Introducción a las RNA

Redes Neuronales Artificiales

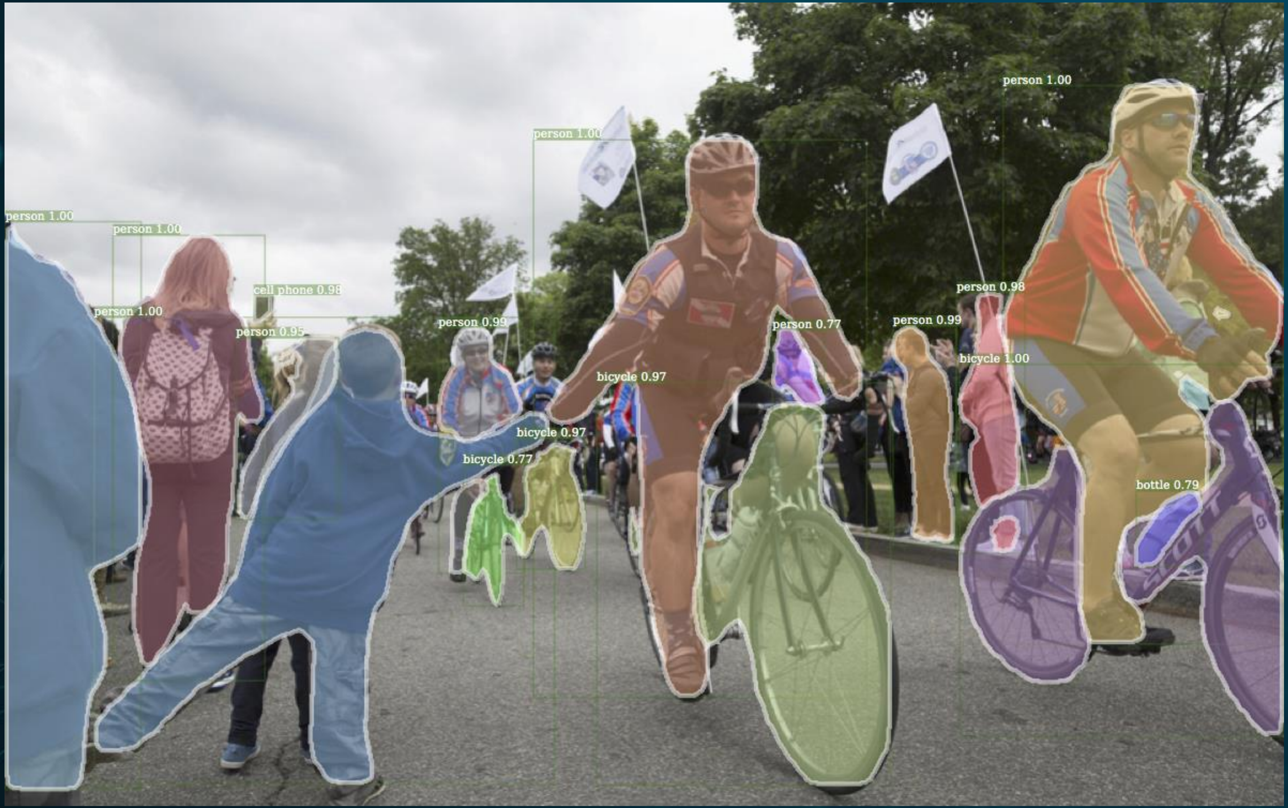
David Barrientos

djbr@ecomp.poli.br

Introducción

01

El inicio de las RNA


person 1.00
person 1.00
cell phone 0.98
person 1.00
person 0.95
person 0.99
person 1.00
bicycle 0.97
bicycle 0.97
bicycle 0.77
person 0.77
person 0.99
person 0.98
bicycle 1.00
person 1.00
bottle 0.79


INPUT
OUTPUT


slice 10
slice 13
slice 16
slice 19
slice 22
slice 25
slice 28
slice 31
slice 34
slice 37
slice 40
slice 43
slice 46
slice 49
slice 52
slice 55
slice 58
slice 61
slice 64
slice 67
slice 70
slice 73
slice 76
slice 79
slice 82
slice 85
slice 88
slice 91
slice 94
slice 97
slice 100
slice 103
slice 106
slice 109
slice 112
slice 115

Primeros Pasos

1943
Primera versión de una neurona artificial propuesta por McCulloch e Pitts.

1949
Donald Hebb propone la regla que serviría como base de la regla de aprendizaje de las RNA

1958
Frank Rosenblatt desarrolla una red neuronal capaz de reconocer patrones: El perceptrón.

1960
Widrow y Hoff presentan una extensión del Perceptrón llamada ADALINE

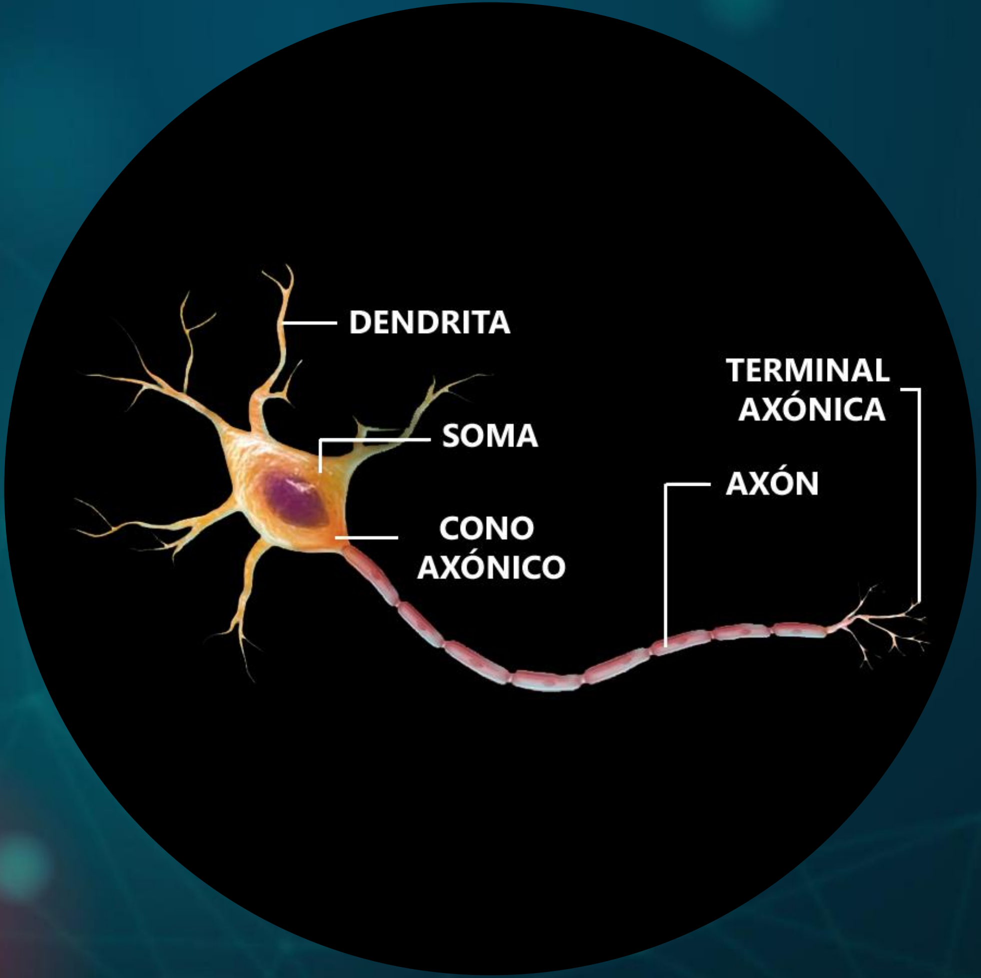
1969
Minsky y Papert ponen en pausa el avance de las redes neuronales artificiales.

El cerebro humano

Compuesto por aproximadamente 10 billones de neuronas.

Es capaz de realizar aproximadamente 60 trillones de conexiones sinápticas.

Es un sistema de procesamiento paralelo.



NEURONA BIOLÓGICA

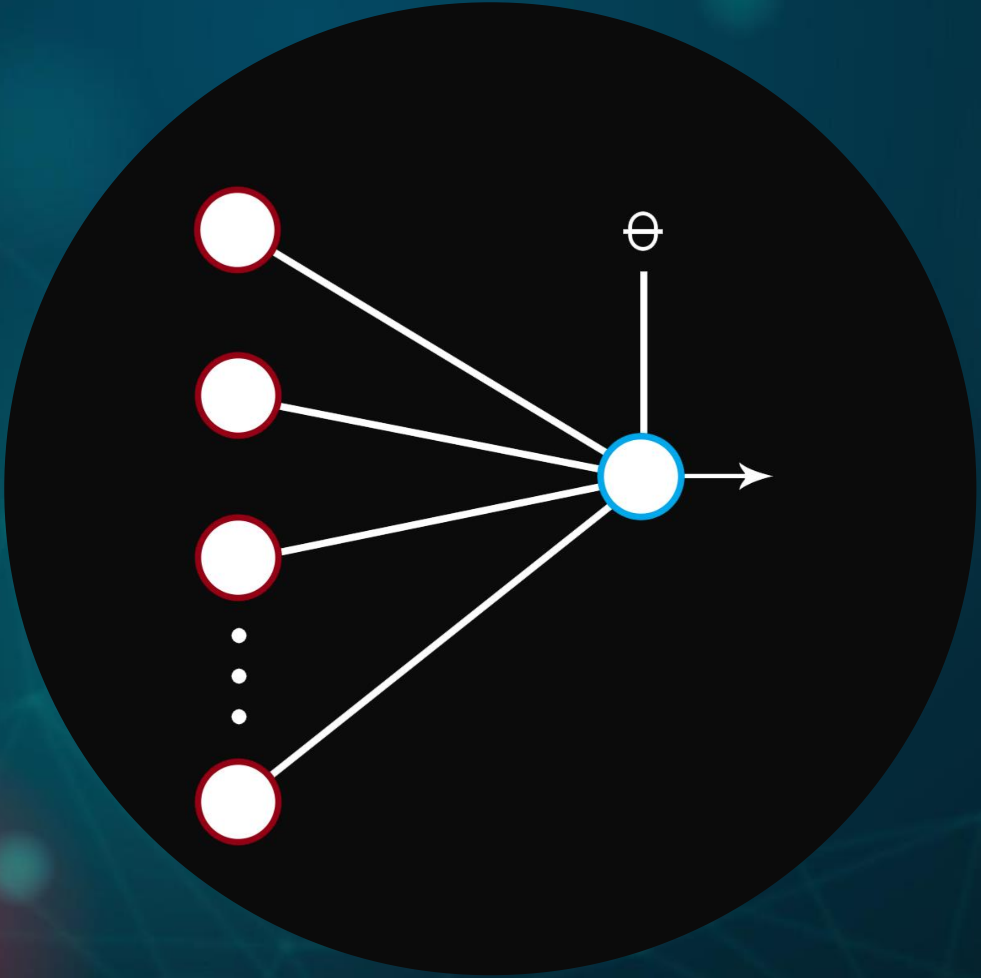
LEY DE TODO O NADA

"Si un estímulo es de la intensidad suficiente para alcanzar o sobrepasar el umbral de excitación de una neurona, se desencadenará un impulso nervioso de la misma magnitud. Si el estímulo es débil entonces no se conseguirá sobrepasar el umbral de excitabilidad y por tanto no se producirá ningún tipo de reacción."

NEURONA ARTIFICIAL

Primer modelo propuesto por McCulloch y Pitts. (1943)

Representa la neurona biológica utilizando una regla de propagación y una función de activación.



NEURONA ARTIFICIAL

RED NEURONAL ARTIFICIAL

$$net_i = \sum_{j=1}^{n} (w_{ij} \cdot x_j) + (1) \cdot (-\theta)$$

$$net_i = \sum_{j=1}^{n} (w_{ij} \cdot x_j) + w_{10} \cdot x_0$$

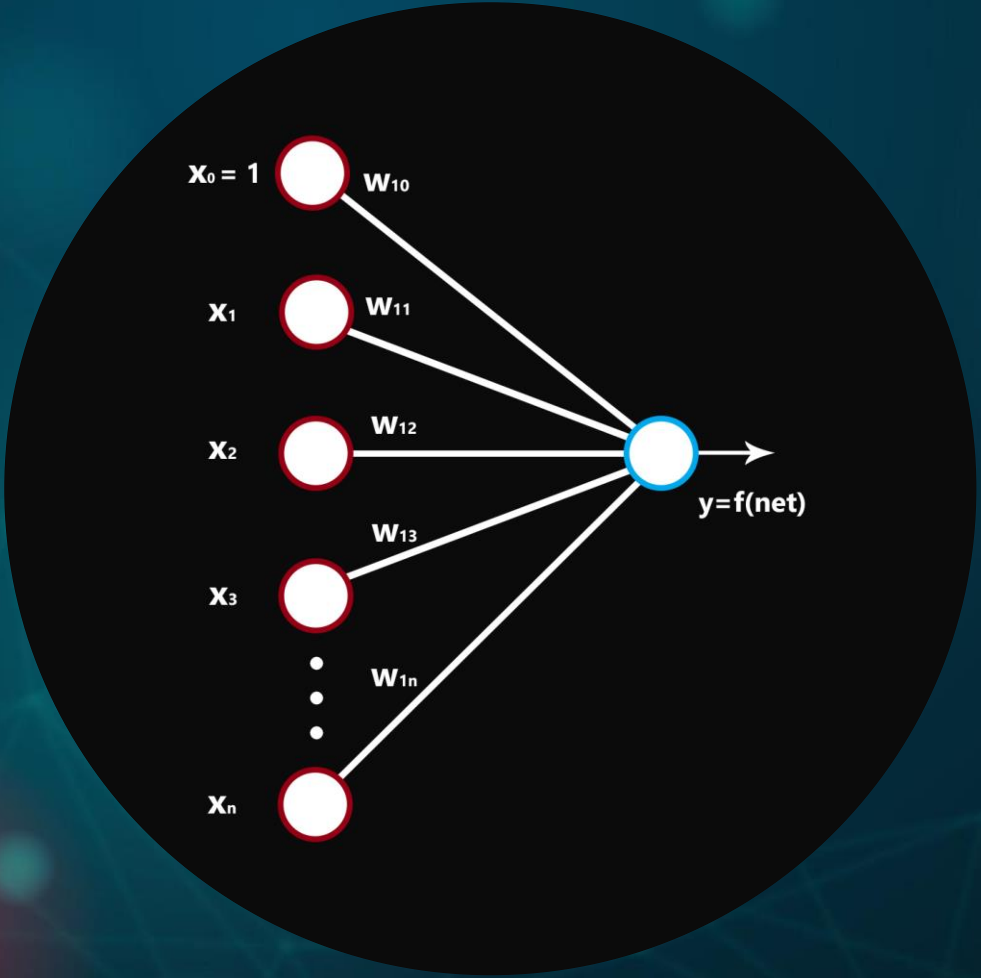
$$net_i = \sum_{j=0}^{n} w_{ij} \cdot x_j$$

“i” representa la neurona que recibe la señal.

“j” representa la neurona que emite la señal.

“n” representa la cantidad de neuronas de entrada.

“θ” representa el umbral de la neurona de salida.



RED NEURONAL ARTIFICIAL

Perceptrón 02

Primera red capaz de reconocer patrones.

ADALINE - 1958

Frank Rosenblatt

PERCEPTRÓN

Definición

Red neuronal capaz de distinguir patrones y así resolver problemas de clasificación.

Características

- Simple
- Capacidad de aprendizaje

Tipos de Aprendizaje

Supervisado

El aprendizaje de la red se realiza con el conocimiento previo del resultado deseado.

Para esto se utiliza un conjunto de ejemplos conformado por las entradas y su respectiva salida.

No Supervisado

El aprendizaje se realiza sin previo conocimiento del resultado deseado.

Por tanto la red no precisa conocer la salida, en cambio aprende con solo los datos de entrada.

Función de Activación

Función encargada de aplicar la ley de todo o nada a la neurona de salida.

Aplica el umbral de salida requerido por la neurona.

Para el Perceptrón esta definida por:

$$y = f(net) = \begin{cases} 1, & net \geq 0 \\ 0, & net < 0 \end{cases}$$

Estructura

Información que alimenta la red (variables de entrada).

Camada de entrada

Salidas de la red. Resultados.

Camada de Salida

Entrada neta. Encargada de ponderar las entradas a través de pesos sinápticos.

Función Suma

Tambien llamada función de activación. Proporciona la señal emitida por la neurona de la camada de salida.

Función de Salida

ENTRENAMIENTO DE UN PECEPTRÓN

ENTRENAMIENTO DE UNA RED

1. CODIFICACIÓN

CODIFICACIÓN

Un sistema se encuentra en funcionamiento normal cuando dos sensores emiten una señal y ambas luces se encuentran encendidas. El sistema entrara en alerta su por lo menos uno de los sensores deja de emitir una señal y alguna luz se apaga.

ENTRADAS		SALIDA
SENSOR 1	**SENSOR 2**	**DESEADO**
APAGADO	APAGADO	ALERTA
APAGADO	ENCENDIDO	ALERTA
ENCENDIDO	APAGADO	ALERTA
ENCENDIDO	ENCENDIDO	NORMAL

ENTRENAMIENTO DE UNA RED

1. CODIFICACIÓN

2. DISEÑO DE LA ARQUITECTURA

ENTRENAMIENTO DE UNA RED

1. CODIFICACIÓN

2. DISEÑO DE LA ARQUITECTURA

3. PROPAGACIÓN HACIA ADELANTE (*FORWARD PROPAGATION*)

PROPAGACION HACIA ADELANTE

Se inicializa la red con pesos sinápticos aleatorios. Usando estos pesos se calcula la salida neta de la red.

Para este ejemplo utilizaremos como valores iniciales:

$$w_{10} = 0$$

$$w_{11} = w_{12} = 3$$

ENTRENAMIENTO DE UNA RED

1. CODIFICACIÓN

2. DISEÑO DE LA ARQUITECTURA

3. PROPAGACIÓN HACIA ADELANTE (*FORWARD PROPAGATION*)

4. AJUSTE DE PESOS

AJUSTE DE PESOS SINÁPTICOS

La gran importancia del trabajo de Frank Rosenblatt en el reconocimiento de patrones constituyó en mostrar que un Perceptrón es capaz de aprender de un conjunto de ejemplos.

Frank Rosenblatt estableció una regla de aprendizaje para la red Perceptrón dada por:

$$\Delta w_{ij} = \propto (d_i - y_i) \cdot x_j$$

El ajuste solamente se realizara si existe un error.

REGLA DE APRENDIZAJE

$$\Delta w_{ij} = \propto (d_i - y_i) \cdot x_j$$

$$\Delta w_{ij} = w_{ij}(nuevo) - w_{ij}(antiguo)$$

$$w_{ij}(nuevo) = w_{ij}(antiguo) + \alpha(d_i - y_i) \cdot x_j$$

donde,

α es la tasa de aprendizaje.
d_i es el valor de salida deseado.
y_i es el valor de salida obtenido.

TASA DE APRENDIZAJE

Es un parámetro que nos indica el tamaño del paso a la hora de realizar un ajuste en los pesos.

Para este ejemplo utilizaremos como valores iniciales:

$$\alpha = 1$$

EJEMPLO

Generalización del Perceptrón

Ejercicio

Una librería desea un sistema capaz de clasificar automáticamente los clientes en cuatro categorías de modo que pueda establecer el descuento adecuado a la hora de realizar el pago en caja. Los descuentos son: Diamante (15%), Oro (10%), Plata (5%) y Bronce (0%).

Cliente	Tarjeta Fidelidad	Compra mayor a $50	Pago en efectivo	Categoría
1	Si	Si	Si	Diamante
2	Si	Si	No	Diamante
3	Si	No	Si	Oro
4	Si	No	No	Oro
5	No	Si	Si	Plata
6	No	Si	No	Plata
7	No	No	Si	Bronce
8	No	No	No	Bronce

Para este ejemplo utilizaremos como valores iniciales:

$$w_{ij} = 1$$

$$\propto = 1$$

ADALINE 03

ADAptative **LIN**ear **E**lement

Limitación del Perceptrón

Una de las limitaciones del perceptrón es que solamente puede proporcionar como salida una señal discreta (si o no).

Esto hace que sea muy malo para trabajar con patrones ruidosos.

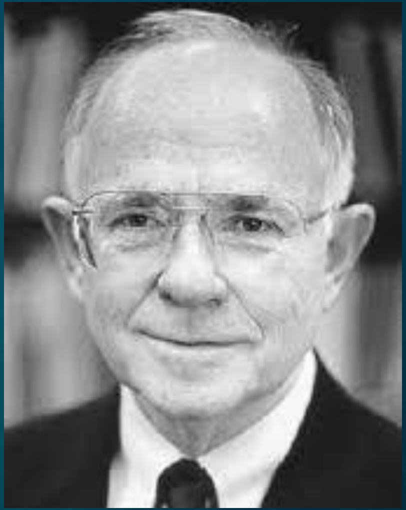
¿Solución?

Tipos de Función de Activación Continuas

Lineales

No Lineales

ADALINE - 1960



Bernard Widrow

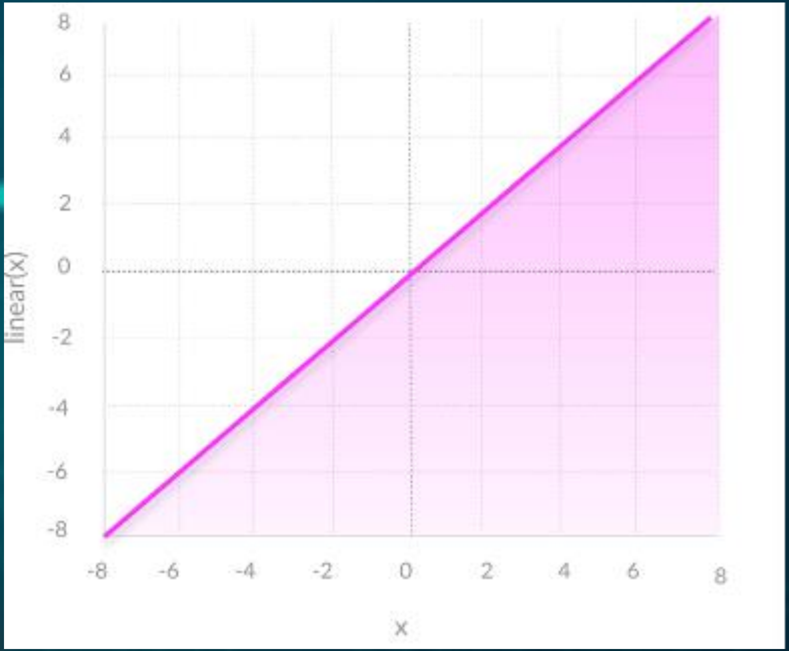
Marcian “Ted” Hoff

Tipos de Función de Activación Continuas

Lineales

No Lineales

Lineales

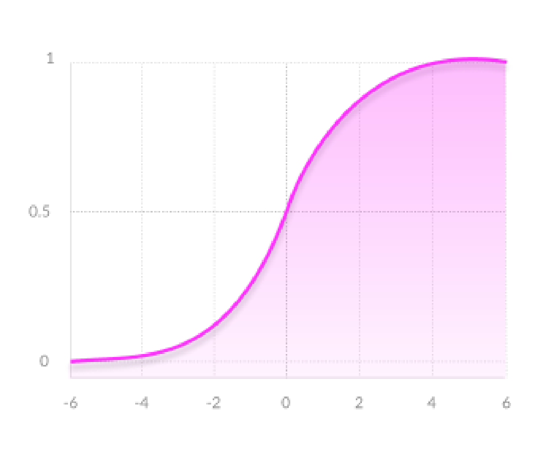


$$y = f(net) = net$$

$$y' = f'(net) = 1$$

No Lineales

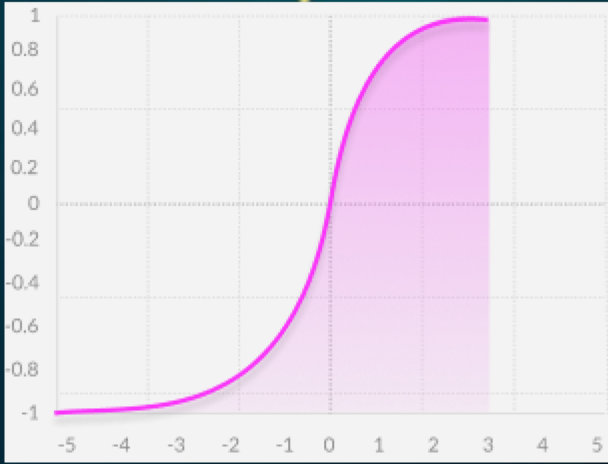
Sigmoide / Logística



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

$$y' = f'(net) = y_i(1 - y_i)$$

Tangente Hiperbólica



$$y = f(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}}$$

$$y' = f'(net) = 1 - {y_i}^2$$

REGLA DE APRENDIZAJE (REGLA DELTA)

$$\Delta w_{ij} = \propto (d_i - y_i) \cdot x_j \cdot \boldsymbol{f'(net_i)}$$

$$w_{ij}(nuevo) = w_{ij}(antiguo) + \alpha(d_i - y_i) \cdot x_j \cdot \boldsymbol{f'(net_i)}$$

donde,

α es la tasa de aprendizaje.
d_i es el valor de salida deseado.
y_i es el valor de salida obtenido.

AJUSTE DE PESOS SINÁPTICOS

"Online"

Se realiza actualizando los pesos al final de cada ejemplo.

N ejemplos implicará N ajustes de pesos.

"Offline" (Lote)

Se realiza actualizando los pesos al terminar un ciclo (época).

Se calcula Δw_{ij} para cada ejemplo. Al terminar el ciclo se actualiza utilizando la suma de todos los valores calculados.

Curva de Aprendizaje

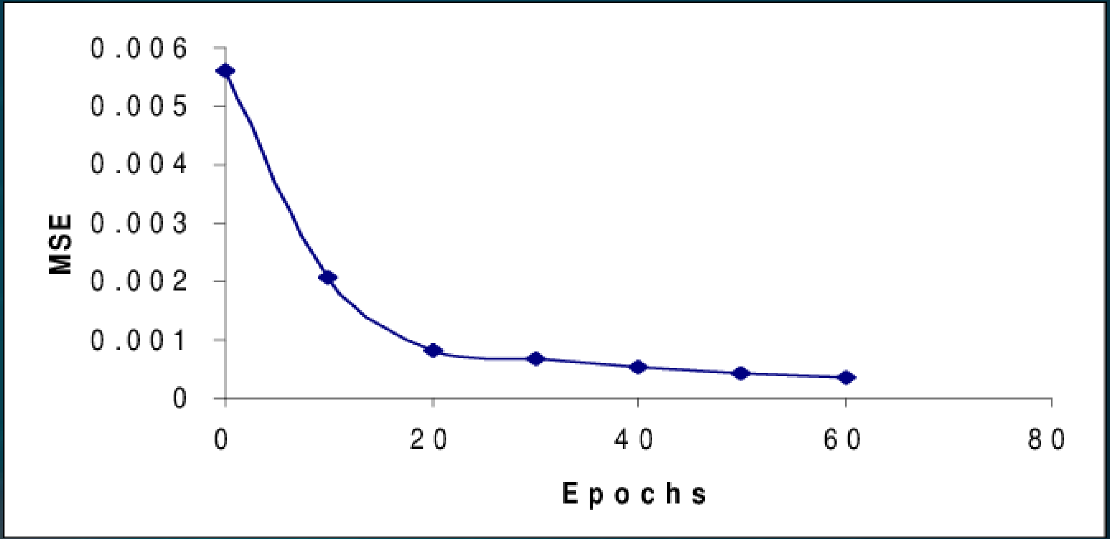
$$e_i{}^2(n) = (d_i - y_i)^2$$

$$MSE = \frac{1}{N}\sum_{k=1}^{N} e_i{}^2(n)$$

$$MSE = \frac{1}{N}\cdot\frac{1}{s}\sum_{k=1}^{N}\sum_{i=1}^{s} e_i{}^2(n)$$

donde,

n es el ejemplo actual.
N es el Total de ejemplos.
s es la cantidad de neuronas de salida.



Curva de Aprendizaje

EJEMPLO

Sistema de Previsión

Sistema de Prevision

Ejemplo	Nivel	(% AREA INUNDADA)/100
1	0.00	0.08
2	0.10	0.09
3	0.20	0.11
4	0.30	0.13
5	0.40	0.15
6	0.70	0.25
7	1.00	0.38
8	1.50	0.62

Para este ejemplo utilizaremos como valores iniciales:

$$w_{10} = -0.5$$

$$w_{11} = 0.5$$

$$\propto = 0.7$$

Y como función de activación la función sigmoide.

ESTANCAMIENTO DE LAS REDES NEURONALES

Python 3

1. Numpy
2. Pandas
3. Scikit-learn
4. Matplotlib

Resumen

MLP 04

Multi-**L**ayer **P**erceptrón

Combinación de Perceptrones

1986: Regla Delta

David E. Rumelhart

Aproximador Universal de Función

ENTRENAMIENTO DE UNA RED

1. DISEÑO DE LA ARQUITECTURA

Estructura

Información de entrada al modelo

Camada de Entrada

Es la encargada de la no linealidad de la red.

Camada Escondida

Provee la respuesta de la red a la información de entrada.

Camada de Salida

Ejemplo

EJEMPLO	ENTRADA	SALIDA
N	x_1	y
1	1.5	0.02
2	2.0	0.08
3	2.5	0.18
4	3.0	0.50
5	3.5	0.65
6	4.0	0.88
7	4.5	0.96

Utilizando los siguientes pesos:

$$w_{10}^{1} = 0.1 \quad w_{11}^{1} = -0.3$$

$$w_{20}^{1} = -0.7 \quad w_{21}^{1} = 0.4$$

$$w_{10}^{2} = -0.6 \; w_{11}^{2} = 0.1$$

$$w_{12}^{2} = -0.8$$

$$\alpha = 0.7$$

3 Neuronas en la camada escondida

ENTRENAMIENTO DE UNA RED

1. DISEÑO DE LA ARQUITECTURA

2. PROPAGACIÓN HACIA ADELANTE (*FORWARD PROPAGATION*)

Forward Propagation

$$net_i^1 = \sum_{j=0}^{N_{in}} w_{ij}^1 \cdot x_j$$

$$net_i^2 = \sum_{j=0}^{N_{hid}} w_{ij}^2 \cdot f^1(net_j^1)$$

“i” representa la neurona que recibe la señal.

“j” representa la neurona que emite la señal.

“N_{in}” representa la cantidad de neuronas de en la camada de entrada.

“N_{hid}” representa la cantidad de neuronas de en la camada escondida.

EL PROBLEMA

ENTRENAMIENTO DE UNA RED

1. DISEÑO DE LA ARQUITECTURA

2. PROPAGACIÓN HACIA ADELANTE (*FORWARD PROPAGATION*)

3. PROPAGACIÓN HACIA ATRAS (*BACKPROPAGATION*)

Backward Propagation

Cálculo de Sensibilidades

$$\delta_i^M = f'^{\,M}(net_i^M) \cdot e_i(n)$$

$$\delta_i^{m-1} = f'^{\,m-1}(net_i^{m-1}) \sum_{i=1}^{N} w_{ij}^m \cdot \delta_i^m$$

“i” representa la neurona que recibe la señal.

“j” representa la neurona que emite la señal.

“M” representa la ultima camada de la red.

“m” representa el numero de camada.

"δ“ representa la sensibilidad.

ENTRENAMIENTO DE UNA RED

1. DISEÑO DE LA ARQUITECTURA

2. PROPAGACIÓN HACIA ADELANTE (*FORWARD PROPAGATION*)

3. PROPAGACIÓN HACIA ATRAS (*BACKPROPAGATION*)

4. AJUSTE DE PESOS

Ajuste de Pesos

$$w_{ij}^{m}(nuevo) = w_{ij}^{m}(antiguo) + \alpha \cdot \delta_i^m \cdot f^{m-1}(net_i^{m-1})$$

$$w_{ij}^{m}(nuevo) = w_{ij}^{m}(antiguo) + \alpha \cdot \delta_i^m \cdot x_j$$

“i” representa la neurona que recibe la señal.

“j” representa la neurona que emite la señal.

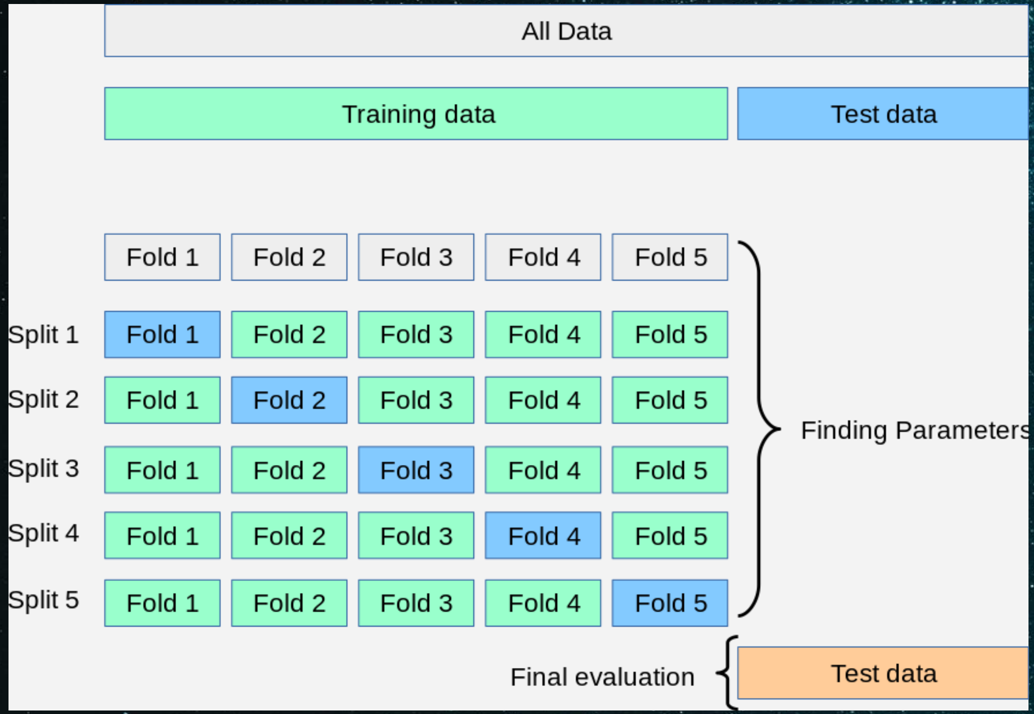
“M” representa la ultima camada de la red.

“m” representa el numero de camada.

"δ“ representa la sensibilidad.

Ejemplo Python

OVERFIT

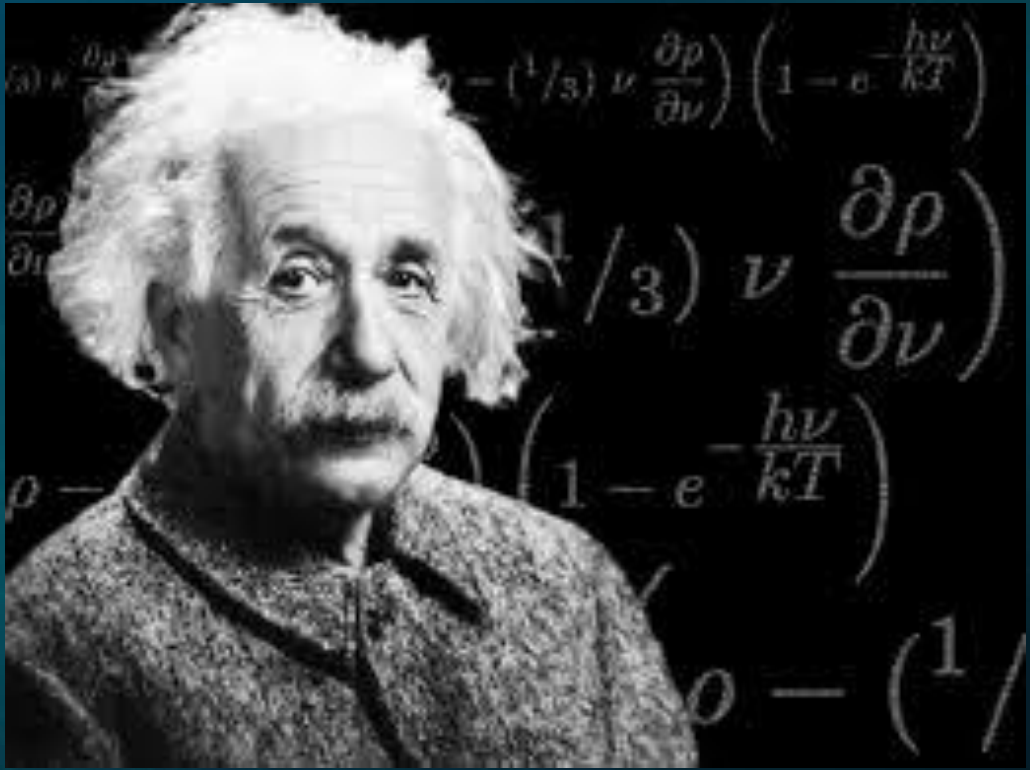

All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

TENSORFLOW

KERAS

CNN 05

Convolutional Neural Networks




CLOUD9
VIP


CLOUD9
VIP

Extracción de Características

Kernel

Filtro

Detector de Características

Extractor por Convolución

Kernel

0.33	0.17	-0.94
0.54	-0.47	0.24
-0.89	-0.68	0.25

Convolución

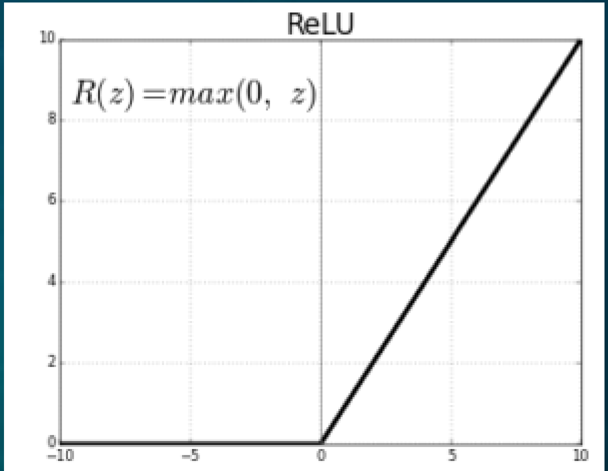
-1	-1	-1	-1	-1
-1	1	1	1	-1
-1	-1	1	-1	-1
-1	-1	1	-1	-1
-1	-1	1	-1	-1

-1	-1	-1
1	1	1
-1	-1	-1

-1	1	-1
-1	1	-1
-1	1	-1

Padding

-1	1	1	1	-1
-1	-1	1	-1	-1
-1	-1	1	-1	-1
-1	-1	1	-1	-1
-1	-1	-1	-1	-1



ReLU

Rectified Linear Unit

Pooling

Realiza un “down sampling”

Max

Mean

Min

Nuevos Parámetros

“F” el tamaño del filtro/kernel.

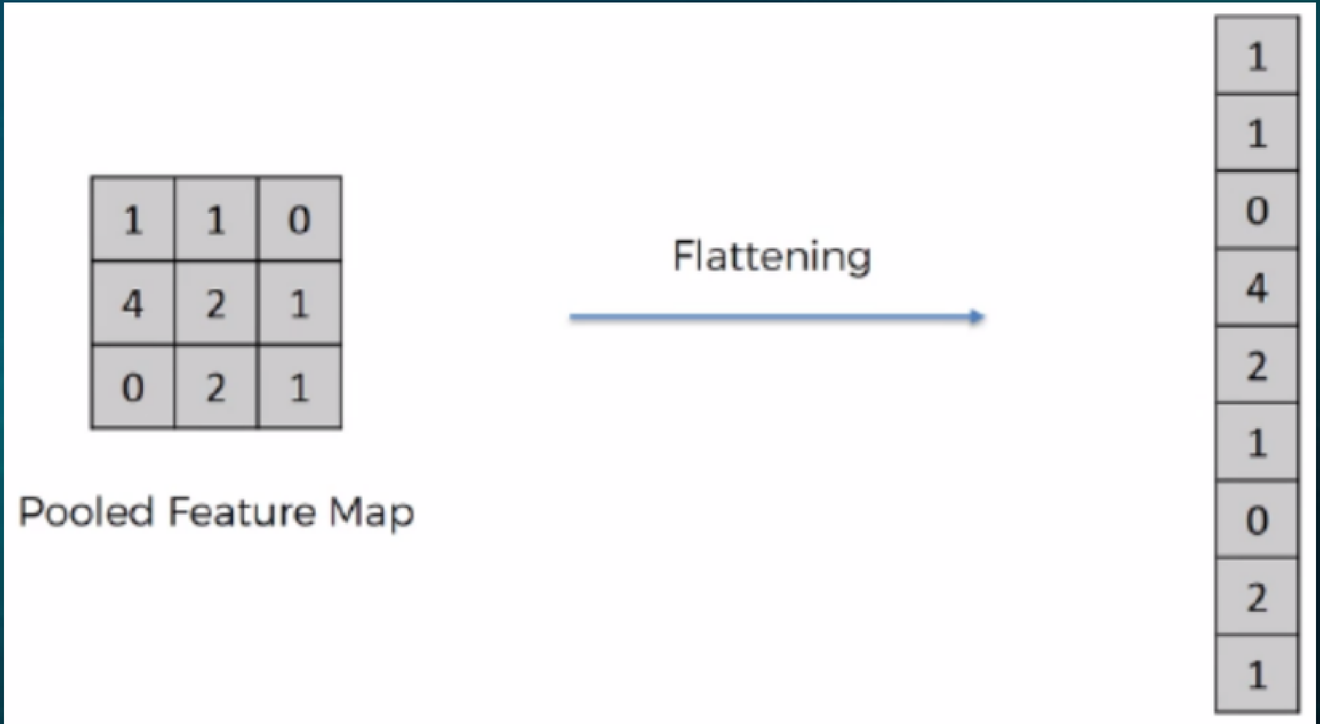
“S” representa el “Stride”.

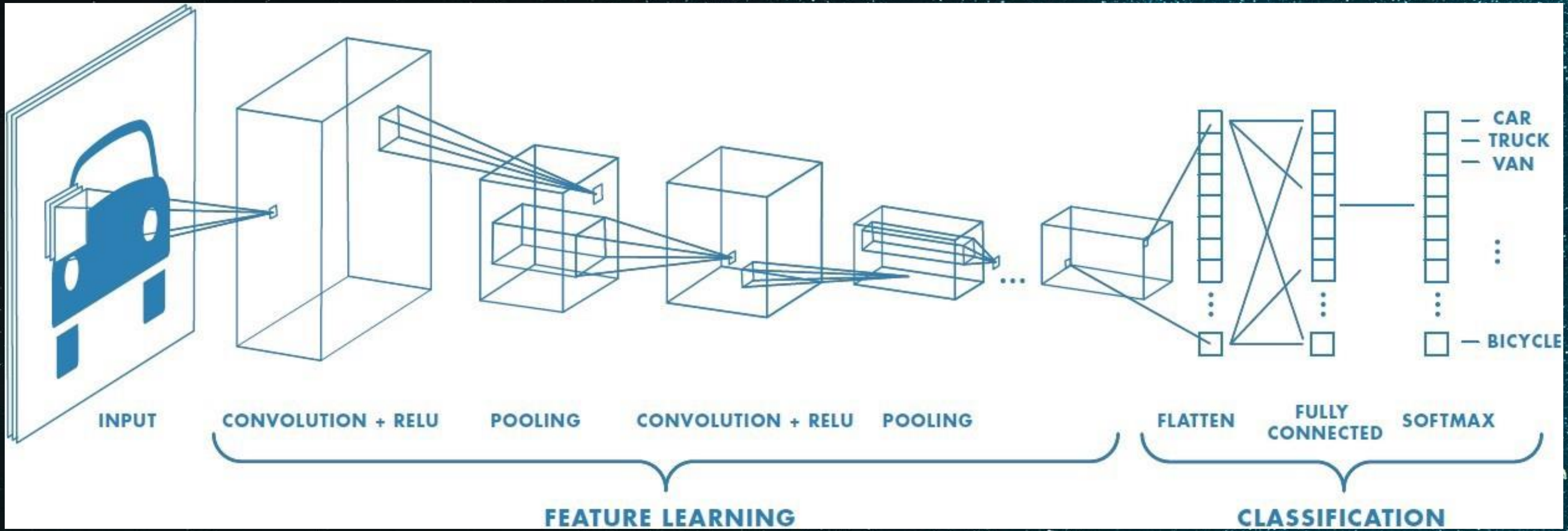
“K” representa el numero de filtros/kernels.

“P” representa el “Zero Padding”.

Pooling

Flattening

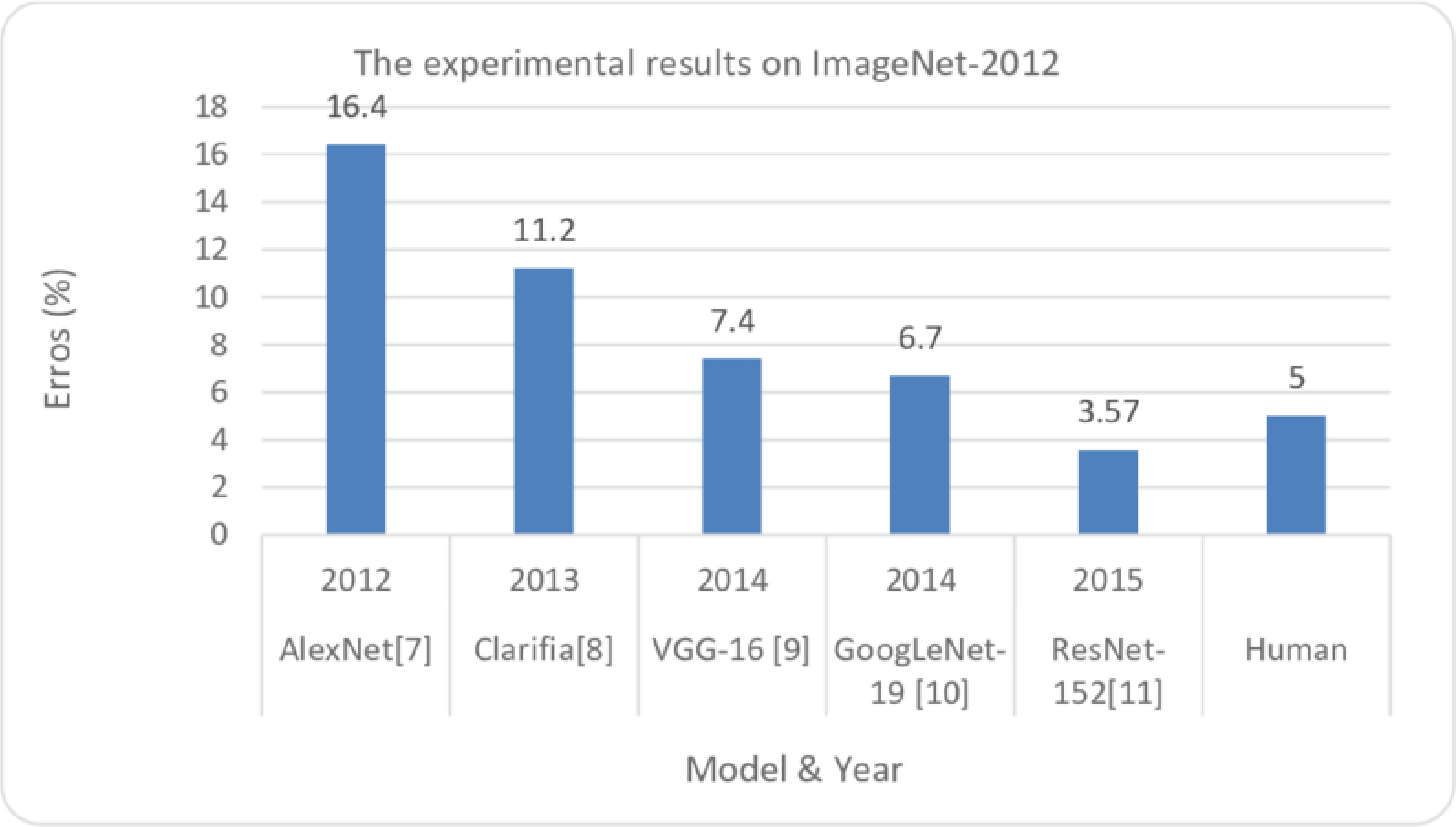



CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.


The experimental results on ImageNet-2012
Erros (%)
18
16
14
12
10
8
6
4
2
0
16.4
11.2
7.4
6.7
3.57
5
2012
AlexNet[7]
2013
Clarifia[8]
2014
VGG-16 [9]
2014
GoogLeNet-19 [10]
2015
ResNet-152[11]
Human
Model & Year