Nama : Ronggur Mahendra Widya Putra
NIM   : 13519008

Deliverable :
Lampiran (Video, data setelah prosessing, dan data setelah di klasifikasi):
https://drive.google.com/drive/folders/1a2KeAgAA_yl6i7O70CVS5kvLCyGhmcl3?usp=sharing
Github: https://github.com/ronggurmahendra/TF4012-TugasBesarHandSign.git

Cat : cara eksekusi program terdapat pada README.md pada repository

1. Teori dasar

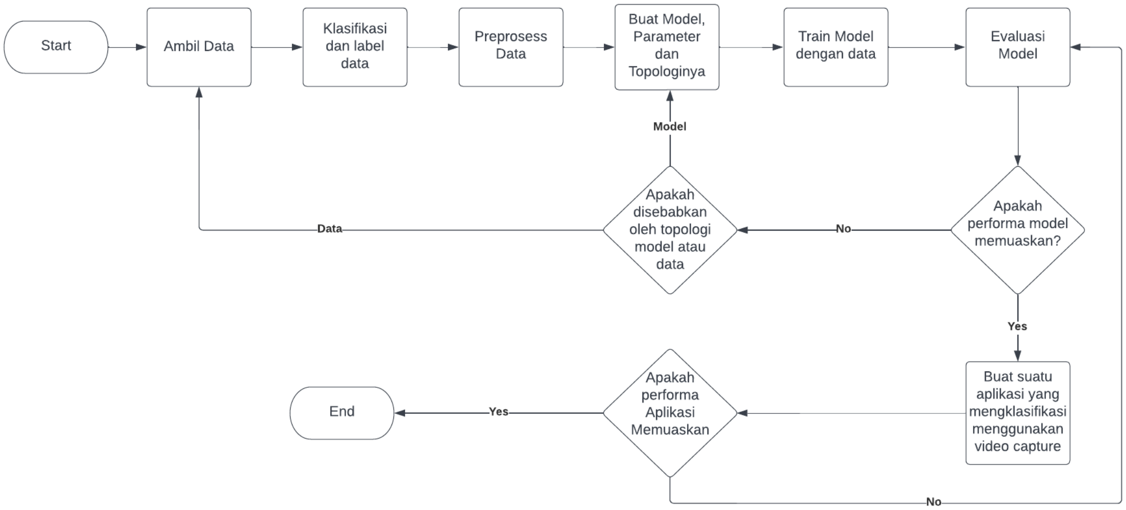
a. CNN(Convolutional Neural Network)

CNN(Convolutional Neural Network) merupakan suatu jenis dari ANN(Artificial Neural Network) yang di design untuk diterapkan pada input citra. CNN menentukan aspek apa saja dalam sebuah gambar yang bisa digunakan algoritma untuk belajar mengklasifikasi gambar.

b. Tensorflow dan keras

Tensorflow dan keras adalah suatu library artificial neural network. Tensorflow dan keras memudahkan developer untuk membuat model dan topologinya tanpa harus mengimplementasi banyak jenis layer pada ANN seperti layer konvolusi dan lain - lain.

2. Desain Eksperimen


Start
Ambil Data
Klasifikasi dan label data
Preprosess Data
Buat Model, Parameter dan Topologinya
Train Model dengan data
Evaluasi Model
Model
Apakah disebabkan oleh topologi model atau data
Data
No
Apakah performa model memuaskan?
Yes
Buat suatu aplikasi yang mengklasifikasi menggunakan video capture
Apakah performa Aplikasi Memuaskan
Yes
End
No

a. Pengambilan Data

Data diambil dengan pertama merekam suatu video hand sign, video tersebut kemudian diambil seriap framenya menjadi gambar dan dilabelkan ke kelas huruf alphabet(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y']). Gambar-gambar tersebut kemudian dilakukan preprocessing resize dan scale ke spesifikasi model yang dibuat. Kumpulan data tersebut kemudian dibagi menjadi 3 bagian yaitu training data, validation data, dan test data dengan rasio 8 : 1 : 1.

b. Pembuatan Model dan Training

Model pertama di buat definisi topologinya. Model tersebut kemudian di compile menggunakan beberapa parameter dan optimizer. Kemudian model tersebut di train menggunakan data yang sudah di buat. Model tersebut kemudian dievaluasi performanyanya dan di save untuk nanti aplikasi load.

c. Pembuatan Aplikasi

Aplikasi pertama meload model yang sudah di train, lalu menginisialisasi video capture lalu pada setiap frame video capture pertama-tama men preprosess frame sama dengan preprocessing training data lalu menggunakan model yang sudah dibuat sebelumnya men predict frame tersebut dan memberikannya ke user.

3. Algoritma

a. Topologi Model_1

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 64)        36928

 flatten (Flatten)           (None, 50176)             0

 dense (Dense)               (None, 120)               6021240

 dense_1 (Dense)             (None, 24)                2904

=================================================================
Total params: 6,080,464
Trainable params: 6,080,464
Non-trainable params: 0
```

