MASTER CSA : Introduction to Embedded systems

"an overall of what is deep learning"

Geoffrey ROMAN JIMENEZ

IRIT - Université Paul Sabatier

19/03/18 - 20/03/18

What is it?

Deep-learning is a branch of Machine-learning which is a type of Artificial intelligence.

Machine-learning

Contruct algorithm that can performed a task without explicitly programming it to do so.
Learning a **task** to a machine.

Deep-learning

Aim at doing the same... but in a more **deeper** way!

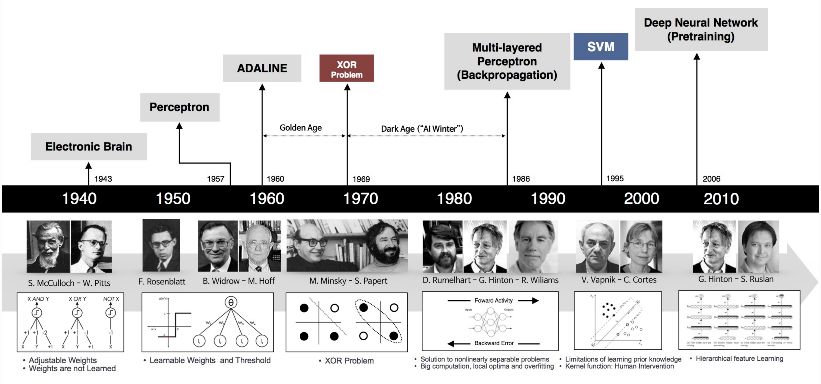
Deep-learning is based on the learning data representation **task**.

What is it?

A task in data science :

- Data Classification
- Data Clustering
- Data Modeling (Regression)
- Data Representation (encoding, compressing, visualization)

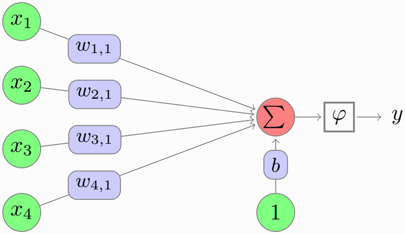
A brief history



- 1943 : Mc Culloch and Pitts $\rightarrow$ 1^{st} generation Perceptron
- 1963 : Hebb's rule $\rightarrow$ cells that fire together, wire together
- 1986 : Multi-layered ~~Perceptron~~ Artifical Neuron
- ~~August 4, 1997 : Skynet's activation (Judgment's Day)~~
- 2006 : Deep neural Network
- 2012 :
- 2018 : Master CSA

Artificial Neural Network

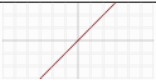
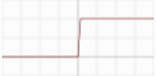
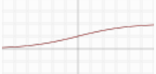
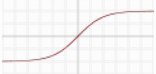
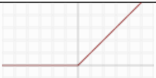
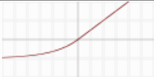
Artificial Neural (AN):



$$y = \mathcal{N}(\mathbf{x}, \mathbf{w}, b) = \varphi\left(\sum_i x_i.w_i + b\right) = \varphi\left(\begin{bmatrix} w_1 & \dots & w_n \end{bmatrix} . \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + b\right)$$

with φ an activation function (e.g. *tanh*, *relu*)

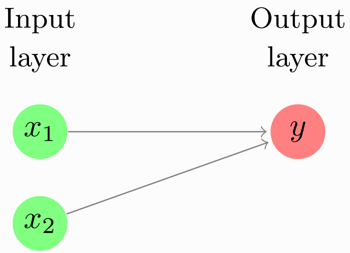
Activation function φ

Linear		$\varphi(x)=x$
Heaviside (Perceptron)		$\varphi(x)=\begin{cases} 0, & \text{if } x < 0. \\ 1, & \text{if } x \geq 0. \end{cases}$
Sigmoid		$\varphi(x)=\frac{1}{1+e^{-x}}$
Tangent Hyperbolic		$\varphi(x)= tanh(x)= \frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$
REctify Linear Unit (ReLU)		$\varphi(x)=max(0,x)$
Exponential Linear Unit (ELU)		$\varphi(x)=\begin{cases} \alpha(e^{x}-1), & \text{if } x < 0. \\ x, & \text{if } x \geq 0. \end{cases}$

