Hand gesture recognition

By Mohamed Jamyl

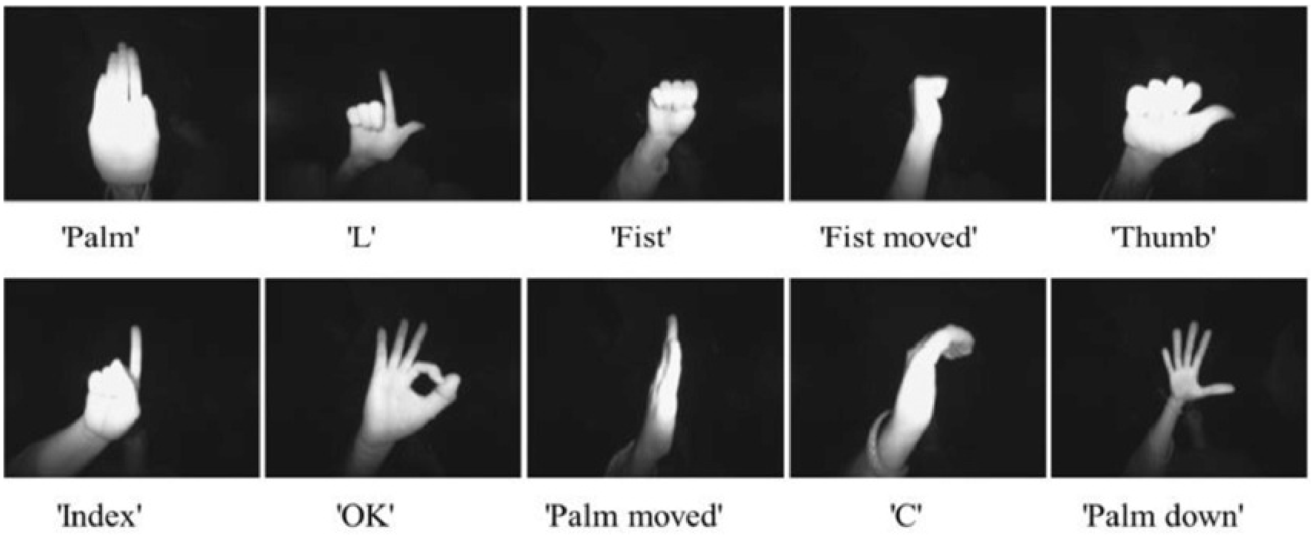
http://linkedin.com/in/mohamed-jamyl

https://www.kaggle.com/mohamedjamyl

https://github.com/Mohamed-Jamyl

In [1]:
```
from IPython.display import Image
Image(filename='img.png')
```

Out[1]:



Context

Hand gesture recognition database is presented, composed by a set of near infrared images acquired by the Leap Motion sensor.

Content

The database is composed by 10 different hand-gestures (showed above) that were performed by 10 different subjects (5 men and 5 women).

Import Libraries

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import os
from cv2 import imread, resize, IMREAD_GRAYSCALE
from pathlib import Path
import shutil
from sklearn.model_selection import train_test_split

from tensorflow import nn
from keras import models, layers,regularizers,optimizers, callbacks
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

from tqdm import tqdm
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
train_data = '/kaggle/input/leapgestrecog/leapGestRecog'
```

```
for fold in os.listdir(train_data):
    print(os.path.join(train_data, fold))
```

```
/kaggle/input/leapgestrecog/leapGestRecog/07
/kaggle/input/leapgestrecog/leapGestRecog/05
/kaggle/input/leapgestrecog/leapGestRecog/06
/kaggle/input/leapgestrecog/leapGestRecog/02
/kaggle/input/leapgestrecog/leapGestRecog/04
/kaggle/input/leapgestrecog/leapGestRecog/00
/kaggle/input/leapgestrecog/leapGestRecog/08
/kaggle/input/leapgestrecog/leapGestRecog/09
/kaggle/input/leapgestrecog/leapGestRecog/03
/kaggle/input/leapgestrecog/leapGestRecog/01
```

In [3]:
```
for fold in os.listdir(train_data):
    subfold = os.path.join(train_data, fold)
    for smallfold in os.listdir(subfold):
        subsmallfold = os.path.join(subfold, smallfold)
        print(subsmallfold)
    print('------------------------------------------------------------')
```

```
/kaggle/input/leapgestrecog/leapGestRecog/07/02_l
/kaggle/input/leapgestrecog/leapGestRecog/07/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/07/09_c
/kaggle/input/leapgestrecog/leapGestRecog/07/10_down
/kaggle/input/leapgestrecog/leapGestRecog/07/06_index
/kaggle/input/leapgestrecog/leapGestRecog/07/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/07/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/07/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/07/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/07/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/05/02_l
/kaggle/input/leapgestrecog/leapGestRecog/05/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/05/09_c
/kaggle/input/leapgestrecog/leapGestRecog/05/10_down
/kaggle/input/leapgestrecog/leapGestRecog/05/06_index
/kaggle/input/leapgestrecog/leapGestRecog/05/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/05/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/05/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/05/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/05/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/06/02_l
/kaggle/input/leapgestrecog/leapGestRecog/06/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/06/09_c
/kaggle/input/leapgestrecog/leapGestRecog/06/10_down
/kaggle/input/leapgestrecog/leapGestRecog/06/06_index
/kaggle/input/leapgestrecog/leapGestRecog/06/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/06/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/06/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/06/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/06/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/02/02_l
/kaggle/input/leapgestrecog/leapGestRecog/02/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/02/09_c
/kaggle/input/leapgestrecog/leapGestRecog/02/10_down
/kaggle/input/leapgestrecog/leapGestRecog/02/06_index
/kaggle/input/leapgestrecog/leapGestRecog/02/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/02/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/02/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/02/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/02/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/04/02_l
/kaggle/input/leapgestrecog/leapGestRecog/04/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/04/09_c
/kaggle/input/leapgestrecog/leapGestRecog/04/10_down
/kaggle/input/leapgestrecog/leapGestRecog/04/06_index
/kaggle/input/leapgestrecog/leapGestRecog/04/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/04/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/04/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/04/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/04/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/00/02_l
/kaggle/input/leapgestrecog/leapGestRecog/00/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/00/09_c
/kaggle/input/leapgestrecog/leapGestRecog/00/10_down
/kaggle/input/leapgestrecog/leapGestRecog/00/06_index
/kaggle/input/leapgestrecog/leapGestRecog/00/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/00/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/00/05_thumb
```

```
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/00/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/08/02_l
/kaggle/input/leapgestrecog/leapGestRecog/08/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/08/09_c
/kaggle/input/leapgestrecog/leapGestRecog/08/10_down
/kaggle/input/leapgestrecog/leapGestRecog/08/06_index
/kaggle/input/leapgestrecog/leapGestRecog/08/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/08/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/08/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/08/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/08/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/09/02_l
/kaggle/input/leapgestrecog/leapGestRecog/09/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/09/09_c
/kaggle/input/leapgestrecog/leapGestRecog/09/10_down
/kaggle/input/leapgestrecog/leapGestRecog/09/06_index
/kaggle/input/leapgestrecog/leapGestRecog/09/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/09/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/09/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/09/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/09/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/03/02_l
/kaggle/input/leapgestrecog/leapGestRecog/03/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/03/09_c
/kaggle/input/leapgestrecog/leapGestRecog/03/10_down
/kaggle/input/leapgestrecog/leapGestRecog/03/06_index
/kaggle/input/leapgestrecog/leapGestRecog/03/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/03/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/03/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/03/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/03/03_fist
------------------------------------------------------------
/kaggle/input/leapgestrecog/leapGestRecog/01/02_l
/kaggle/input/leapgestrecog/leapGestRecog/01/04_fist_moved
/kaggle/input/leapgestrecog/leapGestRecog/01/09_c
/kaggle/input/leapgestrecog/leapGestRecog/01/10_down
/kaggle/input/leapgestrecog/leapGestRecog/01/06_index
/kaggle/input/leapgestrecog/leapGestRecog/01/08_palm_moved
/kaggle/input/leapgestrecog/leapGestRecog/01/07_ok
/kaggle/input/leapgestrecog/leapGestRecog/01/05_thumb
/kaggle/input/leapgestrecog/leapGestRecog/01/01_palm
/kaggle/input/leapgestrecog/leapGestRecog/01/03_fist
------------------------------------------------------------
```

In [4]:
```python
os.makedirs('all_images', exist_ok=True)
```

In [5]:
```python
classes = ['01_palm','02_l','03_fist','04_fist_moved','05_thumb','06_index','07_ok','08_palm_moved','09_c','10_
root = Path('/kaggle/working/all_images')

for cls in classes:
    (root / cls).mkdir(parents=True, exist_ok=True)
```

In [ ]:
```python
src_roots = [
    Path('/kaggle/input/leapgestrecog/leapGestRecog/00'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/01'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/02'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/03'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/04'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/05'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/06'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/07'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/08'),
    Path('/kaggle/input/leapgestrecog/leapGestRecog/09')
]

dst_root = Path('/kaggle/working/all_images')

subfolders = ['01_palm','02_l','03_fist','04_fist_moved','05_thumb','06_index','07_ok','08_palm_moved','09_c','

img_suffixes = {'.png'}

for sub in subfolders:
    dst_dir = dst_root / sub
    dst_dir.mkdir(parents=True, exist_ok=True)

    for src_root in src_roots:
        src_dir = src_root / sub
        if not src_dir.exists():
            print(f'Not exist: {src_dir}')
            continue

        for file in src_dir.iterdir():
            if img_suffixes and file.suffix.lower() not in img_suffixes:
                continue

            dst_path = dst_dir / file.name

            if dst_path.exists():
                prefix = f"{src_root.name}_"
                dst_path = dst_dir / f"{prefix}{file.name}"

            shutil.copy2(file, dst_path)
            print(f'{file}  to  {dst_path}')
```

```
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0025.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0025.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0045.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0045.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0070.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0070.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0125.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0125.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0086.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0086.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0140.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0140.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0004.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0004.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0156.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0156.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0067.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0067.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0076.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0076.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0087.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0087.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0155.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0155.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0107.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0107.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0062.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0062.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0104.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0104.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0115.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0115.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0024.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0024.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0182.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0182.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0029.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0029.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0167.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0167.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0170.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0170.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0153.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0153.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0112.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0112.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0194.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0194.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0109.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0109.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0073.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0073.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0055.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0055.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0033.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0033.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0199.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0199.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0038.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0038.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0129.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0129.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0053.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0053.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0022.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0022.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0181.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0181.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0130.png  to  /kaggle/working/all_images/01_pal
```

