Deep Learning for dummies book

Introducció a la intel·ligència artificial - 3

Aina Palacios

Aina Palacios

- Enginyera de Telecomunicacions especialitzada en Audiovisuals
- Màster en Tecnologías Avançades especialitzada en deep learning en Multimèdia!
- Experiència en programació web i machine learning.
- Mentora a IT Academy de **DataScience**

https://www.linkedin.com/in/ainapc/

ainaPali#2617

Deep Learning

1

Deep Learning

Nova evolució del Machine Learning basat en la forma que aprènen els humans. Basat en Neural Networks

2

Machine Learning

Algoritmes que analitzen les dades, aprenen d'elles i fan decisions respecte al que han après

3

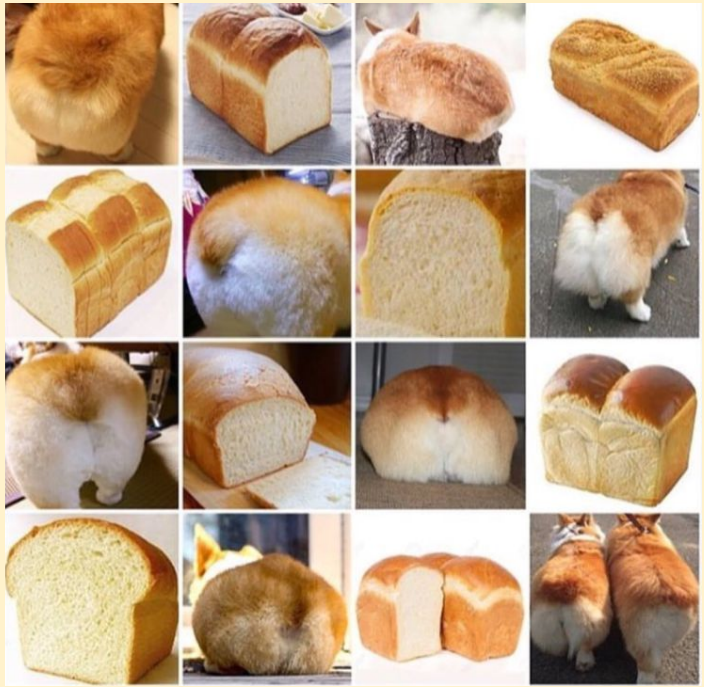
Intel·ligència artificial

Tot allò que engloba comportaments intel·ligents.

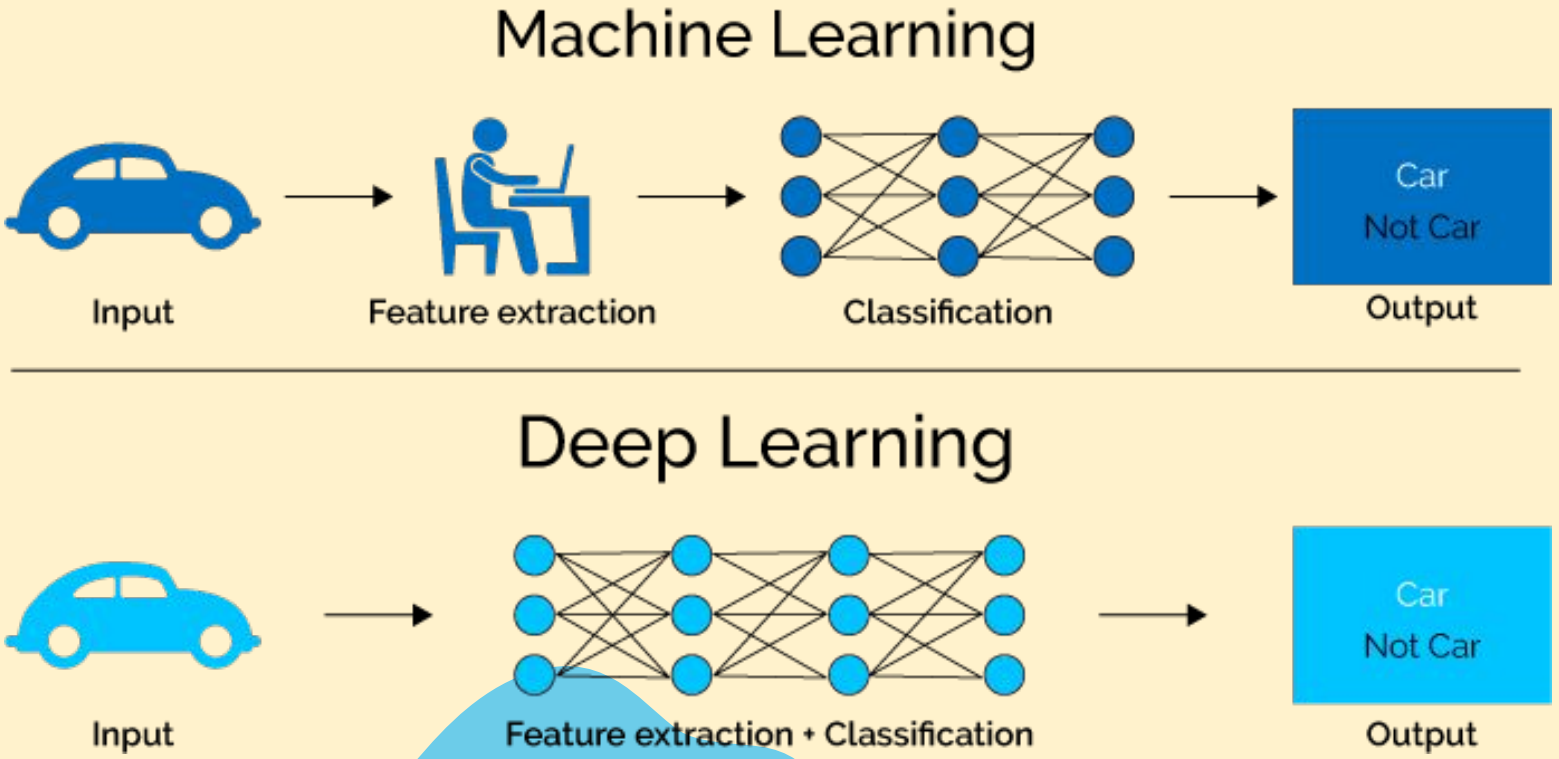
Perquè volem utilitzar deep learning?

