

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN

**NGHIÊN CỨU CHẾ TẠO
XE LĂN ĐIỀU KHIỂN BẰNG MẮT**

TSV2023-125

Sinh viên Lương Đức Huy

Cần Thơ, 09/2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN

NGHIÊN CỨU CHẾ TẠO XE LĂN ĐIỀU KHIỂN BẰNG MẮT TSV2023-125

Sinh viên chủ nhiệm đề tài: Lương Đức Huy Nam, Nữ: Nam
Dân tộc: Kinh
Lớp, khoa: DI20Z6A1, Trường Công nghệ thông & Truyền thông
Năm thứ: 4/4,5:
Ngành học: Khoa học máy tính

Người hướng dẫn: PGS. TS. Phạm Nguyên Khang

Cần Thơ, 09/2023

DANH SÁCH THÀNH VIÊN THAM GIA NGHIÊN CỨU ĐỀ TÀI

HỌ VÀ TÊN	MSSV	NHIỆM VỤ
Lương Đức Huy	B2007184	Chủ nhiệm đề tài
Nguyễn Huy Cường	B2016949	Thành viên chính
Kim Minh Thắng	B2007210	Thành viên chính

ĐƠN VỊ PHỐI HỢP CHÍNH

- ✓ Trường Công nghệ thông tin & Truyền thông – Trường Đại học Cần Thơ
- ✓ Phòng Quản lý Khoa học – Trường Đại học Cần Thơ

MỤC LỤC

PHẦN 1: MỞ ĐẦU	13
1. TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU THUỘC LĨNH VỰC ĐỀ TÀI Ở TRONG VÀ NGOÀI NƯỚC	13
1.1. Trong nước	13
1.2. Ngoài nước	14
2. TÍNH CẤP THIẾT CỦA ĐỀ TÀI:	14
3. MỤC TIÊU ĐỀ TÀI	15
4. PHƯƠNG PHÁP NGHIÊN CỨU	15
5. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU	15
5.1. Đối tượng nghiên cứu	15
5.2. Phạm vi nghiên cứu	15
PHẦN 2: KẾT QUẢ NGHIÊN CỨU	16
CHƯƠNG I: TỔNG QUAN LÝ THUYẾT DEEP LEARNING	16
1. Giới thiệu về Deep Learning	16
2. Cách thức hoạt động của Deep Learning	16
3. Những vấn đề Deep Learning có thể giải quyết	17
4. Convolutional Neural Network (CNN) – Mạng neural tích chập	18
4.1. Giới thiệu	18
4.2. Cấu trúc mạng CNN	19
4.2.1. Lớp tích chập (Convolutional layer)	19
4.2.2. Lớp Pooling	24
4.2.3. Lớp kết nối đầy đủ (Fully Connected)	25
4.3. Mạng xương sống CNN (CNN backbone)	25
4.3.1. Mạng neural tích chập sâu AlexNet	26
4.3.2. Mạng phần dư ResNet	28
CHƯƠNG II: TỔNG QUAN LÝ THUYẾT THỊ GIÁC MÁY TÍNH	30
1. Giới thiệu thị giác máy tính	30
2. Camera Calibration	31
3. Phát hiện khuôn mặt và các điểm mốc khuôn mặt (face landmark)	33
3.1. MediaPipe – Phát hiện khuôn mặt	34
3.2. MediaPipe – Phát hiện các điểm mốc trên khuôn mặt	34
CHƯƠNG III: TỔNG QUAN LÝ THUYẾT VỀ ARDUINO	36
1. Giới thiệu	36
2. Cấu tạo của Arduino	36
3. Ứng dụng Arduino	37
CHƯƠNG IV: XÂY DỰNG CHƯƠNG TRÌNH VÀ CHẾ TẠO SẢN PHẨM	38
1. Huấn luyện mô hình	38

1.1. Giới thiệu tập dữ liệu.....	38
1.2. Tiền xử lý dữ liệu	39
1.3. Huấn luyện và đánh giá mô hình.....	39
2. Xây dựng module ước tính hướng mắt.....	43
3. Xây dựng chương trình điều khiển xe lăn	49
4. Chế tạo sản phẩm.....	53
4.1. Chế tạo xe lăn điện	53
4.2. Kết nối chương trình điều khiển xe lăn bằng hướng mắt với xe lăn điện	55
5. Kết quả và đánh giá	57
5.1. Kết quả.....	57
5.2. Đánh giá.....	57
PHẦN 3: KẾT LUẬN VÀ KIẾN NGHỊ.....	58
1. Kết quả đóng góp.....	58
1.1. Về khoa học và đào tạo.....	58
1.2. Về phát triển kinh tế	58
1.3. Về xã hội.....	58
2. Triển khai ứng dụng	58
3. Hướng phát triển.....	58
TÀI LIỆU THAM KHẢO	59
GIẤY PHÉP TẬP DỮ LIỆU.....	60

DANH MỤC HÌNH

Hình 1: Cách hoạt động của Deep Learning (http://nganhtrituenhantaohp.edu.vn/deep-learning-la-gi)	17
Hình 2: Cấu trúc mạng CNN (https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5).....	19
Hình 3: Ví dụ về lớp tích chập	20
Hình 4: Ví dụ về kernel làm sắc nét (https://viblo.asia/p/ung-dung-convolutional-neural-network-trong-bai-toan-phan-loai-anh-4dbZNg8ylYM).....	20
Hình 5: Ví dụ về kernel tăng cường biên độ (https://viblo.asia/p/ung-dung-convolutional-neural-network-trong-bai-toan-phan-loai-anh-4dbZNg8ylYM)	21
Hình 6: Ví dụ về kernel phát hiện biên độ (https://viblo.asia/p/ung-dung-convolutional-neural-network-trong-bai-toan-phan-loai-anh-4dbZNg8ylYM).....	21
Hình 7: Ví dụ về stride	22
Hình 8: Ví dụ về stride	22
Hình 9: Ví dụ về stride	23
Hình 10: Ví dụ về stride	23
Hình 11: Ví dụ về padding	23
Hình 12: Đồ thị hàm ReLU: $g(z)=\max(0,z)$ (https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks)	24
Hình 13: Cấu trúc mạng CNN (https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5).....	25
Hình 14: Biểu đồ phân bố kết quả sử dụng mạng CNN trên ImageNet (https://www.researchgate.net/figure/The-evolution-of-the-winning-entries-on-the-ImageNet-Large-Scale-Visual-Recognition_fig1_321896881)	26
Hình 15: Cấu trúc mạng AlexNet (https://d2l.ai/chapter_convolutional-modern/alexnet.html)	27
Hình 16: Sự khác nhau giữa khối thông thường (trái) và khối phần dư (phải) (https://d2l.ai/chapter_convolutional-modern/resnet.html)	28
Hình 17: Cấu trúc mạng ResNet-18 (https://www.researchgate.net/figure/Architecture-of-the-ResNet-18-model-used-in-this-study_fig3_354432343).....	29
Hình 18: Hình ảnh bị biến dạng (https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html)	32
Hình 19: Hình ảnh sau khi hiệu chỉnh (https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html)	33
Hình 20: Ví dụ về phát hiện khuôn mặt (https://developers.google.com/mediapipe/solutions/vision/face_detector)	34
Hình 21: Ví dụ về phát hiện các điểm mốc trên khuôn mặt (https://developers.google.com/mediapipe/solutions/vision/face_landmarker)	35
Hình 22: Cấu tạo của Arduino Uno R3 (https://mesidas.com/arduino-la-gi/).....	36
Hình 23: Tập dữ liệu MPIIFaceGaze	38
Hình 24: Kết quả đánh giá mô hình CNN trên tập MPIIFaceGaze (AlexNet).....	42
Hình 25: Kết quả đánh giá mô hình CNN trên tập MPIIFaceGaze (ResNet)	42
Hình 26: Sơ đồ hoạt động của module ước tính hướng mắt.....	43
Hình 27: Kết quả nhận diện khuôn mặt và các điểm mốc trên khuôn mặt.....	44
Hình 28: Kết quả ước tính tư thế đầu	45
Hình 29: Sáu điểm mốc quan trọng trên khuôn mặt (https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/appearance-based-gaze-estimation-in-the-wild)	45
Hình 30: Kết quả chuẩn hóa hình ảnh	47
Hình 31: Biểu diễn hướng nhìn trên hình ảnh đầu ra	49
Hình 32: Phát thảo giao diện chương trình điều khiển xe lăn	50

Hình 33: Sơ đồ hoạt động của chương trình điều khiển xe lăn	51
Hình 34: Giao diện hoàn chỉnh của chương trình điều khiển xe lăn	52
Hình 35: Hướng nhìn người dùng biểu diễn trên chương trình.....	52
Hình 36: Xe lăn Lucass X9 (https://www.moby.com.vn/xe-lan-tay-lucass-x-9-tieu-chuan_16899.html)	53
Hình 37: Động cơ giảm tốc 12V-50W	53
Hình 38: Bình ắc quy 12V-30Ah.....	54
Hình 39: Bộ nhông sên đĩa	54
Hình 40: Mạch điều khiển động cơ BTS7960 43A (https://nshopvn.com/product/mach-dieu-khien-dong-co-dc-bts7960-43a-1-dong-co).....	55
Hình 41: Mạch Arduino Uno R3 (https://nshopvn.com/product/arduino-uno-r3-dip-kem-cap/?variant=100977)	55
Hình 42: Sơ đồ hoạt động của xe lăn điều khiển bằng mắt	56
Hình 43: Sản phẩm sau khi hoàn thiện	56

DANH MỤC BẢNG BIỂU

Bảng 1: So sánh 2 kiểu Pooling.....	24
Bảng 2: Kết quả huấn luyện mô hình	41

DANH MỤC NHỮNG TỪ VIẾT TẮT

STT	Thuật ngữ	Diễn giải
1	CNN	Convolutional Neural Network
2	RNN	Recurrent Neural Network
3	ReLU	Rectified Linear Unit
5	ILSVRC	ImageNet Large Scale Visual Recognition Challenge
7	ResNet	Residual Networks
8	OpenCV	Open Computer Vision
9	NMS	non-maximum suppression

THÔNG TIN KẾT QUẢ NGHIÊN CỨU CỦA ĐỀ TÀI

1. Thông tin chung:

- Tên đề tài: Nghiên cứu chế tạo xe lăn điều khiển bằng mắt
- Sinh viên chủ nhiệm đề tài: Lương Đức Huy
- Lớp: Khoa học máy tính 01 K46
- Khoa: Trường Công nghệ thông tin & Truyền thông
- Năm thứ: 4 Số năm đào tạo: 4,5
- Người hướng dẫn: PGS. TS. Phạm Nguyên Khang

2. Mục tiêu đề tài:

Sử dụng công nghệ học sâu kết hợp với thị giác máy tính để xây dựng phần mềm nhận dạng hướng nhìn của mắt.

Sử dụng lập trình nhúng, lập trình Arduino để thiết kế mạch điều khiển nhận tín hiệu từ phần mềm nhận dạng hướng mắt đã xây dựng để điều khiển bánh xe.

Tích hợp phần mềm nhận dạng hướng nhìn của mắt và mạch điều khiển vào xe lăn truyền thống để điều khiển hướng di chuyển của xe lăn bằng mắt.

3. Tính mới và sáng tạo:

Ứng dụng công nghệ học sâu kết hợp thị giác máy tính vào việc nhận diện hướng nhìn của mắt.

Chế tạo sản phẩm xe lăn có thể vận hành không cần điều khiển bằng tay, mở rộng phạm vi hỗ trợ cho những người khuyết tật.

4. Kết quả nghiên cứu:

Tìm hiểu công nghệ học sâu và thị giác máy tính, xây dựng phần mềm nhận dạng hướng nhìn của mắt.

Tìm hiểu lập trình nhúng, lập trình Arduino, cách giao tiếp giữa hệ điều hành máy tính và mạch Arduino.

5. Sản phẩm:

TÊN SẢN PHẨM	SỐ LƯỢNG
Xe lăn điều khiển bằng mắt	01
Bản tin	01
Báo cáo tóm tắt	01
Video clips	01

6. Công bố khoa học từ kết quả nghiên cứu của đề tài, hoặc nhận xét, đánh giá của cơ sở đã áp dụng các kết quả nghiên cứu (nếu có): Không

7. Đóng góp về mặt kinh tế - xã hội, giáo dục và đào tạo, an ninh, quốc phòng và khả năng áp dụng của đề tài:

7.1. Về mặt kinh tế - xã hội:

Khi sản phẩm được ứng dụng vào thực tế sẽ góp một phần không nhỏ trong việc cải thiện đời sống của một bộ phận những người tàn tật, thiếu may mắn, giúp họ tự tin hòa nhập cuộc sống và có một cuộc sống trọn vẹn. Cải thiện được tinh thần của con người cũng đồng nghĩa cải thiện chất lượng sống của quốc gia, thế giới.

7.2. Về giáo dục và đào tạo:

Phát triển mở rộng phạm vi nghiên cứu về các lĩnh vực trí tuệ nhân tạo, thị giác máy tính.

7.3. Về khoa học và công nghệ có liên quan:

Thúc đẩy các sinh viên có thêm động lực làm nghiên cứu khoa học, góp phần làm đa dạng hóa các công trình nghiên cứu của Đại học Cần Thơ.

7.4. Về khả năng áp dụng của đề tài:

Ngoài xe lăn, có thể áp dụng kết quả nhận diện hướng mắt của đề tài để áp dụng điều khiển các thiết bị, sản phẩm khác. Ví dụ như: máy tính, điện thoại, TV,... Giúp những người khuyết tật có thể điều khiển các thiết bị xung quanh mà chỉ cần dùng mắt, giúp cuộc sống những người khuyết tật tiện lợi và cải thiện hơn.

8. Hiệu quả, phương thức chuyển giao kết quả nghiên cứu và khả năng áp dụng:

8.1. Phương thức chuyển giao:

Chuyển giao trực tiếp sản phẩm cho Trường Công nghệ Thông tin & Truyền thông – Trường Đại học Cần Thơ quản lý và sử dụng.

8.2. Địa chỉ ứng dụng:

Trường Công nghệ Thông tin & Truyền thông - Trường Đại học Cần Thơ.
Khu 2, đường 3/2, Phường Xuân Khánh, Q. Ninh Kiều, TP. Cần Thơ, Việt Nam.

8.3. Hiệu quả, khả năng áp dụng:

Sản phẩm có thể triển khai thí nghiệm tại các trung tâm người khuyết tật. Việc khai thác và sử dụng ứng dụng sẽ giúp người kém may mắn có cơ hội tiếp cận đến các sản phẩm thông minh giúp cuộc sống dễ dàng hơn.

Ngày tháng năm

Sinh viên chịu trách nhiệm chính

thực hiện đề tài

(ký, họ và tên)

Lương Đức Huy

[illegible]

(ký tên và đóng dấu)

Người hướng dẫn

(*ký, họ và tên*)

10

THÔNG TIN VỀ SINH VIÊN
CHỊU TRÁCH NHIỆM CHÍNH THỰC HIỆN ĐỀ TÀI

I. SƠ LƯỢC VỀ SINH VIÊN:

Họ và tên: Lương Đức Huy

Sinh ngày: 08 tháng 05 năm 2002

Nơi sinh: Cần Thơ

Lớp: Khoa học máy tính 01

Khóa: 46

Khoa: Trường Công nghệ thông tin & Truyền thông

Địa chỉ liên hệ: 132/29, 3/2, Phường Hưng Lợi, Quận Ninh Kiều, Tp. Cần Thơ

Điện thoại: 0358 669 057

Email: huyb2007184@student.ctu.edu.vn

Ảnh 4x6

II. QUÁ TRÌNH HỌC TẬP (kê khai thành tích của sinh viên từ năm thứ 1 đến năm đang học):

*** Năm thứ 1:**

Ngành học: Khoa học máy tính Khoa: Trường Công nghệ thông tin & Truyền thông

Kết quả xếp loại học tập: HKI : 3.19 (Khá), HKII : 3.90 (Xuất sắc)

Sơ lược thành tích:

+ Đạt danh hiệu Đoàn viên ưu tú.

*** Năm thứ 2:**

Ngành học: Khoa học máy tính Khoa: Trường Công nghệ thông tin & Truyền thông

Kết quả xếp loại học tập: HKI : 3.88 (Xuất sắc), HKII : 3.50 (Giỏi)

Sơ lược thành tích:

+ Giấy khen Đoàn khoa “Đã có nhiều thành tích đóng góp trong công tác Đoàn và phong trào thanh niên Khoa CNTT & TT 2020-2021”.

+ Giấy khen Đoàn Trường “Đã có nhiều thành tích trong công tác Đoàn và phong trào thanh niên nhiệm kỳ 2019 – 2022”.

*** Năm thứ 3:**

Ngành học: Khoa học máy tính Khoa: Trường Công nghệ thông tin & Truyền thông

Kết quả xếp loại học tập: HKI : 3.80 (Xuất sắc), HKII : 3.50 (Giỏi)

Sơ lược thành tích:

+ Giấy khen Đoàn khoa “Đã có nhiều thành tích đóng góp trong công tác Đoàn và phong trào thanh niên năm học 2021-2022”.

+ Giấy khen Đoàn khoa “Đã có nhiều đóng góp tích cực trong lớp Tập huấn cán bộ đoàn trường Công nghệ Thông tin & Truyền Thông năm 2023”.

- + Giấy khen Đoàn khoa “Đã có đóng góp xuất sắc trong phong trào và hoạt động Đoàn năm học 2022-2023”.
- + Giấy khen Đoàn Trường “Có thành tích xuất sắc trong công tác Đoàn và phong trào thanh niên năm học 2021-2022”.

**Xác nhận của Trường Đại học Cần
Thơ**
(ký tên và đóng dấu)

Ngày tháng năm
**Sinh viên chịu trách nhiệm chính
thực hiện đề tài**
(ký, họ và tên)

Lương Đức Huy

PHẦN 1: MỞ ĐẦU

1. TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU THUỘC LĨNH VỰC ĐỀ TÀI Ở TRONG VÀ NGOÀI NƯỚC

1.1. Trong nước

Trong những năm gần đây, việc ứng dụng các công nghệ học sâu, thị giác máy tính để nghiên cứu và chế tạo các sản phẩm thông minh phục vụ đời sống con người đang dần được chú ý và trở thành xu hướng ở Việt Nam. Một trong số đó là các dự án nghiên cứu xe lăn thông minh được các nhóm nghiên cứu từ các đơn vị, trường khác nhau thực hiện nhằm mục đích chế tạo ra loại xe lăn tiện dụng và an toàn với người sử dụng, góp phần giúp đỡ những người kém may mắn có cơ hội hòa nhập với cuộc sống, có một cuộc sống thuận tiện hơn. Đã có nhiều nghiên cứu chế tạo ra các xe lăn thông minh có thể kể đến như: xe lăn điều khiển bằng cử chỉ đầu, điều khiển bằng tay sử dụng cần điều khiển,...

Xe lăn điện điều khiển bằng cần điều khiển và các động cơ gắn ở 2 bên bánh xe giúp người bệnh điều khiển xe bằng cách gạt cần điều khiển tới những hướng cần đi. Tuy nhiên đối với những bệnh nhân không thể sử dụng tay thì không thể sử dụng được sản phẩm này. Vì thế đã có thêm một số nghiên cứu với góc nhìn khác đó là thiết kế xe lăn điều khiển bằng cử chỉ đầu, trong thiết kế này người sử dụng sẽ phải đội một bộ cảm biến ở trên đầu để xác định được cử chỉ đầu nhằm điều khiển xe theo ý của người sử dụng. Tuy nhiên thiết kế này có một khuyết điểm là sẽ gây khó chịu cho người sử dụng khi phải đội bộ cảm biến trên đầu trong thời gian dài, bên cạnh đó còn có một số bệnh nhân không thể sử dụng cả tay và đầu thì sản phẩm này vẫn chưa là giải pháp tối ưu để giúp những người kém may mắn dễ dàng hòa nhập với cuộc sống.

Để giải quyết những vấn đề còn tồn đọng, một số nghiên cứu đã chuyển hướng sang việc điều khiển bằng mắt, giúp mở rộng phạm vi sử dụng cho những người bị liệt cả tay và đầu. Một số công trình nghiên cứu có thể kể đến:

- Nghiên cứu của kỹ sư Nguyễn Hữu Cường^[1] sáng chế xe lăn điều khiển bằng cử động mắt. Trong sáng chế này kỹ sư Cường sử dụng kỹ thuật xác định hướng hình của mắt dựa trên ảnh thu được từ webcam đeo vào đầu.

