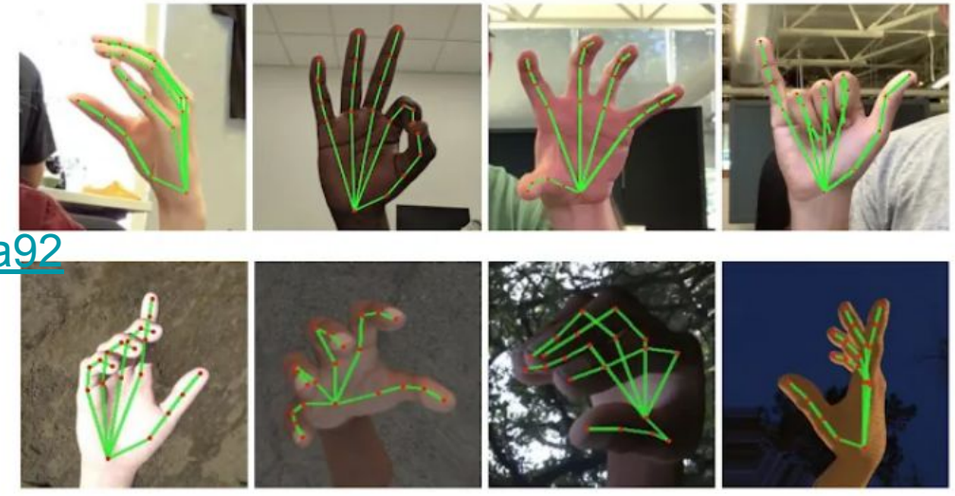
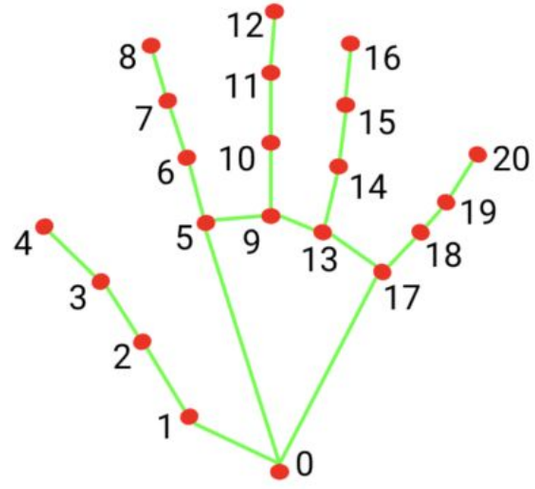
AI VIETNAM
All-in-One Course
(TA Session)

Light Controlling Using Hand Gestures

https://chatgpt.com/c/67271026-a92c-8011-a73e-2a58da8f60ca



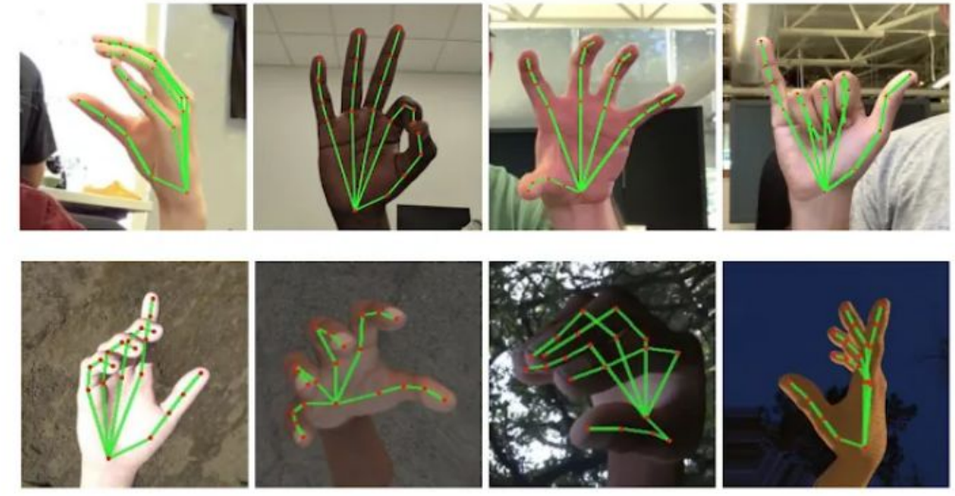


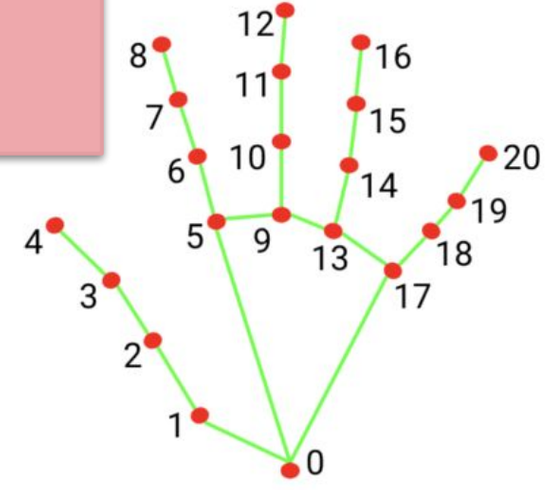
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Nguyen Tho Anh Khoa - Nguyen Dang Nha

Objectives

- Develop a Natural Interaction System
- Set Up the Project Environment
- Collect and Prepare Gesture Data
- Build and Train an MLP Model
- Evaluate Model Performance
- Deploy Real-Time Gesture Recognition





0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Outline

1. Hand Gestures Recognition
2. Project Overview
3. Step 0: Generate Data
4. Step 1: Hand Gesture
5. Step 2: Control Lights (Simulation + Reality)
6. AI Application in Internet of Things (IoT)

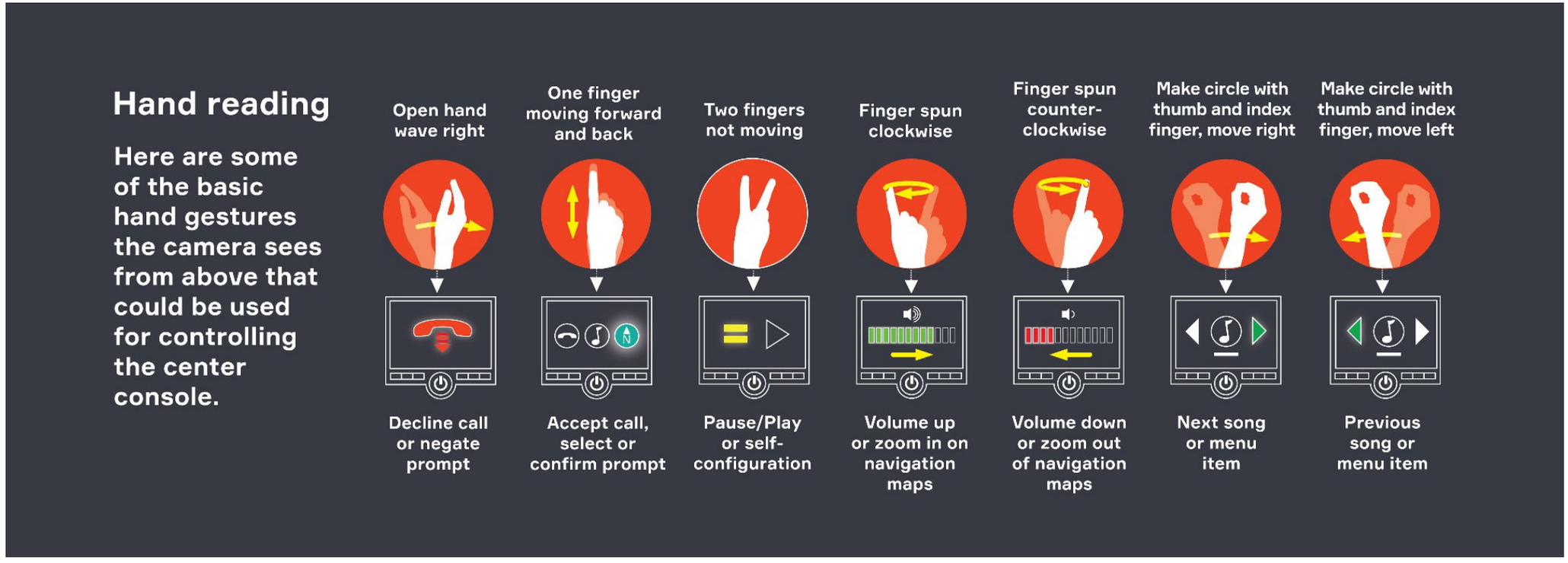
AI VIETNAM
All-in-One Course
(TA Session)

Hand Gestures Recognition

➢ What Is Hand Gestures Recognition?

Definition: HGR is technology that identifies and interprets hand and finger movements to enable intuitive interaction between humans and computers or robotic systems.

Significance: It enhances human-computer interaction (HCI) and human-robot interaction (HRI), with applications in healthcare, education, entertainment, and more.



AI VIETNAM
All-in-One Course
(TA Session)

Hand Gestures Recognition

➢ What Is Hand Gestures Recognition?

Key Components of HGR:

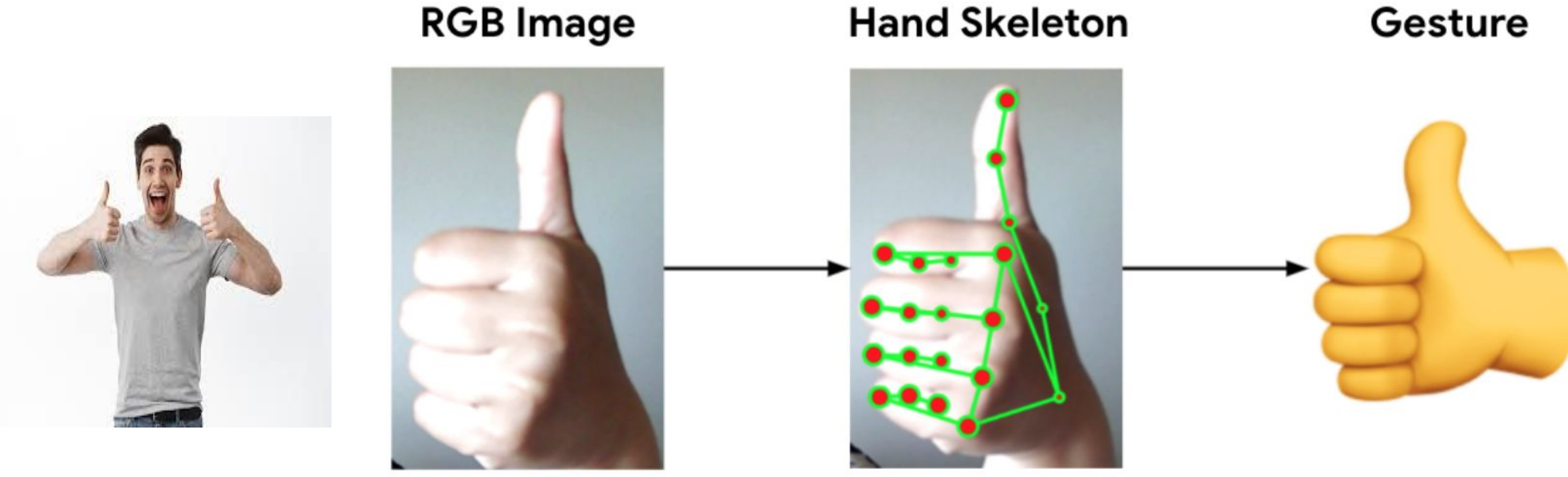
1. **Data Acquisition**:
   - Utilizes cameras or sensors (standard video cameras or specialized depth cameras) to capture visual signals of hand gestures.
2. **Feature Extraction**:
   - Processes captured images to identify key features such as hand positions, shapes, and movements using computer vision algorithms.
3. **Classification**:
   - Applies machine learning or deep learning algorithms (e.g., MLP, CNN, SVM) to classify gestures into predefined categories.

Types of Hand Gestures:

- **Static Gestures**:
  - Fixed hand positions like pointing or forming specific shapes.
  - Recognition focuses on identifying these positions at a specific moment.
- **Dynamic Gestures**:
  - Movements of hands over time, such as waving or drawing patterns.
  - Recognition requires tracking the motion paths of the hands.

Hand Gestures Recognition

➢ What Is Hand Gestures Recognition?



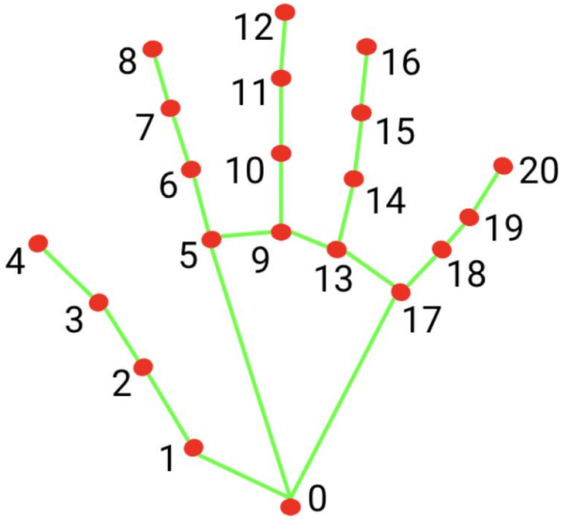
Hình 1: Hand Gesture Recognition

AI VIETNAM
All-in-One Course
(TA Session)

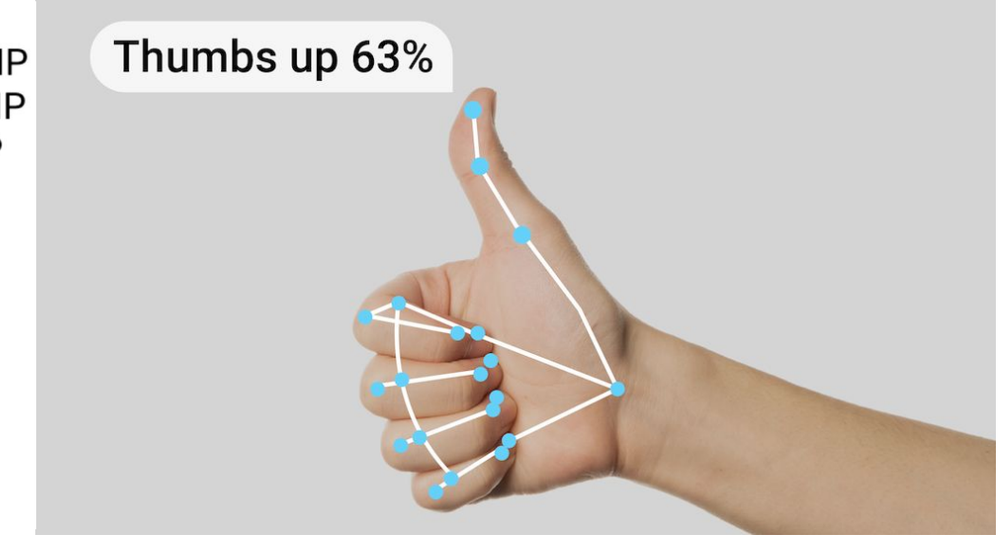
Hand Gestures Recognition

➢ What Is Hand Gestures Recognition?

Mediapipe: palm detection model and hand landmarks detection mode
Hand landmark detection model:
- The keypoint localization of 21 hand-knuckle coordinates
- Model was trained on approximately 30K real-world images

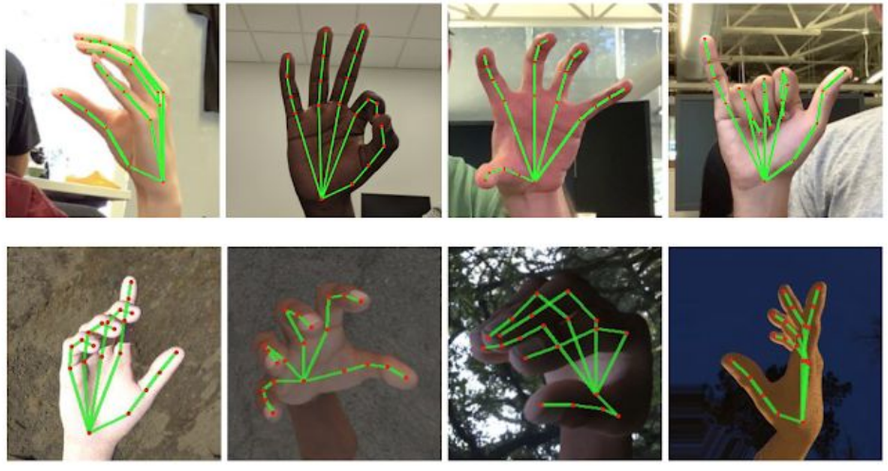


0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

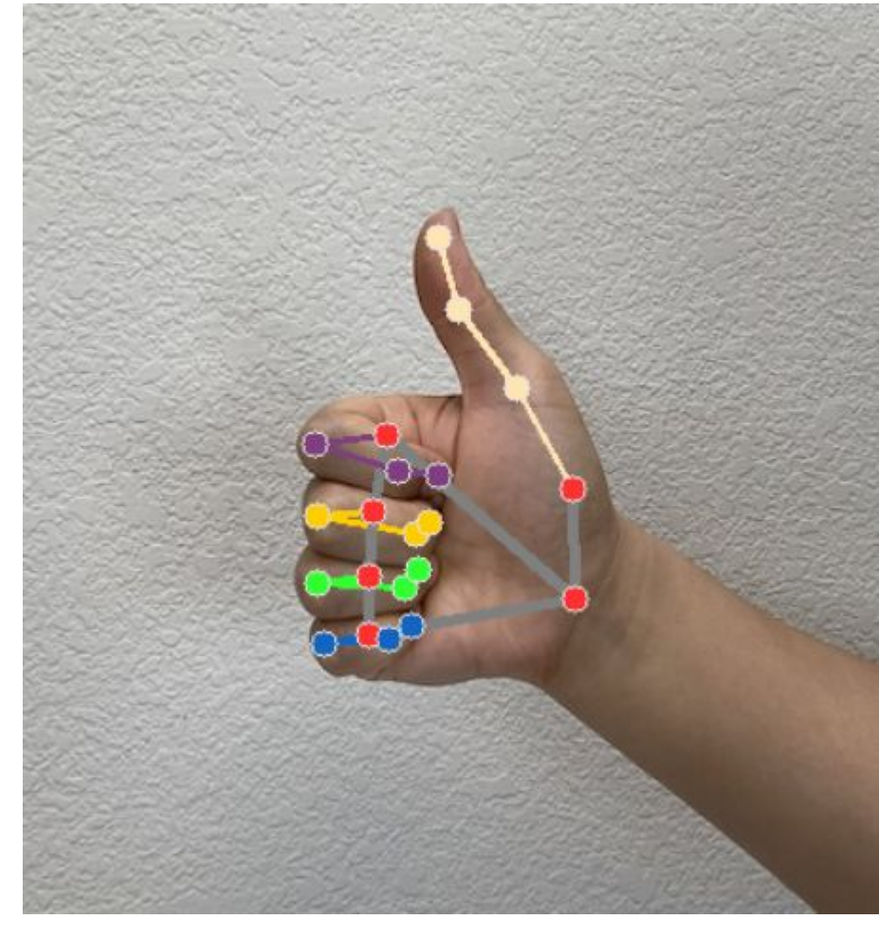


Hand Gestures Recognition

➢ What Is Hand Gestures Recognition?



```
GestureRecognizerResult:
  Handedness:
    Categories #0:
      index         : 0
      score         : 0.98396
      categoryName  : Left
  Gestures:
    Categories #0:
      score         : 0.76893
      categoryName  : Thumb_Up
  Landmarks:
    Landmark #0:
      x             : 0.638852
      y             : 0.671197
      z             : -3.41E-7
    Landmark #1:
      x             : 0.634599
      y             : 0.536441
      z             : -0.06984
    ... (21 landmarks for a hand)
  WorldLandmarks:
    Landmark #0:
      x             : 0.067485
      y             : 0.031084
      z             : 0.055223
    Landmark #1:
      x             : 0.063209
      y             : -0.00382
      z             : 0.020920
    ... (21 world landmarks for a hand)
```



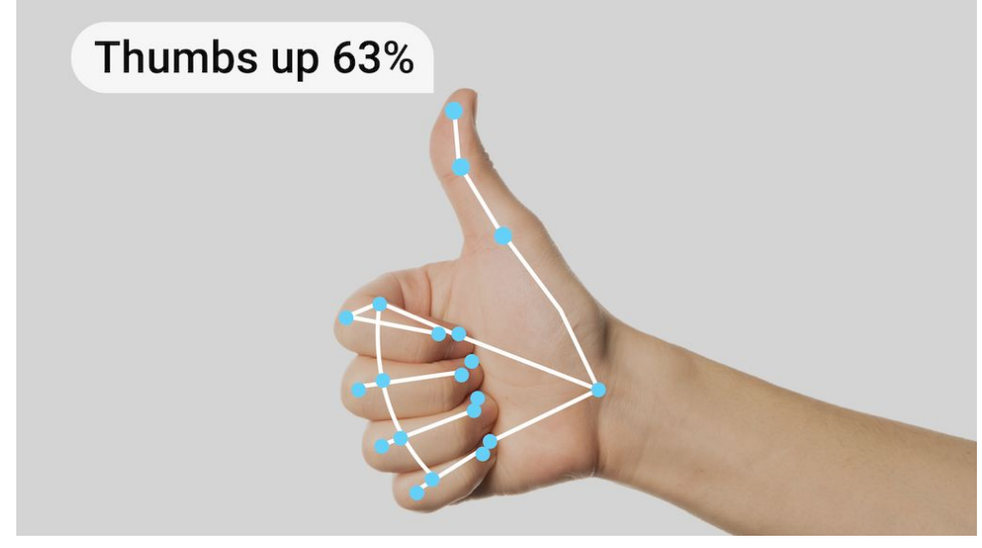
AI VIETNAM
All-in-One Course
(TA Session)

Hand Gestures Recognition

➢ What Is Hand Gestures Recognition?

