

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM
KHOA CÔNG NGHỆ THÔNG TIN



LUẬN VĂN TỐT NGHIỆP
NGHIÊN CỨU ỨNG DỤNG OPENCV ĐỂ NHẬN
DẠNG VÀ HỖ TRỢ GIẢI TOÁN

Ngành : Công nghệ thông tin
Niên khóa : 2016 - 2020
Lớp : DH16DT
Sinh viên thực hiện : Trần Thanh Điền
Nguyễn Chí Phong

Tp. Hồ Chí Minh, ngày 17 tháng 09 năm 2020

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM
KHOA CÔNG NGHỆ THÔNG TIN



LUẬN VĂN TỐT NGHIỆP
NGHIÊN CỨU ÚNG DỤNG OPENCV ĐỂ NHẬN
DẠNG VÀ HỖ TRỢ GIẢI TOÁN

Giảng viên hướng dẫn:

TS. Nguyễn Văn Dũ

Sinh viên thực hiện:

Trần Thanh Điền - 16130326

Nguyễn Chí Phong - 16130514

Tp. Hồ Chí Minh, ngày 17 tháng 09 năm 2020

**Nghiên cứu ứng dụng OpenCV để nhận dạng
và hỗ trợ giải toán**

**Năm
2020**

CÔNG TRÌNH HOÀN TẤT TẠI TRƯỜNG ĐẠI HỌC NÔNG LÂM TP. HỒ CHÍ MINH

Cán bộ hướng dẫn: **T.S Nguyễn Văn Dũ**

Cán bộ phản biện: **Th.S Trần Quốc Việt**

Luận văn cử nhân được bảo vệ tại HỘI ĐỒNG CHẤM LUẬN VĂN CỬ NHÂN TRƯỜNG
ĐẠI HỌC NÔNG LÂM TP HCM ngày 17 tháng 09 năm 2020

Nhận xét của giáo viên hướng dẫn

Nhận xét của giáo viên phản biện

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

NHIỆM VỤ LUẬN VĂN CỦ NHÂN

Họ tên sinh viên: **TRẦN THANH ĐIỀN**

Phái: Nam

Ngày tháng năm sinh: 25/05/1998

Nơi sinh: Bà Rịa Vũng Tàu

Chuyên ngành: Công Nghệ Thông Tin

Ngành: Công Nghệ Thông Tin

Điện thoại liên lạc: 0357352255

Email: 16130326@st.hcmuaf.edu.vn

Họ tên sinh viên: **NGUYỄN CHÍ PHONG**

Phái: Nam

Ngày tháng năm sinh: 17/06/1998

Nơi sinh: Đồng Tháp

Chuyên ngành: Công Nghệ Thông Tin

Ngành: Công Nghệ Thông Tin

Điện thoại liên lạc: 0847773831

Email: 16130514@st.hcmuaf.edu.vn

I. TÊN ĐỀ TÀI: **Nghiên cứu ứng dụng opencv để nhận dạng và hỗ trợ giải toán**

II. NHIỆM VỤ VÀ NỘI DUNG

- Nhiệm vụ: Xây dựng ứng dụng di động Android cho phép nhận dạng và hỗ trợ giải toán thông qua ảnh chụp camera điện thoại.
- Nội dung:
 - + Nghiên cứu việc xử lý hình ảnh bằng thư viện OpenCV.
 - + Nghiên cứu Deep Learning cho quá trình xây dựng mô hình nhận dạng để phát hiện các ký hiệu, biểu thức toán học.

III. NGÀY GIAO NHIỆM VỤ: **02/02/2020**

IV. NGÀY HOÀN THÀNH NHIỆM VỤ: **31/08/2020**

V. HỌ VÀ TÊN CÁN BỘ HƯỚNG DẪN: **T.S Nguyễn Văn Dũ**

Ngày .../.../...

Ngày .../.../...

Ngày .../.../...

CÁN BỘ HƯỚNG DẪN

CÁN BỘ PHẢN BIỆN

KHOA CNTT

LỜI CẢM ƠN



Chúng em xin chân thành cảm ơn các thầy cô khoa Công nghệ thông tin trường Đại học Nông Lâm TP.Hồ Chí Minh, với những kiến thức quý báu và sự tận tâm, nhiệt huyết mà thầy cô đã truyền đạt cho chúng em trong suốt những năm đại học.

Chúng con xin gửi đến cha mẹ lời ghi ơn sâu sắc, những người đã sinh ra và dạy bảo chúng con trưởng thành như ngày hôm nay.

Đặc biệt, chúng em xin chân thành cảm ơn Thầy Nguyễn Văn Dũ đã tận tình hướng dẫn, chỉ bảo và giúp đỡ chúng em trong suốt quá trình thực hiện đề tài nghiên cứu này. Cảm ơn vì thầy đã truyền tải cho chúng em không chỉ về kiến thức chuyên ngành mà còn cả về cách sống, cách ứng xử, cách suy nghĩ giúp chúng em trưởng thành hơn.

Cũng xin cảm ơn tất cả những bạn bè đã chia sẻ kiến thức và tận tình giúp đỡ chúng tôi hoàn thành đề tài này.

Trong quá trình thực hiện đề tài nghiên cứu, mặc dù chúng em đã có những cố gắng nỗ lực thực hiện nhưng chúng em chắc chắn tránh được những sai sót nhất định. Kính mong sự thông cảm và tận tình chỉ bảo của quý Thầy Cô. Xin chân thành cảm ơn mọi người.

TP. HCM, ngày 17 tháng 09 năm 2020

NHÓM THỰC HIỆN LUẬN VĂN TỐT NGHIỆP

DANH MỤC CÁC HÌNH

Hình 1: Các bước cơ bản trong xử lý ảnh.....	6
Hình 2: Ảnh minh họa học sâu	12
Hình 3: Ảnh minh họa não bộ con người	31
Hình 4: Giới thiệu thư viện OpenCV	33
Hình 5: Giới thiệu thư viện Tensorflow	34
Hình 6: Giới thiệu ngôn ngữ Python	36
Hình 7: Giới thiệu thư viện cắt ảnh Crop Image	42
Hình 8: Giới thiệu thư viện vẽ đồ thị GraphLib	43
Hình 9: Giới thiệu ngôn ngữ đánh dấu LaTeX	44
Hình 10: Ví dụ đầu vào của ngôn ngữ LaTeX	46
Hình 11: Kết quả đầu ra của ví dụ	47
Hình 12: Giới thiệu thư viện MathView.....	48
Hình 13: Giới thiệu cơ sở dữ liệu SQLite.....	49
Hình 14: Sơ đồ của hệ thống.....	52
Hình 15: Lưu đồ giải thuật của hệ thống	54
Hình 16: Các ký tự dùng để huấn luyện mô hình nhận dạng	55
Hình 17: Kiến trúc mạng neuron dùng để huấn luyện	56
Hình 18: Các tham số của mô hình	58
Hình 19: Độ chính xác của mô hình.....	59
Hình 20: Độ mượt mà của mô hình	60
Hình 21: Giao diện màn hình chính của ứng dụng.....	61
Hình 22: Chức năng máy ảnh của ứng dụng	62
Hình 23: Chức năng đọc ảnh từ thư viện.....	63
Hình 24: Màn hình cắt ảnh biểu thức	64
Hình 25: Cắt vùng ảnh chứa biểu thức	65
Hình 26: Ảnh gốc chứa biểu thức	66
Hình 27: Ảnh sau khi khử nhiễu	66
Hình 28: Ảnh xám	66
Hình 29: Ảnh sau khi phân ngưỡng	67
Hình 30: Ảnh sau khi làm giãn nở	67
Hình 31: Ảnh chứa hình bao (contour) của các ký tự	67

Hình 32: Ảnh chứa các ký tự được bao đóng bởi các hình chữ nhật	68
Hình 33: Các ký tự sau khi được phân tách	68
Hình 34: Nhận dạng ký tự bằng Tensorflow Lite.....	69
Hình 35: Nhận dạng lũy thừa	70
Hình 36: Nhận dạng lũy thừa với mũ là biểu thức	71
Hình 37: Nhận dạng dấu bằng	72
Hình 38: Nhận dạng dấu chia.....	73
Hình 39: Nhận dạng phân số	74
Hình 40: Nhận dạng dấu chia dạng phân số	74
Hình 41: Nhận dạng căn bậc hai	75
Hình 42: Nhận dạng hệ phương trình	76
Hình 43: Ví dụ giải hệ phương trình	78
Hình 44: Ví dụ giải hệ phương trình	79
Hình 45: Ví dụ giải hệ phương trình	80
Hình 46: Ví dụ giải hệ phương trình	81
Hình 47: Ví dụ giải hệ phương trình	82
Hình 48: Minh họa đồ thị hàm số bậc nhất	83
Hình 49: Minh họa đồ thị hàm số bậc hai	84
Hình 50: Nhận dạng giọng nói	85
Hình 51: Giọng nói được gửi đến Google để xử lý	86
Hình 52: Giải biểu thức giọng nói	87
Hình 53: Màn hình lịch sử biểu thức đã lưu	88
Hình 54: Người dùng hoàn toàn có thể xóa biểu thức đã lưu	89
Hình 55: Xóa tất cả biểu thức	90
Hình 56: Màn hình hướng dẫn sử dụng	91
Hình 57: Màn hình hướng dẫn sử dụng	92
Hình 58: Màn hình hướng dẫn sử dụng	93
Hình 59: Màn hình hướng dẫn sử dụng	94
Hình 60: Màn hình thông tin ứng dụng	95

DANH MỤC CÁC BẢNG

	Trang
Bảng 1: Các lệnh định nghĩa dữ liệu của SQLite	50
Bảng 2: Các lệnh thao tác dữ liệu của SQLite	51
Bảng 3: Các lệnh truy vấn dữ liệu của SQLite	51
Bảng 4: Kết quả nhận dạng một số biểu thức	101

TÓM TẮT

Giải toán là một trong những vấn đề mà con người thường phải đối mặt hàng ngày, đặc biệt là đối với học sinh cấp 1, cấp 2, ngay cả những phép tính đơn giản như cộng, trừ, nhân, chia đôi khi cũng làm mất không ít thời gian. Năm bắt được tầm quan trọng đó, hiện nay đã có một số ứng dụng ra đời cho phép giải toán hiệu quả. Tuy nhiên, phần lớn trong số chúng vẫn còn hạn chế như phải nhập bài toán thủ công, đúng theo định dạng và phải có kết nối mạng. Để khắc phục điều này, chúng tôi đã phát triển một ứng dụng hoàn toàn offline cho phép giải toán nhanh chóng thông qua camera điện thoại. Ứng dụng cho phép giải các bài toán như cộng, trừ, nhân, chia, phương trình bậc 1, 2, 3. Trong bài báo này, chúng tôi sẽ trình bày các bước xử lý hình ảnh có chứa biểu thức thông qua thư viện xử lý ảnh OpenCV, phương pháp nhận dạng biểu thức toán học cũng như cách giải của chúng. Kết quả thử nghiệm và đánh giá ứng dụng cũng được đề cập trong tài liệu này.

ABSTRACT

Solving math is one of the problems that people often face every day, especially for elementary and middle school students, even simple calculations like addition, subtraction, multiplication, and division sometimes do. take a lot of time. Grasping that importance, nowadays, there are a number of applications that allow to solve problems effectively. However, most of them are still limited such as having to enter the problem manually, following the correct format and having a network connection. To overcome this, we have developed a completely offline application that allows quick math problems via phone camera. The application allows solving problems such as addition, subtraction, multiplication, division, quadratic equations, 3. In this paper, we will present the steps for processing images containing expressions through the library. OpenCV image processing, methods of recognizing mathematical expressions as well as their solutions. Test results and application evaluation are also mentioned in this document.

DANH SÁCH CHỮ VIẾT TẮT

Tù viết tắt	Tù tiếng Anh	Tù tiếng Việt
CNN	Convolutional Neural Network	Mạng neuron tích chập
DL	Deep Learning	Học sâu
ML	Machine Learning	Học máy

MỤC LỤC

Trang

LỜI CẢM ƠN	iv
DANH MỤC CÁC HÌNH	v
DANH MỤC CÁC BẢNG	vii
DANH SÁCH CHỮ VIẾT TẮT	ix
MỤC LỤC.....	x
CHƯƠNG 1. MỞ ĐẦU.....	1
1.1. Lý do chọn đề tài	1
1.2. Các công trình nghiên cứu liên quan	2
1.2.1. Trong nước	2
1.2.2. Ngoài nước	2
1.3. Mục tiêu của đề tài	3
1.4. Nội dung và phạm vi nghiên cứu	3
1.5. Sản phẩm của đề tài	4
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	5
2.1. Tổng quan về xử lý ảnh	5
2.1.1. Giới thiệu về xử lý ảnh	5
2.1.1.1. Phân thu nhận ảnh (Image Acquisition)	6
2.1.1.2. Tiền xử lý (Image Processing).....	7
2.1.1.3. Phân đoạn (Segmentation) hay phân vùng ảnh	7
2.1.1.4. Biểu diễn ảnh (Image Representation)	8
2.1.1.5. Nhận dạng và nội suy ảnh (Image Recognition and Interpretation)	8
2.1.1.6. Cơ sở tri thức (Knowledge Base)	8
2.1.2. Những vấn đề cơ bản trong hệ thống xử lý ảnh	9
2.1.2.1. Điểm ảnh (Picture Element).....	9
2.1.2.2. Độ phân giải của ảnh	9
2.1.2.3. Mức xám của ảnh	10
2.1.2.4. Định nghĩa ảnh số	11
2.1.3. Những vấn đề khác trong xử lý ảnh	11
2.1.3.1. Biến đổi ảnh (Image Transform)	11
2.1.3.2. Nén ảnh.....	11
2.2. Tìm hiểu về học sâu	11

2.2.1. Giới thiệu.....	11
2.2.1.1. Định nghĩa	11
2.2.1.2. Khái niệm cơ bản	14
2.2.2. Lịch sử	14
2.2.3. Các mạng Neuron nhân tạo	16
2.2.4. Kiến trúc	19
2.2.4.1. Các mạng neuron sâu	19
2.2.4.2. Mạng niềm tin sâu (Deep belief network).....	21
2.2.4.3. Mạng nơ ron tích chập (Convolutional neural networks)	21
2.2.4.4. Các mạng niềm tin sâu tích chập.....	22
2.2.4.5. Mạng neuron lưu trữ và truy xuất bộ nhớ lớn	22
2.2.4.6. Các mạng xếp chồng sâu	23
2.2.4.7. Mạng lập trình sâu (deep coding network)	24
2.2.4.8. Mạng bộ nhớ	24
2.2.5. Ứng dụng.....	26
2.2.5.1. Nhận dạng tiếng nói tự động.....	26
2.2.5.2. Nhận dạng hình ảnh	28
2.2.5.3. Xử lý ngôn ngữ tự nhiên	29
2.2.5.4. Khám phá dược phẩm và độc chất học.....	29
2.2.5.5. Quản lý quan hệ khách hàng (CRM)	30
2.2.5.6. Các hệ thống khuyến cáo (gợi ý)	30
2.2.5.7. Tin sinh học.....	30
2.2.6. Lý thuyết về bộ não con người	30
2.3. Thư viện xử lý ảnh OpenCV	32
2.4. Thư viện học máy Tensorflow	34
2.4.1. Thư viện Tensorflow là gì?	34
2.4.2. Đặc điểm nổi bật của Tensorflow	35
2.4.3. Ứng dụng của Tensorflow.....	36
2.5. Ngôn ngữ lập trình Python.....	36
2.5.1. Sơ lược về Python	36
2.5.2. Các đặc điểm của ngôn ngữ lập trình Python	38
2.5.3. Ứng dụng của Python	41
2.6. Thư viện cắt ảnh Crop Image	41
2.6.1. Tổng quan.....	41

2.6.2. Tính năng.....	42
2.7. Thư viện vẽ đồ thị GraphLib	43
2.7.1. Tổng quan.....	43
2.7.2. Đặc điểm.....	43
2.7.3. Hạn chế	44
2.8. Thư viện hiển thị biểu thức toán học MathView	44
2.8.1. Giới thiệu LaTeX	44
2.8.2. Thư viện MathView	47
2.9. Tìm hiểu hệ quản trị cơ sở dữ liệu SQLite	48
2.9.1. Giới thiệu SQLite	48
2.9.2. Ưu điểm của SQLite	49
2.9.3. Nhược điểm của SQLite	50
2.9.4. SQLite Commands	50
CHƯƠNG 3. TRIỂN KHAI THỰC HIỆN	52
3.1. Sơ đồ của hệ thống.....	52
3.2. Lưu đồ giải thuật của hệ thống	53
3.3. Xây dựng mô hình nhận dạng bằng Tensorflow	54
3.3.1. Giới thiệu data set	54
3.3.2. Xử lý data set	55
3.3.3. Mô hình CNN	55
3.3.4. Huấn luyện mô hình.....	58
3.4. Xây dựng ứng dụng Android	60
3.4.1. Màn hình chính	60
3.4.2. Chức năng chụp ảnh.....	61
3.4.3. Đọc ảnh từ thư viện ảnh	62
3.4.4. Cắt ảnh	63
3.5. Tiền xử lý ảnh sử dụng OpenCV	64
3.6. Tích hợp mô hình nhận dạng lên ứng dụng	68
3.6.1. Chuyển đổi mô hình Tensorflow thành mô hình Tensorflow Lite	69
3.6.2. Tích hợp mô hình lên ứng dụng	69
3.7. Phương pháp nhận dạng để cho ra biểu thức hoàn chỉnh	70
3.8. Giải và đưa ra từng bước giải	76
3.9. Bổ sung thêm các chức năng cho ứng dụng	82
3.9.1. Biểu diễn đồ thị phương trình, hệ phương trình	83

3.9.2. Tích hợp nhận dạng giọng nói	84
3.9.3. Lưu lịch sử biểu thức đã giải	87
3.9.4. Hướng dẫn sử dụng	90
3.9.5. Thông tin ứng dụng	94
CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC	96
4.1. Sản phẩm	96
4.2. Kết quả thực nghiệm.....	96
CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	102

CHƯƠNG 1. MỞ ĐẦU

1.1. Lý do chọn đề tài

Từ xa xưa, toán học là một ngành, một môn học vô cùng quan trọng đòi hỏi suy luận và trí thông minh cao. Nó chứa tất cả những gì thách thức đến bộ não của chúng ta. Học toán hay nghiên cứu toán học là vận dụng khả năng suy luận và trí óc thông minh của chúng ta.

Có thể nói rằng, môn toán là nền tảng cho tất cả các ngành khoa học tự nhiên khác. Toán học không chỉ là một môn học quan trọng trong nhà trường mà còn được ứng dụng rộng rãi trong cuộc sống hàng ngày.

Ngày nay, với sự phát triển không ngừng của khoa học công nghệ, đặc biệt là trong các lĩnh vực về trí tuệ nhân tạo (AI) đã giúp ích rất nhiều cho cuộc sống con người như: chấm công nhân viên bằng phương pháp nhận diện khuôn mặt, vân tay, bãi giữ xe thông minh,... Đặc biệt trong lĩnh vực giáo dục, một số trường học đã áp dụng trí tuệ nhân tạo để hỗ trợ công tác quản lý (điểm danh học sinh/sinh viên) và chấm điểm thi trắc nghiệm tự động.

Bên cạnh đó, một số ứng dụng giúp học sinh tự học các môn học có thể kể đến như: Duolingo (học tiếng anh), Chemical Reactions (học hóa học), Math Solve (học toán),... Trong số đó, Math Solve là phần mềm toán do chính người Việt tạo ra, nó có thể giải gần như tất cả các dạng toán dưới phổ thông và một phần toán cao cấp trên Đại học. Tuy nhiên, khi sử dụng ứng dụng này người dùng gặp khó khăn trong việc nhập liệu các biểu thức toán học. Math Solve đòi hỏi các biểu thức phải nhập đúng định dạng đã được quy định sẵn. Ngoài ra, việc giải một số phép tính rất đơn giản như phương trình bậc 2 trong trường số thực, nó cũng giải theo cách rất không tự nhiên, chứ không tính biệt thức delta như bình thường. Vì vậy, để góp phần khắc phục những hạn chế trên của Math Solve đồng thời hỗ trợ học sinh trong việc tự học môn toán tốt hơn, đề tài “Nghiên cứu ứng dụng OpenCV để nhận dạng và hỗ trợ giải toán” được hình thành.

Ứng dụng sẽ giúp giải toán thông qua camera điện thoại. Chỉ cần chụp lại bài toán, hệ thống sẽ quét để phát hiện bài toán thông qua ảnh chụp và sau đó xử lý để đưa ra từng bước giải cũng như kết quả cho người dùng. Hoặc người dùng có thể tự nhập nếu muốn. Ngoài ra, đối với các phương trình, ứng dụng còn có thể hiển thị được đồ thị hàm số của phương trình đó để người dùng có cái nhìn trực quan hơn. Một khía cạnh khác ứng dụng có thể giúp cho phụ huynh học sinh kiểm tra, đối chiếu được kết quả làm bài tập của con em mình một cách dễ dàng.

1.2. Các công trình nghiên cứu liên quan

1.2.1. Trong nước

Hiện nay trên thị trường trong nước đã có nhiều sản phẩm phần mềm hay các ứng dụng điện thoại cho phép hỗ trợ con người giải toán một cách nhanh chóng và tiện lợi. Tuy nhiên hầu hết các ứng dụng đó đòi hỏi chúng ta phải nhập biểu thức một cách thủ công và theo đúng định dạng, điều này dẫn tới sự bất tiện cho người dùng.

Ở Việt Nam, chúng tôi chưa tìm thấy đề tài về nhận dạng biểu thức toán học.

1.2.2. Ngoài nước

PhotoMath và PhotoSolver là hai ứng dụng giải toán thông qua camera điện thoại phổ biến hiện nay. Hai ứng dụng này đã có mặt trên nhiều nền tảng di động, giải quyết những bài toán về số học, số thập phân, phương trình đường thẳng và hàm lượng giác, giúp việc học toán trên điện thoại của học sinh, sinh viên được thuận tiện và dễ dàng hơn.

Tuy nhiên điểm hạn chế của PhotoMath là mỗi lần muốn nhận dạng biểu thức người dùng phải chụp ảnh, không thể đọc ảnh từ thư viện ảnh của điện thoại và do cơ chế chọn vùng ảnh chứa biểu thức trước rồi mới chụp nên người dùng không thể chụp nhiều biểu thức cùng một lúc. Đó cũng chính là nhược điểm của

PhotoSolver, ngoài ra nó còn một nhược điểm lớn hơn nữa là yêu cầu phải có kết nối mạng thì mới nhận dạng và giải được.

1.3. Mục tiêu của đề tài

- Nghiên cứu và sử dụng thư viện OpenCV cho việc xử lý hình ảnh ký tự toán học.
- Tìm hiểu về Tensorflow để xây dựng mô hình nhận dạng.
- Xây dựng ứng dụng trên thiết bị di động để nhận dạng và hỗ trợ giải toán.

1.4. Nội dung và phạm vi nghiên cứu

- **Đối tượng nghiên cứu:** Đề tài tập trung nghiên cứu xung quanh các đối tượng sau:
 - Nghiên cứu tổng quan về xử lý ảnh số, nhận dạng các ký hiệu và biểu thức toán học.
 - Nghiên cứu các phương pháp, thuật toán phục vụ cho việc phát hiện và nhận dạng biểu thức, phương trình trên hình ảnh.
 - Bộ thư viện nhận dạng ảnh OpenCV và các công cụ hỗ trợ như hiển thị đồ thị phương trình trên nền tảng di động (Android).
- **Phạm vi nghiên cứu:**
 - Nhận dạng các biểu thức, phương trình chữ in và viết tay.
 - Ứng dụng được xây dựng trên nền tảng di động, phát hiện và giải, sau đó đưa ra từng bước thực hiện.
 - Đôi với phương trình, sau khi có kết quả sẽ vẽ thêm đồ thị giúp người dùng có cái nhìn trực quan hơn.
 - Việc xử lý ảnh, nhận dạng bài toán phải thỏa mãn các điều kiện sau:
 - + Các ký hiệu, biểu thức toán học phải rõ ràng, không bị mờ hay mất chữ, số.

- + Chỉ hỗ trợ các bài toán với các biến x, y hoặc z; các phương trình bậc 1, 2, 3.
- + Góc ảnh: trực diện (frontal) hoặc góc nghiêng không đáng kể.
- + Ảnh có chất lượng cao và không bị che khuất.

➤ **Nội dung nghiên cứu:**

- Nghiên cứu các công nghệ xử lý và các thư viện hỗ trợ nhận dạng để phát hiện các ký hiệu, biểu thức toán học.
- Thu thập bộ dữ liệu liên quan đến ký hiệu toán học, tiền xử lý chúng sau đó trích xuất những đặc trưng trong hình ảnh để phục vụ cho việc nhận dạng.
- Nghiên cứu các framework để xây dựng ứng dụng di động.

