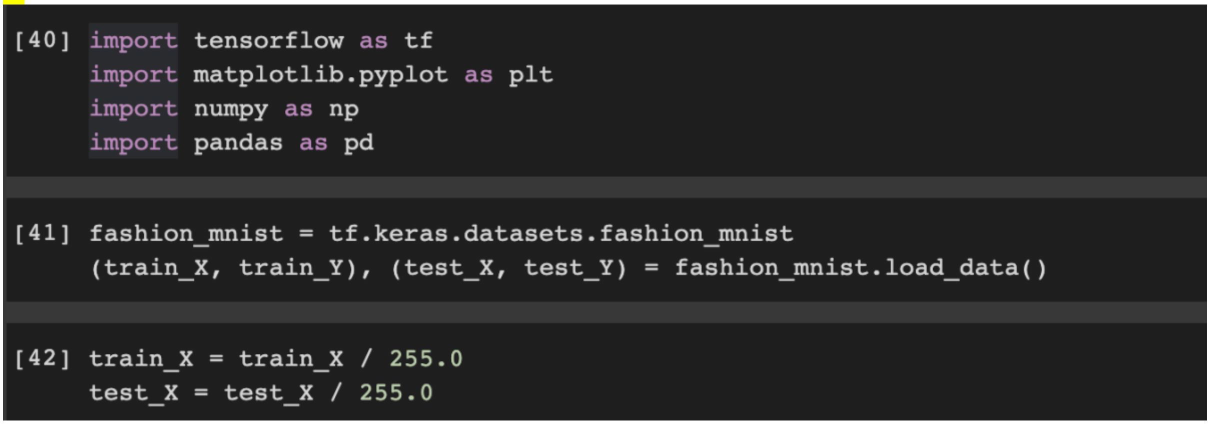
머신비전시스템 과제 6

18011789 조혜수

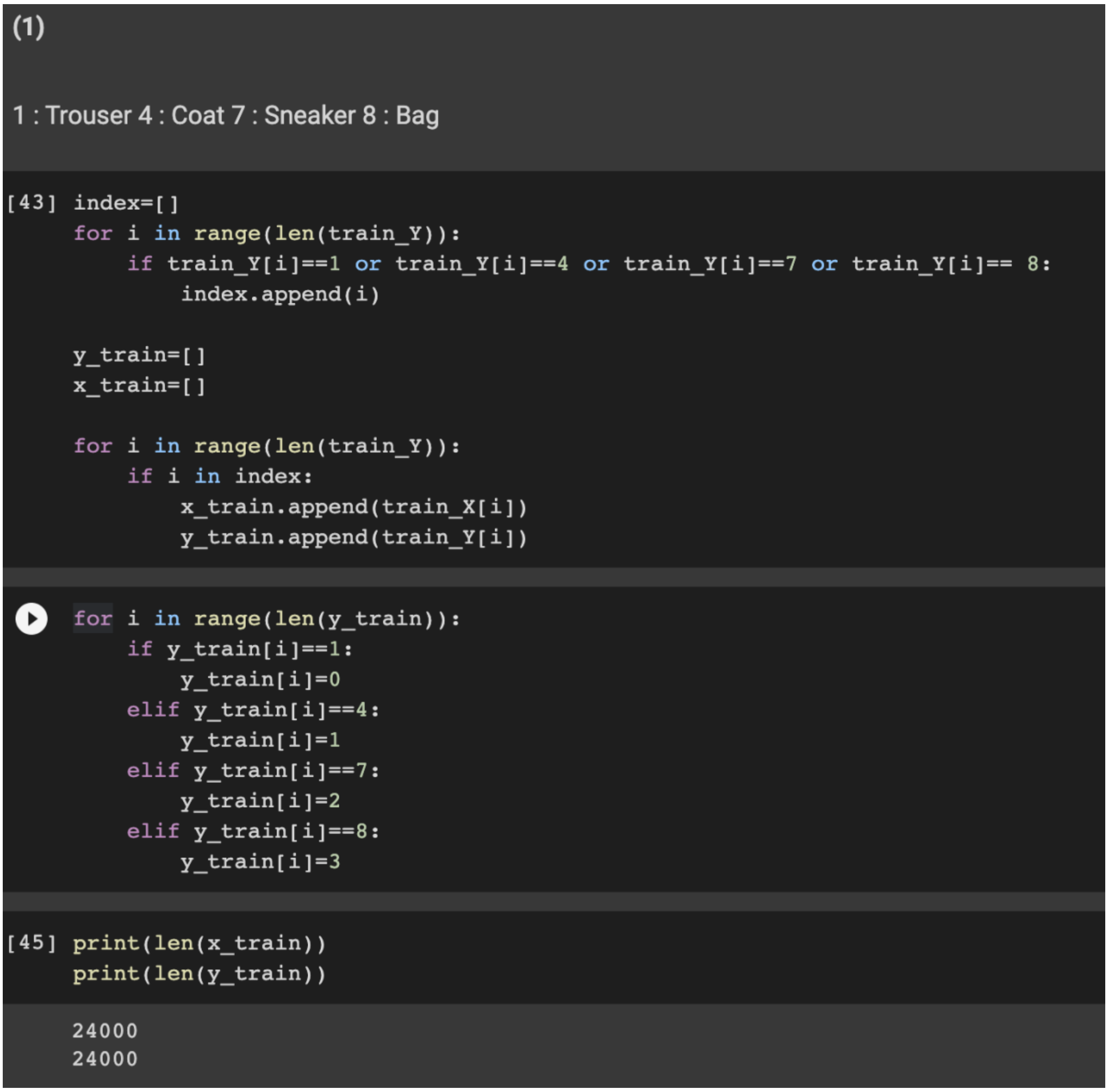
1.



```
[40] import tensorflow as tf
     import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd
```

```
[41] fashion_mnist = tf.keras.datasets.fashion_mnist
     (train_X, train_Y), (test_X, test_Y) = fashion_mnist.load_data()
```

```
[42] train_X = train_X / 255.0
     test_X = test_X / 255.0
```



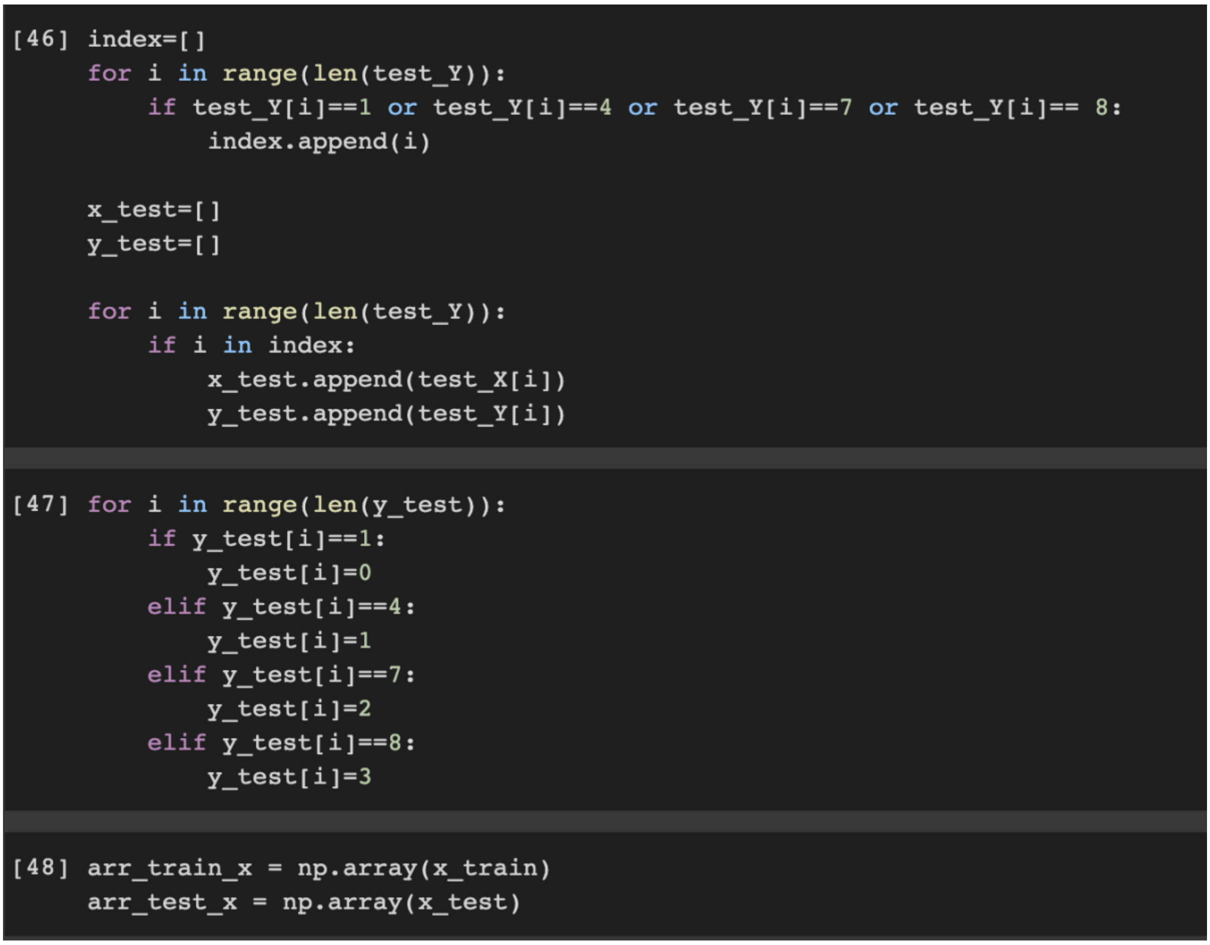
(1)

1 : Trouser 4 : Coat 7 : Sneaker 8 : Bag

```
[43] index=[]
     for i in range(len(train_Y)):
         if train_Y[i]==1 or train_Y[i]==4 or train_Y[i]==7 or train_Y[i]== 8:
             index.append(i)

     y_train=[]
     x_train=[]

     for i in range(len(train_Y)):
         if i in index:
             x_train.append(train_X[i])
             y_train.append(train_Y[i])
```

```
     for i in range(len(y_train)):
         if y_train[i]==1:
             y_train[i]=0
         elif y_train[i]==4:
             y_train[i]=1
         elif y_train[i]==7:
             y_train[i]=2
         elif y_train[i]==8:
             y_train[i]=3
```