```
_________________________________________________________________
```

b. Topologi Model_2

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_3 (Conv2D)           (None, 128, 128, 16)      448       
                                                                 
 conv2d_4 (Conv2D)           (None, 128, 128, 16)      2320      
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 127, 127, 16)     0         
 2D)                                                             
                                                                 
 dropout (Dropout)           (None, 127, 127, 16)      0         
                                                                 
 conv2d_5 (Conv2D)           (None, 125, 125, 32)      4640      
                                                                 
 conv2d_6 (Conv2D)           (None, 123, 123, 32)      9248      
                                                                 
 batch_normalization (BatchN  (None, 123, 123, 32)     128       
 ormalization)                                                   
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 61, 61, 32)       0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 61, 61, 32)        0         
                                                                 
 conv2d_7 (Conv2D)           (None, 59, 59, 32)        9248      
                                                                 
 conv2d_8 (Conv2D)           (None, 57, 57, 32)        9248      
                                                                 
 batch_normalization_1 (Batc  (None, 57, 57, 32)       128       
 hNormalization)                                                 
                                                                 
 conv2d_9 (Conv2D)           (None, 55, 55, 32)        9248      
                                                                 
 conv2d_10 (Conv2D)          (None, 53, 53, 32)        9248      
                                                                 
 batch_normalization_2 (Batc  (None, 53, 53, 32)       128       
 hNormalization)                                                 
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 26, 26, 32)       0         
 2D)                                                             
                                                                 
 dropout_2 (Dropout)         (None, 26, 26, 32)        0         
 flatten_1 (Flatten)        (None, 21632)            0

 dense_2 (Dense)            (None, 120)              2595960

 dense_3 (Dense)            (None, 120)              14520

 dense_4 (Dense)            (None, 24)               2904

=================================================================
Total params: 2,667,416
Trainable params: 2,667,224
Non-trainable params: 192
_________________________________________________________________
```

c. Topologi Model_3

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_11 (Conv2D)          (None, 128, 128, 16)      9424

 conv2d_12 (Conv2D)          (None, 115, 115, 32)      100384

 max_pooling2d_5 (MaxPooling  (None, 114, 114, 32)     0
 2D)

 conv2d_13 (Conv2D)          (None, 101, 101, 32)      200736

 conv2d_14 (Conv2D)          (None, 88, 88, 32)        200736

 max_pooling2d_6 (MaxPooling  (None, 44, 44, 32)       0
 2D)

 dropout_3 (Dropout)         (None, 44, 44, 32)        0

 conv2d_15 (Conv2D)          (None, 31, 31, 64)        401472

 conv2d_16 (Conv2D)          (None, 18, 18, 64)        802880

 batch_normalization_3 (Batc  (None, 18, 18, 64)       256
 hNormalization)

 max_pooling2d_7 (MaxPooling  (None, 9, 9, 64)         0
 2D)
 dropout_4 (Dropout)        (None, 9, 9, 64)         0