1.5. Sản phẩm của đề tài

- Bộ tài liệu nghiên cứu công nghệ nhận dạng và cách xử lý dữ liệu hình ảnh.
- Ứng dụng trên nền tảng di động cho phép nhận dạng và hỗ trợ giải toán thông qua ảnh chụp camera.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về xử lý ảnh

2.1.1. Giới thiệu về xử lý ảnh

Xử lý ảnh là một lĩnh vực mang tính khoa học và công nghệ. Nó là một ngành khoa học mới mẻ so với nhiều ngành khoa học khác nhưng tốc độ phát triển của nó rất nhanh, kích thích các trung tâm nghiên cứu, ứng dụng, đặc biệt là máy tính chuyên dụng riêng cho nó.

Xử lý ảnh được đưa vào giảng dạy ở bậc đại học ở nước ta khoảng chục năm nay. Nó là môn học liên quan đến nhiều lĩnh vực và cần nhiều kiến thức cơ sở khác.

Đầu tiên phải kể đến Xử lý tín hiệu số là một môn học hết sức cơ bản cho xử lý tín hiệu chung, các khái niệm về tích chập, các biến đổi Fourier, biến đổi Laplace, các bộ lọc hữu hạn...

Thứ hai, các công cụ toán như Đại số tuyến tính, Xác suất, thống kê. Một số kiến thức cần thiết như Trí tuệ nhân tạo, Mạng nơ ron nhân tạo cũng được đề cập trong quá trình phân tích và nhận dạng ảnh.

Các phương pháp xử lý ảnh bắt đầu từ các ứng dụng chính: nâng cao chất lượng ảnh và phân tích ảnh. Ứng dụng đầu tiên được biết đến là nâng cao chất lượng ảnh báo được truyền qua cáp từ Luân đôn đến New York từ những năm 1920. Vấn đề nâng cao chất lượng ảnh có liên quan tới phân bố mức sáng và độ phân giải của ảnh. Việc nâng cao chất lượng ảnh được phát triển vào khoảng những năm 1955.

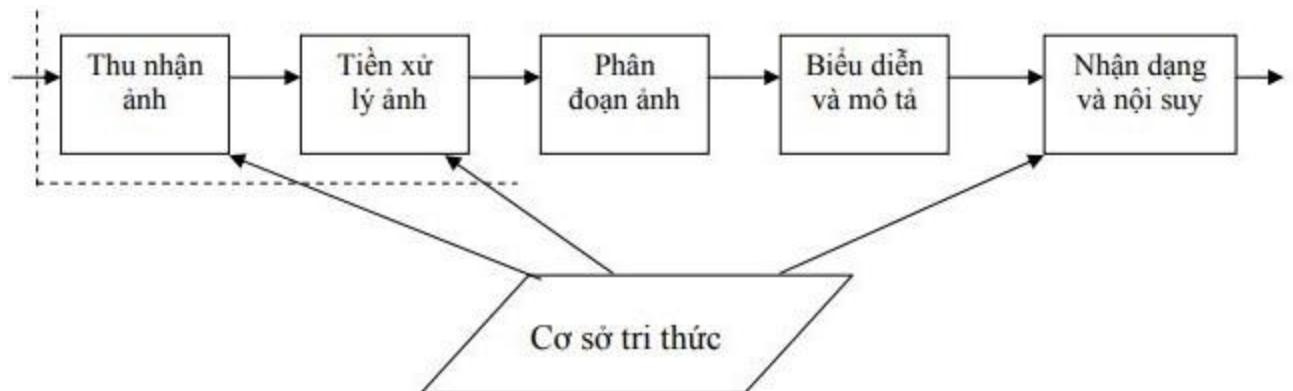
Điều này có thể giải thích được vì sau thế chiến thứ hai, máy tính phát triển nhanh tạo điều kiện cho quá trình xử lý ảnh số thuận lợi. Năm 1964, máy tính đã có khả năng xử lý và nâng cao chất lượng ảnh từ mặt trăng và vệ tinh Ranger 7 của Mỹ bao gồm: làm nổi đường biên, lưu ảnh.

Từ năm 1964 đến nay, các phương tiện xử lý, nâng cao chất lượng, nhận dạng ảnh phát triển không ngừng. Các phương pháp tri thức nhân tạo như mạng nơ ron nhân tạo, các thuật toán xử lý hiện đại và cải tiến, các công cụ nén ảnh ngày càng được áp dụng rộng rãi và thu nhiều kết quả khả quan.

Để dễ tưởng tượng, xét các bước cần thiết trong xử lý ảnh. Đầu tiên, ảnh tự nhiên từ thế giới ngoài được thu nhận qua các thiết bị thu (như Camera, máy chụp ảnh). Trước đây, ảnh thu qua Camera là các ảnh tương tự (loại Camera ống kiều CCIR).

Gần đây, với sự phát triển của công nghệ, ảnh màu hoặc đen trắng được lấy ra từ Camera, sau đó nó được chuyển trực tiếp thành ảnh số tạo thuận lợi cho xử lý tiếp theo. (Máy ảnh số hiện nay là một ví dụ gần gũi). Mặt khác, ảnh cũng có thể tiếp nhận từ vệ tinh; có thể quét từ ảnh chụp bằng máy quét ảnh.

Dưới đây mô tả các bước cơ bản trong xử lý ảnh.



Hình 1: Các bước cơ bản trong xử lý ảnh

Sơ đồ này bao gồm các thành phần sau:

2.1.1.1. Phần thu nhận ảnh (Image Acquisition)

Ảnh có thể nhận qua camera màu hoặc đen trắng. Thường ảnh nhận qua camera là ảnh tương tự (loại camera ống chuẩn CCIR với tần số 1/25, mỗi ảnh 25

dòng), cũng có loại camera đã số hoá (như loại CCD – Change Coupled Device) là loại photodiot tạo cường độ sáng tại mỗi điểm ảnh.

Camera thường dùng là loại quét dòng; ảnh tạo ra có dạng hai chiều. Chất lượng một ảnh thu nhận được phụ thuộc vào thiết bị thu, vào môi trường (ánh sáng, phong cảnh). Camera càng chất lượng sẽ thu được những bức ảnh càng sắc nét, giúp cho quá trình nhận dạng có độ chính xác cao hơn.

2.1.1.2. Tiề̂n xử lý (Image Processing)

Sau quá trình thu nhận, ảnh có thể bị nhiễu, mờ, độ tương phản thấp, kích thước quá rộng hoặc quá nhỏ nên cần đưa chúng vào bộ tiền xử lý để nâng cao chất lượng. Chức năng chính của bộ tiền xử lý là lọc nhiễu, tăng độ tương phản, ... để làm ảnh rõ, sắc nét hơn.

Tiề̂n xử lý ảnh là bước vô cùng quan trọng và không thể thiếu cho hầu hết mọi bài toán về thị giác máy tính (Computer Vision).

2.1.1.3. Phân đoạn (Segmentation) hay phân vùng ảnh

Phân vùng ảnh là tách một ảnh đầu vào thành các vùng thành phần để biểu diễn phân tích, nhận dạng ảnh. Ví dụ: để nhận dạng chữ (hoặc mã vạch) trên phong bì thư cho mục đích phân loại bưu phẩm, cần chia các câu, chữ về địa chỉ hoặc tên người thành các từ, các chữ, các số (hoặc các vạch) riêng biệt để nhận dạng.

Giai đoạn này phân tích ảnh thành những thành phần có cùng tính chất nào đó dựa theo biên hay các vùng liên thông. Tiêu chuẩn để xác định các vùng liên thông là cùng màu, cùng mức xám, ... Mục tiêu của phân đoạn ảnh là để có một miêu tả tổng hợp về nhiều phần tử khác nhau cấu tạo lên ảnh thô. Vì lượng thông tin chứa trong ảnh rất lớn, trong khi đa số các ứng dụng chúng ta chỉ cần trích một vài đặc trưng nào đó, do vậy cần có một quá trình để giảm lượng thông tin không lồ đó.

Đây là phần phức tạp khó khăn nhất trong xử lý ảnh và cũng dễ gây lỗi, làm mất độ chính xác của ảnh. Kết quả nhận dạng ảnh phụ thuộc rất nhiều vào công đoạn này.

2.1.1.4. Biểu diễn ảnh (Image Representation)

Đầu ra ảnh sau phân đoạn chứa các điểm ảnh của vùng ảnh (ảnh đã phân đoạn) cộng với mã liên kết với các vùng lân cận. Việc biến đổi các số liệu này thành dạng thích hợp là cần thiết cho xử lý tiếp theo bằng máy tính.

Việc chọn các tính chất để thể hiện ảnh gọi là trích chọn đặc trưng (Feature Selection) gắn với việc tách các đặc tính của ảnh dưới dạng các thông tin định lượng hoặc làm cơ sở để phân biệt lớp đối tượng này với đối tượng khác trong phạm vi ảnh nhận được. Ví dụ: trong nhận dạng ký tự trên phong bì thư, chúng ta miêu tả các đặc trưng của từng ký tự giúp phân biệt ký tự này với ký tự khác.

2.1.1.5. Nhận dạng và nội suy ảnh (Image Recognition and Interpretation)

Nhận dạng ảnh là quá trình xác định ảnh. Quá trình này thường thu được bằng cách so sánh với mẫu chuẩn đã được học (hoặc lưu) từ trước. Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng. Ví dụ: một loạt chữ số và nét gạch ngang trên phong bì thư có thể được nội suy thành mã điện thoại. Có nhiều cách phân loại ảnh khác nhau về ảnh. Theo lý thuyết về nhận dạng, các mô hình toán học về ảnh được phân theo hai loại nhận dạng ảnh cơ bản:

- Nhận dạng theo tham số.
- Nhận dạng theo cấu trúc.

Một số đối tượng nhận dạng khá phổ biến nay đang được áp dụng trong khoa học và công nghệ là: nhận dạng ký tự (chữ in, chữ viết tay, chữ ký điện tử), nhận dạng văn bản (Text), nhận dạng vân tay, nhận dạng mã vạch, nhận dạng mặt người...

2.1.1.6. Cơ sở tri thức (Knowledge Base)

Như đã nói ở trên, ảnh là một đối tượng khá phức tạp về đường nét, độ sáng tối, dung lượng điểm ảnh, môi trường để thu ảnh phong phú kéo theo nhiều.

Trong nhiều khâu xử lý và phân tích ảnh ngoài việc đơn giản hóa các phương pháp toán học đảm bảo tiện lợi cho xử lý, người ta mong muốn bắt chước quy trình tiếp nhận và xử lý ảnh theo cách của con người.

Trong các bước xử lý đó, nhiều khâu hiện nay đã xử lý theo các phương pháp trí tuệ con người.

2.1.2. Những vấn đề cơ bản trong hệ thống xử lý ảnh

2.1.2.1. Điểm ảnh (Picture Element)

Gốc của ảnh (ảnh tự nhiên) là ảnh liên tục về không gian và độ sáng. Để xử lý bằng máy tính (số), ảnh cần phải được số hóa.

Số hóa ảnh là sự biến đổi gần đúng một ảnh liên tục thành một tập điểm phù hợp với ảnh thật về vị trí (không gian) và độ sáng (mức xám).

Khoảng cách giữa các điểm ảnh đó được thiết lập sao cho mắt người không phân biệt được ranh giới giữa chúng. Mỗi một điểm như vậy gọi là điểm ảnh (PEL: Picture Element) hay gọi tắt là Pixel. Trong khuôn khổ ảnh hai chiều, mỗi pixel ứng với cặp tọa độ (x, y).

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

2.1.2.2. Độ phân giải của ảnh

Định nghĩa: Độ phân giải (Resolution) của ảnh là mật độ điểm ảnh được xác định trên một ảnh số được hiển thị.

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

Ví dụ: Độ phân giải của ảnh trên màn hình CGA (Color Graphic Adaptor) là một lưới điểm theo chiều ngang màn hình: 320 điểm chiều dọc * 200 điểm ảnh (320*200). Rõ ràng, cùng màn hình CGA 12" ta nhận thấy mịn hơn màn hình CGA 17" độ phân giải 320*200. Lý do: cùng một mật độ (độ phân giải) nhưng diện tích màn hình rộng hơn thì độ mịn (liên tục của các điểm) kém hơn.

2.1.2.3. Mức xám của ảnh

Một điểm ảnh (pixel) có hai đặc trưng cơ bản là vị trí (x, y) của điểm ảnh và độ xám của nó. Dưới đây chúng ta xem xét một số khái niệm và thuật ngữ thường dùng trong xử lý ảnh.

a) Định nghĩa: Mức xám của điểm ảnh là cường độ sáng của nó được gán bằng giá trị số tại điểm đó.

b) Các thang giá trị mức xám thông thường: 16, 32, 64, 128, 256 (Mức 256 là mức phổ dụng. Lý do: từ kỹ thuật máy tính dùng 1 byte (8 bit) để biểu diễn mức xám: Mức xám dùng 1 byte biểu diễn: $2^8=256$ mức, tức là từ 0 đến 255).

c) Ảnh đen trắng: là ảnh có hai màu đen, trắng (không chứa màu khác) với mức xám ở các điểm ảnh có thể khác nhau.

d) Ảnh nhị phân: ảnh chỉ có 2 mức đen trắng phân biệt tức dùng 1 bit mô tả 2 mức khác nhau. Nói cách khác: mỗi điểm ảnh của ảnh nhị phân chỉ có thể là 0 hoặc 1.

e) Ảnh màu: trong khuôn khổ lý thuyết ba màu (Red, Blue, Green) để tạo nên thế giới màu, người ta thường dùng 3 byte để mô tả mức màu, khi đó các giá trị màu: $2^8 * 2^8 * 2^8 \approx 16,7$ triệu màu.

2.1.2.4. Định nghĩa ảnh số

Ảnh số là tập hợp các điểm ảnh với mức xám phù hợp dùng để mô tả ảnh gần với ảnh thật.

2.1.3. Những vấn đề khác trong xử lý ảnh

2.1.3.1. Biến đổi ảnh (Image Transform)

Trong xử lý ảnh, do số điểm ảnh lớn, các tính toán nhiều (độ phức tạp tính toán cao) đòi hỏi dung lượng bộ nhớ lớn, thời gian tính toán lâu. Các phương pháp khoa học kinh điển áp dụng cho xử lý ảnh hầu hết khó khả thi.

Người ta sử dụng các phép toán tương đương hoặc biến đổi sang miền xử lý khác để dễ tính toán. Sau khi xử lý dễ dàng hơn được thực hiện, dùng biến đổi ngược để đưa về miền xác định ban đầu, các biến đổi thường gặp trong xử lý ảnh gồm:

- Biến đổi Fourier, Cosin, Sin
- Biến đổi (mô tả) ảnh bằng tích chập, tích Kronecker (theo xử lý số tín hiệu)
- Các biến đổi khác như KL (Karhumen Loeve), Hadamard.

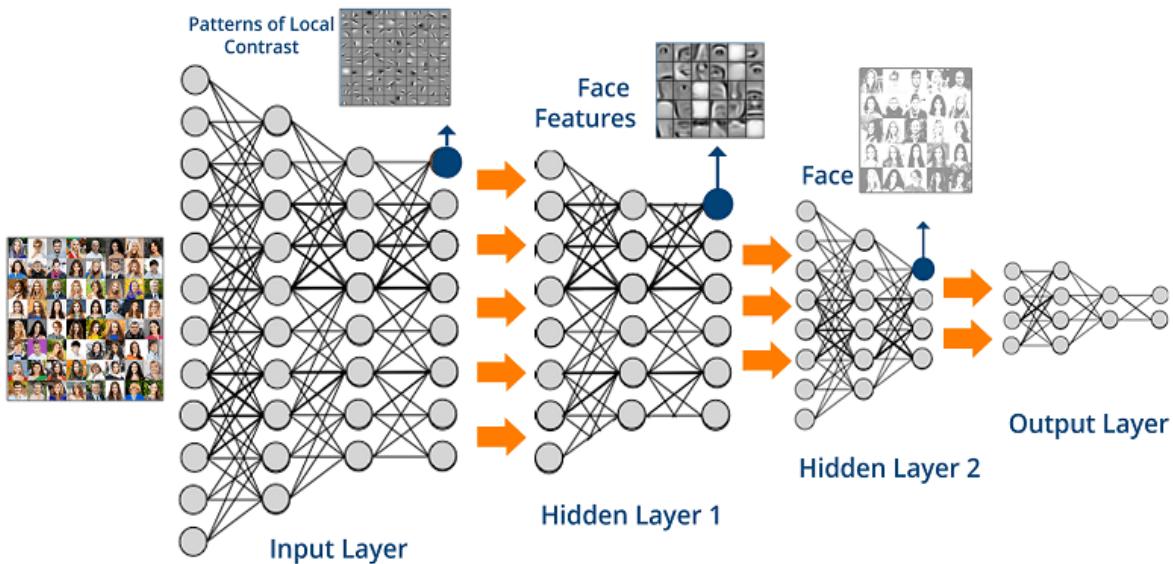
2.1.3.2. Nén ảnh

Ảnh cho dù ở dạng nào đi nữa thì vẫn chiếm không gian nhớ rất lớn. Khi mô tả ảnh người ta đã đưa kỹ thuật nén ảnh vào. Các giai đoạn nén ảnh có thể chia ra thê hệ 1, thê hệ 2. Hiện nay, các chuẩn MPEG được dùng với ảnh đang phát huy hiệu quả.

2.2. Tìm hiểu về học sâu

2.2.1. Giới thiệu

2.2.1.1. Định nghĩa



Hình 2: *Ảnh minh họa học sâu*

Học sâu (Deep Learning): là một chi của ngành máy học dựa trên một tập hợp các thuật toán để có gắng để mô hình dữ liệu trừu tượng hóa ở mức cao bằng cách sử dụng nhiều lớp xử lý với cấu trúc phức tạp, hoặc bằng cách khác bao gồm nhiều biến đổi phi tuyến

Học sâu là một phần của một họ các phương pháp học máy rộng hơn dựa trên đại diện học của dữ liệu. Một quan sát (ví dụ như, một hình ảnh) có thể được biểu diễn bằng nhiều cách như một vector của các giá trị cường độ cho mỗi điểm ảnh, hoặc một cách trừu tượng hơn như là một tập hợp các cạnh, các khu vực hình dạng cụ thể, ... Một trong những hứa hẹn của học sâu là thay thế các tính năng thủ công bằng các thuật toán hiệu quả đối với học không có giám sát hoặc nửa giám sát và tính năng phân cấp.

Có một số cách để mô tả học sâu. Học sâu là một lớp của các thuật toán máy học mà:

- Sử dụng một tầng (cascade) nhiều lớp các đơn vị xử lý phi tuyến để trích tách đặc điểm và chuyển đổi. Mỗi lớp kế tiếp dùng đầu ra từ lớp trước làm đầu vào. Các thuật toán này có thể được giám sát hoặc không cần giám sát

và các ứng dụng bao gồm các mô hình phân tích (không có giám sát) và phân loại (giám sát).

- Dựa trên học (không có giám sát) của nhiều cấp các đặc điểm hoặc đại diện của dữ liệu. Các tính năng cao cấp bắt nguồn từ các tính năng thấp cấp hơn để tạo thành một đại diện thứ bậc.
- Là một phần của lĩnh vực máy học rộng lớn hơn về việc học đại diện dữ liệu.
- Học nhiều cấp độ đại diện tương ứng với các mức độ trừu tượng khác nhau; các mức độ hình thành một hệ thống phân cấp của các khái niệm.

Các định nghĩa này có điểm chung là nhiều lớp các đơn vị xử lý phi tuyến và học có giám sát hoặc không có giám sát của biểu diễn đặc tính ở mỗi lớp, với các lớp hình thành một hệ thống các tính năng phân cấp từ thấp đến cao cấp. Các thành phần của một lớp của đơn vị xử lý phi tuyến sử dụng một thuật toán học sâu tùy theo vấn đề cần được giải quyết. Các lớp được sử dụng trong học sâu bao gồm các lớp ẩn của một mạng neuron nhân tạo và tập các công thức mệnh đề phức tạp. Chúng cũng có thể bao gồm các biến tiềm ẩn được tổ chức thành các lớp chọn lọc trong các mô hình thế sinh (có khả năng sinh ra) sâu như các nút trong Deep Belief Networks và Deep Boltzmann Machines.

Các thuật toán học sâu tương phản với các thuật toán học nông bởi số biến đổi được tham số hóa một tín hiệu gấp phải khi nó lan truyền từ các lớp đầu vào đến lớp đầu ra, nơi một biến đổi được tham số hóa là một đơn vị xử lý có các thông số có thể huấn luyện được, chẳng hạn như trọng số và ngưỡng. Một chuỗi các biến đổi từ đầu vào đến đầu ra là một đường gán kế thừa (CAP- credit assignment path). CAP mô tả các kết nối quan hệ nhân quả tiềm năng giữa đầu vào và đầu ra và có thể thay đổi chiều dài. Đổi với một mạng neuron nuôi tiến (feedforward), độ sâu của CAP, và do đó độ sâu của mạng đó, là số lượng các lớp ẩn cộng 1 (lớp đầu ra cũng là tham số hóa). Đổi với mạng neuron tái phát, trong đó một tín hiệu có thể truyền thông qua một lớp nhiều hơn một lần, CAP có khả năng không bị giới hạn

chiều dài. Không có sự thống nhất chung về ngưỡng của độ sâu chia học nông với học sâu, nhưng hầu hết các nhà nghiên cứu trong lĩnh vực đồng ý rằng học sâu có nhiều lớp phi tuyến ($CAP > 2$) và coi $CAP > 10$ là học rất sâu.

2.2.1.2. Khái niệm cơ bản

Các thuật toán học sâu dựa trên các đại diện phân phôi. Giả định tiềm ẩn đằng sau các đại diện phân phôi là các dữ liệu được quan sát là được tạo ra bởi sự tương tác của các yếu tố được tổ chức theo lớp. Học sâu thêm giả định rằng các lớp của các yếu tố này tương ứng với các mức độ trừu tượng hay theo thành phần. Các con số khác nhau của các lớp và kích thước của lớp có thể được sử dụng để quy định các lượng trừu tượng khác.

Học sâu khai thác ý tưởng thứ bậc các yếu tố giải thích này ở cấp cao hơn, những khái niệm trừu tượng hơn được học từ các cấp độ thấp hơn. Những kiến trúc này thường được xây dựng với một phương pháp lớp chồng lớp tham lam. Học sâu giúp để tháo gỡ những khái niệm trừu tượng này và chọn ra những đặc điểm cần thiết cho việc học.

Đối với các nhiệm vụ học có giám sát, các phương pháp học sâu sẽ tránh kỹ thuật đặc điểm (feature engineering), bằng cách dịch các dữ liệu vào các đại diện trung gian nhỏ gọn giống như các thành phần chính, và lấy được các cấu trúc lớp mà loại bỏ sự thừa thãi trong đại diện.

Rất nhiều các thuật toán học sâu được áp dụng cho các nhiệm vụ học không có giám sát. Đây là một lợi ích quan trọng bởi vì dữ liệu không dán nhãn (chưa phân loại) thường phong phú hơn các dữ liệu dán nhãn. Một ví dụ của một cấu trúc sâu có thể được đào tạo theo cách không có giám sát là một mạng lưới tin sâu (deep belief network).

2.2.2. Lịch sử

Các kiến trúc học sâu, đặc biệt là những kiến trúc được xây dựng từ mạng neuron nhân tạo (ANN), đã từng thống trị ít nhất là tới Neocognitron được giới

thiệu bởi Masahiko Fukushima vào năm 1980. Chính các ANN lại thông trị thậm chí lâu hơn nữa. Thách thức là làm thế nào để đào tạo mạng lưới này với nhiều lớp. Năm 1989, Yann Le Cun và các cộng sự đã có thể áp dụng các thuật toán truyền ngược tiêu chuẩn, khoảng từ năm 1974, đối với một mạng neuron sâu với mục đích nhận dạng chữ viết tay mã ZIP trong các bức thư. Mặc dù sự thành công trong việc áp dụng thuật toán này, thời gian để đào tạo mạng dựa trên số liệu này mất khoảng 3 ngày, làm cho việc sử dụng nó vào các mục đích bình thường trở nên không thực tế. Năm 1995, Brendan Frey đã chứng minh rằng có thể đào tạo một mạng nơ ron bao gồm đầy đủ sáu lớp kết nối và vài trăm đơn vị ẩn bằng cách sử dụng thuật toán đánh thức giấc ngủ, nó được hợp tác phát triển với Peter Dayan và Geoffrey Hinton. Tuy nhiên, việc huấn luyện phải mất hai ngày.