Hand Gesture classification model: Hand gesture classification involves the recognition of specific hand movements, which can be used in various applications such as human-computer interaction, sign language interpretation, and virtual reality. Several models and techniques have been developed to enhance the accuracy and efficiency of gesture recognition.

```
0 - Unrecognized gesture, label: Unknown
1 - Closed fist, label: Closed_Fist
2 - Open palm, label: Open_Palm
3 - Pointing up, label: Pointing_Up
4 - Thumbs down, label: Thumb_Down
5 - Thumbs up, label: Thumb_Up
6 - Victory, label: Victory
7 - Love, label: ILoveYou
```



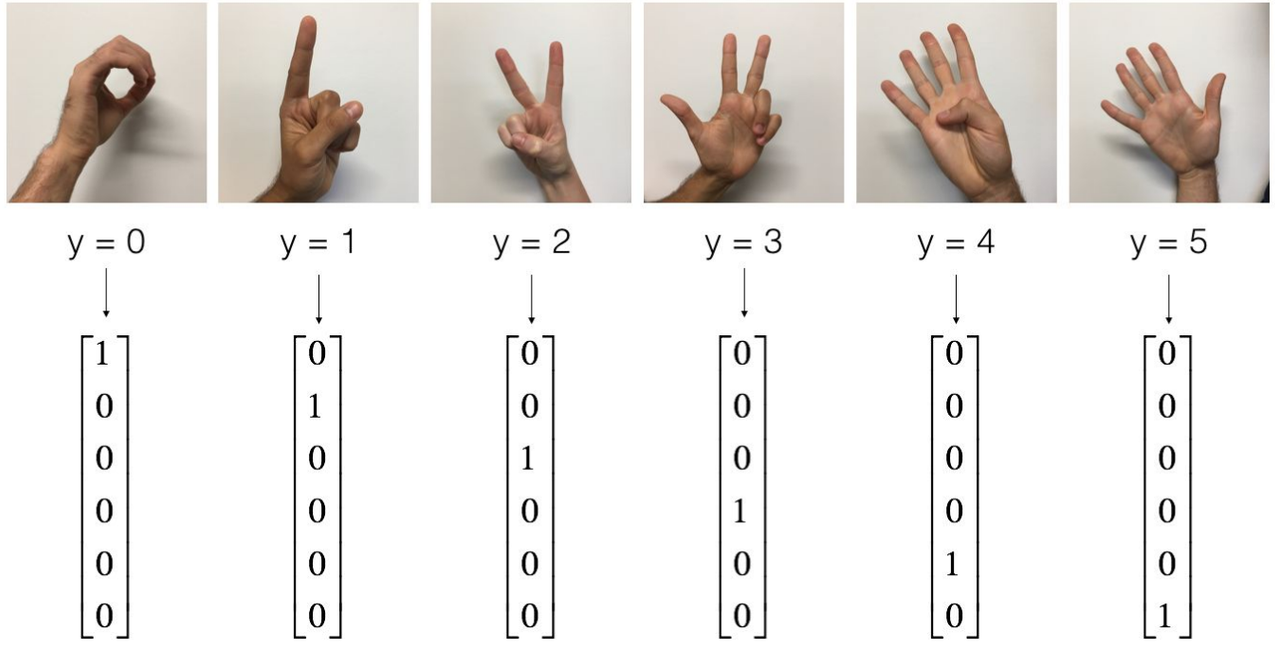
AI VIETNAM
All-in-One Course
(TA Session)

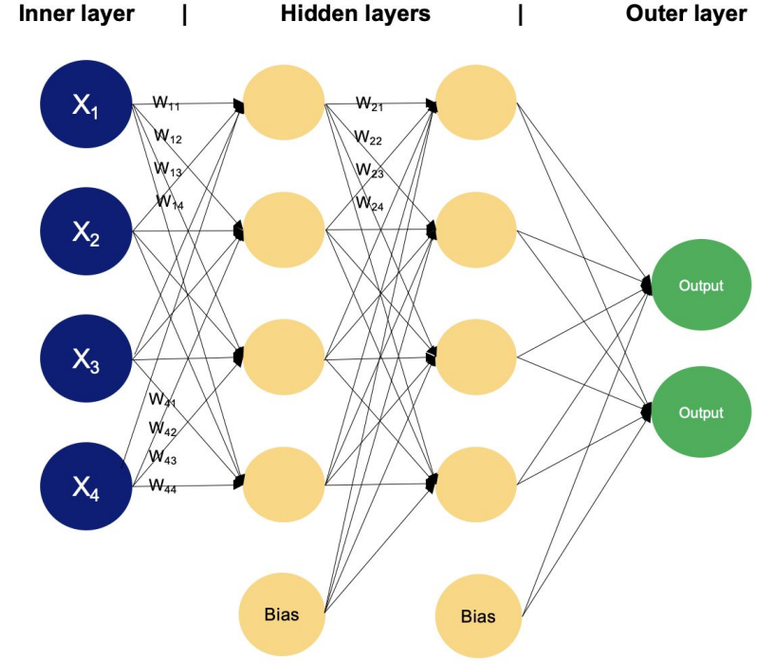
Hand Gestures Recognition

➢ What Is Hand Gestures Recognition?

MLP Model Structure

- Input Layer: Receives features extracted from gesture data
- Hidden Layers: One or more layers that process the inputs through weighted connections and nonlinear activation functions.
- Output Layer: Produces the final classification output, indicating the recognized gesture.



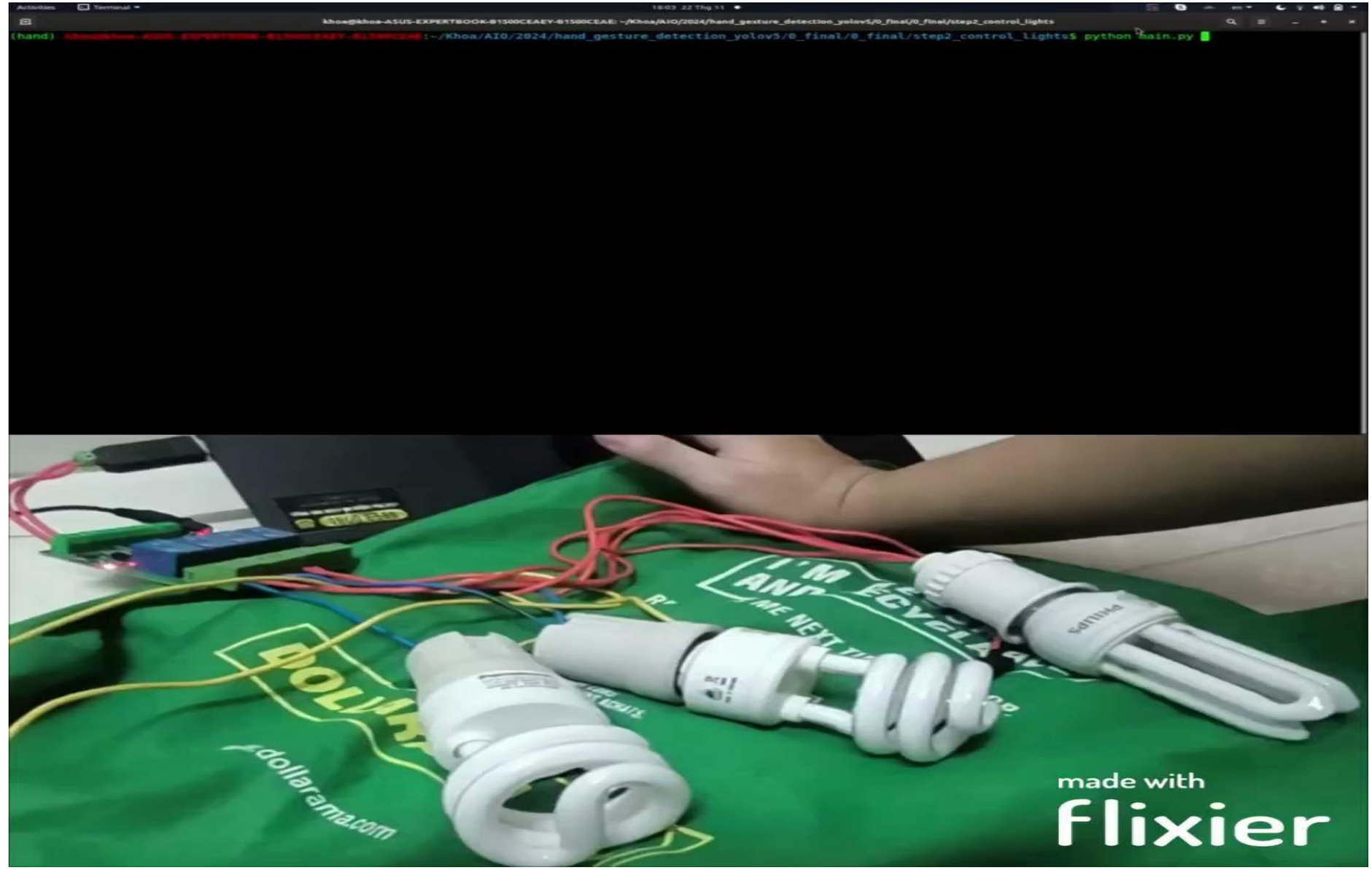


Project Overview

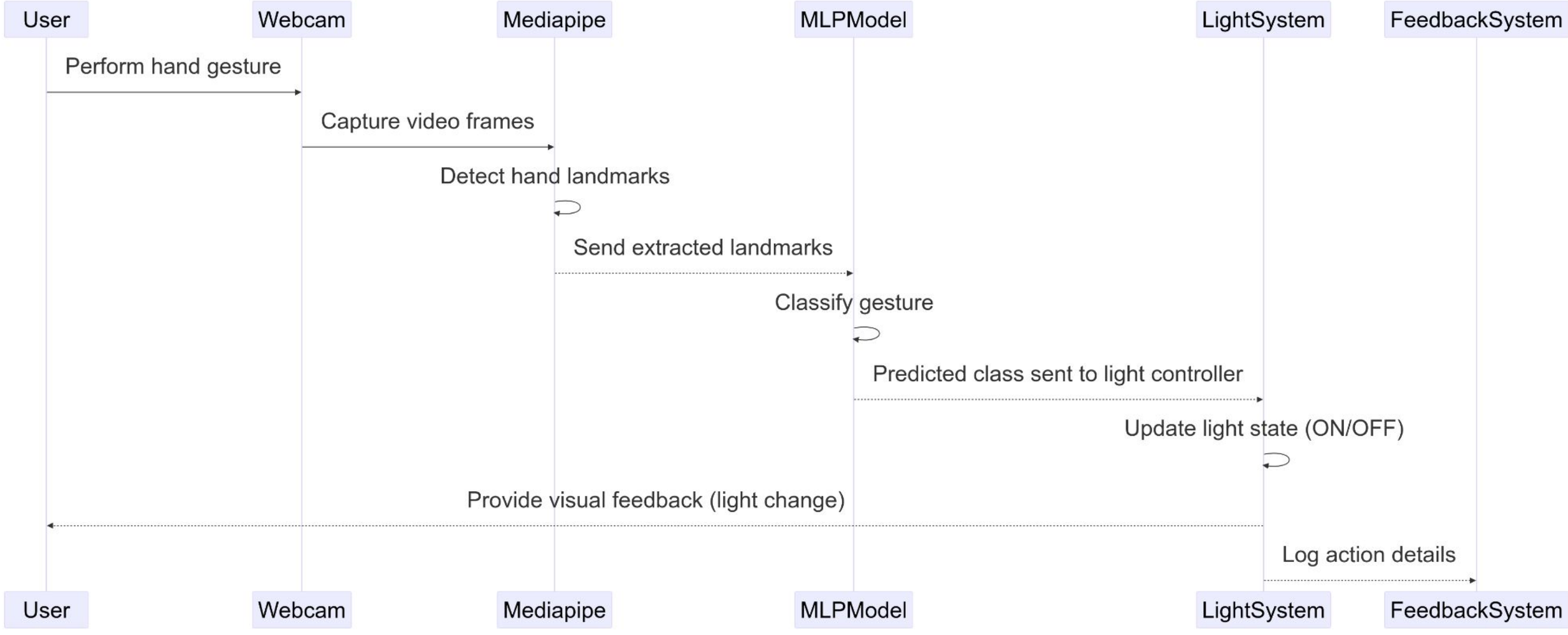
➢ Demo

```
step2_control_lights/
├── controller.py
├── detect_simulation.py
└── hand_gesture.yaml
```

