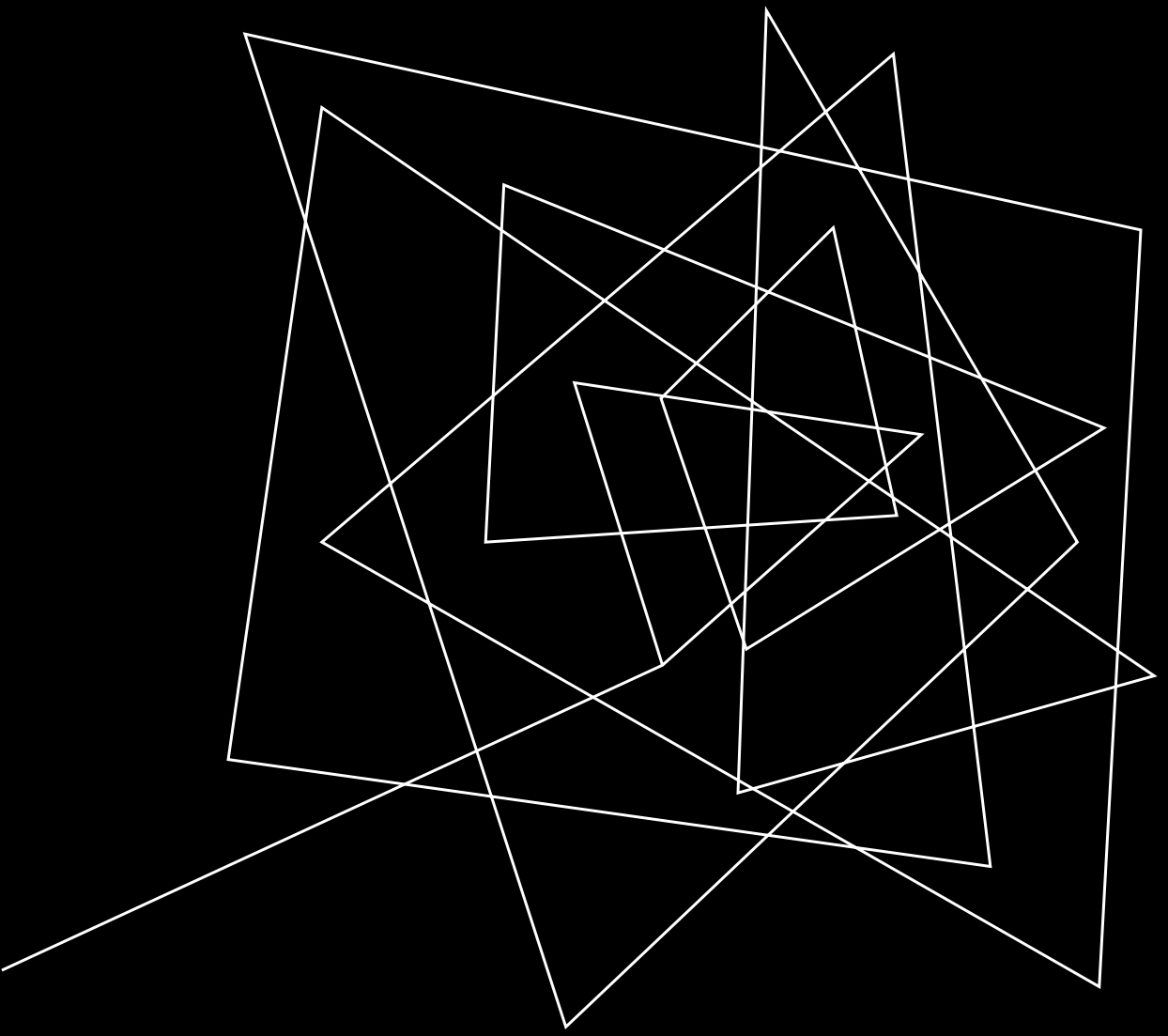
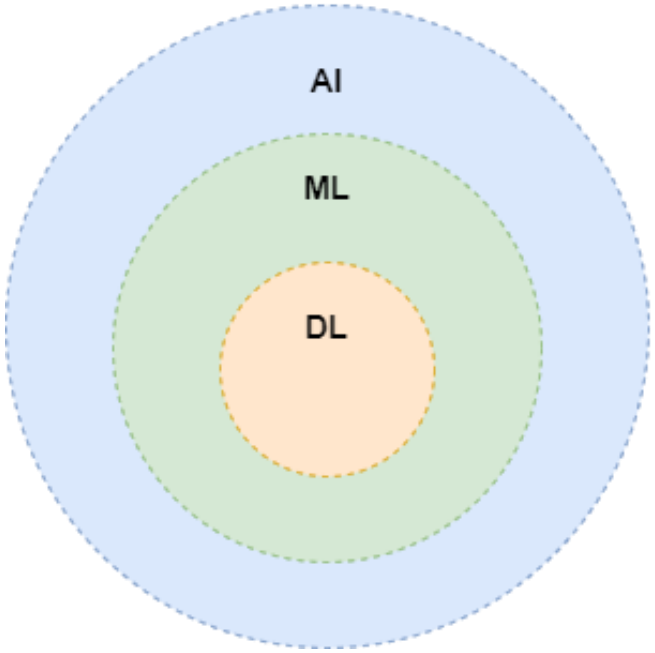


SESSION 5: INTRODUCTION TO DEEP LEARNING

Introduction to deep learning and different types of architectures

Introduction: Why Deep Learning?

The first question someone can ask after knowing about "classical" machine learning (linear regression, logistic regression, decision tree, KNN, ...etc.) is, **why do we need deep learning?**



Artificial Intelligence

Class of problems we can solve when computers think/act like humans

Machine Learning

The science of getting computers to learn without being explicitly programmed. Data + Algorithms

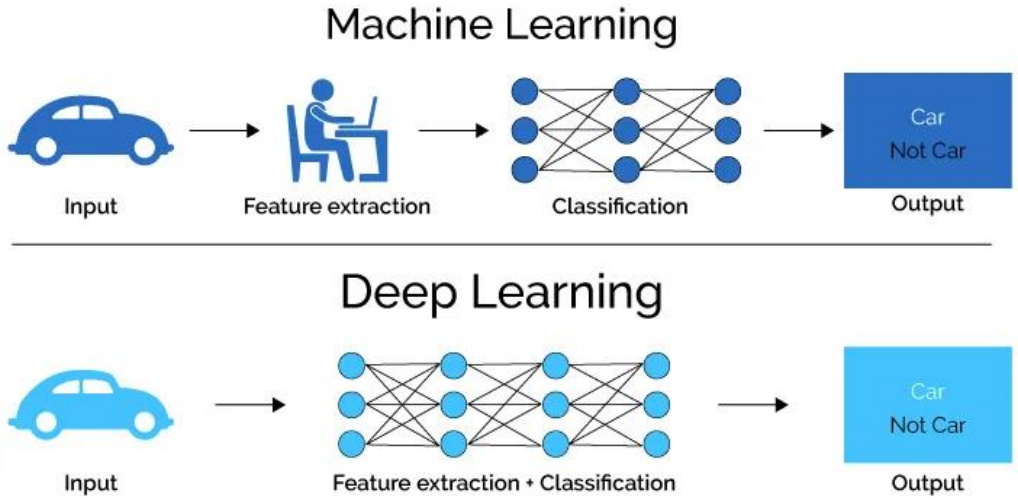
Deep Learning

Subset of machine learning that is based on neural networks to mimic the human brain. Generally outperforms classical machine learning on unstructured data

Introduction: Why Deep Learning?
<Automatic Feature Extraction>

Automatic Feature Extraction:

- Deep learning algorithms automatically learn relevant features during training, alleviating the burden of manually selecting and engineering features. This is especially advantageous when dealing with high-dimensional data.
- Feature extraction (aka feature engineering) is the process of putting domain knowledge into the creation of feature extractors to reduce the complexity of the data and make patterns more visible to learning algorithms. This process is difficult and expensive in terms of time and expertise.

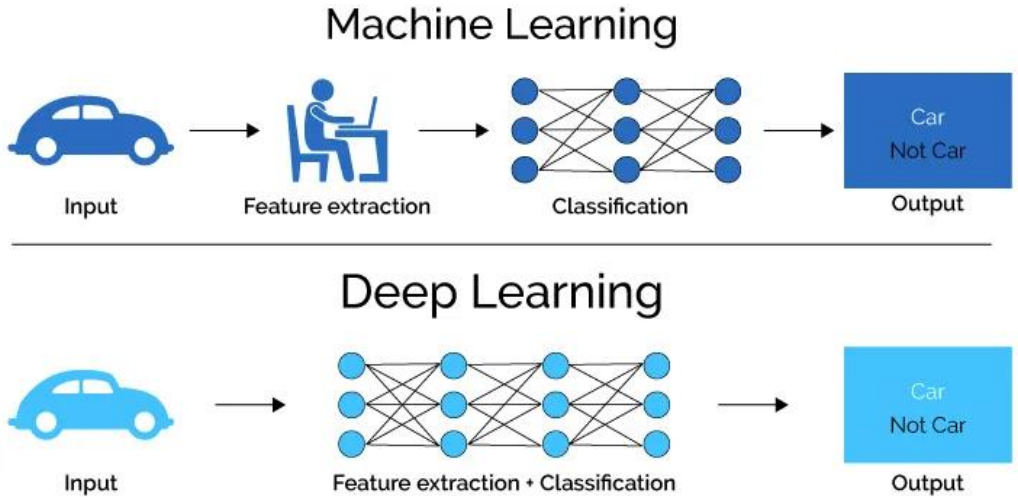


Feature extraction: Machine learning vs Deep learning

Introduction: Why Deep Learning?
<Automatic Feature Extraction>

This is best explained via an example:

- Assume you are building a system that will learn to classify images as either Car or Not Car.
- In classical ML, the algorithmic approach will use data to learn whether the image is Car or Not Car. To help this along, it might have been possible for a human to label constituent features indicative of Car (e.g. wheels) in the images thereby providing extra features with which the system can assess.
- By contrast, a DL solution will also attempt to determine which parts of the image make up the car, e.g. wheels, wing mirrors, headlamps, windscreen etc.
- As a result, DL can reduce the amount of hard coding humans have to apply to define features in datasets. This is the difference between having to label the image as Car vs. Not Car and having to do that plus label other data about the image such as wheels, windscreen, wing mirror etc that indicate Car or Not Car.

