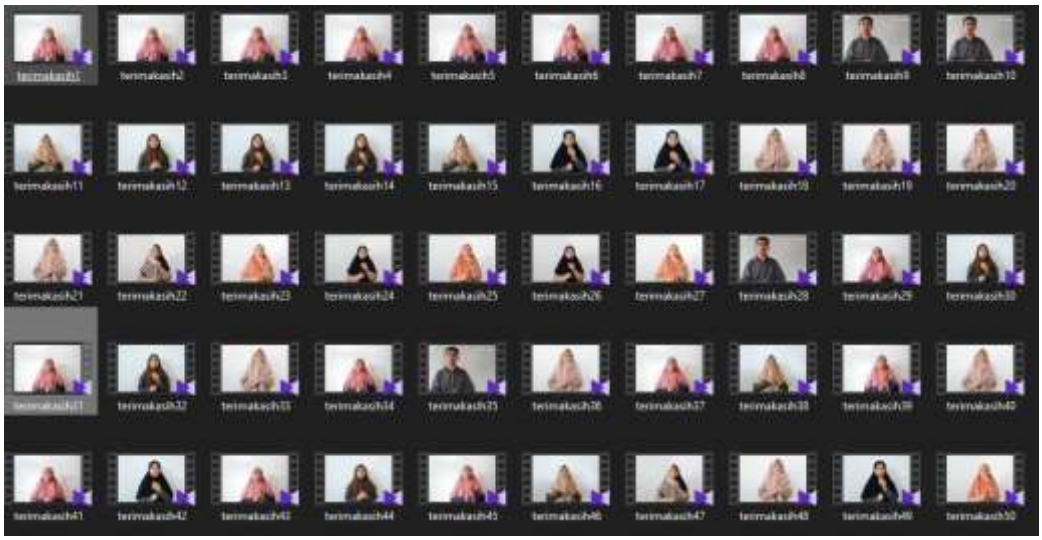
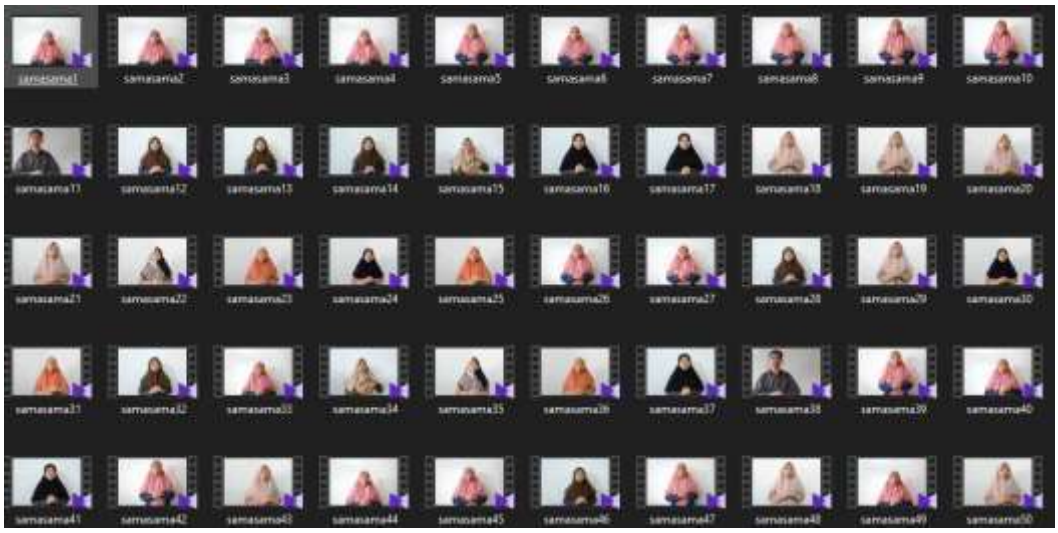
LAMPIRAN

