<https://www.youtube.com/watch?v=aircAruvnKk>

Cada imagen es de 28x28 pixels

Se clasifica la imagen para escoger entre 10 numeros

Una red de 2 capas

PRIMERA CAPA:

DE ENTRADA 28X28 = 784 NODOS, por cada pixel

Activacion:ReLu

CAPA DE SALIDA:

10 NODOS, POR SER 10 NUMEROS POSIBLES

Activacion: softmax

Funcion de error: categorical crossentropy: por ser de clasificacion

Optimizer: rmsprop

import numpy as np

import matplotlib.pyplot as plt

#modelo red neural

from keras import layers, models

from tensorflow.keras.utils import to\_categorical

from keras.datasets import mnist

#Cargar los datos y separarlos en datos de entrenamiento y prueba.

(train\_data, train\_labels), (test\_data, test\_labels) = mnist.load\_data()

train\_data.shape

#6000, 28,28 son 60000 imagenes de 28x28'

def ImagenNumero(pos)

# pos = 0 da numero 5

#para confirmar el numero '5' de imagen

train\_labels[pos]

#ver imagen de posicion

plt.imshow(train\_data[pos])

model = models.Sequential() #Modelo red neural

neuronas=512 #numero de neuronas

#PRIMER CAPA

#512 neuronas con activacion ReLu, tamano de imagen 28x28

# 28X28 = 784 nodos de entrada

model.add(layers.Dense(neuronas, activation='relu', input\_shape = (28\*28,)))

#CAPA SALIDA

#son 10 posibles salidas, de 0..9 , Fx = softmax

model.add(layers.Dense(10,activation='softmax'))

model.compile(optimizer='rmsprop',

loss='categorical crossentropy',

metrics = 'accuracy')

model.summary()

RESUMEN

========

PRIMERA CAPA:

DE ENTRADA 28X28 = 784 NODOS, por cada pixel

Activacion:ReLu

CAPA DE SALIDA:

10 NODOS, POR SER 10 NUMEROS POSIBLES

Activacion: softmax

Funcion de error: categorical crossentropy: por ser de clasificacion

Optimizer: rmsprop

= = = = = = = = = = = = = = = = = = = = = = = = = = =

#Transformacion, en vez de 3 dimensiones a 2

#Pasa valores a un valor menor, de 1 - muchos decimales

# en train y en test

x\_train = train\_data.reshape((60000, 28\*28))

x\_train = x\_train.astype('float32')/255

x\_test = test\_data.reshape((10000, 28\*28))

x\_test = x\_test.astype('float32')/255

#En vez de valores 0..9 tiene arreglos

#Si es 5 es 0,0,0,0,0,1,0,0,0,0

y\_train = to\_categorical(train\_labels)

y\_test = to\_categorical(test\_labels)

model.fit(x\_train, y\_train, epochs=5, batch\_size=128)

#Evaluar si datos de test son muy seguros de ser reconocidos

model.evaluate(x\_test, y\_test)

1. Neurona: función lineal que recibe muchos parámetros, en forma a*x1+b*x2+…+B (x y B son parámetros que la red modifica para entrenar. B es conocido como bias o termino independiente) que lo que nuestra neurona de entrada recibe es el el arreglo de 28\*28 que configuramos.
2. Capas densas: son una serie de neuronas aplicas entre si, en este caso tenemos 2 capas, una con 523 neuronas y otra con 10.
3. la función de activación: elimina datos no relevantes además de conferir cierto grado de no linealidad que es bueno en general.  
   **Parametros para entrenar con model.compile**
4. optimizador o optimizer: modifica los valores X1, X2, …B, de las redes neuronales para disminuir la función de perdida.
5. Función de perdida o loss: en pocas palabras mide cuan lejos está el valor estimadopor la red nueronal del valor real, siempre se busca disminuir lo máximo antes de caer en overfitting.  
   3.metrics: con esta medimos la precisión del modelo, en este caso accuray mide cuantos predicciones acertadas se han dado sobre el total de predicciones, pero hay muchas mas me´tricas.