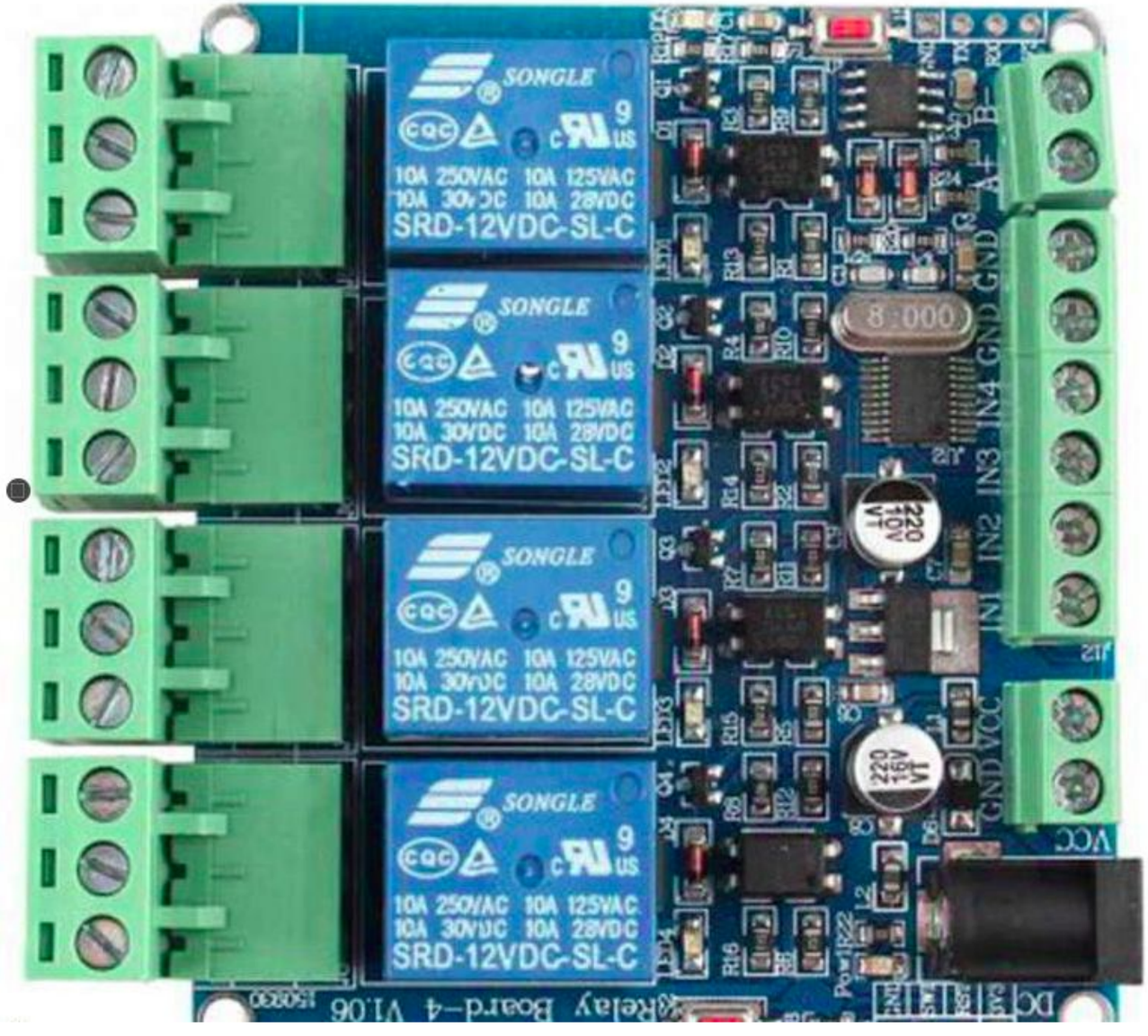
là file main.py
trong clip demo



Project Overview

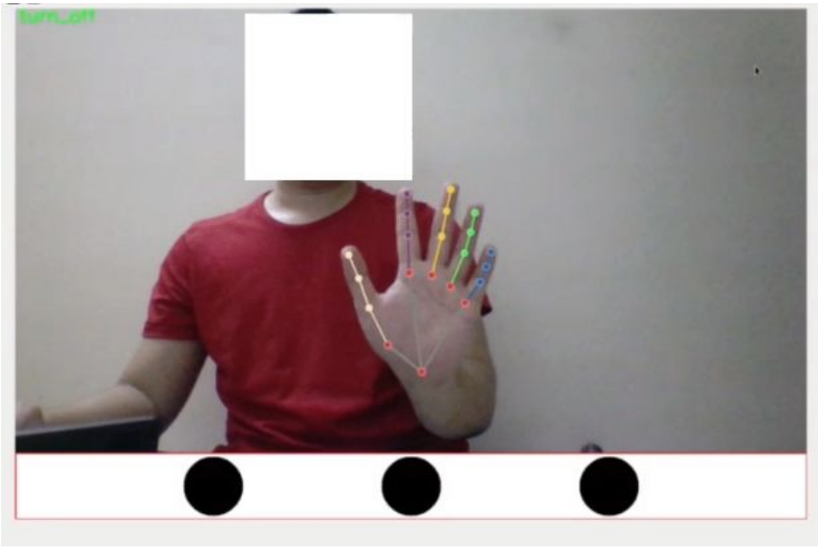


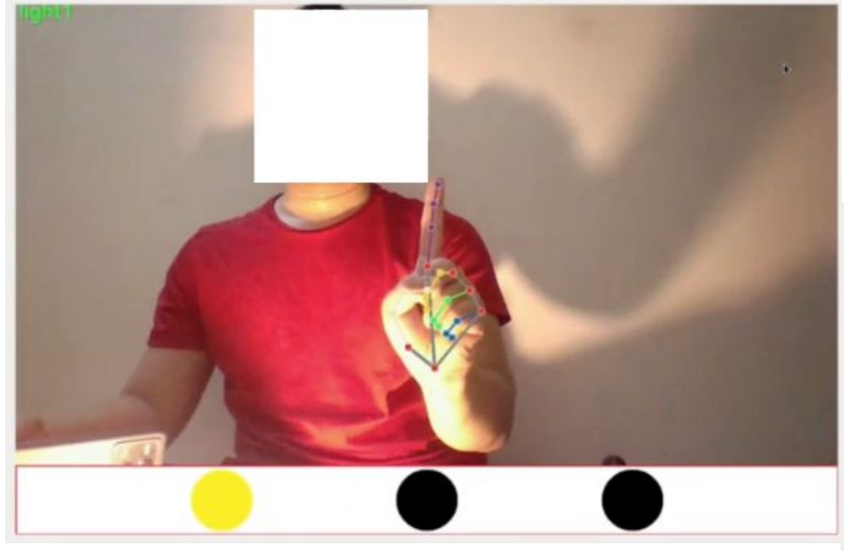
Project Overview

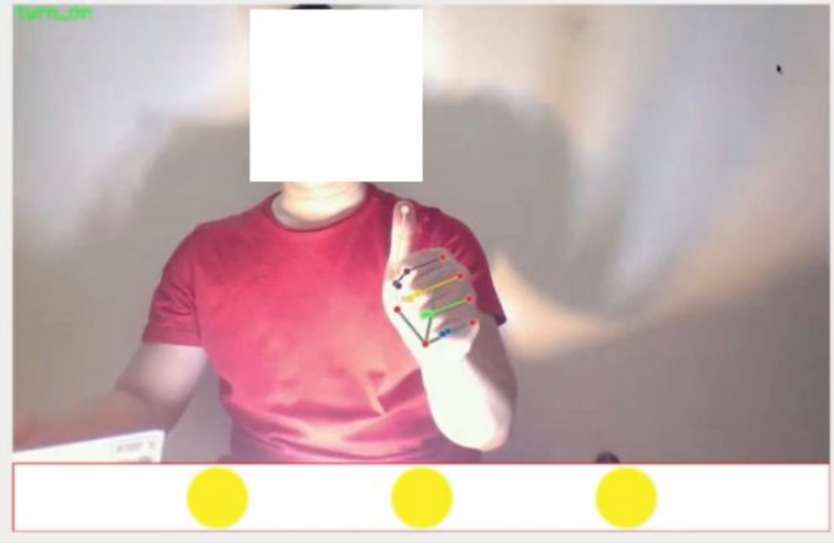


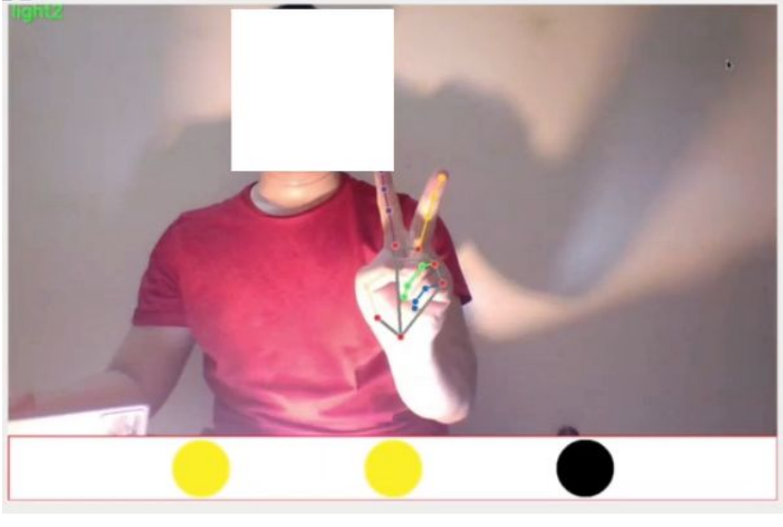
AI VIETNAM
All-in-One Course
(TA Session)

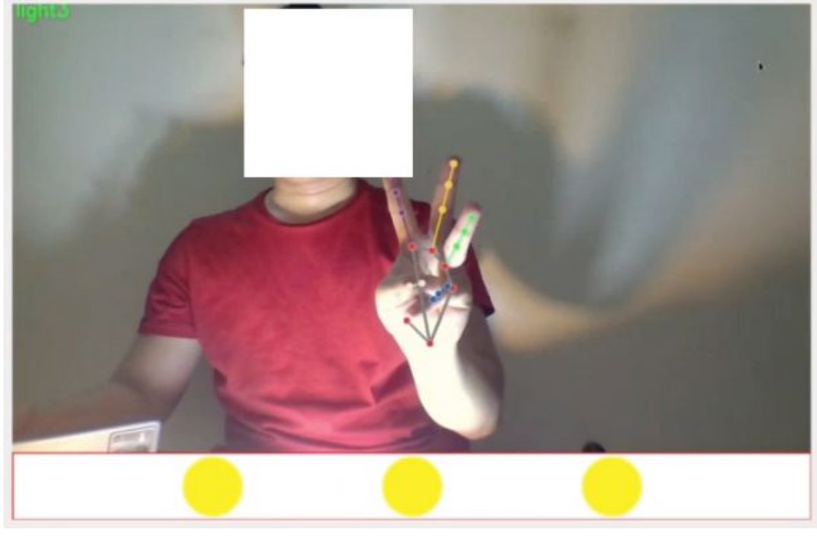
Project Overview











Quiz

Thư viện nào được sử dụng để detect hand landmarks

(A) . Pytorch

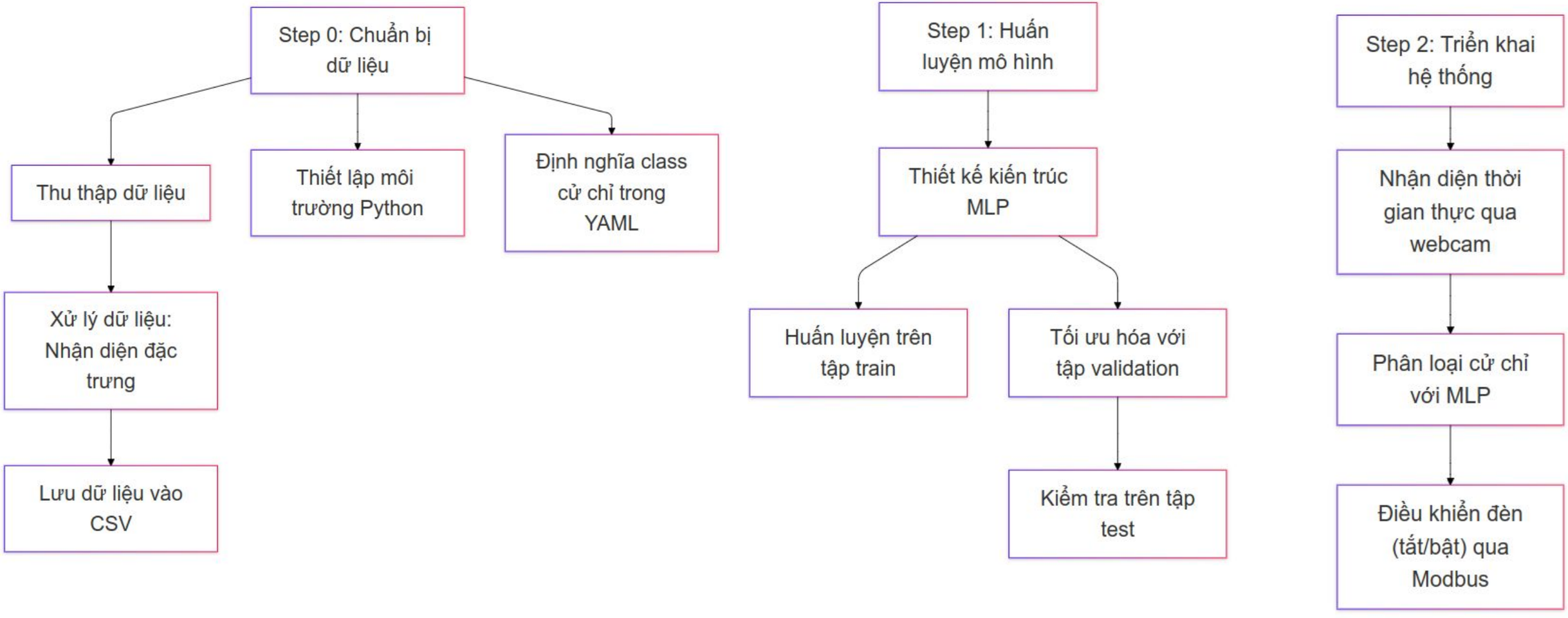
(B). Tensorflow

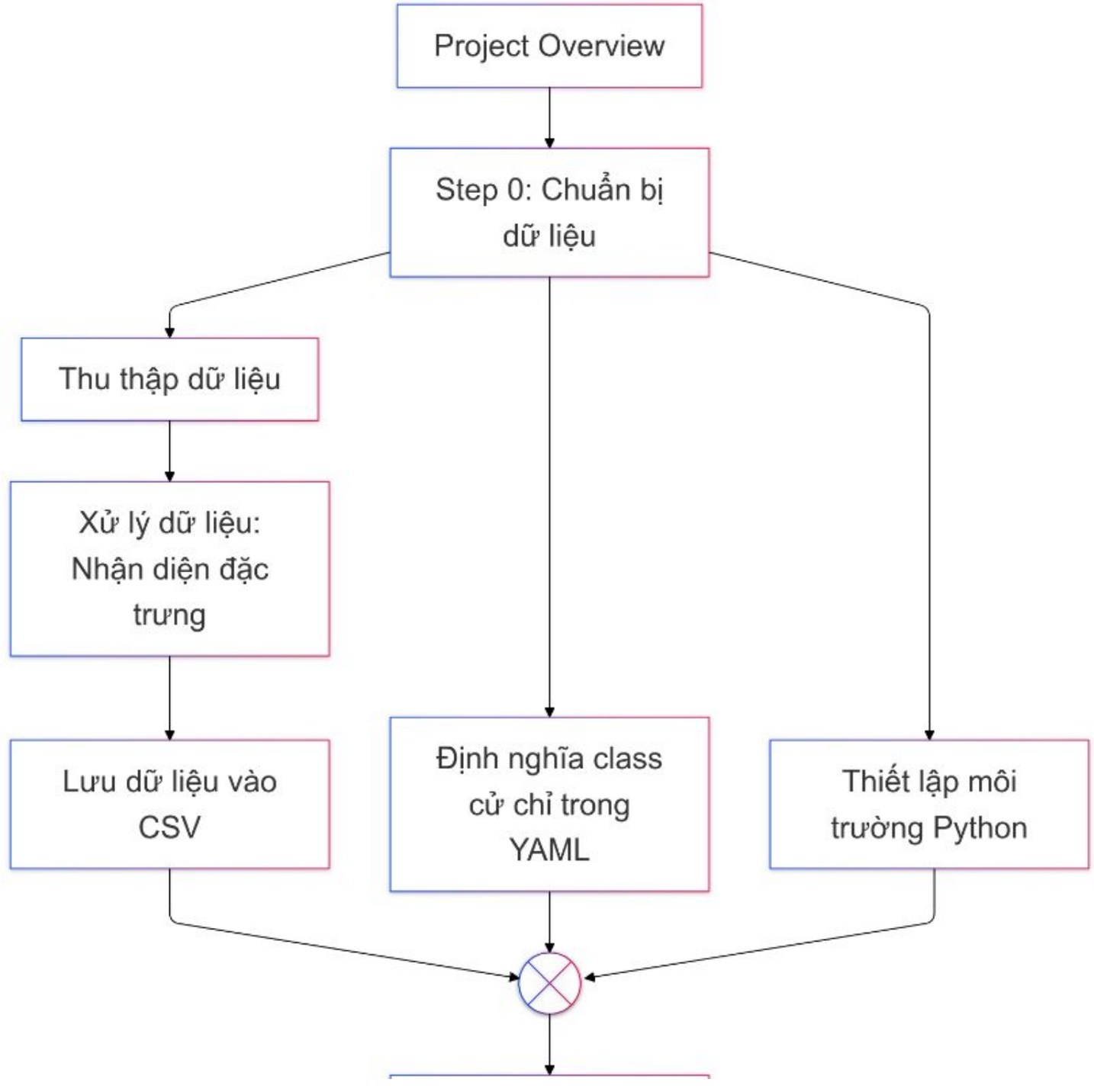
(C). Scikit Learn

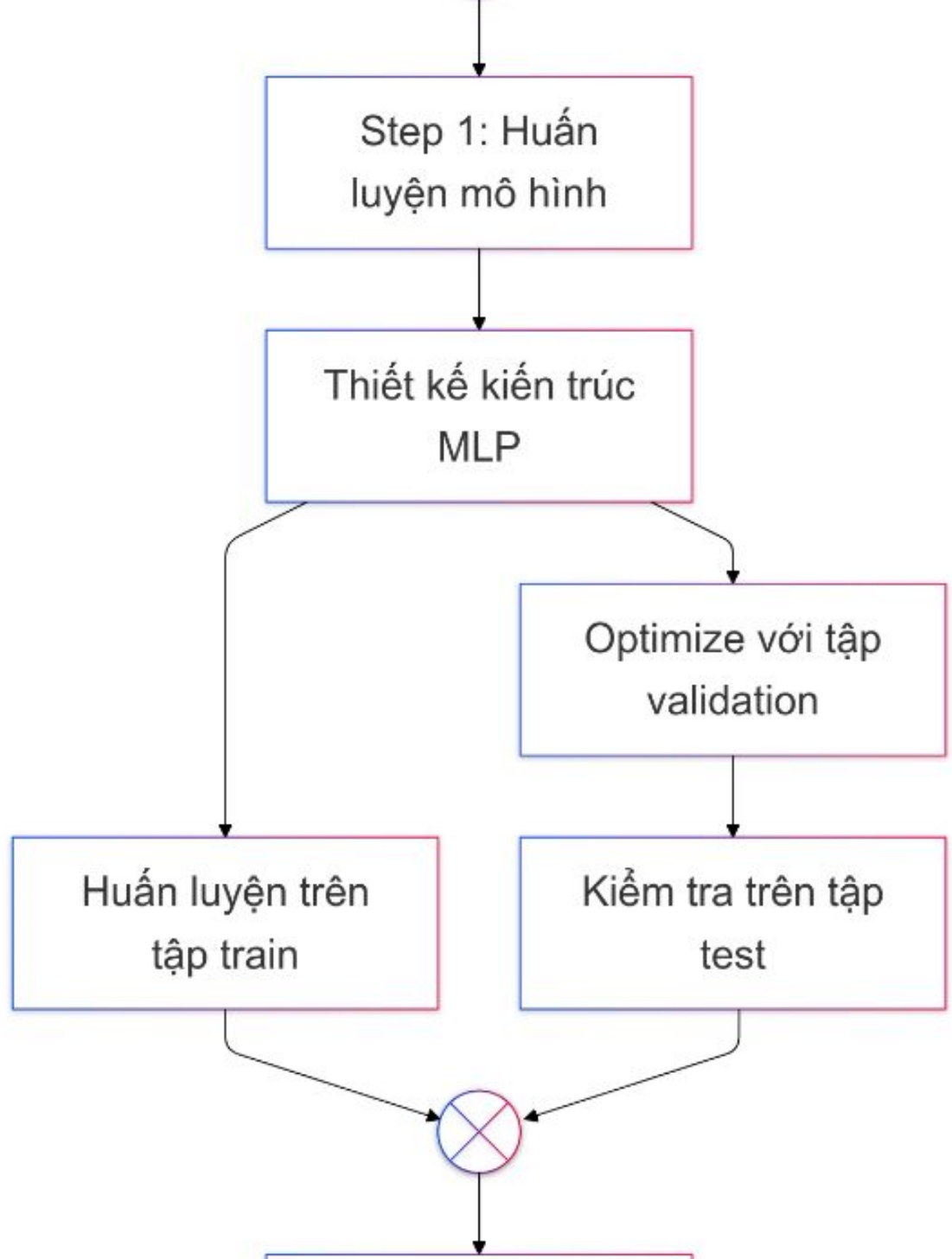
(D). Mediapipe

Project Overview

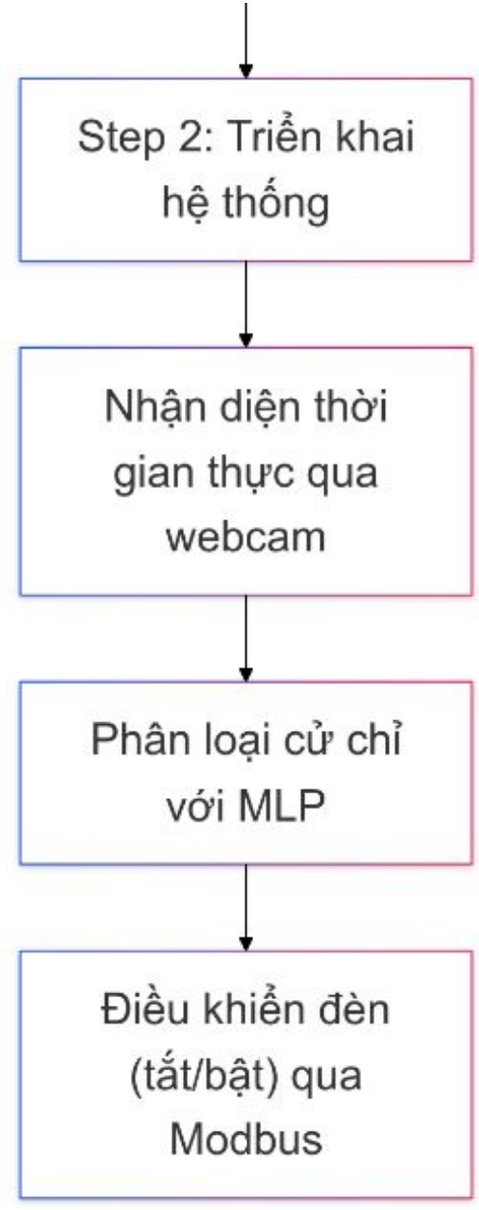
➢ Steps




Project Overview
Step 0: Chuẩn bị dữ liệu
Thu thập dữ liệu
Xử lý dữ liệu: Nhận diện đặc trưng
Lưu dữ liệu vào CSV
Định nghĩa class cử chỉ trong YAML
Thiết lập môi trường Python


Step 1: Huấn luyện mô hình
Thiết kế kiến trúc MLP
Optimize với tập validation
Huấn luyện trên tập train
Kiểm tra trên tập test

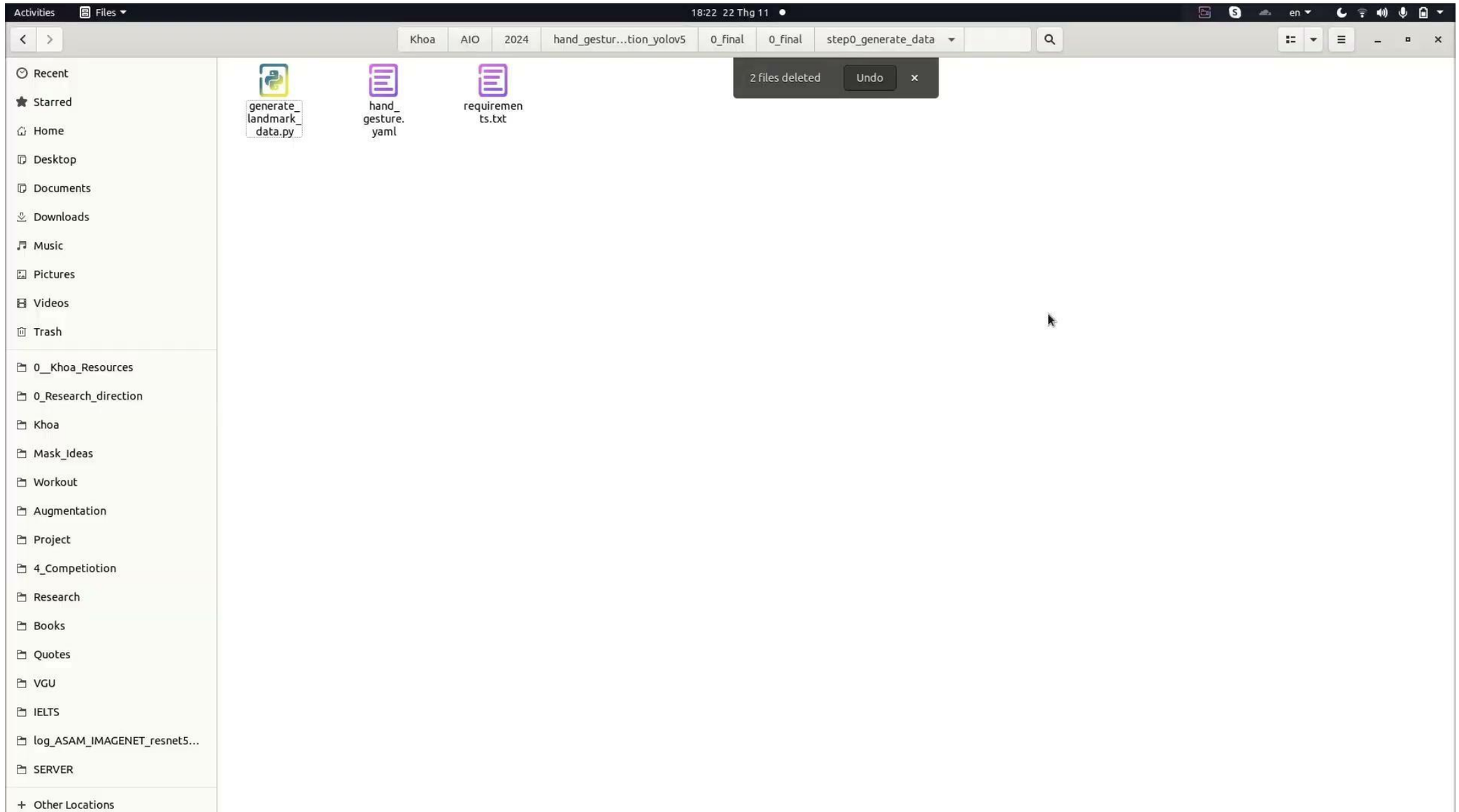
Project Overview



AI VIETNAM
All-in-One Course
(TA Session)

