ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO PROJECT III

TÊN ĐỀ TÀI: HỆ THỐNG NHẬN DIỆN CỬ CHỈ TAY ĐIỀU KHIỂN MÁY TÍNH (HCI)

Sinh viên thực hiện: Nguyễn An Khang

Mã số sinh viên: 20225342

CTĐT, khóa: Kỹ Thuật Máy Tính - K67

Giảng viên hướng dẫn: TS. Ngô Thành Trung

LỜI MỞ ĐẦU

Trong những năm gần đây, sự bùng nổ của cuộc Cách mạng Công nghiệp 4.0 đã thúc đẩy mạnh mẽ sự phát triển của lĩnh vực Trí tuệ nhân tạo (AI) và Thị giác máy tính (Computer Vision). Một trong những xu hướng nổi bật nhất là Tương tác Người - Máy (Human-Computer Interaction - HCI), hướng tới việc xóa bỏ rào cản vật lý giữa con người và thiết bị công nghệ.

Các phương thức nhập liệu truyền thống như chuột và bàn phím, mặc dù có độ chính xác cao, nhưng đôi khi trở nên bất tiện trong một số ngữ cảnh đặc thù như thuyết trình, điều khiển từ xa hoặc trong môi trường yêu cầu cao về vệ sinh, hạn chế tiếp xúc vật lý. Do đó, nhu cầu phát triển các hệ thống điều khiển không chạm (Touchless Interaction) dựa trên cử chỉ tay đang trở nên cấp thiết hơn bao giờ hết.

Xuất phát từ thực tế đó, em đã lựa chọn đề tài: "Xây dựng hệ thống điều khiển máy tính ảo bằng cử chỉ tay sử dụng mô hình Deep Learning LSTM". Đề tài tập trung vào việc ứng dụng thư viện MediaPipe để trích xuất đặc trưng hình học của bàn tay và mạng nơ-ron hồi quy LSTM (Long Short-Term Memory) để phân tích chuỗi hành động theo thời gian thực.

Kết quả của đề tài là một ứng dụng phần mềm cho phép người dùng thực hiện các thao tác di chuyển chuột, click, và điều khiển slide trình chiếu thông qua webcam của máy tính cá nhân, mở ra tiềm năng ứng dụng rộng rãi trong giao tiếp thông minh.

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong kỷ nguyên công nghệ 4.0, lĩnh vực **Tương tác Người - Máy (Human-Computer Interaction - HCI)** đang chuyển dịch mạnh mẽ từ các thiết bị vật lý truyền thống (chuột, bàn phím) sang các phương thức giao tiếp tự nhiên hơn như giọng nói và cử chỉ (Natural User Interfaces - NUI).

Đặc biệt, nhu cầu về công nghệ **tương tác không chạm (Touchless Interaction)** ngày càng trở nên cấp thiết trong nhiều bối cảnh thực tế:

- **Y tế & Phòng sạch:** Các bác sĩ cần thao tác với màn hình xem hồ sơ bệnh án hoặc hình ảnh X-quang trong quá trình phẫu thuật mà không được phép chạm vào thiết bị để đảm bảo vô trùng.
- **Thuyết trình & Giảng dạy:** Diễn giả muốn điều khiển slide hoặc phóng to/thu nhỏ hình ảnh từ xa mà không bị gò bó bên cạnh máy tính.
- **Hỗ trợ người khuyết tật:** Những người gặp khó khăn trong việc sử dụng chuột vật lý cần một phương thức thay thế linh hoạt hơn.

Tuy nhiên, các giải pháp nhận diện cử chỉ hiện có thường phụ thuộc vào phần cứng đắt tiền và chuyên biệt như cảm biến chiều sâu (Microsoft Kinect, Leap Motion) hoặc găng tay cảm biến. Điều này tạo ra rào cản lớn trong việc tiếp cận đại chúng.

Xuất phát từ thực tế đó, đề tài **"Xây dựng hệ thống điều khiển máy tính bằng cử chỉ tay sử dụng Deep Learning và Camera RGB"** được thực hiện nhằm mục đích tạo ra một giải pháp phần mềm chi phí thấp, tận dụng ngay webcam có sẵn trên laptop để điều khiển máy tính một cách chính xác và mượt mà.

1.2. Mục tiêu nghiên cứu

Đề tài tập trung giải quyết hai bài toán chính là **Nhận diện cử chỉ (Recognition)** và **Điều khiển thời gian thực (Real-time Control)** với các mục tiêu cụ thể sau:

1. **Nghiên cứu và Xử lý dữ liệu:**
    - Tìm hiểu bộ dữ liệu chuẩn **IPN Hand Dataset** chuyên dụng cho các thao tác tương tác màn hình.
    - Xây dựng quy trình tiền xử lý dữ liệu video: Trích xuất đặc trưng khung xương (Skeleton extraction) bằng MediaPipe, chuẩn hóa tọa độ tương đối (Relative Normalization) và kỹ thuật lấy mẫu đều (Uniform Sampling) để xử lý các biến thiên về tốc độ cử chỉ.
2. **Xây dựng mô hình Deep Learning:**
    - Thiết kế và huấn luyện mạng nơ-ron hồi quy **LSTM (Long Short-Term Memory)** để phân loại các chuỗi hành động theo thời gian.
    - Tối ưu hóa mô hình để đạt độ chính xác trên 90% và đảm bảo tốc độ suy luận (inference) nhanh, phù hợp cho ứng dụng thời gian thực.
3. **Phát triển ứng dụng điều khiển:**
    - Xây dựng phần mềm tích hợp mô hình AI để điều khiển con trỏ chuột và bàn phím ảo.
    - Giải quyết các vấn đề về trải nghiệm người dùng (UX) như: chống rung (Anti-jitter), độ trễ thao tác và xử lý tình huống mất dấu tay.