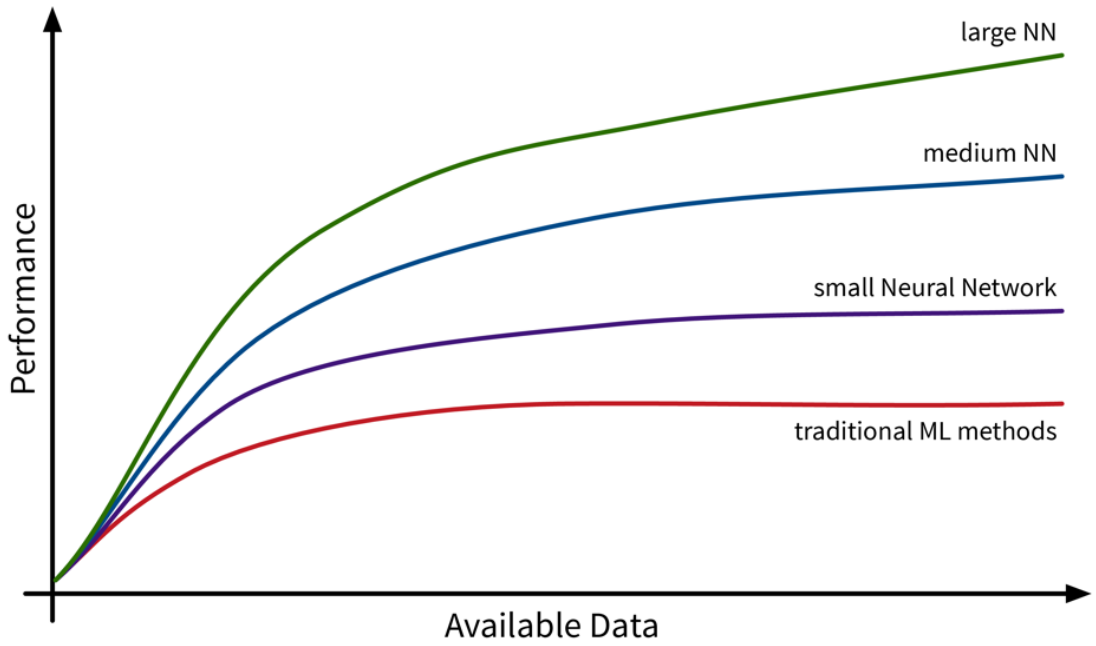


Feature extraction: Machine learning vs Deep learning

Introduction: Why Deep Learning?
<Handling Big Data>

Handling Big Data:

- Deep learning algorithms thrive on large datasets. They can effectively utilize massive amounts of data to generalize well and make accurate predictions, whereas traditional machine learning models might struggle with scalability.
- Another factor driving the spread of ML is the availability of (digital) data. Companies like Google, Amazon, and Meta have had a head start here, as their business model was built around data from the start, but other companies are starting to catch up. While traditional ML models do not benefit much from all this available data, large neural network models with many degrees of freedom can now show their full potential by learning from all the texts and images posted every day on the Internet



Introduction: Why Deep Learning?

1. **Representation Learning:** Deep learning excels at learning hierarchical representations of data. It automatically learns features from raw data, eliminating the need for manual feature engineering, which is common in traditional machine learning.
2. **Complex Pattern Recognition:** Deep learning is particularly effective in handling complex patterns and relationships within data. The multiple layers in neural networks enable the model to capture intricate and abstract features.
3. **Handling Big Data:** Deep learning algorithms thrive on large datasets. They can effectively utilize massive amounts of data to generalize well and make accurate predictions, whereas traditional machine learning models might struggle with scalability.
4. **Feature Abstraction:** Deep learning models automatically abstract and extract relevant features from the input data. This allows the model to focus on the most important aspects, leading to better generalization.
5. **Improved Performance in Unstructured Data:** For tasks involving unstructured data like images, audio, and text, deep learning models, especially Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have demonstrated superior performance compared to traditional methods.

Introduction: Why Deep Learning?

1. **End-to-End Learning:** Deep learning models can learn end-to-end, meaning they can take raw input data and produce a final output without the need for intermediate processing steps. This simplifies the modeling process and can lead to more accurate results.
2. **Adaptability to Varied Tasks**: Deep learning models are versatile and can be applied to a wide range of tasks, from image and speech recognition to natural language processing. Their adaptability makes them suitable for a variety of domains.
3. **Automatic Feature Extraction:** Deep learning algorithms automatically learn relevant features during training, alleviating the burden of manually selecting and engineering features. This is especially advantageous when dealing with high-dimensional data.
4. **Continuous Improvement with Data:** Deep learning models can continuously improve with more data. As the amount of available data increases, the model's performance often improves, making it a scalable solution for evolving datasets.
5. **State-of-the-Art Performance:** In many domains, deep learning models have achieved state-of-the-art performance, outperforming traditional machine learning methods. This makes them the go-to choice for cutting-edge applications.

Introduction: Why Deep Learning?
<Classical ML vs. Deep Learning>

Aspect	Classical machine learning	Deep learning
Data size	Can perform well with smaller datasets.	Often requires large amounts of data for effective training.
Feature Engineering	Often requires domain knowledge for manual feature creation.	Automatically learns hierarchical features.
Computation	Computationally less intensive compared to deep learning.	Requires powerful hardware, like GPUs, for efficient training.
NLP + Computer vision + Audio/Speech recognition	Limited and less effective in these tasks	Highly effective and represents state-of-the-art for these tasks
Human-like Learning	Relies on human-engineered features and rules.	Can automatically learn complex patterns like humans.

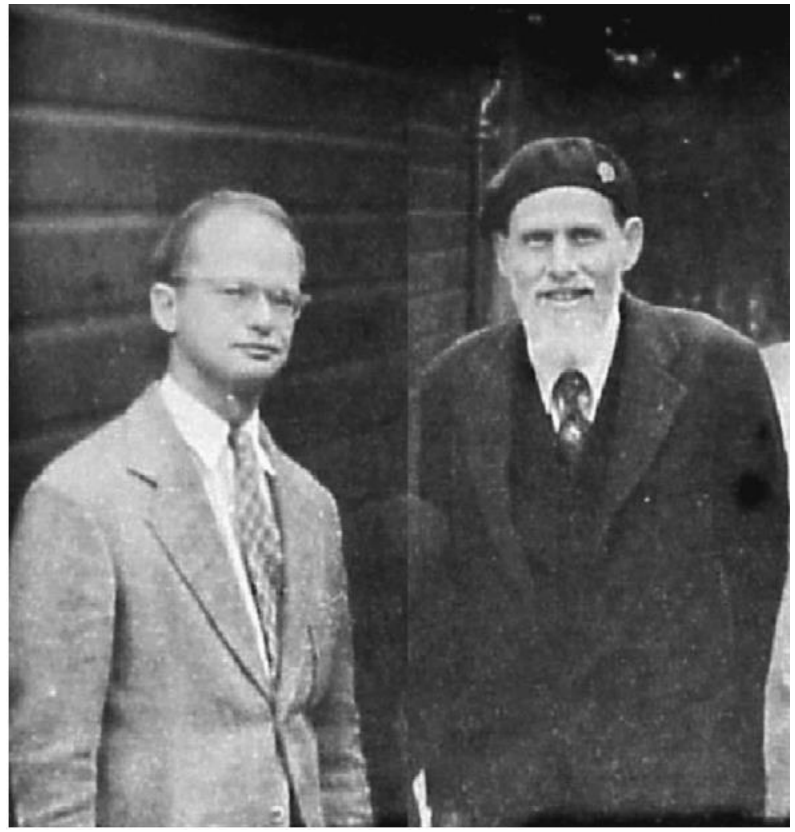
Table. The comparative table between classical machine learning (ML) and deep learning (DL) based on various aspects

History of Deep Learning: 1943: Warren McCulloch and Walter Pitts

McCulloch-Pitts Neuron (1943):The concept of an artificial neuron was first introduced by Warren McCulloch and Walter Pitts. They proposed a mathematical model of a simplified neuron, capable of binary decision-making based on weighted inputs.

Paper title: A logical calculus of the ideas immanent in nervous activity

In the paper, they explain how they were able to program artificial neurons inspired by the functioning of biological neurons



McCulloch (right) and Pitts (left) in 1949
Image Source: www.semanticscholar.org

History of Deep Learning: 1943: Warren McCulloch and Walter Pitts

Neurons are excitable cells connected to each other and whose role is to transmit information in our nervous system

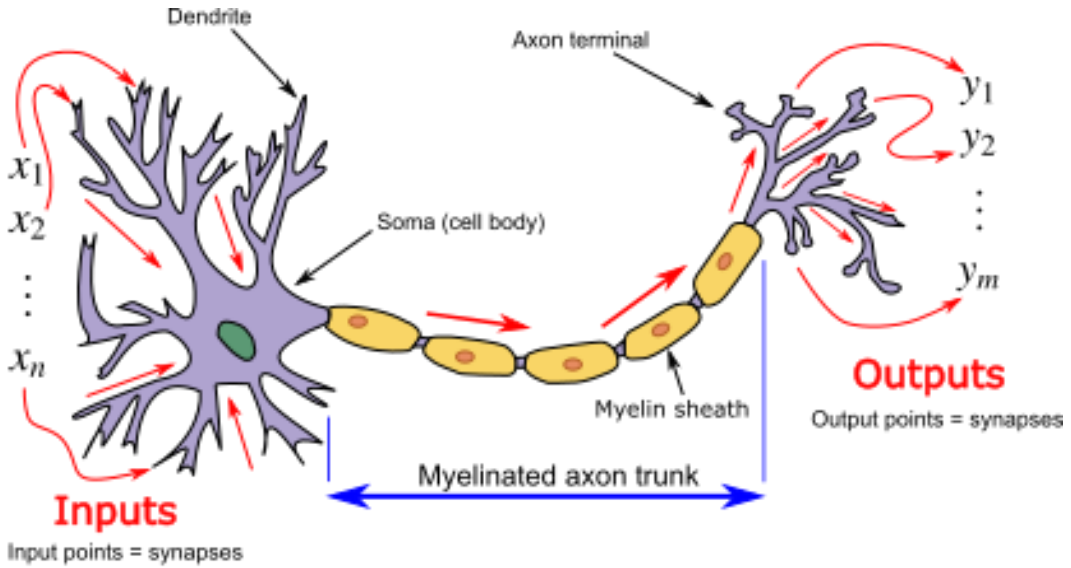


Fig. Neuron and myelinated axon, with signal flow from inputs at dendrites to outputs at axon terminals. The signal is a short electrical pulse called action potential or 'spike'.
Source: Wikipedia.org

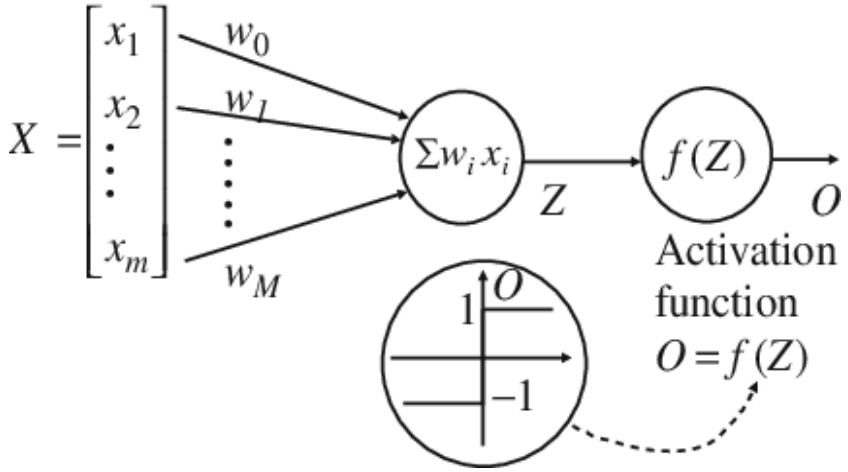
History of Deep Learning: 1943: Warren McCulloch and Walter Pitts <Threshold Logic Unit>

Mathematical Definition:

McCulloch and Pitts developed a mathematical formulation know as linear threshold gate (or threshold logic unit), which describes the activity of a single neuron with two states, firing or not-firing. In its simplest form, the mathematical formulation is as follows:

$$Sum = \sum_{i=1}^{N} I_i W_i$$

$$y(Sum) = \begin{cases} 1, & \text{if } Sum \geq T \\ 0, & \text{otherwise} \end{cases}$$