Perceptron

Perceptron : an artifical neuron with binary output

Each output -> linear classifier



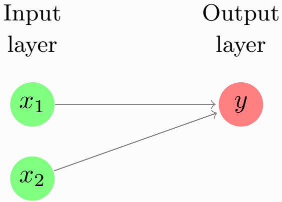
$$y = \varphi\left(\begin{bmatrix} w_1 & w_2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b \right)$$

Note

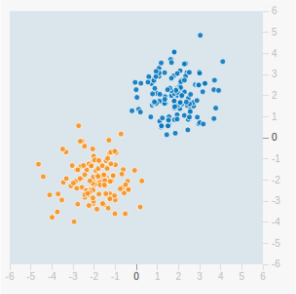
Unlike all layers in a Neural Network, the output layer neurons most commonly do not have an activation function (or you can think of them as having a linear identity activation function).

Perceptron

What Perceptron can do : linear separation

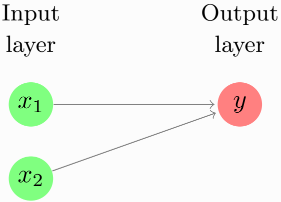


$$y = \varphi\left(\begin{bmatrix} w_1 & w_2 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b \right)$$



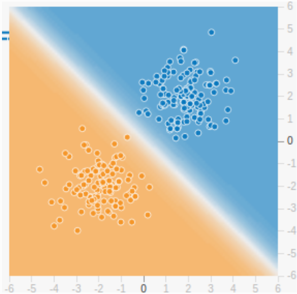
Perceptron

What Perceptron can do : linear separation



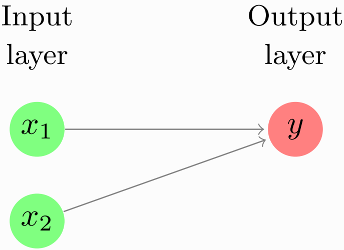
$$y = \varphi\left(\begin{bmatrix} w_1 & w_2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b \right)$$

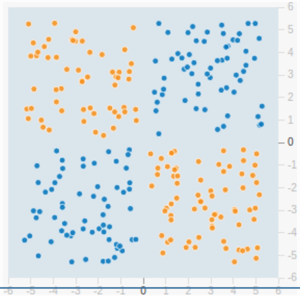
$\rightarrow$



Perceptron

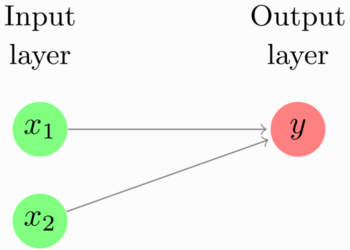
What Perceptron **can't do** : non-linear separation

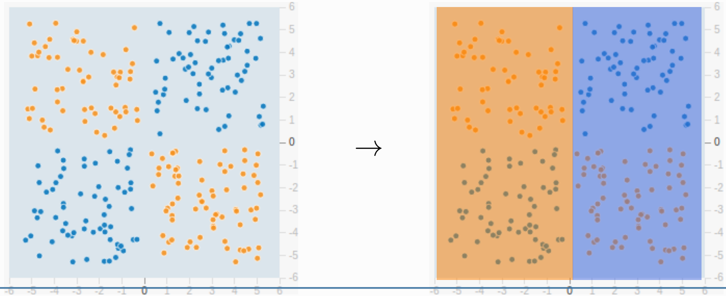




Perceptron

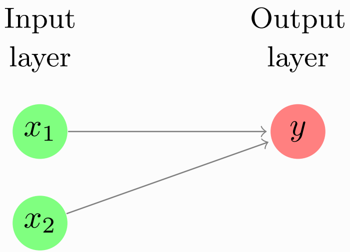
What Perceptron **can't do** : non-linear separation

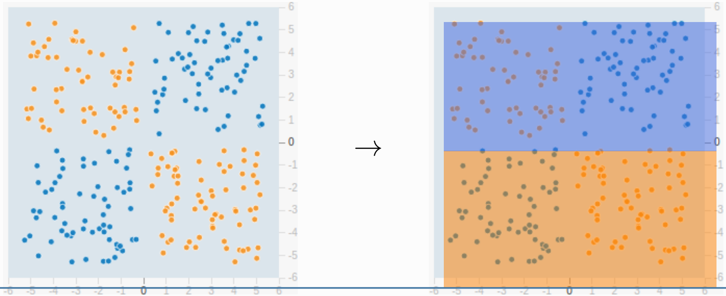




Perceptron

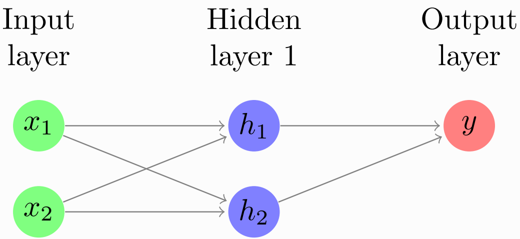
What Perceptron **can't do** : non-linear separation