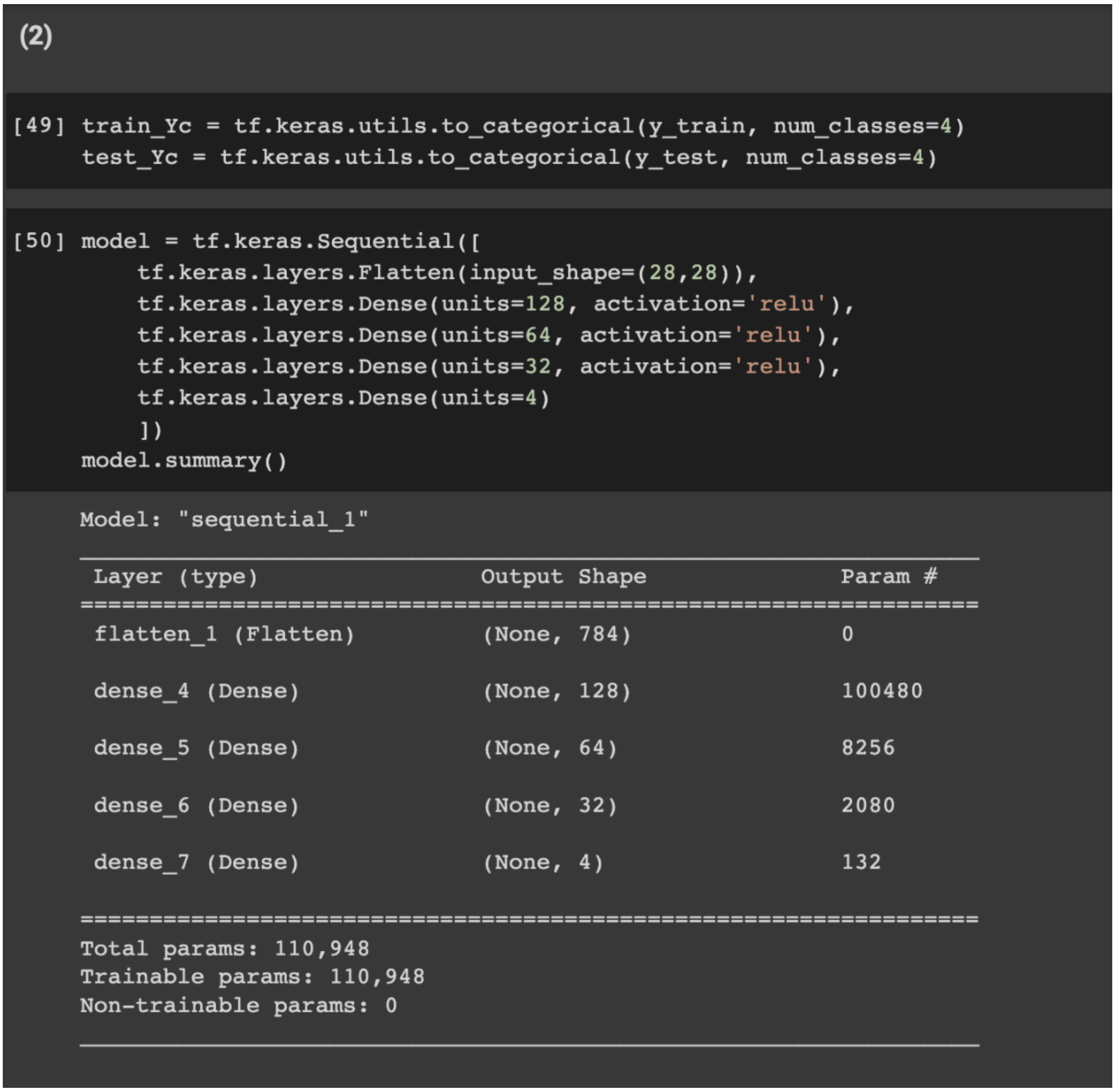
```
[45] print(len(x_train))
     print(len(y_train))

     24000
     24000
```



```
[46] index=[]
     for i in range(len(test_Y)):
         if test_Y[i]==1 or test_Y[i]==4 or test_Y[i]==7 or test_Y[i]== 8:
             index.append(i)

     x_test=[]
     y_test=[]

     for i in range(len(test_Y)):
         if i in index:
             x_test.append(test_X[i])
             y_test.append(test_Y[i])
```

```
[47] for i in range(len(y_test)):
         if y_test[i]==1:
             y_test[i]=0
         elif y_test[i]==4:
             y_test[i]=1
         elif y_test[i]==7:
             y_test[i]=2
         elif y_test[i]==8:
             y_test[i]=3
```

```
[48] arr_train_x = np.array(x_train)
     arr_test_x = np.array(x_test)
```

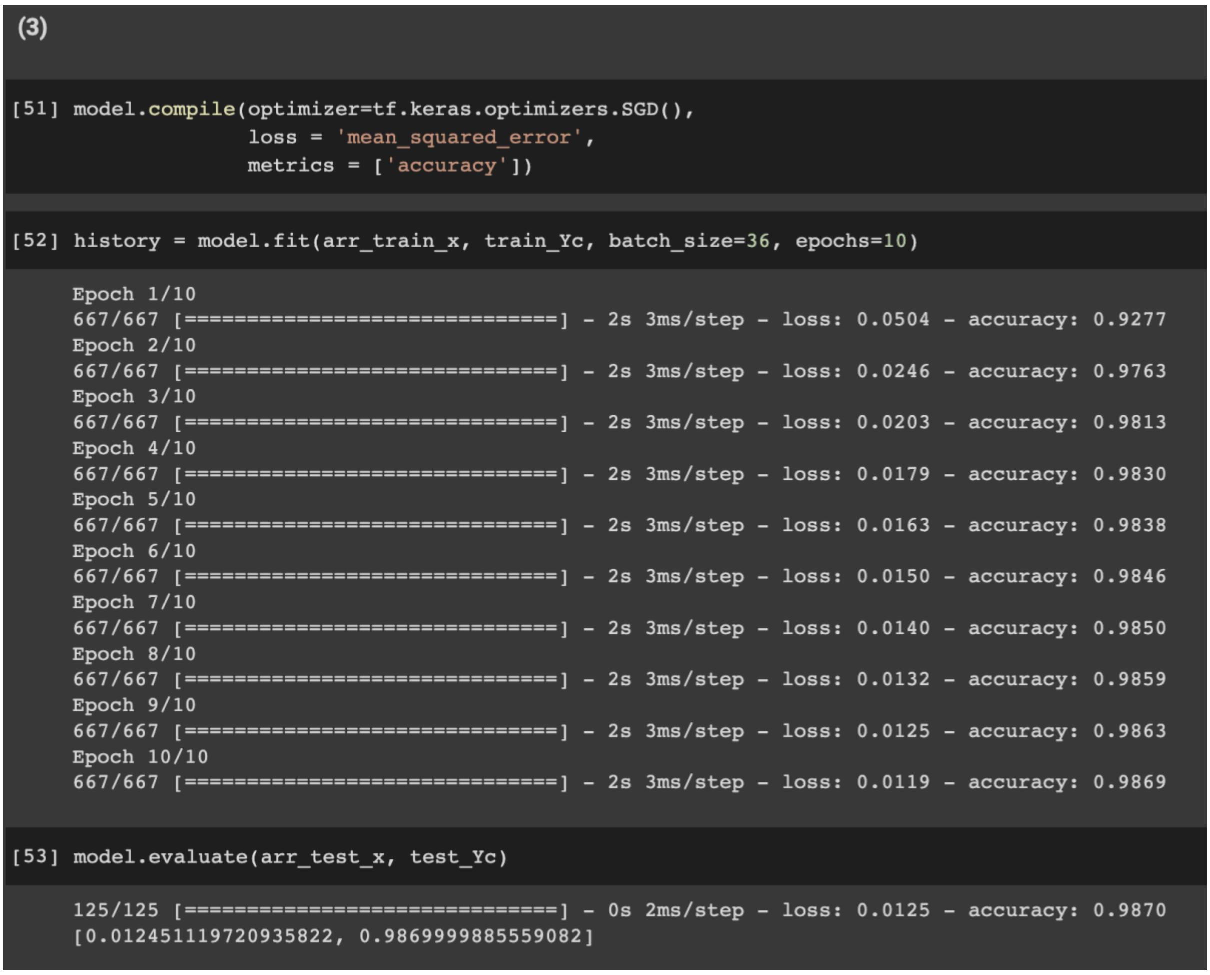


(2)

```
[49] train_Yc = tf.keras.utils.to_categorical(y_train, num_classes=4)
     test_Yc = tf.keras.utils.to_categorical(y_test, num_classes=4)
```

```
[50] model = tf.keras.Sequential([
         tf.keras.layers.Flatten(input_shape=(28,28)),
         tf.keras.layers.Dense(units=128, activation='relu'),
         tf.keras.layers.Dense(units=64, activation='relu'),
         tf.keras.layers.Dense(units=32, activation='relu'),
         tf.keras.layers.Dense(units=4)
         ])
     model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten_1 (Flatten)         (None, 784)               0         
                                                                 
 dense_4 (Dense)             (None, 128)               100480    
                                                                 
 dense_5 (Dense)             (None, 64)                8256      
                                                                 
 dense_6 (Dense)             (None, 32)                2080      
                                                                 
 dense_7 (Dense)             (None, 4)                 132       
                                                                 
=================================================================
Total params: 110,948
Trainable params: 110,948
Non-trainable params: 0
_________________________________________________________________
```



(3)

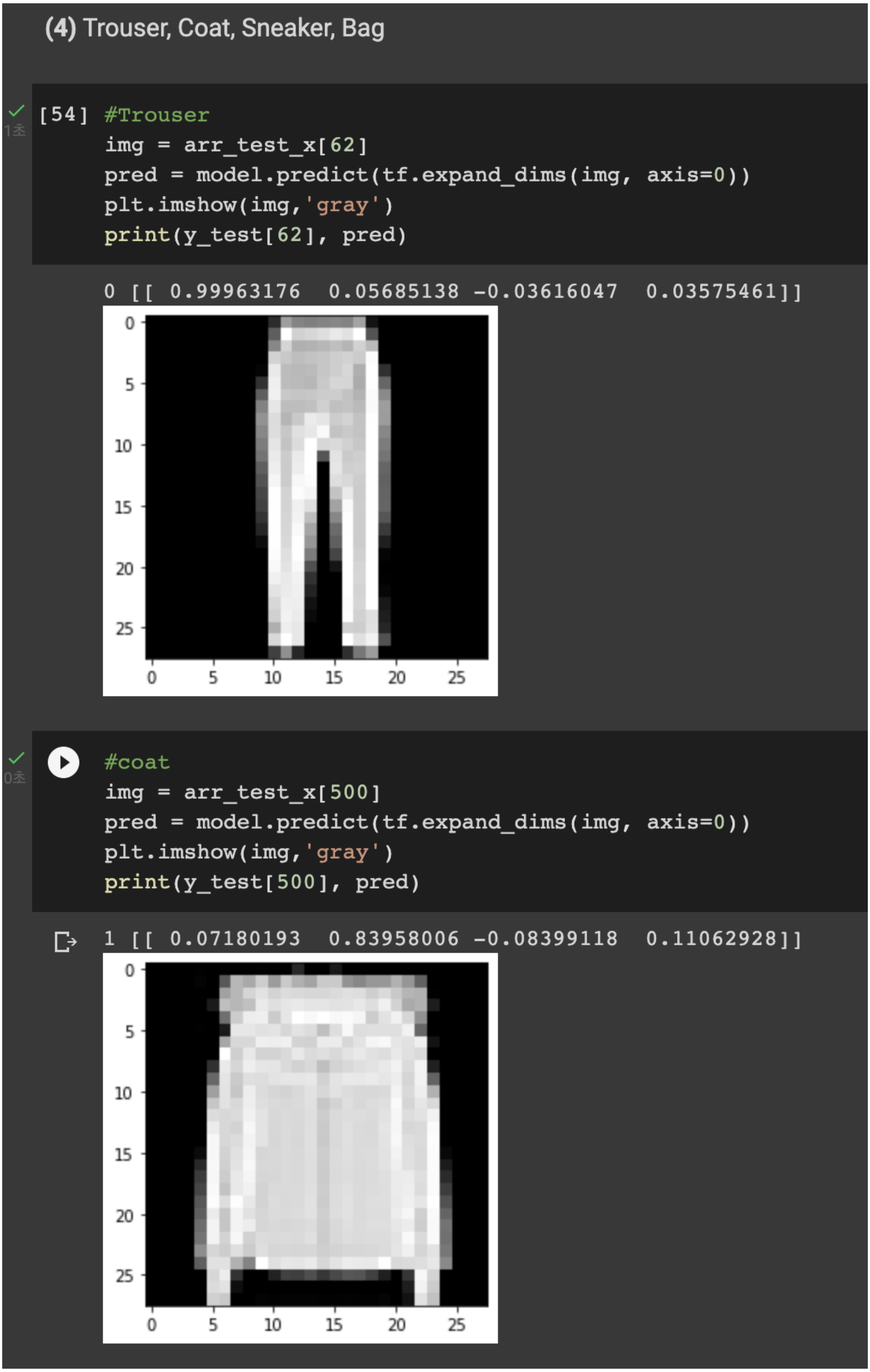
```
[51] model.compile(optimizer=tf.keras.optimizers.SGD(),
                   loss = 'mean_squared_error',
                   metrics = ['accuracy'])
```

```
[52] history = model.fit(arr_train_x, train_Yc, batch_size=36, epochs=10)
```

```
     Epoch 1/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0504 - accuracy: 0.9277
     Epoch 2/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0246 - accuracy: 0.9763
     Epoch 3/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0203 - accuracy: 0.9813
     Epoch 4/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0179 - accuracy: 0.9830
     Epoch 5/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0163 - accuracy: 0.9838
     Epoch 6/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0150 - accuracy: 0.9846
     Epoch 7/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0140 - accuracy: 0.9850
     Epoch 8/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0132 - accuracy: 0.9859
     Epoch 9/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0125 - accuracy: 0.9863
     Epoch 10/10
     667/667 [==============================] - 2s 3ms/step - loss: 0.0119 - accuracy: 0.9869
```

```
[53] model.evaluate(arr_test_x, test_Yc)
```

```
     125/125 [==============================] - 0s 2ms/step - loss: 0.0125 - accuracy: 0.9870
     [0.012451119720935822, 0.9869999885559082]
```



