Carlos Fernández de la Torre

Trabajo 1: Introducción a las redes neuronales con TensorFlow y Keras

En este primer trabajo, vamos a utilizar una red neuronal para clasificar imágenes de prendas de ropa. Para ello, utilizaremos Keras con TensorFlow.

El dataset a utilizar es Fashion MNIST, un problema sencillo con imágenes pequeñas de ropa, pero más interesante que el dataset de MNIST. Puedes consultar más información sobre el dataset en este enlace.

El código utilizado para contestar tiene que quedar claramente reflejado en el Notebook. Puedes crear nuevas cells si así lo deseas para estructurar tu código y sus salidas. A la hora de entregar el notebook, **asegúrate de que los resultados de ejecutar tu código han quedado guardados** (por ejemplo, a la hora de entrenar una red neuronal tiene que verse claramente un log de los resultados de cada epoch).

```
import keras
from keras.datasets import fashion_mnist
from keras.models import Sequential
from keras.layers import Dense
import tensorflow as tf

import matplotlib.pyplot as plt
```

Primero, vamos a obtener los datos. Por suerte para nosotros, estos pueden ser descargados directamente desde Keras, por lo que no tendremos que preocuparnos de tratar con ficheros.

```
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

Acto seguido, normalizamos esos datos de manera similar a como hemos visto con MNIST, obteniendo valores entre 0 y 1. Este paso es muy importante para el correcto funcionamiento de nuestra red.

```
x_train = x_train / 255.0
x_test = x_test / 255.0
```

1. Información sobre el dataset

Una vez tenemos los datos cargados en memoria, vamos a obtener información sobre los mismos.

** Pregunta 1.1 *(0.5 puntos) * ¿Cuántas imágenes hay de *training* y de *test*? ¿Qué tamaño tienen las imágenes?

```
### Tu código aquí ###
print(x_train.shape)
### ¿Cuántas imágenes hay de *training y de test? ###
print(x_train.shape[0],'training samples')
print(x_test.shape[0],"test samples")
### ¿Qué tamaño tienen las imágenes? ###
print(x_train.shape[1],'x',x_train.shape[2],' puntos')
print (type(x_train[99][3][15])) #pixel posición (3,15) de la imagen 100 es un número real
```

```
(60000, 28, 28)
60000 training samples
10000 test samples
28 x 28  puntos
<class 'numpy.float64'>
```

Tu respuesta aquí ¿Cuántas imágenes hay de *training y de test? Training set de 60,000 ejemplos y test set de 10,000 ejemplos. ¿Qué tamaño tienen las imágenes? 28x28 píxeles en escala de grises.

**Pregunta 1.2 *(0.5 puntos)* ** Realizar una exploración de las variables que contienen los datos. Describir en qué consiste un example del dataset (qué información se guarda en cada imagen) y describir qué contiene la información en y.

```
### Tu código aquí ###
print(x_train[7][0:3]) # muestra solo las tres primeras filas de la imagen
# describir qué contiene la información en y
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
print(class_names[y_train[7]])
```

