



ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

**NHẬN DIỆN VẬT THỂ TRONG ẢNH  
NHẬN DIỆN HƯỚNG NHÌN TRONG ẢNH**

HỘI ĐỒNG: Khoa học máy tính 1  
GVHD: TS. NGUYỄN ĐỨC DŨNG  
GVPB: TS. TRẦN GIANG SƠN  
—00—  
SVTH: NGÔ THỊ TIỀN (1413986)

### **Lời cam đoan**

Nhận diện hướng nhìn trong ảnh (Nhận diện vật thể trong ảnh) không phải là một đề tài mới nhưng vẫn là một thách thức bởi: trong các ứng dụng: việc nhận diện hướng nhìn của con người qua hình ảnh đòi hỏi kết quả chính xác cao, ở Việt Nam, hiện tại không thực sự có nhiều nghiên cứu chuyên sâu về đề tài. Trong quá trình nghiên cứu đề tài có rất nhiều kiến thức không nằm trong chương trình giảng dạy ở bậc Đại học tuy vậy chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi dưới sự hướng dẫn của tiến sĩ Nguyễn Đức Dũng. Nội dung nghiên cứu và các kết quả đều là trung thực và chưa từng được công bố trước đây. Các số liệu được sử dụng cho quá trình phân tích, nhận xét được chính tôi thu thập từ nhiều nguồn khác nhau và sẽ được ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, tôi cũng có sử dụng một số nhận xét, đánh giá và số liệu của các tác giả khác, cơ quan tổ chức khác. Tất cả đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kì sự gian lận nào, tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình. Trường đại học Bách Khoa thành phố Hồ Chí Minh không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện.

### **Lời cảm ơn / Lời ngỏ**

Để hoàn thành kì đề cương luận văn này, tôi tỏ lòng biết ơn sâu sắc đến tiến sĩ Nguyễn Đức Dũng đã hướng dẫn tận tình trong suốt quá trình nghiên cứu.

Chúng tôi chân thành cảm ơn quý thầy, cô trong khoa Khoa Học Vật Kỹ Thuật Máy Tính, trường đại học Bách Khoa thành phố Hồ Chí Minh đã tận tình truyền đạt kiến thức trong những năm chúng tôi học tập ở trường. Với vốn kiến thức tích lũy được trong suốt quá trình học tập không chỉ là nền tảng cho quá trình nghiên cứu mà còn là hành trang để bước vào đời một cách tự tin.

Cuối cùng, tôi xin chúc quý thầy, cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý.

## **Tóm tắt nội dung**

Nội dung chính của luận văn nhằm tìm hiểu, nghiên cứu xây dựng hệ thống nhận diện hướng nhìn thông qua ảnh chụp dựa trên những công trình, công nghệ mới được nghiên cứu và phát triển trong những năm gần đây của lĩnh vực Deep Learning. Trong quá trình nghiên cứu, tôi đã tiến hành tổng hợp, đánh giá ưu và nhược điểm của cách phương pháp, công nghệ đã và đang được nghiên cứu, sử dụng. Tiếp cận vấn đề theo nhiều hướng khác nhau, tôi thực hiện một số phương pháp sử dụng học sâu (CNN) để phát hiện hướng nhìn của con người qua hình ảnh. Bên cạnh việc hoàn thành nội dung của đề tài, nhóm chúng tôi đã nghiên cứu thêm một số phần để từ đó đặt nền móng cho các nghiên cứu sau này. Phần còn lại của luận văn tập trung vào việc đánh giá mô hình, kết quả đạt được, đồng thời phân tích ưu nhược điểm của mô hình thực hiện và thảo luận những vấn đề mà mô hình còn gặp phải. Cuối cùng, nhóm chúng tôi đề xuất hướng phát triển tiếp theo của đề tài trong tương lai.

# Mục lục

<b>1 Giới thiệu</b>	<b>1</b>
1.1 Giới thiệu đề tài . . . . .	1
1.2 Mục tiêu của đề tài . . . . .	1
1.3 Cấu trúc luận văn . . . . .	2
<b>2 Tổng quan</b>	<b>3</b>
2.1 Hiệu quả đánh máy bằng mắt với 9-hướng Gaze Estimation [1] . . . . .	3
2.2 Appearance-Based Gaze Estimation in the Wild [2] . . . . .	5
2.3 Rendering of Eyes for Eye-Shape Registration and Gaze Estimation [3]	6
2.4 Learning an appearance-based gaze estimator from one million synthesised images [4] . . . . .	8
<b>3 Kiến thức nền tảng</b>	<b>13</b>
3.1 Khái niệm chung . . . . .	13
3.2 Convolution Neural Network [9] . . . . .	14
3.2.1 Các thành phần cơ bản của mạng CNN . . . . .	14
3.2.2 Kiến trúc mạng CNN cơ bản . . . . .	18
3.3 Các kiến trúc mạng CNN nâng cao . . . . .	21
3.3.1 Mạng LeNet-5 . . . . .	21
3.3.2 Mạng AlexNet . . . . .	22
3.3.3 Mạng GoogLeNet . . . . .	24
3.3.4 Mạng ResNet [18] . . . . .	26
<b>4 Hướng tiếp cận và mô hình đề xuất</b>	<b>27</b>
4.1 Hướng tiếp cận . . . . .	27
4.2 Công cụ sử dụng . . . . .	27
<b>5 Thực nghiệm và đánh giá mô hình</b>	<b>29</b>
5.1 Chuẩn bị dữ liệu, nghiên cứu, phân tích đề tài . . . . .	29
5.1.1 Chuẩn bị kiến thức nền tảng . . . . .	29
5.1.2 Nghiên cứu kỹ thuật dõi chuyển động mắt hiện nay . . . . .	29
5.1.3 Chuẩn bị và xử lý dữ liệu . . . . .	32
5.2 Tiến hành thực hiện . . . . .	34
5.2.1 Tập dữ liệu GazeCaptureEyeTracking . . . . .	35
5.2.2 Tập dữ liệu MPIIFaceGaze . . . . .	36
5.3 Đánh giá kết quả, mô hình . . . . .	38
<b>6 Thảo luận</b>	<b>39</b>

# **Danh sách bảng**

# Danh sách hình vẽ

2.1	Ảnh hưởng sau khi tăng chỉnh sáng . . . . .	3
2.2	Ứng dụng mô hình CNN trong phát hiện hướng nhìn . . . . .	4
2.3	Bảng thống kê kết quả . . . . .	4
2.4	Xử lý đầu vào . . . . .	5
2.5	Hình ảnh được đưa qua mạng học sâu để xử lý . . . . .	6
2.6	Bảng thống kê kết quả . . . . .	6
2.7	Chúng tôi tạo ra một số lượng lớn các hình ảnh photorealistic của mắt bằng cách sử dụng một mô hình vùng mắt động. . . . .	7
2.8	Tổng quan về quá trình chuẩn bị mô hình. . . . .	7
2.9	Mô hình mắt và mô hình đầu (3D). . . . .	8
2.10	Vị trí máy ảnh. . . . .	8
2.11	Sự biến đổi từ việc thay đổi ánh sáng được mô hình hóa với ảnh chụp có môi trường có phạm vi năng động cao. . . . .	9
2.12	Ước lượng lỗi trên MPIIGaze . . . . .	9
2.13	Ví dụ về độ phù hợp (fits) với SynthesEyes eye-CLNF . . . . .	9
2.14	Biểu đồ hiệu suất trên MPIIGaze . . . . .	10
2.15	Sử dụng phương pháp tiếp cận hàng xóm gần nhất(nearest-neighbor approach) để ước lượng cái nhìn. . . . .	10
2.16	Hình ảnh về lưới mắt: (a) được hiển thị với các vật liệu dựa trên vật lý và các hiệu ứng khúc xạ, mô hình co rút đồng tử (b) và giãn nở (c). . . . .	11
2.17	Mô hình khúc xạ iris bằng cách thay đổi kết cấu tra cứu: Một điểm ảnh được xem khúc xạ chính xác để hiển thị màu đen (tròng mắt) thay vì màu xanh (bề mặt hình học). . . . .	11
2.18	Mô hình khởi tạo vùng mắt: (a) Cho thấy cấu trúc liên kết vùng mắt chung của chúng ta (229 đỉnh) trên một quét thô (khoảng 5M đỉnh). (b) Cho thấy cấu trúc liên kết trong không gian kết cấu uv với các vòng cạnh quan trọng được đánh dấu. . . . .	11
2.19	Kết quả của phương pháp tổng hợp những hình ảnh cận cảnh thực tế (Nearest-neighbour pairs) được dán nhãn hoàn hảo của mắt người . . . . .	12
2.20	Lỗi pixel và điểm nhìn của phương pháp nearest-neighbor tương ứng với tập UnityEyes và MPIIGaze. Nhóm nguyên cứu đạt được lỗi thấp nhất ở 1.4 triệu hình ảnh. . . . .	12
3.1	Đặc trưng - Feature trong CNN [9] . . . . .	14
3.2	Bộ lọc trong CNN [9] . . . . .	15
3.3	Tầng Convolution trong CNN [6] . . . . .	15
3.4	Ví dụ mẫu tính toán tích chập [9] . . . . .	15
3.5	Max Pooling trong CNN [8] . . . . .	17
3.6	Các hàm truyền trong CNN [13] . . . . .	17

3.7	Tầng kết nối đầy đủ [10] . . . . .	18
3.8	Kiến trúc mạng CNN cơ bản [20] . . . . .	19
3.9	Nhận diện hình ảnh với CNN [11] . . . . .	19
3.10	Mô hình mạng LeNet [7] . . . . .	22
3.11	Mạng AlexNet [7] . . . . .	23
3.12	Mô hình mạng GoogLeNet [17] . . . . .	25
3.13	Module Inception [18] . . . . .	25
3.14	Một block ResNet [18] . . . . .	26
4.1	Một số khái niệm trong TensorFlow [19] . . . . .	28
5.1	Camera Video thông thường và Camera hồng ngoại có đèn LED hồng ngoại. [23] . . . . .	30
5.2	Phân loại sự vận động của mắt [5] . . . . .	30
5.3	Mối quan hệ giữa hướng nhìn và tư thế đầu [5] . . . . .	31
5.4	Sơ đồ của một hệ thống theo dõi hướng nhìn. P là đồng tử của bóng mắt người và G là vị trí phản chiếu được hình thành trên giác mạc, được chụp trên mặt phẳng camera Hình ảnh này cho thấy lỗi trong ước tính ánh nhìn cũng như độ lệch giữa các vị trí nhìn thực tế và ước tính. [5] .	31
5.5	Mô hình 3D: a. Mô hình bóng mắt người, thông số mắt và các yếu tố thiết lập được sử dụng trong theo dõi mắt 3D. Trục quang được hiển thị như là đường thẳng nối giữa tâm của giác mạc với trung tâm đồng tử mắt. Trục thị giác đi qua fovea và tâm của đường cong giác mạc. Góc Kappa là độ lệch góc giữa trục quang và thị giác. b. Một mô hình phi cầu của giác mạc, như một bề mặt của chuyển đổi về trục quang của mắt. [5]	31
5.6	Phương pháp dựa trên hình dạng: Mẫu của một vùng mắt: Xc, Yc, Ye đại diện cho trung tâm của đồng tử và mắt tương ứng P1 và P2 là tiêu điểm của hai phần parabol và a, b, c và là tham số, r là bán kính của đồng tử [5] . . . . .	32
5.7	Tập dữ liệu ảnh MPIIFaceGaze. . . . .	32
5.8	Tập dữ liệu ảnh GazeCaptureEyeTracking: thư mục frames chứa các file hình ảnh, và các file Json. . . . .	33
5.9	Tập dữ liệu ảnh GazeCaptureEyeTracking: các file hình ảnh trong thư mục frames. . . . .	33
5.10	Các hướng nhìn [1] . . . . .	34
5.11	Mô hình CNNs sử dụng training tập dữ liệu GazeCaptureEyeTracking. .	35
5.12	Mô hình (model) thực hiện phát hiện ánh nhìn . . . . .	36
5.13	Kết quả nhận được tập ảnh chứa các hình ảnh cắt mắt phải, trái . . . . .	36
5.14	Kết quả thu được (dữ liệu ảnh MPIIFaceGaze). . . . .	37
5.15	Tập dữ liệu ảnh MPIIFaceGaze. . . . .	37

# **List of Algorithms**

# **Chương 1**

## **Giới thiệu**

### **1.1 Giới thiệu đề tài**

Sự tương tác giữa con người và máy tính ngày càng trở nên quan trọng khi các thiết bị thông minh như điện thoại, máy tính, máy tính bảng ngày càng gia tăng. Khoa học nghiên cứu về nó cũng được chú trọng phát triển theo, trong đó việc phát hiện hướng nhìn của người dùng thông qua đôi mắt trở thành một lĩnh vực trong thị giác máy tính ngày nay.

Qua các hình ảnh thu thập bằng máy quay trên thiết bị thông minh như điện thoại hay máy tính, ta có thể phát hiện được mục tiêu hay đối tượng mà người dùng đang hướng tới màn hình. Ứng dụng này có thể giúp người khuyết tật điều khiển các thiết bị chỉ bằng ánh mắt mà không cần đến đôi tay.

Tương tự như vậy khi gắn một máy quay nhỏ trên xe hơi theo dõi người lái xe, dựa vào hướng nhìn đôi mắt của tài xế mà ta có thể phát hiện kịp thời người này ngủ gật hay không chú ý khi lái xe (khi ngủ gật mắt sẽ trùng xuống, khi không tập trung mắt sẽ hướng đi nơi khác).

Trong một lớp học hay một buổi dự thảo ta có thể đo được độ tập trung của người tham gia để kịp thời thay đổi nội dung cho phù hợp hơn, tránh lãng phí thời gian của mọi người.

Ngoài ra, việc dự đoán chính xác hướng nhìn của đôi mắt có thể phân tích hành vi mua hàng của khách như khách hay tập trung vào tầm nhìn nào nhiều nhất mà ta có thể sắp xếp mặt hàng quan trọng vừa tầm. Hay đơn giản như việc lướt web của người dùng Internet để sắp xếp nội dung quan trọng hay không quan trọng vào các vị trí phù hợp.

Đó là tầm quan trọng của việc "dự đoán hướng nhìn" ứng dụng vào thực tế mà nhóm tiến hành nghiên cứu và hiện thực hóa đề tài.

### **1.2 Mục tiêu của đề tài**

Mục tiêu của đề tài là nghiên cứu, hiểu và hiện thực một số phương pháp học sâu để phát hiện hướng nhìn của con người qua hình ảnh.

Một số vấn đề đặt ra:

- Làm thế nào để giải quyết bài toán trên?
- Cách tiếp cận như thế nào?

- 
- Những công nghệ nào đã và hiện đang được sử dụng?
  - Hướng cải tiến?...

Như vậy để thực hiện theo đúng mục tiêu của đề tài cần xác định một số công việc phải giải quyết như sau:

- Tìm kiếm và thu thập dữ liệu phù hợp với nội dung đề tài.
- Tìm hiểu các phương pháp tiếp cận đã được hiện thực
- Lựa chọn mô hình phù hợp
- Lên kế hoạch hiện thực, phát triển hệ thống nhận diện huấn luyện và kiểm thử.

### 1.3 Cấu trúc luận văn

Trong giai đoạn luận văn đề tài nhóm đã thực hiện được một số công việc liên quan sẽ trình bày trong báo cáo như sau:

- Chương 1: Giới thiệu tổng quan về nhận diện hướng nhìn, cũng là chương hiện hành. Trong chương này sẽ đưa đến cái nhìn tổng quát về đề tài, tiềm năng và ứng dụng thực tế trong tương lai.
- Chương 2: Tổng quan một số công trình nghiên cứu liên quan tới đề tài mà nhóm tìm hiểu được qua ba phần: hướng tiếp cận, mô hình sử dụng và kết quả đạt được.
- Chương 3: Các kiến thức nền tảng phục vụ như mạng nơ-ron tích chập trong học sâu (Deep Learning), các kiến trúc mạng nâng cao của mạng tích chập.
- Chương 4: Trình bày hướng tiếp cận bài toán: chương này sẽ đi vào chi tiết cách tiếp cận thực hiện mô hình, bao gồm công cụ sử dụng.
- Chương 5: Tổng kết những công việc nhóm đã làm được, đánh giá và định hướng kế hoạch mà nhóm tiếp tục phát triển trong luận văn.

# Chương 2

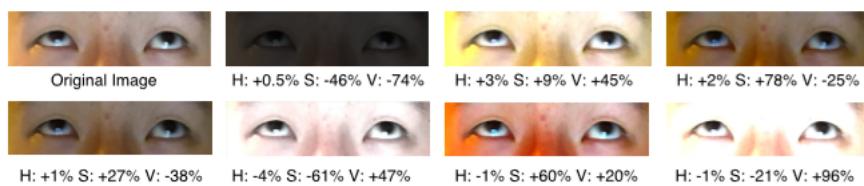
## Tổng quan

Qua quá trình thực hiện luận văn Tôi đã tìm hiểu một số công trình nghiên cứu, bài báo khoa học nước ngoài, Tôi chưa tìm được báo cáo khoa học nào trong nước đi sâu vào đề tài nhóm đang hiện thực.

