BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil

Pada bab ini akan dibahas hasil dari penelitian deteksi tangan otomatis pada video percakapan bahasa indonesia (BISINDO). Serta akan dijelaskan terkait dengan metodologi yang telah disebutkan di bab sebelumnya yang meliputi hasil pengumpulan *dataset*, hasil *preprocessing*, hasil dari implementasi metode *deep gated recurrent unit* (GRU), dan hasil dari uji coba pada metode yang telah digunakan.

4.2 Pengumpulan *Dataset*

Pada penelitian ini data yang diambil adalah dengan membuat sebuah video berbahasa isyarat indonesia (BISINDO) berupa isyarat percakapan sehari- hari dimana di dalamnya terdapat 3 gerakan isyarat percakapan sehari- hari. Di Dalam setiap video berisi relawan yang mempraktekkan video bahasa isyarat sehari hari seperti hallo, terimakasih, dan sama-sama. Pengambilan video ini dilakukan dengan jarak kurang lebih 70 cm antara objek dan kamera dengan posisi kamera tetap harus lurus dengan objek. Pakaian yang digunakan disini bebas dan tangan tidak menutupi wajah saat pengambilan video berlangsung. Hasil video yang telah diambil ini nantinya akan dijadikan data *training* dan data *testing.*

Resolusi video adalah 1280 x 720 *pixel* dengan durasi video 39 menit dengan format *.mp4. Video gerakan bahasa isyarat indonesia ini belum bisa diolah ke dalam sistem karena ukuran video yang terlalu besar maka perlu dilakukan *compression* agar video dapat di proses oleh sistem dengan baik dan cepat. Data video gerakan bahasa isyarat indonesia asli ini harus diproses dalam tahap *pre-processing*. Hasil data video BISINDO yang terkumpul terlihat pada Gambar 4.1.

Gambar 4.1. Hasil video bisindo yang terkumpul

4.3 *Pre-Processing*

Pada bagian ini data yang diperoleh dari pengambilan *dataset* kemudian diolah terlebih dahulu agar mendapatkan kualitas video yang baik. Hal ini dilakukan agar memudahkan proses selanjutnya. Pada tahap *preprocessing* ini adalah memotong video dengan membuang hal-hal yang tidak dibutuhkan dalam video tersebut sehingga dapat menjadi sebuah video yang sempurna dengan penggunaan bahasa isyarat yang benar. Setelah itu mengumpulkan video sesuai dengan gerakan yang akan digunakan, serta merubah format video ke dalam bentuk *.AVI.

Data video yang terkumpul berjumlah 81 video yang di dalamnya ada 3 gerakan bahasa isyarat sehari hari. Video tersebut akan dibagi menjadi data *training* dan data *testing*, untuk data *training* menggunakan 45 video dan data *testing* menggunakan 36 video. Ukuran data video harus disamakan yaitu 704 x 384 *pixel* agar sistem dapat membaca lebih baik ketika diproses pada metode *Gated Recurrent Unit.* Adapun proses pemotongan video ditunjukkan pada Gambar 4.2.

Gambar 4.2. Video asli

Gambar 4.3. Proses pemotongan video

4.4 Hasil Implementasi Metode *Deep Gated Recurrent Unit* (GRU)

Pada subbab ini akan dijelaskan hasil dari implementasi metode *deep gated recurrent unit* (GRU) dengan menggunakan *google colab* yang terdiri dari beberapa tahapan:

4.4.1 *Import Tensorflow*

Tahap awal yang harus dilakukan ialah memasukkan *library tensorflow* kedalam *google colab*. Keberadaan *tensorflow* ini digunakan untuk mempercepat performa pada platform *Google colab*, serta dapat menangani detail dibalik layar dengan baik. *Code import tensorflow* terlihat pada Segmen Program 4.1.

```
%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
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__)
```