 flatten_2 (Flatten)        (None, 5184)             0

 dense_5 (Dense)            (None, 128)              663680

 dense_6 (Dense)            (None, 128)              16512

 dense_7 (Dense)            (None, 24)               3096

=================================================================
Total params: 2,399,176
Trainable params: 2,399,048
Non-trainable params: 128
_________________________________________________________________
```

4. Hasil Evaluasi

Model 1 :

Train Accuracy : 100.00 %
Validation Accuracy : 99.68%
Test Accuracy : 95.83%

Model 2 :

Train Accuracy : 100.00 %
Validation Accuracy : 6.49 %
Test Accuracy : 6.09%

Model 3 :

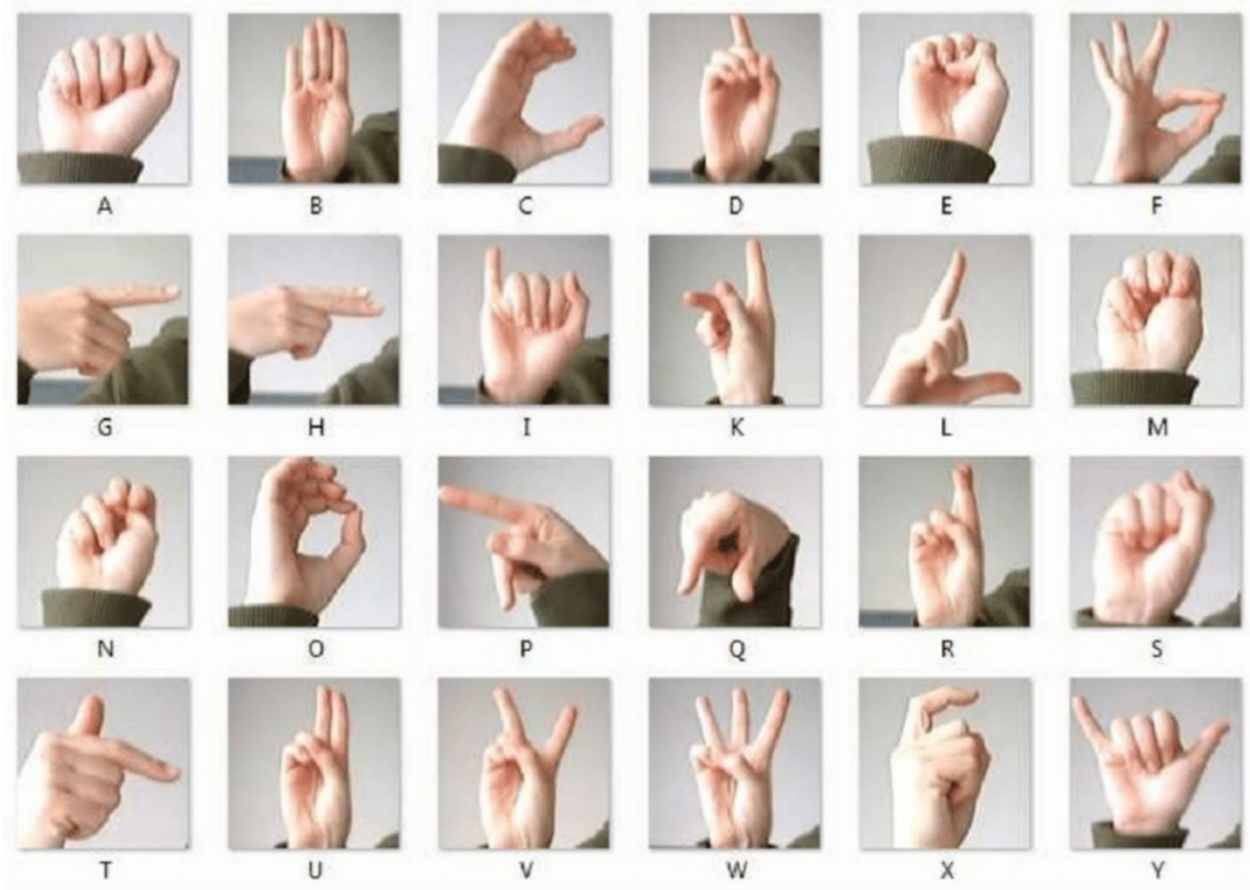
Train Accuracy : 99.42%
Validation Accuracy : 99.68%
Test Accuracy : 100.00%

5. Analisis

Berdasarkan hasil evaluasi Model_1 dan Model_3 yang mendapatkan accuracy yang memuaskan dan Model_2 walaupun mendapatkan training accuracy yang bagus mendapatkan validation accuracy dan test accuracy yang rendah yang mengindikasikan pada Model_2 terjadi overfit. Pada aplikasi yang dibuat Model_3 yang memberikan klasifikasi yang terbaik dibandingkan 2 model lainya. Aplikasi berhasil mengklasifikasi secara real-time namun memiliki kesulitan mengidentifikasi handsign dengan feature yang mirip i.e. : A, E, M, S(menggenggam), hal ini terjadi karena training data yang homogen karena berasal dari video yang sama dan bisa diselesaikan dengan menambah variasi pada training data.

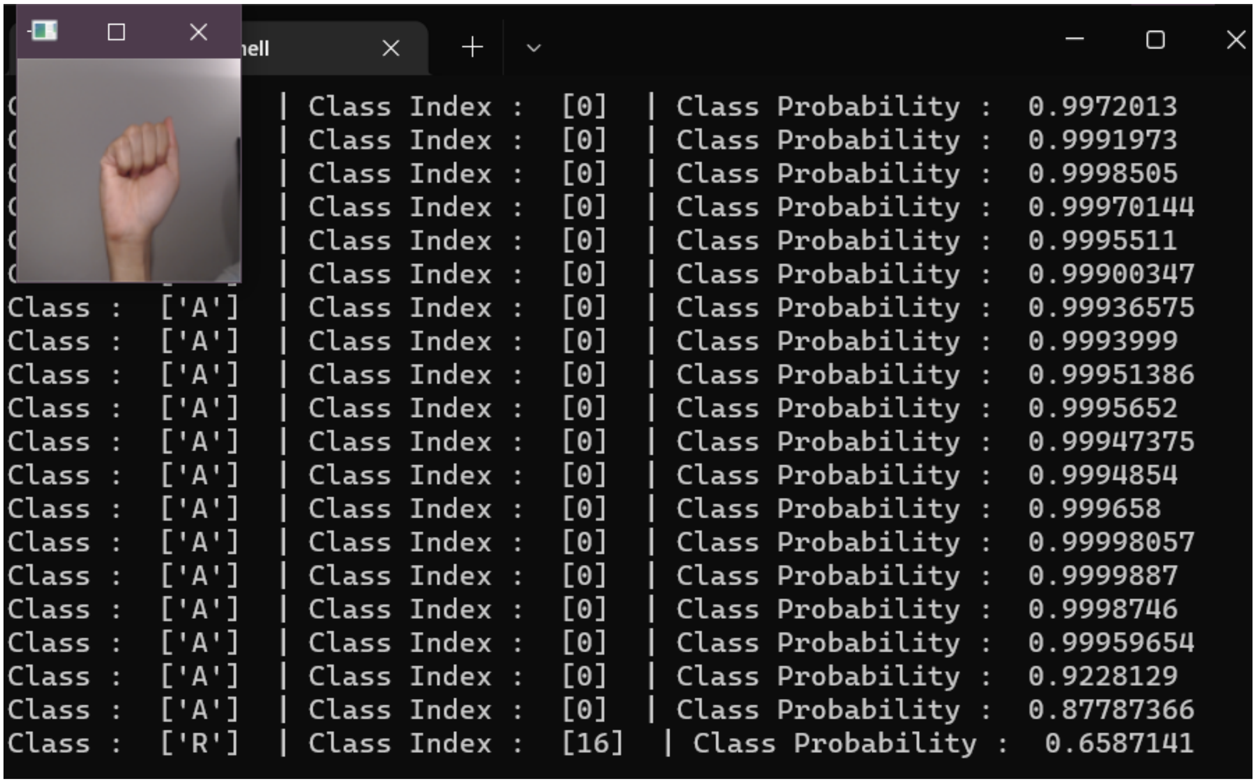
Lampiran

Referensi Hand Sign



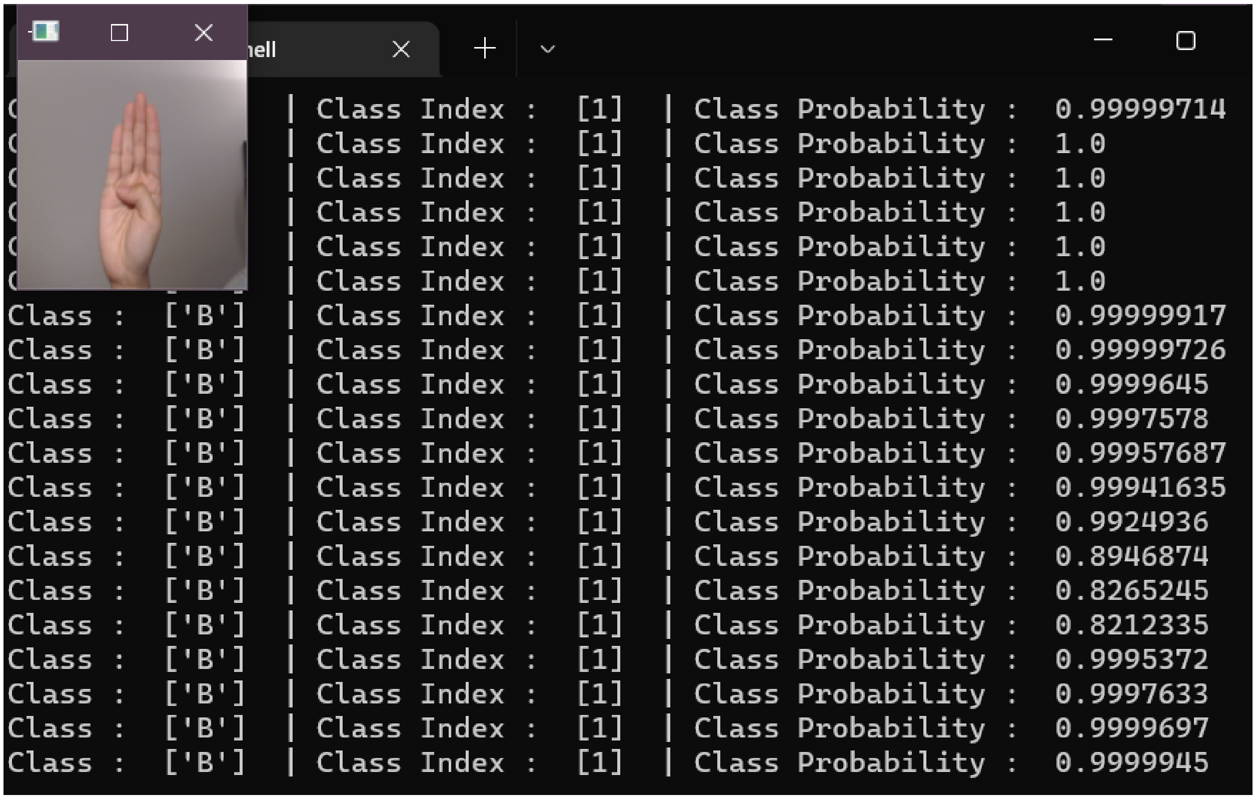
Screenshot aplikasi yang dibuat

Test Case “A” :



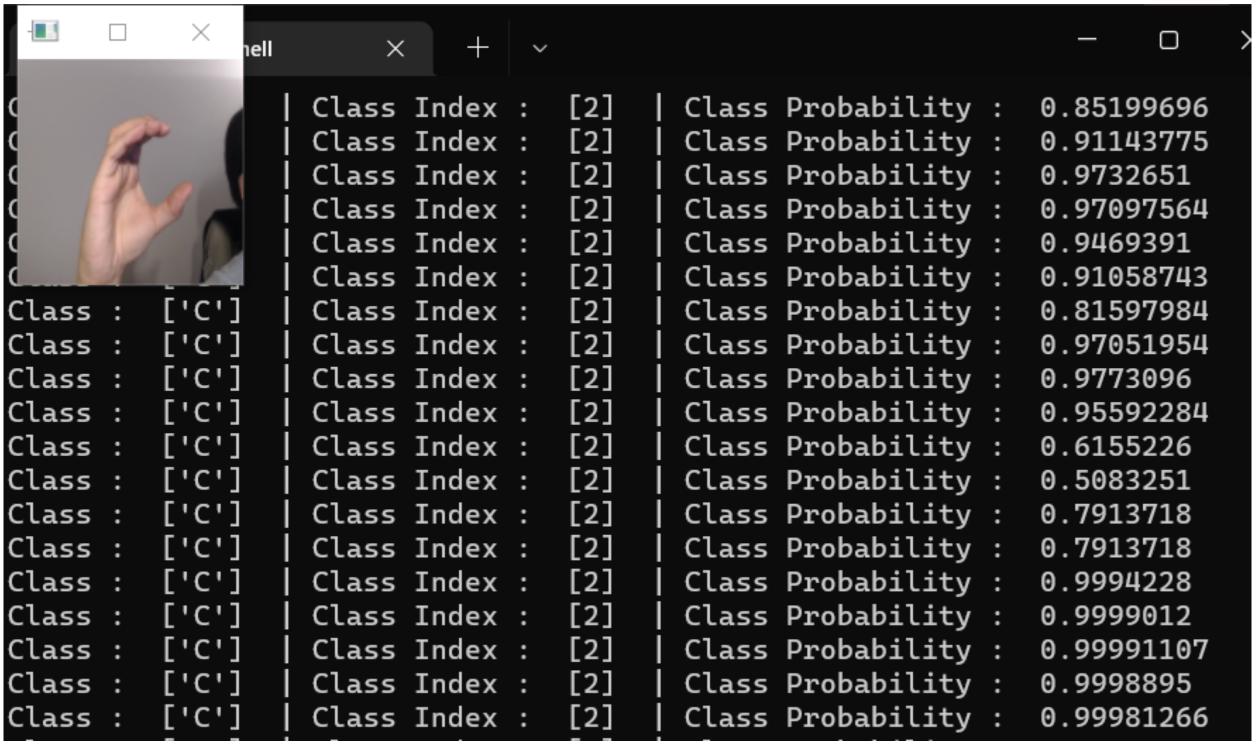
```
C               | Class Index :  [0]  | Class Probability :  0.9972013
C               | Class Index :  [0]  | Class Probability :  0.9991973
C               | Class Index :  [0]  | Class Probability :  0.9998505
C               | Class Index :  [0]  | Class Probability :  0.99970144
C               | Class Index :  [0]  | Class Probability :  0.9995511
C               | Class Index :  [0]  | Class Probability :  0.99900347