- Nghiên cứu của PGS. TS. Huỳnh Thái Hoàng – Trường Đại học Bách Khoa TPHCM^[2] thiết kế và thực hiện xe lăn điện điều khiển bằng mắt. Trong thiết kế này tác giả sử dụng webcam thông thường gắn cố định vào khung xe để thu ảnh mặt người. Từ ảnh mặt người thu được sẽ sử dụng để xử lý hướng nhìn của mắt.

Nhìn chung các đề tài nghiên cứu về xe lăn điện thông minh trong nước khá đa dạng nhưng đa phần hướng tới các phương pháp như điều khiển bằng cử chỉ đầu, điều

khởi bằng giọng nói,... Chưa có nhiều đề tài hướng đến phương pháp điều khiển bằng hướng mắt.

1.2. Ngoài nước

Trong khoảng từ những năm 2000 đến nay có rất nhiều nghiên cứu chế tạo xe lăn điện thông minh dành cho người khuyết tật trên thế giới. Các hướng và phương pháp nghiên cứu cũng rất đa dạng như điều khiển bằng giọng nói, điều khiển bằng suy nghĩ, điều khiển bằng cử chỉ đầu,...trong đó phương pháp điều khiển bằng mắt cũng là một trong số các phương pháp thu được nhiều sự chú ý. Một số công trình nghiên cứu sử dụng các phương pháp trên có thể kể đến như:

- Aleksandar Pajkanović và Branko Dokić^[3] đã nghiên cứu chế tạo xe lăn điều khiển bằng cử chỉ đầu.

- Anoop.K.J và các cộng sự^[4] đã nghiên cứu chế tạo xe lăn điều khiển bằng giọng nói.

- Ehtisham Shahid, Muhammad Zain Ur Rehman, Muhammad Sheraz, Wasim Khan và Muhammad Aqib^[5] đã nghiên cứu chế tạo xe lăn điều khiển bằng chuyển động mắt.

- Anagha Dwajan B, Sowmya M S, Bhavani S và Usha M R Priya Reddy N^[6] đã nghiên cứu chế tạo xe lăn điều khiển bằng hướng mắt.

Nhìn chung các chủ đề nghiên cứu về xe lăn điện thông minh luôn là một vấn đề được chú ý cao trên thế giới. Một trong những phương pháp chế tạo được hướng tới nhiều nhất trong các nghiên cứu là điều khiển bằng mắt.

2. TÍNH CẤP THIẾT CỦA ĐỀ TÀI:

Sự phát triển mạnh mẽ của trí tuệ nhân tạo, thị giác máy tính ở Việt Nam trong các năm gần đây đã thúc đẩy các công trình nghiên cứu áp dụng các công nghệ trên để chế tạo ra các sản phẩm thông minh ngày càng phổ biến và mang tính cấp thiết hơn. Các lĩnh vực nghiên cứu được áp dụng từ các công nghệ này cũng đa dạng hơn đặc biệt là các lĩnh vực liên quan đến việc kết hợp giữa học sâu và xử lý ảnh để nhận dạng, phân loại. Ví dụ như phát hiện trộm thông qua nhận dạng hành động bất thường, mở khóa điện thoại thông qua nhận diện khuôn mặt,...

Việc ứng dụng học sâu và thị giác máy tính để chế tạo xe lăn điều khiển bằng mắt đang là vấn đề luôn được nghiên cứu và cải thiện liên tục trong và ngoài nước. Hiện nay các mẫu xe lăn điện điều khiển bằng tay sử dụng cần điều khiển có giá thành cao, không có tính thực tiễn cao với đại bộ phận người khuyết tật, hoặc những nghiên cứu chế tạo xe lăn thông minh điều khiển bằng cử chỉ đầu cũng gặp vấn đề tương tự, sẽ không thực tiễn với những người liệt toàn thân, không cử động được các chi để điều khiển. Chính

vì vậy, việc nghiên cứu chế tạo ra xe lăn điều khiển bằng mắt sẽ giải quyết được các vấn đề đang gặp phải cũng như mang tính thực tiễn cao hơn, nhằm giúp cuộc sống của những người kém may mắn được cải thiện, dễ hòa nhập và có một cuộc sống trọn vẹn hơn.

3. MỤC TIÊU ĐỀ TÀI

Sử dụng công nghệ học sâu kết hợp với thị giác máy tính để xây dựng phần mềm nhận dạng hướng nhìn của mắt.

Sử dụng lập trình nhúng, lập trình Arduino để thiết kế mạch điều khiển nhận tín hiệu từ phần mềm nhận dạng hướng mắt đã xây dựng để điều khiển bánh xe.

Tích hợp phần mềm nhận dạng hướng nhìn của mắt và mạch điều khiển vào xe lăn truyền thống để điều khiển hướng di chuyển của xe lăn bằng mắt.

4. PHƯƠNG PHÁP NGHIÊN CỨU

- Lên ý tưởng, viết đặc tả.
- Thiết kế, xây dựng, cài đặt, kiểm thử, thực nghiệm.
- Chỉnh sửa, tối ưu độ chính xác của phần mềm.
- Triển khai thành sản phẩm, kiểm thử và viết báo cáo.

5. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

5.1. Đối tượng nghiên cứu

Đối tượng hướng đến của đề tài là trí tuệ nhân tạo nói chung và công nghệ học sâu, thị giác máy tính nói riêng, ngôn ngữ lập trình Python, các thư viện trong Python hỗ trợ trong việc học sâu như Pytorch hay hỗ trợ trong thị giác máy tính như MediaPipe, các mô hình học sâu tiên tiến như mạng neural tích chập (CNN), mạng thần kinh hồi quy (RNN), lập trình nhúng. Hỗ trợ những người liệt toàn thân, cụt tay,... không thể sử dụng các hệ thống điều khiển như cần điều khiển, cử chỉ đầu,...

5.2. Phạm vi nghiên cứu

Kết hợp công nghệ học sâu và thị giác máy tính để xây dựng phần mềm nhận dạng hướng mắt.

Ứng dụng lập trình nhúng để làm trung gian truyền thông tin xử lý điều khiển giữa phần mềm nhận dạng hướng mắt với mạch điều khiển xe.

PHẦN 2: KẾT QUẢ NGHIÊN CỨU

CHƯƠNG I: TỔNG QUAN LÝ THUYẾT DEEP LEARNING

1. Giới thiệu về Deep Learning

Deep Learning là một phần quan trọng của lĩnh vực trí tuệ nhân tạo, là một phương pháp học máy mạnh mẽ được thúc đẩy bởi kiến thức về cấu trúc và hoạt động của não người, được xem là một lĩnh vực con của Machine Learning. Nó được thiết kế để mô hình hóa và giải quyết các vấn đề phức tạp bằng cách sử dụng các mạng neural nhân tạo sâu (Deep Neural Networks). Điều đặc biệt về Deep Learning là khả năng tự học cấu trúc và đặc điểm của dữ liệu thông qua việc tự điều chỉnh các trọng số và tham số của mạng neural.

Một điểm đặc biệt của Deep Learning là khả năng xử lý dữ liệu không cấu trúc, như hình ảnh, âm thanh, văn bản và video. Bằng cách sử dụng các lớp neural ẩn trong mạng, Deep Learning có khả năng tự động học lên các đặc trưng phức tạp từ dữ liệu đầu vào. Điều này đã tạo ra những đột phá lớn trong nhiều lĩnh vực, bao gồm nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự đoán chuỗi thời gian và nhiều ứng dụng khác.

Các mô hình Deep Learning như Convolutional Neural Network (CNN) và Recurrent Neural Network (RNN) đã đạt được những thành tựu ấn tượng trong việc giải quyết các vấn đề phức tạp. Chẳng hạn, trong lĩnh vực nhận dạng hình ảnh, Deep Learning đã giúp máy tính nhận biết vật thể, khuôn mặt và thậm chí đạt độ chính xác cao hơn cả con người.

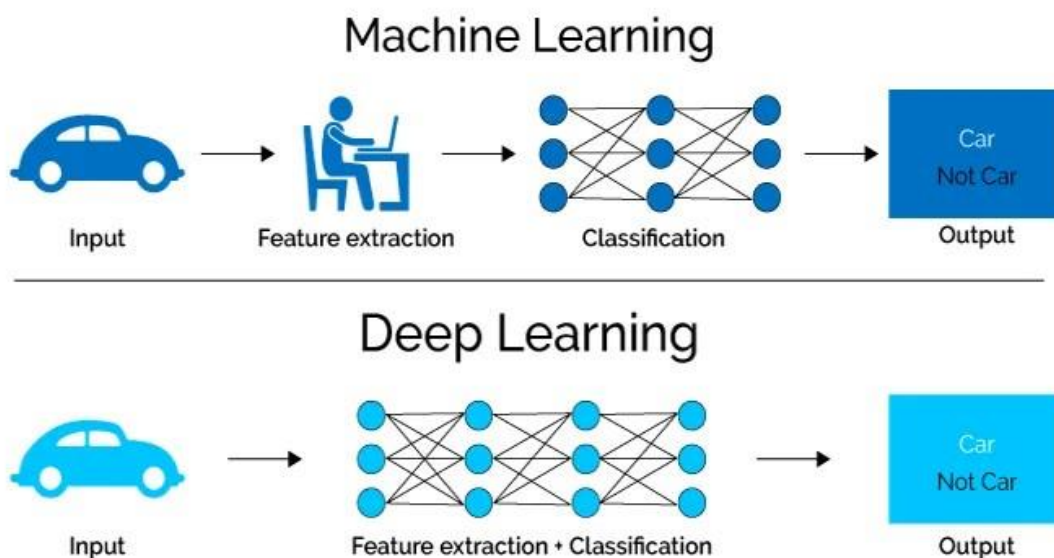
Tuy Deep Learning mang lại nhiều lợi ích, nhưng cũng đòi hỏi tài nguyên tính toán lớn, đặc biệt là khi xử lý các mạng neural sâu và dữ liệu lớn. Điều này đã thúc đẩy sự phát triển của các phần cứng và công nghệ mới để hỗ trợ việc huấn luyện và triển khai các mô hình Deep Learning.

Tóm lại, Deep Learning đang là một lĩnh vực đóng vai trò quan trọng trong cách chúng ta tiếp cận và giải quyết các vấn đề phức tạp sử dụng máy tính. Nhờ khả năng tự học và mô hình hóa dữ liệu phức tạp, nó đã đưa ra những giải pháp sáng tạo cho nhiều lĩnh vực và tiếp tục định hình tương lai của trí tuệ nhân tạo.

2. Cách thức hoạt động của Deep Learning

Deep Learning sử dụng mạng neural nhân tạo được xây dựng để mô phỏng khả năng tư duy của não bộ con người. Một mạng sẽ bao gồm nhiều lớp (layer) khác nhau, số lượng lớp càng nhiều mạng sẽ càng sâu. Trong từng lớp sẽ chứa các nút mạng (node) và các lớp được liên kết kề với nhau. Mỗi kết nối giữa các nút mạng chứa một trọng số tương ứng, trọng số này càng cao thì ảnh hưởng của kết nối này đến mạng neural càng lớn.

Mỗi neural chứa một hàm kích hoạt có nhiệm vụ chuẩn hóa đầu ra từ neural này. Dữ liệu được đưa vào mạng sẽ đi qua tất cả các lớp và trả về kết quả ở lớp cuối cùng, gọi là output layer.



Hình 1: Cách hoạt động của Deep Learning
<http://nganhtrituenhantaohp.edu.vn/deep-learning-la-gi>

Để dễ hình dung về Deep Learning, thông qua ví dụ sau:

Làm sao để nhận biết hình nào là hình vuông trong một tập hình? Để giải quyết vấn đề này điều cần thiết đầu tiên là kiến thức tối thiểu về hình vuông, cụ thể là các đặc trưng về hình vuông khác biệt với các dạng hình khối khác. Khi đã có đủ điều kiện cần, bắt đầu kiểm tra:

- Bước 1: Hình này có 4 cạnh không?
- Bước 2: Nếu hình này có 4 cạnh thì 4 cạnh này có kết nối với nhau không?
- Bước 3: Nếu hình này có 4 cạnh kết nối với nhau, vậy các góc giữa 4 cạnh này có vuông không?
- Bước 4: Nếu hình này có 4 cạnh, vuông góc giữa các cạnh, vậy kích thước của 4 cạnh này có bằng nhau không?

Thông qua việc đi từng bước trong 4 bước cơ bản trên sẽ trả lời được hình nào là hình vuông. Deep Learning cũng hoạt động tương tự như vậy, nhưng ở một quy mô lớn hơn như nhận diện vật thể, nhận diện con người, ước tính khoảng cách,...

3. Những vấn đề Deep Learning có thể giải quyết

Một số vấn đề Deep Learning đã giải quyết được trong thực tế có thể kể đến:

- **Xe tự lái:** Hiện nay một số mẫu xe tự lái đã được phép vận hành, các công nghệ tự lái sẽ được xây dựng dựa trên các mạng neural cao cấp, mô hình Deep Learning

được sử dụng sẽ nhận diện các đối tượng, vật thể ở môi trường xung quanh xe, tính toán khoảng cách giữa xe so với các đối tượng, vật thể hay phương tiện xung quanh khác. Thêm vào đó là xác định làn đường, tín hiệu giao thông,... từ đó mô hình sẽ đưa ra các quyết định tối ưu và nhanh chóng nhất có thể. Một trong các hãng xe đi đầu trong việc sản xuất xe tự lái hiện nay chính là Tesla.

- **Phân tích cảm xúc:** Deep Learning cũng có thể phân tích cảm xúc con người thông qua việc xử lý ngôn ngữ tự nhiên, phân tích văn bản và thống kê. Các mô hình này có thể được ứng dụng để phán đoán cảm xúc khách hàng thông qua các bình luận, đánh giá, tweet,... từ đó đưa ra những chiến lược kinh doanh và marketing phù hợp cho từng nhóm đối tượng cụ thể.
- **Trợ lý ảo:** Trợ lý ảo hiện đang rất phổ biến trên toàn cầu, chúng được tích hợp trực tiếp vào điện thoại, máy tính, hay robot. Một trong số đó có thể kể đến như Cortana, Siri, Google Assistant, robot Vector,... Các trợ lý ảo này được xây dựng dựa trên Deep Learning với các thuật toán nhận diện văn bản, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên,... từ đó phân tích và đưa ra kết quả tối ưu cho người dùng.
- **Mạng xã hội:** Từ lâu các trang mạng xã hội như Facebook, Zalo, Tiktok, Twitter,.. đã trở lên quá phổ biến trong thời đại công nghệ 4.0. Các trang mạng này cũng áp dụng các thuật toán Deep Learning để cải thiện dịch vụ của mình. Một số có thể kể đến như nhận diện khuôn mặt, khi một tấm hình được đăng tải lên, các mô hình sẽ được áp dụng để nhận diện các khuôn mặt xuất hiện trong hình là ai dựa vào các dữ liệu về hình ảnh được tải lên trước đó. Thông qua các dữ liệu về tìm kiếm, cảm xúc, bình luận của người dùng. Các trang mạng này cũng phân tích và đề xuất, gợi ý các sản phẩm, mặt hàng liên quan phù hợp thông qua các sản phẩm mại điện tử có liên kết.

Ngoài ra còn rất nhiều phương diện trong cuộc sống được áp dụng Deep Learning trên toàn thế giới. Sự phát triển của Deep Learning đã giải quyết được nhiều vấn đề liên quan, từ đó cải thiện được chất lượng sống, thời gian, nhân lực,...

4. Convolutional Neural Network (CNN) – Mạng neural tích chập

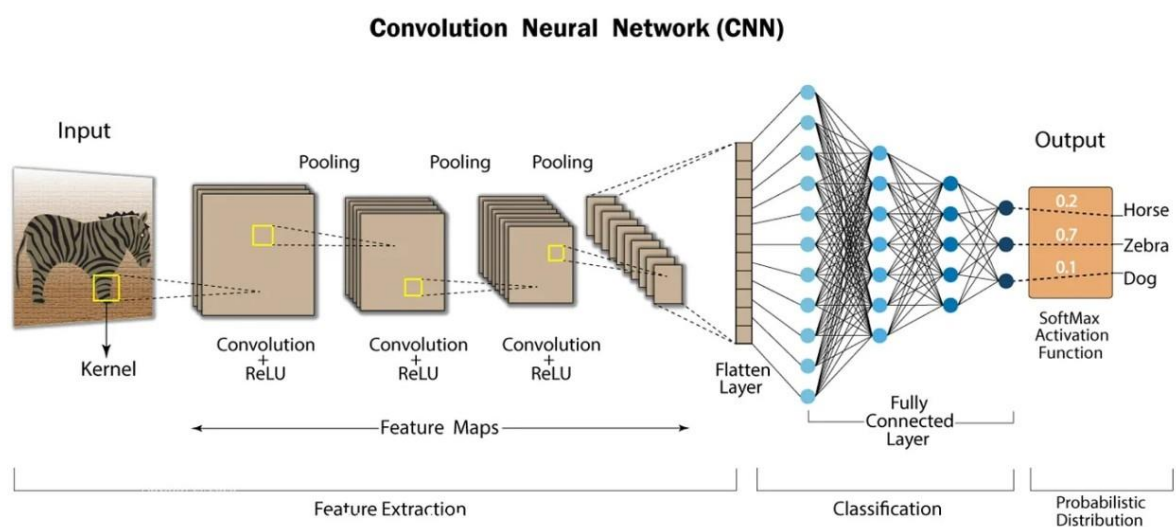
4.1. Giới thiệu

Convolutional Neural Network (CNN) là một loại mạng neural nhân tạo được thiết kế đặc biệt để xử lý và phân loại hình ảnh. CNN đã đạt được sự phát triển đáng kể trong lĩnh vực thị giác máy tính và xử lý hình ảnh, và chúng đã góp phần quan trọng vào sự thành công của nhiều ứng dụng thời gian thực như nhận dạng khuôn mặt, xe tự lái, và phát hiện đối tượng. CNN được lấy cảm hứng từ cách mà não người xử lý hình ảnh thông qua các tầng thụ động và tiếp thu thần kinh.

CNN đã đạt được kết quả ấn tượng trong nhiều cuộc thi và ứng dụng thực tế như phát hiện vật thể, phân loại hình ảnh y tế và thậm chí cả trong tự động đặt tên cho hình ảnh. Chúng đã trở thành một công cụ quan trọng trong lĩnh vực trí tuệ nhân tạo và thị giác máy tính, giúp chúng ta hiểu và xử lý hình ảnh một cách thông minh và hiệu quả.

4.2. Cấu trúc mạng CNN

Mạng CNN là một tập các lớp tích chập chồng lên nhau và sử dụng các hàm phi tuyến như ReLU và tanh để kích hoạt các trọng số trong các nút mạng. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.



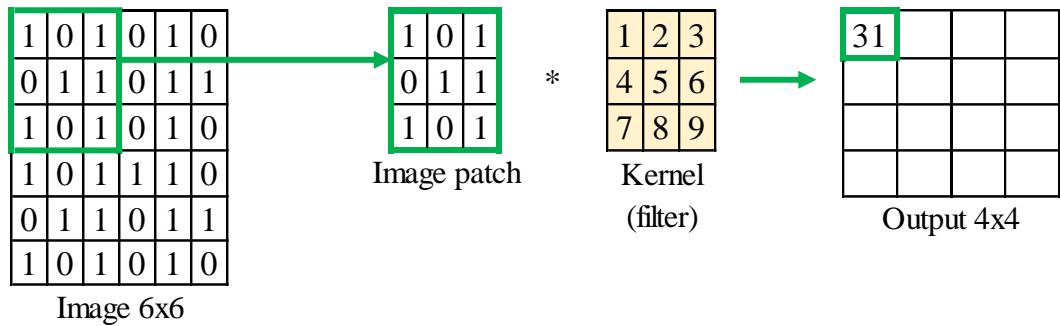
Hình 2: Cấu trúc mạng CNN

(<https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5>)

Số lượng các lớp tích chập sẽ tùy thuộc vào người xây dựng và sự phức tạp của dữ liệu đầu vào. Một mạng CNN cơ bản thường sẽ có 3 lớp chính: lớp tích chập (Convolutional layer), lớp pooling và lớp kết nối đầy đủ (Fully connected).

4.2.1. Lớp tích chập (Convolutional layer)

Lớp này sử dụng các bộ lọc để thực hiện phép tích chập khi đưa qua đầu vào theo các chiều của nó. Cụ thể, với đầu vào là một hình ảnh, được xem như một ma trận dữ liệu, lớp này sẽ là một ma trận quét qua ma trận dữ liệu, sau đó đưa vào các hàm kích hoạt phi tuyến như ReLU để thu lại một ma trận mới được gọi là feature map. Các siêu tham số của bộ lọc này là kích thước bộ lọc và bước nhảy (stride).