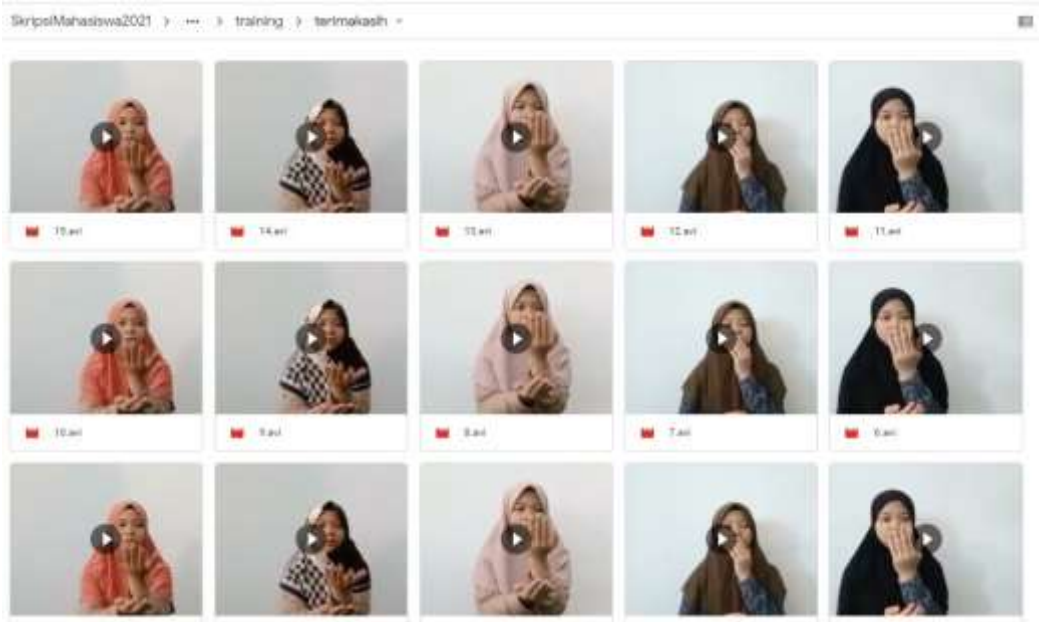
Lampiran 1 *Dataset*



Lanjutan lampiran 1

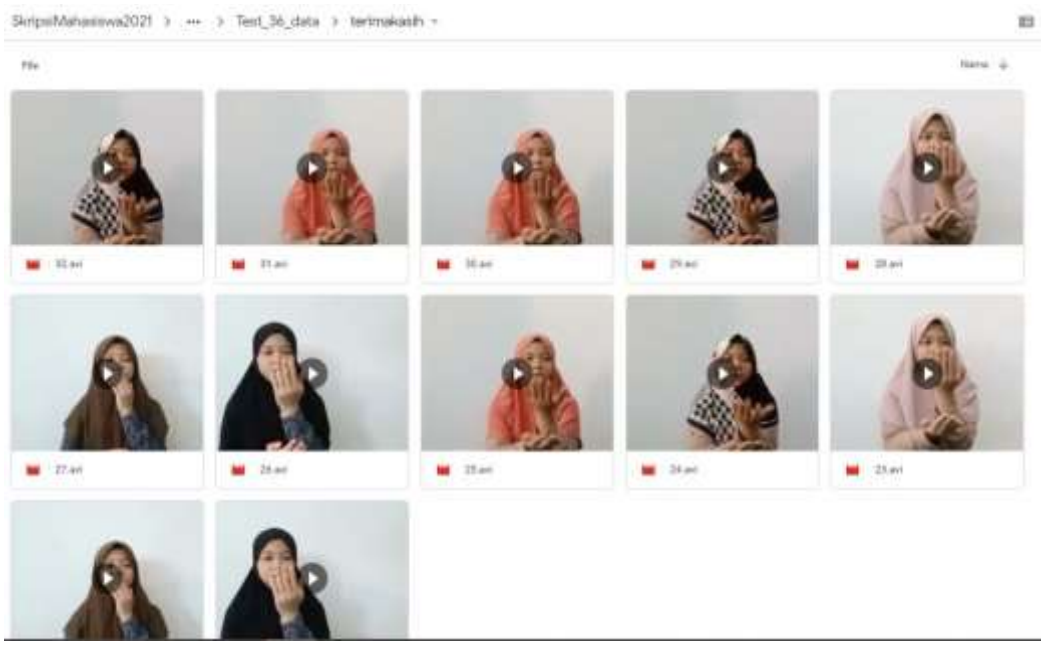


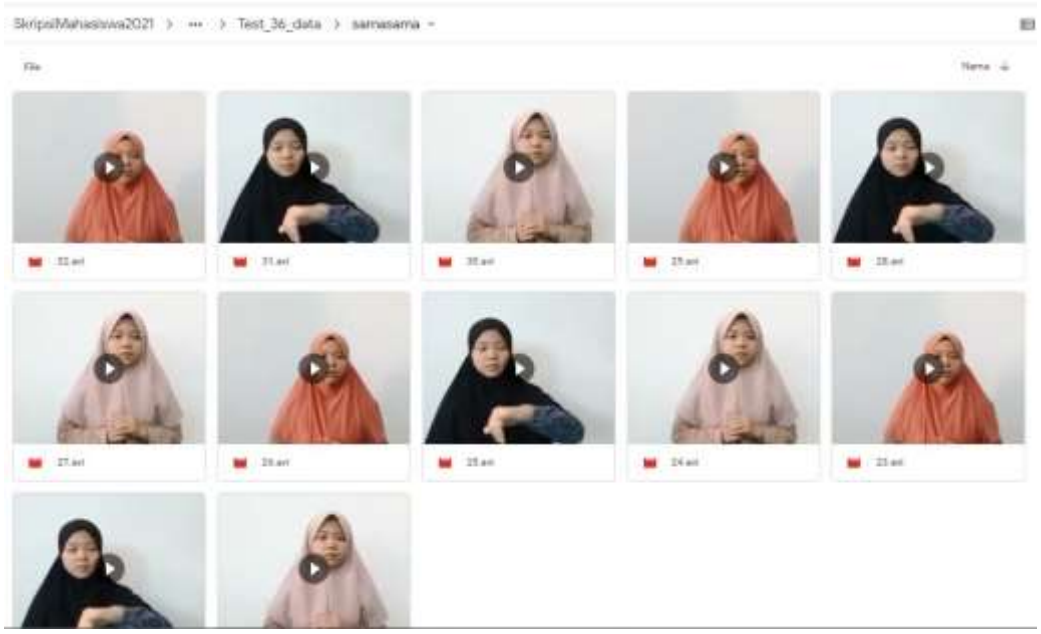
Lampiran 2. Data *Training*



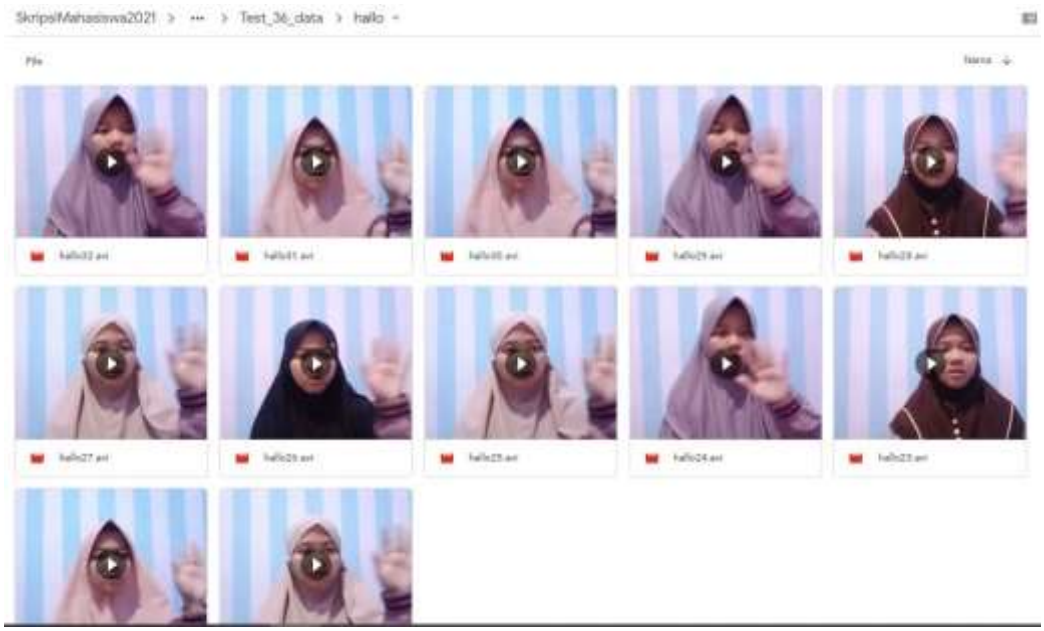
Lanjutan lampiran 2

Lampiran 3. Data *Testing*





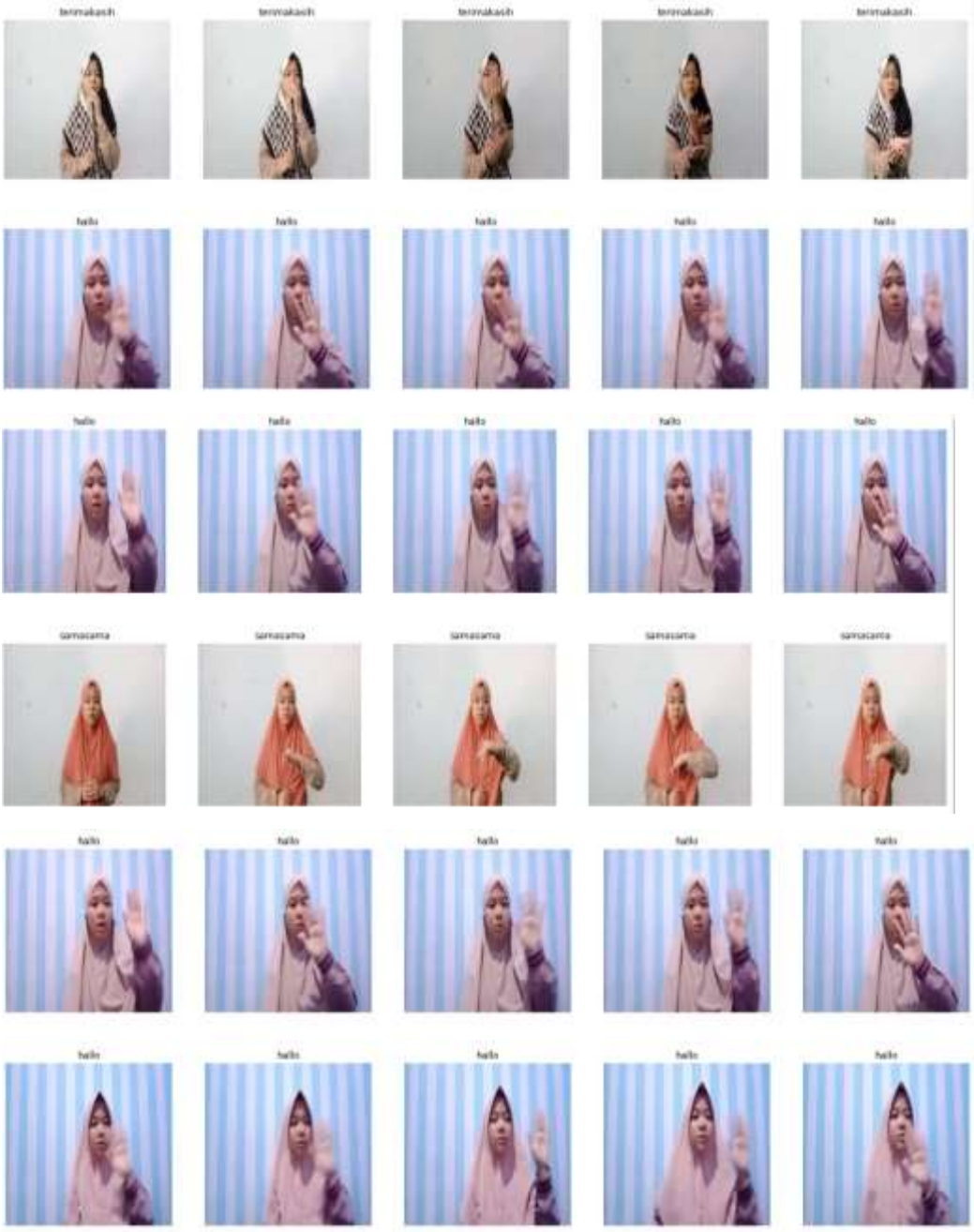
Lanjutan lampiran 3



Lampiran 4 Hasil *Training*

Lanjutan lampiran 4

Lampiran 5 Hasil uji coba



Lampiran 6 Segmen Program Data Training

```
#import Tensorflow
%tensorflow_version 1.x
import tensorflow
import matplotlib.pyplot as plt
print(tensorflow.version)
!pip install 'h5py<3.0.0'
!pip install keras-video-generators
from google.colab import drive
drive.mount('/content/drive')
import os
import glob
import keras
import torch
import keras_video
from keras_video import VideoFrameGenerator
from keras_video import utils as ku
print(keras.__version__)
# Check PyTorch dan GPU
from IPython.display import Image, clear_output  # to display
images
clear_output()
print(f"Setup complete. Using torch {torch.__version__} ({torc
h.cuda.get_device_properties(0).name if torch.cuda.is_availabl
e() else 'CPU'})")
## use sub directories names as classes
classes = [i.split(os.path.sep)[8] for i in glob.glob('/conten
t/drive/MyDrive/SKRIPSI 2021/SKRIPSI PROGRAM/DATASET/training/
*')]
classes.sort()
print(classes)
# some global params
SIZE = (224, 224)
CHANNELS = 3
NBFRAME = 5
