<https://www.kaggle.com/alarcon7a/mi-primera-red-neuronal-convolucional>

Active runtime GPU

Importando librerías

import tensorflow as tf

from tensorflow.keras.datasets import fashion\_mnist

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

*# Helper libraries*

import numpy as np

import matplotlib.pyplot as plt

Cargando dataset de Fashion MNIST, imágenes de ropa

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

train\_images.shape

Out[3]:

60,000 imágenes de 28x28 de ROPA

(60000, 28, 28)

In [4]:

Ver imagen no. 99

plt.imshow(train\_images[99])

Out[4]:

A picture containing text, clock

Description automatically generated

0s

train\_images[99]

array([[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 57,

78, 87, 47, 25, 4, 0, 0, 0, 1, 0, 0, 0, 0,

0, 0],

[ 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 12, 154, 123,

33, 57, 48, 82, 118, 56, 0, 0, 1, 0, 0, 0, 0,……

La imagen 99, esta hecha de 28 x 28, y en cada celda pueden tener valores de 0 .. 255

## Transformacion de datos

Los valores iniciales son de 0 – 255, se cambia porque se maneja mejor con decimales que con valores mayores a 1. Antes valores enteros y los pasa a float

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

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

## Nuevos Valores, a decimales

train\_images[99][27][27]

[0. , 0. , 0. , 0. , 0. ,

0. , 0. , 0. , 0. , 0.01176471,

0. , 0. , 0.22352941, 0.30588236, 0.34117648,

0.18431373, 0.09803922, 0.01568628, 0. , 0. ,…….

**Canal es 1 porque son todos de grises, si fuera a colores será 3 en vez de 1**

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

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

## Etiquetas

train\_labels.shape

60000,

train\_labels

60,000 valores del 0 al 9.

array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)

## Transformación

## En vez de 7 sera 0,0,0,0,0,0,0,1,0,0

## En vez de 2 sera 0,0,1,0,0,0,1,0,0,0

## En vez de un arreglo de 60,000 sera uno de 60,000 x 10

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

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

train\_labels

array([0., 0., 0., 0., 0., 0., 0., 0., 1., 0.], dtype=float32)

## Creando red neurona

model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network

**#capa de convolución, es 28,28,1 por ser tonos grises, sino 28,28,3**

model.add(Conv2D(filters=64, kernel\_size=2, padding='same',

activation='relu', input\_shape=(28,28,1)))

model.add(MaxPooling2D(pool\_size=2))

model.add(Dropout(0.3))

model.add(Conv2D(filters=32, kernel\_size=2, padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=2))

model.add(Dropout(0.3))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

**#10 tipos de ropa**

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

**# Take a look at the model summary**

model.summary()

## Compilando la red neuronal

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

             optimizer='rmsprop',

             metrics=['accuracy'])

## Entenando la red

model.fit(train\_images,

train\_labels,

batch\_size=64,

epochs=10)

## Análisis de Resultados

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

score

[0.2863689064979553, 0.8980000019073486]

## CON OTRA FUNCION DE PERDIDA

## Compilando la red neuronal

from keras.losses import SparseCategoricalCrossentropy

model.compile(loss='SparseCategoricalCrossentropy(from\_logits=True)),

             optimizer='rmsprop',

             metrics=['accuracy'])

## Entenando la red

model.fit(train\_images,

train\_labels,

batch\_size=64,

epochs=10)

## 0.2954 0.903 <- MEJORO PRESICION DE 0.898

## Callbacks

early = tf.keras.callbacks.EarlyStopping(monitor='accuracy',patience=1)

model.fit(train\_images,

         train\_labels,

         batch\_size=64,

        callbacks=[early],

         epochs=10)

checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='mi\_primer\_red\_conv.hdf5',

                                verbose=1,

                                  monitor='accuracy',

                                  save\_best\_only=True)

model.fit(train\_images,

         train\_labels,

         batch\_size=64,

         callbacks=[checkpoint],

         epochs=10)

model2 = model

model2.load\_weights('./mi\_primer\_red\_conv.hdf5')

model2.evaluate(test\_images, test\_labels)

[0.6963484287261963, 0.8723000288009644]