

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



HUỶNH MINH NHỰT

**NGHIÊN CỨU MÔ HÌNH NHẬN DẠNG
CHỮ KÝ VIẾT TAY SỬ DỤNG HỌC SÂU CNN**

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

TP. HỒ CHÍ MINH – NĂM 2022

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



HUỲNH MINH NHỰT

**NGHIÊN CỨU MÔ HÌNH NHẬN DẠNG
CHỮ KÝ VIẾT TAY SỬ DỤNG HỌC SÂU CNN**

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

MÃ SỐ: 8.48.01.04

LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC:

TS. NGUYỄN XUÂN SÂM

TP. HỒ CHÍ MINH – NĂM 2022

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi. Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Ngoại trừ những tài liệu tham khảo được trích dẫn thì không có đoạn văn nào được công bố trên các tạp chí khoa học, luận văn trong và ngoài nước.

Chương trình demo là chương trình đã được cải thiện tốt nhất và chưa được công bố hay bất kỳ một ai đã làm như kết quả đạt được.

TPHCM, ngày 25 tháng 01 năm 2022

Học viên thực hiện luận văn

Huỳnh Minh Nhật

LỜI CẢM ƠN

Trước tiên, tôi xin chân thành cảm ơn Thầy **TS.Nguyễn Xuân Sâm** đã tận tình hướng dẫn và tạo mọi điều kiện thuận lợi để tôi hoàn thành tốt luận văn này.

Tôi cũng xin gửi lời cảm ơn đến Quý Thầy Cô tại Học Viện Công Nghệ Bưu Chính Viễn Thông Cơ sở Thành phố Hồ Chí Minh đã tận tình giảng dạy và trang bị cho tôi những kiến thức quý báu trong quá trình tham gia học tập tại Trường.

Tôi chân thành biết ơn sâu sắc đến gia đình và bạn bè đã động viên và giúp đỡ tôi hoàn thành khóa học này.

TPHCM, ngày 25 tháng 01 năm 2022

Học viên thực hiện luận văn

Huỳnh Minh Nhựt

MỤC LỤC

LỜI CAM ĐOAN	i
LỜI CẢM ƠN	ii
MỤC LỤC	iii
DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT	v
DANH SÁCH HÌNH VẼ	vi
MỞ ĐẦU	1
1. Tính cấp thiết của đề tài	1
2. Tổng quan vấn đề nghiên cứu	1
3. Mục tiêu nghiên cứu	1
4. Đối tượng, phạm vi nghiên cứu	2
5. Phương pháp nghiên cứu	2
6. Bố cục luận văn	2
CHƯƠNG 1: TỔNG QUAN VỀ XÂY DỰNG HỆ THỐNG NHẬN DẠNG CHỮ KÝ VIẾT TAY	3
1.1 Các phương pháp nhận dạng chữ ký viết tay truyền thống	3
1.2 Machine Learning trong nhận dạng chữ ký viết tay	6
1.2.1 Giới thiệu về machine learning	6
1.2.2 Input và Output	7
1.2.3 Random forest trong nhận dạng chữ ký viết tay	8
1.3 Deep Learning (DL) trong nhận dạng chữ ký viết tay	9
1.3.1 Giới thiệu về deep learning	9
1.3.2 Deep learning trong xây dựng hệ thống nhận dạng chữ ký viết tay	10
1.4 Kết luận chương	11
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CÁC CÔNG TRÌNH LIÊN QUAN.....	12
2.1 Cơ sở lý thuyết	12
2.1.1 Mạng nơ-ron (neural).....	12
2.1.2 Một số kiểu mạng nơ-ron.....	16
2.1.3 Các phương pháp huấn luyện mạng nơ-ron.....	17
2.2 Các công trình liên quan.	24
2.2.1 Mạng LeNet-5 (1998)	24
2.2.2 AlexNet	25
2.2.3 VGG-16	29
2.3 Kết luận chương	35

CHƯƠNG 3: MÔ HÌNH NHẬN BIẾT CHỮ KÝ VIẾT TAY	37
3.1 Mô phỏng chương trình nhận biết chữ ký viết tay	37
3.1.1 Giới thiệu chung	37
3.1.2 Công cụ sử dụng	37
3.2 Môi trường mô phỏng thực nghiệm	38
3.3 Kết luận chương	46
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	47
DANH MỤC CÁC TÀI LIỆU THAM KHẢO	49

DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
CNN	Convolutional Neural Network	Mạng thần kinh tích chập
ANN	Artificial Neural Networks	Mạng Nơ-ron
SNN	Simulated Neural nNetworks	Mạng Nơ-ron mô phỏng
kNN	K-nearest Neighbors	Giải thuật hàng xóm gần nhất
SVM	Support vector machine	Máy vector hỗ trợ
GPU	Graphics Processing Unit	Bộ xử lý đồ họa
AI	Artificial Intelligence	Trí thông minh nhân tạo
RGB	Red Green Blue	Mô hình màu Red Green Blue
ML	Machine Learning	Máy học
DL	Deep Learning	Học sâu
HMM	Hidden Markov Model	Mô hình Markov ẩn
SOM	Self Organizing Map	Bản đồ tự tổ chức

DANH SÁCH HÌNH VẼ

Hình 1.1. Tổng quan về Quy trình Nghiên cứu	3
Hình 1.2. Hình ảnh chữ ký trong quá trình thu thập.	4
Hình 1.3. Ví dụ về Chữ ký được Cắt và Dán nhãn.	5
Hình 1.4. Tổng quan về xây dựng thuật toán xác minh chữ ký.	5
Hình 1.5. Kiến trúc của 1 mô hình nhận dạng chữ ký [1].....	8
Hình 1.6. Mô tả thuật toán random forest	8
Hình 1.7. Mẫu training set trong random forest.....	9
Hình 1.8. Áp dụng random forest vào nhận dạng chữ ký	9
Hình 1.9. Mối liên hệ AI, ML, DL.....	10
Hình 1.10. Deep learning trong nhận dạng chữ ký viết tay.	10
Hình 2.1. Cấu tạo một nơ-ron	12
Hình 2.2. Hàm đồng nhất (Identify function)	14
Hình 2.3. Hàm bước nhị phân	14
Hình 2.4. Hàm Sigmoid	15
Hình 2.5. Hàm sigmoid lưỡng cực	15
Hình 2.6. Mạng nơ-ron truyền thẳng nhiều lớp (Feed-forward neural network).....	16
Hình 2.7. Mạng nơ-ron hồi quy (Recurrent neural network).....	17
Hình 2.8. Mô hình học có giám sát (Supervised learning model)	18
Hình 2.9. Mô phỏng mạng nơ-ron tích chập.....	20
Hình 2.10. Minh họa tích chập.....	21
Hình 2.11. Ảnh mờ sau khi chập.....	22
Hình 2.12. Mô phỏng số làm mờ ảnh (bước 1).....	22
Hình 2.13. Mô phỏng số làm mờ ảnh (bước 2).....	23
Hình 2.14. Ảnh được phát hiện biên sau khi chập	23
Hình 2.15. Kiến trúc LeNet-5.....	24
Hình 2.16. Kiến trúc AlexNet.....	26
Hình 2.17. Kiến trúc VGG-16.....	30
Hình 3.1. Quá trình import thư viện.....	39

Hình 3.2. Chia dữ liệu đầu vào thành 2 nhóm thật và giả.....	40
Hình 3.3. Hàm lấy ra 2 chữ thật và 1 chữ ký giả từ bộ dữ liệu đã được chia.....	40
Hình 3.4. Hàm tạo các cặp dữ liệu theo kích thước batch_size.....	41
Hình 3.5. Mô hình CNN.....	42
Hình 3.6. Training model.....	43
Hình 3.7. Tính toán ngưỡng và độ chính xác sau khi train model.....	43
Hình 3.8. Hàm kiểm tra chữ ký thật giả.....	44
Hình 3.9. Kết quả chữ ký thật lần 1.....	44
Hình 3.10. Kết quả chữ ký thật lần 2.....	45
Hình 3.11. Kết quả chữ ký thật lần 3.....	45
Hình 3.12. Kết quả chữ ký thật lần 4.....	45
Hình 3.14. Kết quả chữ ký giả lần 1.....	46
Hình 3.15. Kết quả chữ ký giả lần 2.....	46
Hình 3.16. Kết quả chữ ký giả lần 3.....	46
Hình 3.17. Kết quả chữ ký giả lần 4.....	47

MỞ ĐẦU

1. Tính cấp thiết của đề tài

Trong lĩnh vực sinh trắc học hành vi, việc xác minh chữ ký là quy trình tham chiếu để xác thực một người. Chữ ký được coi là “con dấu phê duyệt” để xác minh sự chấp thuận của người dùng và vẫn là phương tiện xác thực được ưa chuộng nhất. Chữ ký viết tay vốn áp đặt trách nhiệm pháp lý về tài chính và đạo đức, là một kỹ thuật xác thực vẫn được sử dụng rộng rãi ngày nay, đặc biệt trong các văn bản pháp lý, giao dịch ngân hàng và thương mại. Do đó, chữ ký viết tay thường bị các kẻ xấu lợi dụng và sử dụng để lừa đảo.

Để ngăn chặn gian lận và mục đích xấu, xác minh chữ ký được sử dụng nhằm xác minh việc phân biệt chữ ký giả mạo với chữ ký thật. Hạn chế quan trọng của chữ ký viết tay là chúng không thể sao chép theo cùng một cách. Các chữ ký có thể khác nhau tùy thuộc vào dụng cụ viết được sử dụng (bút chì, giấy..). Ngay cả những người tài năng nhất cũng không bao giờ có thể ký cùng một chữ ký theo cùng một cách.

Mục tiêu kết quả đạt của hệ thống là: Phân biệt chữ ký viết tay thật hay giả dựa trên kỹ thuật học sâu. Phương pháp học sâu được nghiên cứu là thuật toán Mạng Nơ-ron tích chập (CNN).

2. Tổng quan vấn đề nghiên cứu

Mạng Nơ-ron tích chập (CNN) là một trong những mô hình của Học sâu (DL). Tác dụng của thuật toán này chính là giúp chúng ta tạo ra những hệ thống thông minh, có sự phản ứng với độ chính xác cao.

Hệ thống phân tích kết quả xác thực chữ ký viết tay được chứa bởi những phương pháp học sâu (DL).

3. Mục tiêu nghiên cứu

Xác thực chữ ký viết tay để phân biệt chữ ký viết tay thật hay giả để ngăn chặn việc kẻ xấu lợi dụng trong các giao dịch ngân hàng, văn bản pháp lý và thương mại....

Nội dung chính là việc biến đổi dữ liệu thu thập này thành thông tin bằng mô hình học sâu nhằm phân tích kết quả tốt nhất của chữ ký thật và giả mạo.

Vấn đề nghiên cứu trên đang là một chủ đề thời sự và được nhiều nhóm nghiên cứu trong và ngoài nước rất quan tâm.

4. Đối tượng, phạm vi nghiên cứu

Đối tượng nghiên cứu: Các chữ ký viết tay của người Việt Nam

Phạm vi nghiên cứu: Nghiên cứu kỹ thuật xử lý ảnh các chữ ký viết tay của người Việt.

5. Phương pháp nghiên cứu

Phương pháp lý thuyết: Dựa trên các tài liệu về cơ sở lý thuyết về xử lý ảnh số, lọc trích ảnh số. Sử dụng ngôn ngữ Python để xây dựng hệ thống dựa trên thuật toán Mạng Nơ-ron về nhận dạng chữ ký viết tay.

Phương pháp thực nghiệm: Xây dựng chương trình thử nghiệm và kiểm thử tính hiệu của chương trình với những chữ ký viết tay của người Việt.

6. Bố cục luận văn

Ngoài phần mở đầu, mục lục, kết luận và tài liệu tham khảo, nội dung chính của luận án được chia thành 3 chương, cụ thể như sau:

Chương 1: Tổng quan về xây dựng hệ thống.

Chương 2: Cơ sở lý thuyết và các công trình liên quan.

Chương 3: Mô hình nhận biết chữ ký viết tay.

CHƯƠNG 1: TỔNG QUAN VỀ XÂY DỰNG HỆ THỐNG NHẬN DẠNG CHỮ KÝ VIẾT TAY

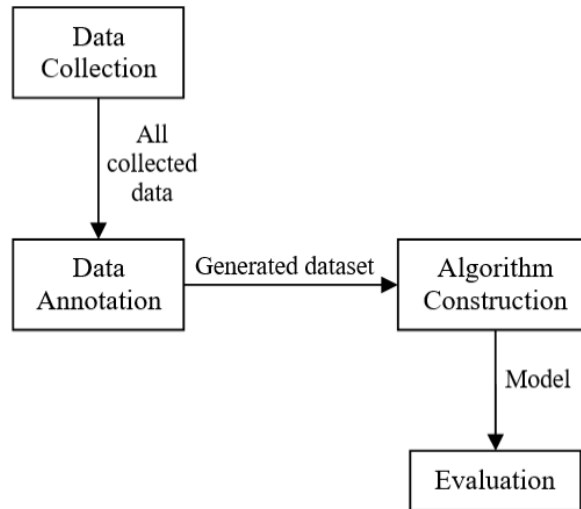
1.1 Các phương pháp nhận dạng chữ ký viết tay truyền thống

Chữ ký viết tay là một biểu tượng viết tay của con người. Chữ ký [1] còn là một công cụ phổ biến để xác định danh tính của một người, đặc biệt là về mặt phê chuẩn các tài liệu mật. Điều này dẫn đến nhu cầu nghiên cứu thêm về xác minh chữ ký, nhằm ngăn chặn việc các bên thiếu trách nhiệm lợi dụng chữ ký.

Mỗi con người có một cái gì đó có thể được sử dụng làm bằng chứng về danh tính, chẳng hạn như: chữ ký, dấu vân tay, khuôn mặt, Ngày nay, hầu hết các công ty trong lĩnh vực ngân hàng và các lĩnh vực khác dựa vào chữ ký để xác minh danh tính của ai đó. Do tầm quan trọng của chữ ký của một người nào đó trong hệ thống, luôn có nguy cơ sử dụng sai chữ ký của các bên thiếu trách nhiệm. Việc lạm dụng bao gồm giả mạo chữ ký để làm sai lệch danh tính của ai đó để xâm nhập vào hệ thống hoặc truy cập các tài liệu bí mật. Bởi vì tình trạng đáng lo ngại này, các nhà nghiên cứu đã tiến hành một số nghiên cứu về chữ ký xác minh bằng cách sử dụng các phương pháp khác nhau [1],[2]. Nghiên cứu tập trung vào xác minh xem chữ ký đã cho là thật hay giả.

Mặc dù có nhiều nghiên cứu về xác minh chữ ký, hiệu suất của các phương pháp đã được phát triển vẫn chưa đạt yêu cầu. Ngoài ra, hầu hết các phương pháp chỉ sử dụng máy học truyền thống. Do đó, chúng tôi phát triển các mô hình và xác minh chữ ký các thuật toán sử dụng học sâu. Người ta mong đợi rằng sự phát triển của mô hình này và thuật toán sẽ cải thiện hiệu suất và độ chính xác của các xác minh chữ ký hiện tại. Xác minh chữ ký có thể được chia thành hai loại, tức là, xác minh chữ ký ngoại tuyến và xác minh chữ ký trực tuyến. Xác minh chữ ký ngoại tuyến là một quá trình xác minh thực hiện trên bản quét chữ ký của một người mà trước đó đã được thực hiện trên giấy. Mặt khác, xác minh chữ ký trực tuyến có thể được thực hiện trực tiếp khi ai đó ký một cái gì đó thông qua công cụ (ví dụ: máy tính bảng) cũng có thể được tích hợp trực tiếp vào một hệ thống có thể xác minh chữ ký.