Source: pabloinsente.github.io

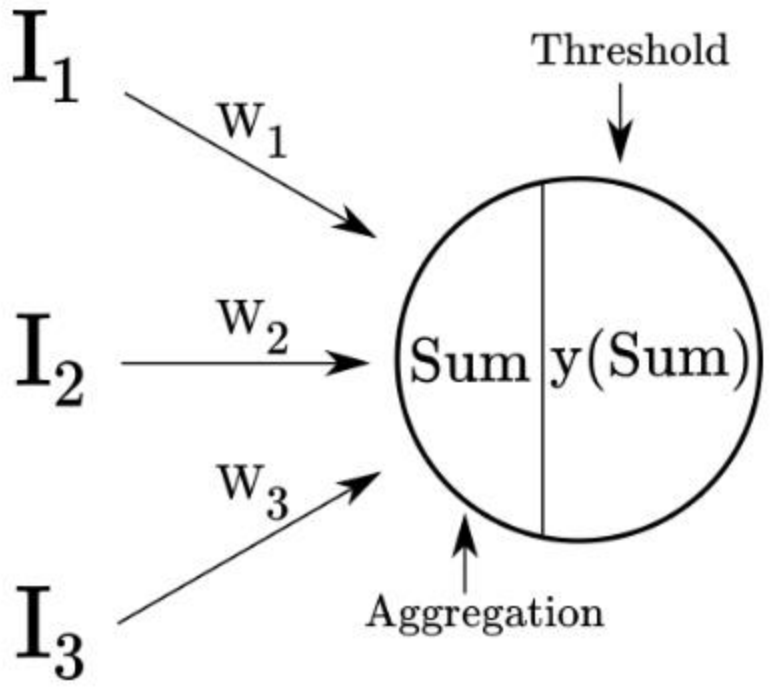
History of Deep Learning: 1943: Warren McCulloch and Walter Pitts <Threshold Logic Unit>

Mathematical Definition:

Where $I_1, I_2, \ldots, I_N$ are binary input values $\in \{0,1\}$; $W_1, W_2, \ldots, W_N$ are weights associated with each input $\in \{-1,1\}$; *Sum* is the weighted sum of inputs; and T is a predefined threshold value for the neuron activation (i.e., firing).

$$Sum = \sum_{i=1}^{N} I_i W_i$$

$$y(Sum) = \begin{cases} 1, & \text{if } Sum \geq T \\ 0, & \text{otherwise} \end{cases}$$



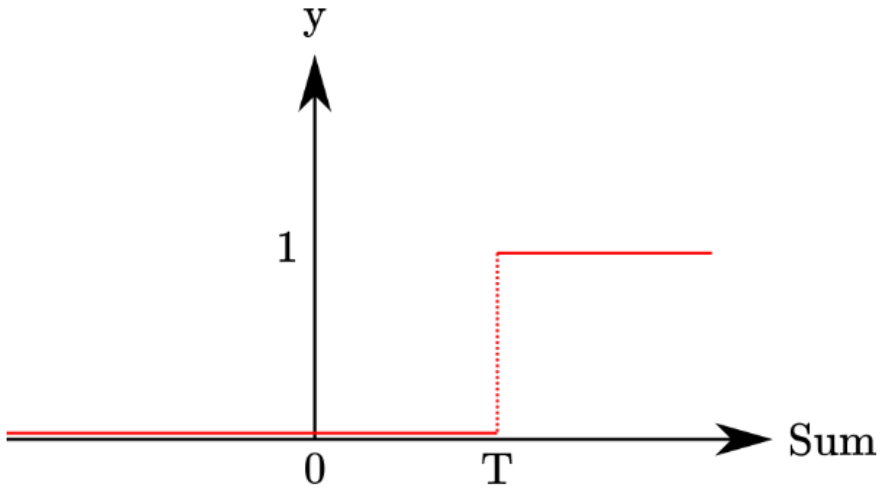
History of Deep Learning: 1943: Warren McCulloch and Walter Pitts <Threshold Logic Unit>

Mathematical Definition:

An input is considered **excitatory** when its contribution to the weighted sum is **positive**, for instance, $I_1 * W_1 = 1 * 1 = 1$; whereas an input is considered **inhibitory** when its contribution to the weighted sum is **negative**, for instance, $I_1 * W_1 = 1 * -1 = -1$. If the value of Sum is $\geq$ T, the neuron fires, otherwise, it does not.

$$Sum = \sum_{i=1}^{N} I_i W_i$$

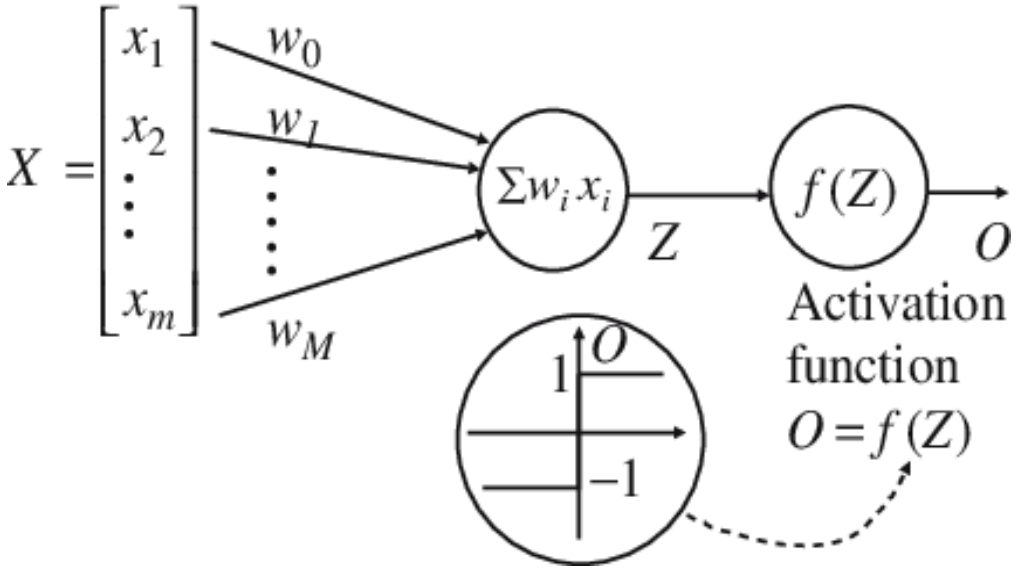
$$y(Sum) = \begin{cases} 1, & \text{if } Sum \geq T \\ 0, & \text{otherwise} \end{cases}$$



This is known as a step-function, where the y-axis encodes the activation-state of the neuron, and the x-axis encodes the output of the weighted sum of inputs.

History of Deep Learning: 1943: Warren McCulloch and Walter Pitts <Threshold Logic Unit>

In the McCulloch-Pitts neuron model, the weights are typically set manually by the designer or researcher. The weights represent the strength of the connections between the inputs and the neuron. Each input is multiplied by its corresponding weight, and the sum of these weighted inputs is compared to a threshold. Setting the weights involves some trial and error or knowledge of the problem domain. The choice of weights determines the neuron's sensitivity to each input feature. Adjusting the weights allows you to influence how much influence each input has on the neuron's decision.



History of Deep Learning: 1957: Frank Rosenblatt's Perceptron

Frank Rosenblatt's perceptron is a type of artificial neural network and a fundamental concept in the history of artificial intelligence and machine learning. It was introduced by Frank Rosenblatt in 1957. The perceptron is a simple algorithm designed for binary classification problems.

Who is Frank Rosenblatt? Frank Rosenblatt (July 11, 1928 – July 11, 1971) was an American psychologist notable in the field of artificial intelligence. He is sometimes called the father of deep learning for his pioneering work on neural networks.

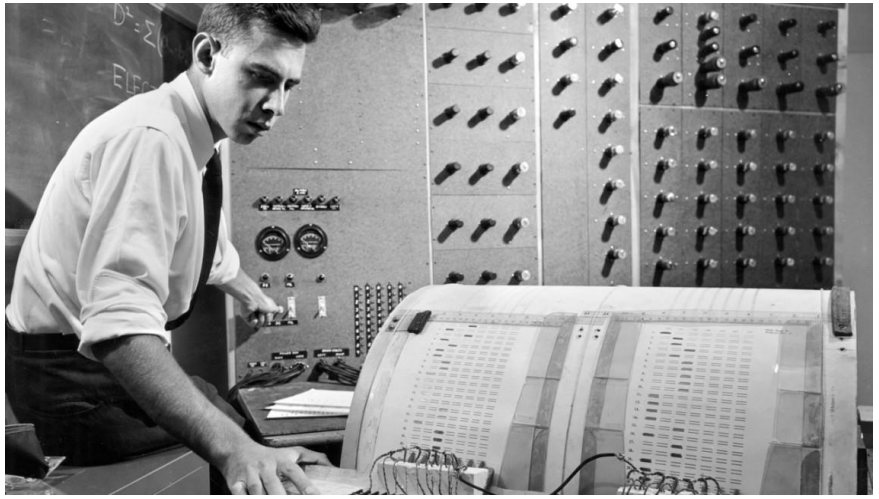


Fig. Frank Rosenblatt '50, Ph.D. '56, works on the "perceptron" – what he described as the first machine "capable of having an original idea."
Source: News.cornell.edu

History of Deep Learning: 1957: Frank Rosenblatt's Perceptron

Here are the basic components and working principles of the perceptron:

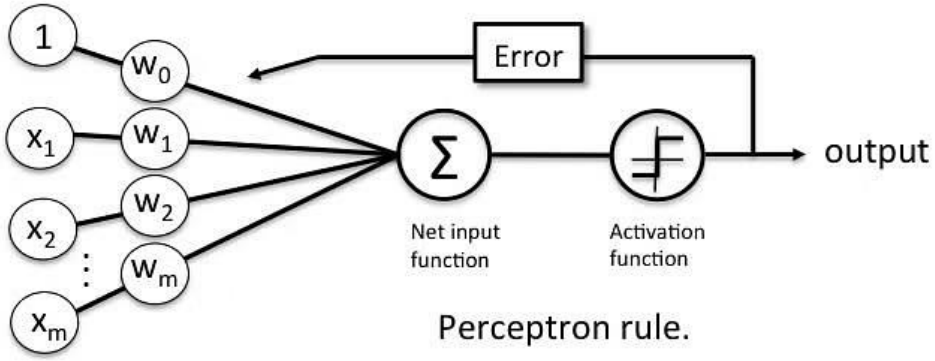
Inputs: The perceptron takes multiple binary inputs (0 or 1). Each input is multiplied by a corresponding weight.

Weights: Each input is associated with a weight, which is a parameter that the algorithm learns during training. The weights determine the strength of the influence of the corresponding input.

Summation: The weighted inputs are summed up, and a bias term is added. The bias is another parameter that the perceptron learns during training.

Activation Function: The summed value is then passed through an activation function. The traditional perceptron uses a step function as its activation function. If the result is above a certain threshold, the perceptron outputs 1; otherwise, it outputs 0.

Learning: The perceptron is trained using a supervised learning approach. During training, it adjusts its weights and bias based on the error in its predictions compared to the true output. This process is often referred to as the perceptron learning rule or the delta rule.



Perceptron rule.

