

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC NHA TRANG  
KHOA CÔNG NGHỆ THÔNG TIN



**NGUYỄN TẤT CHỦ**

**XÂY DỰNG ỨNG DỤNG HỌC TỪ  
VỰNG TIẾNG ANH SỬ DỤNG GIẢI  
THUẬT PHÂN LỚP DỮ LIỆU**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**Ngành Công nghệ thông tin**

**Giảng viên hướng dẫn: Phạm Thị Kim Ngoan**

*Nha Trang – 2017*

## LỜI CẢM ƠN

Lời cảm ơn đầu tiên em xin gửi đến quý thầy cô Khoa Công nghệ Thông tin trường Đại học Nha Trang đã truyền dạy cho em những kiến thức cho em trong thời gian qua để em có thể hoàn thành quá trình nghiên cứu và thực hiện đề tài đồ án. Và hơn hết em xin chân thành cảm ơn cô Phạm Thị Kim Ngoan, người đã tận tình hướng dẫn cho em trong suốt quá trình làm đề tài. Bên cạnh đó em xin gửi lời cảm ơn đến Công ty TNHH Bizzon đã tạo điều kiện tốt nhất để em có thể hoàn thành đồ án này.

Mặc dù đã cố gắng hoàn thành tốt đề tài nhưng do điều kiện thời gian thực hiện có hạn, khả năng nghiên cứu và kinh nghiệm thực tế còn hạn chế nên bài báo cáo sẽ có nhiều thiếu sót. Em rất mong nhận được sự đóng góp như sự cảm thông, chia sẻ của quý Thầy cô và các bạn để em có điều kiện bổ sung, nâng cao kiến thức tốt hơn cho việc học tập, nghiên cứu và công việc sau này.

Cuối cùng em kính chúc quý Thầy cô dồi dào sức khỏe, niềm tin để tiếp tục sự nghiệp cao quý của mình là truyền đạt kiến thức cho thế hệ mai sau.

Em xin chân thành cảm ơn!

Nha Trang, ngày 25 tháng 06 năm 2017

Sinh viên thực hiện

***Nguyễn Tất Chủ***

## NHẬN XÉT

(Của giảng viên hướng dẫn)

## NHẬN XÉT

## (Của giảng viên phản biện)

## MỤC LỤC

<b>CHƯƠNG 1: TỔNG QUAN .....</b>	<b>1</b>
1.1. Giới thiệu .....	1
1.2. Đối tượng và phạm vi nghiên cứu .....	2
1.2.1. Lý thuyết .....	2
1.2.2. Thực tiễn .....	2
<b>CHƯƠNG 2: CƠ SỞ LÝ THUYẾT .....</b>	<b>3</b>
2.1. Nền tảng xây dựng ứng dụng NodeJS .....	3
2.1.1. Đặc điểm của NodeJS .....	4
2.1.2. Cài đặt môi trường NodeJS và các công cụ phát triển.....	5
2.1.3. Phát triển ứng dụng với NodeJS .....	5
2.2. AngularJS Framework .....	9
2.3. NoSQL và hai hệ quản trị CSDL MongoDB, Redis.....	13
2.3.1. Cơ sở dữ liệu NoSQL .....	13
2.3.2. Hệ quản trị CSDL MongoDB .....	16
2.3.3. Hệ thống lưu trữ cache máy chủ Redis.....	19
2.4. Thuật toán phân lớp dữ liệu bằng Neural Network .....	20
2.4.1. Ý tưởng xây dựng mạng Neural nhân tạo.....	22
2.4.2. Cấu trúc Neural nhận tạo .....	26
2.4.3. Mạng Neural nhân tạo và khái niệm học .....	29
2.4.4. Các mô hình mạng neural .....	32
2.4.5. Giải thuật lan truyền ngược .....	33
2.4.6. Các vấn đề trong xây dựng mạng và phương pháp lan truyền ngược	36
2.4.7. Thư viện hỗ trợ xây dựng mạng Neural trong môi trường NodeJS ...	40
2.5. Bài toán nhận dạng chữ viết tay.....	41
2.5.1. Các giai đoạn xây dựng hệ thống nhận dạng chữ viết tay .....	41
2.5.2. Các kỹ thuật trích rút đặc trưng .....	43
2.6. Xử lý ảnh .....	44
2.6.1. Chuyển xám ảnh .....	45

2.6.2. Nhị phân ảnh.....	46
2.6.3. Nhiễm ảnh .....	47
2.6.4. Xác định đối tượng trong ảnh .....	47
2.6.5. Tách ghép ký tự .....	48
<b>CHƯƠNG 3: XÂY DỰNG ÚNG DỤNG .....</b>	<b>50</b>
3.1. Xây dựng bộ dữ liệu .....	50
3.2. Xây dựng mạng Neural nhận dạng ký tự .....	53
3.2.1. Thực nghiệm chọn số lớp ẩn.....	54
3.2.2. Thực nghiệm xác định tốc độ học.....	55
3.2.3. Xác định ngưỡng lỗi và số lần lặp tối đa .....	56
3.2.4. Kết quả nhận dạng ký tự .....	57
3.3. Xây dựng ứng dụng .....	58
3.3.1. Khảo sát chương trình đào tạo lớp 4 .....	58
3.3.2. Xác định yêu cầu.....	60
3.3.3. Thiết kế cơ sở dữ liệu .....	60
3.3.4. Cấu trúc xây dựng ứng dụng.....	64
3.3.5. Các sơ đồ chức năng .....	67
3.4. Xây dựng ứng dụng .....	70
3.4.1. Xây dựng và lưu trữ mạng neural .....	70
3.4.2. Tổ chức cấu trúc và cài đặt ứng dụng .....	71
3.5. Kết quả thực hiện .....	79
3.5.1. Các chức năng của học viên .....	79
3.5.2. Các chức năng nhóm quản lý.....	83
<b>CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>87</b>
4.1. Kết quả đạt được .....	87
4.2. Hướng phát triển .....	87
<b>DANH MỤC TÀI LIỆU THAM KHẢO.....</b>	<b>89</b>

## **DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT**

<b>STT</b>	<b>Ký hiệu viết tắt</b>	<b>Điễn giải</b>
1	I/O	Input/Ouput
2	API	Application Programming Interface
3	JSON	JavaScript Object Noattion
4	IDE	Integrated Development Environment
5	MVC	Model – View – Controller
6	MVVM	Model – View – View Model
7	MVW	Model – View – Whatever
8	CSDL	Cơ sở dữ liệu
9	DOM	Document Object Model
10	IIS	Internet Information Services

## **DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH**

### **Danh sách bảng**

Bảng 2.1: Một số directive thường dùng .....	12
Bảng 2.2: Bảng so sánh CSDL NoSQL và cơ sở dữ liệu quan hệ truyền thống .....	13
Bảng 2.3: Bảng phân loại hệ quản trị CSQL NoSQL .....	16
Bảng 2.4: Các khái niệm tương đương giữa SQL DB và MongoDB .....	17
Bảng 2.5: Các thao tác với MongoDB .....	18
Bảng 2.6: Các hàm kích hoạt thường dùng trong mô hình neural nhân tạo .....	28
Bảng 2.7: Bảng chân trị của phép toán XOR.....	29
Bảng 2.8: Minh họa quá trình tách xử lý tách ký tự .....	49
Bảng 3.1: Bảng kết quả thu được từ quá trình thử số neural lớp ẩn .....	54
Bảng 3.2: Kết quả thử nghiệm tốc độ học .....	55
Bảng 3.3: Kết quả thử nghiệm xác định ngưỡng lỗi và số lần lặp .....	56
Bảng 3.4: Kết quả nhận dạng trên các mẫu ký tự .....	58
Bảng 3.5: Cấu trúc Collection Vocabulary .....	61
Bảng 3.6: Cấu trúc Collection Unit.....	62
Bảng 3.7: Cấu trúc Collection User .....	63
Bảng 3.8: Cấu trúc Collection Blog.....	63
Bảng 3.9: Các module sử dụng xây dựng ứng dụng phía máy chủ .....	64
Bảng 3.10: Kịch bản luyện tập từ vựng .....	68

### **Danh sách sơ đồ**

Sơ đồ 2.1: Cấu trúc chung của hệ thống nhận dạng chữ viết tay.....	42
Sơ đồ 3.1: Sơ đồ trình tự chức năng luyện tập từ vựng .....	69

## **Danh sách hình**

Hình 2.1: Kết quả chạy ứng dụng Helloworld với AngularJS .....	12
Hình 2.2: Cấu trúc của một Neural sinh học .....	25
Hình 2.3: Mô hình cấu trúc neural nhân tạo .....	27
Hình 2.4: Minh họa một Neural nhân tạo .....	28
Hình 2.5: Cấu trúc Neural của phép toán XOR .....	29
Hình 2.6: Mô hình chung của mạng Neural đa lớp .....	30
Hình 2.7: Mô hình Perceptron .....	33
Hình 2.8: Minh họa quá trình huấn luyện với gradient descent .....	39
Hình 2.9: Minh họa kỹ thuật Zoning .....	44
Hình 3.1: Bộ 26 ký tự mẫu .....	50
Hình 3.2: Ảnh scan mẫu của ký tự “x” .....	51
Hình 3.3: Mô tả quá trình tiền xử lý tập dữ liệu .....	51
Hình 3.4: Minh họa phương pháp trích chọn đặc trưng .....	52
Hình 3.5: Minh họa sự biến thiên khả năng nhận dạng theo độ lỗi .....	57
Hình 3.6: Trích phần Vocabulary sách giáo Tiếng Anh khoa lớp 4 .....	59
Hình 3.7: Mô hình xây dựng ứng dụng .....	66
Hình 3.8: Use case mức 1 nhóm học viên .....	67
Hình 3.9: Use case mức 1 nhóm quản trị .....	68
Hình 3.10: Mô hình tổ chức lưu trữ mã xử lý ứng dụng .....	72
Hình 3.11: Cấu trúc ứng dụng .....	72
Hình 3.12: Giao diện trang học từ vựng .....	79
Hình 3.13: Giao diện luyện tập từ vựng .....	80

Hình 3.14: Giao diện trang luyện tập từ vựng theo câu.....	80
Hình 3.15: Giao diện trang tra từ điển .....	81
Hình 3.16: Giao diện trang đọc tin tức, bài viết.....	81
Hình 3.17: Giao diện trang yêu cầu nhận dạng chữ học viên .....	82
Hình 3.18: Giao diện đăng nhập hệ thống .....	83
Hình 3.19: Giao diện trang chỉnh sửa từ vựng.....	83
Hình 3.20: Trang danh sách từ vựng.....	84
Hình 3.21: Giao diện trang chỉnh sửa thông tin người dùng .....	84
Hình 3.22: Giao diện trang nhật ký hệ thống.....	85
Hình 3.23: Giao diện trang thêm bài viết.....	85
Hình 3.24: Giao diện trang phân quyền .....	86
Hình 3.25: Giao diện trang cập nhật mô hình nhận dạng .....	86

## CHƯƠNG 1: TỔNG QUAN

### 1.1. Giới thiệu

Thế giới ngày nay đã có nhiều tiến bộ mạnh mẽ về công nghệ thông tin từ một tiềm năng thông tin đã trở thành một tài nguyên thực sự, cùng với đó từ khi ra đời máy tính đã nhanh chóng phát triển và đóng một vai trò rất quan trọng trong nghiên cứu khoa học kỹ thuật cũng như trong đời sống. Nhưng một máy tính dù có mạnh đến đâu chăng nữa, cũng chỉ có thể làm việc theo một chương trình đã được hoạch định sẵn bởi lập trình viên. Nó vẫn không có khả năng liên tưởng, kết nối sự việc này với sự việc khác, và quan trọng hơn hết là khả năng sáng tạo như con người. Ngày nay, với sự phát triển với tốc độ rất nhanh của lĩnh vực công nghệ thông tin, lĩnh vực học máy không chỉ dừng ở mức độ nghiên cứu mà hơn thế ngày càng được đưa vào các ứng dụng thực tế. Trong lĩnh vực học máy, bài toán nhận dạng mẫu được quan tâm rất nhiều và cũng đạt được rất nhiều thành công rực rỡ có ý nghĩa thực tế lớn có thể kể đến như nhận dạng chữ in dùng trong quá trình xử lý tự động của các thư viện, cơ quan hành chính, nhận dạng chữ viết tay dùng trong các khâu xử lý bưu phẩm tại bưu điện hay, xử lý điểm tại các trường học. Do đó xu hướng xây dựng các ứng dụng sử dụng các giải thuật học máy đang là một ngành công nghiệp hứa hẹn đầy tiềm năng.

Bên cạnh sự phát triển của công nghệ thông tin trong xã hội hiện nay toàn cầu, tiếng Anh đã trở thành yếu tố thiết yếu cho sự phát triển của mỗi cá nhân và của toàn xã hội, theo đó nhu cầu học tiếng Anh ngày càng phát triển, bằng chứng là tiếng Anh đã được đưa vào chương trình giáo dục từ cấp 1. Hiện nay có nhiều ứng dụng được xây dựng để hỗ trợ các học sinh học tiếng anh có thể kể đến như các website topical.vn, antoree.com,... tuy nhiên những ứng dụng bám sát chương trình học của các em gần như chưa. Vì vậy tôi chọn đề tài “Xây dựng ứng dụng học tiếng Anh dùng giải thuật phân lớp dữ liệu” hỗ trợ các em học sinh lớp 4 học tiếng Anh nhằm mục đích dùng giải thuật phân lớp dữ liệu để nhận dạng chữ viết tay kết hợp với những công nghệ phát triển ứng dụng website mới.

Có nhiều giải thuật khác nhau được sử dụng trong lĩnh vực học máy, một trong số các giải thuật được nhiều người quan tâm là mô hình mạng Neural nhân tạo có thể

xây dựng được những hệ thống thông minh với độ chính xác cao. Trong đồ án tốt nghiệp này tôi đi vào nghiên cứu mạng Neural nhân tạo và tích hợp vào ứng dụng trên nền tảng NodeJS và các hệ quản trị CSDL NoSQL thế hệ mới để xây dựng ứng dụng giúp các em học sinh khối 4 có thể tự học từ vựng và tự kiểm tra bài tập về từ vựng tiếng Anh theo chương trình đào tạo của bộ Giáo dục và đào tạo.

## **1.2. Đối tượng và phạm vi nghiên cứu**

### **1.2.1. Lý thuyết**

Nhằm xây dựng một ứng dụng phù hợp với vấn đề đặt ra, về mặt lý thuyết đề tài sẽ tìm hiểu và nghiên cứu các lĩnh vực sau:

- Tìm hiểu nền tảng NodeJS, ngôn ngữ lập trình JavaScript và các thư viện trong xây dựng ứng dụng website
- Nghiên cứu CSDL NoSQL và hệ quản trị CSDL MongoDB, Redis
- Nghiên cứu kỹ thuật phân lớp dữ liệu bằng mạng Neural và các ứng dụng
- Tìm hiểu các kỹ thuật xử lý ảnh, trích chọn đặc trưng trong quá trình xây dựng mạng Neural

### **1.2.2. Thực tiễn**

Từ cơ sở lý thuyết đề tài sẽ xây dựng ứng dụng hỗ trợ học từ vựng tiếng Anh bằng NodeJS với các tính năng:

- Tổ chức các từ vựng theo bài, hiển thị trực quan sinh động trên ứng dụng
- Xây dựng các dạng bài tập về từ vựng theo chương trình học của khối 4
- Dùng giải thuật phân lớp để nhận dạng chữ viết tay giúp học sinh có thể tự kiểm tra bài tập mình làm
- Hỗ trợ tra cứu từ điển, phát âm từ vựng

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Trong chương này sẽ tập trung nghiên cứu tìm hiểu nền tảng phát triển ứng dụng NodeJS cùng với các kỹ thuật có liên quan về CSDL, khả năng phát triển ứng dụng trên nền tảng này trong lĩnh vực trí tuệ nhân tạo mà cụ thể là mô hình mạng nơ-ron nhân tạo.

Để xây dựng hệ thống với ý tưởng đặt ra, đề tài nghiên cứu một nền tảng để phát triển ứng dụng sau đó sẽ nghiên cứu cách thực hiện nhận dạng chữ viết tay cùng kỹ thuật phân lớp dữ liệu bằng mạng Neural nhân tạo và từ đó tìm cách tích hợp kỹ thuật này vào ứng dụng.

### 2.1. Nền tảng xây dựng ứng dụng NodeJS

Node.js là một nền tảng dựa vào Chrome Javascript runtime để xây dựng các ứng dụng nhanh, có độ lớn. Node.js sử dụng các phần phát sinh các sự kiện (event-driven), mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.

NodeJS là một mã nguồn mở, đa nền tảng được thiết kế để xây dựng các ứng dụng Internet nhanh, có độ lớn, có khả năng mở rộng phía máy chủ dựa trên Chrome Javascript Engine V8 được phát triển bởi Ryan Dahl năm 2009 dưới sự bảo trợ của Joyent.

Những môi trường tương tự được viết trong các ngôn ngữ khác bao gồm Twisted cho Python, Perl Object Environment cho Perl, libevent cho C và EventMachine cho Ruby. Khác với hầu hết các chương trình Javascript, Nodejs không chạy trên một trình duyệt mà chạy trên Server. Node.js sử dụng nhiều chi tiết kỹ thuật của CommonJS. Nó cung cấp một môi trường REPL cho kiểm thử tương tác.

NodeJS cung cấp cho nhà phát triển các module JavaScript đa dạng, giúp cho việc phát triển ứng dụng dễ dàng hơn, tuy nhiên NodeJS chỉ là môi trường, điều này có nghĩa là nhà phát triển phải làm mọi thứ. Không có một máy chủ nào mặc định, một đoạn script sẽ xử lý tất cả các kết nối từ máy khách. Điều này làm giảm được đáng kể tài nguyên được sử dụng trong ứng dụng.

### **2.1.1. Đặc điểm của NodeJS**

Một vài đặc điểm quan trọng của NodeJS khuyên cho NodeJS trở thành lựa chọn hàng đầu cho các nhà phát triển phần mềm:

**Không đồng bộ và Phát sinh sự kiện (Event Driven):** Khác với ngôn ngữ C#, PHP hay Java, tất cả các APIs của thư viện Node.js đều bất đồng bộ (non-blocking). Điều này rất cần thiết vì Node.js không bao giờ đợi một API trả về dữ liệu. Server chuyển sang một API sau khi gọi nó và có cơ chế thông báo về Sự kiện của Node.js giúp Server nhận dựa phản hồi từ các API gọi trước đó.

**Chạy rất nhanh:** Dựa trên V8 Javascript Engine của Google Chrome, nền tảng NodeJS rất nhanh trong các quá trình thực thi các đoạn code.

**Các tiến trình đơn giản nhưng hiệu năng cao:** Node.js sử dụng một mô hình đơn luồng (single thread) với các sự kiện lặp. Các cơ chế sự kiện giúp server trả lại các phản hồi với một cách không khóa và tạo cho máy chủ hiệu quả cao ngược lại với các cách truyền thống tạo ra một số lượng luồng hữu hạn để quản lý request. Node.js sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các Server truyền thống như Apache HTTP Server.

**Không đệm:** Ứng dụng NodeJS không lưu trữ các dữ liệu buffer.

**Có giấy phép:** mã nguồn mở NodeJS<sup>[1]</sup> được phát hành dựa vào MIT License.

Với những đặc điểm này NodeJS là một sự lựa chọn hoàn hảo cho các ứng dụng website trên các lĩnh vực công nghệ bao gồm:

- Các ứng dụng về I/O
- Các ứng dụng về luồng dữ liệu
- Các ứng dụng về dữ liệu hướng đến thời gian thực
- Các ứng dụng dựa vào JSON APIs
- Các ứng dụng Single Page Application

Hiện tại có rất nhiều hệ thống lớn hiện tại đang sử dụng NodeJS, trong số này có thể kể đến như: eBay, GE, GoDaddy, Microsoft, Yahoo, Linkedin, Paypal, Uber,...

### 2.1.2. Cài đặt môi trường NodeJS và các công cụ phát triển

Để cài đặt môi trường phát triển NodeJS, có hai cách là tải phiên bản cài đặt từ trang chủ của NodeJS<sup>[2]</sup> theo hệ điều hành tương ứng hoặc sử dụng các công cụ hỗ trợ cài đặt gói thông qua dòng lệnh:

- Trên Windows sử dụng Chocolatey<sup>[3]</sup>: *choco install nodejs*
- Trên MacOS sử dụng Homebrew<sup>[4]</sup>: *brew install node*
- Trên Linux sử dụng trình cài đặt gói mặc định: *sudo apt-get install nodejs*

Sau khi cài đặt sử dụng lệnh *node -v*, để kiểm tra đã cài đặt thành công hay chưa, nếu thành công dòng lệnh này sẽ in ra màn hình console phiên bản NodeJS đã cài trên hệ thống.

Ngoài điều kiện tiên quyết là cài đặt môi trường, ngoài các ứng dụng soạn thảo mặc định của hệ điều hành để thuận tiện cho quá trình phát triển cần có thêm những trình soạn thảo code (Editor) hoặc IDE, các IDE miễn phí và nổi bật thường được các lập trình viên trên thế giới sử dụng bao gồm: Visual Studio Code, Sublime Text, Notepad++, ...

### 2.1.3. Phát triển ứng dụng với NodeJS

Bất kỳ lập trình viên nào muốn xây dựng ứng dụng với NodeJS cũng cần phải biết và hiểu được các khái niệm như module, lập trình không đồng bộ, function callback, promise, các khái niệm về giao thức giữa máy khách với máy chủ (client – server), cách giao tiếp của chúng với nhau, cũng như các đối tượng, các lớp, các hàm được cung cấp sẵn của framework này. Phần dưới đây sẽ đưa ra những định nghĩa và ví dụ minh họa.

**Modules** trong NodeJS giống như các thư viện trong C, C#, Java, ... mỗi module là một đối tượng (Object) chứa các hàm, các hằng số, ... Điều này làm cho việc xây dựng các ứng dụng phức tạp trở nên đơn giản hơn. Mỗi module đóng gói tập các phương thức, thông tin, ... liên quan đến một đối tượng, chức năng của Module. Ví dụ **fs** là Module chứa các hàm, các hằng số cụ thể liên quan đến việc đọc ghi dữ liệu hệ thống. Ngoài những module được NodeJs cung cấp sẵn để thao tác với hệ thống và một số tiện ích khác. NodeJs cho phép bất kỳ ai cũng có thể tạo ra module của riêng mình, hơn thế

nữa NodeJs đi kèm với trình quản lý gói **npm**<sup>[5]</sup> hoặc **yarn**<sup>[6]</sup> giúp cho việc cài đặt tích hợp các module vào ứng dụng của mình trở nên vô cùng đơn giản.

Các thao tác cần thiết để sử dụng một module đó là cài đặt module và tham chiếu tới module.

- Cú pháp cài đặt bằng npm: **npm install <tên package>**. Ví dụ để cài đặt module hapi(module hỗ trợ xây dựng ứng dụng web phía máy chủ): *npm install hapi*
- Cú pháp gọi tham chiếu module: **var <tên biến> = require('<tên module>')**.  
Ví dụ: *var hapi = require('hapi')*

Trong ví dụ trên hàm *require* trả về tham chiếu tới module *http* được cài đặt từ internet về thông qua lệnh *npm install*. Để tham chiếu tới một Module cục bộ do trên hệ thống phải chỉ rõ đường dẫn tới tập tin của Module đó. Ví dụ: *var myModule = require('./path/to/file/moduleName.js')*

Có thể xem mỗi module là một đoạn mã được đóng gói lại với nhau, mã lệnh bên trong module có phạm vi là *private*. Gần giống như việc public một hàm của một Class, bằng việc sử dụng đối tượng exports, sẽ giúp các hàm, các hằng private có thể để đưa ra ngoài sử dụng cho trên ứng dụng. Xem xét ví dụ dưới đây:

```
const PI = Math.PI;
exports.dienTich = function (r) {
    return PI * r * r;
};
exports.chuVi = function (r) {
    return 2 * PI * r;
};
```

Đoạn mã trên tạo ra hằng số *PI* và hai function, hằng số *PI* chỉ có thể sử dụng trong nội bộ module đang định nghĩa ở trên. Qua việc sử dụng đối tượng *exports*, hai hàm *dienTich* và *chuVi* có thể sử dụng khi tham chiếu tới Module này.

**Global Object:** các object có sẵn ở tất cả các module bao gồm: Buffer, \_\_dirname, \_\_filename, console, exports, global, module, process; các function có sẵn bao gồm: require, setTimeout, ... và một class có sẵn là Buffer.

**Asynchronous** (bất đồng bộ): Không giống như các ngôn ngữ lập trình truyền thống C#, Java, C++,... chương trình sẽ chạy tuần tự từng lệnh và chỉ thực hiện lệnh tiếp theo khi lệnh trước đó đã thực hiện xong, điều này sẽ sinh ra một trang thái hay gọi là trạng thái chờ, Javascript là ngôn ngữ lập trình bất đồng bộ, khi thực thi chương trình có thể bỏ qua một số bước chờ không cần thiết, thực hiện nhiều công việc song song cùng lúc.

Một ví dụ về lập trình bất đồng bộ:

```
const fs = require('fs'); // tham chiếu module file system
let filePath = './text.txt'; // khai báo đường dẫn tập tin
// Thực hiện đọc tập tin
fs.readFile(filePath, function done(error, data) {
  console.log(data.toString()); // Xuất kết quả đọc được
});
console.log('Finished'); // Thông báo kết thúc
```

Trong ví dụ trên theo trình tự nội dung tập tin sẽ được ghi lên màn hình trước dòng thông báo kết thúc “Finished”, tuy nhiên kết quả thực tế khi thực thi dòng thông báo kết thúc sẽ thực hiện trước bởi bì hàm `fs.readFile` là hàm bất đồng bộ. Khi thực hiện lệnh này chương trình sẽ thực hiện gọi sự kiện thực hiện lệnh này và không chờ lệnh này thực hiện xong để thực thi lệnh tiếp theo mà sẽ thực hiện các lệnh tiếp theo tiếp, khi nào hàm này thực hiện xong sẽ trả về một sự kiện được thực hiện trong hàm `done`. Hàm `done` nếu có lỗi, biến lỗi sẽ được nhận trong tham số thứ nhất là `error`, ngược lại nếu thành công, biến `error` sẽ nhận giá trị `null` và giá trị đọc được sẽ nhận vào tham số thứ hai là `data`. Đây là quy ước chung cho các hàm bất đồng bộ nó được gọi là cơ chế callback.

Như vậy để thực hiện các lệnh một cách tuần tự, nhà phát triển sẽ lồng các câu lệnh vào trong hàm callback. Trong trường hợp cần rất nhiều các câu hàm, câu lệnh thực hiện tuần tự sẽ tạo nên một cơ chế callback lồng nhau rất phức tạp, rất khó sử dụng và phát hiện lỗi.

Để giải quyết tình trạng trên có một cơ chế được xây dựng đó là `Promise`, tuy nhiên `Promise` không phải là cách giải quyết tối ưu nhất, ở phiên bản mới nhất của NodeJS đã hỗ trợ cơ chế `async/await` theo chuẩn ES7. Đây là cách giải quyết tối ưu nhất

trong vấn đề bắt đồng bộ của Javascript. Phần này sẽ được trình bày cụ thể ở chương tiếp theo.

