---

**Descripción del Código de clasificación con el dataset Fashion MNIST utilizando TensorFlow** y **Keras**.

## 1. Importación de Librerías

```
python
Copiar código
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
```

- **`keras`**: Biblioteca de alto nivel para construir y entrenar redes neuronales.
- **`fashion_mnist`**: Dataset con imágenes de prendas de ropa, similar a MNIST pero más complejo.
- **`Sequential`**: API para crear modelos de redes neuronales capa por capa.
- **`Dense`**: Capa completamente conectada.
- **`matplotlib.pyplot`**: Para visualizar imágenes y gráficos.

---

## 2. Carga del Dataset

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

- **`x_train`** y **`y_train`**: Imágenes y etiquetas para el conjunto de entrenamiento (60,000 ejemplos).
- **`x_test`** y **`y_test`**: Imágenes y etiquetas para el conjunto de prueba (10,000 ejemplos).

---

## 3. Normalización de Datos

```
python
Copiar código
x_train = x_train / 255.0
x_test = x_test / 255.0
```

- Las imágenes son de 28x28 píxeles con valores entre 0 y 255. Dividir por 255 normaliza los datos para que estén entre 0 y 1, lo cual mejora el entrenamiento de la red.

---

## 4. Exploración del Dataset

```
python
Copiar código
print(x_train.shape)  # (60000, 28, 28)
print(x_train.shape[0], 'training samples')  # 60000 training samples
print(x_test.shape[0], "test samples")  # 10000 test samples
print(x_train.shape[1], 'x', x_train.shape[2], 'puntos')  # 28 x 28
puntos
```

Esto muestra la cantidad de imágenes y el tamaño de cada imagen (28x28).

---

## 5. Construcción del Modelo

```
python
Copiar código
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),  # Aplana la imagen de
28x28 a un vector de 784 elementos
    keras.layers.Dense(128, activation='sigmoid'),  # Capa oculta con
128 neuronas y activación sigmoide
    keras.layers.Dense(64, activation='sigmoid'),  # Segunda capa
oculta con 64 neuronas y activación sigmoide
    keras.layers.Dense(10, activation='softmax')  # Capa de salida con
10 neuronas para las 10 clases
])
model.summary()
```

- **Capa `Flatten`**: Convierte la imagen 2D en un vector 1D.
- **Capas `Dense`**: Neuronas conectadas completamente. Se usa **activación sigmoide** en las capas ocultas.
- **Capa `softmax`**: Convierte las salidas en probabilidades para cada clase (10 clases).

---

## 6. Compilación del Modelo

```
python
Copiar código
model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

- **`optimizer='sgd'`**: Utiliza el optimizador **Stochastic Gradient Descent**.
- **`loss='sparse_categorical_crossentropy'`**: Función de pérdida adecuada para clasificación multiclase con etiquetas enteras (0 a 9).
- **`metrics=['accuracy']`**: Métrica para evaluar el rendimiento del modelo.

---

## 7. Entrenamiento del Modelo

```
python
```

```
Copiar código
model.fit(x_train, y_train, epochs=20, validation_data=(x_test,
y_test))
```

- **`epochs=20`**: Entrena por 20 iteraciones completas sobre el conjunto de datos.
- **`batch_size=64`** (por defecto): Tamaño del lote utilizado en cada iteración.

---

## 8. Evaluación del Modelo

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

- Calcula la pérdida y precisión del modelo en el conjunto de prueba.

---

## 9. Predicciones

```
python
Copiar código
predictions = model.predict(x_test)
import numpy as np
print(np.argmax(predictions[0]))  # Índice de la clase predicha con
mayor probabilidad
```

- **`predict`** devuelve un vector de 10 probabilidades.
- **`argmax`** selecciona el índice con la mayor probabilidad.

---

## 10. Visualización de Ejemplos

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

visualize_example(x_train[7])
```

- Muestra una imagen del conjunto de entrenamiento para inspección visual.

---

# Conclusión

Este código implementa una red neuronal sencilla para clasificar imágenes de ropa utilizando el dataset Fashion MNIST. La estructura de dos capas ocultas con activación sigmoide permite que el modelo alcance una precisión razonable, generalmente superior al **60%** en el conjunto de prueba.