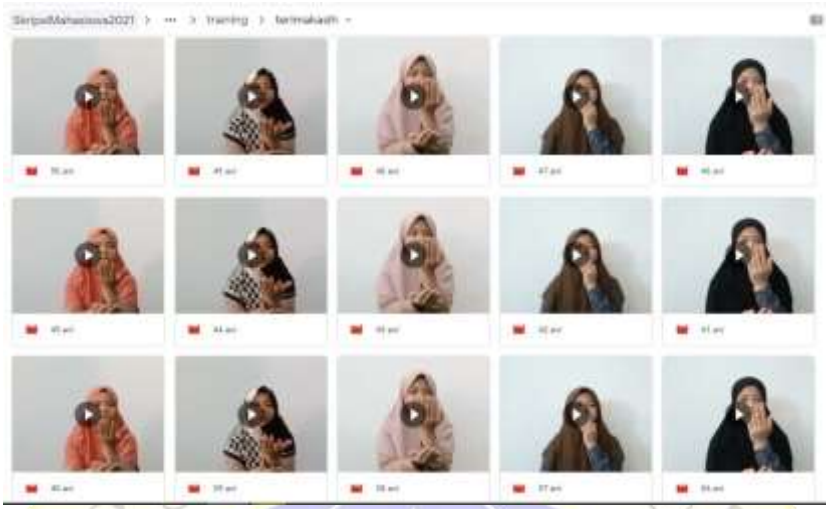
Segmen program 4.1 *import tensorflow*

4.4.2 *Memanggil data video dan class*

Proses selanjutnya adalah memanggil data video atau *class* yang telah di upload ke Google Drive. Fungsi dari pemanggilan data ini ialah agar dapat mengetahui data apa saja yang akan kita olah atau kita proses pada metode *Gated Recurrent Unit* (GRU). Pada Segmen program 4.2. menunjukkan *code input* data video atau *class.* Pada gambar 4.4. menunjukkan data *training* berupa video BISINDO yang akan diinputkan dalam metode GRU.

```
classes = [i.split(os.path.sep)[8] for i in glob.glob('/co
    ntent/drive/MyDrive/SKRIPSI 2021/SKRIPSI PROGRAM/DATASET/
    training/*')]
classes.sort()
print(classes)
# pattern to get videos and classes
glob_pattern_train='/content/drive/MyDrive/SKRIPSI 2021/SK
    RIPSI PROGRAM/DATASET/training/{classname}/*.avi'
glob_pattern_test='/content/drive/MyDrive/SKRIPSI 2021/SKR
    IPSI PROGRAM/DATASET/Test_36_data/{classname}/*.avi'
```

Segmen program 4.2. Input data video dan *class*



Gambar 4.4. **Video atau *class* yang diinput**

4.4.3 *Sliding frame*

Sliding frame ini berfungsi untuk menslide atau menggeser setiap gerakan yang dilakukan oleh tangan. Dengan adanya proses *sliding* ini video akan benar benar diseleksi dengan baik untuk menghasilkan akurasi atau hasil yang bagus. Pada Segmen program 4.3. Menunjukkan c*ode sliding frame.*

```
# Create video frame generator
    gen  =  keras_video . SlidingFrameGenerator ( sequence_time = .5 ,  batch_size
     = 8 ,nb_frames = 5 ,  glob_pattern ='/content/drive/MyDrive/SKRIPSI 2021/S
    KRIPSI PROGRAM/DATASET/training/{classname}/*', split_val = .2)
#tampil gambar
    print(len(gen))
# 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}/*',)
    print(len(test))
```

Segmen program 4.3. *Sliding frame generator*

4.4.4 *Inisialisasi hyperparameter model GRU*

Hyperparameter merupakan salah satu faktor yang berpengaruh dalam mendapatkan performa yang lebih tinggi dalam melatih model *Gated Recurrent Unit*. Untuk model penelitian ini *hyperparameter* yang di inisialisasikan ialah *batch_size*, *nb_frames*, *split_val*, *momentum*, dan *epoch*. Pada Tabel 4.1. menunjukkan hasil inisialisasi yang telah dilakukan pada penelitian ini yaitu dengan nilai *batch_size* 8, *nb_frames* 5, *split_val 0.2, momentum* 0.2 , dan memakai *epoch* 10.

Tabel 4.1. Nilai dari hyperparameter yang dipilih secara manual

Inisialisasi hyperparameter	**Nilai**
Batch Size	8
Nb_Frames	5
Split_val	0.2
Momentum	0.2
Epoch	10

4.4.5 *Menampilkan video BISINDO*

Setelah data diproses oleh sistem dengan dilakukan *sliding frame generator* maka data video akan ditampilkan sesuai dengan jumlah *frame* yang diinginkan. Pada Gambar 4.5. menunjukkan hasil data *training* yang telah dilakukan proses *sliding* dengan 5 *frame* setiap video.