Trong năm 1991 những mạng neuron như vậy được sử dụng để nhận diện chữ số viết tay 2-D cách ly, nhận dạng đối tượng 3-D được thực hiện bằng cách kết hợp các hình ảnh 2-D với một mô hình đối tượng 3-D thủ công. Juyang Weng và các cộng sự đề xuất rằng một bộ não người không sử dụng một mô hình đối tượng 3-D nguyên khôi, và vào năm 1992, họ xuất bản Cresceptron, một phương pháp để thực hiện nhận dạng đối tượng 3-D trực tiếp từ các hậu trường lộn xộn. Cresceptron là một ghép tầng của các lớp tương tự như Neocognitron. Nhưng trong khi Neocognitron yêu cầu một lập trình viên con người can thiệp, Cresceptron sẽ tự động học được một số đặc điểm không có giám sát trong mỗi lớp, nơi mà mỗi đặc điểm được đại diện bởi một nhân tích chập. Cresceptron cũng phân đoạn từng đối tượng học được từ một cảnh nền lộn xộn thông qua việc phân tích ngược mạng đó. Thăm dò max, bây giờ thường được thông qua bởi các mạng neuron sâu (ví dụ: các kiểm tra ImageNet), lần đầu tiên sử dụng trong Cresceptron để giảm độ phân giải vị trí bởi của một hệ số (2×2) đến 1 thông qua việc ghép tầng tổng quát hóa tốt hơn. Mặc dù có những lợi thế như thế, các mô hình đơn giản hơn sử dụng nhiệm vụ cụ thể có đặc điểm thủ công như bộ Gabor và các máy hỗ trợ vector (SVM-support vector machines) đã là lựa chọn phổ biến trong thập niên 1990 và thập niên 2000, bởi vì chi phí tính toán bởi các ANN và vì thiếu sự hiểu biết về cách thức bộ não tự quản các kết nối mạng sinh học của nó.

Trong lịch sử lâu dài của nhận dạng giọng nói, cả học nông và học sâu (ví dụ, các mạng tái phát) của mạng neuron nhân tạo đã được khám phá trong nhiều năm. Nhưng những phương pháp này không bao giờ thắng được công nghệ mô hình hỗn hợp/mô hình Markov ẩn Gaussian (GMM-HMM) thủ công-nội bộ dựa trên các mô hình thể sinh của việc huấn luyện nhận dạng giọng nói một cách rõ ràng. Một số khó khăn chính đã được phân tích một cách có phương pháp, bao gồm giảm bớt gradient và cấu trúc tương quan thời gian yếu và trong các mô hình tiên đoán neuron. Những khó khăn bổ sung đó là thiếu dữ liệu huấn luyện lớn và khả năng tính toán yếu trong thời gian ban đầu. Vì vậy, hầu hết nhà nghiên cứu nhận dạng giọng nói đã hiểu rõ các rào cản như vậy đã chuyển ra khỏi các mạng nơ ron để theo đuổi mô hình thể sinh, cho đến khi một sự hồi sinh gần đây của học sâu đã vượt qua tất cả những khó khăn này.

2.2.3. Các mạng Neuron nhân tạo

Một trong số các phương pháp học sâu thành công nhất là mạng neuron nhân tạo. Mạng neuron nhân tạo được lấy cảm hứng từ các mô hình sinh học năm 1959 được đề xuất bởi người đoạt giải Nobel David H. Hubel & Torsten Wiesel, 2 người đã tìm thấy hai loại tế bào trong vỏ não thị giác chính: các tế bào đơn giản và các tế bào phức tạp. Nhiều mạng neuron nhân tạo có thể được xem như là các mô hình ghép tầng của các tế bào lấy cảm hứng từ những quan sát sinh học.

Neocognitron của Fukushima giới thiệu các mạng neuron tích chập được đào tạo một phần bởi học không có giám sát với các đặc điểm được con người hướng dẫn trong mặt phẳng neuron. Yann LeCun 1989 áp dụng truyền ngược có giám sát cho các kiến trúc như vậy. Năm 1992 công bố các mạng neuron tích chập Cresceptron để nhận dạng các đối tượng 3-D từ các hình ảnh có hậu trường lộn xộn và phân khúc của các đối tượng từ hình ảnh đó.

Một nhu cầu rõ ràng để nhận dạng các đối tượng 3-D nói chung là ít nhất là thay đổi tính bất biến và khả năng chịu biến dạng. Thăm dò Max (Max-pooling) xuất hiện lần đầu tiên được đề xuất bởi Cresceptron để kích hoạt mạng để chịu

được sự biến dạng từ nhỏ đến lớn theo một cách phân cấp, trong khi sử dụng tích chập. Thăm dò mã đã hoạt động tốt, nhưng không đảm bảo, dịch chuyển bất định ở mức điểm ảnh.

Với sự ra đời của thuật toán truyền ngược được khám phá ra một cách độc lập bởi nhiều nhóm trong thập niên 1970 và 1980, nhiều nhà nghiên cứu đã có gắng để đào tạo các mạng neuron nhân tạo sâu có giám sát từ đầu, ban đầu với rất ít thành công. Luận văn tốt nghiệp cao đẳng của Sepp Hochreiter năm 1991 chính thức xác định lý do cho sự thất bại này là vấn đề biến mất gradient, ảnh hưởng đến các mạng nuôi tiến nhiều lớp và các mạng neuron hồi qui. Các mạng tái phát (hồi qui) được huấn luyện bằng cách trải chúng ra vào các mạng nuôi tiến rất sâu, nơi một lớp mới được tạo ra cho mỗi bước thời gian của một chuỗi đầu vào được xử lý bởi mạng này. Khi các sai số truyền từ lớp này sang lớp khác, chúng co lại theo cấp số nhân với số lượng lớp, ngăn cản điều chỉnh trọng số nơ ron, dựa trên những sai số này.

Để khắc phục vấn đề này, một số phương pháp đã được đề xuất. Một là thứ bậc đa cấp của mạng của Jürgen Schmidhuber (1992), cấp độ một được đào tạo trước tại một thời điểm bởi học không có giám sát, điều chỉnh bởi truyền ngược. Ở đây, mỗi cấp học một đại diện bị nén của các quan sát được đưa đến cấp độ tiếp theo.

Phương pháp khác là mạng bộ nhớ dài ngắn hạn (LSTM) của Hochreiter & Schmidhuber (1997). Trong năm 2009, các mạng LSTM đa chiều sâu đã chiến thắng ba cuộc thi ICDAR năm 2009 trong nhận dạng chữ viết tay, mà không có bất kỳ kiến thức sẵn có về ba ngôn ngữ để được học.

Sven Behnke vào năm 2003 dựa chỉ vào các dấu hiệu của gradient (Rprop) khi đào tạo Kim tự tháp Trùu tượng Nơ ron của mình để giải bài toán giống như tái tạo hình ảnh và định vị khuôn mặt.

Các phương pháp khác cũng sử dụng đào tạo trước không có giám sát để tạo ra một mạng nơ ron, khiến nó lần đầu tiên học được bộ dò đặc điểm nói chung là hữu ích. Sau đó mạng này được đào tạo tiếp tục bằng cách truyền ngược có giám sát để phân loại dữ liệu có dán nhãn. Mô hình sâu này của Hinton và các cộng sự (2006) liên quan đến việc học phân phối của một đại diện cao cấp bằng cách sử dụng các lớp kế tiếp của các biến tiềm ẩn nhị phân hoặc giá trị thực. nó sử dụng một máy Boltzmann hạn chế (Smolensky, 1986) để mô hình hóa mỗi lớp mới của các đặc điểm cao cấp hơn. Mỗi lớp mới đảm bảo một sự tăng trưởng trong biên thấp của kiểm tra tỷ lệ giống của dữ liệu, do đó tăng cường cho mô hình, nếu được huấn luyện đúng cách. Một khi đã đủ nhiều lớp đã được học, kiến trúc sâu có thể được sử dụng như là một mô hình thế sinh bằng cách tái tạo dữ liệu khi lấy mẫu xuống mô hình đó (một "sự vượt qua tổ tiên") từ các kích hoạt tính năng cấp định. Hinton báo cáo rằng các mô hình của mình là trích xuất các đặc điểm hiệu quả tính theo chiều cao, cấu trúc dữ liệu.

Nhóm Google Brain do Andrew Ng và Jeff Dean đã tạo ra một mạng nơ ron học cách để nhận dạng được những khái niệm cao cấp hơn, chẳng hạn như con mèo, chỉ từ xem những hình ảnh không được dán nhãn từ các video trên YouTube.

Tính đến năm 2011, tiến bộ trong các mạng nuôi tiên học sâu đã thay thế các lớp tích chập và các lớp thăm dò tối da (max-pooling), đứng đầu bởi một số lớp có đầy đủ kết nối hoặc kết nối từng phần sau bởi một lớp phân loại cuối cùng. Việc huấn luyện thường được thực hiện mà không có bất kỳ đào tạo trước không có giám sát nào. Từ năm 2011, các thực thi dựa trên GPU của hướng tiếp cận này đã thắng nhiều cuộc thi nhận dạng hình mẫu, bao gồm cuộc thi IJCNN 2011 Traffic Sign Recognition Competition, ISBI 2012 Segmentation of neuronal structures in EM stacks challenge, và các cuộc thi khác.

Các phương pháp học sâu có giám sát như vậy cũng đã là bộ nhận dạng mô hình nhân tạo đầu tiên đạt được hiệu suất có thể cạnh tranh lại được với con người trong những công việc nhất định.

Để vượt qua những rào cản của AI được đại diện bằng học sâu, cần phải vượt qua các kiến trúc học sâu, bởi vì bộ não sinh học sử dụng cả mạch học nồng và học sâu theo báo cáo của ngành giải phẫu não bộ chỉ ra một loạt các tính bất biến. Weng lập luận rằng não tự kết nối chủ yếu theo các thông kê tín hiệu và do đó, một phân tầng nối tiếp không thể bắt tất cả các vật phụ thuộc thống kê chủ yếu. Các ANN đã có thể đảm bảo sự thay đổi bất biến để đối phó với các đối tượng tự nhiên lớn và nhỏ trong hậu trường có sự xáo trộn lớn, chỉ khi các bất định mở rộng vượt ra ngoài sự thay đổi, tới tất cả các khái niệm ANN đã học được, chẳng hạn như vị trí, loại (nhãn lớp đối tượng), quy mô, ánh sáng. Điều này được thực hiện trong các Mạng Phát triển (Development Network) có biểu hiện là Where-What Networks, WWN-1 (2008) cho đến WWN-7 (2013).

2.2.4. Kiến trúc

Có một lượng rất lớn các biến thể của kiến trúc sâu. Hầu hết chúng là nhánh sinh ra từ một số kiến trúc cha ban đầu. Không phải là luôn luôn có thể so sánh hiệu suất của nhiều kiến trúc cùng với nhau, vì chúng không phải là tất cả đánh giá trên cùng một tập dữ liệu. Học sâu là một lĩnh vực phát triển nhanh, và các kiến trúc, biến thể, hoặc các thuật toán mới xuất hiện mỗi vài tuần.

2.2.4.1. Các mạng neuron sâu

Mạng neuron sâu (DNN-Deep Neural Network) là một mạng neuron nhân tạo (ANN) với nhiều đơn vị lớp ẩn giữa lớp đầu vào và đầu ra. Tương tự như các ANN nồng, các DNN nồng có thể mô hình hóa mối quan hệ phi tuyến phức tạp. Các kiến trúc DNN, ví dụ như để phát hiện và phân tích đối tượng tạo ra các mô hình hỗn hợp trong đó đối tượng này được thể hiện như một thành phần được xếp lớp của các hình ảnh nguyên thủy. Các lớp phụ cho phép các thành phần của các đặc điểm từ các lớp thấp hơn, đem lại tiềm năng của mô hình hóa dữ liệu phức tạp với các đơn vị ít hơn so với một mạng lưới nồng thực hiện tương tự như vậy.

Các DNN thường được thiết kế như các mạng nuôi tiến, nhưng nghiên cứu gần đây đã áp dụng thành công kiến trúc học sâu đối với các mạng nơ ron tái phát

cho các ứng dụng chẳng hạn như mô hình hóa ngôn ngữ. Các mạng neuron sâu tích chập (CNN) được sử dụng trong thị giác máy tính nơi thành công của chúng đã được ghi nhận. Gần đây hơn, các CNN đã được áp dụng để mô hình hóa âm thanh cho nhận dạng giọng nói tự động (ASR), nơi chúng đã cho thấy sự thành công trong các mô hình trước đó.

Các vấn đề với các mạng neuron sâu:

Như với các ANN, nhiều vấn đề có thể nảy sinh với các DNN nếu chúng được huấn luyện thô sơ. Hai vấn đề phổ biến là overfitting (nhiều hoặc sai số ngẫu nhiên) và thời gian tính toán.

Các DNN có thiên hướng overfitting vì được thêm các lớp trừu tượng, mà cho phép chúng thực hiện mô hình hóa phụ thuộc hiếm hoi vào dữ liệu huấn luyện. Các phương pháp regularization (quy tắc hóa) như phân rã trọng số hoặc sparsity (rãi) có thể được áp dụng trong quá trình huấn luyện để giúp chống lại overfitting. Một phương pháp regularization gần đây được áp dụng cho các DNN là dropout regularization. Trong dropout, một số lượng đơn vị được bỏ qua ngẫu nhiên từ các lớp ẩn trong quá trình đào tạo. Điều này giúp phá vỡ các phụ thuộc hiếm hoi có thể xảy ra trong dữ liệu đào tạo.

Phương pháp chủ đạo cho việc huấn luyện các cấu trúc là sửa lỗi huấn luyện (chẳng hạn như truyền ngược với gradient descent) do dễ thực hiện và xu hướng hội tụ tốt hơn local optima (tối ưu cục bộ) hơn so với các phương pháp huấn luyện khác. Tuy nhiên, những phương pháp này có thể tốn công tính toán hơn, đặc biệt là cho các DNN. Có rất nhiều tham số huấn luyện để được xem xét với một DNN, chẳng hạn như kích thước (số lượng lớp và số lượng đơn vị trên mỗi lớp), tốc độ học và trọng số ban đầu. Quét thông qua không gian tham số cho các thông số tối ưu có thể không khả thi do chi phí trong thời gian và tài nguyên tính toán. Nhiều 'mẹo vặt' chẳng hạn như bằng cách sử dụng mini-batching (tính toán gradient trên nhiều ví dụ huấn luyện khác nhau cùng một lúc chứ không phải là từng ví dụ một) đã được chỉ ra để tăng tốc độ tính toán. Lượng xử lý lớn thông qua GPU đã tăng

tốc đáng kể trong việc huấn luyện, do tính toán ma trận và vector rất thích hợp với các GPU. Lựa chọn thay thế triệt để cho truyền ngược là Extreme Learning Machines (Siêu máy học, các mạng "No-prop", huấn luyện không cần truy ngược, các mạng "không trọng số", và mạng nơron không kết nối (non-connectionist neural network) đang thu hút được sự chú ý.

2.2.4.2. Mạng niềm tin sâu (Deep belief network)

Một mạng niềm tin sâu (DBN) là một mô hình xác suất thế sinh, tạo thành bởi nhiều đơn vị ẩn nhiều lớp. Nó có thể được coi là một hàm hợp các mô-đun học đơn giản tạo thành mỗi lớp.

Một DBN có thể được sử dụng để huấn luyện trước khả sinh một DNN bằng cách sử dụng các trọng số DBN học như các trọng số DNN ban đầu. Các thuật toán truyền ngược hoặc suy xét khác sau đó có thể được áp dụng để điều chỉnh những trọng số này. Điều này đặc biệt hữu ích khi dữ liệu đào tạo giới hạn là có sẵn, vì các trọng số khởi tạo nghèo nàn có thể cản trở đáng kể hiệu suất của mô hình được học. Các trọng số đào tạo trước này là một vùng không gian trọng số là gần gũi hơn với trọng số tối ưu hơn là các trọng số ban đầu được chọn ngẫu nhiên. Điều này cho phép cả mô hình hóa được cải thiện và hội tụ tinh chỉnh pha nhanh hơn.

Khi một RBM được huấn luyện, RBM khác là "xếp chồng" trên nó, đưa đầu vào của nó từ cuối lớp đã được huấn luyện. Lớp hiện mới này được khởi tạo với một vector hiện, và các giá trị cho các đơn vị trong các lớp đã được huấn luyện phân công bằng cách sử dụng trọng số hiện tại và các độ lệch. RBM mới này sau đó lại được huấn luyện với chu trình như trên. Toàn bộ quá trình này được lặp lại cho đến khi một số tiêu chí mong muốn chặn lại được đáp ứng.

Mặc dù xấp xỉ của CD để tối đa khả năng là rất thô (CD đã được chỉ ra là theo gradient của bất kỳ hàm nào), nó đã được kinh nghiệm chỉ ra là có hiệu quả trong huấn luyện các kiến trúc sâu.

2.2.4.3. Mạng nơ ron tích chập (Convolutional neural networks)

Một CNN gồm có một hoặc nhiều hơn các lớp tích chập với các lớp đầy đủ kết nối (đáp ứng phù hợp với những mạng neuron nhân tạo tiêu biểu) trên đỉnh. Nó cũng sử dụng trọng số gắn liền và các lớp thăm dò. Kiến trúc này cho phép các CNN tận dụng lợi thế của cấu trúc 2D của dữ liệu đầu vào. So với những kiến trúc sâu khác, mạng neuron tích chập đang bắt đầu thể hiện kết quả vượt trội trong các ứng dụng hình ảnh và giọng nói. Chúng cũng có thể được huấn luyện với tiêu chuẩn truyền ngược. CNN dễ dàng được đào tạo hơn các mạng nơ ron sâu nuôi tiên thông thường khác, và có ít thông số ước tính hơn, khiến cho chúng trở thành một kiến trúc rất hấp dẫn để sử dụng. Các ví dụ về ứng dụng trong Thị Giác máy tính bao gồm DeepDream.

2.2.4.4. Các mạng niềm tin sâu tích chập

Sử dụng mạng niềm tin sâu (CDBN) là một thành tựu gần đây của học sâu. Các CDBN có cấu trúc rất giống với một mạng neuron tích chập và được huấn luyện tương tự như các mạng niềm tin sâu. Vì vậy, chúng khai thác cấu trúc 2D của hình ảnh, giống như CNN làm, và làm cho việc sử dụng đào tạo trước giống như mạng niềm tin sâu. Chúng quy định một cấu trúc chung mà có thể được sử dụng trong nhiều tác vụ xử lý hình ảnh và tín hiệu. Gần đây, nhiều kết quả benchmark (tiêu chuẩn) dựa trên tập dữ liệu hình ảnh chuẩn như CIFAR đã được thu được kết quả bằng cách sử dụng CDBN.

2.2.4.5. Mạng neuron lưu trữ và truy xuất bộ nhớ lớn

Mạng nơ ron lưu trữ và truy xuất bộ nhớ lớn (LAMSTAR) là các mạng nơ ron học sâu nhanh gồm nhiều lớp mà có thể sử dụng đồng thời nhiều bộ lọc. Các bộ lọc này có thể là phi tuyến, ngẫu nhiên, logic, không cố định, hoặc thậm chí không có tính phân tích. Chúng là học sinh học năng động và liên tục.

Mạng neuron LAMSTAR có thể phục vụ như là một mạng nơ ron năng động trong không gian hay miền thời gian, hoặc cả hai. Tốc độ của nó được quy định bởi các liên kết-trọng số Hebbian (chương 9 của D. Graupe, 2013), dùng để tích hợp các bộ lọc khác nhau và thường khác nhau (các hàm tiền xử lý) vào nó nhiều lớp

và để xếp hạng năng động tâm quan trọng của các lớp khác nhau và các hàm liên quan đến nhiệm vụ nhất định cho việc học sâu. Điều này hiển nhiên bắt chước học sinh học mà tích hợp các bộ tiền lý đầu ra khác nhau (Ốc tai, võng mạc, vv) và vỏ não (thính giác, thị giác, vv) và của các vùng khác nhau của chúng. Khả năng học sâu của nó tăng cường hơn nữa bằng cách sử dụng sự ức chế, sự tương quan và bởi khả năng đối phó với dữ liệu không đầy đủ của nó, hoặc "mất" nơ ron hoặc lớp ngay cả khi đang thực thi một tác vụ. Hơn nữa, nó hoàn toàn minh bạch do trọng số liên kết của nó. Các trọng số liên kết cho phép xác định năng động sáng tạo và thura thai, và tạo thuận lợi cho việc xếp hạng của các lớp, các bộ lọc hoặc các nơ ron đơn lẻ tương ứng với một nhiệm vụ.

LAMSTAR đã được áp dụng cho nhiều dự đoán y tế và tài chính (xem Graupe, 2013 Phần 9C), bộ lọc thích nghi nhiều nhận dạng giọng nói với tiếng ồn không xác định, nhận dạng ảnh tĩnh (Graupe, 2013 Phần 9D), nhận dạng ảnh video, bảo mật phần mềm, điều khiển thích nghi của các hệ thống phi tuyến, vv. LAMSTAR có tốc độ tính toán nhanh hơn nhiều và có lỗi hơi ít hơn so với một mạng nơ ron tích chập dựa trên các bộ lọc hàm-ReLU và thăm dò max, trong một nghiên cứu nhận dạng ký tự so sánh.

Các ứng dụng này chứng minh đào sâu vào các khía cạnh của các dữ liệu đó là bị ẩn từ các mạng học nông hoặc thậm chí từ những giác quan của con người (mắt, tai), chẳng hạn như trong trường hợp của dự đoán sự bắt đầu của hiện tượng ngừng thở khi ngủ, của một biểu đồ điện tâm đồ một thai nhi như được ghi chép từ các điện cực gắn trên da được đặt trên bụng người mẹ trong thời gian đầu của thai kỳ, của dự đoán tài chính (Phần 9C trong Graupe, 2013), hoặc trong lọc mù của nhiễu trong nhận dạng giọng nói.

2.2.4.6. Các mạng xếp chồng sâu

Một kiến trúc sâu dựa trên một hệ thống phân cấp của các khối mô-đun mạng neuron đơn giản là một mạng sâu lồi, được giới thiệu vào năm 2011. Ở đây, bài toán học các trọng số được xây dựng như một bài toán tối ưu hóa lồi với lời giải

dạng đóng. Kiến trúc này còn được gọi là một mạng xếp chồng sâu (DSN), nhấn mạnh các cơ chế tương tự với tổng quát hóa xếp chồng. Mỗi khối DSN là một module đơn giản đó là dễ dàng để huấn luyện chính nó trong một kiểu có giám sát mà không cần truyền ngược cho toàn bộ các khối.

2.2.4.7. Mạng lập trình sâu (deep coding network)

Có những lợi thế của một mô hình mà có thể chủ động cập nhật bản thân từ ngữ cảnh trong dữ liệu. Mạng lập trình (DPCN) là một chương trình lập trình tiên đoán, trong đó thông tin từ trên xuống được sử dụng để điều chỉnh theo kinh nghiệm của những cái trước đó cần thiết cho một thủ tục suy luận từ dưới lên bằng cách phương tiện của một mô hình thể sinh kết nối cục bộ sâu. Điều này hoạt động bằng cách chiết tách các đặc điểm rời rạc các quan sát biến đổi theo thời gian bằng cách sử dụng một mô hình động học tuyến tính. Sau đó, một chiến lược thăm dò được sử dụng để học các đại diện đặc điểm bất biến. Các đơn vị này tập hợp lại để tạo thành một kiến trúc sâu và được huấn luyện bởi học không giám sát layer-wise tham lam. Các lớp tạo thành một loại xích Markov mà các trạng thái tại bất kỳ lớp nào cũng chỉ phụ thuộc vào các lớp trước và các lớp sau (ké thừa).

Mạng lập trình dự đoán sâu (DPCN) dự đoán đại diện của lớp, bằng cách sử dụng một cách tiếp cận từ trên xuống bằng cách sử dụng thông tin ở lớp trên và các phụ thuộc thời gian từ các trạng thái trước đó.

DPCN có thể được mở rộng để tạo thành một mạng tích chập.

2.2.4.8. Mạng bộ nhớ

Bộ nhớ ngoài tích hợp với các mạng neuron nhân tạo tính đến nghiên cứu đầu tiên trong đại diện phân phối và các bản đồ tự tổ chức. Ví dụ, trong bộ nhớ phân tán hoặc bộ nhớ phân cấp thời gian, các mô hình được mã hóa bởi các mạng neuron được sử dụng như là các địa chỉ cho bộ nhớ có khả năng định địa chỉ nội dung, với các "nơ ron" chủ yếu phục vụ như là các bộ mã hóa và giải mã.

Bộ nhớ ngắn-hạn dài:

Trong thập niên 1990 và thập niên 2000, đã có nhiều công trình liên quan đến bộ nhớ ngắn-hạn dài (LSTM - thêm bộ nhớ khả vi cho các hàm hồi qui). Ví dụ:

- Các hành động đẩy và lấy ra khả vi cho các mạng bộ nhớ thay thế được gọi là các máy ngắn xếp nơ ron
- Memory networks where the control network's external differentiable storage is in the fast weights of another network
- LSTM "forget gates"
- Self-referential recurrent neural networks (RNNs) with special output units for addressing and rapidly manipulating each of the RNN's own weights in differentiable fashion (internal storage)
- Learning to transduce with unbounded memory

Các mạng bộ nhớ:

Các mạng bộ nhớ một mở rộng khác của các mạng nơ ron nhân tạo kết hợp với bộ nhớ dài hạn, được phát triển bởi nhóm nghiên cứu Facebook. Bộ nhớ dài hạn có thể được đọc và ghi vào đó, với mục đích sử dụng cho việc dự báo. Các mô hình này đã được áp dụng trong bối cảnh hỏi đáp (QA) nơi bộ nhớ dài hạn hoạt động hiệu quả như một cơ sở kiến thức (năng động), và đầu ra là một đáp ứng văn bản.