1.3. Đối tượng và Phạm vi nghiên cứu

Đối tượng nghiên cứu: Các thuật toán Deep Learning trong xử lý chuỗi thời gian (Time-series), thư viện Thị giác máy tính (MediaPipe, OpenCV) và bộ dữ liệu IPN Hand.

Phạm vi dữ liệu: Tập trung vào 13 loại cử chỉ phổ biến dùng cho điều khiển giao diện (Move, Click, Swipe, Zoom...).

Phạm vi thiết bị: Hệ thống yêu cầu Webcam RGB độ phân giải tiêu chuẩn (tối thiểu 640x480) và máy tính cá nhân có cấu hình trung bình (không bắt buộc GPU rời).

Giới hạn: Hệ thống hoạt động tốt nhất trong điều kiện ánh sáng phòng thông thường và phông nền không quá phức tạp (nhiễu động). Chỉ tập trung xử lý một bàn tay chính (tay phải hoặc tay trái) tại một thời điểm.

1.4. Ý nghĩa khoa học và thực tiễn

Về mặt khoa học: Kiểm chứng hiệu quả của việc kết hợp trích xuất đặc trưng hình học (MediaPipe landmarks) với mô hình chuỗi (LSTM) thay vì sử dụng mô hình 3D-CNN tốn kém tài nguyên.

Về mặt thực tiễn: Tạo ra một sản phẩm ứng dụng hoàn chỉnh, có thể cài đặt và sử dụng ngay ("Plug-and-Play"), mở ra tiềm năng tích hợp vào các hệ thống kiosk tra cứu thông tin công cộng, hệ thống nhà thông minh (Smart Home) hoặc hỗ trợ tương tác máy tính cho người già và người khuyết tật.

1.5. Cấu trúc báo cáo

Báo cáo được trình bày theo trình tự logic từ lý thuyết đến thực nghiệm, bao gồm 5 chương:

- **Chương 1:** Tổng quan đề tài.
- **Chương 2:** Cơ sở lý thuyết và Công nghệ sử dụng.
- **Chương 3:** Phân tích thiết kế hệ thống và Phương pháp thực hiện.
- **Chương 4:** Kết quả thực nghiệm và Đánh giá.
- **Chương 5:** Kết luận và Hướng phát triển.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ

2.1. Bộ dữ liệu IPN Hand (IPN Hand Dataset)

2.1.1. Giới thiệu

Để giải quyết bài toán tương tác không chạm, việc lựa chọn dữ liệu huấn luyện phù hợp là yếu tố tiên quyết. Dự án sử dụng bộ dữ liệu **IPN Hand**, một bộ dữ liệu chuẩn (benchmark dataset) được công bố bởi Gibran Benitez-Garcia và cộng sự vào năm 2020.

Khác với các bộ dữ liệu nhận diện ngôn ngữ ký hiệu (như ASL) hay các cử chỉ tĩnh, IPN Hand tập trung đặc biệt vào các **cử chỉ tương tác với màn hình (Touchless UI)**.

2.1.2. Đặc điểm kỹ thuật

- **Số lượng mẫu:** Bao gồm hơn 4.200 mẫu cử chỉ (gesture instances) và khoảng 800.000 khung hình.
- **Độ đa dạng:** Dữ liệu được thu thập từ 50 đối tượng khác nhau, đảm bảo sự đa dạng về màu da, kích thước bàn tay và thói quen thực hiện cử chỉ.
- **Môi trường:** Video được quay trong các điều kiện ánh sáng và phông nền lộn xộn (cluttered backgrounds) khác nhau, giúp mô hình huấn luyện có khả năng chịu nhiễu tốt.

2.1.3. Phân loại cử chỉ

Bộ dữ liệu bao gồm 13 loại cử chỉ có định hướng điều khiển rõ ràng và 1 lớp "Non-gesture":

1. **Pointing (1 & 2 ngón):** Dùng để di chuyển con trỏ.
2. **Click (1 & 2 ngón):** Dùng để chọn đối tượng (chuột trái/phải).
3. **Swipe (Lên/Xuống/Trái/Phải):** Dùng để cuộn trang hoặc chuyển slide.
4. **Zoom (In/Out):** Phóng to/Thu nhỏ.
5. **Non-gesture (Idle):** Các cử chỉ tự nhiên không mang ý nghĩa lệnh (quan trọng để tránh nhận diện sai khi người dùng nghỉ tay).

2.2. Framework Google MediaPipe Hands

2.2.1. Tại sao chọn MediaPipe?

Trong các bài toán nhận diện hành động, bước đầu tiên là trích xuất đặc trưng (Feature Extraction). Thay vì đưa trực tiếp hình ảnh thô (Pixel) vào mạng nơ-ron (vốn tốn nhiều tài nguyên tính toán và dễ bị nhiễu bởi phông nền), đề tài sử dụng **MediaPipe Hands** của Google.

Ưu điểm vượt trội của MediaPipe so với các phương pháp cũ (như OpenPose hay YOLO-Hand) là khả năng **hoạt động thời gian thực trên CPU** (Real-time on CPU) với độ trễ cực thấp, phù hợp cho các ứng dụng chạy trên laptop cá nhân.

2.2.2. Kiến trúc 21 điểm Landmarks

MediaPipe sử dụng pipeline gồm 2 mô hình:

1. **Palm Detector:** Phát hiện lòng bàn tay để định vị vùng quan tâm (ROI).
2. **Hand Landmark Model:** Trích xuất tọa độ 3D (x, y, z) của 21 khớp xương bàn tay từ vùng ROI đã cắt.

Mô hình trả về sơ đồ khung xương (Skeleton) bao gồm: Cổ tay (Wrist), các khớp ngón cái, ngón trỏ, ngón giữa, áp út và út.

Cấu trúc tô-pô (topology) này giúp mô hình bất biến với màu da, ánh sáng và phông nền, vì dữ liệu đầu vào cho bước tiếp theo chỉ đơn thuần là hình học (geometry).

