```
# If this does not work append "tensorflow." before keras.
# example: tensorflow.keras.models
from keras.datasets import fashion_mnist
import numpy as np

(train_x, train_y), (test_x, test_y) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 2us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 9s 0us/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 [==============================] - 1s 0us/step
```

```
# If this does not work append "tensorflow." before keras.
# example: tensorflow.keras.models
from keras.models import Sequential
from keras.layers import Dense, Flatten, MaxPooling2D, Conv2D

model = Sequential()

model.add(Conv2D(filters=64,kernel_size=(3,3),activation='relu',input_
shape=(28, 28, 1)))

# Adding maxpooling layer to get max value within a matrix
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(128, activation = "relu"))
model.add(Dense(10, activation = "softmax"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 flatten (Flatten)           (None, 10816)             0

 dense (Dense)               (None, 128)               1384576
```

```
 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 1,386,506
Trainable params: 1,386,506
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer = 'adam', loss =
'sparse_categorical_crossentropy', metrics = ['accuracy'])

model.fit(train_x.astype(np.float32), train_y.astype(np.float32),
epochs = 5, validation_split = 0.2)
```

```
Epoch 1/5
1500/1500 [==============================] - 37s 24ms/step - loss:
0.9416 - accuracy: 0.8491 - val_loss: 0.3418 - val_accuracy: 0.8820
Epoch 2/5
1500/1500 [==============================] - 37s 25ms/step - loss:
0.2850 - accuracy: 0.8983 - val_loss: 0.3258 - val_accuracy: 0.8886
Epoch 3/5
1500/1500 [==============================] - 37s 25ms/step - loss:
0.2503 - accuracy: 0.9099 - val_loss: 0.3431 - val_accuracy: 0.8873
Epoch 4/5
1500/1500 [==============================] - 39s 26ms/step - loss:
0.2284 - accuracy: 0.9161 - val_loss: 0.3932 - val_accuracy: 0.8736
Epoch 5/5
1500/1500 [==============================] - 38s 25ms/step - loss:
0.2035 - accuracy: 0.9263 - val_loss: 0.3718 - val_accuracy: 0.8905

<keras.callbacks.History at 0x12446cdf310>
```

```
loss, acc = model.evaluate(test_x, test_y)
```

```
313/313 [==============================] - 3s 9ms/step - loss: 0.4053
- accuracy: 0.8792
```

```
labels = ['t_shirt', 'trouser', 'pullover', 'dress', 'coat', 'sandal',
'shirt', 'sneaker',   'bag', 'ankle_boots']

predictions = model.predict(test_x[:1])
```

```
1/1 [==============================] - 0s 91ms/step
```

```
label = labels[np.argmax(predictions)]

import matplotlib.pyplot as plt
print(label)
plt.imshow(test_x[:1][0])
plt.show
```

```
ankle_boots
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```