BS = 8
# pattern to get videos and classes
glob_pattern_train='/content/drive/MyDrive/SKRIPSI 2021/SKRIPS
I PROGRAM/DATASET/training/{classname}/*.avi'
glob_pattern_test='/content/drive/MyDrive/SKRIPSI 2021/SKRIPSI
 PROGRAM/DATASET/Test_36_data/{classname}/*.avi'
# for data augmentation
data_aug = keras.preprocessing.image.ImageDataGenerator(
    zoom_range=.1,
    horizontal_flip=True,
    rotation_range=8,
```

```
    width_shift_range=.2,
    height_shift_range=.2)
# Create video frame generator
gen  =  keras_video . SlidingFrameGenerator ( sequence_time =
.5 ,  batch_size = 8 ,nb_frames = 5 ,  glob_pattern ='/content
/drive/MyDrive/SKRIPSI 2021/SKRIPSI PROGRAM/DATASET/training/{
classname}/*',split_val = .2)
 # Create video frame generator
test  =  keras_video . SlidingFrameGenerator ( sequence_time =
 .5 ,  batch_size = 8 ,
nb_frames = 5 ,  glob_pattern ='/content/drive/MyDrive/SKRIPSI
 2021/SKRIPSI PROGRAM/DATASET/Test_36_data/{classname}/*',)
ku.show_sample(gen, random=True)
# Pembuatan Model
from keras.layers import Conv2D, BatchNormalization, \
    MaxPool2D, GlobalMaxPool2D
def build_convnet(shape=(224, 224, 3)):
    momentum = 0.2
    model = keras.Sequential()
    model.add(Conv2D(64, (3,3), input_shape=shape,
        padding='same', activation='relu'))
    model.add(Conv2D(64, (3,3), padding='same', activation=
'relu'))
    model.add(BatchNormalization(momentum=momentum))