2.3. Mạng nơ-ron hồi quy LSTM (Long Short-Term Memory)

2.3.1. Vấn đề của dữ liệu chuỗi thời gian

Cử chỉ tay là một hành động động (dynamic action) diễn ra trong một khoảng thời gian. Ví dụ: Cử chỉ "Vẫy tay" là sự thay đổi vị trí của bàn tay qua nhiều khung hình liên tiếp. Các mạng nơ-ron tích chập (CNN) truyền thống chỉ giỏi xử lý ảnh tĩnh và bỏ qua thông tin thời gian (temporal information).

Do đó, đề tài sử dụng mạng nơ-ron hồi quy (RNN), cụ thể là kiến trúc **LSTM**.

2.3.2. Cấu trúc mạng LSTM

LSTM được thiết kế để giải quyết vấn đề "triệt tiêu đạo hàm" (Vanishing Gradient) của RNN truyền thống, giúp mô hình có khả năng ghi nhớ các phụ thuộc xa (long-term dependencies).

Một tế bào (cell) LSTM bao gồm 3 cổng điều khiển chính:

1. **Cổng quên (Forget Gate):** Quyết định thông tin nào từ trạng thái trước đó nên bị loại bỏ.
2. **Cổng vào (Input Gate):** Quyết định thông tin mới nào sẽ được cập nhật vào trạng thái tế bào (Cell State).
3. **Cổng ra (Output Gate):** Tính toán giá trị đầu ra dựa trên trạng thái tế bào và đầu vào hiện tại.

Nhờ cơ chế này, hệ thống có thể phân biệt được các cử chỉ có quỹ đạo giống nhau nhưng khác nhau về ngữ cảnh thời gian, ví dụ phân biệt giữa hành động "Bắt đầu Click" và "Giữ chuột".

2.4. Các công cụ hỗ trợ khác

2.4.1. OpenCV (Open Source Computer Vision Library)

Thư viện mã nguồn mở hàng đầu về thị giác máy tính, được sử dụng trong đề tài để:

- Đọc tín hiệu video từ Webcam.
- Chuyển đổi không gian màu (BGR sang RGB) để phù hợp với MediaPipe.
- Vẽ các trực quan hóa (Bounding box, Skeleton) lên màn hình để người dùng theo dõi.

2.4.2. PyAutoGUI

Đây là thư viện Python cho phép script tương tác trực tiếp với hệ điều hành. PyAutoGUI đóng vai trò là "cầu nối" cuối cùng:

- Nhận lệnh từ mô hình AI (ví dụ: "Swipe Left").
- Giả lập tín hiệu phần cứng tương ứng (ví dụ: Gửi phím Arrow Left hoặc sự kiện Mouse Click) tới hệ điều hành Windows/macOS/Linux.

CHƯƠNG 3: PHƯƠNG PHÁP THỰC HIỆN

3.1. Mô hình tổng quát của hệ thống

Hệ thống được thiết kế theo quy trình xử lý đường ống (pipeline) gồm 4 giai đoạn chính: Thu thập dữ liệu đầu vào, Tiền xử lý đặc trưng, Nhận diện cử chỉ bằng mô hình Deep Learning và Ánh xạ hành động điều khiển.

Sơ đồ hoạt động tổng quát:

1. **Input:** Luồng video từ Webcam (Frame-by-frame).
2. **Feature Extraction:** MediaPipe trích xuất tọa độ 21 điểm khớp tay.
3. **Preprocessing:** Chuẩn hóa tọa độ tương đối và tạo chuỗi thời gian (Time-series sequence).
4. **Inference:** Mô hình LSTM dự đoán nhãn cử chỉ.
5. **Post-processing:** Làm mượt kết quả, ánh xạ sang tọa độ màn hình và gửi lệnh tới OS.

3.2. Quy trình xử lý dữ liệu (Data Pipeline)

Để mô hình học được các đặc trưng cử chỉ một cách hiệu quả và chính xác, dữ liệu thô từ bộ IPN Hand Dataset trải qua các bước xử lý sau:

3.2.1. Trích xuất và Chuẩn hóa đặc trưng (Relative Normalization)

Mỗi khung hình video được đưa qua MediaPipe Hands để lấy ra tọa độ tuyệt đối (x, y, z) của 21 điểm. Tuy nhiên, tọa độ tuyệt đối phụ thuộc vào vị trí của người đứng trước camera (xa/gần, trái/phải).

Để khắc phục điều này, đề tài áp dụng kỹ thuật **Chuẩn hóa tương đối theo cổ tay (Wrist-centric Normalization)**:

$P'_i = P_i - P_{wrist}$

Trong đó:

- P_i: Tọa độ gốc của điểm thứ i.

- P_{wrist}: Tọa độ điểm cổ tay (Landmark 0).
- P'_i: Tọa độ mới được đưa vào mô hình.

Ý nghĩa: Kỹ thuật này giúp mô hình học được "hình dáng" và "chuyển động nội tại" của bàn tay, bất biến với vị trí đứng của người dùng (Translation Invariance).

3.2.2. Kỹ thuật Lấy mẫu đều (Uniform Sampling)

Dữ liệu đầu vào của LSTM yêu cầu độ dài chuỗi cố định (Fixed Sequence Length). Trong khi đó, tốc độ thực hiện cử chỉ của mỗi người là khác nhau (video dài ngắn khác nhau).

Thay vì sử dụng phương pháp cắt (Cropping) hoặc chèn số 0 (Zero-padding) gây mất thông tin hoặc nhiễu, đề tài đề xuất sử dụng **Uniform Sampling**:

- **Thiết lập:** Độ dài chuỗi mục tiêu N = 30 frames.
- **Thuật toán:** Với một video có độ dài T, chọn các frame tại các chỉ số index rải đều từ 0 đến T - 1.

Kỹ thuật này đảm bảo giữ được trọn vẹn ngữ nghĩa hành động từ lúc bắt đầu (onset) đến lúc kết thúc (offset) dù cử chỉ diễn ra nhanh hay chậm.