Các mạng mã hóa-giải mã:

Một framework mã hóa-giải mã là một framework dựa trên các mạng neuron nhằm mục đích lập bản đồ đầu vào cấu trúc cao tới đầu ra có cấu trúc cao. Nó đã được đề xuất gần đây trong bối cảnh của máy dịch, trong đó đầu vào và đầu ra được viết thành câu bằng hai ngôn ngữ tự nhiên. Trong đó, một mạng nơ ron tái phát (RNN) hoặc mạng neuron tích chập (CNN) được sử dụng như một bộ mã hóa để tóm tắt một câu nguồn và tóm tắt này được giải mã bằng cách sử dụng một mô hình ngôn ngữ mạng neuron tái phát có điều kiện để tạo ra bản dịch. Tất cả các hệ thống này có các khối xây dựng tương tự: cỗng RNN và CNN, và các cơ chế tập trung được huấn luyện.

2.2.5. Ứng dụng

2.2.5.1. Nhận dạng tiếng nói tự động

Các kết quả hiển thị trong bảng dưới đây là nhận dạng tiếng nói tự động trên tập dữ liệu TIMIT phổ biến. Đây là một tập hợp dữ liệu phổ biến được sử dụng để đánh giá ban đầu các kiến trúc học sâu. Toàn bộ tập dữ liệu này có 630 người nói từ tám phương ngữ chính của tiếng Anh Mỹ, trong đó mỗi người đọc 10 câu. Kích thước nhỏ của nó cho phép nhiều cấu hình được thử nghiệm một cách hiệu quả. Quan trọng hơn, nhiệm vụ của TIMIT liên quan đến việc nhận dạng trình tự-điện thoại, trong đó, không giống như việc nhận dạng trình tự-từ, cho phép các "mô hình ngôn ngữ" rất yếu và do đó là điểm yếu trong mô hình hóa âm thanh trong các khía cạnh của nhận dạng giọng nói có thể được phân tích dễ dàng hơn. Các phân tích như vậy trên TIMIT bởi Li Deng và các cộng tác viên khoảng năm 2009-2010, tương phản với GMM (và các mô hình thể sinh khác của giọng nói) với cá mô hình DNN, kích thích đầu tư công nghiệp sớm vào học sâu cho nhận dạng giọng nói từ quy mô nhỏ cho đến quy mô lớn, cuối cùng dẫn đến việc sử dụng phổ biến và chi phối trong ngành công nghiệp đó. Phân tích đó được thực hiện với sự so sánh hiệu suất (ít hơn 1,5% tỷ lệ lỗi) giữa các DNN tách biệt và các mô hình thể sinh.

Trong năm 2010, các nhà nghiên cứu công nghiệp đã mở rộng học sâu từ TIMIT để nhận dạng giọng nói với số lượng từ vựng lớn, bằng việc áp dụng các lớp sản lượng lớn DNN dựa trên các trạng thái HMM phụ thuộc vào ngữ cảnh được xây dựng bởi cây quyết định. Đánh giá toàn diện sự phát triển và tiến bộ này tới thời điểm năm 2014 là cuốn sách gần đây Springer từ Microsoft Research. Một bài báo đánh giá về nền tảng của nhận dạng giọng nói tự động và tác động của các mô hình máy học, bao gồm cả học sâu.

Một trong những nguyên tắc cơ bản của học sâu là để thoát khỏi kỹ thuật đặc tính thủ công và sử dụng các đặc tính thô. Nguyên tắc này được khám phá thành công đầu tiên trong kiến trúc của tự mã hóa sâu trên ảnh phô "thô" hoặc các đặc điểm dải lọc tuyến tính, hiển thị sự vượt trội của nó hơn các tính năng Mel-

Cepstral mà có chứa một vài giai đoạn chuyển đổi cố định từ ảnh phô. Các tính năng thực sự "thô" của tiếng nói, dạng sóng, gần đây đã được chỉ ra để tạo ra các kết quả nhận dạng giọng nói tuyệt vời ở quy mô lớn.

Kể từ khi ra mắt thành công ban đầu của DNN cho nhận dạng tiếng nói khoảng 2009-2011, tiến độ (và hướng đi trong tương lai) có thể được tóm tắt vào 8 lĩnh vực chính:

- Mở rộng quy mô lên/ra và tăng tốc quá trình đào tạo và giải mã DNN;
- Huấn luyện suy luận có trình tự cho các DNN;
- Xử lý đặc điểm bởi các mô hình sâu với sự hiểu biết vững chắc các cơ chế tiềm ẩn;
- Thích nghi của các DNN và các mô hình sâu có liên quan;
- Học đa tác vụ và học có chuyên giao bởi các DNN và các mô hình sâu liên quan; Các mạng neuron tích chập và làm thế nào để thiết kế chúng để khai thác tốt nhất kiến thức miền của giọng nói;
- Mạng neuron tái phát và các biến thể giàu LSTM;
- Các loại mô hình sâu bao gồm các mô hình dựa trên tensor và các mô hình tích hợp sâu thể sinh/suy xét.

Trường hợp nhận dạng tiếng nói tự động quy mô lớn lần đầu tiên và thuyết phục nhất thành công của học sâu trong lịch sử gần đây, chấp nhận bởi cả công nghiệp và hàn lâm trong tất cả các lĩnh vực. Từ năm 2010 đến năm 2014, hai hội nghị lớn về xử lý tín hiệu và nhận dạng giọng nói, IEEE-ICASSP và Interspeech, đã thấy một sự gia tăng lớn các báo cáo được chấp nhận trong các báo cáo hội nghị thường niên tương ứng về chủ đề học sâu trong nhận dạng giọng nói. Quan trọng hơn, tất cả các hệ thống nhận dạng giọng nói thương mại chính (ví dụ: Microsoft Cortana, Xbox, Skype Translator, Google Now, Apple Siri, Baidu và iFlyTek tìm kiếm bằng giọng nói và một loạt các sản phẩm của Nuance speech, vv) được dựa trên phương pháp học sâu. Xem thêm các cuộc phỏng vấn trên phương tiện truyền thông với CTO của Nuance Communications.

Thành công lây lan rộng trong nhận dạng tiếng nói đã đạt được vào năm 2011 được kế tiếp liền sau đó là nhận dạng hình ảnh ở quy mô lớn.

2.2.5.2. Nhận dạng hình ảnh

Một tập đánh giá phổ biến cho phân loại hình ảnh là tập hợp dữ liệu cơ sở dữ liệu MNIST. MNIST bao gồm các chữ số viết tay và bao gồm 60000 ví dụ huấn luyện và 10000 ví dụ kiểm tra. Như TIMIT, kích thước nhỏ của nó cho phép nhiều câu hình được kiểm tra. Một danh sách đầy đủ các kết quả trên tập này có thể được tìm thấy trong. Kết quả tốt nhất hiện nay trên MNIST là tỷ lệ lỗi 0,23%, đạt được bởi Ciresan và các cộng sự vào năm 2012.

Tác động thực sự của học sâu trong nhận dạng hình ảnh hoặc đối tượng, một chi chín của thị giác máy tính, đã cảm thấy được vào mùa thu năm 2012 sau khi đội của Geoff Hinton và sinh viên của ông thắng trong cuộc thi quy mô lớn ImageNet bởi một biên độ đáng kể bằng phương pháp máy học nông tiên tiến nhất. Công nghệ này dựa trên các mạng tích chập sâu 20 tuổi, nhưng với quy mô lớn hơn nhiều trên một nhiệm vụ lớn hơn nhiều, vì nó đã học được rằng học sâu làm việc tốt đối với nhận dạng giọng nói quy mô lớn. Trong năm 2013 và 2014, tỷ lệ lỗi trong tác vụ của ImageNet bằng cách sử dụng học sâu tiếp tục giảm xuống nhanh chóng, theo một xu hướng tương tự trong nhận dạng giọng nói quy mô lớn.

Khi tham vọng này di chuyển từ nhận dạng giọng nói tự động sang các bản dịch giọng nói tự động và hiểu được, phân loại hình ảnh gần đây đã được mở rộng với nhiệm vụ khó khăn hơn đó là tạo phụ đề cho hình ảnh tự động, trong đó có học sâu là công nghệ cơ bản thiết yếu.

Một ứng dụng ví dụ là một máy tính xe hơi cho biết được đào tạo bằng học sâu, có thể cho phép xe diễn giải các hình ảnh 360° từ camera. Một ví dụ khác là công nghệ được gọi là Facial Dysmorphology Novel Analysis (FDNA) (Phân tích các dị tật của khuôn mặt) sử dụng để phân tích các trường hợp dị dạng của con người kết nối với cơ sở dữ liệu lớn của các hội chứng di truyền.

2.2.5.3. Xử lý ngôn ngữ tự nhiên

Mạng neuron đã được sử dụng cho việc thực hiện các mô hình ngôn ngữ kể từ đầu những năm 2000. Các kỹ thuật quan trọng trong lĩnh vực này là lấy mẫu âm và nhúng chữ (word embedding). Nhúng chữ, chẳng hạn như word2vec, có thể được dùng như một lớp đại diện trong một kiến trúc học sâu, điều này sẽ biến đổi một từ đơn thành một đại diện vị trí của từ đó liên quan đến các từ khác trong bộ dữ liệu; vị trí được đại diện như là một điểm trong một không gian vector. Sử dụng một từ nhúng như là một lớp đầu vào với một mạng lưới neuron đệ quy (RNN-recursive neuron network) cho phép đào tạo mạng để phân tích cú pháp câu và cụm từ bằng cách sử dụng một ngữ pháp vector tổng hợp có hiệu quả. Một ngữ pháp vector tổng hợp có thể được coi là ngữ pháp không phụ thuộc ngữ cảnh xác suất (PCFG-probabilistic context free grammar) được thực hiện bởi một mạng neuron đệ quy. Tự động-mã hóa đệ qui được xây dựng trên đỉnh từ nhúng đã được đào tạo để đánh giá câu tương tự và phát hiện các chủ giải dài dòng. Các kiến trúc neuron sâu đã đạt được những kết quả tiên tiến nhất trong nhiều tác vụ xử lý ngôn ngữ tự nhiên như phân tích thống kê, phân tích tình cảm, tra cứu thông tin, dịch máy, liên kết thực thể ngữ cảnh, và.v.v.

2.2.5.4. Khám phá dược phẩm và độc chất học

Ngành công nghiệp dược phẩm phải đối mặt với vấn đề mà một tỷ lệ lớn các loại thuốc tiềm năng thất bại khi tiếp cận với thị trường. Những thất bại của các hợp chất hóa học này gây ra bởi không đủ hiệu quả trên mục tiêu phân tử sinh học (có hiệu lực với mục tiêu), có các tương tác không bị phát hiện và không mong muốn với các phân tử sinh học khác (chênh mục tiêu tác động), hoặc các hiệu ứng độc dược ngoài dự tính. Trong năm 2012, một nhóm dẫn đầu bởi George Dahl đã chiến thắng "Merck Molecular Activity Challenge" sử dụng các mạng neuron sâu đa tác vụ để dự đoán mục tiêu phân tử sinh học của một hợp chất. Trong năm 2014, nhóm của Sepp Hochreiter sử dụng học sâu để phát hiện ra mục tiêu lạ và các ảnh hưởng độc dược của các môi trường hóa chất trong các chất dinh dưỡng, sản phẩm gia dụng và thuốc men và đã chiến thắng "Tox21 Data Challenge" của NIH, FDA

và NCATS. Những thành công ám tượng chỉ ra rằng học sâu có thể vượt trội so với các phương pháp kiểm tra ảo khác. Các nhà nghiên cứu đến từ Google và Stanford đã mở rộng học sâu để khám phá được phẩm bằng cách kết hợp dữ liệu từ nhiều nguồn khác nhau. Năm 2015, Atomwise giới thiệu AtomNet, mạng neuron học sâu đầu tiên dành cho thiết kế dược phẩm dựa trên cấu trúc hợp lý. Sau đó, AtomNet đã được sử dụng để dự đoán các phân tử sinh học được chọn mới lạ đối với nhiều mục tiêu bệnh tật, đặc biệt là phương pháp điều trị bệnh do virus Ebola và bệnh đa xơ cứng.

2.2.5.5. Quản lý quan hệ khách hàng (CRM)

Thành công gần đây đã được báo cáo với ứng dụng của học tăng cường sâu trong các thiết lập tiếp thị trực tiếp, thể hiện sự phù hợp của phương pháp này dành cho tự động hóa CRM. Một mạng nơ ron được sử dụng để ước tính giá trị của các hành động có thể trực tiếp thị trên không gian trạng thái khách hàng, được định nghĩa trong điều khoản của biến RFM. Hàm giá trị ước tính được chỉ ra để có một giải thích tự nhiên như là giá trị khách hàng suốt đời.

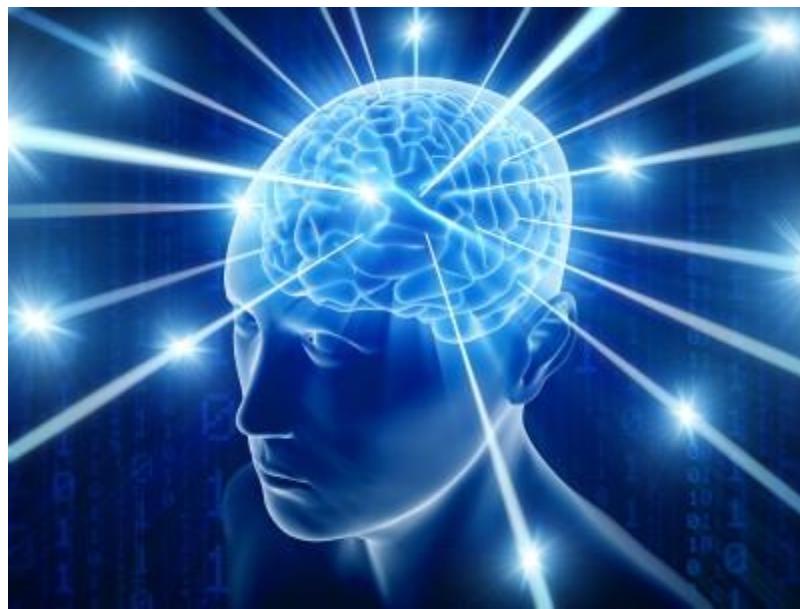
2.2.5.6. Các hệ thống khuyến cáo (gợi ý)

Các hệ thống khuyến cáo đã sử dụng học sâu để trích xuất các đặc điểm sâu có ý nghĩa cho mô hình yếu tố tiềm ẩn đối với khuyến cáo dựa trên nội dung cho âm nhạc. Gần đây, một cách tiếp cận tổng quát hơn cho việc học tập sở thích người dùng từ nhiều miền bằng cách sử dụng học sâu đa góc nhìn đã được đưa ra. Mô hình này sử dụng một cộng tác lai và tiếp cận dựa trên nội dung và tăng cường các khuyến nghị trong nhiều nhiệm vụ.

2.2.5.7. Tin sinh học

Gần đây, một cách tiếp cận học sâu dựa trên một mạng neuron nhân tạo tự mã hóa đã được sử dụng trong tin sinh học, để dự đoán các mối quan hệ chức năng gen và các chú thích Bản thể gen.

2.2.6. Lý thuyết về bộ não con người



Hình 3: Ảnh minh họa não bộ con người

Tính toán học sâu có liên hệ chặt chẽ đến học thuyết về sự phát triển của não bộ (cụ thể, phát triển neocortical) do các nhà khoa học neuron nhận thức đề xuất trong đầu thập niên 1990. Một bản tóm tắt dễ tiếp cận của ý tưởng này là tác phẩm của Elman và các cộng sự vào năm 1996 "Xem xét lại Tính bẩm sinh" (Xem thêm: Shrager và Johnson; Quartz và Sejnowski). Những lý thuyết phát triển này cũng được thuyết minh cụ thể trong các mô hình tính toán, chúng là những kỹ thuật tiền nhiệm của các mô hình học sâu được thúc đẩy bởi tính toán (bằng máy tính) đơn thuần. Những mô hình phát triển này chia sẻ thuộc tính thú vị mà nhiều động lực học (learning dynamics) khác nhau được đề xuất trong nghiên cứu não bộ (Ví dụ, một làn sóng của yếu tố tăng trưởng neuron) để hỗ trợ việc tự tổ chức của các loại mạng nơ ron có liên quan với nhau được sử dụng trong các mô hình học sâu thuần tính toán sau đó; và các mạng neuron tính toán như vậy có vẻ tương tự như quan điểm của ngành nghiên cứu vỏ não mới như một hệ thống phân cấp của bộ lọc trong đó mỗi lớp chụp một số thông tin trong môi trường hoạt động, và sau đó đi qua phần còn lại, cũng như tín hiệu cơ bản được sửa đổi, tới các lớp khác cao hơn trong hệ thống phân cấp. Quá trình này mang lại một chồng tự tổ chức các cảm biến, cũng như điều chỉnh để hoạt động môi trường của họ. Như được mô tả trên tờ New York Times vào năm 1995: "...bộ não của những trẻ sơ sinh dường như tự tổ chức riêng chính nó dưới ảnh hưởng của các sóng của cái gọi là các yếu tố - dinh dưỡng..."

các khu vực khác nhau của não trở nên kết nối tuần tự, với một lớp mô trưởng thành trước các mô khác và cho đến khi toàn bộ não là trưởng thành."

Tầm quan trọng của học sâu đối với sự tiến hóa và phát triển của nhận thức của con người đã không thoát khỏi sự chú ý của các nhà nghiên cứu. Một khía cạnh của phát triển con người là phân biệt chúng ta với những người hàng xóm trong họ linh trưởng gần nhất của mình có thể thay đổi trong thời gian phát triển. Trong số các loài linh trưởng, bộ não con người vẫn còn tương đối mềm dẻo cho đến cuối thời kỳ sau khi sinh, trong khi bộ não của họ hàng gần nhất của chúng ta hoàn toàn cố định hơn ngay sau khi sinh. Vì vậy, con người có khả năng truy cập lớn hơn vào những kinh nghiệm phức tạp đang diễn ra trên thế giới trong giai đoạn hình thành nhất của sự phát triển não bộ. Điều này có thể cho phép chúng ta "điều chỉnh" để thay đổi nhanh chóng môi trường mà các động vật khác, nhiều bị hạn chế bởi cơ cấu tiến hóa của bộ não của chúng, không thể để thực hiện được. Đến mức mà những thay đổi này được phản ánh trong các thay đổi thời gian tương tự trong sóng được giả thuyết của sự phát triển vỏ não, chúng cũng có thể dẫn đến những thay đổi trong việc khai thác thông tin từ môi trường kích thích trong thời gian đầu tự tổ chức của bộ não. Tất nhiên, cùng với tính linh hoạt này đến một giai đoạn kéo dài chưa thành thực, trong đó chúng ta phụ thuộc vào người chăm sóc và cộng đồng của mình để hỗ trợ và đào tạo. Lý thuyết của học sâu do đó thấy sự cùng tiến hóa đồng thời của văn hóa và nhận thức như là một điều kiện cơ bản của sự tiến hóa của con người.

2.3. Thư viện xử lý ảnh OpenCV

❖ Tổng quan về OpenCV



Hình 4: Giới thiệu thư viện OpenCV

OpenCV (Open Computer Vision) là một thư viện mã nguồn mở hàng đầu cho xử lý về thị giác máy tính, machine learning, xử lý ảnh. OpenCV chính thức được ra mắt đầu tiên vào năm 1999, OpenCV là thư viện mã nguồn mở miễn phí cho cả học thuật và thương mại. OpenCV được viết bằng C/C++, vì vậy có tốc độ tính toán rất nhanh, có thể sử dụng với các ứng dụng liên quan đến thời gian thực. Thư viện có hơn 2500 thuật toán được tối ưu hóa. Opencv có các interface cho C/C++, Python Java vì vậy hỗ trợ được cho Window, Linux, MacOs lẫn Android, iOS OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần.

❖ **Chức năng có trong thư viện OpenCV**

Thư viện OpenCV bao gồm một số tính năng nổi bật như:

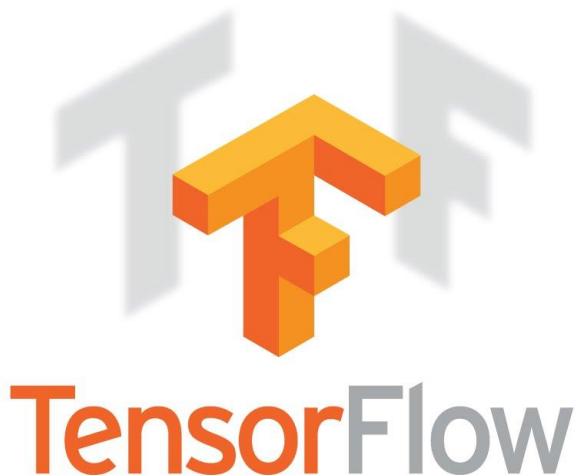
- + Bộ công cụ hỗ trợ 2D và 3D
- + Nhận diện khuôn mặt
- + Nhận diện cử chỉ
- + Nhận dạng chuyển động, đối tượng, hành vi
- + Tương tác giữa con người và máy tính
- + Điều khiển Robot
- + Hỗ trợ thực tế tăng cường.

❖ **Ứng dụng:** Opencv có rất nhiều ứng dụng:

- + Nhận dạng ảnh
- + Xử lý hình ảnh
- + Phục hồi hình ảnh/video
- + Thực tế ảo
- + Các ứng dụng khác

2.4. Thư viện học máy Tensorflow

2.4.1. Thư viện Tensorflow là gì?



Hình 5: Giới thiệu thư viện Tensorflow

Tensorflow là một thư viện mã nguồn mở mạnh mẽ cho Machine Learning được phát triển bởi các nhà nghiên cứu của Google. Thư viện cho phép xây dựng nhiều mô hình học sâu khác nhau.

Tensorflow cũng cho phép tính toán song song trên nhiều máy tính khác nhau, tất nhiên là cũng có thể trên nhiều CPU, GPU trong cùng một máy. Tensorflow cung cấp các api làm việc với Python, C++.

Nó hiện đang được sử dụng cho cả nghiên cứu lẫn sản xuất bởi 50 đội khác nhau trong hàng tá sản phẩm thương mại của Google, như nhận dạng giọng nói,

Gmail, Google Photos và tìm kiếm, nhiều trong số đó đã từng sử dụng chương trình tiền nhiệm DistBelief của nó.

2.4.2. Đặc điểm nổi bật của Tensorflow

❖ Xây dựng mô hình dễ dàng

TensorFlow cung cấp nhiều cấp độ trừu tượng để có thể chọn cấp độ phù hợp với nhu cầu. Xây dựng và đào tạo các mô hình bằng cách sử dụng API Keras cấp cao, giúp dễ dàng bắt đầu với TensorFlow và học máy.

Nếu cần linh hoạt hơn, việc thực thi háo hức cho phép lắp lại ngay lập tức và gỡ lỗi trực quan. Đối với các nhiệm vụ đào tạo ML lớn, sử dụng API chiến lược phân phối để đào tạo phân tán trên các cấu hình phần cứng khác nhau mà không thay đổi định nghĩa mô hình.

❖ Sản xuất Machine Learning mạnh mẽ ở mọi nơi

TensorFlow luôn cung cấp một đường dẫn trực tiếp đến quá trình sản xuất. Cho dù đó là trên máy chủ, thiết bị di động hay web, TensorFlow cho phép bạn đào tạo và triển khai mô hình của mình một cách dễ dàng, bất kể bạn sử dụng ngôn ngữ hoặc nền tảng nào.

Sử dụng TensorFlow Extended (TFX) nếu bạn cần một đường ống Machine Learning sản xuất đầy đủ. Để chạy được trên thiết bị di động và thiết bị cầm tay, hãy sử dụng TensorFlow Lite. Đào tạo và triển khai các mô hình trong môi trường JavaScript bằng TensorFlow.js.

❖ Thủ nghiệm mạnh mẽ để nghiên cứu

Xây dựng và đào tạo các mô hình hiện đại mà không ảnh hưởng đến tốc độ hoặc hiệu suất. TensorFlow cung cấp cho bạn sự linh hoạt và khả năng kiểm soát với các tính năng như API chức năng Keras và API phân lớp mô hình để tạo các cấu trúc liên kết phức tạp.