Neural network

Shallow Neural Network (SNN) : a network with one hidden layer



$$y=\varphi\left(\begin{bmatrix} w_{out_1} & w_{out_2} & w_{out_3} \end{bmatrix} . \varphi\left(\begin{bmatrix} w_{in(1,1)} & w_{in(1,2)} \\ w_{in(2,1)} & w_{in(2,2)} \\ w_{in(3,1)} & w_{in(3,2)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}\right) + b\right)$$

$$\mathbf{y}=\varphi(\mathbf{W}_{out}.\varphi(\mathbf{W}_{in}.\mathbf{x}+\mathbf{b}_{in})+\mathbf{b}_{out})$$

Deep Learning

How to learn the weight? -> Backpropagation Algorithm (Supervised learning)

Requirements

- The inputs $\mathbf{x}$ and their corresponding targets $\mathbf{t}$
- loss function : error measure $\mathcal{E}$ (e.g. Squared Error : $\mathcal{E} = \sum_i (y_i - t_i)^2$)
- Random initialization of the weights $\mathcal{W}$ and $\mathbf{B}$

Backpropagation Algorithm

Repeat until convergence (e.g. $\mathcal{E} < \delta$)

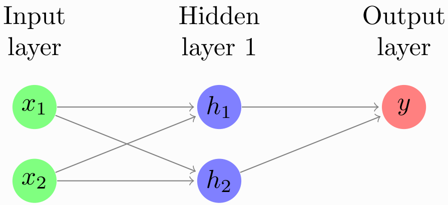
1. Compute the output of the network $\mathbf{y}$ w.r.t $\mathcal{W}$
2. Compute the error $\mathcal{E}$ w.r.t $\mathbf{y}$ and $\mathbf{t}$
3. **Update the weights $\mathcal{W}$ and $\mathbf{B}$**