    model.add(MaxPool2D())

    model.add(Conv2D(128, (3,3), padding='same', activation=
'relu'))
    model.add(Conv2D(128, (3,3), padding='same', activation='
relu'))
    model.add(BatchNormalization(momentum=momentum))

    model.add(MaxPool2D())

    model.add(Conv2D(256, (3,3), padding='same', activation=
'relu'))
    model.add(Conv2D(256, (3,3), padding='same', activation=
'relu'))
    model.add(BatchNormalization(momentum=momentum))

    model.add(MaxPool2D())

    model.add(Conv2D(512, (3,3), padding='same', activation=
'relu'))
    model.add(Conv2D(512, (3,3), padding='same', activation='
relu'))
    model.add(BatchNormalization(momentum=momentum))
```

```
     # flatten...
    model.add(GlobalMaxPool2D())
    return model
from keras.layers import TimeDistributed, GRU, Dense, Dropout
```

```
def action_model(shape=(5, 224, 224, 3), nbout=3):
    # Create our convnet with (224, 224, 3) input shape
    convnet = build_convnet(shape[1:])

    # then create our final model
    model = keras.Sequential()
    # add the convnet with (10, 224, 224, 2) shape
    model.add(TimeDistributed(convnet, input_shape=shape))
    # here, you can also use GRU or LSTM
    model.add(GRU(64))
    # and finally, we make a decision network
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(nbout, activation='softmax'))
    return model
INSHAPE=(NBFRAME,) + SIZE + (CHANNELS,) # (5, 224, 224, 3)