TensorFlow cũng hỗ trợ một hệ sinh thái gồm các thư viện và mô hình tiện ích bổ sung mạnh mẽ để thử nghiệm, bao gồm Ragged Tensors, TensorFlow Probability, Tensor2Tensor và BERT.

2.4.3. Ứng dụng của Tensorflow

Một số project nổi tiếng sử dụng thư viện Tensorflow như:

- Phân loại ung thư da – Dermatologist-level classification of skin cancer with deep neural networks (Esteva et al., Nature 2017)
- WaveNet: Text to speech – Wavenet: A generative model for raw audio (Oord et al., 2016)
- Vẽ hình – Draw Together with a Neural Network (Ha et al., 2017)
- Image Style Transfer Using Convolutional Neural Networks (Gatys et al., 2016)

2.5. Ngôn ngữ lập trình Python

2.5.1. Sơ lược về Python



Hình 6: Giới thiệu ngôn ngữ Python

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình

thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, Python dần mở rộng sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python.

Sự phát triển Python đến nay có thể chia làm các giai đoạn:

Python 1: Bao gồm các bản phát hành 1.x. Giai đoạn này, kéo dài từ đầu đến cuối thập niên 1990. Từ năm 1990 đến 1995, Guido làm việc tại CWI (Centrum voor Wiskunde en Informatica - Trung tâm Toán-Tin học tại Amsterdam, Hà Lan). Vì vậy, các phiên bản Python đầu tiên đều do CWI phát hành. Phiên bản cuối cùng phát hành tại CWI là 1.2.

Vào năm 1995, Guido chuyển sang CNRI (Corporation for National Research Initiatives) ở Reston, Virginia. Tại đây, ông phát hành một số phiên bản khác. Python 1.6 là phiên bản cuối cùng phát hành tại CNRI.

Sau bản phát hành 1.6, Guido rời bỏ CNRI để làm việc với các lập trình viên chuyên viết phần mềm thương mại. Tại đây, ông có ý tưởng sử dụng Python với các phần mềm tuân theo chuẩn GPL. Sau đó, CNRI và FSF (Free Software Foundation - Tổ chức phần mềm tự do) đã cùng nhau hợp tác để làm bản quyền Python phù hợp với GPL. Cùng năm đó, Guido được nhận Giải thưởng FSF vì Sự phát triển Phần mềm tự do (Award for the Advancement of Free Software).

Phiên bản 1.6.1 ra đời, là phiên bản đầu tiên tuân theo bản quyền GPL. Tuy nhiên, bản này hoàn toàn giống bản 1.6, trừ một số sửa lỗi cần thiết.

Python 2: Vào năm 2000, Guido và nhóm phát triển Python dời đến BeOpen.com và thành lập BeOpen PythonLabs team. Phiên bản Python 2.0 được phát hành tại đây. Sau khi phát hành Python 2.0, Guido và các thành viên PythonLabs gia nhập Digital Creations.

Python 2.1 ra đời kế thừa từ Python 1.6.1 và Python 2.0. Bản quyền của phiên bản này được đổi thành Python Software Foundation License. Từ thời điểm này trở đi, Python thuộc sở hữu của Python Software Foundation (PSF), một tổ chức phi lợi nhuận được thành lập theo mẫu Apache Software Foundation.

Python 3 (còn gọi là **Python 3000** hoặc **Py3K**): Dòng 3.x sẽ không hoàn toàn tương thích với dòng 2.x, tuy vậy có công cụ hỗ trợ chuyển đổi từ các phiên bản 2.x sang 3.x. Nguyên tắc chủ đạo để phát triển Python 3.x là "bỏ cách làm việc cũ nhằm hạn chế trùng lặp về mặt chức năng của Python". Trong PEP (Python Enhancement Proposal) có mô tả chi tiết các thay đổi trong Python.

2.5.2. Các đặc điểm của ngôn ngữ lập trình Python

- + **Dễ học, dễ đọc:** Python được thiết kế để trở thành một ngôn ngữ dễ học, mã nguồn dễ đọc, bỏ cục trực quan, dễ hiểu.
- + **Từ khóa:** Python tăng cường sử dụng từ khóa tiếng Anh, hạn chế các ký hiệu và cấu trúc cú pháp so với các ngôn ngữ khác, là một ngôn ngữ phân biệt kiểu chữ HOA, chữ thường cũng như C/C++, các từ khóa của Python đều ở dạng chữ thường.
- + **Khối lệnh:** Trong các ngôn ngữ khác, khối lệnh thường được đánh dấu bằng cặp ký hiệu hoặc từ khóa. Ví dụ, trong C/C++, cặp ngoặc nhọn { } được dùng để bao bọc một khối lệnh. Python, trái lại, có một cách rất đặc biệt để tạo khối lệnh, đó là thụt các câu lệnh trong khối vào sâu hơn (về bên phải) so với các câu lệnh của khối lệnh cha chứa nó.

- + **Khả năng mở rộng:** Python có thể được mở rộng nếu ta biết sử dụng C, ta có thể dễ dàng viết và tích hợp vào Python nhiều hàm tùy theo nhu cầu. Các hàm này sẽ trở thành hàm xây dựng sẵn (built-in) của Python. Ta cũng có thể mở rộng chức năng của trình thông dịch, hoặc liên kết các chương trình Python với các thư viện chỉ ở dạng nhị phân (như các thư viện đồ họa do nhà sản xuất thiết bị cung cấp). Hơn thế nữa, ta cũng có thể liên kết trình thông dịch của Python với các ứng dụng viết từ C và sử dụng nó như là một mở rộng hoặc một ngôn ngữ dòng lệnh phụ trợ cho ứng dụng đó.
- + **Trình thông dịch:** Python là một ngôn ngữ lập trình dạng thông dịch, do đó có ưu điểm tiết kiệm thời gian phát triển ứng dụng vì không cần phải thực hiện biên dịch và liên kết. Trình thông dịch có thể được sử dụng để chạy file script, hoặc cũng có thể được sử dụng theo cách tương tác. Ở chế độ tương tác, trình thông dịch Python tương tự shell của các hệ điều hành họ Unix, tại đó, ta có thể nhập vào từng biểu thức rồi gõ Enter, và kết quả thực thi sẽ được hiển thị ngay lập tức. Đặc điểm này rất hữu ích cho người mới học, giúp họ nghiên cứu tính năng của ngôn ngữ; hoặc để các lập trình viên chạy thử mã lệnh trong suốt quá trình phát triển phần mềm. Ngoài ra, cũng có thể tận dụng đặc điểm này để thực hiện các phép tính như với máy tính bỏ túi.
- + **Lệnh và cấu trúc điều khiển:** Mỗi câu lệnh trong Python nằm trên một dòng mã nguồn. Ta không cần phải kết thúc câu lệnh bằng bất kì ký tự gì. Cũng như các ngôn ngữ khác, Python cũng có các cấu trúc điều khiển. Chúng bao gồm:
 - Cấu trúc rẽ nhánh: cấu trúc if (có thể sử dụng thêm elif hoặc else), dùng để thực thi có điều kiện một khối mã cụ thể.
 - Cấu trúc lặp, bao gồm:
 - **Lệnh while:** chạy một khối mã cụ thể cho đến khi điều kiện lặp có giá trị false.
 - **Vòng lặp for:** lặp qua từng phần tử của một dãy, mỗi phần tử sẽ được đưa vào biến cục bộ để sử dụng với khối mã trong vòng lặp.
 - Python cũng có từ khóa class dùng để khai báo lớp (sử dụng trong lập trình hướng đối tượng) và lệnh def dùng để định nghĩa hàm.

+ Hệ thống kiểu dữ liệu

Python sử dụng hệ thống kiểu duck typing, còn gọi là latent typing (tự động xác định kiểu). Có nghĩa là, Python không kiểm tra các ràng buộc về kiểu dữ liệu tại thời điểm dịch, mà là tại thời điểm thực thi. Khi thực thi, nếu một thao tác trên một đối tượng bị thất bại, thì có nghĩa là đối tượng đó không sử dụng một kiểu thích hợp.

Python cũng là một ngôn ngữ định kiểu mạnh. Nó cấm mọi thao tác không hợp lệ, ví dụ cộng một con số vào chuỗi ký tự. Sử dụng Python, ta không cần phải khai báo biến. Biến được xem là đã khai báo nếu nó được gán một giá trị lần đầu tiên. Căn cứ vào mỗi lần gán, Python sẽ tự động xác định kiểu dữ liệu của biến. Python có một số kiểu dữ liệu thông dụng sau:

- **int, long:** số nguyên (trong phiên bản 3.x long được nhập vào trong kiểu int). Độ dài của kiểu số nguyên là tùy ý, chỉ bị giới hạn bởi bộ nhớ máy tính.
- **float:** số thực
- **complex:** số phức, chẳng hạn $5+4j$
- **list:** dãy trong đó các phần tử của nó có thể được thay đổi, chẳng hạn [8, 2, 'b', -1.5]. Kiểu dãy khác với kiểu mảng (array) thường gặp trong các ngôn ngữ lập trình ở chỗ các phần tử của dãy không nhất thiết có kiểu giống nhau.

Ngoài ra phần tử của dãy còn có thể là một dãy khác.

- **tuple:** dãy trong đó các phần tử của nó không thể thay đổi.
- **str:** chuỗi ký tự. Từng ký tự trong chuỗi không thể thay đổi. Chuỗi ký tự được đặt trong dấu nháy đơn, hoặc nháy kép.
- **dict:** từ điển, còn gọi là "hashtable": là một cặp các dữ liệu được gắn theo kiểu {từ khóa: giá trị}, trong đó các từ khóa trong một từ điển nhất thiết phải khác nhau. Chẳng hạn {1: "Python", 2: "Pascal"}.
- **set:** một tập không xếp theo thứ tự, ở đó, mỗi phần tử chỉ xuất hiện một lần.

+ **Các kiểu dữ liệu:**

- Kiểu số
- Kiểu chuỗi (string)
- Kiểu bộ (tuple)
- Kiểu danh sách (list)
- Kiểu từ điển (dictionary)

2.5.3. Ứng dụng của Python

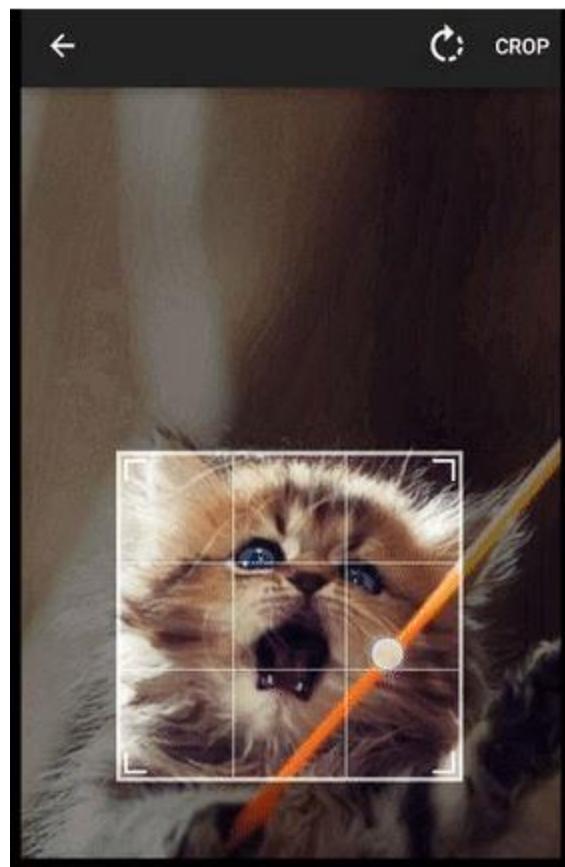
Python được ứng dụng nhiều trong các lĩnh vực khác nhau:

- + Làm web: Python cung cấp nhiều framework để ta có thể lựa chọn để phát triển ứng dụng web tùy theo mô hình của ứng dụng, Django framework. Udacity, youtube, dropbox được xây dựng (một phần lớn) sử dụng python.
- + Làm game: Pygame. Có một game do người Việt làm, gần đây khá nổi là Millia 44 cũng được xây dựng bằng python, đương nhiên python không phải là lựa chọn tốt nhất để xây dựng game.
- + Máy học: Theano, tensorflow, scikit-learn...
- + Khoa học máy tính: Python Opencv, numpy, panda, scipy...
- + Lập trình cho bo mạch: Arduiuno, raspberry pi.

2.6. Thư viện cắt ảnh Crop Image

2.6.1. Tổng quan

Là thư viện cắt ảnh đơn giản, hỗ trợ mạnh mẽ (Thu phóng, Xoay, Đa nguồn), có thể tùy chỉnh (Hình dạng, Giới hạn, Kiểu), tối ưu hóa (Không đồng bộ, Lấy mẫu, Ma trận) cho Android.



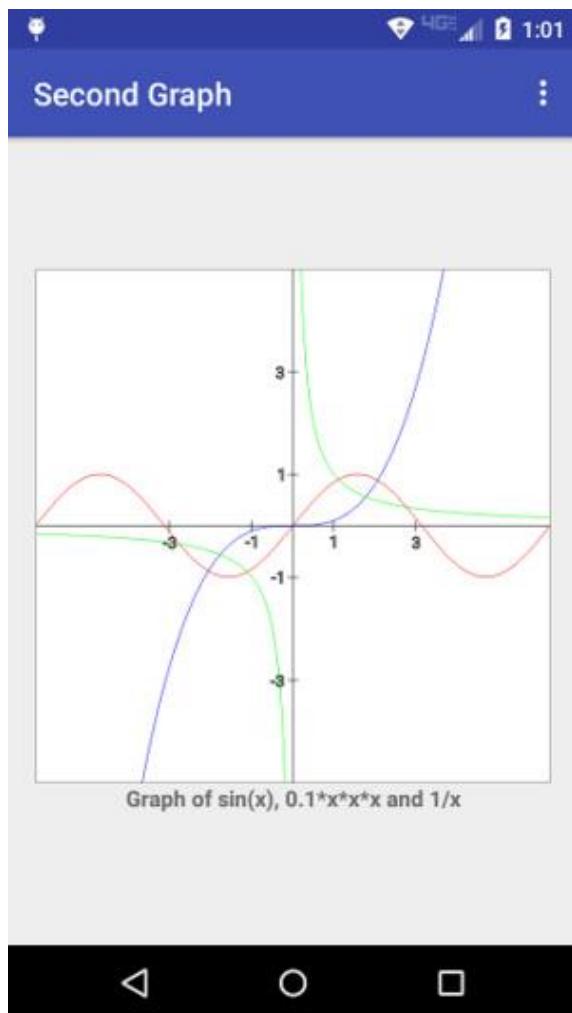
Hình 7: Giới thiệu thư viện cắt ảnh Crop Image

2.6.2. Tính năng

- Tự động tích hợp CropImageActivity.
- Đặt ảnh cắt dưới dạng Bitmap, Resource hoặc Android URI (Thư viện ảnh, Máy ảnh, Dropbox, v.v.).
- Cho phép xoay / lật hình ảnh trong khi cắt.
- Tự động phóng to / thu nhỏ vùng cắt có liên quan.
- Tự động xoay ảnh bitmap theo dữ liệu hình ảnh.
- Đặt giới hạn tối thiểu / tối đa cho hình ảnh, kết quả tính bằng pixel.
- Cho phép thiết lập kích thước / vị trí cửa sổ cắt ban đầu.
- Yêu cầu thay đổi kích thước hình ảnh đã cắt thành kích thước cụ thể.
- Tối ưu hóa bộ nhớ bitmap
- Hỗ trợ phiên bản Android có API 14 trở lên.

2.7. Thư viện vẽ đồ thị GraphLib

2.7.1. Tổng quan



Hình 8: Giới thiệu thư viện vẽ đồ thị GraphLib

GraphLib là thư viện hỗ trợ vẽ đồ thị cho ứng dụng Adroid. Nhiều ứng dụng Android có thể sử dụng hiệu quả các biểu đồ và vùng dữ liệu để giúp minh họa các mối quan hệ, mô tả xu hướng dữ liệu và theo dõi tiến độ so với mục tiêu.

GraphLib là một thư viện dễ sử dụng để vẽ đồ thị hàm, vẽ các điểm dữ liệu và vẽ biểu đồ đường. Thư viện này được cung cấp hoàn toàn miễn phí.

2.7.2. Đặc điểm

- Dung lượng nhỏ, không ảnh hưởng đến tốc độ ứng dụng
- Không cần quá nhiều tác vụ xử lý

- Để vẽ được đồ thị, chỉ cần tính toán và truyền vào tham số cho nó
- Vẽ được nhiều đồ thị cùng lúc: phương trình đường thẳng, parabol, đồ thị hình sin, ...

2.7.3. Hạn chế

- Giao diện hiển thị còn chưa đẹp mắt, chưa đáp ứng được các trải nghiệm người dùng.

2.8. Thư viện hiển thị biểu thức toán học MathView

2.8.1. Giới thiệu LaTeX



Hình 9: Giới thiệu ngôn ngữ đánh dấu LaTeX

LaTeX là ngôn ngữ đánh dấu khi soạn thảo tài liệu. Khi viết, người viết sử dụng văn bản thuần túy thay vì văn bản có định dạng có trong các trình xử lý văn bản "What You See Is What You Get" như Microsoft Word, LibreOffice Writer và Apple Pages. Người viết sử dụng các quy ước gán thẻ ngôn ngữ đánh dấu để xác định cấu trúc chung của tài liệu (chẳng hạn như bài báo, sách và thư), để cách điệu văn bản trong toàn bộ tài liệu (chẳng hạn như in đậm và in nghiêng), thêm trích dẫn và tham chiếu chéo. Các bản phân phối của TeX như TeX Live hoặc MikTeX được sử dụng để tạo tệp đầu ra (chẳng hạn như PDF hoặc DVI) phù hợp để in hoặc phân phối kỹ thuật số.

LaTeX được sử dụng rộng rãi trong học viện để truyền thông và xuất bản các tài liệu khoa học trong nhiều lĩnh vực, bao gồm toán học, thống kê, khoa học máy

tính, kỹ thuật, vật lý, kinh tế học, ngôn ngữ học, tâm lý học định lượng, triết học và khoa học chính trị. Nó cũng có một vai trò nổi bật trong việc chuẩn bị và xuất bản các sách và bài báo chứa các tài liệu đa ngôn ngữ phức tạp, chẳng hạn như tiếng Phạn và tiếng Hy Lạp. LaTeX sử dụng chương trình sáp chữ TeX để định dạng đầu ra của nó và bản thân nó được viết bằng ngôn ngữ macro TeX.

LaTeX có thể được sử dụng như một hệ thống chuẩn bị tài liệu độc lập hoặc như một định dạng trung gian. Ví dụ, ở vai trò thứ hai, nó đôi khi được sử dụng để dịch DocBook và các định dạng dựa trên XML khác sang PDF. Hệ thống sáp chữ cung cấp các tính năng xuất bản trên máy tính để bàn có thể lập trình và các phương tiện mở rộng để tự động hóa hầu hết các khía cạnh của việc sáp chữ và xuất bản trên máy tính để bàn, bao gồm đánh số và tham chiếu chéo các bảng và hình, tiêu đề chương và phần, bao gồm đồ họa, bố cục trang, lập chỉ mục và thư mục.

Giống như TeX, LaTeX bắt đầu như một công cụ soạn thảo cho các nhà toán học và khoa học máy tính, nhưng ngay từ khi mới phát triển, nó cũng đã được các học giả sử dụng, những người cần viết các tài liệu bao gồm các biểu thức toán học phức tạp hoặc các chữ viết không phải Latinh, chẳng hạn như tiếng Ả Rập, Devanagari và tiếng Trung.

LaTeX nhằm cung cấp một ngôn ngữ đánh dấu mô tả, cấp độ cao giúp tiếp cận giá trị của TeX theo cách dễ dàng hơn cho người viết. Về bản chất, TeX xử lý mặt bối cục, trong khi LaTeX xử lý mặt nội dung để xử lý tài liệu. LaTeX bao gồm một bộ sưu tập các macro TeX và một chương trình để xử lý tài liệu LaTeX và vì các lệnh định dạng TeX thuần túy là cơ bản, nó cung cấp cho tác giả các lệnh sẵn sàng cho các yêu cầu về định dạng và bố cục như tiêu đề chương, chú thích cuối trang, tham chiếu chéo và thư mục.

LaTeX ban đầu được viết vào đầu những năm 1980 bởi Leslie Lamport tại SRI International. Phiên bản hiện tại là LaTeX2e (cách điệu là LATEX2 ε). LaTeX là miễn phí và được phân phối theo Giấy phép Công cộng Dự án LaTeX (LPPL).

Ví dụ dưới đây cho thấy đầu vào LaTeX và đầu ra tương ứng:

+ Đầu vào:

```

\documentclass{article} % Bắt đầu một bài báo
\usepackage{amsmath} % Khai báo gói lệnh làm việc trong môi
                     % trường toán học
\title{\LaTeX} % Tiêu đề

\begin{document} % Bắt đầu một tài liệu
\maketitle
\LaTeX{} is a document preparation system for
the \TeX{} typesetting program. It offers
programmable desktop publishing features and
extensive facilities for automating most
aspects of typesetting and desktop publishing,
including numbering and cross-referencing,
tables and figures, page layout,
bibliographies, and much more. \LaTeX{} was
originally written in 1984 by Leslie Lamport
and has become the dominant method for using
\TeX; few people write in plain \TeX{} anymore.
The current version is \LaTeXe.

% Đây là nhận xét, không hiển thị trong đầu ra
% Phần dưới cho thấy khả năng sắp xếp các biểu thức của LaTeX:
\begin{align}
E_0 &= mc^2 \\
E &= \frac{mc^2}{\sqrt{1-\frac{v^2}{c^2}}}
\end{align}
\end{document}

```

Hình 10: Ví dụ đầu vào của ngôn ngữ LaTeX

+ Đầu ra:

L^AT_EX

L^AT_EX is a document preparation system for the T_EX typesetting program. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout, bibliographies, and much more. L^AT_EX was originally written in 1984 by Leslie Lamport and has become the dominant method for using T_EX; few people write in plain T_EX anymore. The current version is L^AT_EX 2 _{ε} .

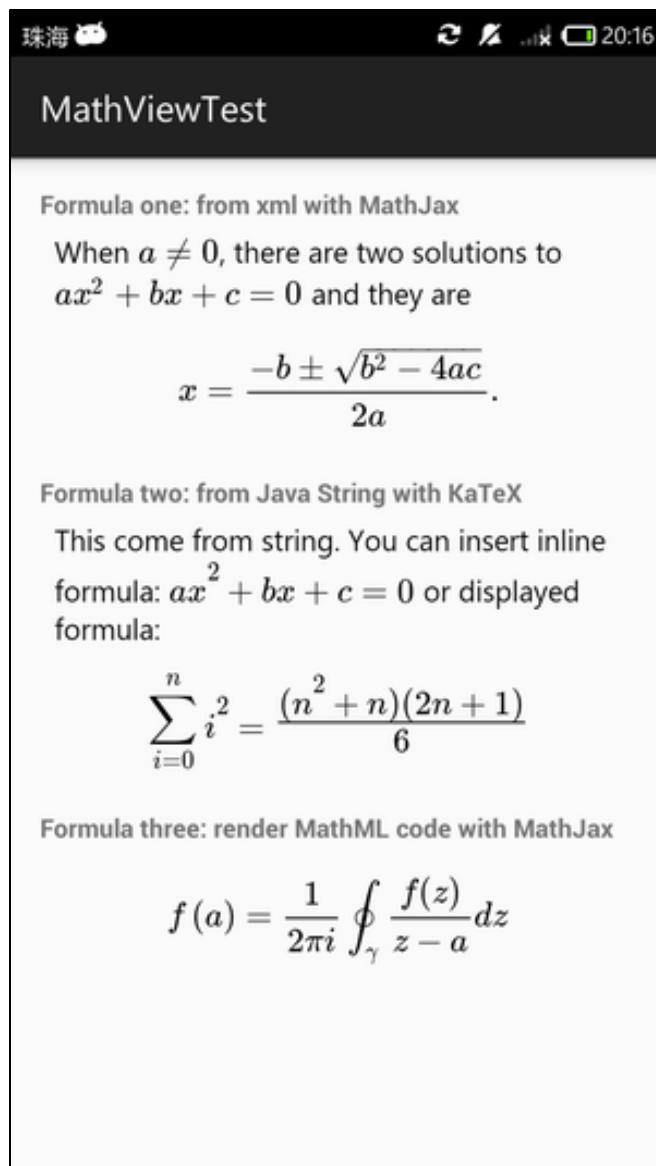
$$E_0 = mc^2 \quad (1)$$

$$E = \frac{mc^2}{\sqrt{1 - \frac{v^2}{c^2}}} \quad (2)$$

Hình 11: Kết quả đầu ra của ví dụ

2.8.2. Thư viện MathView

MathView là 1 thư viện bên thứ 3 hỗ trợ việc hiển thị văn bản LaTeX, cụ thể trong đê tài này là các công thức, biểu thức toán học lên trên các ứng dụng Android. Hiện tại, MathView hỗ trợ 2 engines là MathJax và KaTex. MathView tương thích với Android version 4.1 (Jelly Bean) trở lên



Hình 12: Giới thiệu thư viện MathView

Cách hoạt động của MathView gần giống như TextView, ngoại trừ việc nó có thể tự động hiển thị mã Tex (hoặc MathML) thành công thức toán học.