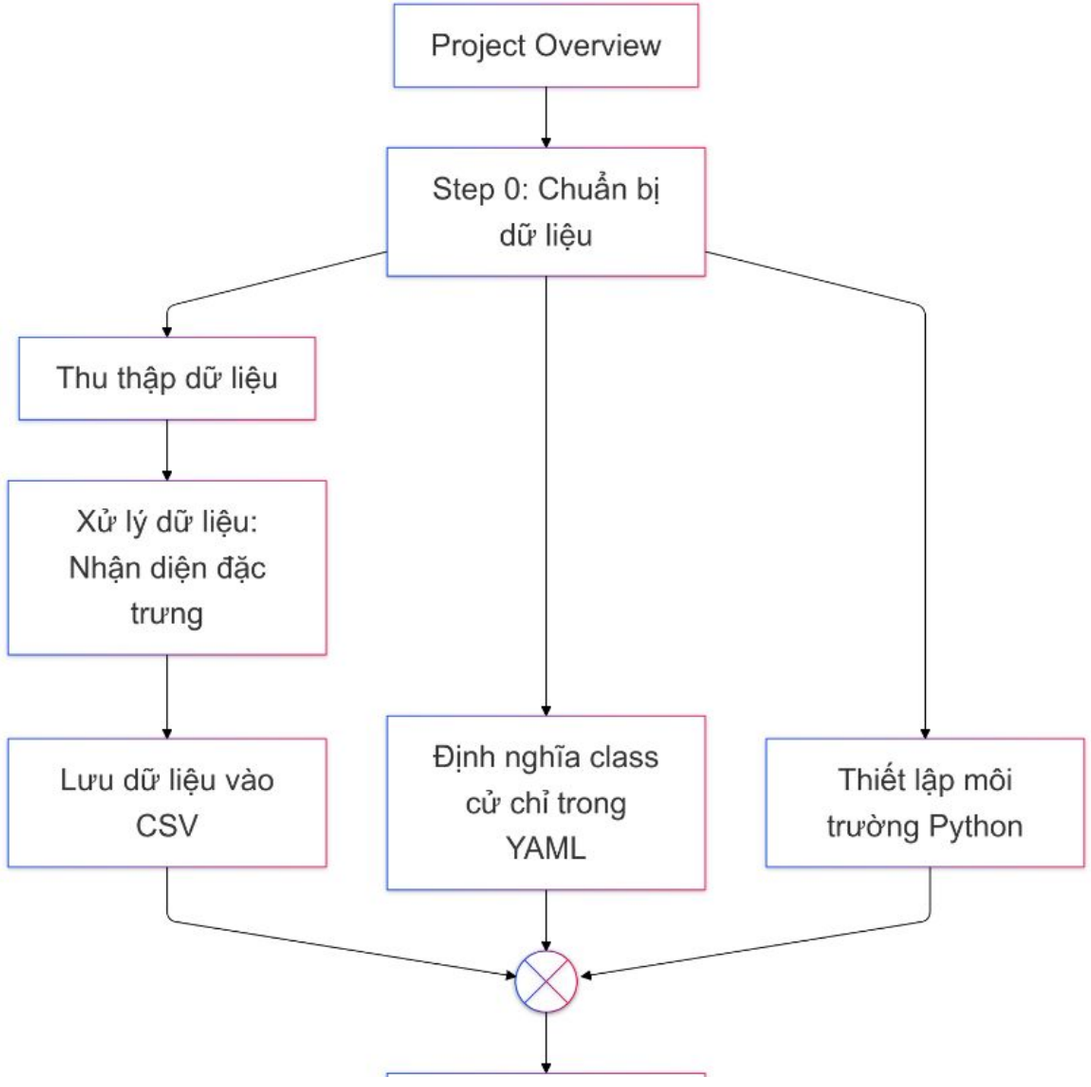
Step 0: Generate Data

➢ Generate Data



Step 0: Generate Data

➢ Generate Data

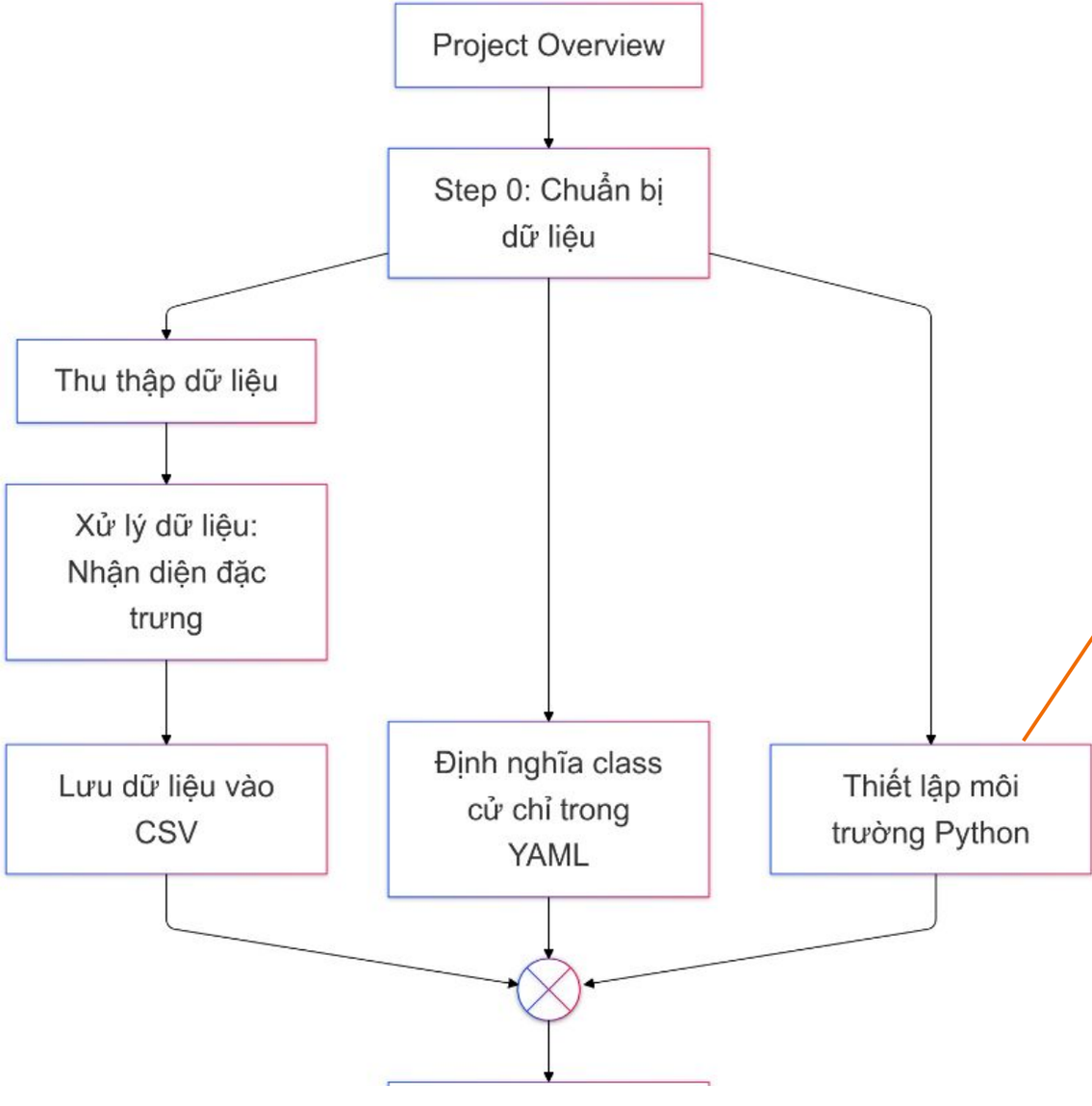



```
step0_generate_data
├── generate_landmark_data.py
├── hand_gesture.yaml
└── requirements.txt
```

```
step0_generate_data
├── data
│   ├── landmark_test.csv
│   ├── landmark_train.csv
│   └── landmark_val.csv
├── generate_landmark_data.py
├── hand_gesture.yaml
├── requirements.txt
└── sign_imgs
    ├── light1.jpg
    ├── light2.jpg
    ├── light3.jpg
    ├── turn_off.jpg
    └── turn_on.jpg
```

Step 0: Generate Data

➢ Generate Data




```
step0_generate_data
├── generate_landmark_data.py
├── hand_gesture.yaml
└── requirements.txt
```

1. Thiết Lập Môi Trường Làm Việc

1.1. Sử Dụng Conda Với Python 3.10: Chhúng ta sẽ sử dụng Conda để tạo môi trường ảo cho project với yêu cầu sử dụng Python phiên bản 3.10

- Cài đặt Anaconda hoặc Miniconda: Nếu bạn chưa có Conda trên máy tính, hãy tải và cài đặt từ trang chủ:
  - Anaconda: https://www.anaconda.com/download
  - Miniconda: https://docs.anaconda.com/miniconda/
- Tạo môi trường mới với Python 3.10:

```
conda create -n gesture_env python=3.10.0
```

- Kích hoạt môi trường:

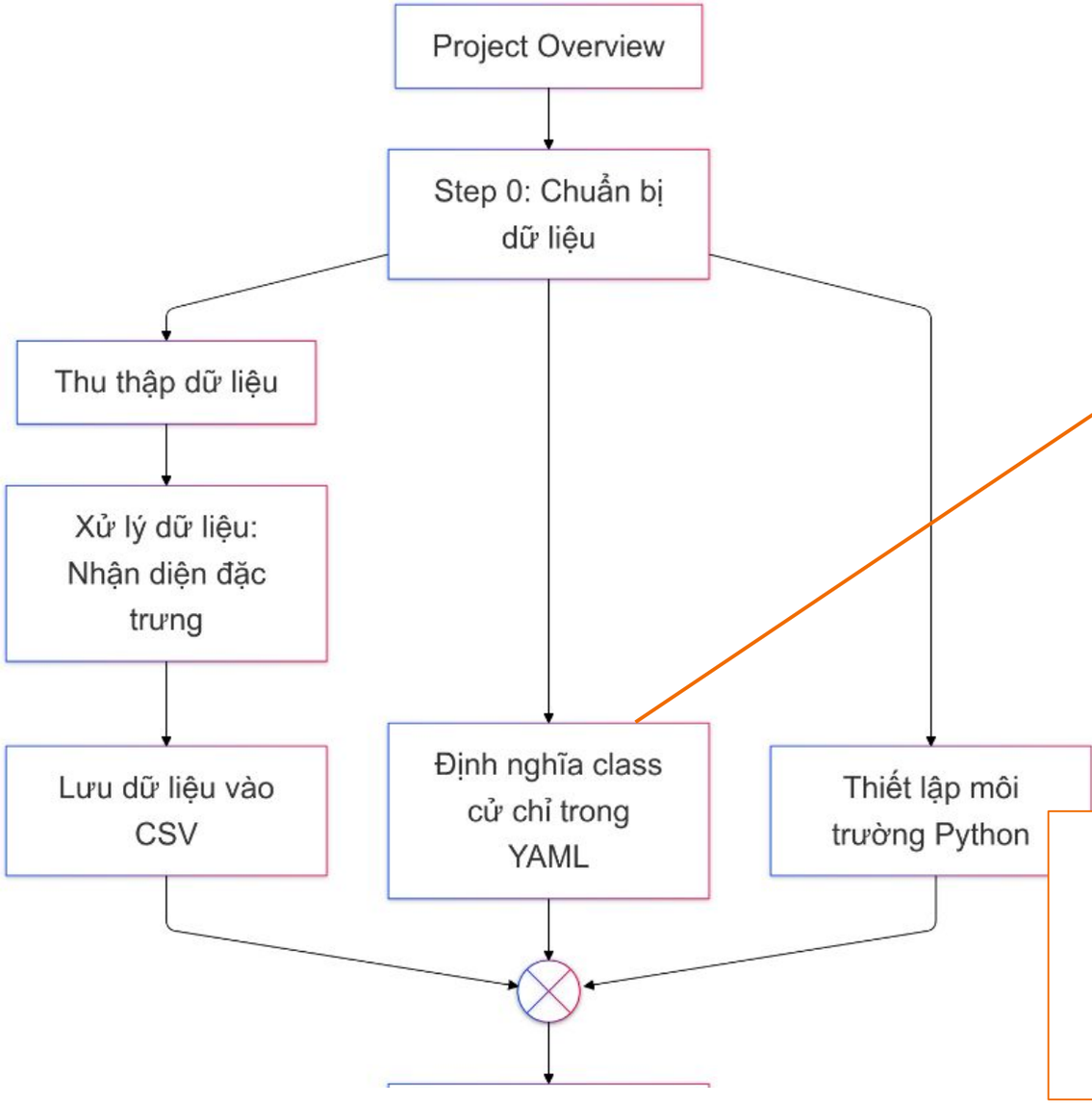
```
conda activate gesture_env
```

1.2. Cài Đặt Các Thư Viện Từ requirements.txt: Sau khi môi trường được activate, chúng ta tiến hành cài đặt các thư viện cần thiết được liệt kê trong file requirements.txt.

```
pip install -r requirements.txt
```

Step 0: Generate Data

➢ Generate Data




```
step0_generate_data
├── generate_landmark_data.py
├── hand_gesture.yaml
└── requirements.txt
```

```yaml
gestures:
  0: "turn_off"
  1: "light1" #
  2: "light2" #
  3: "light3" #
  4: "turn_on"
```

Giải thích:

- **gestures:** Đây là key, chứa danh sách các cử chỉ.
- **0, 1, 2, 3, 4:** Các số đại diện cho class của từng loại cử chỉ.
- **"turn_off", "light1", "light2", "light3", "turn_on":** Tên của các cử chỉ, đồng thời cũng là hành động điều khiển đèn tương ứng.

Step 0: Generate Data

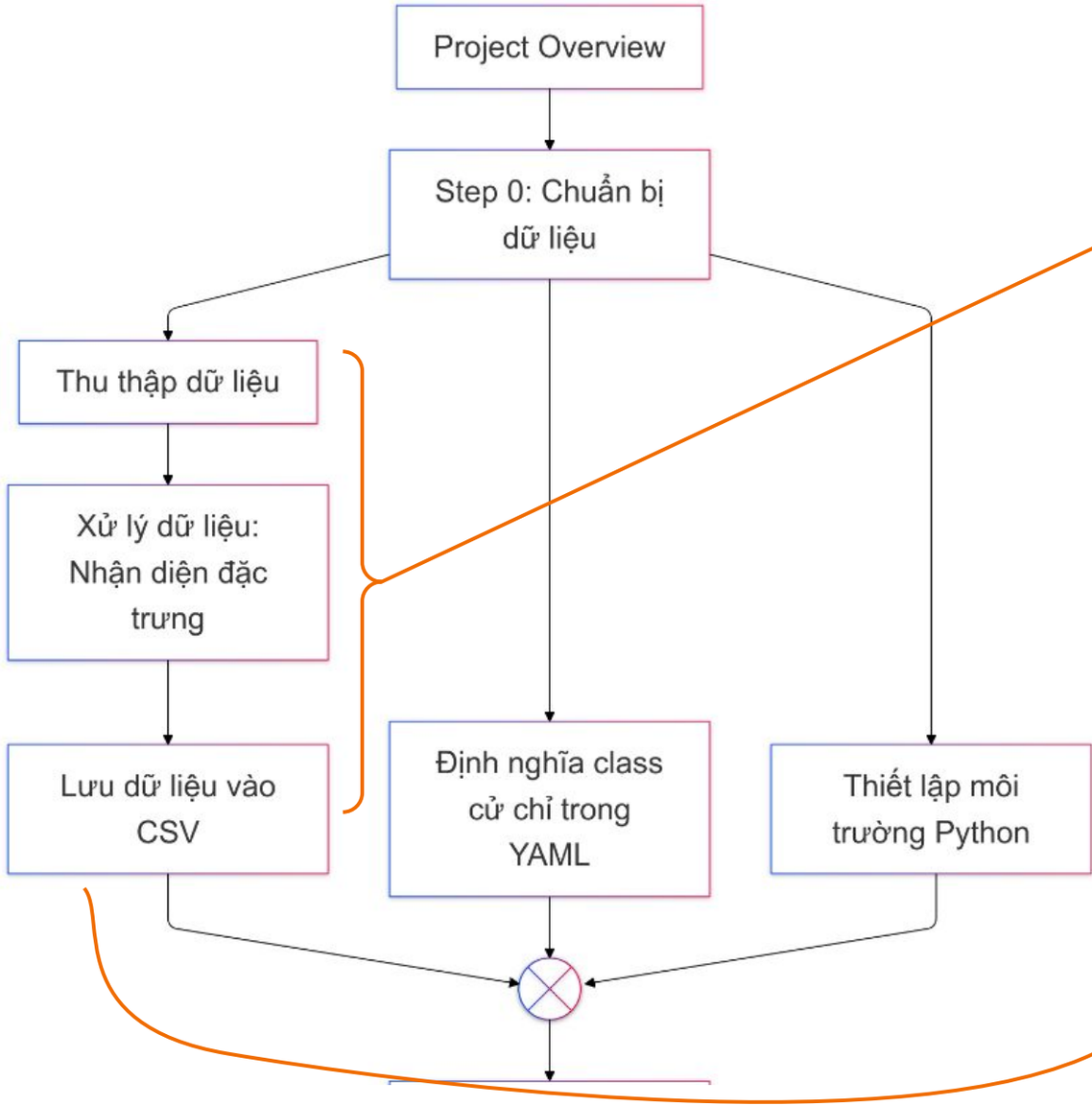
➢ Generate Data

```
gestures:
  0: "turn_off"
  1: "light1" #
  2: "light2" #
  3: "light3" #
  4: "turn_on"
```

Class	Action	Hand Gesture
0	Turn All the Lights Off	
1	Turn the Lights 1 On	
2	Turn the Lights 2 On	
3	Turn the Lights 3 On	
4	Turn All the Lights On	

