```
from sklearn.model_selection import train_test_split
import numpy as np
import glob
from PIL import Image
import tensorflow as tf
import matplotlib.pyplot as plt

```

```
def resize_images(img_path):
    images = glob.glob(img_path + "/*.jpg")
    target_size = (28,28)
    for img in images:
        old = Image.open(img)
        new = old.resize(target_size, Image.ANTIALIAS)
        new.save(img,"jpeg")

    print(len(images), " images resized.")

```

```
resize_images(img_path+'scissors/')
resize_images(img_path+'rock/')
resize_images(img_path+'paper/')
```

```
600  images resized.
614  images resized.
598  images resized.
```

```
num_data = len(os.listdir('/content/rock/') + os.listdir('/content/scissors/') + os.listdir('/co
def load_data(img_path, number_of_data):
    #가위 : 0, 바위 : 1, 보 : 2
    img_size = 28
    color = 3
    imgs = np.zeros(number_of_data * img_size * img_size * color, dtype=np.int32).reshape(number
    labels = np.zeros(number_of_data, dtype=np.int32)

    idx = 0
    for file in glob.iglob(img_path+'scissors/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img # 데이터 영역에 이미지 복사
        labels[idx]=0 #가위 : 0
        idx = idx + 1

    for file in glob.iglob(img_path+'rock/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img # 데이터 영역에 이미지 복사
        labels[idx]=1 #바위 : 1
        idx = idx + 1

    for file in glob.iglob(img_path+'paper/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img # 데이터 영역에 이미지 복사
```

```
        labels[idx]=2 #보 : 2
        idx = idx + 1

    return imgs, labels

img_path = '/content/'


```

```
images, labels = load_data(img_path, num_data)
```

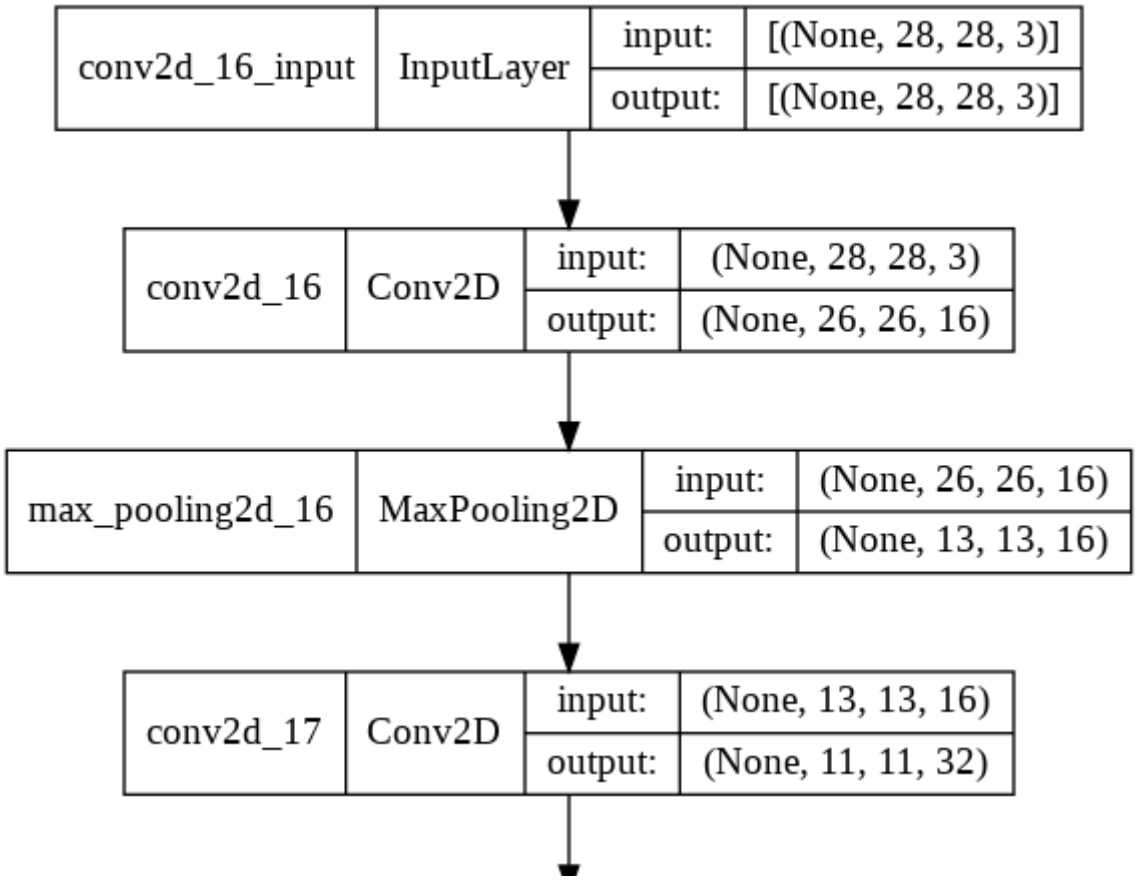
```
normed_images = images / 255
```

```
X_train, X_test, y_train, y_test = train_test_split(normed_images, labels,test_size=0.2,random_s
```

```
# define sequence
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(16, (3,3),activation='relu', input_shape=(28,28,3))) # 16
model.add(tf.keras.layers.MaxPool2D(2,2))
model.add(tf.keras.layers.Conv2D(32,(3,3), activation='relu')) # 32
model.add(tf.keras.layers.MaxPooling2D((2,2)))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(32, activation='relu')) # 32
model.add(tf.keras.layers.Dense(3, activation='softmax'))