```
m/frame_00_01_0130.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0059.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0059.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0192.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0192.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0050.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0050.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0010.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0010.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0009.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0009.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0013.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0013.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0044.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0044.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0066.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0066.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0193.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0193.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0071.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0071.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0177.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0177.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0197.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0197.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0171.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0171.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0012.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0012.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0123.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0123.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0175.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0175.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0003.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0003.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0136.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0136.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0017.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0017.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0196.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0196.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0166.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0166.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0116.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0116.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0099.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0099.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0040.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0040.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0028.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0028.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0114.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0114.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0023.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0023.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0092.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0092.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0021.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0021.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0031.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0031.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0134.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0134.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0168.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0168.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0149.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0149.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0187.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0187.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0047.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0047.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0179.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0179.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0015.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0015.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0174.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0174.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0132.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0132.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0105.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0105.png
/kaggle/input/leapgestrecog/leapGestRecog/00/01_palm/frame_00_01_0145.png  to  /kaggle/working/all_images/01_pal
m/frame_00_01_0145.png
```

```
ame_09_07_0141.png
```

In [ ]:
```
train_data = '/kaggle/working/all_images'
```

In [73]:
```
categories = ['04_fist_moved','02_l', '10_down', '05_thumb', '08_palm_moved', '06_index', '09_c', '01_palm','07_
folds = [os.path.join(train_data, category) for category in categories]
folds
```

Out[73]:
```
['/kaggle/working/all_images/04_fist_moved',
 '/kaggle/working/all_images/02_l',
 '/kaggle/working/all_images/10_down',
 '/kaggle/working/all_images/05_thumb',
 '/kaggle/working/all_images/08_palm_moved',
 '/kaggle/working/all_images/06_index',
 '/kaggle/working/all_images/09_c',
 '/kaggle/working/all_images/01_palm',
 '/kaggle/working/all_images/07_ok',
 '/kaggle/working/all_images/03_fist']
```

In [74]:
```
for f in folds:
    print(f.split('/')[4], ' : ' ,len(os.listdir(f)))
```

```
04_fist_moved  :  2000
02_l  :  2000
10_down  :  2000
05_thumb  :  2000
08_palm_moved  :  2000
06_index  :  2000
09_c  :  2000
01_palm  :  2000
07_ok  :  2000
03_fist  :  2000
```

In [75]:
```
x = 0
for img in os.listdir(folds[0]):
    x += 1
    img_array = imread(os.path.join(folds[0],img))
    print(img_array)
    if x == 1:
        break
```

```
[[[ 4   4   4]
  [ 5   5   5]
  [ 4   4   4]
  ...
  [ 4   4   4]
  [ 4   4   4]
  [ 6   6   6]]

 [[ 3   3   3]
  [ 4   4   4]
  [ 5   5   5]
  ...
  [ 3   3   3]
  [ 3   3   3]
  [ 3   3   3]]

 [[ 4   4   4]
  [ 4   4   4]
  [ 5   5   5]
  ...
  [ 4   4   4]
  [ 4   4   4]
  [ 3   3   3]]

 ...

 [[ 5   5   5]
  [ 4   4   4]
  [ 5   5   5]
  ...
  [ 6   6   6]
  [ 3   3   3]
  [ 3   3   3]]

 [[ 4   4   4]
  [ 4   4   4]
  [ 4   4   4]
  ...
  [ 3   3   3]
  [ 4   4   4]
  [ 4   4   4]]

 [[ 4   4   4]
  [ 6   6   6]
  [ 5   5   5]
  ...
  [25 25 25]
  [12 12 12]
  [16 16 16]]]
```

In [76]:

```
x = 0
for img in os.listdir(folds[0]):
    x += 1
    img_array = imread(os.path.join(folds[0],img))
    print(img_array.shape)
    if x == 1:
        break
```

```
(240, 640, 3)
```

In [77]:

```
x = 0
for img in os.listdir(folds[0]):
    x += 1
    img_array = imread(os.path.join(folds[0],img))
    print(img_array[0])
    if x == 1:
        break
```

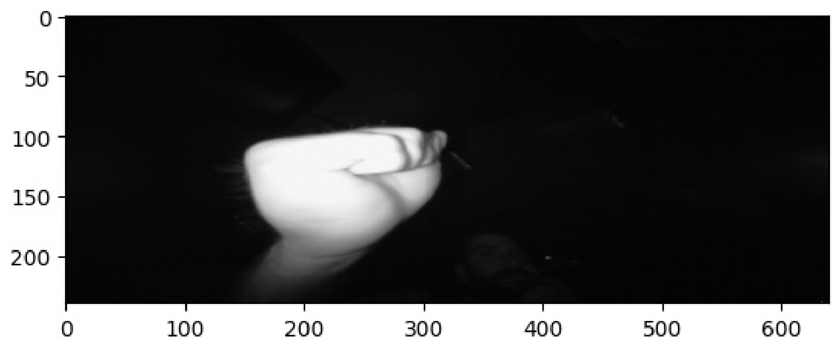
```
[[4 4 4]
 [5 5 5]
 [4 4 4]
 ...
 [4 4 4]
 [4 4 4]
 [6 6 6]]
```

In [78]:

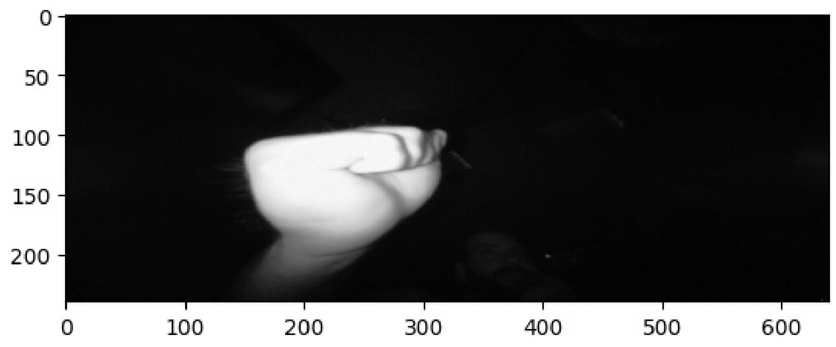
```
x = 0
for img in os.listdir(folds[0]):
    x += 1
    img_array = imread(os.path.join(folds[0],img))
    print(img_array[0][0])
    if x == 1:
        break
```

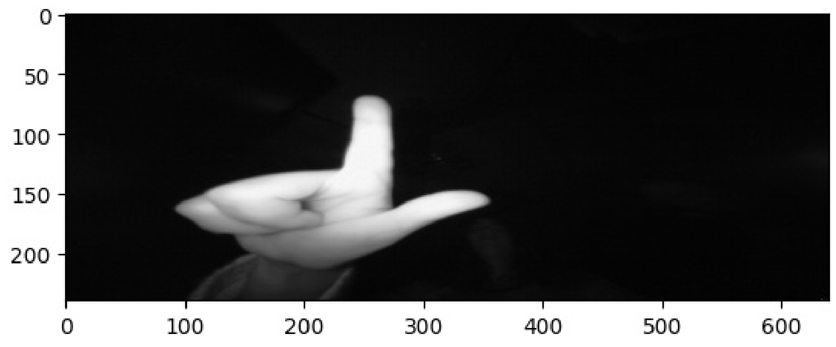
```
[4 4 4]
```

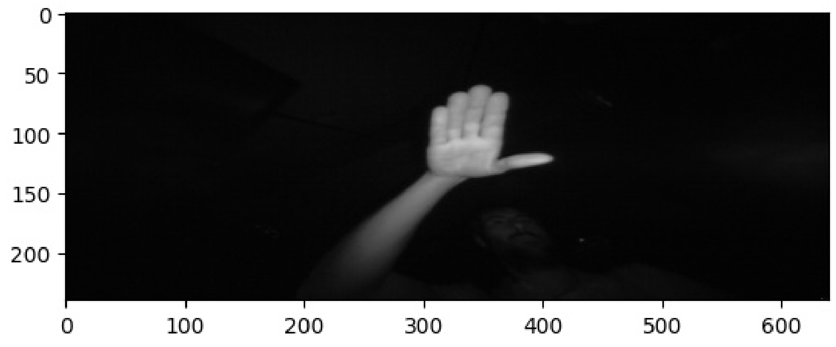
In [ ]:
```
x = 0
for img in os.listdir(folds[0]):
    x += 1
    img_array = imread(os.path.join(folds[0],img))
    plt.imshow(img_array, cmap='gray')
    plt.show()
    if x == 1:
        break
```

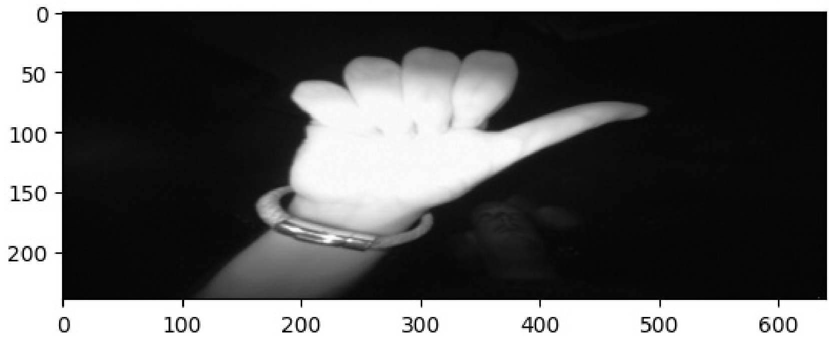


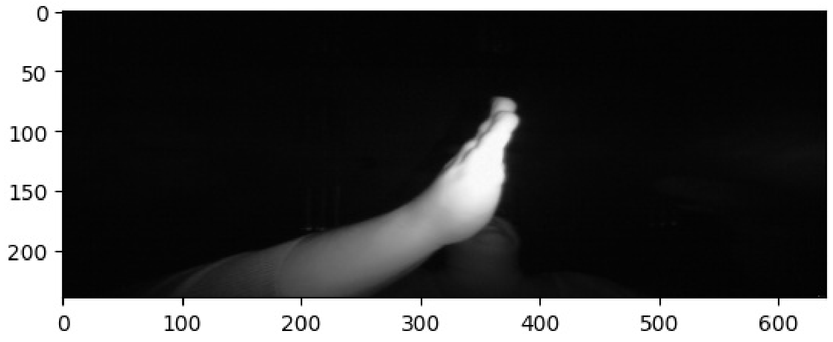
In [ ]:
```
for fold in folds:
    x = 0
    for img in os.listdir(fold):
        x +=1
        img_array = imread(os.path.join(fold,img))
        plt.imshow(img_array, cmap='gray')
        plt.show()
        if x == 1:
            break
```

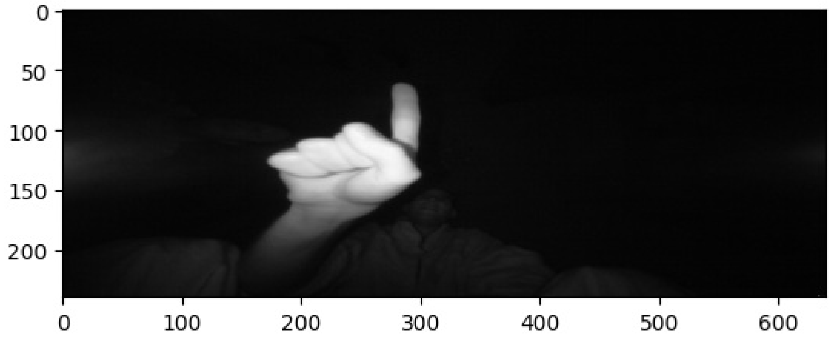


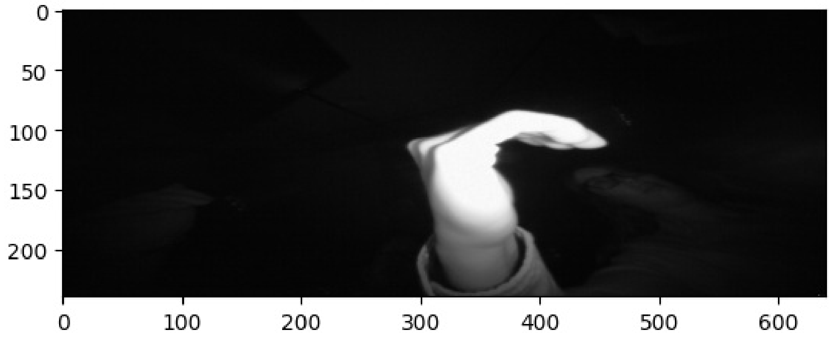


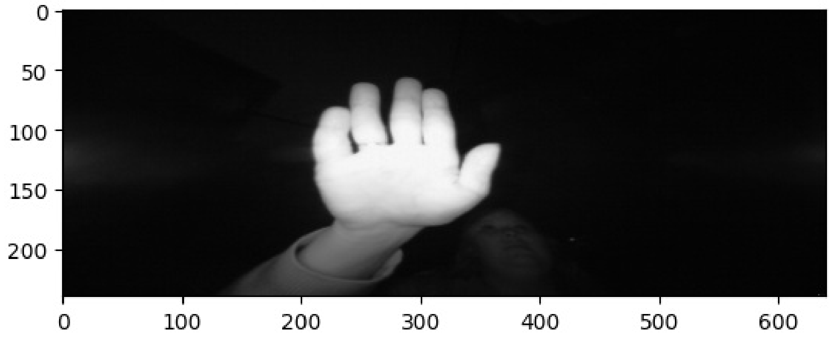


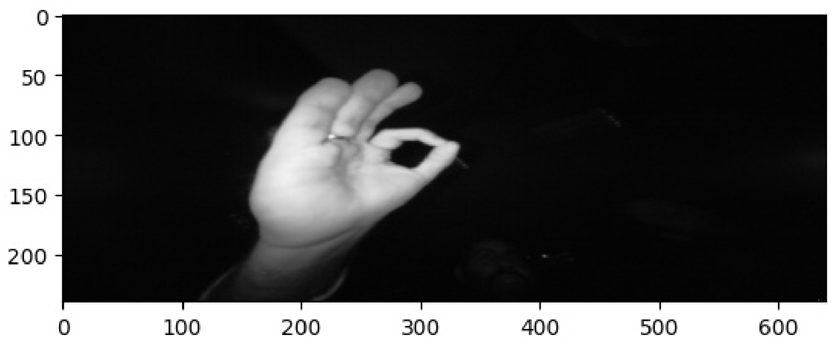

0
50
100
150
200
0
100
200
300
400
500
600

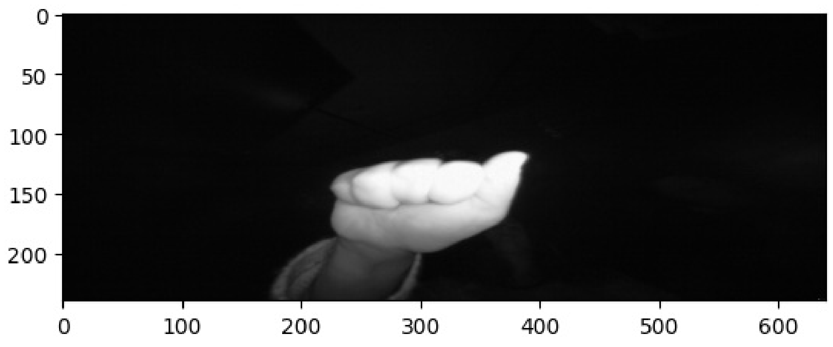

0
50
100
150
200
0
100
200
300
400
500
600


0
50
100
150
200
0
100
200
300
400
500
600


0
50
100
150
200
0
100
200
300
400
500
600


0
50
100
150
200
0
100
200
300
400
500
600

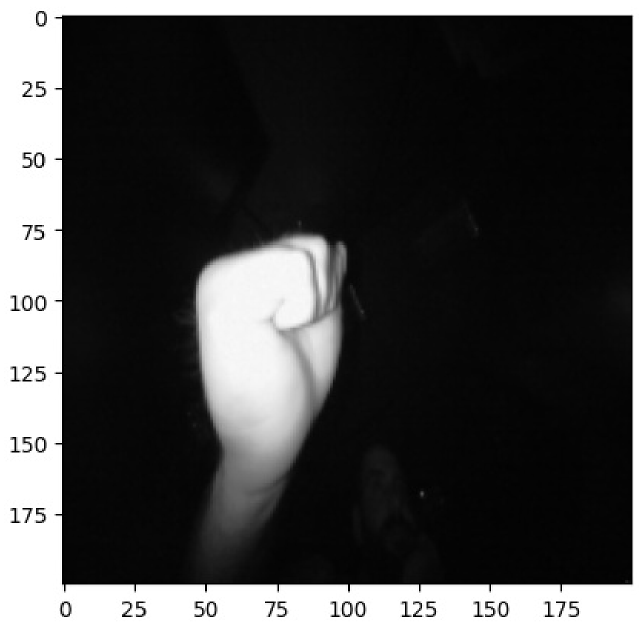




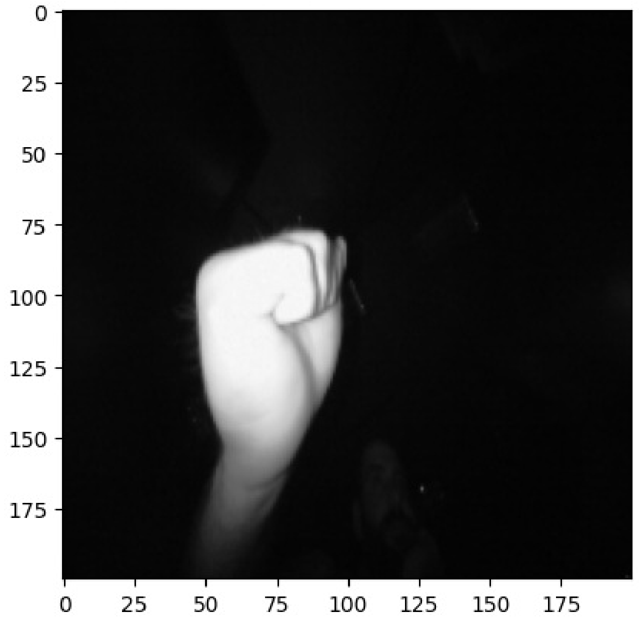
In [81]:
```python
# checking sizes of images
AllSizes = []

for fold in folds:
    for img in os.listdir(fold):
        img_array = imread(os.path.join(fold,img))
        AllSizes.append(img_array.shape)
set(AllSizes)
```

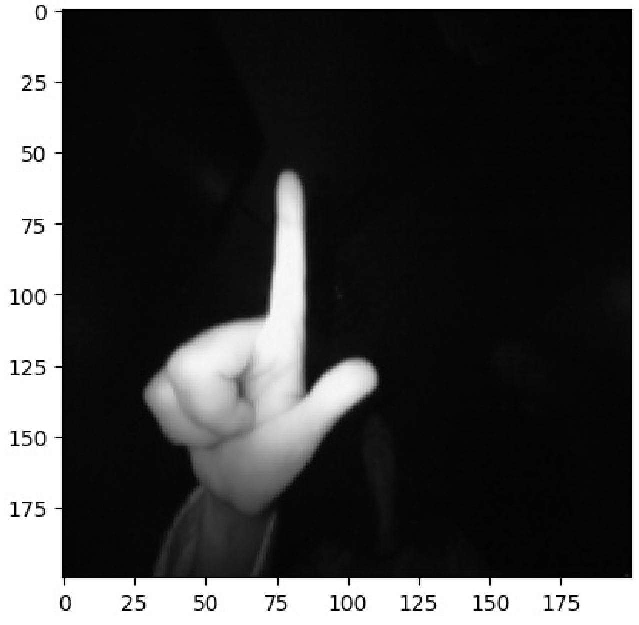
Out[81]:
```
{(240, 640, 3)}
```

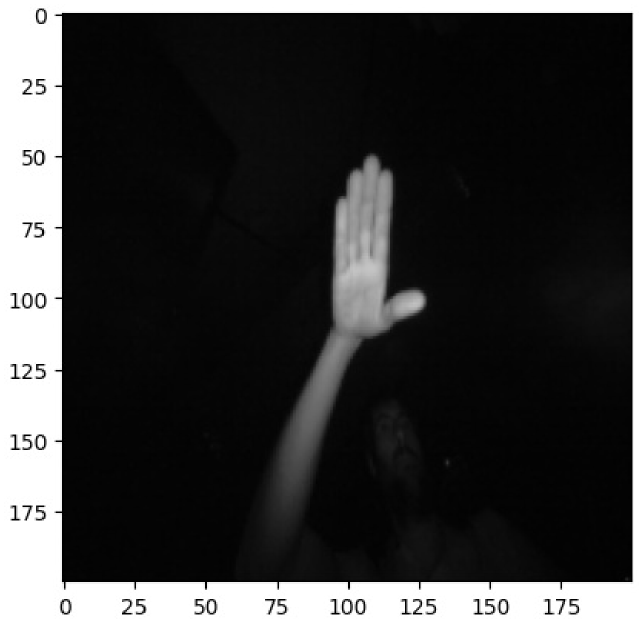
In [ ]:
```python
width, height = 200, 200

x = 0
for img in os.listdir(folds[0]):
    x += 1
    img_array = imread(os.path.join(folds[0],img))
    img_array_resize = resize(img_array,(width,height))
    plt.imshow(img_array_resize)
    plt.show()
    if x == 1:
        break
```

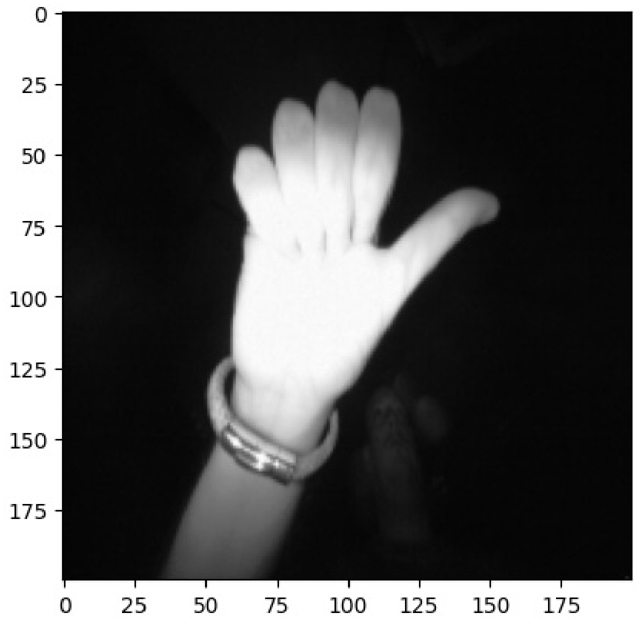


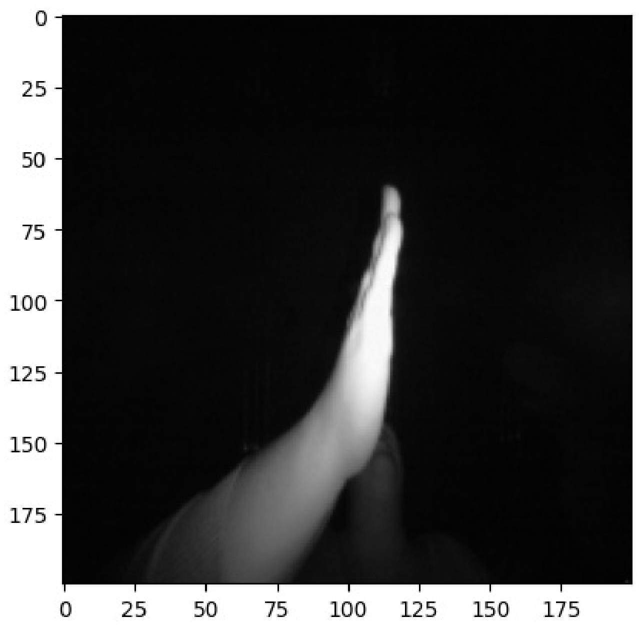
In [ ]:
```
for fold in folds:
    x = 0
    for img in os.listdir(fold):
        x += 1
        img_array = imread(os.path.join(fold,img))
        img_array_resize = resize(img_array, (width,height))
        plt.imshow(img_array_resize)
        plt.show()
        if x==1:
            break
```

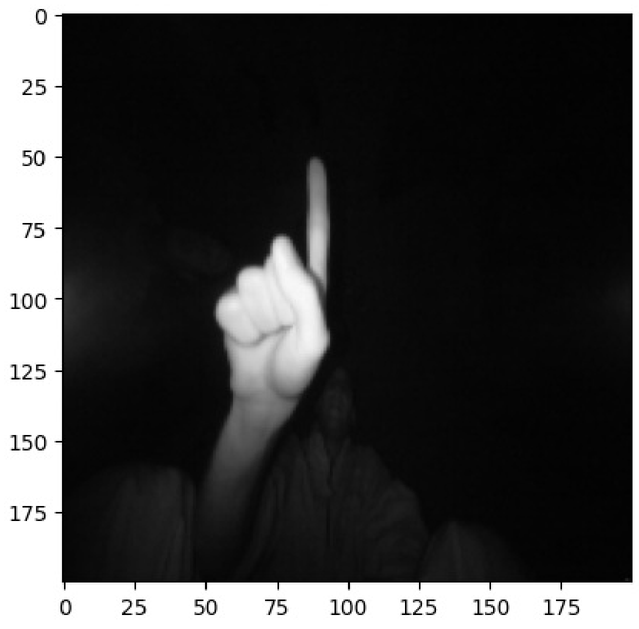


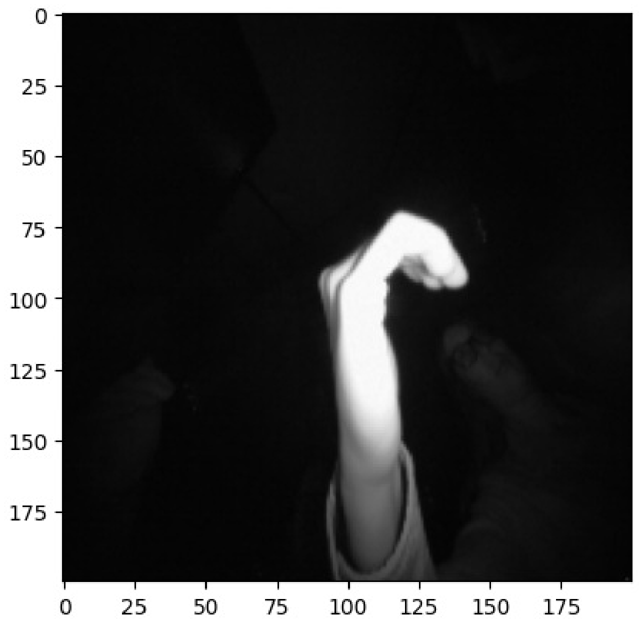


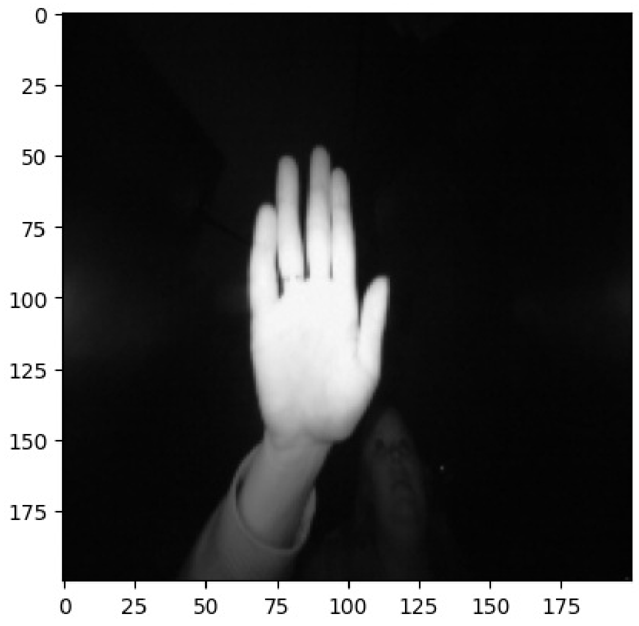

0
25
50
75
100
125
150
175

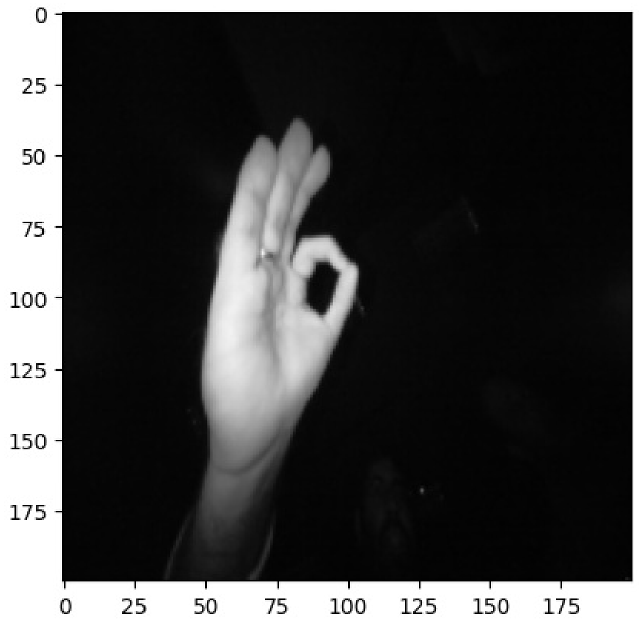
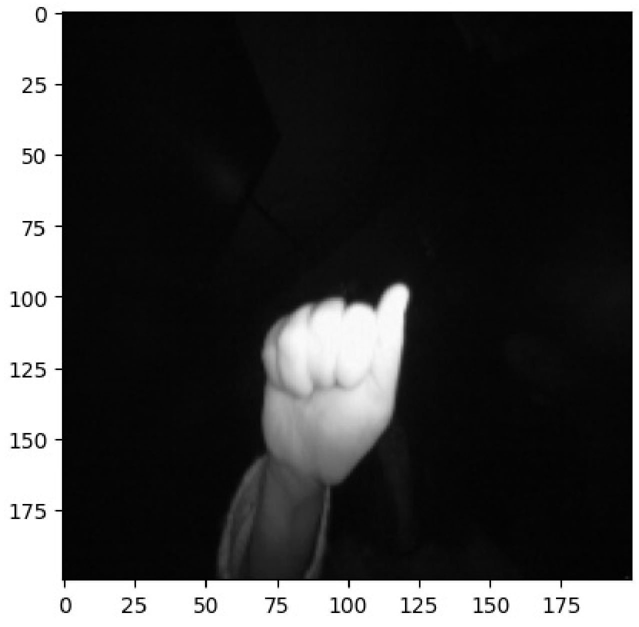

0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

In [84]:
```
training_data=[]

def create_training_data():
    for fold in folds:
        class_num = folds.index(fold)
        for img in tqdm(os.listdir(fold)):
            img_array = imread(os.path.join(fold,img))
            img_array_resize = resize(img_array,(width,height))
            training_data.append([img_array_resize,class_num])

create_training_data()
```

```
100%|██████████| 2000/2000 [00:05<00:00, 380.53it/s]
100%|██████████| 2000/2000 [00:05<00:00, 382.73it/s]
100%|██████████| 2000/2000 [00:04<00:00, 423.66it/s]
100%|██████████| 2000/2000 [00:05<00:00, 367.71it/s]
100%|██████████| 2000/2000 [00:04<00:00, 433.78it/s]
100%|██████████| 2000/2000 [00:05<00:00, 371.53it/s]
100%|██████████| 2000/2000 [00:05<00:00, 368.03it/s]
100%|██████████| 2000/2000 [00:04<00:00, 406.82it/s]
100%|██████████| 2000/2000 [00:05<00:00, 387.23it/s]
100%|██████████| 2000/2000 [00:05<00:00, 360.81it/s]
```

```
In [85]: print(training_data[:1])
```

```
[[array([[[ 5,  5,  5],
         [ 4,  4,  4],
         [ 5,  5,  5],
         ...,
         [ 3,  3,  3],
         [ 4,  4,  4],
         [ 4,  4,  4]],

        [[ 4,  4,  4],
         [ 4,  4,  4],
         [ 5,  5,  5],
         ...,
         [ 4,  4,  4],
         [ 4,  4,  4],
         [ 3,  3,  3]],

        [[ 4,  4,  4],
         [ 4,  4,  4],
         [ 5,  5,  5],
         ...,
         [ 4,  4,  4],
         [ 4,  4,  4],
         [ 4,  4,  4]],

        ...,

        [[ 4,  4,  4],
         [ 5,  5,  5],
         [ 5,  5,  5],
         ...,
         [ 4,  4,  4],
         [ 4,  4,  4],
         [ 4,  4,  4]],

        [[ 4,  4,  4],
         [ 5,  5,  5],
         [ 5,  5,  5],
         ...,
         [ 4,  4,  4],
         [ 4,  4,  4],
         [ 3,  3,  3]],

        [[ 6,  6,  6],
         [ 4,  4,  4],
         [ 5,  5,  5],
         ...,
         [ 3,  3,  3],
         [35, 35, 35],
         [12, 12, 12]]], dtype=uint8), 0]]
```

```
In [86]: training_data[0][0][0]
```

```
Out[86]: array([[ 5,  5,  5],
                [ 4,  4,  4],
                [ 5,  5,  5],
                [ 4,  4,  4],
                [ 4,  4,  4],
                [ 4,  4,  4],
                [ 5,  5,  5],
                [ 5,  5,  5],
                [ 5,  5,  5],
                [ 5,  5,  5],
                [ 5,  5,  5],
                [ 6,  6,  6],
                [ 5,  5,  5],
                [ 7,  7,  7],
                [ 6,  6,  6],
                [ 5,  5,  5],
                [ 6,  6,  6],
                [ 6,  6,  6],
                [ 5,  5,  5],
                [ 4,  4,  4],
                [ 6,  6,  6],
                [ 5,  5,  5],
                [ 7,  7,  7],
                [ 5,  5,  5],
                [ 7,  7,  7],
                [ 9,  9,  9],
                [ 5,  5,  5],
                [ 6,  6,  6],
                [ 6,  6,  6],
       [ 8,  8,  8],
       [ 7,  7,  7],
       [ 6,  6,  6],
       [ 6,  6,  6],
       [ 6,  6,  6],
       [ 6,  6,  6],
       [ 6,  6,  6],
       [ 5,  5,  5],
       [ 5,  5,  5],
       [ 6,  6,  6],
       [ 6,  6,  6],
       [ 6,  6,  6],
       [ 6,  6,  6],
       [ 6,  6,  6],
       [ 5,  5,  5],
       [ 3,  3,  3],
       [ 3,  3,  3],
       [ 4,  4,  4],
       [ 5,  5,  5],
       [ 4,  4,  4],
       [ 6,  6,  6],
       [ 5,  5,  5],
       [ 5,  5,  5],
       [ 7,  7,  7],
       [ 4,  4,  4],
       [ 4,  4,  4],
       [ 8,  8,  8],
       [ 9,  9,  9],
       [ 8,  8,  8],
       [10, 10, 10],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 9,  9,  9],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 9,  9,  9],
       [ 9,  9,  9],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 9,  9,  9],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 7,  7,  7],
       [ 8,  8,  8],
       [ 9,  9,  9],
       [ 8,  8,  8],
       [ 9,  9,  9],
       [ 7,  7,  7],
       [ 8,  8,  8],
       [ 7,  7,  7],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 6,  6,  6],
       [ 7,  7,  7],
       [ 9,  9,  9],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 9,  9,  9],
       [ 7,  7,  7],
       [ 6,  6,  6],
       [ 8,  8,  8],
       [10, 10, 10],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 8,  8,  8],
       [ 7,  7,  7],
       [ 7,  7,  7],
       [ 6,  6,  6],
       [ 7,  7,  7],
       [ 7,  7,  7],
       [ 5,  5,  5],
       [ 7,  7,  7],
       [ 6,  6,  6],
       [ 8,  8,  8],
       [ 6,  6,  6],
       [ 7,  7,  7],
       [ 7,  7,  7],
       [ 6,  6,  6],
[ 6,  6,  6],
[ 6,  6,  6],
[ 5,  5,  5],
[ 7,  7,  7],
[ 6,  6,  6],
[ 7,  7,  7],
[ 6,  6,  6],
[ 4,  4,  4],
[ 6,  6,  6],
[ 5,  5,  5],
[ 6,  6,  6],
[ 6,  6,  6],
[ 6,  6,  6],
[ 5,  5,  5],
[ 3,  3,  3],
[ 4,  4,  4],
[ 4,  4,  4],
[ 4,  4,  4],
[ 6,  6,  6],
[ 6,  6,  6],
[ 7,  7,  7],
[ 7,  7,  7],
[ 6,  6,  6],
[ 6,  6,  6],
[ 6,  6,  6],
[ 7,  7,  7],
[ 6,  6,  6],
[ 8,  8,  8],
[ 6,  6,  6],
[ 6,  6,  6],
[ 5,  5,  5],
[ 6,  6,  6],
[ 6,  6,  6],
[ 6,  6,  6],
[ 6,  6,  6],
[ 7,  7,  7],
[ 5,  5,  5],
[ 6,  6,  6],
[ 4,  4,  4],
[ 5,  5,  5],
[ 4,  4,  4],
[ 4,  4,  4],
[ 4,  4,  4],
[ 4,  4,  4],
[ 4,  4,  4],
[ 4,  4,  4],
[ 3,  3,  3],
[ 4,  4,  4],
[ 4,  4,  4],
[ 5,  5,  5],
[ 3,  3,  3],
[ 4,  4,  4],
[ 5,  5,  5],
[ 4,  4,  4],
[ 3,  3,  3],
[ 4,  4,  4],
[ 3,  3,  3],
[ 4,  4,  4],
[ 5,  5,  5],
[ 3,  3,  3],
[ 3,  3,  3],
[ 3,  3,  3],
[ 4,  4,  4],
[ 4,  4,  4],
[ 3,  3,  3],
[ 3,  3,  3],
[ 3,  3,  3],
[ 4,  4,  4],
[ 3,  3,  3],
[ 4,  4,  4],
[ 5,  5,  5],
[ 4,  4,  4],
[ 5,  5,  5],
[ 3,  3,  3],
[ 3,  3,  3],
[ 3,  3,  3],
[ 4,  4,  4],
[ 4,  4,  4],
[ 5,  5,  5],
[ 3,  3,  3],
[ 3,  3,  3],
[ 4,  4,  4],
[ 3,  3,  3],
        [ 4,  4,  4],
        [ 3,  3,  3],
        [ 3,  3,  3],
        [ 4,  4,  4],
        [ 4,  4,  4]], dtype=uint8)
```

In [87]:
```
import random
random.shuffle(training_data)
for sample in training_data[:10]:
    print(sample[1])
```

```
6
8
0
5
5
0
0
9
5
7
```

In [23]:
```
x= []
y= []

for label, fold in enumerate(folds):
    for img_name in tqdm(os.listdir(fold)):
        img_path = os.path.join(fold,img_name)
        img = imread(img_path, IMREAD_GRAYSCALE)
        img = resize(img,(width,height))
        x.append(img)
        y.append(label)
```

```
100%|██████████| 2000/2000 [00:03<00:00, 545.79it/s]
100%|██████████| 2000/2000 [00:03<00:00, 545.69it/s]
100%|██████████| 2000/2000 [00:03<00:00, 535.02it/s]
```

In [24]:
```
len(x)
```

Out[24]:
```
20000
```

In [25]:
```
len(y)
```

Out[25]:
```
20000
```

In [26]:
```
x[:1]
```

Out[26]:
```
[array([[ 5,  4,  5, ...,  3,  4,  4],
        [ 4,  4,  5, ...,  4,  4,  3],
        [ 4,  4,  5, ...,  4,  4,  4],
        ...,
        [ 4,  5,  5, ...,  4,  4,  4],
        [ 4,  5,  5, ...,  4,  4,  3],
        [ 6,  4,  5, ...,  3, 35, 12]], dtype=uint8)]
```

In [27]:
```
# print(y)
```

In [28]:
```
x = np.array(x).reshape(-1, 200, 200, 1)
y = np.array(y)
```

In [29]:
```
x
```

Out[29]:
```
array([[[[ 5],
         [ 4],
         [ 5],
         ...,
         [ 3],
         [ 4],
         [ 4]],

        [[ 4],
         [ 4],
         [ 5],
         ...,
         [ 4],
         [ 4],
         [ 3]],

        [[ 4],
         [ 4],
         [ 5],
         ...,
         [ 4],
          [ 4],
          [ 4]],

         ...,

         [[ 4],
          [ 5],
          [ 5],
          ...,
          [ 4],
          [ 4],
          [ 4]],

         [[ 4],
          [ 5],
          [ 5],
          ...,
          [ 4],
          [ 4],
          [ 3]],

         [[ 6],
          [ 4],
          [ 5],
          ...,
          [ 3],
          [35],
          [12]]],


        [[[ 5],
          [ 5],
          [ 4],
          ...,
          [ 5],
          [ 4],
          [ 6]],

         [[ 5],
          [ 4],
          [ 4],
          ...,
          [ 5],
          [ 4],
          [ 4]],

         [[ 5],
          [ 4],
          [ 5],
          ...,
          [ 5],
          [ 5],
          [ 5]],

         ...,

         [[ 5],
          [ 5],
          [ 5],
          ...,
          [ 6],
          [ 6],
          [ 6]],

         [[ 5],
          [ 5],
          [ 5],
          ...,
          [ 6],
          [ 6],
          [ 6]],

         [[ 5],
          [ 5],
          [ 5],
          ...,
          [ 5],
          [40],
          [12]]],


        [[[ 3],
          [ 4],
         [ 3],
         ...,
         [ 5],
         [ 5],
         [ 4]],

        [[ 3],
         [ 4],
         [ 3],
         ...,
         [ 4],
         [ 5],
         [ 4]],

        [[ 4],
         [ 4],
         [ 5],
         ...,
         [ 4],
         [ 5],
         [ 4]],

        ...,

        [[ 4],
         [ 3],
         [ 4],
         ...,
         [ 5],
         [ 4],
         [ 5]],

        [[ 4],
         [ 2],
         [ 4],
         ...,
         [ 3],
         [ 4],
         [ 4]],

        [[ 3],
         [ 2],
         [ 3],
         ...,
         [ 4],
         [35],
         [12]]],


       ...,


       [[[ 5],
         [ 5],
         [ 4],
         ...,
         [ 4],
         [ 4],
         [ 4]],

        [[ 4],
         [ 4],
         [ 4],
         ...,
         [ 4],
         [ 4],
         [ 2]],

        [[ 4],
         [ 4],
         [ 5],
         ...,
         [ 5],
         [ 3],
         [ 4]],

        ...,

        [[21],
         [19],
         [20],
         ...,
         [ 5],
          [ 4],
          [ 5]],

         [[19],
          [17],
          [18],
          ...,
          [ 4],
          [ 6],
          [ 5]],

         [[17],
          [17],
          [18],
          ...,
          [ 5],
          [42],
          [12]]],


        [[[ 6],
          [ 5],
          [ 5],
          ...,
          [ 5],
          [ 4],
          [ 5]],

         [[ 5],
          [ 4],
          [ 5],
          ...,
          [ 5],
          [ 4],
          [ 4]],

         [[ 5],
          [ 6],
          [ 5],
          ...,
          [ 4],
          [ 4],
          [ 4]],

         ...,

         [[ 6],
          [ 7],
          [ 6],
          ...,
          [ 5],
          [ 5],
          [ 5]],

         [[ 6],
          [ 6],
          [ 7],
          ...,
          [ 4],
          [ 4],
          [ 4]],

         [[ 5],
          [ 6],
          [ 6],
          ...,
          [ 5],
          [34],
          [12]]],


        [[[ 5],
          [ 5],
          [ 5],
          ...,
          [ 3],
          [ 3],
          [ 4]],

         [[ 6],
          [ 6],
          [ 5],
          ...,
         [ 4],
         [ 4],
         [ 3]],

        [[ 5],
         [ 6],
         [ 5],
         ...,
         [ 4],
         [ 4],
         [ 4]],

        ...,

        [[ 5],
         [ 5],
         [ 5],
         ...,
         [ 5],
         [ 4],
         [ 4]],

        [[ 5],
         [ 5],
         [ 5],
         ...,
         [ 5],
         [ 4],
         [ 4]],

        [[ 5],
         [ 4],
         [ 5],
         ...,
         [ 5],
         [35],
         [12]]]], dtype=uint8)
```

In [30]:
```
y
```

Out[30]:
```
array([0, 0, 0, ..., 9, 9, 9])
```

In [31]:
```
x = x.astype('float32') / 255.0

print(f"x shape: {x.shape}")
print(f"y shape: {y.shape}")
```

```
x shape: (20000, 200, 200, 1)
y shape: (20000,)
```

In [32]:
```
x[:1]
```

Out[32]:
```
array([[[[0.01960784],
         [0.01568628],
         [0.01960784],
         ...,
         [0.01176471],
         [0.01568628],
         [0.01568628]],

        [[0.01568628],
         [0.01568628],
         [0.01960784],
         ...,
         [0.01568628],
         [0.01568628],
         [0.01176471]],

        [[0.01568628],
         [0.01568628],
         [0.01960784],
         ...,
         [0.01568628],
         [0.01568628],
         [0.01568628]],

        ...,

        [[0.01568628],
         [0.01960784],
         [0.01960784],
         ...,
         [0.01568628],
         [0.01568628],
         [0.01568628]],

        [[0.01568628],
         [0.01960784],
         [0.01960784],
         ...,
         [0.01568628],
         [0.01568628],
         [0.01176471]],

        [[0.02352941],
         [0.01568628],
         [0.01960784],
         ...,
         [0.01176471],
         [0.13725491],
         [0.04705882]]]], dtype=float32)
```

In [33]:
```
y[:1]
```

Out[33]:
```
array([0])
```

Spliting Data

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, train_size=0.8,random_state=1234)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(16000, 200, 200, 1)
(4000, 200, 200, 1)
(16000,)
(4000,)
```

Building Convolutional Neural Network

In [ ]:
```
model = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), strides=(2, 2), padding='VALID', input_shape=(width, height, 
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Dropout(0.7),
    layers.Conv2D(filters=64, kernel_size=(3, 3), strides=(2, 2), padding='VALID'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.Dropout(0.7),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Flatten(),
    layers.Dense(128, activation='relu',kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.5),
    layers.Dense(10, activation='softmax')
])
```

In [ ]:
```
model.compile(
    optimizer=optimizers.Adam(learning_rate=0.0001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
```

In [50]:
```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 99, 99, 32)	320
batch_normalization_4 (BatchNormalization)	(None, 99, 99, 32)	128
activation_4 (Activation)	(None, 99, 99, 32)	0
max_pooling2d_4 (MaxPooling2D)	(None, 49, 49, 32)	0
dropout_6 (Dropout)	(None, 49, 49, 32)	0
conv2d_5 (Conv2D)	(None, 24, 24, 64)	18,496
batch_normalization_5 (BatchNormalization)	(None, 24, 24, 64)	256
activation_5 (Activation)	(None, 24, 24, 64)	0
dropout_7 (Dropout)	(None, 24, 24, 64)	0
max_pooling2d_5 (MaxPooling2D)	(None, 12, 12, 64)	0
flatten_2 (Flatten)	(None, 9216)	0
dense_4 (Dense)	(None, 128)	1,179,776
dropout_8 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 10)	1,290

Total params: 1,200,266 (4.58 MB)

Trainable params: 1,200,074 (4.58 MB)

Non-trainable params: 192 (768.00 B)

Training Model

In [ ]:
```
early_stopping = callbacks.EarlyStopping(
                                monitor='val_accuracy',
                                patience=5,
                                restore_best_weights=True)

history=model.fit(
                x_train,y_train,
        batch_size=65,
                steps_per_epoch =50,
                epochs=50,
        validation_split=0.1,
                verbose=1,
                callbacks=[early_stopping])
```

```
Epoch 1/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 38s 684ms/step - accuracy: 0.1067 - loss: 6.8945 - val_accuracy: 0.0950 - val_loss: 4
.7905
Epoch 2/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 666ms/step - accuracy: 0.1342 - loss: 4.7667 - val_accuracy: 0.0950 - val_loss: 4
.7192
```

```
Epoch 3/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 666ms/step - accuracy: 0.1437 - loss: 4.6368 - val_accuracy: 0.0994 - val_loss: 4
.6332
Epoch 4/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 676ms/step - accuracy: 0.1860 - loss: 4.4887 - val_accuracy: 0.1088 - val_loss: 4
.5376
Epoch 5/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 16s 305ms/step - accuracy: 0.2165 - loss: 4.3647 - val_accuracy: 0.1656 - val_loss: 4
.4900
Epoch 6/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 686ms/step - accuracy: 0.2156 - loss: 4.2984 - val_accuracy: 0.2037 - val_loss: 4
.3877
Epoch 7/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 676ms/step - accuracy: 0.2670 - loss: 4.1085 - val_accuracy: 0.2344 - val_loss: 4
.2772
Epoch 8/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 672ms/step - accuracy: 0.2834 - loss: 3.9634 - val_accuracy: 0.2325 - val_loss: 4
.1545
Epoch 9/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 673ms/step - accuracy: 0.2973 - loss: 3.8186 - val_accuracy: 0.3413 - val_loss: 4
.0134
Epoch 10/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 16s 317ms/step - accuracy: 0.3190 - loss: 3.6679 - val_accuracy: 0.4319 - val_loss: 3
.9420
Epoch 11/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 35s 696ms/step - accuracy: 0.3380 - loss: 3.5493 - val_accuracy: 0.4363 - val_loss: 3
.7998
Epoch 12/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 662ms/step - accuracy: 0.3536 - loss: 3.4409 - val_accuracy: 0.5750 - val_loss: 3
.6467
Epoch 13/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 661ms/step - accuracy: 0.3605 - loss: 3.3409 - val_accuracy: 0.6431 - val_loss: 3
.4780
Epoch 14/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 656ms/step - accuracy: 0.4079 - loss: 3.1549 - val_accuracy: 0.6662 - val_loss: 3
.3201
Epoch 15/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 15s 300ms/step - accuracy: 0.4323 - loss: 3.0280 - val_accuracy: 0.7106 - val_loss: 3
.2621
Epoch 16/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 87s 664ms/step - accuracy: 0.4126 - loss: 3.0303 - val_accuracy: 0.6800 - val_loss: 3
.0884
Epoch 17/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 653ms/step - accuracy: 0.4586 - loss: 2.8437 - val_accuracy: 0.7300 - val_loss: 2
.9583
Epoch 18/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 659ms/step - accuracy: 0.4636 - loss: 2.7879 - val_accuracy: 0.7231 - val_loss: 2
.8243
Epoch 19/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 672ms/step - accuracy: 0.4881 - loss: 2.6667 - val_accuracy: 0.7469 - val_loss: 2
.7278
Epoch 20/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 17s 328ms/step - accuracy: 0.4934 - loss: 2.5948 - val_accuracy: 0.7175 - val_loss: 2
.6953
Epoch 21/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 673ms/step - accuracy: 0.5043 - loss: 2.4969 - val_accuracy: 0.7781 - val_loss: 2
.6030
Epoch 22/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 665ms/step - accuracy: 0.5074 - loss: 2.4380 - val_accuracy: 0.7681 - val_loss: 2
.4894
Epoch 23/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 682ms/step - accuracy: 0.5274 - loss: 2.3305 - val_accuracy: 0.7669 - val_loss: 2
.3461
Epoch 24/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 670ms/step - accuracy: 0.5755 - loss: 2.1976 - val_accuracy: 0.8225 - val_loss: 2
.2815
Epoch 25/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 15s 303ms/step - accuracy: 0.5688 - loss: 2.1369 - val_accuracy: 0.7912 - val_loss: 2
.2403
Epoch 26/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 664ms/step - accuracy: 0.5734 - loss: 2.1141 - val_accuracy: 0.8175 - val_loss: 2
.1853
Epoch 27/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 663ms/step - accuracy: 0.6175 - loss: 1.9473 - val_accuracy: 0.8288 - val_loss: 2
.0466
Epoch 28/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 662ms/step - accuracy: 0.5996 - loss: 1.9373 - val_accuracy: 0.8256 - val_loss: 2
.0231
Epoch 29/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 658ms/step - accuracy: 0.6310 - loss: 1.8571 - val_accuracy: 0.8319 - val_loss: 1
.9527
Epoch 30/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 16s 309ms/step - accuracy: 0.6259 - loss: 1.8394 - val_accuracy: 0.8331 - val_loss: 1
```