### 2.1 Hiệu quả đánh máy bằng mắt với 9-hướng Gaze Estimation [1]

#### Hướng tiếp cận

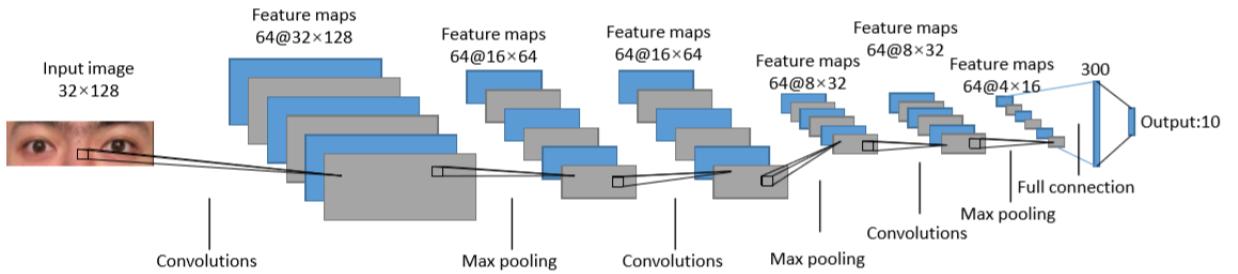
Đầu tiên nhóm nguyên cứu tiến hành lấy mẫu (dataset) thực hiện chụp ảnh trực tiếp hay quay video người làm mẫu thực hiện 3 lần (3 top). Chia hướng nhìn thành 9 hướng và nhắm mắt. Tiến hành xử lý mẫu (dataset), phát hiện vị trí mắt trong ảnh, tiến hành cắt ảnh. Sau đó, xử lý ảnh trước khi đưa vào model hiện thực như: chỉnh độ sáng ảnh sử dụng HSV,... Tiếp theo, hiện thực kết quả với mô hình CNN (Lenet-5, Support Vector Machine (SMV),...)



Hình 2.1: Ảnh hưởng sau khi tăng chỉnh sáng

#### Mô hình thuật toán sử dụng

Sử dụng các hình ảnh RGB với kích thước  $32 \times 128$  pixel làm đầu vào mạng. Đôi với lớp xoắn (convolution) đầu tiên, kích thước bộ lọc (filter) là  $3 \times 3$  và bộ lọc (filter) dịch chuyển dọc theo ảnh và quét toàn bộ ảnh. Số lượng các bộ lọc là 64. Sau lớp xoắn, tiếp theo là ReLU và lớp tổng hợp tối đa (max pooling) thu được kích thước  $64 \times 16 \times 64$ . Lặp lại hoạt động tương tự trên các lớp xoắn thứ hai và thứ ba. Lớp kết nối hoàn chỉnh (full connection) có 300 đơn vị ẩn và mỗi đơn vị có kết nối đến tất cả các bản đồ



Hình 2.2: Ứng dụng mô hình CNN trong phát hiện hướng nhìn

đặc trưng của lớp xoắn thứ ba. Đầu ra của mạng là mười loại (9 hướng mắt và nháy mắt), sử dụng chức năng softmax để tính toán tối ưu hóa.

### Kết quả đạt được

Bảng thống kê dữ liệu thu được sau hai lần nguyên cứu lấy mẫu.

model	Data Augmentation	Test set of known users		Test set of unknown users	
		Top1 Acc.	Top2 Acc.	Top1 Acc.	Top2 Acc.
Our model	No	82.44	95.61	78.34	95.85
Our model	Yes	95.01	99.80	91.71	98.39
Lenet-5	No	68.46	90.02	67.28	88.94
Lenet-5	Yes	80.44	94.81	81.57	93.32
SVM	No	70.66	87.62	70.51	89.63
Our model (SE)	Yes	76.27	93.31	73.15	92.84

Hình 2.3: Bảng thống kê kết quả

Nhóm nguyên cứu đã quan sát thấy các hình ảnh ước lượng sai và phát hiện ra hướng sai, ước tính thường gần đúng hướng. Suy luận rằng có thể nguyên nhân ước lượng không chính xác, do người dùng định hướng sai cạnh của hai hướng liền kề. Nếu tính toán lỗi trên lần 2 (đợt lấy mẫu 2), tỷ lệ lỗi sẽ giảm xuống còn 0.2%. Loại lỗi này có thể chấp nhận được vì khi sử dụng hệ thống để nhập từ (input words), người dùng luôn nhìn vào phần giữa của mọi hướng thay vì nhìn vào cạnh của hướng. Theo kết quả kiểm tra, mô hình với dữ liệu mắt đơn có tỷ lệ lỗi cao hơn. Từ tất cả các dữ liệu mắt người, ước tính sai, tìm thấy ba lý do sau đây: thứ nhất, dữ liệu mắt đơn nhạy cảm hơn với sự chuyển động nhẹ của đầu. Sự nghiêng đầu ảnh hưởng đến kết quả ước tính. Thứ hai, dưới độ phân giải thấp và điều kiện ánh sáng yếu, hình ảnh của mắt đơn có thể quá mờ để sử dụng ước lượng, ngay cả con người không thể nhận ra hướng. Thứ ba, mắt trái và phải không phải lúc nào cũng đồng bộ. Một số hình ảnh kết quả ước lượng của hai mắt là khác nhau nhưng cả hai đều hợp lý.

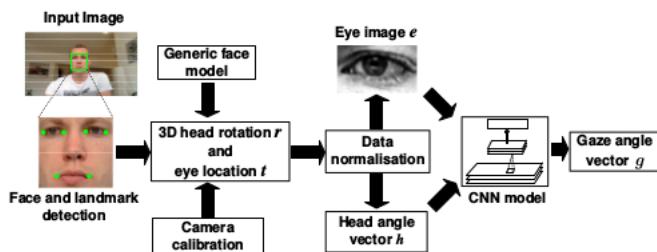
Để giải quyết những vấn đề tìm thấy, nhóm nguyên cứu đã thiết kế hệ thống bằng cách giải quyết cả vấn đề theo dõi mắt và các vấn đề nhập văn bản, phương pháp nhập văn bản dựa trên thị giác mà người dùng có thể nhập văn bản bằng chuyển động mắt và nhấp nháy, chọn các thanh điện thoại thông thường 9-key T9 bố trí bàn phím như giao diện. Một bộ dữ liệu quy mô lớn đã được thu thập bằng cách sử dụng một số thiết bị chụp và bao gồm các điều kiện ánh sáng khác nhau, địa điểm và thời gian. Sử dụng hình ảnh của cả hai mắt để tránh các vấn đề đồng bộ hóa hình ảnh mắt đơn, làm tăng độ chính xác ước lượng. Đối với các mục đích ứng dụng, xác định 9 hướng đi và học từ một mô hình CNN để đánh giá sự quan sát, có thể chính xác ước tính của người được

biết đến (người tham gia lấy mẫu lần 2 trở lên) và người lạ (người tham gia lấy mẫu đầu tiên). Các phương pháp tăng dữ liệu được dùng cung cấp tính mạnh mẽ và tăng độ chính xác ước lượng. Phương pháp nhập văn bản này sử dụng các thiết bị có chi phí thấp, đơn giản hóa các bước hiệu chỉnh và lọc tiếng ồn do đường cong, nháy mắt tự nhiên và ước lượng sai lệch tạm thời trong suốt quá trình hoạt động, ngoài ra có thể chạy ở hai chế độ: chế độ ngoài màn hình và chế độ màn hình. Ở chế độ ngoài màn hình, người dùng có thể tự di chuyển và thiết bị cầm tay. Tốc độ nhập văn bản của chế độ màn hình có thể đạt 20 chữ trong một phút và tắt màn hình chế độ có thể đạt 18 chữ trong một phút với tỷ lệ lỗi thấp. Công việc trong tương lai của nhóm nghiên cứu này là giới thiệu mô hình ngôn ngữ và thêm các chức năng dự đoán và hoàn thành từ cho hệ thống hiện tại và đào tạo ước tính quan sát bắt biến.

## 2.2 Appearance-Based Gaze Estimation in the Wild [2]

### Hướng tiếp cận

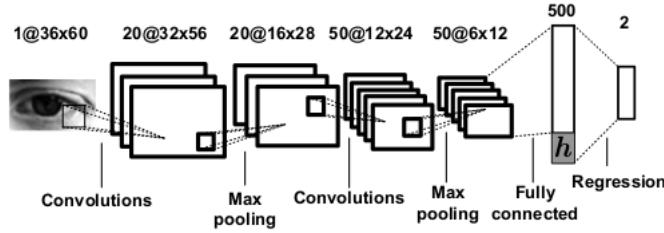
Hình ảnh đầu vào bước đầu được xử lý nhận diện khuôn mặt và điểm 6 điểm mốc (bao gồm hai điểm của mỗi mắt và hai mốc điểm miệng). Sau đó dùng khuôn mặt 3D chung để ước tính các vị trí 3D của các điểm mốc đã được phát hiện và áp dụng công nghệ kỹ thuật chuẩn hóa không gian để cắt và biến dạng đứng đầu và hình ảnh mắt vào không gian đào tạo bình thường. Bước cuối cùng CNN được đưa vào để học và đưa ra kết quả dự đoán.



Hình 2.4: Xử lý đầu vào

### Mô hình thuật toán sử dụng

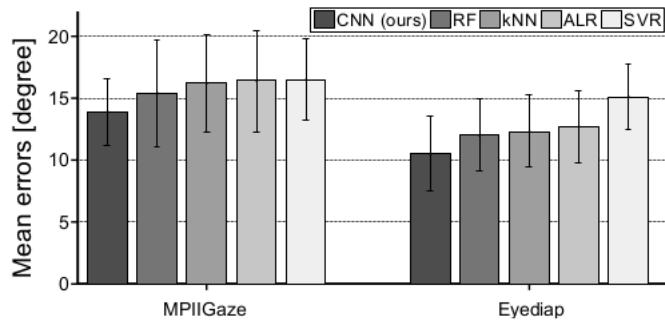
Trong bài báo khoa học này tác giả sử dụng mạng LeNet bao gồm một mạng tích chập (convolution) - hàm tổng hợp tối đa (max-pooling) - mạng tích chập - hàm tổng hợp tối đa - tầng kết nối đầy đủ (fully - connected). Lớp cuối cùng là một tuyến tính để đưa ra output cuối cùng. Đầu vào là hình ảnh màu xám có kích thước 60x36 pixels. Mỗi tích chập có kích thước feature 5x5 pixels, trong đó số feature của tầng thứ nhất là 20, tầng thứ 2 là 50. Đầu ra là hai góc nhìn, trái và phải.



Hình 2.5: Hình ảnh được đưa qua mạng học sâu để xử lý

### Kết quả đạt được

Bảng bên dưới là sai số trung bình đạt được với một số mô hình khác. Bao gồm Random Forests (RF), k-Nearest Neighbours(kNN), Adaptive Linear Regression (ALR), Support Vector Regression (SVR).



Hình 2.6: Bảng thống kê kết quả

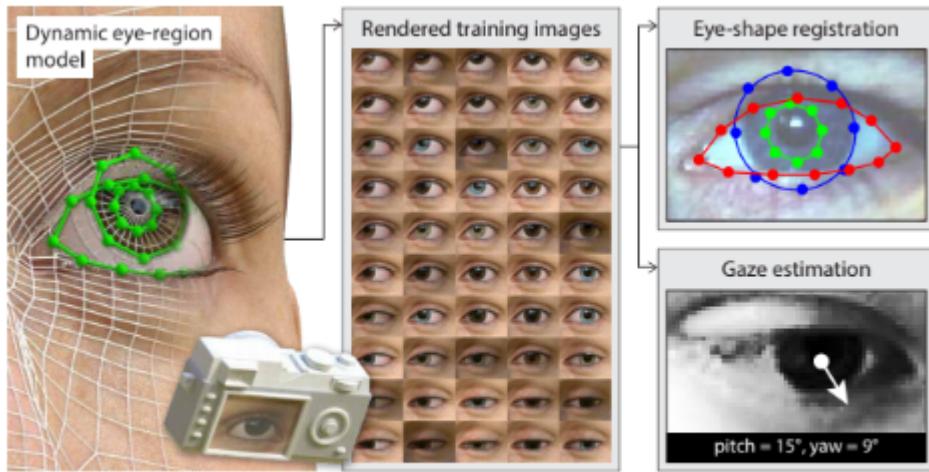
## 2.3 Rendering of Eyes for Eye-Shape Registration and Gaze Estimation [3]

Nhóm nghiên cứu tạo ra một số lượng lớn hình ảnh thực tế của mắt bằng cách sử dụng mô hình vùng mắt động. Chúng được sử dụng làm dữ liệu đào tạo để đăng ký hình dạng mắt và ước tính dựa trên sự xuất hiện dựa trên sự xuất hiện.

### Hướng tiếp cận

- (a) Quét hình ảnh 3D dày đặc (1,4 triệu đa giác (polygons))
- (b) Được tái hiện lại thành dạng tối ưu cho hoạt ảnh (9,005 đa giác (polygons))
- (c) Các chi tiết bề mặt da có độ phân giải cao được khôi phục bằng bản đồ dịch chuyển
- (d) Các điểm mốc mí mắt và mí mắt 3D được chú giải bằng tay
- (e) Hiển thị mẫu được hiển thị

Mô hình mắt nghiên cứu bao gồm các sclera, học sinh, móng mắt, và giác mạc (a) (the sclera, pupil, iris, and cornea) và có thể thể hiện sự thay đổi thực tế trong cả hai hình dạng (giãn nở đồng tử) và kết cấu (màu iris, tĩnh mạch scleral) (b).



**Figure 1:**

Hình 2.7: Chúng tôi tạo ra một số lượng lớn các hình ảnh photorealistic của mắt bằng cách sử dụng một mô hình vùng mắt động.

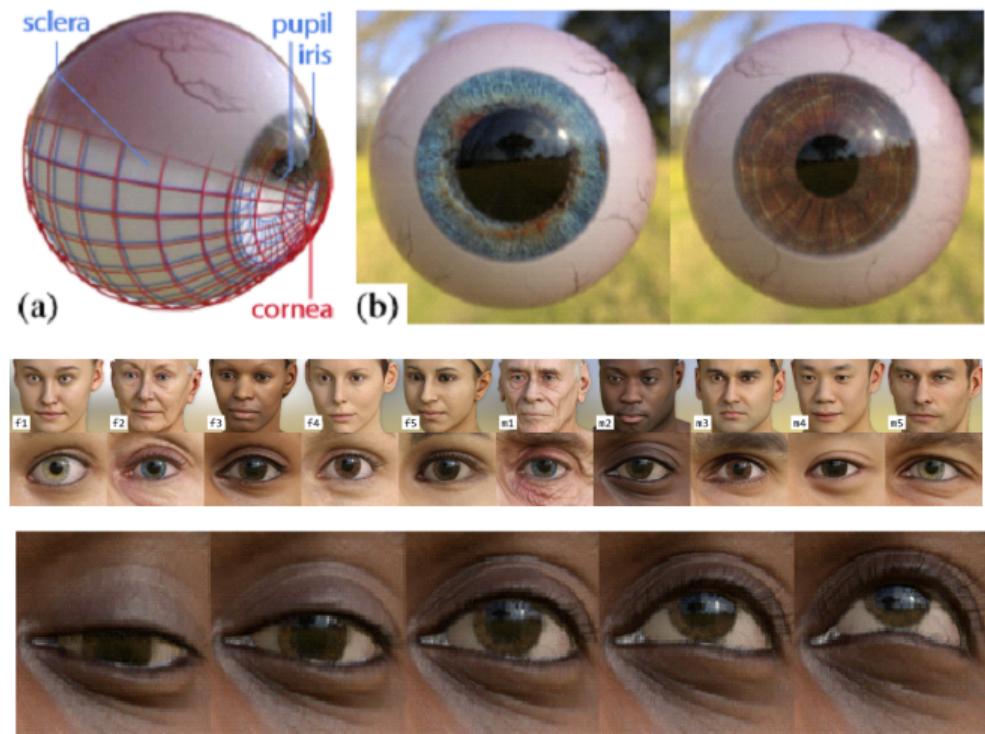


**Hình 2.8: Tổng quan về quá trình chuẩn bị mô hình.**

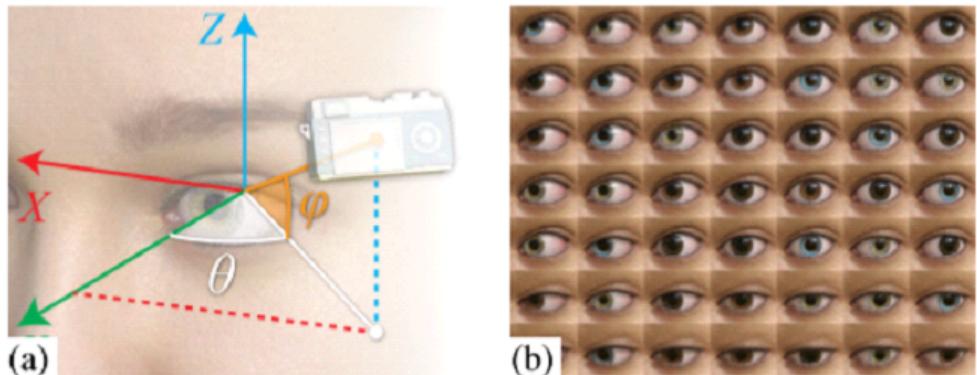
Máy ảnh được định vị để mô phỏng các thay đổi trong tư thế đầu (a). Tại vị trí này, chúng tôi hiển thị nhiều hình ảnh mắt cho các hướng nhìn khác nhau bằng cách đặt mô hình nhãn cầu (b):

Bốn bản đồ môi trường HDR, sử dụng cho ánh sáng thực tế: sáng / ngoài trời nhiều mây và sáng / tối trong nhà. Màu đỏ thể hiện điểm số tối tê nhất.

