In [2]:
```
import tensorflow as tf
import numpy as np
import pandas as pd
from tensorflow.keras.datasets import fashion_mnist
```

In [3]:
```
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

X_train = X_train.reshape((-1, 28, 28, 1))
X_train = X_train / 255.0
X_test = X_test.reshape((-1, 28, 28, 1))
X_test = X_test / 255.0
```

In [4]:
```
batch_size = 8
n_epochs = 20
learn_rate = 0.0001
```

In [5]:
```
model = tf.keras.Sequential()
model.add(tf.keras.Input(shape=(28, 28, 1)))
model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

In [6]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learn_rate),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [7]:
```
model.fit(X_train, y_train, batch_size=batch_size, epochs=n_epochs, validation_data
Epoch 1/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 33s 4ms/step - accuracy: 0.7307 - loss: 0.7734 - val_
accuracy: 0.8556 - val_loss: 0.3929
Epoch 2/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 29s 4ms/step - accuracy: 0.8726 - loss: 0.3574 - val_
accuracy: 0.8786 - val_loss: 0.3264
Epoch 3/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 31s 4ms/step - accuracy: 0.8908 - loss: 0.3055 - val_
accuracy: 0.8823 - val_loss: 0.3172
Epoch 4/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 30s 4ms/step - accuracy: 0.9015 - loss: 0.2734 - val_
accuracy: 0.8985 - val_loss: 0.2770
Epoch 5/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 33s 4ms/step - accuracy: 0.9078 - loss: 0.2508 - val_
accuracy: 0.8993 - val_loss: 0.2803
Epoch 6/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9197 - loss: 0.2261 - val_
accuracy: 0.9054 - val_loss: 0.2659
Epoch 7/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9207 - loss: 0.2191 - val_
accuracy: 0.9011 - val_loss: 0.2652
Epoch 8/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9275 - loss: 0.1999 - val_
accuracy: 0.9094 - val_loss: 0.2514
Epoch 9/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9311 - loss: 0.1881 - val_
accuracy: 0.9084 - val_loss: 0.2528
Epoch 10/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 33s 4ms/step - accuracy: 0.9381 - loss: 0.1736 - val_
accuracy: 0.9133 - val_loss: 0.2418
Epoch 11/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 33s 4ms/step - accuracy: 0.9399 - loss: 0.1657 - val_
accuracy: 0.9135 - val_loss: 0.2432
Epoch 12/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9435 - loss: 0.1534 - val_
accuracy: 0.9177 - val_loss: 0.2299
Epoch 13/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9496 - loss: 0.1414 - val_
accuracy: 0.9171 - val_loss: 0.2365
Epoch 14/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9515 - loss: 0.1336 - val_
accuracy: 0.9166 - val_loss: 0.2402
Epoch 15/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9554 - loss: 0.1254 - val_
accuracy: 0.9113 - val_loss: 0.2565
Epoch 16/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9573 - loss: 0.1159 - val_
accuracy: 0.9175 - val_loss: 0.2462
Epoch 17/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 33s 4ms/step - accuracy: 0.9612 - loss: 0.1083 - val_
accuracy: 0.9164 - val_loss: 0.2593
Epoch 18/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 33s 4ms/step - accuracy: 0.9653 - loss: 0.0985 - val_
accuracy: 0.9172 - val_loss: 0.2486
Epoch 19/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 32s 4ms/step - accuracy: 0.9668 - loss: 0.0915 - val_
```

```
accuracy: 0.9184 - val_loss: 0.2582
Epoch 20/20
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 30s 4ms/step - accuracy: 0.9685 - loss: 0.0853 - val_
accuracy: 0.9169 - val_loss: 0.2554
```

```
Out[7]: <keras.src.callbacks.history.History at 0x26e7ac427b0>
```

```
In [8]: # Evaluar en test
        test_loss, test_acc = model.evaluate(X_test, y_test)
        print(f"\nPrecisión en datos de prueba: {test_acc:.4f}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9170 - loss: 0.2675

Precisión en datos de prueba: 0.9169
```