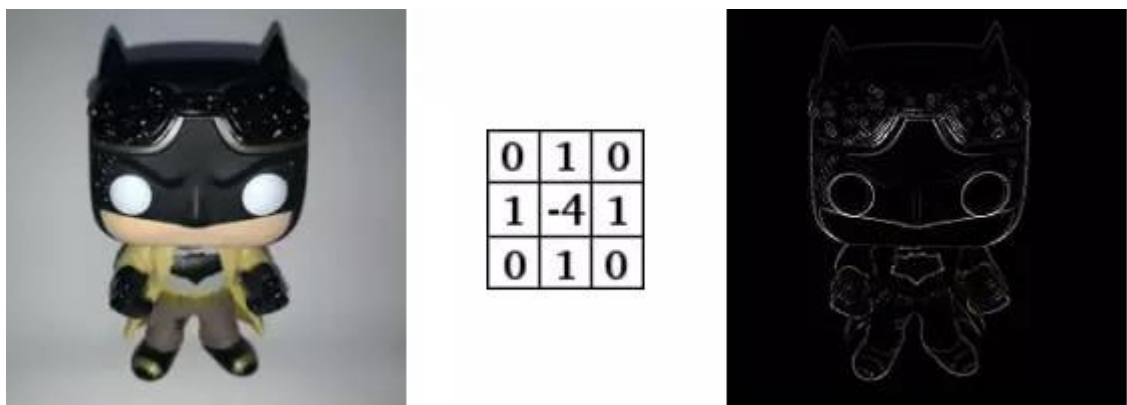


Hình 3: Ví dụ về lớp tích chập

Giả sử đầu vào là một hình ảnh với ma trận dữ liệu tương đương kích thước 6x6. Với kernel là ma trận quét kích thước 3x3, ma trận quét sẽ quét lần lượt theo bước nhảy (stride) trên ma trận dữ liệu. Mỗi lần quét sẽ lấy được một ma trận con (Image patch) trong ma trận dữ liệu có kích thước bằng với kích thước kernel. Sau đó tiến hành tích chập ma trận con với ma trận quét và trả về kết quả cho ma trận đầu ra kích thước 4x4. Ví dụ với minh họa trên ta có:

$$\begin{aligned}
 \text{Output}[0][0] &= (1 \times 1) + (0 \times 2) + (1 \times 3) + (0 \times 4) + (1 \times 5) + (1 \times 6) + (1 \times 7) + (0 \times 8) + (1 \times 9) \\
 &= 1 + 0 + 3 + 0 + 5 + 6 + 7 + 0 + 9 \\
 &= 31
 \end{aligned}$$

Các giá trị trong ma trận kernel không cố định, tùy vào mục đích trích xuất đặc tính hình ảnh để tìm các giá trị này. Ví dụ với 3 kernel bên dưới, khi tích chập sẽ cho ra 3 kết quả với đặc tính riêng biệt từ input ban đầu:

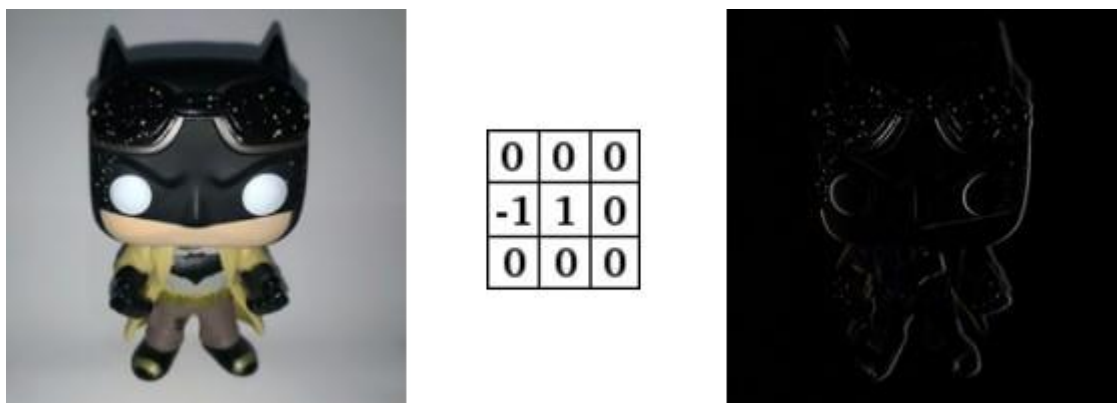


Hình 4: Ví dụ về kernel làm sắc nét

(<https://viblo.asia/p/ung-dung-convolutional-neural-network-trong-bai-toan-phan-loai-anh-4dbZNg8ylYM>)

Convolution Sharpen (tạm dịch: Làm sắc nét bằng tích chập): là một kỹ thuật xử lý hình ảnh thường được sử dụng để tăng cường độ tương phản và sắc nét của

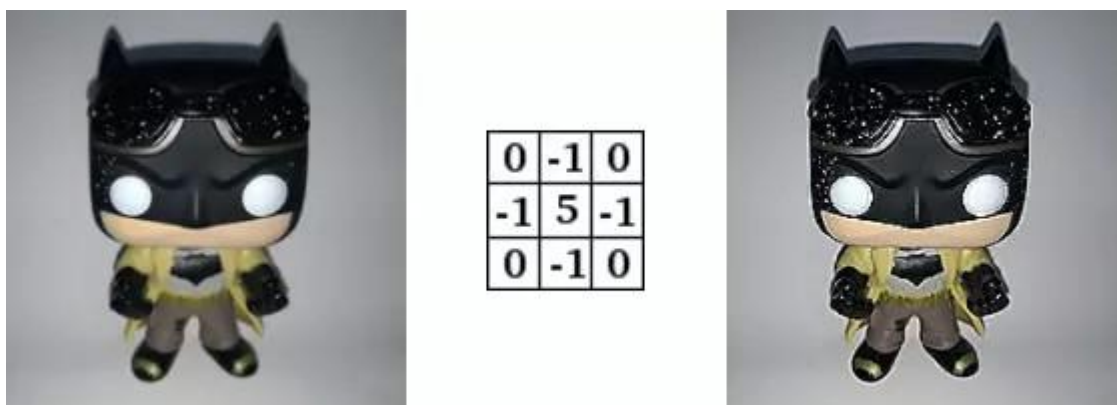
một hình ảnh. Kernel được áp dụng có khả năng tạo ra một hiệu ứng tăng cường cạnh và độ tương phản của hình ảnh.



Hình 5: Ví dụ về kernel tăng cường biên độ

(<https://viblo.asia/p/ung-dung-convolutional-neural-network-trong-bai-toan-phan-loai-anh-4dbZNg8ylYM>)

Convolution Edge Enhance (tạm dịch: Tăng cường biên độ bằng tích chập): đây là kỹ thuật xử lý hình ảnh dùng để tạo ra hiệu ứng tăng cường cạnh và biên độ trong hình ảnh. Kernel của Convolution Edge Enhance được thiết kế để tôn lên sự khác biệt giữa các vùng có độ tương phản cao và thấp, tạo ra hiệu ứng tăng cường cạnh.



Hình 6: Ví dụ về kernel phát hiện biên độ

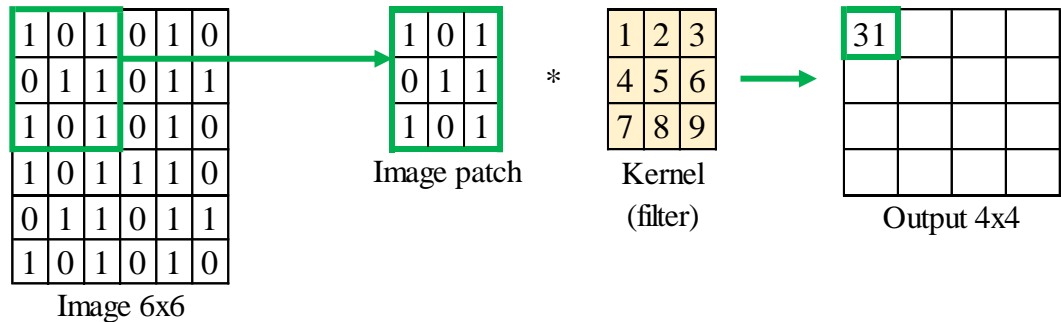
(<https://viblo.asia/p/ung-dung-convolutional-neural-network-trong-bai-toan-phan-loai-anh-4dbZNg8ylYM>)

Convolution Edge Detect (tạm dịch: Phát hiện biên độ bằng phép tích chập) là một kỹ thuật xử lý hình ảnh dùng để tìm và làm nổi bật các biên hoặc đường nét trong hình ảnh. Kỹ thuật này sử dụng phép tích chập và một kernel được thiết kế đặc biệt để xác định các vùng có sự thay đổi đột ngột trong độ sáng hoặc màu sắc, coi chúng như là các đường biên. Khi kernel này được áp dụng lên hình ảnh, nó sẽ

tạo ra một hình ảnh mới trong đó các biên sẽ được biểu thị bằng các điểm sáng hoặc tối tùy thuộc vào hướng thay đổi độ sáng hay màu sắc.

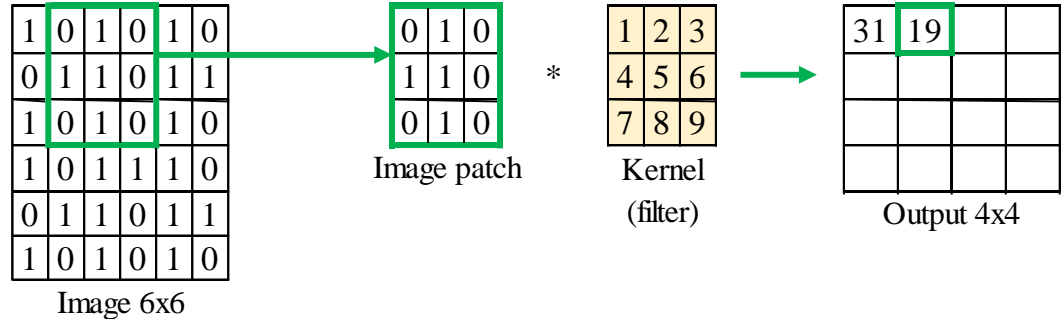
Các siêu tham số:

- **Stride**: là sai bước hay độ trượt của kernel khi quét qua ma trận dữ liệu đầu vào. Giả sử với Stride(1, 1), khi kernel quét qua ma trận đầu vào theo chiều ngang, mỗi lần trượt sẽ dịch kernel sang một ô, lần lượt như vậy cho đến cuối. Lúc này kernel sẽ trở lại vị trí ban đầu nhưng dịch xuống một ô và tiếp tục quét theo chiều ngang với Stride = 1. Quay lại ví dụ ban đầu:



Hình 7: Ví dụ về stride

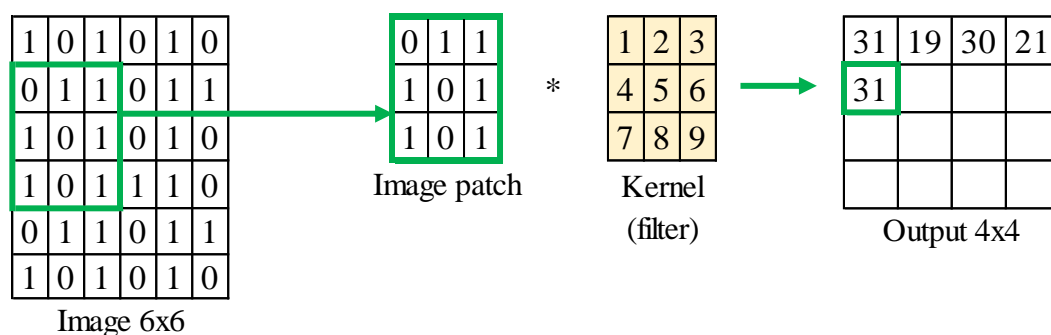
Sau khi kernel tích chập với ma trận con 3x3 từ ma trận đầu vào 6x6 và trả về kết quả cho ma trận đầu ra 4x4: $\text{Output}[0][0] = 31$.



Hình 8: Ví dụ về stride

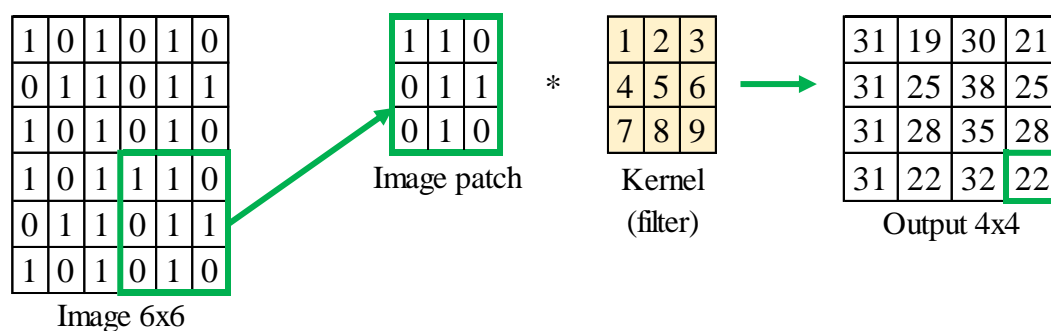
Trượt kernel sang phải một ô để lấy ma trận con tiếp theo, tiếp tục tích chập với kernel và trả về kết quả cho ma trận đầu ra:

$$\begin{aligned}
 \text{Output}[0][1] &= (0 \times 1) + (1 \times 2) + (0 \times 3) + (1 \times 4) + (1 \times 5) + (0 \times 6) + (0 \times 7) + (1 \times 8) + (0 \times 9) \\
 &= 0 + 2 + 0 + 4 + 5 + 0 + 0 + 8 + 0 \\
 &= 19
 \end{aligned}$$



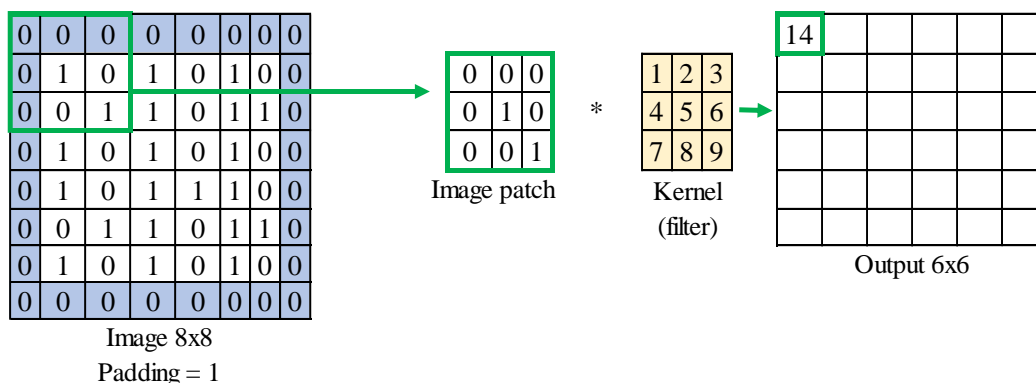
Hình 9: Ví dụ về stride

Sau khi trượt kernel đến cuối tương đương với hàng đầu của ma trận đầu ra. Kernel sẽ trở về vị trí ban đầu nhưng trượt xuống một ô và tiếp tục tích chập theo chiều ngang để trả về kết quả cho hàng thứ hai của ma trận đầu ra. Liên tục quét kernel qua ma trận đầu vào cho đến hết sẽ thu được ma trận đầu ra đầy đủ:



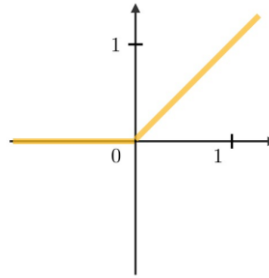
Hình 10: Ví dụ về stride

- Padding: Nếu chọn stride và kích thước của kernel càng lớn thì kích thước của ma trận đầu ra càng nhỏ, một vấn đề rắc rối có thể xảy ra là mất một số điểm ảnh trên biên của ảnh, sự mất mát này có thể tích lũy dần ra nhiều lớp tích chập. Và có một giải pháp cho vấn đề này là Padding, ta sẽ chèn các điểm ảnh bọc xung quanh đường biên của ảnh ảnh đầu vào, nhờ đó làm tăng kích thước sử dụng của bức ảnh. Ví dụ với Padding = 1, tiến hành bọc các điểm ảnh với giá trị thêm vào (thường là 0) xung quanh biên của ma trận ảnh đầu vào như sau:



Hình 11: Ví dụ về padding

Các lớp tích chập có tác dụng trích xuất các đặc trưng của hình ảnh đầu vào, sau khi đi qua các lớp tích chập, các thông tin được trích xuất từ các lớp tích chập sẽ được đưa vào các hàm kích hoạt phi tuyến để tạo tính phi tuyến và không gian trước khi đi đến các lớp tiếp theo. Nếu không có các hàm kích hoạt phi tuyến này, dù mạng neural có nhiều lớp đến đâu vẫn chỉ có hiệu quả như một lớp tuyến tính. Một số hàm phi tuyến được sử dụng trong CNN như ReLU, tanh, sigmoid,... Trong đó ReLU thường được sử dụng vì nó có hiệu suất tốt.



Hình 12: Đồ thị hàm ReLU: $g(z)=\max(0,z)$

(<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>)

4.2.2. Lớp Pooling

Lớp Pooling thường được sử dụng sau lớp tích chập và hàm kích hoạt phi tuyến, lớp này là một phép downsampling, giúp tăng tính bất biến không gian, giảm kích thước đầu vào và chi phí tính toán. Trong đó, max pooling và average pooling là những dạng pooling đặc biệt, hay được sử dụng (thường là max pooling). Cách thức hoạt động của lớp này gần giống với lớp tích chập. Cụ thể, lớp này sử dụng ma trận quét với stride tương tự như lớp tích chập, nhưng mỗi lần quét qua sẽ lấy giá trị lớn nhất (đối với max pooling) hoặc giá trị trung bình (đối với average pooling) để trả về ma trận đầu ra.

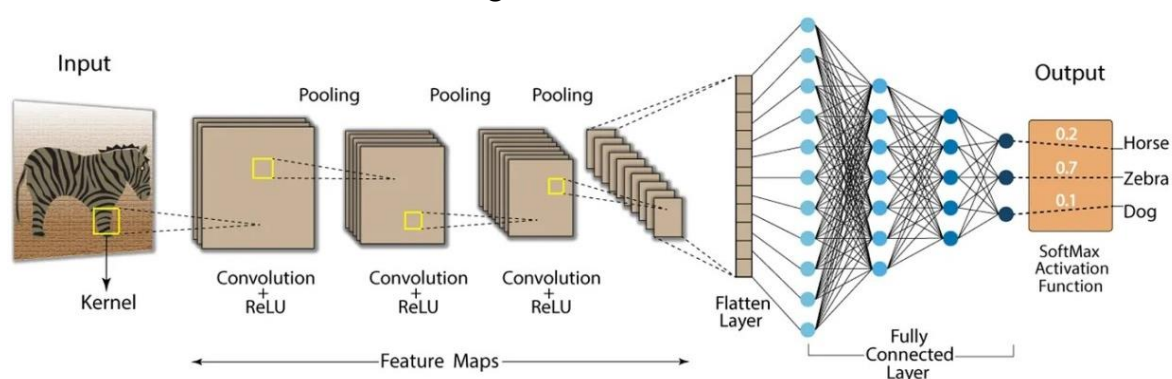
Kiểu	Max Pooling	Average Pooling
Chức năng	Từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng	Từng phép pooling tính trung bình các giá trị trong khu vực mà nó được áp dụng
Minh họa	<div> <div> <div>31</div><div>19</div><div>30</div><div>21</div> <div>31</div><div>25</div><div>38</div><div>25</div> <div>31</div><div>28</div><div>35</div><div>28</div> <div>31</div><div>22</div><div>32</div><div>22</div> </div> <div> <div>max pool (2x2)</div><div>stride = 2</div> </div> <div> <div>31</div><div>38</div> <div>31</div><div>35</div> </div> </div>	<div> <div> <div>31</div><div>19</div><div>30</div><div>21</div> <div>31</div><div>25</div><div>38</div><div>25</div> <div>31</div><div>28</div><div>35</div><div>28</div> <div>31</div><div>22</div><div>32</div><div>22</div> </div> <div> <div>avg pool (2x2)</div><div>stride = 2</div> </div> <div> <div>26</div><div>28</div> <div>28</div><div>29</div> </div> </div>
Nhận xét	<ul style="list-style-type: none"> ➤ Bảo toàn các đặc trưng đã phát hiện ➤ Thường được sử dụng 	<ul style="list-style-type: none"> ➤ Giảm kích thước feature map ➤ Được sử dụng trong mạng LeNet

Bảng 1: So sánh 2 kiểu Pooling

4.2.3. Lớp kết nối đầy đủ (Fully Connected)

Các lớp kết nối đầy đủ thường được đặt ở cuối của kiến trúc CNN, sau một loạt các lớp tích chập và lớp pooling. Chúng thường được gọi là "dense" (đầy đủ) vì mọi neural trong một lớp này được kết nối với mọi neural trong lớp liền trước. Điều này có nghĩa là mỗi neural trong một lớp kết nối đầy đủ nhận đầu vào từ tất cả các neural trong lớp trước đó.

