**Clasificación de imágenes de ropa con codelab**

a. Importar el set de datos de moda de MNIST, explore el set de datos

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

fashion\_mnist = keras.datasets.fashion\_mnist # importar dataset

(x\_train, y\_train), (x\_test, y\_test) = fashion\_mnist.load\_data() # explorar el ds: (images, labels)

class\_names = ['T-shirt/top', 'Trouser','Pullover','Dress','Coat','Sandal','Shirt','Sneaker','Bag','Ankle boot'] # 10 clases

print(x\_train.shape) # arrays de imagenes

print(x\_test.shape)

print(y\_train) # array de etiquetas

print(y\_test)

# Escalar valores de imagenes en un rango de 0 a 1

x\_train = x\_train / 255.0

x\_test = x\_test / 255.0

b. Pre-procese el set de Datos, Construir el modelo (configurar las capas, compile el modelo).

print(len(x\_train))  # filas

print(len(x\_train[0])) # columnas

col = len(x\_train[0]) # 28

plt.figure(figsize=(10,10))

for i in range(0, col):

  plt.subplot(7,4,i+1) # distribuir el total de modelos un grid de 7 filas, 4 columnas

  plt.grid(False)

  plt.imshow(x\_train[i], cmap=plt.cm.binary)

plt.show()

# configurar las capas

model= keras.Sequential([

  keras.layers.Flatten(input\_shape=(col,col)), # primera capa que transforma el arreglo bidimensional en unidimensional en valores [0,1]

  keras.layers.Dense(128, activation='relu'), # 128 neuronas de la primera capa

  keras.layers.Dense(10, activation='softmax') # 10 tipos de datos, cada uno tendra una clasificacion

 ])

# compilacion

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

c. Entrenar el modelo, evaluar exactitud.

Se justa el modelo a el set de datos de entrenamiento:

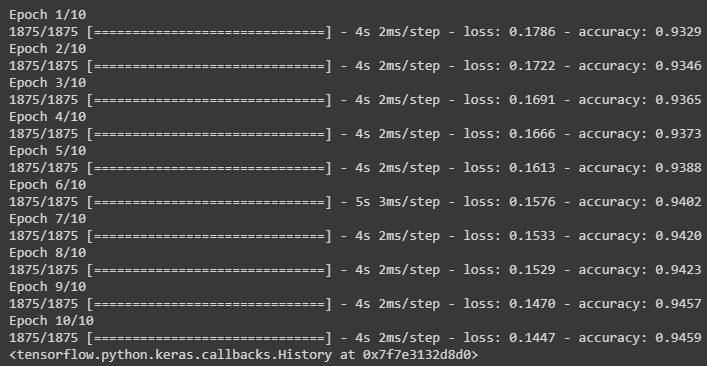
model.fit(x\_train, y\_train, epochs=10) # evaluar(imagenes, etiquetas)) de entrenamiento

test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=2) # evaluar(imagenes, etiquetas)) de test

print('Test accuracy:',test\_acc)

# el modelo alcanza una exactitud de %88.8

Comparamos el rendimiento del modelo sobre el set de datos:



El modelo alcanza una exactitud de %88.5 (Promediando los accuracy de las 10 Epoch)

La exactitud del set de entrenamiento resulta de % 88.8. Ocurre el sobreajuste: cuando un modelo de aprendizaje de maquina (ML) tiene un rendimiento peor sobre un set de datos nuevo, que nunca antes ha visto comparado con el de entrenamiento.



d. Hacer predicciones.

# una prediccion es un arreglo de 10 numeros. Estos representan el nivel de "confianza" del modelo

# sobre las imagenes de cada uno de los 10 articulos de moda/ropa.

predictions = model.predict(x\_test)

 # para la prediccion 0, la que tiene el nivel mas alto de confianza: bota de talon

print(class\_names[np.argmax(predictions[0])])

# para la prediccion 1, la que tiene el nivel mas alto de confianza: sueter

print(class\_names[np.argmax(predictions[1])])

Para la predicción 0, la que tiene el nivel más alto de confianza: bota de talón

Para la predicción 1, la que tiene el nivel más alto de confianza: sueter