2.9. Tìm hiểu hệ quản trị cơ sở dữ liệu SQLite

2.9.1. Giới thiệu SQLite

SQLite là một cơ sở dữ liệu SQL mã nguồn mở, nó lưu trữ dữ liệu vào một tập tin văn bản trên một thiết bị. Nó mặc định đã được tích hợp trên thiết bị Android và không cần cài đặt gì thêm. Để truy cập dữ liệu này, bạn không cần phải thiết lập bất

kỳ loại kết nối nào cho nó như JDBC, ODBC, ... SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C.



Hình 13: Giới thiệu cơ sở dữ liệu SQLite

SQLite là một thư viện phần mềm mà triển khai một SQL Database Engine truyền thống, không cần mô hình client-server nên rất nhỏ gọn. SQLite thường được sử dụng vào rất nhiều chương trình từ desktop đến mobile hay là website.

2.9.2. Ưu điểm của SQLite

SQLite có các ưu điểm sau:

- SQLite không cần mô hình client – server để hoạt động.
- Với SQLite, database được lưu trữ trên một tập tin duy nhất.
- Các thao tác dữ liệu trên SQLite chạy nhanh hơn so với các hệ quản trị cơ sở dữ liệu theo mô hình client – server.
- SQLite rất đơn giản và dễ dàng sử dụng.
- Tin cậy: các hoạt động transaction (chuyển giao) trong cơ sở dữ liệu được thực hiện trọn vẹn, không gây lỗi khi xảy ra sự cố phần cứng.
- Tuân theo chuẩn SQL92 (chỉ có một vài đặc điểm không hỗ trợ).
- Không cần cài đặt cấu hình.
- Kích thước chương trình gọn nhẹ, với cấu hình đầy đủ chỉ không đầy 300 kB.
- Không cần phần mềm phụ trợ.

- Phần mềm tự do với mã nguồn mở, được chú thích rõ ràng.
- Với đặc tính nhỏ gọn, truy xuất dữ liệu nhanh SQLite thường được sử dụng để nhúng vào các dự án.

2.9.3. Nhược điểm của SQLite

Ngoài những ưu điểm đã kể ra ở trên thì SQLite cũng có một số mặt hạn chế nếu đem so sánh với các hệ quản trị khác.

Do sử dụng cơ chế riêng biệt nên trong cùng một thời điểm SQLite có thể hỗ trợ nhiều người đọc dữ liệu, nhưng chỉ có 1 người có thể ghi dữ liệu.

Trong thời đại bùng nổ thông tin như hiện nay, SQLite không phải là lựa chọn hoàn hảo để đáp ứng các nhu cầu xử lý trên một khối lượng dữ liệu lớn, phát sinh liên tục.

2.9.4. SQLite Commands

Các lệnh SQLite tiêu chuẩn để tương tác với cơ sở dữ liệu quan hệ tương tự như SQL. Chúng là CREATE, SELECT, INSERT, UPDATE, DELETE và DROP.

Các lệnh này có thể được phân loại thành các nhóm dựa trên tính chất hoạt động của chúng như sau:

- DDL - Ngôn ngữ định nghĩa dữ liệu (Data Definition Language)

Lệnh	Mô tả
CREATE	Tạo mới một bảng, view của bảng hoặc đối tượng khác trong cơ sở dữ liệu.
ALTER	Sửa đổi một đối tượng cơ sở dữ liệu đang tồn tại, chẳng hạn như bảng.
DROP	Xóa toàn bộ bảng, view của bảng hoặc đối tượng khác trong cơ sở dữ liệu.

Bảng 1: Các lệnh định nghĩa dữ liệu của SQLite

- DML - Ngôn ngữ thao tác dữ liệu (Data Manipulation Language)

Lệnh	Mô tả
INSERT	Tạo một bảng ghi (record)
UPDATE	Sửa một bảng ghi (record)

DELETE	Xóa một bảng ghi (record)
--------	---------------------------

Bảng 2: Các lệnh thao tác dữ liệu của SQLite

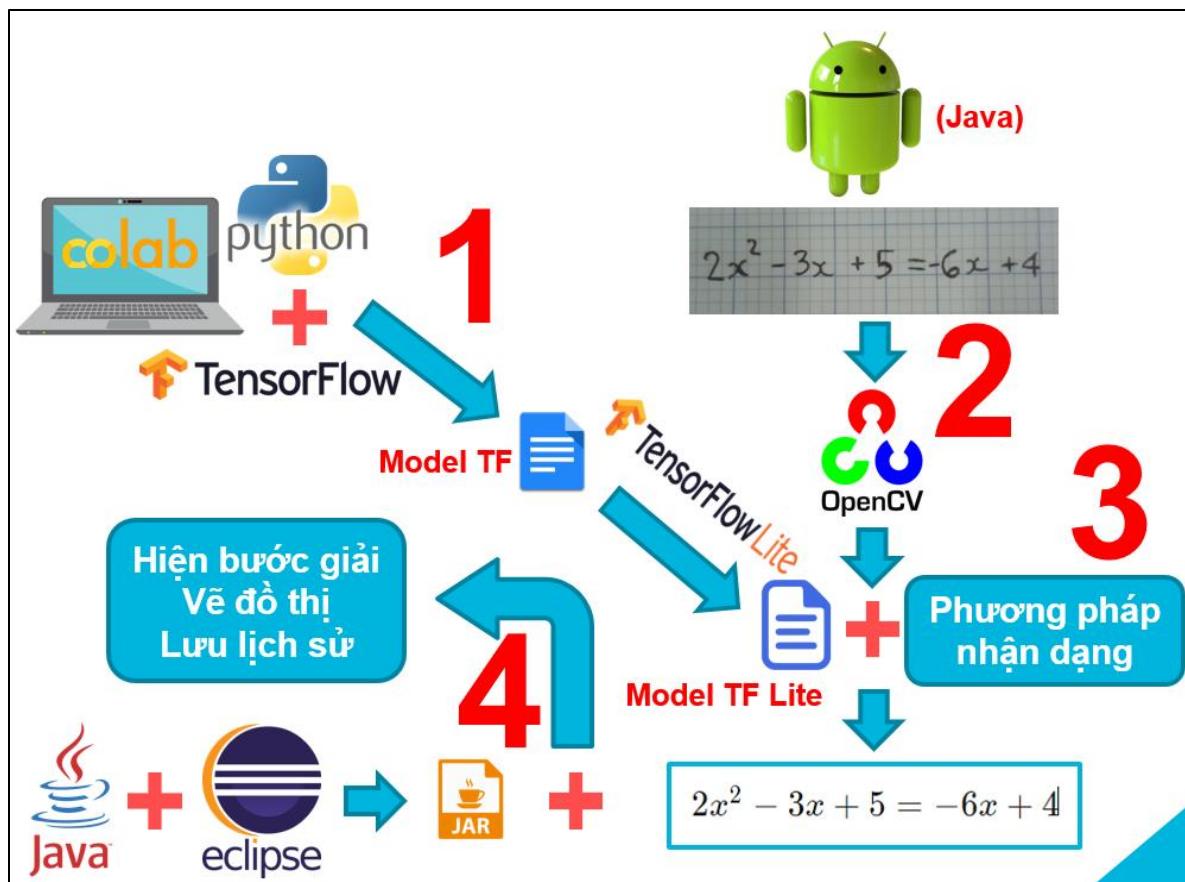
- DQL - Ngôn ngữ truy vấn dữ liệu (Data Query Language)

Lệnh	Mô tả
SELECT	Lấy một số bảng ghi nhất định từ một hoặc nhiều bảng (record)

Bảng 3: Các lệnh truy vấn dữ liệu của SQLite

CHƯƠNG 3. TRIỂN KHAI THỰC HIỆN

3.1. Sơ đồ của hệ thống



Hình 14: Sơ đồ của hệ thống

- ❖ Giai đoạn 1: Xây dựng mô hình nhận dạng ký tự với mạng CNN bằng Tensorflow, sau đó chuyển đổi về mô hình Tensorflow Lite để chuẩn bị cho việc tích hợp lên ứng dụng di động.
 - Ngôn ngữ lập trình: Python
 - Thư viện ML: Tensorflow
 - Môi trường thực thi: Google Colab

- ❖ Giai đoạn 2: Xây dựng ứng dụng di động cho phép đọc ảnh từ camera hoặc thư viện ảnh của điện thoại, sau đó sử dụng OpenCV để tiền xử lý ảnh, phân tách biểu thức thành các ký tự riêng lẻ.
 - Ngôn ngữ lập trình: Java

- Thư viện cắt ảnh: Crop Image
- Thư viện xử lý ảnh: OpenCV
- Môi trường thực thi: Android

❖ **Giai đoạn 3: Nhận dạng các ký tự đã tách và kết nối lại cho phù hợp tạo thành một biểu thức hoàn chỉnh.**

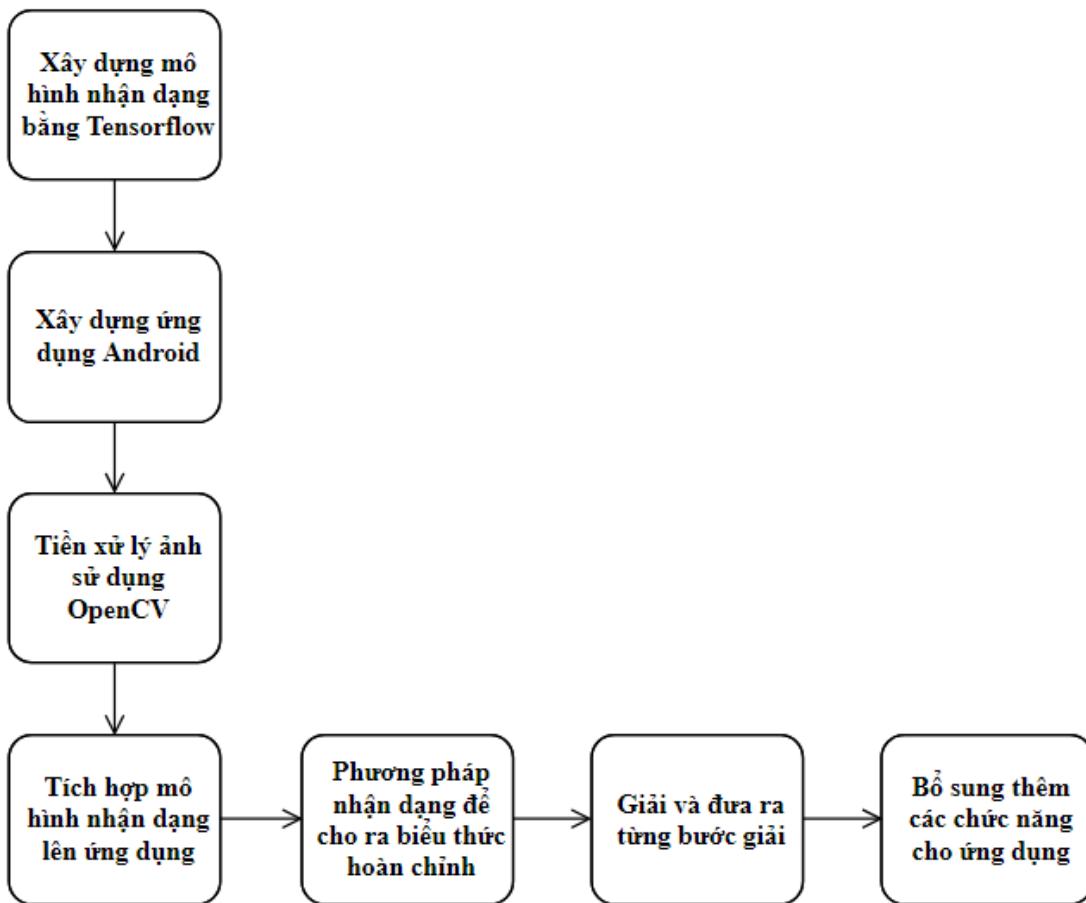
- Ngôn ngữ lập trình: Java
- Thư viện dùng để chạy mô hình nhận dạng: Tensorflow Lite
- Môi trường thực thi: Android

❖ **Giai đoạn 4: Viết mã giải toán, xuất ra dưới dạng thư viện có đuôi .jar và tích hợp lên ứng dụng di động để giải biểu thức có được ở Giai đoạn 3. Giải và hiển thị từng bước giải, lưu lại lịch sử biểu thức để người dùng có thể truy cập lại sau này, vẽ đồ thị phương trình, hệ phương trình, tích hợp nhận dạng giọng nói.**

- Ngôn ngữ lập trình: Java
- Công cụ viết mã và xuất file jar: Eclipse
- Thư viện vẽ đồ thị: Graphlib
- Thư viện hiển thị biểu thức toán học: MathView
- Cơ sở dữ liệu lưu lịch sử: SQLite
- Thư viện nhận dạng giọng nói: Google API

3.2. Lưu đồ giải thuật của hệ thống

Tuần tự các bước xây dựng đề tài được thực hiện theo sơ đồ dưới đây:

*Hình 15: Lưu đồ giải thuật của hệ thống*

3.3. Xây dựng mô hình nhận dạng bằng Tensorflow

Như đã được giới thiệu ở mục 2.4, Tensorflow là thư viện mạnh mẽ được hỗ trợ bởi Google dùng để xây dựng các mô hình Deep Learning. Ngoài Tensorflow, còn một số thư viện khác như Pytorch, MXNet, Theano, Caffe2 cũng hỗ trợ xây dựng mô hình nhưng chúng tôi quyết định dùng Tensorflow vì một lý do vô cùng thiết thực và quan trọng: Tensorflow hỗ trợ chuyển đổi mô hình đã xây dựng thành Tensorflow Lite để chạy được trên thiết bị di động tốt hơn so với các thư viện khác.

Trong mục này, chúng tôi sẽ trình bày quá trình xây dựng mạng CNN để nhận dạng ký tự từ việc tìm data set đến khi xây dựng hoàn tất cho ra mô hình hoàn chỉnh.

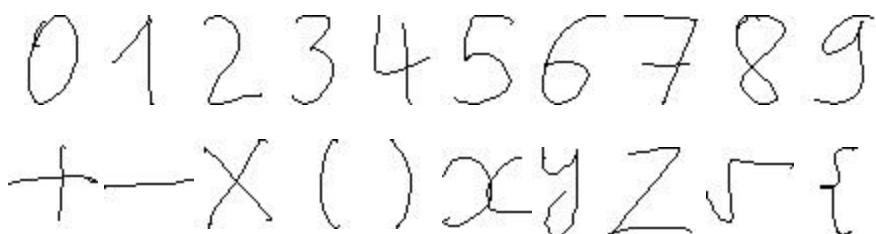
3.3.1. Giới thiệu data set

Dữ liệu được sử dụng trong đề tài này được trích từ tập dữ liệu Handwritten math symbols của Kaggle:

<https://www.kaggle.com/xainano/handwrittenmathsymbols>

Tập dữ liệu trên gồm 375.974 ảnh jpg cỡ 45x45 của 82 lớp như: các số từ 0 đến 9, +, -, times, div, sin, cos, tan, sqrt, x, y, z, (,), {}, ...

Đề tài này sẽ sử dụng 20 lớp (gồm 70875 ảnh) trong tổng số 82 lớp của tập dữ liệu trên, bao gồm các ký tự sau: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, x, (,), chữ x, y, z, căn và {



Hình 16: Các ký tự dùng để huấn luyện mô hình nhận dạng

3.3.2. Xử lý data set

Dữ liệu 20 lớp với 70875 ảnh 45x45, chữ đen nền trắng sẽ được chia thành 2 tập dữ liệu nhỏ là tập huấn luyện (train) chiếm 80% và tập xác nhận (validation) chiếm 20% số lượng ảnh của mỗi lớp.

Ảnh trước khi đưa vào huấn luyện (training) sẽ được chuyển về thang ảnh xám (grayscale). Mục đích của việc này là bởi vì khi chuyển về ảnh xám, sẽ giảm số lượng phép tính mà máy tính cần phải xử lý trong khi ảnh xám vẫn giữ được đặc trưng của ảnh nên không làm ảnh hưởng đến độ chính xác của mô hình.

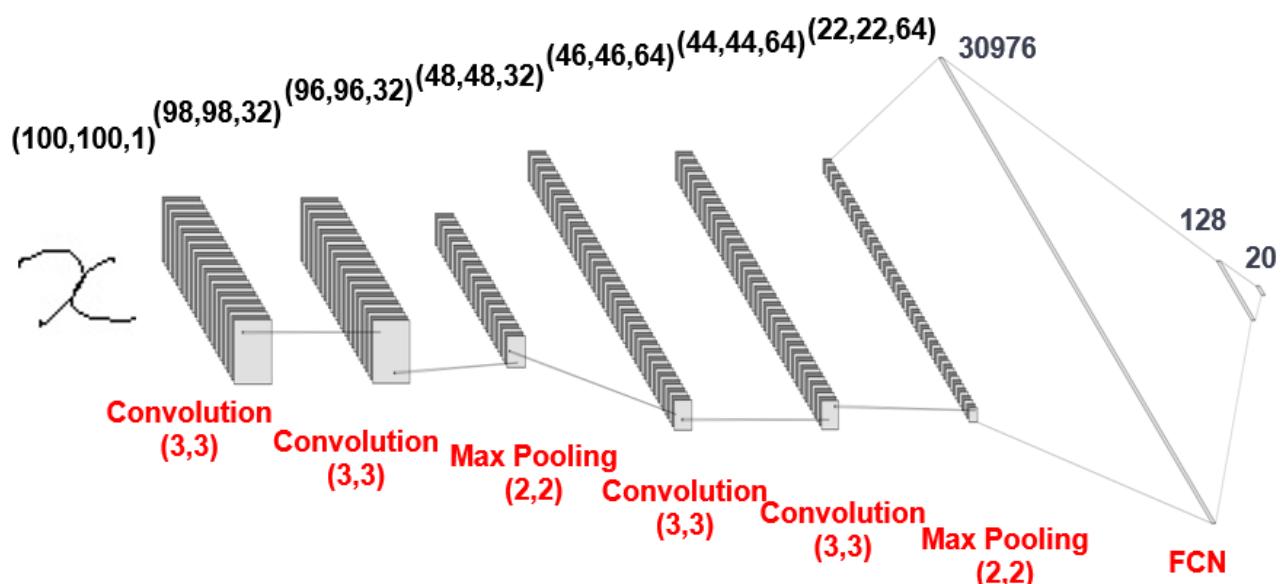
Đồng thời, ta cũng sẽ chỉnh lại kích cỡ ảnh (resize) từ 45x45 thành 100x100 vì trong quá trình học, dữ liệu ở phần biên ảnh thường bị mất nên tăng cỡ ảnh sẽ làm giảm lượng điểm ảnh bị mất liên quan đến ký tự. Cỡ 100x100 đã được chúng tôi thử nghiệm và cho độ chính xác cao nhất so với các kích cỡ khác như: 50x50, 60x60, ...

Bên cạnh đó, ta sẽ chuẩn hóa dữ liệu về đoạn [0,1] để dữ liệu được đồng nhất, tạo điều kiện tốt nhất để huấn luyện ra mô hình có độ chính xác cao.

3.3.3. Mô hình CNN

* Đây là bài toán phân loại ảnh nên mô hình phù hợp nhất mà chúng tôi dùng là Mạng neuron tích chập (CNN).

* Vì để xây dựng được một mô hình có khả năng nhận diện tốt đòi hỏi rất nhiều kiến thức chuyên sâu về lý thuyết toán nên bằng cách sử dụng kiến thức có được trong khả năng của bản thân, chúng tôi đã cát công tìm hiểu, xây dựng, trải qua rất nhiều lần huấn luyện và thử nghiệm để cuối cùng có được một mô hình như hình dưới đây:



Hình 17: Kiến trúc mạng neuron dùng để huấn luyện

* Mô hình CNN chúng tôi xây dựng bên trên được mô tả như sau:

- Bước 1: Dữ liệu ảnh với cỡ 100x100, số kênh màu là 1 (do đã được chuyển về thang màu xám ở bước Xử lý data set) sẽ đi qua lớp tích chập (convolution) đầu tiên để trích xuất đặc trưng của ảnh, lớp này có 32 kernel và kích cỡ của mỗi kernel là 3x3, tức là một ma trận có 3 hàng và 3 cột. Ma trận này sẽ trượt trên ảnh để tính toán lấy ra giá trị đặc trưng của ảnh. Quá trình trích xuất đặc trưng thực chất là đang nhân hai ma trận, một ma trận của phần ảnh mà đang được kernel đè lên (apply) và ma trận còn lại chính là kernel. Đầu ra (output) của lớp này sẽ là ảnh có kích thước 98x98, với số kênh màu (channel) là 32 (do lớp tích chập này có 32 kernel). Lý do ảnh từ 100x100 xuống còn 98x98 là bởi vì chúng tôi không cài đặt tham số padding cho ảnh nên qua lớp tích chập với padding mặc định bằng 1, ảnh sẽ bị mất 1 pixel ở bốn cạnh của ảnh. Ở lớp này, hàm kích hoạt (activation) được dùng là ReLU (Rectified Linear Unit).

- Bước 2: Tiếp theo, dữ liệu đầu ra của lớp tích chập đầu tiên sẽ được đi qua lớp tích chập thứ hai, cũng với các thông số là 32 kernel cỡ 3×3 . Đầu ra là ảnh $(96, 96, 32)$. Hàm kích hoạt ở lớp này vẫn sẽ là ReLU.

- Bước 3: Đầu ra của lớp tích chập thứ hai sẽ làm đầu vào của lớp Max Pooling. Với thông số được cho là 2×2 , lớp này có tác dụng làm giảm kích thước của ảnh xuống còn phân nửa so với ảnh ban đầu trong khi số kênh màu của ảnh vẫn được giữ nguyên nên đầu ra của lớp này sẽ là ảnh $(48, 48, 32)$. Giống như tên gọi, Max Pooling sẽ giảm kích thước ảnh bằng cách chỉ giữ lại giá trị lớn nhất (đặc trưng tốt nhất) của ảnh.

- Bước 4: Sau khi qua lớp Max Pooling, ảnh đã được thu nhỏ nhưng vẫn giữ được các đặc trưng tốt nhất cho việc nhận dạng, tiếp đến các ảnh này sẽ qua tiếp lớp tích chập với 64 kernel cỡ 3×3 . Đầu ra là dữ liệu ảnh có kích thước $(46, 46, 64)$. Lớp này sẽ dùng hàm kích hoạt là ReLU.

- Bước 5: Dữ liệu sẽ tiếp tục đi qua một lớp tích chập nữa với 64 kernel cỡ 3×3 , hàm phi tuyến ReLU vẫn được dùng ở lớp này. Đầu ra của lớp sẽ là ảnh $(44, 44, 64)$.

- Bước 6: Lớp Max Pooling tiếp theo với tham số 2×2 sẽ giảm cỡ ảnh từ $(44, 44, 64)$ xuống còn $(22, 22, 64)$.

- Bước 7: Sau lớp Max Pooling trên, quá trình trích xuất đặc trưng (feature extraction) kết thúc. Dữ liệu ảnh $(22, 22, 64)$ sẽ là ảnh còn giữ lại được những đặc trưng tốt nhất, thích hợp cho việc phân loại. Dữ liệu này sẽ đi vào phần kết nối đầy đủ (FCN - fully connected network), phần này có nhiệm vụ phân loại ảnh sau khi đi qua trình trích xuất đặc trưng. Trong phần phân loại, đầu tiên dữ liệu sẽ được làm phẳng (Flatten) thành $22 \times 22 \times 64 = 30976$ lớp.

- Bước 8: Thông qua lớp Dense với hàm kích hoạt ReLU, 30976 lớp này sẽ được thu nhỏ xuống còn 128 lớp.

- Bước 9: Và qua lớp Dense cuối cùng, 128 lớp được thu nhỏ thành 20 lớp. Đây chính là số lớp của dữ liệu ảnh mà chúng tôi đã đề cập ở mục Giới thiệu data set. Hàm kích hoạt được dùng ở lớp này là softmax, dùng hàm này vì nó sẽ cho ra xác suất nhận dạng của từng lớp.

- Ngoài ra, trong mô hình, sau mỗi lớp Max Pooling, chúng tôi sử dụng dropout 0.25 và sau lớp Dense đầu tiên, chúng tôi sử dụng dropout 0.5. Mục đích của việc này là để tránh mô hình quá khớp (overfitting).