Step 0: Generate Data

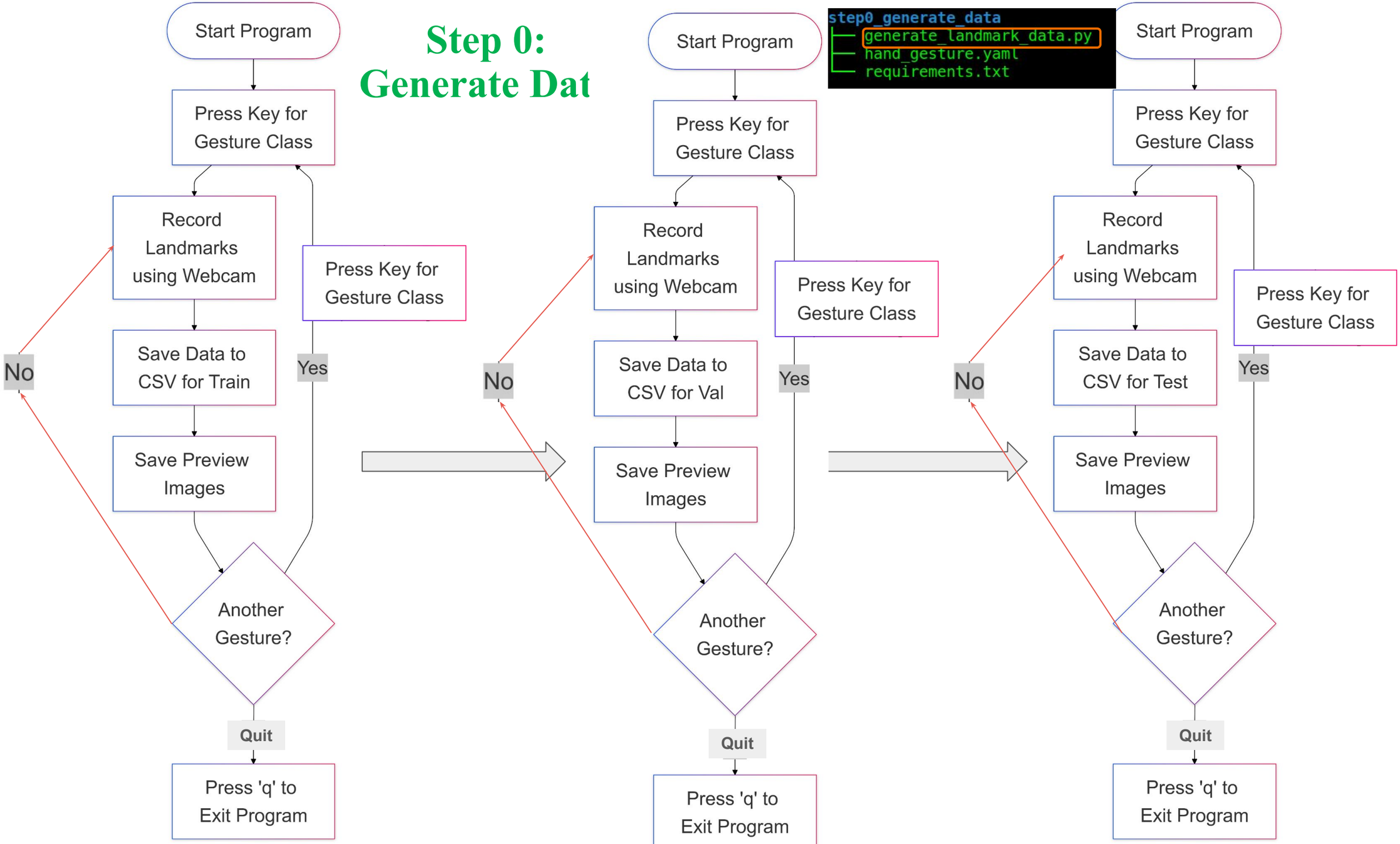
➢ Generate Data




```
step0_generate_data
├── generate_landmark_data.py
├── hand_gesture.yaml
└── requirements.txt
```

```
step0_generate_data
├── data
│   ├── landmark_test.csv
│   ├── landmark_train.csv
│   └── landmark_val.csv
├── generate_landmark_data.py
├── hand_gesture.yaml
├── requirements.txt
└── sign_imgs
    ├── light1.jpg
    ├── light2.jpg
    ├── light3.jpg
    ├── turn_off.jpg
    └── turn_on.jpg
```

Step 0: Generate Dat



Step 0: Generate Data

```
step0_generate_data
├── generate_landmark_data.py
├── hand_gesture.yaml
└── requirements.txt
```

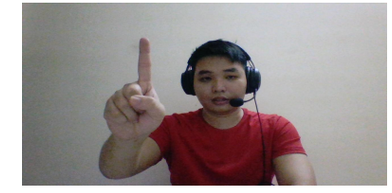
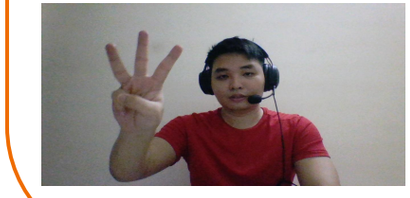
➢ Generate Data

```
step0_generate_data
├── data
│   ├── landmark_test.csv
│   ├── landmark_train.csv
│   └── landmark_val.csv
├── generate_landmark_data.py
├── hand_gesture.yaml
├── requirements.txt
└── sign_imgs
    ├── light1.jpg
    ├── light2.jpg
    ├── light3.jpg
    ├── turn_off.jpg
    └── turn_on.jpg
```

Sử dụng cho step 1

Kiểm tra các cử chỉ có chính xác




Quiz

Tại Step 0 file nào định nghĩa các class cử chỉ để điều khiển đèn?

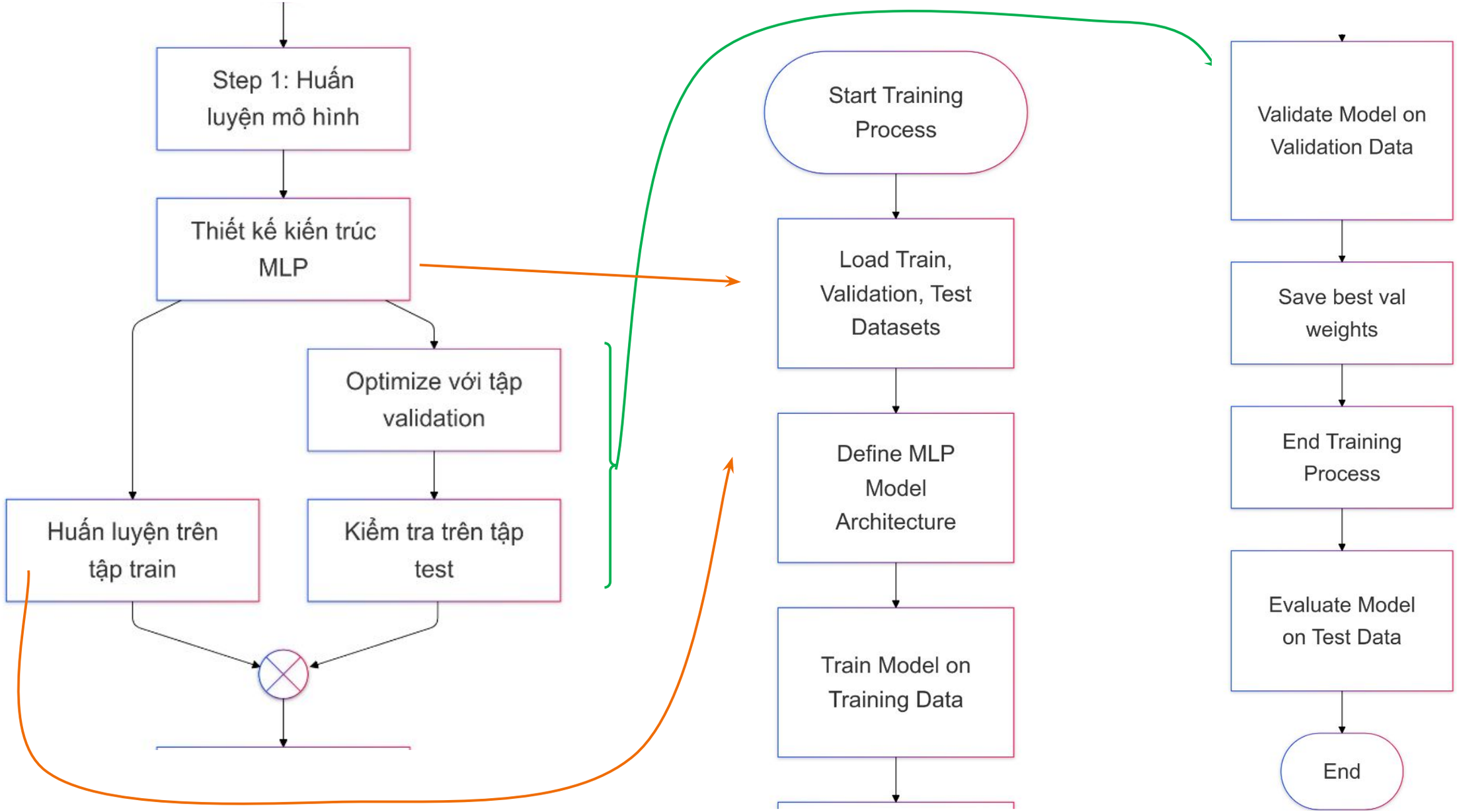
(A) . requirements.txt

(B). generate_landmark_data.py

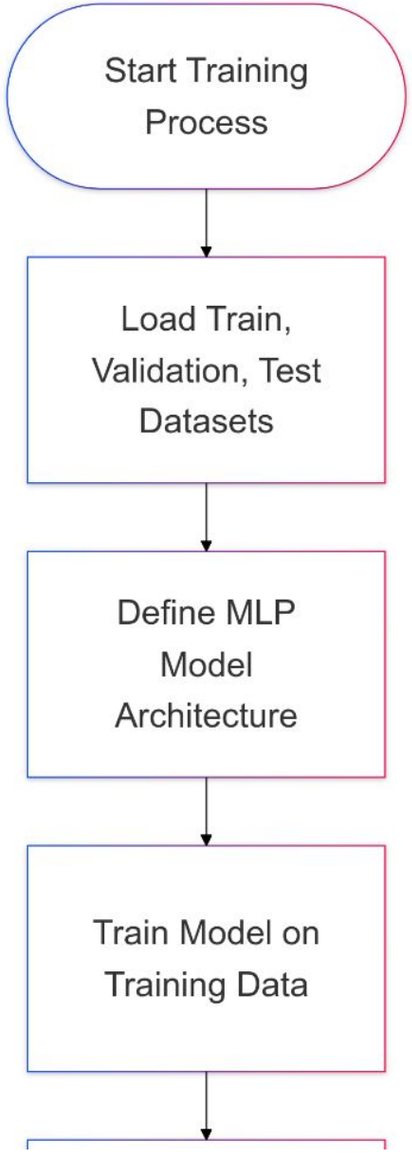
(C). hand_gesture.yaml

(D). gesture_classes.csv

Step 1: Hand Gesture

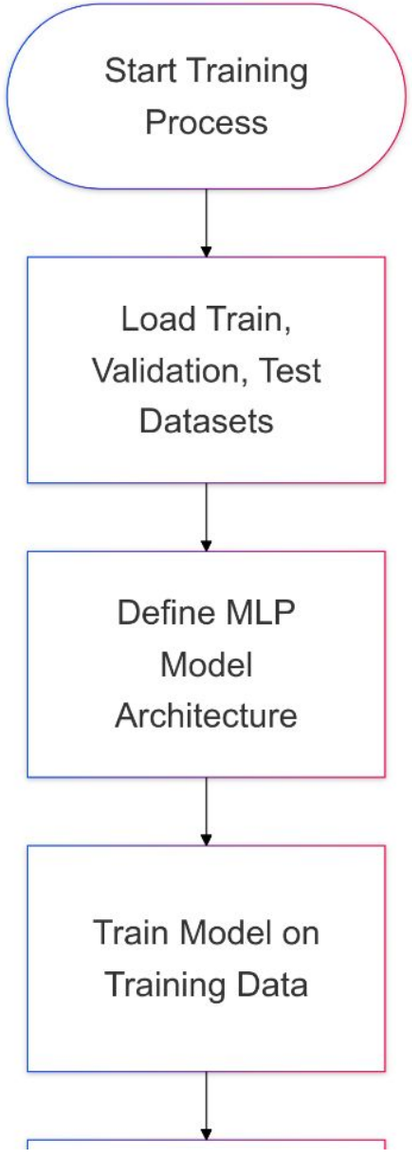


Step 1: Hand Gesture




```
DATA_FOLDER_PATH="./data/"
LIST_LABEL = label_dict_from_config_file("hand_gesture.yaml")
train_path = os.path.join(DATA_FOLDER_PATH,"landmark_train.csv")
val_path = os.path.join(DATA_FOLDER_PATH,"landmark_val.csv")
save_path = './models'
os.makedirs(save_path,exist_ok=True)

trainset = CustomImageDataset(train_path)
################## Your Code Here ################## Q3
    '''Hoàn thành code để thực hiện khởi tạo DataLoader cho trainset với
    batch_size 40 và cho phép xáo trộn
    '''
trainloader =
#####################################################

valset = CustomImageDataset(os.path.join(val_path))
val_loader = torch.utils.data.DataLoader(valset,batch_size=50, shuffle=False)
```

Step 1: Hand Gesture

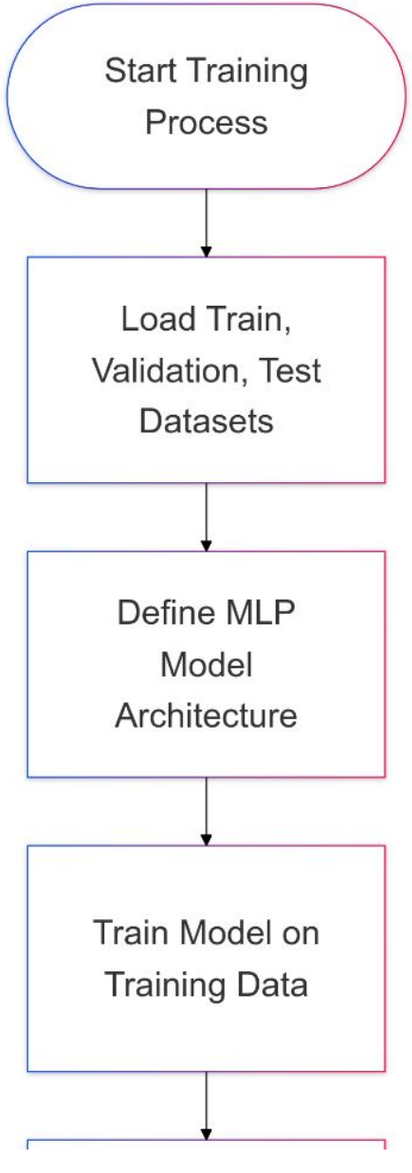


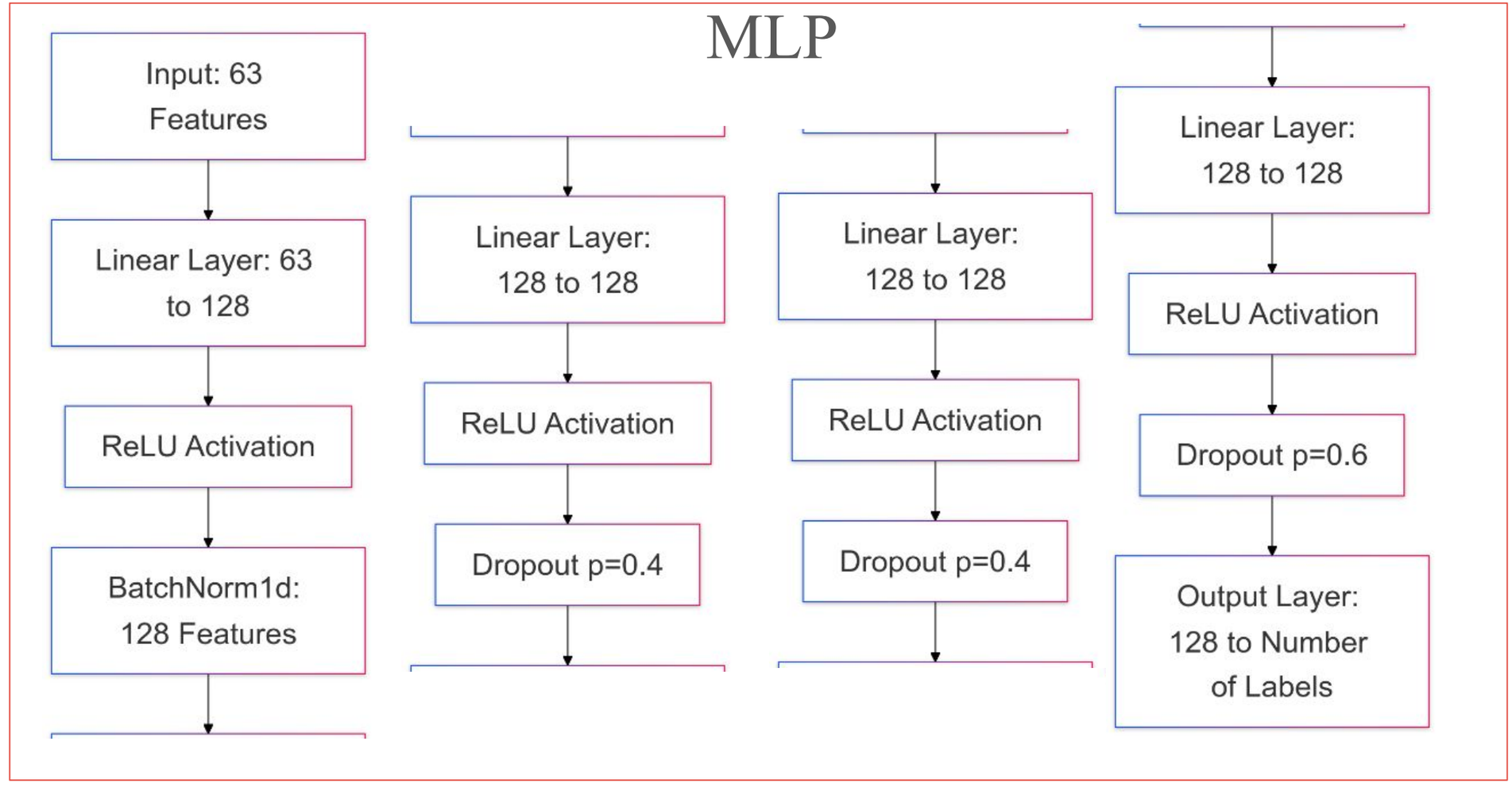

```
# Tạo folder data để các bạn upload file 3 file data được tạo ra ở step 0 vào folder data
# Các bạn upload file hand_gesture.yaml ngoài folder data
!mkdir data/
```

```
# Nếu đã chạy thành công step 0 thì các bạn không chạy cell này và chỉ upload 3 file data của các bạn vào thư mục data
# Và upload file hand_gesture.yaml bên ngoài thư mục data
# Nếu không chạy được ở step 0 thì các bạn run cell này để download data
!gdown  1gzWOtABiVmJ38usCSDe5F9gR2tECt3zu -O data/
!gdown  15lwipssmC_K82ukRfb0uVCiDH1TZ3QCf -O data/
!gdown  1nIo1_wBmkovz-u_BCsV5c1Kbz6ZqoKwq -O data/
# Download file hand_gesture.yam
!gdown  1ZteHYSgbuZu_GcUJHW8ZzoZv1DE8-oLw
```

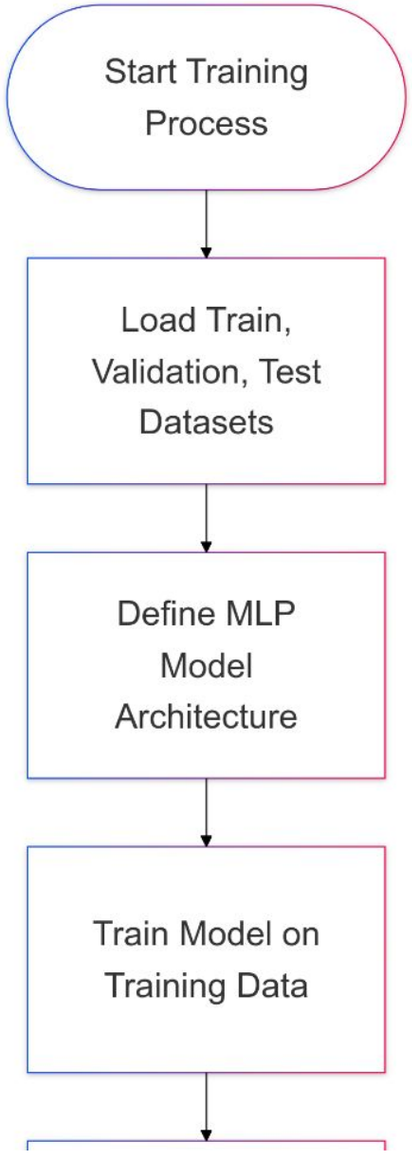
```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        list_label = label_dict_from_config_file("hand_gesture.yaml")

        self.linear_relu_stack = nn.Sequential(
            ################## Your Code Here ################## Q1
            '''Hoàn thành đoạn code để xây dựng một model gồm có 4 hidden layer,
            lần lượng input và output là (63, 128), (128, 128), (128, 128), (128, 128).
            Layer đầu tiên được theo sau bởi một Relu và Batchnorm1d.
            Layer thứ 2, 3, và 4 được theo sau bởi Relu và Dropout với rate lần lượt là 0.4, 0.4, 0.6.
            Output layer có nhiệm vụ phân loại với input là 128 và output là số lượng class cần chỉ
            '''
            ####################################################
        )
```