Gambar 4.5. Hasil *sliding* video terimakasih

4.4.6 *Augumentasi data dan simpan model*

Proses *augmentasi* data ini biasa disebut dengan *pre-processing* dan pembangkit data. Model *fit generator* ini berfungsi untuk membangkitkan data untuk setiap *epoch* sampai semua *epoch* dapat memenuhi jumlah sampelnya. *Validation data* ini berfungsi untuk memanggil data *training* yang sudah di *sliding frame generator*. Setelah itu model akan disimpan pada folder baru yang telah

dibuat dan model ini akan digunakan untuk mengklasifikasi data baru/data *testing*. Segmen program 4.4. Menunjukkan cara *augmentasi* data serta simpan model *training* dalam bentuk .h5.

(a)

```
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
)
```

(b)

```
%cd/content/drive/MyDrive/SKRIPSI 2021/SKRIPSI PROGRAM/DATASET/
```

Segmen program 4.4. (a) *Augmentasi* data video BISINDO, (b) Save model training

4.4.7 *Training Data BISINDO*

Pada tahapan ini data yang telah diproses pada tahapan diatas maka akan dilanjutkan pada tahapan *training* data dengan menggunakan metode *Deep Gated Recurrent Unit*. Dari hasil tahapan diatas maka peneliti telah memiliki bekal untuk

mengimplementasikan metode GRU pada tahap ini. Setelah melakukan proses *sliding* maka data akan di *running*, proses *running* ini dilakukan selama 24 jam atau 1 hari. Proses *training* ini menggunakan *epoch* 10 yang modelnya nanti akan disimpan dalam bentuk ".h5" yang akan menjadi bekal untuk tahapan *testing*. Segmen program 4.5. berfungsi untuk menampilkan grafik yang telah diperoleh dari *epoch* 10. Pada Gambar 4.6. menunjukkan grafik dari *epoch* 10 yang terdiri dari nilai *loss*, *accurasi*, *val_loss*, dan *val_accuracy*.

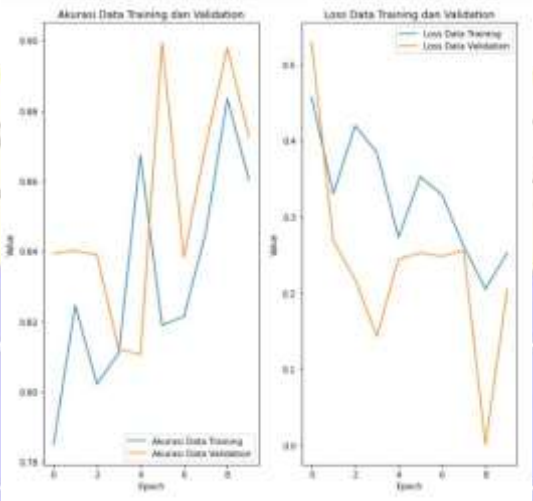
Segmen program4.5. Menampilkan grafik

```
#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 plot berdasarkan
data
plt.plot(val_accuracy, label='Akurasi Data Validation')
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 berdasarkan data
```

```
plt.plot(val_loss, label='Loss Data Validation')#membuat plot berdasarkan d
ata
plt.legend(loc="upper right")
plt.show()
```

Gambar 4.6. Grafik hasil nilai *epoch* 1



4.4.8 *Nilai Accuracy*

Nilai *accuracy* ini diperoleh dari hasil *epoch* yang dijalankan yaitu *epoch* 1 sampai *epoch* 10 untuk mencari nilai *accuracy* tertinggi. Hasil dari epoch 10 ini mendapat nilai *loss* sebesar 0,2034, nilai *accuracy* sebesar 0,9115, dan dengan waktu 794 *second*. Hasil grafik pada tabel 4.1. menunjukkan bahwa nilai *accuracy* yang diperoleh lebih besar daripada nilai *loss* yang diperoleh. Untuk lebih jelasnya ada pada tabel 4.1.

Tabel 4.2. Grafik nilai *accurasi* dan loss *epoch* 10

Nilai Epoch	Nilai Accuracy	Nilai Loss
1	0,8269	0,3939
2	0,8157	0,4561
3	0,8373	0,3181
4	0,8566	0,2815

5	0,8954	0,2355
6	0,8898	0,2237
7	0,8682	0,3123
8	0,8770	0,2926
9	0,9115	0,2034
10	0,9099	0,2340