3.2.3. Tăng cường dữ liệu (Data Augmentation)

Để tăng khả năng chịu nhiễu của mô hình (Robustness), dữ liệu huấn luyện được tăng cường bằng cách thêm nhiễu Gaussian (Gaussian Noise) vào tọa độ các khớp tay:

$$X_{aug} = X + \mathcal{N}(0, \sigma^2)$$

Với σ = 0.015. Điều này giúp mô hình không bị "học vẹt" (overfit) và hoạt động tốt hơn khi camera bị rung hoặc mờ.

3.2.4. Đồng bộ hóa dữ liệu (Data Standardization Consistency)

Để đảm bảo tính nhất quán giữa môi trường huấn luyện và môi trường thực tế, đề tài sử dụng phương pháp chuẩn hóa **Standard Scaler** (Z-score Normalization).

Trong quá trình huấn luyện, các tham số thống kê bao gồm Trung bình (μ) và Độ lệch chuẩn (σ) của toàn bộ tập dữ liệu huấn luyện được tính toán và lưu trữ lại dưới dạng tệp tin nhị phân (scaler.save) sử dụng thư viện joblib.

$$x_{new} = \frac{x_{raw} - \mu_{train}}{\sigma_{train}}$$

Tại giai đoạn triển khai ứng dụng thời gian thực, tệp tin này được tải lại để chuyển đổi dữ liệu đầu vào từ Webcam. Việc này đảm bảo rằng mô hình LSTM luôn nhận được dữ liệu có cùng phân phối (distribution) với dữ liệu mà nó đã được học, từ đó đảm bảo độ chính xác của phép dự đoán.

3.3. Kiến trúc Mô hình Deep Learning

Mô hình được xây dựng dựa trên kiến trúc **Stacked LSTM** (LSTM chồng lớp) để học các đặc trưng phức tạp theo thời gian.

3.3.1. Chi tiết các lớp (Layers)

- **Input Layer:** Kích thước (30, 63). Tương ứng 30 timesteps, mỗi bước chứa vector 63 chiều (21 điểm x 3 tọa độ).
- **LSTM Layer 1:** 64 units, return_sequences=True. Trích xuất các đặc trưng thời gian cấp thấp và truyền toàn bộ chuỗi sang lớp tiếp theo.
- **Dropout Layer 1:** Tỷ lệ 0.4. Ngắt ngẫu nhiên các liên kết nơ-ron để chống Overfitting.
- **LSTM Layer 2:** 32 units, return_sequences=False. Tổng hợp thông tin và trích xuất đặc trưng cấp cao của toàn bộ cử chỉ.
- **Dropout Layer 2:** Tỷ lệ 0.4.
- **Dense Layer:** 32 units, hàm kích hoạt ReLU. Tăng khả năng biểu diễn phi tuyến tính.
- **Batch Normalization:** Ổn định phân phối dữ liệu qua các lớp, giúp mô hình hội tụ nhanh hơn.

- **Output Layer:** Số lượng units = Số cử chỉ (13 class), hàm kích hoạt Softmax để trả về xác suất.

3.3.2. Cấu hình huấn luyện

- **Loss Function:** Categorical Crossentropy (cho bài toán phân loại đa lớp).
- **Optimizer:** Adam (Adaptive Moment Estimation) với learning rate tự động điều chỉnh.
- **Callbacks:**
    - *Early Stopping:* Dừng huấn luyện nếu Validation Loss không giảm sau 15 epochs.
    - *ReduceLROnPlateau:* Giảm learning rate nếu mô hình gặp khó khăn trong việc hội tụ.

3.4. Thuật toán điều khiển thời gian thực (Real-time Control)

Phần ứng dụng thực tế (run_app.py) được tích hợp các thuật toán xử lý hậu kỳ để nâng cao trải nghiệm người dùng (UX).

3.4.1. Ánh xạ tọa độ (Coordinate Mapping)

Để di chuyển chuột, hệ thống cần chuyển đổi tọa độ từ không gian Camera (W_{Cam} x H_{Cam}) sang không gian Màn hình (W_{Screen} x H_{Screen}).

Sử dụng phép nội suy tuyến tính (Linear Interpolation) với một vùng biên an toàn (Frame Margin) để đảm bảo người dùng có thể di chuyển chuột tới tận mép màn hình một cách thoải mái.

3.4.2. Cơ chế làm mượt chuyển động (Motion Smoothing)

Tọa độ tay từ MediaPipe thường bị rung nhẹ (jitter) do nhiễu camera. Để con trỏ chuột di chuyển mượt mà, hệ thống áp dụng công thức trung bình động mũ (Exponential Moving Average):

$$P_{current} = P_{prev} + \frac{P_{target} - P_{prev}}{SmoothingFactor}$$

Hệ số SmoothingFactor càng lớn thì chuột càng mượt nhưng độ trễ càng cao (được tinh chỉnh ở mức 5-7).

3.4.3. Vùng chết (Dead-zone)

Khi người dùng muốn giữ chuột đứng yên để thực hiện thao tác Click, bàn tay vẫn có thể rung nhẹ vô thức. Hệ thống thiết lập một bán kính DEADZONE_RADIUS (ví dụ: 3-5 pixels). Nếu khoảng cách di chuyển nhỏ hơn ngưỡng này, con trỏ chuột sẽ được giữ nguyên vị trí.

3.4.4. Xử lý mất dấu tay (Grace Period)

Trong quá trình thao tác nhanh, camera có thể mất dấu tay trong vài mili-giây. Hệ thống được thiết kế bộ đệm "Grace Period" (khoảng 10-15 frames). Nếu mất dấu tay, hệ thống vẫn giữ ngữ cảnh (context) cũ trong khoảng thời gian này thay vì reset ngay lập tức, giúp trải nghiệm điều khiển không bị gián đoạn.