Lớp kết nối đầy đủ đóng vai trò quan trọng trong việc tổng hợp thông tin và học các mối quan hệ phức tạp giữa các đặc trưng để thực hiện các tác vụ như phân loại, dự đoán và điều chỉnh mô hình trong CNN.



Hình 13: Cấu trúc mạng CNN

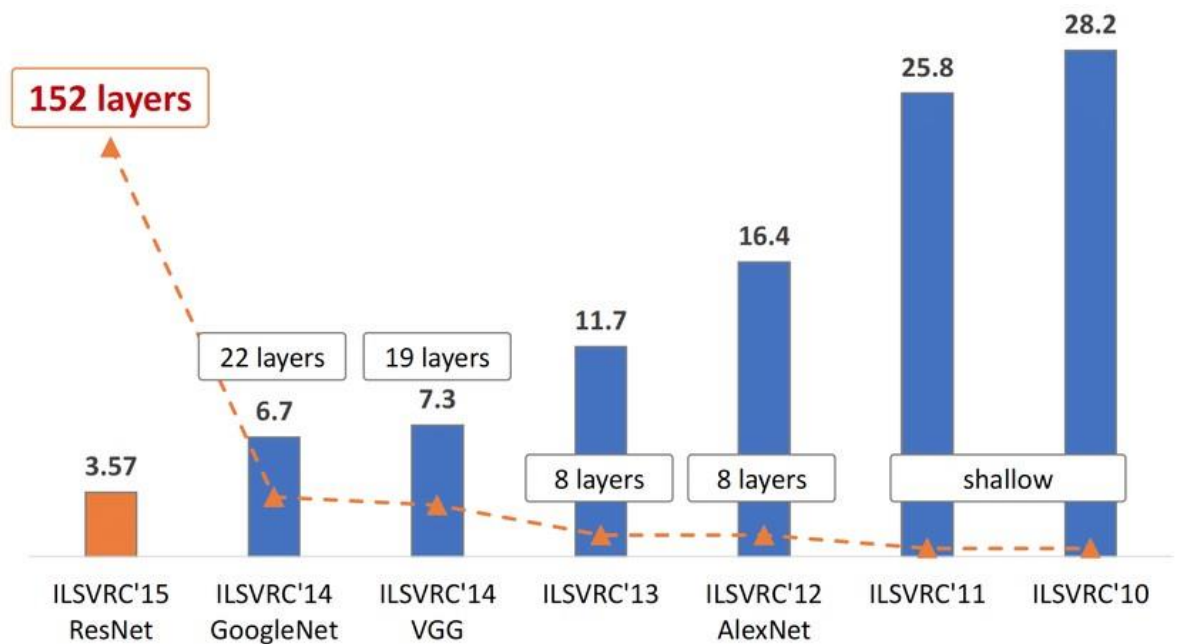
(<https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5>)

4.3. Mạng xương sống CNN (CNN backbone)

CNN backbone (Convolutional Neural Network backbone) là một phần quan trọng trong kiến trúc của mạng neural dùng cho các nhiệm vụ liên quan đến thị giác máy tính, chẳng hạn như nhận dạng vật thể trong hình ảnh, phân loại ảnh, hoặc các tác vụ khác trong lĩnh vực thị giác máy tính.

Backbone thường bao gồm một loạt các lớp convolutional và lớp pooling (thường là Convolutional Layers và MaxPooling Layers) được xếp chồng lên nhau. Các lớp này được thiết kế để trích xuất các đặc trưng quan trọng từ hình ảnh đầu vào, bằng cách áp dụng các phép tích chập và tổng hợp thông tin từ các vùng nhỏ của ảnh để tạo ra biểu đồ đặc trưng (feature map). Các feature map này sau đó được sử dụng bởi các lớp mạng neural sau để thực hiện các tác vụ như phân loại hoặc nhận dạng.

Một số CNN backbone phổ biến trong lĩnh vực thị giác máy tính bao gồm AlexNet, VGG, ResNet,... Mỗi loại backbone có các đặc điểm riêng biệt và hiệu suất khác nhau, tùy thuộc vào nhiệm vụ cụ thể và khả năng tính toán. Việc chọn backbone phù hợp là một phần quan trọng trong việc xây dựng các mô hình thị giác máy tính hiệu quả.



Hình 14: Biểu đồ phân bố kết quả sử dụng mạng CNN trên ImageNet

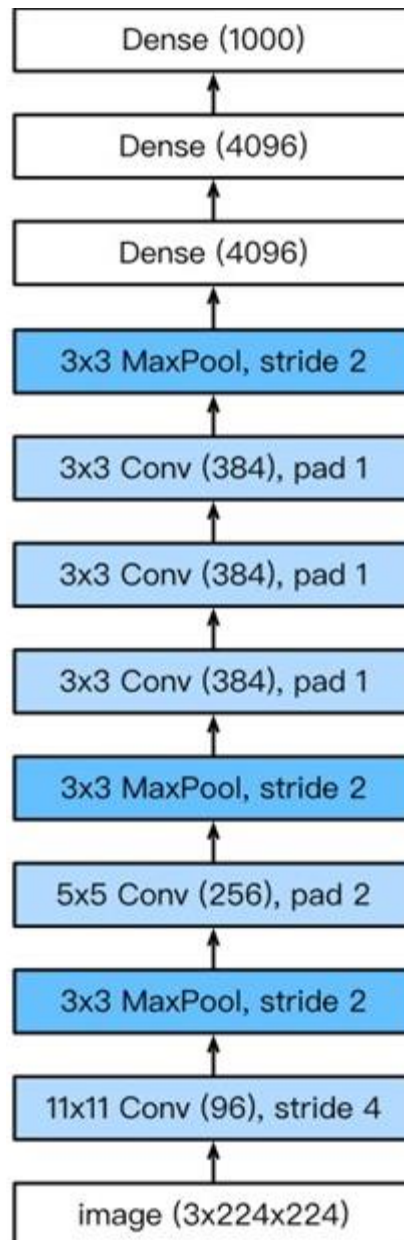
(https://www.researchgate.net/figure/The-evolution-of-the-winning-entries-on-the-ImageNet-Large-Scale-Visual-Recognition_fig1_321896881)

Biểu đồ trên phân bố kết quả sử dụng mạng CNN trong những năm từ 2010 đến 2015 trên bộ ImageNet. Từ năm 2012, AlexNet bắt đầu thắng thế giúp giảm xác suất lỗi trên bộ ImageNet xuống còn 16.4. Đến năm 2014 bắt đầu sử dụng mạng VGG và đã giảm xác suất lỗi xuống 7.3, sau đây là GoogleNet giảm xuống 6.7 và cuối cùng là ResNet giảm chỉ còn 3.57. Mạng ResNet cũng là mạng được sử dụng chính trong kết quả của bài nghiên cứu khoa học này.

4.3.1. Mạng neural tích chập sâu AlexNet

AlexNet là một mạng neural convolutional (CNN) nổi tiếng và đột phá trong lĩnh vực thị giác máy tính và học sâu. Đây là một trong những mạng neural đầu tiên và mạnh mẽ nhất được giới thiệu bởi Alex Krizhevsky, Ilya Sutskever và Geoffrey Hinton trong cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2012. Mạng này đã đặt nền móng cho sự phát triển mạnh mẽ của Deep Learning trong thị giác máy tính.

AlexNet đã giành chiến thắng trong cuộc thi ILSVRC 2012 với kết quả xuất sắc và đã mở đường cho việc sử dụng deep learning trong nhiều ứng dụng thị giác máy tính. Cấu trúc của AlexNet đã trở thành một kiểu mẫu cho các kiến trúc mạng CNN sau này và đóng vai trò quan trọng trong sự phát triển của lĩnh vực này.



Hình 15: Cấu trúc mạng AlexNet

(https://d2l.ai/chapter_convolutional-modern/alexnet.html)

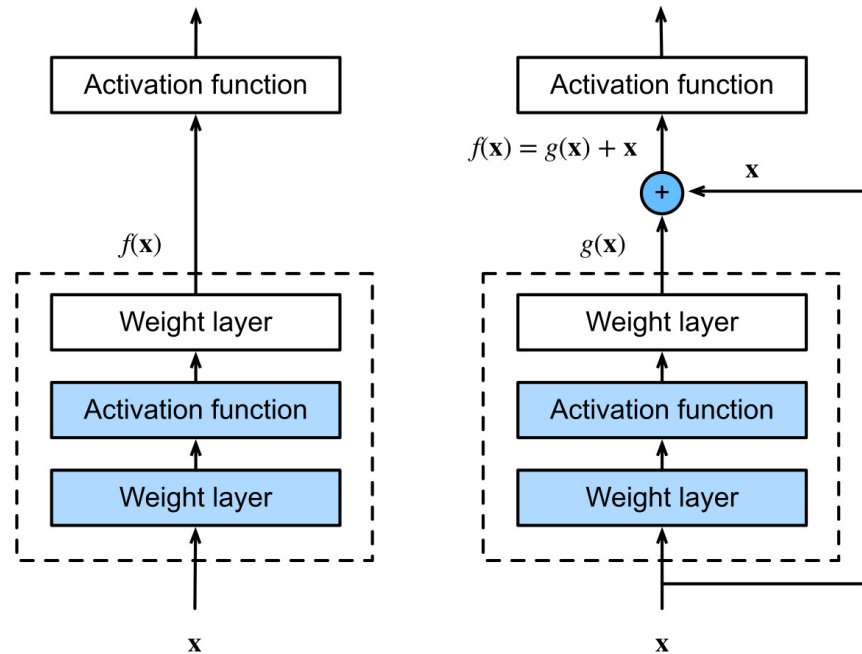
Kiến trúc AlexNet rất sâu so với các mạng trước đó tại thời điểm ra mắt, bao gồm 8 lớp tích chập và 3 lớp kết nối đầy đủ. Trong đó, ở tầng thứ nhất, kích thước của sổ tích chập là $11 \times 11 \times 3$ (3 là kích thước của ảnh RGB). Kích thước của sổ tích chập trong tầng thứ hai được giảm xuống còn 5×5 và sau đó là 3×3 . Ngoài ra, theo sau các tầng chập thứ nhất, thứ hai và thứ năm là các tầng Max Pooling với kích thước 3×3 và stride bằng 2.

Sau tầng tích chập cuối cùng là hai tầng kết nối đầy đủ với đầu ra là 4096, hai tầng này tạo ra gần 1GB các tham số mô hình. Cuối cùng là tầng kết nối đầy đủ với đầu ra 1000 dùng cho việc phân loại các đối tượng.

Alexnet cũng đã thay hàm kích hoạt sigmoid bằng ReLU. Điều này vừa giúp giảm việc tính toán vì ReLU không có phép lũy thừa như hàm kích hoạt sigmoid, vừa giúp việc huấn luyện mô hình trở nên dễ dàng hơn vì sử dụng các phương thức khởi tạo tham số khác nhau.

4.3.2. Mạng phân dư ResNet

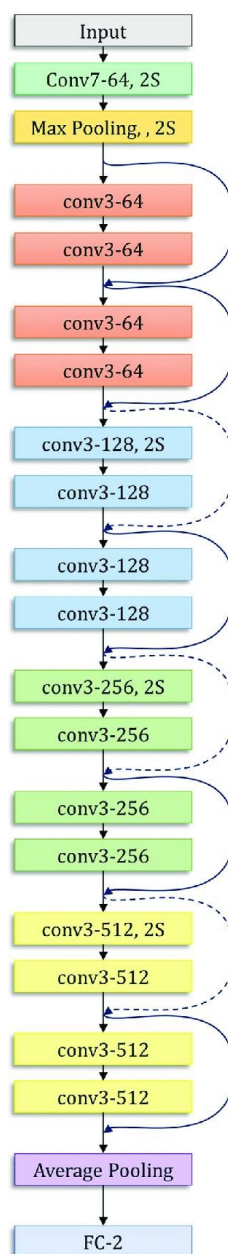
Mạng ResNet (Residual Network) là một kiểu kiến trúc mạng neural sâu (Deep Neural Network) chú trọng vào việc xử lý sự đánh mất gradient (vanishing gradient) trong quá trình đào tạo các mạng sâu. Ý tưởng chính của ResNet là sử dụng các khối còn gọi là "residual blocks" để giảm thiểu sự mất mát thông tin trong quá trình lan truyền ngược (backpropagation) và cho phép xây dựng các mạng sâu hơn mà vẫn có khả năng học tốt.



Hình 16: Sự khác nhau giữa khối thông thường (trái) và khối phân dư (phải)

(https://d2l.ai/chapter_convolutional-modern/resnet.html)

Với đầu vào là x , giả sử hàm ánh xạ lý tưởng muốn học là $f(x)$ được dùng là đầu vào của hàm kích hoạt. Phần nằm trong viền nét đứt (hình bên trái) phải khớp với hàm ánh xạ $f(x)$. Để thực hiện việc đó có thể không đơn giản nếu không cần khối đó và muốn giữ lại đầu vào x . Khi đó, chỉ cần tham số hóa độ lệch khối giá trị x bằng cách trả về $f(x) + x$, việc này giúp chống lại đạo hàm bằng 0 do vẫn còn cộng thêm x . Giải pháp này gọi là sử dụng kết nối “tắt” đồng nhất để xuyên qua nhiều lớp, dùng để giải quyết hiện tượng vanishing gradient. Và một khối như vậy được gọi là khối phân dư (Residual Block).



Hình 17: Cấu trúc mạng ResNet-18

(https://www.researchgate.net/figure/Architecture-of-the-ResNet-18-model-used-in-this-study_fig3_354432343)

Hai tầng đầu tiên trong kiến trúc của ResNet giống với kiến trúc GoogleNet, tầng đầu tiên là tầng tích chập 7x7 với 64 kênh đầu ra và stride bằng 2, theo sau là tầng Max Pooling 3x3 với stride bằng 2. Theo sau là những tầng tích chập kết hợp với khối phần dư. Số lượng các lớp theo sau tùy thuộc vào kiểu ResNet sử dụng. Một số kiểu ResNet như ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-101, ResNet-152. Các con số theo sau ResNet tượng trưng cho số lượng lớp có trong kiến trúc ResNet đó.

CHƯƠNG II: TỔNG QUAN LÝ THUYẾT THỊ GIÁC MÁY TÍNH

1. Giới thiệu thị giác máy tính

Thị giác máy tính (Computer Vision) là một lĩnh vực trong trí tuệ nhân tạo (AI) liên quan đến việc giải quyết các vấn đề liên quan đến xử lý hình ảnh và video bằng máy tính. Nó có mục tiêu là cho máy tính khả năng nhận biết, hiểu và xử lý thông tin từ hình ảnh và video giống như con người. Một số khía cạnh quan trọng của thị giác máy tính:

- Phân đoạn hình ảnh (Image Segmentation): Quá trình chia một hình ảnh thành các phần khác nhau dựa trên các đặc điểm như màu sắc, kích thước, hình dạng, và vị trí. Điều này thường được sử dụng trong việc nhận dạng đối tượng trong hình ảnh.
- Phát hiện đối tượng (Object Detection): Nhận biết và xác định vị trí của các đối tượng cụ thể trong hình ảnh hoặc video. Điều này có ứng dụng trong việc theo dõi đối tượng, phân loại đối tượng, và nhiều ứng dụng khác.
- Nhận dạng đối tượng (Object Recognition): Xác định xem đối tượng trong hình ảnh hoặc video là gì. Điều này có thể bao gồm việc phân loại đối tượng thành các danh mục cụ thể hoặc nhận dạng đối tượng theo tên riêng.
- Phân tích khuôn mặt (Face Analysis): Phân tích các đặc điểm của khuôn mặt như nhận dạng khuôn mặt, nhận dạng biểu cảm, và theo dõi chuyển động khuôn mặt.
- Xử lý video (Video Processing): Liên quan đến xử lý và phân tích video, bao gồm việc theo dõi đối tượng trong video, nhận biết hành động, và xử lý video theo thời gian thực.
- Tạo hình ảnh tổng hợp (Image Generation): Sử dụng máy tính để tạo ra hình ảnh hoặc video mới dựa trên các mô hình học máy.
- Nhận dạng chữ viết (Optical Character Recognition - OCR): Nhận biết và chuyển đổi văn bản từ hình ảnh hoặc tài liệu in thành dạng văn bản có thể xử lý.
- Tăng cường thực tế (Augmented Reality - AR): Kết hợp thông tin thị giác với thế giới thực để tạo ra trải nghiệm tăng cường thực tế, ví dụ như trò chơi AR hoặc ứng dụng hướng dẫn sử dụng.
- Xử lý hình ảnh y tế (Medical Image Processing): Áp dụng thị giác máy tính trong lĩnh vực y tế để phát hiện bệnh, theo dõi tiến triển của bệnh, và hỗ trợ trong quá trình chẩn đoán.
- Tổng hợp dữ liệu từ nhiều nguồn (Multimodal Fusion): Kết hợp thông tin từ nhiều nguồn dữ liệu như hình ảnh, âm thanh, và văn bản để đưa ra quyết định hoặc tạo ra hiểu biết phức tạp hơn về môi trường.

Thị giác máy tính là một lĩnh vực đang phát triển mạnh mẽ và có nhiều ứng dụng thú vị trong nhiều lĩnh vực khác nhau như công nghiệp, y tế, xe tự động, giám sát an ninh, và nhiều ứng dụng khác. Để đạt được các mục tiêu trong thị giác máy tính, các nhà

ngiên cứu thường sử dụng các mô hình học máy và mạng neural sâu (deep learning) để giải quyết các vấn đề phức tạp liên quan đến xử lý hình ảnh và video.