Nhóm nghiên cứu đã trình bày một phương pháp mới để tổng hợp những hình ảnh cận cảnh thực tế được dán nhãn hoàn hảo của mắt người. Một đường ống đồ họa máy tính sử dụng một bộ sưu tập các mô hình vùng mắt động thu được từ quét đầu để tạo ra các hình ảnh cho thấy phạm vi của các tư thế đầu, hướng nhìn và điều kiện chiếu sáng. Nhóm nghiên cứu đã chứng minh rằng phương pháp nghiên cứu hoạt động tốt hơn các phương pháp hiện đại để đăng ký hình dạng mắt và ước tính dựa trên sự xuất hiện dựa trên dữ liệu chéo trong tự nhiên. Những kết quả này hứa hẹn và nhấn mạnh tiềm năng quan trọng của các phương pháp tổng hợp học tập như vậy, đặc biệt là sự bứt phá với các phương pháp giám sát quy mô lớn gần đây. **Kết quả đạt được** Ví dụ về độ phù hợp (fits) với SynthesEyes eye-CLNF về hình ảnh trong tự nhiên (a) và hình ảnh webcam (b). Hai hàng hình ảnh trên cùng minh họa cho việc nhận dạng hình dạng mắt thành công, trong khi hàng dưới cùng minh họa các trường hợp thất bại, bao gồm các vệt chưa được tạo ra (tóc), các tư thế chưa được chỉnh sửa (mắt kín), kính và khói tạo mô hình không chính xác.



Hình 2.9: Mô hình mắt và mô hình đầu (3D).



Hình 2.10: Vị trí máy ảnh.

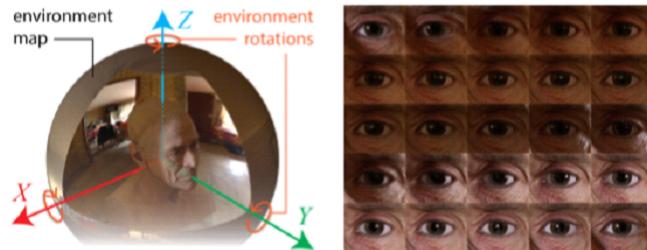
Trục x đại diện cho tập huấn luyện được sử dụng. Đầu châm là các lõi trung bình và đường màu đỏ thể hiện một giới hạn thấp hơn về mặt xác thực (số điểm xác thực chéo trong tập dữ liệu).

## 2.4 Learning an appearance-based gaze estimator from one million synthesised images [4]

**Hướng tiếp cận**



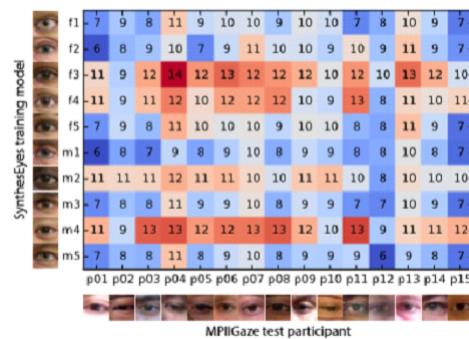
(a) The four HDR environment maps we use for realistic lighting: bright/cloudy outdoors, and bright/dark indoors



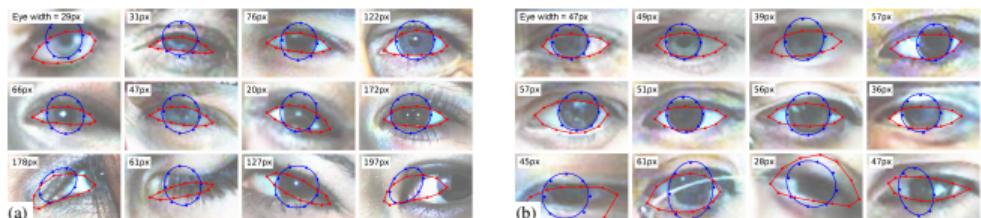
(b) The environment is rotated to simulate different head poses

(c) Renders using a single environment, rotated about  $Z$

Hình 2.11: Sự biến đổi từ việc thay đổi ánh sáng được mô hình hóa với ảnh chụp có môi trường có phạm vi năng động cao.



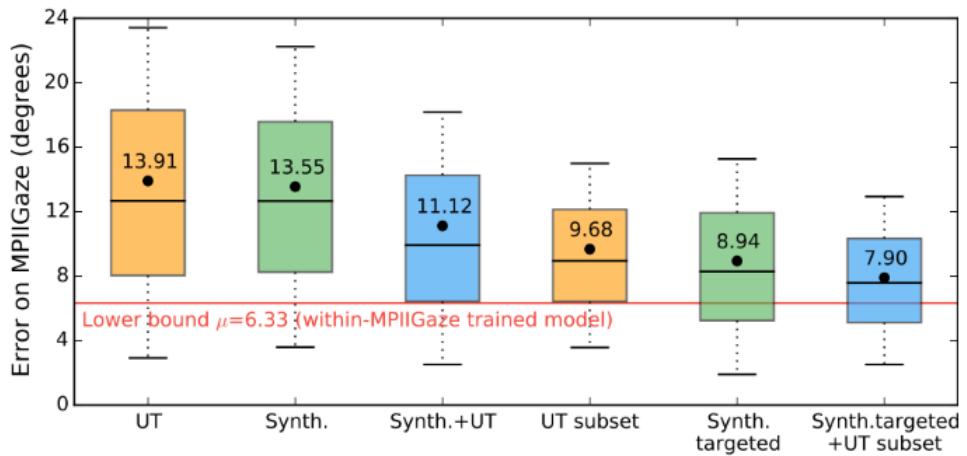
Hình 2.12: Ước lượng lỗi trên MPIIGaze



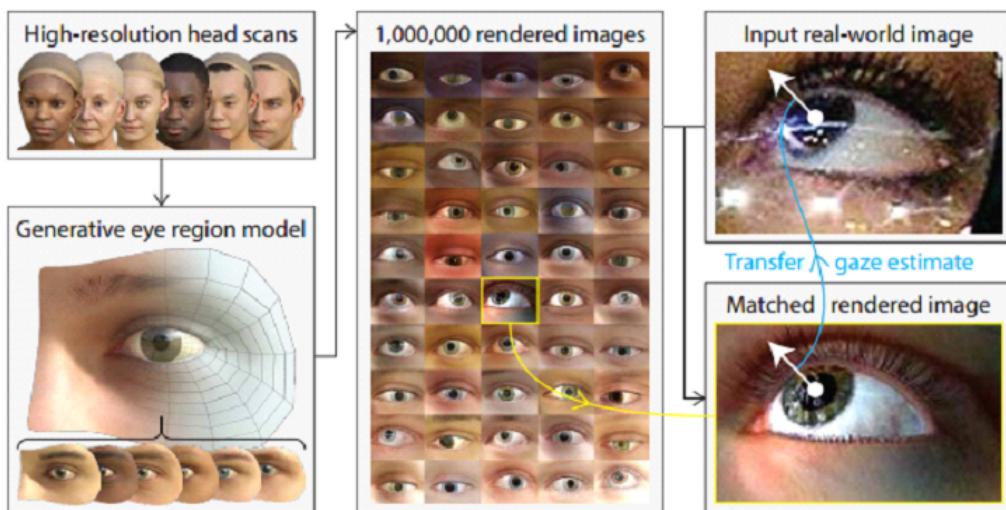
Hình 2.13: Ví dụ về độ phù hợp (fits) với SynthesEyes eye-CLNF

Nghiên cứu đã cung cấp một triệu hình ảnh thực tế về mắt bằng cách sử dụng mô hình vùng mắt được mô phỏng. Chúng được kết hợp với một hình ảnh đầu vào bằng cách sử dụng một phương pháp tiếp cận hàng xóm gần nhất (nearest-neighbor approach) để ước lượng cái nhìn. Mô hình quản lý này tìm ra các kết quả phù hợp ngay cả với các góc nhìn cực độ và ánh sáng chói từ kính.

Nhóm nghiên cứu trình bày về UnityEyes, một phương pháp mới để tổng hợp nhanh chóng số lượng lớn các hình ảnh vùng mắt thay đổi cho dữ liệu huấn luyện. Phương



Hình 2.14: Biểu đồ hiệu suất trên MPIIGaze

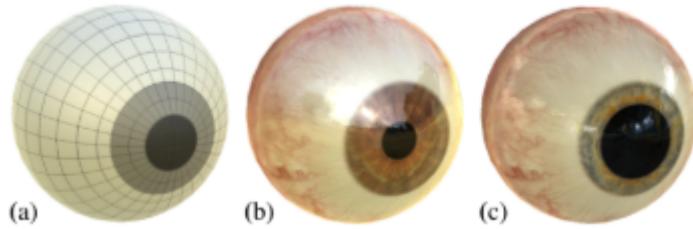


Hình 2.15: Sử dụng phương pháp tiếp cận hàng xóm gần nhất(nearest-neighbor approach) để ước lượng cái nhìn.

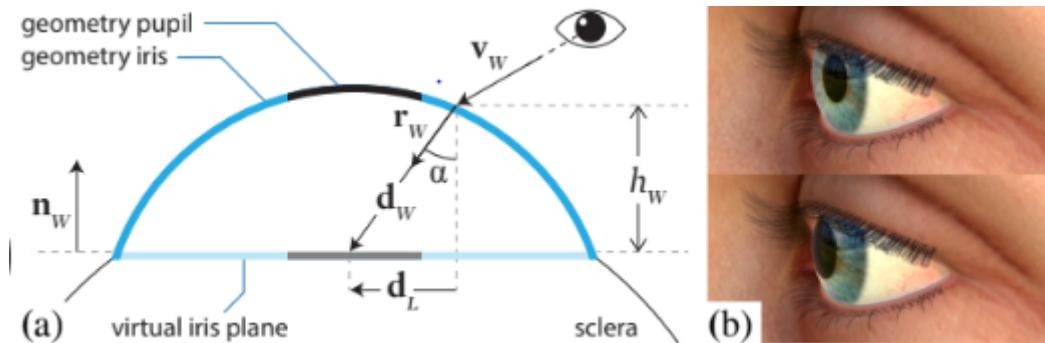
pháp nghiên cứu này: kết hợp một mô hình 3D sinh học mới lạ của vùng mắt người với khung thời gian thực. Mô hình vùng mắt có nguồn gốc từ quét khuôn mặt 3D có độ phân giải cao và sử dụng xấp xỉ thời gian thực cho các vật liệu và cấu trúc nhãn cầu phức tạp, cũng như các phương pháp hình học thủ tục lấy cảm hứng từ giải phẫu cho hoạt ảnh mí mắt. Tổng hợp hình ảnh bằng cách sử dụng trình kết xuất đồ họa và ánh sáng dựa trên hình ảnh thực tế và đa dạng điều kiện chiếu sáng. Những hình ảnh tổng hợp này có thể được khớp với hình ảnh đầu vào trong thế giới thực bằng cách sử dụng các phương pháp tiếp cận gần nhất để ước lượng ánh mắt.

### Kết quả đạt được

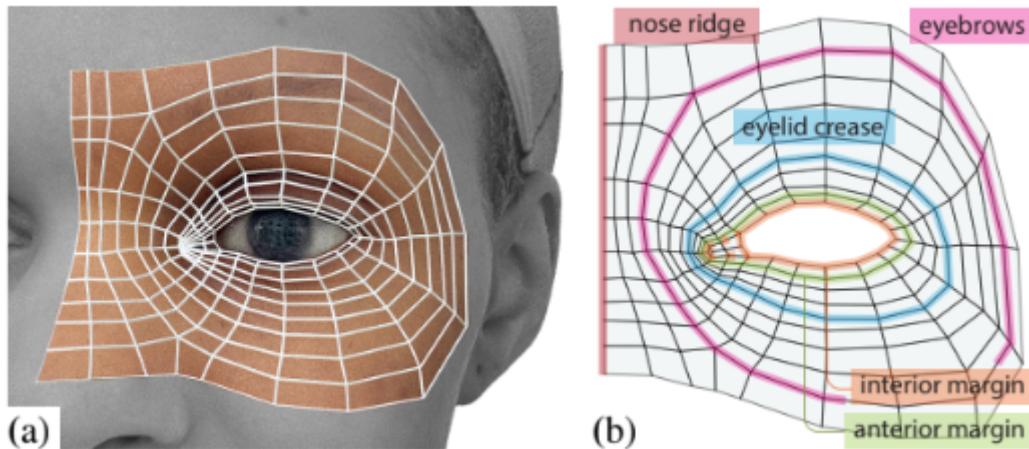
Nearest-neighbour pairs hiển thị hình ảnh tự nhiên (trên cùng) và phần kết của chúng tôi (dưới cùng) cùng với ánh mắt được ước tính (màu xanh lá cây). Ba hàng đầu



Hình 2.16: Hình ảnh về lưới mắt: (a) được hiển thị với các vật liệu dựa trên vật lý và các hiệu ứng khúc xạ, mô hình co rút đồng tử (b) và giãn nở (c).



Hình 2.17: Mô hình khúc xạ iris bằng cách thay đổi kết cấu tra cứu: Một điểm ảnh được xem khúc xạ chính xác để hiển thị màu đen (tròng mắt) thay vì màu xanh (bề mặt hình học).



Hình 2.18: Mô hình khởi tạo vùng mắt: (a) Cho thấy cấu trúc liên kết vùng mắt chung của chúng ta (229 đỉnh) trên một quét thô (khoảng 5M đỉnh). (b) Cho thấy cấu trúc liên kết trong không gian kết cấu uv với các vòng cạnh quan trọng được đánh dấu.

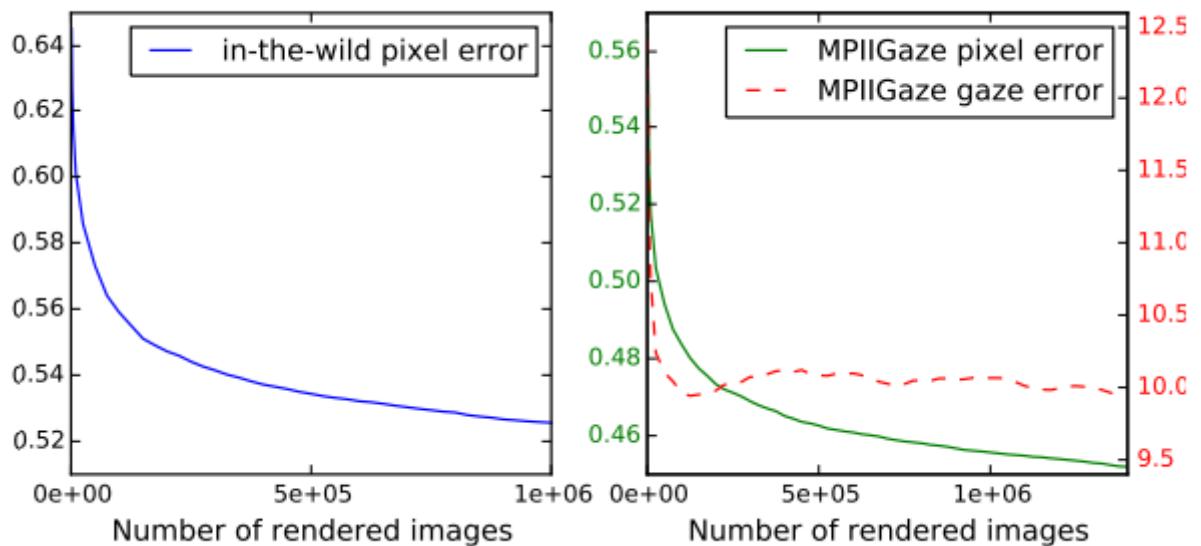
hiển thị ước tính chất lượng tốt, ngay cả dưới ánh sáng khó khăn (thiếu ánh sáng), độ phân giải thấp và góc nhìn cực đoan. Hàng dưới cùng hiển thị các trường hợp lỗi từ biến thể chưa được sửa đổi, ví dụ: trang điểm và tóc.

Lỗi pixel và điểm nhìn của phương pháp nearest-neighbor tương ứng với tập Uni-



Hình 2.19: Kết quả của phương pháp tổng hợp những hình ảnh cận cảnh thực tế (Nearest-neighbour pairs) được dán nhãn hoàn hảo của mắt người

tyEyes và MPIIGaze. Nhóm nguyên cứu đạt được lỗi thấp nhất ở 1.4 triệu hình ảnh.



Hình 2.20: Lỗi pixel và điểm nhìn của phương pháp nearest-neighbor tương ứng với tập UnityEyes và MPIIGaze. Nhóm nguyên cứu đạt được lỗi thấp nhất ở 1.4 triệu hình ảnh.