**Kiểu dữ liệu JSON:** là một dạng dữ liệu dùng để trao đổi dữ liệu giữa các ngôn ngữ, nền tảng với nhau. Kiểu dữ liệu JSON có đặc điểm dễ viết, dễ đọc, dễ phân tích và có thể tạo ra một cách dễ dàng. JSON được xây dựng bởi hai cấu trúc:

- Một tập hợp các cặp tên - giá trị. Trong các ngôn ngữ khác nhau, nó được xây dựng như một đối tượng, bản ghi, struct, vector, danh sách, hoặc mảng kết hợp.
- Là 1 tập hợp các giá trị đã được sắp xếp.

Một ví dụ về kiểu dữ liệu JSON:

```
{
    "MSSV": 55133917,
    "ho_ten": "Nguyễn Tất Chủ",
    "gioi_tinh_nam": true,
    "tong_ket": null,
    "cac_mon_da_hoc": [
        "Kỹ thuật lập trình",
        "Tin học cơ sở",
        "Trí tuệ nhân tạo"
    ],
    "thong_tin_lop": {
        "lop": "55-CNTT1",
        "khoa-hoc": "2013-2017"
    }
}
```

Trong ví dụ trên giá trị bên trái dấu “:” là khoá, bên phải là giá trị. Các giá trị được lưu trữ ở nhiều dạng khác nhau, có thể là một số, một giá trị Boolean, giá trị một mảng hoặc một object khác, cặp ngoặc “{}” biểu thị cho 1 object, “[ ]” được hiểu như một mảng.

Có một kiểu ngôn ngữ tương tự là XML, tuy nhiên trong phạm vi đề này không đề cập đến.

**REST:** viết tắt của Representaion State Transfer, là một kiến trúc trong việc thiết kế hệ thống phân tán (distributed system) kiểu trao đổi dữ liệu dạng JSON hoặc XML mà sử dụng các ràng buộc chặt chẽ. Một số tiêu chuẩn của REST bao gồm:

- Mô hình client - server
- Phi trạng thái (stateless interation), có thể lưu nhưng ko đủ điều kiện xác thực.
- Thông nhất giao thức (uniform interface).

Các cơ bản quy tắc được quy định chung của RESP thông qua giao thức HTTP bao gồm:

- Để tạo một tài nguyên trên máy chủ sử dụng phương thức POST.
- Để truy xuất một tài nguyên sử dụng GET.
- Để cập nhật một tài nguyên sử dụng PUT.
- Để xoá một tài nguyên sử dụng DELETE.

## 2.2. AngularJS Framework

Công nghệ HTML hỗ trợ tốt cho các trang web tĩnh, kiểu như trước năm 2000 vậy. Khi bạn xây dựng 1 trang web với PHP, Node/Express, hay Ruby thì nó cũng chỉ là một trang web tĩnh với nội dung được thay đổi khi bạn gửi request về máy chủ, máy chủ sẽ render 1 trang với nội dung tương ứng. Tuy nhiên mọi thứ đã thay đổi nhiều từ sự phát triển của HTML5, nhất là khi có sự chống lưng từ những công ty lớn như Google, Yahoo, Facebook, và sự tập hợp đông đảo của cộng đồng mã nguồn mở.

AngularJS là một framework viết bằng Javascript được sử dụng phía client có cấu trúc cho các ứng dụng web động. Nó cho phép bạn sử dụng HTML như là ngôn ngữ mẫu và cho phép mở rộng cú pháp của HTML để diễn đạt các thành phần ứng dụng một cách rõ ràng súc tích.

AngularJS là framework mã nguồn mở hoạt động dưới giấy phép Apache Lincense được đưa ra lần đầu năm 2009 bởi Misko Hevery và Adam Abrons. Hiện tại thư viện này được duy trì và phát triển bởi hãng Google.

Các tính năng cốt lõi của Framework AngularJS bao gồm:

- Data-binding: Nó tự động đồng bộ hóa dữ liệu giữa thành phần model và view.
- Scope: Là những đối tượng hướng đến model, nó hoạt động như là cầu nối giữa controller và view

- Controller: Đây là những tính năng của AngularJS mà được giới hạn tới một scope cụ thể
- Service: AngularJS hoạt động với một vài dịch vụ (service) có sẵn , ví dụ \$http để tạo XMLHttpRequests. Nó là các singleton object mà được khởi tạo duy nhất một lần trong ứng dụng
- Filter: Nó lựa chọn (hay là lọc) các tập con từ tập item trong các mảng và trả về các mảng mới
- Directive: Directive là các marker trong các phần tử DOM (như các phần tử, thuộc tính, css và nhiều hơn thế). Nó có thể dùng để tạo các thẻ HTML riêng phục vụ những mục đích riêng. AngularJS có những directive có sẵn như ngBind, ngModel, ngController, ngApp...
- Template:Là các rendered view với các thông tin từ controller và model. Nó có thể được sử dụng trong các file riêng rẽ (ví dụ như index.html) hoặc nhiều view với một trang sử dụng “partials”
- Routing: Là khái niệm của sự chuyển đổi qua lại các view
- Deep Linking: Cho phép bạn mã hóa trạng thái các ứng dụng trên địa chỉ URL để nó có thể được bookmark. Các ứng dụng có thể được phục hồi lại từ các địa chỉ URL với cùng một trạng thái
- Dependency Injection: AngularJS có sẵn một hệ thống con dependency injection để giúp các lập trình viên tạo ra các ứng dụng dễ phát triển, dễ hiểu và kiểm tra

AngularJS được thiết kế kết hợp giữa mô hình MVC và MVVM hay còn được là MVW. Những ưu điểm nổi bật của AngularJS đó là:

- AngularJS cung cấp khả năng tạo ra các Single Page Application với API service một cách rất rõ ràng và dễ dàng để duy trì, nâng cấp.
- AngularJS cung cấp khả năng Data binding tới HTML do đó giúp người dùng cảm giác linh hoạt, thân thiện, trực quan.
- AngularJS code dễ dàng trong giai đoạn unit test, functional testing.
- AngularJS sử dụng dependency injection.

- AngularJS cung cấp khả năng tái sử dụng các component (thành phần).
- Với AngularJS, lập trình viên sẽ viết ít code hơn, với nhiều chức năng hơn.
- Với AngularJS, view là thành phần trong trang HTML thuận, trong khi controller được viết bởi JavaScript với quá trình xử lý nghiệp vụ.
- AngularJS có thể sử dụng kết hợp với các framework, thư viện khác mà không gây xung đột

Bên cạnh những ưu điểm, AngularJS cũng có những nhược điểm riêng:

- Không an toàn là một JavaScript Framework, ứng dụng được viết bởi AngularJS nên không an toàn. Vấn đề này được giải quyết nếu phía máy chủ web có cơ chế chứng thực và phân quyền phù hợp
- Được xây dựng bằng JavaScript nên khi người sử dụng vô hiệu hóa tính năng Javascript thì ứng dụng bị vô hiệu hóa.

Các thành phần của AngularJS quan trọng khi xây dựng ứng dụng bao gồm các directive: ngApp, ngController, ngModel,... Ví dụ ứng dụng HelloWord dưới đây minh họa một ứng dụng viết bằng AngularJS. Ứng dụng có cấu trúc gồm 2 tập tin:

- view.html là tập tin chứa mã html và các chỉ thị để hiển thị lên trình duyệt có nội dung như sau:

```
<html>
<head>
  <script src="angular.min.js"></script>
  <script src="controllers.js"></script>
</head>
<body ng-app="HelloApp"> <h1>AngularJS Demo</h1>
  <div ng-controller="HelloCtrl">
    Tên bạn là: <input type="text" ng-model="name" />
    <p>Xin chào: <b>{{ name }}</b></p>
  </div>
</body>
</html>
```

- controller.js chứa mã xử lý có nội dung:

```
var App = angular.module('HelloApp', []);
App.controller('HelloCtrl', function HelloCtrl($scope){
  $scope.name = 'Nguyễn Tất Chủ'
});
```

Khi chạy ứng dụng sẽ được kết quả như hình 2.1

# AngularJS Demo

Tên bạn là:

Xin chào: **Nguyễn Tất Chủ**

*Hình 2.1: Kết quả chạy ứng dụng Helloworld với AngularJS*

Trong ví dụ trên mỗi thuộc tính mở rộng có dạng `ng-*` được gọi là một directive, ví dụ directive `ng-app="HelloApp"` thể hiện nơi đánh dấu cho AngularJS biết ứng dụng bắt đầu từ đâu, `ng-controller="HelloCtrl"`, sẽ là đánh dấu phần controller tương ứng được sử dụng, ngoài ra còn một số directive thường dùng:

*Bảng 2.1: Một số directive thường dùng*

STT	Directive	Mục đích sử dụng
1	<code>ng-repeat</code>	Duyệt danh sách
2	<code>ng-if</code>	Kiểm tra điều kiện
3	<code>ng-model</code> , <code>ng-bind</code>	Khai báo/Sử dụng Model
4	<code>ng-disable</code> , <code>ng-enable</code>	Cho phép thao tác với DOM hoặc không
5	<code>ng-show</code> , <code>ng-hide</code>	Ẩn hiện DOM
6	<code>ng-click</code>	Bắt sự kiện click DOM
7	<code>ng-change</code>	Bắt sự kiện Model của DOM thay đổi

**Mô hình MVC trong AngularJS:** Ý tưởng đằng sau MVC là để chia rõ 3 thành phần chính là model(cấu trúc dữ liệu), view(giao diện hiển thị), và controller(phần xử

lý logic). Đối với AngularJS view là DOM, controller là các tập tin Javascript, còn model là các dữ liệu được gán trong biến \$scope của mỗi controller.

Trong AngularJS để hiển thị dữ liệu từ controller ra view sử dụng 2 cặp ngoặc “{{}}” và để đồng bộ dữ liệu từ trên view vào model sử dụng ng-model.

Điều đặc biệt ở ví dụ trên là dù trong mã xử lý controller.js không hề tạo ra bất kỳ một sự kiện eventListener cho thẻ input nhưng khi thay đổi nội dung thẻ input này thì lời chào sẽ tự động được cập nhật tên tương ứng.

## 2.3. NoSQL và hai hệ quản trị CSDL MongoDB, Redis

### 2.3.1. Cơ sở dữ liệu NoSQL

Với các công nghệ phát triển website, hệ quản trị cơ sở dữ liệu quan hệ dựa trên SQL đã thống trị hầu hết các hệ Quản trị Cơ sở dữ liệu. Tuy nhiên thời gian gần đây, một cách tiếp cận mới đã bắt đầu biết đến là NoSQL, tạo ra sự thay thế cho các hệ quản trị cơ sở dữ liệu quan hệ truyền thống

Thuật ngữ NoSQL có nghĩa là Non-Relation hoặc Not Only SQL – không rỗng buộc hoặc phi quan hệ, ám chỉ những đến CSDL không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu trong lĩnh vực phần mềm. Bảng so sánh 2.1 dưới đây sẽ phân biệt NoSQL và SQL:

Bảng 2.2: Bảng so sánh CSDL NoSQL và cơ sở dữ liệu quan hệ truyền thống

	NoSQL	SQL
Mô hình cấu trúc	<ul style="list-style-type: none"> <li>- Không có quan hệ</li> <li>- Lưu trữ dữ liệu dưới dạng JSON, key-value, graph, ...</li> </ul>	<ul style="list-style-type: none"> <li>- Có quan hệ</li> <li>- Lưu trữ dữ liệu dưới dạng các bảng</li> </ul>
Dữ liệu	<ul style="list-style-type: none"> <li>- Linh hoạt, dữ liệu không cần lưu những thuộc tính không cần thiết</li> <li>- Có thể bổ xung thuộc tính bất cứ khi nào một cách dễ dàng</li> </ul>	<ul style="list-style-type: none"> <li>- Thêm thuộc tính có thể yêu cầu thay đổi cấu trúc các bảng hoặc dữ liệu bị ghi đè</li> </ul>

	<ul style="list-style-type: none"> <li>- Các quan hệ thường được tóm gọn và trình bày trong một đối tượng trên mỗi dòng dữ liệu</li> <li>- Tốt cho các trường hợp dữ liệu không có cấu trúc, phức tạp hoặc lồng nhau</li> </ul>	<ul style="list-style-type: none"> <li>- Quan hệ được tóm gọn và tổng quát để sử dụng kết nối và tham chiếu tới các bảng</li> <li>- Tốt cho dữ liệu có cấu trúc và các thuộc tính thường được cố định, không thay đổi</li> </ul>
--	---	--

### Những đặc điểm của CSDL NoSQL :

- Looser consistency: NoSQL tổ chức lưu trữ và truy xuất dữ liệu theo cơ chế “thoảng hơn trong đảm bảo tính nhất quán của dữ liệu” so với mô hình dữ liệu quan hệ truyền thống nhằm cải thiện hiệu suất, đảm bảo dữ liệu luôn được đáp ứng tốt hơn, CSDL này chấp nhận sự trùng lặp dữ liệu.
- Eventual consistency: NoSQL không yêu cầu phải đảm bảo tính nhất quán của dữ liệu ngay tức thì mà sẽ hiện thực tính nhất quán của dữ liệu theo cơ chế lan truyền
- Distributed storage: hay còn gọi là lưu trữ phân tán là thay vì dữ liệu được lưu trữ trên một máy chủ duy nhất thì hệ thống sẽ lưu dữ liệu trên nhiều máy khác nhau dưới sự kiểm soát của phần mềm
- Horizontal scalable: hay còn gọi là khả năng mở rộng chiều ngang. Bình thường, với các hệ quản trị cơ sở dữ liệu quan hệ, khi mà dữ liệu quá lớn phương pháp tăng khả năng lưu trữ là sẽ phải mở rộng (nâng cấp máy chủ), còn đối với NoSQL thì chỉ cần bổ sung thêm máy chủ khác vì hệ thống hỗ trợ lưu trữ phân tán trên nhiều máy

### Ưu điểm của NoSQL:

- NoSQL là mã nguồn mở: điều này có nghĩa là bạn sẽ dễ dàng phát triển một ứng dụng có sử dụng NoSQL mà không phải tốn chi phí license.

- Dễ mở rộng quy mô: NoSQL đã thay thế cách mở rộng quy mô truyền thống của các hệ quản trị cơ sở dữ liệu quan hệ bằng hình thức “mở rộng ra ngoài”. Với hình thức mở rộng đặc biệt này, thay vì phải bổ sung thêm các máy chủ lớn hơn vào hệ thống khi dữ liệu lớn lên, thì NoSQL lại hỗ trợ doanh nghiệp phân tán dữ liệu qua nhiều máy chủ khi dữ liệu gia tăng.
- Hỗ trợ các mô hình dữ liệu khác nhau tuỳ mục đích và cách thức lưu trữ dữ liệu như lưu kiểu key-value, BigTable, lưu document hay lưu thông tin graph

Nhược điểm của CSDL NoSQL:

- Thiếu tính tương thích: các CSDL NoSQL khác nhau không tương thích với nhau, mỗi CSDL NoSQL có một giao diện và cung cấp API khác nhau và chưa có một tiêu chuẩn chung nào. Điều này có nghĩa là bạn sẽ gặp khó khăn trong việc chuyển từ nhà cung cấp này sang nhà cung cấp khác
- Hạn chế về nghiệp vụ: NoSQL hiện chưa hỗ trợ các dạng phân tích dữ liệu lớn và mạnh mẽ mà các doanh nghiệp đã quen thuộc trong các RDBMS

Phân loại các CSDL NoSQL: Có nhiều cách phân loại các cơ sở dữ liệu NoSQL khác nhau, mỗi loại với các loại và loại con khác nhau, một số trong số đó có thể chồng chéo lên nhau. Dưới đây là một phân loại cơ bản dựa trên mô hình dữ liệu, chia các hệ quản trị CSDL thành các nhóm khác nhau:

- Column: Accumulo, Cassandra, Druid, HBase, Vertica
- Document: Apache CouchDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, Lotus Notes, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB
- Key-value: Aerospike, CouchDB, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, Berkeley DB
- Graph: AllegroGraph, InfiniteGraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Stardog
- Multi-model: Alchemy Database, ArangoDB, CortexDB, FoundationDB, MarkLogic, OrientDB

Một phân loại chi tiết hơn như sau, dựa trên cách phân loại của Stephen Yen<sup>[7]</sup>:

*Bảng 2.3: Bảng phân loại hệ quản trị CSQL NoSQL*

Loại	Hệ quản trị CSDL
Key-Value Cache	Coherence, eXtreme Scale, GigaSpaces, GemFire, Hazelcast, Infinispan, JBoss Cache, Memcached, Repcached, Terracotta, Velocity
Key-Value Store	Flare, Keyspace, RAMCloud, SchemaFree, Hyperdex, Aerospike
Key-Value Store (Eventually-Consistent)	DovetailDB, Oracle NoSQL Database, Dynamo, Riak, Dynomite, MotionDb, Voldemort, SubRecord
Key-Value Store (Ordered)	Actord, FoundationDB, Lightcloud, LMDB, Luxio, MemcacheDB, NMDB, Scalaris, TokyoTyrant
Máy chủ Cấu trúc dữ liệu	Redis
Tuple Store	Apache River, Coord, GigaSpaces
Cơ sở dữ liệu đối tượng	DB4O, Objectivity/DB, Perst, Shoal, ZopeDB
Document Store	Clusterpoint, Couchbase, CouchDB, DocumentDB, Lotus Notes, MarkLogic, MongoDB, Qizx, RethinkDB, XML-databases
Wide Column Store	BigTable, Cassandra, Druid, HBase, Hypertable, KAI, KDI, OpenNeptune, Qbase

### 2.3.2. Hệ quản trị CSDL MongoDB

#### Các đặc điểm của MongoDB:

- Lưu trữ dữ liệu hướng document dưới dạng BSON gần giống JSON.
- Các Collection(bộ sưu tập) tương tự như các Table trong CSDL quan hệ
- Có chỉ mục cho các bản ghi
- MongoDB có hai phiên bản Community(miễn phí) và Enterprise(trả phí). Mã nguồn mở giấy phép GNU. Được viết bằng C++, Javascript, Python có thể chạy trên tất cả các hệ điều hành hiện nay.

- Hỗ trợ nhiều kiểu dữ liệu: null, integer, long, doubles, decimals, symbol, string, object, array, BinData, ObjectId, Boolean, Date, Timestamp, Regular Expression, MaxKey, MinKey

Ngoài các đặc điểm nổi bật của một CSDL NoSQL, MongoDB còn rất phù hợp cho các ứng dụng cỡ vừa và lớn vì nó được thiết kế để thao tác khá giống với một CSDL SQL, mọi thao tác dữ liệu đều có thể thực hiện giống như một CSDL SQL. Bảng so sánh 2.3 dưới đây thể hiện điều này:

Bảng 2.4: Các khái niệm tương đương giữa SQL DB và MongoDB

SQL DB	MongoDB
Table	Colection
Row	Document
Column	Feild
Joins	Embeded documents, linking
Group	Aggregation
Primary key (Chỉ định cột bất kỳ làm khoá chính)	Primary key, MongoDB mặc định tạo ra feild _id làm khoá
Foreign Key	Reference Schema

Một ví dụ bản ghi (document) với MongoDB:

```
{
  "_id": ObjectId('5816bed4a2b7a9f009f2f2bb'),
  "full_name": 'Nguyễn Tất Chủ',
  "date_of_birthday": "1995-04-10T14:02:20+07:00",
  "is_male": true,
  "average_score": null,
  "detail_scope": [
    {
      "course": "Nhập môn lập trình",
      "scope": 9.0
    },
    {
      "course": "Tin học cơ sở",
      "scope": 9.0
    }
  ]
}
```

```

    ]
}
```

Các thao tác với các bản ghi cũng được thiết kế tương tự. MongoDB còn hỗ trợ thao tác qua Command line, dưới đây là một số thao tác cơ bản với MongoDB:

Bảng 2.5: Các thao tác với MongoDB

SQL DB	MongoDB
Create table <i>CREATE TABLE TableName(&lt;columns - type&gt;)</i>	Add Collection <i>db.createCollection(" &lt;CollectionName&gt; ")</i>
Read <i>SELECT feild1, feild2 from TableName WHERE feild3 = "value" LIMIT 10 SKIP 20</i>	Find <i>db.CollectionName.find({feild3: "value"}, {feild1, feild2}).limit(10).skip(20)</i>
Insert <i>INSERT INTO TableName (column1, column2) VALUES (value1, value2)</i>	Insert <i>db.CollectionName.insert({column1: value1, column2: value2})</i>
Update <i>UPDATE TableName SET column1 = value1, column2 = value2, ... WHERE condition;</i>	Update <i>db.CollectionName.updateOne( condition, { column1: value1, column2: value2}, &lt;options&gt; )</i>
Delete <i>DELETE FROM TableName WHERE condition;</i>	Remove <i>db.CollectionName.deleteMany(condition)</i>
Search <i>SELECT * FROM users WHERE name LIKE "Prefix%"</i>	Find (sử dụng biểu thức RegExp) <i>db.users.find({name:/^Prefix/})</i>
Order <i>SELECT column1, column2, ... FROM TableName</i>	Sort <i>db.CollectionName.find({}).sort('column1 - column2')</i>

<i>ORDER BY column1, column2, ... ASC DESC</i>	
Index <i>CREATE INDEX index_name ON table_name (column1, column2, ...);</i>	Index <i>db.CollectionName.createIndex(keys, &lt;options&gt;)</i>

### Cách cài đặt và sử dụng:

MongoDB có thể cài đặt bằng cách tải bản cài đặt đầy đủ tại website [8] hoặc dưới dạng host services của hệ thống bằng các trình cài đặt gói:

- Trên MacOS: *brew install mongodb*
- Trên Windows: *choco install mongodb*
- Trên Linux: *sudo apt-get install -y mongodb-org*

Sau khi cài đặt, MongoDB thường chạy dịch vụ ở địa chỉ 127.0.0.1 port 27017(<http://localhost:27017>). Để sử dụng MongoDB với các ngôn ngữ lập trình, cần thêm bộ driver tương ứng. Hiện nay MongoDB có thể sử dụng với hầu hết các ngôn ngữ, NodeJS sử dụng với module mongoose, C# sử dụng Nuget với các package MongoDB.Driver, MongoDB.Driver.Core, MongoDB.Driver.BSON; PHP sử dụng extension mongodb,...

#### 2.3.3. Hệ thống lưu trữ cache máy chủ Redis

Redis là một lựa chọn tốt nhất trong những việc xây dựng các ứng dụng cần lưu trữ dữ liệu dạng Memcached. Memcached là một dạng lưu trữ có các đặc điểm: lưu trữ dữ liệu theo dạng key-value, tất cả dữ liệu trên RAM, dữ liệu có thể hết hạn hoặc không và đặc điểm quan trọng nhất của Memcached đó là có thể truy xuất rất nhanh.

Redis giống MongoDB thuộc dạng CSDL NoSQL lưu trữ dữ liệu dạng key – value. Bên cạnh lưu trữ key-value trên RAM với hiệu năng cao, redis còn hỗ trợ lưu trữ dữ liệu trên đĩa cứng (persistent redis) cho phép phục hồi dữ liệu khi gặp sự cố. Ngoài tính năng replication (sao chép giữa master-client), tính năng cluster (sao lưu master-master) cũng đang được phát triển.

Ngoài những đặc điểm phù hợp cho Memcached, Redis còn có các ưu điểm:

- Hỗ trợ nhiều Databases
- Truy vấn theo Key
- Hỗ trợ counters dữ liệu kiểu integer
- Cấu trúc dữ liệu cấp cao: key là một string nhưng value thì không giới hạn ở một string mà có thể là List, Sets, Sorted sets, ...
- Thao tác dữ liệu chuyên biệt
- Tự động phân trang danh sách
- Redis hỗ trợ mở rộng master-slave nếu chúng ta muốn sự an toàn hoặc mở rộng, co giãn trong việc lưu trữ data

Các cài đặt và sử dụng Redis: Redis là một phần mềm mã nguồn mở nên nhà phát triển có thể cài tự biên dịch chương trình để chạy theo cơ chế services của hệ điều hành hoặc sử dụng các trình cài đặt gói. Ví dụ:

- Trên macOS: *brew install redis*
- Trên Windows: *choco install redis-64*
- Trên Linux cần phải biên dịch mã nguồn từ [<sup>9</sup>]

Việc sử dụng Redis được tích hợp tùy theo framework khác nhau. Bộ thư viện làm việc có thể tham khảo tại [<sup>10</sup>]

## **2.4. Thuật toán phân lớp dữ liệu bằng Neural Network**

Mạng neural nhân tạo (Artificial Neural Networks : ANN) ra đời xuất phát từ ý tưởng mô phỏng hoạt động của bộ não con người. Mạng neural nhân tạo là sự tái tạo bằng kỹ thuật những chức năng của hệ thần kinh con người với vô số các neural được liên kết truyền thông với nhau qua mạng. Giống như con người, ANN được học bởi kinh nghiệm, lưu những kinh nghiệm đó và sử dụng trong các tình huống. Trong một vài năm trở lại đây đã được nhiều người quan tâm và đã áp dụng thành công trong nhiều lĩnh vực khác nhau, như tài chính, y tế, địa chất và vật lý. Thật vậy, bất cứ ở đâu có vấn đề về dự báo, phân loại và điều khiển, mạng neural đều có thể ứng dụng được. Ví dụ như khả năng nhận dạng mặt người trong các hệ thống quản lý thông tin liên quan đến con người

(quản lý nhân sự ở các công sở, doanh nghiệp; quản lý học sinh, sinh viên trong các trường trung học, đại học và cao đẳng,...); các ngành khoa học hình sự, tội phạm; khoa học tướng số, tử vi,... Trong phần này sẽ trình bày ý tưởng xây dựng mạng nơ-ron từ mô hình nơ-ron sinh học, cùng với đó là cơ sở toán học của giải thuật và các vấn đề trong quá trình xây dựng mạng.

Nghiên cứu về noron thần kinh từ lâu đã trở thành đề tài được nhiều nhà khoa học quan tâm. Nhưng ký nguyên của mạng nơ-ron chính thức được bắt đầu với báo cáo khoa học của Mc Culloch và Pitts năm 1943<sup>[11]</sup> mô tả một phép tính logic của mạng nơ-ron. Báo cáo này được công chúng đón nhận cho đến năm 1949 học thuyết về mạng nơ-ron chính thức của Mc. Culloch và Pitts được mô tả chủ yếu trong bài giảng thứ hai trong bốn bài giảng mà Von Neumann đã phát biểu tại trường đại học Illinois. Sự phát triển tiếp theo của quá trình nghiên cứu mạng nơ-ron được đánh dấu vào năm 1949 với việc xuất bản cuốn sách “The Organization of Behavior: A neuropsychological Theory” của Donald Olding Hebb. Đến năm 1952, cuốn sách của Ashby “Design for a Brain” đã mô tả những điều kiện cần và đủ đối với một hệ thống hoạt động giống như bộ não “đó là phải học để còn tồn tại trong môi trường luôn thay đổi và nhận được những cái nó cần”. Tiếp đó, năm 1954, Minsky đã viết luận án tiến sĩ mang tên “Theory of Neural-analog Reinforcement Systems and Application to Brain-Model Problem” tại trường Đại học Princeton, sau đó là bài báo của ông “Steps Toward Artificial Intelligence” năm 1961 về việc học củng cố trong mạng nơ-ron hiện nay. Một chủ đề khác được đánh giá cao là phát triển về bộ nhớ liên kết của Taylor vào năm 1956, mở đầu một loạt các kết quả phát triển to lớn về sau. Các kết quả có thể kể đến là sự ra đời của mạng Perceptron được Frank Rosenblatt công bố vào năm 1957 và được coi là “mạng nơ-ron truyền thông đơn giản nhất”. Tiếp đó năm 1960, mạng nơ-ron khác được Bernard Widrow và Marcian Hoff giới thiệu là ADALINE (ADAptive LINear Element). Với mạng ADALINE lần đầu tiên kiểu hội tụ các mạch con chứa trọng số trước node tổng được sử dụng để phân lớp các mẫu. Năm 1969, Minsky và Papert xuất bản cuốn “Perceptron, An Introduction to Computational Geometry” chỉ ra những giới hạn trong mạng Perceptron một lớp đơn và đề nghị khắc phục trong mạng Perceptron nhiều lớp. Những năm 1970 nổi bật với các sự kiện: các mạng liên kết của Kohonen và Anderson (1972), Cognitron – mạng tự

tổ chức nhiều lớp đầu tiên do Kunihiko Fukushima người Nhật giới thiệu năm 1975. Thời kì phát triển vượt bậc của quá trình nghiên cứu mạng nơron là những năm 80 với sự ra đời của một loạt các mạng nơron có giá trị: mạng Hopfield của John Hopfield (1982), SOM (Self-Organization Map) của Kohonen, máy Boltzmann của Ackley, Hinton và Sejnowski. Tiếp theo là sự ra đời của mạng Back- Propagation năm 1986 do D. Rumelhart, G. Hilton và R. Williams giới thiệu, sau đó là mạng ART (Adaptive Resonance Networks) vào năm 1987 của Gail Carpenter và Stephen Grossberg tại đại học Boston. Trong những năm 1990 việc ứng dụng các mô hình mạng nơron vào phục vụ các lĩnh vực trong cuộc sống được phát triển mạnh mẽ.