Tuy nhiên để xác minh chữ ký thường thực hiện theo 4 giai đoạn sau: thu thập dữ liệu, chú thích dữ liệu, xây dựng mô hình và đánh giá mô hình [3]:



Hình 1.1: Tổng quan về Quy trình Nghiên cứu

Quy trình nghiên cứu của chúng tôi gồm bốn thành phần chính, các thành phần được giải thích chi tiết trong phần này:

Thu thập dữ liệu:

Là tập hợp các quy trình phân tích thu thập dữ liệu của người dùng.

<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2
<i>Chris</i>	<i>Chris</i>	<i>Chris</i>	2

Hình 1.2: Hình ảnh chữ ký trong quá trình thu thập [3]

Ví dụ về dữ liệu được thu thập, hai Cột đầu tiên từ bên trái là chữ ký ban đầu, sau đó là cột thứ ba và thứ tư là chữ ký bắt chước của người khác mà tình nguyện viên đã giả mạo.

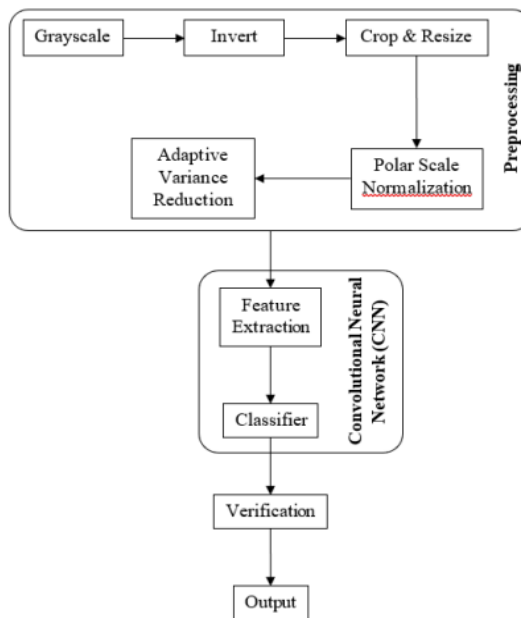
Chú thích dữ liệu:

Sau khi hoàn tất việc thu thập dữ liệu, tất cả các chữ ký sẽ được quét, chú thích và dán nhãn để sử dụng trong giai đoạn này. Mỗi trang của tài liệu được quét bao gồm 20 chữ ký gốc của người 1, 10 chữ ký giả mạo của người 2 và 10 chữ ký giả mạo của người 3, như trong Hình 1.2:



Hình 1.3: Ví dụ về Chữ ký được Cắt và Dán nhãn

Xây dựng mô hình:



Hình 1.4: Tổng quan về xây dựng thuật toán xác minh chữ ký [4]

Đây là giai đoạn quan trọng, sau khi tập dữ liệu thông qua quá trình tiền xử lý, dữ liệu được sử dụng trực tiếp làm đầu vào cho CNN sẽ được xử lý để trích xuất các tính năng. Sau đó, các đặc điểm đã được trích xuất có thể được gửi đến bộ phân loại

và quá trình xác minh sẽ tạo ra mô hình được sử dụng để xác định cho dù chữ ký là thật hay giả.

Đánh giá mô hình:

Giai đoạn này là giai đoạn cuối cùng trong quá trình triển khai mô hình. Tại giai đoạn này, mỗi mô hình sẽ được đánh giá, sau đó sẽ tính độ chính xác và tỷ lệ lỗi với một số cấu hình. Để biết chi tiết về sự sắp xếp và kết quả của các cuộc đánh giá.

1.2 Machine Learning trong nhận dạng chữ ký viết tay

1.2.1 Giới thiệu về machine learning

Machine Learning [4],[5] là việc sử dụng và phát triển các hệ thống máy tính có khả năng học hỏi và thích ứng mà không cần tuân theo các chỉ dẫn rõ ràng, bằng các sử dụng các thuật toán và mô hình thống kê để phân tích và rút ra suy luận từ các mẫu trong dữ liệu đầu vào.

Máy học là một lĩnh vực con của trí tuệ nhân tạo (AI). Mục tiêu của máy học nói chung là hiểu cấu trúc của dữ liệu và điều chỉnh dữ liệu đó thành các mô hình mà con người có thể hiểu và sử dụng. Mặc dù máy học là một lĩnh vực trong khoa học máy tính, nó khác với các phương pháp tiếp cận tính toán truyền thống. Trong máy tính truyền thống, thuật toán là tập hợp các lệnh được lập trình rõ ràng được máy tính sử dụng để tính toán hoặc giải quyết vấn đề. Thay vào đó, các thuật toán học máy cho phép máy tính đào tạo về đầu vào dữ liệu và sử dụng phân tích thống kê để đưa ra các giá trị nằm trong một phạm vi cụ thể. Do đó, học máy tạo điều kiện cho máy tính xây dựng mô hình từ dữ liệu mẫu để tự động hóa quy trình ra quyết định dựa trên dữ liệu đầu vào.

Xác minh chữ ký [6] trực quan được xây dựng một cách tự nhiên như một nhiệm vụ học máy. Một chương trình được cho là thể hiện khả năng máy học trong việc thực hiện một nhiệm vụ nếu nó có thể học hỏi từ những người mẫu, cải thiện khi số lượng người mẫu tăng lên, v.v. [3].

Xác minh chữ ký tự động [4] là một nhiệm vụ trong đó máy học có thể được sử dụng như một phần tự nhiên của quy trình. Hai cách tiếp cận học máy khác nhau, một phương pháp liên quan đến độ chính xác và một phương pháp khác chỉ liên quan

đến độ chính xác cho một trường hợp cụ thể đã được mô tả. Cả hai cách tiếp cận đều liên quan đến việc sử dụng một thước đo tương tự để tính toán khoảng cách giữa các đặc điểm của hai chữ ký.

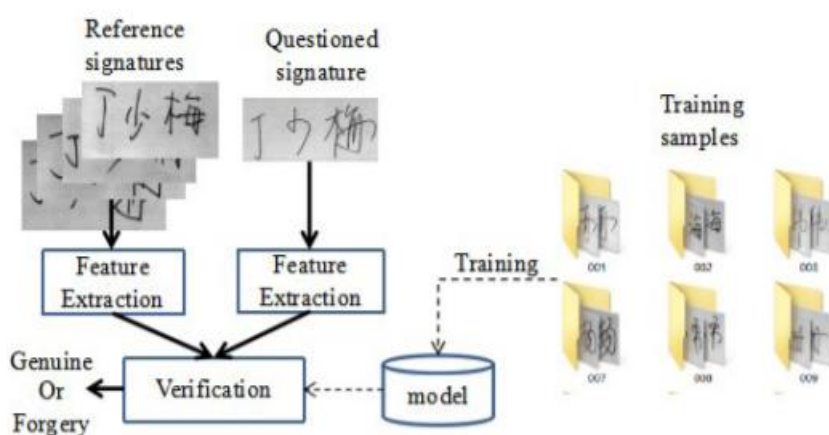
Trong trường hợp học thông thường, mục tiêu là học từ một lượng lớn các mẫu chữ ký thật và giả. Trọng tâm là phân biệt sự khác biệt giữa hàng thật-chính hãng và sự khác biệt giữa hàng thật-giả mạo. Bài toán học được phát biểu là học một bài toán phân loại hai lớp trong đó đầu vào bao gồm sự khác biệt giữa một cặp chữ ký. Nhiệm vụ xác minh được thực hiện bằng cách so sánh chữ ký được hỏi với mỗi chữ ký đã biết. Vấn đề học tập nói chung có thể được xem như một vấn đề mà việc học tập diễn ra với những điểm gần như thiếu sót như một ví dụ phản chứng [5]. Học tập đặc biệt tập trung vào việc học từ các mẫu chính chủ của một người cụ thể. Trọng tâm là tìm hiểu sự khác biệt giữa các thành viên của lớp genuines. Nhiệm vụ xác minh về cơ bản là một bài toán một lớp để xác định xem chữ ký được hỏi có thuộc lớp đó hay không.

1.2.2 Input và Output

Machine Learning sử dụng trong nhận dạng chữ ký viết tay trước đây chủ yếu sử dụng cây quyết định (Decision Tree) hoặc cao hơn là sử dụng random forest để nhận dạng chúng.

Input: Là các hình ảnh chữ ký được gắn nhãn.

Output: Đưa ra dự đoán về chữ ký của ai trong tập training ở đây là các nhãn 001, 002, 003?

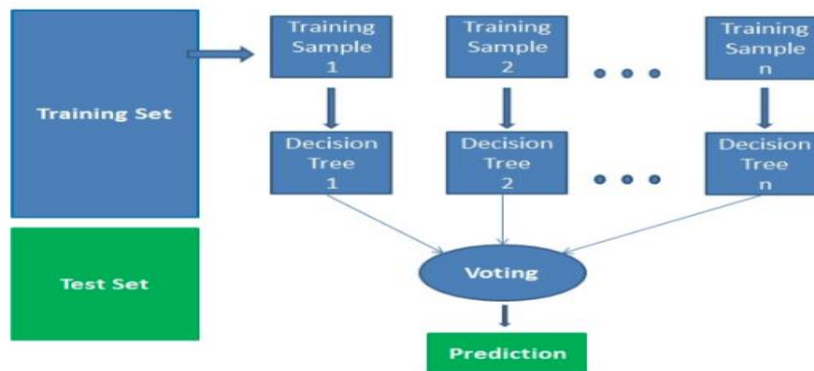


Hình 1.5: Kiến trúc của 1 mô hình nhận dạng chữ ký [1]

1.2.3 Random forest trong nhận dạng chữ ký viết tay

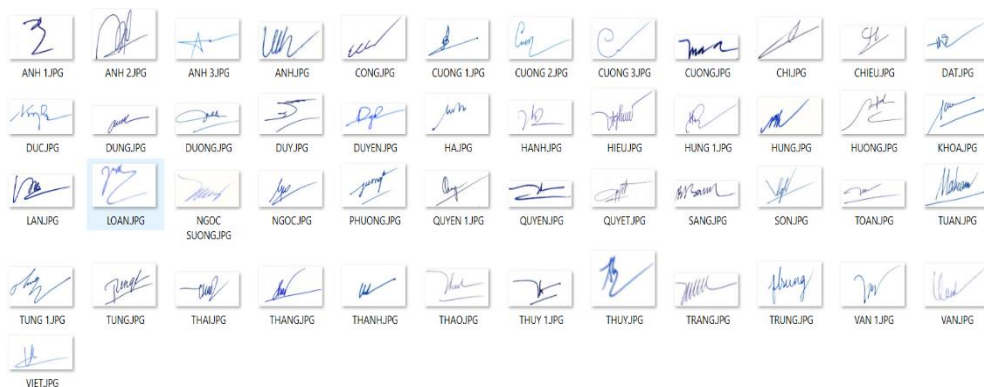
Thuật toán random forest trong nhận dạng chữ ký viết tay sẽ hoạt động theo 5 bước:

- Bước 1: Tiền xử lý hình ảnh
- Bước 2: Chọn các mẫu chữ ký ngẫu nhiên từ tập dữ liệu huấn luyện
- Bước 3: Xây dựng cây quyết định cho mẫu chữ ký
- Bước 4: Các cây quyết định đưa ra các dự đoán
- Bước 5: Chọn kết quả là chữ ký được dự đoán nhiều nhất là dự đoán cuối cùng



Hình 1.6: Mô tả thuật toán random forest [5]

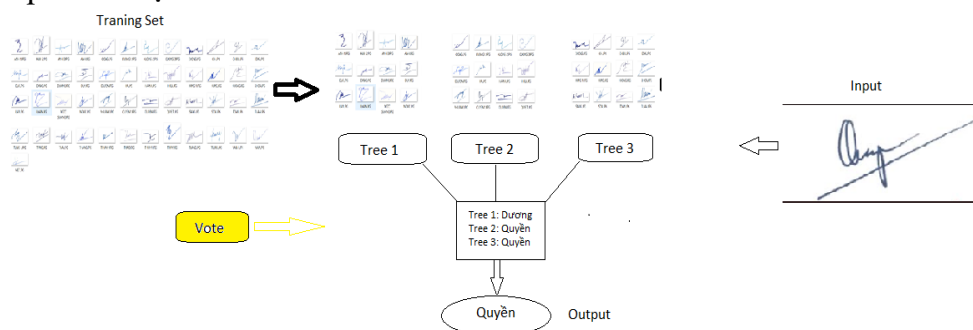
Training Set: Là tập các hình ảnh nằm trong các thư mục có gắn nhãn hoặc các hình ảnh có nhãn.



Hình 1.7: Mẫu training set trong random forest [5]

Training Sample: Là các tập được lựa chọn ngẫu nhiên từ tập training set từ đó tạo thành các cây quyết định chuẩn bị cho quá trình voting.

Test: Đưa dữ liệu là hình ảnh chữ ký vào mô hình. Kết thúc quá trình voting sẽ là kết quả của dự đoán



Hình 1.8: Áp dụng random forest vào nhận dạng chữ ký [4]

Ưu điểm:

- Thuật toán mang tính chất mô phỏng dễ hình dung, không phức tạp nhiều về thuật toán.
- Thuật toán được hỗ trợ trên nhiều thuật toán, trên nhiều lĩnh vực khác nhau.

Nhược điểm:

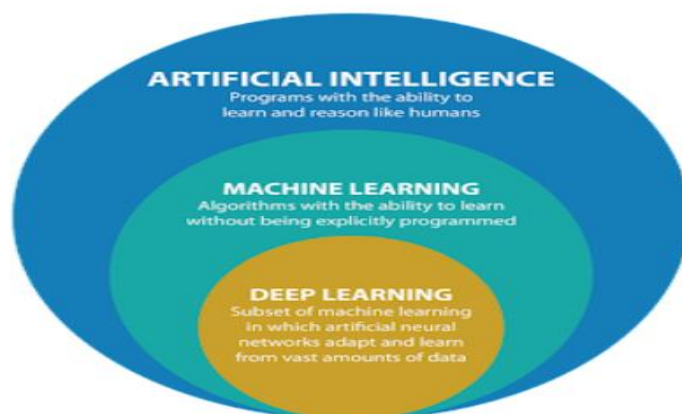
- Độ chính xác không cao.

1.3 Deep Learning (DL) trong nhận dạng chữ ký viết tay

1.3.1 Giới thiệu về deep learning

Có thể nói deep learning là một loại machine learning dựa trên mạng nơron nhân tạo, trong đó nhiều lớp xử lý được sử dụng để trích xuất đặc trưng và nhiều thứ hơn từ dữ liệu.

Hiện nay, AI [7] đang phát triển mạnh mẽ, để đạt được bước tiến như hiện tại thì deep learning như là một chìa khóa thúc đẩy AI ngày càng tiến xa hơn và sử dụng rộng rãi với đời sống con người hơn.

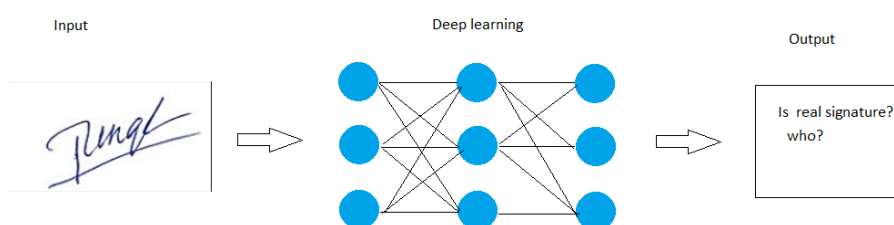


Hình 1.9: Mối liên hệ AI, ML, DL [7]

1.3.2 Deep learning trong xây dựng hệ thống nhận dạng chữ ký viết tay

Bài toán đặt ra: Chúng ta sẽ có 1 bộ các chữ ký thật, giả có gắn nhãn là tên của các nhân vật đại diện cho từng chữ ký. Bài toán đặt ra là khi đưa ảnh của 1 chữ ký vào có thể nhận dạng đó là chữ ký thật hay giả? Và chữ ký đó là của ai?