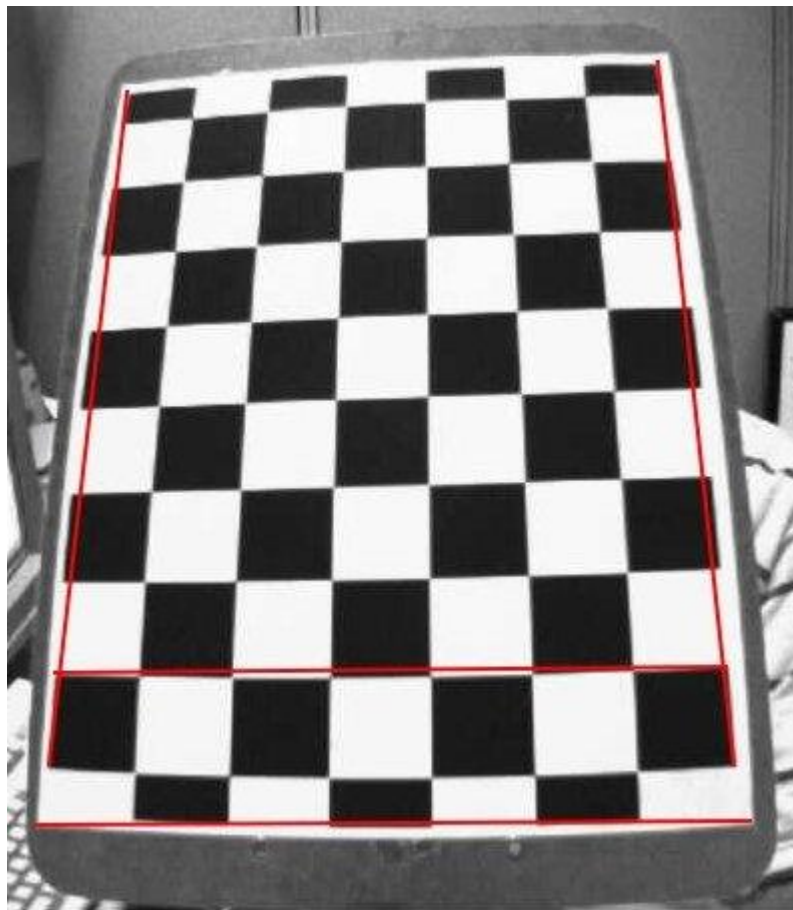
2. Camera Calibration

Camera calibration (hiệu chỉnh máy ảnh) là một quá trình quan trọng trong lĩnh vực thị giác máy tính, được thực hiện để biến đổi hình ảnh từ camera thành hình ảnh thế giới thực, giúp máy tính hiểu và xử lý thông tin trong không gian ba chiều (3D). Quá trình này đặc biệt quan trọng khi bạn muốn đo đạc hoặc theo dõi các đối tượng trong thế giới thực dựa trên hình ảnh chụp từ camera. Một số khía cạnh quan trọng trong quá trình calibrate camera:

- **Nhiệm vụ chính:** Nhiệm vụ chính của camera calibration là tìm ra các thông số và biểu đồ biến đổi giữa không gian 3D (thế giới thực) và không gian hình ảnh (hình ảnh trên camera). Cụ thể, điều này bao gồm xác định các thông số như tỷ lệ tiêu cự, distortion coefficients (hệ số méo), và vị trí của trung tâm của ống kính.
- **Biến đổi 2D sang 3D:** Sau khi calibrate camera, có thể biến đổi các điểm trong hình ảnh 2D thành không gian 3D dựa trên các thông số đã xác định. Điều này cho phép đo đạc các kích thước và khoảng cách trong không gian thực.
- **Loại bỏ méo ảnh (Distortion Correction):** Hệ số méo (distortion) là sự méo mó không mong muốn trong hình ảnh chụp từ camera và có thể làm sai lệch các thông tin về khoảng cách và hình dạng. Calibration giúp loại bỏ hoặc giảm thiểu méo ảnh này, đảm bảo hình ảnh đúng với thực tế.
- **Ứng dụng trong thị giác máy tính:** Camera calibration được sử dụng rộng rãi trong thị giác máy tính để các ứng dụng như theo dõi chuyển động, phát hiện đối tượng, tái tạo mô hình 3D và thực hiện đo lường trong không gian thực.
- **Phương pháp calibration:** Có nhiều phương pháp calibrate camera, bao gồm việc sử dụng mẫu đặc biệt (chẳng hạn như một dấu chấm hoặc một bảng chứa các hình dạng đã biết trước) và thu thập các hình ảnh từ nhiều góc độ khác nhau.
- **Công cụ phần mềm:** Có nhiều công cụ phần mềm và thư viện như OpenCV, MATLAB, và Python với các thư viện như Camera Calibration Toolbox cho việc thực hiện camera calibration.
- **Quá trình calibration là bước quan trọng để đảm bảo rằng thông tin từ camera có thể được sử dụng chính xác trong các ứng dụng thị giác máy tính, đặc biệt là khi cần đo lường và phân tích không gian ba chiều.**

Một số máy ảnh lỗi kim gây ra hiện tượng méo hình ảnh đáng kể và hai loại biến dạng phổ biến nhất là biến dạng radial và biến dạng tangential.

Biến dạng radial làm cho các đường thẳng trở nên cong. Biến dạng radial trở nên lớn hơn khi các điểm xa hơn khỏi trung tâm của hình ảnh. Ví dụ, bên dưới đây là một hình ảnh trong đó hai cạnh của bàn cờ đang được đánh dấu bằng các đường đỏ. Nhưng có thể thấy rằng biên của bàn cờ không phải là một đường thẳng và không khớp với đường đỏ. Tất cả các đường thẳng đều bị nở ra.



Hình 18: Hình ảnh bị biến dạng

(https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html)

Biến dạng radial có thể được biểu diễn như sau:

$$x_{distorted} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

$$y_{distorted} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

Tương tự vậy, hiện tượng biến dạng tangential xảy ra do thấu kính chụp ảnh không được đặt song song hoàn hảo với mặt phẳng hình ảnh. Dẫn đến việc một số vùng trong ảnh có thể trong gần hơn. Mức độ biến dạng tangential có thể được biểu diễn như sau:

$$x_{distorted} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

Tóm lại, để có thể hiệu chỉnh máy ảnh, cần tìm năm tham số gọi là hệ số biến dạng được cho bởi:

$$Distortion\ coefficients = (k_1\ k_2\ p_1\ p_2\ k_3)$$

Ngoài ra, cần một số thông tin khác như các thông số biên trong và ngoài máy ảnh. Các thông số đặc biệt dành riêng cho máy ảnh bao gồm như tỷ lệ tiêu cự (f_x , f_y) và vị trí tâm (c_x , c_y). Các thông số này sẽ được biểu diễn dưới dạng ma trận gọi là ma trận camera như sau:

$$\text{camera matrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$



Hình 19: Hình ảnh sau khi hiệu chỉnh

(https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html)

Dựa vào các tham số camera matrix và distortion coefficients kết hợp với những thư viện hỗ trợ như OpenCV trong python có thể hiệu chỉnh hình ảnh, loại bỏ các biến dạng. Đây là tiền đề để tăng sự chính xác trong các vấn đề liên quan đến thị giác máy tính.

3. Phát hiện khuôn mặt và các điểm mốc khuôn mặt (face landmark)

Phát hiện khuôn mặt và các điểm mốc khuôn mặt (face landmark) là hai nhiệm vụ quan trọng trong lĩnh vực thị giác máy tính và xử lý ảnh, đặc biệt trong lĩnh vực trí tuệ nhân tạo và ứng dụng của nó, như nhận dạng khuôn mặt, dự đoán cảm xúc, tự động nhận diện và điều khiển gương mặt trên các thiết bị di động.

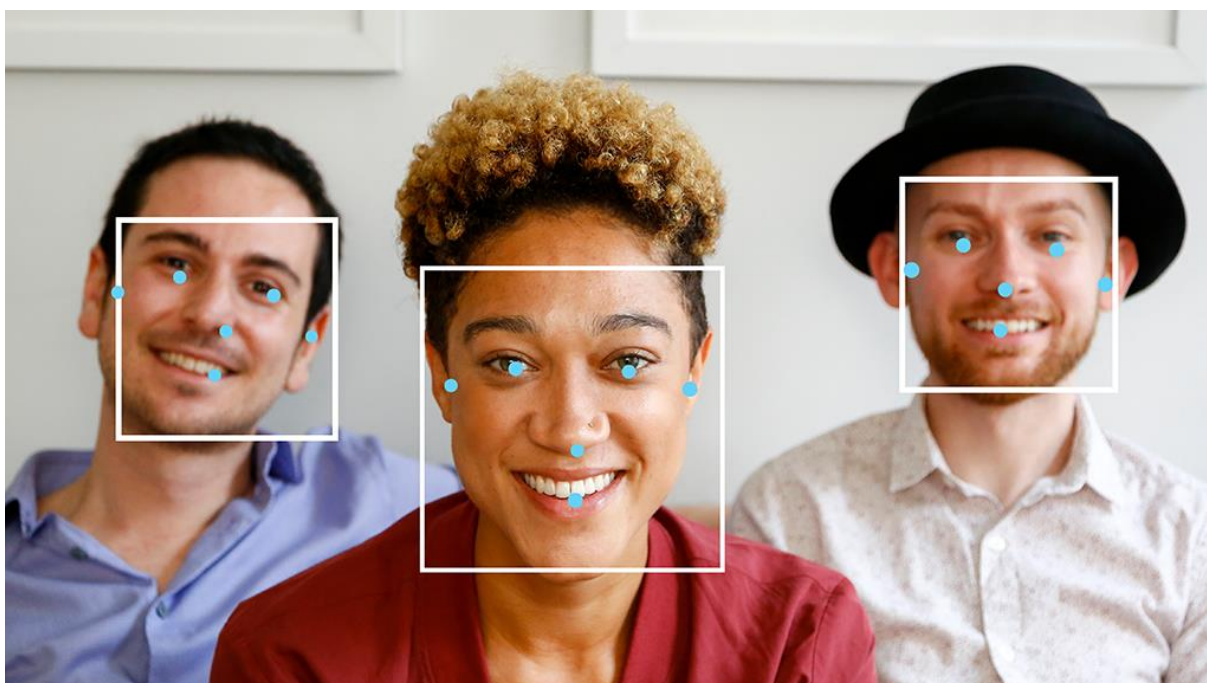
Hiện nay có rất nhiều nền tảng được xây dựng và phát triển cho các mục đích liên quan đến thị giác máy tính nói chung và phát hiện khuôn mặt, điểm mốc khuôn mặt nói riêng. Một trong các nền tảng được sử dụng phổ biến nhất là MediaPipe.

MediaPipe là một nền tảng mã nguồn mở được phát triển bởi Google, là tập hợp các giải pháp Machine Learning và trí tuệ nhân tạo như phát hiện vật thể, phân loại hình ảnh, phát hiện khuôn mặt, phát hiện các điểm mốc trên khuôn mặt, phân loại văn bản,...

Mỗi giải pháp bao gồm một hoặc nhiều mô hình được huấn luyện trước với độ chính xác cao.

3.1. MediaPipe – Phát hiện khuôn mặt

Phát hiện khuôn mặt là một bài toán cực kỳ quen thuộc và lâu đời tính đến thời điểm hiện tại. Nhiệm vụ của bài toán này là làm sao từ một ảnh, video hoặc webcam trực tiếp làm đầu vào, tìm ra được vị trí và bounding box những khuôn mặt con người xuất hiện trong đầu vào đấy, sau đấy đánh dấu các điểm quan trọng trên khuôn mặt như mũi, miệng, mắt,...(MediaPipe sử dụng 5-landmark). MediaPipe Face Detection sử dụng mạng BlazeFace làm nền tảng nhưng thay đổi backbones. Ngoài ra, thuật toán NMS (non-maximum suppression) cũng được thay thế bởi một chiến thuật khác, giúp giảm đáng kể thời gian xử lý.

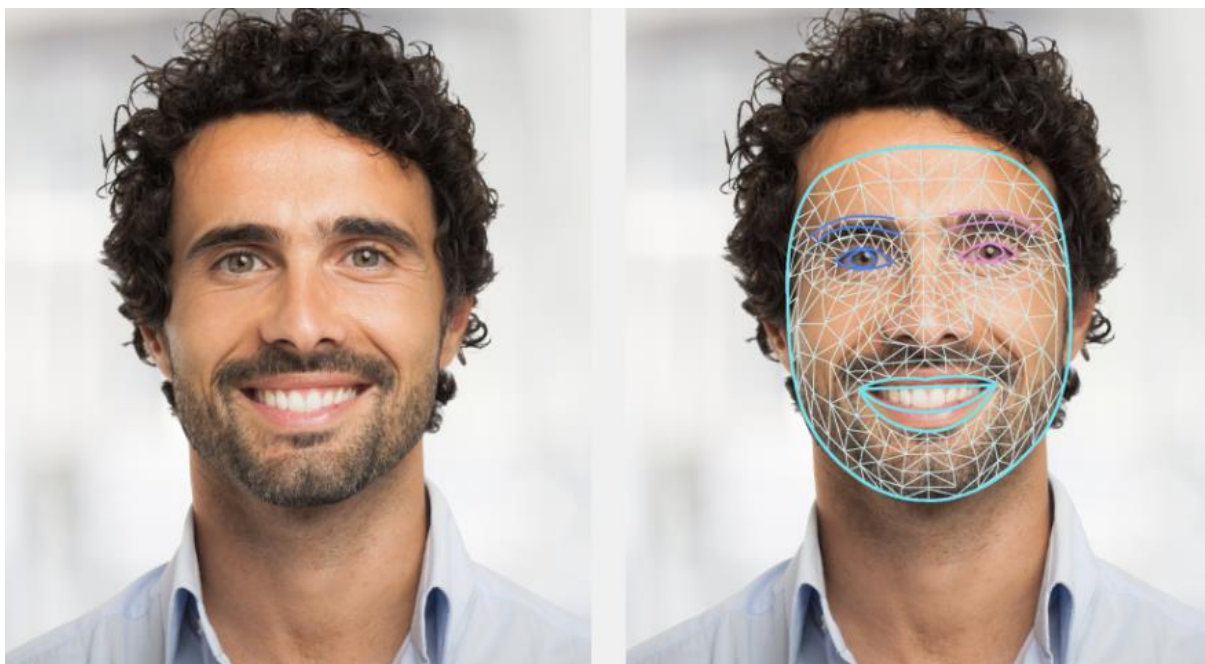


Hình 20: Ví dụ về phát hiện khuôn mặt

(https://developers.google.com/mediapipe/solutions/vision/face_detector)

3.2. MediaPipe – Phát hiện các điểm mốc trên khuôn mặt

Khác với phát hiện khuôn mặt, thay vì tìm vị trí và bounding box bao quanh mặt, MediaPipe Face Landmark Detection sẽ nhận diện một loạt các điểm trên khuôn mặt để tạo thành một lưới (mesh) của mặt. Lưới này thường được áp dụng vào các bài toán chỉnh sửa mặt 3D hay các tác vụ liên quan đến 3D Alignment và Anti-spoofing. Giải pháp của MediaPipe sinh ra 468 điểm trên mặt và tạo thành lưới mà không đòi hỏi quá nhiều năng lực tính toán cũng như số lượng camera.



Hình 21: Ví dụ về phát hiện các điểm mốc trên khuôn mặt
(https://developers.google.com/mediapipe/solutions/vision/face_landmarker)

CHƯƠNG III: TỔNG QUAN LÝ THUYẾT VỀ ARDUINO

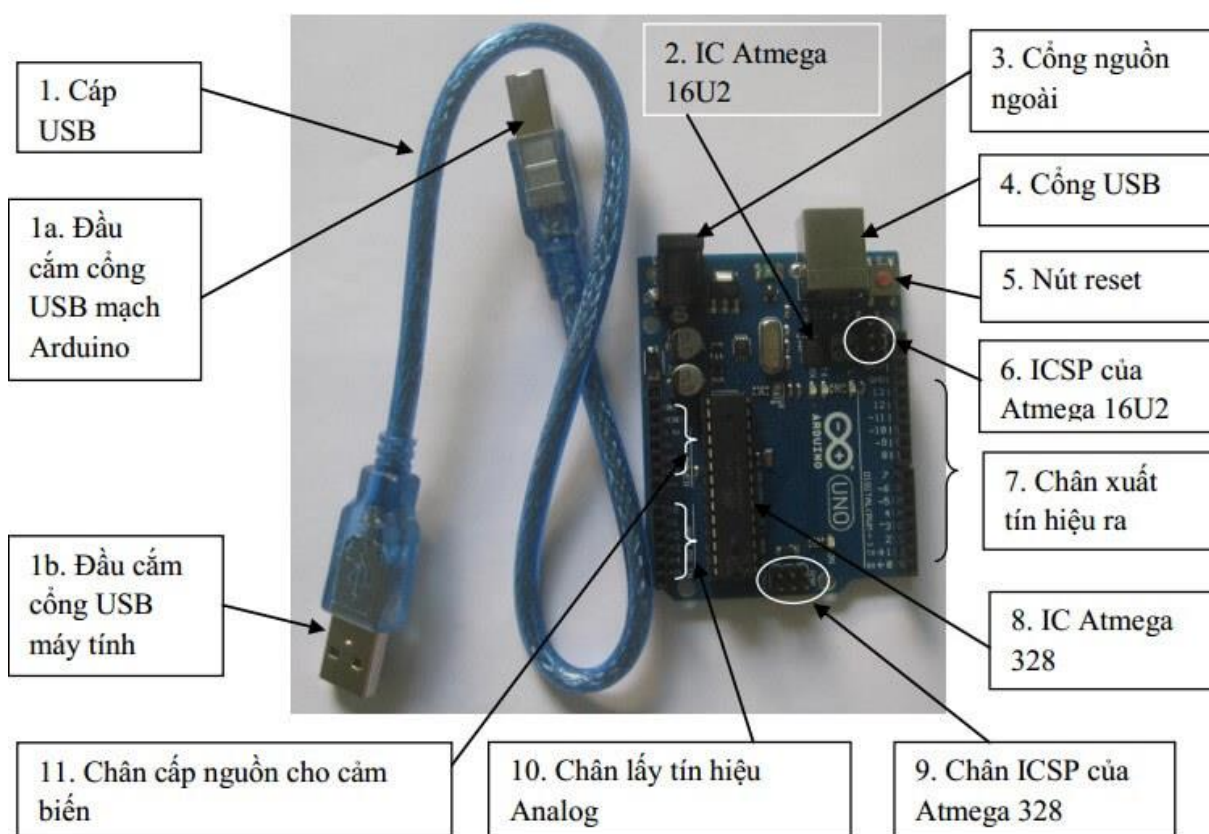
1. Giới thiệu

Arduino là một nền tảng phát triển điện tử mã nguồn mở phổ biến, được sáng tạo để đơn giản hóa quá trình phát triển các dự án và ứng dụng điện tử. Mạch Arduino được sử dụng để cảm nhận và điều khiển nhiều đối tượng khác nhau. Nó có thể thực hiện nhiều nhiệm vụ lấy tín hiệu từ cảm biến đến điều khiển đèn, động cơ, và nhiều đối tượng khác. Ngoài ra mạch còn có khả năng liên kết với nhiều module khác nhau như module đọc thẻ từ, ethernet shield, sim900A, để tăng khả năng ứng dụng của mạch.

Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit, hoặc ARM, Atmel 32-bit,.... Hiện phần cứng của Arduino có tất cả 6 phiên bản, Tuy nhiên phiên bản thường được sử dụng nhiều nhất là Arduino Uno và Arduino Mega.

2. Cấu tạo của Arduino

Hiện Arduino có rất nhiều loại, ví dụ với Arduino Uno R3 sẽ bao gồm cấu tạo và các thông số cơ bản sau:



Hình 22: Cấu tạo của Arduino Uno R3

(<https://mesidas.com/arduino-la-gi/>)

Vi điều khiển	Atmega 328 (họ 8 bit)
Điện áp hoạt động	5V – DC (cấp qua cổng USB)
Tần số hoạt động	16 MHz
Dòng tiêu thụ	30mA
Điện áp vào khuyến dùng	7 – 12V – DC
Điện áp vào giới hạn	6 – 20V – DC
Số chân Digital I/O	14 (6 chân PWM)
Số chân Analog	6 (độ phân giải 10 bit)
Dòng tối đa trên mỗi chân I/O	30 mA
Dòng ra tối đa (5V)	500 mA
Dòng ra tối đa (3.3V)	50 mA
Bộ nhớ flash	32 KB (Atmega328) với 0.5KB dùng bởi bootloader
SRAM	2KB (Atmega328)
EEPROM	1KB (Atmega328)

3. Ứng dụng Arduino

Arduino có nhiều ứng dụng trong đời sống, trong việc chế tạo các thiết bị điện tử chất lượng cao. Một số ứng dụng có thể kể đến như:

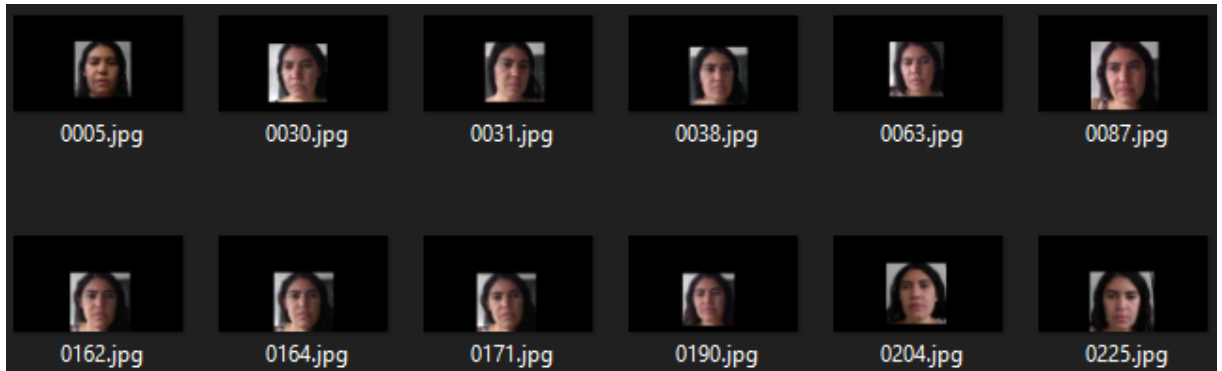
- Lập trình robot: Arduino chính là một phần quan trọng trong trung tâm xử lý giúp điều khiển được hoạt động của robot.
- Lập trình máy bay không người lái. Có thể nói đây là ứng dụng có nhiều kỳ vọng trong tương lai.
- Game tương tác: chúng ta có thể dùng Arduino để tương tác với Joystick, màn hình,... để chơi các trò như Tetrix, phá gạch, Mario...
- Arduino điều khiển thiết bị ánh sáng cảm biến tốt. Là một trong những bộ phận quan trọng trong cây đèn giao thông, các hiệu ứng đèn nháy được cài đặt làm nổi bật các biển quảng cáo.
- Arduino cũng được ứng dụng trong máy in 3D và nhiều ứng dụng khác tùy thuộc vào khả năng sáng tạo của người sử dụng.