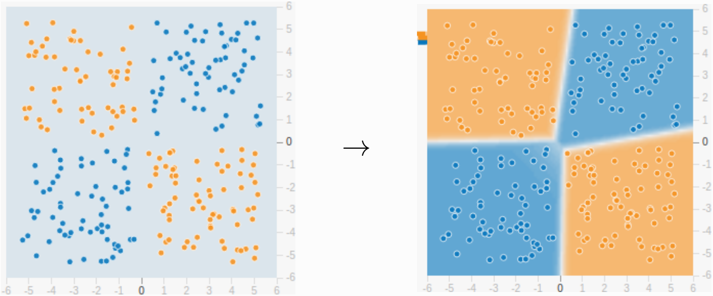
Neural network

Let's try

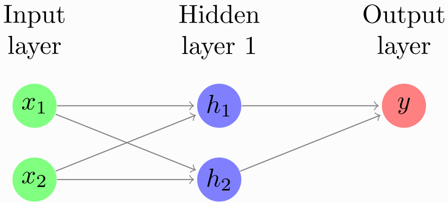
http://playground.tensorflow.org

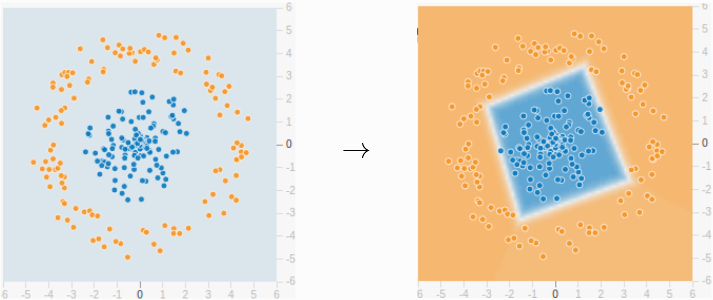
Shallow Neural network



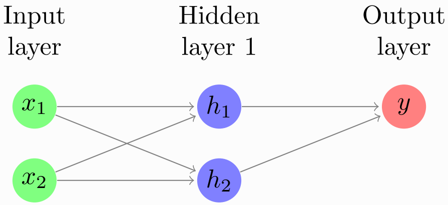


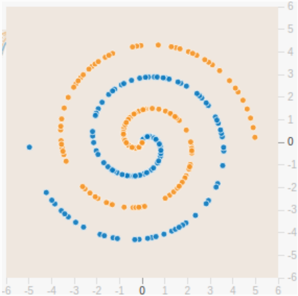
Shallow Neural network





Shallow Neural network



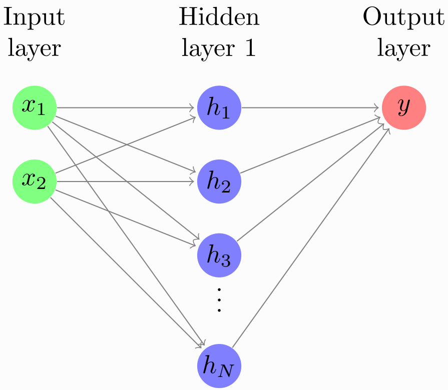


$\rightarrow$?

Shallow Neural network

Cybenko 1989: Universal approximation theorem

A feed-forward network with a single hidden layer (Shallow Neural network or SNN) containing a finite number of neurons N, can approximate any continuous functions on compact subsets of $\mathbb{R}^n$



Neural network

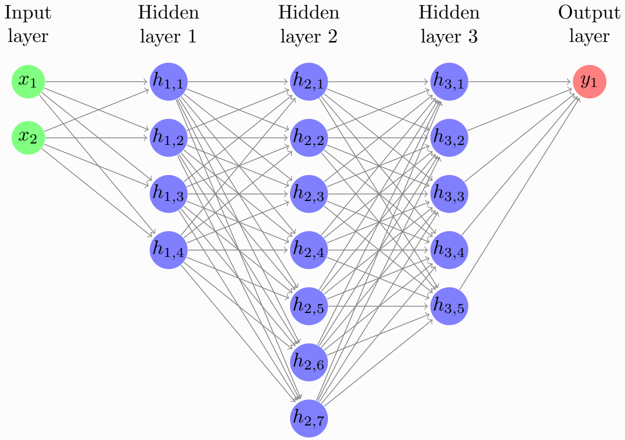
- Historically : Perceptron (binary output unit)
- Network without hidden unit are very limited in the input-output mappings they can perform
- Shallow neural network can approximate any continuous mapping

limits

- We can perform any mapping with SNN...
  - BUT how many AN are needed in the hidden layer ?
  - BUT hard to learn large SNN

Deep Neural network

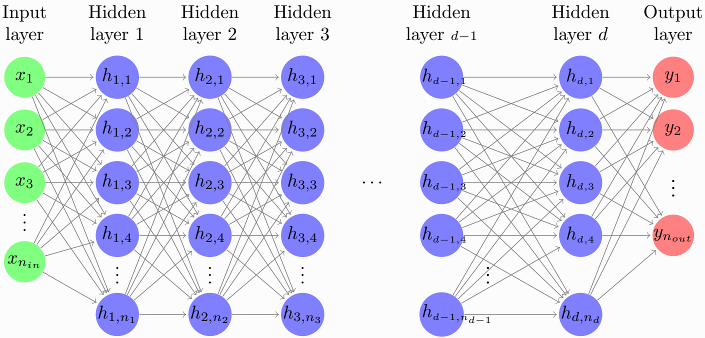
Deep neural network : more than One hidden layer Can help to solve hard problem by performing deeper mix of information



$$y(\mathbf{x},\mathcal{W},\mathbf{B})=.\varphi\left(\mathbf{W_4}.\varphi\left(\mathbf{W_3}.\varphi\left(\mathbf{W_2}.\varphi(\mathbf{W_1}.\mathbf{x}+\mathbf{b_1})+\mathbf{b_2}\right)+\mathbf{b_3}\right)+b_4\right)$$

Deep Neural network

in a more general way ...



$$\mathbf{y}(\mathbf{x},\mathcal{W},\mathbf{B})=\varphi\left(\mathbf{W}_d.\varphi\left(\mathbf{W}_{d-1}\ \dots\ \varphi\left(\mathbf{W}_2.\varphi(\mathbf{W}_1.\mathbf{x}+\mathbf{b}_1)+\mathbf{b}_2\right)\ \dots\ +\mathbf{b}_{d-1}\right)+\mathbf{b}_d\varphi\right)$$