Để giải quyết vấn đề trên được triệt để đưa ra các dự đoán có độ chính xác cao, có độ tin cậy và an toàn hơn. Trong việc xây dựng này, đề xuất thực hiện bằng deep learning, sử dụng CNN hình thành nên các mạng nơ-ron nhân tạo, tiến hành phân tích dự đoán.



Hình 2.10: Deep learning trong nhận dạng chữ ký viết tay [9]

Ta nhận thấy thuật toán deep learning đem đến cho chúng ta kết quả xác thực với mạng nơ-ron nhân tạo. Đem đến sự chính xác và mức độ khả quan hơn so với machine learning.

1.4 Kết luận chương

Hiểu biết được những khái niệm tổng quan về trí tuệ nhân tạo. Thấy được tầm quan trọng và ý nghĩa của trí tuệ nhân tạo. Từ đó, có cái nhìn về học máy, học sâu các lợi ích về sau mà trí tuệ nhân tạo sẽ mang lại cho con người. Áp dụng vào việc sử dụng để nhận dạng chữ ký viết tay.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CÁC CÔNG TRÌNH LIÊN QUAN

2.1 Cơ sở lý thuyết

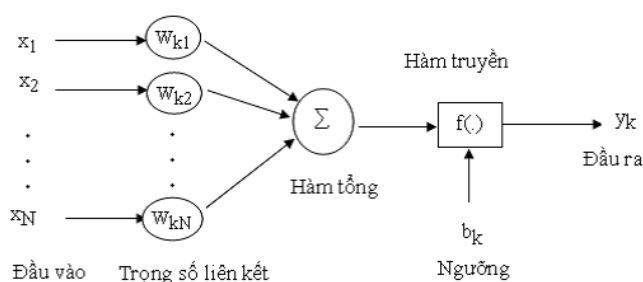
2.1.1 Mạng nơ-ron (neural)

2.1.1.1 Giới thiệu về mạng nơ-ron

Mạng nơ-ron, còn được gọi là mạng nơ-ron nhân tạo (ANN) hoặc mạng nơ-ron mô phỏng (SNN), là một tập hợp con của học máy và là trung tâm của các thuật toán học sâu. Tên và cấu trúc của chúng được lấy cảm hứng từ não người, bắt chước cách các tế bào thần kinh sinh học truyền tín hiệu cho nhau.

Mạng nơ-ron nhân tạo (ANN) bao gồm một lớp nút, chứa một lớp đầu vào, một hoặc nhiều lớp ẩn và một lớp đầu ra. Mỗi nút, hoặc nơ-ron nhân tạo, kết nối với một nút khác và có trọng số và ngưỡng liên quan. Nếu đầu ra của bất kỳ nút riêng lẻ nào vượt quá giá trị ngưỡng được chỉ định, nút đó sẽ được kích hoạt, gửi dữ liệu đến lớp tiếp theo của mạng. Nếu không, không có dữ liệu nào được chuyển đến lớp tiếp theo của mạng.

Cấu trúc nơ-ron nhân tạo:



Hình 2.1: Cấu tạo một nơ-ron [2]

Các thành phần cơ bản của một nơ-ron nhân tạo bao gồm:

- **Input signals:** Là các tín hiệu đầu vào của nơ-ron, các tín hiệu này thường được đưa vào dưới dạng một vector N chiều.
- **Thành phần liên kết:** được kết nối bởi trọng số liên kết – synaptic weight. Các trọng số kết nối giữa tín hiệu vào thứ j với nơ-ron k thường được kí

hiệu là w_{kj} . Thông thường, các trọng số liên kết này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục.

- Thành phần tổng (Summing function): được dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó.
- Thành phần ngưỡng (còn gọi là một độ lệch - bias): thành phần này thường được đưa vào như một thành phần của hàm truyền.
- Thành phần truyền (Transfer function): Thành phần này được dùng để giới hạn phạm vi đầu ra của mỗi nơ-ron. Nó nhận đầu vào là kết quả của hàm tổng và ngưỡng.
- Đầu ra: Là tín hiệu đầu ra của một nơ-ron, với mỗi nơ-ron sẽ có tối đa là một đầu ra.

Xét về mặt toán học, cấu trúc của một nơ-ron k , được mô tả bằng cặp biểu thức sau:

$$u_k = \sum_{j=1}^p w_{kj} x_j \quad \text{và} \quad y_k = f(u_k - b_k) \quad (2.1)$$

Trong đó: x_1, x_2, \dots, x_p : là các tín hiệu vào; $(w_{k1}, w_{k2}, \dots, w_{kp})$ là các trọng số liên kết của nơ-ron thứ k ; u_k là hàm tổng; b_k là một ngưỡng; f là hàm truyền và y_k là tín hiệu đầu ra của nơ-ron.

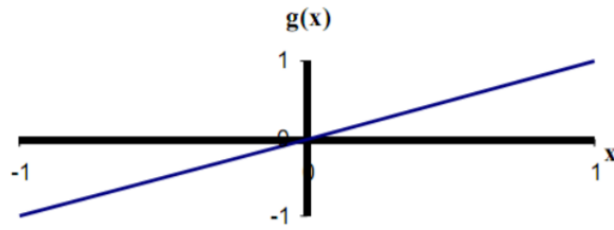
2.1.1.2 Một số hàm truyền thông dụng

Phần lớn các đơn vị trong mạng nơ-ron chuyển net input bằng cách sử dụng một hàm vô hướng (scalar-to-scalar function) [9] gọi là hàm kích hoạt, kết quả của hàm này là một giá trị gọi là mức độ kích hoạt của đơn vị (unit's activation). Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác. Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm bẹp (squashing). Các hàm kích hoạt hay được sử dụng là:

Hàm đồng nhất (Linear function, Identity function)

$$g(x) = x$$

Nếu coi các đầu vào là một đơn vị thì chúng sẽ sử dụng hàm này. Đôi khi một hằng số được nhân với net-input để tạo ra một hàm đồng nhất.



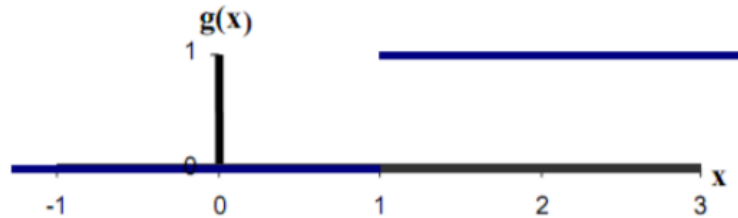
Hình 2.2: Hàm đồng nhất (Identify function) [3]

Hàm bước nhị phân (Binary step function, Hard limit function)

Hàm này cũng được biết đến với tên "Hàm ngưỡng" (Threshold function hay Heaviside function). Đầu ra của hàm này được giới hạn vào một trong hai giá trị:

$$g(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Hàm này được sử dụng một lớp. Trong hình vẽ sau, θ được chọn bằng 1.

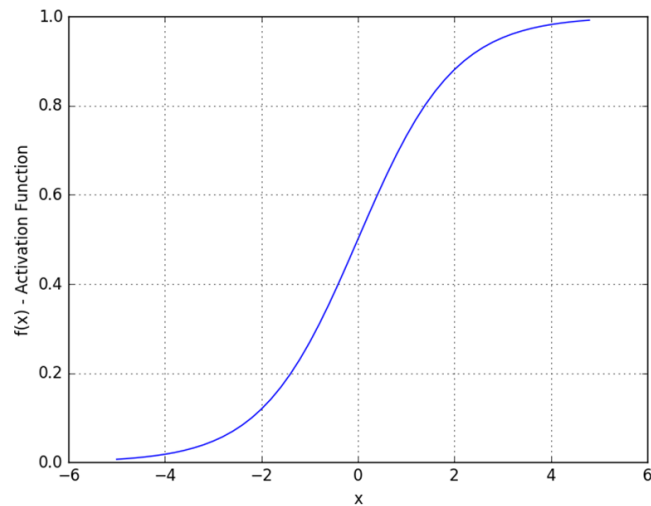


Hình 2.3: Hàm bước nhị phân [3]

Hàm sigmoid (Sigmoid function (logsig))

$$g(x) = \frac{1}{1 + e^{-x}}$$

Hàm này rất thuận lợi khi sử dụng vì được huấn luyện bởi thuật toán Lan truyền ngược, nó dễ lấy đạo hàm. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng $[0,1]$.

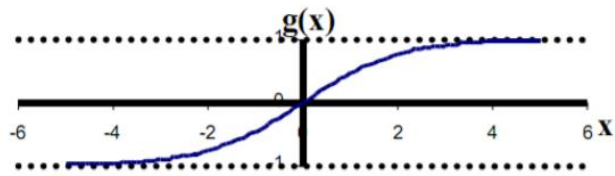


Hình 2.4: Hàm Sigmoid [6]

Hàm sigmoid lưỡng cực (Bipolar sigmoid function (tansig)).

$$g(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

Hàm này có các thuộc tính tương tự hàm sigmoid. Nó làm việc tốt đối với các ứng dụng có đầu ra yêu cầu trong khoảng $[-1, 1]$.



Hình 2.5: Hàm sigmoid lưỡng cực [7]

Hàm này rất thuận lợi khi sử dụng vì được huấn luyện bởi thuật toán Lan truyền ngược, nó dễ lấy đạo hàm. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng $[0, 1]$.

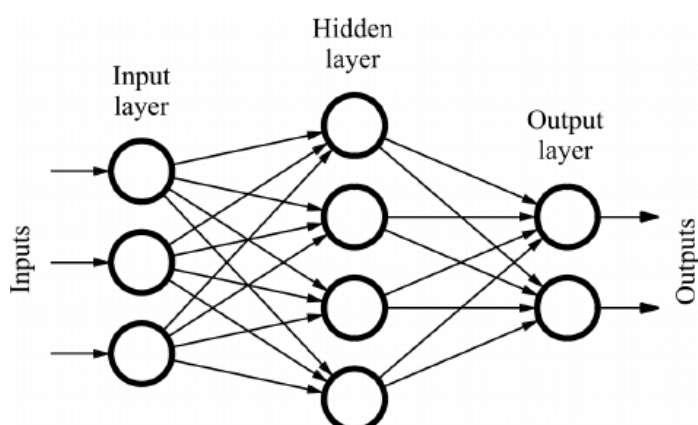
Đối với các đơn vị đầu ra (output units), các hàm chuyển cần được chọn sao cho phù hợp với sự phân phối của các giá trị đích mong muốn. Chúng ta đã thấy rằng đối với các giá trị ra trong khoảng $[0, 1]$, hàm sigmoid là có ích; đối với các giá trị đích mong muốn là liên tục trong khoảng đó thì hàm này cũng vẫn có ích, nó có thể cho ta các giá trị ra hay giá trị đích được căn trong một khoảng của hàm kích hoạt đầu ra. Nhưng nếu các giá trị đích không được biết trước khoảng xác định thì hàm

hay được sử dụng nhất là hàm đồng nhất (identity function). Nếu giá trị mong muốn là dương nhưng không biết cận trên thì nên sử dụng một hàm kích hoạt dạng mũ (exponential output activation function).

2.1.2 Một số kiểu mạng nơ-ron

2.1.2.1 Mạng truyền thẳng (Feed-forward neural network)

Mạng nơ-ron truyền thẳng là một mạng nơ-ron nhân tạo trong đó các kết nối giữa các nút không tạo thành một chu trình. Đối lập với mạng nơ-ron truyền thẳng là mạng nơ-ron tuần hoàn, trong đó các đường dẫn nhất định được tuần hoàn. Mô hình truyền thẳng là dạng mạng nơ-ron đơn giản nhất vì thông tin chỉ được xử lý theo một hướng. Mặc dù dữ liệu có thể đi qua nhiều nút ẩn nhưng nó luôn di chuyển theo một hướng và không bao giờ ngược.

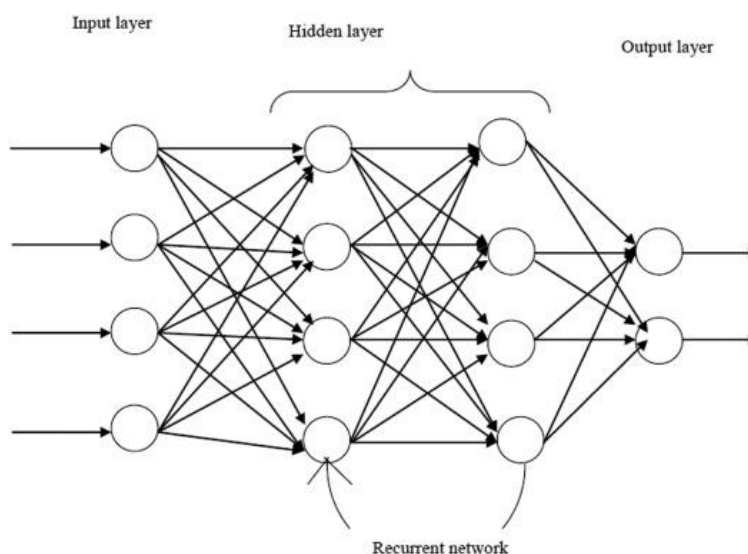


Hình 2.6: Mạng nơ-ron truyền thẳng [1]

2.1.2.2 Mạng hồi quy (Recurrent neural network)

Mạng nơ-ron tuần hoàn (RNN) là một loại mạng Nơ-ron, được sử dụng rộng rãi để thực hiện quá trình phân tích trình tự vì RNN được thiết kế để trích xuất thông tin ngữ cảnh bằng cách xác định sự phụ thuộc giữa các tem thời gian khác nhau. RNN bao gồm nhiều lớp lặp lại liên tiếp và các lớp này được lập mô hình tuần tự để ánh xạ trình tự với các trình tự khác. RNN có một khả năng mạnh mẽ để thu thập dữ liệu theo ngữ cảnh từ chuỗi. Tuy nhiên, các dấu hiệu ngữ cảnh trong cấu trúc mạng là ổn định và được sử dụng hiệu quả để đạt được quá trình phân loại dữ liệu. RNN có thể

vận hành các chuỗi với độ dài tùy ý. Hình 2.7 đại diện cho kiến trúc của bộ phân loại RNN.



Hình 2.7: Mạng nơ-ron hồi quy (Recurrent neural network) [1]

Trong các ứng dụng khác, cách chạy động tạo thành đầu ra của mạng thì những sự thay đổi các giá trị kích hoạt là đáng quan tâm.

2.1.3 Các phương pháp huấn luyện mạng nơ-ron

Học là quá trình thay đổi hành vi của các vật theo một cách nào đó làm cho chúng có thể thực hiện tốt hơn trong tương lai.