```
[[0.         0.         0.         0.         0.         0.00392157
  0.00392157 0.         0.         0.         0.         0.24705882
  0.10980392 0.         0.         0.         0.12941176 0.33333333
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.00784314
  0.         0.         0.10980392 0.49411765 0.94509804 1.
  1.         1.         1.         1.         1.         0.98823529
  0.97254902 0.43529412 0.         0.         0.         0.00784314
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.00784314 0.
  0.         0.80784314 0.95686275 0.98431373 0.94509804 0.90196078
  0.93333333 0.86666667 0.80392157 0.90196078 0.94117647 0.90196078
  0.9372549  0.98431373 0.91372549 0.64705882 0.         0.
  0.00784314 0.         0.         0.        ]]
Pullover
```

Tu respuesta aquí. ¿Qué información se guarda en cada imagen? Un array de 28*28 con valores entre 0 y 1

La salida **y** contiene un número correspondiente con una prenda de ropa.

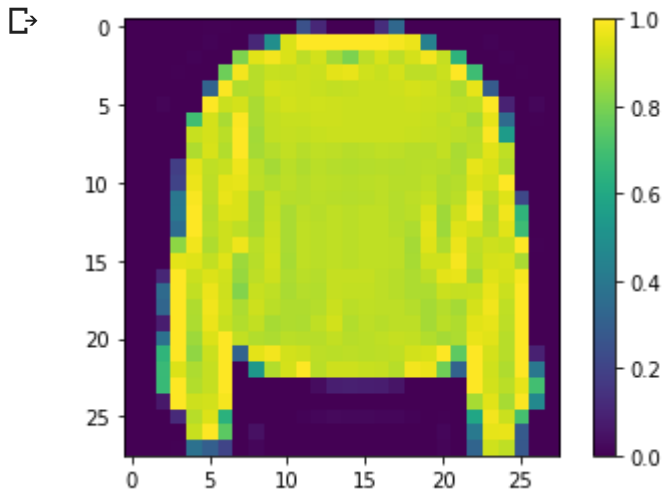
- 0 T-shirt/top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot

Por ejemplo x_train[7] es un Pullover

Vamos a **visualizar** una imagen de ejemplo. Prueba tu mismo a cambiar la imagen en uso para explorar el

```
def visualize_example(x):
    plt.figure()
    plt.imshow(x)
    plt.colorbar()
    plt.grid(False)
    plt.show()


visualize_example(x_train[7])
```



2. Entrenamiento de una red neuronal simple

Pregunta 2 ***(7 puntos)***. Utilizando Keras, y preparando los datos de X e y como fuera necesario, define y entrena una red neuronal que sea capaz de clasificar imágenes de Fashion MNIST con las siguientes características:

- Dos hidden layers de tamaños 128 y 64, utilizando unidades **sigmoid**
- Optimizador **sgd**.
- Durante el entrenamiento, la red tiene que mostrar resultados de **loss** y **accuracy** por cada epoch.
- La red debe entrenar durante **20 epochs** y batch size de **64**.
- La última capa debe de ser una capa **softmax**.

Tu red tendría que ser capaz de superar fácilmente 60% de accuracy.

```
### Tu código aquí ###
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='sigmoid'),
    keras.layers.Dense(64, activation='sigmoid'),
    keras.layers.Dense(10, activation='softmax')
])
model.summary()