- Està basat en Neural Network (Xarxes Neuronals), per tant és el més proper al comportament humà que tenim!!
- Quan la data és desestructurada o és complexa, deep learning és la solució!
- Molt utilitzat sobretot en image recognition, speech, and computer vision applications.
- Són molt molt ràpides a l'hora de funcionar!
- Podem fer servir xarxes ja entrenades

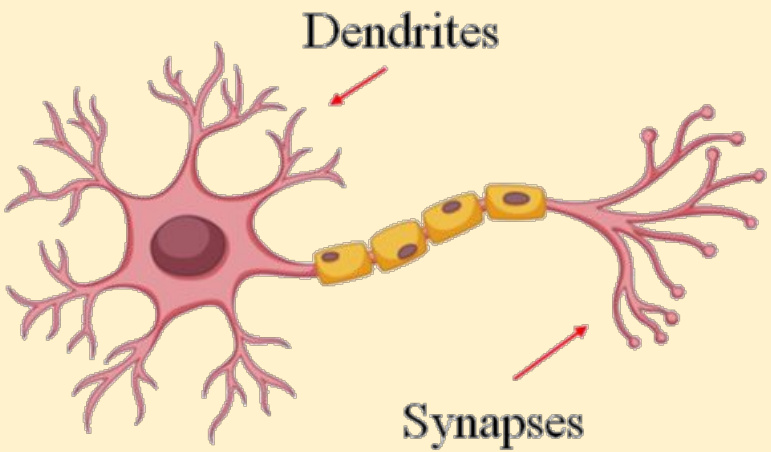
Desavantatges: Necessita molta molta dada i complex i llarg entrenament



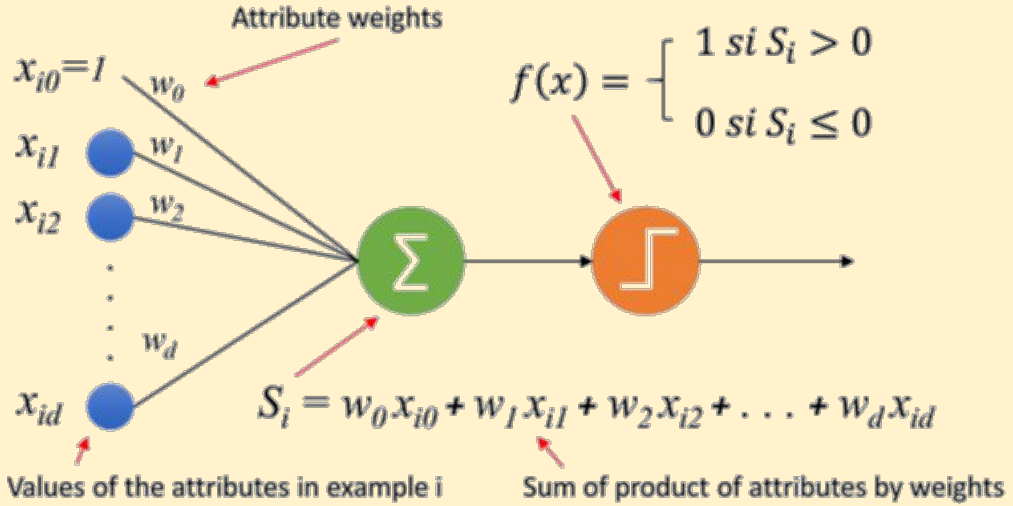
Machine Learning vs Deep Learning



Neural Networks: THE PERCEPTRON



NEURON



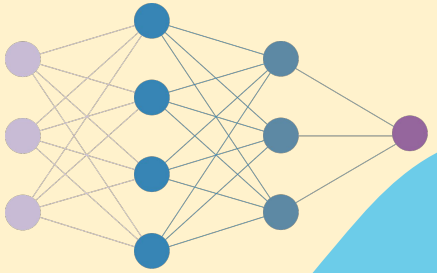
PERCEPTRON

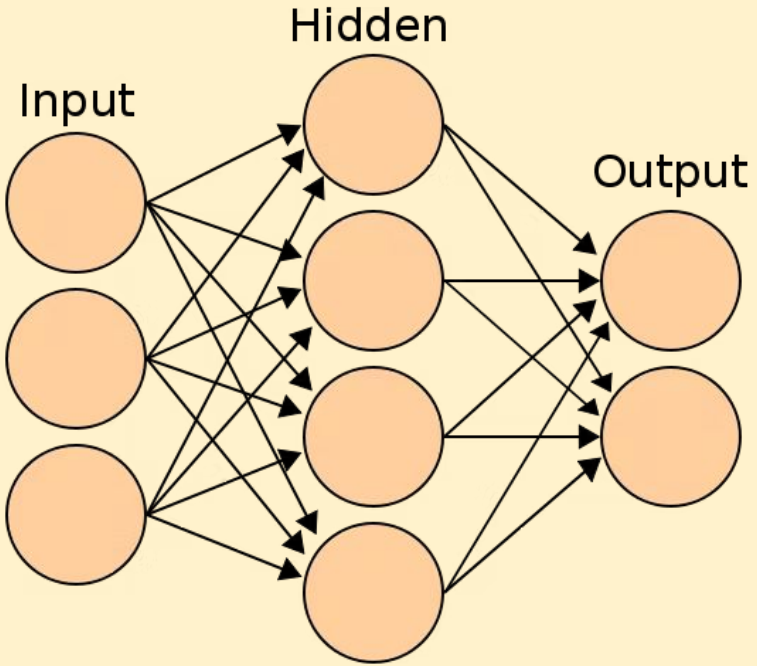
The hidden layer!

Amb una neurona, no podem predir res. Però què passa si ajuntem un munt de Perceptrons?

La idea de les **Neural Networks** és crear una estructura amb una layer d'entrada, ona o moltes hidden layers i una output layer.

Així, les entrades de els nostres perceptrons seran primerament les input layers i seguidament les hidden layers.