#### **2.4.1. Ý tưởng xây dựng mạng Neural nhân tạo**

Mạng nơron nhân tạo được thiết kế dựa trên mô hình mạng nơron thần kinh với các phần tử nơron nhân tạo của nó là sự mô phỏng nơron sinh học. Các mức tổ chức bộ não và cấu trúc mạng nơron sinh vật có thể được tham khảo trong tài liệu. Trong phần này chỉ tập trung tìm hiểu cấu tạo của nơron sinh học và nơron nhân tạo để thấy được sự tương quan giữa chúng.

Nơron sinh vật có nhiều dạng khác nhau như dạng hình tháp, dạng tò ong, dạng rẽ cây. Tuy khác nhau về hình dạng, chúng có cấu trúc và nguyên lý hoạt động chung. Một tế bào nơron gồm bốn phần cơ bản sau:

Các nhánh và rẽ: Các nhánh và rẽ là các bộ phận nhận thông tin, các đầu nhạy hoặc các đầu ra của các nơron khác bám vào rẽ hoặc nhánh của một nơron. Khi các đầu vào từ ngoài này có sự chênh lệch về nồng độ  $K^+$ ,  $Na^+$  hay  $Cl^-$  so với nồng độ bên trong của nó thì xảy ra hiện tượng thẩm thấu từ ngoài vào trong thông qua một cơ chế màng thẩm đặc biệt. Hiện tượng thẩm thấu như vậy tạo nên một cơ chế truyền đạt thông tin với hàng ngàn hàng vạn lối vào trên một nơron sinh vật, ứng với hàng nghìn hàng vạn liên kết khác nhau. Mức độ thẩm thấu được đặc trưng bởi cơ chế màng tượng trưng bằng một tỷ lệ. Tỷ lệ đó được gọi là tỷ trọng hay đơn giản gọi là trọng (Weight).

Thân thần kinh (Soma): Thân thần kinh chứa các nhân và cơ quan tổng hợp protein. Các ion vào được tổng hợp và biến đổi. Khi nồng độ các ion đạt đến một giá trị

nhất định, xảy ra quá trình phát xung (hay kích thích). Xung đó được phát ở các đầu ra của nơron. Dây dẫn đầu ra xung được gọi là dây thần kinh.

Dây thần kinh (Axon): Dây thần kinh là đầu ra. Đó là phương tiện truyền dẫn tín hiệu. Dây thần kinh được cấu tạo gồm các đốt và có thể dài từ micro mét đến vài mét tùy từng kết cấu cụ thể. Đầu ra này có thể truyền tín hiệu đến các nơron khác.

Khớp thần kinh là bộ phận tiếp xúc của đầu ra nơron với rễ, nhánh của các nơron khác. Chúng có cấu trúc màng đặc biệt để tiếp nhận các tín hiệu khi có sự chênh lệch về nồng độ ion giữa bên trong và bên ngoài. Nếu độ lệch về nồng độ càng lớn thì việc truyền các ion càng nhiều và ngược lại. Mức độ thẩm thấu của các ion có thể coi là một đại lượng thay đổi tùy thuộc vào nồng độ như một giá trị đo thay đổi được gọi là trọng.

Trong não người có khoảng 15 tỷ nơron, mỗi nơron được nối với nhiều nơron khác bằng những khớp thần kinh (synapses), một nơron có thể có đến hơn nghìn synapses, và số synapses tổng cộng lại được ước lượng khoảng 1 triệu tỷ. Tín hiệu được truyền qua thân nơron tới các synapses, và tuỳ theo trạng thái của chúng mà một hay nhiều nơron khác sẽ được kích thích để tiếp tục truyền tín hiệu. Đó là mức thấp nhất, nhưng từ đó tổ chức lên các tầng trên như thế nào, có thể có được một bản đồ chi tiết của một bộ não tới từng nơron không?

Mạng nơron sinh vật tổ chức thành từng lớp (layer). Bao gồm:

- **Mạng một lớp:** là tập hợp các phần tử nơron có đầu vào và đầu ra trên mỗi một phần tử. Nếu mạng nối đầu ra của các phần tử này với đầu vào của phần tử kia gọi là mạng tự liên kết (autoassociative)
- **Mạng hai lớp:** gồm một lớp đầu vào và một lớp đầu ra riêng biệt.
- **Mạng nhiều lớp:** gồm một lớp đầu vào và một lớp đầu ra riêng biệt. Các lớp nằm giữa lớp đầu vào và lớp đầu ra gọi là lớp ẩn (hidden layers)
- **Mạng truyền thẳng:** là mạng hai hay nhiều lớp mà quá trình truyền tín hiệu từ đầu ra lớp này đến đầu vào lớp kia theo một hướng
- **Mạng truyền ngược:** là mạng mà trong đó một hoặc nhiều đầu ra của các phần tử lớp sau truyền ngược tới đầu vào của lớp trước

- Mạng truyền ngược: là mạng mà trong đó một hoặc nhiều đầu ra của các phần tử lớp sau truyền ngược tới đầu vào của lớp trước

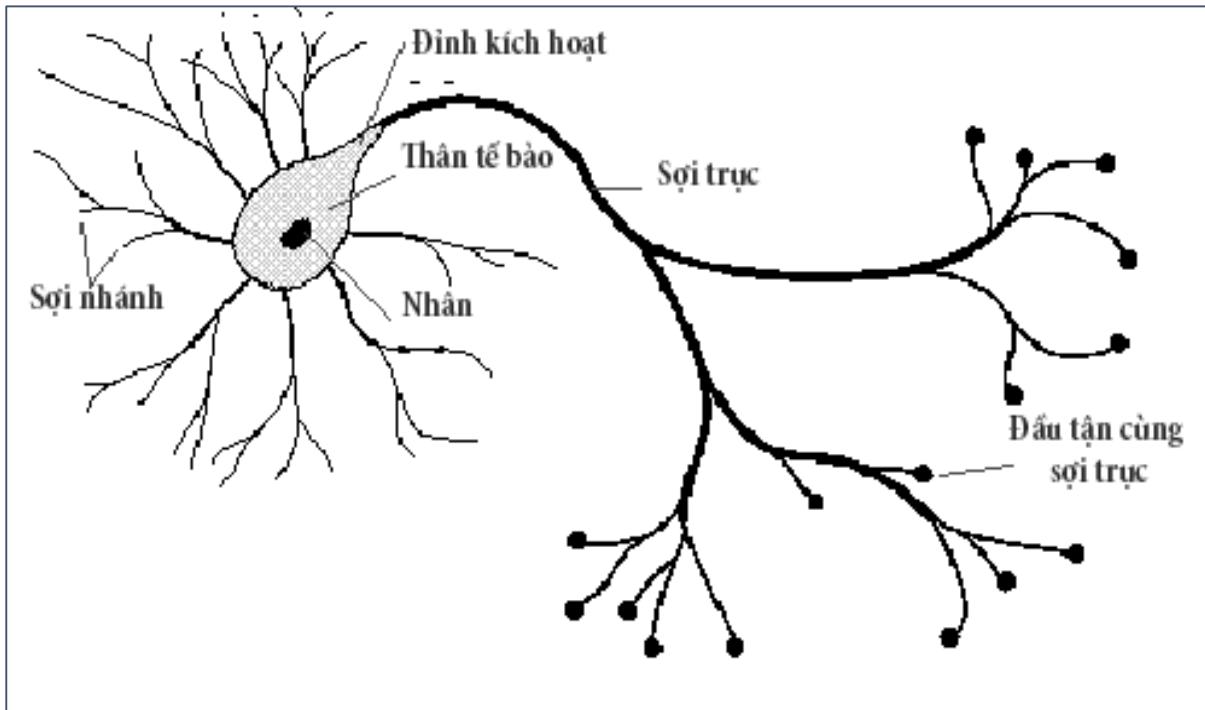
Một tế bào noron gồm các phần cơ bản sau:

Nhánh và rẽ: Đây là bộ phận tiếp nhận thông tin của tế bào noron. Các đầu nhạy hoặc các đầu ra của các noron khác bám vào rẽ hoặc nhánh của một noron. Khi các đầu vào từ ngoài này có sự chênh lệch về nồng độ  $K^+$ ,  $Na^+$  hay  $Cl^-$  so với nồng độ bên trong của nó thì xảy ra hiện tượng thấm từ ngoài vào trong thông qua cơ chế màng thấm đặc biệt. Hiện tượng thấm thấu như vậy tạo nên một cơ chế truyền đạt thông tin với hàng ngàn hàng vạn lối vào trên một noron sinh học ứng với hàng nghìn hàng vạn liên kết khác nhau. Mức độ thấm thấu của các ion có thể coi là một đại lượng thay đổi tùy thuộc vào nồng độ gọi là tỷ trọng hay đơn giản là trọng số (weight)

Dây thần kinh: Đây là phương tiện truyền dẫn tín hiệu của giữa các noron. Dây thần kinh được cấu tạo gồm các đốt và có thể dài từ vài micro mét đến vài mét tùy kết cấu cụ thể.

Thân thần kinh: Thân thần kinh chứa các nhân và cơ quan tổng hợp protein. Trong quá trình các ion vào được tổng hợp và biến đổi khi nồng độ các ion đạt đến một giá trị nhất định, xảy ra quá trình phát xung (hay kích thích). Xung đó được phát ở đầu ra của noron thông qua dây thần kinh

**Khớp thần kinh:** Là bộ phận tiếp xúc của đầu ra nơron với nhánh, rẽ của các nơron khác. Chúng có cấu trúc màng đặc biệt để tiếp nhận các tín hiệu khi có sự chênh lệch về nồng độ ion giữa bên trong và bên ngoài. Nếu độ lệch càng lớn thì việc truyền các ion càng nhiều và ngược lại.



Hình 2.2: Cấu trúc của một Neural sinh học

Ở mức độ đơn giản nhất, khi bị kích thích từ các tác nhân (có thể là các nơ-ron khác hoặc ảnh hưởng bên ngoài) các nơ-ron sản sinh ra các xung thần kinh mà ta gọi là điện thế hoạt động.

Khi một nơ-ron ở trạng thái nghỉ ngơi, nó được phân cực. Mặc dù nó không nhận được bất kỳ tín hiệu điện nào từ các nơ-ron khác, nó vẫn được nạp và sẵn sàng phát ra các xung thần kinh. Mỗi nơ-ron có một ngưỡng kích thích riêng, nếu tín hiệu tiếp nhận vượt qua ngưỡng này sẽ khiến nơ-ron sản sinh ra các xung thần kinh. Do đó, chỉ khi một nơ-ron nhận đủ kích thích (từ một hay nhiều nguồn) nó mới tạo ra một xung.

Cơ chế truyền tải các xung dựa trên sự trao đổi các ion trong dung môi xung quanh tế bào. Các tín hiệu di chuyển rất chậm chạp, khoảng vài trăm mét mỗi giây.

Các xung thần kinh chỉ kéo dài khoảng vài mili giây. Nếu một nơ-ron nhận rất nhiều kích thích thì sẽ tự sản xuất ra một tín hiệu mạnh, gồm nhiều xung mỗi giây.

Xi-nap là phần tử quan trọng đối với chức năng của một nơ-ron và quá trình học. Sợi trực kết thúc trong một bầu nhỏ gọi là bầu xi-nap và phân cách với nơ-ron kẽ bởi một khe gọi là khe xi-nap. Khi một xung di chuyển tới đoạn cuối của sợi trực, nó kích thích việc tạo ra chất dẫn truyền thần kinh trong bầu xi-nap và chất này di chuyển qua khe, kích thích nơ-ron tiếp theo.

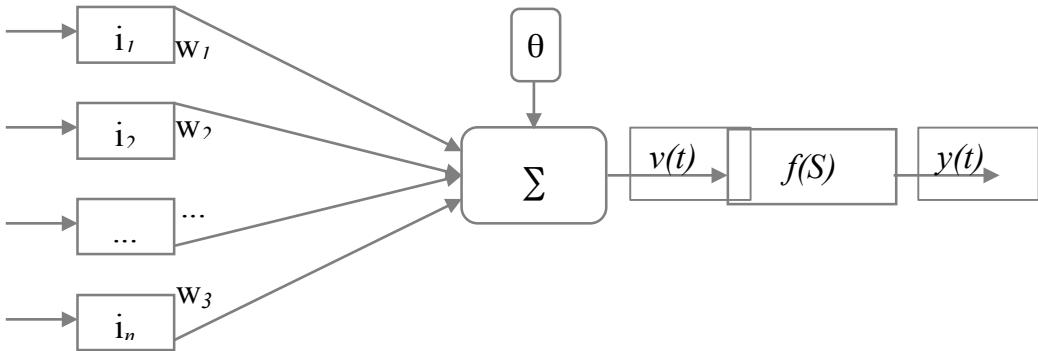
Năm 1949, Donald Hebb chỉ ra một cách để mạng nơ-ron học. Nếu một xi-nap hoạt động nhiều sẽ trở nên mạnh và sản xuất ra nhiều hơn chất dẫn truyền thần kinh. Từ đó, tuyến dẫn truyền này trở nên mạnh hơn và những tuyến khác yếu đi. Ta có thể xem như có một trọng số liên kết, giá trị này càng lớn sẽ sản xuất ra nhiều kích thích hơn. Đây là những bước đầu tiên để hiểu cơ chế học của mạng thần kinh

Những nghiên cứu trên đây là cơ sở mở đường cho con người nghiên cứu phương pháp mô phỏng lại mạng nơron sinh vật, đó chính là mạng nơron nhân tạo.

#### **2.4.2. Cấu trúc Neural nhận tạo**

Năm 1943, Warren McCulloch và Walter Pitts đưa ra một mô hình đơn giản các nơ-ron nhân tạo. Đây cũng chính là bước khởi đầu lịch sử của ANN. Cho tới tận ngày nay, mô hình này vẫn được xem như là nền tảng cho hầu hết các ANN. Ở đây, các nơ-ron được gọi là các Perceptron.

Trên cơ sở cấu trúc của nơron sinh học người ta đề xuất mô hình nơron nhân tạo gồm 3 phần chính: Bộ tổng liên kết đầu vào, bộ động học tuyến tính và bộ phi tuyến. Nơron nhân tạo là sự mô phỏng nơron sinh học. Sơ đồ dưới đây minh họa cụ thể cấu trúc của một Neural:



Hình 2.3: Mô hình cấu trúc neural nhân tạo

Trong sơ đồ trên:

- $n$ : Số lượng đầu vào mô tả tín hiệu vào từ các đầu nhạy thần kinh hoặc các nơron khác
- $x_i(t)$ : Các đầu vào ngoài; với  $i$  là chỉ số chạy,  $i=1,2,\dots,n$
- $w_i$ : Trọng số liên kết ngoài giữa các đầu vào  $i$  tới nơron  $t$
- $\theta$ : Độ lệch (bias), là hằng số xác định ngưỡng kích thích hay ức chế
- $v(t) = \sum_{i=1}^n w_i x_i(t) - \theta = S$ : hàm tổng các đầu vào tác động ở thân nơron  $t$
- $y(t)$ : Tín hiệu đầu ra nơron  $t$
- $f$ : hàm kích hoạt

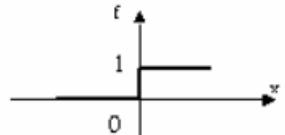
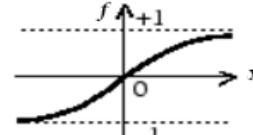
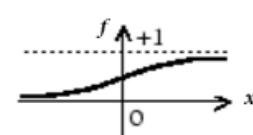
Biểu đồ cho ta thấy đầu vào của một nơ-ron được biểu diễn bởi  $x$  (theo quan điểm sinh học, đây chính là các tác động từ các nơ-ron khác kết nối đến hoặc từ thế giới bên ngoài, truyền đến nơ-ron thông qua các sợi nhánh). Mỗi đầu vào tỉ trọng với một hệ số biểu diễn cường độ kết nối xi-nap của các sợi nhánh của nơ-ron, được kí hiệu là  $w$ . Tổng các tín hiệu vào và trọng số của nó được gọi là sự hoạt hóa nơ-ron, được kí hiệu bởi  $v(t)$ .

Sau khi đã tính toán được  $v(t)$ , ta sử dụng một ngưỡng để xác định đầu ra của nơ-ron.

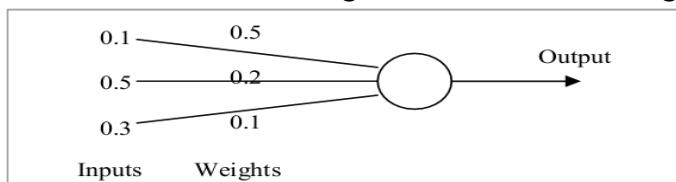
Tóm lại, ta thấy các nơ-ron tiếp nhận tín hiệu vào, tùy thuộc vào cường độ kết nối mà mỗi tín hiệu được tăng cường hay kìm hãm theo giá trị trọng số, rồi tổng hợp chúng lại và kiểm tra với ngưỡng kích hoạt  $f(S)$ , tùy theo hàm ngưỡng này mà giá trị truyền đi là khác nhau.

Hàm kích hoạt là hàm thể hiện đặc điểm kích hoạt hay ức chế một neural. Đặc điểm chung của hàm này là bị chặn, đơn điệu tăng và tăng liên tục. Các hàm này được chia thành hai nhóm là hàm bước nhảy và hàm liên tục. Bảng 2.5 sau đây liệt kê một số hàm thường được sử dụng:

Bảng 2.6: Các hàm kích hoạt thường dùng trong mô hình neural nhân tạo

Tên hàm	Công thức	Đồ thị minh họa
Hàm ngưỡng	$f(x) = \begin{cases} 1 & \text{nếu } x \geq \alpha \\ 0 & \text{nếu } x < \alpha \end{cases}$	 $\alpha = 1$
Hàm tang-hyperbol	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
Hàm sigmoid nhị phân	$f(x) = \frac{1}{1 + e^{-x}}$	

Ví dụ tính toán trên một nơ-ron đơn giản với một neural có giá trị ngưỡng là 0.5.



Hình 2.4: Minh họa một Neural nhân tạo

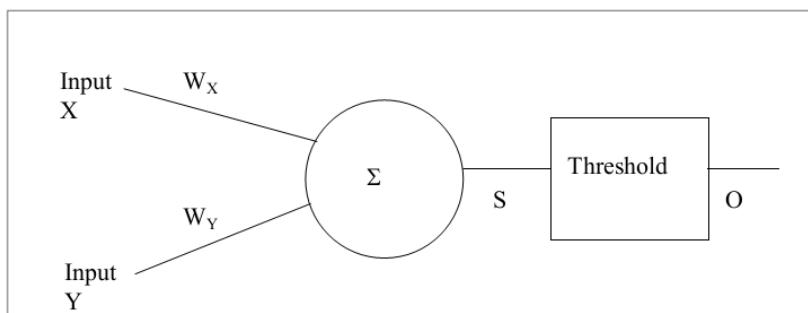
Ta có tổng  $S = v() = (0.1 \times 0.5) + (0.5 \times 0.2) + (0.3 \times 0.1) = 0.05 + 0.1 + 0.03 = 0.18$

- Với hàm ngưỡng vì  $0.18 < 0.5$  nên Output sẽ là:  $y() = 0$ ;
- Nếu áp dụng hàm sigmoid Output sẽ là  $y() = f_{(0.18)} = \frac{1}{1+e^{-0.18}} = 0.545$

### 2.4.3. Mạng Neural nhân tạo và khái niệm học

Những năm cuối thập niên 1950, Rosenblatt thành công trong việc tạo ra các mạng nơ-ron nhân tạo thực nghiệm đầu tiên. Các mạng này mang những đặc tính đáng kinh ngạc và một trong số chúng được ứng dụng trong nhận dạng mẫu.

Để biết được tại sao chúng ta thường nghiên cứu cấu trúc của một mạng các nơ-ron thay vì các nơ-ron đơn lẻ, ta cần xem xét một bài toán kinh điển trong mạng Nơ-ron: phép toán XOR. Cấu trúc nơ-ron của bài toán này là một nơ-ron với hai đầu vào ứng với hai toán tử của phép toán và một đầu ra.



Hình 2.5: Cấu trúc Neural của phép toán XOR

Dù ta có thay đổi các số  $W_x$  và  $W_y$  như thế nào đi nữa thì nơ-ron cũng thế không cho ra kết quả đúng được bảng chân trị.

Bảng 2.7: Bảng chân trị của phép toán XOR

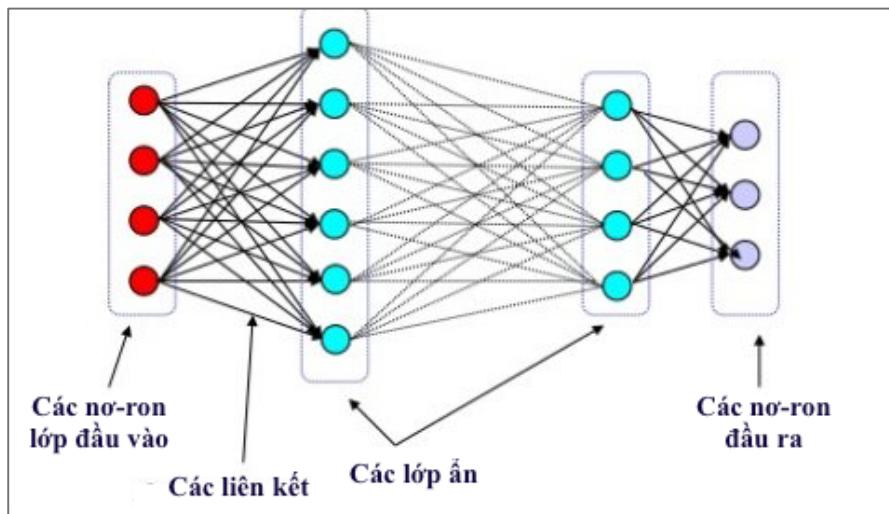
X1	X2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Đó là lý do mà chúng ta thường nghiên cứu cấu trúc mạng neural chứ không phải là những neural đơn lẻ. Năm 1969, Marvin Minsky và Seymour Papert xuất bản cuốn sách “Perceptrons”, đưa ra các phê bình cũng như các lí lẽ tranh cãi về mạng Nơ-ron,

khiến cho việc nghiên cứu trong lĩnh vực này bị tác động nặng nề. Lý lẽ chính được sử dụng trong “Perceptrons” là việc một nơ-ron Perceptron đơn giản không có khả năng mô phỏng một cổng XOR hai đầu vào, từ đó họ chỉ ra rằng phải chăng một bộ xử lý quá hạn chế đến nỗi không thể thực hiện một công việc đơn giản đến vậy.

Mặc dù hạn chế này khiến cho quá trình nghiên cứu mạng nơ-ron bị trì hoãn một khoảng thời gian dài nhưng ta lại dễ dàng giải quyết nó bằng cách đơn giản là sử dụng một mạng nơ-ron thay cho một nơ-ron riêng lẻ. Thực tế, nhà toán học Kolmogorov người Nga chứng minh được rằng mạng nơ-ron ba lớp có thể học bất kì hàm nào miễn cung cấp đủ nơ-ron cho nó. Do vậy ngày nay một mạng các nơ-ron được sử dụng thay vì các nơ-ron đơn giản để nhận dạng các mẫu phức tạp.

Lấy ý tưởng từ mô hình nơron sinh học liên kết với nhau tạo thành mạng nơron sinh học mạng nơ-ron nhân tạo, mạng nơ-ron với cấu trúc tương tự trong đó một neural là một khối tính toán gồm nhiều đầu vào, mỗi đầu vào có một trọng số đặc trưng cho tính ức chế hay kích hoạt giữa các neural, mỗi neural còn được gọi là các nút (node). Có nhiều cách kết hợp các nút để thành một mạng neural, mỗi cách kết hợp tạo thành một lớp mạng nơron nhân tạo khác nhau, chức năng của mạng được xác định bởi các đặc điểm như cấu trúc liên kết, quá trình xử lý bên trong từng nút và mức độ liên kết các nút với nhau.



Hình 2.6: Mô hình chung của mạng Neural đa lớp

Kiến trúc mạng neural nhân tạo được xác định bởi các yếu tố:

- Số lượng các tín hiệu đầu vào và đầu ra
- Số lượng các lớp
- Số lượng nút của mỗi lớp
- Trọng số liên kết
- Cách thức liên kết giữa các neural trên một lớp hoặc giữa các lớp với nhau

Dựa trên kiến trúc của mạng neural người ta chia mạng neural thành các nhóm khác nhau, có nhiều cách phân loại mạng dựa trên các tiêu chí khác nhau.

- Dựa trên số lượng lớp trong mạng chia thành 2 loại là mạng neural một lớp và mạng nhiều lớp
- Dựa trên cách thức liên kết, phân thành 3 loại: mạng truyền thẳng, mạng hồi quy và mạng tự tổ chức.

Tương tự như bộ não con người có thể lưu trữ và tiếp nhận thông tin có thể xem trọng số là phương pháp để mạng neural lưu trữ thông tin tri thức giúp máy tính có thể phân biệt và dự đoán được những thông tin đã được học. Khái niệm học được hiểu theo hai nghĩa đó là **học cấu trúc** và **học tham số**.

**Học tham số** là thay đổi, cập nhật các trọng số liên kết theo một cách nào đó làm cho chúng có thể thực hiện tốt hơn trong tương lai. Hầu hết các luật học tồn tại thuộc kiểu học tham số, mục tiêu của quá trình học là cập nhật các trọng số để tìm ra một liên kết hoàn chỉnh cùng với một bộ trọng số tối ưu nhất. Thông thường, luật học tham số được chia thành ba dạng chính, đó là: *học giám sát*, *học không giám sát* và *học củng cố*.