Deep Neural Network

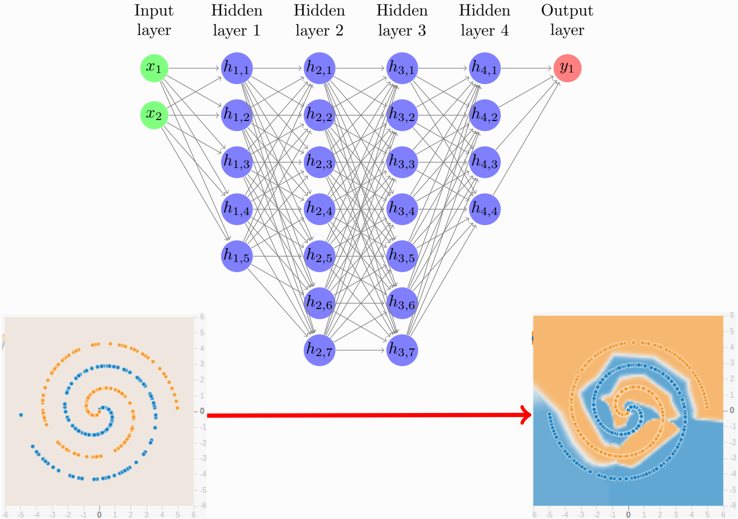
$$\mathbf{y}(\mathbf{x},\mathcal{W},\mathbf{B})=\varphi\left(\mathbf{W}_d.\varphi\left(\mathbf{W}_{d-1}\ \dots\ \varphi\left(\mathbf{W}_2.\varphi(\mathbf{W}_1.\mathbf{x}+\mathbf{b}_1)+\mathbf{b}_2\right)\ \dots\ +\mathbf{b}_{d-1}\right)+\mathbf{b}_d\varphi\right)$$

with

$$\mathbf{x}=\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_{n_{in}} \end{pmatrix} \quad \mathbf{y}=\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_{n_{out}} \end{pmatrix} \quad \mathcal{W}=\{\mathbf{W}_1,\mathbf{W}_2,\dots,\mathbf{W}_{d-1},\mathbf{W}_d\} \quad \mathbf{B}=\{\mathbf{b}_1,\mathbf{b}_2,\dots,\mathbf{b}_{d-1},\mathbf{b}_d\}$$

$$\mathbf{W}_k=\begin{pmatrix} w_{k(1,1)} & w_{k(1,2)} & \cdots & w_{k(1,n_{k-1})} \\ w_{k(2,1)} & w_{k(2,2)} & \cdots & w_{k(2,n_{k-1})} \\ \vdots & \vdots & \vdots & \vdots \\ w_{k(n_k,1)} & w_{k(n_k,2)} & \cdots & w_{k(n_k,n_{k-1})} \end{pmatrix} \qquad \mathbf{b}_k=\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{k(n_k-1)} \\ b_{k(n_k)} \end{pmatrix}$$