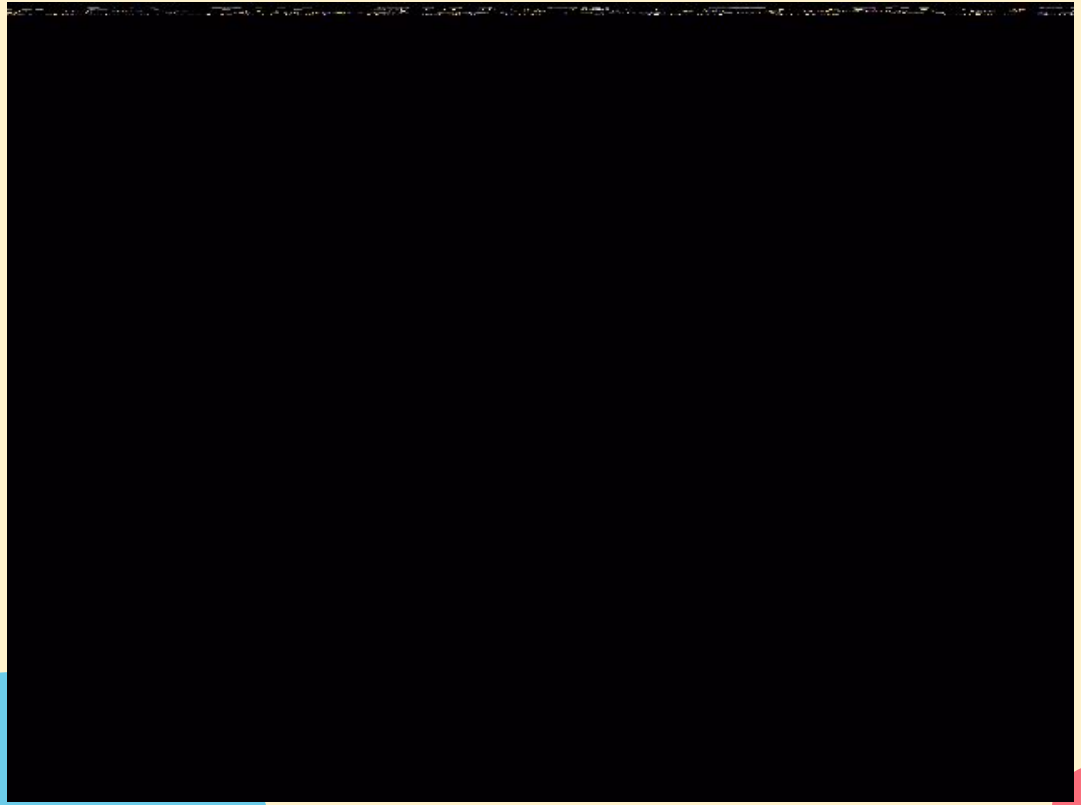




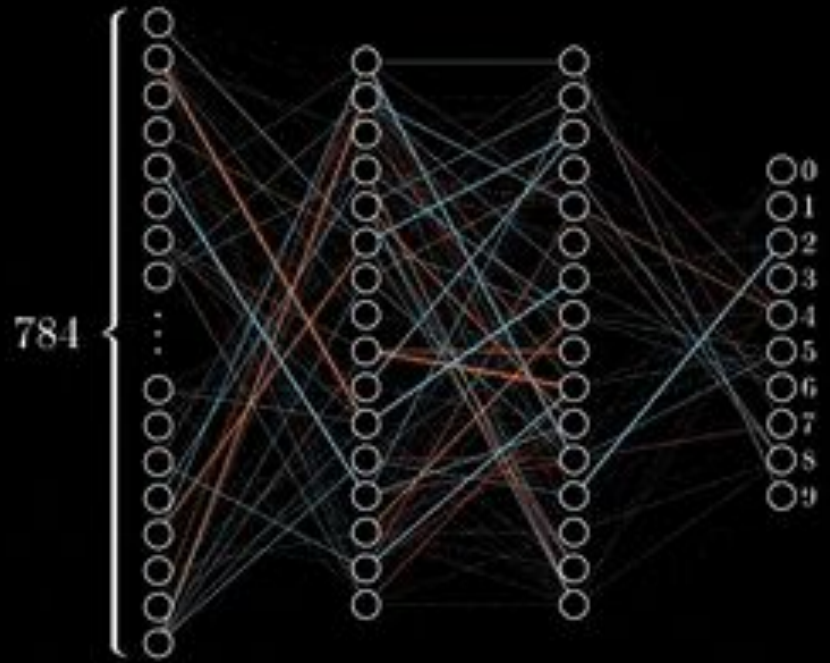
MATHS!

Durant l’entrenament, les xarxes modifiquen els seus pesos per intentar reduïr l’error al màxim! És el que en diem Backpropagation.

Més info:
https://hmkcode.com/ai/backpropagation-step-by-step/



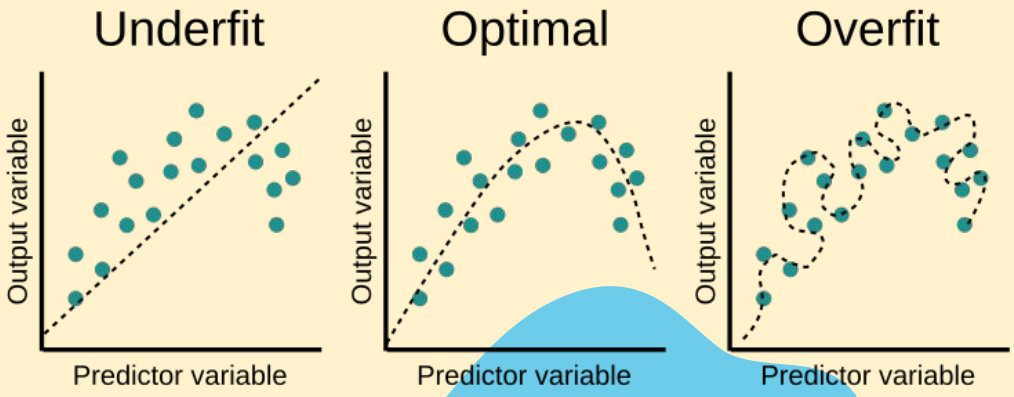
Training in progress...