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.99936575
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.9993999
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.99951386
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.9995652
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.99947375
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.9994854
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.999658
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.99998057
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.9999887
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.9998746
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.99959654
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.9228129
Class :  ['A']  | Class Index :  [0]  | Class Probability :  0.87787366
Class :  ['R']  | Class Index :  [16]  | Class Probability :  0.6587141
```

Test Case “B” :



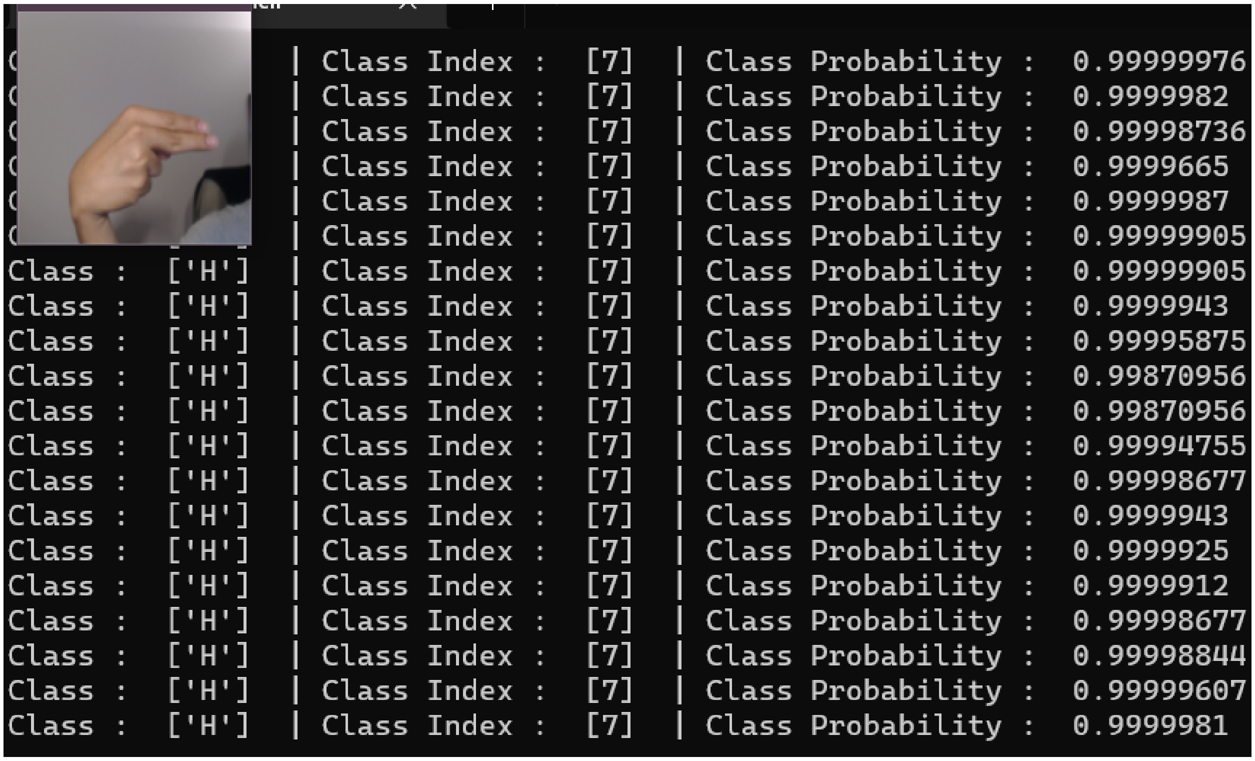
```
C                  | Class Index :  [1]  | Class Probability :  0.99999714
C                  | Class Index :  [1]  | Class Probability :  1.0
C                  | Class Index :  [1]  | Class Probability :  1.0
C                  | Class Index :  [1]  | Class Probability :  1.0
C                  | Class Index :  [1]  | Class Probability :  1.0
C                  | Class Index :  [1]  | Class Probability :  1.0
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.99999917
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.99999726
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.9999645
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.9997578
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.99957687
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.99941635
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.9924936
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.8946874
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.8265245
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.8212335
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.9995372
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.9997633