* Số lượng tham số khi huấn luyện mô hình được biểu diễn như hình dưới đây:

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 98, 98, 32)	320
conv2d_1 (Conv2D)	(None, 96, 96, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 48, 48, 32)	0
dropout (Dropout)	(None, 48, 48, 32)	0
conv2d_2 (Conv2D)	(None, 46, 46, 64)	18496
conv2d_3 (Conv2D)	(None, 44, 44, 64)	36928
max_pooling2d_1 (MaxPooling2 (None, 22, 22, 64)		0
dropout_1 (Dropout)	(None, 22, 22, 64)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 128)	3965056
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 20)	2580
<hr/>		
Total params:	4,032,628	
Trainable params:	4,032,628	
Non-trainable params:	0	

Hình 18: Các tham số của mô hình

- Tổng số lượng params của mô hình là 4032628. Ngoài các lớp không phải học (Max Pooling, Dropout, Flatten) thì không có params, các lớp còn lại đều phải học với số lượng params được biểu diễn như trong hình.

3.3.4. Huấn luyện mô hình

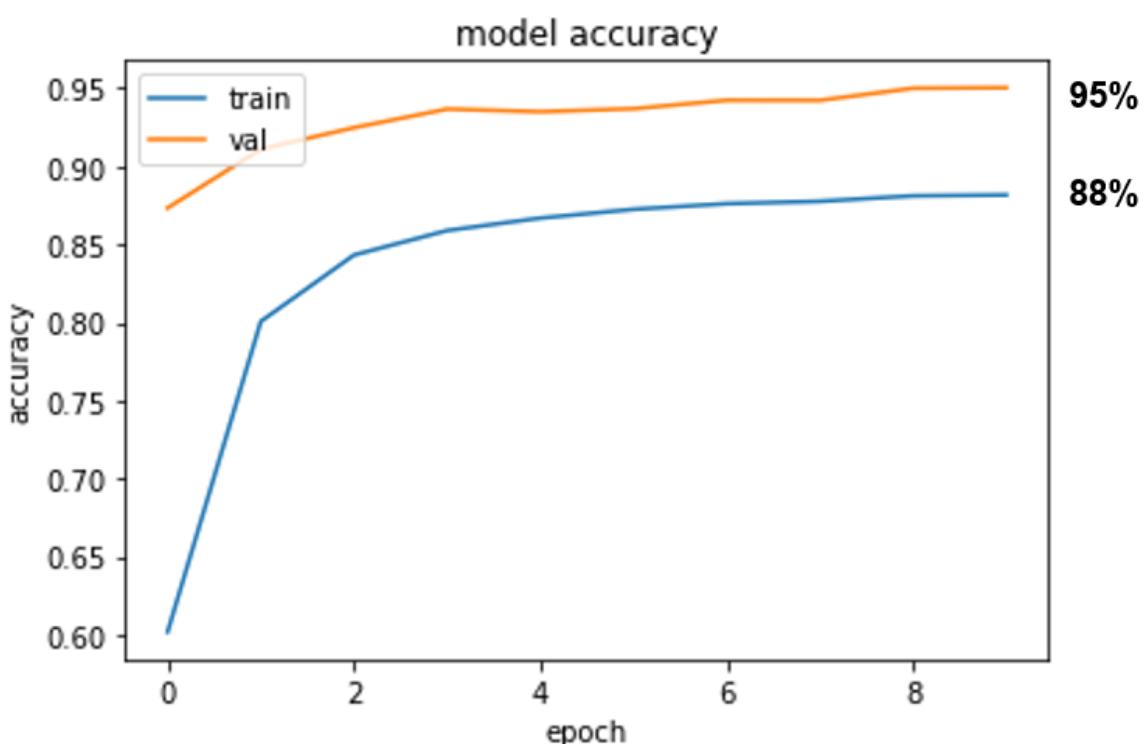
- Sau khi có được dữ liệu (80% train; 20% validation) và thiết lập xong mô hình CNN, ta đến bước quan trọng tiếp theo là huấn luyện mô hình bằng cách sử dụng ngôn ngữ Python và thư viện Tensorflow.

- Hàm tối ưu (optimization) được dùng là RMSprop (Root Mean Square prop)

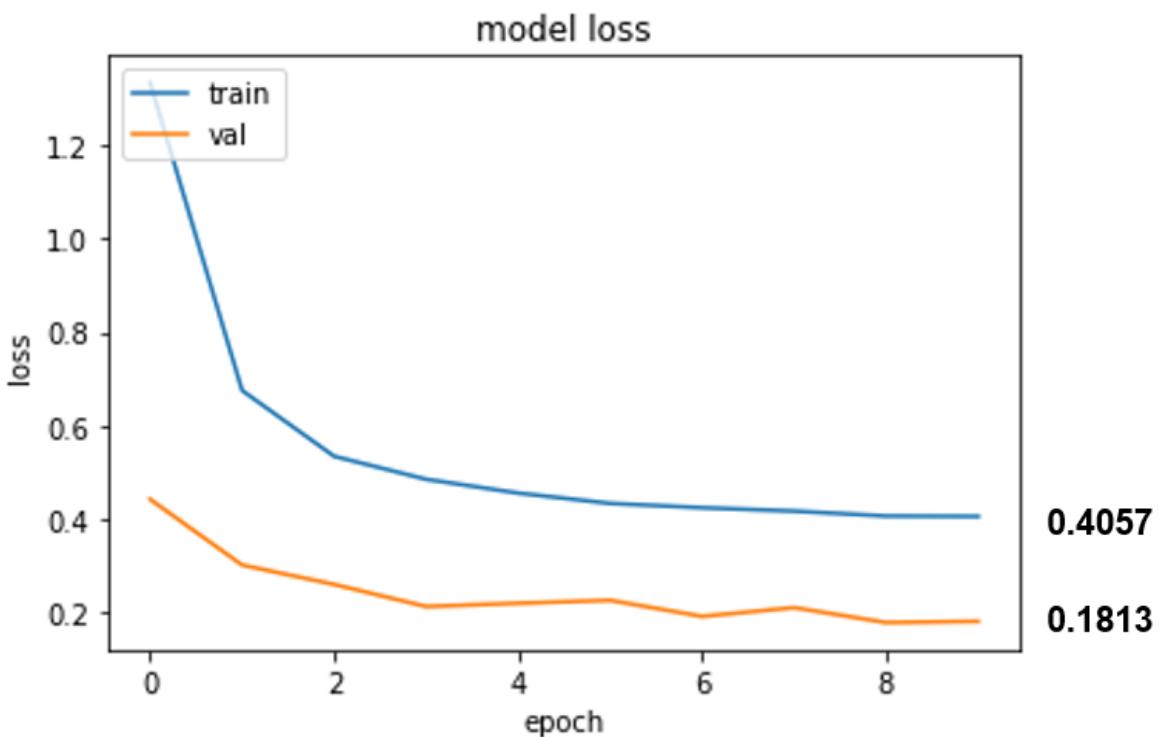
- Vì dữ liệu của chúng ta là đa lớp nên hàm mất mát (loss function) thích hợp khi huấn luyện mô hình là categorical cross entropy.

- Mô hình được huấn luyện 10 epochs (mỗi epoch là một lần mô hình học qua hết toàn bộ dữ liệu, 10 epochs nghĩa là mô hình học qua tập dữ liệu 10 lần).

- Sau khi huấn luyện ta sẽ có được mô hình Tensorflow với độ chính xác (accuracy) và giá trị mất mát như sau:



Hình 19: Độ chính xác của mô hình



Hình 20: Độ mất mát của mô hình

=> Quan sát 2 hình bên trên, ta có thể thấy mô hình nhận dạng đạt độ chính xác khoảng 88% trên tập train, và 95% trên tập validation. Giá trị mất mát trên tập train là 0.4057 và tập validation là 0.1813. Như vậy, mô hình có khả năng nhận diện khá tốt.

=> Bên cạnh đó, ta cũng dễ dàng nhận thấy từ khoảng epoch thứ 8 đến 10 trở đi thì mô hình dần ổn định nên dù có huấn luyện thêm nhiều epoch nữa thì độ chính xác của mô hình sẽ không tăng hoặc chỉ tăng rất ít. Và đó cũng chính là lý do vì sao chúng tôi chỉ huấn luyện mô hình với 10 epochs.

3.4. Xây dựng ứng dụng Android

3.4.1. Màn hình chính

Ban đầu trong mỗi ứng dụng Android sẽ có một giao diện màn hình chính để cho phép người dùng thực hiện các thao tác chung. Ở đây sẽ là nơi chứa các chức năng của toàn bộ ứng dụng. Một số chức năng đặc trưng nổi bật của ứng dụng như: chụp ảnh, tái ảnh lên từ thư viện,... Sau đây là giao diện chính của ứng dụng.

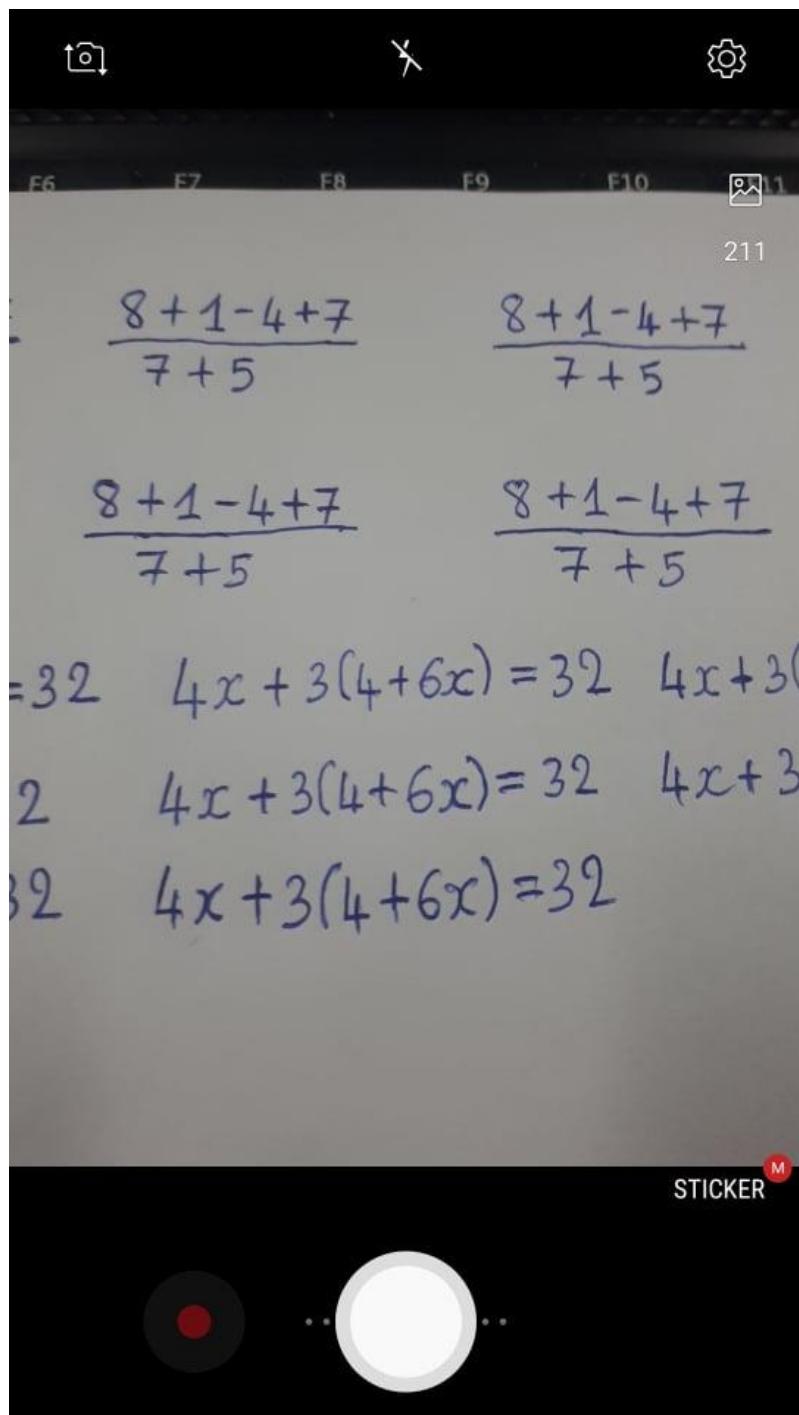


Hình 21: Giao diện màn hình chính của ứng dụng

Như màn hình phía trên thì ta thấy giao diện ban đầu bao gồm logo nằm trên cùng, bên dưới sẽ là các button cùng với các icon tương ứng với từng chức năng khác nhau.

3.4.2. Chức năng chụp ảnh

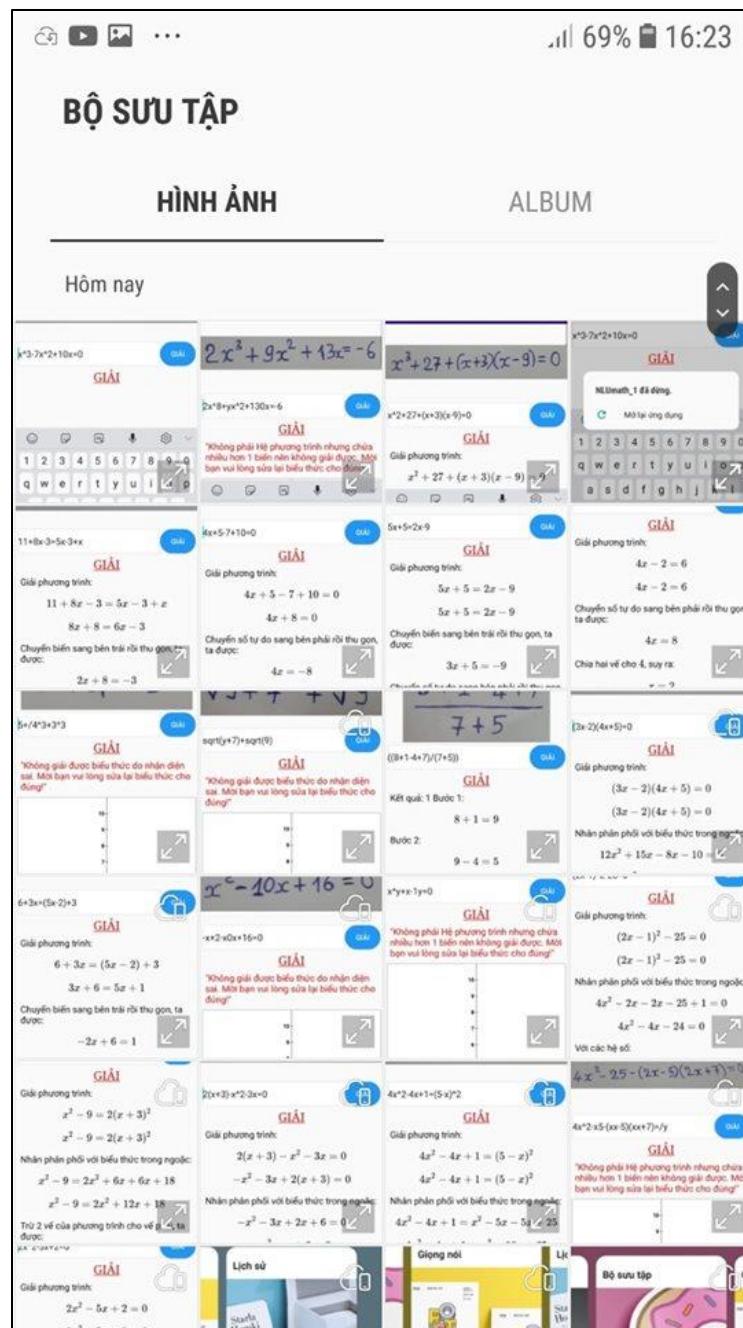
Chức năng này cho phép mở máy ảnh của điện thoại, chụp ảnh bài toán



Hình 22: Chức năng máy ảnh của ứng dụng

3.4.3. Đọc ảnh từ thư viện ảnh

Bên cạnh việc chụp ảnh bài toán, người dùng có thể tải ảnh đã có từ trước trong bộ sưu tập lên ứng dụng

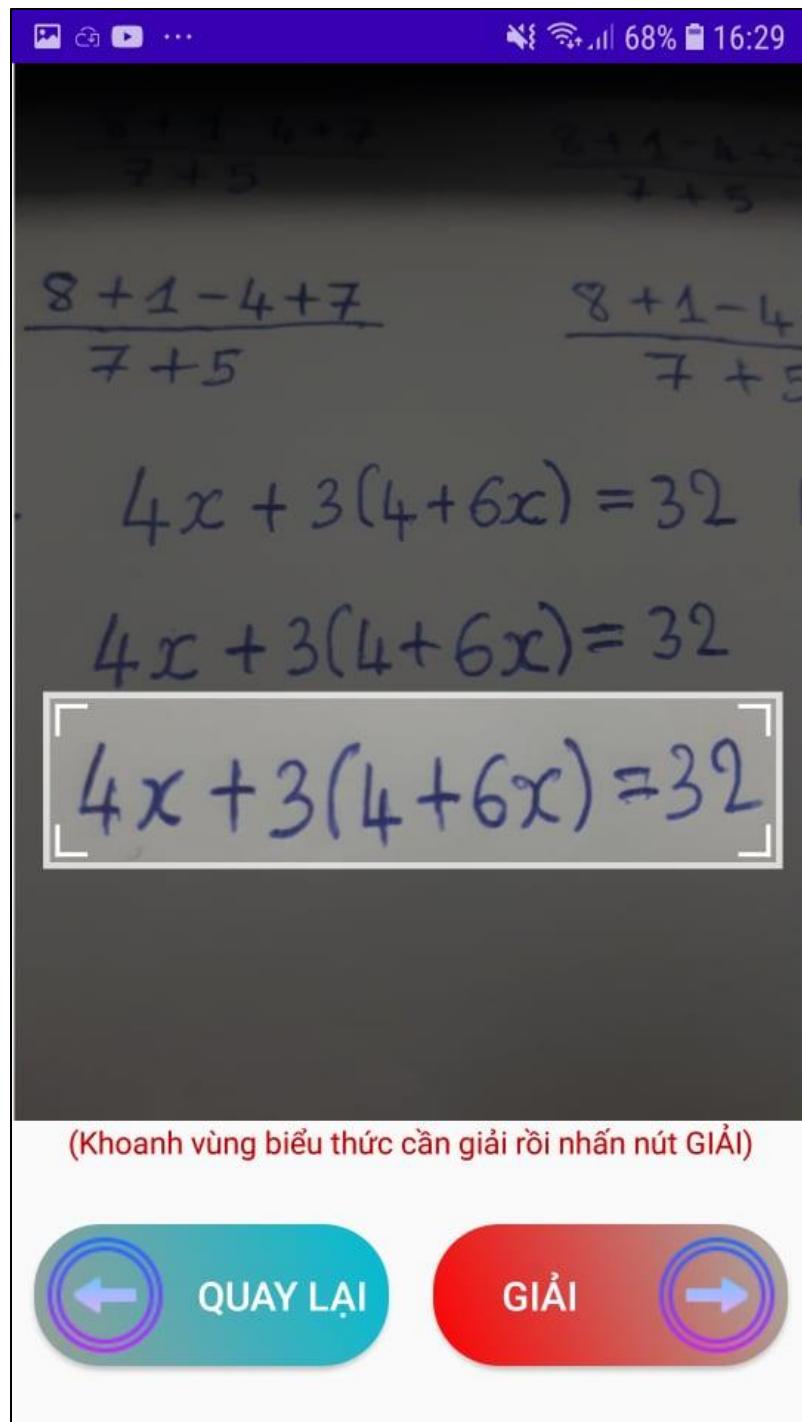


Hình 23: Chức năng đọc ảnh từ thư viện

3.4.4. Cắt ảnh

Sau khi có được ảnh bài toán bằng cách chụp ảnh hoặc tải ảnh từ thư viện, ứng dụng sẽ chuyển qua màn hình cắt ảnh. Tại đây ở giữa màn hình chính sẽ hiển thị một khung hình viền trắng cho phép di chuyển sang trái, phải, lên, xuống và thực hiện bước cắt ảnh. Bên dưới màn hình sẽ có 2 button, 2 button này được pha trộn màu sắc và kèm theo các icon mũi tên tương ứng. Nếu nhấn vào button QUAY LẠI thì sẽ

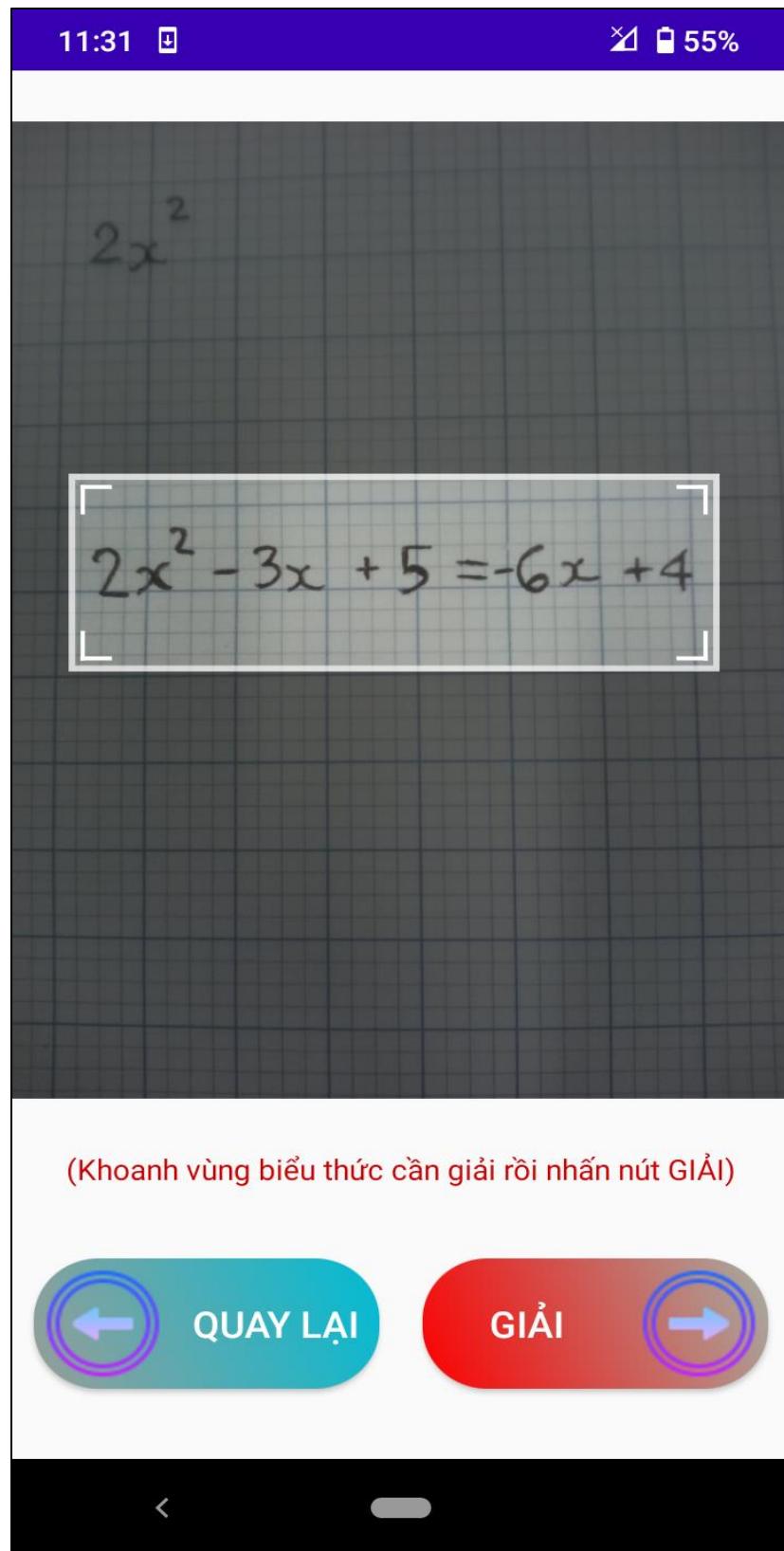
chuyển màn hình về lại trạng thái người dùng thực hiện trước đó, nếu nhấn vào button GIẢI thì sẽ mở sang màn hình hiện thị kết quả và các bước giải của bài toán:



Hình 24: Màn hình cắt ảnh biểu thức

3.5. Tiền xử lý ảnh sử dụng OpenCV

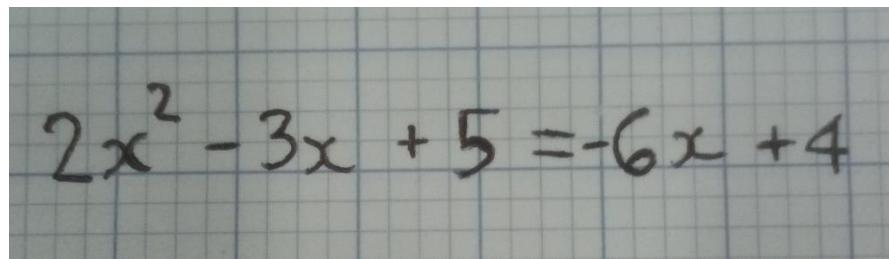
- ❖ **Ảnh sau khi có được từ camera hoặc thư viện ảnh của điện thoại sẽ được cắt bằng thư viện Crop Image:**