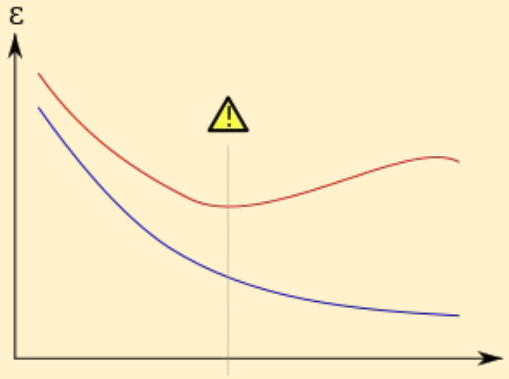


Train and Test!

Utilitzarem, si són dades molt senzilles, poques hidden layers, per evitar overfitting (que s'adapti massa al train).

Hi ha molts mètodes per saber quan parar l'entrenament: Si l'error és molt petit, si l'error ja no disminueix en el temps, si detactem overfitting, nombre d'iteracions..





Deep learning process

Training data

Feature vector Label

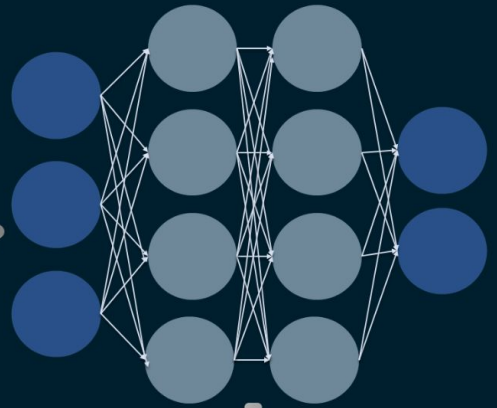
(135, 57, 3, 98, ...) (Bread)

(256, 67, 15, 46, ...) (Corgi)

(1, 354, 2, 479, ...) (Bread)

(576, 32, 6, 441, ...) (Corgi)

Deep learning NN layers



Testing data

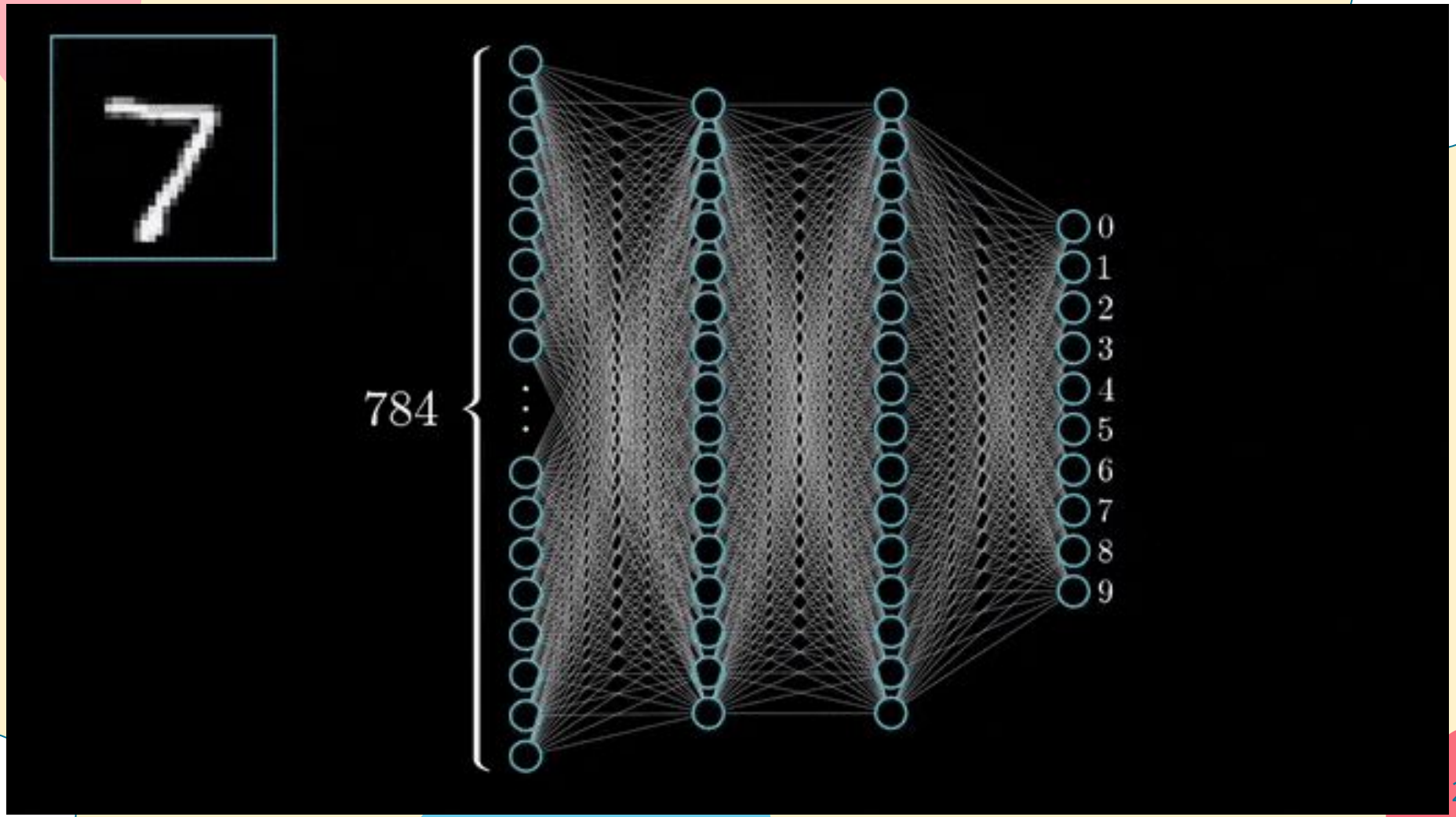
(39, 198, 95, 81, ...) (?)