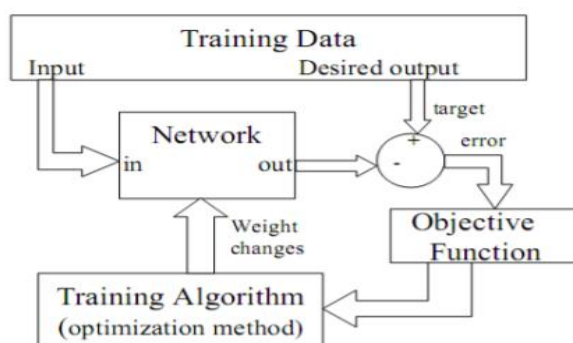
Một mạng nơ-ron được huấn luyện sao cho với một tập các vector đầu vào X , mạng có khả năng tạo ra tập các vector đầu ra mong muốn Y của nó. Tập X được sử dụng cho huấn luyện mạng được gọi là tập huấn luyện (training set). Các phần tử x thuộc X được gọi là các mẫu huấn luyện (training example). Quá trình huấn luyện bản chất là sự thay đổi các trọng số liên kết của mạng.

Trong quá trình này, các trọng số của mạng sẽ hội tụ dần tới các giá trị sao cho mỗi vector đầu vào x từ tập huấn luyện, mạng sẽ cho ra vector đầu ra y như mong muốn.

Có ba phương pháp học phổ biến là học có giám sát (Supervised learning), học không giám sát (Unsupervised learning) và học củng cố (Reinforcement learning).

2.1.3.1 Học có giám sát (Supervised Learning)

Mạng được huấn luyện [12] bằng cách cung cấp cho nó các cặp mẫu đầu vào và các đầu ra mong muốn (target values). Các cặp được cung cấp bởi "thầy giáo", hay bởi hệ thống trên đó mạng hoạt động. Sự khác biệt giữa các đầu ra thực tế so với các đầu ra mong muốn được thuật toán sử dụng để thích ứng các trọng số trong mạng. Điều này thường được đưa ra như một bài toán xấp xỉ hàm số - cho dữ liệu huấn luyện bao gồm các cặp mẫu đầu vào x , và một đích tương ứng t , mục đích là tìm ra hàm $f(x)$ thỏa mãn tất cả các mẫu học đầu vào.



Hình 2.8: Mô hình học có giám sát (Supervised learning model) [9]

2.1.3.2 Học không giám sát (Unsupervised Learning)

Là việc học không cần có bất kỳ một sự giám sát nào. Trong bài toán học không giám sát, tập dữ liệu huấn luyện được cho dưới dạng:

$D = \{(x_1, x_2, \dots, x_N)\}$, với (x_1, x_2, \dots, x_N) là vector đặc trưng của mẫu huấn luyện. Nhiệm vụ của thuật toán là phải phân chia tập dữ liệu D thành các nhóm con, mỗi nhóm chứa các vector đầu vào có đặc trưng giống nhau. Như vậy với học không giám sát, số lớp phân loại chưa được biết trước, và tùy theo tiêu chuẩn đánh giá độ tương tự giữa các mẫu mà ta có thể có các lớp phân loại khác nhau.

2.1.3.3 Học củng cố (Reinforcement learning)

Đôi khi còn được gọi là học thưởng-phạt [12][13] (rewardpenalty learning), là sự tổ hợp của cả hai mô hình trên. Phương pháp này cụ thể như sau: với vector đầu vào, quan sát vector đầu ra do mạng tính được. Nếu kết quả được xem là “tốt” thì mạng sẽ được thưởng theo nghĩa tăng các trọng số kết nối lên; ngược lại mạng sẽ bị phạt, các trọng số kết nối không thích hợp sẽ được giảm xuống. Do đó học tăng cường là học theo nhà phê bình (critic), ngược với học có giám sát là học theo thầy giáo (teacher).

2.1.3.4 Học có giám sát trong mạng nơ-ron

Học có giám sát có thể được xem như việc xấp xỉ một ánh xạ: $X \rightarrow Y$, trong đó X là tập các vấn đề và Y là tập các lời giải tương ứng cho vấn đề đó. Các mẫu (x, y) với $x = (x_1, x_2, \dots, x_n) \in X$, $y = (y_1, y_2, \dots, y_m) \in Y$ được cho trước. Học có giám sát trong các mạng nơ-ron thường được thực hiện theo các bước sau:

B1: Xây dựng cấu trúc thích hợp cho mạng nơ-ron, chẳng hạn có $(n + 1)$ nơ-ron vào (n nơ-ron cho biến vào và 1 nơ-ron cho ngưỡng x_0), m nơ-ron đầu ra, và khởi tạo các trọng số liên kết của mạng.

B2: Đưa một vector x trong tập mẫu huấn luyện X vào mạng

B3: Tính vector đầu ra o của mạng

B4: So sánh vector đầu ra mong muốn y (là kết quả được cho trong tập huấn luyện) với vector đầu ra o do mạng tạo ra; nếu có thể thì đánh giá lỗi.

B5: Hiệu chỉnh các trọng số liên kết theo một cách nào đó sao cho ở lần tiếp theo khi đưa vector x vào mạng, vector đầu ra o sẽ giống với y hơn.

B6: Nếu cần, lặp lại các bước từ 2 đến 5 cho tới khi mạng đạt tới trạng thái hội tụ. Việc đánh giá lỗi có thể thực hiện theo nhiều cách, cách dùng nhiều nhất là sử dụng lỗi tức thời: $Err = (o - y)$, hoặc $Err = |o - y|$; lỗi trung bình bình phương (MSE: mean-square error): $Err = (o - y)^2/2$; Có hai loại lỗi trong đánh giá một mạng nơ-ron. Thứ nhất, gọi là lỗi rõ ràng (apparent error), đánh giá khả năng xấp xỉ các

mẫu huấn luyện của một mạng đã được huấn luyện. Thứ hai, gọi là lỗi kiểm tra (test error), đánh giá khả năng tổng quát hóa của một mạng đã được huấn luyện, tức khả năng phản ứng với các vector đầu vào mới. Để đánh giá lỗi kiểm tra chúng ta phải biết đầu ra mong muốn cho các mẫu kiểm tra.

Thuật toán tổng quát ở trên cho học có giám sát trong các mạng nơ-ron có nhiều cài đặt khác nhau, sự khác nhau chủ yếu là cách các trọng số liên kết được thay đổi trong suốt thời gian học. Trong đó tiêu biểu nhất là thuật toán lan truyền ngược.

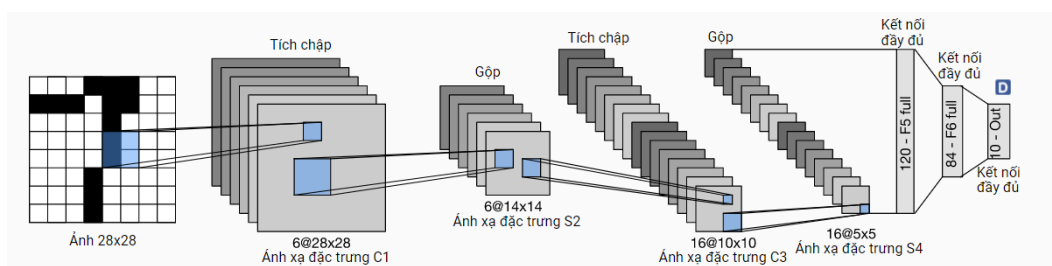
2.1.3.5 Mạng nơ-ron tích chập (CNN)

Mạng nơ-ron tích chập (CNN) là một loại mạng nơ-ron nhân tạo được sử dụng trong nhận dạng và xử lý hình ảnh được thiết kế đặc biệt để xử lý dữ liệu pixel.

CNN là công cụ xử lý hình ảnh mạnh mẽ, trí thông minh nhân tạo (AI) sử dụng học sâu để thực hiện cả tác vụ mô tả và mô tả, thường sử dụng máy tính bao gồm nhận dạng hình ảnh và video, cùng với hệ thống đề xuất và xử lý ngôn ngữ tự nhiên (NLP). Yann LeCun, Leon Bottou, Yosuha Bengio và Patrick Haffner đã đề xuất kiến trúc mạng thần kinh để nhận dạng ký tự viết tay và in bằng máy vào năm 1990 mà họ gọi là LeNet-5. Kiến trúc đơn giản và dễ hiểu, đó là lý do tại sao nó chủ yếu được sử dụng như một bước đầu tiên để giảng dạy.

Một Mạng nơ-ron tích chập bao gồm 3 lớp chính: lớp tích chập (Convolution), lớp gộp (pooling) và lớp kết nối đầy đủ (full-connected).

Một cách đơn giản, ta có thể xem LeNet gồm hai phần: (i) một khối các tầng tích chập; và (ii) một khối các tầng kết nối đầy đủ. Trước khi đi vào các chi tiết cụ thể, hãy quan sát tổng thể mô hình bên dưới

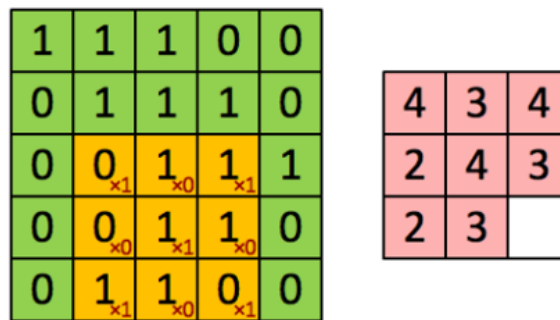


Hình 2.9: Mô phỏng mạng nơ-ron tích chập [10]

2.1.3.6 LỚp tích chậP (Convolution)

Tích chậP đượC sử dụng đầu tiên trong xử lý tín hiệu số (Signal processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kỹ thuật này vào xử lý ảnh và video số.

Để dễ hình dung, ta có thể xem tích chậP như một cửa sổ trượt (sliding window) áp đặt lên một ma trận. Bạn có thể theo dõi cơ chế của tích chậP qua hình minh họa bên dưới.



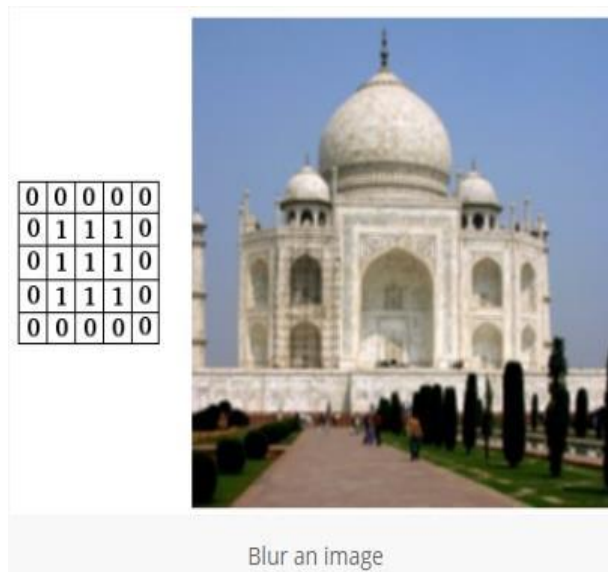
Hình 2.10: Minh họa tích chậP [6]

Ma trận bên trái là một bức ảnh đen trắng. Mỗi giá trị của ma trận tương đương với một điểm ảnh (pixel), 0 là màu đen, 1 là màu trắng (nếu là ảnh grayscale thì giá trị biến thiên từ 0 đến 255).

Sliding window còn có tên gọi là kernel, filter hay feature detector. Ở đây, ta dùng một ma trận filter 3×3 nhân từng thành phần tương ứng (element-wise) với ma trận ảnh bên trái. Giá trị đầu ra do tích của các thành phần này cộng lại.

Kết quả của tích chậP là một ma trận [13] (convolved feature) sinh ra từ việc trượt ma trận filter và thực hiện tích chậP cùng lúc lên toàn bộ ma trận ảnh bên trái. Dưới đây là một vài ví dụ của phép toán tích chậP.

Ta có thể làm mờ bức ảnh ban đầu bằng cách lấy giá trị trung bình của các điểm ảnh xung quanh cho vị trí điểm ảnh trung tâm.



Hình 2. 7: Ảnh mờ sau khi chụp [5]

Về bản chất thực hiện làm mờ ảnh chính là tạo ra ảnh mới sao cho giá trị mức xám của mỗi pixel ở ảnh mới đúng bằng giá trị trung bình của điểm tương ứng và 8 điểm lân cận trên ảnh ban đầu. Nói cách khác, với mỗi điểm trên hình ban đầu, bạn tính giá trị trung bình của nó (tại hàng i cột j) với 8 điểm xung quanh rồi viết lại giá trị mức xám ở vị trí tương ứng (cũng tại hàng i cột j) lên ảnh mới, sau đó tương tự với các điểm tiếp theo.

2	6	8	6	3	4	7	9	
9	0	3	1	2	4	1	4	
1	3	5	2	5	1	3	3	
9	0	8	7	8	9	0	3	
1	3	5	5	3	8	9	3	
6	7	3	2	4	2	1	4	
3	5	5	6	5	4	3	5	
3	6	7	1	2	4	5	6	

tính trung bình các điểm
xung quanh

$$\frac{2+5+1+7+8+9+5+3+8}{9}$$

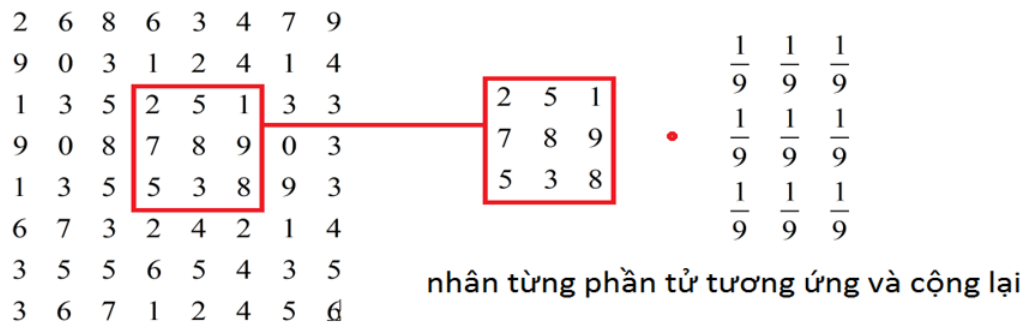
bằng

$$\frac{2}{9} + \frac{5}{9} + \frac{1}{9} + \frac{7}{9} + \frac{8}{9} + \frac{9}{9} + \frac{5}{9} + \frac{3}{9} + \frac{8}{9}$$

Hình 2.8: Mô phỏng số làm mờ ảnh (bước 1) [10]

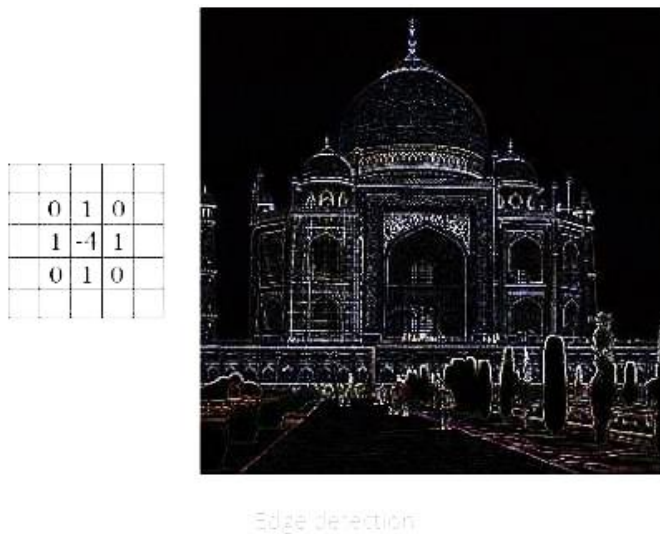
Vậy có nghĩa là ta đang tính trung bình cộng của 9 pixel (pixel tại điểm đó và 8 pixel lân cận), vậy phép tính đó cũng giống như nhân từng giá trị mức sáng của các pixel lân cận với $1/9$ sau đó cộng lại với nhau. Vậy, nếu có một ma trận 3×3 với tất cả các con số trong ma trận đều là $1/9$, ta nhân từng phần tử của ma trận này với mức sáng của pixel tương ứng và cộng lại, ta sẽ có kết quả giống nhau (xem hình vẽ bên dưới).

Áp dụng tương tự cho mọi pixel trên ảnh ban đầu và lấy từng kết quả cho từng pixel của ảnh mới, ta sẽ được ảnh mới chính là ảnh mờ của ảnh ban đầu.



Hình 2.9: Mô phỏng số làm mờ ảnh (bước 2) [10]

Ngoài ra, ta có thể phát hiện biên cạnh bằng cách tính vi phân (độ dị biệt) giữa các điểm ảnh lân cận.

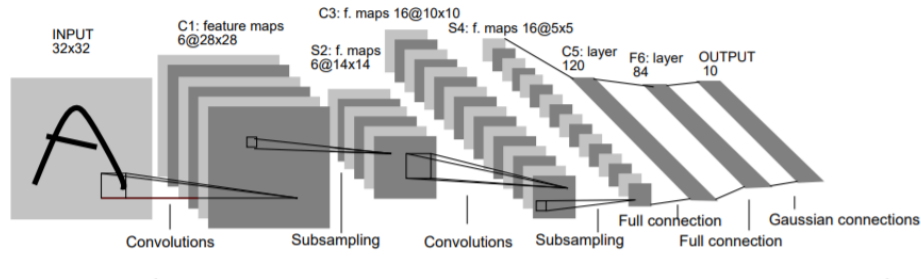


Hình 2.10: Ảnh được phát hiện biên sau khi chụp [7]

2.2 Các công trình liên quan

2.2.1 Mạng LeNet-5 (1998)

Lenet-5 là một trong những [15] mạng CNN đầu tiên được Yann Lecun và các cộng sự phát triển vào năm 1998 để xử lý hình ảnh.



Hình 2.15: Kiến trúc LeNet-5 [3]

Lenet-5 gồm 7 lớp :

Lớp đầu tiên là lớp đầu vào – đây thường không được coi là lớp của mạng vì không có gì được học tại đây. Hiểu theo cách đơn giản đây là lớp đầu vào với kích thước sẽ được đưa về 32x32.

Lớp C1: Đây là lớp conv đầu tiên có 6 đặc trưng với các bước là một. Sử dụng công thức tính thứ nguyên đầu ra của mạng nơ ron tích tụ:

$$n_{out} = \left[\frac{n_{in} + 2p - k}{s} \right] + 1$$

trong đó:

- n_{out} : output of ConvNet
- n_{in} : Input image dimension
- p : padding dimension
- k : kernel dimension
- s : strides size

Công thức tính tham số:

$$n_{param} = (n \times m \times l + 1) \times f$$

Trong đó:

Sử dụng 2 công thức trên người ta có thể tính được kích thước đầu ra và số

- n_{param} : number of parameters
- n : Kernel size dimension 1
- m : Kernel size dimension 2
- l : Input image
- f : Filter size

lượng tham số của mô hình LeNet-5. Chức năng kích hoạt của lớp này là hàm $\tanh(x)$.

Lớp S2: Là lớp gộp trung bình. Lớp này ánh xạ các giá trị trung bình từ lớp chuyển đổi sang lớp chuyển đổi tiếp theo. Lớp Pooling được sử dụng để giảm sự phụ thuộc của mô hình vào vị trí của các đối tượng hơn là hình dạng của các đối tượng. Lớp gộp trong LeNet-5 có kích thước là 2 và bước tiến là 2.

Lớp C3: Là tập thứ hai của lớp phức hợp với 16 bản đồ đặc trưng. Kích thước đầu ra của lớp này là 10 với 2.416 tham số. Chức năng kích hoạt của lớp này là $\tanh(x)$.

Lớp S4: Là một lớp tổng hợp trung bình khác có kích thước là 2 và kích thước bước tiến là 2. Lớp tiếp theo chịu trách nhiệm làm phẳng đầu ra của lớp trước thành mảng 1 chiều. Kích thước đầu ra của lớp này là 400 ($5 \times 5 \times 16$).

Lớp C5: Là một khối dày đặc với 120 kết nối và 48.120 tham số (400×120). Chức năng kích hoạt của khối này là $\tanh(x)$.

Lớp F6: Là một khối dày đặc khác với 84 tham số và 10.164 tham số ($84 \times 120 + 84$). Chức năng kích hoạt của lớp này là $\tanh(x)$.

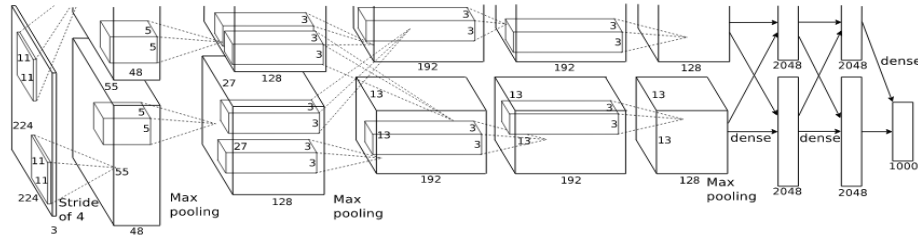
Lớp đầu ra: Có 10 kích thước (bằng số lớp trong cơ sở dữ liệu) với 850 tham số ($10 \times 84 + 10$). Chức năng kích hoạt của lớp này là sigmoid.

2.2.2 AlexNet

AlexNet là một mô hình mạng nơ-ron tích chập được đề xuất vào năm 2012 bởi Alex Krizhevsky.

AlexNet bao gồm 8 lớp (5 lớp tích chập và 3 lớp kết nối đầy đủ). Ma trận lọc (kernel) của AlexNet có khả năng trích xuất các đặc trưng từ các ảnh đơn có kích cỡ

chiều rộng (W) \times chiều cao (H) \times màu (C). AlexNet có khả năng xử lý ảnh RGB (3 lớp) với kích cỡ chuẩn là $224 \times 224 \times 3 = 150.528$ giá trị.



Hình 2.16: Kiến trúc AlexNet [13]

AlexNet là một kiến trúc sâu, đầu vào cho mô hình này là hình ảnh có kích thước $227 \times 227 \times 3$.

Lớp tích chập và lớp gộp tối đa

Ở đây, ta nhận thấy rằng lớp tích chập đầu tiên với 96 bộ lọc có kích thước 11×11 với 4 bước. Chức năng kích hoạt được sử dụng trong lớp này là ReLu. Bản đồ tính năng đầu ra là $55 \times 55 \times 96$.

Trong trường hợp này, cách tính kích thước đầu ra của một lớp tích chập:

$$\text{Output} = ((\text{Input-filter size}) / \text{stride}) + 1$$

Ngoài ra, số lượng bộ lọc sẽ trở thành kênh trong bản đồ tính năng đầu ra.

Ở lớp gộp tối đa đầu tiên, có kích thước 3×3 và bước 2. Bản đồ tính năng kết quả với kích thước $27 \times 27 \times 96$.

Sau đó, trong hình ta ta sẽ nhận thấy sử dụng phép toán tích chập thứ hai. Lần này kích thước bộ lọc được giảm xuống 5×5 và ta có 256 bộ lọc như vậy. Chức năng kích hoạt được sử dụng lại là relu. Bây giờ kích thước đầu ra là $27 \times 27 \times 56$.

Tiếp theo, áp dụng lớp gộp tối đa có kích thước 3×3 với bước 2. Bản đồ tính năng thu về có hình dạng $13 \times 13 \times 56$.

Bây giờ, ta áp dụng phép toán tích chập thứ ba với 384 bộ lọc có kích thước 3×3 . Chức năng kích hoạt được sử dụng là *relu*. Bây giờ kích thước đầu ra vẫn không thay đổi, tức là $13 \times 13 \times 384$.

Sau đó, ta có lớp tích chập cuối cùng có kích thước 3×3 với 256 bộ lọc như vậy. Bản đồ tính năng kết quả có hình dạng $13 \times 13 \times 256$.

Vì vậy, nếu nhìn vào kiến trúc cho đến bây giờ, số lượng bộ lọc đang tăng lên khi ta đi sâu hơn. Do đó, hệ thống sẽ trích xuất nhiều tính năng hơn khi ta đi sâu hơn vào kiến trúc. Ngoài ra, kích thước bộ lọc đang giảm, có nghĩa là bộ lọc ban đầu lớn hơn và khi ta tiếp tục, kích thước bộ lọc sẽ giảm, dẫn đến hình dạng bản đồ đối tượng giảm. Tiếp theo, ta áp dụng lớp gộp tối đa thứ ba có kích thước 3×3 và bước 2. Kết quả là bản đồ đặc trưng của hình dạng $6 \times 6 \times 256$.

AlexNet 1 có khối kích hoạt sử dụng hàm kích hoạt ReLu cho phép giảm thời gian đào tạo 6 lần (so với tanh) khi so sánh cùng độ chính xác. AlexNet 1 sử dụng kỹ thuật “dropout” để có thể bỏ qua các khối (units) trong quá trình đào tạo và chống lại vấn đề “overfitting”. AlexNet 1 được tổ chức và sử dụng lớp “overlap polling” để giảm kích cỡ của mạng và giảm độ lỗi top-1 0.4% và top-5 là 0.3% (so với việc sử dụng pooling có kích thước 2×2 và sai bước 2).

Để xem xét khả năng xử lý của AlexNet 1, ta xem xét số lượng nơ-ron ($N_i = W * H * M$) tại đầu ra lớp thứ i , trọng số ($W_i = K^2 * C * M$) tại lớp thứ i , và số lượng kết nối ($U_i = W * H * K^2$) tại lớp thứ i . Trong đó $W = H = 55$, $C = 3$, M (số đầu ra) = 96, K (kích thước vectơ một chiều của ma trận lọc) = 11. Việc đánh giá kích cỡ (mạng) của 8 lớp được mô tả như sau:

Lớp L1: gồm 96 ma trận lọc (kernels) và mỗi ma trận lọc có kích cỡ $11 \times 11 \times 3$.

- $N_1 = 55 \times 55 \times 96 = 290.400$;
- $P_1 = 96 \times 11 \times 3 = 34.848$;
- $U_1 = 55 \times 11 \times 3 \times 96 = 105.415.200$.

Lớp L2: 256 kernels có kích cỡ $5 \times 5 \times 48$. Lớp gộp (Max pooling) là $5/2 = 2,5$ Vậy ta có:

- $N2 = 272 \times 256 = 186.624$;
- $P2 = 2(52 \times 48 \times 128) = 307.200$;
- $U2 = 223\,948\,800/2 = 111.974.400$.

Lớp L3: 384 kernels có kích cỡ $3 \times 3 \times 256$. Lớp gộp (Max pooling) là $27/2 = 13,5$ Vậy ta có:

- $N3 = 132 \times 384 = 64.896$;
- $P3 = 3 \times 2 \times 256 \times 384 = 884.736$;
- $U3 = 132 \times 3 \times 2 \times 256 \times 384 = 149.520.384$.

Lớp L4: 384 kernels có kích cỡ $3 \times 3 \times 192$. Vậy ta có:

- $N4 = 132 \times 384 = 64.869$;
- $P4 = 2(32 \times 192 \times 128) = 663.552$;
- $U4 = 132 \times 32 \times 384 \times 2/2 = 112.140.288$.

Lớp L5: 256 kernels có kích cỡ $3 \times 3 \times 192$. Vậy ta có:

- $N5 = 132 \times 256 = 43.264$;
- $P5 = 2(32 \times 192 \times 128) = 442.368$;
- $U5 = 132 \times 32 \times 384 \times 256/2 = 74.760.192$.

Lớp L6: 4096 units. Lớp gộp (Max pooling) là $13/2 = 6,5$.

- $N6 = 4096$;
- $P6 = U6 = 6 \times 6 \times 256 \times 4096 = 37.748.736$

Lớp L7:

- $N7 = 4096$ units.
- $P7 = U7 = 4096 \times 4096 = 16.777.216$

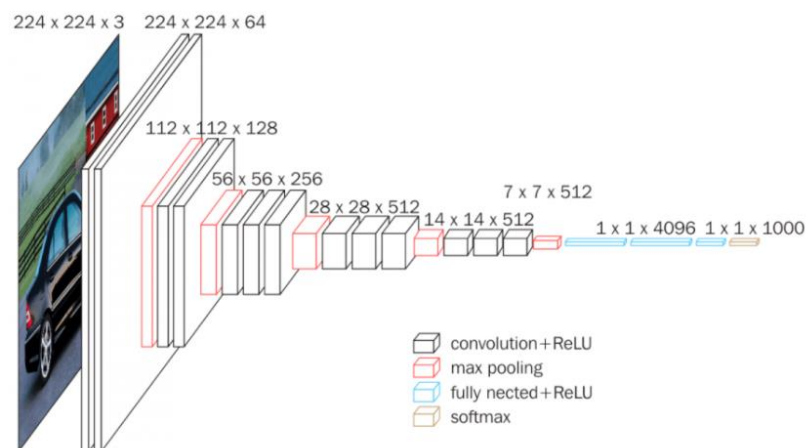
Lớp L8:

- $N8 = 1000$ units.
- $P8 = U8 = 4096 \times 1000 = 4.096.000$

2.2.3 VGG-16

VGG-16 là một mô hình mạng nơ-ron tích chập được đề xuất bởi K. Simonyan và A. Zisserman năm 2014. Mô hình này đạt độ chính xác trong bài kiểm tra top 5 là 92,5% trong ImageNet, đây là một tập dữ liệu của hơn 14 triệu hình ảnh thuộc 1000 lớp. Nó cải tiến so với AlexNet bằng cách thay thế các bộ lọc có kích thước hạt nhân lớn (11 và 5 trong lớp chập đầu tiên và thứ hai, tương ứng) bằng nhiều bộ lọc kích thước hạt nhân 3×3 lần lượt nữa.

