An explanation of how you approached the given problem and Solution for the problem

```
In [1]: import os
        import glob
        import cv2
        import numpy as np
        import matplotlib.pyplot as plt
        import tensorflow as tf
        import handshape_feature_extractor
        import frameextractor as fe
        from sklearn.metrics.pairwise import cosine_similarity
```

At first, I import library related to this project part2.

```
In [2]: hfe = handshape_feature_extractor.HandShapeFeatureExtractor.get_instance()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 32)      320

 max_pooling2d (MaxPooling2D  (None, 99, 99, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 97, 97, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 48, 48, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 46, 46, 64)        36928

 flatten (Flatten)           (None, 135424)            0

 dense (Dense)               (None, 64)                8667200

 dense_1 (Dense)             (None, 27)                1755

=================================================================
Total params: 8,724,699
Trainable params: 8,724,699
Non-trainable params: 0
_________________________________________________________________
None
```

call the get_Intsance() method of the class HandShapeFeatureExtractor from handshape_feature_extractor.py
check the architecture of pre-trained machine learning model usage for classification.
There are 27 feature vectors of the penultimate(last) layer of the model.

Task 1: Generate the penultimate layer for the training videos.

```
In [3]: mp4_path = 'traindata/'
        image_path = 'traindata/frames/'

        count = 0
        train_list = []
        for filename in glob.glob(os.path.join(mp4_path, '*.mp4')):
            #with open(os.path.join(os.getcwd(), filename), 'r') as f: # open in readonly mode
            fe.frameExtractor(filename, image_path, count)
            image = cv2.imread(image_path + "%#05d.png" % (count+1), cv2.IMREAD_GRAYSCALE)
            train_list.append(hfe.extract_feature(image))

            count += 1
            print(filename, count)
```

Extract the middle frame of each gesture video from the part1 on TrainData folder and save image files onto TrainImage folder. Then extract the feature of the image files. The image files should be Grayscale to satisfy img_arr = img_arr.reshape(1, 300, 300, 3) on the below method.

```
def extract_feature(self, image):
    try:
        img_arr = self.__pre_process_input_image(image)
        #input = tf.keras.Input(tensor=image)

        return self.model.predict(img_arr)
        #return self.model.predict(input)
    except Exception as e:
        raise
```

```
@staticmethod
def __pre_process_input_image(crop):
    try:
        img = cv2.resize(crop, (300, 300))
        img_arr = np.array(img) / 255.0
        img_arr=np.stack((img_arr,)*3,axis=-1)
        #img_arr = img_arr.reshape(1, 200, 200, 1)
        img_arr = img_arr.reshape(1,300, 300,3)
        return img_arr
    except Exception as e:
        print(str(e))
        raise
```

By using two methods, 27 Feature vectors of all the gestures can be generated like below.

```
[array([[6.8940618e-04, 7.4863746e-03, 2.5794252e-08, 3.3339587e-04,
        8.2353256e-07, 3.9329734e-08, 8.3504163e-02, 7.8741409e-02,
        7.1419818e-05, 5.9156611e-07, 7.7087629e-01, 5.7461396e-02,
        7.7632721e-15, 1.2249328e-12, 2.8160642e-13, 4.6115965e-05,
        6.2888159e-07, 1.4866944e-05, 8.1145236e-13, 5.3492339e-09,
        7.2934927e-04, 2.0924533e-06, 1.2317844e-07, 3.2516644e-05,
        3.1876066e-09, 1.4245443e-21, 8.9841578e-06]], dtype=float32),
```

There are total 51 gestures in the train list.

Task 2: Generate the penultimate layer for the test videos
Testing video also do the same step as training video.

```
In [4]: mp4_path = 'test/'
        image_path = 'test/frames/'

        count = 0
        test_list = []
        for filename in glob.glob(os.path.join(mp4_path, '*.mp4')):

            fe.frameExtractor(filename, image_path, count)
            image = cv2.imread(image_path + "%#05d.png" % (count+1), cv2.IMREAD_GRAYSCALE)
            test_list.append(hfe.extract_feature(image))

            count += 1
            print(filename, count)
```

Extract the middle frame of each gesture video from the part1 on TestData folder and save image files onto TestImage folder. Then extract the feature of the image files. The image files should be Grayscale to satisfy img_arr = img_arr.reshape(1, 300, 300, 3) on the below method.

```
def extract_feature(self, image):
    try:
        img_arr = self.__pre_process_input_image(image)
        #input = tf.keras.Input(tensor=image)

        return self.model.predict(img_arr)
        #return self.model.predict(input)
    except Exception as e:
        raise
```

```
@staticmethod
def __pre_process_input_image(crop):
    try:
        img = cv2.resize(crop, (300, 300))
        img_arr = np.array(img) / 255.0
        img_arr=np.stack((img_arr,)*3,axis=-1)
        #img_arr = img_arr.reshape(1, 200, 200, 1)
        img_arr = img_arr.reshape(1,300, 300,3)
        return img_arr
    except Exception as e:
        print(str(e))
        raise
```

By using two methods, 27 Feature vectors of all the gestures can be generated like below.

```
[array([[5.4760629e-05, 4.7887769e-02, 2.4446506e-05, 8.4034473e-02,
        6.0546267e-01, 5.3823134e-03, 2.0766791e-02, 1.2735669e-02,
        1.6813083e-01, 5.6940420e-03, 7.0452346e-03, 1.2704379e-04,
        5.1560486e-04, 1.9992114e-04, 7.9075014e-04, 3.1240226e-03,
        1.7421192e-03, 2.4140323e-02, 1.4181981e-03, 4.3805057e-04,
        1.8890580e-03, 7.7209515e-06, 5.4389922e-08, 8.3222976e-03,
        4.3169479e-05, 1.6637679e-11, 2.2665323e-05]], dtype=float32),
```

Task 3: Gesture recognition of the test dataset.

```
In [5]: i = 0             #test_data label
        acc = 0           #count for accuracy
        label_list = []   #list for results.csv
        for train_data in train_list:
            c = 0         #count for train_data label
            cs_max = 0
            #print("train data", train_data[0])
            for test_data in test_list:
                #print("test data", test_data[0])

                cs_current = cosine_similarity(train_data, test_data)[0][0]
                if cs_max < cs_current:
                    cs_max = cs_current
                    label = c
                #print(i, c, cs_current, cs_max)
                c = (c + 1) % 17

            print(i, label, cs_max)
            label_list.append(label)

            if i == label:
                acc = acc + 1
            i = (i + 1) % 17
        print("Accuracy % = ", round(acc/51*100, 2))
        print(label_list)
        np.savetxt("results.csv", label_list, delimiter =",", fmt='%i')
```

Apply cosine similarity between the penultimate layer of the testing set and the penultimate layer of the training set. Corresponding gesture of the training set vector with max cosine similarity is the recognition of the gesture. Recognize the gestures for all the test dataset videos and save the results(labels) to the results.csv file.

```
Accuracy % =  15.69
[14, 15, 4, 3, 4, 15, 15, 15, 4, 4, 15, 12, 12, 15, 14, 14, 0, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 14, 15, 13, 14, 14, 1
4, 0, 9, 15, 3, 1, 11, 4, 2, 3, 2, 2, 10, 0, 15, 3, 14, 16, 16]
```

I used the practice gesture videos generated in project Part 1, but my accuracy was super low 15.69%. I tried to record gesture videos again several time like Part 1, but the accuracy was not changed significantly. This is because the model performance is not great due to the overfit or underfit. Also, my hand shape and size, skin color, and background are totally different from the given test data. Since the model is pre-trained by someone, I cannot access and modify the deep learning CNN model.