Inference

Output

(Corgi)

altexsoft
software r&d engineering

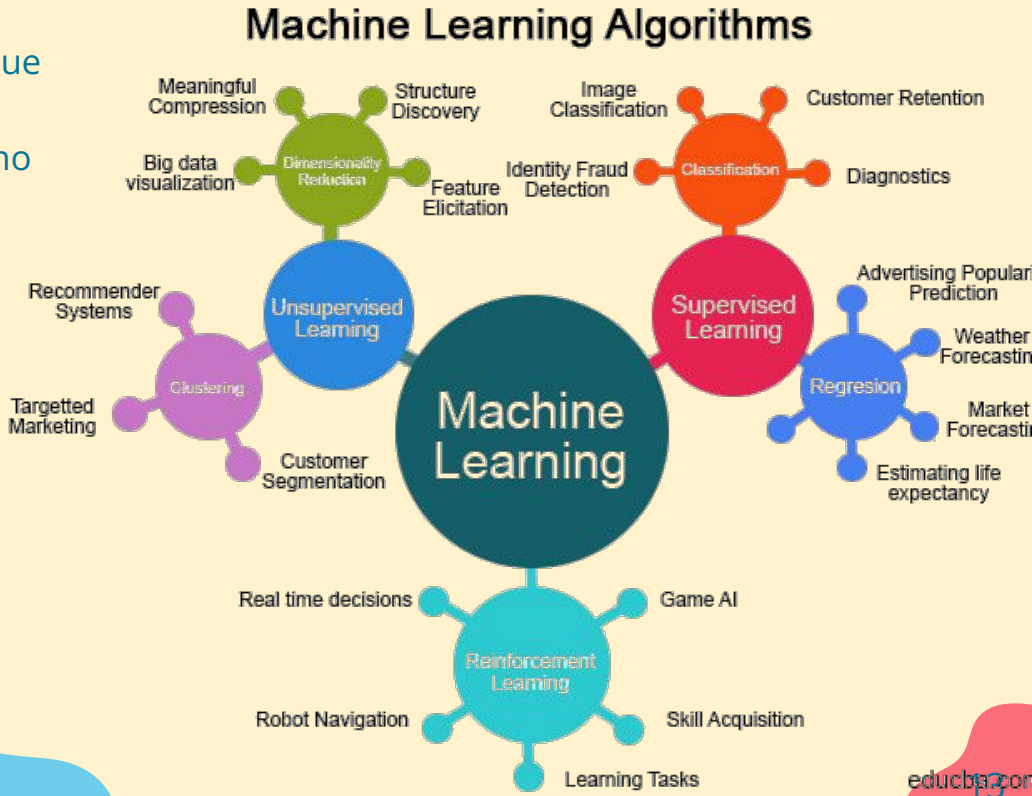

784
0
1
2
3
4
5
6
7
8
9

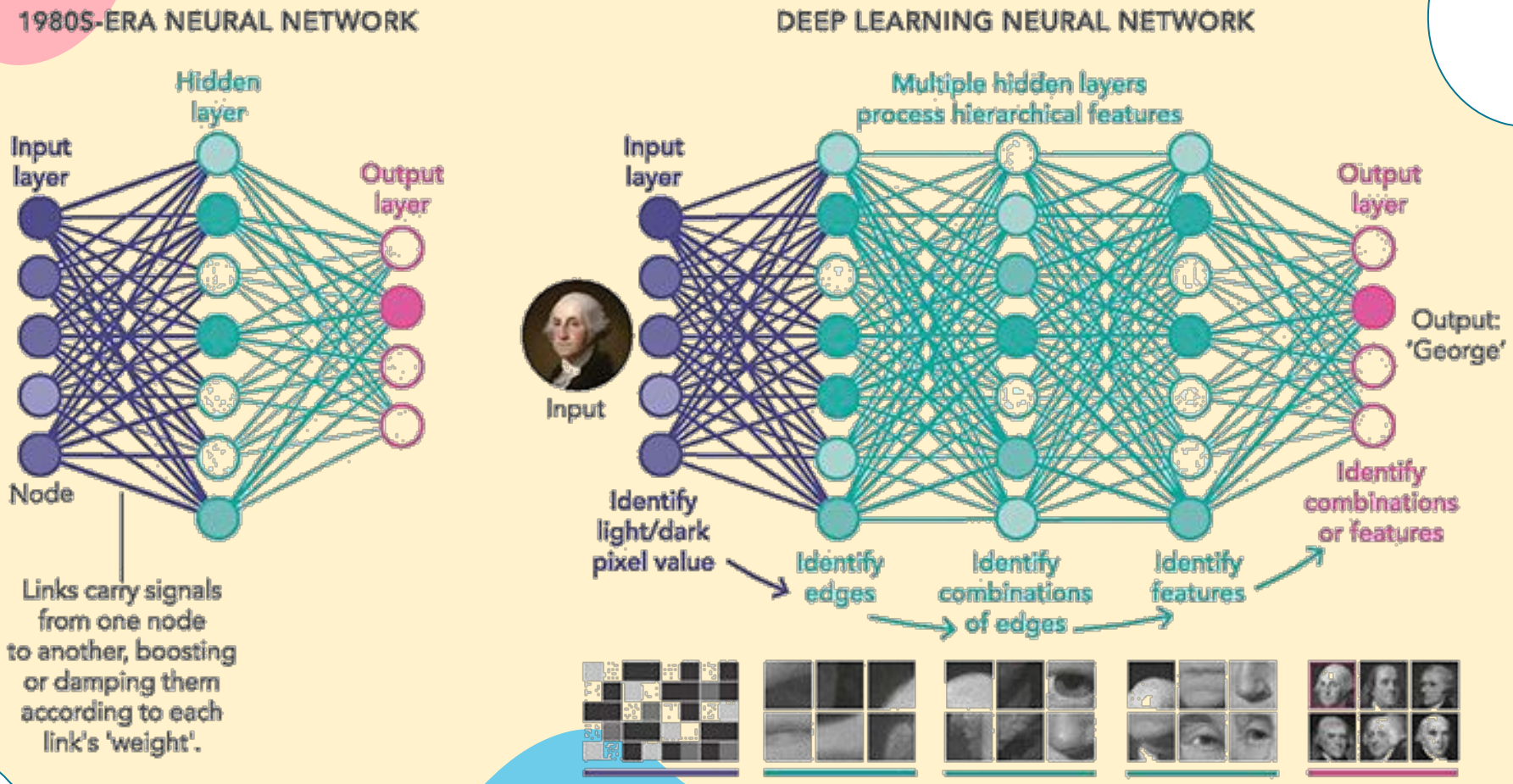
Deep Learning

Una xarxa neuronal pot extreure informació que altres algoritmes no poden, sobretot quan les dades no segueixen una estructura normal o no tenim etiquetes.