CHƯƠNG IV: XÂY DỰNG CHƯƠNG TRÌNH VÀ CHẾ TẠO SẢN PHẨM

1. Huấn luyện mô hình

1.1. Giới thiệu tập dữ liệu

Tập dữ liệu MPIIFaceGaze là tập dữ liệu dựa trên tập dữ liệu MPIIGaze, dùng cho mục đích ước tính hướng nhìn của người trong tự nhiên. Tập dữ liệu này chứa 213,659 hình ảnh được thu thập từ 15 người tham gia trong quá trình sử dụng máy tính xách tay tự nhiên hằng ngày và được thu thập trong hơn ba tháng.



Hình 23: Tập dữ liệu MPIIFaceGaze

Ngoài ra, tập dữ liệu cũng đã được chuẩn hóa và được lưu trữ dưới dạng MATLAB bao gồm các hình ảnh khuôn mặt đã được chuẩn hóa có kích thước 448x448 pixel và vector góc nhìn 2D. Cấu trúc của thư mục tập dữ liệu như sau:

MPIIFaceGaze_normalized/

- |— p00.mat
- |— p01.mat
- |— p02.mat
- |— p03.mat
- |— p04.mat
- |— p05.mat
- |— p06.mat
- |— p07.mat
- |— p08.mat
- |— p09.mat
- |— p10.mat
- |— p11.mat
- |— p12.mat
- |— p13.mat
- |— p14.mat

Trong đó, mỗi tập .mat có cấu trúc Data bao gồm Data.data và Data.label

- **Data.data:** chứa hình ảnh khuôn mặt đã được chuẩn hóa với kích thước 448x448 pixel, có dạng [chiều cao hình ảnh, chiều rộng hình ảnh, kênh màu, số lượng mẫu] là [448, 448, 3, 3000] cho mỗi tập.
- **Data.label:** chứa nhãn của dữ liệu có dạng [kênh, số lượng mẫu] là [16, 3000]. Hai chiều đầu tiên là hướng nhìn đã được chuẩn hóa, chiều 3~4 là tư thế đầu chuẩn hóa và chiều 5~16 là các điểm mốc trên khuôn mặt dưới dạng (x, y) cho 6 điểm mốc.

1.2. Tiền xử lý dữ liệu

Tiền xử lý dữ liệu là quá trình chuẩn bị và làm sạch dữ liệu trước khi nó được sử dụng cho mục đích phân tích hoặc xây dựng mô hình máy học. Tiền xử lý dữ liệu là một phần quan trọng trong quá trình khai phá dữ liệu và có thể ảnh hưởng lớn đến chất lượng và hiệu suất của các ứng dụng và mô hình dự đoán.

Đối với tập dữ liệu MPIIFaceGaze dù đã được chuẩn hóa, vẫn cần một bước để tiền xử lý dữ liệu nữa là lấy những dữ liệu cần thiết để huấn luyện mô hình và lưu trữ nó thành một tệp tin mới với định dạng HDF5 (.h5, HDF5 là một định dạng tệp tin để lưu trữ dữ liệu có cấu trúc). Cụ thể:

- **Bước 1:** Tiến hành đọc lần lượt các tệp tin MATLAB trong thư mục tập dữ liệu.
- **Bước 2:** Đối với từng tệp tin MATLAB trích xuất các dữ liệu sau:
 - **Data.data:** Lấy ra danh sách hình ảnh được lưu trữ, sau đó chuyển đổi danh sách hình ảnh từ kiểu dữ liệu gốc sang kiểu số nguyên 8-bit không dấu. Mục đích của việc chuyển đổi là giới hạn dải giá trị của mỗi pixel trong hình ảnh từ 0 đến 255, điều này sẽ giúp tiết kiệm bộ nhớ và giảm dung lượng lưu trữ.
 - **Data.label:** Lấy ra danh sách hướng nhìn đã được chuẩn hóa và danh sách tư thế đầu đã được chuẩn hóa.
- **Bước 3:** Sau khi lấy được 3 danh sách bao gồm: hình ảnh, hướng nhìn và tư thế đầu thì tiến hành lưu trữ vào tệp tin với định dạng HDF5.
- **Bước 4:** Làm lần lượt như vậy cho 15 mẫu có trong tập dữ liệu và lưu trữ vào cùng một tệp tên HDF5, đối với mỗi 3 danh sách lấy ra được từ từng mẫu khi lưu trữ cần đánh số thứ tự theo thứ tự của mẫu đang duyệt.

1.3. Huấn luyện và đánh giá mô hình

Sau khi tiền xử lý dữ liệu và có tập dữ liệu mới được lưu trữ trong một tệp tin dưới định dạng HDF5, tiến hành huấn luyện mô hình bằng CNN sử dụng hai backbone là AlexNet và ResNet. Sử dụng phương pháp leave-one-out để chia tập dữ liệu thành 2

tập huấn luyện và kiểm tra, sau đó đánh giá mô hình dựa trên sai số góc (angle error) và sai số góc trung bình (mean angle error), vì mô hình được huấn luyện dựa trên nhãn là vector góc và khi dự đoán mô hình cũng trả về kết quả là vector góc.

Cụ thể, với một tập dữ liệu có N mẫu, phương pháp leave-one-out sẽ huấn luyện và đánh giá mô hình N lần. Mỗi lần tiến hành chia tập dữ liệu thành hai tập huấn luyện và kiểm tra bằng cách loại một mẫu trong N mẫu ra để làm tập kiểm tra, tiếp theo sử dụng N-1 mẫu còn lại để làm tập huấn luyện. Sau khi huấn luyện được mô hình sẽ dùng mẫu ban đầu bị loại ra để kiểm tra, đánh giá mô hình và lưu trữ kết quả đánh giá. Thực hiện huấn luyện và đánh giá N lần như vậy, mỗi lần thực hiện sẽ loại ra một mẫu khác với những lần trước để làm tập kiểm tra. Sau khi hoàn thành N lần huấn luyện và đánh giá tiến hành lấy N kết quả đánh giá để tính độ trung bình đánh giá và so sánh với các mô hình khác.

Để tính được sai số góc (angle error), giả sử nhãn dự đoán của mô hình là prediction, nhãn góc của tập dữ liệu là label. Các bước để tính sai số góc (angle error) như sau:

- Trích xuất góc nghiêng (pitch) và góc quay (yaw) từ các nhãn:

$$prediction \rightarrow prediction_{pitch}, prediction_{yaw}$$

$$label \rightarrow label_{pitch}, label_{yaw}$$

- Sau khi đã có pitch và yaw, tính các thành phần x, y, z của vector đơn vị:

$$x = -\cos(\text{pitch}) * \sin(\text{yaw})$$

$$y = -\sin(\text{pitch})$$

$$z = -\cos(\text{pitch}) * \cos(\text{yaw})$$

- Tính độ dài vector (sử dụng công thức Euclid) và chia các thành phần x, y, z cho độ dài để đảm bảo độ dài của vector kết quả bằng 1:

$$norm = \sqrt{x^2 + y^2 + z^2}$$

$$x = \frac{x}{norm}$$

$$y = \frac{y}{norm}$$

$$z = \frac{z}{norm}$$

- Sau khi đã có các thành phần x, y, z của vector đơn vị prediction ($pred_x, pred_y, pred_z$) và label ($label_x, label_y, label_z$), tính cosin của góc giữa hai vector đơn vị của prediction và label bằng cách sử dụng phép tích vô hướng:

$$angle = pred_x * label_x + pred_y * label_y + pred_z * label_z$$

- Chuyển đổi góc thành đơn vị độ, kết quả cuối cùng chính là sai số góc (angle error):

$$angle\ error = \frac{\arccos(angle) * 180}{\pi}$$

Tương tự vậy với tập dữ liệu MPIIFaceGaze đã tiền xử lý, trong tập dữ liệu chứa 15 mẫu của 15 người đã tham gia vào việc thu thập dữ liệu. Tiến hành thực hiện việc huấn luyện và đánh giá theo phương pháp leave-one-out. Cụ thể, thực hiện huấn luyện và đánh giá 15 lần, với mỗi lần thực hiện tiến hành loại bỏ một mẫu trong 15 mẫu để làm tập kiểm tra, sử dụng 14 mẫu còn lại để huấn luyện, cuối cùng sử dụng mô hình đã huấn luyện được để kiểm tra và đánh giá với mẫu được loại ra ban đầu, sau đó lưu kết quả đánh giá lại. Trong các lần thực hiện tiếp theo sẽ loại bỏ mẫu khác với các mẫu đã được loại ở những lần trước. Tổng quan hóa các lần thực hiện như sau:

- Lần 1:
 - Chọn mẫu có ID là 00 trong tập dữ liệu để làm tập kiểm thử
 - Huấn luyện mô hình với các mẫu có ID từ 01 đến 14.
 - Sử dụng mô hình đã huấn luyện kiểm thử với mẫu có ID là 00, tính sai số góc (angle error) và lưu trữ lại.
- Lần 2:
 - Chọn mẫu có ID là 01 để làm tập kiểm thử
 - Huấn luyện mô hình với các mẫu có ID là 00 và ID từ 02 đến 14.
 - Sử dụng mô hình đã huấn luyện kiểm thử với mẫu có ID 01, tính sai số góc (angle error) và lưu trữ lại.
- ...
- Lần 15:
 - Chọn mẫu có ID 14 để làm tập kiểm thử
 - Huấn luyện mô hình với các mẫu có ID từ 00 đến 13.
 - Sử dụng mô hình đã huấn luyện kiểm thử với mẫu có ID 14, tính sai số góc (angle error) và lưu trữ lại.

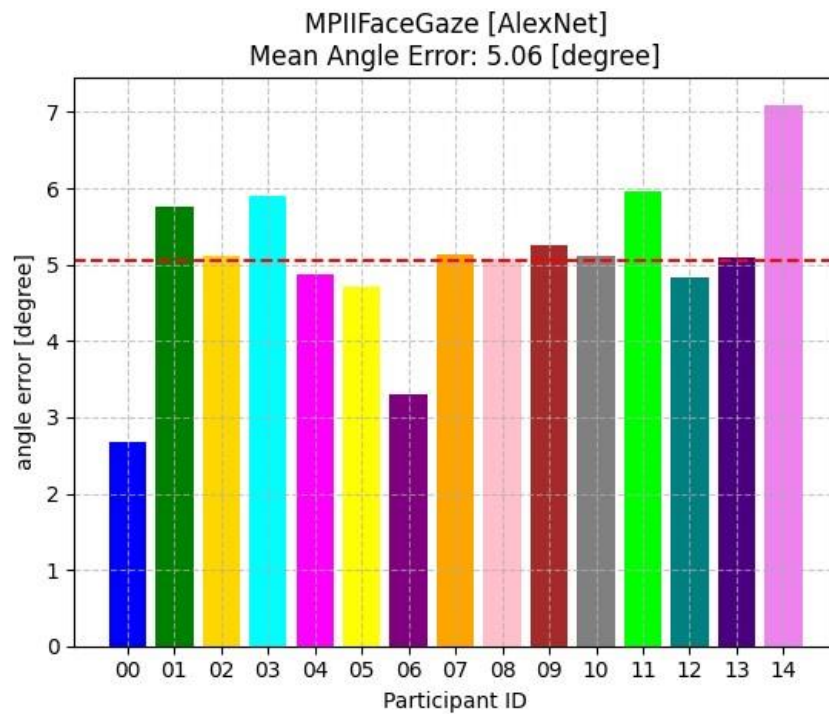
Cuối cùng, dựa vào danh sách 15 sai số góc đã tính được để tính sai số góc trung bình (mean angle error), lấy kết quả đó so sánh với các mô hình khác và chọn ra mô hình tối ưu nhất để sử dụng.

Với các thông số huấn luyện mô hình cơ bản như epoch bằng 15, batch size = 32. Kết quả huấn luyện và đánh giá mô hình dựa trên hai model AlexNet và ResNet như sau:

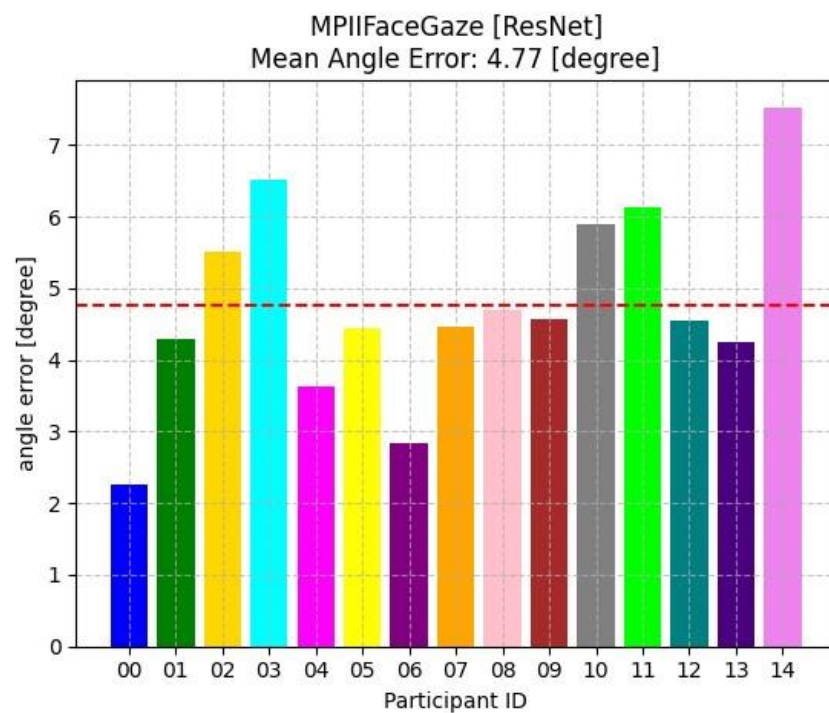
Model	Thời gian huấn luyện trung bình (s/epoch)
AlexNet	936 s/epoch
ResNet	278 s/epoch

Bảng 2: Kết quả huấn luyện mô hình

Thời gian huấn luyện trung bình trên được tính khi huấn luyện mô hình với NVIDIA GeForce GTX 1650. Theo phương pháp leave-one-out tổng cộng có 15 lần huấn luyện, mỗi lần bao gồm 15 epoch. Vậy với thời gian huấn luyện trung bình của AlexNet là 936 s/epoch thì tổng thời gian hoàn thành việc huấn luyện khoảng 58 tiếng, còn ResNet khoảng 17 tiếng. Qua đây nhận thấy mô hình ResNet giảm chi phí và thời gian tính toán hơn so với AlexNet. Tiếp theo về kết quả đánh giá mô hình:



Hình 24: Kết quả đánh giá mô hình CNN trên tập MPIIFaceGaze (AlexNet)

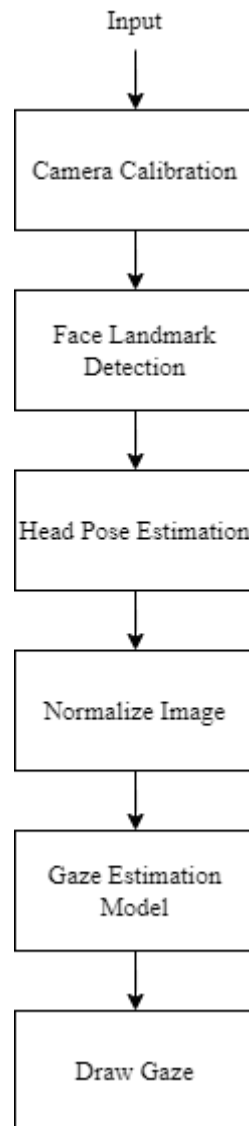


Hình 25: Kết quả đánh giá mô hình CNN trên tập MPIIFaceGaze (ResNet)

Sai số góc trung bình của AlexNet là 5.06, còn ResNet là 4.77. Nhận thấy rằng ngoài việc tối ưu về chi phí và thời gian tính toán thì ResNet cũng giảm tỉ lệ sai số hơn so với AlexNet.

2. Xây dựng module ước tính hướng mắt

Sau khi xây dựng được mô hình dự đoán hướng mắt, tiến hành xây dựng module ước tính hướng nhìn của mắt theo sơ đồ hoạt động sau:



Hình 26: Sơ đồ hoạt động của module ước tính hướng mắt

Sử dụng thư viện OpenCV trong Python để nhận hình ảnh từ webcam để làm đầu vào cho chương trình với kích thước 640x480. Sau đó tiến hành hiệu chỉnh máy ảnh với các tham số camera matrix và distortion coefficients. Trong đó camera matrix có dạng:

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Để tính các giá trị f_x , f_y , c_x , c_y trong camera matrix cần dựa vào kích thước input ban đầu là 640x480. Giả sử đặt 640 là width và 480 là height, các giá trị được tính như sau:

$$f_x = \text{width}, f_y = \text{width}, c_x = \frac{\text{width}}{2}, c_y = \frac{\text{height}}{2}$$

Vậy với đầu vào từ webcam có kích thước 640x480 sẽ thu về được camera matrix như sau:

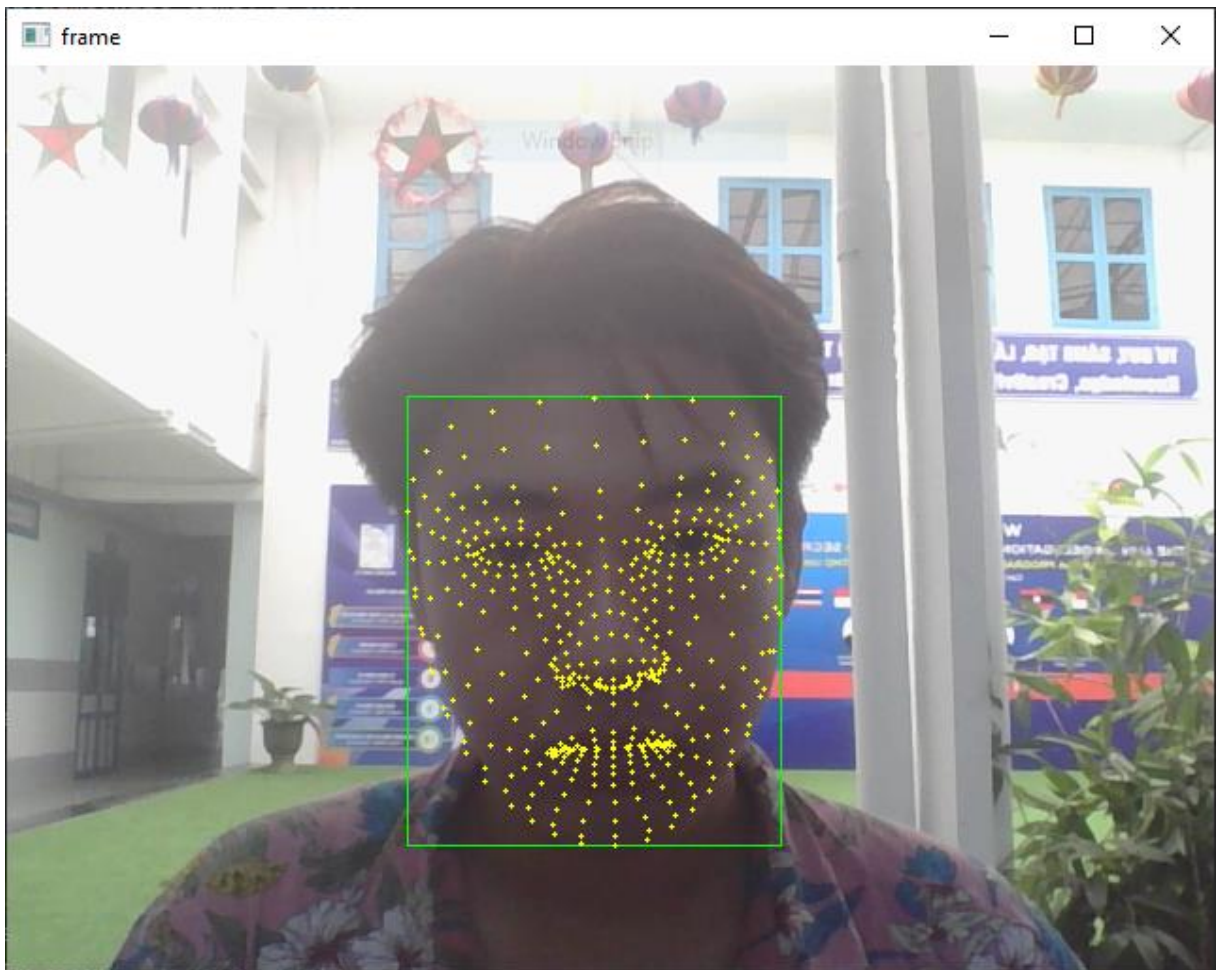
$$\text{camera matrix} = \begin{bmatrix} 640 & 0 & 320 \\ 0 & 640 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

Giả sử không có biến dạng ống kính sẽ có distortion coefficients như sau:

$$\text{Distortion coefficients} = (0 \ 0 \ 0 \ 0 \ 0)$$

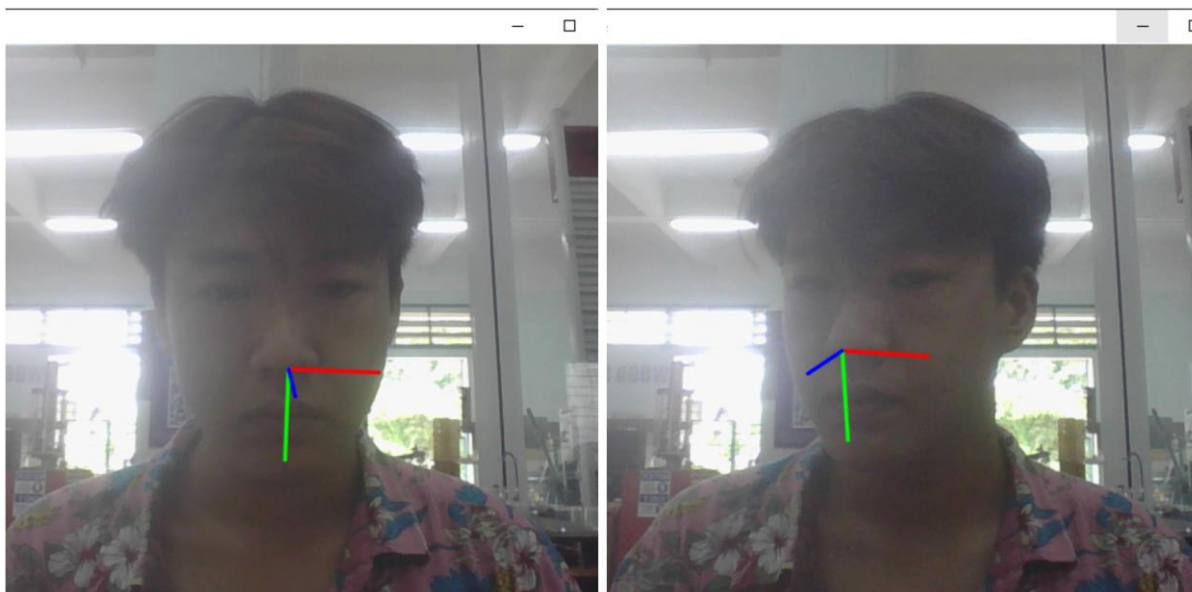
Sau khi có được các tham số camera matrix và distortion coefficients, thực hiện hiệu chỉnh máy ảnh cho hình ảnh đầu vào.