4.4.9 Uji Coba

Tahap uji coba ini dilakukan untuk menguji cobakan data *testing* dengan metode *Gated Recurrent Unit* yang telah diimplementasikan. Pada tahapan uji coba ini kita akan mengetahui tingkat akurasi pada tahapan implementasi dengan menggunakan metode GRU. Model yang akan dipakai adalah model dari data *training* yang telah dilatih sebelumnya. Segmen program 4.6. menunjukkan *script* untuk memanggil data dan model yang akan digunakan pada proses uji coba atau *testing.* Dan pada Segmen program 4.7. menunjukkan *script* untuk menampilkan nilai akurasi yang diperoleh dari hasil uji coba atau *testing*.

```
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_data/*')]
```

```
classes.sort()
print(classes)
glob_pattern_test='/content/drive/MyDrive/SKRIPSI 2021/SKRIPSIPROG
RAM/DATASET/Test_36_data/{classname}/*.avi'
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)
```

Segmen program 4.6. Deskripsi sumber dan target data uji coba

```
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)
```

Segmen program 4.7. *Script* nilai *accurasi*

Menentukan akurasi ini ialah dengan menghitung nilai *frame* yang telah terdeteksi. Video *testing* akan dicocokkan dengan video dari data training. Nilai *accurasi* yang diperoleh ialah berbentuk *persentase* pada akhir dengan menghitung nilai data *testing*. Pada Gambar 4.7. ditunjukkan bahwa model yang disimpan dalam bentuk .h5 menghasilkan nilai parameter sebesar 5,465, 475 *params.*