Aquestes, poden fer de tot!!! Podem fer regressió (extreure un resultat numèric), classificació, unsupervised, reinforcement learning, computer vision...

Quan una xarxa neuronal conté més d'una hidden layer, en diem deep learning, d'aqui prové el nom.



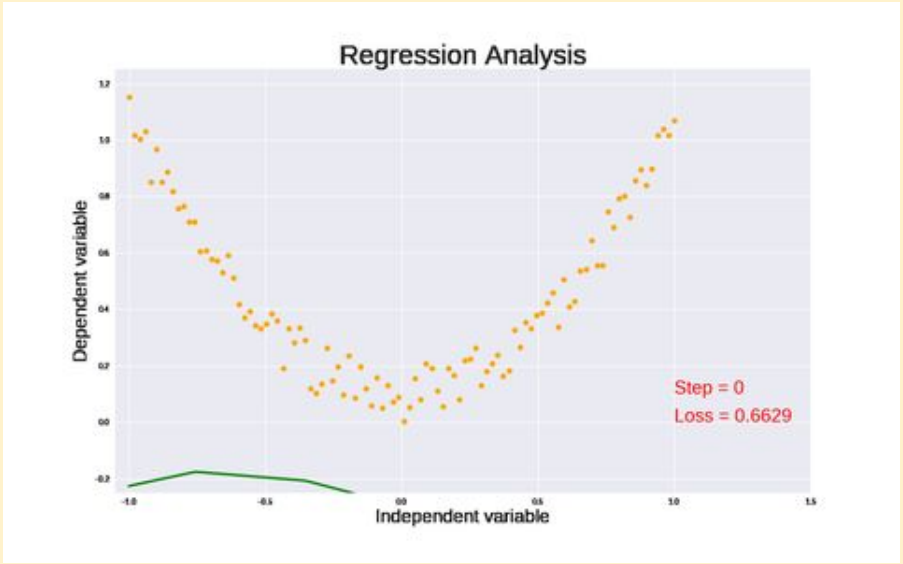

1980S-ERA NEURAL NETWORK
Hidden layer
Input layer
Output layer
Node
Links carry signals from one node to another, boosting or damping them according to each link's 'weight'.
DEEP LEARNING NEURAL NETWORK
Multiple hidden layers process hierarchical features
Input layer
Output layer
Input
Output: 'George'
Identify light/dark pixel value
Identify edges
Identify combinations of edges
Identify features
Identify combinations or features

Aplicar Deep Learning

A vegades, et pots trobar dades que no s'adapten bé als models escollits, ja que tenen una estructura estranya. Sempre pots provar d'utilitzar les Xarxes Neuronals!

Crear una xarxa neuronal des de 0 és bastant complex, però per sort la llibreria sklearn ens permet utilitzar-les! https://scikit-learn.org/stable/modules/neural_networks_supervised.html#

Exemple: https://www.kaggle.com/code/ahmethamzaemra/mlpclassifier-example/notebook



La complexitat de les Xarxes Neuronals

Les Xarxes Neuronals poden ser útils per moltes coses, però com sempre, tot depèn de les dades. Moltes vegades alguns algoritmes matemàtics podran tenir millor prediccions que una NN. Algunes, la combinació de models pot ser la solució.

A més, aquestes conenten un munt de paràmetres que necessiten molt estudi per interpretar bé.

Pensa que qualsevol modificació, qualsevol estat d'inici, pot alterar el resultat final enormement.

L'estudi d'aquest model porta anys estudiant-se, i moltes vegades, ni els professionals entenem perquè pot variar tant les dades al posar noves hidden layers.

Molta gent es dedica únicament a descobrir nous tipus de layers i posicionar-les de manera que alterin el resultat final.

Llibreries per crear Xarxes

Pytorch

PyTorch is an open source machine learning library for Python, based on Torch. It is used for applications such as natural language processing and was developed by Facebook's AI research group.

TensorFlow

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library that is used for machine learning applications like neural networks.

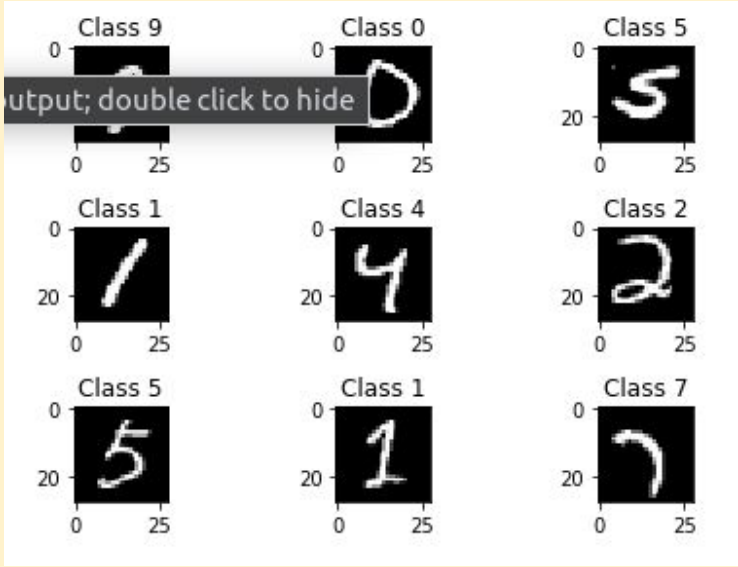
Keras

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow. It is designed to enable fast experimentation with deep neural networks.

Keras API i MNIST

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()

print("X_train shape", X_train.shape)
print("y_train shape", y_train.shape)
print("X_test shape", X_test.shape)
print("y_test shape", y_test.shape)
```

```
X_train shape (60000, 28, 28)
y_train shape (60000,)
X_test shape (10000, 28, 28)
y_test shape (10000,)
```



Modificant la imput layer

Ens interessa crear un unic vector per l'entrada d'aquesta xarxa, per tant haurem de passar d'una imatge de dos dimensions a una. A mésn normalitzarem les imatges per tenir-les al rang [0,1]

```
X_train = X_train.reshape(60000, 784) # reshape 60,000 28 x 28 matrices into 60,000 784-length vectors.
X_test = X_test.reshape(10000, 784)   # reshape 10,000 28 x 28 matrices into 10,000 784-length vectors.