Từ hình ảnh đầu vào đã hiệu chỉnh, sử dụng thư viện MediaPipe để phát hiện khuôn mặt và các điểm mốc trên khuôn mặt. Nếu phát hiện khuôn mặt tiến hành trả về một lớp Face bao gồm hai thông tin là bounding box và landmarks của khuôn mặt. Kết quả khi vẽ bounding box và landmarks của khuôn mặt ở đầu ra:



Hình 27: Kết quả nhận diện khuôn mặt và các điểm mốc trên khuôn mặt

Sử dụng hàm solvePnP trong OpenCV kết hợp với landmarks đã tìm được để ước tính tư thế đầu. Hàm sẽ trả về Rotation vector (vector biểu diễn quay) và Translation vector (vector biểu diễn sự dịch chuyển). Tư thế đầu được biểu diễn dưới dạng hệ trục tọa độ Oxyz thông qua Rotation vector và Translation vector như sau:



Hình 28: Kết quả ước tính tư thế đầu

Sau khi tính toán được Rotation vector và Translation vector, thực hiện việc biến đổi mô hình 3D của khuôn mặt dựa trên Rotation vector và Translation vector của khuôn mặt trong không gian 3D. Để biến đổi cần chuyển Rotation vector về dạng ma trận, sau đó thực hiện biến đổi bằng cách:

$$model3D_{face} = LANDMARKS @ rot + vector_{Translation}$$

Trong đó, LANDMARKS là 468 điểm mốc trên khuôn mặt của MediaPipe được lưu dưới dạng ma trận, rot là ma trận chuyển vị của ma trận Rotation.

Sau khi có được mô hình 3D của khuôn mặt, tiến hành tìm điểm trung tâm của mặt và mắt. Trong trường hợp tập dữ liệu MPIIFaceGaze, điểm trung tâm được định nghĩa là trung bình tọa độ của 6 điểm các góc của 2 mắt và miệng, trong đó 4 điểm của 2 mắt và 2 điểm của miệng.



Hình 29: Sáu điểm mốc quan trọng trên khuôn mặt

(<https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/appearance-based-gaze-estimation-in-the-wild>)

Vì vậy, để tìm được điểm trung tâm của mặt và mắt, cần tính trung bình của 6 điểm trên và tiến hành quy chiếu trên mô hình 3D của khuôn mặt để tìm tọa độ điểm trung tâm trong không gian 3D.

Vì đặc tính của tập dữ liệu MPIIFaceGaze là được thu thập tự nhiên từ thói quen sử dụng máy tính xách tay hằng ngày của người tham gia. Nên dữ liệu thu vào gần như không có những hiện tượng xoay tư thế đầu hoặc ngã nghiêng đầu quá nhiều, vì vậy trước khi ước lượng hướng nhìn của mắt cần chuẩn hóa đầu vào để tối ưu hóa kết quả ước lượng. Các bước để chuẩn hóa đầu vào:

Bước 1: Chuẩn hóa Rotation vector:

- Từ điểm trung tâm tính được trong quá trình ước tính tư thế đầu, chuyển hóa nó thành vector đơn vị và lấy làm đại diện cho trục Z của hệ trục tọa độ Oxyz trong không gian 3D.
- Chuyển hóa Rotation vector thành ma trận 3x3 và trích xuất trục X của Rotation.
- Lấy tích có hướng giữa trục Z và X vừa tính được ở trên, chuyển hóa kết quả tích có hướng trở thành vector đơn vị và lấy làm đại diện cho trục Y của hệ trục tọa độ Oxyz trong không gian 3D.
- Lấy tích có hướng giữa trục Z và Y trong không gian 3D vừa tính được và lấy kết quả làm đại diện cho trục X của hệ trục tọa độ Oxyz trong không gian 3D.
- Từ X, Y và Z tạo thành ma trận quay mới và chuyển về dạng Rotation vector.

Bước 2: Chuẩn hóa hình ảnh:

- Tính ma trận nghịch đảo từ camera matrix đầu vào:

$$camera\ matrix = \begin{bmatrix} 640 & 0 & 320 \\ 0 & 640 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

$$camera\ matrix\ inverse = \begin{bmatrix} \frac{1}{640} & 0 & -\frac{1}{2} \\ 0 & \frac{1}{640} & -\frac{3}{8} \\ 0 & 0 & 1 \end{bmatrix}$$

- Sử dụng camera matrix đã được chuẩn hóa, đây là camera matrix đã chuẩn hóa được cung cấp từ tập dữ liệu MPIIFaceGaze:

$$normalized\ camera\ matrix = \begin{bmatrix} 1600 & 0 & 112 \\ 0 & 1600 & 112 \\ 0 & 0 & 1 \end{bmatrix}$$

- Tính khoảng cách từ điểm trung tâm mô hình 3D khuôn mặt đến gốc tọa độ của hệ trục tọa độ Oxyz trong không gian 3D. Từ khoảng cách tính được tìm ma trận tỉ lệ của khuôn mặt.
- Tìm ma trận biểu diễn phép chuyển đổi bằng cách nhân ma trận tỉ lệ với Rotation vector đã chuẩn hóa ở bước 1.

$$\text{conversion matrix} = \text{scale} @ \text{normalized rotation}$$

- Sau khi có được ma trận biểu diễn phép chuyển đổi, tìm ma trận chiếu bằng cách lấy tích 3 ma trận: conversion matrix, normalized camera matrix và camera matrix inverse.

$$\text{projection matrix} =$$

$$\text{normalized camera matrix} @ \text{conversion matrix} @ \text{camera matrix inverse}$$

- Cuối cùng, sử dụng hàm warpPerspective trong thư viện OpenCV trên ma trận chiếu để tạo ra hình ảnh chuẩn hóa với kích thước 224x224 (bằng với kích thước đầu vào của mô hình đã huấn luyện).



Hình 30: Kết quả chuẩn hóa hình ảnh

Hình ảnh sau khi được chuẩn hóa sẽ luôn được cố định theo hệ trục tọa độ trong không gian, dù tư thế đầu của hình ảnh đầu vào có nghiêng thì hình ảnh chuẩn hóa vẫn luôn cố định theo trục đứng. Mục đích của việc chuẩn hóa là để đầu vào của mô hình có tư thế đầu tương đương với đặc tính tư thế đầu của tập dữ liệu gốc, tạo tiền đề cho việc nhận dạng hướng nhìn của mắt trong mô hình đã huấn luyện được chính xác hơn.

Sử dụng hình ảnh đã được chuẩn hóa làm đầu vào cho mô hình đã huấn luyện, mô hình sẽ dự đoán và trả về hướng mắt dưới dạng góc, từ kết quả trả về trích xuất pitch (góc nghiêng) và yaw (góc quay) để làm tham số trong việc dự đoán hướng nhìn và vẽ hướng nhìn.

Để vẽ hướng nhìn cần thực hiện các bước:

Bước 1: Chuyển hướng nhìn từ dạng góc sang dạng vector:

- Trích xuất pitch (góc nghiêng) và yaw (góc quay) từ hướng nhìn dự đoán.
- Từ pitch và yaw tính hướng nhìn dưới dạng vector trong không gian 3D:

normalized gaze vector

$$= -[\cos(\text{pitch}) * \sin(\text{yaw}), \sin(\text{pitch}), \cos(\text{pitch}) * \cos(\text{yaw})]$$

Bước 2: Để vẽ hướng nhìn đúng cho hình ảnh đầu vào ban đầu, cần đảo ngược quá trình chuẩn hóa cho vector hướng nhìn bằng cách chuyển vector hướng nhìn về dạng ma trận và nhân với ma trận Rotation đã được chuẩn hóa.

$$\text{gaze vector} = \text{normalized gaze vector} @ \text{normalized rotation}$$

Bước 3: Vẽ hướng nhìn trong không gian với 2 điểm, điểm bắt đầu là điểm trung tâm của khuôn mặt, điểm kết thúc được tính như sau:

$$\text{end point} = \text{face}_{\text{center}} + \text{length} * \text{gaze vector}$$

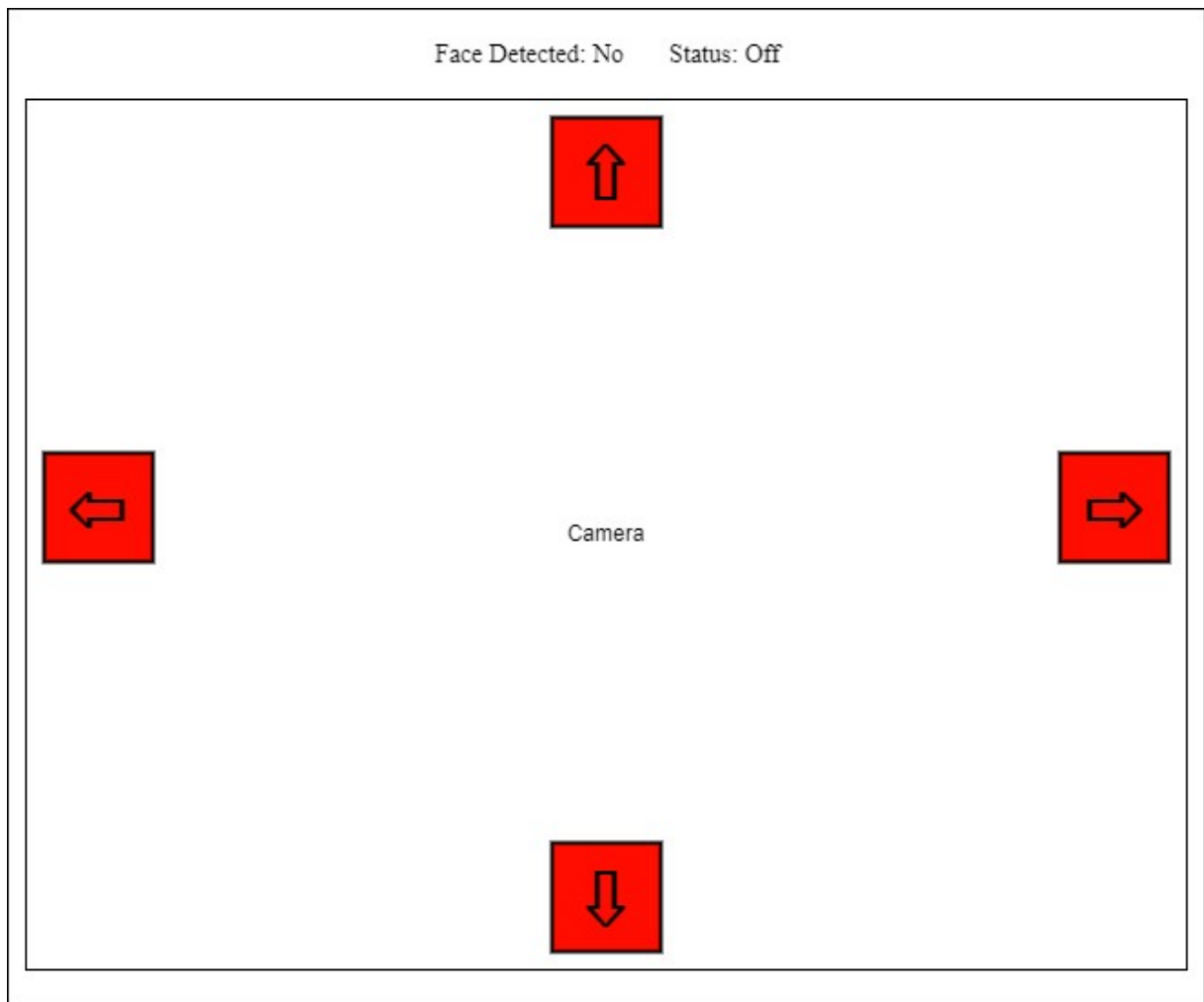
Trong đó, $\text{face}_{\text{center}}$ là điểm trung tâm của khuôn mặt, gaze vector là hướng nhìn sau khi đảo ngược chuẩn hóa.



Hình 31: Biểu diễn hướng nhìn trên hình ảnh đầu ra

3. Xây dựng chương trình điều khiển xe lăn

Sau khi xây dựng module ước tính hướng nhìn từ mô hình đã huấn luyện, tiến hành xây dựng chương trình điều khiển xe lăn bằng cách trực diện hóa giao diện. Giao diện chương trình được phát thảo như sau:

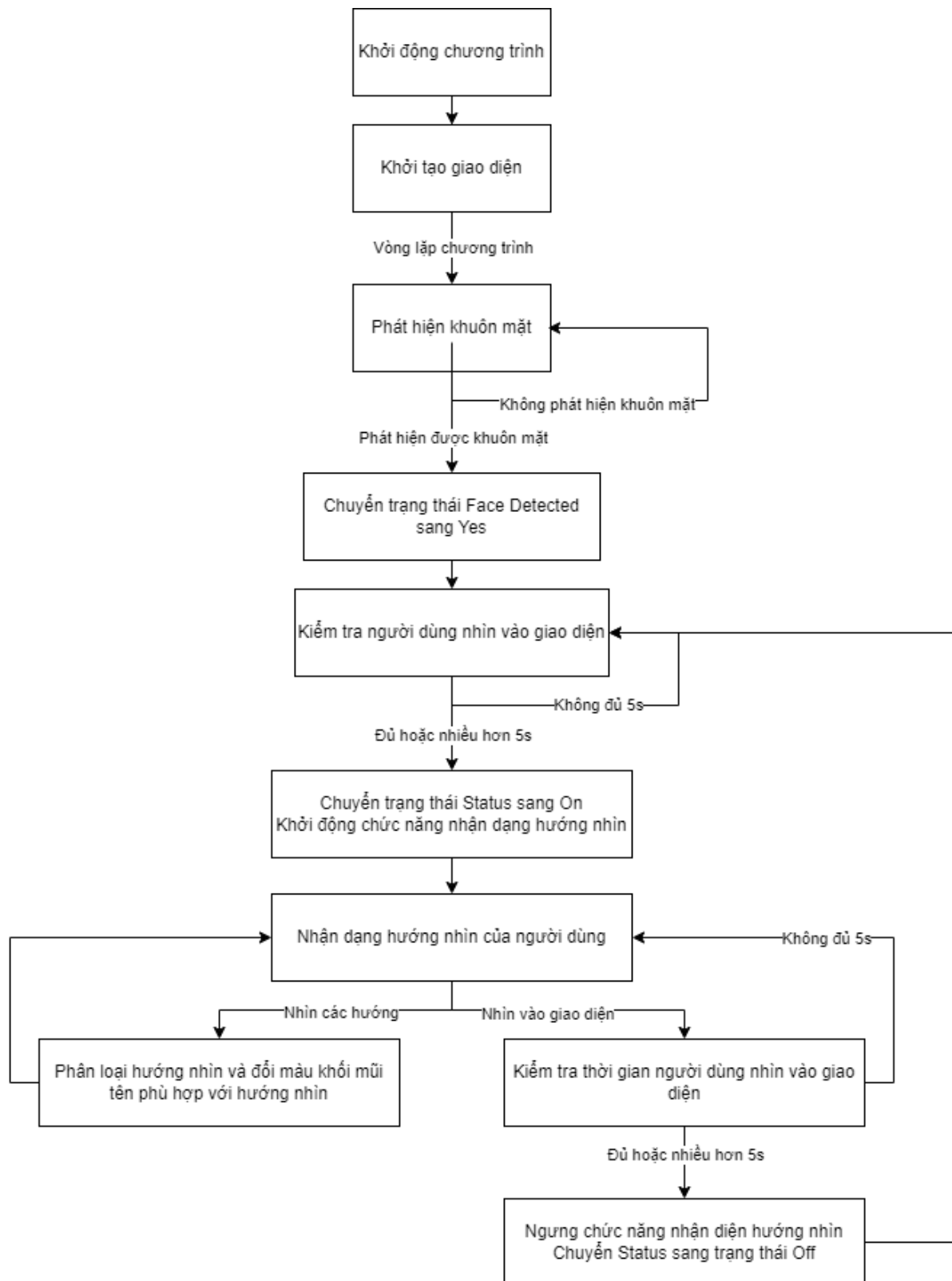


Hình 32: Phát thảo giao diện chương trình điều khiển xe lăn

Các thành phần của giao diện bao gồm:

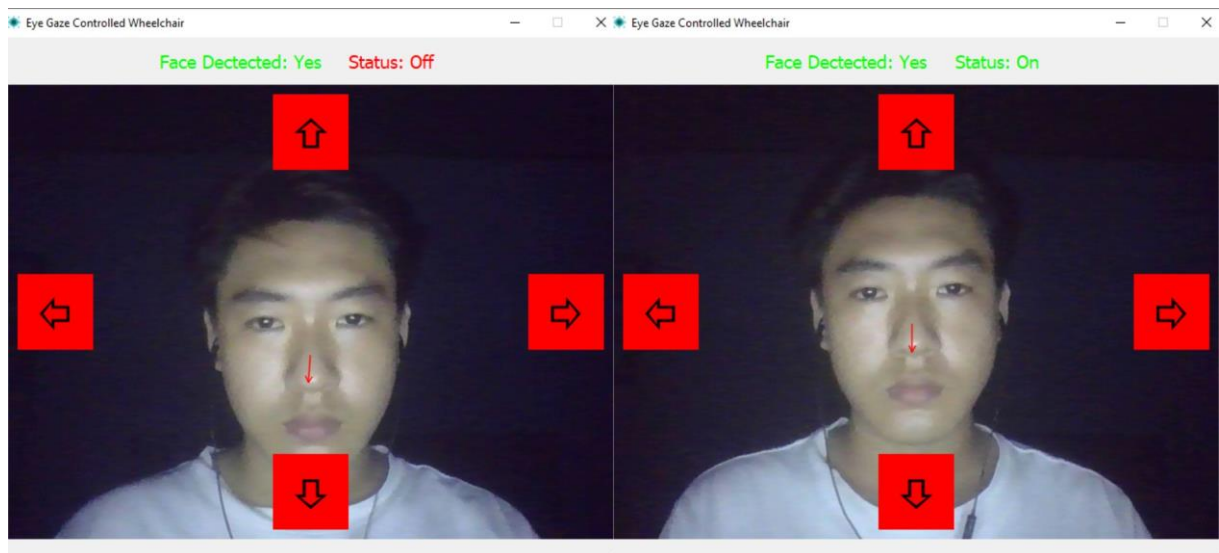
- Face Detected: Thông báo chương trình có phát hiện được khuôn mặt nào hay không, nếu phát hiện được thì thông báo Yes, ngược lại thông báo No.
- Status: Thông báo trạng thái của hoạt động của chương trình, nếu phát hiện được khuôn mặt của người dùng và người dùng nhìn vào giao diện trong khoảng 5 giây, chương trình sẽ được kích hoạt và trạng thái Status sẽ chuyển thành On, lúc này chức năng nhận diện hướng nhìn sẽ được thực thi. Nếu Status đang ở trạng thái On, người dùng có thể tắt chương trình bằng cách nhìn vào giao diện khoảng 5 giây, lúc này Status sẽ chuyển sang trạng thái Off và chức năng nhận dạng hướng nhìn cũng được tắt.
- Khởi mũi tên: Đại diện cho hướng nhìn của người dùng khi chức năng nhận diện hướng nhìn được thực thi.