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.9999697
Class :  ['B']  | Class Index :  [1]  | Class Probability :  0.9999945
```

Test Case “C” :



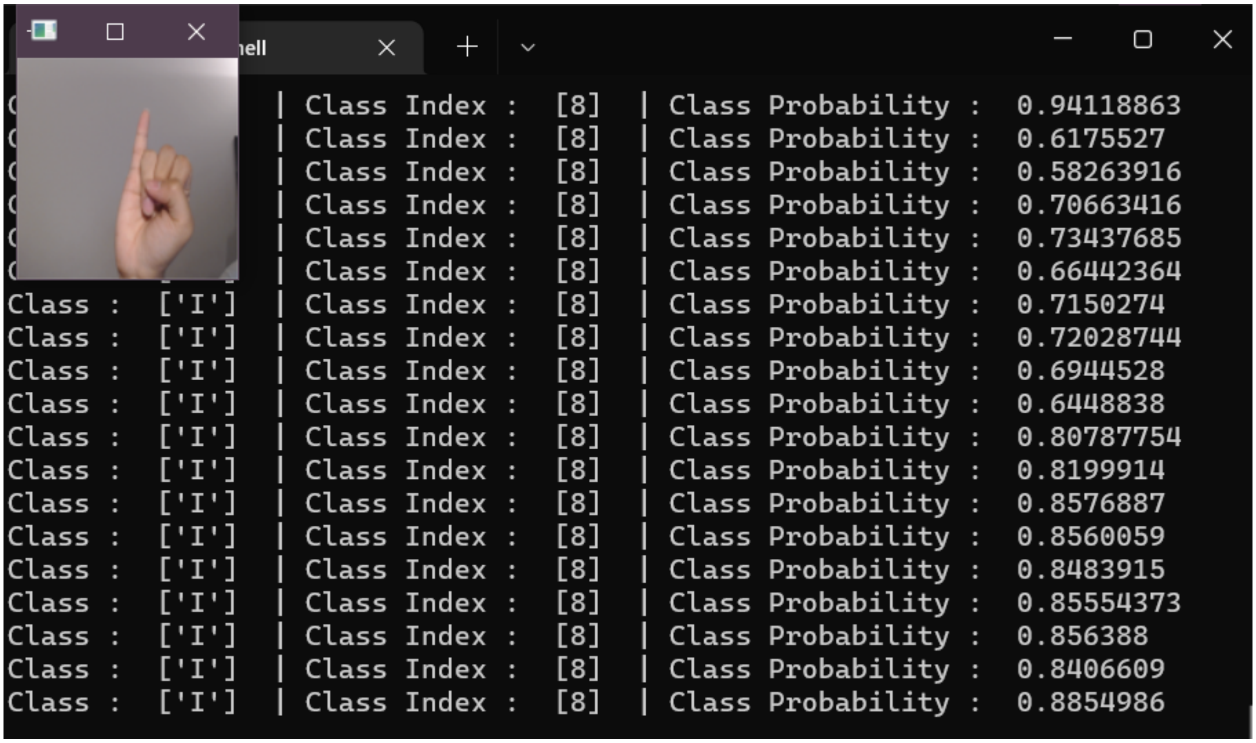
```
C                  | Class Index :  [2]  | Class Probability :  0.85199696
C                  | Class Index :  [2]  | Class Probability :  0.91143775
C                  | Class Index :  [2]  | Class Probability :  0.9732651
C                  | Class Index :  [2]  | Class Probability :  0.97097564
C                  | Class Index :  [2]  | Class Probability :  0.9469391
C                  | Class Index :  [2]  | Class Probability :  0.91058743
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.81597984
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.97051954
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.9773096
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.95592284
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.6155226
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.5083251
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.7913718
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.7913718
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.9994228
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.9999012
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.99991107
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.9998895
Class :  ['C']  | Class Index :  [2]  | Class Probability :  0.99981266
```

Test Case “H” :



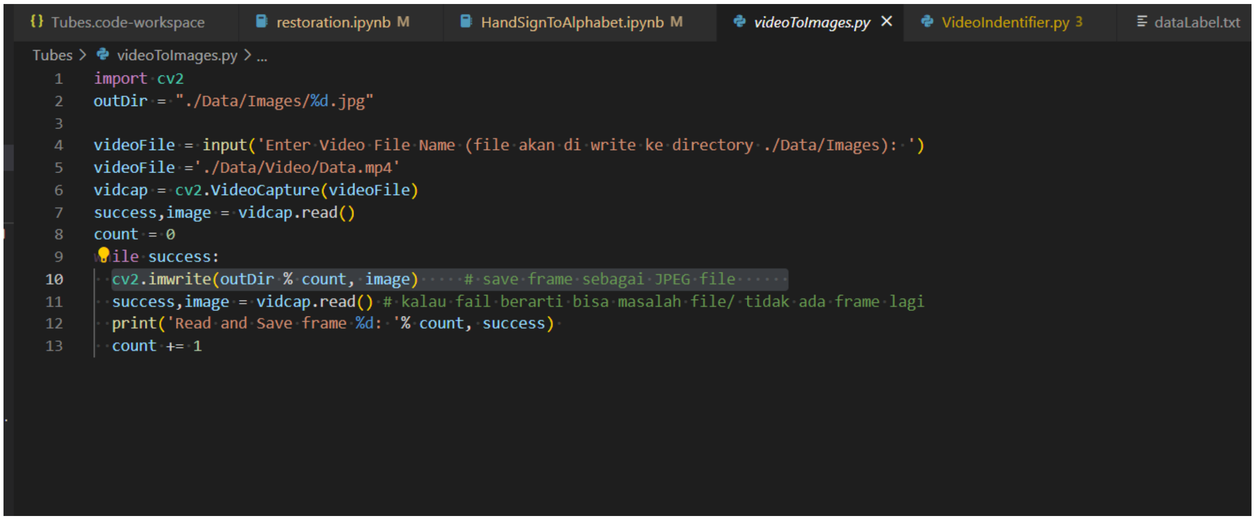
```
C                | Class Index :  [7]  | Class Probability :  0.99999976
C                | Class Index :  [7]  | Class Probability :  0.9999982
C                | Class Index :  [7]  | Class Probability :  0.99998736
C                | Class Index :  [7]  | Class Probability :  0.9999665
C                | Class Index :  [7]  | Class Probability :  0.9999987
C                | Class Index :  [7]  | Class Probability :  0.99999905
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99999905
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.9999943
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99995875