History of Deep Learning: 1957: Frank Rosenblatt's Perceptron

The perceptron learning rule, also known as the perceptron update rule or perceptron training rule, is used to adjust the weights and bias of a perceptron during the training process. The goal is to minimize the error in the perceptron's predictions. The formula for updating the weights and bias for a single training example is as follows:

For each weight w_i and the bias b, the update rule is:

$w_i \leftarrow w_i + \text{learning rate} \times (\text{target} - \text{prediction}) \times \text{input}_i$

Here:

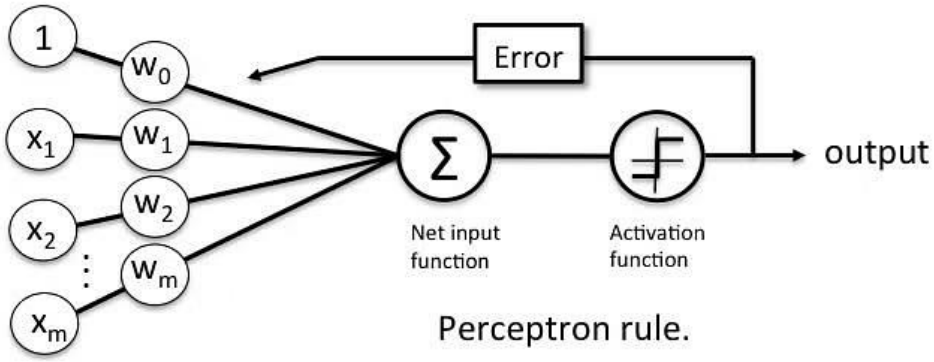
w_i is the weight associated with the i-th input feature.

b is the bias.

The learning rate is a hyperparameter that determines the step size of the update.

The target is the true class label for the training example.

The prediction is the output of the perceptron for the given input.



Perceptron rule.

McCulloch-Pitts Artificial Neuron Model and Rosenblatt's Perceptron <DEMO>

Demo: [Session 5 – Introduction to Deep Learning]

[Perceptron Research from the 50's & 60's, clip]

[The Man who forever changed Artificial Intelligence]

Rosenblatt's Perceptron: The New York Times Reported;

WASHINGTON, July 7, 1958 (UPI) -- "**The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.**"

Link: nytimes.com

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

1958 New York Times...

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

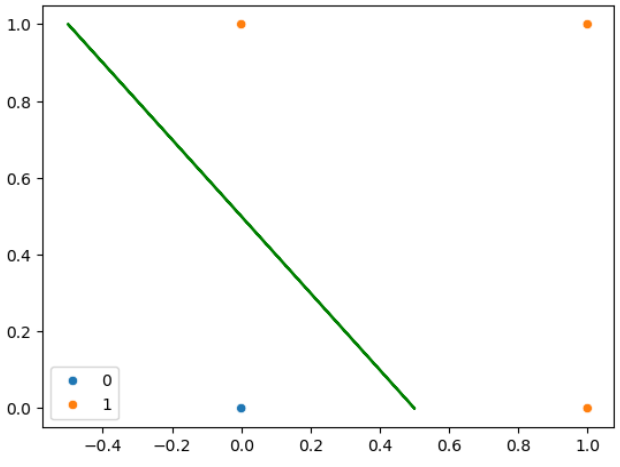
Rosenblatt's Perceptron: XOR Problem

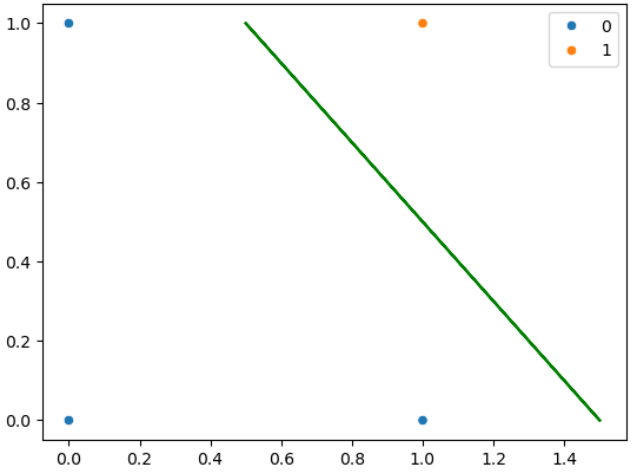
The XOR problem and its implications for perceptrons were first highlighted by Marvin Minsky and Seymour Papert in their book "Perceptrons," published in 1969. Minsky and Papert demonstrated that a single-layer perceptron (with a linear activation function) was unable to learn and represent certain types of logical functions, specifically those that were not linearly separable. The XOR function is a classic example of a logical operation that cannot be represented by a single-layer perceptron.

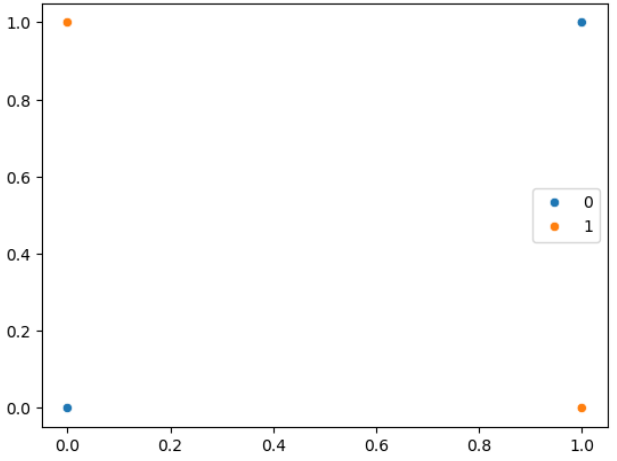
Fig. Marvin Minsky (left) and Seymour Papert (right) in 1971.
Source: npr.org

Rosenblatt's Perceptron: XOR Problem

The XOR (exclusive OR) function takes two binary inputs and outputs 1 if the inputs are different and 0 if they are the same. The problem with representing XOR using a single-layer perceptron arises because the decision boundary for XOR is not a straight line; it requires a non-linear decision boundary.



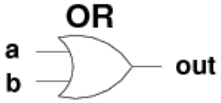




AND

a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

OR

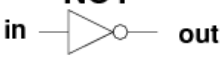


a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

XOR

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

NOT



in	out
0	1
1	0

Rosenblatt's Perceptron: XOR Problem

Minsky and Papert's work was influential because it highlighted the limitations of the perceptron model and led to a period of reduced interest (**dark age of AI** also called **AI winter**) in neural networks during the 1970s and 1980s. It wasn't until the development of multi-layer perceptrons and the backpropagation algorithm in the 1980s that neural networks regained popularity and were shown to be capable of learning more complex, non-linear patterns, including XOR.

Fig. Marvin Minsky (left) and Seymour Papert (right) in 1971.
Source: npr.org

Rosenblatt's Perceptron: XOR Problem
<What is The Solution?>

Well, the solution is to use multi-layer perceptron instead of a single perceptron

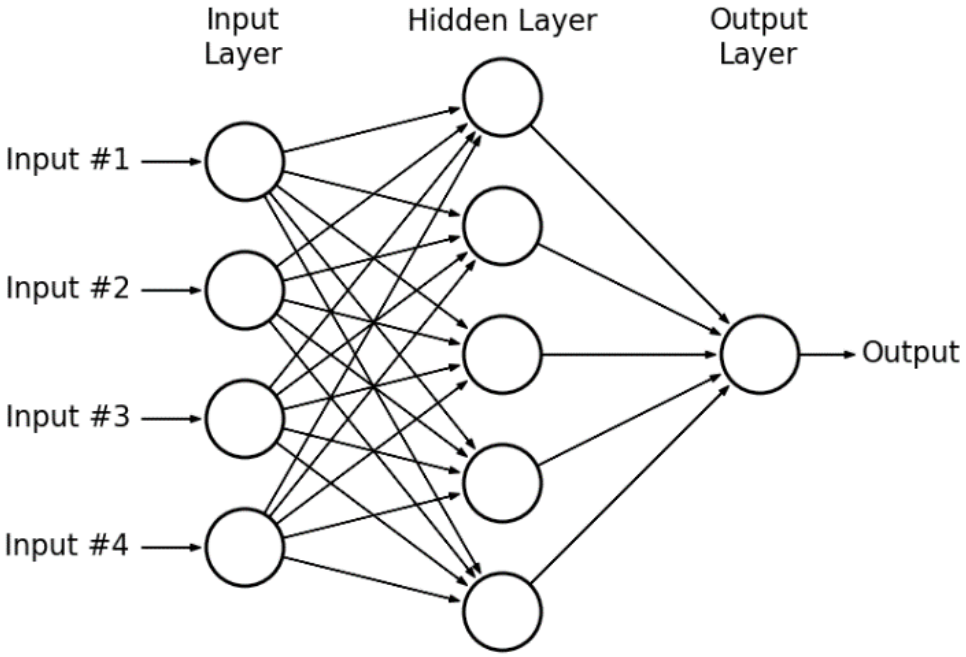
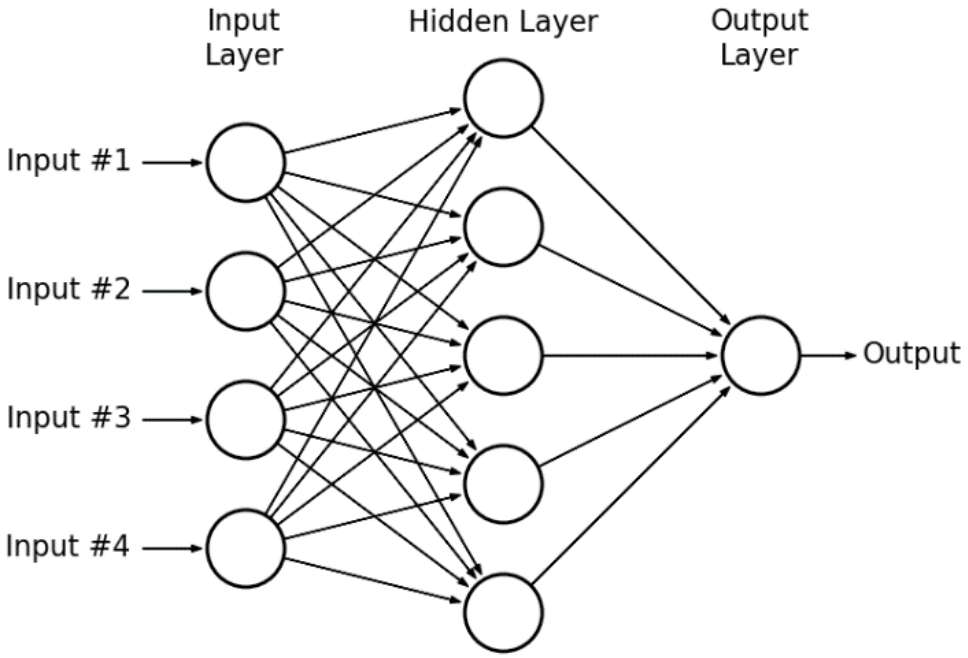