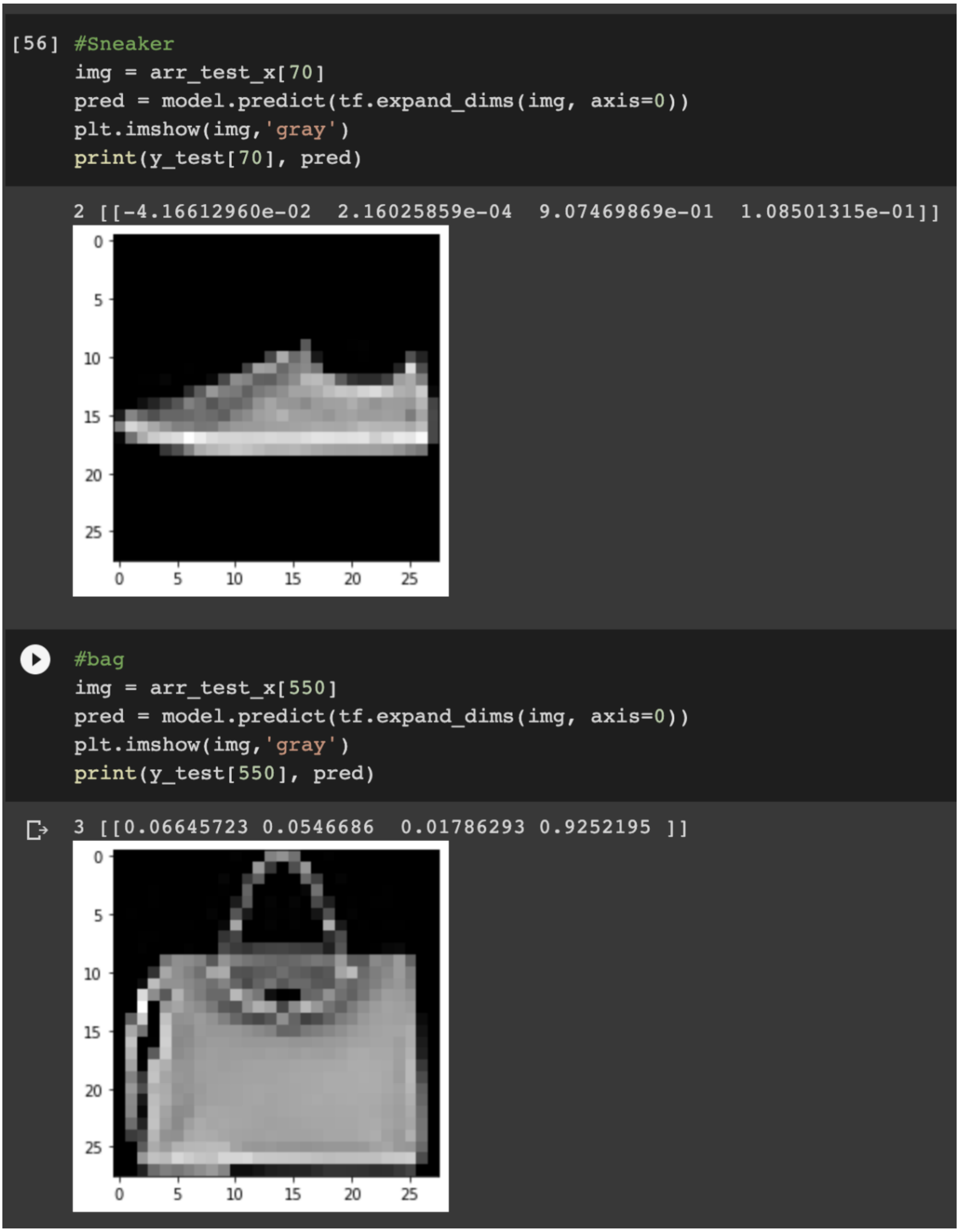
(4) Trouser, Coat, Sneaker, Bag

```
#Trouser
img = arr_test_x[62]
pred = model.predict(tf.expand_dims(img, axis=0))
plt.imshow(img,'gray')
print(y_test[62], pred)
```

```
0 [[ 0.99963176  0.05685138 -0.03616047  0.03575461]]
```

```
#coat
img = arr_test_x[500]
pred = model.predict(tf.expand_dims(img, axis=0))
plt.imshow(img,'gray')
print(y_test[500], pred)
```

```
1 [[ 0.07180193  0.83958006 -0.08399118  0.11062928]]
```



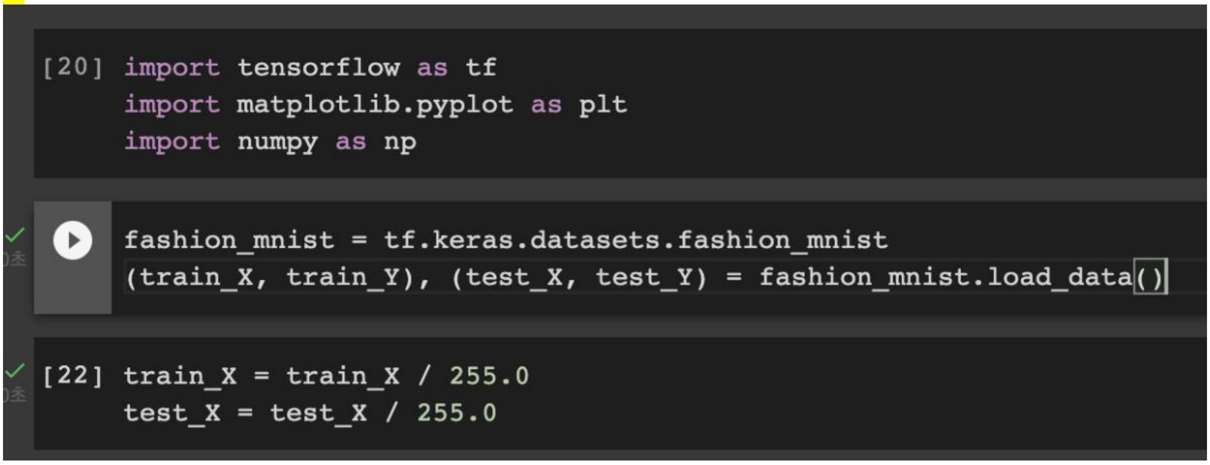
```
[56] #Sneaker
     img = arr_test_x[70]
     pred = model.predict(tf.expand_dims(img, axis=0))
     plt.imshow(img,'gray')
     print(y_test[70], pred)
```

```
2 [[-4.16612960e-02  2.16025859e-04  9.07469869e-01  1.08501315e-01]]
```

```
#bag
img = arr_test_x[550]
pred = model.predict(tf.expand_dims(img, axis=0))
plt.imshow(img,'gray')
print(y_test[550], pred)
```

```
3 [[0.06645723 0.0546686  0.01786293 0.9252195 ]]
```

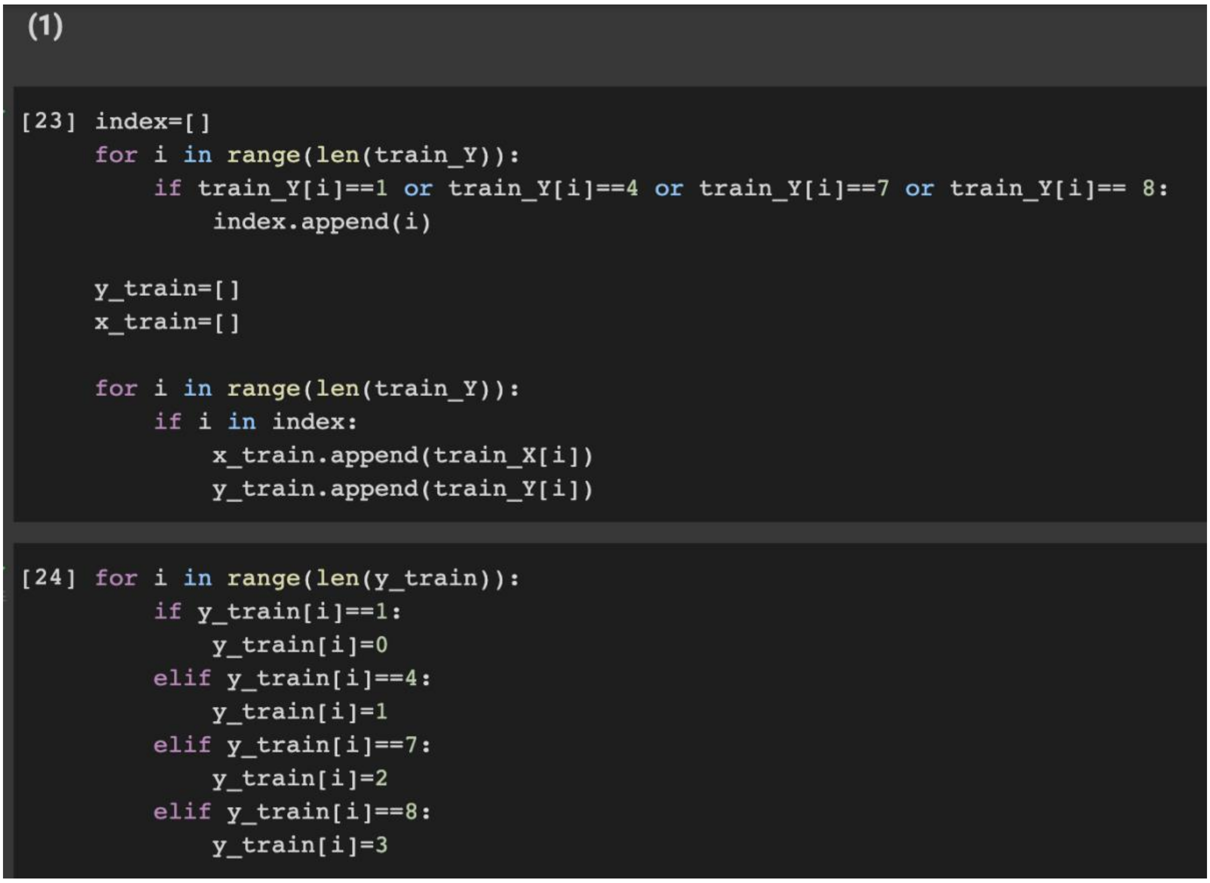
2.



```
[20] import tensorflow as tf
     import matplotlib.pyplot as plt
     import numpy as np
```