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99870956
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99870956
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99994755
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99998677
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.9999943
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.9999925
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.9999912
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99998677
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99998844
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.99999607
Class :  ['H']  | Class Index :  [7]  | Class Probability :  0.9999981
```

Test Case “I” :



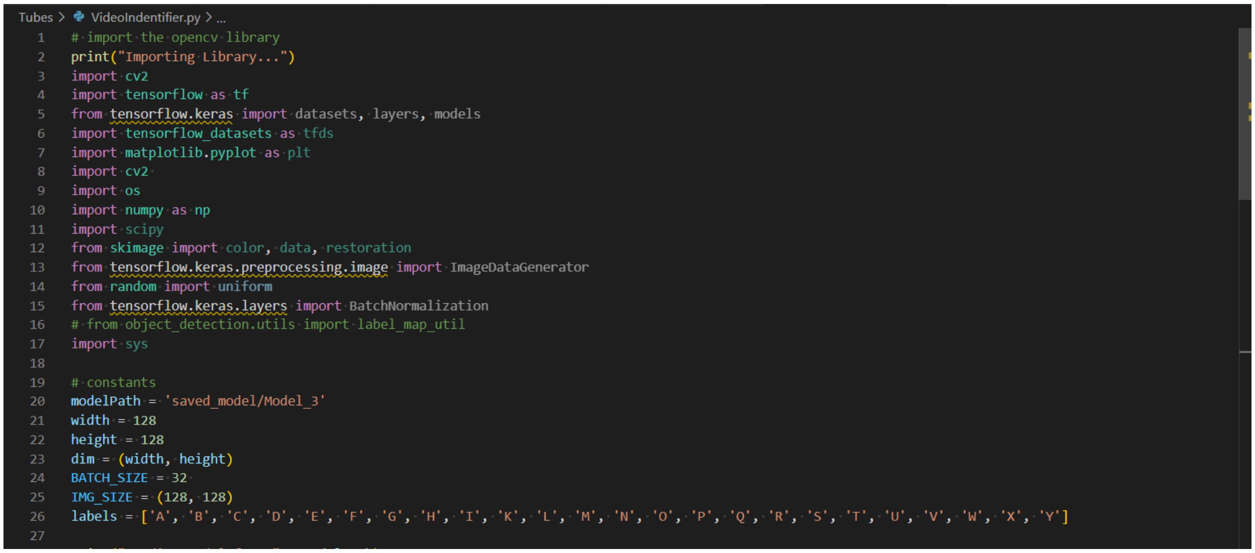
```
C                | Class Index :  [8]  | Class Probability :  0.94118863
C                | Class Index :  [8]  | Class Probability :  0.6175527
C                | Class Index :  [8]  | Class Probability :  0.58263916
C                | Class Index :  [8]  | Class Probability :  0.70663416
C                | Class Index :  [8]  | Class Probability :  0.73437685
C                | Class Index :  [8]  | Class Probability :  0.66442364
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.7150274
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.72028744
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.6944528
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.6448838
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.80787754
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.8199914
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.8576887
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.8560059
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.8483915
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.85554373
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.856388
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.8406609
Class :  ['I']  | Class Index :  [8]  | Class Probability :  0.8854986
```

Skrip VideoToImages.py (mengubah video menjadi images):

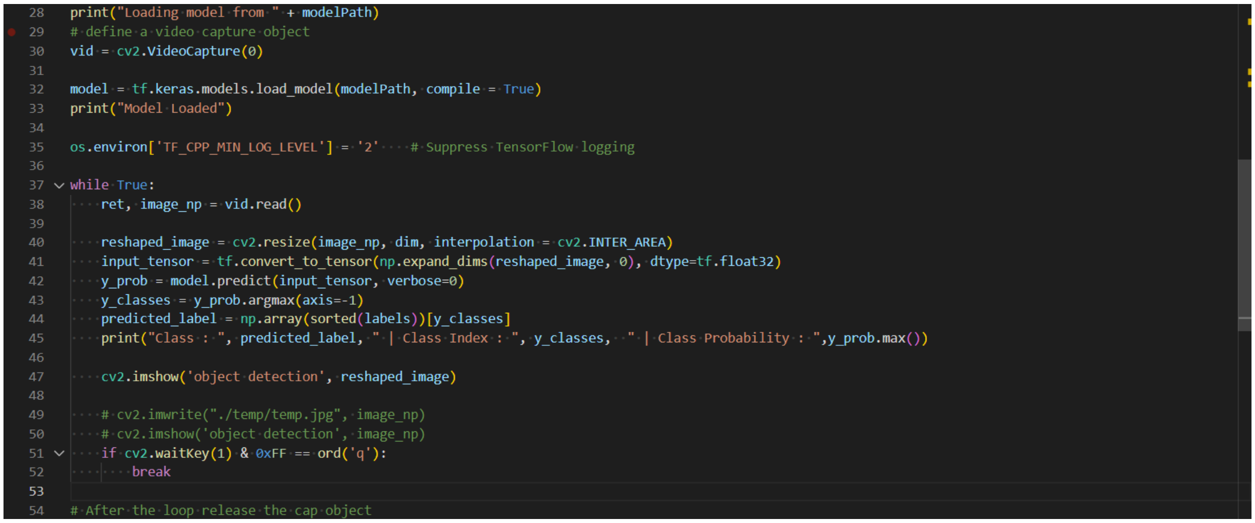