Sơ đồ hoạt động của chương trình:



Hình 33: Sơ đồ hoạt động của chương trình điều khiển xe lăn

Sử dụng thư viện PyQt5 trong Python để thực hiện hóa chương trình, chương trình sau khi hoàn thiện có giao diện như sau:



Hình 34: Giao diện hoàn chỉnh của chương trình điều khiển xe lăn

Khi người dùng nhìn vào giao diện tối thiểu 5 giây, Status sẽ chuyển sang trạng thái On và chức năng nhận dạng hướng nhìn được thực thi.

Các trạng thái hướng nhìn của người dùng được biểu diễn như sau:



Hình 35: Hướng nhìn người dùng biểu diễn trên chương trình

4. Chế tạo sản phẩm

4.1. Chế tạo xe lăn điện

Xe lăn được sử dụng để chế tạo xe lăn điện là xe lăn truyền thống Lucass X9:



Hình 36: Xe lăn Lucass X9

(https://www.moby.com.vn/xe-lan-tay-lucass-x-9-tieu-chuan_16899.html)

Động cơ được sử dụng để điều khiển 2 bánh xe sau của xe lăn là 2 động cơ giảm tốc 12V-50W:



Hình 37: Động cơ giảm tốc 12V-50W

Đề cấp điện cho động cơ hoạt động sử dụng bình ắc quy 12V-30Ah:



Hình 38: Bình ắc quy 12V-30Ah

Ngoài ra để kết nối giữa động cơ và bánh xe cần 2 bộ nhông sên dĩa:

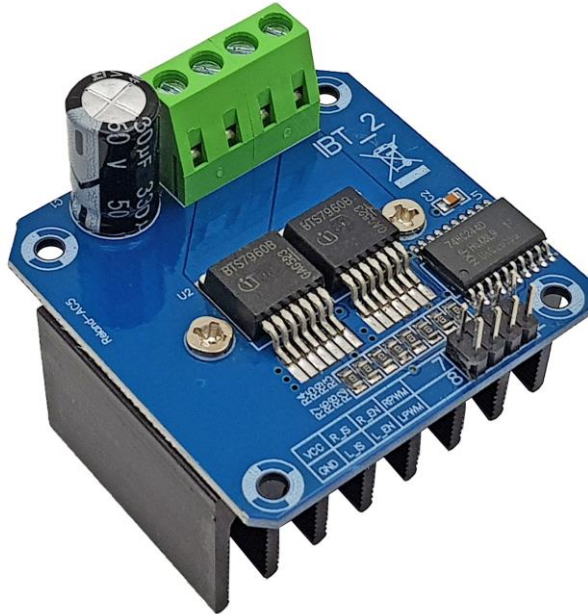


Hình 39: Bộ nhông sên dĩa

Tiến hành lắp đặt và hàn các nguyên liệu vào xe lăn để xe lăn có thể hoạt động bằng động cơ.

4.2. Kết nối chương trình điều khiển xe lăn bằng hướng mắt với xe lăn điện

Để điều khiển động cơ xe lăn bằng chương trình điều khiển, sử dụng 2 mạch điều khiển động cơ BTS7960 43A để điều động cơ (mỗi mạch điều khiển một động cơ), một mạch Arduino Uno R3 để truyền tín hiệu điều khiển từ chương trình đến 2 mạch điều khiển động cơ.



Hình 40: Mạch điều khiển động cơ BTS7960 43A

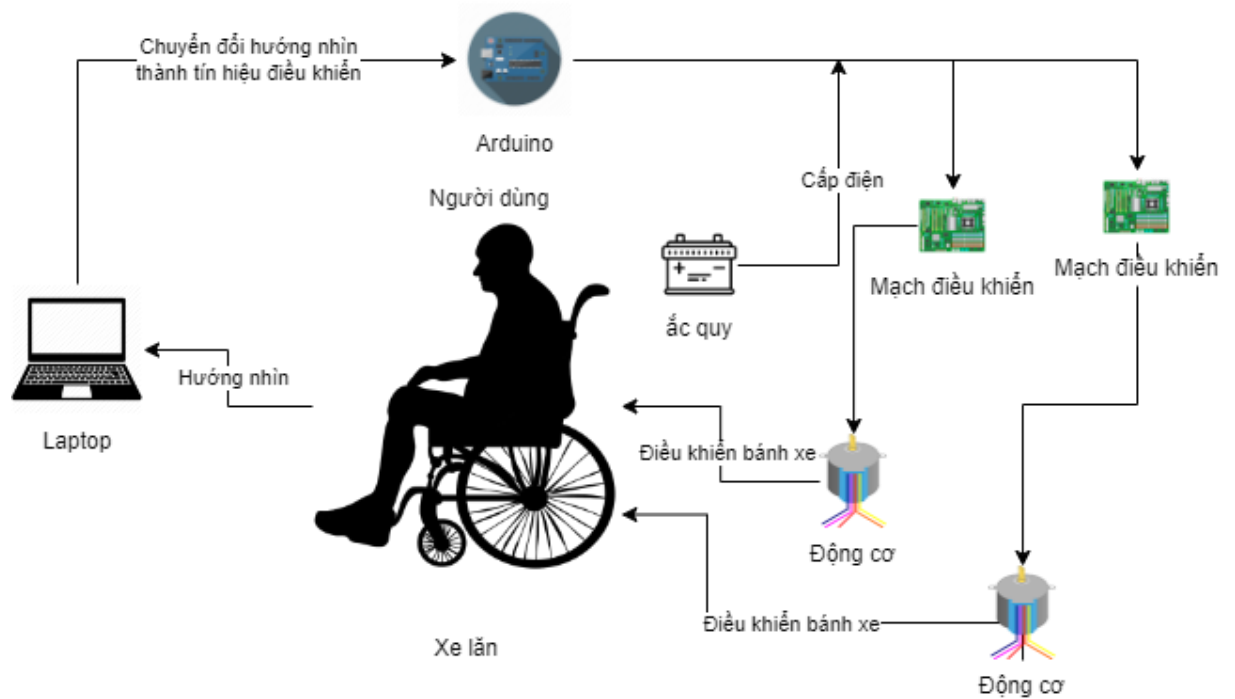
(<https://nshopvn.com/product/mach-dieu-khien-dong-co-dc-bts7960-43a-1-dong-co>)



Hình 41: Mạch Arduino Uno R3

(<https://nshopvn.com/product/arduino-uno-r3-dip-kem-cap/?variant=100977>)

Tiến hành kết nối giữa chương trình điều khiển và xe lăn điện, sản phẩm cuối cùng hoạt động theo sơ đồ sau:



Hình 42: Sơ đồ hoạt động của xe lăn điều khiển bằng mắt

Sản phẩm sau khi hoàn thiện:



Hình 43: Sản phẩm sau khi hoàn thiện

5. Kết quả và đánh giá

5.1. Kết quả

Chế tạo thành công xe lăn có thể điều khiển bằng hướng mắt với tỉ lệ chính xác khá cao, có thể hỗ trợ cho người khuyết tật nói chung và một bộ phận người khuyết tật không thể sử dụng tay để điều khiển xe lăn thông thường nói riêng.

5.2. Đánh giá

Tuy xe lăn có thể vận hành nhưng xe được nhóm nghiên cứu tự chế tạo từ các linh kiện nên tính chính xác về mặt di chuyển và các vấn đề liên quan như sức chịu tải của xe, tuổi thọ của xe,... sẽ không được như các mẫu xe điện được tính toán và chế tạo ở các dây chuyền sản xuất hàng loạt.

PHẦN 3: KẾT LUẬN VÀ KIẾN NGHỊ

1. Kết quả đóng góp

1.1. Về khoa học và đào tạo

Chế tạo được xe lăn điều khiển bằng hướng mắt mà không cần sử dụng tay để điều khiển và không cần người hỗ trợ di chuyển trong quá trình xe đang vận hành. Tính năng nhận diện hướng mắt của sản phẩm tạo tiền đề, làm cơ sở để chế tạo các thiết bị và phần mềm khác để hỗ trợ cuộc sống con người nói chung và người khuyết tật nói riêng.

Áp dụng các công nghệ học sâu, các kỹ thuật về xử lý ảnh trong thị giác máy tính trong việc xây dựng phần mềm và chế tạo sản phẩm.

1.2. Về phát triển kinh tế

Với chi phí chế tạo sản phẩm gần 4.000.000 đồng, rẻ hơn so với các mẫu xe điện điều khiển bằng tay hiện tại, giúp việc sản phẩm dễ dàng trong việc tiếp cận đến bộ phận người khuyết tật có hoàn cảnh khó khăn. Sản phẩm có thể giúp bộ phận người khuyết tật thoải mái trong việc di chuyển mà không cần quá nhiều sự trợ giúp từ gia đình và người thân, có thể làm những việc họ muốn, nếu những việc đó liên quan đến kinh tế sẽ là một phần trong việc giúp thúc đẩy phát triển kinh tế.

1.3. Về xã hội

Sản phẩm giúp cuộc sống của bộ phận người khuyết tật được cải thiện hơn, có thể di chuyển không cần quá nhiều sự giúp đỡ từ người thân và gia đình. Góp phần xoa dịu nỗi đau của những người khuyết tật cũng như cải thiện đời sống tinh thần của họ. Ngoài ra, những người khuyết tật cũng sẽ cảm thấy tự tin hơn và không còn cảm thấy bản thân là gánh nặng cho gia đình và xã hội.

2. Triển khai ứng dụng

Thiết bị có thể triển khai thí nghiệm tại các trung tâm người khuyết tật.

Chương trình được công bố dưới dạng mã nguồn mở trên Internet.

3. Hướng phát triển

Dự kiến trong tương lai, nhóm nghiên cứu sẽ cải thiện sản phẩm cả về mặt phần mềm và phần cứng. Tăng độ chính xác trong việc nhận dạng hướng mắt trong nhiều điều kiện ngoại cảnh khác nhau. Tính toán lại trong việc chế tạo sản phẩm để cải thiện sản phẩm khi sử dụng trên nhiều môi trường và địa hình khác nhau.

TÀI LIỆU THAM KHẢO

[1] [What are convolutional neural networks?](#)

(<https://www.ibm.com/topics/convolutional-neural-networks>)

[2] [Convolutional Neural Networks cheatsheet](#)

(<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>)

[3] [Dive into Deep Learning](#)

(<https://www.d2l.ai>)

[4] [OpenCV: Camera Calibration](#)

(https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html)

[5] [Appearance-based Gaze Estimation in the Wild \(MPIIGaze\)](#)

(<https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/appearance-based-gaze-estimation-in-the-wild>)

[6] [It's Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation](#)

(<https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/its-written-all-over-your-face-full-face-appearance-based-gaze-estimation>)

[7] [Revisiting Data Normalization for Appearance-Based Gaze Estimation](#)

(<https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/revisiting-data-normalization-for-appearance-based-gaze-estimation>)

[8] [PYTORCH DOCUMENTATION](#)

(<https://pytorch.org/docs/stable/index.html>)

GIẤY PHÉP TẬP DỮ LIỆU

- It's Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation, X. Zhang, Y. Sugano, M. Fritz and A. Bulling, Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops(CVPRW), 2017.
- [PDF](#)
- @inproceedings{zhang2017s,
title={It's written all over your face: Full-face appearance-based gaze estimation},
author={Zhang, Xucong and Sugano, Yusuke and Fritz, Mario and Bulling, Andreas},
booktitle={Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on},
pages={2299--2308},
year={2017},
organization={IEEE}
}

MINH CHỨNG SẢN PHẨM




Hình 44: Xe lăn điều khiển bằng mắt

Thực nghiệm tại trường Công nghệ thông tin & Truyền thông – Đại học Cần Thơ



**BẢN TIN ĐỀ TÀI
NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

- Mã số đề tài: TSV2023-125
 - Tên đề tài: Nghiên cứu chế tạo xe lăn điều khiển bằng mắt
 - Thời gian thực hiện: 06 tháng
 - Kinh phí đã sử dụng: 15.000.000 VNĐ
 - Chủ nhiệm đề tài: Lương Đức Huy (0358669057)
 - Thành viên tham gia nghiên cứu:
 - Nguyễn Huy Cường
 - Kim Minh Thắng
 - Cán bộ hướng dẫn: PGS. TS. Phạm Nguyên Khang (pnkhang@cit.ctu.edu.vn)
- 
- The image shows a young man with dark hair, wearing a blue and purple long-sleeved shirt and black pants, sitting in a blue and silver wheelchair. He is looking at a laptop screen which displays a video call interface. The laptop is placed on a small stand next to the wheelchair. The background is a plain, light-colored wall.
- Tính cấp thiết: Việc ứng dụng các công nghệ học sâu, thị giác máy tính vào các công trình nghiên cứu, sản phẩm giúp thúc đẩy việc công nghệ hóa và hiện đại hóa đất nước. Khi ứng dụng vào việc chế tạo xe lăn điều khiển bằng mắt sẽ giúp cải thiện đời sống vật chất cũng như tinh thần của bộ phận người khuyết tật, kém may mắn
 - Mục tiêu: Ứng dụng các công nghệ học sâu, thị giác máy tính vào việc chế tạo xe lăn điều khiển bằng mắt
 - Phương pháp nghiên cứu: Nghiên cứu lý thuyết => Lên ý tưởng => Chế tạo sản phẩm => Kiểm thử => Thực nghiệm
 - Nội dung nghiên cứu: Các công nghệ học sâu, thị giác máy tính, lập trình Arduino
 - Kết quả đạt được: Chế tạo được xe lăn có thể điều khiển bằng hướng nhìn của mắt
 - Ý nghĩa: Mở rộng phạm vi phục vụ cho người khuyết tật, những người không thể điều khiển xe bằng tay. Giúp họ không phụ thuộc quá nhiều vào người thân và gia đình trong quá trình sử dụng sản phẩm, cải thiện đời sống vật chất và tinh thần cũng như giúp họ không còn cảm thấy tự ti và là gánh nặng của gia đình và xã hội
 - Khả năng ứng dụng: Sản phẩm có thể triển khai thực nghiệm ở các trung tâm người khuyết tật để cải tiến dần dần độ chính xác. Khi kết quả thực nghiệm đạt mức tốt có thể đưa vào sản xuất để phục vụ cho những người khuyết tật. Ngoài ra việc nhận diện hướng mắt của sản phẩm có thể ứng dụng vào những công trình nghiên cứu khác, tạo tiền đề, làm cơ sở để chế tạo các chương trình và sản phẩm khác nhằm mục đích phục vụ đời sống con người nói chung và hỗ trợ những người khuyết tật, kém may mắn nói riêng



**BÁO CÁO TÓM TẮT
ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

1. Thông tin chung:

- Tên đề tài: Nghiên cứu chế tạo xe lăn điều khiển bằng mắt
- Sinh viên chủ nhiệm đề tài: Lương Đức Huy
- Lớp: Khoa học máy tính 01 K46
- Khoa: Trường Công nghệ thông tin & Truyền thông
- Năm thứ: 4 Số năm đào tạo: 4/4,5
- Người hướng dẫn: PGS. TS. Phạm Nguyên Khang

2. Mục tiêu đề tài:

Sử dụng công nghệ học sâu kết hợp với thị giác máy tính để xây dựng phần mềm nhận dạng hướng nhìn của mắt.

Sử dụng lập trình nhúng, lập trình Arduino để thiết kế mạch điều khiển nhận tín hiệu từ phần mềm nhận dạng hướng mắt đã xây dựng để điều khiển bánh xe.

Tích hợp phần mềm nhận dạng hướng nhìn của mắt và mạch điều khiển vào xe lăn truyền thống để điều khiển hướng di chuyển của xe lăn bằng mắt.

3. Tính mới và sáng tạo:

Ứng dụng công nghệ học sâu kết hợp thị giác máy tính vào việc nhận diện hướng nhìn của mắt.

Chế tạo sản phẩm xe lăn có thể vận hành không cần điều khiển bằng tay, mở rộng phạm vi hỗ trợ cho những người khuyết tật.

4. Kết quả nghiên cứu:

Tìm hiểu công nghệ học sâu và thị giác máy tính, xây dựng phần mềm nhận dạng hướng nhìn của mắt.

Tìm hiểu lập trình nhúng, lập trình Arduino, cách giao tiếp giữa hệ điều hành máy tính và mạch Arduino.

5. Sản phẩm:

TÊN SẢN PHẨM	SỐ LƯỢNG
Xe lăn điều khiển bằng mắt	01
Bản tin	01
Báo cáo tóm tắt	01
Video clips	01

6. Công bố khoa học từ kết quả nghiên cứu của đề tài, hoặc nhận xét, đánh giá của cơ sở đã áp dụng các kết quả nghiên cứu (nếu có): Không

7. Đóng góp về mặt kinh tế - xã hội, giáo dục và đào tạo, an ninh, quốc phòng và khả năng áp dụng của đề tài:

7.1. Đối với lĩnh vực giáo dục và đào tạo

Phát triển mở rộng phạm vi nghiên cứu về các lĩnh vực trí tuệ nhân tạo, thị giác máy tính.

7.2. Đối với lĩnh vực khoa học và công nghệ có liên quan

Thúc đẩy các sinh viên có thêm động lực làm nghiên cứu khoa học, góp phần làm đa dạng hóa các công trình nghiên cứu của trường Đại học Cần Thơ.

7.3. Đối với phát triển kinh tế - xã hội

Khi sản phẩm được ứng dụng vào thực tế sẽ góp một phần không nhỏ trong việc cải thiện đời sống của một bộ phận những người tàn tật, thiếu may mắn, giúp họ tự tin hòa nhập cuộc sống và có một cuộc sống trọn vẹn. Cải thiện được tinh thần của con người cũng đồng nghĩa cải thiện chất lượng sống của quốc gia, thế giới.

7.4. Đối với tổ chức chủ trì và các cơ sở ứng dụng kết quả nghiên cứu

Việc khai thác và sử dụng ứng dụng sẽ giúp người kém may mắn có cơ hội tiếp cận đến các sản phẩm thông minh giúp cuộc sống dễ dàng hơn.

7.5. Về khả năng áp dụng của đề tài:

Ngoài xe lăn, có thể áp dụng kết quả nhận diện hướng mắt của đề tài để áp dụng điều khiển các thiết bị, sản phẩm khác. Ví dụ như: máy tính, điện thoại, TV,... Giúp những người khuyết tật có thể điều khiển các thiết bị xung quanh mà chỉ cần dùng mắt, giúp cuộc sống những người khuyết tật tiện lợi và cải thiện hơn.

8. Hiệu quả, phương thức chuyển giao kết quả nghiên cứu và khả năng áp dụng:

8.1. Phương thức chuyển giao:

Chuyển giao trực tiếp sản phẩm cho Trường Công nghệ Thông tin & Truyền thông – Trường Đại học Cần Thơ quản lý và sử dụng.

8.2. Địa chỉ ứng dụng:

Trường Công nghệ Thông tin & Truyền thông - Trường Đại học Cần Thơ. Khu 2, đường 3/2, Phường Xuân Khánh, Q. Ninh Kiều, TP. Cần Thơ, Việt Nam.

8.3. Hiệu quả, khả năng áp dụng:

Sản phẩm có thể triển khai thí nghiệm tại các trung tâm người khuyết tật. Việc khai thác và sử dụng ứng dụng sẽ giúp người kém may mắn có cơ hội tiếp cận đến các sản phẩm thông minh giúp cuộc sống dễ dàng hơn.