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_X, train_Y), (test_X, test_Y) = fashion_mnist.load_data()
```

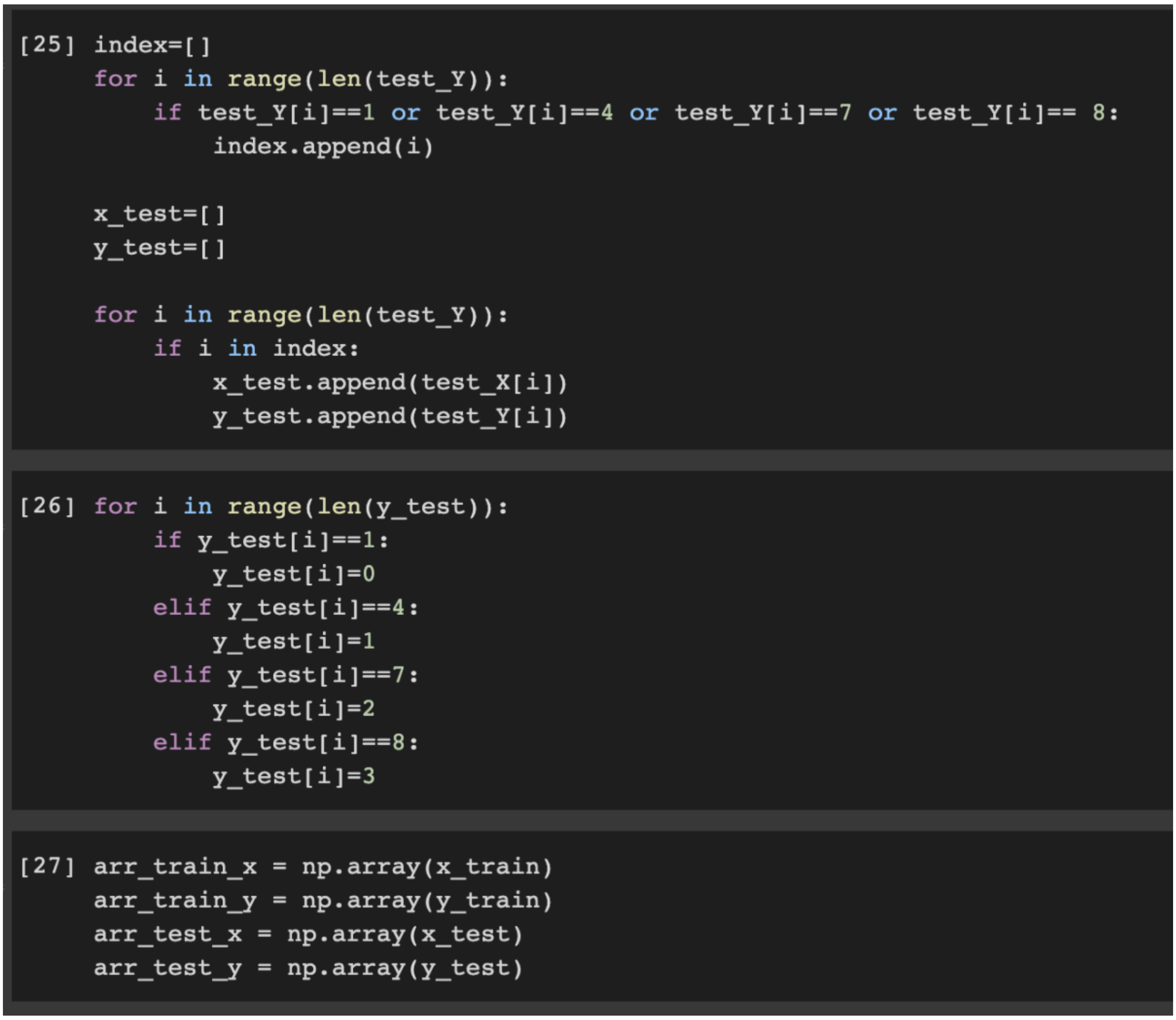
```
[22] train_X = train_X / 255.0
     test_X = test_X / 255.0
```



(1)

```
[23] index=[]
     for i in range(len(train_Y)):
         if train_Y[i]==1 or train_Y[i]==4 or train_Y[i]==7 or train_Y[i]== 8:
             index.append(i)

     y_train=[]
     x_train=[]

     for i in range(len(train_Y)):
         if i in index:
             x_train.append(train_X[i])
             y_train.append(train_Y[i])
```

```
[24] for i in range(len(y_train)):
         if y_train[i]==1:
             y_train[i]=0
         elif y_train[i]==4:
             y_train[i]=1
         elif y_train[i]==7:
             y_train[i]=2
         elif y_train[i]==8:
             y_train[i]=3
```

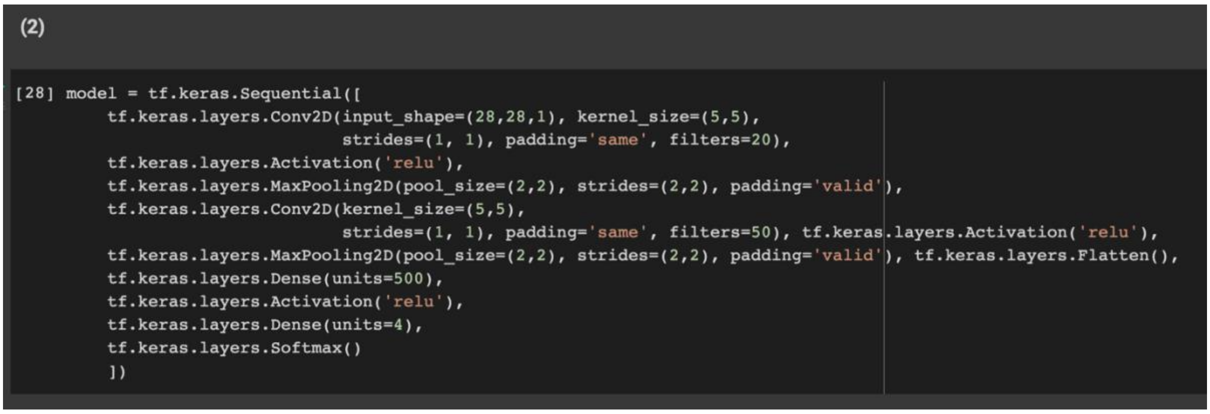


```
[25] index=[]
     for i in range(len(test_Y)):
         if test_Y[i]==1 or test_Y[i]==4 or test_Y[i]==7 or test_Y[i]== 8:
             index.append(i)

     x_test=[]
     y_test=[]

     for i in range(len(test_Y)):
         if i in index:
             x_test.append(test_X[i])
             y_test.append(test_Y[i])
```

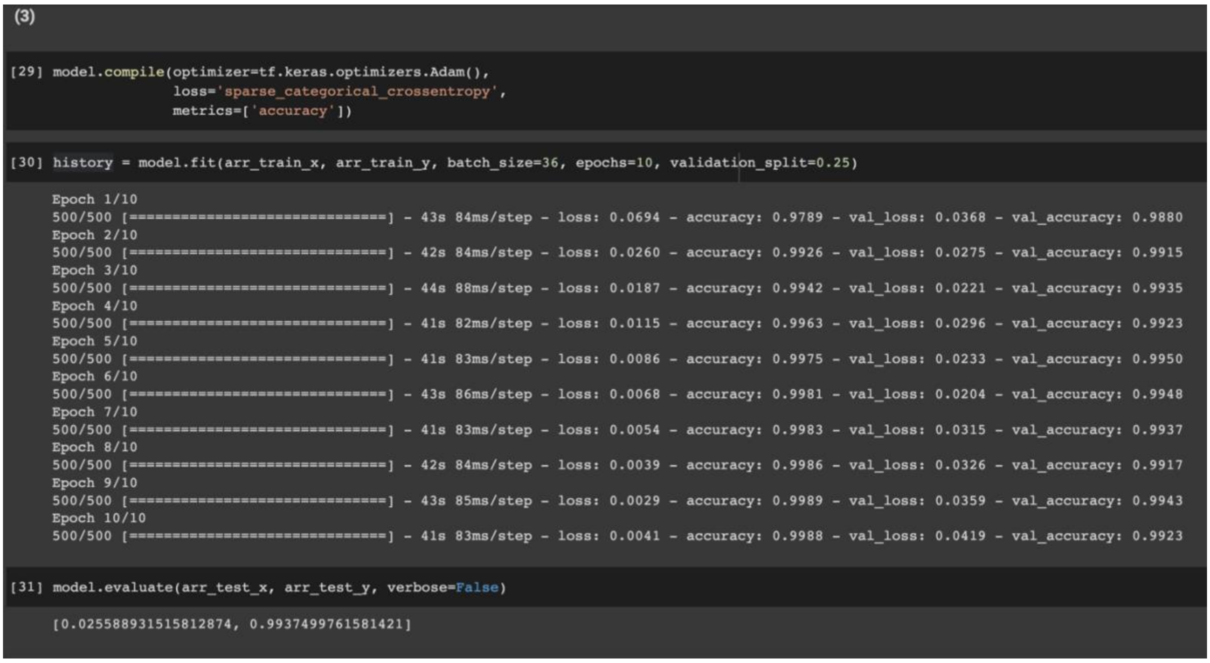
```
[26] for i in range(len(y_test)):
         if y_test[i]==1:
             y_test[i]=0
         elif y_test[i]==4:
             y_test[i]=1
         elif y_test[i]==7:
             y_test[i]=2
         elif y_test[i]==8:
             y_test[i]=3
```

```
[27] arr_train_x = np.array(x_train)
     arr_train_y = np.array(y_train)
     arr_test_x = np.array(x_test)
     arr_test_y = np.array(y_test)
```

(2)