```python
import cv2
outDir = "./Data/Images/%d.jpg"

videoFile = input('Enter Video File Name (file akan di write ke directory ./Data/Images): ')
videoFile ='./Data/Video/Data.mp4'
vidcap = cv2.VideoCapture(videoFile)
success,image = vidcap.read()
count = 0
while success:
  cv2.imwrite(outDir % count, image)     # save frame sebagai JPEG file
  success,image = vidcap.read() # kalau fail berarti bisa masalah file/ tidak ada frame lagi
  print('Read and Save frame %d: '% count, success)
  count += 1
```

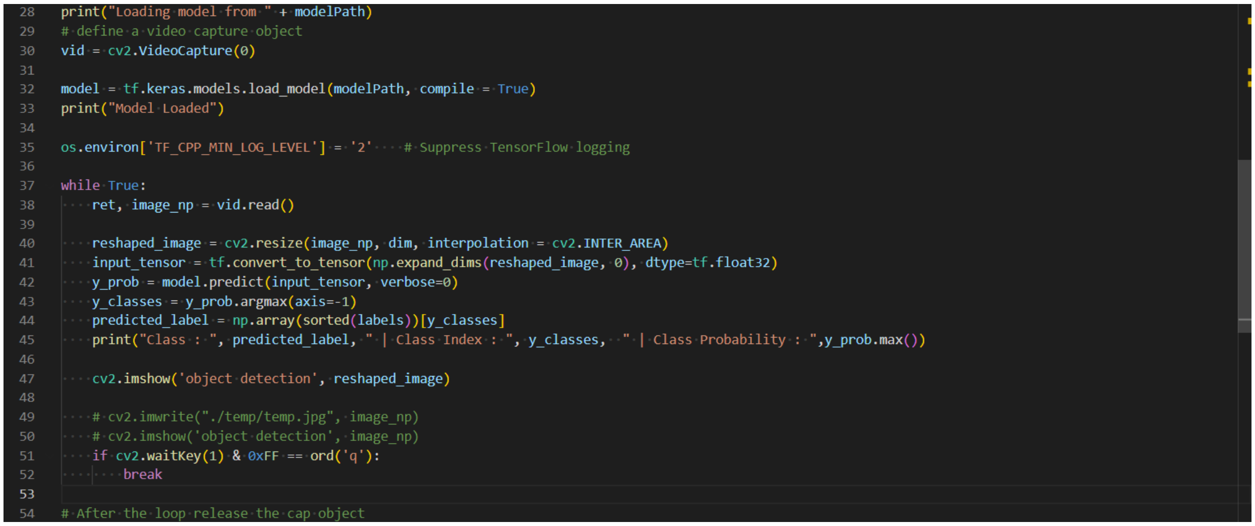
Skrip VideoIndentifier.py (untuk indentifikasi video realtime)



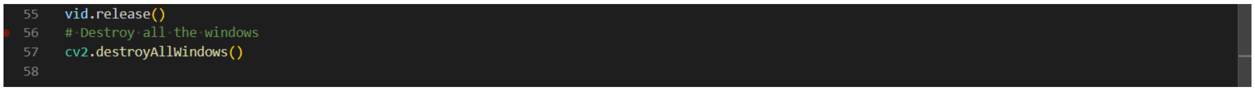
```python
# import the opencv library
print("Importing Library...")
import cv2
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
import cv2
import os
import numpy as np
import scipy
from skimage import color, data, restoration
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from random import uniform
from tensorflow.keras.layers import BatchNormalization
# from object_detection.utils import label_map_util
import sys

