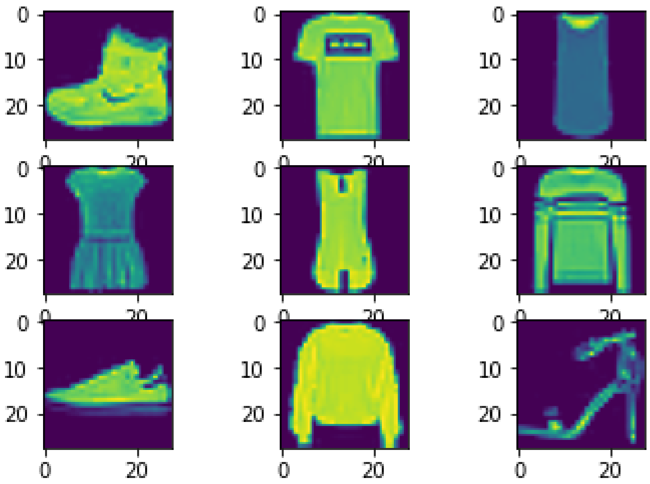
ANN_fashion

```
from keras.datasets import fashion_mnist
import matplotlib.pyplot as plt
```

```
(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
32768/29515 [=================================] - 0s 0us/step
40960/29515 [=========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-
16384/5148 [==================================================================================
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

```
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x_train[i])
plt.show()
```



```
print(x_train.shape,x_test.shape)
```

```
(60000, 28, 28) (10000, 28, 28)
```

```
x = x_test
x_train = x_train.reshape(60000,784)
x_test = x_test.reshape(10000,784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

```
from tensorflow.keras.utils import to_categorical
```

```
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)
```

```
print(x_train.shape, x_test.shape)
```

```
(60000, 784) (10000, 784)
```

```
from keras.models import Sequential
from keras.layers import Activation, Dropout, Dense
from tensorflow.keras.optimizers import RMSprop
model = Sequential()
```

```
model.add(Dense(512,activation='relu',input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='categorical_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
history = model.fit(x_train,y_train,batch_size=128,epochs=100,verbose=1,validation_data=(x_test,y_te
```

```
Epoch 1/100
469/469 [==============================] - 6s 6ms/step - loss: 0.5596 - accuracy: 0.7951 - v
Epoch 2/100
469/469 [==============================] - 3s 6ms/step - loss: 0.4051 - accuracy: 0.8515 - v
Epoch 3/100
469/469 [==============================] - 3s 6ms/step - loss: 0.3669 - accuracy: 0.8660 - v
Epoch 4/100
469/469 [==============================] - 3s 6ms/step - loss: 0.3484 - accuracy: 0.8734 - v
Epoch 5/100
469/469 [==============================] - 3s 6ms/step - loss: 0.3386 - accuracy: 0.8792 - v
Epoch 6/100
469/469 [==============================] - 3s 6ms/step - loss: 0.3270 - accuracy: 0.8821 - v
Epoch 7/100
```