model = action_model(INSHAPE, len(classes))
optimizer = keras.optimizers.Adam(0.001)
model.compile(
    optimizer,
    'categorical_crossentropy',
    metrics=['acc']
)
EPOCHS=10
# create a "chkp" directory before to run that
# because ModelCheckpoint will write models inside
callbacks = [
    keras.callbacks.ReduceLROnPlateau(verbose=1),
]
log = model.fit_generator(
    gen,
    validation_data=gen,
    verbose=1,
    epochs=EPOCHS,
    # steps_per_epoch=train.files_count*NBFRAME//BS,
    # validation_steps=gen.files_count*NBFRAME//BS,
    # callbacks=callbacks
)
#mengambil data hasil akurasi dan loss dari model
accuracy = log.history['acc']
val_accuracy = log.history['val_acc']
loss = log.history['loss']
```

```
val_loss = log.history['val_loss']

#menampilkan pada figur
plt.figure(figsize=(10,10))
plt.subplot(1, 2, 1)
plt.title('Akurasi Data Training dan Validation')
plt.xlabel('Epoch')
plt.ylabel('Value')
plt.plot(accuracy, label='Akurasi Data Training') #membuat plo
t berdasarkan data
plt.plot(val_accuracy, label='Akurasi Data Validation')  #memb
ua
plt.legend(loc="lower right")

plt.subplot(1, 2, 2)
plt.title('Loss Data Training dan Validation')
plt.xlabel('Epoch')
plt.ylabel('Value')
plt.plot(loss, label='Loss Data Training')  #membuat plot berd
asarkan data
plt.plot(val_loss, label='Loss Data Validation')  #membuat plo
t berdasarkan data
plt.legend(loc="upper right")

plt.show()
```

