In [1]:

```
import keras
from keras.datasets import fashion_mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
```

In [2]:

```
# Load Data
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 4us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 38s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0s/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 7s 1us/step
```

In [3]:

```
# Preprocessing
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')

X_train /= 255
X_test /= 255

n_classes = 10
y_train = keras.utils.to_categorical(y_train, n_classes)
y_test = keras.utils.to_categorical(y_test, n_classes)
```

In [4]:

```
# Design neural network architecture
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='softmax'))
```

```
In [5]: # Model summary
        model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320       
                                                                 
 conv2d_1 (Conv2D)           (None, 24, 24, 64)        18496     
                                                                 
 max_pooling2d (MaxPooling2D  (None, 12, 12, 64)       0         
 )                                                               
                                                                 
 dropout (Dropout)           (None, 12, 12, 64)        0         
                                                                 
 flatten (Flatten)           (None, 9216)              0         
                                                                 
 dense (Dense)               (None, 128)               1179776   
                                                                 
 dropout_1 (Dropout)         (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 10)                1290      
                                                                 
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
```

In [6]:

```
# Configure the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [7]:

```
# Train the model
model.fit(X_train, y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_test, y_test))
```

```
Epoch 1/10
469/469 [==============================] - 96s 200ms/step - loss: 0.5164 - accuracy: 0.8167 - val_loss: 0.3439 - val_accuracy:
0.8759
Epoch 2/10
469/469 [==============================] - 94s 201ms/step - loss: 0.3393 - accuracy: 0.8787 - val_loss: 0.2844 - val_accuracy:
0.8964
Epoch 3/10
469/469 [==============================] - 86s 183ms/step - loss: 0.2920 - accuracy: 0.8943 - val_loss: 0.2706 - val_accuracy:
0.8999
Epoch 4/10
469/469 [==============================] - 81s 173ms/step - loss: 0.2604 - accuracy: 0.9054 - val_loss: 0.2453 - val_accuracy:
0.9094
Epoch 5/10
469/469 [==============================] - 99s 211ms/step - loss: 0.2350 - accuracy: 0.9145 - val_loss: 0.2339 - val_accuracy:
0.9124
Epoch 6/10
469/469 [==============================] - 94s 201ms/step - loss: 0.2171 - accuracy: 0.9195 - val_loss: 0.2312 - val_accuracy:
0.9145
Epoch 7/10
469/469 [==============================] - 92s 196ms/step - loss: 0.2006 - accuracy: 0.9251 - val_loss: 0.2168 - val_accuracy:
0.9228
Epoch 8/10
469/469 [==============================] - 92s 196ms/step - loss: 0.1836 - accuracy: 0.9314 - val_loss: 0.2206 - val_accuracy:
0.9209
```

In [8]:

```
# Evaluate the model
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.20950645208358765
Test accuracy: 0.9273999929428101
```