# constants
modelPath = 'saved_model/Model_3'
width = 128
height = 128
dim = (width, height)
BATCH_SIZE = 32
IMG_SIZE = (128, 128)
labels = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y']

```



```python
print("Loading model from " + modelPath)
# define a video capture object
vid = cv2.VideoCapture(0)

model = tf.keras.models.load_model(modelPath, compile = True)
print("Model Loaded")

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging

while True:
    ret, image_np = vid.read()

    reshaped_image = cv2.resize(image_np, dim, interpolation = cv2.INTER_AREA)
    input_tensor = tf.convert_to_tensor(np.expand_dims(reshaped_image, 0), dtype=tf.float32)
    y_prob = model.predict(input_tensor, verbose=0)
    y_classes = y_prob.argmax(axis=-1)
    predicted_label = np.array(sorted(labels))[y_classes]
    print("Class : ", predicted_label, " | Class Index : ", y_classes,  " | Class Probability : ",y_prob.max())

    cv2.imshow('object detection', reshaped_image)

    # cv2.imwrite("./temp/temp.jpg", image_np)
    # cv2.imshow('object detection', image_np)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# After the loop release the cap object
```



```python
print("Loading model from " + modelPath)
# define a video capture object
vid = cv2.VideoCapture(0)

model = tf.keras.models.load_model(modelPath, compile = True)
print("Model Loaded")

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging

while True:
    ret, image_np = vid.read()

    reshaped_image = cv2.resize(image_np, dim, interpolation = cv2.INTER_AREA)
    input_tensor = tf.convert_to_tensor(np.expand_dims(reshaped_image, 0), dtype=tf.float32)
    y_prob = model.predict(input_tensor, verbose=0)
    y_classes = y_prob.argmax(axis=-1)
    predicted_label = np.array(sorted(labels))[y_classes]
    print("Class : ", predicted_label, " | Class Index : ", y_classes,  " | Class Probability : ",y_prob.max())

    cv2.imshow('object detection', reshaped_image)

    # cv2.imwrite("./temp/temp.jpg", image_np)
    # cv2.imshow('object detection', image_np)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# After the loop release the cap object
```



```python
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()

```

Skrip HardSignToAlphabet.ipynb (untuk training) :