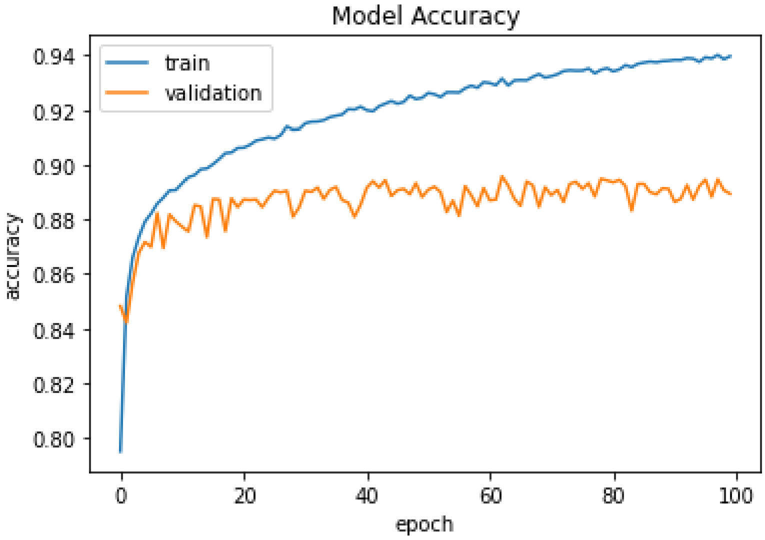
```
469/469 [==============================] - 3s 6ms/step - loss: 0.3205 - accuracy: 0.8858 - v
Epoch 8/100
469/469 [==============================] - 3s 6ms/step - loss: 0.3132 - accuracy: 0.8880 - v
Epoch 9/100
469/469 [==============================] - 3s 6ms/step - loss: 0.3048 - accuracy: 0.8906 - v
Epoch 10/100
469/469 [==============================] - 3s 6ms/step - loss: 0.3055 - accuracy: 0.8907 - v
Epoch 11/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2985 - accuracy: 0.8931 - v
Epoch 12/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2922 - accuracy: 0.8955 - v
Epoch 13/100
469/469 [==============================] - 3s 7ms/step - loss: 0.2924 - accuracy: 0.8963 - v
Epoch 14/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2842 - accuracy: 0.8984 - v
Epoch 15/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2864 - accuracy: 0.8987 - v
Epoch 16/100
469/469 [==============================] - 3s 7ms/step - loss: 0.2809 - accuracy: 0.9003 - v
Epoch 17/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2770 - accuracy: 0.9021 - v
Epoch 18/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2756 - accuracy: 0.9043 - v
Epoch 19/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2761 - accuracy: 0.9045 - v
Epoch 20/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2670 - accuracy: 0.9062 - v
Epoch 21/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2676 - accuracy: 0.9063 - v
Epoch 22/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2676 - accuracy: 0.9074 - v
Epoch 23/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2633 - accuracy: 0.9089 - v
Epoch 24/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2599 - accuracy: 0.9094 - v
Epoch 25/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2600 - accuracy: 0.9100 - v
Epoch 26/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2585 - accuracy: 0.9096 - v
Epoch 27/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2538 - accuracy: 0.9109 - v
Epoch 28/100
469/469 [==============================] - 3s 6ms/step - loss: 0.2569 - accuracy: 0.9140 - v
Epoch 29/100
```

```
score=model.evaluate(x_test, y_test, verbose=1)
print('Test loss =', score)
print('Test accuracy =', score[1])
```

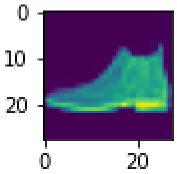
```
313/313 [==============================] - 1s 3ms/step - loss: 0.9888 - accuracy: 0.8894
Test loss = [0.9888068437576294, 0.8894000053405762]
Test accuracy = 0.8894000053405762
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','validation'], loc='upper-left')
```

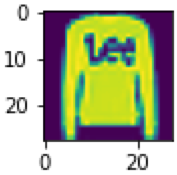
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: MatplotlibDeprecationWarning:
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will raise an exception in 3.3.

<matplotlib.legend.Legend at 0x7fdd9c6a3f10>
```



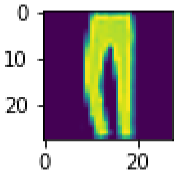
```
import numpy as np
y_pred = model.predict(x_test)
for i in range (9):
  plt.subplot(330+i+1)    # 330 mean: 3 hang 3 cot
  plt.imshow(x[i])
  plt.show()
  print(np.round(y_pred[i]))
```



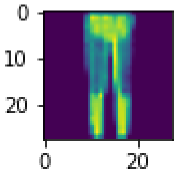
```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```



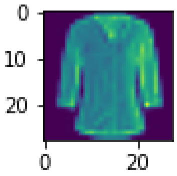
```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```



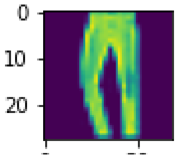
```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```

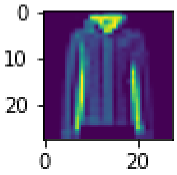


```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```

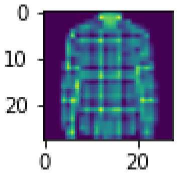


```
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
```

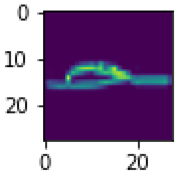




```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```



```
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
```



```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```