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_3 (TimeDist (None, 5, 512)            4689216
_________________________________________________________________
gru_3 (GRU)                  (None, 64)                110784
_________________________________________________________________
dense_11 (Dense)             (None, 1024)              66560
_________________________________________________________________
dropout_7 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_12 (Dense)             (None, 512)               524800
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_13 (Dense)             (None, 128)               65664
_________________________________________________________________
dropout_9 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_14 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_15 (Dense)             (None, 3)                 195
=================================================================
Total params: 5,465,475
Trainable params: 5,463,555
Non-trainable params: 1,920
```

Gambar 4.7. Hasil *load* model yang disimpan dalam .h5

a) Uji coba dengan 24 data *testing*

Uji coba yang akan dilakukan disini ialah dengan menggunakan 24 data *testing* video BISINDO yang terdiri dari 3 *class* dimana dalam setiap *class* terdapat 8 video. Hasil uji coba ditunjukkan pada tabel 4.2.

Tabel 4.3. Hasil uji coba dengan 24 data

Banyak data uji coba	Banyak data benar	Banyak data salah	Nilai Akurasi
24	16	8	67%

Hasil dari tahapan uji coba diatas dengan menggunakan 24 data sebagai uji coba, menghasilkan data benar (akurat) atau data yang dapat terdeteksi sebanyak 16 data dan data video yang tidak dapat terdeteksi sebanyak 8 video dengan menggunakan model *epoch* 10. Hasil akurasi yang diperoleh dari uji coba tersebut ialah sebanyak 67 %. Dari data uji coba tersebut pola gerakan tangan dapat dikenali sesuai dengan data *training* yang telah dilatih sebelumnya. Gambar 4.8 menunjukkan hasil uji coba dengan 24 data video (a) hasil dari data *testing* dan gambar (b) hasil dari data *training*.

(a)

(b)

Gambar 4.8. Hasil uji coba (a) Video terimakasih, (b) Video sama sama

b) Uji coba dengan 36 data *testing*

Uji coba yang dilakukan dengan menggunakan 36 video BISINDO yang terdiri dari tiga *class* yang mana setiap *class* terdapat 12 data video. Hasil uji coba ini menghasilkan akurasi yang baik, lebih jelasnya dapat dilihat pada Tabel 4.3.

Tabel 4.4. Hasil uji coba 36 data video

Banyak data uji coba	Banyak data benar	Banyak data salah	Nilai akurasi
36	34	4	88%

Hasil dari tahap uji coba pada Tabel 4.3. dengan data *testing* sebanyak 36 video dan menggunakan *epoch* 10 yang telah dilakukan dalam tahap *implementasi* menghasilkan nilai akurasi sebesar 88%. Dengan jumlah data benar atau data yang dapat terdeteksi sebanyak 32 data video dan 4 data video tidak dapat terdeteksi oleh sistem. Pada Gambar 4.7. Menunjukkan hasil video atau data *training* yang telah dilatih sebelumnya. Dan pada Gambar 4.8. Menunjukkan hasil uji coba dengan menggunakan 36 data video, video yang dapat terdeteksi yaitu ucapan hallo, terimakasih, dan sama-sama.

Gambar 4.9. Hasil data *training*

(a)

Gambar 4.10. Hasil data *testing* 36 video

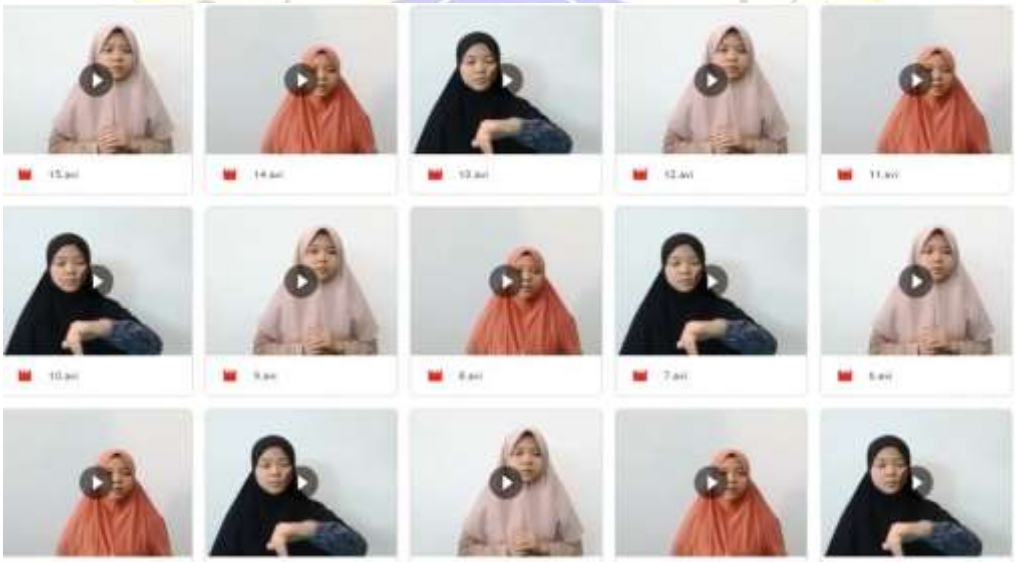
4.5 Analisi Data

Pada tahapan ini akan dibahas hasil deteksi dari pengenalan bahasa isyarat indonesia BISINDO menggunakan metode *deep gated recurrent unit* (GRU). Analisis ini dilakukan dengan menggunakan 24 video BISINDO untuk data *testing* yang terdiri dari 3 *class* yang mana menghasilkan 16 video yang terdeteksi/benar sesuai dengan data *training* dan 8 video tidak dapat terdeteksi atau tidak sesuai dengan data *training* dengan jumlah akurasi 67%. Video yang tidak dapat terdeteksi dengan baik ialah video ucapan terimakasih karena pada video tersebut terdapat dua gerakan tangan yang berbeda sehingga sistem belum bisa membacanya dengan baik. Selanjutnya peneliti mencoba untuk melakukan uji coba kembali dengan menggunakan 36 data video *testing* yang mana dalam uji coba ini mengalami kenaikan tingkat akurasi.

Dengan menggunakan 36 video BISINDO ini data yang sesuai dengan data *training* atau bisa terdeteksi oleh sistem ialah 32 video dengan tingkat akurasi 94% serta video yang tidak sesuai dengan data *training* ialah 4 video. Hasil akurasi tersebut masih terbilang kecil, hal ini disebabkan karena minimnya data atau

terjadinya *underfitting* data serta kapasitas *memory* yang tidak memadai sehingga hasil akurasi yang diperoleh kurang baik. Namun dengan adanya peningkatan akurasi ini maka sistem yang menggunakan metode *deep gated recurrent unit* sudah mulai berlatih dengan gerakan yang diberikan oleh manusia. Pada Gambar 4.9 (a) menunjukan video atau data mentah sebelum terdeteksi oleh sistem dan belum diolah oleh metode GRU. Dan pada gambar 4.9 (b) menunjukkan hasil deteksi video yang dapat terbaca oleh sistem setelah menggunakan metode *Deep Gated Recurrent Unit* (GRU).

(a)



(b)

Gambar 4.11.(a) video BISINDO sebelum terdeteksi oleh sistem, (b) video BISINDO setelah terdeteksi oleh sistem

Pada Gambar 4.9 menunjukan video atau data mentah sebelum terdeteksi oleh sistem dan belum diolah oleh metode GRU, setelah diterapkannya metode GRU maka sistem dapat membaca gerakan tangan yang ada pada video tersebut sesuai dengan *class* yang telah dibagi bagi.