```
[28] model = tf.keras.Sequential([
        tf.keras.layers.Conv2D(input_shape=(28,28,1), kernel_size=(5,5),
                               strides=(1, 1), padding='same', filters=20),
        tf.keras.layers.Activation('relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'),
        tf.keras.layers.Conv2D(kernel_size=(5,5),
                               strides=(1, 1), padding='same', filters=50), tf.keras.layers.Activation('relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'), tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(units=500),
        tf.keras.layers.Activation('relu'),
        tf.keras.layers.Dense(units=4),
        tf.keras.layers.Softmax()
        ])
```



(3)

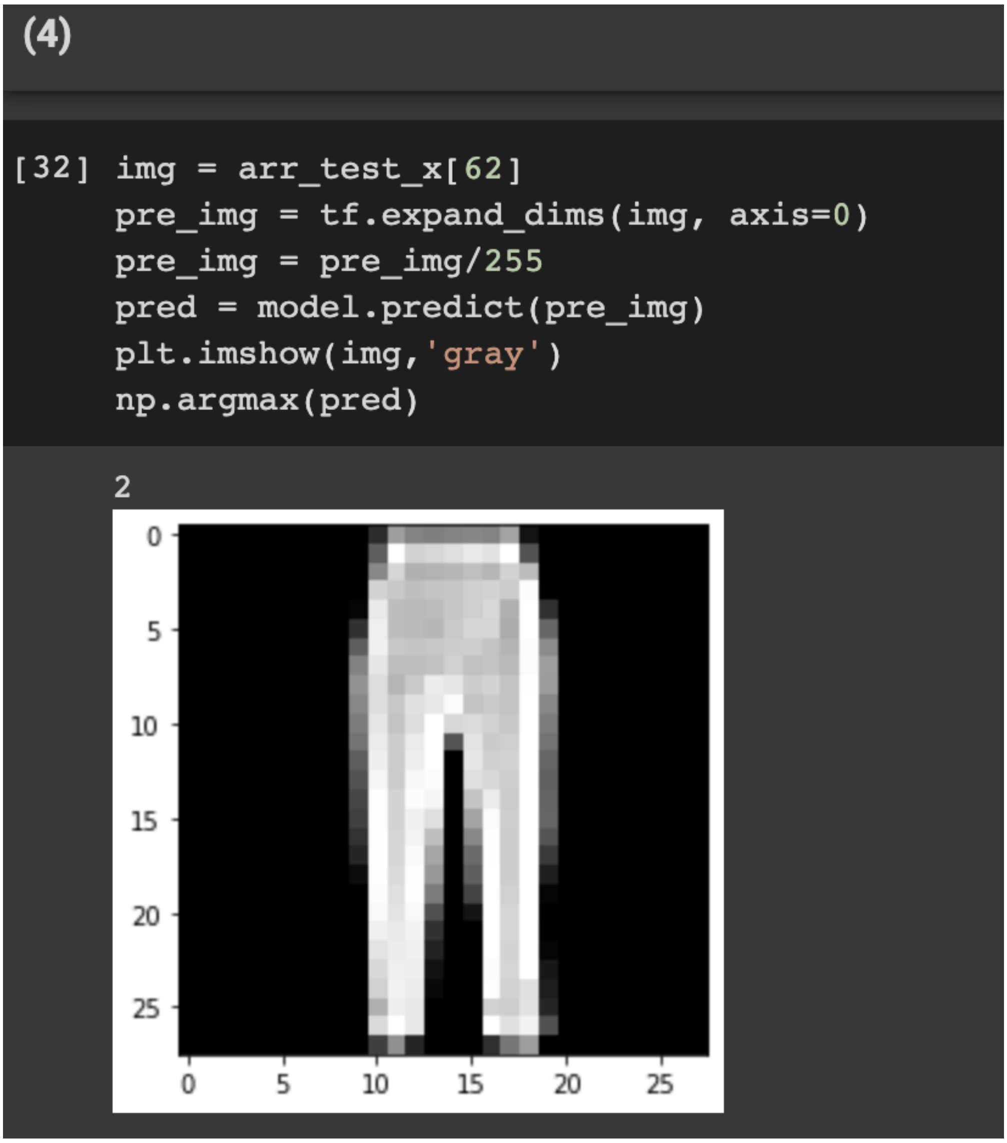
```
[29] model.compile(optimizer=tf.keras.optimizers.Adam(),
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
```

```
[30] history = model.fit(arr_train_x, arr_train_y, batch_size=36, epochs=10, validation_split=0.25)
```

```
Epoch 1/10
500/500 [==============================] - 43s 84ms/step - loss: 0.0694 - accuracy: 0.9789 - val_loss: 0.0368 - val_accuracy: 0.9880
Epoch 2/10
500/500 [==============================] - 42s 84ms/step - loss: 0.0260 - accuracy: 0.9926 - val_loss: 0.0275 - val_accuracy: 0.9915
Epoch 3/10
500/500 [==============================] - 44s 88ms/step - loss: 0.0187 - accuracy: 0.9942 - val_loss: 0.0221 - val_accuracy: 0.9935
Epoch 4/10
500/500 [==============================] - 41s 82ms/step - loss: 0.0115 - accuracy: 0.9963 - val_loss: 0.0296 - val_accuracy: 0.9923
Epoch 5/10
500/500 [==============================] - 41s 83ms/step - loss: 0.0086 - accuracy: 0.9975 - val_loss: 0.0233 - val_accuracy: 0.9950
Epoch 6/10
500/500 [==============================] - 43s 86ms/step - loss: 0.0068 - accuracy: 0.9981 - val_loss: 0.0204 - val_accuracy: 0.9948
Epoch 7/10
500/500 [==============================] - 41s 83ms/step - loss: 0.0054 - accuracy: 0.9983 - val_loss: 0.0315 - val_accuracy: 0.9937
Epoch 8/10
500/500 [==============================] - 42s 84ms/step - loss: 0.0039 - accuracy: 0.9986 - val_loss: 0.0326 - val_accuracy: 0.9917
Epoch 9/10
500/500 [==============================] - 43s 85ms/step - loss: 0.0029 - accuracy: 0.9989 - val_loss: 0.0359 - val_accuracy: 0.9943
Epoch 10/10
500/500 [==============================] - 41s 83ms/step - loss: 0.0041 - accuracy: 0.9988 - val_loss: 0.0419 - val_accuracy: 0.9923
```

```
[31] model.evaluate(arr_test_x, arr_test_y, verbose=False)
```

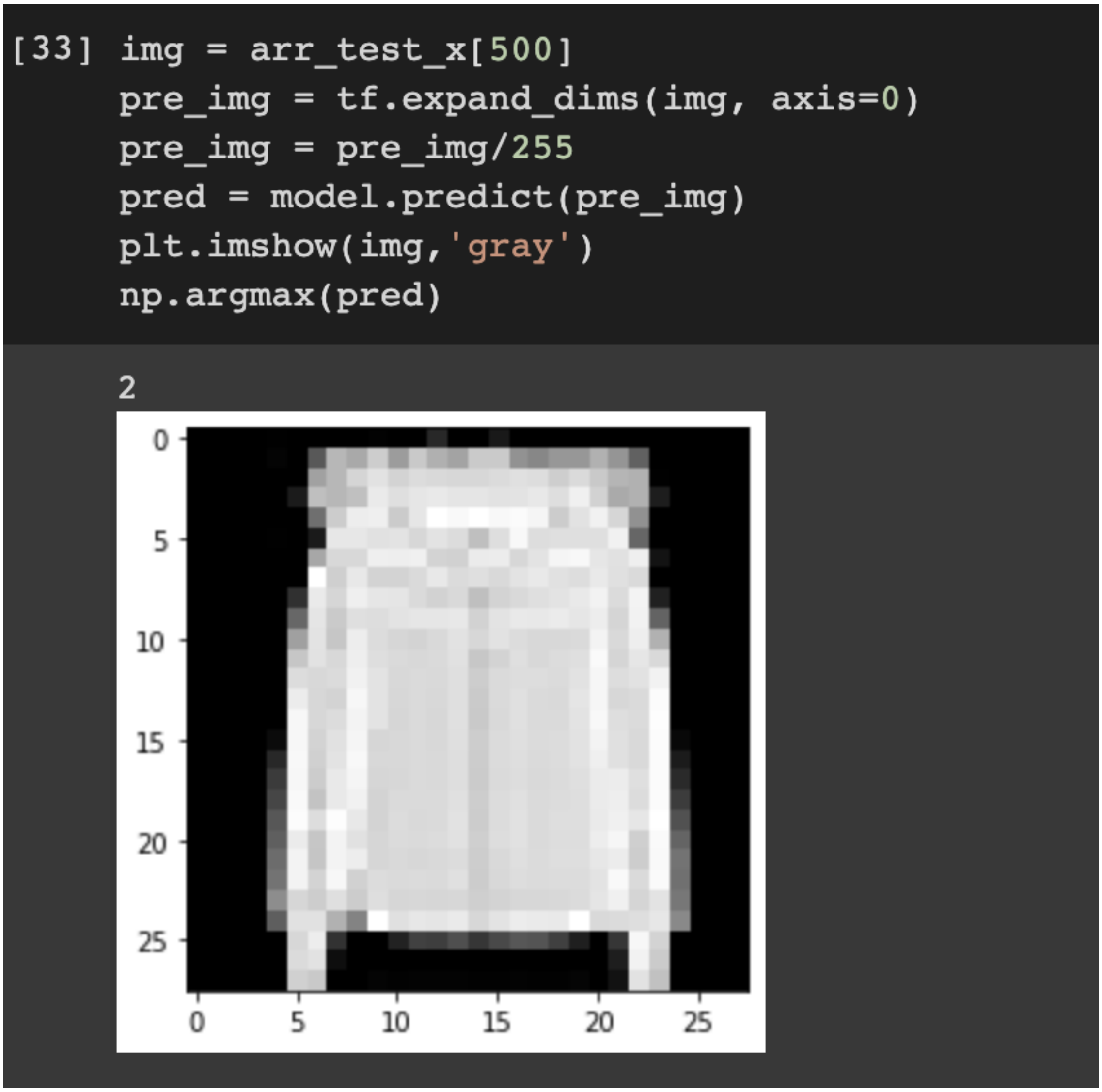
```
[0.025588931515812874, 0.9937499761581421]
```



(4)

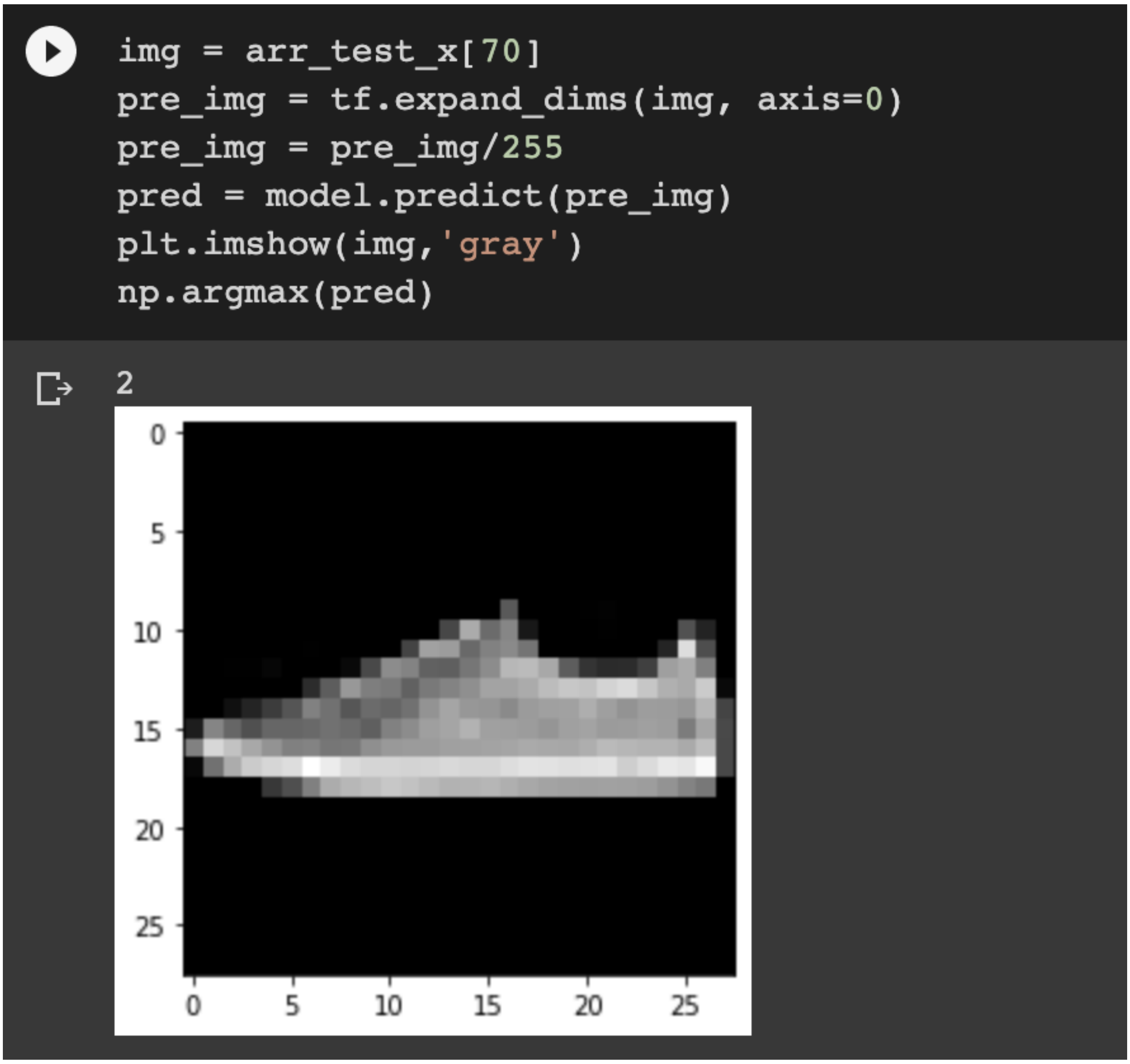
```
[32] img = arr_test_x[62]
     pre_img = tf.expand_dims(img, axis=0)
     pre_img = pre_img/255
     pred = model.predict(pre_img)
     plt.imshow(img,'gray')
     np.argmax(pred)
```