Deep Neural Network



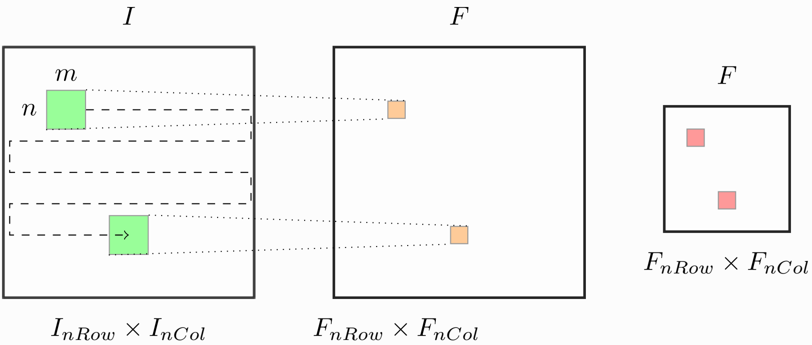
Deep Learning

Convolutional neural network

Deep Learning

Convolutional Neural Network

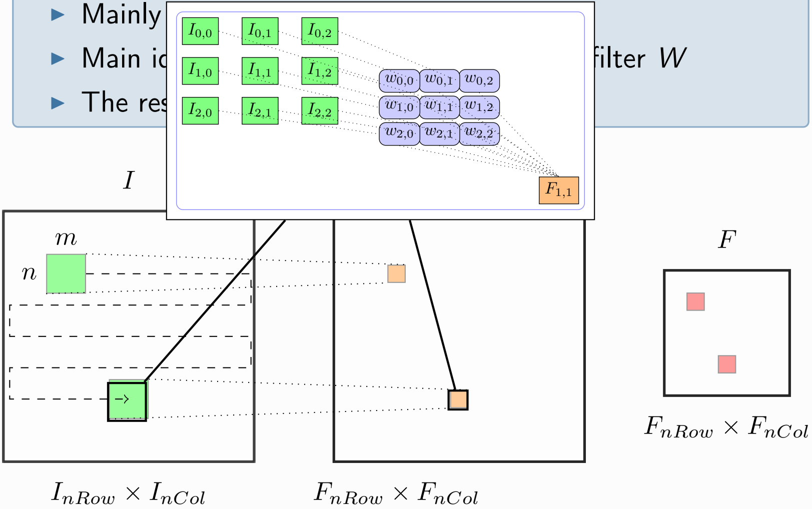
- Mainly performed on images (I)
- Main idea : convolution of the input by a filter W
- The result is a features map F



Deep Learning

Convolutional Neural Network

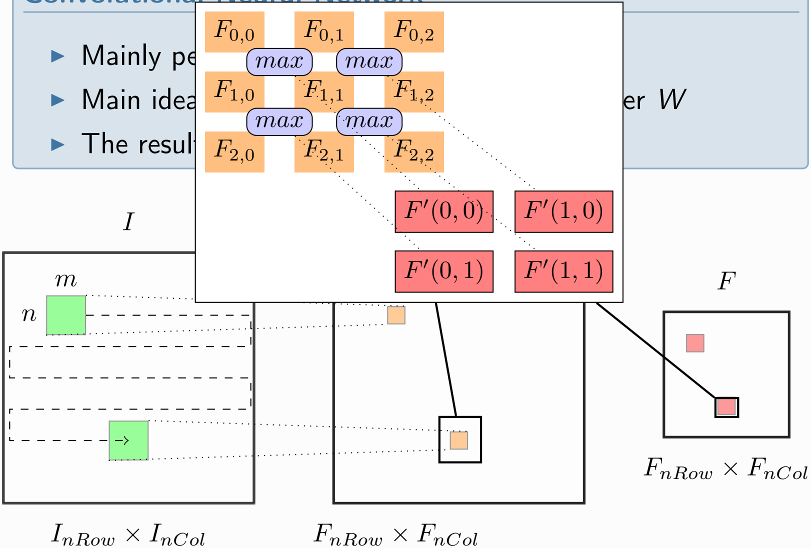
- Mainly
- Main id … filter W
- The res



Deep Learning

Convolutional Neural Network

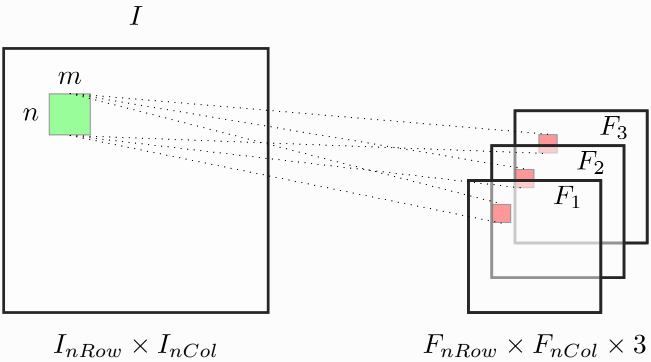
- Mainly pe
- Main idea er W
- The result



Deep Learning

Convolutional Neural Network

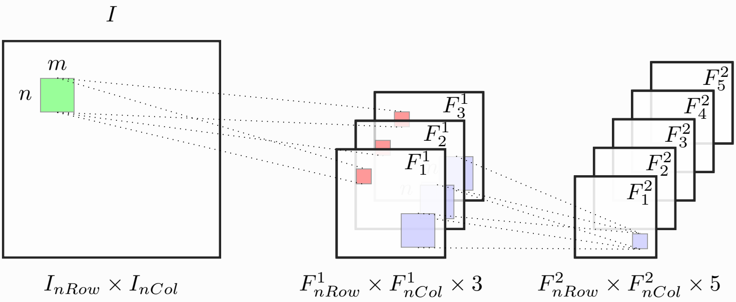
- Several convolution of the input by several filter $\{W_1, W_2, W_3\}$
- The result is **several** features map $\{F_1, F_2, F_3\}$