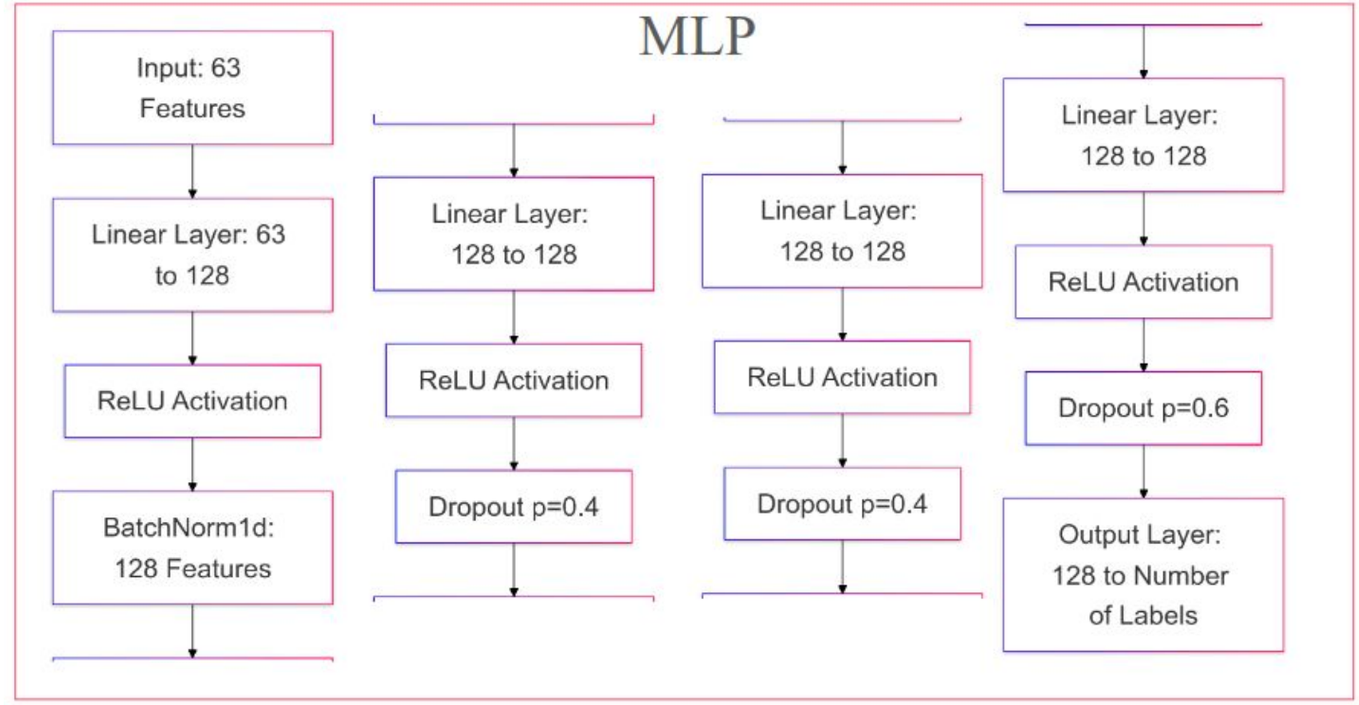
Step 1: Hand Gesture





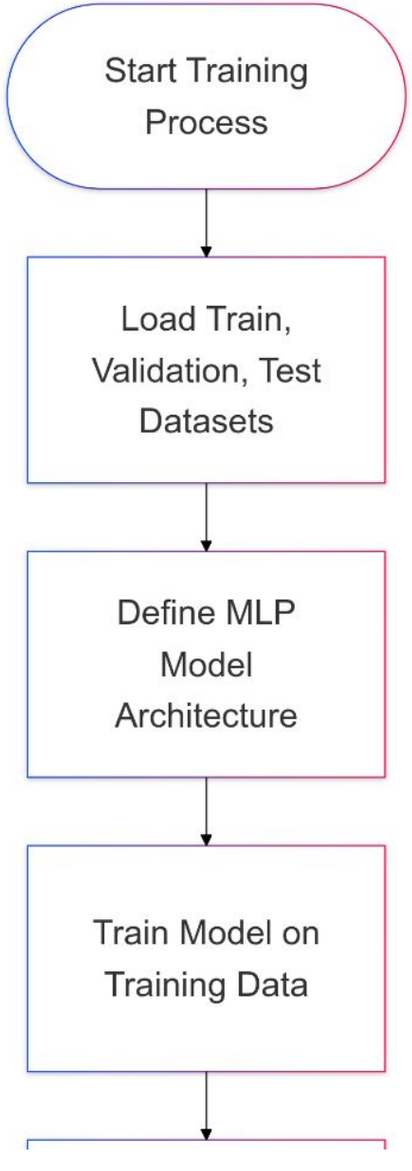
Step 1: Hand Gesture





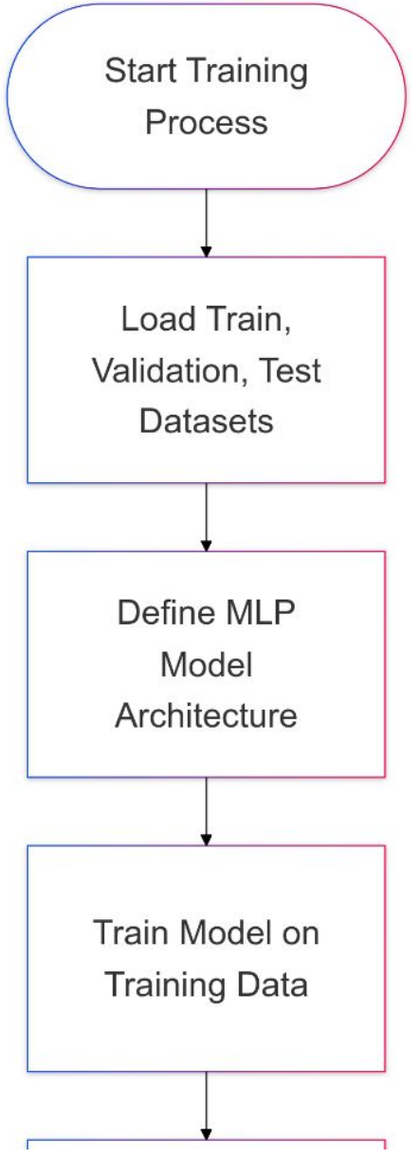

```python
def forward(self, x):
    ################## Your Code Here ################## Q2
    ''' Hoàn thành code để thực hiện forward dự đoán cử chỉ với input x.
    Thực hiện flatten x
    Pass x vừa flatten vào linear_relu_stack
    Return  logits (outputs từ layer cuối cùng)
    '''
    ###################################################
```

Step 1: Hand Gesture



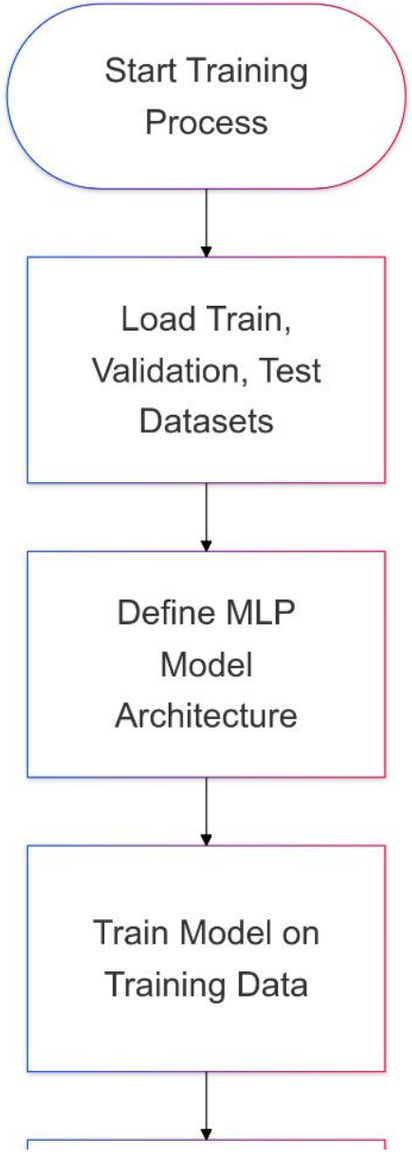

```
19  ################## Your Code Here ################## Q8
20  '''Hoàn thành code để thực hiện khởi tạo NeuralNetwork model đã xây dựng ở trên,
21  khởi tạo hàm loss sử dụng CrossEntropyLoss và khởi tạo early stopper với patience
22  là 30 và min_delta là 0.01
23  '''
24  model =
25  loss_function =
26  early_stopper =
27  ####################################################
28
29  ################## Your Code Here ################## Q4
30  '''Hoàn thành code để thực hiện cấu hình Adam optimizer cho các tham số của
31  model với tốc độ học là 0.0001
32  '''
33  optimizer =
34  ####################################################
35
36
37  model, best_model_path = train(trainloader, val_loader, model, loss_function, early_stopper, optimizer)
```

Step 1: Hand Gesture



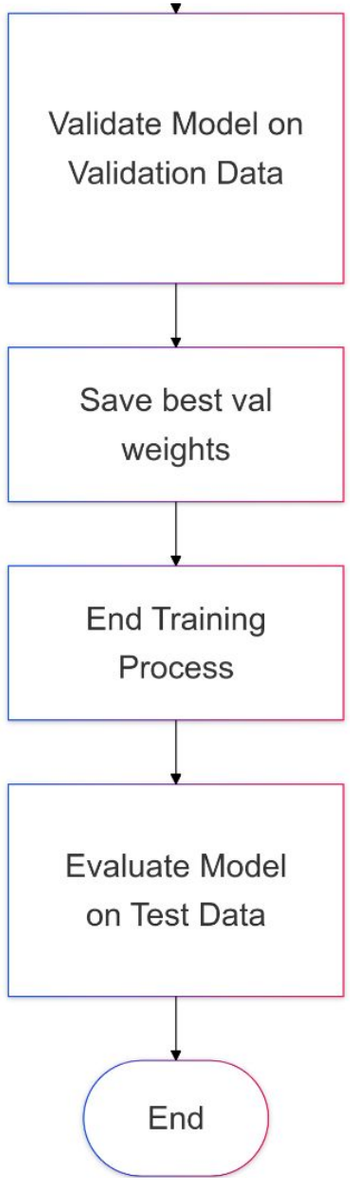

```python
def train(trainloader, val_loader, model, loss_function, early_stopper, optimizer):
    # add auroc score
    best_vloss = 1_000_000
    timestamp = datetime.now().strftime('%d-%m %H:%M')
    for epoch in range(300):
        #training step
        model.train(True)
        running_loss = 0.0
        acc_train = Accuracy(num_classes=len(LIST_LABEL), task='MULTICLASS')
        for batch_number,data in enumerate(trainloader):
            inputs,labels = data

            ################## Your Code Here ################## Q9
            ''' Hoàn thành code để thực hiện reset gradients và dự đoán class cư
            chỉ của inputs
            '''

            preds =
            ####################################################
```

Step 1: Hand Gesture



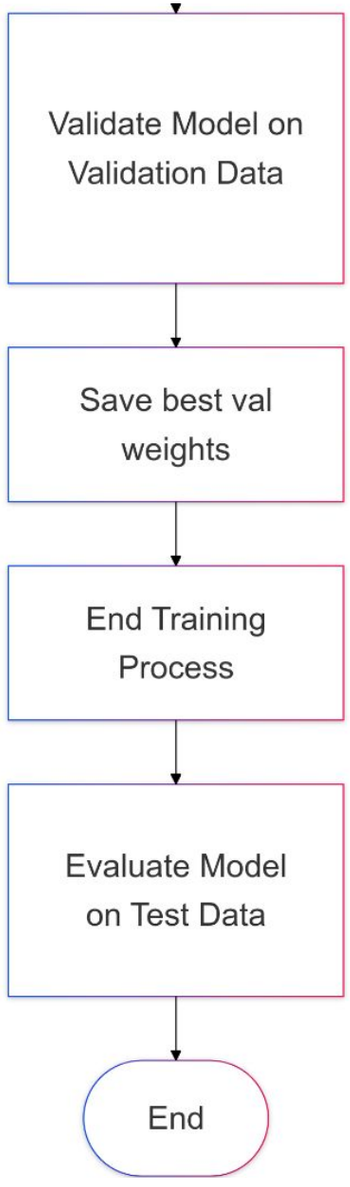

```
21            ################## Your Code Here ################## Q10
22            ''' Hoàn thành code để thực hiện tính loss dựa vào kết quả dự đoán
23            và labels, sau đó thực hiện backwward và update parameters thông qua
24            optimizer
25            '''
26            loss =
27
28
29            ####################################################
30
31            acc_train.update(model.predict_with_known_class(inputs), labels)
32            running_loss += loss.item()
33        avg_loss = running_loss / len(trainloader)
```

Step 1: Hand Gesture



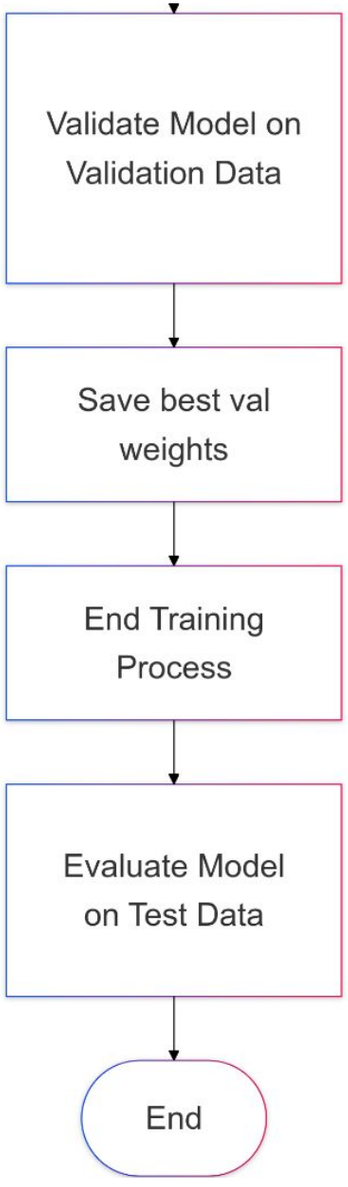

```
46          # Log the running loss averaged per batch
47          # for both training and validation
48          print(f"Epoch {epoch}: ")
49          print(f"Accuracy train:{acc_train.compute().item()}, val:{acc_val.compute().item()}")
50          avg_vloss = running_vloss / len(val_loader)
51          print('LOSS train {} valid {}'.format(avg_loss, avg_vloss))
52          print('Training vs. Validation Loss',
53                            { 'Training' : avg_loss, 'Validation' : avg_vloss },
54                            epoch + 1)
55          print('Training vs. Validation accuracy',
56                            { 'Training' : acc_train.compute().item()
57                            , 'Validation' : acc_val.compute().item() },
58                            epoch + 1)
59
60          # Track best performance, and save the model's state
61          if avg_vloss < best_vloss:
62              best_vloss = avg_vloss
63              best_model_path = f'./{save_path}/model_{timestamp}_{model.__class__.__name__}_best'
64              torch.save(model.state_dict(), best_model_path)
```

Step 1: Hand Gesture



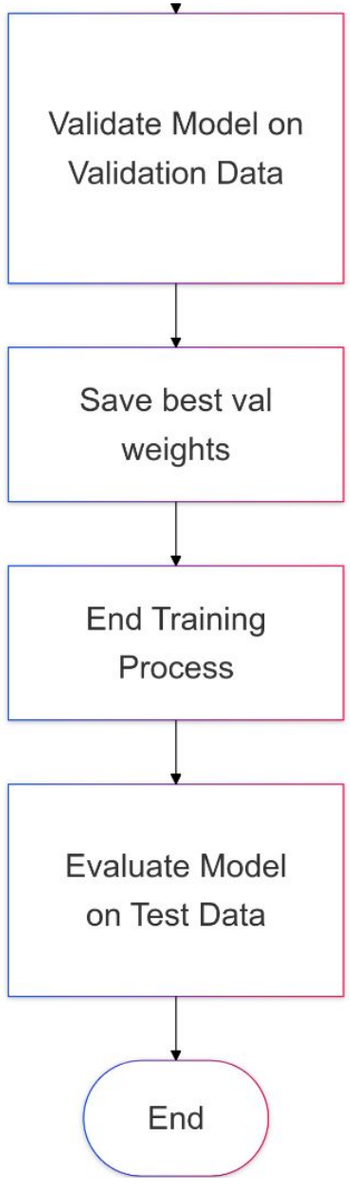

```
60          # Track best performance, and save the model's state
61          if avg_vloss < best_vloss:
62              best_vloss = avg_vloss
63              best_model_path = f'./{save_path}/model_{timestamp}_{model.__class__.__name__}_best'
64              torch.save(model.state_dict(), best_model_path)
65
66          if early_stopper.early_stop(avg_vloss):
67              ################## Your Code Here ################## Q5
68              ''' Hoàn thành đoạn code bên dưới để print ra epoch hiện tại và
69              minimum watched metric và thoát loop
70              '''
71
72              ####################################################
73
74
75
76      model_path = f'./{save_path}/model_{timestamp}_{model.__class__.__name__}_last'
77      torch.save(model.state_dict(), model_path)
```

Step 1: Hand Gesture




```python
list_label = label_dict_from_config_file("hand_gesture.yaml")
DATA_FOLDER_PATH="./data/"
testset = CustomImageDataset(os.path.join(DATA_FOLDER_PATH,"landmark_test.csv"))

# Test DataLoader instantiation
################## Your Code Here ################## Q6
''' Hoàn thành code bên dưới để khởi tạo DataLoader cho testset with batch size
20, không cho phép shuffle
'''
test_loader =
#####################################################
```

Step 1: Hand Gesture




```python
network = NeuralNetwork()
network.load_state_dict(torch.load(best_model_path, weights_only=False))

network.eval()
acc_test = Accuracy(num_classes=len(list_label), task='MULTICLASS')
for i, test_data in enumerate(test_loader):
    test_input, test_label = test_data
    ################## Your Code Here ################## Q7
    '''Hoàn thành code bên dưới để predict class của cử chỉ và update accuracy
    với kết quả predict và true labels
    '''
    preds =

    ####################################################

print(network.__class__.__name__)
print(f"Accuracy of model:{acc_test.compute().item()}")
print("================================================================")
```

```
NeuralNetwork
Accuracy of model:0.9750000238418579
================================================================
```

Quiz

Model được sử dụng đề phân loại cử chỉ có bao nhiêu hidden layer

(A) . 2

(B). 3

(C). 4

(D). 5

Step 2: Control Lights (Simulation + Reality)

➢ Simulation


```
step2_control_lights/
├── controller.py
├── detect_simulation.py
└── hand_gesture.yaml
```

```
step2_control_lights/
├── controller.py
├── detect_simulation.py
├── hand_gesture.yaml
└── models
    └── model_03-11 04_37_NeuralNetwork_best
```

Best model từ step 1

Step 2: Control Lights (Simulation + Reality)

➢ Simulation


```
step2_control_lights/
├── controller.py
├── detect_simulation.py
├── hand_gesture.yaml
└── models
    └── model_03-11_04_37_NeuralNetwork_best
```

```python
if __name__ == "__main__":
    model_path = "./models/model_02-11_15_19_NeuralNetwork_best"
    light = LightGesture(model_path, device=False)
    light.run()
```

Không chạy thiết bị chỉ mô phỏng

Step 2: Control Lights (Simulation + Reality)

➢ Simulation


```
step2_control_lights/
├── controller.py
├── detect_simulation.py
├── hand_gesture.yaml
└── models
    └── model_03-11_04_37_NeuralNetwork_best
```

Đường dẫn MLP model

```
if __name__ == "__main__":
    model_path = "./models/model_02-11_15_19_NeuralNetwork_best"
    light = LightGesture(model_path, device=False)
    light.run()
```