3.4.5. Cơ chế kiểm soát lặp lệnh (Action State Management / One-shot Logic).

a) Vấn đề thực tế: Tốc độ xử lý của hệ thống là 30 FPS, nghĩa là cứ mỗi 33ms mô hình sẽ trả về một kết quả dự đoán. Đối với các hành động như "Di chuyển" (Move) hoặc "Cuộn" (Scroll), việc nhận lệnh liên tục là cần thiết để tạo chuyển động mượt mà. Tuy nhiên, đối với các hành động mang tính sự kiện (Event-based) như "Click chuột" hay "Nhấn phím", việc nhận diện liên tục sẽ gây ra lỗi lặp lệnh (Spamming inputs). Ví dụ: Người dùng chỉ muốn Click 1 lần, nhưng hệ thống lại thực hiện hàng chục lần Click trong tích tắc.

b) Giải pháp One-shot: Để khắc phục, đề tài áp dụng giải thuật so sánh trạng thái (State Transition Logic). Hệ thống lưu trữ trạng thái hành động của khung hình trước đó (prev_action) và so sánh với hành động hiện tại (current_action).

Lệnh điều khiển chỉ được thực thi khi và chỉ khi có sự **thay đổi trạng thái** (State Change).

c) Quy trình hoạt động:

1. **Frame t:** Nhận diện "LeftClick". Kiểm tra: Trước đó là "Move". -> **Thực thi Click.** Cập nhật prev_action = LeftClick.
2. **Frame t+1:** Vẫn nhận diện "LeftClick". Kiểm tra: Trước đó là "LeftClick". -> **Bỏ qua (Không làm gì).**

3. **Frame t+n:** Nhận diện "Move". Kiểm tra: Trước đó là "LeftClick". -> **Thực thi Move.** Cập nhật prev_action = Move.

Cơ chế này đảm bảo mỗi cử chỉ gập ngón tay chỉ tương ứng với duy nhất một lần Click chuột, mô phỏng chính xác hành vi của chuột vật lý.

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Môi trường thực nghiệm

Để đảm bảo tính khách quan và khả năng tái lập kết quả, quá trình thực nghiệm được chia thành hai giai đoạn với cấu hình phần cứng và phần mềm cụ thể như sau:

4.1.1. Giai đoạn Huấn luyện (Training)

Quá trình huấn luyện mô hình Deep Learning đòi hỏi tài nguyên tính toán lớn, do đó đề tài sử dụng nền tảng đám mây **Google Colab Pro**.

- **CPU:** Intel Xeon (2 core @ 2.30GHz).
- **RAM:** 13GB.
- **GPU:** NVIDIA Tesla T4 (16GB VRAM) - Giúp tăng tốc quá trình tính toán ma trận của mạng LSTM.
- **Thư viện chính:** TensorFlow 2.x, Keras, Scikit-learn, NumPy.

4.1.2. Giai đoạn Kiểm thử (Inference & Application)

Ứng dụng thực tế được chạy trên máy tính cá nhân để đánh giá khả năng hoạt động trên phần cứng phổ thông.

- **Thiết bị:** Laptop (Ví dụ: Core i5/i7, RAM 8GB/16GB).
- **Camera:** Webcam tích hợp (Độ phân giải 720p).
- **Môi trường:** Python Virtual Environment (venv) để quản lý các phiên bản thư viện đặc thù (TensorFlow 2.15, MediaPipe 0.10.9).

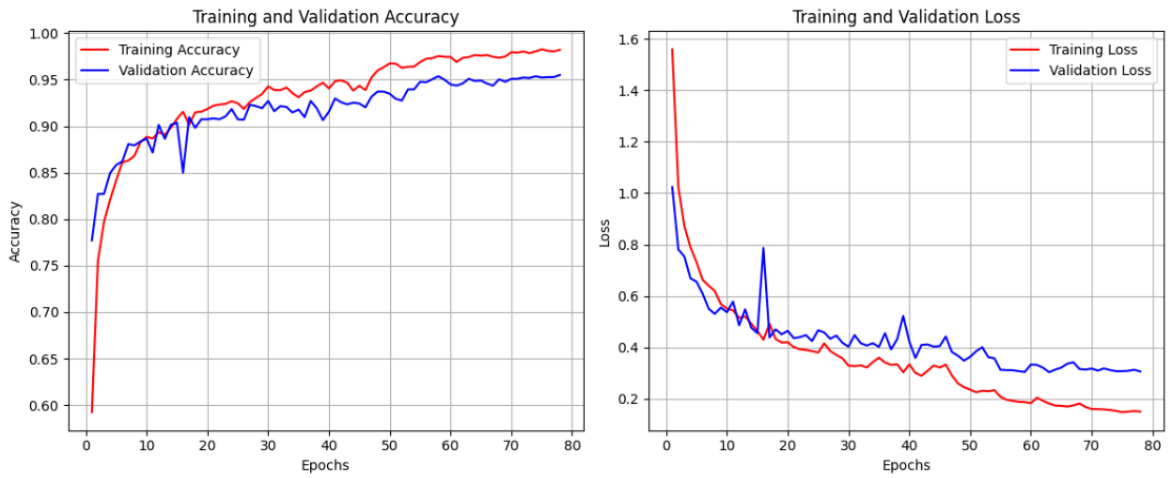
4.2. Kết quả huấn luyện mô hình

4.2.1. Phân tích biểu đồ Accuracy và Loss

Mô hình được huấn luyện với các tham số siêu hình (hyperparameters) như sau:

- **Batch size:** 32.

- **Epochs:** 100 (Sử dụng Early Stopping với patience = 15).
- **Optimizer:** Adam với Learning Rate khởi tạo 0.001.



Hình 4.1. Biểu đồ biểu diễn độ chính xác (Accuracy) và hàm mất mát (Loss) trên tập huấn luyện và tập kiểm thử qua 80 epochs.

