Project Title: Mouse control by Hand gestures Using CNN

```
#code
import mediapipe as mp
import cv2
import mouse
import numpy as np
import tkinter as tk
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import transforms
import torch.nn.functional as F

root = tk.Tk()
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
ssize = (screen_height, screen_width)

class SimpleCNN(nn.Module):
    def init(self):
        super(SimpleCNN, self).init()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1,
padding=1)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1,
padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2,
padding=0)
        self.fc1 = nn.Linear(32 * 120 * 160, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 32 * 120 * 160)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = SimpleCNN()

def frame_pos2screen_pos(frame_size=(480, 640),
screen_size=(768, 1366), frame_pos=None):
    x, y = screen_size[1] / frame_size[0], screen_size[0] /
frame_size[1]
```

```
    screen_pos = [frame_pos[0] * x, frame_pos[1] * y]
    return screen_pos

def euclidean(pt1, pt2):
    d = np.sqrt((pt1[0] - pt2[0]) ** 2 + (pt1[1] - pt2[1]) ** 2)
    return d

cam = cv2.VideoCapture(0)

if not cam.isOpened():
    print("Error: Camera could not be opened.")
    exit()

fsize = (520, 720)
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands

left, top, right, bottom = (200, 100, 500, 400)
events = ["sclick", "dclick", "rclick", "drag", "release",
"scroll_up", "scroll_down"]
check_every = 15
check_cnt = 0
```

```
last_event = None


out = cv2.VideoWriter("out.avi",
cv2.VideoWriter_fourcc(*'XVID'), 30, (fsize[1], fsize[0]))


with mp_hands.Hands(static_image_mode=True,
max_num_hands=1, min_detection_confidence=0.5) as
hands:
    try:
        print("Starting video capture...")
        while True:
            ret, frame = cam.read()
            if not ret:
                print("Failed to capture frame. Exiting...")
                break


            frame = cv2.flip(frame, 1)
            frame = cv2.resize(frame, (fsize[1], fsize[0]))


            h, w, _ = frame.shape
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0,
255), 1)
```

```
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        res = hands.process(rgb)

        if res.multi_hand_landmarks:
            for hand_landmarks in res.multi_hand_landmarks:
                index_tip =
mp_drawing._normalized_to_pixel_coordinates(

hand_landmarks.landmark[mp_hands.HandLandmark.INDEX
_FINGER_TIP].x,

hand_landmarks.landmark[mp_hands.HandLandmark.INDEX
_FINGER_TIP].y,
                    w, h
                )

                index_dip =
mp_drawing._normalized_to_pixel_coordinates(

hand_landmarks.landmark[mp_hands.HandLandmark.INDEX
_FINGER_DIP].x,

hand_landmarks.landmark[mp_hands.HandLandmark.INDEX
_FINGER_DIP].y,
                    w, h
                )

                index_pip = 
np.array(mp_drawing._normalized_to_pixel_coordinates(

hand_landmarks.landmark[mp_hands.HandLandmark.INDEX
_FINGER_PIP].x,

hand_landmarks.landmark[mp_hands.HandLandmark.INDEX
_FINGER_PIP].y,
                    w, h
                ))

                thumb_tip = 
mp_drawing._normalized_to_pixel_coordinates(

hand_landmarks.landmark[mp_hands.HandLandmark.THUM
B_TIP].x,

hand_landmarks.landmark[mp_hands.HandLandmark.THUM
B_TIP].y,
                    w, h
                )
```

```
            middle_tip =
mp_drawing._normalized_to_pixel_coordinates(

hand_landmarks.landmark[mp_hands.HandLandmark.MIDDL
E_FINGER_TIP].x,

hand_landmarks.landmark[mp_hands.HandLandmark.MIDDL
E_FINGER_TIP].y,
                w, h
            )

            ring_tip =
mp_drawing._normalized_to_pixel_coordinates(

hand_landmarks.landmark[mp_hands.HandLandmark.RING_
FINGER_TIP].x,

hand_landmarks.landmark[mp_hands.HandLandmark.RING_
FINGER_TIP].y,
                w, h
            )

            index_tipm = list(index_tip)
```

```
          index_tipm[0] = np.clip(index_tipm[0], left, right)
          index_tipm[1] = np.clip(index_tipm[1], top,
bottom)

          index_tipm[0] = (index_tipm[0] - left) * fsize[0] /
(right - left)
          index_tipm[1] = (index_tipm[1] - top) * fsize[1] /
(bottom - top)

          if check_cnt == check_every:
            if thumb_tip is not None and index_tip is not
None and middle_tip is not None:
              if euclidean(index_tip, middle_tip) < 40:
                last_event = "dclick"
              else:
                if last_event == "dclick":
                  last_event = None
            if thumb_tip is not None and index_pip is not
None:
              if euclidean(thumb_tip, index_pip) < 60:
                last_event = "sclick"
              else:
                if last_event == "sclick":
```

```
                last_event = None
        if thumb_tip is not None and index_tip is not
None:
            if euclidean(thumb_tip, index_tip) < 60:
                last_event = "press"
            else:
                if last_event == "press":
                    last_event = "release"
        if thumb_tip is not None and middle_tip is not
None:
            if euclidean(thumb_tip, middle_tip) < 60:
                last_event = "rclick"
            else:
                if last_event == "rclick":
                    last_event = None

        if thumb_tip is not None and ring_tip is not
None:
            if euclidean(thumb_tip, ring_tip) < 60:
                last_event = "scroll_down"
            else:
                if last_event == "scroll_down":
                    last_event = None
```

```
                        if index_tip is not None and middle_tip is not
None:
                            if euclidean(index_tip, middle_tip) < 60:
                                last_event = "scroll_up"
                            else:
                                if last_event == "scroll_up":
                                    last_event = None

                        check_cnt = 0

                    if check_cnt > 1:
                        last_event = None

                    screen_pos = frame_pos2screen_pos(fsize, ssize,
index_tipm)

                    mouse.move(screen_pos[0], screen_pos[1])

                    if check_cnt == 0:
                        if last_event == "sclick":
                            mouse.click()
                        elif last_event == "rclick":
```

```
                    mouse.right_click()
                elif last_event == "dclick":
                    mouse.double_click()
                elif last_event == "press":
                    mouse.press()
                elif last_event == "scroll_up":
                    mouse.wheel(delta=1)
                elif last_event == "scroll_down":
                    mouse.wheel(delta=-1)
                else:
                    mouse.release()

                print(last_event)
                cv2.putText(frame, last_event, (20, 20),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)
            check_cnt += 1
            mp_drawing.draw_landmarks(frame,
hand_landmarks, mp_hands.HAND_CONNECTIONS)

        cv2.imshow("Window", frame)
        out.write(frame)
```

```
            if cv2.waitKey(1) & 0xFF == 27:
                print("Exiting...")
                break

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        cam.release()
        out.release()
        cv2.destroyAllWindows()
```

##Video Link

https://drive.google.com/file/d/14486HS-XsELdjgQADJOzquM-baFwa2So/view?usp=sharing