# Chương 3

## Kiến thức nền tảng

Để có thể nghiên cứu bài toán nhận diện vật thể trong ảnh, những kiến thức nền tảng quan trọng nhom cần phải nắm bao gồm:

- Convolution Neural Network.
- Các kiến trúc mạng CNN nâng cao

### 3.1 Khái niệm chung

Mạng nơ-ron nhân tạo (Artificial Neural Network- ANN) hay thường gọi ngắn gọn là mạng nơ-ron là một mô hình toán học hay mô hình tính toán được xây dựng dựa trên các mạng nơ-ron sinh học, bao gồm có một nhom các nơ-ron nhân tạo (nút: node) nối với nhau và xử lý thông tin bằng cách truyền theo các kết nối và tính giá trị mới tại các nút. Trong nhiều trường hợp, mạng nơ-ron nhân tạo là một hệ thống thích ứng (adaptive system) tự thay đổi cấu trúc của mình dựa trên các thông tin bên ngoài hay bên trong chảy qua mạng trong quá trình học.

Trong thực tế sử dụng, nhiều mạng nơ-ron là các công cụ mô hình hóa dữ liệu thống kê phi tuyến. Chúng có thể được dùng để mô hình hóa các mối quan hệ phức tạp giữa dữ liệu vào và kết quả hoặc để tìm kiếm các dạng/mẫu trong dữ liệu. ANN giống như bộ não con người, được học bởi kinh nghiệm (thông qua huấn luyện), có khả năng lưu giữ những kinh nghiệm tri thức và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết (unseen data). Kiến trúc chung của một mạng nơron nhân tạo (ANN) gồm 3 thành phần đó là: lớp đầu vào (input layer), lớp ẩn (hidden layer) và lớp đầu ra (output layer).

Khi ta áp dụng mạng ANN lên dữ liệu ảnh đầu vào có kích thước mxn. Với một hidden layer, có một nơ-ron ta sẽ có số trọng số  $w_{ij}$  phải lưu là mxn trọng số. Từ đó ta có thể thấy với một ảnh có kích thước 50x50 thì với mạng như trên ta đã phải nhớ đến 2500 trọng số  $w_{ij}$ . Thế nhưng ảnh bình thường lại có kích thước lớn hơn 50x50 và mạng ANN với một lớp ẩn và một nơ-ron thì kết quả học sẽ không được tốt. Vậy nên khi áp dụng ANN lên dữ liệu ảnh ta sẽ phải nhớ rất nhiều trọng số  $w_{ij}$ , vì vậy khó có thể tạo được một mạng có độ sâu đủ lớn để áp dụng lên dữ liệu ảnh.

Convolutional Neural Network (CNNs - Mạng neural tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao.

## 3.2 Convolution Neural Network [9]

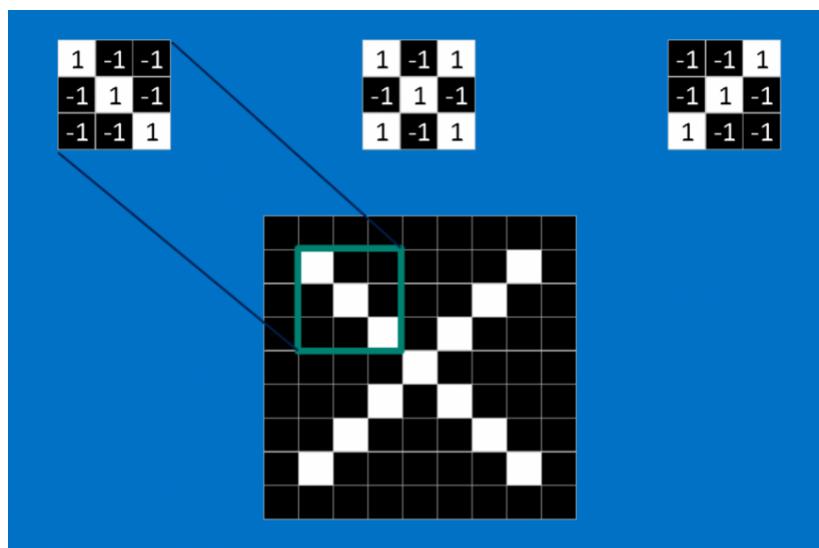
### 3.2.1 Các thành phần cơ bản của mạng CNN

Các thành phần cơ bản của CNN gồm:

- Tầng Convolution
- Pooling
- ReLU
- Fully Connected

Một vài khái niệm cần hiểu:

- Feature: CNNs so sánh hình ảnh theo từng mảng. Các mảng (thuộc tính đặc trưng) mà nó tìm được gọi là các feature. Bằng cách tìm ở mức độ các feature khớp nhau ở cùng vị trí trong hai hình ảnh, CNNs nhìn ra sự tương đồng tốt hơn nhiều so với việc khớp toàn bộ bức ảnh. Mỗi feature giống như một hình ảnh mini - một mảng hai chiều nhỏ. Các feature khớp với các khía cạnh chung của các bức ảnh.



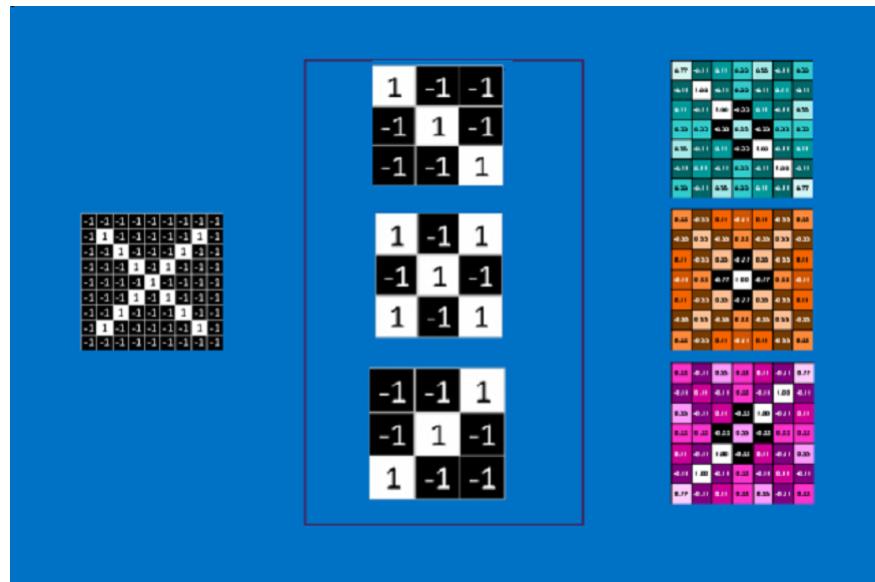
Hình 3.1: Đặc trưng - Feature trong CNN [9]

- Bộ lọc (filter): là một ma trận có kích thước nhỏ thường là  $3 \times 3$  hoặc  $5 \times 5$ , được khởi tạo ban đầu. Khi “trượt” filter trên ảnh đầu vào ứng với mỗi vùng trượt sẽ có một nơ-ron ẩn trong lớp ẩn (hidden layer) đầu tiên, quá trình này được lặp lại liên tục đến khi trượt xong toàn bộ bức ảnh cuối cùng ta sẽ được 1 feature map. Tuy nhiên trong thực tế chúng ta cần rất nhiều feature map.

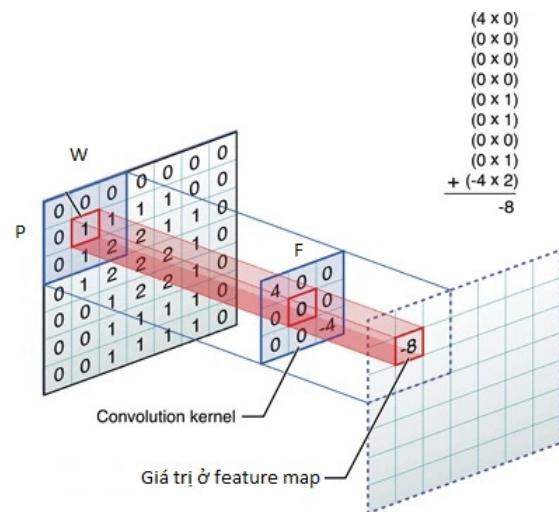
#### Tầng Convolution

Lớp convolution sử dụng bộ lọc (filters) có kích thước nhỏ hơn so với ảnh (thường ma trận  $3 \times 3$  hoặc  $5 \times 5$ ), tiến hành tích chập. Bộ filter sẽ dịch chuyển trên ảnh theo bước trượt chạy dọc theo ảnh và quét toàn bộ ảnh thu được feature map.

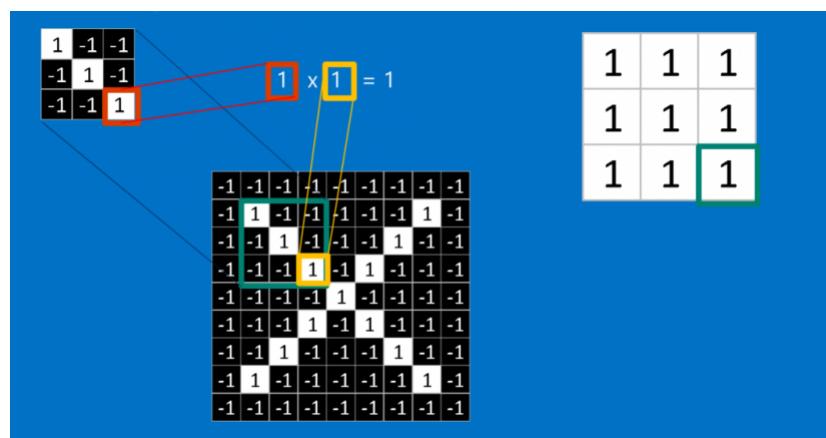
Tính toán sự khớp của một feature đối với một hình ảnh, ta thực hiện nhân mỗi điểm ảnh trong feature với giá trị của điểm ảnh tương ứng trong hình ảnh. Cộng tổng lại và chia cho số lượng điểm ảnh trong feature.



Hình 3.2: Bộ lọc trong CNN [9]



Hình 3.3: Tầng Convolution trong CNN [6]



Hình 3.4: Ví dụ mẫu tính toán tích chập [9]

Ví dụ trong hình mẫu: nếu giá trị điểm ảnh trong feature khớp màu với điểm ảnh trong hình sẽ cho giá trị là 1 ( $1 \times 1 = 1$  hoặc  $(-1) \times (-1) = 1$ ), nếu giá trị điểm ảnh không khớp màu sẽ cho giá trị  $(-1)$  ( $1 \times (-1) = -1$ ). Cộng các kết quả lại, chia cho số lượng điểm ảnh của feature  $3 \times 3 = 9$ .

Một mạng tích chập cần nhiều feature map, thực hiện tương tự quá trình tích chập cho từng feature khác, kết quả thu được là một tập hợp các hình ảnh được lọc, tương ứng với mỗi filter. Chú ý: các giá trị gần 1 cho thấy sự khớp mạnh, các giá trị gần -1 cho thấy sự khớp mạnh với âm bản của feature, và các giá trị gần bằng 0 cho thấy không khớp với bất kỳ loại nào.

Thực hiện tương tự quá trình tích chập cho từng feature khác, kết quả thu được là một tập hợp các hình ảnh được lọc, tương ứng với mỗi filter.

Có hai dạng tính Convolution chính mà ta thường dùng là Valid và Same. Hai dạng cho ảnh kết quả có kích thước khác nhau:

- Dạng Valid: kernel được trượt trên toàn ảnh, sau khi tính tích chập (convolution) kết quả sẽ nhỏ hơn kích thước ảnh ban đầu. Với ảnh đầu vào có kích thước  $m \times m$ , kernel kích thước  $n \times n$  thì ảnh đầu ra sẽ có kích thước  $(m-n+1) \times (m-n+1)$ .
- Dạng Same: trước khi tính Convolution, ảnh sẽ được đệm (padding) thêm các giá trị (thường các giá trị này sẽ là 0 hoặc 1) để tăng kích thước ảnh, sao cho sau khi thực hiện Convolution, kích thước ảnh kết quả sẽ bằng với kích thước ảnh ban đầu. Về mặt toán học, cho hàm  $f(x)$ , tích chập của  $f$  với hàm  $k(x)$  được tính như sau:

$$r(x) = (f * k)(x) = \int f(x-n)k(n)dn \quad (3.1)$$

Thực hiện "trượt" hàm  $k(x)$  lên hàm  $f(x)$  và tính tổng phần chồng lấp của 2 hàm này với nhau, do đó kết quả của phép tích chập là ta được một hàm mới là sự "hoà trộn" của  $k(x)$  vào  $f(x)$ . Trong trường hợp rời rạc, phép tích phân trở thành tổng:

$$r(i) = (f * k)(i) = \sum f(i-n)k(n) \quad (3.2)$$

Với trường hợp 2 chiều, ta chỉ cần thêm chỉ số cho chiều thứ 2:

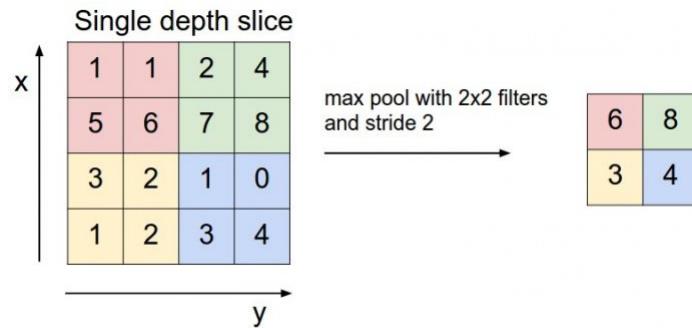
$$r(i, j) = (f * k)(i, j) = \sum \sum f(i-n, j-m)k(n, m) \quad (3.3)$$

## Lớp gộp Pooling

Pooling layer trong mạng CNN giúp loại bỏ những thông tin không cần thiết, rất hữu ích khi ảnh có kích cỡ lớn (dimensional reduction) làm giảm độ phức tạp khi tính toán. Tuy nhiên nếu lạm dụng nhiều có thể làm mất dữ liệu. Trong lớp này sử dụng cửa sổ trượt (thường là  $2 \times 2$ ), mỗi lần trượt theo một bước trượt (stride) là 2. Khác với lớp Convolutional, lớp Pooling không tính tích chập mà tiến hành lấy mẫu (subsampling). Một giá trị đại diện thông tin ảnh tại vùng ảnh đó được giữ lại. Một số phương pháp phổ biến được sử dụng trong lớp Pooling là MaxPooling (lấy giá trị lớn nhất), MinPooling (lấy giá trị nhỏ nhất) và AveragePooling (lấy giá trị trung bình).

Ví dụ minh họa: mô hình có kích thước  $32 \times 32$  (hình minh họa trên), lớp Pooling dùng filter kích thước  $2 \times 2$ , bước trượt (stride) là 2, sử dụng phương pháp MaxPooling. Tiến hành gộp ảnh, giá trị lớn nhất trong vùng cửa sổ  $2 \times 2$  giới hạn bởi filter được giữ lại, làm đầu ra. Sau lớp Pooling, kích thước ảnh giảm đi 2 lần mỗi chiều ( $16 \times 16$ ).

Khi dữ liệu ảnh có kích thước lớn, thực hiện qua nhiều lớp Pooling sẽ thu nhỏ, giảm kích thước dữ liệu làm giảm lượng tham số, tăng hiệu quả tính toán và góp phần kiểm soát hiện tượng quá khớp (overfitting).



Hình 3.5: Max Pooling trong CNN [8]

### Rectified Linear Units - ReLU

Hay còn gọi là hàm tinh chỉnh các đơn vị tuyến tính

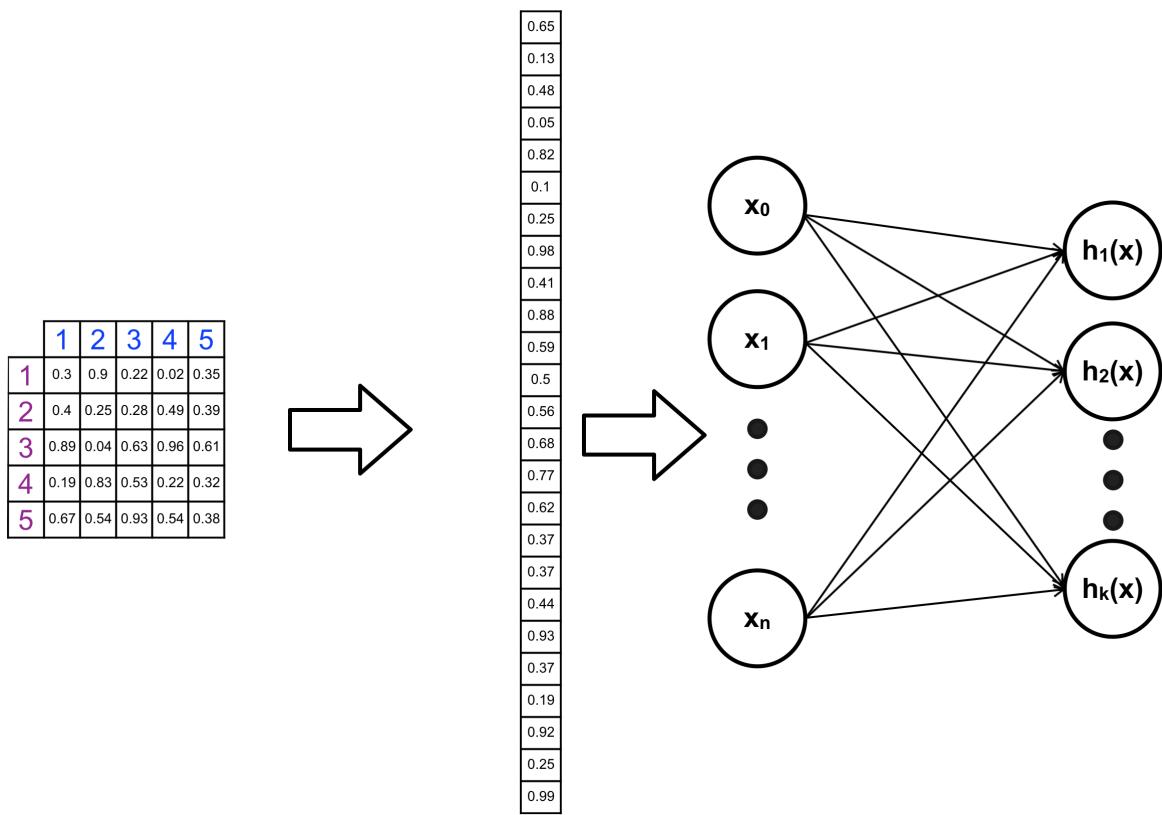
Đóng vai trò nhỏ nhưng quan trọng trong quá trình là Rectified Linear Unit hoặc ReLU. Lớp này thường được cài đặt sau lớp Convolutional. Có nhiệm vụ chuyển toàn bộ giá trị âm của kết quả từ lớp Convolutional thành giá trị 0, sử dụng hàm kích hoạt  $f(x) = \max(0, x)$ . ReLU giúp tạo nên tính phi tuyến cho mô hình, giữ vững sự tin cậy toán học giúp các giá trị khỏi bị mắc kẹt gần 0 hoặc bị về vô tận.

Ngoài ra còn một số hàm truyền khác:

Hàm Truyền	Công thức	Đồ thị
Linear	$f(x) = x$	
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$	
Tan-Sigmoid	$f(x) = \frac{1 - e^x}{1 + e^x}$	
ReLU	$f(x) = \max(0, x)$	

Hình 3.6: Các hàm truyền trong CNN [13]

## Fully-connected Layer



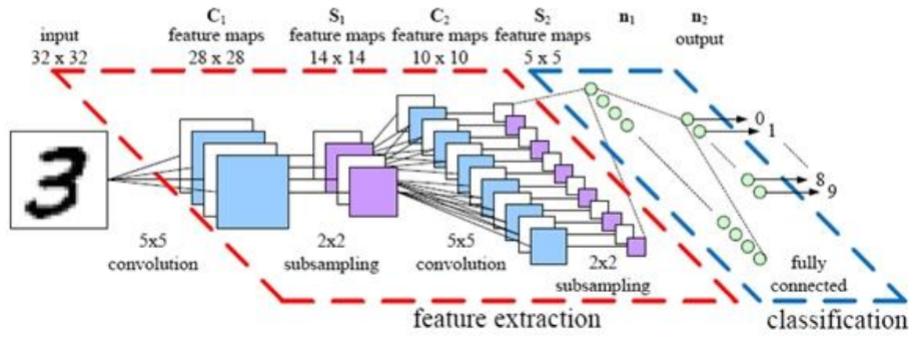
Hình 3.7: Tầng kết nối đầy đủ [10]

Fully-Connected Layer (FC Layer) chính là một mạng NN được gắn vào phần cuối của CNNs. Lớp này tương tự với lớp trong mạng nơ-ron truyền thẳng, các giá trị điểm ảnh liên kết với nhau liên tiếp. Sau khi được xử lý, rút trích, gộp,... từ các lớp trước đó, dữ liệu ảnh không còn quá lớn nên có thể áp dụng mô hình truyền thẳng. Các layer được kết nối đầy đủ lấy các hình ảnh đã lọc ở cấp cao và chuyển thành các phiếu bầu (vote). Thay vì coi đầu vào như một mảng hai chiều, chúng được coi như mảng một chiều và tất cả đều được xử lý giống nhau. Mỗi giá trị hướng đến kết quả cho hình ảnh hiện tại. Tuy nhiên, trong quá trình này các giá trị có mức độ quyết định không giống nhau. Một số giá trị cho biết kết quả của hình ảnh này là tốt hơn nhiều so với những giá trị khác, và một số lại đặc biệt tốt khi cho biết kết quả khác, thể hiện qua trọng số (weight), hoặc là mức độ kết nối...

Như vậy, lớp fully-connected có vai trò như một mô hình phân lớp và thực hiện phân lớp dữ liệu đã được xử lý thông qua các lớp trước đó.

### 3.2.2 Kiến trúc mạng CNN cơ bản

Trong mô hình CNN hai khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Cùng một đối tượng mẫu, nếu đối tượng được chiếu theo các hướng khác nhau (dịch (translation), quay (rotation), co giãn (scaling)) cũng ảnh hưởng đến kết quả thuật toán, độ chính xác sẽ bị ảnh hưởng. Pooling layer



Hình 3.8: Kiến trúc mạng CNN cơ bản [20]

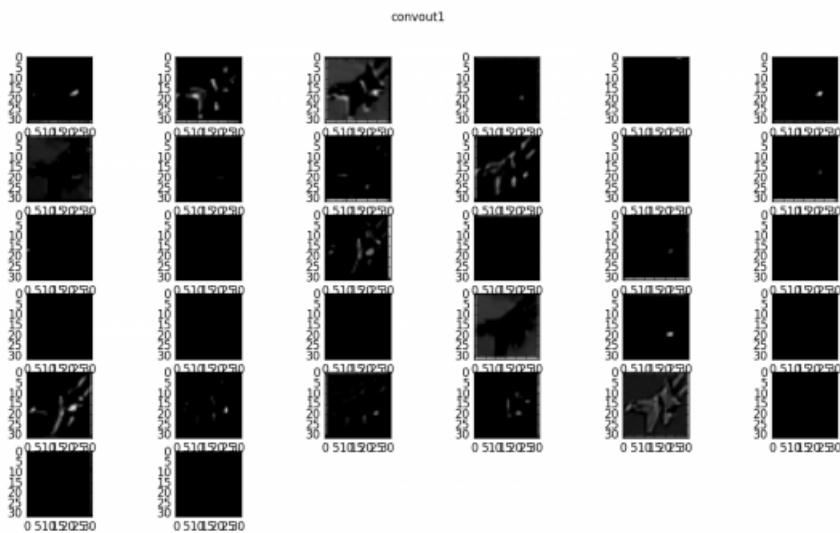
tạo nên tính bất biến với các phép biến đổi của ảnh, vật: phép dịch (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp thể hiện qua việc thông qua việc càng nhiều thông tin bổ sung kết quả dữ liệu cho ta đầu ra càng chính xác, như mạng nơ-ron tích chập được hình thành từ việc ghép bổ xung nhiều feature map.

Mô hình bắt đầu với lớp Convolutional. Lớp RELU thường luôn được cài đặt ngay sau lớp Convolutional hoặc kết hợp cả hai lớp này thành một lớp. Các lớp tiếp theo có thể là Convolutional hay Pooling tùy theo kiến trúc muốn xây dựng. Cuối cùng sẽ là lớp fully-connected để tiến hành phân lớp, đưa ra kết quả đầu ra của mô hình.

Xét một kiến trúc sau đây:

Conv1 (with RELU) – Pooling – Conv2 (with RELU) – Pooling – FC – FC

Hình ảnh cần nhận dạng có kích thước 32x32. Ảnh sẽ được đưa vào lớp Conv1 (Convolutional kết hợp RELU) gồm 32 filter có kích thước 5x5, thực hiện tính tích chập với 32 filter ta sẽ có 32 ảnh kết quả.



Hình 3.9: Nhận diện hình ảnh với CNN [11]

Mỗi ảnh sẽ có kích thước tương ứng là 28x28. Sau đó cho qua lớp Pooling và kết quả trả ra sẽ là 32 ảnh có kích thước 14x14. Tiếp tục cho dữ liệu đi vào lớp Conv2.

Tương tự như Conv1, ảnh sẽ được tính tích chập với filter và trả ra kết quả. Lớp Pooling tiếp theo sẽ tiếp tục giảm kích thước của ảnh xuống còn 5x5. Với kích thước đủ nhỏ, lớp Fully-connected tiếp theo sẽ xử lý và đưa ra kết quả phân lớp hay kết quả nhận dạng.

### Lan truyền ngược (Backpropagation)

Câu hỏi đặt ra: Các feature đến từ đâu? Làm thế nào để tìm trọng số trong các layer được kết nối đầy đủ?

Thuật toán Back – Propagation được sử dụng để điều chỉnh các trọng số kết nối sao cho tổng sai số e nhỏ nhất.

$$E = \sum_{i=1}^n (t(x_i, w) - y(x_i))^2 \quad (3.4)$$

Trong đó:

$t(x_i, w)$ : giá trị của tập mẫu

$(y(x_i))$ : giá trị kết xuất của mạng

Mỗi nơron đều có giá trị vào và ra, mỗi giá trị đều có một trọng số để đánh giá mức độ ảnh hưởng của giá trị vào đó. Thuật toán Back –Propagation sẽ điều chỉnh các trọng số đó để giá trị  $e_j = t_j - y_j$  là nhỏ nhất.

Xác định vị trí của mỗi nơron, nơron nào là của lớp n và nơron nào là của lớp xuất.

Các ký hiệu:

$W_{ij}$ : vector trọng số của nơron j số đầu vào i

$u_j$  : vector giá trị kết xuất của nơron trong lớp j

Giá trị sai số của nơron j tại vòng lặp thứ n

$$e_j(n) = t_j(n) - y_j(n) \quad (3.5)$$

Tổng bình phương sai số của mạng nơron:

$$E(n) = \frac{1}{2} \sum_{j=1}^k e_j^2(n) \quad (3.6)$$

Tổng trọng số input tại nơron j:

$$u_j(n) = \sum_{i=0}^p w_{ij} x_i(n) \quad (3.7)$$

Giá trị kết xuất của neural j:

$$y_j(n) = f_j(u_j(n)) \quad (3.8)$$

Tính toán giá trị đạo hàm sai số cho mỗi neural  $w_{ij}$ :

$$\frac{\partial E(n)}{\partial W_{ij}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} \frac{\partial u_j(n)}{\partial w_{ij}(n)} \quad (3.9)$$

Trong đó

$$\frac{\partial E(n)}{\partial e_j(n)} = \frac{\frac{1}{2} \sum_{j=1}^k e_j^2(n)}{\partial e_j(n)} = \partial e_j(n) \quad (3.10)$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial(t_i(n) - y_i(n))}{\partial y_j(n)} = -1; \quad (3.11)$$

$$\frac{\partial y_j(n)}{\partial u_j(n)} = f'_j(u_j(n)) \quad (3.12)$$

$$\frac{\partial u_j(n)}{\partial w_j(n)} = \frac{\partial(\sum_{i=0}^p W_{ij}x_i(n))}{\partial w_{ij}(n)} = x_i(n) \quad (3.13)$$

$$\Rightarrow \frac{\partial E(n)}{\partial w_{ij}(n)} = -e_j(n).f'(u_j(n))x_i(n) \quad (3.14)$$

Giá trị điều chỉnh trọng số:

Để cập nhật trọng số  $w_{ij}$ : chọn một tốc độ học:  $\eta$ . Tích tích của tốc độ học với gradient, nhân (-1).

$$\Delta w_{ij} = -\eta \frac{\partial E(n)}{\partial w_{ij}(n)} = -\eta e_j(n).f'(u_j(n))x_i(n) \quad (3.15)$$

Trong đó

$$\delta_j = \frac{\partial E(n)}{\partial W_{ij}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} = e_j(n).f'(u_j(n)) \quad (3.16)$$

$$\Delta w_{ij} = -\eta \delta_j(n)x_i(n) \quad (3.17)$$

Công thức điều chỉnh trọng số:

$$w_{ij(n+1)} = w_{ij}(n) + \Delta w_{ij}(n) \quad (3.18)$$

### 3.3 Các kiến trúc mạng CNN nâng cao

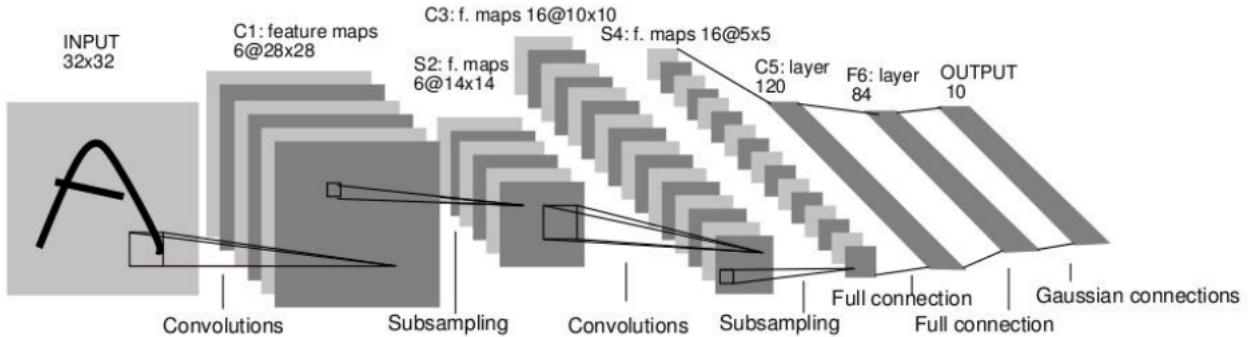
#### 3.3.1 Mạng LeNet-5

Năm 1994 tác phẩm tiên phong của Yann LeCun được đặt tên là LeNet5 sau nhiều lần lặp lại thành công trước đó kể từ năm 1988. Và đây là một trong những mạng thần kinh xoắn nếp đầu tiên, giúp đẩy mạnh lĩnh vực học tập sâu.

Vào thời điểm đó không có GPU để huấn luyện, và thậm chí cả CPU cũng chậm. Vì vậy, có thể lưu các thông số và tính toán là một lợi thế quan trọng. Kiến trúc LeNet là đơn giản và nhỏ gọn (về bộ nhớ), làm cho nó hoàn hảo để giảng dạy những điều cơ bản của CNN - nó thậm chí có thể chạy trên CPU (nếu hệ thống của bạn không có GPU thích hợp).

Các đặc điểm của LeNet5 có thể được tóm tắt như sau [7]:

- Chuỗi sử dụng mạng nơ-ron thần kinh gồm 3 lớp: xoắn chập (convolution), tập hợp gộp (pooling), phi tuyến (non-linearity).
- Sử dụng chập (convolution) để chiết xuất các đặc tính không gian.
- Phi tuyến dùng tanh hoặc sigmoids
- Mạng nơ-ron đa lớp (multi-layer neural network MLP) làm phân loại cuối cùng (final classifier).



Hình 3.10: Mô hình mạng LeNet [7]

- Ma trận kết nối thưa thớt giữa các lớp để tránh chi phí tính toán lớn.
- Hàm gộp lấy giá trị trung bình (Average pooling).

Thực hiện:

Đầu vào là một ảnh 32x32, thực hiện tích chập (convolution) dùng bộ lọc (filter) là 5x5, với 6 feature để có được một khối lớp (layer) kết quả là 28x28x6. Sau đó, chúng ta tiến hành lấy mẫu (subsampling) để có được lượng kích hoạt 14x14x6: giảm một nửa chiều rộng của ảnh, sử dụng ma trận 2x2. Tiếp tục tích chập với 16 ma trận tính năng để có được kích hoạt 10x10x16, dùng các bộ lọc kích thước 5x5. Thực hiện một lần nữa tiến trình lấy mẫu (subsampling), kích thước tổng hợp là 2x2. Lớp xoắn cuối cùng là 120 bộ lọc (filter) có kích thước 5x5, và làm phẳng khôi lượng kích hoạt thành một vector. Cuối cùng, chúng ta thêm một lớp ẩn với 84 nơ-ron và lớp đầu ra với 10 nơ-ron.

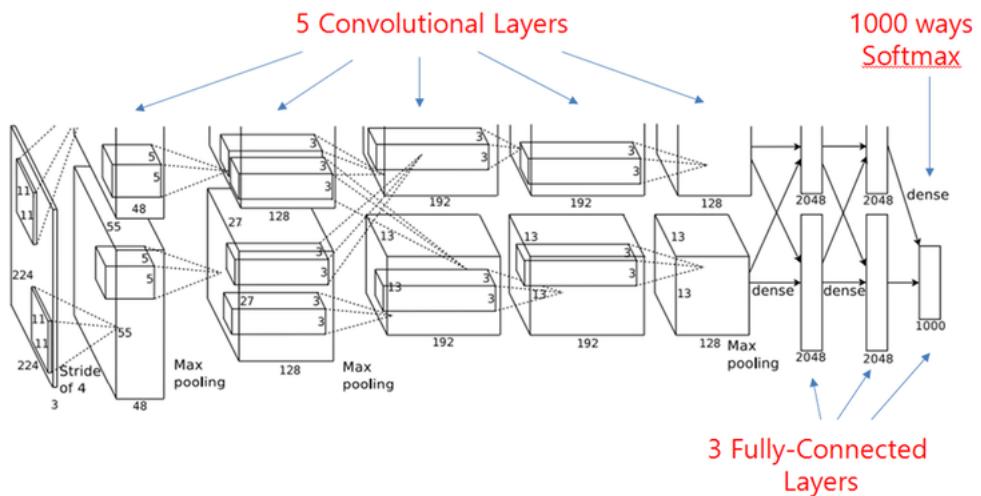
### 3.3.2 Mạng AlexNet

Sơ lược về AlexNet đây là mô hình đánh dấu sự phổ biến của CNN, dành chiến thắng trong cuộc thi ILSVRC 2012 (ImageNet Large Scale Visual Recognition Challenge - một cuộc thi thể vận hội cho thị giác máy tính nơi mà các đội trên thế giới cạnh tranh nhau về mô hình thị giác máy tính theo các nhiệm vụ như phân loại, phát hiện, ...)

Điều đáng nói ở mạng này là chỉ nhận dữ liệu đầu vào là dữ liệu thô và không áp dụng phương pháp trích đặc trưng nào.

Một số thông số cơ bản trong một mạng Alexnet [14]

- Tầng 0: Hình ảnh đầu vào
  - Size: 227 x 227 x 3
- Tầng 1: Mạng tích chập với 96 bộ lọc, kích thước 11x11, bước trượt (stride) S = 4, bộ đệm P = 0
  - Kích thước: 55 x 55 x 96
  - $(227 - 11)/4 + 1 = 55$  size của đầu ra
  - Độ sâu 96 bởi vì 1 tập biểu thị một bộ lọc và có 96 bộ lọc
- Tầng 2: Hàm tổng hợp tối đa (max-pooling) với bộ lọc có kích thước 3x3, bước trượt S = 2



Hình 3.11: Mạng AlexNet [7]

- Kích thước:  $27 \times 27 \times 96$
- $(55 - 3)/2 + 1 = 27$  là kích thước của đầu r
- Độ sâu 96
- Tầng 3: Mạng tích chập với 256 bộ lọc, kích thước 5x5, bước trượt S = 1, bộ đệm P = 2
  - Kích thước:  $27 \times 27 \times 256$
  - Padding = 2 nên kích thước ban đầu được thay đổi l
  - Độ sâu 256 vì có 256 bộ lọc
- Tầng 4: Mạng tích chập với bộ lọc 3x3, bước trượt S = 2
  - Kích thước:  $13 \times 13 \times 256$
  - Kích thước đầu ra:  $(27 - 3)/2 + 1 = 13$
  - Độ sâu 256
- Tầng 5: Mạng tích chập với 384 bộ lọc, kích thước 3x3, bước trượt S = 1, bộ đệm P = 1
  - Kích thước:  $13 \times 13 \times 384$
  - Vì bộ đệm P = 1 nên kích thước được thay đổi
  - Độ sâu 384 vì có 384 bộ lọc
- Tầng 6: Mạng tích chập với 384 bộ lọc, kích thước 3x3, bước trượt S = 1, bộ đệm P = 1
  - Kích thước:  $13 \times 13 \times 384$
  - Vì bộ đệm P = 1 nên kích thước được thay đổi
  - Độ sâu 384 vì có 384 bộ lọc
- Tầng 7: Mạng tích chập với 256 bộ lọc, kích thước 3x3, bước trượt S = 1, bộ đệm P = 1
  - Kích thước:  $13 \times 13 \times 384$
  - Vì bộ đệm P = 1 nên kích thước được thay đổi
  - Độ sâu 256 vì có 256 bộ lọc
- Tầng 8: Hàm tổng tối đa kích thước 3x3, bước trượt S = 2

- Kích thước:  $6 \times 6 \times 256$
- Kích thước đầu ra: 6
- Độ sâu 256
- Tầng 9: Mạng tích chập đầy đủ với 4096 neuron
  - Mỗi ảnh  $6 \times 6 \times 256 = 9216$  pixels được đưa vào mỗi neuron và trọng số được xác định bằng mạng truyền ngược (back - propagation)
- Tầng 10: Mạng tích chập đầy đủ với 4096 neuron
  - Giống tầng 9
- Tầng 11: Mạng tích chập đầy đủ với 1000 neuron
  - Đây là tầng cuối cùng và có 1000 neurons vì dữ liệu IAMGENET có 1000 lớp cần được phân loại

Là mạng đầu tiên làm việc tốt với tập dữ liệu IAMGENET.

### 3.3.3 Mạng GoogLeNet

Tổng quan về GooLeNet [17]

- Gồm 22 tầng
- Ảnh hưởng bởi Inception module
- Không có tầng kết nối đầy đủ (fully - connected layer)
- Chỉ có 5 triệu thông số ít hơn so với AlexNet 12 lần

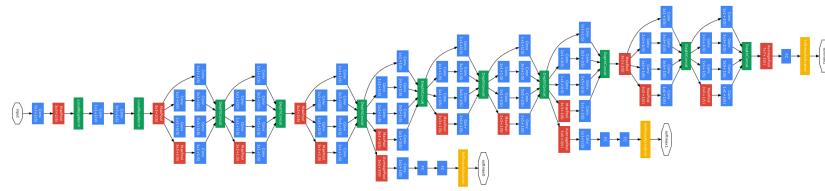
Đối với các mạng tích chập khác việc lựa chọn  $3 \times 3$  hoặc  $5 \times 5$  để được tối ưu là một vấn đề nan giải. Vậy thì sao chúng ta không thể sử dụng tất cả chúng, câu trả lời được hiện thực trong module Inception, ta thực hiện việc này bằng cách thực hiện từng tích chập song song và nối các kết quả lại trước khi tới lớp tiếp theo.

Đến lớp tiếp theo cũng là một module Inception, sau đó mỗi bản đồ đặc trưng (feature maps) sẽ được truyền qua hỗn hợp các xáo trộn của lớp hiện tại. Ý tưởng là không cần phải biết trước thời gian nếu nó được tốt hơn để làm, ví dụ  $3 \times 3$  sau đó là  $5 \times 5$ . Thay vào đó, chỉ cần làm tất cả các tích chập và để mô hình tự chọn được cái tốt nhất. Ngoài ra kiến trúc này cho phép mô hình phục hồi cá tính năng địa phương thông qua các vòng quay nhỏ hơn và các tính năng trừu tượng cao.

Inception module: thiết kế một mạng địa phương với topology và rồi đặt chúng chồng lên nhau. Tất cả các mạng tích chập được bao trong modul Inception, sử dụng hàm tinh chỉnh đơn vị tuyến tính để kích hoạt (hàm ReLU). Tham khảo hình dưới.

Mạng gồm 22 tầng khi chỉ đếm với những tầng có thông số (nếu đếm tất cả ta có 27 tầng). Tổng số lớp (khối xây dựng độc lập) được sử dụng để xây dựng là khoảng 100 lớp. Số chính xác dựa vào cách các lớp được đếm bởi cơ sở hạ tầng học máy. Việc sử dụng tổng hợp trước khi phân loại, mặc dù việc có thêm một lớp tuyến tính. Lớp tuyến tính cho phép chúng ta dễ dàng điều chỉnh mạng với các bộ nhãn khác nhau, tuy nhiên nó được sử dụng chủ yếu để thuận tiện và mong đợi nó có một hiệu ứng lớn. Sự di chuyển từ các lớp kết nối hoàn chỉnh tới tổng hợp đã nâng cao độ chính xác lên khoảng 1%, tuy nhiên việc sử dụng dropout vẫn là điều thiết yếu ngay cả sau khi loại bỏ các lớp kết nối hoàn toàn.

Với độ sâu tương đối lớn của mạng, khả năng truyền của gradient trở lại tất cả các lớp một cách hiệu quả là mối quan tâm. Hiệu năng mạnh mẽ của các mạng lưới nông

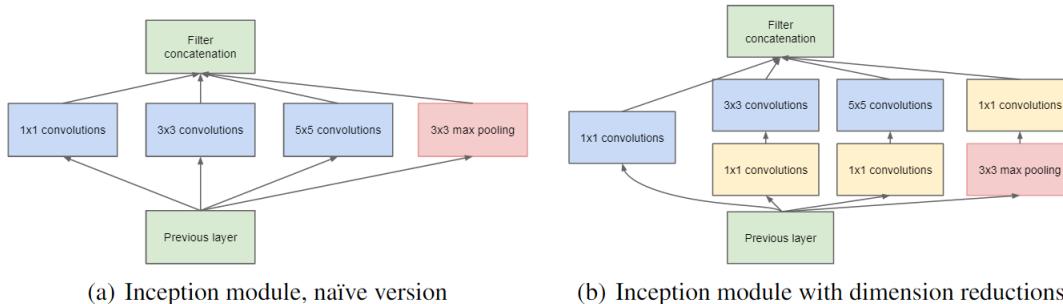


Hình 3.12: Mô hình mạng GoogLeNet [17]

trên công việc này cho thấy các tính năng được sản xuất bởi các lớp ở giữa mạng nên rất phân biệt. Bằng cách thêm các phân lớp phụ trợ kết nối với các lớp trung gian này, sự kì thị trong các giai đoạn dưới của phân loại được dự kiến. Điều này được cho là để chống lại các vấn đề về gradient biến mất trong khi cung cấp sự ổn định. Những phân loại này có dạng của các mạng xoắn nhỏ hơn đặt ra trên đầu ra của các mô đun inception. Trong quá trình huấn luyện, mất mát sẽ được cộng vào tổng số thiệt hại của mạng với trọng lượng chiết khấu (tổn thất của các tầng được trọng số = 0.3. Tại thời điểm suy luận, các phụ trợ mạng sẽ bị loại bỏ. Các thí nghiệm kiểm soát sau này có cho thấy hiệu quả của các lối phụ trợ là tương đối nhỏ khoảng 0.5% và rằng nó chỉ yêu cầu một trong số họ để được hiệu quả tương tự.

Cấu trúc chính xác của mạng phụ ở bên cạnh, bao gồm phân loại phụ trợ như sau:

- Một lớp tổng hợp trung bình với kích thước bộ lọc  $5 \times 5$  và bước trước  $S = 3$ , kết quả đầu ra  $4 \times 4 \times 512$  cho hình a, và  $4 \times 4 \times 528$  cho hình b.



Hình 3.13: Module Inception [18]

- Một tích chập  $1 \times 1$  với 128 bộ lọc để giảm kích thước và tinh chỉnh các đơn vị tuyến tính
- Một lớp kết nối hoàn chỉnh với 1024 đơn vị và tinh chỉnh các đơn vị tuyến tính.
- Một tầng loại bỏ (dropout layer) 70% tỷ lệ đầu ra.
- Một lớp tuyến tính với tổn thất softmax như là phân loại (dự đoán các lớp 1000 tương tự như phân loại chính, nhưng loại bỏ tại thời gian suy luận)

Đây là một trong những kiến trúc CNN đầu tiên thực sự bị lạc khỏi cách tiếp cận tổng quát của việc xếp chồng lên nhau và xếp các lớp lên nhau theo cấu trúc tuần tự. Các tác giả của nhấn mạnh rằng mô hình mới này chú trọng đến việc sử dụng bộ nhớ và điện năng (Lưu ý quan trọng đôi khi chúng ta cũng quên rằng: xếp tất cả các lớp này và thêm một số lượng lớn các bộ lọc có chi phí tính toán và bộ nhớ, cũng như tăng cơ hội overfitting).

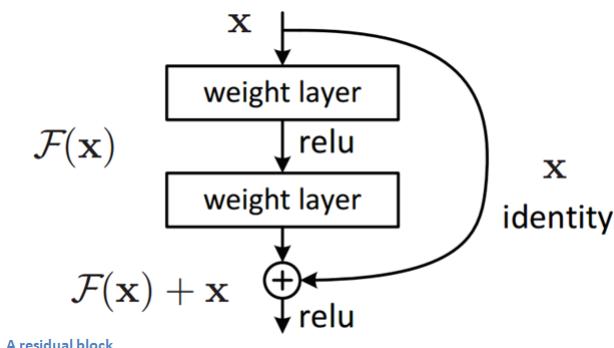
### 3.3.4 Mạng ResNet [18]

Như ta đã biết độ sâu có tầm quan trọng đặc biệt trong kiến trúc mạng neuron, nhưng các mạng càng sâu thì càng khó đào tạo.

Vấn đề với độ sâu của mạng tăng lên, độ chính xác bị bão hòa và sau đó giảm nhanh chóng. Sự đi xuống không phải do overfitting, và thêm càng nhiều lớp vào học sâu dẫn đến lỗi đào tạo cao hơn.

Xem xét vấn đề với một mạng nông hơn. Có một giải pháp cho mô hình sâu hơn bằng cách xây dựng các lớp được sao chép từ mô hình nông hơn, và các lớp thêm vào được ánh xạ nhận dạng. Sự tồn tại của giải pháp xây dựng này chỉ ra rằng một mô hình sâu hơn sẽ không tạo ra lỗi đào tạo cao hơn so với mô hình nông hơn. Một lý do khác cho lý do tại sao ResNet lại này có thể hiệu quả là trong suốt quá trình quay ngược trở lại của backpropagation, gradient sẽ dễ dàng chảy qua đồ thị bởi vì chúng ta có các phép toán cộng, phân phối gradient.

Ý tưởng trong mạng này là sau một khối dư bạn có đầu vào của mình là  $x$  đi qua chuỗi conv-relu-conv. Điều này sẽ cho ta hàm số  $F(x)$ . Kết quả sau đó được thêm vào đầu vào ban đầu  $x$ . Gọi đó là  $H(x) = F(x) + x$ . Trong CNN truyền thống,  $H(x)$  bằng  $F(x)$ , bây giờ ta không chỉ đơn thuần tính toán sự chuyển đổi thẳng từ  $x$  sang  $F(x)$ . Về cơ bản, mô đun mini bên dưới hình ảnh thay đổi một chút đầu vào để có một phiên bản mới của CNN.



Hình 3.14: Một block ResNet [18]

Cho đến năm 2016 thì đây là mạng CNN tốt nhất, giải quyết một phần vấn đề học càng sâu càng tốt.

# Chương 4

## Hướng tiếp cận và mô hình đề xuất

### 4.1 Hướng tiếp cận

Trong bài toán phân loại hình ảnh hiện hai có hướng tiếp cận chính:

- Phương pháp thứ nhất: nhận dạng hình ảnh dựa trên tri thức của chuyên gia. Đối với hướng tiếp cận này người ta xây dựng hệ thống phân loại dựa trên các công thức toán học, rút trích đặc trưng của hình ảnh để đưa ra phân loại. Đây là phương pháp truyền thống trước kia, yêu cầu kinh nghiệm của chuyên gia cao, độ chính xác thấp, không thể áp dụng cho mô hình thay đổi, hay với tập dữ liệu lớn. Tuy nhiên cũng có một số ưu điểm như các công thức tính toán có thể được tính toán dễ dàng, chi phí thấp, không yêu cầu phần cứng cao.
- Phương pháp thứ hai: nhận dạng hình ảnh dựa trên dữ liệu. Phương pháp này đòi hỏi lượng dữ liệu ban đầu lớn cùng với giải thuật học máy phù hợp để dự đoán, phân loại. Tuy nhiên đối với cách tiếp cận này có thể đảm bảo hệ thống tự động cập nhật các quy tắc nhận dạng mà không cần xây dựng lại công thức tính toán mới. Với sự phát triển công nghệ như hiện nay thì việc thu thập dữ liệu đã không còn là vấn đề nên cách tiếp cận này hiện trở thành hướng tiếp cận chính trong thị giác máy tính. Chính vì thế đây cũng là hướng mà nhóm sẽ đi theo.

### 4.2 Công cụ sử dụng

Cùng với sự phát triển của xử lý ảnh các công cụ phục vụ cũng dần được phát triển trong đó phải kể đến một số framework nổi tiếng như caffe của UC Berkeley, Torch (NYU/Facebook), Theano (U Montreal), TensorFlow (Google),...

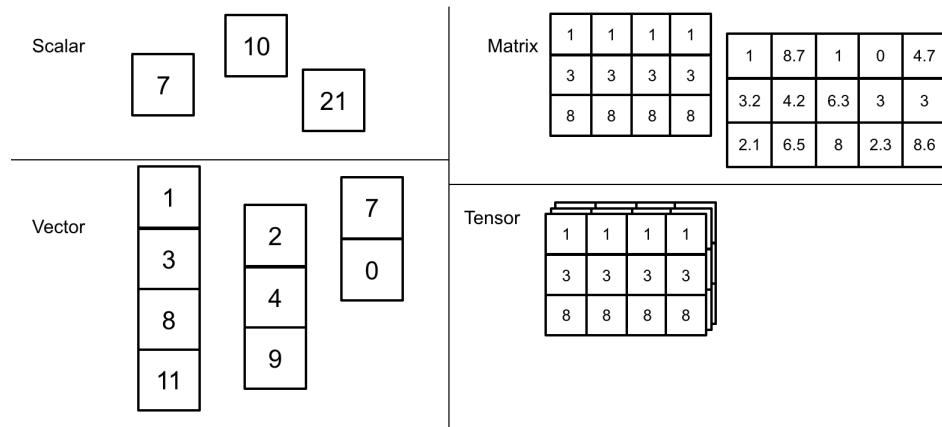
Theo định hướng tôi sẽ sử dụng TensorFlow để hiện thực luận văn. Đây là một công cụ mã nguồn mở của Google được hỗ trợ rất mạnh - dễ sử dụng. Hiện tại, cộng đồng hiện sử dụng nó càng được mở rộng. Ngoài ra còn sử dụng: OpenCV 3.4.4, Keras 2.2.4,...

Một số khái niệm cơ bản cần biết trong Tensorflow:

- Scalar : số vô hướng hay các số trong hệ thập phân (5, 10, 7.2,...)
- Vector : là một tập các số vô hướng. Số lượng số vô hướng là số chiều của vector
- Matrix : gồm nhiều vector có cùng số chiều.
- Tensor : phát triển lên từ các khái niệm trên ta có định nghĩa tensor, mỗi tensor n chiều là 1 tập các tensor n-1 chiều có cùng kích thước. Chẳng hạn Scalar là tensor

0D (0 Dimension – 0 chiều), Vector là tensor 1D, Matrix là tensor 2D... Tensor là cấu trúc dữ liệu được sử dụng trong toàn TensorFlow, đại diện cho tất cả các loại dữ liệu. Hay nói cách khác là tất cả các loại dữ liệu đều là tensor. Việc trao đổi dữ liệu trong quá trình xử lý chỉ thông qua tensor. Hiểu đơn giản thì tensor là mảng n chiều hay list cộng thêm 1 vài thứ thú vị khác.

- Variable lưu trạng thái (state) sau khi tính toán đồ thị.



Hình 4.1: Một số khái niệm trong TensorFlow [19]

# **Chương 5**

## **Thực nghiệm và đánh giá mô hình**

### **5.1 Chuẩn bị dữ liệu, nghiên cứu, phân tích đề tài**

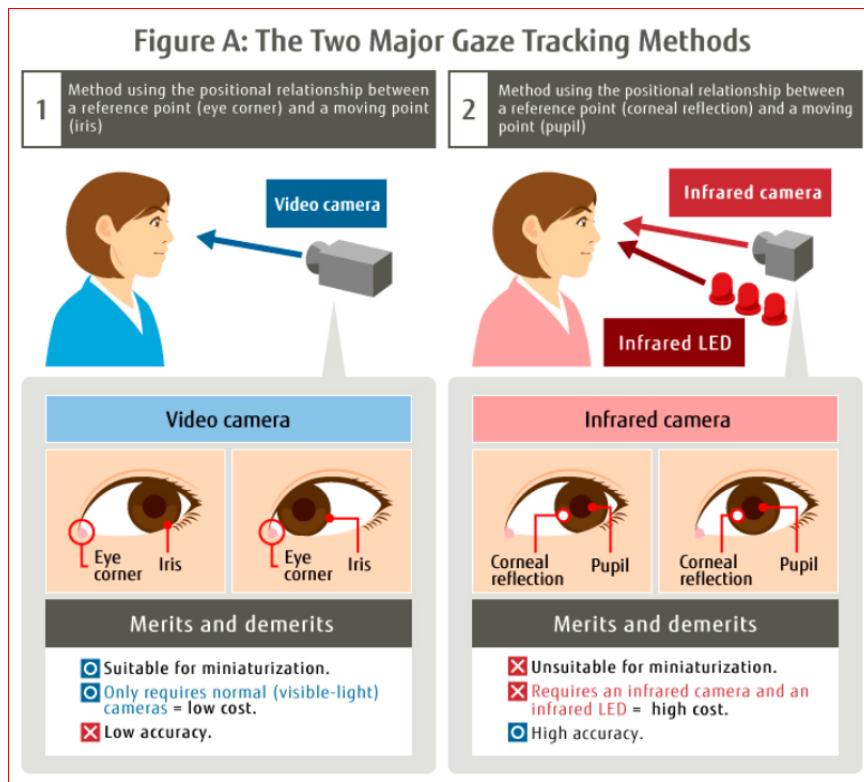
#### **5.1.1 Chuẩn bị kiến thức nền tảng**

- Nghiên cứu các công trình liên quan tới đề tài mà mình đang thực hiện.
- Lựa chọn hướng tiếp cận.
- Hiểu được căn bản và cách hoạt động của mạng CNN.
- Hiểu được kiến trúc nâng cao của mạng CNN cũng như các vấn đề và cách giải quyết của nó.
- Tìm hiểu công cụ sử dụng TensorFlow

#### **5.1.2 Nghiên cứu kỹ thuật dõi chuyển động mắt hiện nay**

Có hai phương pháp để theo dõi chuyển động mắt hiện nay: Camera Video thông thường và Camera hồng ngoại có đèn LED hồng ngoại. Camera hồng ngoại được phát triển vì phản chiếu để định vị mắt chính xác. Nhưng với sự phát triển của các công nghệ mới, điều này có thể được thay thế bằng một số cách như nhận dạng tính năng bằng cách sử dụng mạng nơ ron thần kinh tích chập (CNNs).

Camera hồng ngoại- có đèn hồng ngoại: (infrared LED) là camera thông thường được trang bị thêm các đèn hồng ngoại có cảm biến ánh sáng tên tiếng anh là (light sensor). Cảm biến này sẽ tự nhảy nếu điều kiện môi trường thiếu sáng, khi bật cảm biến hồng ngoại sẽ bật lên giúp camera quay được những cảnh trong bóng tối giúp hình ảnh chính xác và ổn định nhưng tốn nhiều chi phí hơn camera thông thường.

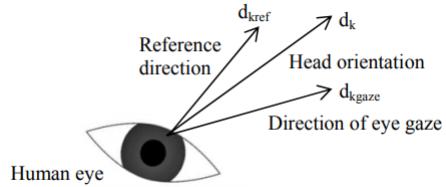


Hình 5.1: Camera Video thông thường và Camera hồng ngoại có đèn LED hồng ngoại. [23]

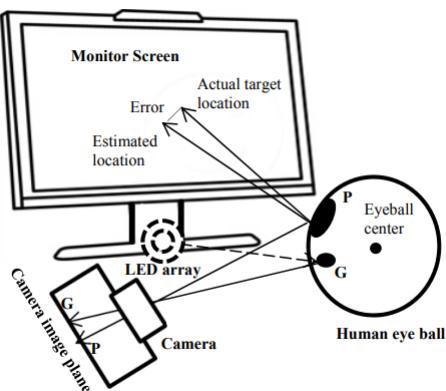
TABLE I  
CLASSIFICATION OF EYE MOVEMENTS

Eye movement type	Movement rate	Latency/duration of occurrence	Functionality/ Significance	Applications in Human Computer interaction
Fixation	< 15 – 100 deg/ms	180- 275 ms	Acquiring information, Cognitive processing, attention	Browsing information, reading, scene perception
Saccade	100- 700 deg/sec	Latency<200 ms, duration: 20–200 ms	Moving between targets	Visual search
Smooth pursuit	< 100 deg/sec based on target speed	100 ms	Following moving targets	Gaze based drawing, steering
Scanpath	--	--	Scanning, direct search	Assessing user behavior, user interface and layout quality
Gaze duration	--	--	Cognitive processing, conveying intent	Item selection, text/number entry
Blink	12 -15 per min	300 ms	Indicates behavioral states, stress	Eye liveliness detection, activate command/control
Pupil size change	4-7 mm/sec	140 ms	Cognitive effort, representing micro-emotions	Assessing cognitive workload, user fatigue, command/control

Hình 5.2: Phân loại sự vận động của mắt [5]



Hình 5.3: Mối quan hệ giữa hướng nhìn và tư thế đầu [5]



Hình 5.4: Sơ đồ của một hệ thống theo dõi hướng nhìn. P là đồng tử của bóng mắt người và G là vị trí phản chiếu được hình thành trên giác mạc, được chụp trên mặt phẳng camera. Hình ảnh này cho thấy lỗi trong ước tính ánh nhìn cũng như độ lệch giữa các vị trí nhìn thực tế và ước tính. [5]

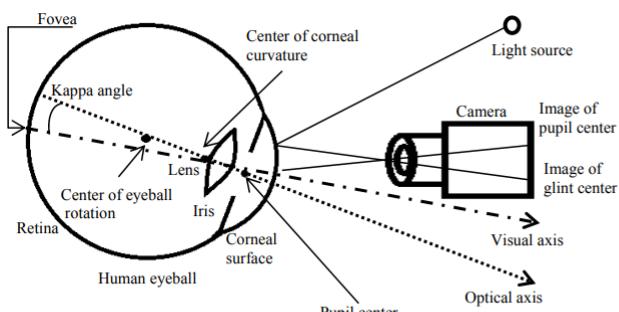


Fig. 4a

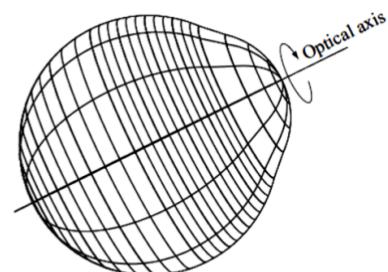
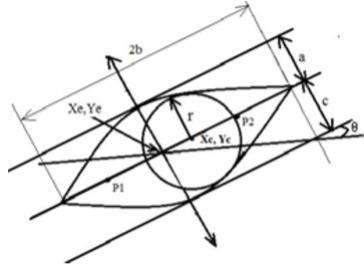


Fig. 4b

Hình 5.5: Mô hình 3D: a. Mô hình bóng mắt người, thông số mắt và các yếu tố thiết lập được sử dụng trong theo dõi mắt 3D. Trục quang được hiển thị như là đường thẳng nối giữa tâm của giác mạc với trung tâm đồng tử mắt. Trục thị giác đi qua fovea và tâm của đường cong giác mạc. Góc Kappa là độ lệch góc giữa trục quang và thị giác. b. Một mô hình phi cầu của giác mạc, như một bề mặt của chuyển đổi về trục quang của mắt. [5]



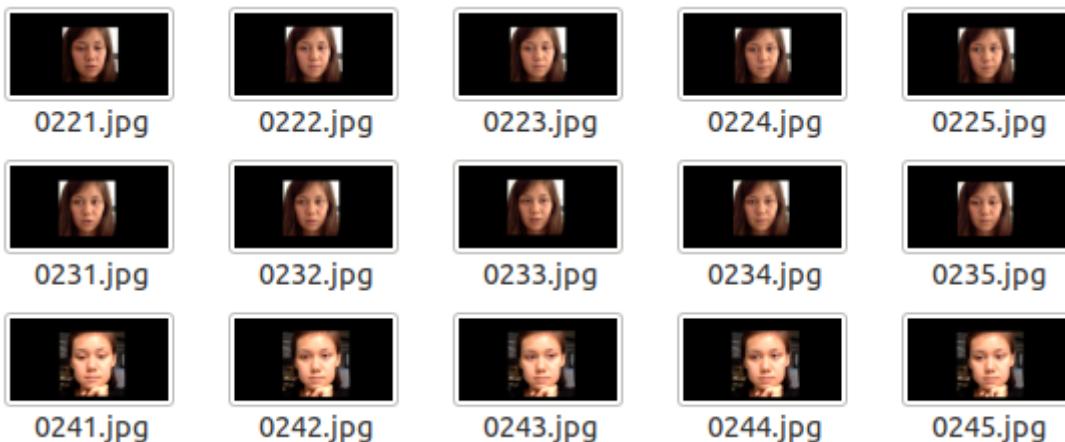
Hình 5.6: Phương pháp dựa trên hình dạng: Mẫu của một vùng mắt:  $X_c$ ,  $Y_c$  đại diện cho trung tâm của đồng tử và mắt tương ứng  $P_1$  và  $P_2$  là tiêu điểm của hai phần parabol và  $a$ ,  $b$ ,  $c$  và là tham số,  $r$  là bán kính của đồng tử [5]

### 5.1.3 Chuẩn bị và xử lý dữ liệu

Thu thập xử lý dữ liệu, tập dữ liệu sử dụng trong đề tài:

- Chuẩn bị tập dữ liệu MPIIFaceGaze:[15]

Đối với bài toán phát hiện hướng nhìn, ban đầu nhận hình ảnh đầu vào là ảnh chỉ có mặt người mà không có nhiễu (không có các vật thể khác phía sau khuôn mặt). Tiếp theo đó xử lý hình ảnh để đánh nhãn ảnh hướng nhìn trên khuôn mặt, đưa vào mạng tiên hành training cho để đưa ra đầu ra cho hình ảnh bất kỳ.



Hình 5.7: Tập dữ liệu ảnh MPIIFaceGaze.

- Chuẩn bị tập dữ liệu GazeCaptureEyeTracking:[21]

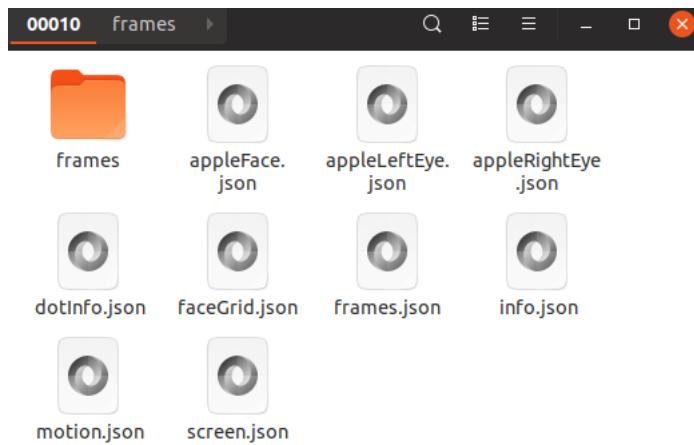
Bộ dữ liệu bao gồm dữ liệu cho các đối tượng duy nhất. Mỗi thư mục được đánh số đại diện cho một đối tượng. Các số được gán liên tục.

Bên trong mỗi thư mục là một tập hợp các hình ảnh được đánh số liên tục (trong thư mục con của thư mục frames) và các tệp JSON chứa thông tin cho các phần dữ liệu ảnh trong cùng thư mục. Nhiều biến trong các tệp JSON là các mảng, trong đó mỗi phần tử được liên kết với khung (frames) được đánh số giống như chỉ mục.

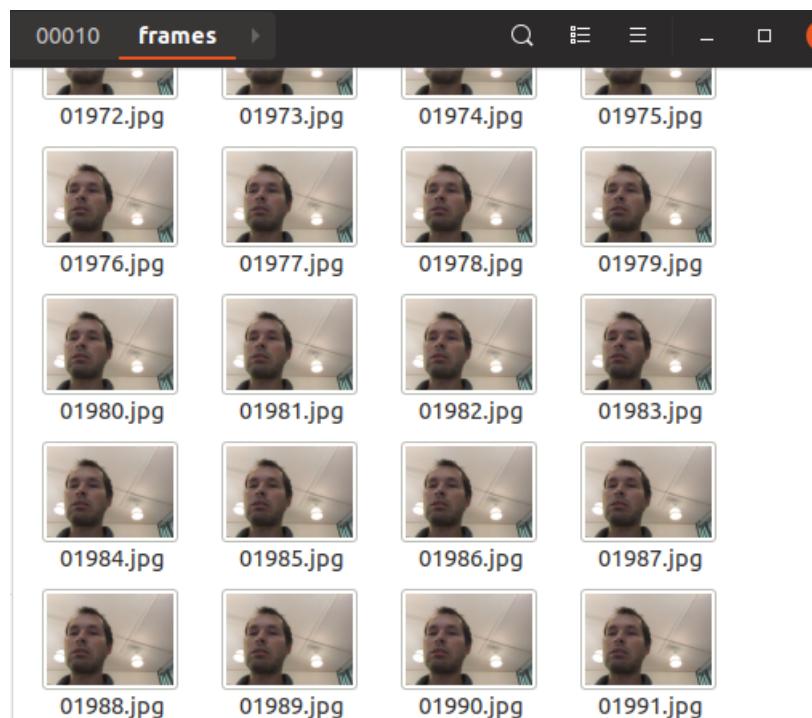
Thông tin các file Json:

+ **appleFace.json, appleLeftEye.json, appleRightEye.json:**

Các tệp này mô tả các hộp giới hạn xung quanh khuôn mặt và mắt được phát hiện. X, Y: Vị trí của góc trên cùng bên trái của hộp giới hạn (tính bằng pixel).



Hình 5.8: Tập dữ liệu ảnh GazeCaptureEyeTracking: thư mục frames chứa các file hình ảnh, và các file Json.



Hình 5.9: Tập dữ liệu ảnh GazeCaptureEyeTracking: các file hình ảnh trong thư mục frames.

W, H: Chiều rộng và chiều cao của khung giới hạn (tính bằng pixel).

#### + dotInfo.json:

DotNum: Số thứ tự của dấu chấm (bắt đầu từ 0) được hiển thị trong khung đó.

XPts, YPts: Vị trí trung tâm của dấu chấm (tính theo điểm; xem tài liệu screen.json bên dưới để biết thêm thông tin về đơn vị này) từ góc trên cùng bên trái của màn hình.

XCam, YCam: Vị trí của tâm điểm trong không gian dự đoán của chúng tôi. Vị trí được đo bằng centimet và liên quan đến trung tâm camera, giả sử camera vẫn ở

---

vị trí cố định trong không gian trên tất cả các hướng của thiết bị. Tức là, các giá trị YCam sẽ âm đối với các khung chế độ dọc (Định hướng == 1) vì màn hình nằm dưới camera, nhưng các giá trị sẽ dương ở chế độ chân dung lộn ngược (Định hướng == 2) vì màn hình nằm phía trên camera.

Time: Thời gian (tính bằng giây) kể từ khi dấu chấm hiển thị xuất hiện lần đầu tiên trên màn hình.

**+ faceGrid.json:**

Các giá trị này mô tả các tính năng đầu vào "lưới mặt", được tạo ra từ các phát hiện khuôn mặt. X, Y: Vị trí của góc trên cùng bên trái của hộp mặt (được lập chỉ mục 1, trong lưới 25 x 25).

W, H: Chiều rộng và chiều cao của hộp mặt.

IsValid: Dữ liệu có hợp lệ (1) hay không (0).

**+ frames.json:**

Tên tệp của các khung (frames) trong thư mục khung. Thông tin này cũng có thể được tạo từ số thứ tự đếm từ 0 đến TotalFrames - 1 (xem info.json).

**+ info.json:**

TotalFrames: Tổng số khung cho chủ đề này.

NumFaceDetections: Số lượng khung hình trong đó một khuôn mặt được phát hiện.

NumEyeDetections: Số lượng khung hình trong đó mắt được phát hiện.

Bộ dữ liệu: "đào tạo", "val" hoặc "kiểm tra" ("train," "val," or "test.")

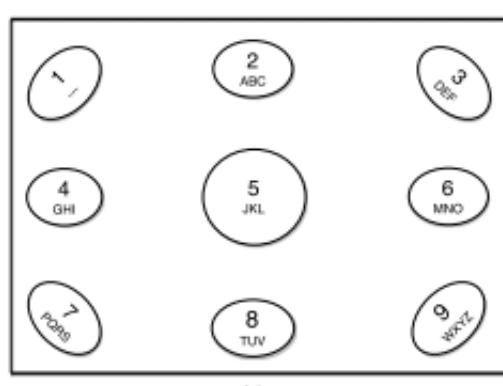
DeviceName: Tên của thiết bị được sử dụng trong bản ghi.

**+ motion.json, screen.json:**

Thông tin về luồng dữ liệu chuyển động và định hướng của giao diện

## 5.2 Tiến hành thực hiện

Từ những bức ảnh chụp khuôn mặt người thông qua các mô hình có thể phát hiện ra hướng của mắt. Nhóm chia hướng nhìn của mắt làm 9 hướng và nhắm mắt.

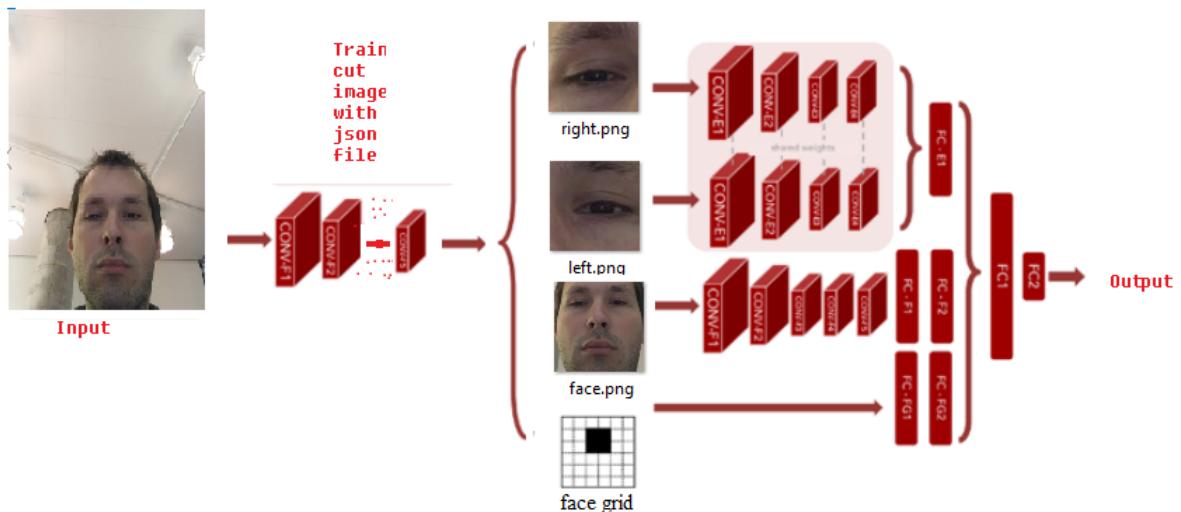


Hình 5.10: Các hướng nhìn [1]

### 5.2.1 Tập dữ liệu GazeCaptureEyeTracking

Phương pháp sử dụng học sâu với mô hình CNNs: Từ tập hình ảnh với các tệp dữ liệu json , tiến hành training cho để đưa ra kết quả. Mô hình đã thực hiện với cấu hình sau:

Python 3.6.7  
OpenCV 3.4.4  
Keras 2.2.4  
TensorFlow 1.11.0



Hình 5.11: Mô hình CNNs sử dụng training tập dữ liệu GazeCaptureEyeTracking.

```

# eye model
def get_eye_model(img_ch, img_cols, img_rows):
    eye_img_input = Input(shape=(img_ch, img_cols, img_rows))

    h = Conv2D(96, (11, 11), activation=activation)(eye_img_input)
    h = MaxPool2D(pool_size=(2, 2))(h)
    h = Conv2D(256, (5, 5), activation=activation)(h)
    h = MaxPool2D(pool_size=(2, 2))(h)
    h = Conv2D(384, (3, 3), activation=activation)(h)
    h = MaxPool2D(pool_size=(2, 2))(h)
    out = Conv2D(64, (1, 1), activation=activation)(h)

    model = Model(inputs=eye_img_input, outputs=out)
    print("model get_eye_model: ", model)
    return model


# final model
def get_eye_tracker_model(img_ch, img_cols, img_rows):
    # get partial models
    print("get_eye_model")
    eye_net = get_eye_model(img_ch, img_cols, img_rows)
    face_net_part = get_face_model(img_ch, img_cols, img_rows)

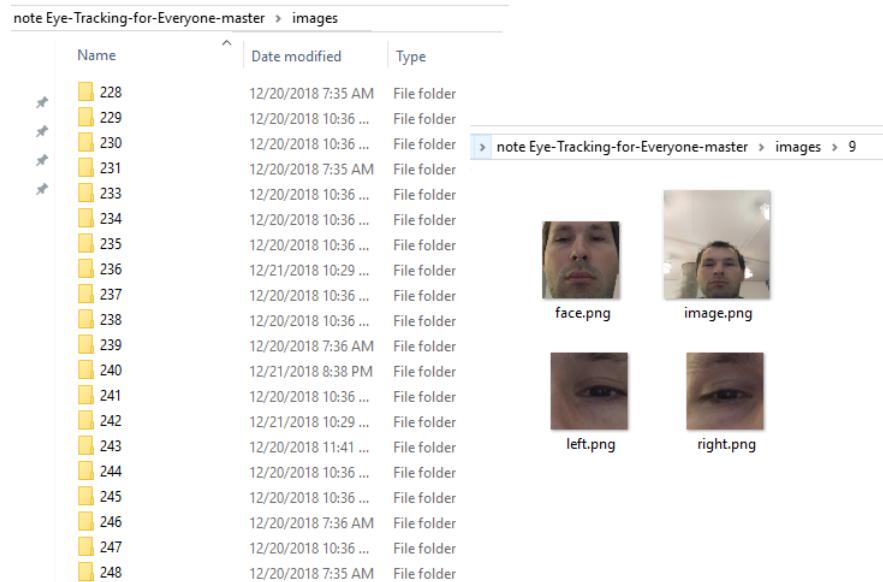
    # right eye model
    right_eye_input = Input(shape=(img_ch, img_cols, img_rows))
    right_eye_net = eye_net(right_eye_input)

    # left eye model
    left_eye_input = Input(shape=(img_ch, img_cols, img_rows))
    left_eye_net = eye_net(left_eye_input)

    # face model
    face_input = Input(shape=(img_ch, img_cols, img_rows))
    face_net = face_net_part(face_input)

```

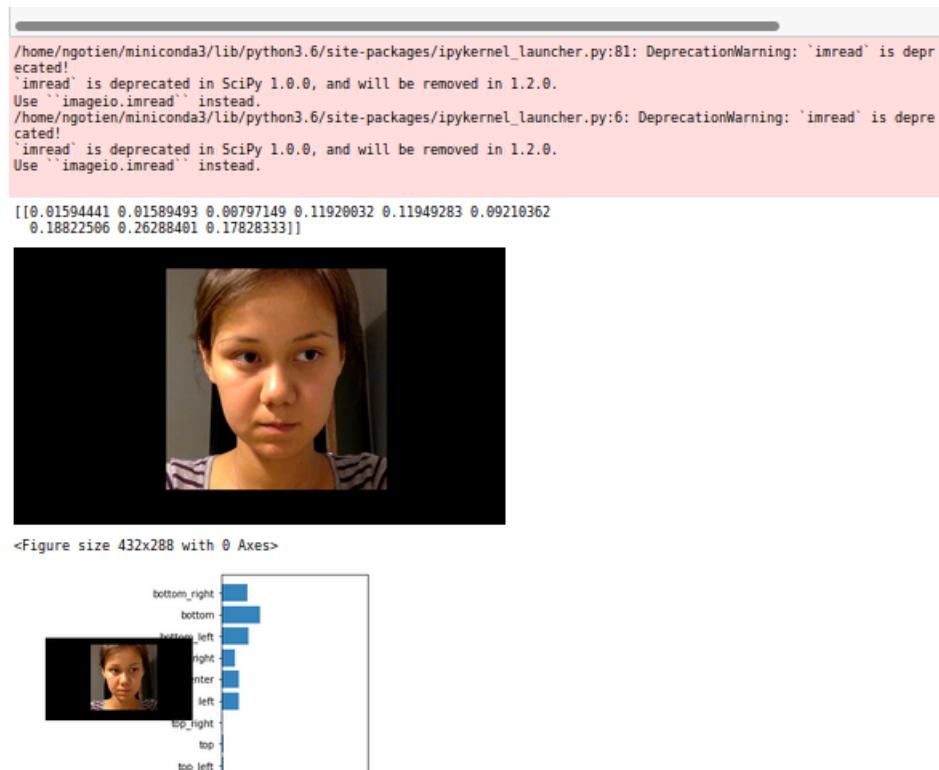
Hình 5.12: Mô hình (model) thực hiện phát hiện ánh nhìn



Hình 5.13: Kết quả nhận được tập ảnh chứa các hình ảnh cắt mắt phải, trái

## 5.2.2 Tập dữ liệu MPIIFaceGaze

Sử dụng kết hợp mô hình đơn giản linear model để train: Xử lý hình ảnh đánh nhãn ảnh khuôn mặt, tiến hành training cho để đưa ra kết quả. Một số kết quả nhận được:



Hình 5.14: Kết quả thu được (dữ liệu ảnh MPIIFaceGaze).



Hình 5.15: Tập dữ liệu ảnh MPIIFaceGaze.

---

### **5.3 Đánh giá kết quả, mô hình**

Trong quá trình nghiên cứu, tôi đã:

- Hiểu được căn bản và cách hoạt động của mạng CNN.
- Hiểu được kiến trúc nâng cao của mạng CNN cũng như các vấn đề và cách giải quyết
- Tổng hợp, đánh giá ưu và nhược điểm của cách phương pháp, công nghệ đã và đang được nghiên cứu, sử dụng.
- Tiếp cận vấn đề theo nhiều hướng khác nhau, các nghiên cứu kỹ thuật đổi chuyển động mắt hiện nay
- Thực hiện một số phương pháp sử dụng học sâu (CNN) để phát hiện hướng nhìn của con người qua hình ảnh nhưng độ chính xác chưa cao do đòi hỏi yêu cầu kỹ thuật cao (Kỹ năng phân tích hình dạng mắt, Mô hình 3D, Camera,...)
- Cuối cùng, tôi đề xuất hướng phát triển tiếp theo của đề tài trong tương lai: Phân tích hình ảnh mắt, cấu trúc mắt đồng tử, điều chỉnh hướng camera,... tăng độ chính xác của kết quả, phát triển thành ứng dụng trên thiết bị máy tính, điện thoại có thể giúp người khuyết tật điều khiển các thiết bị chỉ bằng ánh mắt mà không cần đến đôi tay, theo dõi người lái xe có thể phát hiện kịp thời người này ngủ gật hay không chú ý khi lái xe,...

# **Chương 6**

## **Thảo luận**

Sự phát triển nghiên cứu về công nghệ theo dõi mắt là một xu hướng trong tương lai gần.

Đôi mắt và các chuyển động của chúng truyền đạt sự chú ý của chủ thể và đóng vai trò trong việc truyền đạt thông tin và cảm xúc. Do đó, chúng rất quan trọng đối với một loạt các ứng dụng, bao gồm sự tương tác giữa con người và máy tính dựa trên sự quan sát, giám sát hành vi trực quan... Một số ứng dụng trong ngành thị giác máy tính (computer vision) liên quan đến mắt bao gồm việc ước lượng mắt: xác định nơi ai đó đang tìm kiếm, nhận diện người qua câu tạo mắt (móng mắt, mí mắt,...).

Khoa học về mắt là một tiềm năng lớn, có nhiều ứng dụng trong tương lai. Theo dõi bằng mắt sẽ trở thành một công cụ hỗ trợ quan trọng trên nhiều lĩnh vực. Nó giải phóng bàn tay con người trong một số phạm vi, làm cho công việc hiệu quả hơn.

# Tài liệu tham khảo

- [1] Chi Zhang, Rui Yao, Jinpeng Cai *Efficient Eye Typing with 9 direction Gaze Estimation.*
- [2] Xucong Zhang, Yusuke Sugano, Mario Fritz, Andreas Bulling *Appearance-Based Gaze Estimation in the Wild.*
- [3] University of Cambridge, United Kingdom- Rendering of Eyes for Eye-Shape Registration and Gaze Estimation eww23 iccv2015  
[https://www.cv-foundation.org/openaccess/content\\_iccv\\_2015/papers/Wood\\_Rendering\\_of\\_Eyes\\_ICCV\\_2015\\_paper.pdf](https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Wood_Rendering_of_Eyes_ICCV_2015_paper.pdf)
- [4] University of Cambridge and Carnegie Mellon University and Max Planck Institute for Informatics, Learning an appearance-based gaze estimator from one million synthesised images  
<https://www.d2.mpi-inf.mpg.de/content/learning-appearance-based-gaze-estimator>
- [5] Anuradha Kar and Peter M. Corcoran, A Review and Analysis of Eye-Gaze Estimation Systems Algorithms and Performance Evaluation Methods in Consumer Platforms  
<https://www.semanticscholar.org/paper/A-Review-and-Analysis-of-Eye-Gaze-Estimation-Systems-Algorithms-and-Performance-Evaluation-Methods-in-Consumer-Platforms>
- [6] <https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html>
- [7] Y. Lecun, L.Boutou, and Y.Bengio, Gradient-based learning applied to document recognition, Proceedings of the IEEE, vol. 88, no. 11, pp. 2278 – 2324, Nov. 1998.
- [8] Denny Britz, <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks/>
- [9] Brandon Rohrer, [http://brohrer.github.io/how\\_convolutional\\_neural\\_networks\\_work.html](http://brohrer.github.io/how_convolutional_neural_networks_work.html)
- [10] Trần Thé Anh, <http://labs.septeni-technology.jp/technote/ml-20-convolution-neural-network-part-3/>
- [11] Lương Quốc An, <http://nhiethuyettre.net/mang-no-ron-tich-chap-convolutional-neural-networks/>
- [12] <https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/googlenet.html>
- [13] Giáo trình Mạng neural, Tác giả: Phan Văn Hiền – Trường Đại học Bách khoa Đà Nẵng, 2013
- [14] Aarshay Jain, <https://www.analyticsvidhya.com/blog/2016/04/deep-learning-computer-vision-introduction-convolutional-neural-networks/>
- [15] <https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing-research/gaze-based-human-computer-interaction/its-written-all-over-your-face-full-face-appearance-based-gaze-estimation/>

- 
- [16] [https://www.tensorflow.org/versions/r0.12/get\\_started/basic\\_usage](https://www.tensorflow.org/versions/r0.12/get_started/basic_usage)
  - [17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. *Going deeper with convolutions*
  - [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition*
  - [19] Trần Thé Anh, <http://labs.septeni-technology.jp/technote/ml-18-convolution-neural-network-part-1/>
  - [20] <https://www.kernix.com/blog/a-toy-convolutional-neural-network-for-image-classification-p14>
  - [21] Kyle Kafka- Aditya Khosla- Petr Kellnhofer- Harini Kannan- Suchendra Bhandarkar- Wojciech Matusik- Antonio Torralba, Eye Tracking for Everyone <http://gazecapture.csail.mit.edu/>
  - [22] Kyle Kafka and Aditya Khosla and Petr Kellnhofer and Harini Kannan and Suchendra Bhandarkar and Wojciech Matusik and Antonio Torralba, Eye Tracking for Everyone Code Dataset and Models <https://github.com/CSAILVision/GazeCapture>
  - [23] [https://medium.com/@taolu\\_99738/developing-of-eye-tracking-application-for-smartphone-44a2a2a2a2a2](https://medium.com/@taolu_99738/developing-of-eye-tracking-application-for-smartphone-44a2a2a2a2a2)