*Học có giám sát* là quá trình học có sự tham gia giám sát của một “thầy giáo”. Cũng giống như việc ta dạy một em nhỏ các chữ cái. Ta đưa ra một chữ “a” và bảo với em đó rằng đây là chữ “a”. Việc này được thực hiện trên tất cả các mẫu chữ cái, lặp đi lặp lại một số lần nhất định cho đến khi em nhỏ có thể nhớ được chữ đó dựa vào hình dáng của nó. Sau đó khi kiểm tra ta sẽ đưa ra một chữ cái bất kỳ (với cách có thể viết hơi khác đi) và hỏi em đó đây là chữ gì?

Với học có giám sát, số lớp cần phân loại đã được biết trước. Nhiệm vụ của thuật toán là phải xác định được một cách thức phân lớp sao cho với mỗi đầu vào sẽ được phân loại chính xác vào lớp của nó.

*Học cũng có* 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, mỗi nhóm chứa các vector đầu vào có đặc trưng giống nhau.

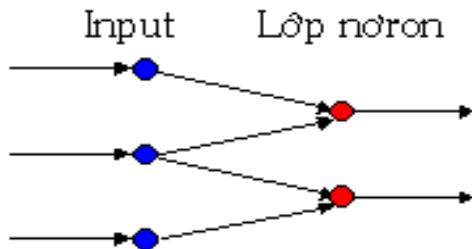
Với học không giám sát, số lớp phân loại chưa được biết trước, và tùy theo tiêu chuẩn đánh giá độ tương tự giữa các mẫu mà ta có thể có các lớp phân loại khác nhau.

*Học cũng có* là sự tổ hợp của mô hình trên đôi khi còn được gọi là học thưởng – phạt. 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).

#### **2.4.4. Các mô hình mạng neural**

Phần này trình bày các mô hình mạng dựa trên các kiểu lén của mạng bao gồm cấu trúc truyền thẳng Feedforward network gồm có mạng một lớp ẩn Single-layer perceptron – SLP, mạng đa lớp ẩn Multi-layer perceptron – MLP. Ngoài ra còn các cấu mạng nâng cao Recurrent Neural Networks – RNN, mạng tự tổ chức Self organizing Map – SOM, Time Delay Neural Network – TDNN, Learning Vector Quantisation – LVQ, Hopfield Network, Wavelet Neural Network, Auto-Associative Neural Networks trong phạm vi đề tài không đề cập tới.

**Mạng Perceptron đơn lớp** là một mô hình đơn giản nhất của mạng nơ-ron truyền thẳng chỉ với một lớp đầu ra. Trong lớp này có thể có một hoặc nhiều nơ-ron và các thông số đầu vào sẽ được kết nối trực tiếp đến các nơ-ron này.



Hình 2.7: Mô hình Perceptron

**Mạng Perceptron đa lớp MLP** là mạng có cấu trúc một lớp đầu vào, một lớp đầu ra, và một số lớp ẩn nằm giữa đầu vào vào đầu ra.

#### 2.4.5. Giải thuật lan truyền ngược

Mạng nơ-ron có khả năng học thông qua quá trình điều chỉnh giá trị các trọng số của nó. Mạng lan truyền ngược học thông qua ví dụ, tức là từ các ví dụ mẫu đầu vào, thuật toán điều chỉnh dần các trọng số của mạng sao cho phù hợp. Có thể xem các trọng số như là một ẩn số cần phải tìm ra sao cho ứng với các đầu vào thông qua mạng ta nhận được một kết quả xấp xỉ với đầu ra mong muốn. Các học đang đề cập tới chính là phương pháp học có giám sát.

Sau khi xác định cấu trúc mạng có bao nhiêu đầu vào, bao nhiêu đầu ra. Người ta bắt đầu tiến hành cho mạng học, phương pháp học có giám sát thông thường thực hiện qua các bước:

- Bước 1: Khởi tạo trọng số liên kết nhỏ ngẫu nhiên, nằm trong khoảng (-1, 1)
- Bước 2: Dựa các vector đầu vào của tập mẫu huấn luyện vào mạng tính vector đầu ra (quá trình lan truyền tiến)
- Bước 3: So sánh vector đầu ra mong muốn với kết quả thực tế nhận được
- Bước 4: Hiệu chỉnh các trọng số liên kết theo một cách nào đó sao cho trong lần tiếp theo vector đầu ra sẽ giống với kết quả mong muốn hơn
- Bước 5: Lặp đi lặp lại các bước từ 2 đến 4 một số lần nhất định hoặc dừng lại nếu đạt được một trạng thái mong muốn.

Trạng thái mong muốn tốt nhất trong việc học đó là các đầu ra thực tế đúng bằng các đầu ra mong muốn. Tuy nhiên đây là việc hầu như không thể, do đó trong thực tế người ta cần thiết lập các tiêu chuẩn lỗi để đánh giá hiệu quả mạng và để cho quá trình lặp có thể dừng lại được.

Thuật toán tổng quát hoá của phương pháp học có giám sát trong các mạng neural có nhiều cách cài đặt khác nhau. Trong đó tiêu biểu là thuật toán lan truyền ngược.

Trong tiếng Anh, truyền ngược là Backpropagation, là một từ viết tắt cho "backward propagation of errors" tức là "truyền ngược của sai số", là một phương pháp phổ biến để huấn luyện các mạng thần kinh nhân tạo được sử dụng kết hợp với một phương pháp tối ưu hóa như gradient descent. Phương pháp này tính toán gradient của hàm tổn thất với tất cả các trọng số có liên quan trong mạng nơ ron đó. Gradient này được đưa vào phương pháp tối ưu hóa, sử dụng nó để cập nhật các trọng số, để cực tiểu hóa hàm tổn thất.

Có hai chế độ học để lựa chọn: một loạt và ngẫu nhiên. Trong chế độ học ngẫu nhiên, mỗi một lan truyền được sau ngay bởi một cập nhật trọng số. Trong học một loạt, nhiều lan truyền xảy ra trước khi cập nhật các trọng số, các sai số tích lũy trên các mẫu nằm trong một gói. Học ngẫu nhiên đưa vào các "nhiều" vào quá trình gradient descent, sử dụng gradient địa phương tính từ một điểm dữ liệu. Điều này sẽ giúp giảm khả năng bị mắc kẹt trong một cực tiểu cục bộ cho mạng lưới. Tuy nhiên, học một loạt thường đạt được độ dốc nhanh hơn, ổn định hơn với một cực tiểu địa phương, vì mỗi cập nhật được thực hiện theo hướng sai số trung bình của các mẫu một loạt. Trong các ứng dụng hiện đại, một lựa chọn thỏa hiệp phổ biến là sử dụng các "mini-batch", nghĩa là học một loạt, nhưng với một gói có kích thước nhỏ và mẫu được lựa chọn ngẫu nhiên

Mô hình mạng thường sử dụng phương pháp lan truyền ngược có cấu trúc 3 lớp:

- Lớp vào (Input Layer) với số node vào là số thuộc tính của đối tượng cần phân lớp.
- Lớp ra (Output Layer) – Số node ra là số đặc điểm cần hướng tới của đối tượng để phân biệt các đầu vào với nhau.

- Lớp ẩn (Hidden Layer) – Số node ẩn thường là không xác định trước, nó thường là do kinh nghiệm của người thiết kế mạng và thay đổi dựa trên các phương pháp thử sai thực nghiệm. Nếu số node ẩn quá nhiều mạng sẽ cồng kềnh, quá trình học sẽ chậm, còn nếu số node ẩn quá ít làm mạng học không chính xác.

Để mô tả thuật toán lan truyền ngược ta sử dụng một số ký hiệu sau đây:

- $I_{li}$  – Input của node đầu vào thứ i
- $O_{li}$  – Output của node đầu ra thứ i
- $I_{Hi}$  – Input của node lớp ẩn thứ i
- $O_{Hi}$  – Output của node lớp ẩn thứ i
- $I_{Oi}$  – Input của node lớp đầu ra thứ i
- $O_{Oi}$  – Output của node lớp ẩn thứ i

Đầu vào của thuật toán:

- Mạng neural 3 lớp, với n đầu vào, m nút ẩn và đầu ra
- $\eta$ : Hằng số học
- $\varepsilon$ : Độ lỗi chấp nhận
- Một tập dữ liệu huấn luyện (training set):  $D = \{x_i - \text{là vector đầu vào}, d_k - \text{là vector đầu ra}\}$
- T: số lần lặp tối đa

Đầu ra: các vector trọng số sau khi huấn luyện

Thuật toán lan truyền ngược<sup>[12]</sup>:

- Bước 1: Khởi tạo trọng số  $V_{ij}$ ,  $W_{jk}$  trong khoảng (-1;1)
- Bước 2: Lan truyền tiến tính đầu ra thực tế:
  - + Tại lớp đầu vào node thứ i:  $I_{li} = x_i$ ,  $O_{li} = I_{li}$
  - + Tại lớp ẩn node thứ p:  $I_{Hp} = \sum O_{li} V_{ip}$ ,  $O_{Hp} = \frac{1}{1+e^{-I_{Hp}}}$
  - + Tại lớp ra nó thứ q:  $I_{Oq} = \sum O_{Hi} V_{ip}$ ,  $O_{Oq} = \frac{1}{1+e^{-I_{Oq}}}$

- Bước 3: Đánh giá độ lỗi – theo độ độ lỗi trung bình phương (MSE - mean-square error):

$$+ E = \frac{1}{L * \text{sqrt}(\sum(d_k - y_k)^2)}$$

+ Tại bước này nếu  $E \leq \epsilon$  thì dừng quá trình học

- Bước 4: Lan truyền ngược điều chỉnh trọng số:

+ Với mỗi nút q thuộc lớp ra, tính đạo hàm sai số thành phần  $\partial_q$  theo công thức  $\partial_q = (d_q - y_q)y_q(1 - y_q)$

+ Cập nhật các trọng số từ tầng ẩn tới tầng ra  $W_{jk}$

$$\quad \quad \quad \blacksquare W_{pq}^{(new)} = W_{pq}^{(old)} + \Delta W_{pq} \text{ với } \Delta W_{pq} = \eta \partial_q O_{Hq}$$

+ Với mỗi node p thuộc tầng ẩn, tính đạo hàm sai số  $\partial_p$  theo công thức

$$\quad \quad \quad \blacksquare \partial_p = O_{Hp}(1 - O_{Hp}) \sum_{k=1}^L W_{pk}^{(old)} \partial_k$$

+ Cập nhật trọng số từ tầng vào tới tầng ẩn  $V_{ij}$

$$\quad \quad \quad \blacksquare V_{ip}^{(new)} = V_{ip}^{(old)} + \Delta V_{ip} \text{ với } \Delta V_{ip} = \eta \partial_p O_{Ii}$$

- Bước 5: Lặp lại các bước 2, 3, 4 cho tới khi dừng ở bước 3 hoặc đủ T lần.

Nếu quá trình huấn luyện thành công, các trọng số trong mạng được điều chỉnh phù hợp nhất với bộ dữ liệu huấn luyện. Tuy nhiên việc huấn luyện mạng theo phương pháp lan truyền ngược cũng có những khó khăn, phần dưới đây sẽ đề cập đến các vấn đề trong quá trình xây dựng mạng và một số các giải quyết.

#### **2.4.6. Các vấn đề trong xây dựng mạng và phương pháp lan truyền ngược**

Lợi thế lớn nhất của ANN là khả năng được sử dụng như một cơ chế xấp xỉ hàm tuỳ ý mà chúng học được từ các dữ liệu huấn luyện. Tuy nhiên việc sử dụng ANN thực tế không đơn giản như vậy. Độ chính xác của phụ thuộc vào rất nhiều yếu tố để tìm ra mạng phù hợp không phải là một việc hết sức khó khăn.

Vấn đề đầu tiên khi giải quyết bài toán với ANN đó là chọn mô hình phù hợp, việc chọn mô hình sẽ phụ thuộc vào cách tổ chức dữ liệu của ứng dụng cụ thể, mô hình đơn giản làm cho hiệu quả bài toán thấp nhưng nếu mô hình quá phức tạp thì sẽ dẫn đến thách thức lớn trong quá trình huấn luyện. Trong phạm vi đề tài sử dụng mạng neural

đa lớp MLP học theo phương pháp có giám sát dùng giải thuật lan truyền ngược nên phần dưới đây đề tài chỉ đề cập đến các vấn đề khó trong xây dựng mạng theo mô hình này tham khảo từ [<sup>11</sup>]

*Vấn đề mạng không đủ năng lực(underfilling):* là hiện tượng mạng không thể tổng quát hoá được mẫu, vấn đề này thường xảy ra với một mạng với số node lớp ẩn quá ít, giải quyết bằng cách tăng số nơ-ron lớp ẩn lên. Thông thường khi tăng số nơ-ron lớp ẩn lên thì độ chính xác cũng tăng lên, tuy nhiên mô hình mạng sẽ trở nên phức tạp. Sự phức tạp ảnh hưởng tới hai khía cạnh, thứ nhất là tốc độ tính toán sẽ chậm đi rất nhiều, thứ hai khi mô hình phức tạp nó có thể xây dựng được hàm tổng quát rất tốt bộ dữ liệu huấn luyện, tuy nhiên nó lại không tốt đối với bộ dữ liệu kiểm tra. Hiện tượng này được gọi là *overfitting* sẽ được trình bày dưới đây

*Vấn đề quá khớp (overfitting):* xảy ra khi mạng được huấn luyện quá khớp với dữ liệu huấn luyện kể cả nhiễu, nên nó sẽ trả lời chính xác những gì đã học, còn những gì chưa được học thì hoàn toàn không đúng. Vấn đề quá khớp thường xảy ra khi năng lực mạng quá lớn. Có 3 cách để hạn chế vấn đề này:

- Giảm bớt số nút ẩn
- Ngăn không cho mạng sử dụng các trọng số lớn
- Giảm số lần huấn luyện

Khi mạng được luyện, nó chuyển từ các hàm ánh xạ tương đối đơn giản đến các hàm ánh xạ tương đối phức tạp. Nó sẽ đạt được một cấu hình tổng quát hóa tốt nhất tại một điểm nào đó. Sau điểm đó mạng sẽ học để mô hình hóa nhiễu, những gì mạng học được sẽ trở thành quá khớp. Nếu ta phát hiện ra thời điểm mạng đạt đến trạng thái tốt nhất này, ta có thể ngừng tiến trình luyện trước khi hiện tượng quá khớp xảy ra. Ta biết rằng, chỉ có thể để đánh giá mức độ tổng quát hóa của mạng bằng cách kiểm tra mạng trên các mẫu nó không được học. Ta thực hiện như sau: chia mẫu thành tập mẫu huấn luyện và tập mẫu kiểm tra. Luyện mạng với tập mẫu huấn luyện nhưng định kỳ dừng lại và đánh giá sai số trên tập mẫu kiểm tra. Khi sai số trên tập mẫu kiểm tra tăng lên thì quá khớp đã bắt đầu và ta dừng tiến trình luyện.

*Vấn đề xác định kích thước mẫu:* Không có nguyên tắc nào hướng dẫn kích thước mẫu phải là bao nhiêu đối với một bài toán cụ thể. Hai yếu tố quan trọng dẫn đến kích thước mẫu:

- Dạng hàm đích: khi hàm đích càng phức tạp thì kích thước mẫu càng tăng
- Nhiều: khi dữ liệu bị nhiễu kích thước mẫu cùng cần được tăng

*Việc xác định các tham số huấn luyện:* Các tham số huấn luyện ngoài số tác động của số lần lặp tối đa có thể dẫn đến hiện tượng quá khớp hoặc không tổng quát được đàm mê ở trên thì các giá trị hằng số học cũng ảnh hưởng đến kết quả của quá trình huấn luyện. Việc xác định các giá trị thường do kinh nghiệm và các thực nghiệm đưa ra.

Các tham số huấn luyện cần quan tâm đó là ngưỡng lỗi và tốc độ học. Từ thuật toán có thể thấy rằng nếu ngưỡng lỗi chấp nhận lớn thì quá trình huấn luyện sẽ được dừng sớm, nhưng điều này không đảm bảo mạng sẽ đủ khả năng tổng quát mà ta mong muốn. Khó khăn khi chọn tham số ngưỡng lỗi đó là nếu chọn lớn thì có thể mạng sẽ rơi vào trường hợp *overfitting*.

Tham số thứ hai đó là tốc độ học, tốc độ học càng vào thì sự điều chỉnh càng lớn. Điều này sẽ làm cho quá trình điều chỉnh lỗi nhanh hơn, tuy nhiên tới một mức nào đó nó lại làm cho sự hội tụ của mạng trở nên khó khăn hơn, đôi khi là tốc độ học lớn sẽ làm cho quá trình học không hội tụ được.

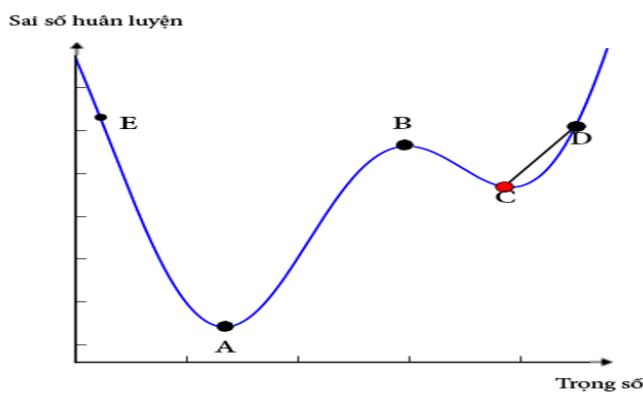
*Vấn đề khởi tạo trọng số:* trọng số thường được khởi tạo một cách ngẫu nhiên bằng phương pháp thử sai, mang tính chất kinh nghiệm tuỳ thuộc vào bài toán, việc định nghĩa một bộ trọng số tốt cũng không dễ. Một số kinh nghiệm để khởi tạo trọng số:

- Khởi tạo trọng số cho mạng nơron thu được là cân bằng (với đầu vào ngẫu nhiên sai số lan truyền ngược cho các ma trận trọng số là xấp xỉ bằng nhau). Nếu mạng nơron không cân bằng thì quá trình thay đổi trọng số ở một số ma trận là rất nhanh trong khi ở một số ma trận khác lại rất chậm, thậm chí không đáng kể. Do đó để các ma trận này đạt tới giá trị tối ưu sẽ mất rất nhiều thời gian.

- Tạo trọng sao cho giá trị kết xuất của các nút có giá trị trung gian, để tránh các giá trị thái quá trong quá trình lan truyền

*Vấn đề lãng quên(catastrophic forgetting):* là hiện tượng các mạng neural quên các mẫu cũ trong khi học mẫu mới, nguyên nhân là do sự thay đổi trọng số cho mẫu mới do các mẫu cũ một thời gian không được đưa vào huấn luyện. Để tránh điều này ta phải thực hiện huấn luyện luân phiên hoặc đồng loạt các mẫu mới và cũ.

*Vấn đề cực tiểu địa địa phương:* vì mạng học theo phương pháp Gradient descent nên việc tìm ra cực tiểu toàn cục (cực trị của hàm số) có thể không xảy ra mà chỉ dừng lại ở một cực tiểu cục bộ.



Hình 2.8: Minh họa quá trình huấn luyện với gradient descent

Trong hình minh họa phía trên điểm A là cực tiểu cục bộ mà quá trình huấn luyện cần tìm, điểm C là cực tiểu mà ta có thể gặp trong quá trình huấn luyện. Thuật toán học Gradient descent thường được ví như tác dụng của trọng lực lên một hoàn bi đặt trên một mặt có dạng như hình thung lũng như hình trên. Thuật toán huấn luyện ban đầu sẽ khởi tạo một bộ trọng số bất kỳ, do đó điểm bắt đầu sẽ bất kỳ có thể là D hoặc E. Qua quá trình huấn luyện độ lỗi sẽ giảm từng bước  $\eta$ (hàng số học), ví dụ như một bước học  $\eta$  bằng khoảng cách từ D đến C. Theo cách này nếu ta khởi tạo trọng số ban đầu là E thì quá trình huấn luyện sẽ dừng lại ở A, nhưng nếu quá trình khởi tạo là D thì rất có thể quá trình huấn luyện sẽ dừng lại ở cực tiểu cục bộ C.

Theo góc nhìn vật lý nếu bài quá trình đi từ D xuống có tốc độ đủ lớn thì, viên bi sẽ theo đà lăn qua B để đi xuống A, điều này là tuỳ thuộc vào độ dốc của đỉnh B. Dựa

trên hiện tượng này, một thuật toán ra đời nhằm khắc phục tình trạng thuật toán dừng ở điểm cực tiểu cục bộ có tên là Momentum.

Trong thuật toán Gradient descent, giả sử ta khởi tạo một điểm khởi tạo là  $\theta$  với  $\theta = \theta_0$ . Quá trình huấn luyện sẽ cập nhật sai số cho tới một mức chấp nhận được theo công thức:  $\theta = \theta_0 - \eta \nabla_0 E$  (với  $E$  là hàm mất mát trong thuật toán phần trước và  $\nabla_0 E$  là đạo hàm của  $E$ ).

Để biểu diễn thuật toán Gradient descent với momentum, ta giả sử trong gradient ta cần tính lượng thay đổi ở thời điểm  $t$  để cập nhật vị trí mới cho hòn bi. Nếu chúng ta coi đại lượng này vận tốc  $v_t$  trong vật lý vị trí mới của hòn bi sẽ là  $\theta_{t+1} = \theta_t - v_t$ . Công việc lúc này là tính đại lượng  $v$  sao cho nó vừa mang thông tin độ dốc – đạo hàm vừa mang thông tin đà – vận tốc trước đó( $v_{t-1}$ ). Một cách đơn giản ta có thể cộng hai đại lượng này với nhau:

$$v_t = \omega v_{t-1} - \eta \nabla_0 E$$

Trong đó  $\omega$  là một hằng số chung ta cung cấp trong khoảng  $(0.2; 0.5)$ . Lúc này vị trí mới của hòn bi được xác định bởi công thức:

$$\theta_t = \theta_{t-1} - v_t$$

Thuật toán này tỏ ra rất hiệu quả trong các bài toán thực tế. Phần này tham khảo từ [<sup>13</sup>][<sup>14</sup>][<sup>15</sup>].

#### **2.4.7. Thư viện hỗ trợ xây dựng mạng Neural trong môi trường NodeJS**

NodeJS là một môi trường rất tốt để xây dựng các ứng dụng với tốc độ nhanh phần đầu đã nêu ra. Hiện nay các thư viện (module) được cộng đồng phát triển ngày càng nhiều, trong đó có cả những thư viện về máy học. Trong lĩnh vực mạng nơ-ron nhân tạo tại thời điểm thực hiện có khoảng trên 30 module hỗ trợ việc xây dựng mạng và phát triển ứng dụng bao gồm các thư viện tiêu biểu:

- Brain.js[<sup>16</sup>]
- Neuralnetwork[<sup>17</sup>]
- Deep Learning(dnn)[<sup>18</sup>]
- Synaptic[<sup>19</sup>]

- Node Neural Network<sup>[20]</sup>
- Mind<sup>[21]</sup>
- ConvNetJS<sup>[22]</sup>
- Digital Neural Networks Architecture<sup>[23]</sup>

Trong các thư viện vừa đưa ra, trong quá trình nghiên cứu và thử nghiệm nhận thấy rằng thư viện hỗ trợ tương đối tốt việc xây dựng mạng nơ-ron theo mô hình MLP lan truyền ngược, hơn thế nữa module này còn hỗ trợ rất tốt trong việc lưu trữ mô hình nhận dạng và khởi tạo mạng từ mô hình nhân dạng lưu trữ.

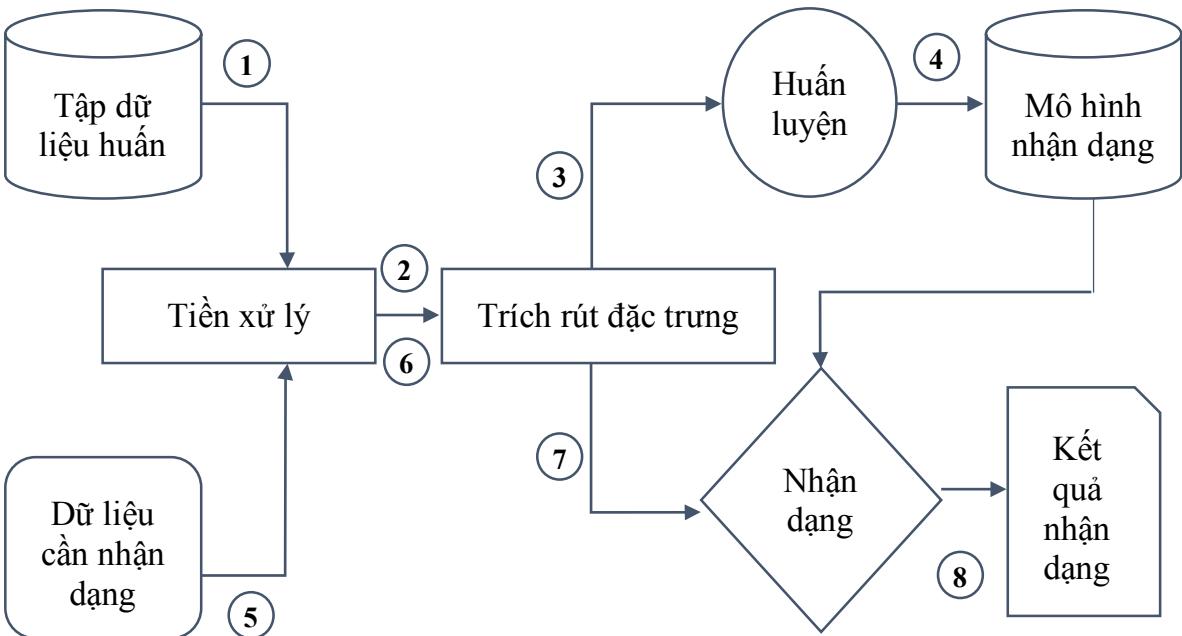
## **2.5. Bài toán nhận dạng chữ viết tay**

Chữ viết tay mỗi người có một cách viết riêng, thậm chí là cùng một người nhưng tại mỗi thời điểm chữ viết cũng khác nhau. Hiện nay các hệ thống ứng dụng sử dụng việc nhận dạng chữ viết tay vẫn chưa phổ biến vì nhiều lý do liên quan đến mức độ chính xác của việc nhận dạng. Vấn đề nhận dạng chữ viết tay đang đặt ra những thách thức lớn trong vấn đề đa dạng của chữ viết tay, vì vậy việc xây dựng một hệ thống nhận dạng đáng tin cậy là điều không dễ dàng.

### **2.5.1. Các giai đoạn xây dựng hệ thống nhận dạng chữ viết tay**

Tuỳ vào mục đích cuối cùng của bộ nhận dạng sẽ có những sơ đồ tổng quát khác nhau. Dưới đây là một sơ đồ hệ thống nhận dạng được xem là phổ biến, trong đó các thành phần chuyên biệt riêng cho từng hệ thống cụ thể đã được lược bỏ, sơ đồ tham khảo từ [24]

Một hệ thống nhận dạng chữ viết tay nhìn chung sẽ có cấu trúc như sơ đồ 2.3 dưới đây:



Sơ đồ 2.1: Cấu trúc chung của hệ thống nhận dạng chữ viết tay

Tập dữ liệu huấn luyện là tập dữ liệu chứa các mẫu dữ liệu được sử dụng để huấn luyện cho hệ thống nhận dạng, tức là dùng xây dựng cơ sở dữ liệu nhận dạng.

Dữ liệu cần nhận dạng là những dữ liệu cần nhận dạng trong thực tế.