Lampiran 7 Segmen Program Testing

```
%tensorflow_version 1.x
import tensorflow
print(tensorflow.version)
!pip install 'h5py<3.0.0'
!pip install keras-video-generators
from google.colab import drive
drive.mount('/content/drive')
import cv2
import numpy as np
from keras.models import load_model
from keras.preprocessing import image
model = load_model('/content/drive/MyDrive/SKRIPSI 2021/SKRIPSI
 PROGRAM/DATASET/modellagii_NBFrame5BS8_epoch10.h5')
model.summary()
import os
import glob
import keras
from keras_video import VideoFrameGenerator
classes = [i.split(os.path.sep)[8] for i in glob.glob('/content
/drive/MyDrive/SKRIPSI 2021/SKRIPSI PROGRAM/DATASET/Test_36_dat
a/*')]
classes.sort()
print(classes)
glob_pattern_test='/content/drive/MyDrive/SKRIPSI 2021/SKRIPSI
PROGRAM/DATASET/Test_36_data/{classname}/*.avi'
# some global params
SIZE = (224, 224)
CHANNELS = 3
NBFRAME = 5
BS = 8
test = VideoFrameGenerator(
  classes=classes,
  glob_pattern=glob_pattern_test,
  nb_frames=NBFRAME,
  shuffle=False,
  batch_size=BS,
  target_shape=SIZE,
  nb_channel=CHANNELS,
  use_frame_cache=True)
import keras_video.utils
keras_video.utils.show_sample(test)
```

```
from sklearn.metrics import classification_report

y_test = []
y_predict = []
for x in range(test.__len__()):
    batch = test.__getitem__(x)[1]
    batch_predicted = model.predict(test.__getitem__(x)[0])

    for y in range(BS):
        y_test.append(batch[y])
        y_predict.append(batch_predicted[0])

y_test = np.argmax(y_test, axis=1)
y_predict = np.argmax(y_predict, axis=1)
y_hasil = y_test==y_predict
print(classification_report(y_test, y_predict))
print("Data Testing")
print(y_test)
print("Hasil Prediksi")
print(y_predict)
print("Hasil Akurasi")
print(y_hasil)
```