```
.9086
Epoch 31/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 36s 731ms/step - accuracy: 0.6745 - loss: 1.7283 - val_accuracy: 0.8431 - val_loss: 1
.8085
Epoch 32/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 47s 944ms/step - accuracy: 0.6569 - loss: 1.7070 - val_accuracy: 0.8519 - val_loss: 1
.7565
Epoch 33/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 37s 738ms/step - accuracy: 0.6968 - loss: 1.6038 - val_accuracy: 0.8475 - val_loss: 1
.6958
Epoch 34/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 683ms/step - accuracy: 0.7047 - loss: 1.5499 - val_accuracy: 0.8694 - val_loss: 1
.6074
Epoch 35/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 16s 310ms/step - accuracy: 0.6866 - loss: 1.5144 - val_accuracy: 0.8719 - val_loss: 1
.5572
Epoch 36/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 674ms/step - accuracy: 0.7102 - loss: 1.4806 - val_accuracy: 0.8825 - val_loss: 1
.5264
Epoch 37/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 676ms/step - accuracy: 0.6902 - loss: 1.5059 - val_accuracy: 0.8788 - val_loss: 1
.4542
Epoch 38/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 668ms/step - accuracy: 0.7255 - loss: 1.3919 - val_accuracy: 0.8737 - val_loss: 1
.3804
Epoch 39/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 33s 668ms/step - accuracy: 0.7299 - loss: 1.3697 - val_accuracy: 0.8969 - val_loss: 1
.3259
Epoch 40/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 16s 317ms/step - accuracy: 0.7530 - loss: 1.3042 - val_accuracy: 0.9106 - val_loss: 1
.3308
Epoch 41/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 678ms/step - accuracy: 0.7743 - loss: 1.2301 - val_accuracy: 0.8975 - val_loss: 1
.2880
Epoch 42/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 686ms/step - accuracy: 0.7746 - loss: 1.1881 - val_accuracy: 0.9038 - val_loss: 1
.2022
Epoch 43/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 684ms/step - accuracy: 0.7567 - loss: 1.2177 - val_accuracy: 0.9225 - val_loss: 1
.1885
Epoch 44/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 690ms/step - accuracy: 0.7623 - loss: 1.1677 - val_accuracy: 0.9231 - val_loss: 1
.1389
Epoch 45/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 16s 314ms/step - accuracy: 0.7725 - loss: 1.1509 - val_accuracy: 0.9119 - val_loss: 1
.1055
Epoch 46/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 682ms/step - accuracy: 0.7948 - loss: 1.0892 - val_accuracy: 0.9425 - val_loss: 1
.0642
Epoch 47/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 37s 736ms/step - accuracy: 0.8020 - loss: 1.0466 - val_accuracy: 0.9269 - val_loss: 1
.0067
Epoch 48/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 682ms/step - accuracy: 0.8002 - loss: 1.0458 - val_accuracy: 0.9331 - val_loss: 0
.9891
Epoch 49/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 34s 674ms/step - accuracy: 0.8214 - loss: 0.9958 - val_accuracy: 0.9312 - val_loss: 0
.9590
Epoch 50/50
50/50 ━━━━━━━━━━━━━━━━━━━━ 16s 306ms/step - accuracy: 0.8160 - loss: 0.9485 - val_accuracy: 0.9400 - val_loss: 0
.9180
```

Model Evaluation

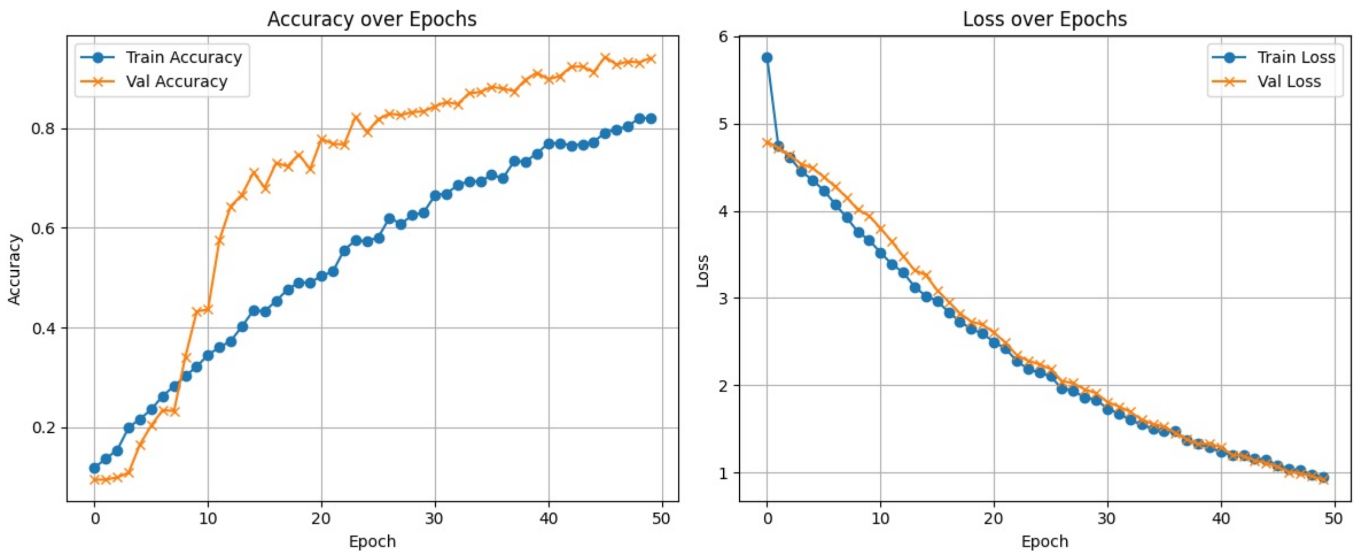
In [57]:
```
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=1)

print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 7s 55ms/step - accuracy: 0.9458 - loss: 1.0514
Test Loss: 1.0560
Test Accuracy: 0.9467
```

In [59]:
```
# **Loading Model**
#from keras.models import load_model

#model.save('Hand_Gesture_Recognition_model.h5')
#print("Model saved successfully!")
```

```
Model saved successfully!
```

In [ ]:
```
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Val Accuracy', marker='x')
plt.title('Accuracy over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss', marker='o')
plt.plot(history.history['val_loss'], label='Val Loss', marker='x')
plt.title('Loss over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```



The plots suggest that the model is learning effectively as both accuracy increases and loss decreases over time. The most notable observation is that the validation accuracy is consistently higher than the training accuracy, and validation loss is often similar to or slightly lower than training loss. This is an unusual but not impossible scenario. It could indicate:

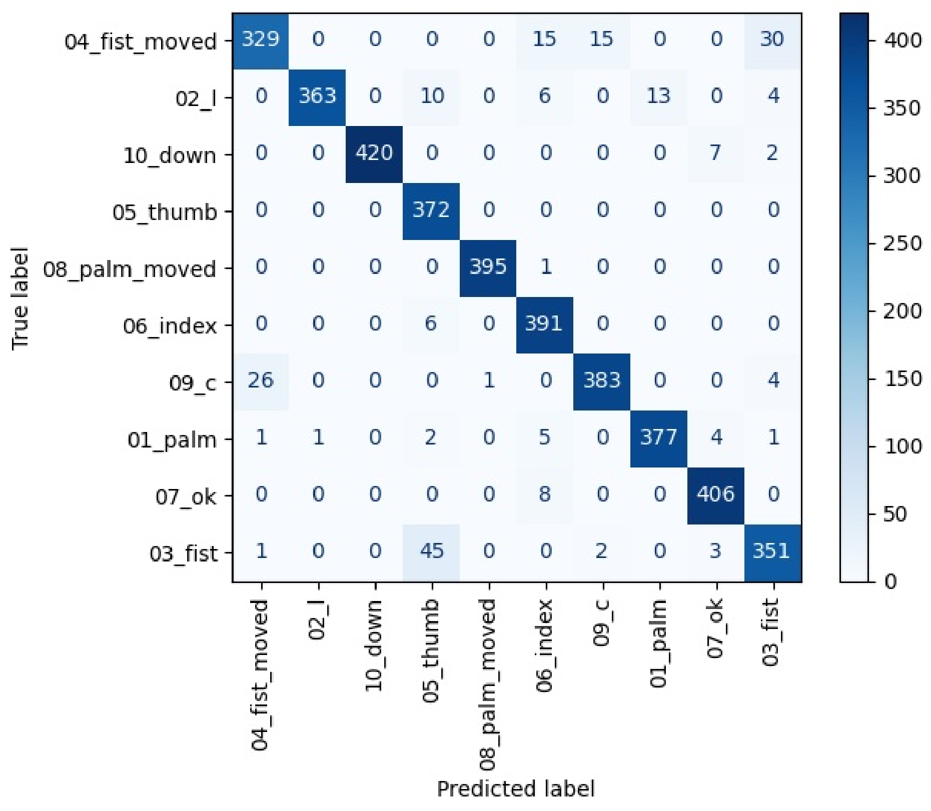
- The validation set is "easier" than the training set.
- The training process incorporates some techniques (e.g., strong regularization, specific data augmentations) that make the training loss higher or accuracy lower during the training phase itself, but which ultimately lead to better generalization on the validation set.
- There might be a slight data mismatch or difference in complexity between the training and validation sets.

In [63]:
```
y_pred_probs = model.predict(x_test)
y_pred = np.argmax(y_pred_probs, axis=1)

y_true = y_test
```

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 6s 47ms/step
```

In [ ]:
```
plt.figure(figsize = (12,6))
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=categories)
disp.plot(xticks_rotation='vertical', cmap='Blues')
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



True label	04_fist_moved	02_l	10_down	05_thumb	08_palm_moved	06_index	09_c	01_palm	07_ok	03_fist
04_fist_moved	329	0	0	0	0	15	15	0	0	30
02_l	0	363	0	10	0	6	0	13	0	4
10_down	0	0	420	0	0	0	0	0	7	2
05_thumb	0	0	0	372	0	0	0	0	0	0
08_palm_moved	0	0	0	0	395	1	0	0	0	0
06_index	0	0	0	6	0	391	0	0	0	0
09_c	26	0	0	0	1	0	383	0	0	4
01_palm	1	1	0	2	0	5	0	377	4	1
07_ok	0	0	0	0	0	8	0	0	406	0
03_fist	1	0	0	45	0	0	2	0	3	351

- **Rows (True Label):** These represent the actual classes of the data. For example, "04_fist_moved", "02_l", "10_down", etc.
- **Columns (Predicted Label):** These represent the classes that the model predicted.
- **Numbers in the cells:** Each cell at the intersection of a true label row and a predicted label column shows how many instances of the true label were predicted as that specific predicted label.
- **Diagonal elements:** The numbers on the diagonal (from top-left to bottom-right) represent the number of correctly classified instances for each class. For example, 329 instances of "04_fist_moved" were correctly predicted as "04_fist_moved".
- **Off-diagonal elements:** These numbers represent misclassifications. For example, in the "04_fist_moved" row, there are instances that were actually "04_fist_moved" but were predicted as other classes (e.g., 15 as "10_down", 15 as "09_c", etc.).
- **Color Bar:** The color bar on the right indicates the count range, with darker shades of blue representing higher counts.

Conclusion

In this project, we developed a Convolutional Neural Network (CNN) model to classify images of hand gestures into 10 distinct categories. After training the model on labeled image data, we evaluated its performance using unseen test images—selecting one sample from each class—and observed accurate predictions for most cases. The model demonstrated good generalization ability and robustness to new inputs, making it suitable for potential real-time gesture recognition applications. This work highlights the effectiveness of CNNs in visual pattern recognition and provides a solid foundation for further improvements or deployment in interactive systems.

```
In [ ]:
```