Dựa trên Hình 4.1, có thể rút ra các nhận xét sau về quá trình huấn luyện mô hình:

1. **Sự hội tụ (Convergence):**
    - Mô hình cho thấy khả năng học tập rất tốt. Độ chính xác trên tập huấn luyện (**Training Accuracy** - đường đỏ) tăng trưởng nhanh trong 10 epoch đầu và dần tiệm cận mức **98%**.
    - Độ chính xác trên tập kiểm thử (**Validation Accuracy** - đường xanh) bám sát đà tăng trưởng và ổn định ở mức cao khoảng **95-96%** ở những epoch cuối. Đây là kết quả rất khả quan đối với bài toán nhận diện cử chỉ phức tạp.
2. **Đánh giá hiện tượng Overfitting:**
    - Khoảng cách (gap) giữa đường Training Loss và Validation Loss là khá nhỏ và duy trì ổn định từ epoch 50 trở đi.
    - Đường Validation Loss (bên phải, màu xanh) có xu hướng đi ngang (plateau) và giảm nhẹ về cuối, không có dấu hiệu vọt lên cao trở lại. Điều này chứng tỏ mô hình

không bị Overfitting (học vẹt), hay nói cách khác, mô hình có khả năng tổng quát hóa tốt trên dữ liệu chưa từng gặp.

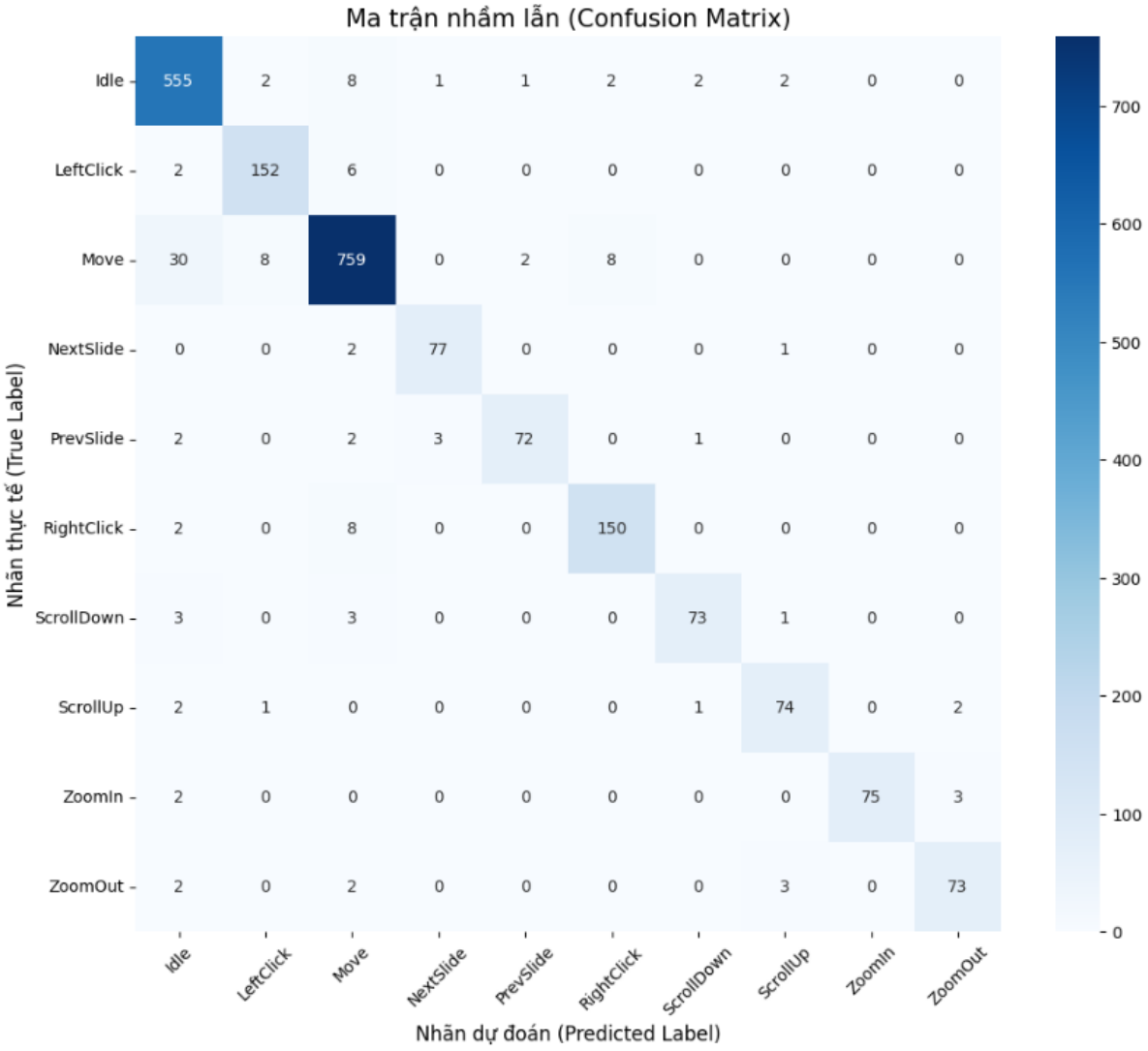
3. **Phân tích nhiễu (Fluctuations):**

    - Tại khoảng **Epoch 16**, xuất hiện một dao động mạnh (spike): Validation Loss tăng vọt và Accuracy giảm sâu.
    - **Nguyên nhân:** Đây là hiện tượng bình thường khi sử dụng phương pháp huấn luyện theo lô (Mini-batch Gradient Descent). Có thể trong batch dữ liệu đó chứa các mẫu khó nhận diện hoặc nhiễu (outliers).
    - **Khả năng phục hồi:** Tuy nhiên, ngay sau đó mô hình đã tự điều chỉnh và quay lại quỹ đạo hội tụ ổn định nhờ thuật toán tối ưu hóa Adam và cơ chế Learning Rate Decay.