X_train = X_train.astype('float32')   # change integers to 32-bit floating point numbers
X_test = X_test.astype('float32')

X_train /= 255                        # normalize each value for each pixel for the entire vector for each input
X_test /= 255

print("Training matrix shape", X_train.shape)
print("Testing matrix shape", X_test.shape)
```

```
Training matrix shape (60000, 784)
Testing matrix shape (10000, 784)
```

Modificant el target

El que volem és aplicar one-hot al target per tenir per cada classe un node, que ens indicarà entre el valor 0 i l'1 la probabilitat d'aquella classe:

```
[0, 0.94, 0, 0, 0, 0, 0.06, 0, 0]
```

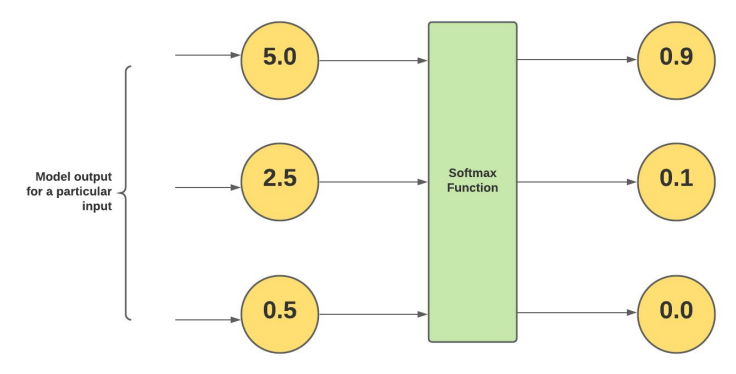
El més probeble és que sigui 1

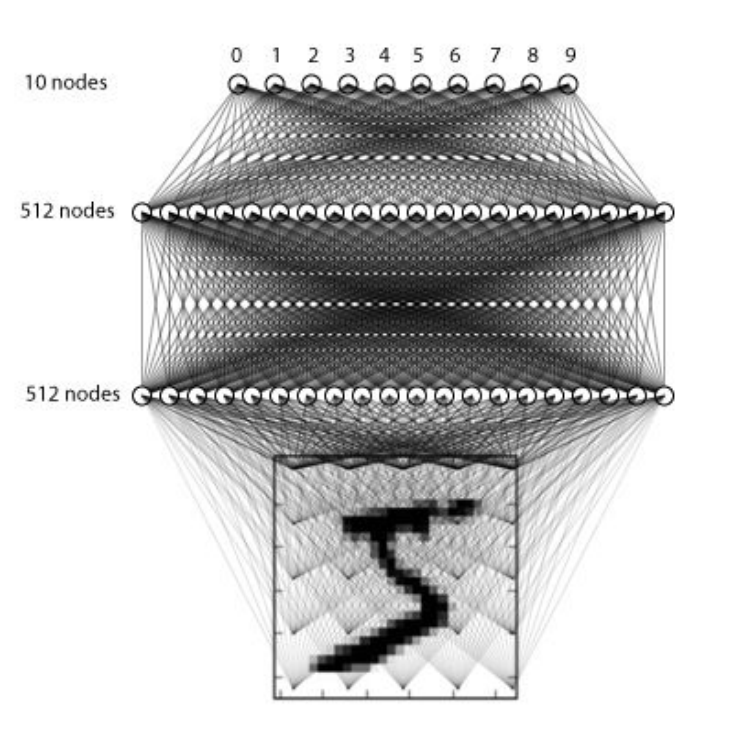
```
nb_classes = 10 # number of unique digits

Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

```
model = Sequential()

model.add(Dense(512, input_shape=(784,))) #(784,) is not a typo -- that represents a 784 length vector!
model.add(Activation('relu'))
model.add(Dropout(0.2))
```

```
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
```

```
model.add(Dense(10))
model.add(Activation('softmax'))
```