```

```
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model.png')
plot_model(model, to_file='model_shapes.png', show_shapes=True)
```




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

```
1 history = model.fit(X_train, y_train, epochs=15)
```

```
Epoch 1/15
46/46 [==============================] - 2s 22ms/step - loss: 1.0903 - accuracy: 0.3761
Epoch 2/15
46/46 [==============================] - 1s 20ms/step - loss: 0.9950 - accuracy: 0.5397
Epoch 3/15
46/46 [==============================] - 1s 20ms/step - loss: 0.7833 - accuracy: 0.7019
Epoch 4/15
46/46 [==============================] - 1s 20ms/step - loss: 0.6299 - accuracy: 0.7453
Epoch 5/15
46/46 [==============================] - 1s 19ms/step - loss: 0.5007 - accuracy: 0.8040
Epoch 6/15
46/46 [==============================] - 1s 19ms/step - loss: 0.4282 - accuracy: 0.8433
Epoch 7/15
46/46 [==============================] - 1s 20ms/step - loss: 0.3726 - accuracy: 0.8875
Epoch 8/15
46/46 [==============================] - 1s 20ms/step - loss: 0.2906 - accuracy: 0.9137
Epoch 9/15
46/46 [==============================] - 1s 20ms/step - loss: 0.2373 - accuracy: 0.9337
Epoch 10/15
46/46 [==============================] - 1s 19ms/step - loss: 0.2124 - accuracy: 0.9268
Epoch 11/15
46/46 [==============================] - 1s 20ms/step - loss: 0.1792 - accuracy: 0.9476
Epoch 12/15
46/46 [==============================] - 1s 21ms/step - loss: 0.1579 - accuracy: 0.9503
Epoch 13/15
46/46 [==============================] - 1s 20ms/step - loss: 0.1324 - accuracy: 0.9648
Epoch 14/15
46/46 [==============================] - 1s 19ms/step - loss: 0.1110 - accuracy: 0.9655
```

```
Epoch 15/15
46/46 [==============================] - 1s 19ms/step - loss: 0.1032 - accuracy: 0.9689
```

```
1 test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
```

```
12/12 - 0s - loss: 0.1066 - accuracy: 0.9532 - 215ms/epoch - 18ms/step
```

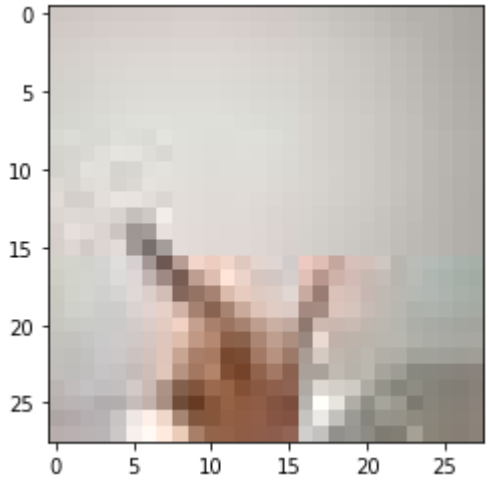
```
1 predictions = model.predict(X_test)
```

```
1 predicted_labels = np.argmax(predictions, axis=1)
```

```
1 # get wrong prediction list
2 import random
3 wrong_predict_list = []
4 for i, _ in enumerate(predicted_labels):
5
6     if predicted_labels[i] != y_test[i]:
7         wrong_predict_list.append(i)
```

```
1 # is prediction, label match
2 idx = 100
3 print(predictions[idx])
4 print(predicted_labels[idx])
5 print(y_test[idx])
6 plt.imshow(X_test[idx])
```

```
[9.5178407e-01 4.7411606e-02 8.0435665e-04]
0
0
<matplotlib.image.AxesImage at 0x7fa70427b3d0>
```

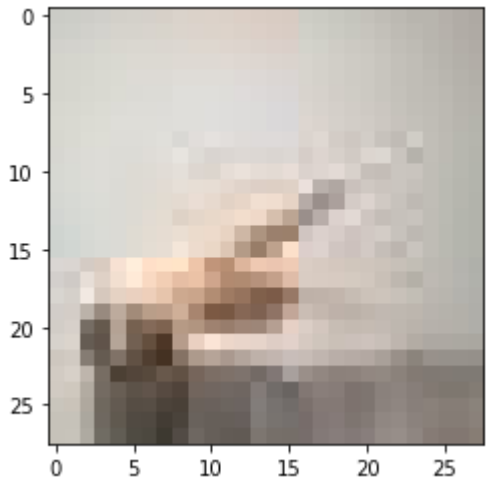



```
1 samples = random.choices(population=wrong_predict_list, k=5)
```

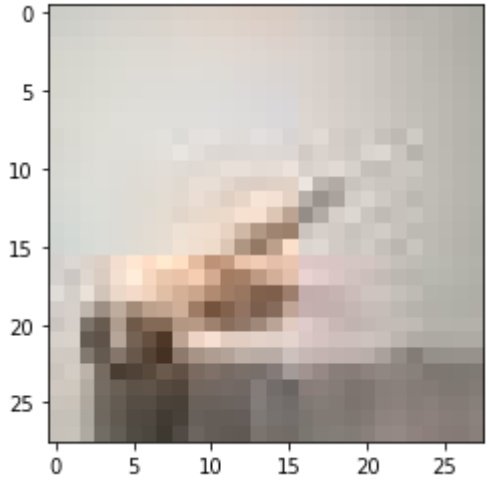
```
1 for n in samples:
2     print("예측확률분포: ", str(predictions[n]))
3     print("라벨 ", str(y_test[n]),"예측결과: ", str(predicted_labels[n]))
```

```
    plt.imshow(X_test[n], cmap=plt.cm.binary)
    plt.show()
```

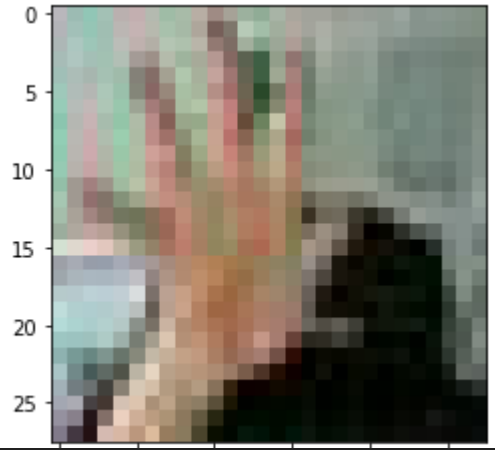
```
예측확률분포:  [3.8555187e-01 6.1432999e-01 1.1810708e-04]
라벨  0  예측결과:  1
```



```
예측확률분포:  [4.4483513e-01 5.5502540e-01 1.3940182e-04]
라벨  0  예측결과:  1
```



```
예측확률분포:  [0.1317438  0.7570609  0.11119536]
라벨  2  예측결과:  1
```



✓ 0초 오전 10:43에 완료됨