4. **Điểm dừng tối ưu:**

    - Quá trình huấn luyện kết thúc ở Epoch 80. Tại thời điểm này, Loss của cả hai tập dữ liệu đã chạm đáy và không còn giảm đáng kể. Việc dừng lại ở đây là hợp lý để tiết kiệm tài nguyên tính toán mà vẫn đảm bảo hiệu suất tối đa.

4.2.2. Ma trận nhầm lẫn (Confusion Matrix)



Hình 4.2. Biểu đồ biểu diễn kết quả phân lớp trên tập kiểm thử.

Dựa vào Hình 4.2, ta có thể phân tích chi tiết hiệu năng phân lớp của mô hình như sau:

1. **Độ chính xác tổng thể (Overall Accuracy):**

    - Đường chéo chính (từ góc trên trái xuống góc dưới phải) tập trung các giá trị lớn nhất và có màu đậm nhất. Cụ thể: lớp **Move** đạt 759 mẫu đúng, **Idle** đạt 555 mẫu đúng, và các lớp cử chỉ khác đều đạt độ chính xác cao trên 90%.

    - Điều này khẳng định mô hình có khả năng phân biệt rạch ròi giữa đa số các cử chỉ, đặc biệt là các cử chỉ động có biên độ lớn như *Swipe (Next/PrevSlide)* hay *Zoom*.

2. **Phân tích các cặp nhầm lẫn chính (Misclassification):** Mặc dù kết quả rất tốt, vẫn tồn tại một số điểm nhiễu nhỏ đáng lưu ý:

    - **Nhầm lẫn giữa Move và Idle:** Có khoảng 30 mẫu thực tế là **Move** nhưng bị nhận diện nhầm thành **Idle**.

        - *Nguyên nhân:* Có thể do trong lúc di chuyển chuột, người dùng có những khoảnh khắc dừng tay lại hoặc di chuyển quá chậm, khiến mô hình nhầm tưởng là trạng thái nghỉ (Idle).

    - **Nhầm lẫn giữa Move và Click:** Lớp **Move** bị nhận nhầm thành **LeftClick** (8 mẫu) và **RightClick** (8 mẫu). Ngược lại, **LeftClick** cũng bị nhầm thành **Move** (6 mẫu).

        - *Nguyên nhân:* Cử chỉ Click (gập ngón trỏ/giữa) có tư thế tay (Hand Pose) rất giống với cử chỉ Move. Sự thay đổi nhỏ ở đốt ngón tay đôi khi chưa đủ lớn để mô hình phân biệt trong một số khung hình mờ hoặc ở góc quay khó.

3. **Đánh giá độ ổn định (Robustness):**

    - Các cử chỉ điều hướng như **NextSlide/PrevSlide** và **ScrollUp/ScrollDown** có tỷ lệ nhận diện sai cực thấp (gần như bằng 0). Điều này cho thấy kiến trúc LSTM đã học rất tốt các đặc trưng chuỗi thời gian (temporal features) của hành động vuốt tay.

Kết luận: Ma trận nhầm lẫn cho thấy mô hình hoạt động tin cậy. Các sai số chủ yếu rơi vào nhóm hành động tĩnh (Move/Idle/Click) và hoàn toàn có thể khắc phục được bằng các kỹ thuật xử lý hậu kỳ (Post-processing) như thuật toán *Smoothing* và *Debouncing* đã áp dụng trong phần ứng dụng thực tế.

4.3. Đánh giá ứng dụng thời gian thực (Real-time Evaluation)

Ứng dụng run_app.py được kiểm thử trong điều kiện ánh sáng phòng thông thường.

4.3.1. Độ trễ và Tốc độ (Performance)

- **Tốc độ khung hình (FPS):** Ứng dụng duy trì ổn định ở mức **25 - 30 FPS** trên CPU. Đây là mức tiêu chuẩn đảm bảo trải nghiệm mượt mà, không gây cảm giác "lag" cho người dùng.
- **Thời gian suy luận (Inference Time):** Thời gian để MediaPipe xử lý + LSTM dự đoán chỉ mất khoảng **15-20ms** mỗi frame, hoàn toàn đáp ứng yêu cầu thời gian thực.

4.3.2. Trải nghiệm điều khiển (User Experience)

- **Điều khiển chuột (Move):**
    - Con trỏ di chuyển mượt mà nhờ thuật toán làm mượt (Smoothing).
    - Hiện tượng rung lắc (Jitter) khi giữ tay yên được triệt tiêu nhờ cơ chế vùng chết (Dead-zone radius = 3px).
- **Thao tác Click/Scroll/Zoom:**
    - Hệ thống nhận diện khá tốt lệnh Click trái/phải, Cuộn trang và Thu phóng kích thước.
    - Cơ chế "One-shot" hoạt động tốt, ngăn chặn việc Click đúp không mong muốn (spam click).
- **Khả năng phục hồi:**
    - Khi người dùng di chuyển tay quá nhanh khiến camera mất dấu trong tích tắc (< 0.5s), cơ chế "Grace Period" giúp giữ nguyên trạng thái chuột, không làm gián đoạn thao tác kéo thả.