Giai đoạn tiền xử lý là giai đoạn loại bỏ những thông tin dư thừa của dữ liệu đầu vào hoặc điều chỉnh dữ liệu đầu vào phù hợp với yêu cầu của giai đoạn tiếp theo, đó có thể là nhiễu của ảnh, chuyển về ảnh xám, làm trơn biên, điều chỉnh độ nghiêng, chuẩn hóa kích thước, nhị phân ảnh ... Hiện nay thư viện openCV hỗ trợ rất tốt các thao tác tiền xử lý này.

Giai đoạn trích trọn đặc trưng, tùy theo thiết kế của hệ thống nhận dạng sẽ có những cách thực hiện khác nhau. Đây là giai đoạn có vai trò rất quan trọng nó ảnh hưởng đến hiệu quả huấn luyện cũng như quá trình nhận dạng ra các đối tượng. Các đặc trưng rút ra từ ảnh phải đạt được mục đích quan trọng là dựa trên đặc trưng đó giải thuật có thể phân biệt được với nhau. Các kỹ thuật trích rút đặc trưng sẽ được trình bày kỹ trong phần tiếp theo.

Quá trình huấn luyện là quá trình sử dụng những đặc trưng đầu vào để đưa ra những thông tin được có thể hiểu là tri thức. Quá trình này dựa vào các phương pháp học máy, có thể là mô hình máy hỗ trợ vector – SVM, cây quyết định, ... trong báo cáo này sử dụng mô hình mạng neural nhân tạo.

Mô hình nhận dạng là những thông tin cần thiết cho quá trình nhận dạng, dữ liệu này tùy phương pháp học ở quá trình huấn luyện. Ví dụ trong mô hình SVM đó là một tập các vector đặc trưng, trong mô hình mạng neural nhân tạo đó là một bộ trọng số.

Nhận dạng là quá trình sử dụng dữ liệu nhận dạng để đưa ra một kết quả nhận dạng tương ứng với mỗi đầu vào.

Kết quả nhận dạng là kết quả thu nhận được từ quá trình nhận dạng.

Một quá trình xây dựng mạng thường đi qua hai pha đó là pha huấn luyện và pha nhận dạng. Trong sơ đồ trên pha huấn luyện bao gồm các bước 1; 2; 3 và 4, kết quả của pha này là một mô hình nhận dạng được sử dụng trong pha nhận dạng. Pha nhận dạng bao gồm các bước 5; 6; 7 và 8.

### **2.5.2. Các kỹ thuật trích rút đặc trưng**

Trích chọn đặc trưng là giai đoạn để tìm ra các đặc điểm trên mỗi dữ liệu đầu vào, điểm quan trọng của giai đoạn này đó là các thuật toán trích rút đặc trưng cần phải tránh tổn thất thông tin, kích thước dữ liệu nhỏ gọn để nâng cao hiệu quả của ứng dụng. Một số loại đặc trưng cơ bản của mẫu hiện tại bao gồm:

- **Đặc trưng thống kê**, là những đặc trưng được trích xuất dựa trên các điểm ảnh.
- **Đặc trưng hình học và hình thái**, là những đặc điểm từ các yếu tố nguyên thủy (đoạn thẳng, cung) tạo ra các ký tự. Các ký tự có thể được phân biệt bằng độ đo của các đại lượng hình học như tỉ số giữa chiều rộng và chiều cao của khung chứa ký tự, quan hệ khoảng cách giữa hai điểm, độ dài một nét, độ dài tương quan giữa hai nét, tỉ lệ giữa các chữ hoa và chữ thường trong một từ, độ dài từ... Vì thế các ký tự được tổ chức thành các tập hợp của các yếu tố nguyên thủy, sau đó đưa các yếu tố nguyên thủy vào các đồ thị liên quan

- **Đặc trưng hướng**, các ký tự được mô tả như các vector mà các phần tử của nó là các giá trị thống kê về hướng. Việc chọn đặc trưng để nâng cao độ chính xác của bài toán nhận dạng là hết sức khó khăn, đòi hỏi rất nhiều thời gian và quyết định rất nhiều đến độ chính xác. Hơn nữa, do biến dạng khá lớn trong chữ viết tay nên để hạn chế người ta thường chia ô trên ảnh và đặc trưng được rút trong các ô đó.

Nổi lên trong nhóm đặc trưng thống kê, kỹ thuật Zoning có cách cài đặt khá đơn giản nhưng cho kết quả khá tốt trong bài toán nhận dạng chữ viết tay. Kỹ thuật này sẽ chia ảnh ký tự ra thành  $k^* k$  vùng, sau đó tính tổng số điểm đen của mỗi vùng để tạo nên vector đặc trưng.



*Hình 2.9: Minh họa kỹ thuật Zoning*

Ví dụ như ban đầu ta có ảnh ký tự có kích thước  $28 \times 28$  điểm ảnh thì với  $k = 2$ , ta sẽ có tất cả là:  $\frac{28}{2} \times \frac{28}{2} = 14 \times 14 = 196$  hình vuông con có cạnh là 2 điểm ảnh. Các hình vuông được lấy lần lượt từ trái sang phải và từ trên xuống dưới. Ứng với tổng số điểm đen trong mỗi ô vuông thứ  $i$  sẽ là giá trị của vector đặc trưng tại chiều  $i$  đó.

## 2.6. Xử lý ảnh

Trong bài toán nhận dạng chữ viết tay, xử lý ảnh nằm trong giai đoạn tiền xử lý. Đây cũng là một trong những vấn đề được nghiên cứu nhằm cải thiện chất lượng nhận dạng. Trong phạm vi đồ án và để hoàn thành đề tài đặt ra, đề tài sử dụng gói hai module opencv<sup>[25]</sup> và jimp<sup>[26]</sup> để hỗ trợ các thao tác tiền xử lý ảnh.

Mục tiêu của giai đoạn tiền xử lý hướng tới đó là:

- Giảm đi các yếu tố ảnh hưởng tới các đường nét của ký tự như các điểm ảnh nhiễu (nhiễu muối tiêu), loại bỏ ảnh hưởng của mức độ ảnh sáng tối, các nếp nhăn giấy, các vệt mờ/dâm không liên quan đến ký tự xuất hiện trong quá trình thu nhận ảnh đầu vào.
- Giảm đi sự phức tạp của ảnh, ảnh hưởng của màu sắc thông qua việc nhị phân ảnh
- Đưa ảnh về kích thước chuẩn phù hợp với bài toán.

Hiện nay bài toán nhận dạng chữ viết tay được chia thành nhiều bài toán nhỏ khác nhau dẫn đến sử dụng một loạt các thuật toán khác nhau trong mỗi giai đoạn xử lý, do vậy trong đề tài chỉ tập trung đến những lý thuyết cả của các kỹ thuật xử nhận dạng chữ viết tay rời rạc, tức là mỗi ký tự viết tay được xử lý riêng biệt. Các kỹ thuật có liên quan bao gồm: chuyển về mức ảnh xám, lọc khử nhiễu, tìm biên và xác định khung đối tượng trong ảnh, nhị phân ảnh, tách ký tự, tách dòng,...

### **2.6.1. Chuyển xám ảnh**

Đơn vị tế bào cấu trúc của ảnh là điểm ảnh (pixel). Đối với ảnh màu từng pixel sẽ mang thông tin của 3 màu cơ bản tạo ra màu của pixel đó là Blue, Green và Red, ngoài ra còn một giá trị đặc trưng cho độ trong suốt của ảnh. Ba màu cơ bản bô trí sát nhau trong từng điểm ảnh và có cường độ khác nhau. Thông thường mỗi màu cơ bản được biểu diễn bằng 8 bit tương ứng với 256 mức độ khác nhau. Như vậy mỗi pixel sẽ có  $2^{8 \times 3}$  màu (khoảng 16.78 triệu màu). Đối với ảnh xám, thông thường mỗi pixel mang thông tin của 256 mức xám, như vậy ảnh xám hoàn toàn có thể tái cấu trúc ảnh một ảnh màu tương ứng.

Trong hầu hết các xử lý chúng ta chỉ quan tâm đến cấu trúc của ảnh mà bỏ qua sự ảnh hưởng của màu sắc của bức ảnh. Do đó bước chuyển ảnh xám là một công đoạn phổ biến và cần thiết để giảm độ phức tạp của ảnh đầu vào và tăng tốc độ tính toán của các giải thuật.

Công thức chuyển ảnh xám của 1 pixel như sau:

$$G = \alpha R + \beta G + \gamma B$$

Trong đó  $\alpha, \beta, \gamma$  là các hằng số; R, G, B là các mức độ màu Blue, Green và Red tương ứng tại điểm ảnh đó.

Trong môi trường NodeJS sử dụng module opencv để chuyển ảnh xám như sau:

```
const cv = require('opencv');
cv.readImage(src, function (err, img) {
    img.convertGrayscale();
    // Do something with img gray
})
```

hoặc đối với module Jimp:

```
const Jimp = require('jimp');
Jimp.read(imgDir + imgName, function (err, img) {
    img greyscale(function (err, imgGray) {
        // Do something with img gray
    })
})
```

## 2.6.2. Nhị phân ảnh

Mặc dù tính đến thời điểm hiện tại việc xử lý ảnh màu hoặc ảnh xám ký tự đang được quan tâm nghiên cứu do bảo toàn được nhiều tính chất của ảnh, nhất là đặc điểm về việc biến thiên mức năng lượng, song các thuật toán nhận dạng hiện nay hầu như đều xử lý trên ảnh nhị phân (ảnh chỉ có hai loại điểm ảnh là hoàn toàn đen hoặc hoàn toàn trắng). Hai lý do có thể kể đến cho vấn đề này là:

- Việc nghiên cứu các thuật toán áp dụng trên ảnh màu bao giờ cũng tốn nhiều chi phí (thời gian, độ phức tạp) hơn so với ảnh nhị phân. Để đơn giản nhất ta xét ví dụ về ảnh màu dùng hệ màu RGB thì phải tiến hành trên ít nhất ba mảng điểm ảnh tương ứng với ba gam màu Red, Green, Blue
- Một vài thuật toán chỉ có thể áp dụng trên ảnh nhị phân

Cách thức để nhị phân ảnh phổ biến nhất là dựa trên dữ liệu hiện thời của ảnh (đặc biệt là dữ liệu về Histogram) để đưa ra một ngưỡng nhị phân, sau đó xét trên toàn bộ các điểm ảnh, điểm ảnh nào có giá trị nhỏ hơn ngưỡng nhị phân thì quy về pixel 0, điểm ảnh nào có giá trị lớn hơn ngưỡng nhị phân thì quy về pixel 1. Một số kỹ thuật nhị phân đang được sử dụng hiện tại tham khảo từ [27] bao gồm:

- Kỹ thuật ngưỡng toàn cục giản đơn (Global fixed threshold): đây là kỹ thuật nhị phân đơn giản nhất:

$$Dest(x, y) = \begin{cases} 1 & \text{nếu } Source(x, y) \geq T \\ 0 & \text{nếu } Source(x, y) < T \end{cases}$$

Trong đó  $Source(x, y)$  là giá trị điểm ảnh tại  $x, y$  của ảnh đầu vào,  $Dest(x, y)$  là giá trị thu được tương ứng,  $T$  là ngưỡng thường không có giá trị cố định mà tùy thuộc vào ảnh đầu vào.

- Kỹ thuật ngưỡng Otsu (Otsu threshold), ý tưởng của kỹ thuật này là tìm ra một ngưỡng toàn cục cho ảnh, sau đó chuẩn hoá ảnh thành ảnh nhị phân.

### 2.6.3. Nhiễu ảnh

Nhiễu là hiện tượng giá trị của một phần tử ảnh nỗi trội hơn so với các phần tử xung quanh. Đối với các ứng dụng sử dụng ảnh đầu vào là camera hoặc thiết bị scan thông thường ảnh thường bị nhiễu hoặc bị mờ vì vậy cần phải xử lý loại bỏ. Một số loại nhiễu thường gặp bao gồm: nhiễu trắng, nhiễu Gauss, nhiễu đều, nhiễu xung, hoặc nhiễu muối tiêu. Trong đó nhiễu muối tiêu là ảnh thường gặp nhất đối với các ứng dụng sử dụng camera làm thiết bị thu thập ảnh.

Một số phương pháp lọc nhiễu bao gồm lọc trung bình, lọc trung vị (Median), lọc sắc nét, lọc hight-boost. Cơ sở toán học của các phép lọc là nhân ma trận ảnh với một ma trận trọng số. Hiện nay thư viện opencv hỗ trợ các hàm khá tốt để xử lý, một số hàm thông dụng bao gồm:

- Sử dụng bộ lọc:

```
img.medianBlur(thresh);
```

- Bộ lọc Gauss:

```
img.gaussianBlur([thresh_min, thresh_max]);
```

### 2.6.4. Xác định đối tượng trong ảnh

Để xác định được hình chữ nhật chứa đối tượng trong ảnh có thể dùng thuật toán Canny Edge Detector<sup>[28]</sup> được phát triển năm 1986 và trở thành thuật toán phổ biến thường được gọi là “máy đo tối ưu”. Thuật toán này đáp ứng 3 tiêu chí:

- Phát hiện cạnh có tỷ lệ lỗi thấp
- Các cạnh chỉ được phát hiện một lần
- Phát hiện lặp tốt

Trong module opencv hiện tại đã hỗ trợ thuật toán này, thao tác để xác định biên thực hiện như sau:

```
// input: img đã chuyển xám
img.canny(lowThresh, highThresh);
img.dilate(iterations);
var contours = img.findContours(); // class contours chứa các thông tin các
vùng biên trong ảnh

// Dựa trên thông tin biên, xác định các hình chữ nhật
for (let i = 0; i < contours.size(); i++) {
    var bound = contours.boundingRect(i);
    // output bound chứa thông tin hình chữ nhật bao quanh đối tượng,
    // bao gồm toạ độ, chiều cao, chiều rộng
}
```

### 2.6.5. Tách ghép ký tự

Có nhiều kỹ thuật tách ký tự như sử dụng lược đồ sáng, sử dụng biên của điểm ảnh. Trong phạm vi ứng dụng của đề tài, để phù hợp với môi trường và đặc trưng của ứng dụng, tôi sử dụng phương pháp tách ký tự bằng kỹ thuật dùng biên của các đối tượng trong ảnh.

Dựa trên quá trình xác định đối tượng đã trình bày ở phần 2.6.4 sẽ thu được một tập các hình chữ nhật bao quanh các đối tượng có trong ảnh. Nhiệm vụ của quá trình tách ghép ký tự là tách các hình chữ nhật thành các hình ảnh các ký tự riêng biệt theo thứ tự từ trái qua phải, từ trên xuống dưới của hình ảnh, và loại bỏ các đối tượng dư do sự chồng biên đối tượng.

Thuật toán loại bỏ các đối tượng dư:

- Input: danh sách tất cả các biên đối tượng, bao gồm toạ độ đỉnh (x, y), chiều cao: height, chiều rộng: width
- Output: danh sách các đối tượng không bị loại bỏ
  1. Khởi tạo mảng rỗng danh sách bị đối tượng biên bị loại bỏ

2. Duyệt các đối tượng có trong danh sách đầu vào

a. Kiểm tra đối tượng có phải là đối tượng có nằm trong một đối tượng khác không:

i. Duyệt qua tất cả các biên

ii. Nếu  $x > x'$  và  $y > y'$   
 và  $x + width < x' + width'$  và  $y + height < y' + height'$  thì  
 thêm đối tượng vào danh sách đối tượng bị loại bỏ. Trong  
 đó  $x, y, width, height$  là thông tin đối tượng đang xét,  $x', y',$   
 $width', height'$  là thông tin các đối tượng trong vòng lặp.

3. Xoá các đối tượng thuộc danh sách xoá

a. Duyệt danh sách đầu vào

i. Tìm đối tượng có nằm trong danh sách xoá. Nếu đối tượng  
 tồn tại trong danh sách xoá thì xoá đối tượng.

Sau khi tìm được danh sách các đối tượng trong ảnh, các đối tượng thông thường  
 không được sắp xếp theo thứ tự như trong ảnh, để ghép ký tự lại ta cần sắp xếp các đối  
 tượng theo thứ tự tăng dần toạ độ đỉnh.

Ưu điểm của phương pháp cắt ảnh bằng là dễ cài đặt, sử dụng hiệu quả cao đối  
 với các hình ảnh rời nét và đó cũng là nhược điểm lớn nhất của phương pháp dùng biên  
 đó là không cắt được hình ảnh với các ký tự liền nét.

Bảng 2.8: Minh họa quá trình tách xử lý tách ký tự

Ảnh đầu vào	Ảnh trước khi loại các đối tượng dư	Ảnh đã loại các ký đối tượng trùng lặp

## CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG

Quá trình xây dựng ứng dụng được thực hiện giai đoạn: xây dựng dữ liệu huấn luyện mạng, trích rút đặc trưng; xây dựng mạng nơ-ron và sau đó sử dụng mô hình nhận dạng tích hợp vào ứng dụng để xây dựng các tính năng.

### 3.1. Xây dựng bộ dữ liệu

Để chuẩn bị cho quá trình nguyên cứu, thử nghiệm và triển khai ứng dụng cần phải có một bộ dữ liệu sử dụng cho quá trình huấn luyện mạng, và kiểm tra mạng. Đề tài đã xây dựng tập dữ liệu gồm 26 ký tự in thường trong bảng chữ cái tiếng Anh. Mỗi ký tự có từ 120-150 hình khác nhau.

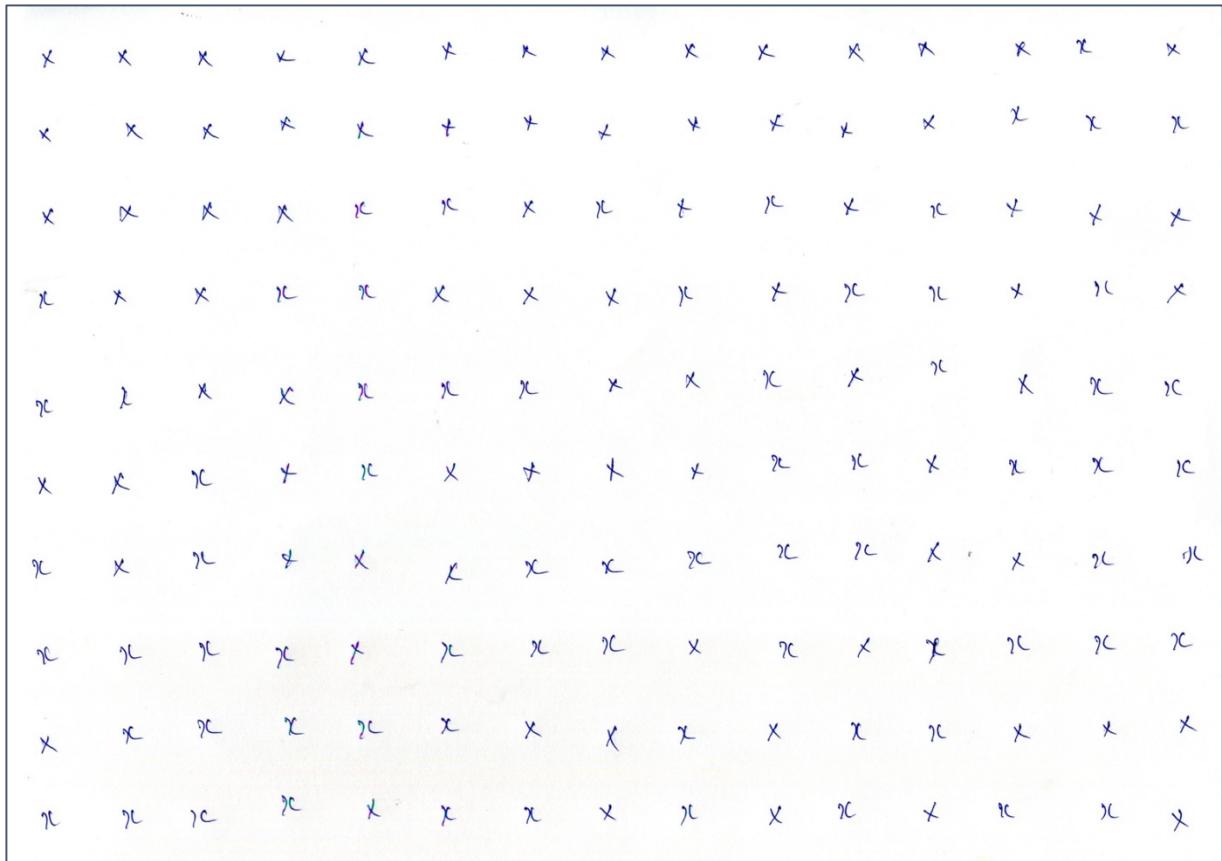
a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z

Hình 3.1: Bộ 26 ký tự mẫu

Quá trình xây dựng tập dữ liệu mẫu gồm các bước:

- Thu thập ảnh 26 ký tự
- Tiền xử lý ảnh (điều chỉnh kích thước, khử nhiễu, làm dày mảnh/đày biên)
- Rút trích đặc trưng theo kỹ thuật phân vùng Zoning và lưu lại dưới dạng JSON.

Quá trình thu thập ảnh được tiến hành bằng cách scan chữ viết tay trên giấy A4. Mỗi tờ A4 được chia thành 150 ô (10 dòng x 15 cột), mỗi ký tự được viết trên một tờ 150 lần vào các ô đã chia. Sau đó scan thành ảnh. Hình dưới đây minh họa ảnh được scan của ký tự “x”.



Hình 3.2: Ảnh scan mẫu của ký tự “x”

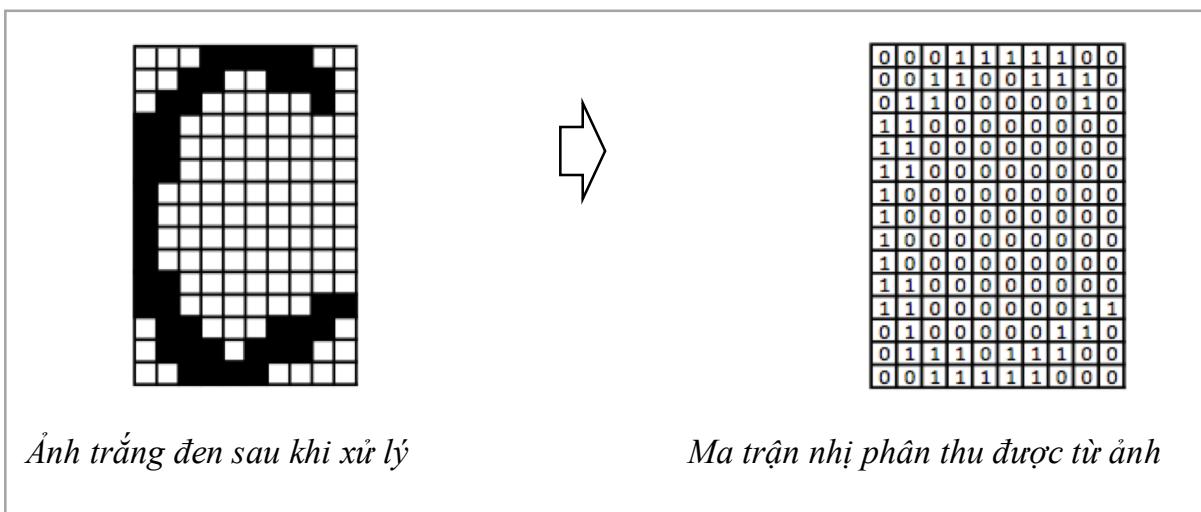
Sau khi thu thập đủ 26 ký tự, mỗi ảnh scan sẽ được đặt lại kích thước 2000x3000px, biến đổi ảnh xám, khử nhiễu, chuyển về ảnh trắng đen và cắt tách thành 150 ảnh, tương đương với 150 ô được chia lúc đầu. Trong quá trình cắt ảnh, mỗi ảnh cắt ra sẽ được tìm biên của chữ trong hình, cắt theo đường biên đó và lưu lại với kích thước 10x15px. Lý do chọn kích thước 10x15px là để tối ưu mạng nơ-ron sẽ được giải thích trong phần xây dựng mạng. Quá trình xử lý ảnh ứng dụng sử dụng thư viện openCV và module Jimp thực thi trên môi trường NodeJS.



Hình 3.3: Mô tả quá trình tiền xử lý tập dữ liệu

Trước khi tiến hành thực hiện trích rút đặc trưng để tập dữ liệu đạt được chất lượng cao không bị nhiều cần phải loại bỏ những ảnh bị cắt lỗi hoặc vẫn còn nhiều quá nhiều, thao tác loại bỏ được thực hiện thủ công.

Giai đoạn rút trích đặc trưng thực hiện theo phương pháp phân vùng. Để đơn giản trong quá trình thực hiện, mỗi ảnh từ quá trình tiền xử lý với kích thước 15x10px sẽ được chia lưới 15 hàng x 10 cột. Mỗi pixel trên ảnh tương ứng với một ô trong lưới. Mỗi ô trong lưới sẽ là tổng số pixel đen trong ô tương ứng. Vì tỉ lệ ô/số pixel ở đây là 1:1 nên một ô sẽ có giá trị là 1 hoặc 0. Giá trị là 1 nếu pixel trong ô là đen và ngược lại giá trị là 0 nếu pixel là trắng. Hình dưới minh họa phương pháp chọn đặc trưng theo phương pháp phân vùng.



Hình 3.4: Minh họa phương pháp trích chọn đặc trưng

Vì đặc trưng của ngôn ngữ JavaScript không có kiểu dữ liệu mảng hai chiều và để đơn giản trong quá trình huấn luyện sau này. Ma trận nhị phân thu được sẽ chuyển thành mảng một chiều kích thước là 150, với 10 phần tử tương ứng sẽ với một dòng trong ma trận. Để truy xuất nhanh trong các giai đoạn sau, kết quả trích xuất sẽ được lưu về định dạng JSON theo kiểu mảng. trong đó mỗi phần tử là một kết quả của quá trình lấy đặc trưng.

Kết quả của quá trình trích chọn đặc trưng thu được 3236 mẫu từ 26 ký tự. Trung bình mỗi ký tự có 120 mẫu. Để thuận tiện cho quá trình huấn luyện mà kiểm tra giải

thuật phân lớp dữ liệu, mỗi đặc trưng thu được sẽ được lưu trong tập JSON gồm n phần tử có cấu trúc như sau:

```
{
    "input": [<đặc trưng: điểm ảnh trắng - 0, điểm ảnh đen - 1>],
    "output": [<mảng xác vị trí số thứ tự ký tự tương ứng>]
}
```

Trong đó *input* là đầu vào của giải thuật chính là các đặc trưng mảng 150 phần tử đã nêu ở trên. *output* là dãy nhị phân gồm 26 phần tử, tương ứng với 26 ký tự, trong dãy sẽ có 1 phần tử mang giá trị là 1 và 25 phần tử mang giá trị 0. Ví dụ ký tự “a” đứng đầu thì phần tử đầu tiên sẽ là 1, các phần tử còn lại là 0.

### 3.2. Xây dựng mạng Neural nhận dạng ký tự

Mục tiêu của phần này thông qua các thực nghiệm để xây dựng một mạng Neural nhận dạng các ký tự viết tay cụ thể và đánh giá hiệu quả của mạng. Sau quá trình xây dựng mạng sẽ lưu trữ mô hình nhận dạng để sử dụng cho quá trình phát triển ứng dụng.