```
2
```



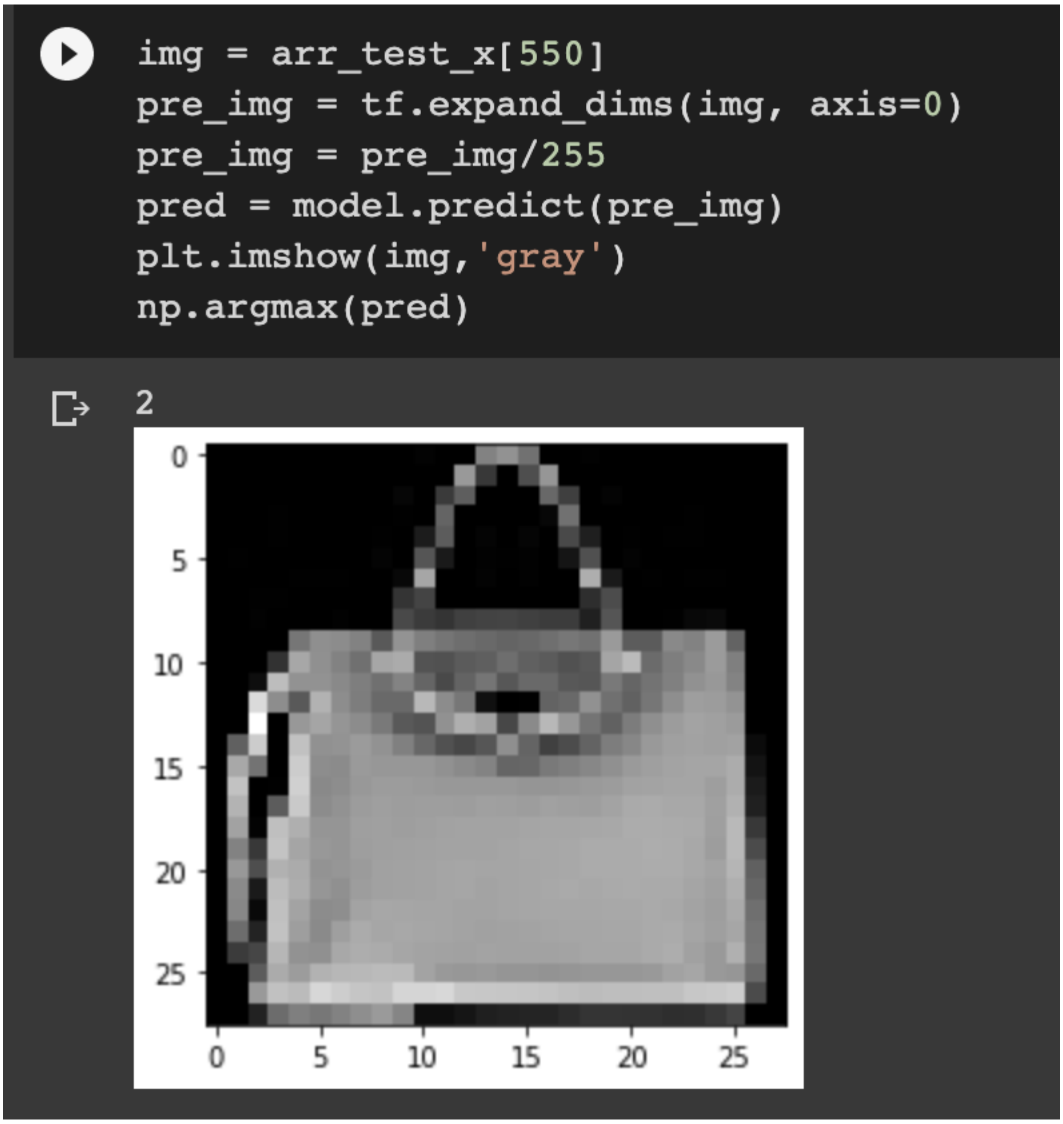
```
[33] img = arr_test_x[500]
     pre_img = tf.expand_dims(img, axis=0)
     pre_img = pre_img/255
     pred = model.predict(pre_img)
     plt.imshow(img,'gray')
     np.argmax(pred)
```

```
2
```



```
img = arr_test_x[70]
pre_img = tf.expand_dims(img, axis=0)
pre_img = pre_img/255
pred = model.predict(pre_img)
plt.imshow(img,'gray')
np.argmax(pred)
```

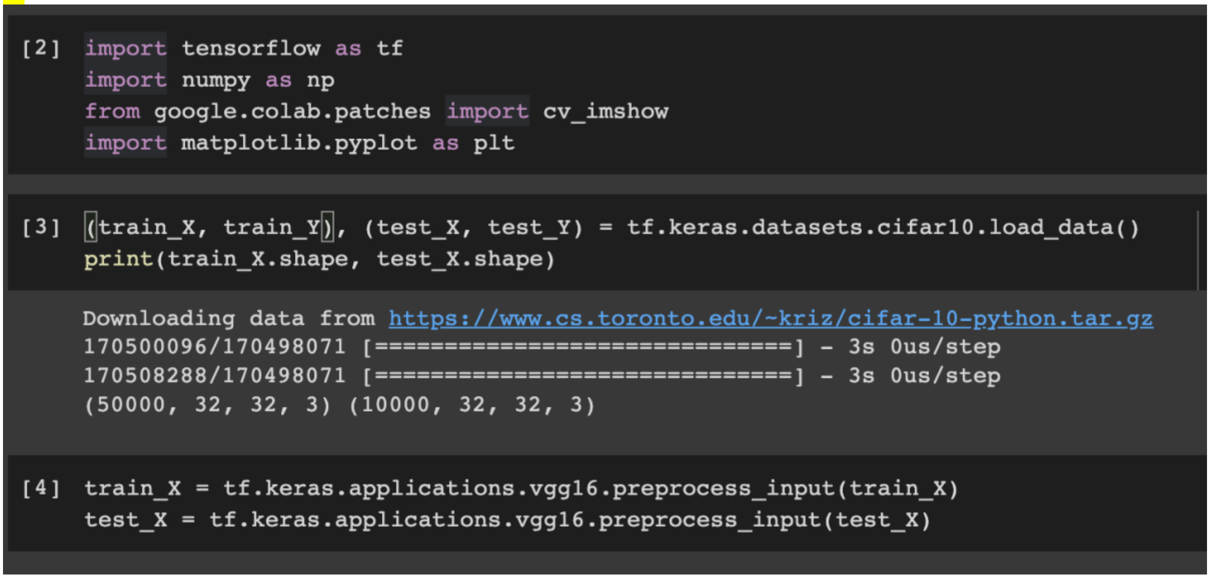
```
2
```



```
img = arr_test_x[550]
pre_img = tf.expand_dims(img, axis=0)
pre_img = pre_img/255
pred = model.predict(pre_img)
plt.imshow(img,'gray')
np.argmax(pred)
```

```
2
```

3.

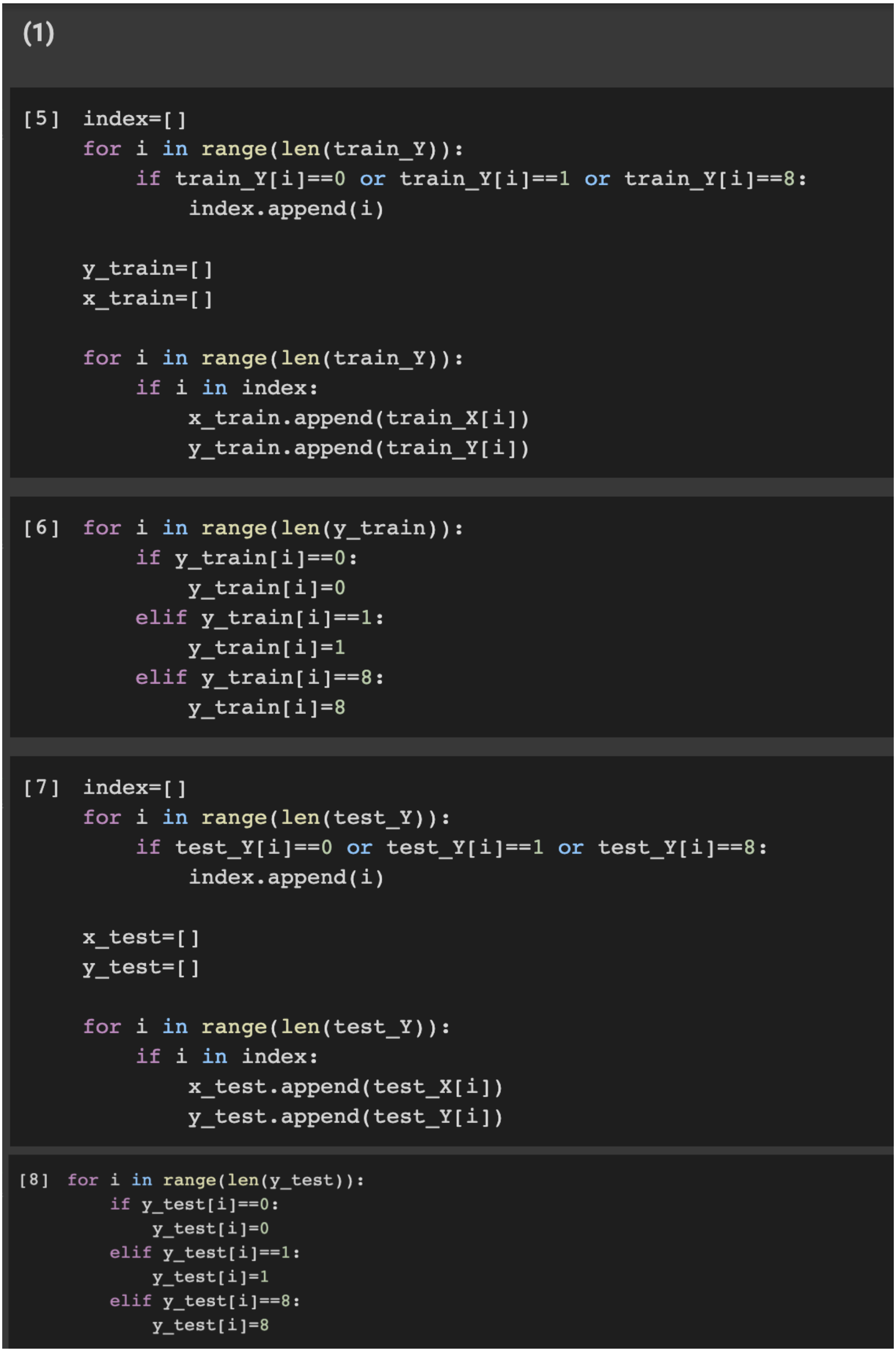


```
[2]  import tensorflow as tf
     import numpy as np
     from google.colab.patches import cv_imshow
     import matplotlib.pyplot as plt
```