Step 2: Control Lights (Simulation + Reality)

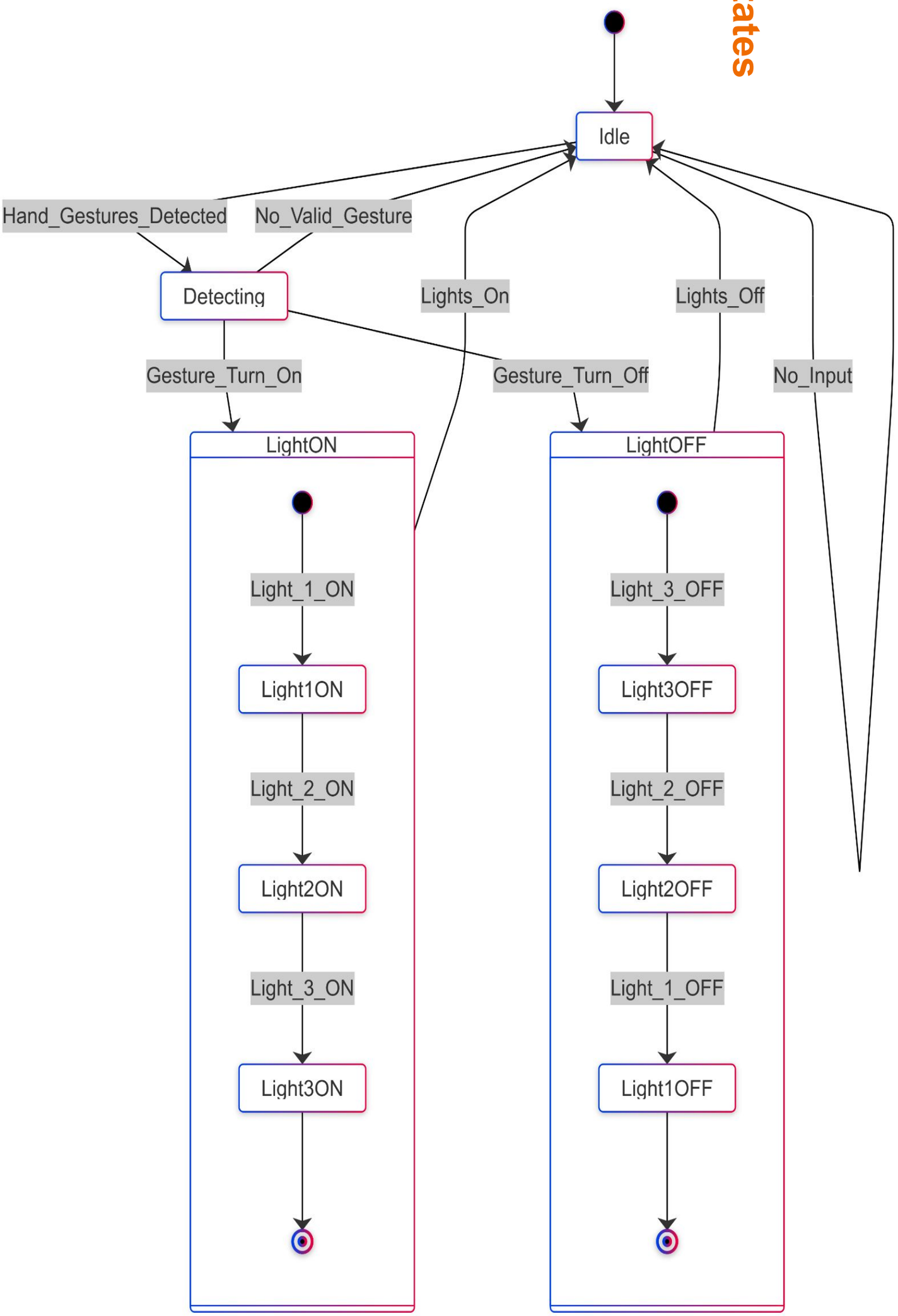
➢ Reality


```
step2_control_lights/
├── controller.py
├── detect_simulation.py
├── hand_gesture.yaml
└── models
    └── model_03-11 04_37_NeuralNetwork_best
```

```
if __name__ == "__main__":
    model_path = "./models/model_02-11_15_19_NeuralNetwork_best"
    light = LightGesture(model_path, device=True)
    light.run()
```

Chạy với thiết bị

States


Idle
Hand_Gestures_Detected
No_Valid_Gesture
Detecting
Lights_On
Lights_Off
Gesture_Turn_On
Gesture_Turn_Off
No_Input
LightON
Light_1_ON
Light1ON
Light_2_ON
Light2ON
Light_3_ON
Light3ON
LightOFF
Light_3_OFF
Light3OFF
Light_2_OFF
Light2OFF
Light_1_OFF
Light1OFF

Quiz

Ở step 2 khi light 1 ở trạng thái ON và nhận lệnh bật light 2 thì trạng thái của light 1 sẽ thay đổi như thế nào

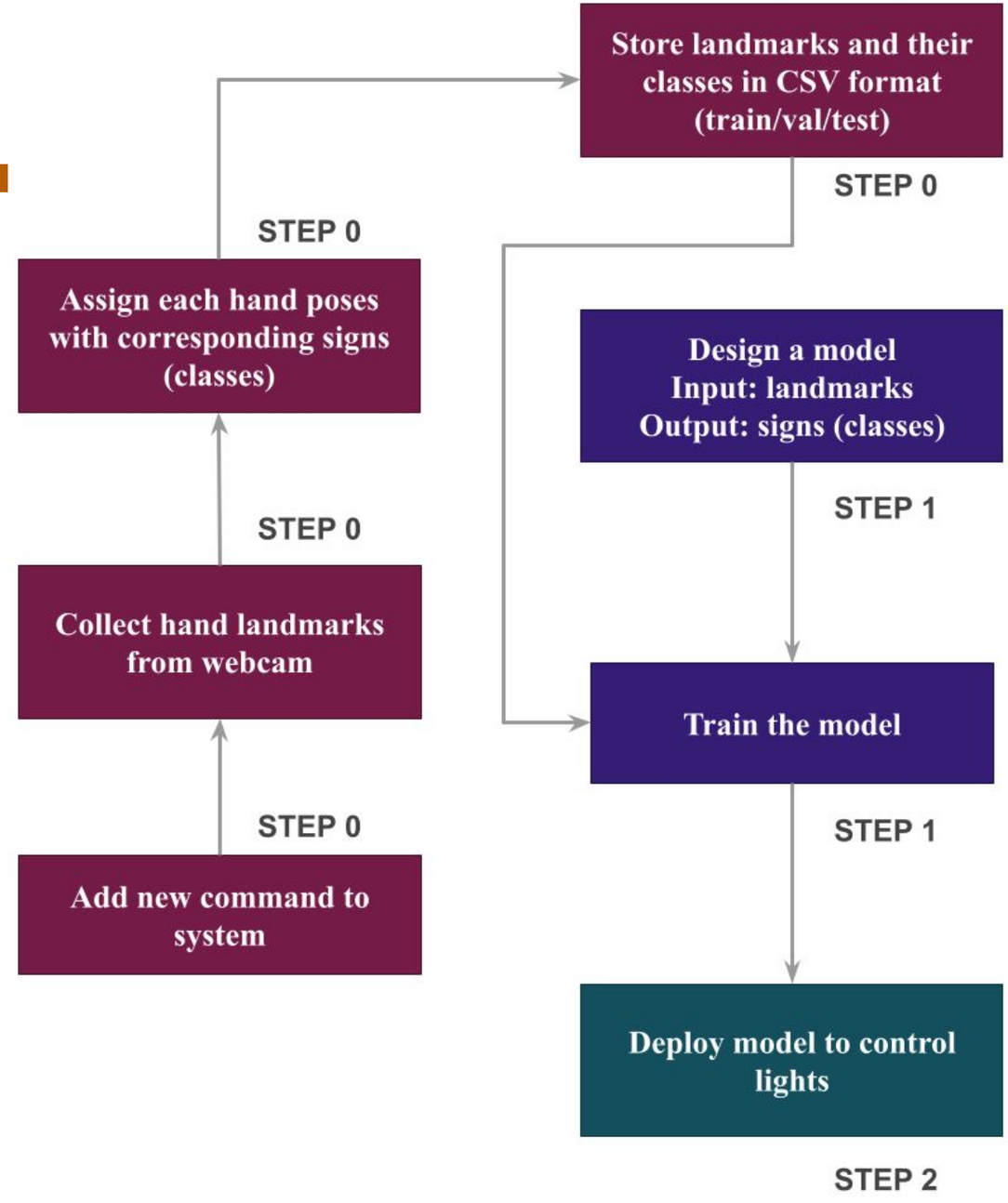
(A) . Không thay đổi

(B). Chuyển sang trạng thái ON

(C). Chuyển sang trạng thái OFF

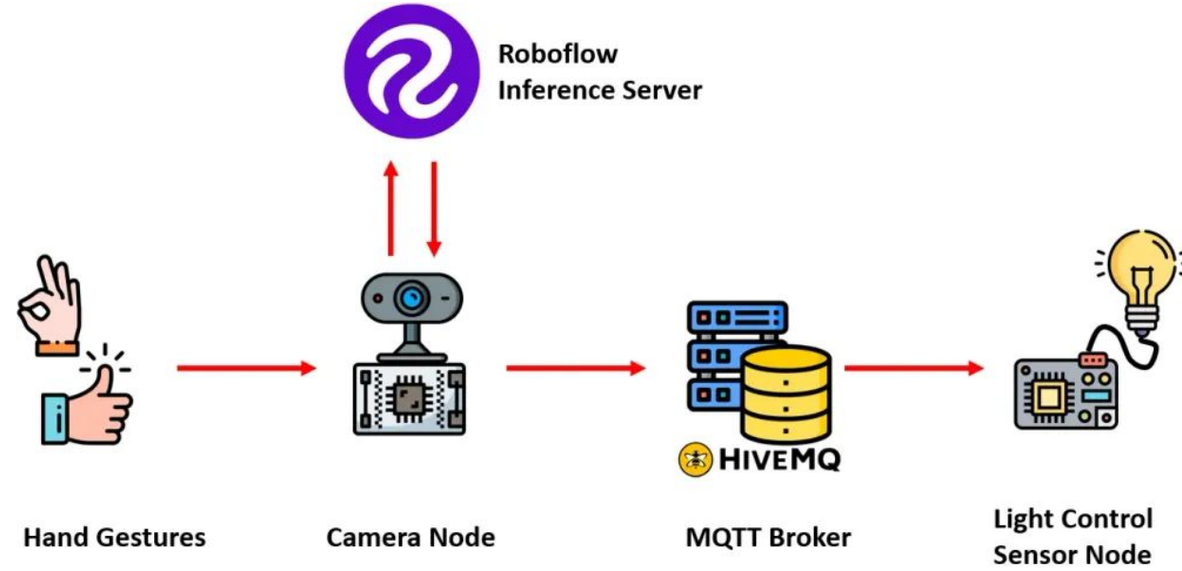
(D). Không xác định

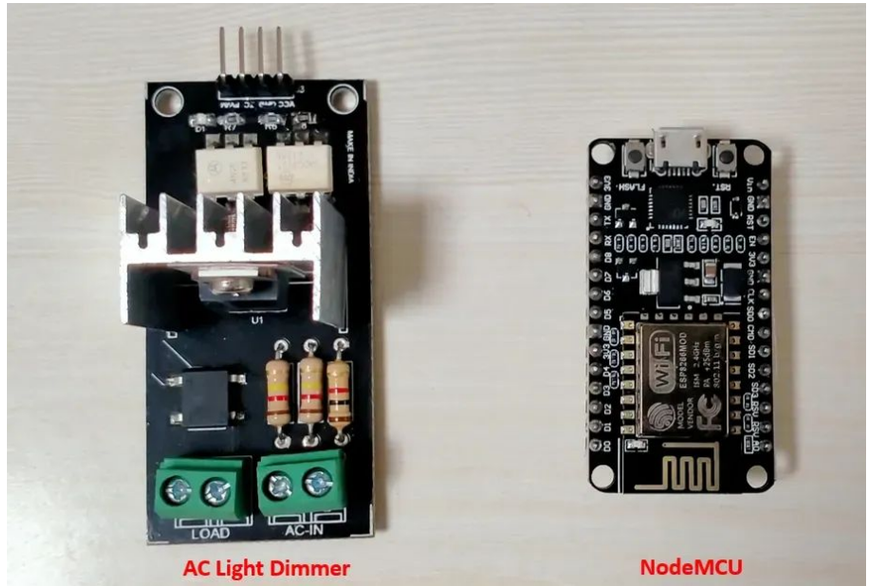
➢ Summary

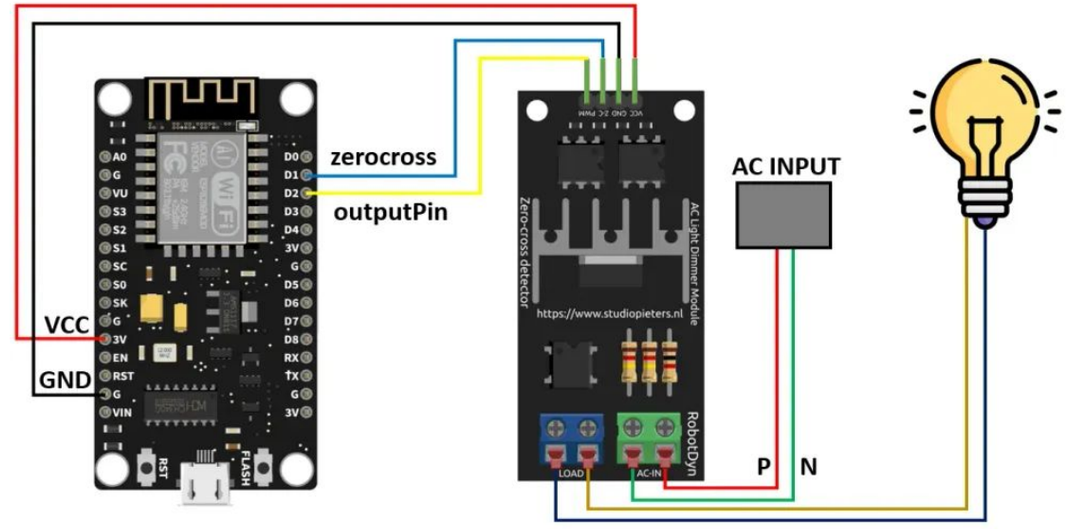

Store landmarks and their classes in CSV format (train/val/test)
STEP 0
Assign each hand poses with corresponding signs (classes)
STEP 0
Collect hand landmarks from webcam
STEP 0
Add new command to system
Design a model
Input: landmarks
Output: signs (classes)
STEP 1
Train the model
STEP 1
Deploy model to control lights
STEP 2


QUIZ
QUIZ
QUIZ

AI Application in Internet of Things (IoT)

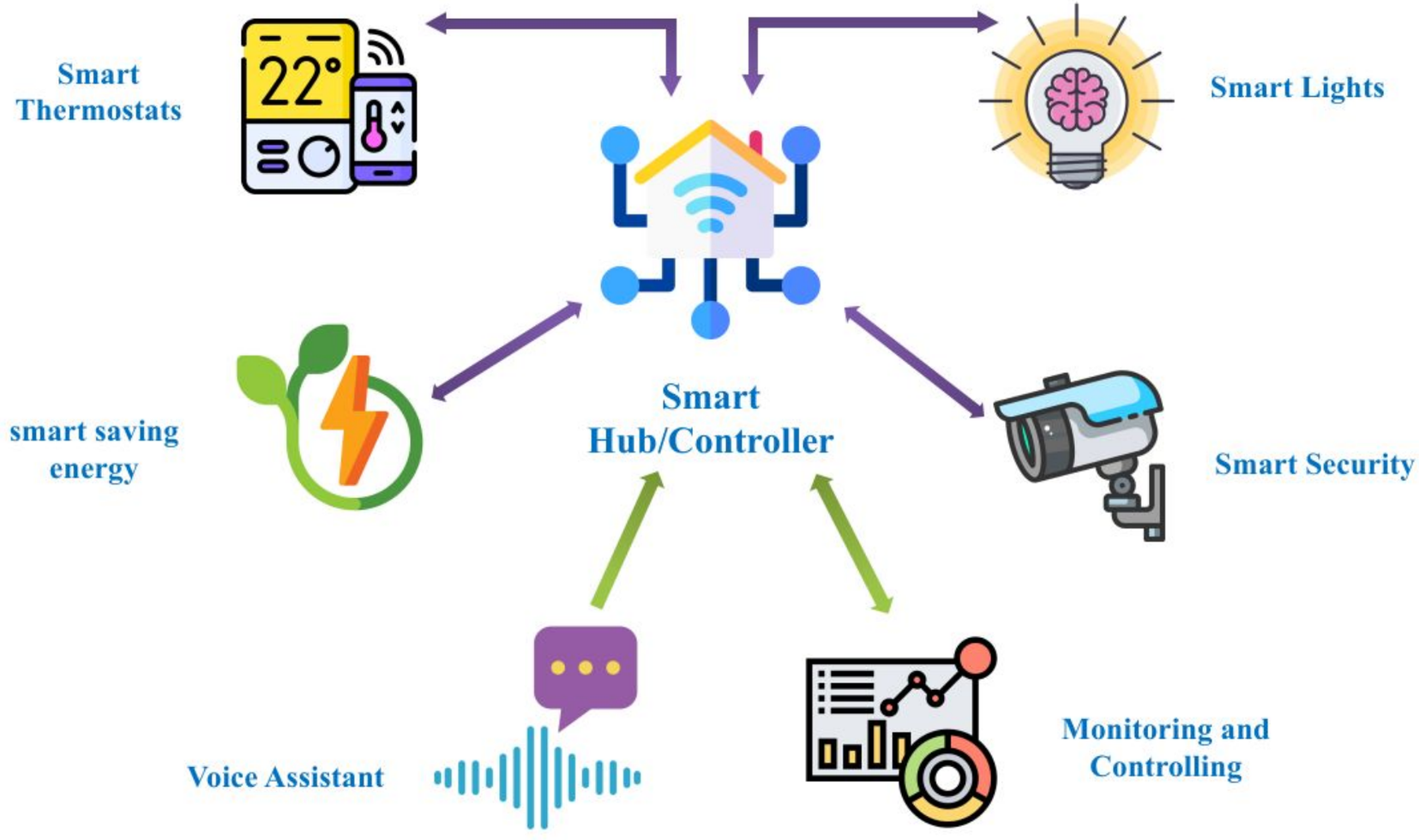






AI Application in Internet of Things (IoT)

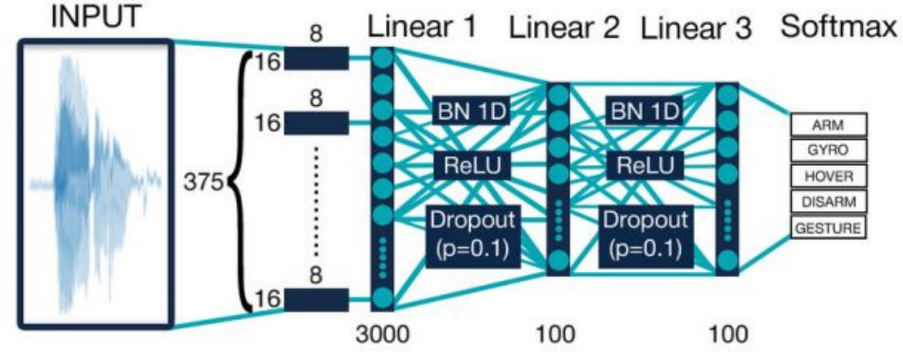
Smart Home Automation

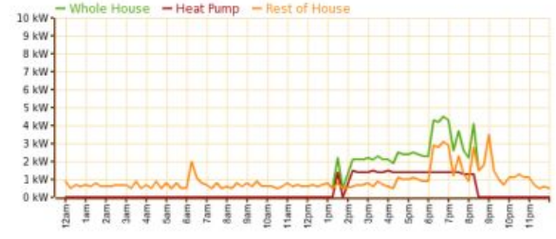


AI Application in Internet of Things (IoT)