Khác với các mạng CNN được nghiên cứu trước đó, VGG-16 được tổ chức chặt chẽ và có số lớp gia tăng một cách đột biến thông qua việc tổ chức các ma trận lọc nhỏ hơn. Cụ thể kiến trúc của VGG-16 được mô tả trong hình 2.2 và các thông số đầu của VGG-16 được chúng tôi tính toán và phân tích chi tiết trong 16 lớp cụ thể như sau:



Hình 2.17: Kiến trúc VGG-16 [15]

Lớp 1 (Tích chập):

- Số bộ lọc: 64
- Kích thước bộ lọc: $3 \times 3 \times 64$
- Bộ nhớ: $224 \times 224 \times 64$

- Số lượng tham số: $(3 \times 3 \times 3) \times 64$

Lớp 2 (Tích chập):

- Đầu vào: $224 \times 224 \times 64$
- Số bộ lọc: 64
- Kích thước bộ lọc: $3 \times 3 \times 64$
- Bộ nhớ: $224 \times 224 \times 64$
- Số lượng tham số: $(3 \times 3 \times 64) \times 64$

Lớp chuyển tiếp sang lớp 3 (Lấy mẫu):

- Kích thước = (2, 2)
- Sai bước = 2
- Độ n = 0
- Bộ nhớ: $112 \times 112 \times 64$
- Kích thước đầu ra của dữ liệu giảm 1/2, từ $(224 \times 224 \times 3)$ xuống $(112 \times 112 \times 3)$, và chiều sâu được giữ nguyên

Lớp 3 (Tích chập):

- Đầu vào: $112 \times 112 \times 3$
- Số bộ lọc: 128
- Kích thước bộ lọc: $3 \times 3 \times 128$
- Bộ nhớ: $112 \times 112 \times 128$
- Số lượng tham số: $(3 \times 3 \times 64) \times 128$

Lớp 4 (Tích chập):

- Đầu vào: $112 \times 112 \times 3$
- Số bộ lọc: 128
- Kích thước bộ lọc: $3 \times 3 \times 128$

- Bộ nhớ: $112 \times 112 \times 128 = 1,6\text{M}$
- Số lượng tham số: $(3 \times 3 \times 128) \times 128$

Lớp chuyển tiếp sang lớp 5 (Lấy mẫu):

- Kích thước = (2, 2)
- Sải bước (Stride) = 2
- Độ đệm (Padding) = 0
- Bộ nhớ: $56 \times 56 \times 128 = 400\text{K}$
- Kích thước đầu ra của dữ liệu giảm 1/2, từ $(112 \times 112 \times 3)$ xuống $(56 \times 56 \times 3)$, và chiều sâu được giữ nguyên

Lớp 5 (Tích chập):

- Đầu vào: $56 \times 56 \times 3$
- Số bộ lọc: 256
- Kích thước bộ lọc: $3 \times 3 \times 256$
- Bộ nhớ: $56 \times 56 \times 256 = 800\text{K}$
- Số lượng tham số: $(3 \times 3 \times 128) \times 256$

Lớp 6 (Tích chập):

- Đầu vào: $56 \times 56 \times 3$
- Số bộ lọc: 256
- Kích thước bộ lọc: $3 \times 3 \times 256$
- Bộ nhớ: $56 \times 56 \times 256 = 800\text{K}$
- Số lượng tham số: $(3 \times 3 \times 256) \times 256$

Lớp 7 (Tích chập):

- Đầu vào: $56 \times 56 \times 3$
- Số bộ lọc: 256

- Kích thước bộ lọc: $3 \times 3 \times 256$
- Bộ nhớ: $56 \times 56 \times 256$
- Số lượng tham số: $(3 \times 3 \times 256) \times 256$

Lớp chuyển tiếp sang lớp 8 (Lấy mẫu):

- Kích thước = (2, 2)
- Sai bước = 2
- Độ n = 0
- Bộ nhớ: $28 \times 28 \times 256$
- Kích thước đầu ra của dữ liệu giảm 1/2, từ $(56 \times 56 \times 3)$ xuống $(28 \times 28 \times 3)$, và chiều sâu được giữ nguyên

Lớp 8 (Tích chập):

- Đầu vào: $28 \times 28 \times 3$
- Số bộ lọc: 512
- Kích thước bộ lọc: $3 \times 3 \times 512$
- Bộ nhớ: $28 \times 28 \times 512$
- Số lượng tham số: $(3 \times 3 \times 256) \times 512$

Lớp 9 (Tích chập):

- Đầu vào: $28 \times 28 \times 3$
- Số bộ lọc: 512
- Kích thước bộ lọc: $3 \times 3 \times 512$
- Bộ nhớ: $28 \times 28 \times 512$
- Số lượng tham số: $(3 \times 3 \times 512) \times 512$

Lớp 10 (Tích chập):

- Đầu vào: $28 \times 28 \times 3$

- Số bộ lọc: 512
- Kích thước bộ lọc: $3 \times 3 \times 512$
- Bộ nhớ: $28 \times 28 \times 512$
- Số lượng tham số: $(3 \times 3 \times 512) \times 512$

Lớp chuyển tiếp sang lớp 11 (Lấy mẫu):

- Kích thước = (2, 2)
- Sai bước = 2
- Độ n = 0
- Bộ nhớ: $14 \times 14 \times 512$
- Kích thước đầu ra của dữ liệu giảm 1/2, từ $(28 \times 28 \times 3)$ xuống $(14 \times 14 \times 3)$, và chiều sâu được giữ nguyên

Lớp 11 (Tích chập):

- Đầu vào: $14 \times 14 \times 3$
- Số bộ lọc: 512
- Kích thước bộ lọc: $3 \times 3 \times 512$
- Bộ nhớ: $14 \times 14 \times 512 = 100K$
- Số lượng tham số: $(3 \times 3 \times 512) \times 512$

Lớp 12 (Tích chập):

- Đầu vào: $14 \times 14 \times 3$
- Số bộ lọc: 512
- Kích thước bộ lọc: $3 \times 3 \times 512$
- Bộ nhớ: $14 \times 14 \times 512 = 100K$
- Số lượng tham số: $(3 \times 3 \times 512) \times 512$

Lớp 13 (Tích chập):

- Đầu vào: $14 \times 14 \times 3$
- Số bộ lọc: 512
- Kích thước bộ lọc: $3 \times 3 \times 512$
- Bộ nhớ: $14 \times 14 \times 512 = 100\text{K}$
- Số lượng tham số: $(3 \times 3 \times 512) \times 512$

Lớp chuyển tiếp sang lớp 14 (Lấy mẫu):

- Kích thước = (2, 2)
- Sai bước = 2
- Độ n = 0
- Bộ nhớ: $7 \times 7 \times 512 = 25\text{K}$
- Kích thước đầu ra của dữ liệu giảm 1/2, từ $(14 \times 14 \times 3)$ xuống $(7 \times 7 \times 3)$, và chiều sâu được giữ nguyên

Lớp 14 (Kết nối đầy đủ):

- Đầu vào: $1 \times 1 \times 4.096$
- Bộ nhớ: 4.096K
- Số lượng tham số: $7 \times 7 \times 512 \times 4.096$

Lớp 15 (Kết nối đầy đủ):

- Đầu vào: $1 \times 1 \times 4.096$
- Bộ nhớ: 4.096K
- Số lượng tham số: 4.096×4.096

Lớp 16 (Kết nối đầy đủ):

- Đầu vào: $1 \times 1 \times 4.096$
- Bộ nhớ: 1.000K

Số lượng tham số: 4.096×1.000

Theo như hình 2.16, ta có thể thấy đầu vào của lớp cov1 có kích thước cố định 224 x 224 RGB. Hình ảnh được chuyển qua một chồng các lớp tích chập, trong đó các bộ lọc được sử dụng với các trường tiếp cận rất nhỏ: 3 x 3 (là kích thước nhỏ nhất để nắm bắt khái niệm trái/phải, lên/xuống, trung tâm).

Một trong các cấu hình, nó cũng sử dụng bộ lọc tích chập 1 x 1, có thể được xem như một phép biến đổi tuyến tính của các kênh đầu vào (theo sau là không tuyến tính). Sải tích chập được cố định thành 1 pixel; phần đệm không gian đầu vào lớp tích chập sao cho độ phân giải không gian được giữ nguyên sau khi tích chập, phần đệm là 1 pixel cho lớp tích chập 3 x 3. Tổng hợp theo không gian được thực hiện tối đa 5 tầng gộp (pooling layer), theo sau một số của lớp tích chập (không phải tất cả các lớp tích chập đều được gộp tối đa). Gộp tối đa (max-pooling) được thực hiện trên cửa sổ 2 x 2 với bước 2.

Ba, lớp kết nối đầy đủ (fully-connected) tuân theo một chồng các lớp tích chập (có độ sâu khác nhau trong các kiến trúc khác nhau): hai lớp đầu tiên có 4096 kênh mỗi lớp, lớp thứ ba thực hiện phân loại ILSVRC 1000 chiều và do đó chứa 1000 kênh.

Lớp cuối cùng là lớp Soft-max, cấu hình của lớp được kết nối đầy đủ là giống nhau trong tất cả các mạng.

Tất cả các lớp ẩn được trang bị tính phi tuyến tính (ReLU). Cùng cần lưu ý rằng không có mạng nào chuẩn hóa phản hồi cục bộ (LRN), việc chuẩn hóa như vậy không cải tiến hiệu suất trên tập dữ liệu ILSVRC, nhưng dẫn đến tiêu thụ bộ nhớ và thời gian tính toán tăng lên.

2.3 Kết luận chương

Sau khi phân tích, làm rõ các khái niệm liên quan đến mạng nơ-ron tích chập và mô hình học tích cực, chúng tôi tiếp tục đề cập đến 3 mạng nơ-ron tích chập phổ biến. Mỗi mạng nơ-ron có ưu và nhược điểm riêng.

- Thứ nhất là mạng LeNet-5: Mạng này khá đơn giản, LeNet-5 có 7 lớp. trong đó có ba lớp chập (C1, C3 và C5) và hai lớp gộp (S2 và S4), và 1 lớp được kết

nối đầy đủ (F6), tiếp theo là lớp đầu ra. LeNet-5 sử dụng hàm sigmoid (tanh) ở mỗi lớp tích chập. Chính vì thế, tốc độ tính toán của LeNet-5 chậm. Tuy vậy, một số bài toán sử dụng mô hình LeNet-5 có thể đạt được độ chính xác cực cao tới 99,05% chẳng hạn như bài toán phát hiện chữ số (0-9).

- Thứ hai là mạng AlexNet: AlexNet bao gồm 8 lớp (5 lớp tích chập và 3 lớp kết nối đầy đủ). AlexNet có một số đặc điểm của AlexNet cho phép giảm thời gian đào tạo 6 lần (so với tanh) khi so sánh cùng độ chính xác. AlexNet có cấu trúc tương tự như LeNet, nhưng vì sử dụng nhiều tầng tích chập hơn, do vậy AlexNet có khả năng xử lý bộ dữ liệu với tham số truyền vào lớn hơn LeNet. Tuy nhiên, chính vì sử dụng nhiều tầng tích chập nên số lượng tham số cần xử lý là rất lớn. Chính vì thế cần phải đưa vào một số kỹ thuật khắc phục những lỗi này như “Dropout” và tiền xử lý.

- Thứ ba là mạng VGG-16: VGG-16 có cửa sổ tích chập nhỏ (3×3). Chính vì sử dụng cửa sổ tích chập nhỏ nên rõ ràng VGG-16 cho hiệu quả cao hơn mạng có cửa sổ tích chập rộng nhưng ít lớp.

CHƯƠNG 3: MÔ HÌNH NHẬN BIẾT CHỮ KÝ VIẾT TAY

3.1 Mô phỏng chương trình nhận biết chữ ký viết tay

3.1.1 Giới thiệu chung

Chương trình nhận dạng chữ ký viết tay dựa trên phương thức học có giám sát (Supervised Learning) theo dạng phân loại (Classification). Theo đó dữ liệu đầu vào sẽ là các hình ảnh chữ ký thật và giả mạo của người dùng đã được gán nhãn cho tập training. Sau quá trình training sẽ được model sử dụng cho việc nhận dạng chữ ký số viết tay với độ chính xác ổn định.

3.1.2 Công cụ sử dụng

- Công cụ: Jupyter Notebook. Trước đây là nó có tên IPython Notebook, đến năm 2014 đổi lại thành Jupyter Notebook. Jupyter hỗ trợ rất nhiều các kernel cho các ngôn ngữ khác nhau, khoảng trên 40 ngôn ngữ trong đó có Python. Việc đổi tên từ IPython sang Jupyter cũng là vì mục đích hỗ trợ đa ngôn ngữ. Phần cơ bản của nó là một ứng dụng chạy trên nền web cho phép chạy Interactive Python (hay IPython), bạn có thể đưa cả code Python và các thành phần văn bản phức tạp như hình ảnh, công thức, video, biểu thức... vào trong cùng một file giúp cho việc trình bày trở lên dễ hiểu, giống như một file trình chiếu nhưng lại có thể thực hiện chạy code tương tác trên đó, cốt lõi của việc này chính là Markdown. Các file "notebook" này có thể được chia sẻ với mọi người và có thể thực hiện lại các công đoạn một cách nhanh chóng và chính xác như những gì bạn đã làm trong quá trình tạo ra file.
- Cài đặt công cụ Jupyter Notebook: Cài đặt Python => Cài đặt Jupyter Notebook bằng lệnh: `pip install jupyter`.
- Tập dữ liệu: tập chữ ký của người Việt thu thập được 160 mẫu tương ứng với 160 người. Tỷ lệ trong 1 mẫu là 24 chữ ký thật và 30 chữ ký giả.
- Tính đủ và độ tin cậy của tập dữ liệu thử nghiệm: Phương án của em là tiên xử lý dữ liệu thu thập của 160 chữ ký của người Việt. Do không đủ thời gian nên

em mới thu thập được dữ liệu của 160 người, hướng tương lai về sau em sẽ sưu tập thêm dữ liệu nhiều hơn để cải thiện độ chính xác.

3.2 Môi trường mô phỏng thực nghiệm

Môi trường:

- Chương trình được thực nghiệm trên Windows 10, máy tính laptop core i7 tốc độ 3.4GHz, bộ nhớ RAM 8GB. Sử dụng bộ thư viện sklearn, tensorflow, keras, cv2 phiên bản 2.7 và các thư viện hỗ trợ khác như: os, time, numpy, matplotlib. Đây đều là các thư viện mã nguồn mở, sử dụng ổn định trên Python 3.8.

Mô phỏng thực nghiệm:

- Mô tả: Đưa tập dữ liệu chữ ký vào mô hình mạng CNN để lấy ra một cặp chữ ký và dựa vào ngưỡng trên khoảng cách để xác định hình ảnh sau là chữ ký thật hay chữ ký giả so với ảnh gốc.

Thực nghiệm và kết quả mô phỏng:

Bước 1: Import các thư viện cần thiết

```
import sys
import numpy as np
import pickle
import os
import matplotlib.pyplot as plt

import cv2
import time
import itertools
import random

from sklearn.utils import shuffle

import tensorflow as tf
from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, RMSprop
from keras.layers import Conv2D, ZeroPadding2D, Activation, Input, concatenate, Dropout
from keras.models import Model

from tensorflow.keras.layers import BatchNormalization
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import Concatenate
from keras.layers.core import Lambda, Flatten, Dense
from keras.initializers import glotot_uniform

#from keras.engine.topology import Layer
from tensorflow.keras.layers import Layer
from keras.regularizers import l2
from keras import backend as K
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLRonPlateau
```

Hình 3.1: Quá trình import thư viện

- Bộ thư viện sklearn, tensorflow, keras, cv2 đều là các thư viện mã nguồn mở và sử dụng ổn định trên Python với các phiên bản. Thư viện keras cung cấp các hàm để tạo nên mô hình mạng CNN.
- Thư viện tensorflow, keras, sklearn đều là phiên bản 2.7.

- ✚ **Bước 2:** Đưa các hình ảnh từ tập dữ liệu thu thập vào thành 2 nhóm: `forg_groups` (chữ ký giả) và `orig_groups` (chữ ký thật). Đồng thời xử lý ảnh trước khi đưa vào model để tranning.

```
#path = "./BHSig260/Hindi/"
path = "./BHSig260/Vietnam/"

# Lấy ra danh sách các thư mục và sắp xếp
dir_list = next(os.walk(path))[1]
dir_list.sort()

# Mỗi người tách biệt chữ ký thật và chữ ký giả
# Chữ ký thật đc Lưu vào danh sách "orig_groups"
# Chữ ký giả đc Lưu vào danh sách "forged_groups"
orig_groups, forg_groups = [], []
for directory in dir_list:
    images = os.listdir(path+directory)
    images.sort()
    images = [path+directory+'/' + x for x in images]
    forg_groups.append(images[:30]) # First 30 signatures in each folder are forged
    orig_groups.append(images[30:]) # Next 24 signatures are genuine
```

Hình 3.2 Chia dữ liệu đầu vào thành hai nhóm thật và giả

```
# All the images will be converted to the same size before processing
img_h, img_w = 155, 220

# hàm có chức năng chọn ngẫu nhiên một chữ ký từ tập train và thiết lập ra 2 bản sao thật và giả
def visualize_sample_signature():
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize = (10, 10))
    k = np.random.randint(len(orig_train))
    orig_img_names = random.sample(orig_train[k], 2)
    forg_img_name = random.sample(forg_train[k], 1)
    orig_img1 = cv2.imread(orig_img_names[0], 0)
    orig_img2 = cv2.imread(orig_img_names[1], 0)
    forg_img = plt.imread(forg_img_name[0], 0)
    orig_img1 = cv2.resize(orig_img1, (img_w, img_h))
    orig_img2 = cv2.resize(orig_img2, (img_w, img_h))
    forg_img = cv2.resize(forg_img, (img_w, img_h))

    ax1.imshow(orig_img1, cmap = 'gray')
    ax2.imshow(orig_img2, cmap = 'gray')
    ax3.imshow(forg_img, cmap = 'gray')

    ax1.set_title('Genuine Copy')
    ax1.axis('off')
    ax2.set_title('Genuine Copy')
    ax2.axis('off')
    ax3.set_title('Forged Copy')
    ax3.axis('off')
```

Hình 3.3: Hàm lấy ra hai chữ thật và một chữ ký giả từ bộ dữ liệu đã được chia

- Bước 3:** Tạo dữ liệu theo kích thước `batch_size` và tạo ra các cặp chữ ký thật - chữ ký thật và chữ ký thật- chữ ký giả. Với mỗi một bộ có 24 chữ ký thật thì chọn ra 2 chữ ký thì có 276 cặp chữ ký thật - chữ ký thật. Đối với cặp chữ ký thật - chữ ký giả thì lấy cặp chữ ký thật của 1 người với 12 chữ ký giả của người đó nên có 300 bộ.

```
In [14]: # hàm tạo dữ liệu theo kích thước batch_size và một nửa dữ liệu là thật và một nửa dữ liệu còn lại là giả
def generate_batch(orig_groups, forg_groups, batch_size = 32):
    while True:
        orig_pairs = []
        forg_pairs = []
        gen_gen_labels = []
        gen_for_labels = []
        all_pairs = []
        all_labels = []

        # Here we create pairs of Genuine-Genuine image names and Genuine-Forged image names
        # For every person we have 24 genuine signatures, hence we have
        # 24 choose 2 = 276 Genuine-Genuine image pairs for one person.
        # To make Genuine-Forged pairs, we pair every Genuine signature of a person
        # with 12 randomly sampled Forged signatures of the same person.
        # Thus we make 24 * 12 = 300 Genuine-Forged image pairs for one person.
        # In all we have 120 person's data in the training data.
        # Total no. of Genuine-Genuine pairs = 120 * 276 = 33120
        # Total number of Genuine-Forged pairs = 120 * 300 = 36000
        # Total no. of data points = 33120 + 36000 = 69120
        for orig, forg in zip(orig_groups, forg_groups):
            orig_pairs.extend(list(itertools.combinations(orig, 2)))
            for i in range(len(forg)):
                forg_pairs.extend(list(itertools.product(orig[i:i+1], random.sample(forg, 12))))

        # Label for Genuine-Genuine pairs is 1
        # Label for Genuine-Forged pairs is 0
        gen_gen_labels = [1]*len(orig_pairs)
        gen_for_labels = [0]*len(forg_pairs)

        # Concatenate all the pairs together along with their labels and shuffle them
        all_pairs = orig_pairs + forg_pairs
        all_labels = gen_gen_labels + gen_for_labels
        del orig_pairs, forg_pairs, gen_gen_labels, gen_for_labels
        all_pairs, all_labels = shuffle(all_pairs, all_labels)

        # Note the lists above contain only the image names and
        # actual images are loaded and yielded below in batches
        # Below we prepare a batch of data points and yield the batch
        # In each batch we load "batch_size" number of image pairs
        # These images are then removed from the original set so that
        # they are not added again in the next batch.

        k = 0
        pairs=[np.zeros((batch_size, img_h, img_w, 1)) for i in range(2)]
        targets=np.zeros((batch_size,))
        for ix, pair in enumerate(all_pairs):
            img1 = cv2.imread(pair[0], 0)
            img2 = cv2.imread(pair[1], 0)
            img1 = cv2.resize(img1, (img_w, img_h))
            img2 = cv2.resize(img2, (img_w, img_h))
            img1 = np.array(img1, dtype = np.float64)
            img2 = np.array(img2, dtype = np.float64)
            img1 /= 255
            img2 /= 255
            img1 = img1[... , np.newaxis]
            img2 = img2[... , np.newaxis]
            pairs[0][k] = img1
            pairs[1][k] = img2
            k += 1
            if k == batch_size:
                yield pairs, targets
                break
```

Hình 3.4: Hàm tạo các cặp dữ liệu theo kích thước `batch_size`

- Bước 4:** Mô hình CNN
 - `Sequential()`: khởi tạo model.
 - Lớp `Convolution2D` với 96 bộ lọc, kích thước bộ lọc `11x11`, `activation='relu'` lọc ra các giá trị nhỏ hơn 0. Tốc độ hội tụ và tính toán nhanh hơn các hàm khác.

- Lớp MaxPooling để lấy ra mẫu con sau lớp tích chập với stride = (2,2).
- Lớp BatchNormalization chuẩn hóa các feature (đầu ra của mỗi layer sau khi đi qua các activation) về trạng thái zero-mean với độ lệch chuẩn 1.
- Lớp ZeroPadding2D dùng để padding trên hình ảnh.
- Lớp Dropout dùng để bỏ qua 1 vài unit trong quá trình training và mục đích chính là để chống over-fitting.
- Lớp Flatten là lớp reshape trong đó tất cả các axes được làm phẳng hoặc được ghép lại với nhau.
- Lớp Dense thể hiện việc tất cả các unit của layer trước được nối toàn bộ với các unit của hiện tại

```
# Mô hình CNN
def create_base_network_signet(input_shape):
    seq = Sequential()
    seq.add(Conv2D(96, kernel_size=(11, 11), activation='relu', name='conv1_1', strides=4, input_shape= input_shape, kernel_initializer='glorot_uniform'))
    seq.add(BatchNormalization(epsilon=1e-06, axis=1, momentum=0.9))
    seq.add(MaxPooling2D((3,3), strides=(2, 2)))
    seq.add(ZeroPadding2D((2, 2), data_format="channels_last"))

    seq.add(Conv2D(128, kernel_size=(5, 5), activation='relu', name='conv2_1', strides=1, kernel_initializer='glorot_uniform'))
    seq.add(BatchNormalization(epsilon=1e-06, axis=1, momentum=0.9))
    seq.add(MaxPooling2D((3,3), strides=(2, 2)))
    seq.add(Dropout(0.3))# added extra
    seq.add(ZeroPadding2D((1, 1), data_format="channels_last"))

    seq.add(Conv2D(256, kernel_size=(3, 3), activation='relu', name='conv3_1', strides=1, kernel_initializer='glorot_uniform'))
    seq.add(ZeroPadding2D((1, 1), data_format="channels_last"))

    seq.add(Conv2D(128, kernel_size=(3, 3), activation='relu', name='conv3_2', strides=1, kernel_initializer='glorot_uniform'))
    seq.add(MaxPooling2D((3,3), strides=(2, 2)))
    seq.add(Dropout(0.3))# added extra
    seq.add(Flatten(name='flatten'))
    seq.add(Dense(512, activation='relu', kernel_initializer='glorot_uniform'))
    seq.add(Dropout(0.5))
    seq.add(Dense(1))

    return seq
```

Hình 3.5: Mô hình CNN

Bước 5: Training model

Chọn Epoch = 10 là số lần chạy để đưa độ chính xác cao nhất.

```
#training dữ liệu
# compile model using RMSProp Optimizer and Contrastive Loss function defined above
rms = RMSprop(lr=1e-4, rho=0.9, epsilon=1e-08)
#model.compile(loss=contrastive_loss, optimizer=rms)
model.compile(optimizer=rms, loss='categorical_crossentropy', metrics=['accuracy'])
#model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

c:\users\hoangnam\appdata\local\programs\python\python38\lib\site-packages\keras\optimizer_v2\rmsprop.py:130: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super(RMSprop, self).__init__(name, **kwargs)

results = model.fit(generate_batch(orig_train, forg_train, batch_sz),
                    steps_per_epoch = num_train_samples//batch_sz,
                    epochs = 10,
                    validation_data = generate_batch(orig_val, forg_val, batch_sz),
                    validation_steps = num_val_samples//batch_sz
                    )
```

Hình 3.6: Tranning model

✚ Bước 6: Xây dựng hàm test dựa vào ngưỡng trên khoảng cách

Dựa vào ngưỡng trên khoảng cách, hình ảnh lấy được so sánh với hình ảnh gốc. Nếu ngưỡng xác định được lớn hơn ngưỡng sau khi tranning model thì đó là chữ ký giả và ngược lại là chữ ký thật.

```
# tính toán độ chính xác và ngưỡng trên khoảng cách
def compute_accuracy_roc(predictions, labels):
    dmax = np.max(predictions)
    dmin = np.min(predictions)
    nsame = np.sum(labels == 1)
    ndiff = np.sum(labels == 0)

    step = 0.01
    max_acc = 0
    best_thresh = -1

    for d in np.arange(dmin, dmax+step, step):
        idx1 = predictions.ravel() <= d
        idx2 = predictions.ravel() > d

        tpr = float(np.sum(labels[idx1] == 1)) / nsame
        tnr = float(np.sum(labels[idx2] == 0)) / ndiff
        acc = 0.5 * (tpr + tnr)
        # print ('ROC', acc, tpr, tnr)

        if (acc > max_acc):
            max_acc, best_thresh = acc, d

    return max_acc, best_thresh

test_gen = generate_batch(orig_test, forg_test, 1)
pred, tr_y = [], []
for i in range(num_test_samples):
    (img1, img2), label = next(test_gen)
    tr_y.append(label)
    pred.append(model.predict([img1, img2])[0][0])

# đưa ra độ chính xác và ngưỡng xác định thật giả dựa vào model sau khi trainin
tr_acc, threshold = compute_accuracy_roc(np.array(pred), np.array(tr_y))
tr_acc, threshold
#Nếu lớn hơn threshold thì là chữ ký giả
```

Hình 3.7: Tính toán ngưỡng và độ chính xác sau khi train model

Kết quả: độ chính xác lên đến 100% và ngưỡng là 0.0 do tập dữ liệu thu thập được ở phần chữ ký thật là 24 hình ảnh giống nhau

✚ Bước 7: Xây dựng hàm kiểm tra chữ ký thật giả dựa vào ngưỡng sau khi train model ở bước 6

```
# hàm demo kiểm tra chữ ký thật giả
def predict_score():
    test_point, test_label = next(test_gen)
    img1, img2 = test_point[0], test_point[1]

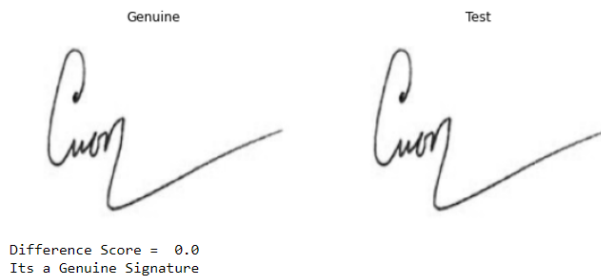
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 10))
    ax1.imshow(np.squeeze(img1), cmap='gray')
    ax2.imshow(np.squeeze(img2), cmap='gray')
    ax1.set_title('Genuine')
    if test_label == 1:
        ax2.set_title('Test')
        #ax2.set_title('Forged')
    else:
        ax2.set_title('Test')
        #ax2.set_title('Genuine')
    ax1.axis('off')
    ax2.axis('off')
    plt.show()
    result = model.predict([img1, img2])
    diff = result[0][0]
    print("Difference Score = ", diff)
    if diff > threshold:
        print("Its a Forged Signature")
        #print("Its a Genuine Signature")
    else:
        print("Its a Genuine Signature")
        #print("Its a Forged Signature")
```

Hình 3.8: Hàm kiểm tra chữ ký thật giả

Kết quả mô phỏng:

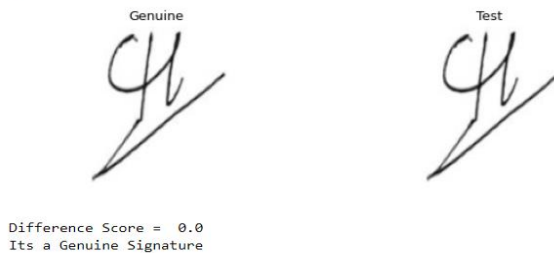
- Chữ ký thật

In [108]: predict_score()

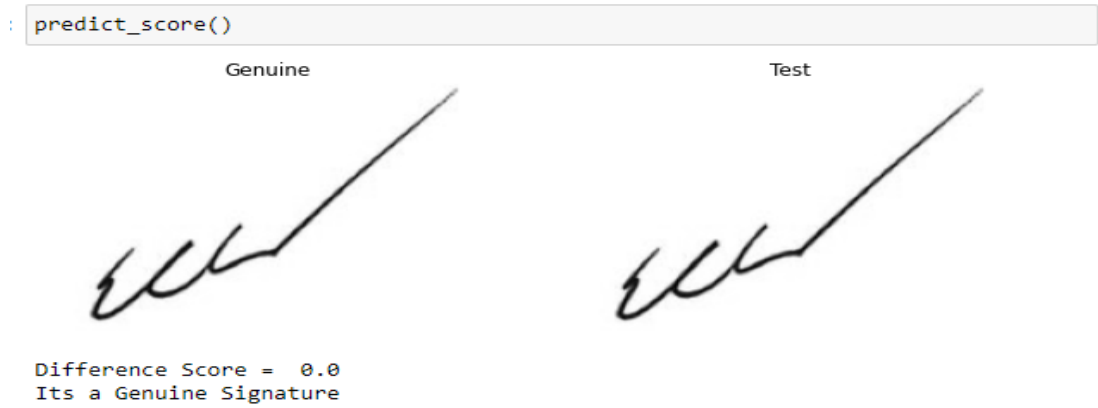


Hình 3.9: Kết quả chữ ký thật lần 1

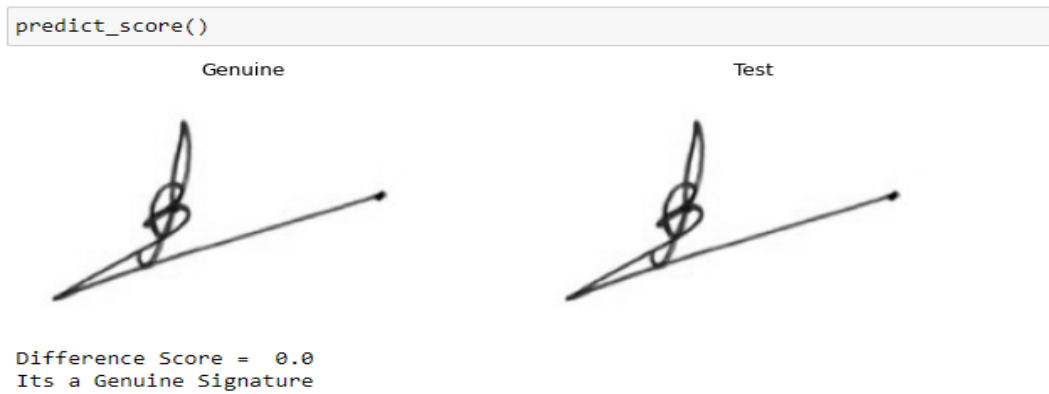
In [29]: predict_score()