Fig. Marvin Minsky (left) and Seymour Papert (right) in 1971.
Source: npr.org

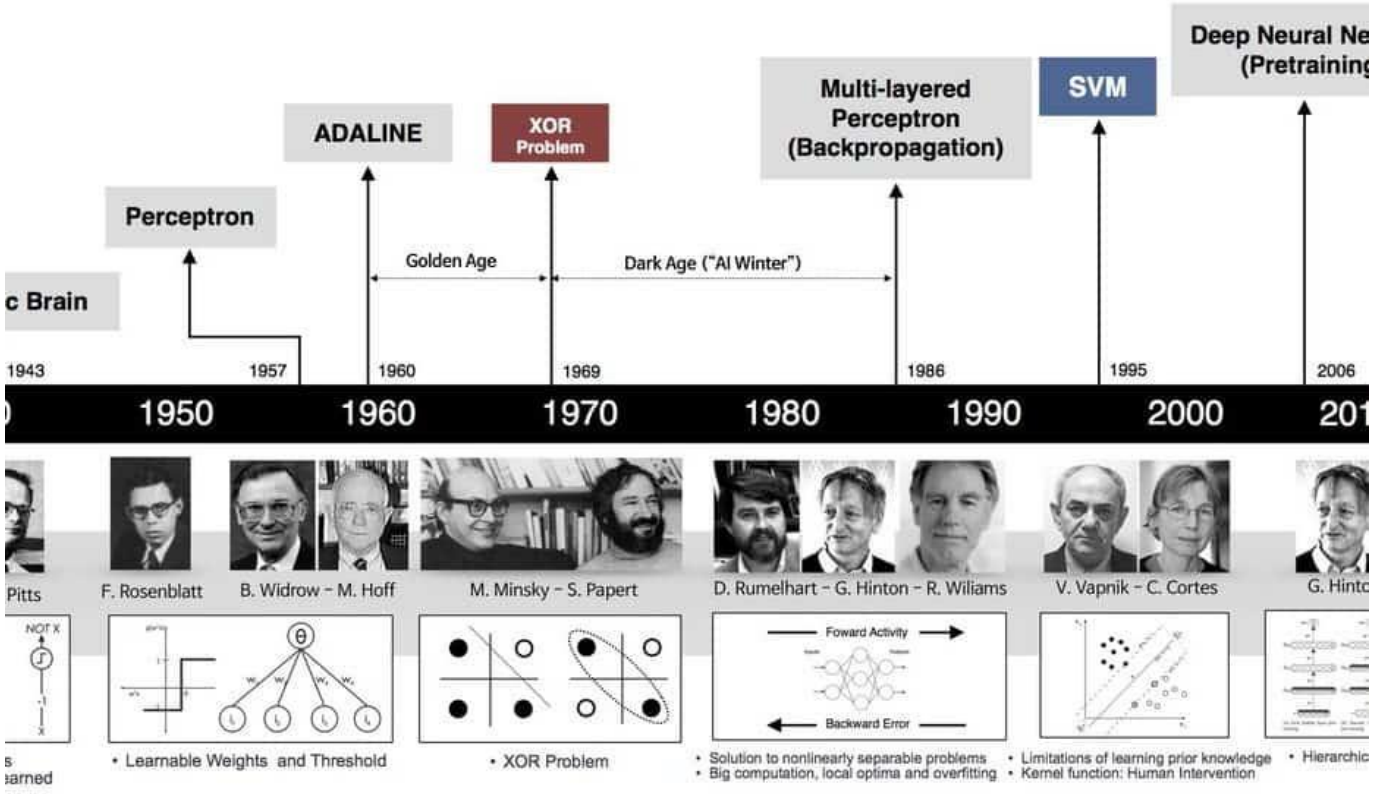
Multi-Layer Perceptron (MLP): Another Challenge!

At that time, there was no training algorithm
that can be used for such architecture!



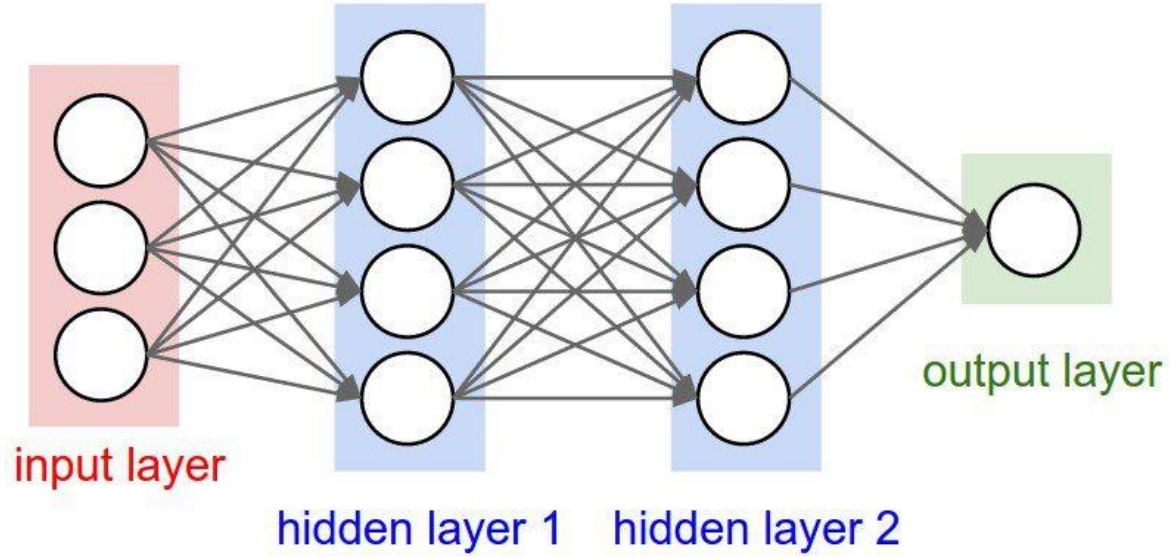
Multi-Layer Perceptron (MLP): Problem Solved and Beginning of New Era!

It took almost 17 years (1969 to 1986) until a training algorithm was re-discovered, known as back-propagation.
Paper Link: [Learning representations by back-propagating errors]



Multi-Layer Perceptron (MLP)

Like in the human brain, the power and robustness of the biological neurons is when they are fully connected to each other, it starts to become more and more complex by adding/stacking more and more hidden layers, at this level we are talking about **Multi-layer perceptron.** **(DEEP LEARNING)**



Multi-Layer Perceptron (MLP): <DEMO>

Demo: Session 5 - Introduction To Deep Learning

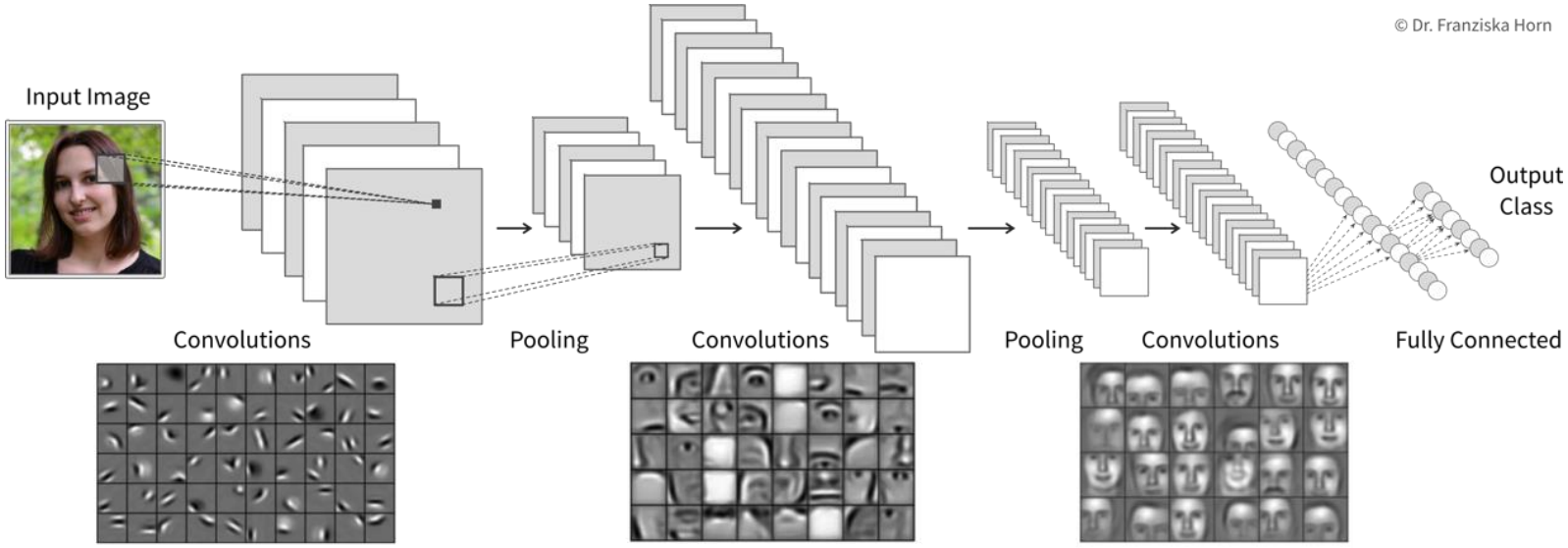
Advanced Neural Network Architectures: Convolutional Neural Network (CNN)

Manual feature engineering for computer vision tasks is incredibly difficult. While humans recognize a multitude of objects in images without effort, it is hard to describe why we can identify what we see, e.g., which features allow us to distinguish a cat from a small dog. Deep learning had its first breakthrough success in this field, because neural networks, in particular CNNs, manage to learn meaningful feature representations of visual information through a hierarchy of layers.

Source: istockphoto.com

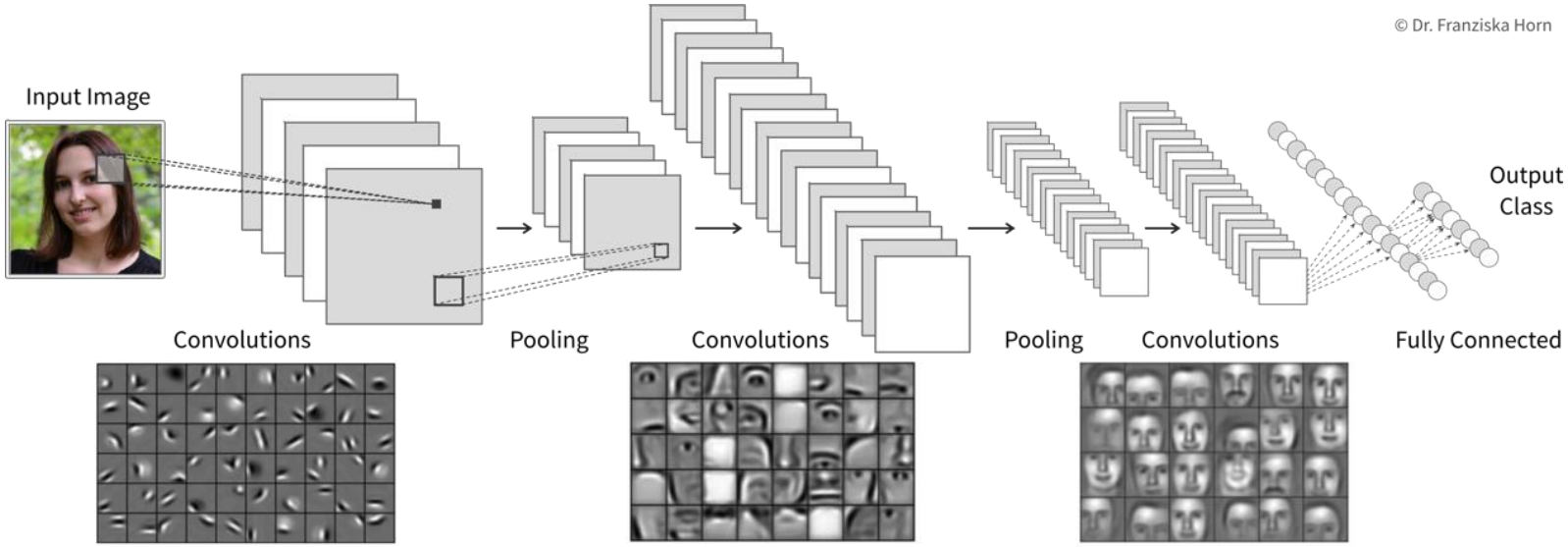
Advanced Neural Network Architectures: Convolutional Neural Network (CNN)

Convolutional neural networks are very well suited for processing visual information, because they can operate on the 2D images directly and do not need the input to be flattened into a vector. Furthermore, they utilize the fact that images are composed of a lot of local information (e.g., eyes, nose, and mouth are all localized components of a face).

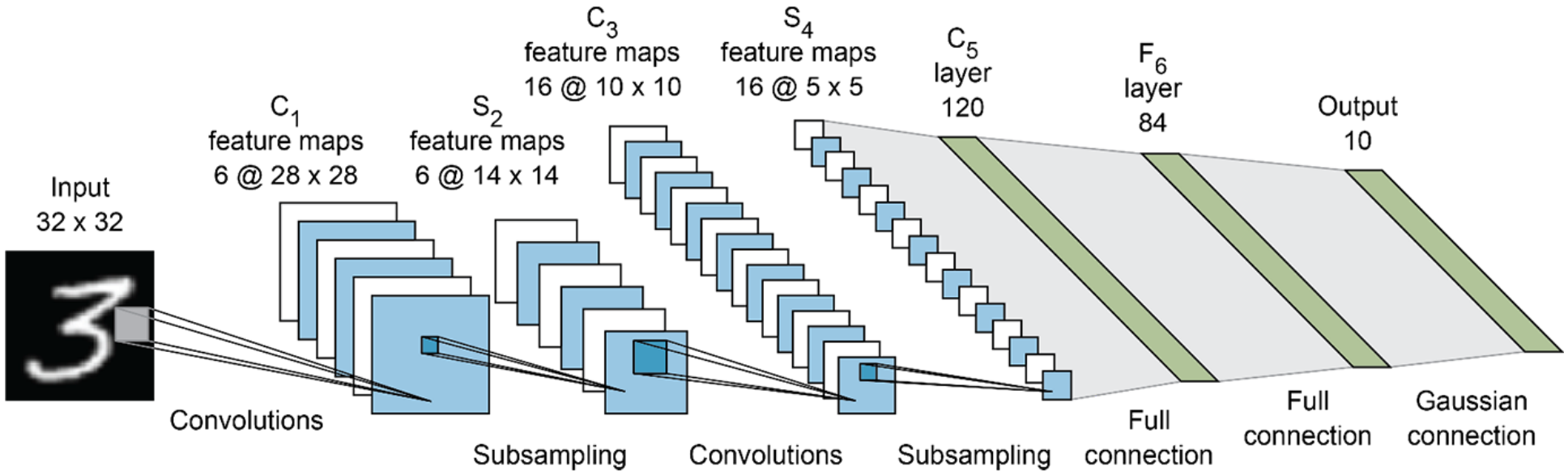


Advanced Neural Network Architectures: Convolutional Neural Network (CNN)

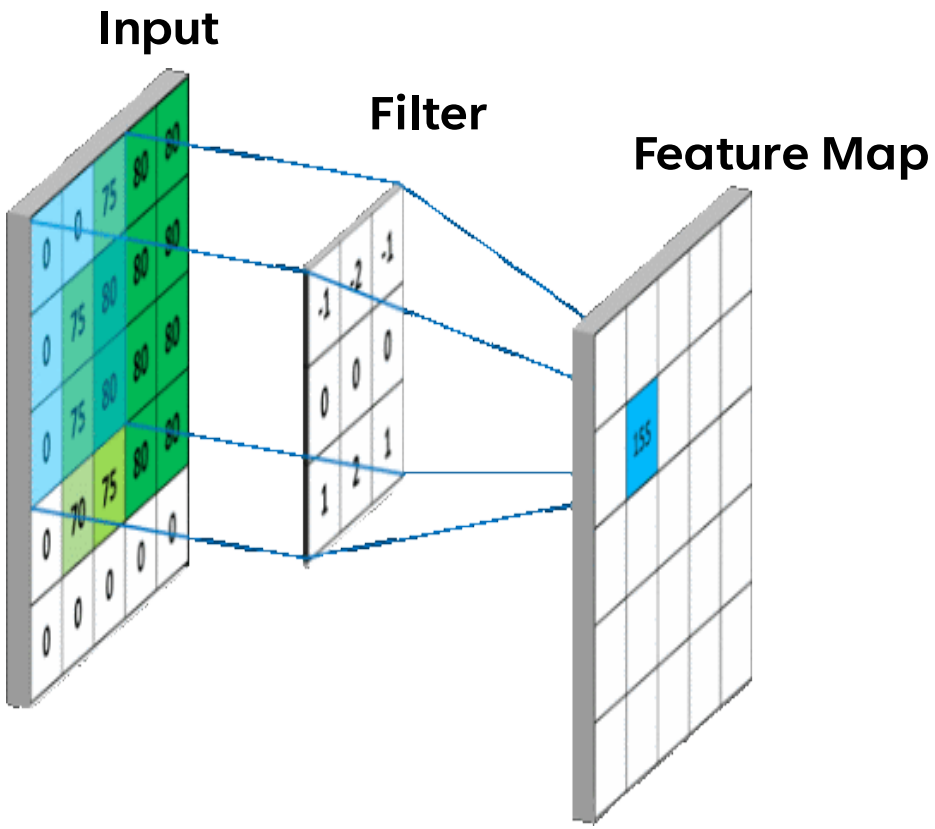
Compared to the dense / fully-connected layers in FFNNs, which consist of one huge matrix mapping from one layer to the next, the filter patches used in convolutional layers are very small, i.e., there are less parameters that need to be learned. Furthermore, the fact that the filters are applied at every position in the image has a regularizing effect, since the filters need to be general enough capture relevant information in multiple areas of the images.



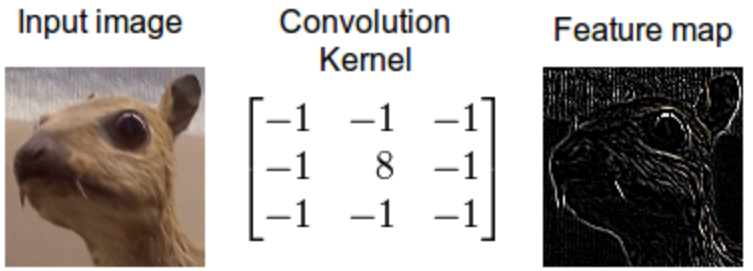
Convolutional Neural Networks (CNNs)



Convolutional Neural Networks (CNNs)



- The term convolution refers to **the mathematical combination of two functions to produce a third function**. It merges two sets of information. In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel (these terms are used interchangeably) to then produce a feature map.
- It is a very important technique to find patterns in images and image processing



Convolutional Neural Networks (CNNs): Filter

Input

3	5	2	8	1
9	7	5	4	3
2	0	6	1	6
6	3	7	9	2
1	4	9	5	1

1	0	0
1	1	0
0	0	1

Filter/Kernel/Feature detector

3×1	5×0	2×0	8	1
9×1	7×1	5×0	4	3
2×0	0×0	6×1	1	6
6	3	7	9	2
1	4	9	5	1

3+0+0+9+7+0+0+0+6

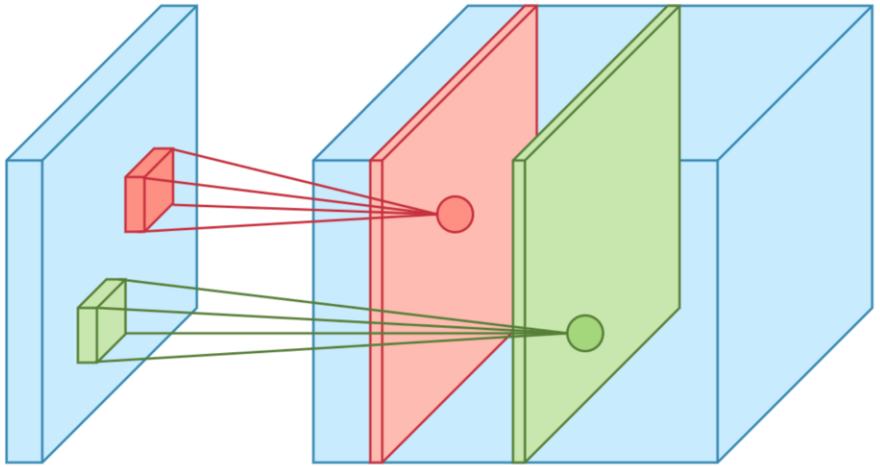
25	18	17
18	22	14
20	15	23

Result: Feature map

Note: Feature map **output size** = [**size** of the **input** - **size** of the kernel/filter] + 1

Convolutional Neural Networks (CNNs): ConvLayers

Conv Layer

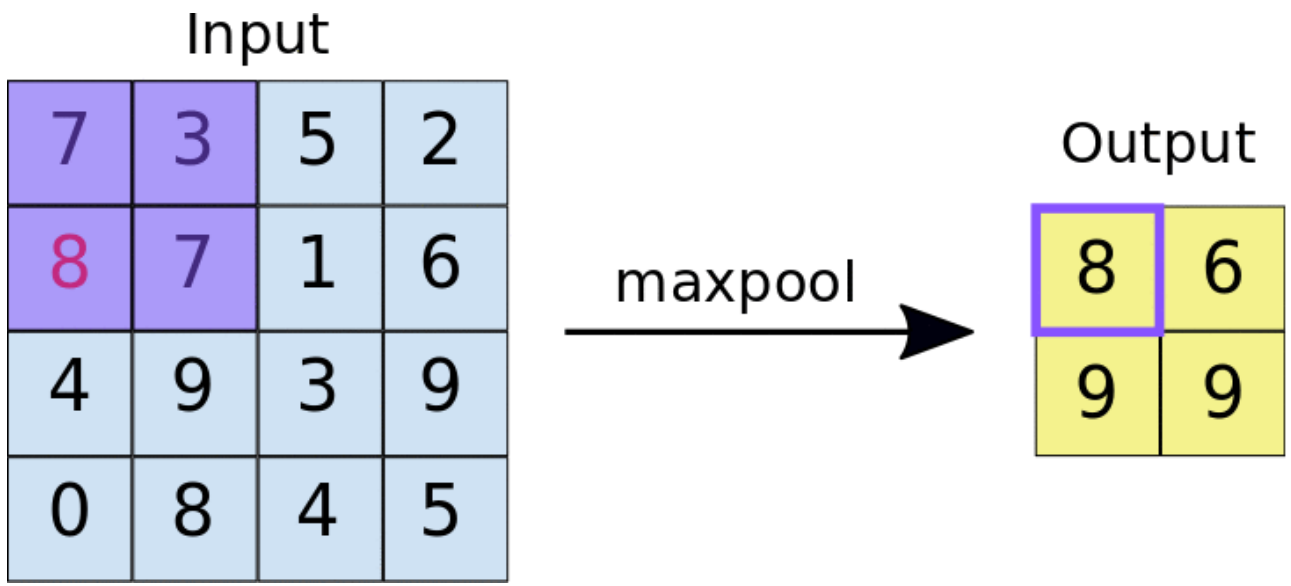


- Convolutional neural networks are a type of neural network in which we are trying to **find** the **filters** and **biases** that **minimize** the **loss function**.
- **Conv Layer** contains different filters/kernels with different lengths and widths.
- The filters are **computed** by the **CNN**, and that is what makes the **CNN** more **powerful**.

Convolutional Neural Networks (CNNs): Pooling

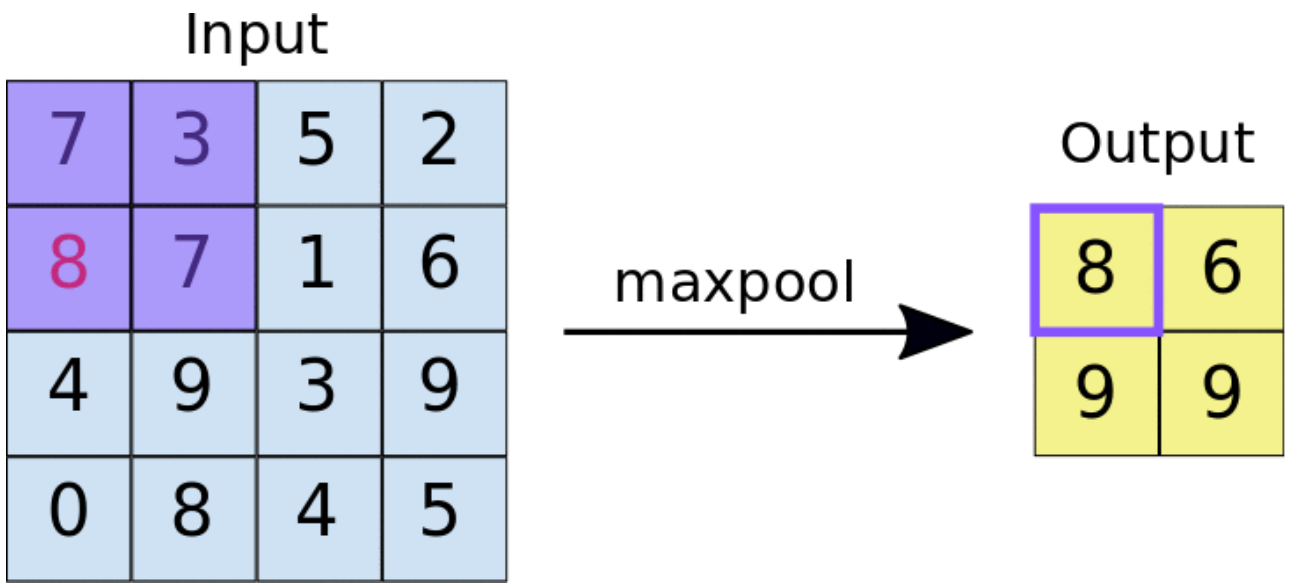
Some used techniques in **ConvNets**: **Pooling**

- **Pooling** is required to down sample the detection of features in feature maps.
- **Pooling layers** provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map.
- Two common pooling methods are **average pooling** and **max pooling**



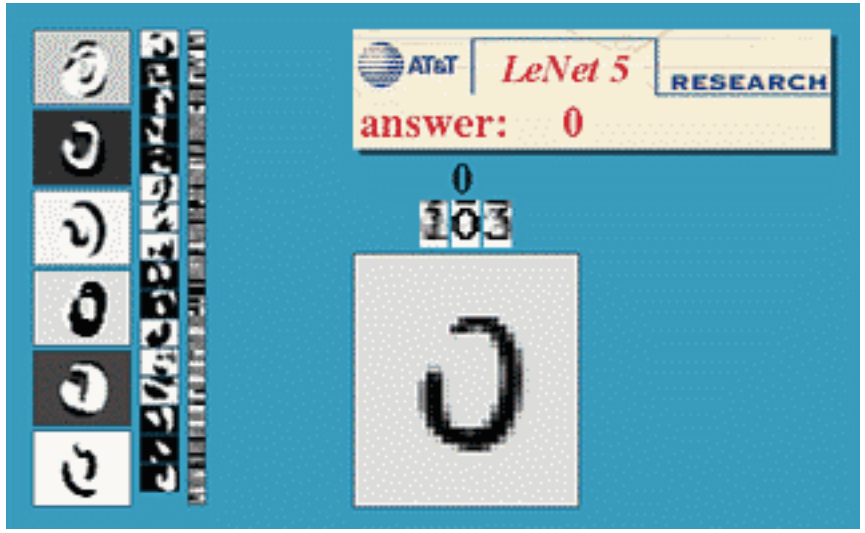
Convolutional Neural Networks (CNNs): MaxPooling

Some used techniques in **ConvNets**: **MaxPooling**



Convolutional Neural Networks (CNNs): LeNet5

LeNet-5 is a convolutional neural network architecture designed for handwritten digit recognition. It was introduced by **Yann LeCun**, Léon Bottou, Yoshua Bengio, and Patrick Haffner in the paper titled "**Gradient-Based Learning Applied to Document Recognition**" published in **1998**.



LeNet5 (1998)

Yann LeCun

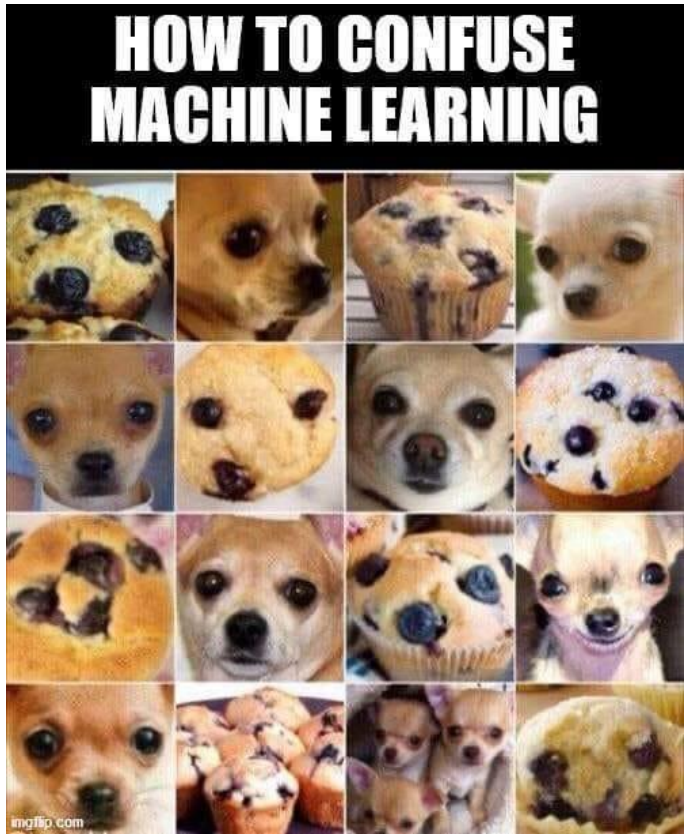
Convolutional Neural Networks (CNNs): LeNet5

LeNet5 Architecture(1998)

1. **Convolutional Layers**: LeNet-5 consists of seven layers, including two convolutional layers. The first convolutional layer applies six filters with a 5x5 receptive field, followed by a 2x2 max-pooling operation.
2. **Activation Functions**: In LeNet-5, the hyperbolic tangent function (tanh) is used as the activation function.
3. **Subsampling Layers**: Following the first convolutional layer, LeNet-5 has a subsampling layer (max-pooling) to reduce the spatial dimensions of the feature maps.
4. **Second Convolutional Layer**: The second convolutional layer applies sixteen 5x5 filters to the subsampled feature maps from the first convolutional layer. This is followed by another 2x2 max-pooling operation.
5. **Fully Connected Layers**: After the convolutional layers, there are three fully connected layers. The first fully connected layer has 120 units, the second has 84 units, and the final output layer has 10 units (corresponding to the 10 possible digits).
6. **Flatten Operation**: Between the convolutional and fully connected layers, there is a flatten operation to convert the 3D volume output from the convolutional layers into a 1D vector.
7. **Softmax Activation**: The output layer uses a softmax activation function, which turns the raw scores (logits) into class probabilities.

Convolutional Neural Networks (CNNs): <DEMO>

Demo: Session 5 - Introduction To Deep Learning



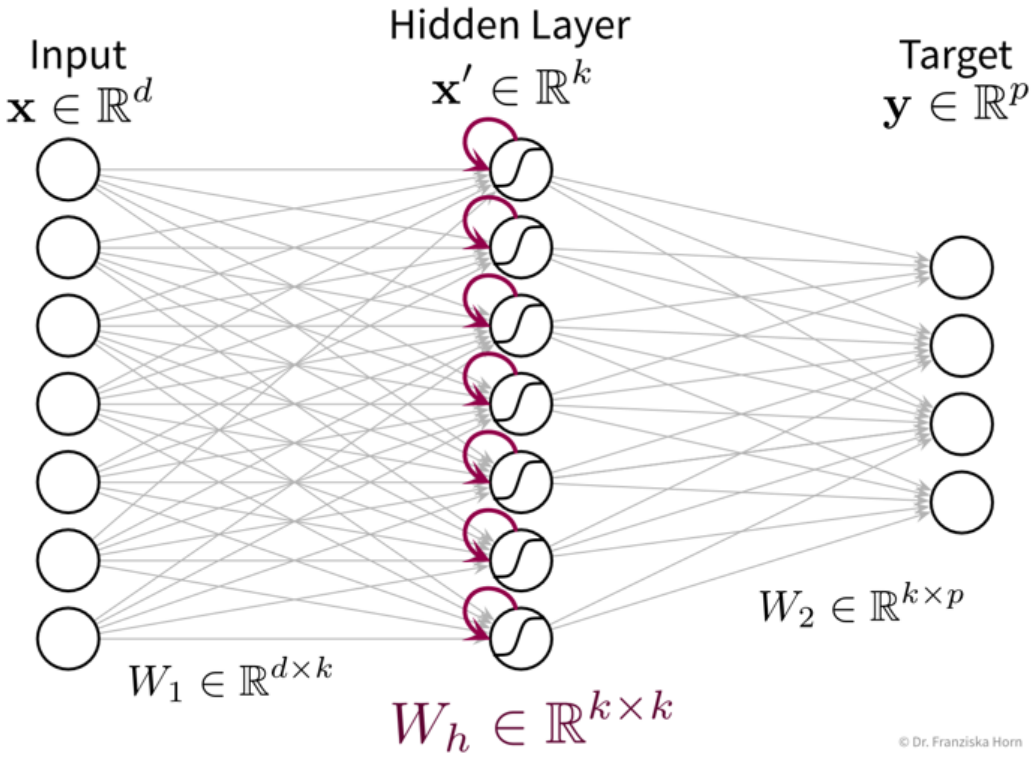
Advanced Neural Network Architectures: Recurrent Neural Network (RNN)

Suited for Sequential Data

Recurrent Neural Networks (RNNs) are a class of neural networks uniquely designed to handle sequential data, making them powerful tools for tasks such as time series analysis, natural language processing, and speech recognition.

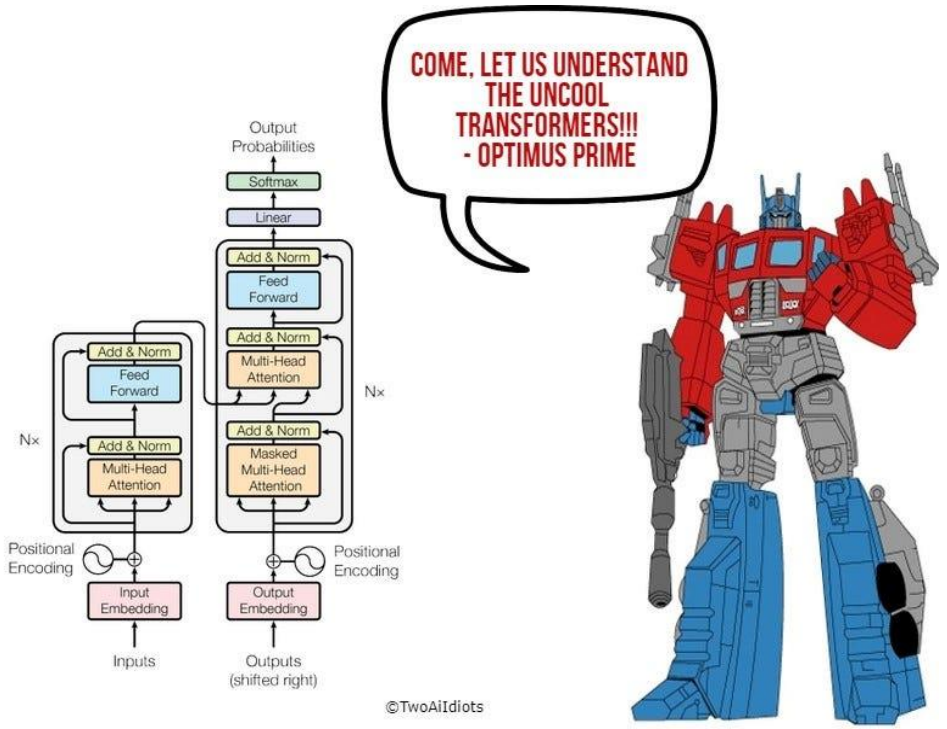
Recurrent neural networks are great for sequential data such as time series data or text (i.e., a sequence of words).

In its simplest form, a RNN is like a FFNN, but with additional recurrent connections W_h in the hidden layer to create a memory of the past:



Advanced Neural Network Architectures: Transformers

Transformers are a type of deep learning model architecture that has gained significant popularity and success in various natural language processing (NLP) tasks. They were introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017. Since then, transformers have become a cornerstone in the field of deep learning and have been extended to various domains beyond NLP.



Advanced Neural Network Architectures: Transformers

Key components of transformers include:

Self-Attention Mechanism:

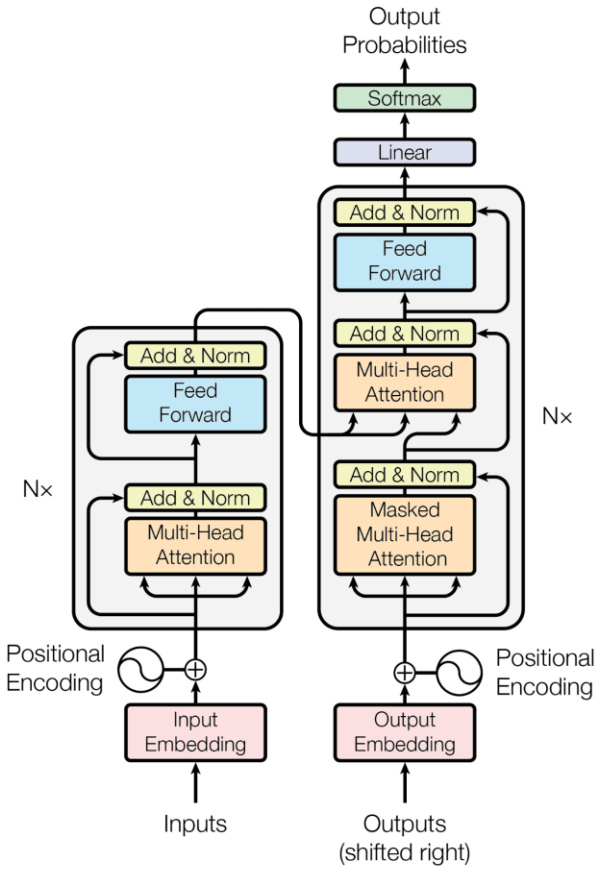
The self-attention mechanism allows the model to weigh the importance of different words in a sequence when processing each word. This helps in capturing long-range dependencies in the data.

Multi-Head Attention:

Transformers use multiple self-attention mechanisms, called attention heads, in parallel. This enables the model to capture different aspects of the relationships between words.

Positional Encoding:

Since transformers don't inherently understand the order of elements in a sequence, positional encodings are added to the input embeddings to provide information about the positions of words in a sentence.



Advanced Neural Network Architectures: Transformers

Key components of transformers include:

Encoder-Decoder Architecture:

Transformers are commonly used in sequence-to-sequence tasks where the input and output sequences can be of different lengths. The model consists of an encoder and a decoder, with each having its own stack of layers.

Feedforward Neural Networks:

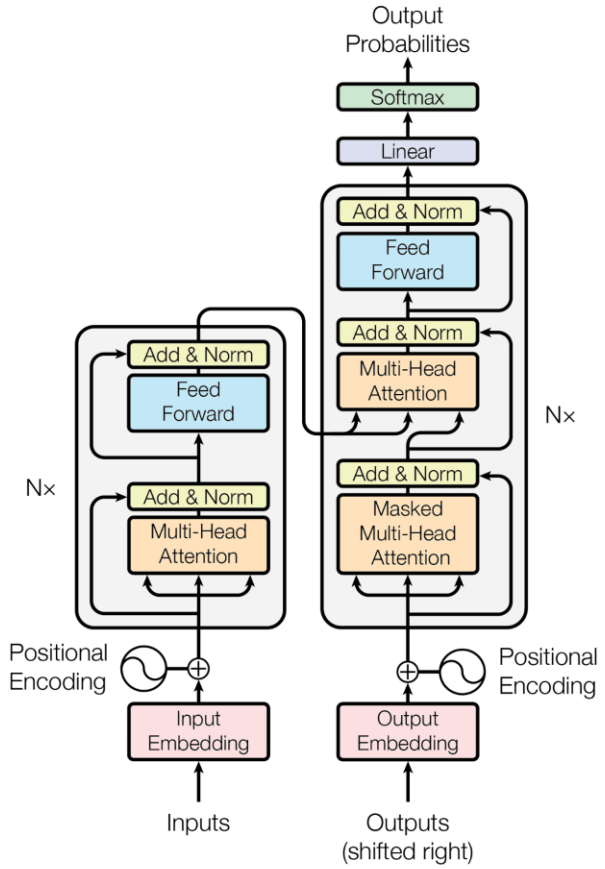
After the self-attention mechanism, there is a feedforward neural network for further processing and non-linearity.

Layer Normalization and Residual Connections:

Each sub-layer (like self-attention or feedforward layer) is followed by layer normalization and connected with a residual connection. This helps in stabilizing the training process.

Attention Masking:

In some applications, attention masking is used to prevent the model from attending to future tokens in the sequence during training.

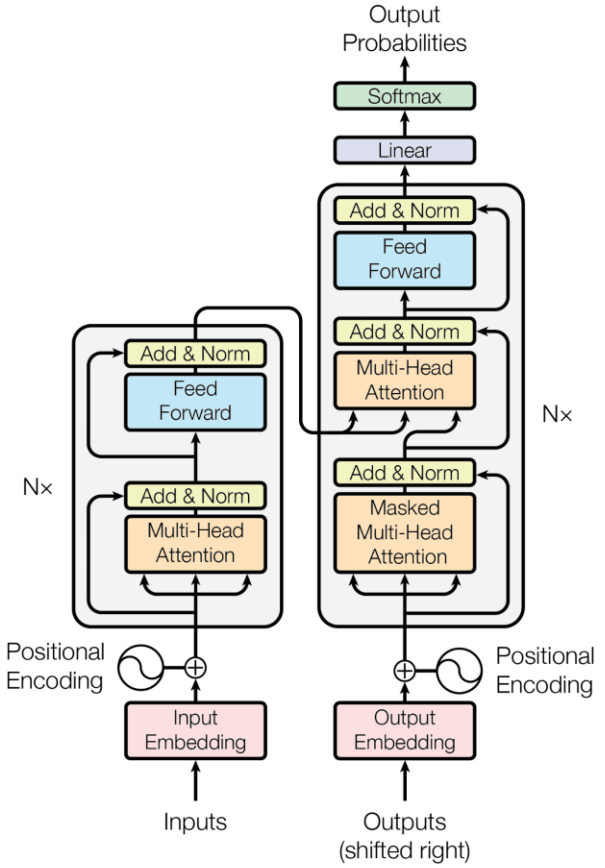


Advanced Neural Network Architectures: Transformers

Applications of transformers extend beyond NLP and include:

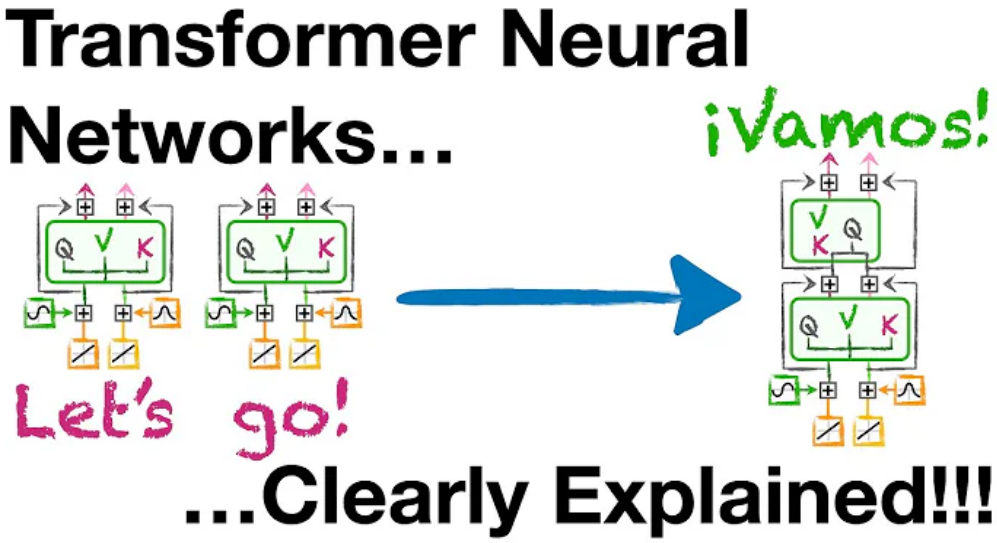
- BERT (Bidirectional Encoder Representations from Transformers): Used for various NLP tasks, such as question answering and sentiment analysis.
- GPT (Generative Pre-trained Transformer): Employed for language modeling and text generation.
- T5 (Text-to-Text Transfer Transformer): Generalized transformer model capable of performing various tasks with a unified text-to-text approach.
- Vision Transformers (ViTs): Applying transformers to computer vision tasks, replacing the traditional convolutional neural networks (CNNs) in some scenarios.

Transformers have shown state-of-the-art performance in many tasks and have become a fundamental building block for a wide range of deep learning applications. Researchers continue to explore and refine transformer architectures for different domains and tasks.



Advanced Neural Network Architectures: Transformers

[Let's build GPT: from scratch, in code, spelled out.]



[Transformer Neural Networks, ChatGPT's foundation, Clearly Explained!!!]