```
[3]  (train_X, train_Y), (test_X, test_Y) = tf.keras.datasets.cifar10.load_data()
     print(train_X.shape, test_X.shape)
```

```
     Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
     170500096/170498071 [==============================] - 3s 0us/step
     170508288/170498071 [==============================] - 3s 0us/step
     (50000, 32, 32, 3) (10000, 32, 32, 3)
```

```
[4]  train_X = tf.keras.applications.vgg16.preprocess_input(train_X)
     test_X = tf.keras.applications.vgg16.preprocess_input(test_X)
```



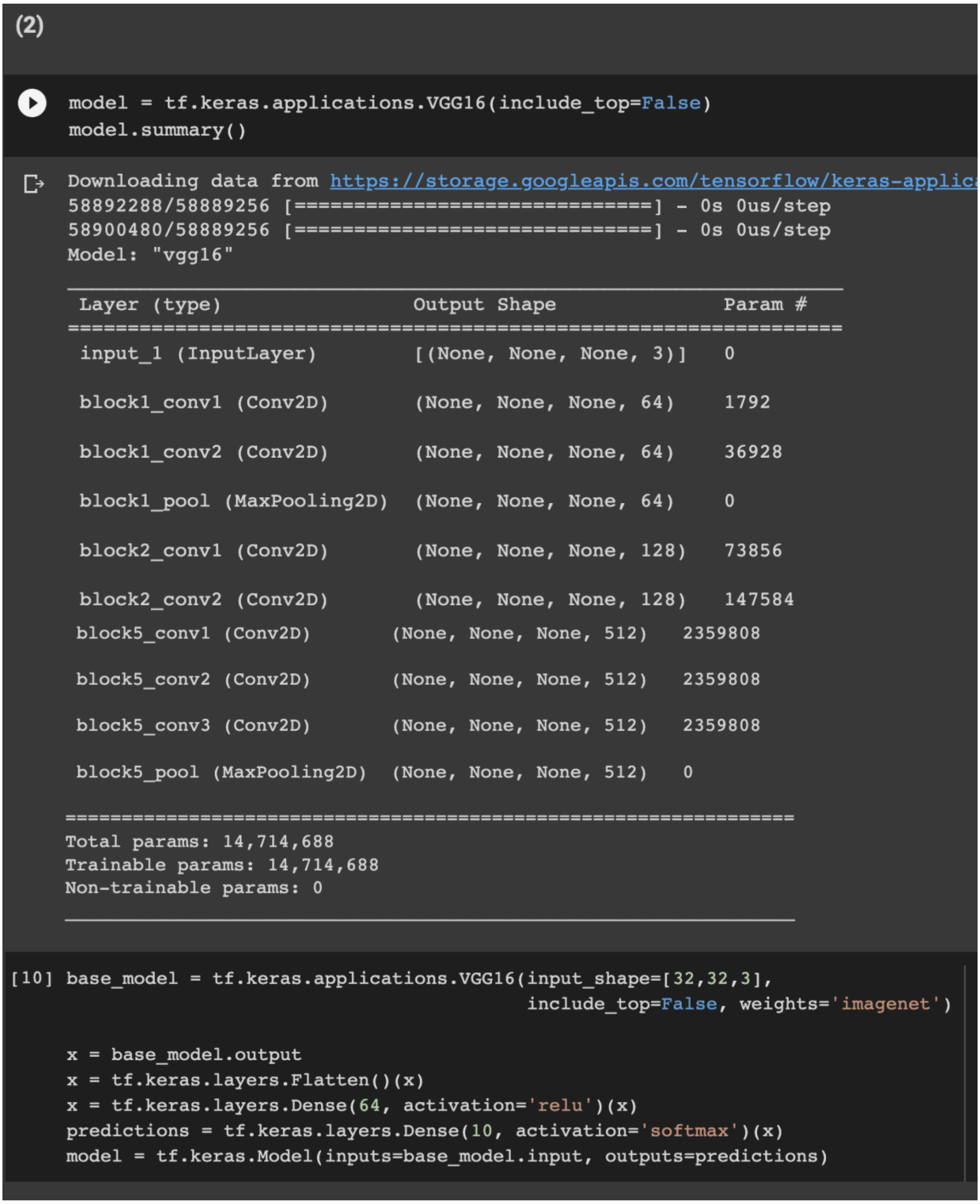
(1)

```
[5]  index=[]
     for i in range(len(train_Y)):
         if train_Y[i]==0 or train_Y[i]==1 or train_Y[i]==8:
             index.append(i)

     y_train=[]
     x_train=[]

     for i in range(len(train_Y)):
         if i in index:
             x_train.append(train_X[i])
             y_train.append(train_Y[i])
```

```
[6]  for i in range(len(y_train)):
         if y_train[i]==0:
             y_train[i]=0
         elif y_train[i]==1:
             y_train[i]=1
         elif y_train[i]==8:
             y_train[i]=8
```

```
[7]  index=[]
     for i in range(len(test_Y)):
         if test_Y[i]==0 or test_Y[i]==1 or test_Y[i]==8:
             index.append(i)

     x_test=[]
     y_test=[]

     for i in range(len(test_Y)):
         if i in index:
             x_test.append(test_X[i])
             y_test.append(test_Y[i])
```

```
[8]  for i in range(len(y_test)):
         if y_test[i]==0:
             y_test[i]=0
         elif y_test[i]==1:
             y_test[i]=1
         elif y_test[i]==8:
             y_test[i]=8
```

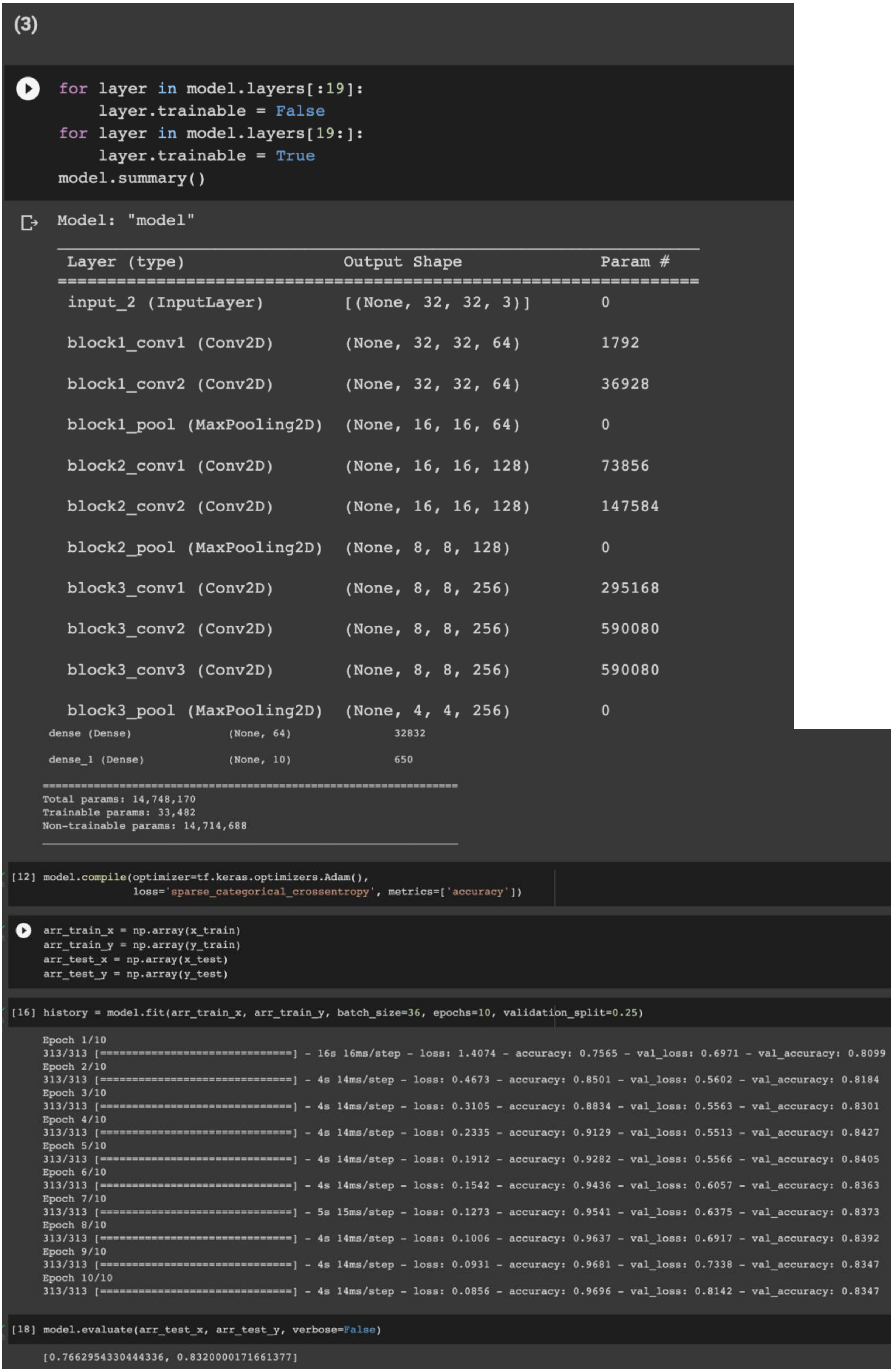


(2)

```
model = tf.keras.applications.VGG16(include_top=False)
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applica
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None, None, 3)]   0

 block1_conv1 (Conv2D)       (None, None, None, 64)    1792

 block1_conv2 (Conv2D)       (None, None, None, 64)    36928

 block1_pool (MaxPooling2D)  (None, None, None, 64)    0

 block2_conv1 (Conv2D)       (None, None, None, 128)   73856

 block2_conv2 (Conv2D)       (None, None, None, 128)   147584
 block5_conv1 (Conv2D)       (None, None, None, 512)   2359808

 block5_conv2 (Conv2D)       (None, None, None, 512)   2359808

 block5_conv3 (Conv2D)       (None, None, None, 512)   2359808

 block5_pool (MaxPooling2D)  (None, None, None, 512)   0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

```
[10] base_model = tf.keras.applications.VGG16(input_shape=[32,32,3],
                                              include_top=False, weights='imagenet')

     x = base_model.output
     x = tf.keras.layers.Flatten()(x)
     x = tf.keras.layers.Dense(64, activation='relu')(x)
     predictions = tf.keras.layers.Dense(10, activation='softmax')(x)
     model = tf.keras.Model(inputs=base_model.input, outputs=predictions)
```