Hình 3.10: Kết quả chữ ký thật lần 2



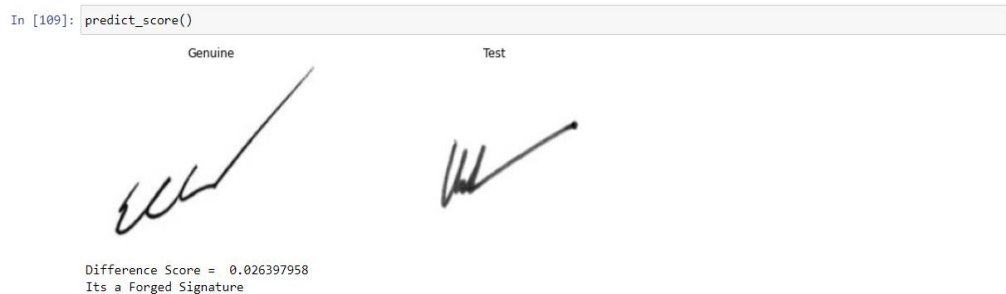
Hình 3.11: Kết quả chữ ký thật lần 3



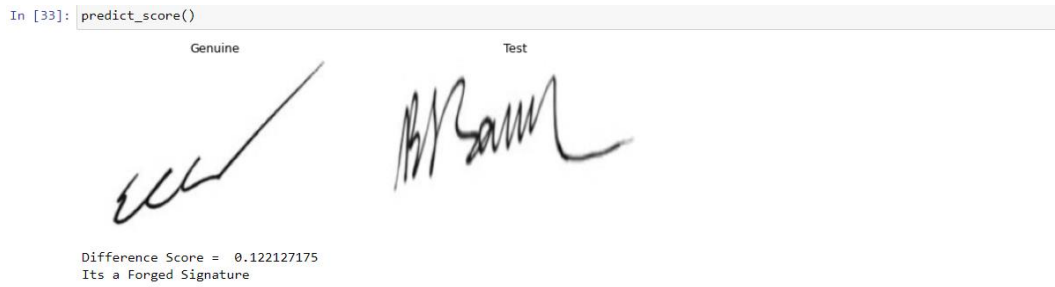
Hình 3.12: Kết quả chữ ký thật lần 4

Nhận xét: do đầu vào chữ ký thật là 24 hình ảnh giống nhau lên ngưỡng trên khoảng cách luôn là 0.0. Ngưỡng trên khoảng cách luôn ≥ 0.0 .

- Chữ ký giả



Hình 3.14: Kết quả chữ ký giả lần 1



Hình 3.15 Kết quả chữ ký giả lần 2



Hình 3.16: Kết quả chữ ký giả lần 3



Hình 3.17: Kết quả chữ ký giả lần 4

Nhận xét: Ngưỡng trên khoảng cách thay đổi 1 cách rõ rệt đối với những chữ ký gần giống hoặc khác hẳn so với chữ ký thật ban đầu.

Kết luận:

Từ những kết quả ở trên, ta thấy rằng mô hình CNN đạt được độ chính xác cao trong việc nhận dạng chữ ký so với các mô hình khác.

Mô hình CNN được huấn luyện với tốc độ học $lr = 1e-4$, sử dụng các hàm kích hoạt relu ở các lớp tích chập và điều chỉnh được các tham số trong các lớp mạng để có được độ chính xác cao nhất.

Để đánh giá độ chính xác, em xây dựng training model để tạo ra hàm test dựa vào ngưỡng trên khoảng cách. Dựa vào ngưỡng trên khoảng cách, hình ảnh lấy được so sánh với hình ảnh gốc. Nếu ngưỡng xác định được lớn hơn ngưỡng sau khi training model thì đó là chữ ký giả và ngược lại là chữ ký thật. Độ chính xác nằm trong khoảng 0.0 đến 0.01 (độ chính xác giảm dần từ 0.0 \rightarrow 0.01) thì được xem là chữ ký thật ngược lại thì nó là chữ ký giả.

3.3 Kết luận chương

Qua đây có thể thấy được việc training model phục vụ cho việc nhận dạng chữ ký số. Có thể sử dụng model này cho những mục đích lớn hơn sau này. Độ chính xác của dự đoán chỉ mang tính chất tương đối chưa thể đạt được độ chính xác lý tưởng (100%).

Qua đề tài nghiên cứu nội dung chính là giải quyết bài toán nhận dạng ở đây là chữ ký viết tay. Với mạng nơ-ron nhân tạo chúng ta có thể thấy được mức độ nhận biết cũng như độ nhạy trong dự đoán. Sử dụng mô hình CNN để training model giúp cải thiện độ chính xác hơn, cải thiện độ tin cậy hơn so với các mô hình machine learning trước đây.

Phương pháp	Độ chính xác
Nearest-neighbor	75%
LIBSVM	77%
1-layer Neural nets	79%
CNN	80%

Từ bảng so sánh [16] trên ta nhận thấy mô hình CNN đưa ra kết quả với độ chính xác cao nhất.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Qua luận văn này, Chúng tôi không những phân tích mô hình, thuật toán, và dữ liệu mà còn khảo sát, đánh giá các lớp, khối hàm, và các kỹ thuật trong mạng nơ-ron tích chập hiện đại. Việc này đáp ứng được cho sự thay đổi kiến trúc, thuật toán có thể giải quyết các bài toán đơn giản, phù hợp với các ứng dụng phân loại chữ ký viết tay trong thực tế.

Nội dung của luận văn thực hiện bao gồm việc cài đặt thuật toán, đánh giá mô hình và phân tích các tập dữ liệu huấn luyện và thử nghiệm. Thông qua một số kịch bản mô phỏng và đánh giá hiệu năng thực thi của hệ thống, các gợi mở cho việc tối ưu một hệ thống thực dựa trên tập mã nguồn mở đã sẵn sàng triển khai cho một hệ thống phân loại các chữ ký tiếng anh hoặc tiếng việt trong thực tế trên một nền tảng học sâu.

Chúng tôi đã thử nghiệm với một số sự thay đổi về các nhiệm vụ xác minh chữ ký. Chúng tôi đã chỉ ra rằng các công trình mạng nơ-ron phức tạp làm việc tốt trong việc xác minh chữ ký trong quá trình đào tạo các ví dụ về chữ ký thật và giả mạo của cùng một người có chữ ký được nhìn thấy tại thời điểm thử nghiệm. Sau đó, chúng tôi đã tiến hành một số thí nghiệm của mình trên chữ ký của những người mới có chữ ký không được nhìn thấy trong quá trình đào tạo cũng cho ra kết quả nhận dạng chữ ký cao. Thêm vào đó, đề xuất phương pháp học tích cực để hệ thống phân loại nhận biết chữ ký thật giả tốt, phù hợp với thực tế nhu cầu hơn. Những dữ liệu về chữ ký viết tay ở Việt Nam còn nhỏ không đủ lớn, cần thời gian bổ sung, cũng như nhu cầu công nghệ hiện đại trong chữ ký số cho các doanh nghiệp, cá nhân thì tiếp cận kiểu bán giám sát học tích cực này là hoàn toàn phù hợp và đúng với xu hướng công nghệ 4.0, và cũng giải quyết cho bài toán nhận dạng chữ ký viết tay dựa theo đó có thể giải quyết được vấn đề xác thực số, khác so với sinh trắc học và face ID,... Chữ ký viết tay gắn liền với đời sống, là phương tiện xác thực có ý nghĩa cao trên tất cả các văn bản hiện hành.

Trong luận văn này, các kết quả cũng đưa ra được độ nhận diện chữ ký viết tay bằng CNN có độ chính xác cao.

Để xây dựng nhận dạng chữ ký số viết tay bằng CNN thì em đã nghiên cứu các thuật toán nhận dạng trước đó, và từ đó xây dựng đưa ra thuật toán cải tiến để nâng cao độ chính xác cao cho nhận dạng chữ ký.

Các bài báo trước đó tập trung nhận dạng chữ ký nước ngoài. Trong đề tài này em tập trung nhận dạng chữ ký người Việt tiếng Việt. Và tập dữ liệu chữ ký Việt Nam rất ít nên phải tìm sưu tầm các chữ ký lại để đưa vào bộ thư viện

Hướng nghiên cứu trong tương lai của chúng tôi là hoàn thiện mô hình lý thuyết nghiên cứu, hiệu chỉnh các thuật toán, các lớp, hàm, hệ số lọc bỏ phù hợp để ứng dụng có độ chính xác cao hơn khi khảo sát hình ảnh các chữ ký viết tay thực tế ở Việt Nam. Nâng cao hiệu quả chương trình, nhận diện nhiều ký tự cùng lúc. Phát triển chương trình thành module phần cứng. Thêm vào đó, chúng tôi sẽ xem xét phát triển tập dữ liệu phù hợp với khả năng triển khai cho các ứng dụng nhận diện chữ ký viết tay cho nhu cầu, xu hướng hiện đại hóa, công nghệ 4.0 ở Việt Nam.

Mặt khác, Chúng tôi đề xuất hướng cụ thể cho công việc trong tương lai là việc tiếp cận nhiều tài nguyên hơn sẽ cho phép chúng tôi đạt được thủ tục tốt hơn trong nhiệm vụ chính của mình. Cụ thể, có thể đào tạo trên một bộ dữ liệu lớn hơn với nhiều ví dụ chữ ký hơn cho mỗi hệ thống có thể đạt được sự thích nghi cao hơn, cũng như đào tạo một mạng lưới lớn hơn, điều mà chúng tôi không thể làm do hạn chế về thời gian và tài nguyên tính toán.

Chúng tôi cũng bị hạn chế ở đây bởi thực tế là bộ dữ liệu của chúng tôi tương đối nhỏ, rất khó để tìm thấy các bộ dữ liệu chữ ký có sẵn tốt. Do hạn chế về thời gian và năng lực tính toán của máy tính, luận văn không thể khảo sát hết các mô hình CNN. Vấn đề nghiên cứu trong luận văn là khá mới nên khi trình bày, dịch thuật tài liệu chắc chắn không tránh khỏi những thiếu sót và hạn chế cần phải khắc phục. Rất mong nhận được sự góp ý của độc giả.

DANH MỤC CÁC TÀI LIỆU THAM KHẢO

- [1]. Dewi Suryani, Michael Reynaldo Phangtrastu, and Yudy Purnama, "An Offline Signature Verification Using Convolutional Neural Networks," in *International Journal of Advanced Science and Technology*, Vol. 29, No. 6, pp. 4764 - 4773 2016.
- [2]. D. Suryani, E. Irwansyah and R. Chindra, "Offline signature recognition and verification system using efficient fuzzy kohonen clustering network (EFKCN) algorithm," *Procedia computer science*, vol. 116, pp. 621-628, 2017.
- [3]. L. G. Hafemann, R. Sabourin and L. S. Oliveira, "Learning features for offline handwritten signature verification using deep convolutional neural networks," *Pattern Recognition*, vol. 70, pp. 163-176, 2017.
- [4]. L. G. Hafemann, R. Sabourin and L. S. Oliveira, "Writer-independent Feature Learning for Offline Signature Verification using Deep Convolutional Neural Networks," in *2016 international joint conference on neural networks (IJCNN)*, 2016.
- [5]. Winston, P.: *Learning structural descriptions from examples*. In Winston, P., ed.: *The Psychology of Computer Vision*. McGraw-Hill (1975) 157–210.
- [6]. R. Sa-Ardship and K. Woraratpanya, "Offline handwritten signature recognition using adaptive variance reduction," in *2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2015.
- [7]. R. Sa-Ardship and K. Woraratpanya, "Offline Handwritten Signature Recognition using Polar-Scale Normalization," in *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2016.
- [8]. S. L. Karanjkar and P. Vasambekar, "Signature recognition on bank cheques using ANN," in *2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, 2016.
- [9]. A. B. Jagtap and H. R. S. Hegadi, "Offline handwritten signature recognition based on upper and lower envelope using eigen values," in *2017 World Congress on Computing and Communication Technologies (WCCCT)*, 2017.

- [10]. Mai Văn Thủy (2015). Nghiên cứu về mô hình thống kê học sâu và ứng dụng trong nhận dạng chữ viết tay hạn chế. Luận văn thạc sỹ, Đại học Công nghệ thông tin và Truyền thông.
- [11]. G. Alvarez, B. Sheffer, and M. Bryant, “Offline Signature Verification with Convolutional Neural Networks,” 2016.
- [12]. Nikolaas N. Oosterhof, Philip K. F. Hölzenspies, and Jan Kuper. Application patterns. A presentation at Trends in Functional Programming, 2005.
- [13]. Vojislav Kecman: "Learning and Soft Computing — Support Vector Machines, Neural Networks, Fuzzy Logic Systems", The MIT Press, Cambridge, MA, 2001
- [14]. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
- [15]. Harish Srinivasan, Sargur N. Srihari, and Matthew J. Beal, “Machine Learning for Signature Verification”, Computer Vision, Graphics and Image Processing pp 761-775, 2006.
- [16]. Phuoc-Hai Huynh, Van Hoa Nguyen, Thanh-Nghi Do, “So sánh mô hình học sâu với các phương pháp học tự động khác trong phân lớp dữ liệu biểu hiện Gen microarray”, Nghiên cứu cơ bản và ứng dụng công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017.

BẢN CAM ĐOAN

Tôi cam đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung luận văn/luận án qua phần mềm Kiểm tra tài liệu một cách trung thực và đạt kết quả mức độ tương đồng 20% toàn bộ nội dung luận văn/luận án. Bản luận văn/luận án kiểm tra qua phần mềm là bản cứng luận văn đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện.

TP. Hồ Chí Minh, ngày 25 tháng 01 năm 2022


HỌC VIÊN CAO HỌC

Huỳnh Minh Nhựt

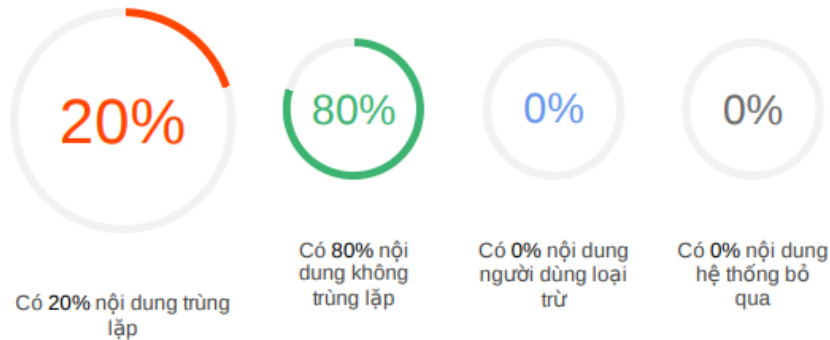


BÁO CÁO KIỂM TRA TRÙNG LẶP

Thông tin tài liệu

Tên tài liệu:	Nghiên cứu mô hình nhận dạng chữ ký viết tay sử dụng học sâu CNN	
Tác giả:	Huỳnh Minh Nhật	
Điểm trùng lặp:	20	
Thời gian tải lên:	22:01 09/02/2022	
Thời gian sinh báo cáo:	22:11 09/02/2022	
Các trang kiểm tra:	52/52 trang	

Kết quả kiểm tra trùng lặp



Nguồn trùng lặp tiêu biểu

123docz.net tailieu.vn vjol.info.vn

Học viên

Người hướng dẫn khoa học

Huỳnh Minh Nhật

TS. Nguyễn Xuân Sâm