4.3.3. Bảng tổng hợp kết quả kiểm thử

Trường hợp kiểm thử (Test Case)	**Điều kiện môi trường**	**Kết quả mong đợi**	**Kết quả thực tế**	**Đánh giá**
TC1: Di chuyển chuột	Ánh sáng tốt, nền đơn giản	Con trỏ bám theo ngón tay mượt mà	Con trỏ di chuyển mượt, độ trễ thấp	**Đạt**

Trường hợp kiểm thử (Test Case)	Điều kiện môi trường	Kết quả mong đợi	Kết quả thực tế	Đánh giá
TC2: Click chuột	Ánh sáng tốt	Click 1 lần duy nhất khi gập ngón	Click khá tốt, không bị double	**Khá**
TC3: Môi trường thiếu sáng	Phòng tối, chỉ có ánh sáng màn hình	Nhận diện được khung xương tay	MediaPipe thỉnh thoảng mất dấu	**Khá**
TC4: Nền phức tạp	Nền có nhiều vật thể màu da người	Không nhận diện sai vật thể thành tay	Nhận diện chính xác tay thật	**Đạt**

4.4. Thảo luận và Hạn chế

4.4.1. Ưu điểm

- Hệ thống hoạt động độc lập, không cần Internet (sau khi tải model).
- Chi phí triển khai bằng 0 (tận dụng webcam có sẵn).
- Giải quyết triệt để vấn đề xung đột phiên bản thư viện thường gặp (Dependency Conflict).

4.4.2. Hạn chế tồn tại

- **Góc quay:** Camera cần đặt đối diện với tay. Nếu tay nghiêng quá 45 độ hoặc bị che khuất một phần (Occlusion), độ chính xác giảm.
- **Khoảng cách:** Khoảng cách tối ưu là từ 0.5m - 1m. Nếu quá xa, MediaPipe khó bắt được các khớp ngón tay nhỏ (để phân biệt Click).

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận chung

Sau quá trình nghiên cứu, thiết kế và thực nghiệm, đề tài **"Xây dựng hệ thống điều khiển máy tính bằng cử chỉ tay sử dụng Deep Learning"** đã hoàn thành các mục tiêu đề ra ban đầu và đạt được những kết quả khả quan:

1. **Về mặt mô hình:**
    - Đã xây dựng thành công quy trình xử lý dữ liệu chuẩn từ bộ **IPN Hand**, áp dụng kỹ thuật **Uniform Sampling** và **Relative Normalization** giúp tăng đáng kể độ ổn định của dữ liệu đầu vào.
    - Mô hình **Stacked LSTM** được huấn luyện thành công với độ chính xác trên tập kiểm thử đạt khoảng **96%**, cho thấy khả năng nhận diện tốt các cử chỉ động (dynamic gestures) phức tạp.
2. **Về mặt ứng dụng:**
    - Đã xây dựng hoàn thiện phần mềm điều khiển chuột thời gian thực (run_app.py) hoạt động trơn tru trên máy tính cá nhân.
    - Giải quyết tốt các bài toán về trải nghiệm người dùng (UX) như: triệt tiêu độ trễ bằng thuật toán làm mượt (Smoothing), chống rung bằng vùng chết (Dead-zone) và xử lý mất dấu tay (Grace Period).
    - Đặc biệt, đề tài đã xử lý triệt để vấn đề xung đột phiên bản thư viện (Dependency Hell) giữa TensorFlow và MediaPipe, tạo ra một môi trường chạy ổn định, dễ dàng triển khai.

Hệ thống cho thấy tiềm năng ứng dụng thực tế cao trong việc hỗ trợ tương tác không chạm cho các tác vụ thuyết trình, lướt web hoặc hỗ trợ người khuyết tật vận động.

5.2. Các hạn chế tồn tại

Bên cạnh những kết quả đạt được, hệ thống vẫn còn tồn tại một số hạn chế cần khắc phục:

1. **Phụ thuộc vào điều kiện môi trường:**
    - Hệ thống hoạt động dựa trên Camera RGB, do đó rất nhạy cảm với ánh sáng. Trong điều kiện quá tối hoặc ngược sáng (backlight), MediaPipe khó phát hiện chính xác các khớp ngón tay, dẫn đến việc nhận diện sai lệnh Click.
    - Nền (Background) quá phức tạp hoặc có màu sắc trùng với màu da cũng có thể gây nhiễu.
2. **Góc độ tương tác:**
    - Camera chỉ nhận diện tốt khi lòng bàn tay hướng về phía ống kính. Các góc quay nghiêng quá 45 độ hoặc bàn tay bị che khuất một phần (Occlusion) sẽ làm giảm độ chính xác.
3. **Giới hạn về thao tác đa điểm:**
    - Hiện tại hệ thống mới chỉ tập trung xử lý một bàn tay chính (tay phải hoặc trái). Chưa hỗ trợ các thao tác phối hợp hai tay (ví dụ: một tay giữ, một tay kéo) như trên màn hình cảm ứng đa điểm.

5.3. Hướng phát triển trong tương lai

Để hoàn thiện sản phẩm và nâng cao tính ứng dụng, các hướng phát triển tiếp theo được đề xuất bao gồm:

1. **Cải tiến Mô hình & Dữ liệu:**
    - Mở rộng bộ dữ liệu với các điều kiện ánh sáng khắc nghiệt hơn để tăng tính bền vững (Robustness).
    - Thử nghiệm các kiến trúc mạng mới nhẹ hơn như **GRU (Gated Recurrent Unit)** hoặc **Transformer** để so sánh hiệu năng và tốc độ.
2. **Nâng cấp tính năng ứng dụng:**

- **Tùy chỉnh cử chỉ (Customization):** Cho phép người dùng tự định nghĩa cử chỉ cho các hành động riêng biệt (ví dụ: Nắm tay để tắt máy, Xòe tay để mở trình duyệt).
- **Đa phương thức (Multimodal):** Kết hợp nhận diện cử chỉ tay với điều khiển bằng giọng nói (Voice Command) để tạo ra trải nghiệm tương tác toàn diện hơn.

3. **Triển khai phần cứng (Deployment):**
    - Đóng gói ứng dụng thành file thực thi (.exe) để người dùng phổ thông có thể sử dụng ngay mà không cần cài đặt Python.
    - Tối ưu hóa để chạy trên các thiết bị nhúng giá rẻ như **Raspberry Pi** hoặc **NVIDIA Jetson Nano**, biến nó thành một thiết bị ngoại vi độc lập ("Smart Camera") có thể cắm vào bất kỳ máy tính hoặc Smart TV nào để điều khiển.