model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=20,validation_data=(x_test,y_test))
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_10 (Flatten)         (None, 784)               0         
_________________________________________________________________
dense_28 (Dense)             (None, 128)               100480    
_________________________________________________________________
dense_29 (Dense)             (None, 64)                8256      
_________________________________________________________________
dense_30 (Dense)             (None, 10)                650       
=================================================================
Total params: 109,386
Trainable params: 109,386
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
60000/60000 [==============================] - 7s 121us/step - loss: 2.0061 - acc:
Epoch 2/20
60000/60000 [==============================] - 6s 104us/step - loss: 1.3659 - acc:
Epoch 3/20
60000/60000 [==============================] - 6s 99us/step - loss: 1.0309 - acc: 0
Epoch 4/20
60000/60000 [==============================] - 6s 99us/step - loss: 0.8548 - acc: 0
Epoch 5/20
60000/60000 [==============================] - 6s 93us/step - loss: 0.7543 - acc: 0
Epoch 6/20
60000/60000 [==============================] - 6s 94us/step - loss: 0.6936 - acc: 0
Epoch 7/20
60000/60000 [==============================] - 5s 90us/step - loss: 0.6523 - acc: 0
Epoch 8/20
60000/60000 [==============================] - 5s 90us/step - loss: 0.6212 - acc: 0
Epoch 9/20
60000/60000 [==============================] - 5s 89us/step - loss: 0.5955 - acc: 0
Epoch 10/20
60000/60000 [==============================] - 5s 88us/step - loss: 0.5735 - acc: 0
Epoch 11/20
60000/60000 [==============================] - 5s 91us/step - loss: 0.5547 - acc: 0
Epoch 12/20
60000/60000 [==============================] - 5s 90us/step - loss: 0.5380 - acc: 0
Epoch 13/20
60000/60000 [==============================] - 5s 90us/step - loss: 0.5232 - acc: 0
Epoch 14/20
60000/60000 [==============================] - 5s 89us/step - loss: 0.5106 - acc: 0
Epoch 15/20
60000/60000 [==============================] - 5s 90us/step - loss: 0.4994 - acc: 0
Epoch 16/20
60000/60000 [==============================] - 5s 91us/step - loss: 0.4895 - acc: 0
Epoch 17/20
60000/60000 [==============================] - 5s 90us/step - loss: 0.4808 - acc: 0
Epoch 18/20
60000/60000 [==============================] - 5s 86us/step - loss: 0.4732 - acc: 0
Epoch 19/20
60000/60000 [==============================] - 5s 89us/step - loss: 0.4661 - acc: 0
Epoch 20/20
60000/60000 [==============================] - 5s 87us/step - loss: 0.4598 - acc: 0
<keras.callbacks.History at 0x7f9152fd2898>
```

3. Evaluación del modelo en datos de test

Una vez hemos entrenado nuestro modelo, vamos a evaluarlo en los datos de test de Fashion MNIST.

*Pregunta 3.1 *(1 punto)* **. Utilizando el modelo recién entrenado, obtener la accuracy resultante en el dataset de test.

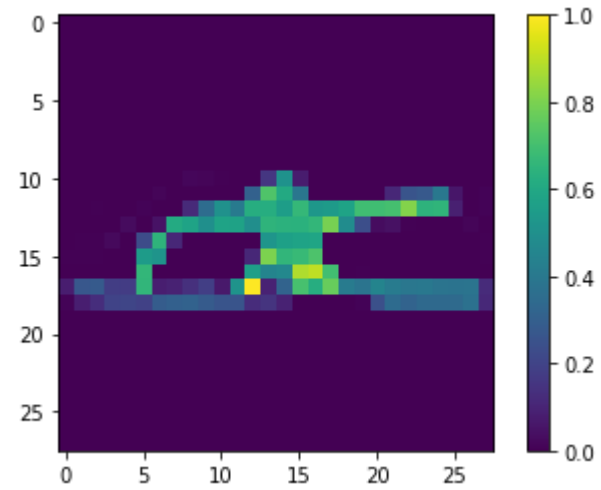
```
### Tu código aquí ###
score=model.evaluate(x_test,y_test,verbose=0)
print("Test loss:", score[0])
print("Test accuracy:",score[1])
```

```
Test loss: 0.4879085629224777
Test accuracy: 0.821
```

Pregunta 3.2 *(1 punto)*. Utilizando el método **predict** de Keras, realizar predicciones para los datos de test. Por cada predicción resultante, ¿qué significan los números que obtenemos al hacer predict? ¿Cómo podemos obtener el valor de la clase resultante? (recordar que estamos utilizando una capa softmax para clasificar).

```
### Tu código aquí ###
import numpy as np
predictions=model.predict(x_test)
np.argmax(predictions[0])

visualize_example(x_test[52])
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
print(class_names[np.argmax(predictions[52])])
```




```
Sandal
```

Tu respuesta aquí ** ¿qué significan los números que obtenemos al hacer predict?** Son un vector de 10 elementos, números entre 0 y 1 (por cada predicción) siendo el índice del vector cuyo valor es más cercano a uno el que el modelo predice o clasifica con mayor probabilidad. **¿Cómo podemos obtener el valor de la clase resultante?** Usando la función **argmax** que devuelve el índice del vector de 10 elementos a los que se les ha asignado una "probabilidad",en el ejemplo la imagen 52 el modelo predice que es una sandalia con la mayor probabilidad.