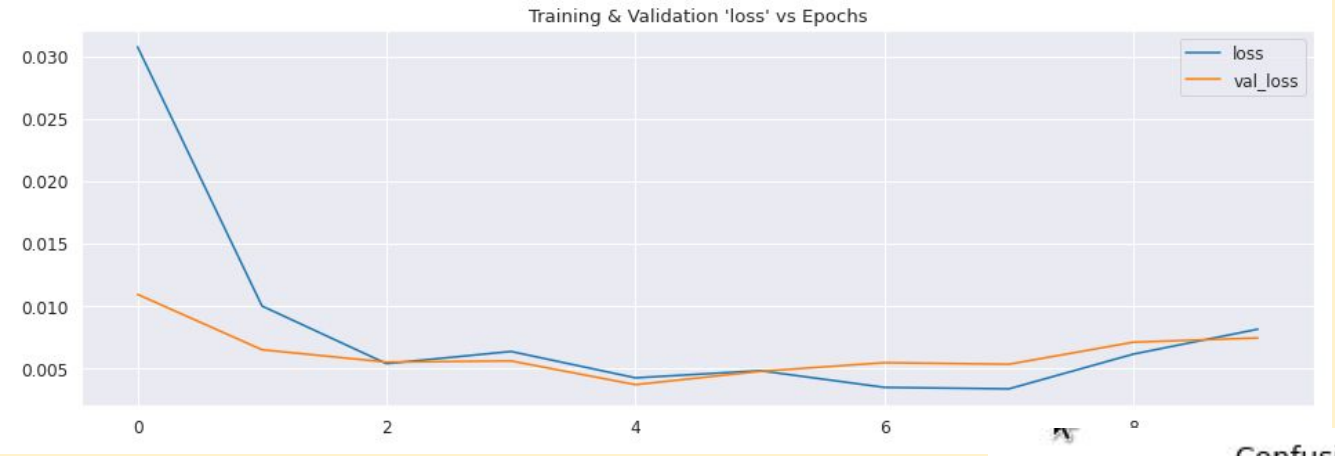


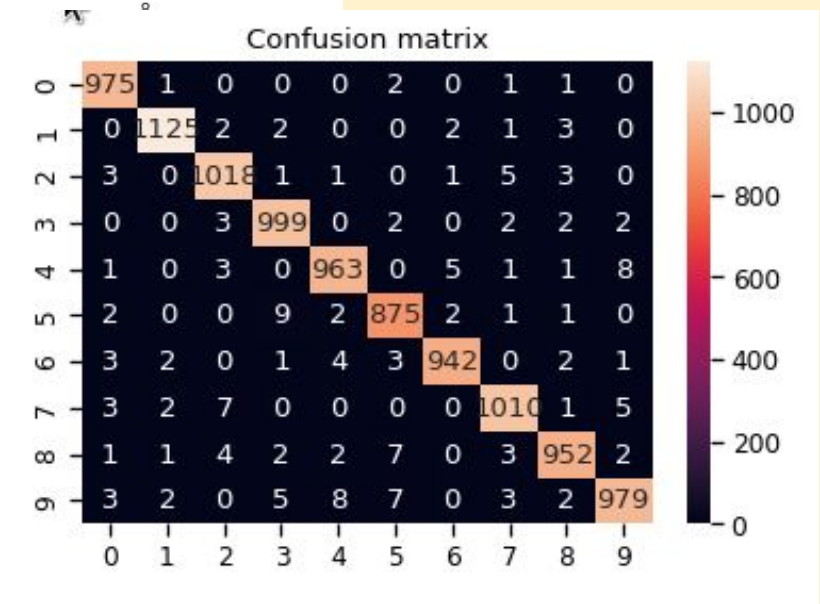


```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
from sklearn.model_selection import train_test_split

X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.15, random_state=42)

results = model.fit(X_train, Y_train,
          batch_size=64, epochs=10,
          verbose=1, validation_data=(X_val,Y_val))
```

```
Epoch 1/10
354/354 [==============================] - 3s 7ms/step - loss: 0.0307 - accuracy: 0.9947 - val_loss: 0.0109 - val_
accuracy: 0.9979
Epoch 2/10
354/354 [==============================] - 3s 8ms/step - loss: 0.0100 - accuracy: 0.9979 - val_loss: 0.0065 - val_
accuracy: 0.9989
Epoch 3/10
354/354 [==============================] - 3s 7ms/step - loss: 0.0054 - accuracy: 0.9986 - val_loss: 0.0055 - val_
accuracy: 0.9985
Epoch 4/10
354/354 [==============================] - 3s 7ms/step - loss: 0.0063 - accuracy: 0.9988 - val_loss: 0.0056 - val_
accuracy: 0.9985
Epoch 5/10
354/354 [==============================] - 3s 8ms/step - loss: 0.0042 - accuracy: 0.9986 - val_loss: 0.0037 - val_
accuracy: 0.9985
Epoch 6/10
```


Training & Validation 'loss' vs Epochs
loss
val_loss
0.030
0.025
0.020
0.015
0.010
0.005
0
2
4
6


Confusion matrix
975 1 0 0 0 2 0 1 1 0
0 1125 2 2 0 0 2 1 3 0
3 0 1018 1 1 0 1 5 3 0
0 0 3 999 0 2 0 2 2 2
1 0 3 0 963 0 5 1 1 8
2 0 0 9 2 875 2 1 1 0
3 2 0 1 4 3 942 0 2 1
3 2 7 0 0 0 0 1010 1 5
1 1 4 2 2 7 0 3 952 2
3 2 0 5 8 7 0 3 2 979
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

References

- MNIST en keras Google Collab [link]
- TensorFlow MNIST [link]
- MNIST — Digits Classification with Keras [link]
- Keras API [link]
- Overffiting en TensorFlow [link]

- NLP en Keras [link]

Fins on pot arribar el Deep Learning?

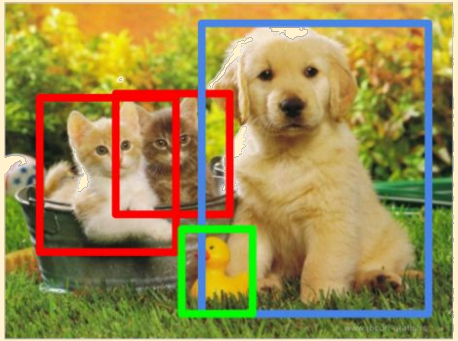
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

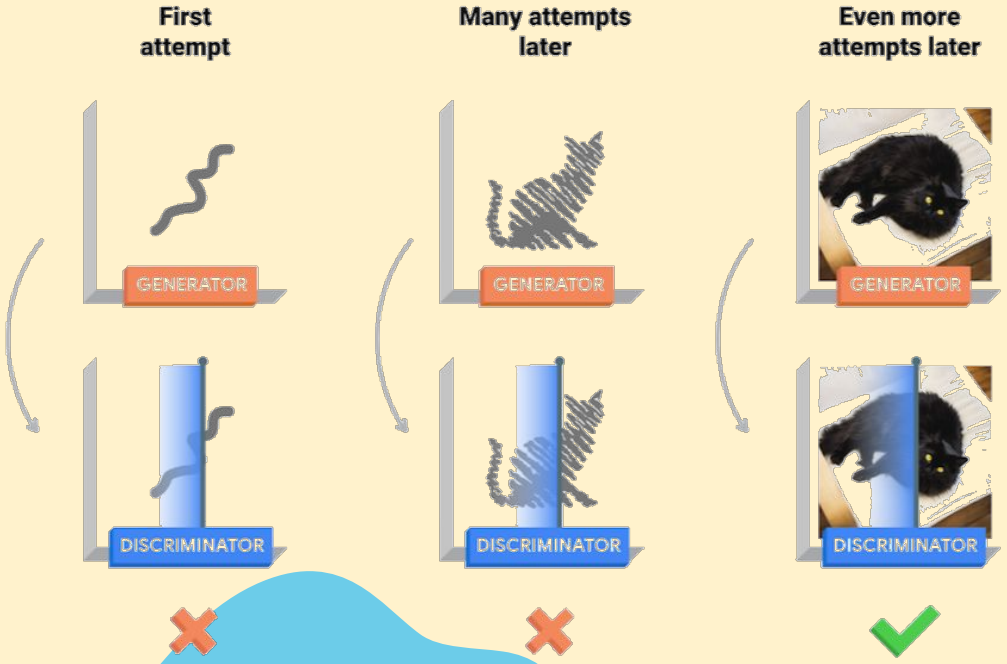
Instance Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

GAN's Generative Adversarial Networks



Feature Visualization

https://distill.pub/2017/feature-visualization/

Ja hem acabat la part 2!

Gràcies a tots!