(3)

```
for layer in model.layers[:19]:
    layer.trainable = False
for layer in model.layers[19:]:
    layer.trainable = True
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 3)]       0

 block1_conv1 (Conv2D)       (None, 32, 32, 64)        1792

 block1_conv2 (Conv2D)       (None, 32, 32, 64)        36928

 block1_pool (MaxPooling2D)  (None, 16, 16, 64)        0

 block2_conv1 (Conv2D)       (None, 16, 16, 128)       73856

 block2_conv2 (Conv2D)       (None, 16, 16, 128)       147584

 block2_pool (MaxPooling2D)  (None, 8, 8, 128)         0

 block3_conv1 (Conv2D)       (None, 8, 8, 256)         295168

 block3_conv2 (Conv2D)       (None, 8, 8, 256)         590080

 block3_conv3 (Conv2D)       (None, 8, 8, 256)         590080

 block3_pool (MaxPooling2D)  (None, 4, 4, 256)         0
 dense (Dense)               (None, 64)                32832

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 14,748,170
Trainable params: 33,482
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
[12] model.compile(optimizer=tf.keras.optimizers.Adam(),
                   loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

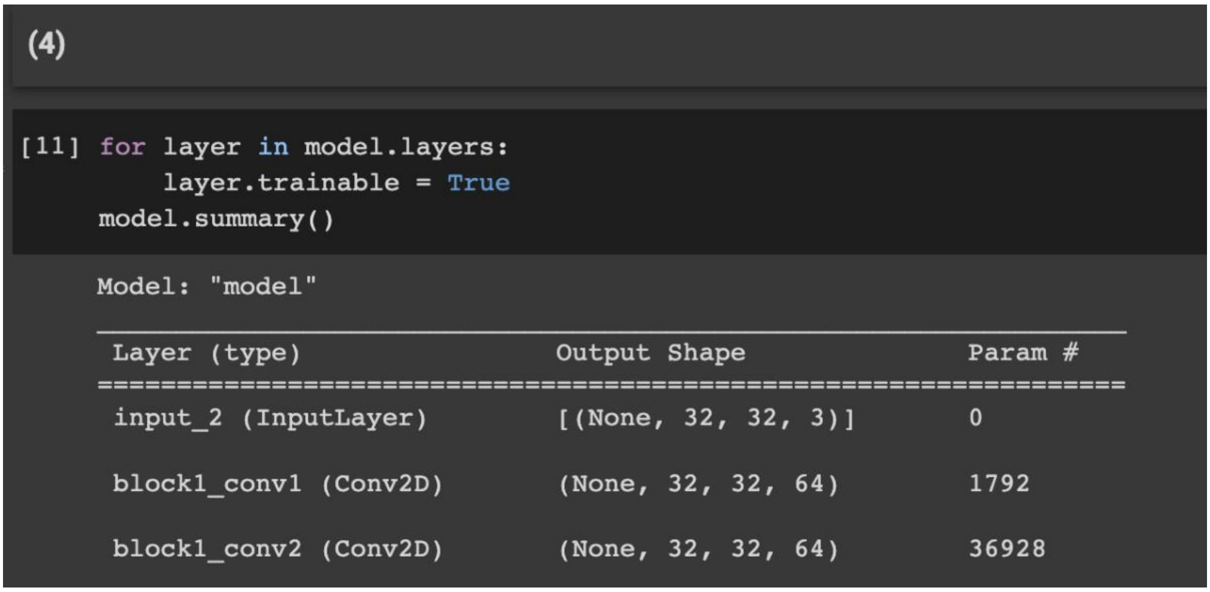
```
arr_train_x = np.array(x_train)
arr_train_y = np.array(y_train)
arr_test_x = np.array(x_test)
arr_test_y = np.array(y_test)
```

```
[16] history = model.fit(arr_train_x, arr_train_y, batch_size=36, epochs=10, validation_split=0.25)
```

```
Epoch 1/10
313/313 [==============================] - 16s 16ms/step - loss: 1.4074 - accuracy: 0.7565 - val_loss: 0.6971 - val_accuracy: 0.8099
Epoch 2/10
313/313 [==============================] - 4s 14ms/step - loss: 0.4673 - accuracy: 0.8501 - val_loss: 0.5602 - val_accuracy: 0.8184
Epoch 3/10
313/313 [==============================] - 4s 14ms/step - loss: 0.3105 - accuracy: 0.8834 - val_loss: 0.5563 - val_accuracy: 0.8301
Epoch 4/10
313/313 [==============================] - 4s 14ms/step - loss: 0.2335 - accuracy: 0.9129 - val_loss: 0.5513 - val_accuracy: 0.8427
Epoch 5/10
313/313 [==============================] - 4s 14ms/step - loss: 0.1912 - accuracy: 0.9282 - val_loss: 0.5566 - val_accuracy: 0.8405
Epoch 6/10
313/313 [==============================] - 4s 14ms/step - loss: 0.1542 - accuracy: 0.9436 - val_loss: 0.6057 - val_accuracy: 0.8363
Epoch 7/10
313/313 [==============================] - 5s 15ms/step - loss: 0.1273 - accuracy: 0.9541 - val_loss: 0.6375 - val_accuracy: 0.8373
Epoch 8/10
313/313 [==============================] - 4s 14ms/step - loss: 0.1006 - accuracy: 0.9637 - val_loss: 0.6917 - val_accuracy: 0.8392
Epoch 9/10
313/313 [==============================] - 4s 14ms/step - loss: 0.0931 - accuracy: 0.9681 - val_loss: 0.7338 - val_accuracy: 0.8347
Epoch 10/10
313/313 [==============================] - 4s 14ms/step - loss: 0.0856 - accuracy: 0.9696 - val_loss: 0.8142 - val_accuracy: 0.8347
```

```
[18] model.evaluate(arr_test_x, arr_test_y, verbose=False)
```

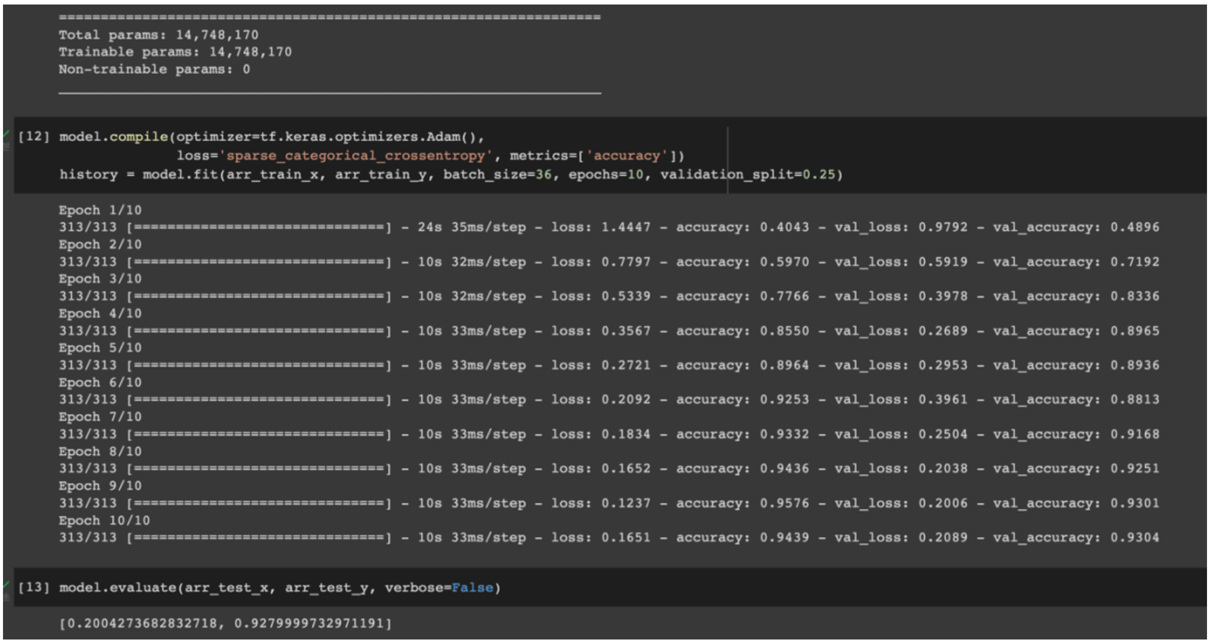
```
[0.7662954330444336, 0.8320000171661377]
```



(4)

```
[11] for layer in model.layers:
         layer.trainable = True
     model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 3)]       0

 block1_conv1 (Conv2D)       (None, 32, 32, 64)        1792

 block1_conv2 (Conv2D)       (None, 32, 32, 64)        36928
```



```
=================================================================
Total params: 14,748,170
Trainable params: 14,748,170
Non-trainable params: 0
_________________________________________________________________
```

```
[12] model.compile(optimizer=tf.keras.optimizers.Adam(),
                   loss='sparse_categorical_crossentropy', metrics=['accuracy'])
     history = model.fit(arr_train_x, arr_train_y, batch_size=36, epochs=10, validation_split=0.25)
```

```
Epoch 1/10
313/313 [==============================] - 24s 35ms/step - loss: 1.4447 - accuracy: 0.4043 - val_loss: 0.9792 - val_accuracy: 0.4896
Epoch 2/10
313/313 [==============================] - 10s 32ms/step - loss: 0.7797 - accuracy: 0.5970 - val_loss: 0.5919 - val_accuracy: 0.7192
Epoch 3/10
313/313 [==============================] - 10s 32ms/step - loss: 0.5339 - accuracy: 0.7766 - val_loss: 0.3978 - val_accuracy: 0.8336
Epoch 4/10
313/313 [==============================] - 10s 33ms/step - loss: 0.3567 - accuracy: 0.8550 - val_loss: 0.2689 - val_accuracy: 0.8965
Epoch 5/10
313/313 [==============================] - 10s 33ms/step - loss: 0.2721 - accuracy: 0.8964 - val_loss: 0.2953 - val_accuracy: 0.8936
Epoch 6/10
313/313 [==============================] - 10s 33ms/step - loss: 0.2092 - accuracy: 0.9253 - val_loss: 0.3961 - val_accuracy: 0.8813
Epoch 7/10
313/313 [==============================] - 10s 33ms/step - loss: 0.1834 - accuracy: 0.9332 - val_loss: 0.2504 - val_accuracy: 0.9168
Epoch 8/10
313/313 [==============================] - 10s 33ms/step - loss: 0.1652 - accuracy: 0.9436 - val_loss: 0.2038 - val_accuracy: 0.9251
Epoch 9/10
313/313 [==============================] - 10s 33ms/step - loss: 0.1237 - accuracy: 0.9576 - val_loss: 0.2006 - val_accuracy: 0.9301
Epoch 10/10
313/313 [==============================] - 10s 33ms/step - loss: 0.1651 - accuracy: 0.9439 - val_loss: 0.2089 - val_accuracy: 0.9304
```

```
[13] model.evaluate(arr_test_x, arr_test_y, verbose=False)
```

```
[0.2004273682832718, 0.9279999732971191]
```