Cấu trúc tổng quát mạng mà để tài sử dụng là một mạng cấu trúc MLP gồm 1 lớp đầu vào, 1 lớp ẩn, và 1 lớp đầu ra. Sau quá trình xây dựng bộ dữ liệu, ta đã thu được các đặc trưng với cấu trúc input là dãy nhị phân 150 phần tử ứng với output là một dãy output 26 phần tử. Như vậy lớp đầu vào của mạng sẽ có 150 nơ-ron và lớp đầu ra gồm 26 nơ-ron. Quá trình huấn luyện mạng bằng phương pháp học có giám sát dùng giải thuật lan truyền ngược.

Trong quá trình tìm hiểu xây dựng mạng số nơ-ron lớp ẩn được chọn tương ứng với 150 phần tử trong ma trận ảnh bởi vì, nếu số lượng nơ-ron quá lớn sẽ dẫn tới việc nhận dạng các ký tự nhỏ hoặc trung bình bị sai, và quá trình lan truyền của mạng cũng chậm ảnh hưởng tới kết quả chương trình cuối cùng. Nếu số lượng nơ-ron quá ít sẽ dẫn tới quá trình nhận dạng các ký tự cỡ lớn cũng dễ bị sai, mạng có thể không học được nhiều mẫu. Qua thử nghiệm nhận thấy số lượng nơ-ron đầu vào 150 là phù hợp nhất

Vấn đề còn lại là xác định số nơ-ron lớp ẩn, các tham số trong quá trình học bao gồm tốc độ học, ngưỡng lỗi và kích thước mẫu. Các phần thực nghiệm dưới đây trình bày các thí nghiệm nhằm tìm ra các giá trị tốt nhất cho các tham số nêu trên.

Các tham số của mạng được lựa chọn:

- Hàm truyền: sigmoid:  $f(x) = 1/(1-e^x)$
- Khởi tạo trọng số: trọng số khởi tạo ngẫu nhiên trong khoảng (-0.5; +0.5)
- Momentum: 0.3

Để đánh giá mức hiệu quả của mạng, bộ dữ liệu sau khi rút trích đặc trưng được chia thành 2 bộ, một bộ dùng để huấn luyện (training set) và một bộ dùng để kiểm tra (testing set) theo tỷ lệ 70% dùng để huấn luyện và 30% dùng để kiểm tra. Các tham số còn lại thông qua quá trình thực nghiệm thử sai để xác định. Quá trình huấn luyện để tải sử dụng module brainjs<sup>[29]</sup> để hỗ trợ thực hiện các thử nghiệm.

### **3.2.1. Thực nghiệm chọn số lớp ẩn**

Qua một số tài liệu nghiên cứu và tham khảo các diễn đàn đưa ra kinh nghiệm để chọn số nơ-ron lớp ẩn sẽ nằm trong khoảng từ kích thước đầu vào tới kích thước đầu ra, gần các giá trị căn bậc 2 tích kích thước đầu vào và đầu ra hoặc 2/3 tổng đầu vào và đầu ra. Để xác định số lớp ẩn phù hợp để tài thực hiện các phép thử với các tham số khởi tạo như sau:

- Ngưỡng lỗi 0.001
- Số lần lặp tối đa 80
- Hằng số học: 0.3
- Số lượng neural lớp ẩn thử từ 10 – 170;
- Quá trình huấn luyện thực hiện trên hệ điều hành macOS Sierra 10.12, CPU core i5 2.7GHz, 8GB RAM

*Bảng 3.1: Bảng kết quả thu được từ quá trình thử số neural lớp ẩn*

<b>Số neural</b>	10	30	50	70	80	90	110	130	150	170
------------------	----	----	----	----	----	----	-----	-----	-----	-----

<b>Phần trăm nhận dạng</b>	76.4	88.7	88.9	90.3	89.6	89.0	89.14	88.9	88.8	88.1
<b>Thời gian huấn luyện (s)</b>	9,6	27,5	43,8	60,6	77,9	78,6	97,8	124,3	138,8	167,1

Qua thử nghiệm có thể thấy rằng số nơ-ron tăng thì độ chính xác và thời gian huấn luyện cũng tăng theo, tuy nhiên tới một thời điểm từ qua khoảng 90 nơ-ron thì kết quả kết quả nhận dạng lại không tăng nữa. Như vậy khoảng lựa chọn phù hợp nhất là từ 70-80 nơ-ron.

### 3.2.2. Thực nghiệm xác định tốc độ học

Mục đích tìm tốc độ học là để quá trình huấn luyện đạt được trạng thái dừng một cách nhanh nhất và quá trình huấn luyện sai số giảm dần tới ngưỡng lỗi mong muốn.

Bảng 3.2: Kết quả thử nghiệm tốc độ học

Tốc độ học Số lần lặp	0.1	0.3	0.5	0.8	1.0	1.5	2.0
1	0.0367	0.0313	0.0267	0.0233	0.0231	0.0222	0.0242
10	0.0058	0.0027	0.0017	0.0015	0.0013	0.0012	0.0017
20	0.0031	0.0013	0.0011	0.0010	0.0010	0.0009	0.0011
50	0.0014	0.0008	0.0006	0.0007	0.0007	0.0007	0.0010
100	0.0009	0.0007	0.0005	0.0006	0.0006	0.0006	0.0009

Quá trình thử nghiệm trên thực hiện với số lần huấn luyện tối đa là 100 lần. Quan sát kết quả trong bảng 3.2 ta nhận thấy rằng khi tăng tốc độ học từ 0.1 đến 0.8 thì tốc độ giảm lỗi cũng nhanh lên, tuy nhiên nếu tăng cao hơn 0.8 thì lỗi có giảm nhưng tốc độ

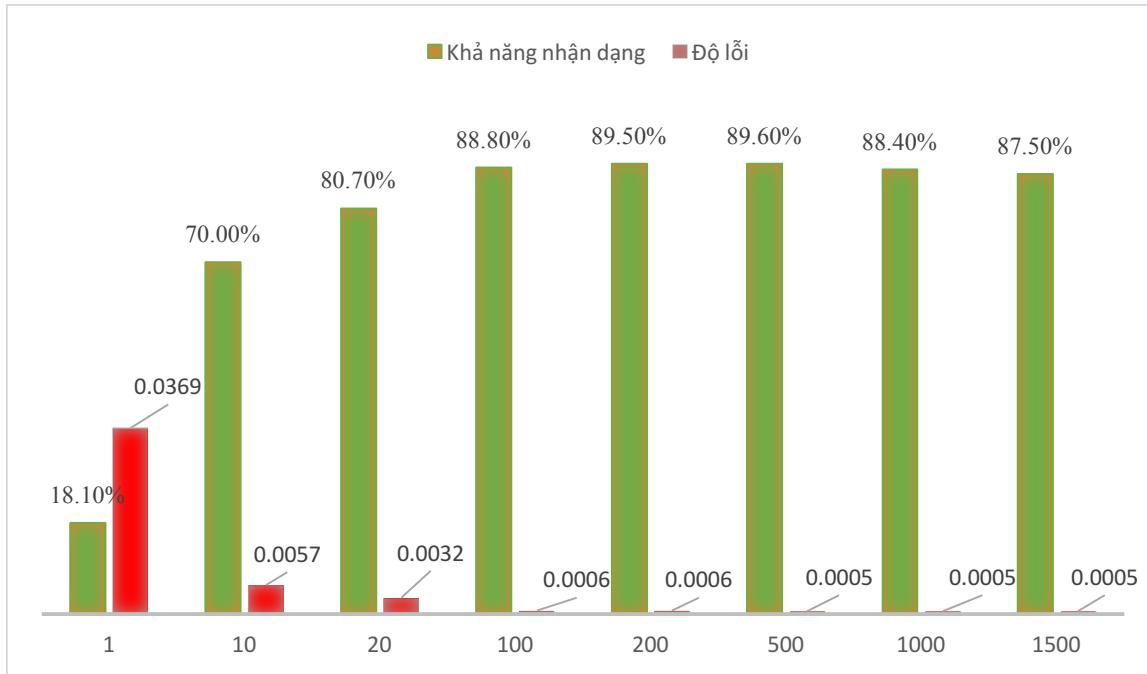
không nhanh, điều này được lý giải là do sự hỗ trợ của momentum giúp quá trình huấn luyện có khả năng giảm theo hướng đi xuống. Do vậy giá trị tốc độ học trong khoảng 0.3 đến 0.8 là phù hợp.

### **3.2.3. Xác định ngưỡng lỗi và số lần lặp tối đa**

Như đã trình bày ở phần lý thuyết, ảnh hưởng của ngưỡng lỗi chấp nhận và số lần lặp tối đa có thể dẫn tới mạng bị hiện tượng mạng quá khớp. Bảng 3.3 ghi nhận kết quả huấn luyện với số nơ-ron lớp ẩn là 80 và tốc độ học là 0.1.

*Bảng 3.3: Kết quả thử nghiệm xác định ngưỡng lỗi và số lần lặp*

Số lần lặp	Độ lỗi	Khả năng nhận dạng trên tập test
1	0.0369	18.1 %
10	0.0057	70.0 %
20	0.0032	80.7 %
100	0.0006	88.8 %
200	0.0006	89.5 %
500	0.0005	89.6 %
1000	0.0005	88.4 %
1500	0.0005	87.5 %



*Hình 3.5: Minh họa sự biến thiên khả năng nhận dạng theo độ lỗi*

Quan sát hình 3.5 nhận thấy rằng khi tăng dần số lần huấn luyện từ 1 đến dưới 1000 thì độ lỗi giảm và kiểm tra khả năng nhận dạng cũng tăng lên, tuy nhiên trên mức từ 1000 trở lên thì độ lỗi giảm rất chậm và khả năng nhận dạng cũng không tăng. Dựa vào đó ta chọn số lần huấn luyện từ 500 đến 1000 và độ lỗi là 0.0004.

### 3.2.4. Kết quả nhận dạng ký tự

Sau quá trình xây dựng mạng với 150 đầu vào, 26 đầu ra, 80 nơ-ron lớp ẩn, và các tham số:

- Nguồn lỗi: 0.0004,
- Tốc độ học: 0.5, mức quán tính (momentum: 0.3)
- Số lần học tối đa: 800
- Hàm nguồn sử dụng hàm sigmoid

Kết quả nhận dạng trung bình trên bộ dữ liệu thử nghiệm gồm 967 mẫu đã nhận dạng đúng 882 mẫu, tỷ lệ 91.2%. Kết quả nhận dạng chi tiết trên các ký tự như sau:

*Bảng 3.4: Kết quả nhận dạng trên các mẫu ký tự*

STT	Ký tự	Tỷ lệ	STT	Ký tự	Tỷ lệ	STT	Ký tự	Tỷ lệ
1	a	87.8	10	j	97.9	19	s	97.2
2	b	97.2	11	k	79.2	20	t	79.5
3	c	90.0	12	l	91.4	21	u	87.5
4	d	92.2	13	m	96.7	22	v	95.0
5	e	86.7	15	n	88.2	23	w	95.2
6	f	80.0	15	o	92.5	24	x	92.7
7	g	94.2	16	p	97.4	25	y	90.2
8	h	88.9	17	q	89.2	26	z	87.8
9	i	91.7	18	r	93.1			

Để sử dụng lại kết quả xây dựng mạng nhằm xây dựng ứng dụng ở giai đoạn sau, đề tài sử dụng tính năng lưu trữ mạng dưới dạng tập tin JSON của module brain.

### 3.3. Xây dựng ứng dụng

#### 3.3.1. Khảo sát chương trình đào tạo lớp 4

Theo chương trình đào tạo của bộ giáo dục và đào tạo, khung chương trình môn Tiếng Anh lớp 4 được thiết kế gồm 20 đơn vị bài tập. Mỗi đơn vị bài tập gồm 5 phần phù hợp để giảng dậy các kỹ năng nghe, nói, đọc viết

- **Phonics** cũng có khả năng phát âm và viết đúng chính tả của học sinh với yêu cầu các em tìm chữ cái phù hợp để điền vào chỗ trống và hoàn thành từ sau đó đọc to từ. Trong phần này gồm hai dạng bài tập hoàn thành từ và hoàn thành câu theo hình ảnh.

**Unit 9**

## In My Classroom

**A. PHONICS**

**1** Complete each word. Say it aloud.

**s**

1. pen \_\_\_\_\_ 2. schoolbag \_\_\_\_\_ 3. book \_\_\_\_\_ 4. desk \_\_\_\_\_

**2** Complete each sentence. Say it aloud.

1. They are \_\_\_\_\_.

Hình 3.6: Trích phần Vocabulary sách giáo Tiếng Anh khoa lớp 4

- **Vocabulary** giúp các em củng cố các từ đã học thông qua nhiều dạng bài tập khác nhau như xếp chữ thành các từ, điền vào chỗ trống và lựa chọn (trắc nghiệm)
- **Sentense Patterns** giúp củng cố các mẫu câu học sinh đã được học thông qua các hình thức lựa chọn, lắp ghép, ...
- **Reading** cũng củng cố khả năng đọc hiểu của học sinh ở cấp độ câu và đoạn hội thoại ngắn thông qua dạng bài tập điền vào chỗ trống hoặc chọn lựa
- **Writing** giúp học sinh viết chữ đúng chính tả các từ ngữ trong câu, sắp xếp từ ngữ thành câu

Trong sách giáo khoa còn có nhiều tranh minh họa sinh động nhằm hỗ trợ học sinh liên kết giữa từ ngữ với hình ảnh, nghĩa của từ và tình huống giao tiếp đồng thời tạo cảm hứng cho học sinh.

### **3.3.2. Xác định yêu cầu**

Qua trình khảo sát chương trình đào tạo lớp 4 đặt ra các yêu cầu xây dựng ứng dụng bao gồm xây dựng một tập từ vựng kèm hình ảnh, phát âm; thiết kế dạng bài tập học từ vựng (phần Vocabulary) theo sách giáo khoa sử dụng nhận dạng chữ viết tay trong ứng dụng để học sinh có thể tự kiểm tra bài làm của mình.

- Yêu cầu chức năng:
  - Quản trị viên:
    - + Hệ thống quản lý các bài học
    - + Hệ thống quản lý từ vựng theo các bài học
    - + Hệ thống quản lý người dùng
    - + Hệ thống quản lý bài viết
  - Học sinh:
    - + Học từ vựng, làm các bài tập và kiểm tra kết quả
    - + Tra cứu từ điển
    - + Theo dõi các bài viết
    - + Quản lý thông tin tài khoản
- Chức năng hệ thống:
  - Đăng nhập, đăng xuất
  - Phân quyền
  - Cấu hình hệ thống
- Các yêu cầu phi chức năng:
  - Đối với người sử dụng: hệ thống dễ sử dụng, dễ thao tác
  - Giao diện thân thiện, bắt mắt, ngôn ngữ sử dụng là tiếng Việt phù hợp với ngữ cảnh
  - Đảm bảo an toàn bảo mật thông tin người sử dụng

### **3.3.3. Thiết kế cơ sở dữ liệu**

Đặc điểm của CSDL NoSQL là kiến trúc không nhất quán và không giới hạn, tuy nhiên trong để nhất quán dữ liệu và dễ dàng cho các thao tác xử lý, ứng dụng sử dụng module mongoose để hỗ trợ tạo cấu trúc và thực hiện thao tác với CSDL MongoDB.

Phần dưới đây minh họa xây dựng cấu trúc của bộ sưu tập Từ vựng (tương đương với cấu trúc một bảng trong SQL)

Bảng 3.5: Cấu trúc Collection Vocabulary

<b>Thuộc tính</b>	<b>Kiểu dữ liệu</b>	<b>Điễn giải</b>
_id	ObjectId	Khoá
Word	String	Từ vựng
Unit	ObjectId reference Unit Model	ID tham chiếu tới bài học tương ứng
Define	String	Nghĩa của từ
Sentense_partern: {sentense, image}	Object	Sentense_partern chứa thông tin mẫu câu bao gồm câu và hình ảnh minh họa
Lang	String	Ngôn ngữ từ vựng, mặc định là 'EN'
Images	Array	Mảng hình ảnh, mỗi phần tử chứa một tên
Status	Boolean	Trạng thái từ vựng
Classes	String	Loại từ vựng

Mã xử lý để nạp cấu trúc trên vào ứng dụng như sau:

```
const mongoose = require('mongoose');
var VocabularySchema = new mongoose.Schema({
  unit: {
    type: Schema.ObjectId,
    ref: 'Unit',
    require: true
  },
  word: {
    type: String,
    require: true
  },
  ...
```

```

define: {
    type: String,
    require: true
},
sentense_pattern: {
    sentense: String,
    image: type: String
},
lang: {
    type: String,
    enum: ['en', 'vi'],
    default: 'en'
},
images: [
    _id: false,
    url: {
        type: String
    }
],
status: {
    type: Boolean,
    default: true
},
classes: {
    type: String,
    enum: ['noun', 'verb', 'adjective', 'adverb', 'pronoun',
           'conjunction', 'determiner', 'exclamation'],
    default: 'noun'
}
},
{
    collection: 'vocabulary',
    timestamps: true // Auto save timestamps, version
});
// Export module
module.exports = mongoose.model('Vocabulary', VocabularySchema);

```

Sau khi xây dựng được cấu trúc như trên, với sự hỗ trợ của module mongoose, chúng ta có thể dễ dàng quản lý các trường và thao tác với CSDL trên các Collection bằng cách tham chiếu ở bất kỳ đâu trên ứng dụng như sau:

```
const Vocabulary = mongoose.model('Vocabulary');
```

Bảng 3.6: Cấu trúc Collection Unit

<i>Thuộc tính</i>	<i>Kiểu dữ liệu</i>	<i>Điễn giải</i>
_id	ObjectId	Khoá

Index	Number	Số thứ tự bài học
Name	String	Tên bài học
Short_description	String	Mô tả ngắn

Bảng 3.7: Cấu trúc Collection User

<b>Thuộc tính</b>	<b>Kiểu dữ liệu</b>	<b>Điễn giải</b>
_id	ObjectId	Khoá
Name	String	Tên người dùng
Classes	String	Lớp
Short_description	String	Mô tả ngắn
Email	String	Email
Phone	String	Điện thoại
Avatar	String	Tên hình ảnh đại diện
Password	String	Mật khẩu
Neural_json	String	Mô hình nhận dạng

Bảng 3.8: Cấu trúc Collection Blog

<b>Thuộc tính</b>	<b>Kiểu dữ liệu</b>	<b>Điễn giải</b>
_id	ObjectId	Khoá
Name	String	Tiêu đề bài viết
Slug	String	Tiêu đề không dấu
Type	String	Loại bài viết
Auth	ObjectId reference User Model	Tác giả viết, tham chiếu tới Model User

Status	Boolean	Trạng thái
Views	Number	Số lượt xem

### 3.3.4. Cấu trúc xây dựng ứng dụng

Không giống như việc xây dựng ứng dụng website với các nền tảng của các ngôn ngữ PHP hay ASP đã có các framework và IDE hỗ trợ sẵn, việc xây dựng ứng dụng với JavaScript trên NodeJS rất linh động, nhà phát triển có thể chọn cách xây dựng ứng dụng của mình với các module hỗ trợ khác nhau. Đồng nghĩa với việc lập trình viên phải làm tất cả, không có bất kỳ một cơ chế hỗ trợ nào trong việc xây dựng ứng dụng website như chứng thực qua session hay các hỗ trợ thao tác với cơ sở dữ liệu. Qua quá trình tìm hiểu và thử nghiệm các module trên môi trường NodeJS, các module để xây dựng ứng dụng:

Bảng 3.9: Các module sử dụng xây dựng ứng dụng phía máy chủ

STT	Tên module	Chức năng
1	Hapi	Khởi tạo server ứng dụng
2	Mongoose	Driver cho MongoDB
3	Sqilte3	Driver cho SQLite
4	Redis	Driver cho Redis
5	Bcrypt	Mã hoá mật khẩu
6	Vision, Boom, Joi, Kue, Inert	Nhóm hỗ trợ xây dựng server
7	Hapi-auth-jwt2	Hỗ trợ chứng thực người dùng
8	Acl	Hỗ trợ phân quyền hệ thống
9	Handlebars	Render html
10	Jsonwebtoken	Hỗ trợ chứng thực client – server
11	Bluebird	Hỗ trợ Promise JS
12	OpenCV	Xử lý ảnh
13	Jimp	Hỗ trợ xử lý ảnh
14	Commander	Hỗ trợ thao tác hệ thống

15	Brain	Hỗ trợ xây dựng mạng nơ-ron
16	Moment	Hỗ trợ thao tác với ngày giờ
17	Audit-log	Lưu nhật ký hệ thống
18	Nodemailer	Gửi email

Cấu trúc ứng dụng được tách biệt thành hai thành phần bao gồm Client, và Server. Client là những gì được hiển thị trên trình duyệt của người sử dụng, cũng chính View trong mô hình MVC phía Server.

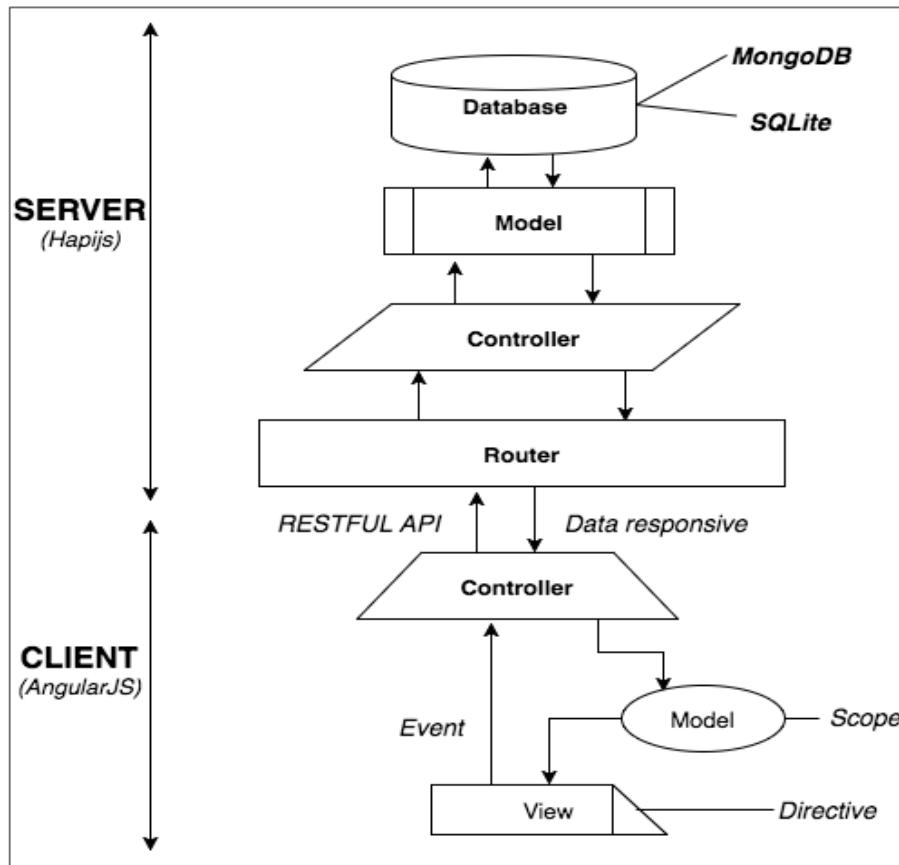
Phía Server bao gồm 4 thành phần là Router, Controller, Model, Database

- Router là một bộ định tuyến các request từ client lên server. Ngoài điều hướng các request tới các xử lý Controller tương ứng, Router còn đảm nhận nhiệm vụ chứng thực trên mỗi request.
- Controller là nơi xử lý logic cho ứng dụng
- Model đảm nhận nhiệm vụ thao tác thêm, xoá, sửa với cơ sở dữ liệu trên hệ thống.
- Hệ quản trị CSDL lưu trữ dữ liệu có thể NoSQL hoặc SQL. Thông thường chọn các hệ quản trị NoSQL.

Phía Client bao gồm 3 thành phần Controller, Model, View

- View bao gồm các đoạn mã HTML kết hợp với các thuộc tính mở rộng (directive), và các Model của Angular hiển thị lên trình duyệt.
- Controller phía client thực hiện các xử lý bao gồm gửi request tới server để lấy truyền hoặc nhận dữ liệu, xử lý logic trên View hiển thị
- Model là một đối tượng đặc biệt trong mô hình, đảm nhận nhiệm vụ đóng bộ dữ liệu từ controller tới View và ngược lại. Model ở phía Client chính là đối tượng scope của Angular.

Mô hình xây dựng ứng dụng MVC phía máy chủ kết hợp với MVVM trên trình duyệt:



Hình 3.7: Mô hình xây dựng ứng dụng

Cơ chế hoạt động chi tiết của ứng dụng như sau:

1. Người sử dụng truy cập vào website thực hiện một hành động (event), event này được gửi tới controller tương ứng
2. Tại controller sẽ tạo một request RESTFUL lên máy chủ thông qua các service của Angular như http, ngResouce, ...
3. Phía máy chủ nhận được một request sẽ chứng thực người dùng và kiểm tra dữ liệu request có được cho phép không. Nếu tất cả đều thoả mãn sẽ gọi Controller tương ứng để xử lý request. Trong ứng dụng xây dựng xử dụng module Hapi cùng với các module mở rộng kết hợp như hapi-auth, acl, json-web-token, redis để thực hiện thao tác chứng thực và phân quyền người sử dụng.
4. Khi thực hiện xử lý tại Controller nếu cần thao tác với CSDL, controller sẽ gọi tới các module hỗ trợ để thực hiện thao tác. Model xử lý các thao tác với dữ liệu tương ứng và trả kết quả về controller. Sau khi có kết quả thực hiện xử lý,

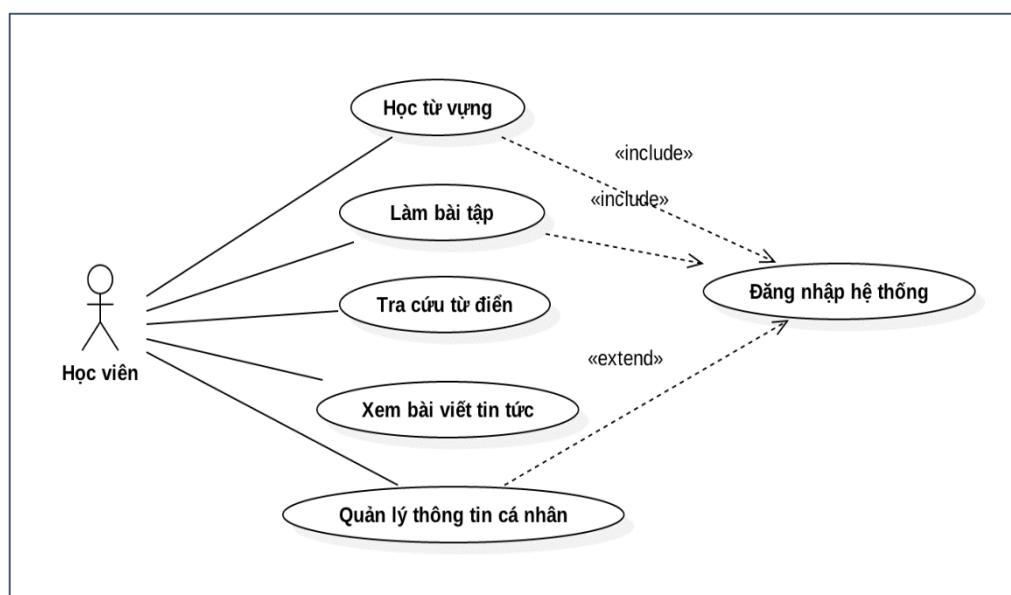
Controller sẽ trả về dữ liệu tương ứng. Trong ứng dụng sử dụng 2 module là mongoose và sqlite3 để hỗ trợ thao tác dữ liệu với các Hệ quản trị CSDL MongoDB và SQLite

5. Khi gửi request có kết quả trả về từ máy chủ, Controller Angular trên trình duyệt xử lý cập nhật đối tượng Model (scope trong Angular) tương ứng.
6. Khi Model được cập nhật sẽ tự động hiển thị lên trình duyệt kết quả tương ứng.

### **3.3.5. Các sơ đồ chức năng**

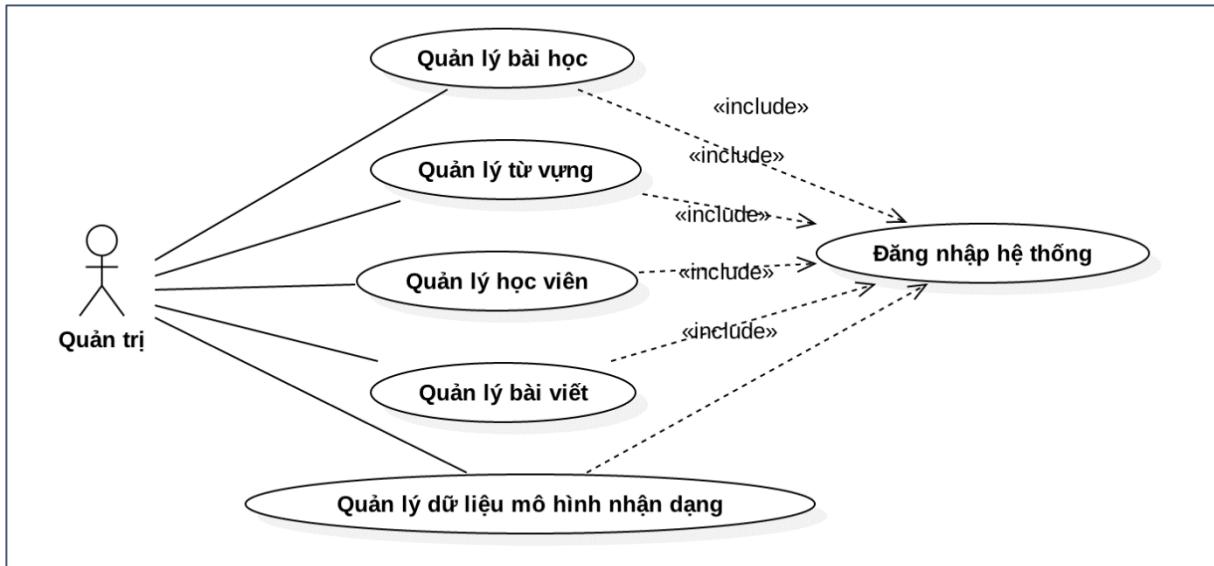
- **Use case tổng quát:** chia làm hai nhóm

- o Nhóm học viên:



Hình 3.8: Use case mirc 1 nhóm học viên

- o Nhóm quản trị



Hình 3.9: Use case mirc 1 nhóm quản trị

#### - Kịch bản

- Tính năng luyện tập từ vựng

Bảng 3.10: Kịch bản luyện tập từ vựng

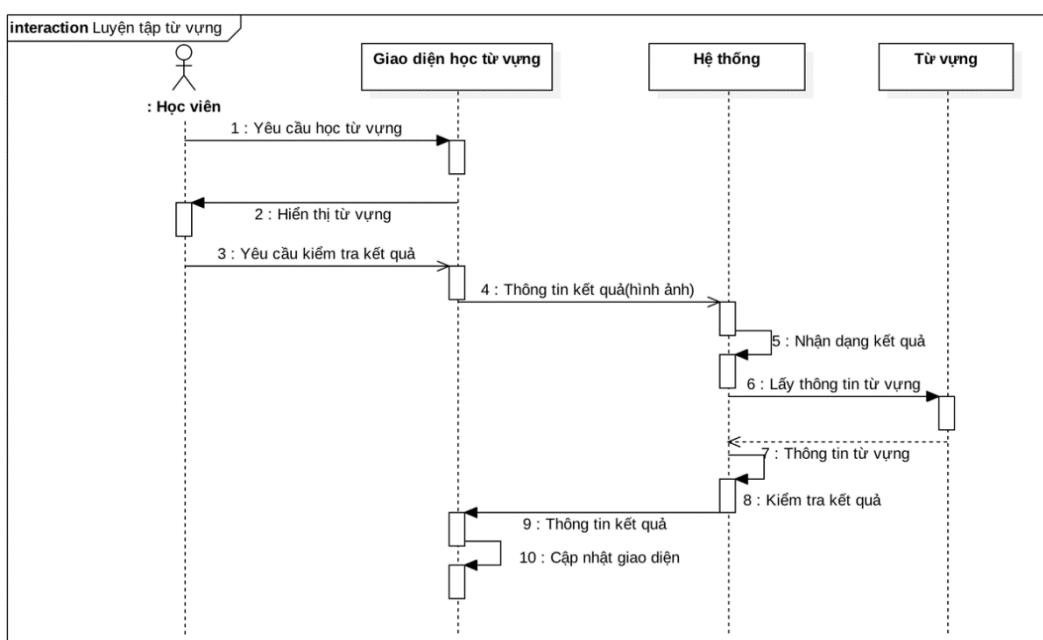
<b>Tên use case</b>	Luyện tập từ vựng
<b>Tác nhân chính</b>	Học viên
<b>Mức</b>	2
<b>Người chịu trách nhiệm</b>	Học viên
<b>Tiền điều kiện</b>	Học viên đã đăng nhập hệ thống
<b>Đảm bảo tối thiểu</b>	Hệ thống đưa về giao diện ban đầu
<b>Đảm bảo thành công</b>	Hiển thị kết quả tương ứng cho học viên
<b>Kích hoạt</b>	Học viên click nút “Luyện tập từ vựng” trên trang “Học từ vựng”
<b>Chuỗi sự kiện chính</b>	
<ol style="list-style-type: none"> <li>1. Hệ thống hiển thị giao diện luyện tập từ vựng</li> <li>2. Học viên đưa thông tin kết quả lên camera và nhấn “Chụp ảnh”</li> <li>3. Hệ thống xử lý hình ảnh đầu vào</li> <li>4. Hệ thống hiển thị kết quả nhận dạng và kết quả bài tập</li> </ol>	

## Ngoại lệ

- 1.1. Hệ thống không thể truy cập camera trên thiết bị
  - a. Hệ thống thông báo không truy cập được camera
  - b. Tắt giao diện học bài
- 3.1. Học viên tắt chức năng tự động xử lý kết quả
  - a. Hệ thống hiển thị popup xử lý hình ảnh
  - b. Học viên xử lý hình ảnh và nhấn “Cắt hình”
  - c. Hệ thống xử lý hình ảnh đã qua xử lý

### - Sơ đồ trình tự:

- o Tính năng học luyện tập từ vựng



Sơ đồ 3.1: Sơ đồ trình tự chức năng luyện tập từ vựng

### 3.4. Xây dựng ứng dụng

#### 3.4.1. Xây dựng và lưu trữ mạng neural

Quá trình huấn luyện sử dụng bộ dữ liệu xây dựng được lưu dưới dạng JSON. Sau khi huấn luyện xong sẽ lưu thông tin trọng số và kết quả đáng giá năng lực của mạng. Mã xử lý:

```

const brain = require('brain.js');           // module hỗ trợ xây dựng mạng
const jsonfile = require('jsonfile')         // module hỗ trợ lưu trữ file
json
const shuffle = require('shuffle-array');    // module hỗ trợ xáo trộn mảng
// Lấy dữ liệu bộ train và test sau quá trình tiền xử lý và trích rút đặc
trung
let trainingSet = require('data-training.json');
let testSet = require('data-testing.json');

// Xáo trộn dữ liệu
shuffle(trainingSet)
shuffle(testSet)

// Khởi tạo mạng
let net = new brain.NeuralNetwork({
  hiddenLayers: 80,           // số Neural lớp ẩn
  learningRate: 0.4          // bước học
});
console.time('train: ')
// Thực hiện huấn luyện
net.train(trainingSet, {
  errorThresh: 0.0001,        // ngưỡng lỗi chấp nhận
  momentum: 0.3,              // momentum
  iterations: 2000,           // số lần huấn luyện tối đa
});
// Tính thời gian huấn luyện
console.timeEnd('train: ')

// Lưu trữ trọng số mạng
let file = 'net1.json'
let obj = net.toJSON();
jsonfile.writeFile(file, obj, function (err) {
  if (err) throw err;
})
// ĐÁNH GIÁ NĂNG LỰC MẠNG
const arrayChart = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'l',
'k', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'];

let totalSetTest = testSet.length;
let OK = 0;

```

```

let detailTest = {
    count: new Array(26).fill(0), // số mẫu test của 26 ký tự
    ok: new Array(26).fill(0), // số mẫu đúng tương ứng
}
// Thực hiện thống kê
testSet.forEach(function (item, index) {
    let output = net.run(item.input);
    let numberOut = output.indexOf(Math.max.apply(null, output));
    detailTest.count[numberOut]++;
    let numberRes = item.output.indexOf(Math.max.apply(null, item.output));
    if (numberOut == numberRes) {
        OK++;
        detailTest.ok[numberOut]++;
    }
});
// Xuất thông tin huấn luyện
console.log('Số mẫu kiểm tra: ', totalSetTest);
console.log('Số mẫu nhận dạng đúng: ', OK);
console.log('Phần trăm nhận dạng: ', (OK / totalSetTest) * 100, '%')
for (let i = 0; i < 26; i++) {
    console.log(` ${arrayChart[i]} : ${detailTest.ok[i]} /
${detailTest.count[i]} :\t ${detailTest.ok[i] * 100 /
detailTest.count[i]} %`)
}

```

### 3.4.2. Tổ chức cấu trúc và cài đặt ứng dụng

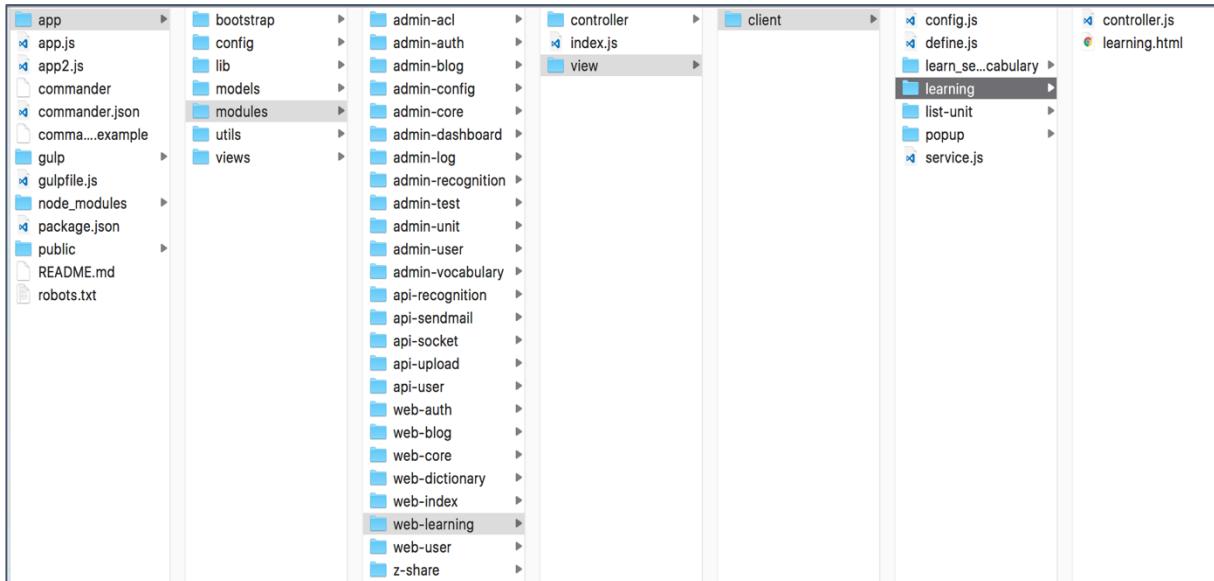
Như đã đưa ra ở phần lý thuyết, NodeJS chỉ là nền tảng do đó không có cấu trúc chung cho một ứng dụng, nhà phát triển sẽ tự xây dựng cấu trúc cho ứng dụng của mình. Hơn thế nữa NodeJS cũng không có công cụ giống như IIS được tích hợp sẵn trong hệ điều hành Windows. Theo mô hình ứng dụng hình 3.7, ứng dụng gồm 3 phần đó là phần ứng dụng cho học viên gọi là ứng dụng phía Web, trang quản trị nội dung gọi là ứng dụng phía Admin và phần dùng chung cho Admin và Web, có thể mở rộng ứng dụng sau này gọi là API. Để dễ quản lý các chức năng, mỗi chức năng được tách lưu trữ trong các thư mục khác nhau. Chi tiết cấu trúc xem hình 3.10 và 3.11.

```

Project
|--ocr # Chứa code xử lý nhận dạng chữ
|---code # Chứa mã xử lý
|   |-- img-process # Chứa mã xử lý ảnh
|   |-- neural-network # Chứa mã xử lý mạng nơ-ron
|   |-- image      # Thư mục chứa ảnh xử lý
|--package.json # Tập tin khai báo module sử dụng trong xử lý mạng nơ-ron
|--web-app # Chứa code website
|--package.json # Tập tin khai báo module sử dụng trong ứng dụng web
|--app # Chứa app chính
|---bootstrap # boot đầu tiên khi vào app
|   |--config # Chứa những config của site
|   |--lib # Chứa config thư viện được sử dụng và các plugin
|   |--model # Chứa tất model (collection) trong mongoses
|   |--modules # Gồm những module admin, web, api
|       |---(admin,web,api)-*
|       |--controller # Chứa controller server của module đó
|       |--util # chứa helper những function sử dụng chung cho module đó phía server
|       |--view # Phần client
|           |---client # Chứa code client của module đó
|           |--index.js    # File chứa chính để load module đó vào hapi
|           |--utils # phần helper dùng chung cho cả app, chứa middleware, event, socket của app
|           |--views # Chứa layout, partial của (app, module) và chứa helper của hadlebar template
|---node_modules # thư viện node
|---gulp # chứa các file phục vụ việc phát triển ứng dụng
|---public # phần công cộng
|   |--assets # Chứa (fonts, images, scripts, styles) của site
|       |---(admin,site)
|   |--bower_components # Chứa thư viện dùng bower.
|   |--fonts # Chứa fonts
|   |--images # Chứa Hình ảnh cắt html
|   |--script # Chứa js util và biến setting của (admin,site) (những phần dùng chung)
|   |--styles # Chứa css những phần dùng chung và file main.scss để import css
|---global # Chứa
|--libs # Chứa thư viện dùng ngoài không dùng bower
|--files # Chứa files upload

```

Hình 3.10: Mô hình tổ chức lưu trữ mã xử lý ứng dụng



Hình 3.11: Cấu trúc ứng dụng

Hình 3.11 thể hiện cách lưu trữ ứng dụng của tính năng học từ vựng của học viên, theo đó mã xử lý chi tiết theo mô hình ứng dụng như sau:

- Phần router phía máy chủ (tập tin index.js hình 3.7)

```
server.route({
  method: 'POST',
  path: '/recognition',
  handler: RecognitionController.recognition,
  config: {
  },
});

```

Phần Controller phía máy chủ:

```
function exercise_1(request, reply) {
  let unit = request.params.unit || null;
  Vocabulary.find({ unit: unit }).exec().then(function (vocabularys) {
    return reply.view('web-learning/view/client/exercise_1/exercise_1',
    {
      vocabularys,
      menu: { learning: true }
    }, { layout: 'web/layout' });
  }).catch(function (err) {
    return reply.redirect('/error404');
  })
}

```

**RecognitonUtil** là module được tách riêng, có nội dung như sau:

```
let brain = require('brain');
const Jimp = require('jimp');
const width = 10, height = 15;
const nguong = 20;

const arrayChart = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'l',
  'k', 'm', 'n', 'o',
  'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'];

var json = require('./network.json');
let net = new brain.NeuralNetwork();
net.fromJSON(json);

module.exports = {
  recognition
}

function recognition(imgSrc) {
  return new Promise(function (resolve, reject) {
    let input = [];
    
```

```
Jimp.read(imgSrc, function (err, image) {
    if (err) return reject(err);
    image.resize(10, 15).scan(0, 0, image.bitmap.width,
image.bitmap.height, function (x, y, idx) {
        var red = this.bitmap.data[idx + 0];
        var green = this.bitmap.data[idx + 1];
        var blue = this.bitmap.data[idx + 2];
        var alpha = this.bitmap.data[idx + 3];
        let tbc = (red + green + blue) / 3;
        if (tbc < nguong) {
            input.push(0);
        }
        else {
            input.push(1);
        }
    }).autocrop(function () {
        return resolve({
            input: input,
            output: output,
            arrayChart: arrayChart,
            charPredict: arrayChart[indexOutput]
        });
    })
})
});
```

- Phần Model phía máy chủ là model mục 3.3.3
  - Phần View phía trình duyệt (tập tin learning.html hình 3.11):

```
<div id="popup-exercise_1" class="bsPopup">
  <div>
    <h3>LUYỆN TẬP TỪ VỰNG</h3>
  </div>
  <div>
    <div>
      <div>
        <div ng-show="vmExercise_1.hasImage">
          <div>
            
            
            <canvas id="snapshot" ng-show="false"></canvas>
          </div>
          <button ng-if="!isProcessed" ng-click="vmExercise_1.processImg>
Xử lý ảnh</button>
```

```

        <button ng-if=" isProcessed " ng-
click="vmExercise_1.checkResult() "Kiểm tra kết quả</button>

        <button ng-click="vmExercise_1.resetImage() ">Chụp lại</button>
    </div>
    <div class="live-video " ng-show="!vmExercise_1.hasImage ">
        <div class="text-center ">
            <webcam on-stream="vmExercise_1.onStream(stream) " on-access-
denied="vmExercise_1.onError(err)
                " on-streaming="vmExercise_1.onSuccess() "
                channel="vmExercise_1.channel ">
            </webcam>
        </div>

        <button type="button " ng-click="vmExercise_1.makeSnapshot() ">
Chụp ảnh</button>

        <button type="button " ng-click="vmExercise_1.exit() ">
Quay lại</button>
    </div>
</div>
</div>
<div class="modal-footer fix-modal-footer ">
    <div class="row ">
        <div class="col-md-4 ">
            <label><input type="checkbox " ng-model="
vmExercise_1.configs.autoCropImage " name="autoCropImg " value=" "> &nbsp
Tự động xử lý ảnh và kiểm tra kết quả</label>
        </div>
        <div class="col-md-3 ">
            <label><input type="checkbox " ng-
change="vmExercise_1.onChangeTimeDelay() " ng-model="
vmExercise_1.configs.delayTime
                " name="autoCapture " value=" ">
Chụp sau 3 giây</label>
        </div>
    </div>
</div>
</div>

```

- Model phía trình duyệt trong mô hình đưa ra chính là Object **vmExcercise** trong đoạn mã xử lý phía trên
- Controller phía trình duyệt (tập tin controller.js hình 3.11):

```
var exercise_1_Ctrl = (function () {
```

```
'use strict';

angular
    .module('bsLearning')
    .controller('exercise_1_Ctrl', exercise_1_Ctrl);

function exercise_1_Ctrl($scope) {
    var vmExercise_1 = this;
    vmExercise_1.processImg = processImg;
    vmExercise_1.checkResult = checkResult;

    function processImg() {
        LearnSvc.processImage({
            directory: settingJs.configs.uploadDirectory.tmp,
            name: vmExercise_1.image.name
        }).then(function (resp) {
            vmExercise_1.imgProcessed.isProcessed = true;
            vmExercise_1.imgProcessed.image = resp.resp;
        }).catch(function (err) {
        })
    }

    function checkResult() {
        LearnSvc.recognition({
            name: vmExercise_1.imgProcessed.image.name
        }).then(function (resp) {
            vmExercise_1.textRecognition.identified = true;
            vmExercise_1.textRecognition.char = resp.charPredict;
            vmExercise_1.textRecognition.data = resp;
        }).catch(function (err) {
        })
    }
}
);

```

Mã xử lý để khởi động ứng dụng (tập tin app.js hình 3.11)

```
// Trong đoạn mã xử lý này đã cắt một số nội dung nâng cao
const Hapi = require('hapi');
const server = new Hapi.Server();
global.BASE_PATH = __dirname;
server.register({
    register: require('hapi-kea-config'),
    options: {
        confPath: BASE_PATH + '/app/config',
        decorateServer: true
    }
});

const config = server.plugins['hapi-kea-config'];
```

```

/* Cài đặt các biến toàn cục cho ứng dụng */
global.config = {};
global.config.web = config.get('web');

let connections = config.get('web.connections');
connections.forEach(function (config) {
    server.connection(config);
}, this);
// Cấu hình template cho ứng dụng
server.views({
    engines: {
        html: require('handlebars')
    },
    relativeTo: global.BASE_PATH + '/app/modules',
    partialsPath: global.BASE_PATH + '/app/views/layouts',
    helpersPath: global.BASE_PATH + '/app/views/helpers',
    layoutPath: global.BASE_PATH + '/app/views/layouts',
    layout: function () {
        return 'web/layout';
    },
    context: config.get('web.context')
});

//Xử lý tự động load các module vào ứng dụng
server.connections.forEach(function(connectionSetting) {

    let labels = connectionSetting.settings.labels;
    labels.forEach(name => {
        let modules = [];
        let modulesName = Glob.sync(BASE_PATH + `/app/modules/${name}--*/index.js`, {});
        modulesName.forEach((item) => {
            modules.push(require(Path.resolve(` ${item}`)));
        });
        if (modules.length) {
            let options = { select: [name] };
            if (name == 'api') {
                options.routes = { prefix:
config.get('web.context.apiprefix') };
            }
            if (name == 'admin') {
                options.routes = { prefix:
config.get('web.context.cmsprefix') };
            }
            server.register(modules, options, (err) => {
                if (err) {
                    server.log(['error', 'server'], err);
                }
            });
        }
    });
});

```

```

        }
    });
}

//start the server
server.start((err) => {
    if (err) {
        throw err;
    }
});

module.exports = server;

```

Mã xử lý chúc trung gian xác thực người dùng:

```

function (server, options, next) {
    var config = server.configManager;

    function validate(decoded, request, callback) {
        const redisClient = server.redis;

        redisClient.get(new String(decoded.id).toString(), function
(rediserror, result) {
            if (rediserror) {
                server.log(['error','redis','validateauth'], rediserror);
            }

            let session;
            if (result) {
                session = JSON.parse(result);
            }else{
                return callback(rediserror, false);
            }
            if (session.valid === true) {
                return callback(rediserror, true);
            }else {
                return callback(rediserror, false);
            }
        });
    }

    function registerJwtAuth2(err) {
        if (err) {
            console.log('erro',err);
        }
        server.auth.strategy('jwt', 'jwt', 'optional', {
            key: config.get('web.jwt.secret'), validateFunc: validate,
            errorFunc: function(err){
                console.log('error_auth', err);
            },
        },
    }
}

```

```

        verifyOptions: { ignoreExpiration: false, algorithms: ['HS256']
    }
});

return next();
};

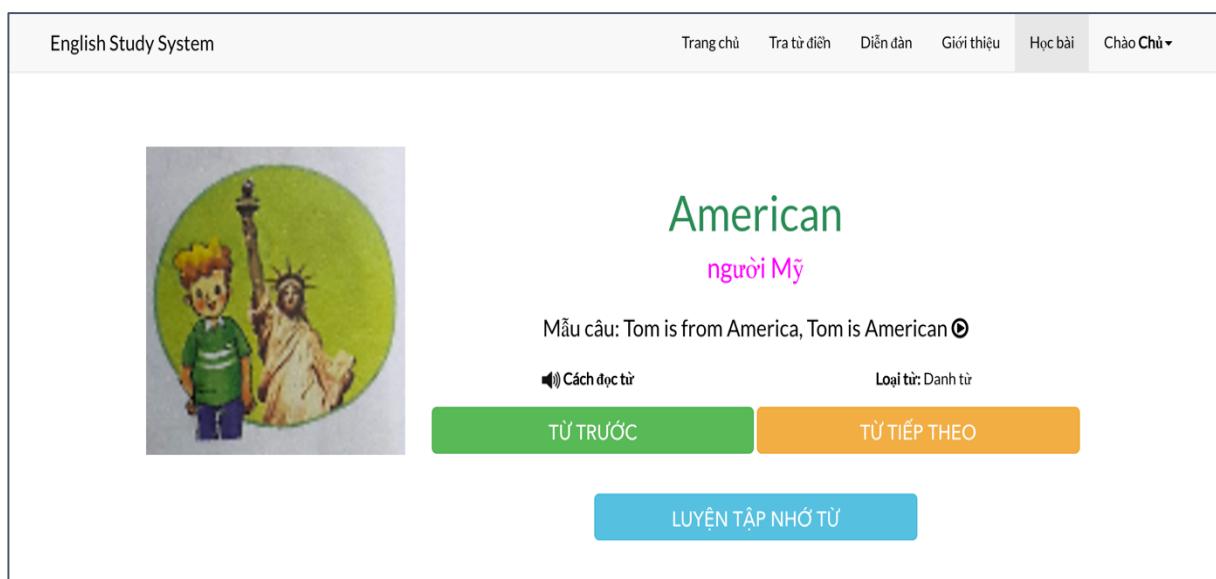
server.register(AuthJwt2, registerJwtAuth2);
}

```

### 3.5. Kết quả thực hiện

#### 3.5.1. Các chức năng của học viên

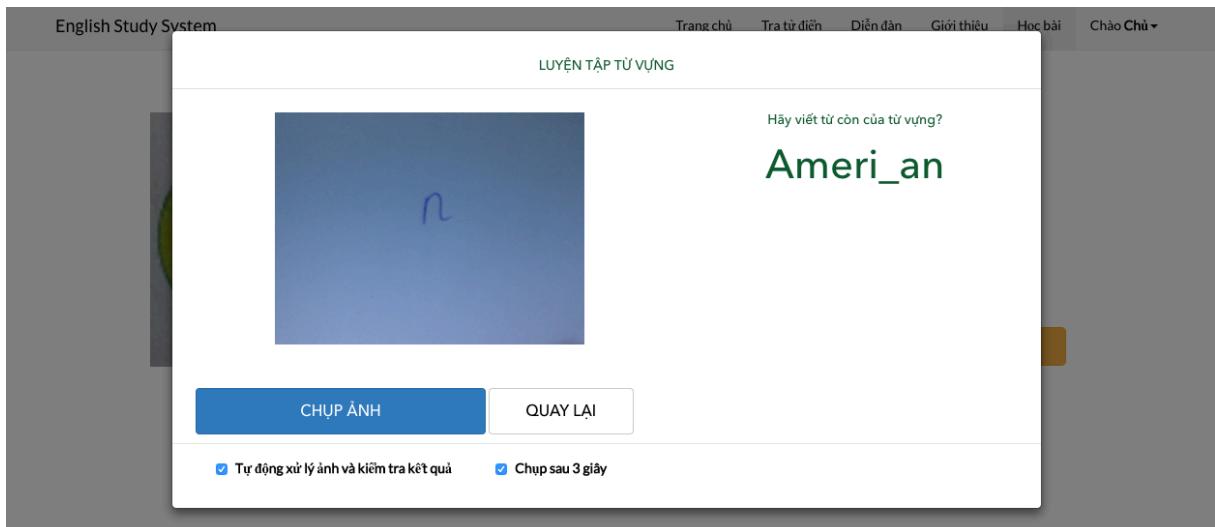
**Chức năng học từ vựng** theo bài cho phép học sinh truy cập vào ứng dụng và học từ vựng theo bài, ứng dụng hỗ trợ học sinh học các phát âm, nhớ loại từ và hiển thị một số câu hỏi thoại đơn giản giúp học sinh có thể nhớ từ theo ngữ cảnh và những câu giao tiếp thông dụng.



Hình 3.12: Giao diện trang học từ vựng

**Chức năng luyện tập từ vựng**, giúp học sinh có thể nhớ tốt hơn bằng cách luyện viết từ tương đương với dạng bài tập trong phần Vocabulary của sách giáo khoa. Mỗi từ vựng sẽ bị khuyết ngẫu nhiên các ký tự. Học sinh sẽ cần phải viết vào giấy phần khuyết của từ vựng sau đó đưa lên trước camera của máy tính để máy tính kiểm tra. Sau khi kiểm tra kết quả bằng cách nhận dạng ký tự mà học sinh viết, hệ thống sẽ hiển thị thông báo cho học sinh, sau đó học sinh có thể làm tiếp từ vựng khác hoặc làm tiếp tục luyện

tập từ vựng hiện tại. Trong quá trình học, học sinh có thể cài đặt hệ thống tự động xử lý kết quả hoặc xử lý thủ công.



Hình 3.13: Giao diện luyện tập từ vựng

**Chức năng luyện tập từ vựng theo mẫu câu:** sau khi học viên đã học qua các từ vựng trong bài, học viên sẽ rèn luyện từ vựng theo các mẫu câu. Cách làm bài tập tương tự như làm bài tập với từ vựng. Hệ thống sẽ hiển thị một câu mẫu và khuyết đi từ vựng, học viên sẽ quan sát một bức tranh minh họa cho từ vựng. Nhiệm vụ của học viên là viết từ còn thiếu trong câu và đưa lên trước camera máy tính. Và nhấn nhấn chụp ảnh hệ thống sẽ kiểm tra và hiển thị kết quả tương ứng.



Hình 3.14: Giao diện trang luyện tập từ vựng theo câu

**Chức năng tra cứu từ vựng** cho phép học sinh tra cứu từ vựng mở rộng, ngoài các từ vựng được học trong bài. Dữ liệu từ điển được lấy từ dự án từ điển mã nguồn mở andict<sup>[30]</sup>. Dữ liệu từ vựng được lưu trong tập tin sqlite với khoảng 380.000 từ Anh – Việt và 390.000 từ Việt - Anh.

English Study System

Trang chủ Tra từ điển Diễn đàn Giới thiệu Học bài Chào Chủ▼

## Tra từ điển trực tuyến

hello

Anh → Việt

hello /hə'lou/ (halloa) /hə'lou/ (hello)  
/'hə'lou/

thân từ

- chào anh!, chào chị!
- này, này
- ô này! (tò ý ngạc nhiên)

danh từ

- tiếng chào
- tiếng gọi "này, này" !
- tiếng kêu ô này ! (tò ý ngạc nhiên)

nội dung từ

- chào
- gọi "này, này"
- kêu "ô này" (tò ý ngạc nhiên)

English Study System

Hệ thống học từ vựng trực tuyến.  
Tra cứu từ điển chuyên ngành,  
Tra cứu bằng hình ảnh  
Nhanh chóng, chính xác.

Apple Android f Q

Hình 3.15: Giao diện trang tra từ điển

English Study System

Trang chủ Tra từ điển Diễn đàn Giới thiệu Học bài Chào Chủ▼

Trang chủ > Diễn đàn > HỌC NÓI TIẾNG ANH HIỆU QUẢ NHANH NHẤT

HỌC NÓI TIẾNG ANH HIỆU QUẢ NHANH NHẤT

Thứ sáu , 5 tháng 6 , 2017

Việc dạy học nói tiếng Anh trẻ em lớp 4 như bạn đã biết chủ yếu là "bắt chước". Tuy nhiên, SPEAKING là kỹ năng đòi hỏi độ khó và đều khiến mọi người choáng ngợp. Nếu như kỹ năng Listening, Writing hay Grammar, bé có thể tìm cách dạy học tiếng Anh lớp 4 hiệu quả nhất bằng hàng đồng tài liệu thì kỹ năng nói lại đòi hỏi người học phải bắt chước linh tục. Vì thế, bố mẹ đừng ngần ngừ mà khuyến khích bé mạnh dạn bắt chước những câu nói, cách nhấn nhá câu của người bản ngữ, chẳng mấy chốc, bé sẽ có thể nói giống y hệt đấy.

**HÃY CHO TRẺ HỌC NÓI TIẾNG ANH TRONG MÔI TRƯỜNG HỌC CHUẨN NHẤT**

Cách dạy học tiếng Anh lớp 4 hiệu quả nhất là bé được tiếp xúc với môi trường nói tiếng Anh và được trực tiếp tự rèn luyện trong môi trường đó. Mỗi trường giúp đóng góp tích cực vào phương pháp học của người học ngôn ngữ. Chỉ cần bé có được môi trường tích cực, bé sẽ "lên tay" một cách chóng mặt. Tiếng Anh là ngôn ngữ toàn cầu nên việc tìm nơi hội tụ các bé ham học tiếng Anh không hề khó. Bố mẹ hãy châm chỉ đưa bé đến các câu lạc bộ để luyện nói tiếng Anh cùng các bạn cùng lứa tuổi, chắc chắn việc học tiếng Anh trẻ em kỹ năng sẽ hiệu quả nhanh chóng.

**DẠY BÉ HỌC NÓI TIẾNG ANH QUA TỐC ĐỘ NỐI**

Một điểm hạn chế của tiếng Anh là những người muốn nói lưu loát thường nói khá nhanh khiến người nghe không thể nắm bắt được hết thông tin. Lời khuyên khi **đạy tiếng Anh lớp 4** cho trẻ là hãy nói thật chậm, rõ ràng, thỉnh thoảng nghỉ giữa các từ với nhau cho những từ khó phát âm.

**TAG NỔI BẬT**

Computer Serv... 162 lượt thích

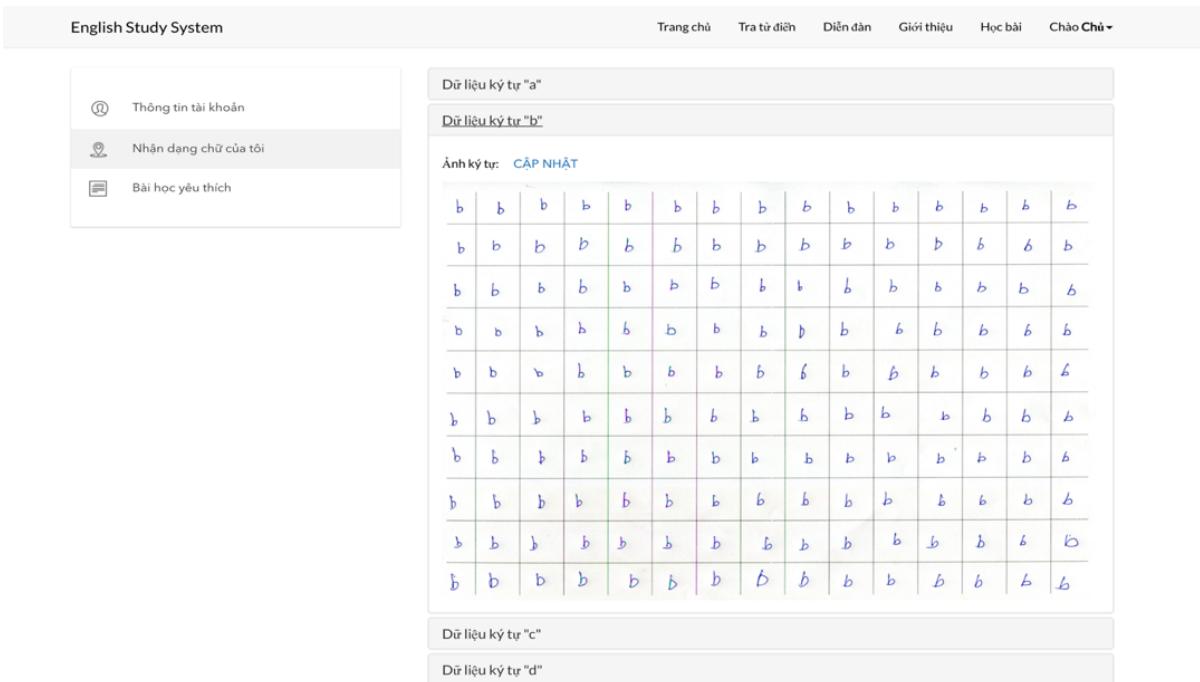
Đã thích

Bạn và 10 người bạn khác thích nội dung này

Hình 3.16: Giao diện trang đọc tin tức, bài viết

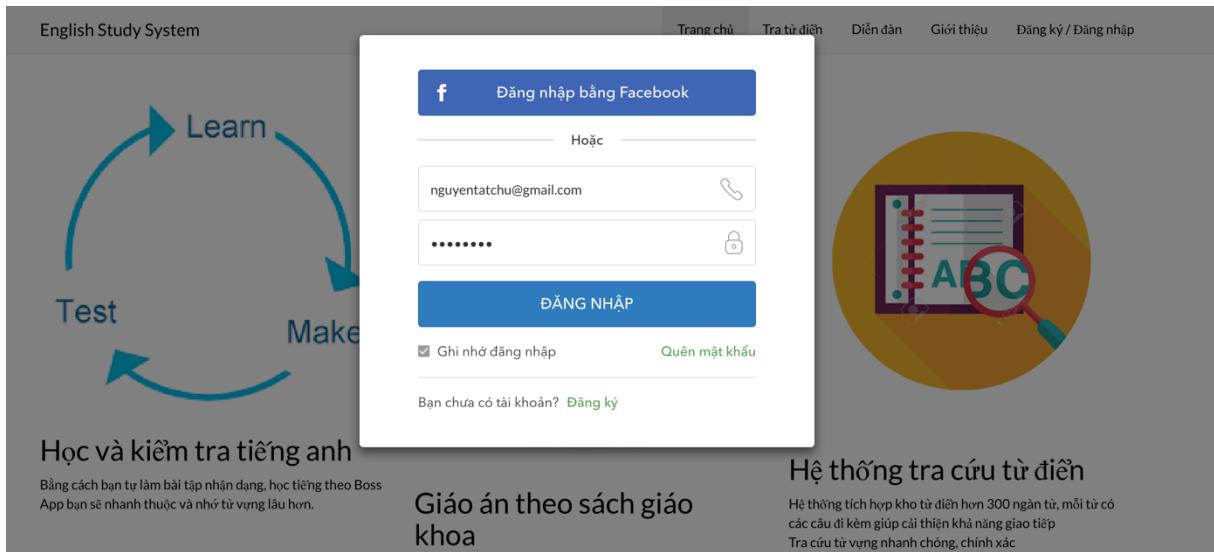
**Tính năng gửi yêu cầu nhận dạng chữ**, để phù với tính đa dạng của chữ viết tay và khắc phục nhược điểm của ứng dụng có bộ dữ liệu hạn chế, hệ thống cho phép

học viên nhận dạng chữ của chính học viên bằng cách tải lên hình ảnh kiểu viết của mình. Mỗi ký tự trong bảng chữ cái, học viên sẽ viết vào một tờ giấy, sau đó chụp ảnh hoặc scan để upload lên hệ thống. Khi đủ 26 ký tự, hệ thống sẽ cho phép học viên gửi yêu cầu tới quản trị viên. Khi nhận được yêu cầu từ học viên, quản trị sẽ kiểm tra yêu cầu. Nếu yêu cầu hợp lý, quản trị viên tải hình ảnh các ký tự về và thực hiện xử lý thủ công, sau đó cập nhật mô hình nhận dạng cho học viên.



Hình 3.17: Giao diện trang yêu cầu nhận dạng chữ học viên

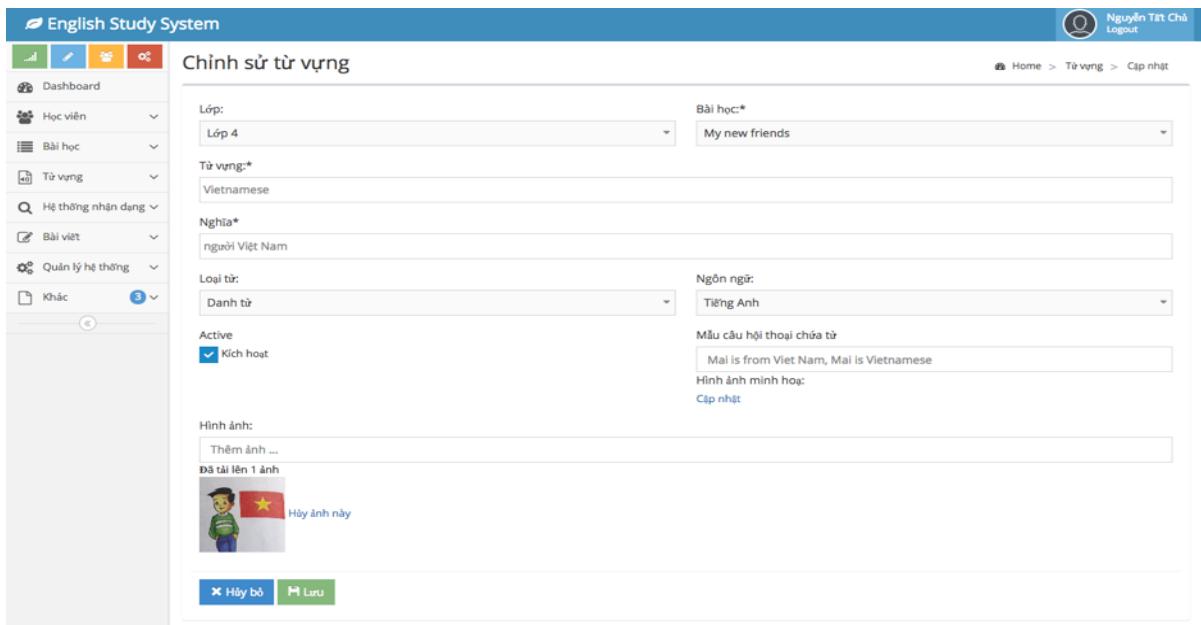
**Chức năng đăng nhập – chứng thực hệ thống:** giúp hệ thống ngăn chặn các thao tác trái phép, và truy sử dụng ứng dụng không đúng mục đích. Cơ chế chứng thực sử dụng module mở rộng hapi-jwt và acl để kiểm tra thông tin chứng thực trên mỗi request từ người dùng gửi lên. Nếu trong request gửi lên có thông tin người dùng bao gồm id người dùng, quyền. Nếu quyền được cho phép thì yêu cầu được thực hiện, ngược lại sẽ trả về kết quả yêu cầu người dùng chứng thực. Thông tin của người lưu trên trình duyệt dùng được mã hoá bằng module json-web-token, giúp bảo vệ thông tin cá nhân.



Hình 3.18: Giao diện đăng nhập hệ thống

### 3.5.2. Các chức năng nhóm quản lý

**Quản lý từ vựng:** cho phép quản trị viên thêm xoá sửa từ vựng, lọc từ vựng theo bài học. Mỗi từ vựng được gán kèm với ít nhất một hình ảnh. Khi thêm hoặc sửa từ vựng hệ thống cho phép quản trị viên tải hình ảnh và điều chỉnh hình ảnh theo ý muốn.



Hình 3.19: Giao diện trang chỉnh sửa từ vựng

**Quản lý bài học** cho phép quản trị viên thêm, xoá, sửa cập nhật các nội dung như tên bài học, mô tả nội dung bài học, ...

STT	Tên	Mô tả ngắn	Trạng thái	Action
1	Good Morning. How are you?	Các em sẽ được học về các buổi trong ngày, các câu chào hỏi trong thường dùng các buổi	Active	
2	My new friends	Học các nước trên thế giới, cách chào hỏi và giới thiệu bản thân với bạn bè trên thế giới	Active	
3	My birthday	Ở bài học này các em sẽ được học về ngày tháng, học cách đọc viết về ngày sinh của mình và bạn bè	Active	
4	Things I can do	Chúng ta sẽ biết được những việc chúng ta có thể làm trong tiếng anh, miêu tả chúng và giới thiệu những công việc chúng ta yêu thích với bạn bè	Active	
5	Our Hobbies	Sở thích của bạn là gì? Từ bài học này chúng ta sẽ biết được sở thích của mình trong tiếng anh để có thể nói chuyện được với các bạn nước ngoài	Active	
6	My school	Bạn học lớp mấy rồi? Trường của bạn ở đâu? Bạn đã biết cách giới thiệu về trường của mình với các bạn nước ngoài chưa	Active	
7	My school subject	Bài học số 7 sẽ cho chúng ta biết các môn học trong tiếng anh	Active	

Hình 3.20: Trang danh sách từ vựng

**Chức năng quản lý người dùng**, cho phép quản trị cập nhật các thông tin và cấp quyền cho các tài khoản trên hệ thống

Hình 3.21: Giao diện trang chỉnh sửa thông tin người dùng

Ngoài các chức năng chính hệ thống còn có các chức năng quản lý bài viết giúp quản trị viên quản lý danh sách các bài viết, thêm xoá sửa trên các bài viết; quản lý dữ

liệu huấn luyện mạng nơ-ron cho phép quản trị viên cập nhật mô hình nhận dạng giúp quản trị chủ động cập nhật dữ liệu cho hệ thống và các chức năng thuộc nhóm yêu cầu chức năng đăng nhập hệ thống, quản lý phân quyền, quản lý nhật ký hệ thống.

STT	Actor	Action	Label	Origin	Description	Date	Action
41	tatchu.it@gmail.com	save	vocabulary	mongoose	add new vocabulary	11/05 lúc 10:43PM	Xem chi tiết
42	tatchu.it@gmail.com	save	vocabulary	mongoose	add new vocabulary	11/05 lúc 10:42PM	Xem chi tiết
43	tatchu.it@gmail.com	save	vocabulary	mongoose	add new vocabulary	11/05 lúc 10:41PM	Xem chi tiết
44	tatchu.it@gmail.com	save	vocabulary	mongoose	add new vocabulary	11/05 lúc 10:39PM	Xem chi tiết
45	tatchu.it@gmail.com	update	vocabulary	mongoose	update vocabulary	11/05 lúc 10:34PM	Xem chi tiết
46	tatchu.it@gmail.com	save	vocabulary	mongoose	add new vocabulary	11/05 lúc 10:34PM	Xem chi tiết
47	tatchu.it@gmail.com	save	vocabulary	mongoose	add new vocabulary	11/05 lúc 10:31PM	Xem chi tiết
48	tatchu.it@gmail.com	save	vocabulary	mongoose	add new vocabulary	11/05 lúc 10:31PM	Xem chi tiết

Hình 3.22: Giao diện trang nhật ký hệ thống

Bài viết - Blog

Tên bài viết\*  
HỌC NÓI TIẾNG ANH HIỆU QUẢ NHANH NHẤT

Slug / Đường dẫn liên kết\*  
hoc-noi-tieng-anh-hieuqua-nhanhnhat

Mô tả ngắn  
bắt chước linh tinh. Vì thế, bố mẹ đừng ngại ngùng mà khuyên khích bé mạnh dạn bắt chước những câu nói, cách nhăn nhá câu của người bản ngữ, chẳng mấy chốc, bé sẽ có thể nói tiếng Anh như một chuyên gia.

Active  Publish Post

Hình ảnh - Hiện có 1 ảnh

Nội dung bài viết

**Hãy cho trẻ học nói tiếng Anh trong môi trường học chuẩn nhất**

Cách dạy học tiếng Anh lớp 4 hiệu quả nhất là bé được tiếp xúc với môi trường nói tiếng Anh và được trực tiếp tự rèn luyện trong môi trường đó. Môi trường giúp đóng góp tích cực vào phương pháp học của người học ngôn ngữ. Chỉ cần bé có được môi trường tích cực, bé sẽ "lên tay" một cách chóng mặt. Tiếng Anh là ngôn ngữ toàn cầu nên việc tìm nơi hội tụ các bé ham học tiếng Anh không hề khó. Bố mẹ hãy chăm chỉ đưa bé đến các câu lạc bộ để luyện nói tiếng Anh cùng các bạn cùng lứa tuổi, chắc chắn việc học tiếng Anh trẻ em kĩ năng nói sẽ hiệu quả nhanh chóng.

Hình 3.23: Giao diện trang thêm bài viết

**Chức năng phân quyền**, quản trị viên tạo ra các nhóm người dùng có vai trò khác nhau, với mỗi nhóm vai trò có thể phân quyền thêm, xoá, sửa, xuất báo cáo chi tiết trên từng nhóm. Sau đó gán nhóm vai trò cho từng người dùng.

Hình 3.24: Giao diện trang phân quyền

Chức năng cập nhật mô hình huấn luyện cho yêu cầu nhận dạng của học viên:

Hình 3.25: Giao diện trang cập nhật mô hình nhận dạng

## CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 4.1. Kết quả đạt được

Sau quá trình tìm hiểu môi trường NodeJS và nghiên cứu giải thuật phân lớp dữ liệu với mạng Neural cùng một số kỹ thuật có liên quan, đề tài đã đạt được mục tiêu đề ra là xây dựng ứng dụng hỗ trợ các em học sinh khối 4 học tiếng Anh học từ vựng, làm bài tập và kiểm tra kết quả trực tiếp, ngoài ra tích hợp thêm các tính năng như tra từ điển, đọc tin bài viết, chi tiết bao gồm:

- Xây dựng tập từ vựng, hình ảnh cho học sinh lớp 4
- Sử dụng kỹ thuật tìm biên để tách ghép ký tự thành từ với sự hỗ trợ của module opencv, jimp và sử dụng phương pháp trích chọn đặc trưng chia lưỡi hình ảnh làm đầu vào cho mạng nơ-ron để xây dựng tính năng luyện tập từ vựng
- Cài đặt ứng dụng với trên nền tảng NodeJS, hệ quản trị CSDL MongoDB, SQLite, Redis và thư viện AngularJS

Bên cạnh kết quả đạt được, do thời gian và kiến thức có hạn nên ứng dụng vẫn có những hạn chế và thiếu sót:

- Khả năng nhận dạng chưa cao đối với các kiểu chữ khác biệt so với bộ dữ liệu huấn luyện do bộ dữ liệu huấn luyện chưa được nhiều và chưa đa dạng
- Các dạng bài tập chưa đa dạng
- Khả năng xử lý ảnh chưa cao, đôi khi không đạt được kết quả theo mong muốn, phương pháp tách ký tự không tách được các ký tự viết liền nhau.
- Hệ thống không nhận dạng các ký tự viết hoa

### 4.2. Hướng phát triển

Để đưa ứng dụng tới các em học sinh, trong thời gian tới cần phải những khuyết điểm của ứng dụng đó là xây dựng bộ dữ liệu huấn luyện đa dạng, đủ lớn, thiết kế thêm các dạng bài tập phù hợp với sách giáo khoa. Để ứng dụng thực sự có hiệu quả và thiết thực hơn nữa cần đưa ứng dụng lên các nền tảng di động Android và iOS bằng cách sử dụng các API sẵn có của ứng dụng hiện tại.

Nghiên cứu các kỹ thuật xử lý ảnh, để cải thiện hiệu xuất và tốc độ nhận dạng của các ứng dụng.

Bên cạnh các hướng phát triển liên quan tới ứng dụng của đề tài vì khả năng ứng dụng của mạng nơ-ron là rất cao, do đó cần những nghiên cứu để ứng dụng thêm vào những ứng dụng thiết thực cho cuộc sống và bên cạnh đó là nghiên cứu các kỹ thuật để cải thiện tối đa mức nhận dạng phân biệt mẫu của mạng.

## DANH MỤC TÀI LIỆU THAM KHẢO

---

- [<sup>1</sup>] “Project NodeJS”, <https://github.com/nodejs/node>
- [<sup>2</sup>] “NodeJS Documentation”, <https://nodejs.org/en/download/>
- [<sup>3</sup>] “Package manager for Windows”, <https://chocolatey.org>
- [<sup>4</sup>] “Package manager for macOS”, <https://brew.sh>
- [<sup>5</sup>] “Node Package manager”, <https://www.npmjs.com>
- [<sup>6</sup>] “Node Package manager”, <https://yarnpkg.com/lang/en/>
- [<sup>7</sup>] Yen, Stephen. “NoSQL is a Horseless Carriage” (PDF). NorthScale.
- [<sup>8</sup>] “MongoDB”, <https://www.mongodb.com/download-center>
- [<sup>9</sup>] “Redis DB”, <http://download.redis.io/redis-stable.tar.gz>
- [<sup>10</sup>] “Project Redis”, <https://github.com/nrk/redis>
- [<sup>11</sup>] McCulloch, Warren; Walter Pitts (1943). “A Logical Calculus of Ideas Immanent in Nervous Activity”. Bulletin of Mathematical Biophysics.
- [<sup>12</sup>] Robert J. Schalkoff, Hill Companies Inc (1997) – Artificial Neural Networks, the McGraw, tr.71-104
- [<sup>13</sup>] “Stochastic gradient descent”,  
[https://en.wikipedia.org/wiki/Stochastic\\_gradient\\_descent](https://en.wikipedia.org/wiki/Stochastic_gradient_descent)
- [<sup>14</sup>] “Newton’s method”, [https://en.wikipedia.org/wiki/Newton's\\_method](https://en.wikipedia.org/wiki/Newton's_method)
- [<sup>15</sup>] “Hướng dẫn thuật toán Gradient descent”,  
<http://machinelearningcoban.com/2017/01/12/gradientdescent/>
- [<sup>16</sup>] “Project module brain”, <https://github.com/harthur-org/brain>
- [<sup>17</sup>] “Module neural network”, <https://www.npmjs.com/package/neuralnetwork>
- [<sup>18</sup>] “Module Dnn”, <https://github.com/junku901/dnn>
- [<sup>19</sup>] “Module sysnatic”, <https://github.com/cazala/synaptic>
- [<sup>20</sup>] “Module node neural network”, <https://www.npmjs.com/package/node-neural-network>
- [<sup>21</sup>] “Module mind”, <https://github.com/stevenmiller888/mind>
- [<sup>22</sup>] “Module converjs”, <https://github.com/karpathy/convnetjs>
- [<sup>23</sup>] “Module dn2a”, <https://github.com/dn2a/dn2a-javascript>
- [<sup>24</sup>] Phạm Anh Phượng (2009), “Một số phương pháp trích chọn đặc trưng hiệu quả cho bài toán nhận dạng chữ viết tay rời rạc”, Tạp chí khoa học Đại học Huế, số 53
- [<sup>25</sup>] “Project module node opencv”, <https://github.com/peterbraden/node-opencv>
- [<sup>26</sup>] “Module jimp”, <https://github.com/oliver-moran/jimp>
- [<sup>27</sup>] Garcia E.K., Gupta M.R. and Jacobson N.P. (2007), “OCR binarization and image pre-processing for searching historical documents”,
- [<sup>28</sup>] “Canny Edge Detector Algorithm”, tr 389 -  
391 [https://en.wikipedia.org/wiki/Canny\\_edge\\_detector](https://en.wikipedia.org/wiki/Canny_edge_detector)
- [<sup>29</sup>] “Module Neural Network NodeJS”, <https://github.com/harthur-org/brain.js>
- [<sup>30</sup>] “Dự án từ điển mã nguồn mở andict”,  
<https://code.google.com/archive/p/andict>