Deep Learning

Convolutional Neural Network

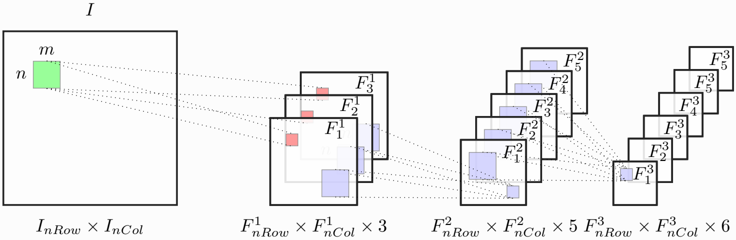
- Deep CNN : Extract features maps on features map
- The result is **several** features map $\{F_1^2, F_2^2, F_3^2, F_4^2, ...,\}$



Deep Learning

Convolutional Neural Network

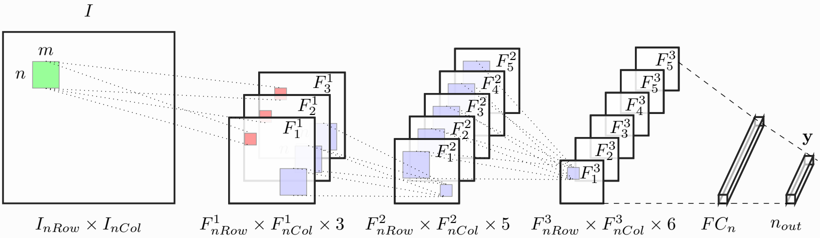
- Deep CNN : Extract features from features maps from features map ...
- The result is **several** features map $\{F_1^3, F_2^3, F_3^3, F_4^3, ..., \}$



Deep Learning

Convolutional Neural Network

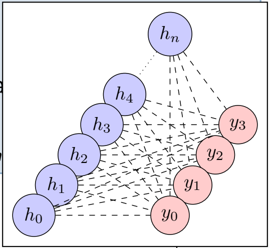
- Deep CNN : connect all of these features to a flatten fully connected layer of neurons
- The result is
  - a vector of neurons $\{h_1, h_2, ..., h_n\}$ (a mix of the features $\{F_1^3, F_2^3, F_3^3, F_4^3, ..., \}$)
  - a vector of outputs $\mathbf{y} = \{y_1, y_2, ..., y_n\}$

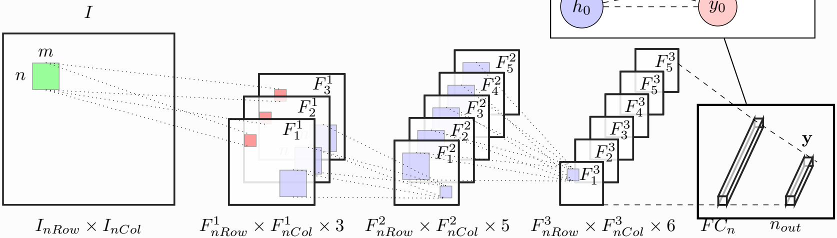


Deep Learning

Convolutional Neural Network

- Deep CNN : connect all of these features to a flatten fully connected layer of neurons
- The result is
  - a vector of neurons $\{h_1, h_2, ..., h_n\}$ (a $\{F^3_1, F^3_2, F^3_3, F^3_4, ..., \}$)
  - a vector of outputs $\mathbf{y} = \{y_1, y_2, ..., y_n$





Deep Learning

example of CNN for digit recognition

http://scs.ryerson.ca/ãharley/vis/conv/flat.html

Convolutional Neural Network

CNN is very usefull to go deeper in the learning of latent space -> better classification performance

1. Extract features
2. Mix features
3. Output classification, Regression, etc.

Deep Neural Network

To resume

Artificial Neuron (AN)

- Formula : $y = \varphi\left(\sum_i x_i.w_i + b\right)$
- Several type of unit : Linear Unit, Perceptron (binary), Sigmoid Unit, Tangent Hyperbolic unit, Rectified Linear Unit

Artificial Neural Network (ANN)

- Shallow Neural Network (SNN)
- Deep neural network (DNN)
  - Convolutional Neural Network (CNN)
  - Recurrent neural network (RNN)