Smart Home Automation Examples

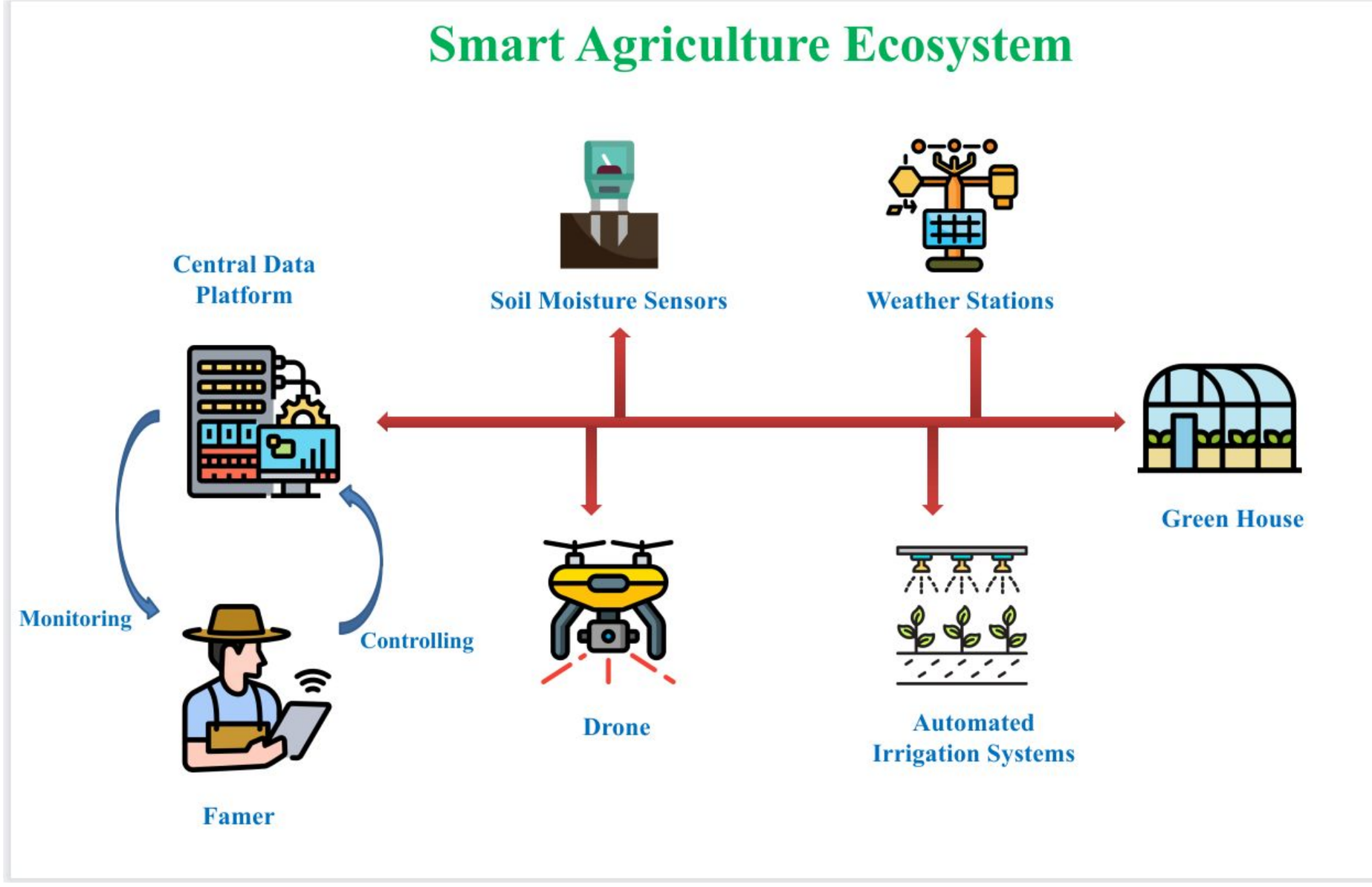
Voice Assistant





Monitoring & Controlling

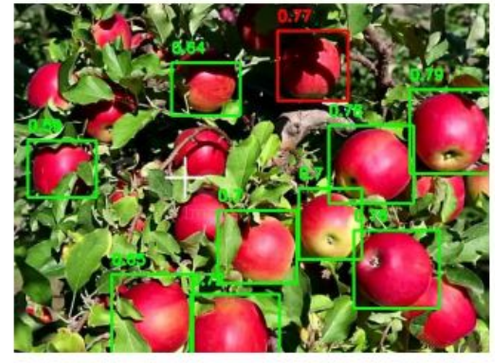
AI Application in Internet of Things (IoT)



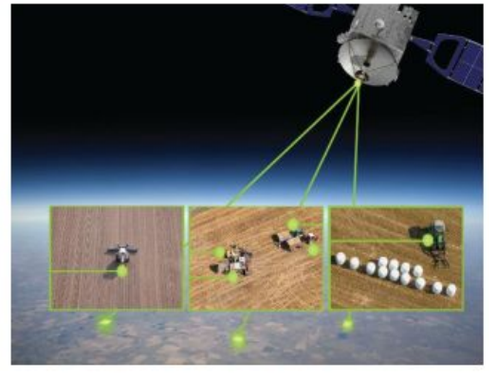
AI Application in Internet of Things (IoT)

Smart Agriculture Ecosystem Examples

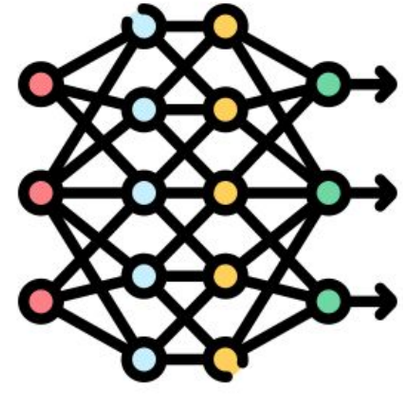
Drone Video Real-time



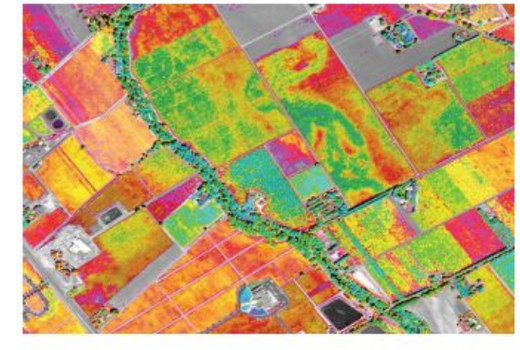
Object Detection



Datellite Images



Deep Learning Model

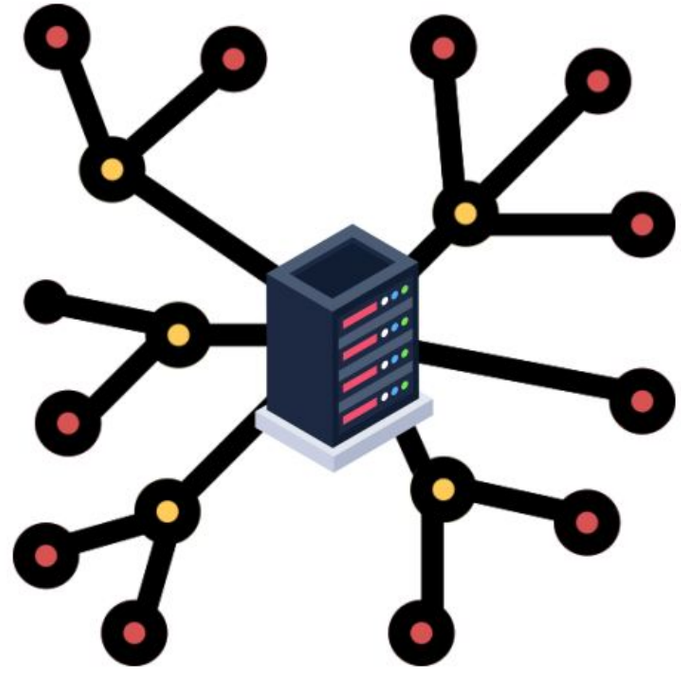


Yield Estimation and Forecasting

AI Application in Internet of Things (IoT)

AI-Enhanced Retail Environment

Smart Shelves



Social Robots

Digital Kiosks

Facial Recognition and
Computer Vision

Self-Checkout
Station

AI Application in Internet of Things (IoT)

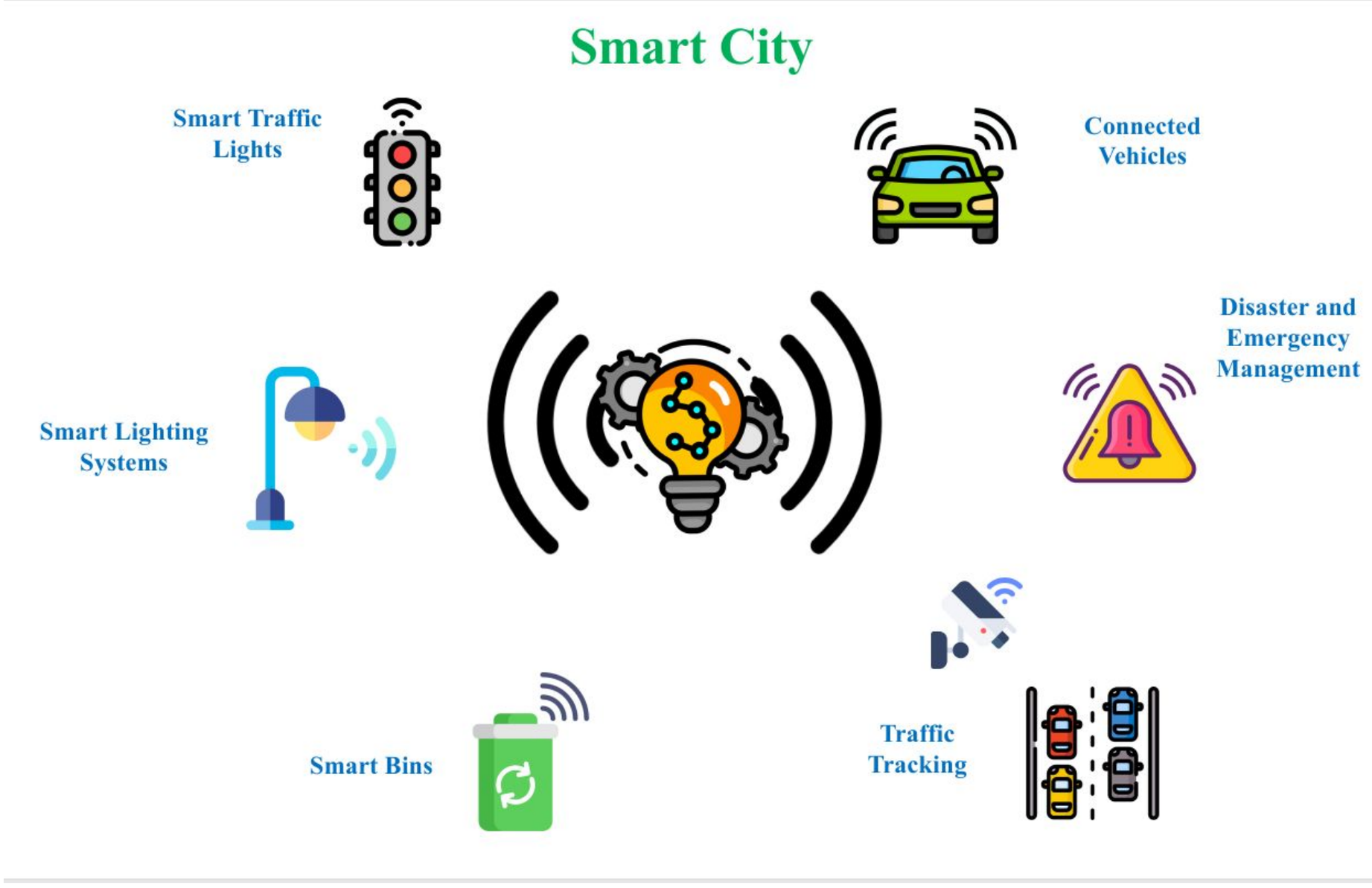
AI-Enhanced Retail Environment Examples

Self-Checkout Station

Facial Recognition and Computer Vision

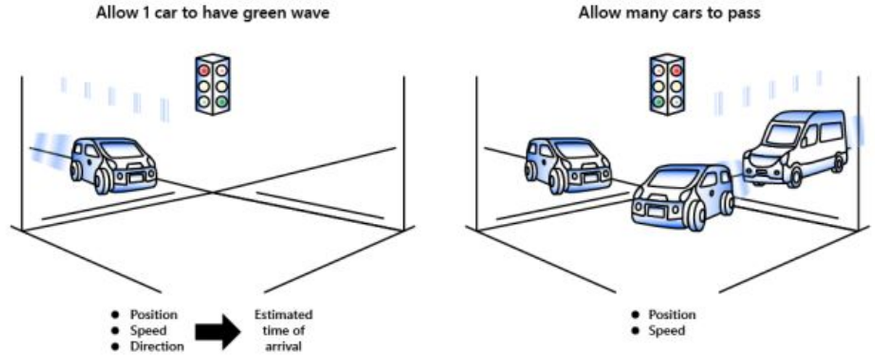
Smart Shelves

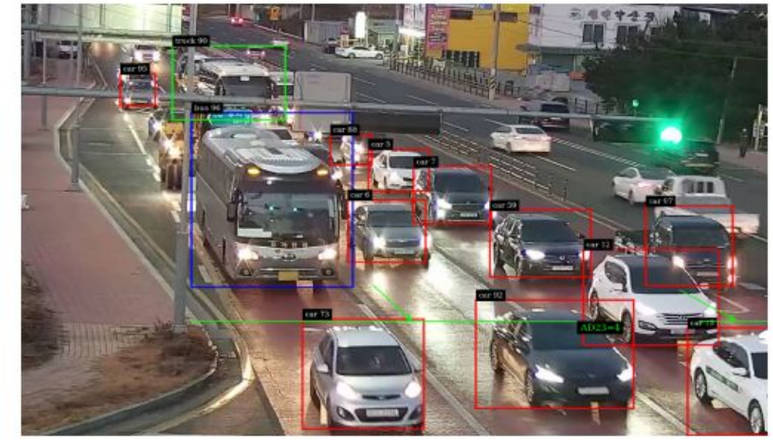
AI Application in Internet of Things (IoT)



AI Application in Internet of Things (IoT)

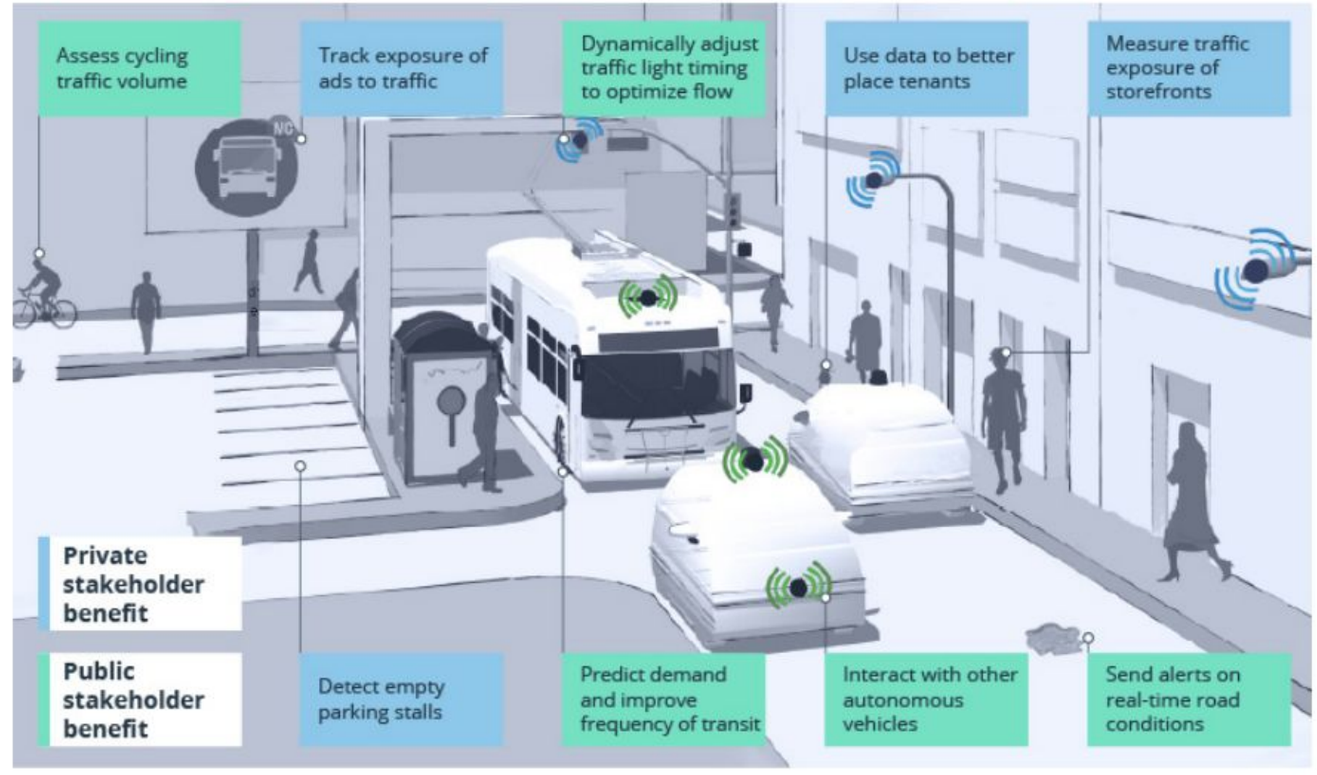
Smart City Examples





AI Application in Internet of Things (IoT)

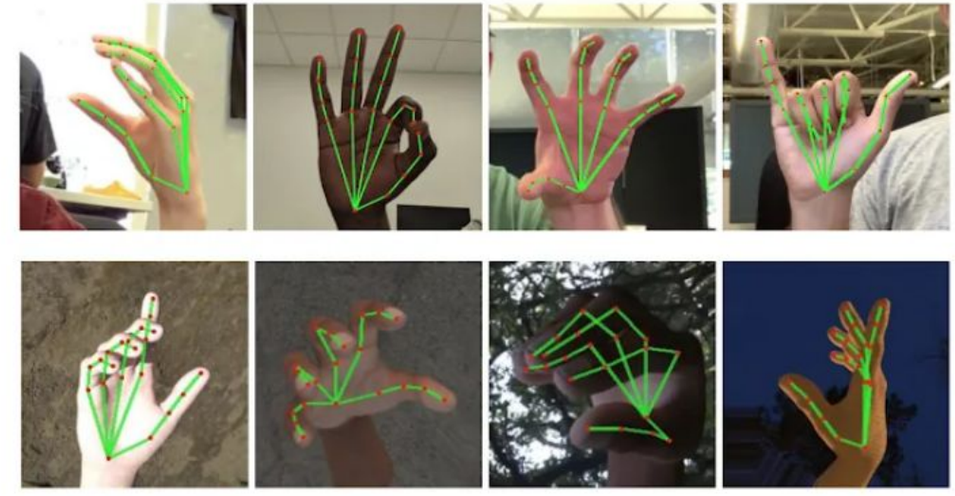
Smart City Examples

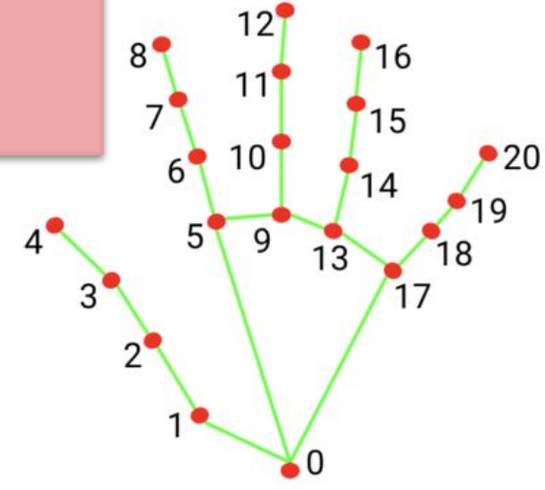


AI-Driven Public Transportation Optimization in Smart City

Objectives

- Develop a Natural Interaction System
- Set Up the Project Environment
- Collect and Prepare Gesture Data
- Build and Train an MLP Model
- Evaluate Model Performance
- Deploy Real-Time Gesture Recognition





0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP