TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN TỐT NGHIỆP

NHẬN DIỆN CHỮ VIẾT TAY SỬ DỤNG MẠNG NƠ-RON NHÂN TẠO

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ - TRUYỀN THÔNG

Sinh viên: **NGUYỄN THANH TUYẾT HÂN**
MSSV: 15141152
CAO VĂN CẢNH
MSSV: 15141104

TP. HỒ CHÍ MINH – 06/2019

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA ĐIỆN ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN TỐT NGHIỆP

NHẬN DIỆN CHỮ VIẾT TAY SỬ DỤNG MẠNG NƠ-RON NHÂN TẠO

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ TRUYỀN THÔNG

Sinh viên: **NGUYỄN THANH TUYẾT HÂN**
MSSV: 1541152
CAO VĂN CẢNH
MSSV: 15141104

Hướng dẫn**: THS. ĐẶNG PHƯỚC HẢI TRANG**

TP. HỒ CHÍ MINH – 6/2019

TRANG THÔNG TIN LUẬN VĂN

1. Thông tin sinh viên

Họ và tên sinh viên: CAO VĂN CẢNH MSSV: 15141104

Email: 15141104@student.hcmute.edu.vn Điện thoại: 0355568890

Họ và tên sinh viên: NGUYỄN THANH TUYẾT HÂN MSSV: 15141152

Email: 15141152@student.hcmute.edu.vn Điện thoại: 0785775567

2. Thông tin đề tài

- Tên của đề tài: NHẬN DIỆN CHỮ VIẾT TAY SỬ DỤNG MẠNG NƠ-RON NHÂN TẠO
- Đơn vị quản lý: Bộ môn Kỹ Thuật Máy Tính - Viễn Thông, Khoa Điện Điện Tử, Trường Đại Học Sư Phạm Kỹ Thuật Tp. Hồ Chí Minh.
- Thời gian thực hiện: Từ ngày 02 / 03 / 2019 đến ngày 15 / 06 / 2019
- Thời gian bảo vệ : Ngày 20 / 06 / 2019

3. Lời cam đoan của sinh viên

Tôi – Cao Văn Cảnh và Nguyễn Thanh Tuyết Hân cam đoan KLTN là công trình nghiên cứu của bản thân dưới sự hướng dẫn của Thạc sĩ Đặng Phước Hải Trang. Kết quả công bố trong KLTN là trung thực và không sao chép từ bất kỳ công trình nào khác.

Tp. HCM, ngày ... tháng ... năm 2019

SV thực hiện đồ án

(*Ký và ghi rõ họ tên*)

Giảng viên hướng dẫn xác nhận quyển báo cáo đã được chỉnh sửa theo đề nghị được ghi trong biên bản của Hội đồng đánh giá Khóa luận tốt nghiệp.

……

Xác nhận của Bộ Môn

Tp. HCM, ngày ... tháng ... năm 20...

Giáo viên hướng dẫn

(Ký, ghi rõ họ tên và học hàm - học vị)

BẢN NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(Dành cho giảng viên hướng dẫn)

Đề tài: Nhận diện chữ viết tay sử dụng mạng nơ-ron nhân tạo

Sinh viên thực hiện: 1. Nguyễn Thanh Tuyết Hân MSSV: 15141152

2. Cao Văn Cảnh MSSV: 15141104

Giảng viên hướng dẫn: ThS. Đặng Phước Hải Trang

Nhận xét bao gồm các nội dung sau đây:

1. Tính hợp lý trong cách đặt vấn đề và giải quyết vấn đề; ý nghĩa khoa học và thực tiễn:
Đặt vấn đề rõ ràng, mục tiêu cụ thể; đề tài có tính mới, cấp thiết; đề tài có khả năng ứng dụng, tính sáng tạo.
— Đề tài có tính ứng dụng

2. Phương pháp thực hiện/ phân tích/ thiết kế:
Phương pháp hợp lý và tin cậy dựa trên cơ sở lý thuyết; có phân tích và đánh giá phù hợp; có tính mới và tính sáng tạo.
— Phương pháp thực hiện hợp lý

3. Kết quả thực hiện/ phân tích và đánh giá kết quả/ kiểm định thiết kế:
Phù hợp với mục tiêu đề tài; phân tích và đánh giá kèm thư thiết kế hợp lý; có tính sáng tạo kiểm định chặt chẽ và đảm bảo độ tin cậy.
— Kết quả phù hợp với mục tiêu đề tài

4. Kết luận và đề xuất:
Kết luận phù hợp với cách đặt vấn đề, đề xuất mang tính cải tiến và thực tiễn; kết luận có đóng góp mới của tác giả, đề xuất sáng tạo và thuyết phục.
— Kết luận phù hợp với cách đặt vấn đề

5. Hình thức trình bày và bố cục báo cáo:
Văn phong nhất quán, bố cục hợp lý, cấu trúc rõ ràng, đúng định dạng mẫu; có tính hấp dẫn, thể hiện năng lực tốt, văn bản trau chuốt.
— Trình bày đúng định dạng mẫu

6. Kỹ năng chuyên nghiệp và tính sáng tạo:
Thể hiện các kỹ năng giao tiếp, kỹ năng làm việc nhóm, và các kỹ năng chuyên nghiệp khác trong việc thực hiện đề tài.
— Có khả năng làm việc nhóm

7. Tài liệu trích dẫn
Tính trung thực trong việc trích dẫn tài liệu tham khảo; tính phù hợp của các tài liệu trích dẫn; trích dẫn theo đúng chỉ dẫn APA.
— Trích dẫn theo đúng quy định

8. Đánh giá về sự trùng lặp của đề tài
(Cần khẳng định đề tài có trùng lặp hay không? Nếu có, đề nghị ghi rõ mức độ, tên đề tài, nơi công bố, năm công bố của đề tài đã công bố.
— Đề tài có kế thừa

9. Những nhược điểm và thiếu sót, những điểm cần được bổ sung và chỉnh sửa*

10. Nhận xét tinh thần, thái độ học tập, nghiên cứu của sinh viên

Đề nghị của giảng viên hướng dẫn
Ghi rõ: "Báo cáo đạt, không đạt yêu cầu của một khóa luận tốt nghiệp kỹ sư, và được phép/ không được phép bảo vệ trước hội đồng"
— Đạt yêu cầu KLTN, cho phép bảo vệ trước hội đồng

Tp. HCM, ngày ... tháng 6. năm 2019

Người nhận xét

(Ký và ghi rõ họ tên)

Đặng Phước Hải Trang

* *Giáo viên hướng dẫn có thể ghi tiếp vào mặt sau*

BẢN NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(Dùng cho giảng viên phản biện)

Đề tài: **Nhận diện chữ viết tay sử dụng mạng nơ-ron nhân tạo**

Sinh viên thực hiện: 1. Nguyễn Thanh Tuyết Hân MSSV: 15141152

2. Cao Văn Cảnh MSSV: 15141104

Giảng viên hướng dẫn: Ths. Đặng Phước Hải Trang

Nhận xét bao gồm các nội dung sau đây:

1. Tính hợp lý trong cách đặt vấn đề và giải quyết vấn đề; ý nghĩa khoa học và thực tiễn [15/100]:
Đặt vấn đề rõ ràng, mục tiêu cụ thể [5]; đề tài có tính mới, cấp thiết [5]; đề tài có khả năng ứng dụng, tính sáng tạo [5].

.....

.....

2. Phương pháp thực hiện/ phân tích/ thiết kế [25/100]:
Phương pháp hợp lý và tin cậy dựa trên cơ sở lý thuyết [10]; có phân tích và đánh giá phù hợp [10]; có tính mới và tính sáng tạo [5].

.....

.....

3. Kết quả thực hiện/ phân tích và đánh giá kết quả/ kiểm định thiết kế [25/100]:
Phù hợp với mục tiêu [10]; phân tích và đánh giá / kiểm thử thiết kế hợp lý [10]; có tính sáng tạo/ kiểm định chặt chẽ và đảm bảo độ tin cậy [5].

.....

.....

4. Kết luận và đề xuất [10/100]:
Kết luận phù hợp với cách đặt vấn đề, đề xuất mang tính cải tiến và thực tiễn [5]; kết luận có đóng góp mới mẻ, đề xuất sáng tạo và thuyết phục[5].

.....

.....

5. Hình thức trình bày, bố cục và chất lượng báo cáo [15/100]:
Văn phong nhất quán, bố cục hợp lý, cấu trúc rõ ràng, đúng định dạng mẫu [5]; có tính hấp dẫn, thể hiện năng lực tốt, văn bản trau chuốt [15].

.....

.....

6. Tài liệu trích dẫn [10/100]
Tính trung thực trong việc trích dẫn tài liệu tham khảo; tính phù hợp của các tài liệu trích dẫn; trích dẫn theo đúng chỉ dẫn APA.

.....

.....

7. Đánh giá về sự trùng lặp của đề tài
Cần khẳng định đề tài có trùng lặp hay không? Nếu có, đề nghị ghi rõ mức độ, tên đề tài, nơi công bố, năm công bố của đề tài đã công bố.

.....

.....

8. Những nhược điểm và thiếu sót, những điểm cần được bổ sung và chỉnh sửa*.....

.....

.....

.....

.....

Câu hỏi sinh viên phải trả lời trước hội đồng* *(ít nhất 02 câu)*

.....

.....

.....

.....

.....

Đánh giá chung

- Điểm (Quy về thang điểm 10 không làm tròn): ……./10.
- Xếp loại chung (Xuất sắc, Giỏi, Khá, Trung bình, Yếu, Kém):…….

Đề nghị của giảng viên phản biện

Ghi rõ: "Báo cáo đạt/ không đạt yêu cầu của một khóa luận tốt nghiệp kỹ sư, và được phép/ không được phép bảo vệ khóa luận tốt nghiệp"

……

Tp. HCM, ngày … tháng …. năm 20…

Người nhận xét

(*Ký và ghi rõ họ tên)*

LỜI CẢM ƠN

Đầu tiên, chúng tôi xin gửi lời cảm ơn chân thành đến Thạc sĩ Đặng Phước Hải Trang đã trực tiếp hướng dẫn cũng như tận tình chỉ bảo và giúp đỡ chúng tôi trong suốt thời gian thực hiện đồ án tốt nghiệp.

Xin gửi lời cảm ơn đến tất cả các quý thầy cô khoa Điện – Điện tử, những bài giảng của thầy cô đã trang bị kiến thức vững chắc để bây giờ chúng tôi vận dụng thực hiện đồ án tốt nghiệp.

Chân thành cảm ơn Bộ môn Máy tính - Viễn Thông, Ban giám hiệu nhà trường đã tạo điều kiện tốt cho chúng tôi về tài liệu học tập và trang thiết bị trong suốt thời gian học tập tại trường, nhất là trong quá trình chúng tôi thực hiện đồ án.

Ngoài ra, khi thực hiện đồ án này, chúng tôi cũng nhận được nhiều sự giúp đỡ từ những sinh viên khoa Điện – Điện tử khác. Xin gửi lời cảm ơn đến các bạn đã giúp đỡ mình trong quá trình làm đồ án.

Một lần nữa, chúng tôi xin chân thành cảm ơn.

Trân Trọng

Nhóm thực hiện

Nguyễn Thanh Tuyết Hân – Cao Văn Cảnh

TÓM TẮT

Hiện nay, sự phát triển của Internet giúp con người có thể truy cập được nhiều nguồn kiến thức khác nhau một cách dễ dàng hơn. Điều đó giúp cho khoa học kỹ thuật phát triển mạnh mẽ hơn bao giờ hết. Trí tuệ nhân tạo (Artificial Intelligence) đã được nghiên cứu từ cuối những năm 1980 và 1990, và hiện nay đang tạo nên những thay đổi lớn trên nhiều mặt của đời sống con người. Một số ứng dụng như nhận diện khuôn mặt, nhận diện giọng nói hay phát hiện ngôn ngữ đã được nhiều hãng công nghệ phát triển và áp dụng. Với mong muốn tìm hiểu về công nghệ này chúng tôi thực hiện nghiên cứu và thực hiện ứng dụng nhận diện chữ viết tay sử dụng công nghệ deep learning.

Ứng dụng nhận diện chữ viết tay có thể ứng dụng trên nhiều mặt đời sống con người như chuyển những tài liệu viết tay thành văn bản trên máy tính. Google Dịch đã ứng dụng công nghệ này trong chương trình của họ. Nhờ vào nhận biết chữ qua hình ảnh hoặc lựa chọn phương pháp viết tay, người dùng có nhiều lựa chọn hơn trong việc nhập dữ liệu, qua đó người sử dụng cảm thấy thuận tiện hơn trong lúc sử dụng phần mềm. Ngoài ra, công nghệ này còn có thể ứng dụng trong nhiều mặt khác như chấm bài thi trắc nhiệm bằng máy tính,....

Trong đề tài này, chúng tôi sẽ tập trung nghiên cứu về mạng nơ-ron nhân tạo và ứng dụng nó vào nhận biết chữ viết tay. Trong mô hình được xem xét, hình ảnh đầu vào là hình ảnh tĩnh được chụp từ webcam của máy tính, sau khi đưa qua hệ thống tách biên thì từng chữ cái sẽ được đưa qua mạng nơ-ron nhân tạo để đưa ra kết quả.

Do kiến thức và thời gian có hạn nên chúng tôi chỉ có thể nghiên cứu một cách cơ bản về công nghệ này, mong nhận được sự góp ý của quý Thầy Cô và các bạn sinh viên.

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

CÁC TỪ VIẾT TẮT

AI	Artificial Intelligence
MLP	Multi-layer perceptron
DBN	Deep Belief Nets
ReLU	Rectified Linear Unit
RoI	Region Of Interest
T.gian	Thời gian đáp ứng của mạng nơ-ron

CHƯƠNG 1

GIỚI THIỆU

1.1 GIỚI THIỆU

Nhận diện là bài toán xuất hiện cách đây khá lâu và vẫn luôn thu hút nhiều sự quan tâm, nghiên cứu. Đặc biệt là trong vài thập niên gần đây, do sự thúc đẩy của quá trình tin học hóa trong mọi lĩnh vực, bài toán nhận dạng không còn dừng lại ở mức độ nghiên cứu nữa mà nó trở thành một lĩnh vực để áp dụng vào thực tế. Các bài toán nhận dạng đang được ứng dụng trong thực tế hiện nay tập trung vào nhận dạng mẫu, nhận dạng tiếng nói và nhận dạng chữ. Trong số này, nhận dạng chữ là bài toán được quan tâm rất nhiều và cũng đã đạt được nhiều thành tựu rực rỡ. Các ứng dụng có ý nghĩa thực tế lớn có thể kể đến như: nhận dạng chữ in dùng trong quá trình sao lưu sách báo trong thư viện, nhận dạng chữ viết tay dùng trong việc phân loại thư ở bưu điện, thanh toán tiền trong nhà bang và lập thư viện sách cho người mù (ứng dụng này có ý nghĩa: scan sách bình thường, sau đó cho máy tính nhận dạng và trả về dạng tài liệu mà người mù có thể đọc được).

Xuất phát từ yêu cầu thực tế, đang rất cần có những nghiên cứu về vấn đề này. Chính vì vậy, chúng tôi đã chọn đề tài nhận dạng ký tự viết tay làm đồ án tốt nghiệp với mong muốn phần nào áp dụng vào bài toán thực tế.

1.2 MỤC TIÊU ĐỀ TÀI

- Xây dựng một mạng nơ-ron nhân tạo có khả năng nhận diện được chữ viết tay với độ chính xác tốt.
- Thu nhận ảnh từ camera và đưa vào hệ thống để xử lý. Sau đó, tách ảnh văn bản thành từng ảnh ký tự theo thứ tự từ trái sang phải. Sau khi đưa vào mạng

nơ-ron, hệ thống sẽ tiến hành nhận dạng chữ đã viết và xuất ra kết quả trên cửa sổ hệ thống.

- Thiết kế một giao diện hoàn chỉnh, thuận tiện cho việc sử dụng.

1.3 GIỚI HẠN ĐỀ TÀI

- Hệ thống không nhận dạng được số đếm cũng như không phát hiện được khoảng trắng giữa các từ.
- Hệ thống phải được chạy trên máy tính có cài sẵn ngôn ngữ Python, Pycharm và một số thư viện cần thiết.
- Số lượng ký tự được nhận dạng cùng một lúc không thể trên 22, các ký tự không được viết dính liền nhau.
- Khoảng cách giữa camera và ký tự cần nhận dạng từ 15cm đến 60cm, ký tự cần nhận dạng có chiều cao từ 1cm đến 15cm, chiều rộng chiếm 2/3 chiều dài ký tự, độ dày tối của ký tự là từ khoảng 0.1cm đến 0.5cm, với độ dày 0.5cm ký tự cần nhận dạng có độ chính xác cao hơn..
- Ảnh sau khi được chụp phải có phông nền trắng và chữ màu đen hoặc xanh. Ảnh không được chứa bất kì thành phần khác ngoài hai thành phần trên.
- Camera phải có chất lượng tốt, ánh sáng trong phòng vừa đủ và đều.

1.4 PHƯƠNG PHÁP NGHIÊN CỨU

- Tìm hiểu về Deep Learning (Học Sâu) và Artificial Neural Network (Mạng nơ-ron nhân tạo).
- Tìm hiểu về ngôn ngữ Python, thư viện OpenCV, Tensorflow và Keras.
- Tìm hiểu về tiền xử lý ảnh và các phương pháp phát hiện biên.
- Xây dựng chương trình trên máy tính cá nhân.

1.5 ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng nghiên cứu:

- Các phương pháp xử lí ảnh đầu vào.
- Các phương pháp nhận diện chữ viết tay.
- Ngôn ngữ lập trình Python, thư viện OpenCV, Tensorflow, Keras.

Phạm vi nghiên cứu:

- Việc nhận dạng chữ viết tay phải thỏa mãn các điều kiện sau:
  + Chữ phải được viết trên cùng một hàng, không viết quá dính liền nhau.
  + Chất lượng camera tốt.
  + Ánh sáng tốt.

1.6 BỐ CỤC QUYỂN BÁO CÁO

Chương 1: Giới thiệu

Trong chương này nêu ra được tình hình nghiên cứu hiện nay, lý do và mục tiêu chọn đề tài, đối tượng và phạm vi nghiên cứu của đề tài, phương pháp nghiên cứu và giới hạn đề tài.

Chương 2: Cơ sở lý thuyết

- Lý thuyết về ảnh số và xử lý ảnh.
- Lý thuyết về lọc nhiễu, nhị phân hóa và phát hiện biên ký tự.
- Các lý thuyết chính liên quan đến phương pháp phát hiện và nhận diện chữ viết.

Chương 3: Thiết kế hệ thống

Trong chương này mục đích là thiết kế phần cứng và xây dựng chương trình cho hệ thống với những yêu cầu đặt ra:

- Thiết kế phần cứng: Tiến hành lựa chọn Camera phù hợp với máy tính cá nhân có sẵn để tiết kiệm chi phí trong quá trình nghiên cứu.
- Thiết kế chương trình: Xây dựng chương trình nhận diện chữ viết trên máy tính cá nhân sử dụng ngôn ngữ lập trình Python và thư viện Tensorflow cũng như Keras.

Chương 4: Kết quả

Trình bày về kết quả thi công phần cứng, phần mềm và đánh giá ưu nhược điểm của hệ thống.

Chương 5: Kết luận và hướng phát triển

Đưa ra các kết luận về những vấn đề mà trong quá trình nghiên cứu đã đạt được.

CHƯƠNG 2

CƠ SỞ LÝ THUYẾT

2.1 TÌNH HÌNH NGHIÊN CỨU

Trong cuộc sống này nay, công nghệ được ứng dụng trong nhiều lĩnh vực từ nông nghiệp cho đến công nghiệp, nhờ sự phát triển mạnh mẽ của ngành công nghệ thông tin, ngành công nghiệp máy tính cũng như sự xuất hiện của Big Data, một số trí tuệ nhân tạo (AI) đã được ra đời. AI là trí tuệ do con người phát triển nhằm mục tiêu giúp máy tính có thể nhận biết và có một số hành vi thông minh như con người. Các ứng dụng của AI hiện nay được nhiều công ty công nghệ áp dụng vào sản phẩm của họ nhằm nâng cao đời sống của con người hiện nay. Google đã xây dựng được một hệ thống có khả năng nhận diện chữ viết cực kì chính xác ở khá nhiều ngôn ngữ khác nhau từ tiếng anh, trung, nhật...

2.2 XỬ LÝ ẢNH

Đầu vào của quá trình xử lý ảnh là các ảnh gốc ban đầu, thu được qua máy ảnh, thiết bị ghi hình hay thiết bị thu nhận hình ảnh khác. Ảnh ban đầu thường có chất lượng thấp (do ảnh hưởng của nhiễu, bị lệch so với ảnh gốc một góc bất kỳ, bị đứt nét). Nguyên nhân là do thiết bị thu nhận không đảm bảo, điều kiện thu nhận không tốt (độ sáng thay đổi, thu nhận trong khi di chuyển) hay quá trình sao lưu bị mất mát thông tin. Nên quá trình tiền xử lý ảnh để nâng cao chất lượng ảnh đầu vào trước khi đưa vào nhận dạng là cực kì cần thiết. Quá trình này bao gồm công đoạn khôi phục ảnh và tăng cường ảnh.

Khôi phục ảnh nhằm mục đích loại bỏ hay làm giảm tối thiểu các ảnh hưởng của môi trường bên ngoài lên ảnh thu nhận được. Công đoạn khôi phục ảnh bao gồm các bước như lọc ảnh, khử nhiễu, quay ảnh,.. qua đó nhằm giảm bớt các biến dạng do quá trình quét ảnh gây ra và đưa ảnh về trạng thái gần như ban đầu.

Tăng cường ảnh là một công đoạn quan trọng, tạo tiền đề cho xử lý ảnh. Tăng cường ảnh không phải làm tăng lượng thông tin trong ảnh mà là làm nỗi bật những đặc trưng của ảnh giúp cho công việc xử lý phía sau được hiệu quả hơn. Công đoạn này bao gồm các công việc như: lọc độ tương phản, làm trơn ảnh, nhị phân hóa,…

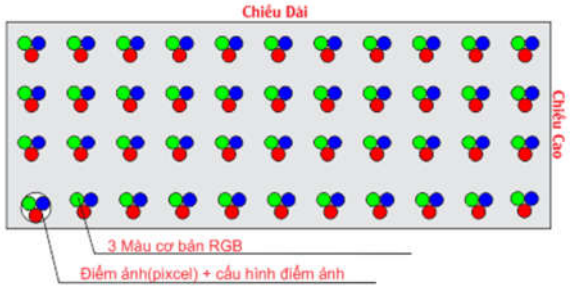
2.2.1 Khái niệm cơ bản về ảnh số

Ảnh số là tập hợp hữu hạn các điểm ảnh với mức xám phù hợp dùng để mô tả ảnh gần với ảnh thật. Số điểm ảnh xác định độ phân giải của ảnh. Ảnh có độ phân giải càng cao thì càng thể hiện rõ nét các đặt điểm của tấm hình càng làm cho tấm ảnh trở nên thực và sắc nét hơn. [1]

Hình 2.1: Ảnh số [1]

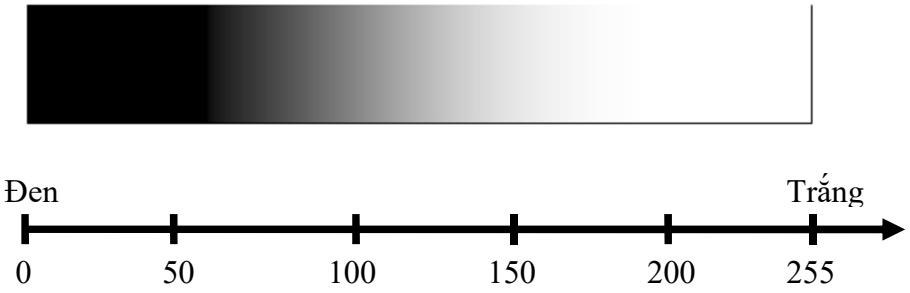
Các thông số cơ bản của ảnh được trình bày như sau:

Điểm ảnh: là một phần tử của ảnh số tại toạ độ (x, y) với độ xám hoặc màu nhất định. Kích thước và khoảng cách giữa các điểm ảnh đó được chọn thích hợp sao cho mắt người cảm nhận sự liên tục về không gian và mức xám (hoặc màu) của ảnh số gần như ảnh thật. Mỗi phần tử trong ma trận được gọi là một phần tử ảnh. [1]



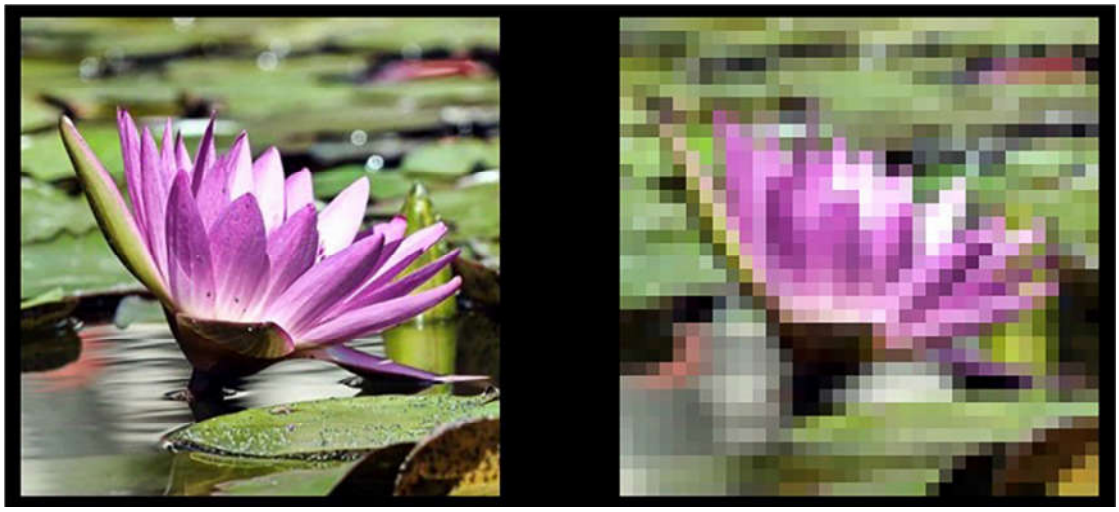
Hình 2.2: Điểm ảnh [3]

Mức xám của ảnh: Là kết quả của sự biến đổi tương ứng 1 giá trị độ sáng của 1 điểm ảnh với 1 giá trị nguyên dương. Thông thường nó xác định trong [0, 255] tuỳ thuộc vào giá trị mà mỗi điểm ảnh được biểu diễn. Các thang giá trị mức xám thông thường: 16, 32, 64, 128, 256 (Mức 256 là mức phổ dụng. Lý do: từ kỹ thuật máy tính dùng 1 byte (8 bit) để biểu diễn mức xám. Mức xám dùng 1 byte biểu diễn: 28 =256 mức, tức là từ 0 đến 255). [1]



Hình 2.3: Mức xám của ảnh

Độ phân giải của ảnh: là mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị. Theo định nghĩa, khoảng cách giữa các điểm ảnh phải được chọn sao cho mắt người vẫn thấy được sự liên tục của ảnh. Việc lựa chọn khoảng cách thích hợp tạo nên một mật độ phân bổ, đó chính là độ phân giải và được phân bố theo trục x và y trong không gian hai chiều. [2]

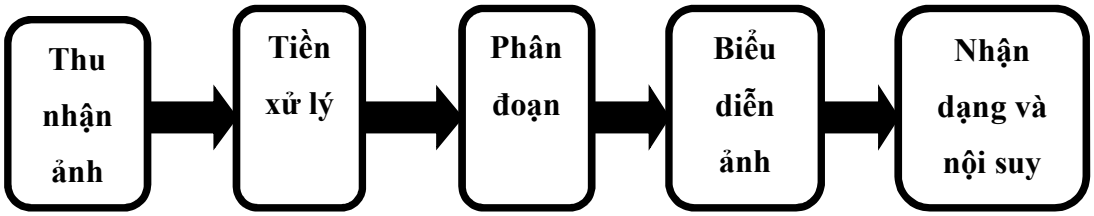


Độ phân giải cao Độ phân giải thấp

Hình 2.4: Độ phân giải [3]

2.2.2 Các giai đoạn xử lý ảnh

Quá trình xử lý ảnh được thể hiện như hình 2.5



Hình 2.5: Quá trình xử lý ảnh

2.2.2.1 Thu nhận ảnh

Ảnh thu được từ nhiều nguồn khác nhau như máy ảnh, máy quay phim, máy quét, các thiết bị ghi hình khác.

Hình 2.6: Một số thiết bị thu hình

2.2.2.2 Tiền xử lý

Sử dụng các kỹ thuật xử lý ảnh để nâng cao chất lượng ảnh. Mục đích là để điều chỉnh độ sáng nhằm khắc phục hậu quả của việc chiếu sáng không đều, giảm nhỏ thành phần nhiễu của ảnh tức là các đối tượng xuất hiện ngoài ý muốn, hiệu chỉnh giá trị độ sáng giữa nền và đối tượng, chuẩn hoá độ lớn, màu, dạng của ảnh. Ảnh thu thập được thường có nhiễu và cần loại bỏ nhiễu hay ảnh thu được không sắc nét, bị mờ cần làm rõ các chi tiết trước khi đưa vào xử lý.

➢ Chuyển xám ảnh

Đơn vị tế bào của ảnh số là pixel. Tùy theo mỗi định dạng là ảnh màu hay ảnh xám mà từng pixel có thông số khác nhau. Đối với ảnh màu từng pixel sẽ mang thông tin của ba màu cơ bản tạo ra bản màu khả kiến là Đỏ (R), Xanh lá (G) và Xanh biển (B). Trong mỗi pixel của ảnh màu, ba màu cơ bản R, G và B được bố trí sát nhau và có cường độ sáng khác nhau. Thông thường, mỗi màu cơ bản được biểu diễn bằng tám bit tương ứng 256 mức độ màu khác nhau. Như vậy mỗi pixel chúng ta sẽ có $2^{8x3} = 2^{24}$ màu (khoảng 16.78 triệu màu). Đối với ảnh xám, thông thường mỗi pixel mang thông tin của 256 mức xám (tương ứng với tám bit) như vậy ảnh xám hoàn toàn có thể tái hiện đầy đủ cấu trúc của một ảnh màu tương ứng thông qua tám mặt phẳng bit theo độ xám.

Ảnh gốc Ảnh xám

Hình 2.7: Ảnh trước và sau khi chuyển xám

Trong hầu hết quá trình xử lý ảnh, cấu trúc của ảnh được quan tâm nhiều và ảnh hưởng của yếu tố màu sắc thì được bỏ qua. Do đó bước chuyển từ ảnh màu thành ảnh xám là một công đoạn phổ biến trong các quá trình xử lý ảnh vì nó làm tăng tốc độ xử lý là giảm mức độ phức tạp của các thuật toán trên ảnh.

Công thức chuyển các thông số giá trị màu của một pixel thành mức xám tương ứng như sau:

$$G = \alpha.CR + \beta.CG + \delta.CB \quad (2.1)$$

Trong đó các giá trị CR, CG và CB lần lượt là các mức độ màu Đỏ, Xanh và Xanh biển của pixel màu.

➢ Phương pháp lọc nhiễu Gaussian Blur

Bộ lọc Gaussian là bộ lọc thông cao (chỉ giữ lại thành phần tần số thấp), một

cách trực quan hơn đây được xem là phương pháp làm mờ mịn giống như hiệu ứng hình ảnh được đặt dưới một lớp màn trong suốt bị mờ. Gaussian blur là một bộ lọc làm mờ ảnh, sử dụng lý thuyết hàm Gaussian để tính toán việc chuyển đổi mỗi pixel của hình, phương trình hàm Gaussian dùng trong không hai chiều:

$$G(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.2)$$

Trong đó x và y là tọa độ theo hai trục đứng và ngang còn σ là giá trị quyết định độ lệch giữa các điểm trên bề mặt Gaussian. [3]

- Nhị phân hóa

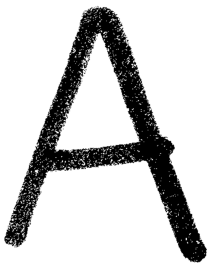
Quá trình này chuyển từ ảnh màu, ảnh đa cấp xám sang ảnh nhị phân (ảnh hai cấp xám, ảnh đen trắng). Thuật toán phân ngưỡng cài đặt ở đây sử dụng hàm phân ngưỡng:

$$g'(x,y) = \begin{cases} 1, & nếu\ g(x,y) \geq T \\ 0, & nếu\ g(x,y) < T \end{cases} \quad (2.3)$$

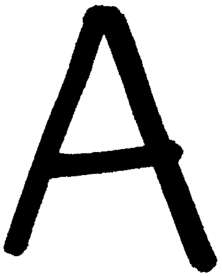
Trong đó, g(x,y) là giá trị điểm ảnh ở vị trí (x,y) của ảnh nguồn, g'(x,y) là giá trị điểm ảnh tương ứng ở vị trí (x,y) của ảnh đích. T là giá trị ngưỡng.

Giá trị cụ thể của ngưỡng phụ thuộc vào từng ảnh, vùng ảnh đầu vào đang xét và không thể lấy cố định. Ví dụ trên hình 2.8, hình a) là ảnh ban đầu, hình b) ,c) ,d) thể hiện ảnh đã được nhị phân hóa với cùng ngưỡng thấp, trung bình và ngưỡng cao. Chúng ta có thể thấy là giá trị ngưỡng trong hình c là thích hợp hơn cả. [4]

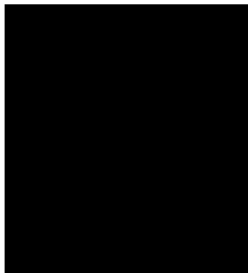
a) Ảnh gốc ban đầu



b) Ngưỡng thấp (50)



c). Ngưỡng trung bình (127)



d). Ngưỡng cao (255)

Hình 2.8: Phương pháp lấy ngưỡng

- Phương pháp giãn ảnh

Là quá trình chập của những bức ảnh nhằm loại bỏ điểm đen bị vây bởi các điểm trắng. Trong kỹ thuật này một ma trận (N+1)x(N+1) được quét trên toàn ảnh và thực hiện so sánh một pixel của ảnh với (N+1)x2-1 điểm lân cận.

$$A \oplus B = \{ c \mid c = a + b, a \in A, b \in B \} \quad (2.4)$$

Trong đó A là ma trận điểm ảnh của ảnh nhị phân, B là phần tử cấu trúc.

2.2.2.3 Phân đoạn ảnh

Là quá trình phân chia nội dung các đối tượng cần khảo sát ra khỏi ảnh, phân chia các đối tượng tiếp giáp nhau, phân tách các đối tượng riêng biệt thành các đối tượng con.

2.2.2.4 Biểu diễn ảnh

Đầu ra ảnh sau phân đoạn chứa các điểm ảnh của vùng ảnh (ảnh đã phân đoạn) cộng với mã liên kết với các vùng lận cận. Việc biến đổi các số liệu này thành dạng thích hợp là cần thiết cho xử lý tiếp theo bằng máy tính. Việc chọn các tính chất để thể hiện ảnh gọi là trích chọn đặc trưng gắn với việc tách các đặc tính của ảnh dưới dạng các thông tin định lượng hoặc làm cơ sở để phân biệt lớp đối tượng này với đối tượng khác trong phạm vi ảnh nhận được.

2.2.2.5 Nhận dạng và nội suy ảnh

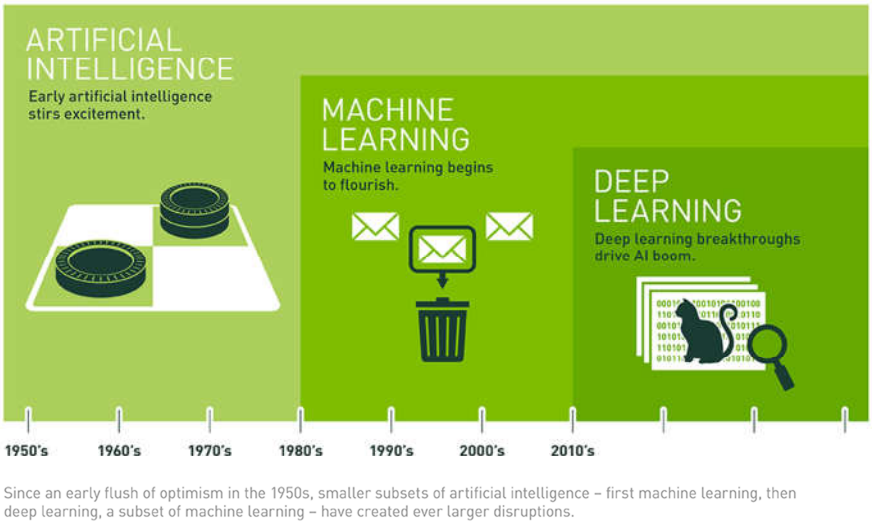
Nhận dạng ảnh là quá trình xác định ảnh. Quá trình này thường thu được bằng cách so sánh với mẫu chuẩn đã được học (hoặc lưu) từ trước. Theo lý thuyết về nhận dạng, các mô hình toán học về ảnh được phân theo hai loại nhận

dạng ảnh cơ bản là nhận dạng theo tham số và nhận dạng theo cấu trúc. Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng.

2.3 DEEP LEARNING

2.3.1 Giới thiệu

Học Sâu (Deep Learning) là một chi của ngành Máy Học (Machine Learning), dưới đây là mối quan hệ giữa Học Sâu, Máy Học, mạng nơ-ron nhân tạo (Artificial Neural Network), và Trí tuệ nhân tạo (Artificial Intelligence viết tắt là AI).



Hình 2.9: Mối quan hệ giữa Học Sâu, Máy Học và Trí tuệ nhân tạo [5]

Trí tuệ nhân tạo có thể được định nghĩa như một ngành của khoa học máy tính, liên quan đến việc tự động hóa các hành vi thông minh. Trong tương lai, AI là một chương trình máy tính có thể tư duy, suy nghĩ, học hỏi, … như con người. Chúng có thể xử lý dữ liệu ở mức rộng lớn, quy mô, hệ thống, khoa học và nhanh hơn so với con người.

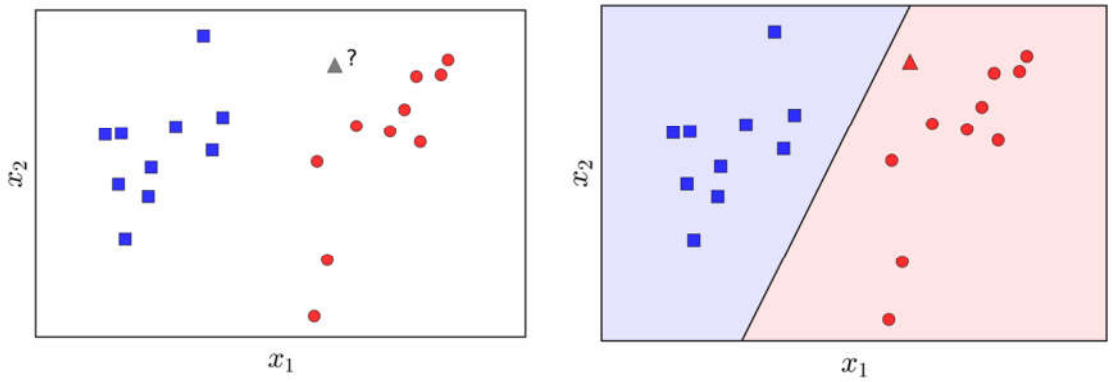
Máy học là một thuật ngữ để chỉ hành động dạy cho máy tính thực hiện một nhiệm vụ nào đó. Máy học sử dụng thuật toán để phân tích những thông tin, dữ liệu có sẵn, học hỏi từ nó rồi đưa ra quyết định hoặc dự đoán về một thứ gì đó có liên quan. Máy học mang lại cho máy tính sức mạnh học hỏi mà không cần các nhà nghiên cứu, lập trình viên phải lập trình một cách rõ ràng.

Học sâu là một tên gọi dễ hiểu hơn của mạng nơ-ron nhân tạo. Chúng được sử dụng để giúp robot học tập tốt hơn. Kỹ thuật này lấy ý tưởng từ bộ não sinh học con người. Một hệ thần kinh của con người bao gồm nhiều nơ-ron, trong đó, một nơ-ron sẽ nhận thông tin từ một số nơ-ron xử lý tín hiệu rồi truyền cho nơ-ron khác liên kết với nó.

2.3.2 Một số dấu mốc quan trọng của Deep Learning

Vào những năm 1960, một trong những nền móng đầu tiên của Học Sâu là Perceptron Learning Algorithm (hoặc gọi là Perceptron) ra đời. Perceptron là một thuật toán của học có giám sát (Supervised Learning) giúp giải quyết bài toán phân lớp nhị phân, được khởi nguồn bởi Frank Rosenblatt năm 1957. Đây là nền tảng quan trọng cho Deep Learning sau này. [6]

Giả sử có hai tập hợp dữ liệu đã được tô màu trong hình bên trái dưới đây, hai tập này là tập hợp của các điểm màu xanh và tập hợp các điểm màu đỏ. Bài toán đặt ra là tìm và vẽ giới hạn cho hai tập hợp dữ liệu để khi có một điểm dữ liệu mới ta có thể dự đoán được màu của nó. [7]



Hình 2.10: Bài toán phân lớp nhị phân [7]

Vào những năm 1980, Geoffrey Hinton và hai tác giả khác xuất bản một bài báo khoa học trên Nature với tựa đề "Learning representations by back-propagating errors". Trong bài báo này, nhóm của ông chứng minh rằng một hệ thống thần kinh nhân tạo với nhiều hidden layer (được gọi là multi-layer perceptron hoặc MLP) có thể được huấn luyện một cách hiệu quả dựa trên một quy trình đơn giản được gọi là Backpropagation. Việc này giúp Deep Learning thoát được những hạn chế của perceptron về việc chỉ biểu diễn được các quan hệ

tuyến tính. Để biểu diễn các quan hệ phi tuyến, phía sau mỗi lớp là một hàm kích hoạt phi tuyến.

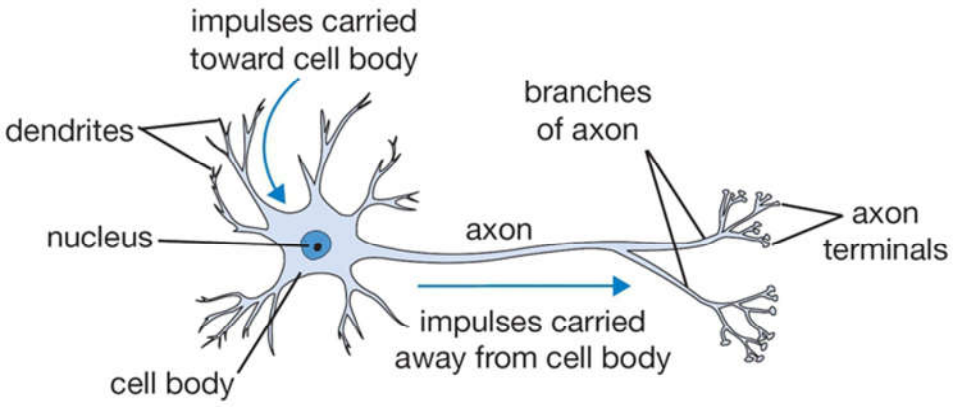
Năm 2006, Hinton giới thiệu ý tưởng của tiền huấn luyện không giám sát (unsupervised pretraining) thông qua deep belief nets (DBN). Qua đó khắc phục được phần nào vấn đề trọng số bị lãng quên (vanishing gradient). Đây là hiện tượng những trọng số ở các lớp đầu tiên thường khó huấn luyện vì tác dụng của chúng là quá nhỏ so với đầu ra nhất là đối với những hệ thống có nhiều lớp nơ-ron.

Năm 2012, một số kỹ thuật như hàm ReLU và dropout được ra đời. Hàm ReLU có cách tính và đạo hàm đơn giản giúp tốc độ huấn luyện tăng đáng kể và cũng giúp cho vấn đề vanishing gradient được giải quyết một phần. Dropout là kỹ thuật mà trong quá trình training, nhiều nơ-ron ẩn bị tắt một cách ngẫu nhiên và mô hình sẽ được huấn luyện với các tham số còn lại. Trong quá trình test, toàn bộ nơ-ron sẽ được sử dụng. Với tổ hợp nơ-ron bị tắt khác nhau, ta thu được nhiều mô hình và việc kết hợp cuối cùng được coi như sự kết hợp của nhiều mô hình, qua đó tránh được overfitting và mô hình cũng mang lại hiệu quả cao hơn. [6]

2.3.3 Mạng nơ-ron nhân tạo

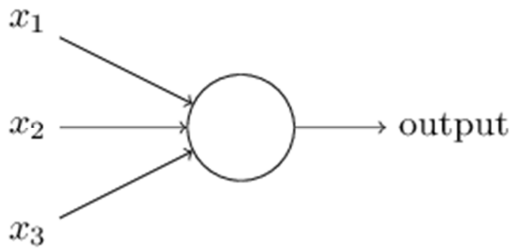
2.4.3.1 Perceptron cơ bản

Một mạng nơ-ron nhân tạo được cấu thành bởi các nơ-ron đơn lẻ được gọi là các perceptron. Nơ-ron nhân tạo được lấy cảm hứng từ nơ-ron sinh học như hình mô tả bên dưới:



Hình 2.11: Nơ-ron sinh học [8]

Như hình trên, một nơ-ron có thể nhận nhiều đầu vào và cho ra một kết quả đầu duy nhất. Mô hình của perceptron cũng tương tự:



Hình 2.12: Mô hình nơ-ron nhân tạo [8]

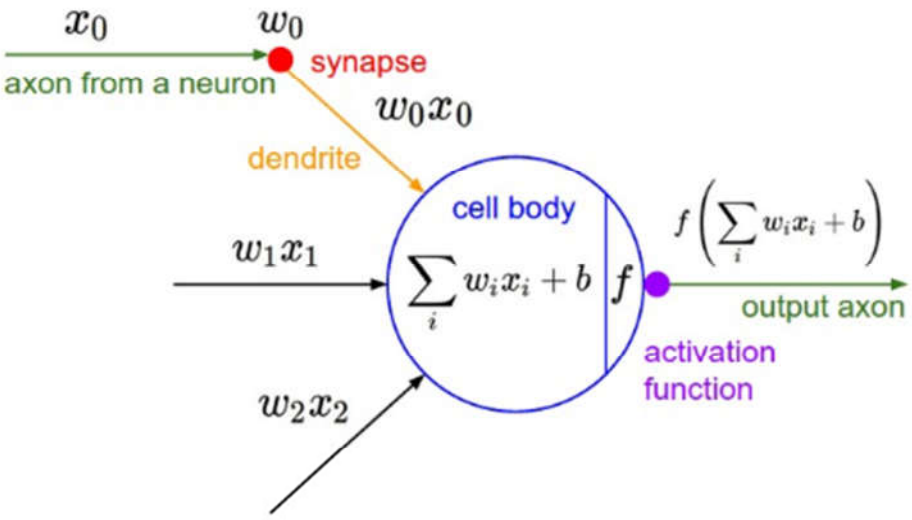
Một perceptron sẽ nhận một hoặc nhiều đầu vào (kí hiệu là x) dưới dạng nhị phân và cho ra một kết quả (kí hiệu là o) duy nhất cũng dưới dạng nhị phân. Tầm ảnh hưởng của mỗi đầu vào khác nhau được quyết định bởi tham số trọng lượng (weight kí hiệu là w) của nó. Kết quả đầu ra được quyết định dựa vào một ngưỡng b (bias) nào đó:

$$o = \begin{cases} 0 & if \ \sum_i \omega_i x_i + b \ \leq 0 \\ 1 & if \ \sum_i \omega_i x_i + b \ > 0 \end{cases} \quad (2.5)$$

2.4.3.2 Nơ-ron với Hàm kích hoạt (Activation Function)

Với đầu vào và ra dạng nhị phân, hệ thống sẽ không có độ linh hoạt nên để tăng độ linh động thì giá trị của đầu vào và đầu ra sẽ ở trong khoảng [0,1]. Đề làm được điều này đầu ra sẽ được quyết định bởi một hàm kích hoạt f(z). Hàm này có thể là hàm sigmoid, tanh, hoặc ReLU,…

Qua đó một nơ-ron nhân tạo có thể được biểu diễn như sau:



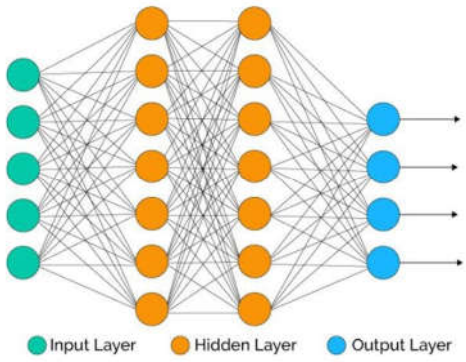
Hình 2.13: Nơ-ron nhân tạo có Hàm Kích Hoạt [8]

2.4.3.3 Kiến trúc của một mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo là sự kết hợp của các nơ-ron đơn lẻ. Một mạng nơ-ron nhân tạo sẽ có 3 kiểu tầng:

- Tầng vào (input layer): thể hiện đầu vào của mạng.
- Tầng ra (output layer): thể hiện đầu ra của mạng.
- Tầng ẩn (hidden layer): nằm giữa tầng vào và tầng ra thực hiện chức năng suy luận logic của mạng.

Mỗi mạng nơ-ron chỉ có tầng vào và 1 tầng ra nhưng có thể có nhiều tầng ẩn. Số lượng nơ-ron mỗi tầng khác nhau tùy thuộc vào công việc mà mạng đó thực hiện. Mỗi nơ-ron trong một lớp thường được liên kết với từng nơ-ron ở lớp tiếp theo. Mạng này còn được gọi là mạng kết nối đầy đủ (full-connected network).[10]

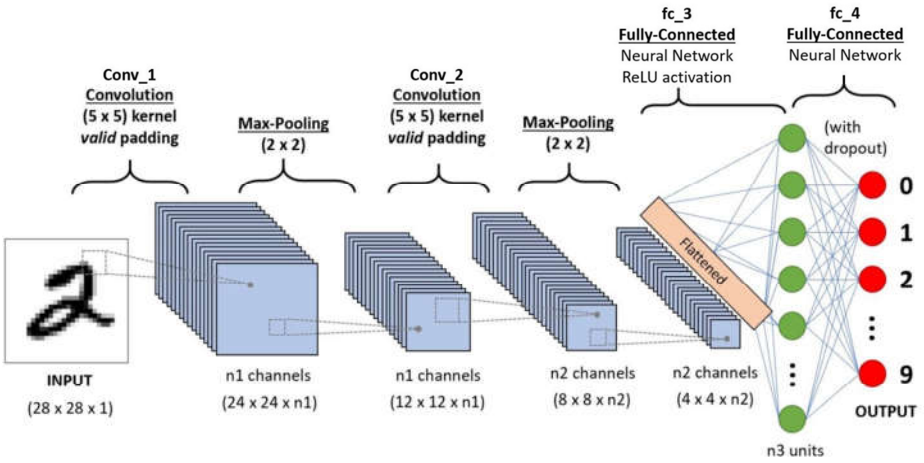


Hình 2.14: Mạng nơ-ron kết nối đầy đủ

Trong mạng nơ-ron nhân tạo, các nơ-ron được kết hợp theo một chiều duy nhất từ tầng vào đến tầng ra. Nói cách khác mỗi nơ-ron trong một tầng sẽ nhận đầu vào là tất cả tín hiệu của nơ-ron của tầng trước đó mà không chuyển tín hiệu ngược lại nên dạng lan truyền này gọi là lan truyền tiến (feedforwark).[9]

Để một mạng nơ-ron nhận biết được quan hệ giữa đầu vào và đầu ra cần một quá trình huấn luyện. Trong quá trình huấn luyện một số lượng lớn dữ liệu sẽ được đưa vào tầng vào, qua một quá trình tính toán mạng sẽ đưa ra một kết quả đầu ra (gọi là prediction). Qua việc tính toán sự khác biệt (loss) giữa kết quả đầu ra và kết quả mong muốn (gọi là expectation) có sẵn, sau đó điều chỉnh tham số trọng lượng cũng như ngưỡng giữa các lớp nơ-ron, sự khác biệt của hai kết quả trên sẽ ngày càng nhỏ và hệ thống sẽ ngày càng chính xác. Quá trình điều chỉnh ngược lại tham số trọng lượng giữa các nơ-ron gọi là quá trình backpropagation.

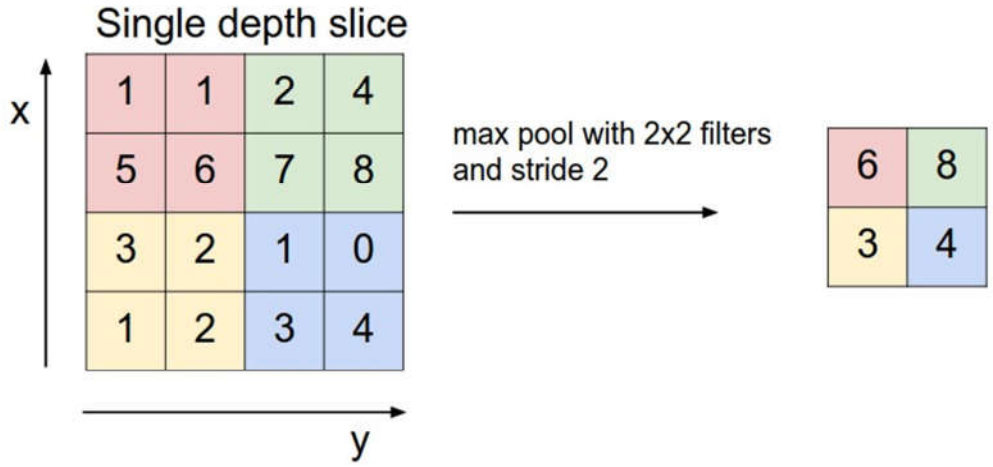
Với một bài toán nhận diện hình ảnh, nếu chỉ sử dụng mạng nơ-ron truyền thống như trên thì hệ thống sẽ chỉ phân biệt được những bức ảnh đơn giản với độ chính xác có thể chấp nhận được. Khi kích thước của bức ảnh tăng hoặc người nghiên cứu muốn tăng độ chính xác của hệ thống, mạng nơ-ron tích chập cho ra kết quả tốt hơn nhiều so với mạng nơ-ron được kết nối đầy đủ.



Hình 2.15: Mạng nơ-ron tích chập

Mạng nơ-ron tích chập nhận diện về không gian cục bộ tốt hơn mạng nơ-ron truyền thống vì thông thường không phải nơ-ron nào của lớp trước cũng được kết nối với nơ-ron của tầng tiếp theo. Trong mạng nơ-ron tích giá trị của mỗi nơ-ron

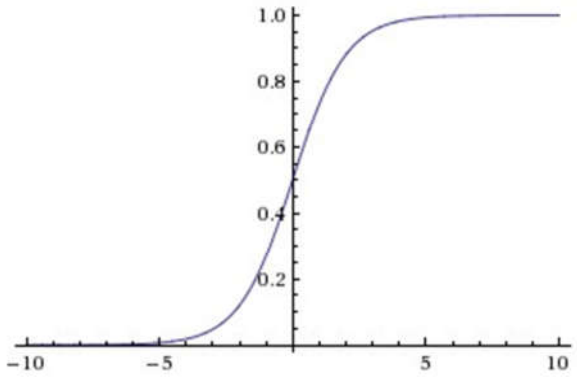
của tầng tiếp theo là kết quả tích chập của tầng trước đó, nhờ đó kết nối cục bộ giữa các pixel được sinh ra. Ngoài ra nhờ vào lớp Tổng hợp (Pooling), thông tin có trong các lớp tích chập sẽ được chắt lọc và tổng hợp. Qua đó nhiễu sẽ được loại bỏ vì lớp này chỉ lấy pixel có giá trị lớn nhất trong một vùng nào đó mà nhiễu thường có giá trị nhỏ. Thêm vào đó lớp này còn có công dụng làm giảm kích thước đầu vào của lớp tiếp theo.[11]



Hình 2.16: Lớp Tổng hợp

2.3.4 Một số Hàm Kích Hoạt

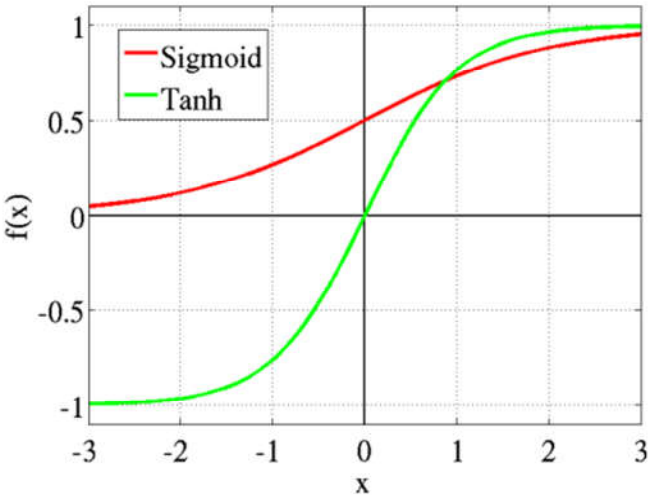
Hàm sigmoid có công thức: f(s) = 1/ (1+exp(-s)) và có đồ thị như sau:



Hình 2.17: Đồ thị hàm sigmoid

Lý do chính hàm sigmoid thường được sử dụng là bởi vì giá trị đầu ra của nó tồn tại giữa 0 và 1. Trong những mô hình mà phải dự đoán đầu ra có dạng xác xuất thì kết quả đầu ra phải là giá trị từ 0 đến 1 (vì xác xuất của mọi thử tồn tại trong khoảng từ 0 đến 1) nên hàm sigmoid đáp ứng được yêu cầu nêu trên.

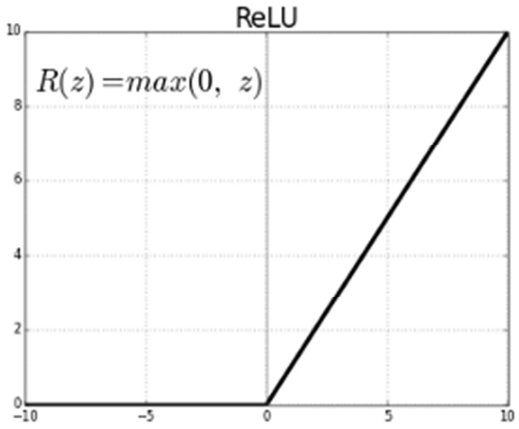
Hàm tanh hoặc còn gọi là hyperbol: có đồ thị khá giống hàm sigmoid nhưng tốt hơn. Kết quả đầu ra của hảm tanh là từ -1 đến 1. Đồ thị của hàm tanh và hàm sigmoid:



Hình 2.20: Đồ thị hàm tanh so với sigmoid [12]

Ưu điểm của hàm này là nếu đầu vào âm rất nhiều thì kết quả là -1, hàm này thường được sử dụng trong những bài toán phân loại giữa hai lớp. [12]

Hàm đơn vị tuyến tính chỉnh lưu (Rectified Linear Unit hay còn gọi là hàm ReLU) là hàm được sử dụng nhiều nhất trong các hệ thống Học Sâu hiện nay. Hàm này có công thức và đồ thị như sau:



Hình 2.18: Công thức và đồ thị hàm ReLU

Vì hàm này là hàm tuyến tính cho tất cả giá trị dương và trả về 0 cho tất cả những giá trị âm nên điều này cho phép máy tính tính toán một cách dễ dàng. Do

đó sẽ mất ít thời gian để đào tạo một hệ thống hơn. Ngoài ra, hàm này hội tụ nhanh hơn, vì nó là hàm tuyến tính nên độ dốc của hàm vẫn giữ ngay khi cả x trở nên lớn hơn. Nhờ đó vấn đề trọng số bị lãng quên được giải quyết. [12]

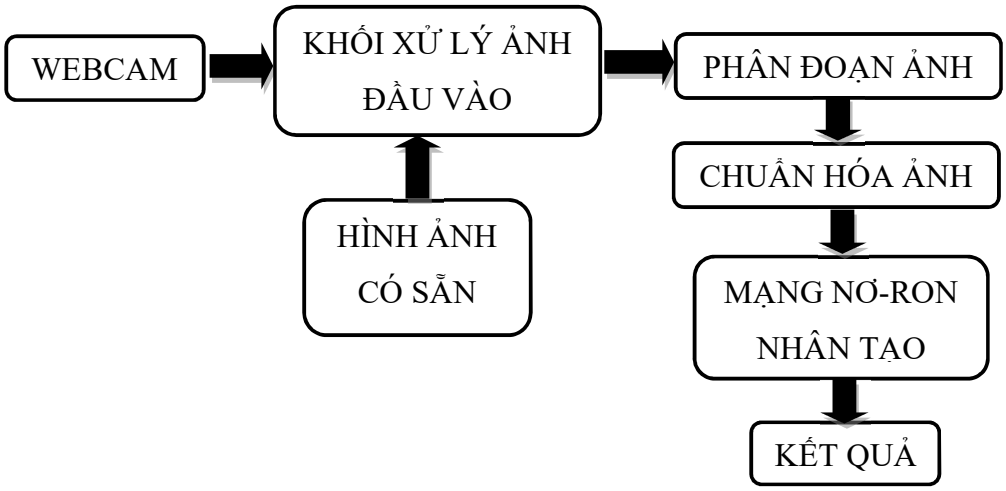
CHƯƠNG 3

THIẾT KẾ HỆ THỐNG

3.1 YÊU CẦU HỆ THỐNG

- Sử dụng webcam được kết nối bằng cáp với máy tính để chụp hình những ký tự cần được nhận diện.
- Khối tiền xử lý phải tách đầy đủ từng ký tự có trong hình theo thứ tự từ trái sang phải, sau đó từng ảnh phải được đưa về kích thước chuẩn 28x28 pixel để phù hợp với đầu vào của mạng nơ-ron.
- Mạng nơ-ron phải có độ chính xác trên 85%.

3.2 SƠ ĐỒ KHỐI HỆ THỐNG



Hình 3.1: Sơ đồ khối hệ thống

3.2.1 Ngõ vào

Gồm hình ảnh từ webcam hoặc hình ảnh có sẵn trong máy tính.

Webcam mà đề tài sử dụng được kết nối với máy tính để chụp ảnh chứa các ký tự. Trên thị trường hiện nay có hai loại webcam phổ biến đó là Webcam có

dây và Webcam không dây, một số dòng webcam có chất lượng tốt được nhiều ngươi ưa chuộng như là Logitech C270HD, Logitech B525 HD, Microsoft Lifecam Cinema, Microsoft Lifecam Studio, Genius Facecam 1000x, …

Hình 3.2: Webcam sử dụng trong đề tài

Để tăng tiến độ cho quá trình xử lý ảnh thì ảnh đầu vào cũng cần một số điều kiện dựa trên tính chất của chữ cái cần xác định. Chúng ta cần dựa vào tính chất màu của ký tự (màu đen), nền chứa ký tự là màu trắng. Việc ảnh chứa một số loại thành phần màu khác, chúng ta có thể loại bỏ bằng cách xử lý ảnh xám và chuyển ảnh về ảnh trắng đen.

Nhóm đã tiến hành sử dụng một chân máy ảnh để cố định webcam như Hình 3.3 nhằm giảm thiểu độ rung khi lấy ảnh đầu vào.

Hình 3.3: Webcam được cố định bằng chân máy ảnh

Ảnh đầu vào là ảnh mang ký tự đen nền trắng, việc giảm thiểu nhiễu là điều rất cần trong xử lý ảnh, nên nhóm cũng sử dụng một chân máy ảnh khác để cố định ảnh đầu vào như hình 3.4.

Hình 3.4: Ảnh đầu vào được cố định bằng chân máy ảnh

Khoảng cách giữa camera và ký tự cần nhận dạng từ 15cm đến 60cm, ký tự cần nhận dạng có chiều cao từ 1cm đến 15cm, chiều rộng chiếm 2/3 chiều dài ký tự, độ dày tối của ký tự là từ khoảng 0.1cm đến 0.5cm.

Đối với các ký tự có chiều cao và chiều rộng dưới 1cm thì chỉ thích hợp để nhận diện với khoảng cách dưới 15cm từ webcam đến ảnh văn bản. Tại khoảng cách 15cm ký tự trên ảnh văn bản có thể nhận dạng là 1 ký tự với chiều cao 4-7cm và chiều rộng từ 3-5cm được minh họa như hình bên dưới:

Hình 3.5: Ảnh văn bản tại khoảng cách 15cm

Đối với các ký tự có chiều cao và chiều rộng trên 1cm thì chỉ thích hợp để nhận diện với khoảng cách từ 15-60cm từ webcam đến ảnh văn bản (số lượng kí tự cũng phụ thuộc vào việc thay đổi khoảng cách từ webcam đến ảnh văn bản). Tại khoảng cách 60cm ký tự trên ảnh văn bản có thể nhận dạng tối đa là 6 ký tự với chiều cao 4-7cm và chiều rộng từ 3-5cm được minh họa như hình bên dưới:

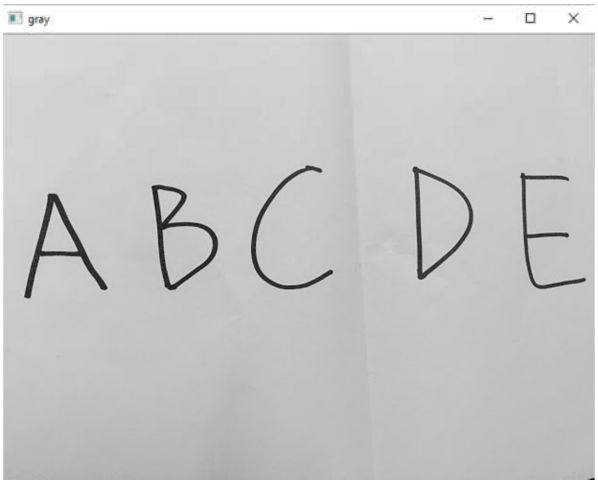
Hình 3.6: Ảnh văn bản tại khoảng cách 60cm

3.2.2 Tiền xử lý

Phần tiền xử lý bao gồm 3 khối: Khối xử lý ảnh đầu vào, khối phân đoạn ảnh và khối chuẩn hóa ảnh.

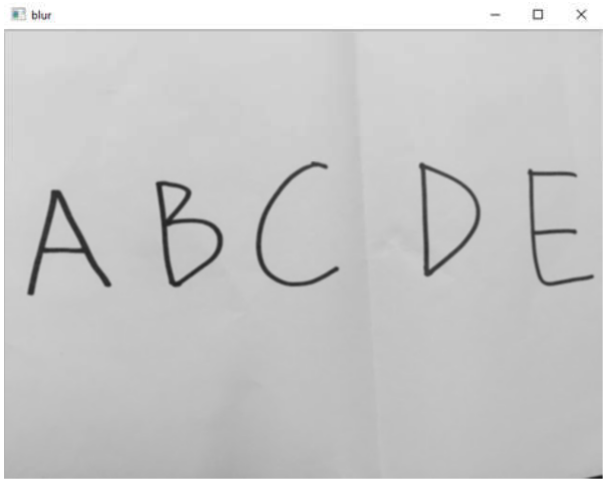
Phần này có chức năng xử lý đối với ảnh đầu vào thô, ảnh đầu vào có kích thước khá lớn sẽ ảnh hưởng đến quá trình huấn luyện và kiểm tra nên nhóm quyết định chuẩn hóa kích thước để phù hợp cho quá trình nhận dạng. Quá trình xử lý ảnh diễn ra như sau:

Ảnh sau khi đọc vào hệ thống tiến hành chuyển ảnh màu BGR sang trường màu xám bằng hàm cv2.cvtColor().



Hình 3.7: Ảnh sau khi chuyển xám

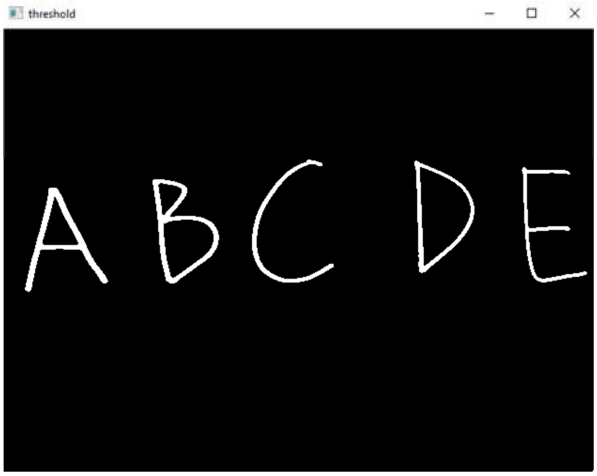
Điều này sẽ dễ dàng hơn cho việc lọc nhiễu Gaussian bằng hàm thông dụng cv2.GaussianBlur() như hình bên dưới:



Hình 3.8: Ảnh sau khi qua bộ lọc Gaussian

Ảnh sau khi chuyển xám và lọc nhiễu thì cần xác được ngưỡng mức xám để chuyển đổi sang giá trị nhị phân. Vì ký tự được nhận diện là màu đen nên sẽ có mức xám thấp. Để chuyển sang ảnh nhị phân, hàm cv2.threshold được dùng để lấy ngưỡng ảnh, nếu giá trị điểm ảnh cao hơn giá trị ngưỡng thì nó được gán giá trị 0 (màu trắng), ngược lại giá trị điểm ảnh nhỏ hơn giá trị ngưỡng thì nó được gán giá trị 1 (màu đen). Ở hàm cv2.threshold, hình ảnh sau khi được làm mờ sẽ được áp dụng threshold = 130, các pixel có cường độ lớn hơn 130 đều được đặt bằng 255, ngược lại thì được đặt bằng 0. Ngoài ra, ảnh có thể được inverse

thresholding với hàm cv2.THRESH_BINARY_INV (hình Threshold Binary Inverse).

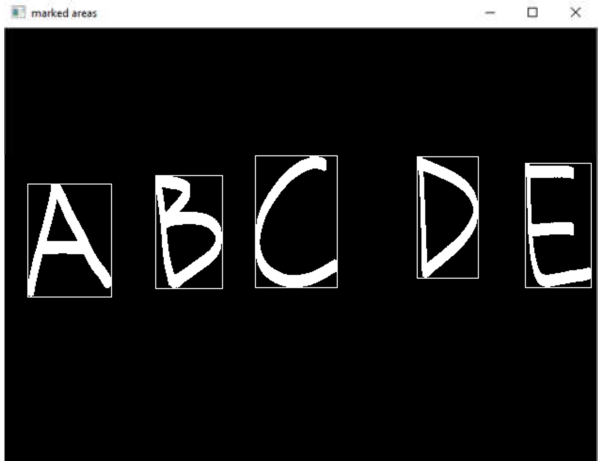


Hình 3.9: Ảnh sau khi chuyển nhị phân

Tiếp đến hệ thống sẽ sử dụng hàm cv2.dilate để thêm pixel vào biên của ký tự, với bộ lọc là ma trận 5x1 duyệt qua tất cả các pixel của ảnh nhị phân để tính toán.

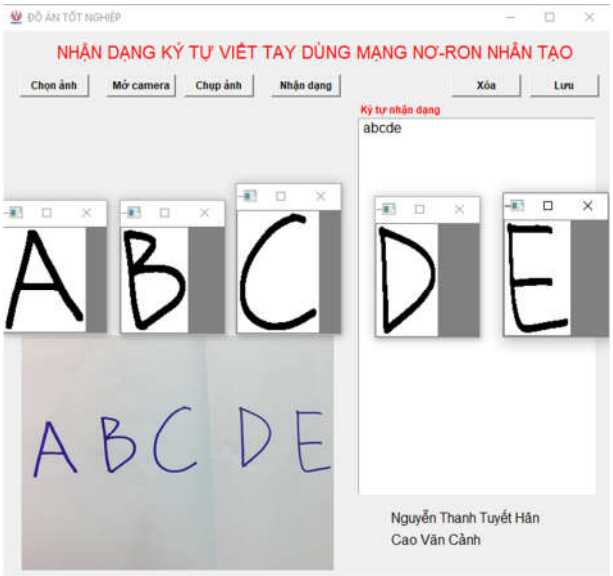
Hình 3.10: Ảnh sau khi giãn nở

Bước tiếp theo hệ thống sử dụng hàm cv2.findContour để tìm tất cả các đường biên của ký tự, sau đó sử dụng hàm sorted để sắp xếp các đường biên từ trái sang phải rồi khôi phục lại các chi tiết của ký tự bằng cách dùng hàm tạo hình chữ nhật cv2.rectangle theo đường bao (contour) đã được tìm thấy trước đó.



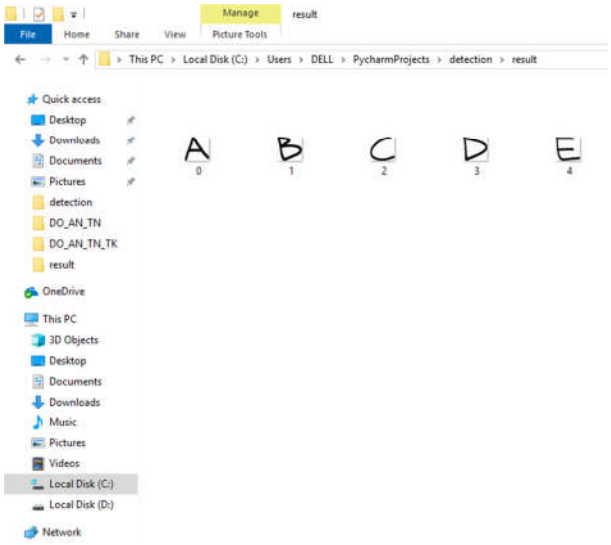
Hình 3.11: Ký tự sau khi được tạo khung

Sau khi tạo hình chữ nhật theo đường bao hệ thống tiến hành tách ký tự khỏi ảnh văn bản, ở đây sẽ dùng hàm RoI để tách từng ký tự riêng biệt. Ký tự sau khi tách khỏi ảnh văn bản sẽ được nhị phân hóa lần 2 nhằm đảo ngược chữ trắng nền đen sang chữ đen nền trắng với ngưỡng được chọn là 127 thông qua hàm cv2.THRESH_BINARY_INV được biểu diễn như hình bên dưới:



Hình 3.12: Ký tự được tách từ ảnh văn bản

Để thuận lợi cho mạng nơ-ron nhận diện chính xác ký tự, nhóm tiến hành chuẩn hóa kích thước từng ký tự về ảnh có chiều rộng và chiều cao là 28pixel, việc chuẩn hóa thông qua hàm cv2.resize(), ảnh sau chuẩn hóa sẽ được lưu đến một thư mục chọn trước để thuận tiện cho mạng nơ-ron trích xuất hình ảnh.



Hình 3.13: Ký tự sau khi chuẩn hóa kích thước

3.2.3 Mạng nơ-ron nhân tạo

Trong lĩnh vực học máy có các phương pháp học sau:

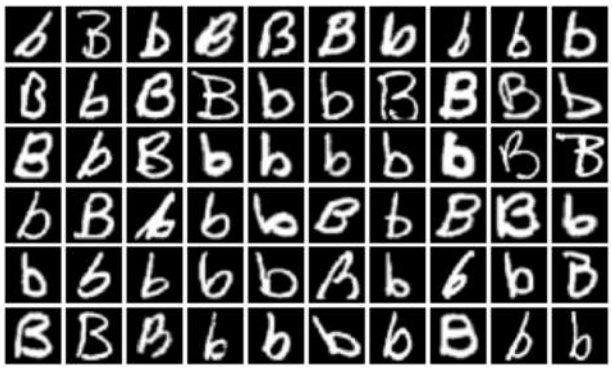
- Học có giám sát (supervised learning).
- Học không có giám sát (unsupervied learning).
- Học bán giám sát (semi-supervised learning).
- Học tăng cường (reinforcement learning).

Học có giám sát dùng thường dùng cho những bài toán phân lớp. Học không có giám sát thường áp dụng cho những bài toán phân cụm. Học bán giám sát là sự pha trộn của hai phương pháp trên, thường được áp dụng khi có dữ liệu lớn nhưng không đồng nhất, lúc được dán nhãn lúc không. Học tăng cường dùng để áp dụng cho những bài toán giúp cho một hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được lợi ích cao nhất, chủ yếu được áp dụng vào Lý Thuyết Trò Chơi.[13]

Vì mục đích của đề tài là nhận diện ký tự viết tay nên học có giám sát là phù hợp nhất. Với học có giám sát thì một tập dữ liệu chữ viết tay có phân loại và dán nhãn sẵn là không thể thiếu. EMNIST là một cơ sở dữ liệu lớn bao gồm các chữ viết tay và thường được sử dụng để đào tạo các hệ thống xử lý hình ảnh. Bộ cơ sở dữ liệu này có nguồn gốc từ NIST Special Database19 và được chuyển những

bức ảnh có kích cỡ là 28x28 pixel. Có sáu bộ dữ liệu khác nhau trong tập dữ liệu này:

- EMNIST ByClass: gồm 814,255 hình, được chia thành 62 tập khác nhau, số lượng hình mỗi tập là không bằng nhau.
- EMNIST ByMerge: gồm 814,255 hình, được chia thành 47 tập khác nhau, số lượng hình mỗi tập là không bằng nhau.
- EMNIST Balanced: gồm 131,600 hình, được chia thành 47 tập khác nhau, số lượng hình mỗi tập là bằng nhau.
- EMNIST Letters: gồm 145,600 hình, được chia thành 26 tập khác nhau, số lượng hình mỗi tập là bằng nhau.
- EMNIST Digits: gồm 280,000 hình, được chia thành 10 tập khác nhau, số lượng hình mỗi tập là bằng nhau.
- EMNIST MNIST: gồm 70,000 hình, được chia thành 10 tập khác nhau, số lượng hình mỗi tập là bằng nhau.



Hình 3.14: Tập dữ liệu EMNIST

Vì đề tài là nhận dạng chữ viết tay nên tập dữ liệu EMNIST Letters là phù hợp nhất.

Thiết kế mạng nơ-ron truyền thống:

Bước đầu tiên là xác định số lượng nơ-ron mỗi tầng cũng như là số lượng tầng ẩn cần thiết. Theo tập dữ liệu EMNIST, mỗi hình đều có kích cỡ là 28x28 pixel do đó số lượng nơ-ron của tầng vào bắt buộc là 28 x 28 = 784 nơ-ron. Tập dữ liệu EMNIST Letters được gắn nhãn từ 1 đến 26 tượng trưng cho ký tự từ A đến Z trong bảng chữ cái. Tuy nhiên trong lập trình, mảng luôn được đánh số từ

0 nên số lượng nơ-ron đầu ra phải là 27 (mảng được đánh số từ 0 đến 26, nơ-ron số 0 là nơ-ron dư).

Số lượng Tầng ẩn thường được dựa vào độ phức tạp của bài toán cần nhận dạng. Với một bài toán ánh xạ liên tục từ một không gian hữu hạn này sang không gian hữu hạn khác thì chỉ cần 1 tầng ẩn là đủ. Đối với bài toán nhận dạng chữ viết thì từ 2 đến 3 Tầng ẩn là phù hợp nhất vì khi số lượng tầng ẩn tăng cao thì số lượng nơ-ron cũng tăng cao. Khi số lượng nơ-ron quá cao, hệ thống sẽ có khả năng ghi nhớ nhiều hơn là học hỏi, việc này gây ra vấn đề over-fitting (trọng số quá khớp). Việc này có thể nhận biết khi có sự khác biệt cực lớp giữa độ chính xác trong quá trình huấn luyện và độ chính xác trong quá trình kiểm tra.[14]

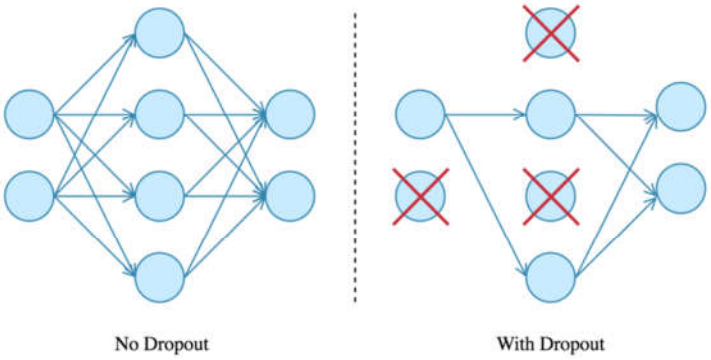
Số lượng nơ-ron mỗi tầng, cũng phải được chọn một cách cẩn thận bởi vì việc này có thể dễ dàng gây ra vấn đề trọng số quá khớp và một số vấn đề khác. Đầu tiên, khi số lượng nơ-ron quá nhiều, dòi hỏi dữ liệu để huấn luyện cũng phải lớn để huấn luyện hết tất cả nơ-ron trong hệ thống. Ngoài ra, việc này cũng làm thời gian cần thiết để huấn luyện tăng lên đáng kể. Tuy nhiên khi sử dụng quá ít nơ-ron thì hệ thống thường sẽ không đủ khả năng để phát hiện đầy đủ những điểm quan trọng của một dữ liệu phức tạp, kết quả là độ chính xác của hệ thống sẽ không cao. [15]

Dưới đây là một số quy tắc để xác định số lượng nơ-ron cần thiết trong Tầng ẩn:

- Số lượng nơ-ron thường nằm giữa kích thước của lớp đầu vào và kích thước lớp đầu ra.
- Số lượng nơ-ron thường bằng 2/3 kích thước lớp đầu vào cộng kích thước lớp đầu ra.
- Số lượng tế bào thần kinh phải nhỏ hơn hai lần kích thước đầu vào.

Từ đây có thể xác định số lượng nơ-ron trong tầng ẩn nằm giữa 50 và 700. Tuy nhiên, để hệ thống tránh được vấn đề trọng số quá khớp, kỹ thuật Dropout thường được áp dụng trong quá trình huấn luyện. Kỹ thuật này sẽ bỏ qua một số nơ-ron một cách ngẫu nhiên trong một chu kì huấn luyện (còn gọi là epochs). Qua quá trình thử nghiệm, số chu kì huấn luyện đối với bài toán này là từ 10 đến

20 chu kì vì sau đó hệ thống không có xu hướng tốt hơn mà vấn đề trọng số quá khớp bắt đầu xảy ra.[15]



Hình 3.15: Kỹ thuật Dropout

Dưới đây là một số kết quả của quá trình huấn luyện mà nhóm đã thực hiện:

Bảng 3.1: Kết quả một số mạng nơ-ron truyền thống

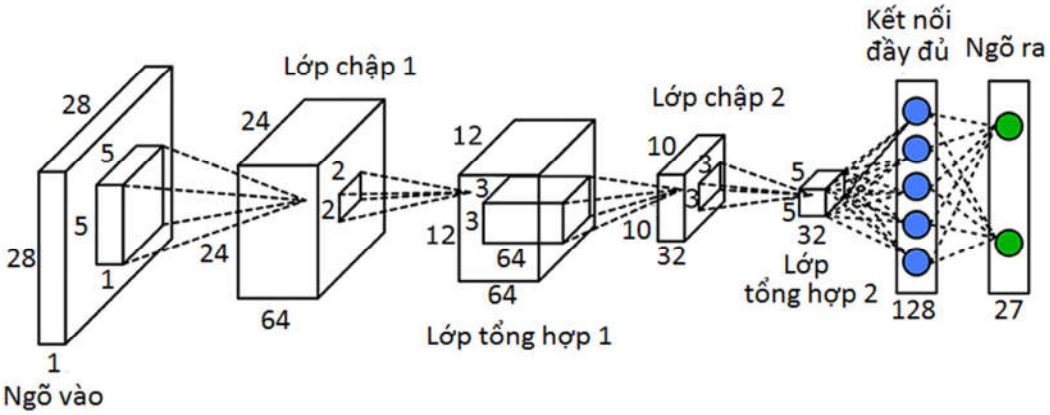
Số Tầng Ẩn	Số lượng nơ-ron mỗi tầng	Tỉ lệ Dropout	Epochs	Độ chính xác
3	600	0.4	50	87,6%
3	200	0.25	20	84.8%
3	300	0.2	15	86,3%
3	230	0.2	15	86,3%
3	300	0.35	10	84,4%
3	300	0.5	10	82,1%
2	512	0.2	10	88,1%
2	300	0.5	10	82,8%

Thiết kế mạng nơ-ron tích chập:

Đầu tiên, cần phải xác định kích cỡ của bộ lọc. Vì hình ảnh đầu vào có kích thước khá nhỏ, bộ lọc có kích thước 3x3 hoặc 5x5 sẽ hợp lí hơn vì bộ lọc kích thước lớn hơn (9x9 hoặc 11x11) thường được sử dụng cho những ảnh có số lượng pixel khá lớn. Trong đồ án này, nhóm quyết định kích cỡ bộ lọc sẽ lần lượt là 5x5 ở lớp nơ-ron tích chập thứ nhất và 3x3 ở lớp nơ-ron tích chập thứ hai.

Số lượng bộ lọc trong lớp tích chập thường được sử dụng nhất là từ 32 đến 1024 bộ lọc. Tuy nhiên, vì hình ảnh đầu vào không quá lớn nên nhóm sẽ sử dụng 32 hoặc 64 bộ lọc ở hai lớp tích chập.[16] [17] [18]

Mô hình mạng nơ-ron đề tài sử dụng được thể hiện qua hình và bảng sau đây:



Hình 3.16: Cấu trúc mạng nơ-ron

Bảng 3.2: Cấu trúc mạng nơ-ron tích chập

Lớp	Thông số	Kích thước ngõ ra
Ngõ vào	Không	28x28x1
Tích chập 1	64 bộ lọc có kích cỡ [5,5]	24x24x64
Tổng hợp 1	Kích cỡ [2x2]	12x12x64
Tích chập 2	32 bộ lọc có kích cỡ [3,3]	10x10x32
Tổng hợp 2	Kích cỡ [2x2]	5x5x32
Dropout	20% Dropout	
Flatten	Không	800x1
Kết nối đầy đủ 1	128 nơ-ron, 20% dropout	128x1
Ngõ ra	Hàm kích hoạt là softmax	27x1

Tổng số lượng tham số là 126139 nơ-ron, trong đó 1662 nơ-ron thuộc lớp Tích chập 1, 18464 nơ-ron thuộc lớp Tích chập 2, 102528 nơ-ron thuộc lớp Kết nối đầy đủ và 3483 nơ-ron thược lớp Ngõ ra.

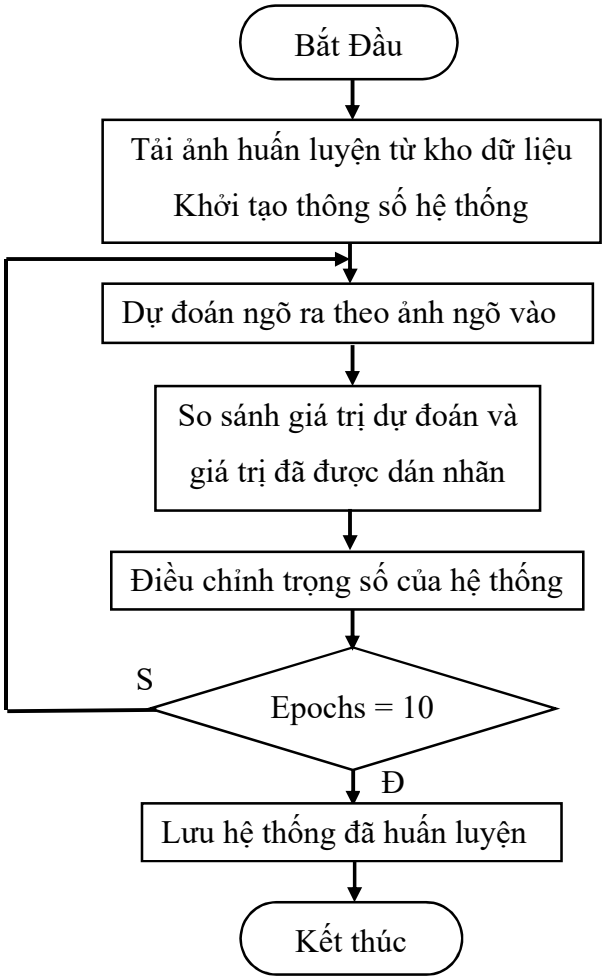
3.2.4 Kết quả

Kết quả của quá trình huấn luyện sẽ là một mạng nơ-ron có thông số được điều chỉnh một cách tự động bởi máy tính trong quá trình huấn luyện. Mạng nơ-ron này sẽ được lưu trong máy tính với đuôi ".model".

Kết quả của quá trình nhận diện chữ viết sẽ là một trong 26 chữ cái từ “a” đến “z”. Chữ cái này sẽ được hiển thị trên Textbox của giao diện phần mềm. Kết quả sẽ được sắp xếp đúng thứ tự từ trái qua phải như trên ảnh đầu vào.

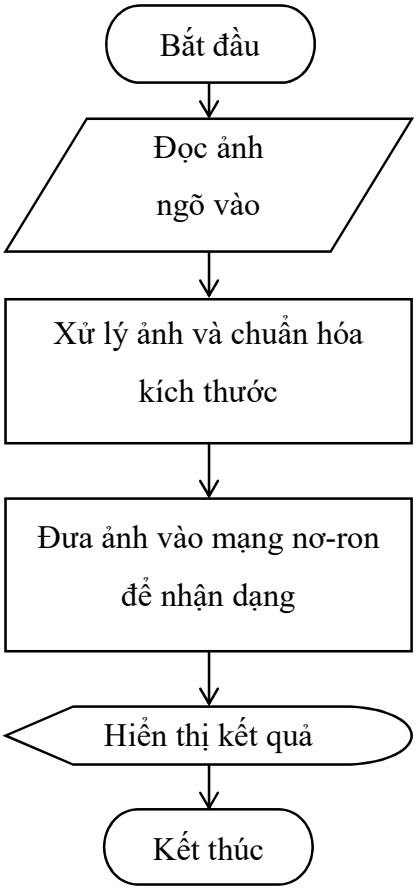
3.3 LƯU ĐỒ HỆ THỐNG

3.3.1 Lưu đồ huấn luyện



Hình 3.17: Lưu đồ quá trình huấn luyện

3.3.2 Lưu đồ chương trình



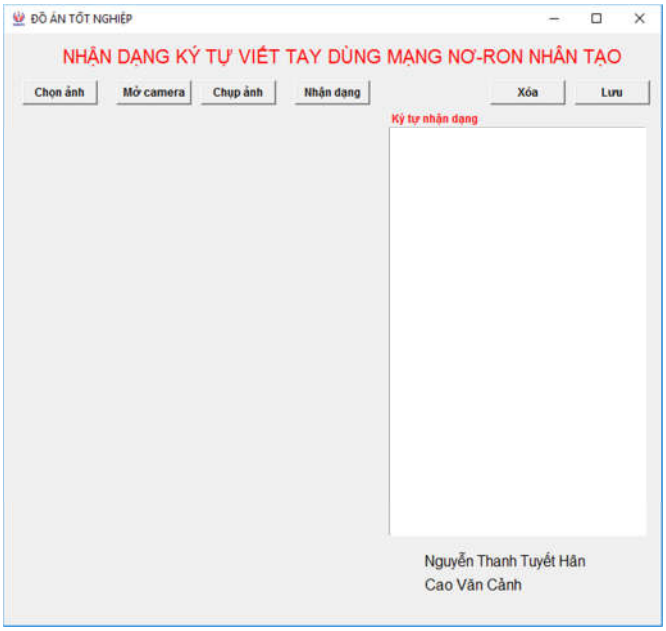
Hình 3.18: Lưu đồ quá trình nhận diện

CHƯƠNG 4

KẾT QUẢ

4.1 GIAO DIỆN CHƯƠNG TRÌNH

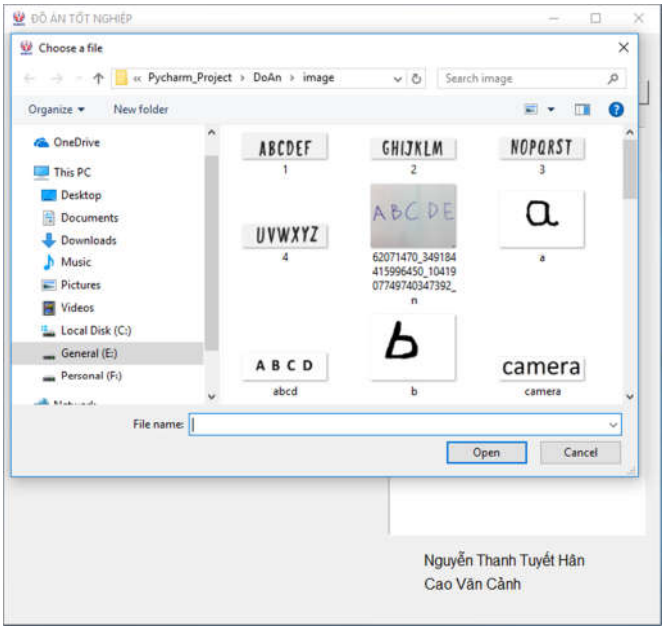
Chương trình nhận dạng ký tự viết tay sử dụng mạng nơ-ron nhân tạo có chức năng là nhận dạng qua hình ảnh tĩnh được người dùng chọn từ thư mục có sẵn hoặc từ webcam.



Hình 4.1: Giao diện chương trình

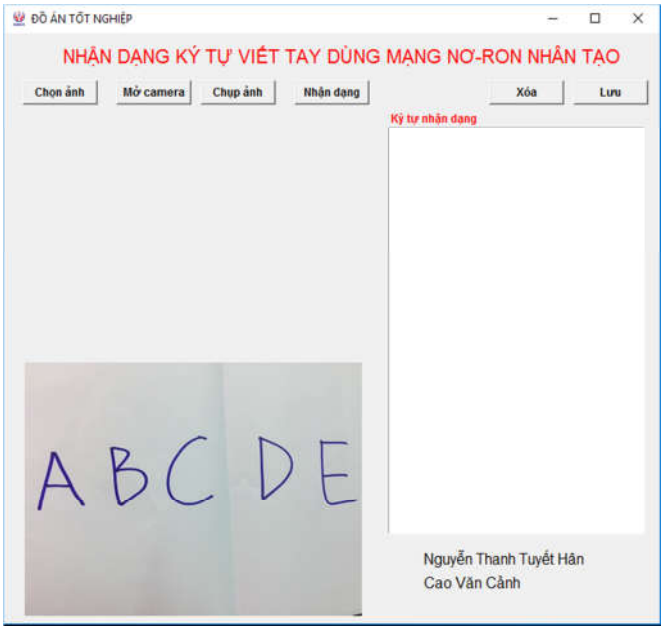
4.2 HOẠT ĐỘNG CỦA HỆ THỐNG

Người sử dụng nhấn vào "Chọn ảnh" để nhập ảnh đầu vào. Chương trình sẽ chuyển đến một đường dẫn với thư mục ngẫu nhiên chứa ảnh được chọn từ người dùng.



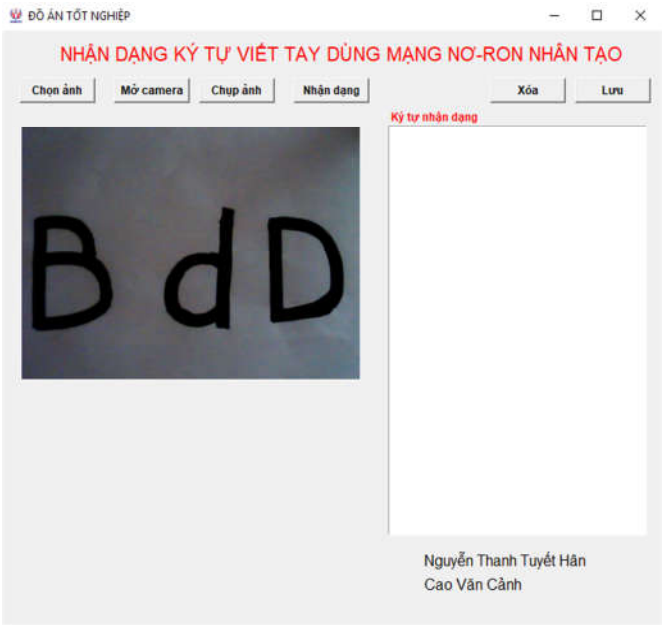
Hình 4.2: Giao diện sau khi nhấn chọn ảnh

Sau đó ảnh mà người dùng đã chọn sẽ hiển thị trên Label với vị trí được thiết kế sẵn như hình sau:



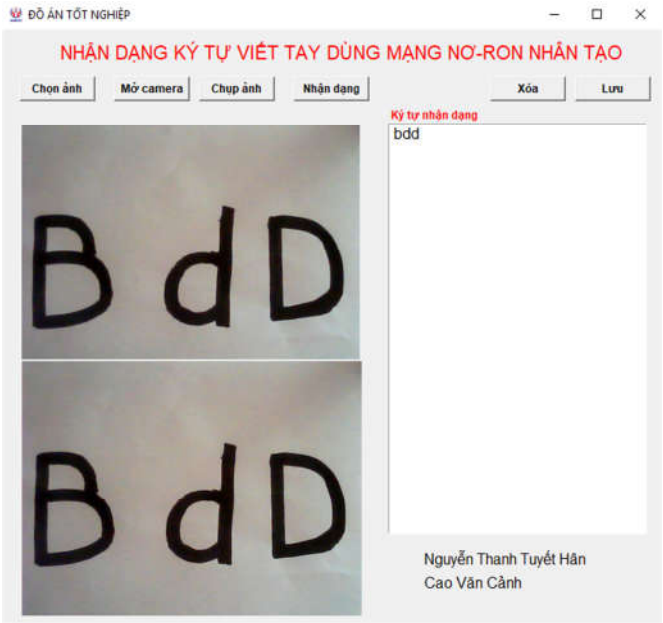
Hình 4.3: Giao diện sau khi chọn ảnh

Nếu người dùng muốn chọn ảnh trực tiếp từ camera thì có thể nhấn nút "Mở camera". Khung camera sẽ xuất hiện lên cửa sổ như hình dưới:



Hình 4.4: Giao diện chương trình khi mở camera

Sau đó người dùng có thể nhấn chụp ảnh để lưu lại ảnh chứa ký tự, ảnh này sau đó sẽ được hiển thị vào Label với vị trí đã được thiết lập sẵn như hình bên dưới:

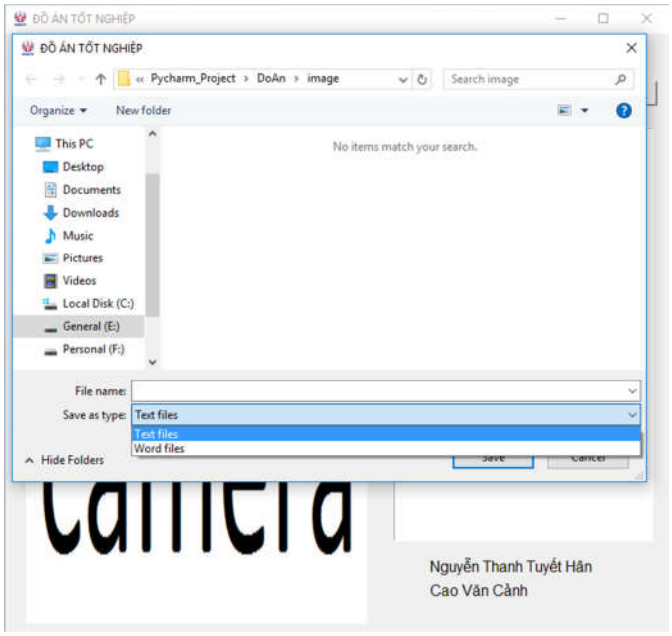


Hình 4.5: Giao diện chương trình sau khi chụp ảnh và nhận dạng

Dù với ảnh đầu vào được chọn từ webcam hay từ file có sẵn, người dùng khi nhấn "Nhận dạng" thì ký tự có trong ảnh sẽ được nhận dạng và hiển thị trên Textbox với vị trí được thiết lập sẵn như hình bên dưới:

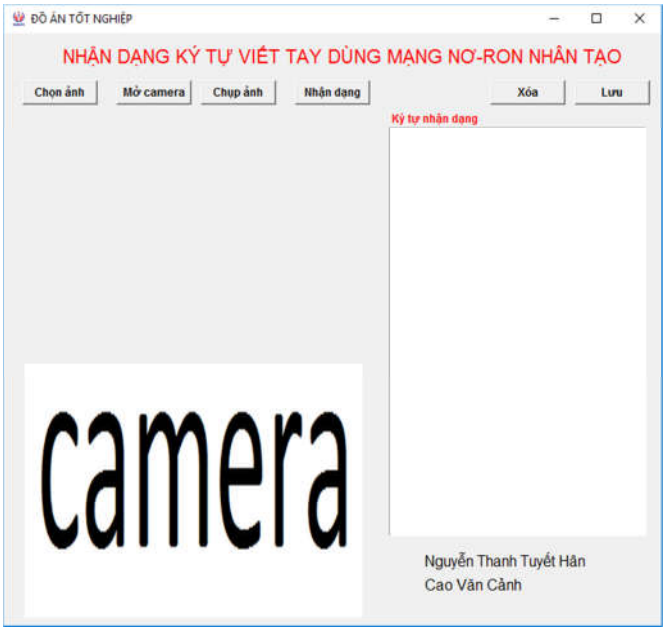
Hình 4.6: Giao diện chương trình sau khi nhận dạng

Người dùng có thể nhấn "Lưu" để lưu lại kết quả nhận dạng với định dạng được hỗ trợ như ".txt" hay ".doc".



Hình 4.7: Giao diện cửa sổ khi nhấn lưu

Mặt khác, người dùng có thể xóa đi kết quả vừa nhận dạng khi không muốn lưu kết quả có thể nhấn "Xóa". Hệ thống sẽ xóa tất cả ký tự có trong textbox khi người dùng nhấn nút này. Ngoài ra, người dùng có thể xóa từ ký tự riêng lẻ bằng bàn phím có sẵn.



Hình 4.8: Giao diện chương trình khi nhấn xóa

4.3 KẾT QUẢ HOẠT ĐỘNG

Sau đây là kết quả của quá trình huấn luyện, hai bảng 4.1 và 4.2 dưới đây thể hiện ma trận tương quan của hệ thống cũng như số lượng hình nhận dạng chính xác trên 100 mẫu của mỗi ký tự.

Bảng 4.1: Kết quả nhận dạng chữ cái từ A đến M

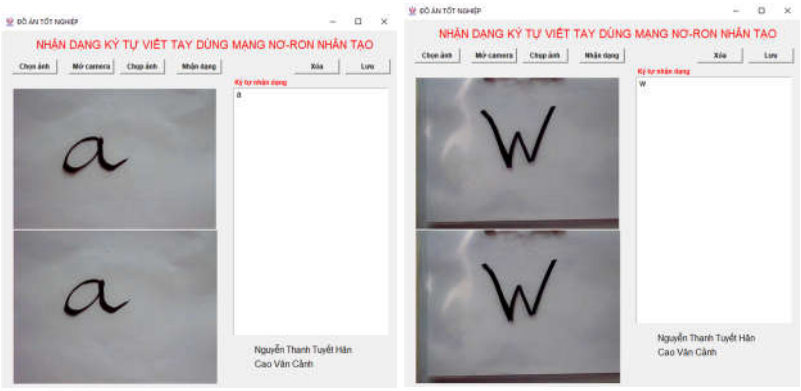
Kết Quả	Chữ Cái												
	A	B	C	D	E	F	G	H	I	J	K	L	M
A	**0.92**	0.01	0	0	0	0	0.02	0	0	0	0	0	0
B	0	**0.91**	0	0	0	0	0	0.01	0	0	0	0	0
C	0	0	**0.95**	0	0.03	0	0.01	0	0	0	0	0	0
D	0.01	0.02	0	**0.92**	0	0	0	0	0.01	0.01	0	0	0
E	0	0.01	0.03	0	**0.95**	0	0	0	0	0	0	0	0
F	0	0	0	0	0	**0.92**	0	0	0	0	0	0	0
G	0.01	0.01	0	0	0	0.01	**0.79**	0	0.01	0.01	0	0	0
H	0	0.01	0	0	0	0	0	**0.95**	0	0	0	0	0.01
I	0	0	0	0	0	0	0	0	**0.69**	0.03	0	0.37	0
J	0	0	0	0	0	0	0	0	0.02	**0.93**	0	0	0
K	0	0.01	0	0	0	0	0	0.01	0	0	**0.96**	0	0
L	0	0.01	0	0	0	0	0	0.01	0.26	0	0	**0.62**	0
M	0	0	0	0	0	0	0	0	0	0	0	0	**0.96**
N	0	0	0	0	0	0	0	0.01	0	0	0	0	0.02
O	0	0	0	0.04	0	0	0	0	0	0	0	0	0
P	0	0	0	0.02	0	0.01	0	0	0	0	0	0	0
Q	0.04	0	0	0	0	0	0.15	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0.01	0	0	0.01	0	0	0
T	0	0	0	0	0	0.04	0	0	0	0	0	0	0
U	0.01	0	0.01	0	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0.01
X	0	0	0	0	0	0	0	0.01	0	0	0.02	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0
Tổng	0.99	0.99	0.99	0.98	0.98	0.98	0.98	1	0.99	0.99	0.98	0.99	1
Số mẫu	100	100	100	100	100	100	100	100	100	100	100	100	100
Số hình đúng	97	93	97	93	97	95	82	97	85	95	98	65	97
T.gian (ms)	6.1	5.3	4.5	4.3	4.3	4.3	4.4	4.4	4.4	4.3	4.4	4.2	4.2

Bảng 4.2: Kết quả nhận dạng chữ cái từ N đến Z

Kết Quả	Chữ Cái												
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	0	0	0	0.05	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0.05	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0.01	0	0	0	0	0	0	0	0	0
F	0	0	0	0.01	0	0	0	0	0	0	0	0	0
G	0	0.01	0	0.15	0	0.01	0	0	0	0	0	0	0
H	0.02	0	0	0	0	0	0	0	0	0.01	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0.01	0.01	0	0.01	0	0	0.01	0
K	0	0	0	0	0	0	0	0.01	0	0	0.02	0	0
L	0	0	0	0	0	0	0	0	0	0	0.01	0	0
M	0.01	0	0	0	0	0	0	0	0	0	0	0	0
N	**0.94**	0	0	0	0.01	0	0	0	0	0.02	0	0	0
O	0	**0.89**	0	0	0	0	0	0	0	0	0	0	0
P	0	0	**0.98**	0.01	0	0	0	0	0	0	0	0	0
Q	0	0.01	0.01	**0.75**	0	0	0	0	0	0	0	0.01	0
R	0	0	0	0	**0.94**	0	0	0	0.01	0	0	0	0
S	0	0	0	0	0	**0.97**	0	0	0	0	0	0	0
T	0	0	0	0	0	0	**0.97**	0	0	0	0	0	0
U	0.01	0.02	0	0	0	0	0	**0.93**	0.01	0.01	0	0	0
V	0.01	0.01	0	0	0.01	0	0	0.05	**0.94**	0	0.01	0.03	0
W	0	0	0	0	0	0	0	0	0	**0.95**	0	0	0
X	0	0	0	0	0	0	0	0	0	0	**0.93**	0	0
Y	0	0	0	0	0	0	0	0	0.02	0	0.02	**0.94**	0
Z	0	0	0	0.01	0	0	0	0	0	0	0	0	**0.99**
Tổng	0.99	0.99	0.99	0.99	0.96	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99
Số mẫu	100	100	100	100	100	100	100	100	100	100	100	100	100
Số hình đúng	95	90	99	82	99	99	100	97	98	95	94	97	100
T.gian (ms)	4.2	4.2	4.3	4.4	4.4	5.6	4.5	4.5	4.4	4.3	4.3	4.2	4.3

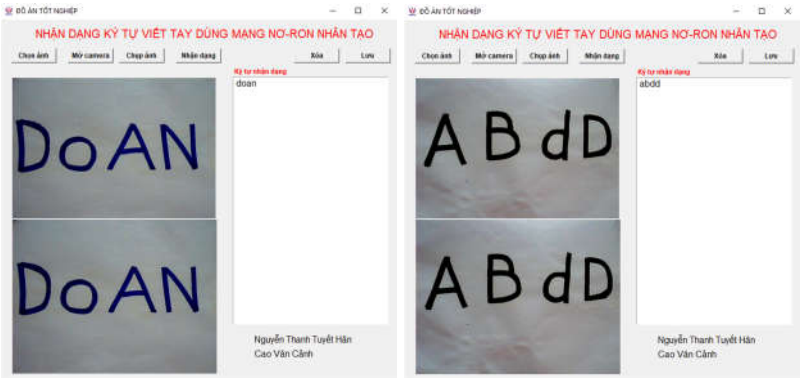
Từ hai bảng trên, có thể thấy hệ thống vẫn còn nhầm lẫm giữa hai chữ cái ‘I’ và ‘L’ vì trong 100 mẫu đầu vào có hình ảnh chữ ‘L’ thì chỉ nhận dạng đúng 65 hình còn 35 hình còn lại cho ra kết quả là chữ ‘I’.

Sau đây là kết quả của một số hình ảnh thực tế, mà hệ thống nhận dạng đúng.



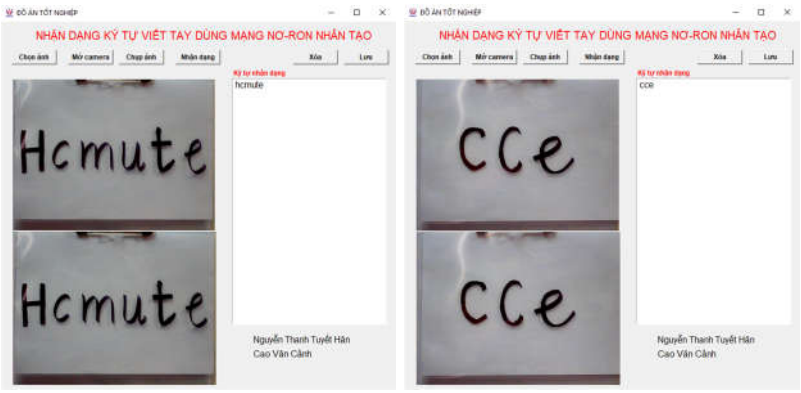
Hình 4.9: Một số hình nhận diện đúng

Nhận xét: Kết quả cho thấy hệ thống có khả năng nhận dạng tốt với các kí tự được viết đầy đủ các nét, độ dày kí tự không quá mỏng, nên việc phân loại nền trắng chữ đen trong quá trình xử lý ảnh diễn ra nhanh hơn.



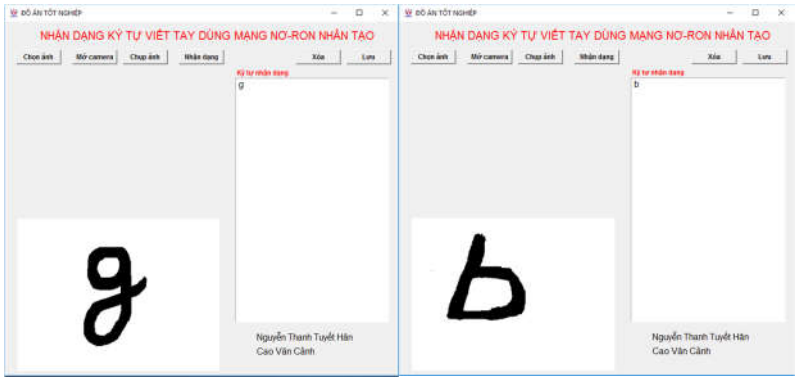
Hình 4.10: Một số hình nhận diện đúng

Nhận xét: Hệ thống sẽ cho kết quả nhận diện nhanh hơn đối với các trường hợp kí tự có nét viết đầy đủ, độ dày vừa phải và các kí tự không dính liền nhau, độ sáng ảnh văn bản không quá tối. Ngược lại, hệ thống sẽ nhận diện sai hoặc một vài trường hợp hệ thống không nhận diện được.



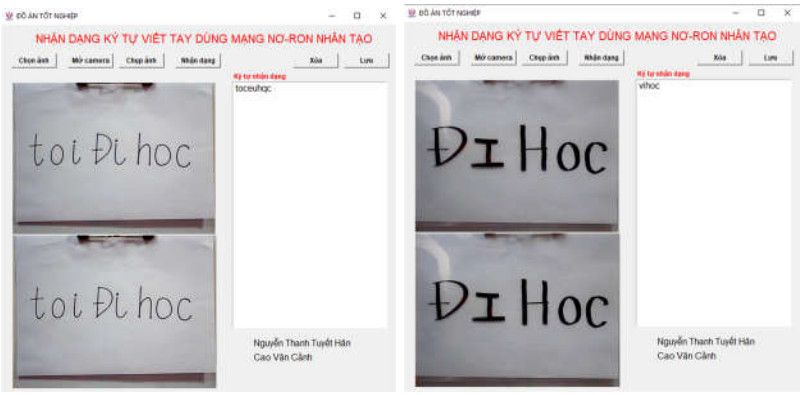
Hình 4.11: Một số hình nhận diện đúng

Nhận xét: Đối với trường hợp này, ảnh văn bản có chứa khung/viền không mong muốn thì việc này cũng không ảnh hưởng đến quá trình nhận diện, bởi vì quá trình tiền xử lý ảnh đã giải quyết vấn đề này. Các ký tự đáp ứng đầy đủ về khoảng cách, về độ dày cũng như nét viết cho thấy quá trình nhận diện sẽ diễn ra nhanh chóng.



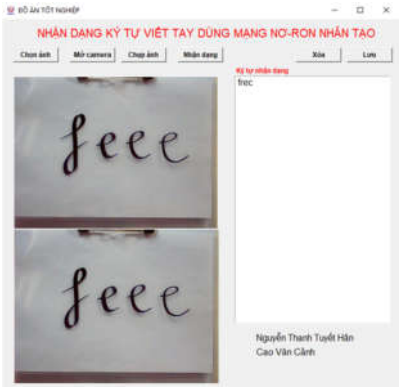
Hình 4.12: Một số hình nhận diện đúng

Nhận xét: Trong trường hợp này, ảnh văn bản đã đáp ứng đầy đủ về độ dày cũng như về định dạng nền trắng chữ đen. Tuy nhiên qua quá trình kiểm tra vấn đề nét viết được yêu cầu cao về trường hợp này nhằm đảm bảo độ chính xác đạt hiệu quả tốt nhất.



Hình 4.13: Một số hình nhận diện sai

Nhận xét: Dựa vào kết quả cho thấy, hệ thống có khả năng nhận dạng kém đối với ký tự có độ dày quá mỏng, trường hợp nét chữ viết cũng ảnh hưởng lớn đến nhận diện vì xác suất ký tự giống nhau xảy ra rất cao.



Hình 4.14: Một số hình nhận diện sai

Nhận xét: Trong trường hợp này ký tự f được viết với độ dày đầy đủ, nét viết cũng đảm bảo nên khả năng đúng rất cao, với ký tự e được chia thành ba trường hợp như minh họa trên cho thấy hệ thống phụ thuộc rất nhiều vào nét viết, độ cao các kí tự và độ dày cần thiết.

Như vậy qua quá trình kiểm tra và thực nghiệm cho thấy hệ thống có khả năng nhận dạng kém đối với những ký tự có độ dày dưới 0.2cm. Thời gian đáp ứng tùy thuộc vào số lượng ký tự trong hình, thời gian nhận dạng một ký tự sẽ từ 0.01 giây đến 0.015 giây. Ngoài ra, hệ thống khó nhận dạng được những ký tự dính liền nhau hoặc những ký tự quá nhỏ (có chiều cao nhỏ hơn 4cm và độ rộng nhỏ hơn 3cm).

CHƯƠNG 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 KẾT LUẬN

Đề tài đã xây dựng một mạng nơ-ron nhân tạo có khả năng nhận diện được chữ viết tay với độ chính xác tốt.

Đề tài đã thiết kế một giao diện chương trình hoàn chỉnh, thuận tiện cho việc sử dụng. Chương trình đã xây dựng cho kết quả nhận dạng với tỷ lệ cao đối với các ảnh đầu vào chụp đủ cường độ sáng, vị trí, góc chụp thích hợp. Giao diện giao tiếp với người dùng được tối ưu hóa, rất đơn giản.

Bên cạnh đó, trong quá trình nhận diện ảnh văn bản cũng còn một số hạn chế như là: ký tự trong ảnh văn bản có độ dày nằm trong khoảng 0.2cm-0.5cm, khi hai ký tự viết liền nhau chương trình sẽ không nhận diện được. Ngoài ra, hệ thống không thể nhận diện được khoảng trắng giữa các từ cũng như số đếm. Bên cạnh đó, ký tự được nhận diện trên một ảnh văn bản còn giới hạn về số lượng.

5.2 HƯỚNG PHÁT TRIỂN

Do thời gian có hạn, đề tài được thiết kế còn mang tính nghiên cứu, nên để triển khai trong đề tài trong thực tế cần phải phát triển thêm các nội dung sau:

- Cải thiện tập mẫu để hệ thống đạt được độ chính xác cao hơn, nhận dạng đúng trong nhiều trường hợp và ở nhiều điều kiện khác nhau.
- Kết hợp thêm các phần cứng hỗ trợ, hệ thống chiếu sáng để tăng cường khả năng nhận dạng trong điều kiện thiếu sáng ,…
- Phát triển hệ thống không phụ thuộc nhiều vào chất lượng camera.
- Nhận diện sự khác biệt giữa chữ in hoa và chữ viết thường.
- Nhận diện được khoảng trắng giữa các ký tự và nhận dạng được số đếm.
- Có khả năng nhận diện hình ảnh có nhiều dòng ký tự.

TÀI LIỆU THAM KHẢO

[1] Ngô Thị Nhung, "*Xử Lý Ảnh Trong Matlab*", Báo cáo thực tập, Trường ĐHCN Hà Nội.

[2] PGS.TS Nguyễn Quang Hoan, "*Xử Lý Ảnh*", NXB Hà Nội, Học Viện Công Nghệ Bưu Chính Viễn Thông 2006.

[3] PGS.TS Nguyễn Thanh Hải, "*Giáo trình Xử lí ảnh*", NXB Đại học Quốc gia TP.HCM, Trường Đại học Sư Phạm Kỹ Thuật TPHCM 09/2015.

[4] Bùi Ngọc Huy, "*Nghiên Cứu Kỹ Thuật Xử Lý Nhiễu, Hiệu Chỉnh Ảnh Nhị Phân và Ứng Dụng Cho Phiếu Thi Trắc Nghiệm*", Luận Văn Thạc Sĩ, Trường Đại Học Công Nghệ Thông Tin Và Truyền Thông Thái Nguyên, 2013.

[5] Michael Copeland, "*What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning*", July 29 2016.

[6] Vũ Hữu Tiệp, "*Machine Learning cơ bản*", Bài 35: Lược sử Deep Learning, 06/2018.

[7] Vũ Hữu Tiệp, "*Machine Learning cơ bản*", Bài 35: Perceptron Learning Algorithm, 01/2017.

[8] Đỗ Minh Hải, "*Mạng nơ-ron nhân tạo - Neural Networks*", < https://dominhhai.github.io/vi/2018/04/nn-intro/>

[9] Kurt Hornik (1991), "*Approximation Capabilities of Multilayer Feedforward Networks*", Neural Networks, 4(2), 251-257.

[10] Neural Smithing (1999), "Supervised Learning in Feedforward Artificial Neural Networks", Page 31.

[11] Alex Krizhevsky, Ilya S., Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", Advances in neural

information processing systems, 2012, pp. 1097 - 1105Christian Gerber, Mokdong Chung, "*Number Plate Detection with a MultiConvolutional Neural Network Approach with Optical Character Recognition for Mobile Devices*", J Inf Process Syst, Vol.12, No.1, pp.100~108, March 2016.

[12] Sagar Sharma, "*Activation Functions in Neural Networks*", About Towards Data Science, September 6 2017.

[13] Vũ Hữu Tiệp, "*Machine Learning cơ bản*", Bài 2: Phân nhóm các thuật toán Machine Learning, 12/2016.

[14] Nitish S., Geoffrey H., Alex Krizhevsky, Ilya Sutskever, Ruslan S., "*Dropout:A Simple Way to Prevent Neural Networks from Overfitting*", Journal of Machine Learning Research 15, 2014, pp. 1929 – 1958.

[15] Hinton, G.E. (2006),"A Fast Learning Algorithm for Deep Belief Nets" (PDF). Neural Computation.18(7):1527-1554.

[16] Fei-Fei Li, Justin Johnson, Serena Yeung, "*Lecture 9: CNN Architecture*", Stanford University, May 2, 2017.

[17] Arden Dertat Applied Deep Learning - Part 4: Convolutional Neural Networks",< https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>.

[18] Shashank Ramesh, " *A guide to an efficient way to build neural network architectures- Part II: Hyper-parameter selection and tuning for Convolutional Neural Networks using Hyperas on Fashion-MNIST*",<https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7>.

PHỤ LỤC

MÃ NGUỒN CHƯƠNG TRÌNH

1. CHƯƠNG TRÌNH HUẤN LUYỆN

```
import tensorflow as tf
from emnist import extract_training_samples
from emnist import extract_test_samples
x_train, y_train = extract_training_samples('letters')
x_test, y_test = extract_test_samples('letters')

x_train = x_train.reshape(124800, 28, 28, 1)
x_test = x_test.reshape(20800, 28, 28, 1)

x_train = tf.keras.utils.normalize(x_train, axis=1)
x_test = tf.keras.utils.normalize(x_test, axis=1)

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(32, kernel_size=5, activation='relu',
input_shape=(28, 28, 1)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Conv2D(64, kernel_size=3,
activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(27, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=10, batch_size=200,
validation_split=0.3)

val_loss, val_acc = model.evaluate(x_test, y_test)
print(val_loss)
print(val_acc)
model.save(Con6432_D02_Den128_D02_Den50.model ')
```

2. CHƯƠNG TRÌNH KIỂM TRA

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from emnist import extract_test_samples
from emnist import extract_training_samples
```

```
import cv2

CATEGORIES = ["none", 'a', 'b', 'c', 'd', 'e', 'f', 'g',
              'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
              'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

x_test, y_test = extract_test_samples('letters')
x_train, y_train = extract_training_samples('letters')
i = 1068
# for convolution model
x_train = tf.keras.utils.normalize(x_train, axis=1)
x_test = tf.keras.utils.normalize(x_test, axis=1)
x_test = x_test.reshape(20800, 28, 28, 1)
new_model =
tf.keras.models.load_model("Con6432_D02_Den128_D02_Den50.model")
#############################################################

# to show accuracy in test database
val_loss, val_acc = new_model.evaluate(x_test, y_test)
print(val_loss)
print(val_acc)
```

3. CHƯƠNG TRÌNH CHÍNH

```
import tkinter
from tkinter.ttk import Frame, Button, Style, Label

from tkinter import *
from tkinter import filedialog, font
from PIL import ImageTk, Image

import PIL.Image
import PIL.ImageTk
import cv2

import numpy as np
import tensorflow as tf

new_model =
tf.keras.models.load_model("Con6432_D02_Den128_D02_Den50.model")

CATEGORIES = ["none", 'a', 'b', 'c', 'd', 'e', 'f', 'g',
              'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
              'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']


# create a window
class Example(Frame):
    img = cv2.imread("1.jpg", cv2.IMREAD_UNCHANGED)

    def __init__(self, parent, **kw):
        Frame.__init__(self, parent)
        super().__init__(**kw)
        # size of window
        parent.minsize(width=700, height=650)
        # define theme as default
        self.style = Style()
        self.style.theme_use("default")
        self.pack(fill=BOTH, expand=1)
        # use parent to save a window
```

```
        self.parent = parent
        self.parent.title("ĐỒ ÁN TỐT NGHIỆP")
        myFont = font.Font(family='Helvetica', size=9, weight='bold')
        # create Label
        self.Label = Label(self, text="NHẬN DẠNG KÝ TỰ VIẾT TAY DÙNG
MẠNG NO-RON NHÂN TẠO", font=('TimesBold', 15))
        self.Label.place(x=60, y=10)
        self.Label = self.Label.config(fg='red')
        self.Label = Label(self, text="Ký tự nhận dạng", font=myFont)
        self.Label.place(x=410, y=80)
        self.Label = self.Label.config(fg='red')
        self.Label = Label(self, text="Nguyễn Thanh Tuyết Hân",
font=('Times13'))
        self.Label.place(x=445, y=550)
        self.Label = Label(self, text="Cao Văn Cảnh", font='Times13')
        self.Label.place(x=445, y=575)
        # create Button
        self.Button = Button(self, text="Chọn ảnh", font=myFont,
command=self.open_img, height=1, width=10)
        self.Button.place(x=20, y=50)
        self.Button = Button(self, text="Mở camera", font=myFont,
command=self.video, height=1, width=10)
        self.Button.place(x=120, y=50)
        self.Button = Button(self, text="Chụp ảnh", font=myFont,
command=self.snapshot, height=1, width=10)
        self.Button.place(x=210, y=50)
        self.Button = Button(self, text="Nhận dạng", font=myFont,
command=self.reg, height=1, width=10)
        self.Button.place(x=310, y=50)
        self.Button = Button(self, text="Lưu", font=myFont,
command=self.save_as_file, height=1, width=10)
        self.Button.place(x=608, y=50)
        self.Button = Button(self, text="Xóa", font=myFont,
command=self.clear, height=1, width=10)
        self.Button.place(x=518, y=50)
        # create text to show result
        self.Text = Text(root, height=24, width=30, font='Times13')
        self.Text.place(x=410, y=100)

    def clear(self):
        textsave = self.Text.delete(1.0, END)

    def save_as_file(self):
        f = filedialog.asksaveasfile(mode='w',
defaultextension=".txt", filetypes=[("Text files", ".txt"),

("Word files", ".doc")],
                                     initialdir="dir", title="ĐỒ ÁN
TỐT NGHIỆP")
        if f is None:
            return
        textsave = self.Text.get(1.0, END)
        f.write(textsave)
        f.close()

    def open_img(self):
        global result
        # choose and save image path
        file = filedialog.askopenfile(parent=self, mode='rb',
title='Choose a file')
```

```
        path = file.name
        # Resize image
        self.img = Image.open(path)
        self.img = self.img.resize((360, 270), Image.ANTIALIAS)  #
Resize
        self.img = ImageTk.PhotoImage(self.img)
        # Create a label to show image
        self.label = Label(image=self.img)
        self.label.place(x=20, y=370)
        image = cv2.imread(path, 1)
        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        lower_white = np.array([0, 0, 140])
        upper_white = np.array([256, 60, 256])
        mask = cv2.inRange(hsv, lower_white, upper_white)
        ctrs, hier = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        for i, ctrs in enumerate(ctrs):
            # Get bounding box
            x, y, w, h = cv2.boundingRect(ctrs)
            if w > 15 and h > 15:
                roi = image[y + 20:y + h - 20, x + 20:x + w - 20]
                cv2.imwrite('cropped.jpg', roi)
        result = cv2.imread('cropped.jpg', 1)

    def video(self, video_source=0):
        self.video_source = video_source
        self.vid = MyVideoCapture(self.video_source)
        ret, frame = self.vid.get_frame()
        resized = cv2.resize(frame, (360, 270),
interpolation=cv2.INTER_AREA)
        if ret:
            self.canvas = tkinter.Canvas(self, width=360, height=270)
            self.canvas.place(x=20, y=100)
            self.photo =
PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(resized))
            self.canvas.create_image(0, 0, image=self.photo,
anchor=tkinter.NW)
        self.delay = 30
        self.update()

    def snapshot(self):
        # Get a frame from the video source
        global result
        ret, frame = self.vid.get_frame()
        if ret:
            cv2.imwrite("frame.jpg", cv2.cvtColor(frame,
cv2.COLOR_RGB2BGR))
            self.img = Image.open("frame.jpg")
            self.img = self.img.resize((360, 270), Image.ANTIALIAS)  #
Resize
            self.img = ImageTk.PhotoImage(self.img)
            # Create a label to show image
            self.label = Label(image=self.img)
            self.label.place(x=20, y=370)
            image = cv2.imread("frame.jpg", 1)
            hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
            lower_white = np.array([0, 0, 140])
            upper_white = np.array([256, 60, 256])
            mask = cv2.inRange(hsv, lower_white, upper_white)
```

```
            ctrs, hier = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            for i, ctrs in enumerate(ctrs):
                # Get bounding box
                x, y, w, h = cv2.boundingRect(ctrs)
                if w > 15 and h > 15:
                    roi = image[y + 20:y + h - 20, x + 20:x + w - 20]
                    cv2.imwrite('cropped.jpg', roi)
        result = cv2.imread('cropped.jpg', 1)

    def reg(self):
        def prepare(filepath):
            IMG_SIZE = 28
            img_array = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
            new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
            inverted_image = 255 - new_array
            inverted_image = inverted_image / 255
            return inverted_image.reshape(1, IMG_SIZE, IMG_SIZE, 1)  #
convolutional neural network
        # image detection
        gray = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)
        blurred = cv2.GaussianBlur(gray, (5, 5), 0)
        ret, threshFilled = cv2.threshold(blurred, 130, 255,
cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
        # dilation
        kernel = np.ones((5, 1), np.uint8)
        img_dilation = cv2.dilate(threshFilled, kernel, iterations=1)
        # find contours
        ctrs, hier = cv2.findContours(img_dilation.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        # sort contours
        sorted_ctrs = sorted(ctrs, key=lambda ctr:
cv2.boundingRect(ctr)[0])
        sentence = " "
        # self.Text.insert(1.0, '\n')
        for i, ctrs in enumerate(sorted_ctrs):
            # Get bounding box
            x, y, w, h = cv2.boundingRect(ctrs)
            # Getting ROI
            roi = img_dilation[y - 3:y + h + 3, x - 3:x + w + 3]
            ret, thresh1 = cv2.threshold(roi, 0, 255,
cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
            if (w > 15 and h > 15) or (w > 28 and h > 28):
                resized = cv2.resize(thresh1, (28, 28),
interpolation=cv2.INTER_NEAREST)
                cv2.imwrite('i.jpg'.format(i), resized)
                cv2.imwrite('result/{}.jpg'.format(i), resized)
                predictions = new_model.predict(prepare("i.jpg"))
                sentence = sentence +
str(CATEGORIES[np.argmax(predictions)])
        self.Text.insert(END, sentence)

    def update(self):
        ret, frame = self.vid.get_frame()  # Get a frame from the
video source
        resized = cv2.resize(frame, (360, 270),
interpolation=cv2.INTER_AREA)
        if ret:
            self.canvas = tkinter.Canvas(self, width=360, height=270)
            self.canvas.place(x=20, y=100)
```

```
            self.photo =
PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(resized))
            self.canvas.create_image(0, 0, image=self.photo,
anchor=tkinter.NW)
        self.parent.after(self.delay, self.update)


class MyVideoCapture:
    def __init__(self, video_source=0):
        # Open the video source
        self.vid = cv2.VideoCapture(video_source)
        if not self.vid.isOpened():
            raise ValueError("Unable to open video source",
video_source)
        # Get video source width and height
        self.width = self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
        self.height = self.vid.get(cv2.CAP_PROP_FRAME_HEIGHT)

    def get_frame(self):
        ret, frame = self.vid.read()
        if self.vid.isOpened():
            if ret:
                # Return a boolean success flag and the current frame
converted to BGR
                return (ret, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
            else:
                return (ret, None)
        else:
            return (ret, None)

    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()

root = Tk()
app = Example(root)
app.parent.iconbitmap('icon.ico')
root.mainloop()
```