**ACTIVAR GTU, SINO MUY LENTO**

import tensorflow as tf

from tensorflow.keras.datasets import fashion\_mnist

from tensorflow.keras.layers import Conv2D, Dropout, MaxPooling2D, Flatten, Dense

from tensorflow.keras.utils import to\_categorical

import numpy as np

import matplotlib.pyplot as plt

#LOAD DATA

(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

train\_images = train\_images.astype('float32') / 255

test\_images = test\_images.astype('float32') / 255

train\_images = train\_images.reshape(train\_images.shape[0], 28, 28, 1)

test\_images = test\_images.reshape(test\_images.shape[0], 28, 28, 1)

train\_labels = tf.keras.utils.to\_categorical(train\_labels, 10)

test\_labels = tf.keras.utils.to\_categorical(test\_labels, 10)

**#SI ES GRIS 28,28,1**

**#SI ES COLOR 28,28,3**

**#SI ES AMPLITUD 28,28,4**

from tensorflow.keras import regularizers

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(75, (3,3), activation = "relu", input\_shape= (28, 28, 1)),

tf.keras.layers.MaxPool2D((2,2)),

tf.keras.layers.**Flatten(),**

**#28 x 28 = 784**

tf.keras.layers.Dense(256, kernel\_regularizer = regularizers.l2(1e-5), activation = "relu"),

tf.keras.layers.**Dropout(0.2),**

tf.keras.layers.Dense(128, kernel\_regularizer= regularizers.l2(1e-5),activation = "relu"),

tf.keras.layers.**Dropout(0.2),**

tf.keras.layers.Dense(10, activation = "softmax")

])

from tensorflow import keras

model.compile(loss='categorical\_crossentropy',

# optimizer='rmsprop',

optimizer = keras.optimizers.Adam(learning\_rate=0.001),

metrics=['accuracy'])

from tensorflow.keras.callbacks import Callback

class TrainingCallback(Callback):

def on\_epoch\_end(self, epoch, logs = {}):

if logs.get("accuracy") > 0.95:

print("Lo logramos, nuestro modelo llego a 95%, detenemos entrenamiento "+str(epoch))

self.model.stop\_training = True

callback = TrainingCallback()

history=model.fit(train\_images,

train\_labels,

batch\_size=64,

callbacks = [callback],

epochs=40)

score = model.evaluate(test\_images, test\_labels, verbose=0)

print(score)

import pandas as pd

pd.DataFrame(history.history).plot(figsize=(10,8));

934/938 [============================>.] - ETA: 0s - loss: 0.1904 - accuracy: 0.9508Lo logramos, nuestro modelo llego a 95%, detenemos entrenamiento 8

938/938 [==============================] - 5s 5ms/step - loss: 0.1905 - accuracy: 0.9509

[0.31086140871047974, 0.9179999828338623]

Epochs

2 a 90%

8 a 95%

15 a 98%

Se logra 95%

* Si tf.keras.layers.**Dropout(0.4),**  en 13 epochs tarda mas en vez de 0.2
* Si  tf.keras.layers.Dropout(0.1), en 7 epochs tarda menos
* Si  tf.keras.layers.Dropout(0.01), en 6 epochs tarda menos

Variar

* Dropout(0.2)
* kernel\_regularizer = regularizers.l2(1e-5)
* optimizer = keras.optimizers.Adam(learning\_rate=0.001),
  + optimizer='rmsprop',