Hình 25: Cắt vùng ảnh chứa biểu thức

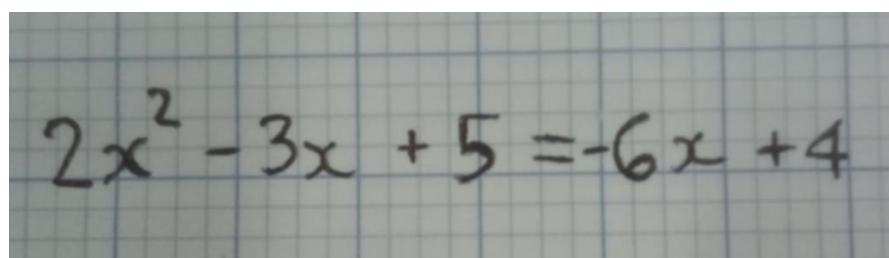
❖ Ảnh sau cắt sẽ được xử lý qua các bước sau đây để phân tách thành từng ký tự riêng lẻ:

- Ảnh gốc:



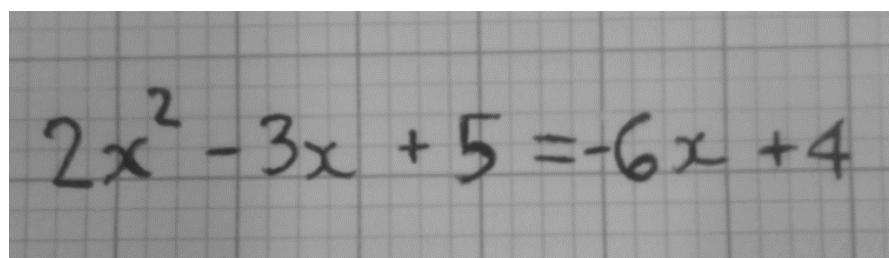
Hình 26: Ảnh gốc chứa biểu thức

- Blur ảnh để khử nhiễu:



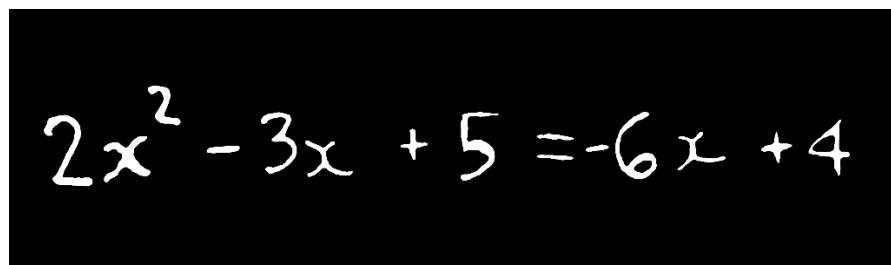
Hình 27: Ảnh sau khi khử nhiễu

- Chuyển ảnh về thang màu xám (để dễ phân ngưỡng ảnh ở bước sau):

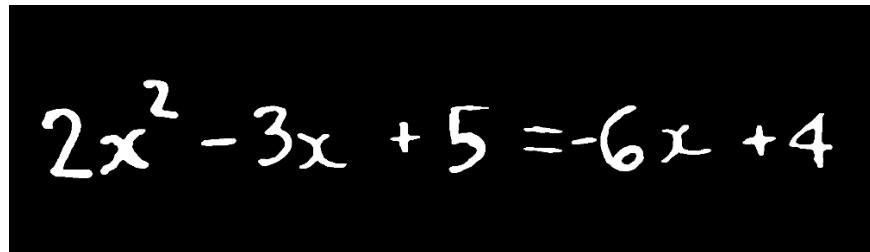


Hình 28: Ảnh xám

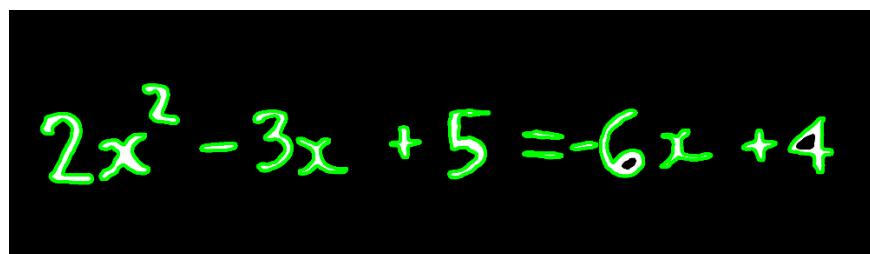
- Phân ngưỡng ảnh (ảnh sẽ ở dạng chữ trắng nền đen). Mục đích của bước này là để dễ dàng tách ký tự khỏi nền:

*Hình 29: Ảnh sau khi phân ngưỡng*

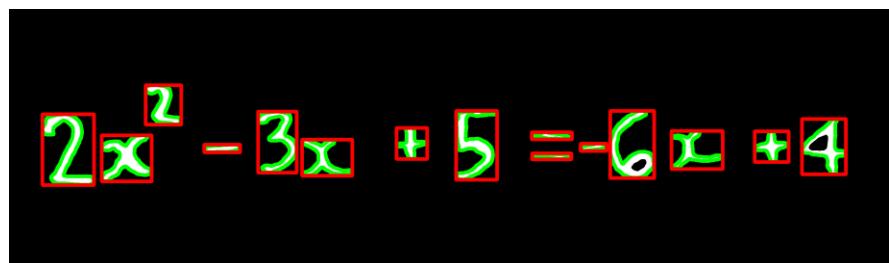
- Làm giãn nở nét vẽ ký tự (Dilate). Bước này hữu ích khi nét vẽ ký tự bị đứt (thường gặp ở bút bi) giúp gắn kết các phần bị đứt của ký tự:

*Hình 30: Ảnh sau khi làm giãn nở*

- Tìm contour cho các ký tự (contour được hiểu là một đường kín bao quanh ký tự, trong hình các contour là các đường màu xanh):

*Hình 31: Ảnh chứa hình bao (contour) của các ký tự*

- Đóng gói (Bounding) ký tự bằng hình chữ nhật nhỏ nhất mà có thể chứa vừa ký tự. Mục đích của bước này là để lát nữa, ta dễ dàng cắt các ký tự ra dựa vào vị trí và kích thước của các hình chữ nhật này:



Hình 32: Ảnh chứa các ký tự được bao đóng bởi các hình chữ nhật

- Phân tách (Segment) thành từng ký tự riêng lẻ. Vì mô hình nhận dạng đã xây dựng là dùng để nhận dạng từng ký tự nên ta cần tách từng ký tự ra khỏi biểu thức:

$2x^2 - 3x + 5 = -6x + 4$

Hình 33: Các ký tự sau khi được phân tách

cũng cần lưu ý rằng, ở bước này ảnh sau khi cắt đã được phân ngưỡng lại thành chữ đen nền trắng. Lý do là bởi vì dữ liệu trong data set là chữ đen nền trắng nên ta chuyển ký tự đã tách về cùng kiểu để kết quả nhận dạng được chính xác nhất. Còn ở bước phân ngưỡng lúc đầu được chuyển thành chữ trắng nền đen là do cách thức tìm contour của OpenCV hoạt động hiệu quả nhất với chữ trắng nền đen.

3.6. Tích hợp mô hình nhận dạng lên ứng dụng

“Có thể nói đây là bước quan trọng nhất trong đề tài của chúng tôi, việc tích hợp được mô hình nhận dạng lên thiết bị di động hoạt động offline thật sự không hề dễ dàng. Đa phần các ứng dụng nhận dạng ví dụ như nhận dạng biển số xe, nhận dạng vân tay, nhận dạng khuôn mặt,... đều được thực hiện trên desktop, còn điện thoại chỉ đóng vai trò thu nhận hình ảnh gửi đến máy chủ hoặc một ứng dụng khác trên desktop để xử lý sau đó trả kết quả về cho điện thoại. Tuy nhiên, vì đây là một ứng dụng học tập và với mong muốn người dùng có thể học mọi lúc mọi nơi, không phụ thuộc kết nối mạng, đặc biệt là trẻ em. Chúng tôi đã cố gắng để cuối cùng bước này cũng hoàn thành tốt đẹp!”

3.6.1. Chuyển đổi mô hình Tensorflow thành mô hình Tensorflow Lite

- Mô hình Tensorflow sau khi được huấn luyện sẽ được lưu thành tệp dưới dạng JSON (có phần mở rộng .json) hoặc HDF5 (có phần mở rộng .h5) tùy thuộc vào định dạng chúng ta muốn lưu. Ở đây chúng tôi sử dụng định dạng JSON để lưu mô hình.

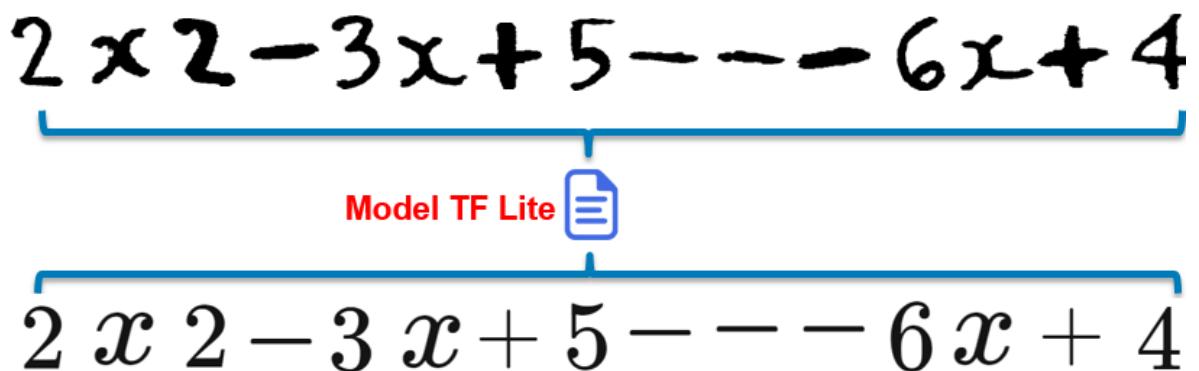
- Từ file JSON ta sẽ sử dụng gói Tensorflow Lite được tích hợp bên trong thư viện Tensorflow để chuyển đổi mô hình Tensorflow thành mô hình ở dạng Tensorflow Lite (có phần mở rộng .tflite)

- Cũng đề cập thêm là do mô hình đã được chuyển đổi nên chúng ta sẽ phải đánh đổi bằng một chút độ chính xác của mô hình, dao động khoảng 4%, tức là mô hình của chúng ta sau khi huấn luyện có độ chính xác 95% thì khi chuyển về Tensorflow Lite chạy trên di động thì độ chính xác của nó sẽ không thể còn nguyên 95% mà sẽ có thể chỉ ở khoảng 91% nhưng vẫn còn khá tốt.

3.6.2. Tích hợp mô hình lên ứng dụng

- Sau khi sao chép tệp .tflite vào dự án Android, ta sẽ sử dụng thư viện Tensorflow Lite để chạy mô hình này với dữ liệu đầu vào là các ảnh ký tự đã được tách. Và dĩ nhiên là ảnh các ký tự này phải được chỉnh lại kích thước 100x100 để phù hợp với input của mô hình mà chúng ta đã huấn luyện.

- Kết quả ví dụ sau khi chạy:



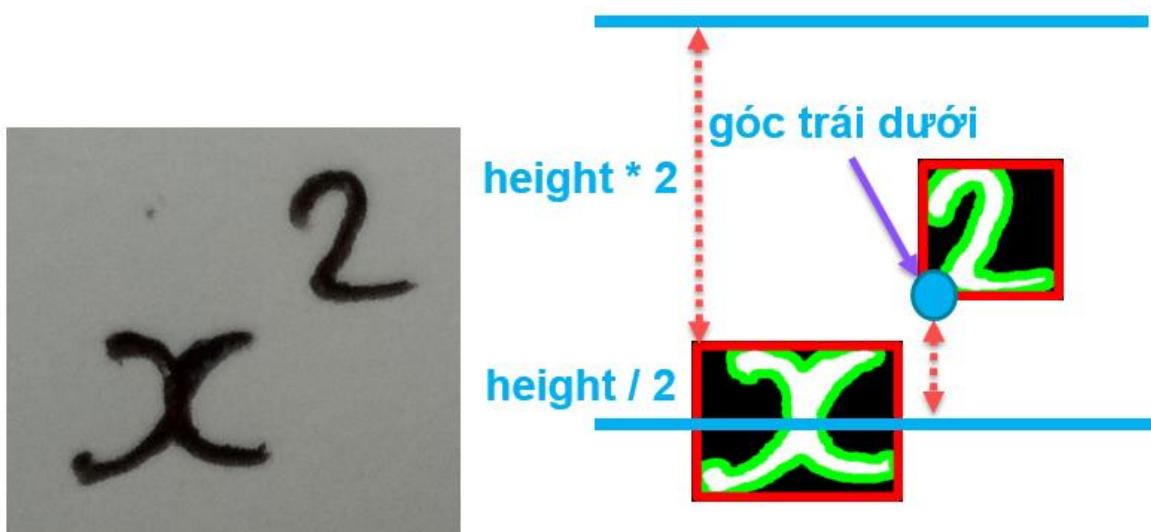
Hình 34: Nhận dạng ký tự bằng Tensorflow Lite

3.7. Phương pháp nhận dạng để cho ra biểu thức hoàn chỉnh

- ❖ Trong quá trình nhận dạng các ký tự, ta cần xác định mối quan hệ không gian của chúng để cho ra một biểu thức hoàn chỉnh.
- ❖ Sau đây là phương pháp mà chúng tôi dùng để xác định mối quan hệ giữa các ký tự:

➤ Lũy thừa:

- Ta giả sử ký tự thứ nhất có tọa độ (x_1, y_1) và kích thước $(width_1, height_1)$, ký tự thứ hai có tọa độ (x_2, y_2) và kích thước $(width_2, height_2)$
- Nếu góc trái dưới (được tính bằng $y_2 + height_2$) của ký tự thứ hai nằm trong khoảng giữa cận dưới ($y_1 - 2 * height_1$) và cận trên ($y_1 + height_1 / 2$) thì ta coi ký tự thứ hai là số mũ của ký tự thứ nhất
- Công thức: $y_1 - 2 * height_1 \leq y_2 + height_2 \leq y_1 + height_1 / 2$

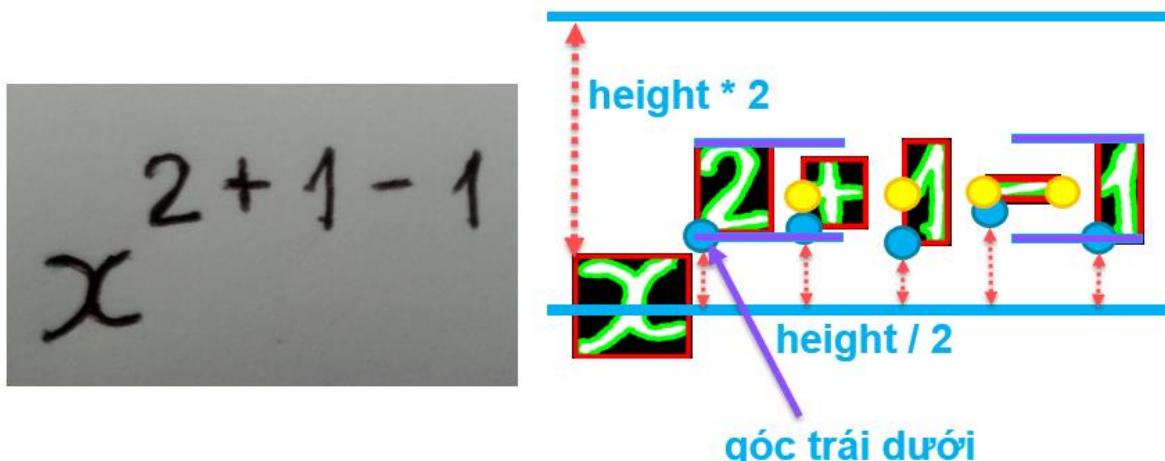


Hình 35: Nhận dạng lũy thừa

- Trong trường hợp ký tự thứ nhất là các dấu “+”, “-”, “*”, “/”, “ \leq ”, “ \geq ” thì ta không cần phải xét mũ cho nó (vì không bao giờ có lũy thừa của dấu)

- Trong trường hợp ký tự thứ hai là các dấu “*”, “/”, “=”, “)”“ thì không xét nó là mũ (vì mũ có thể bắt đầu bằng dấu “+” hoặc “-“ hoặc “(“ nhưng không thể bắt đầu bằng các dấu kể trên)

- Với trường hợp, mũ là một biểu thức như hình sau:



Hình 36: Nhận dạng lũy thừa với mũ là biểu thức

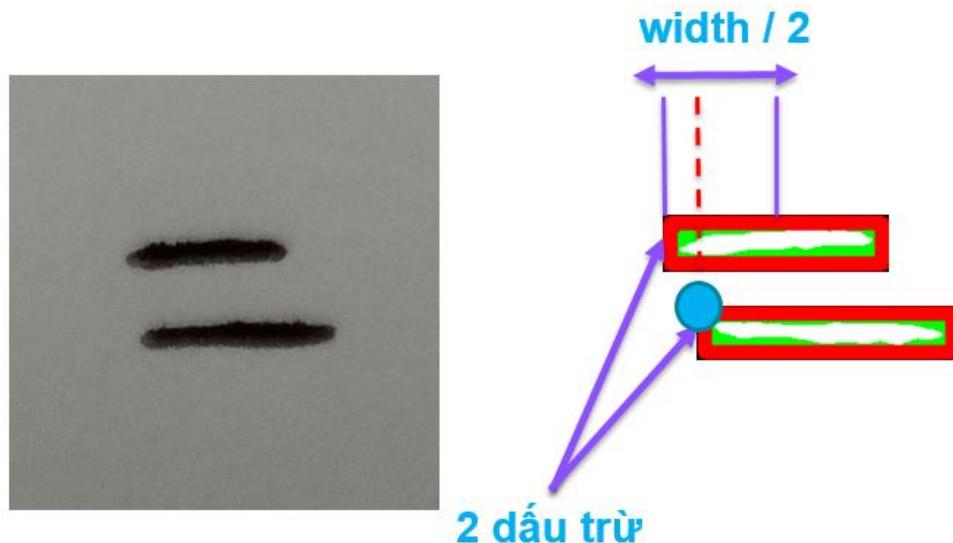
+ Ngoài việc các ký tự phải thỏa điều kiện góc trái dưới như đã nêu, để đảm bảo khả năng nhận diện chính xác, chúng tôi đã ràng buộc thêm mối liên hệ giữa các ký tự là mũ. Cụ thể, điểm giữa ($y + height / 2$) của ký tự tiếp theo là mũ phải nằm trong phạm vi với cận dưới và cận trên lần lượt là: y và $y + height$ của ký tự trước đó. Như ví dụ trong hình, điểm giữa (màu vàng) của dấu “+” phải nằm trong phạm vi của hai đường màu tím.

+ Riêng đối với trường hợp ký tự trước đó là dấu “-“, do chiều cao của dấu “-“ nhỏ nên ta sẽ xét ngược lại. Như trong hình, ta sẽ xét điểm giữa của dấu “-“ có nằm trong phạm vi của số “1” ngoài cùng hay không, nếu phải thì ta công nhận số “1” đó cũng thuộc biểu thức mũ.

➤ Dấu bằng

- Dấu “=” thực chất là sự kết hợp của hai dấu “-“. Sau khi nhận dạng được hai dấu “-“ liên tiếp nhau, ta sẽ xét độ chênh lệch tọa độ theo chiều ngang (tọa độ X) của hai ký tự này. Nếu lệch trong khoảng cho phép thì sẽ xác định đó là dấu “=“.

- Độ lệch mà chúng tôi cho phép là không vượt quá nửa chiều rộng của dấu “-“ thứ nhất. Nói cách khác $x_1 \leq x_2 \leq x_1 + width_1 / 2$

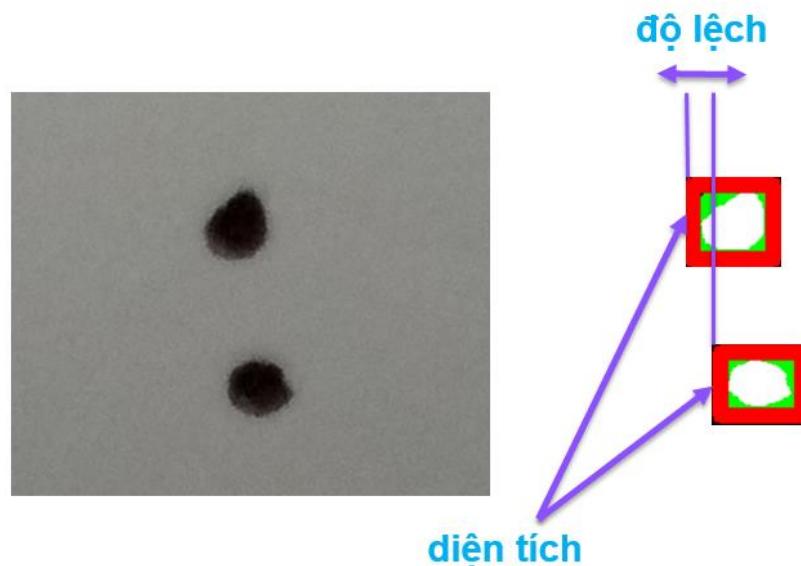


Hình 37: Nhận dạng dấu bằng

➤ Dấu chia

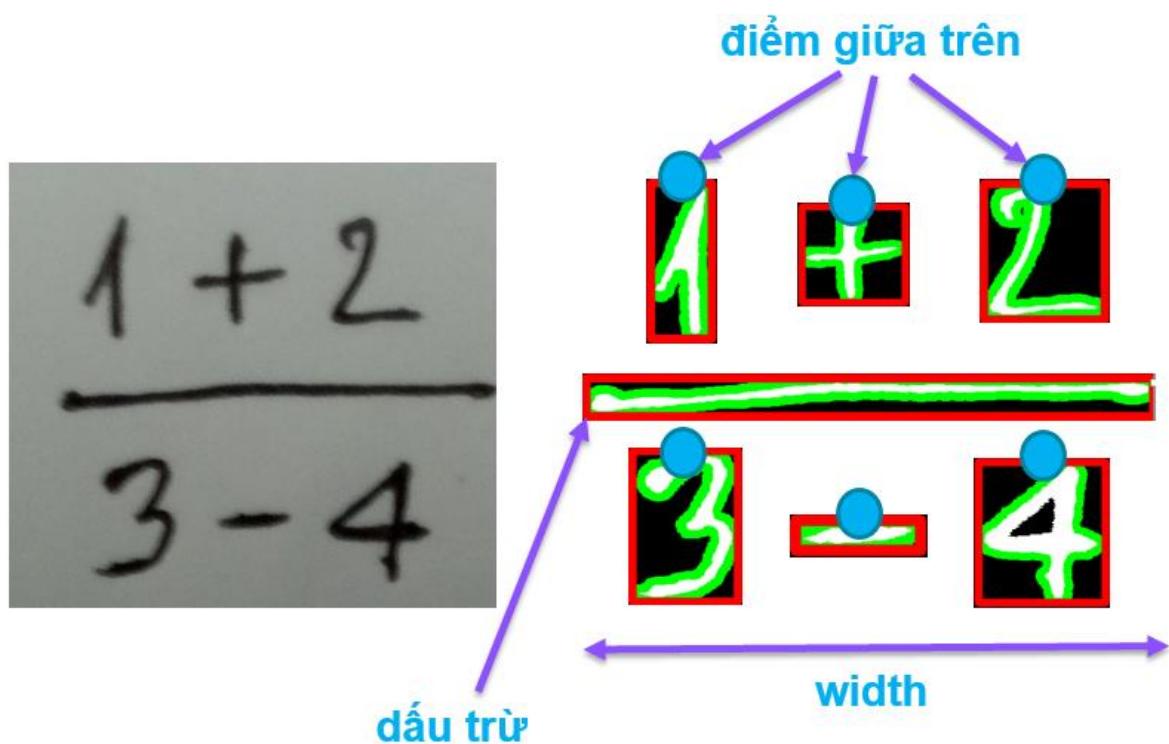
- Dấu “:” là sự kết hợp của hai dấu chấm nên bằng cách xét diện tích và độ lệch theo chiều ngang của hai dấu chấm này, ta sẽ xác định được đó có phải là dấu “:” hay không.

- Ngưỡng giá trị mà chúng tôi chọn sau nhiều lần thử nghiệm như sau:
- + Nếu diện tích ký tự < 5000 pixels thì coi đó là dấu chấm
- + Nếu độ lệch theo chiều ngang của hai dấu chấm $< 500px$ thì xác định đó là dấu “:”

*Hình 38: Nhận dạng dấu chia*

➤ **Phân số**

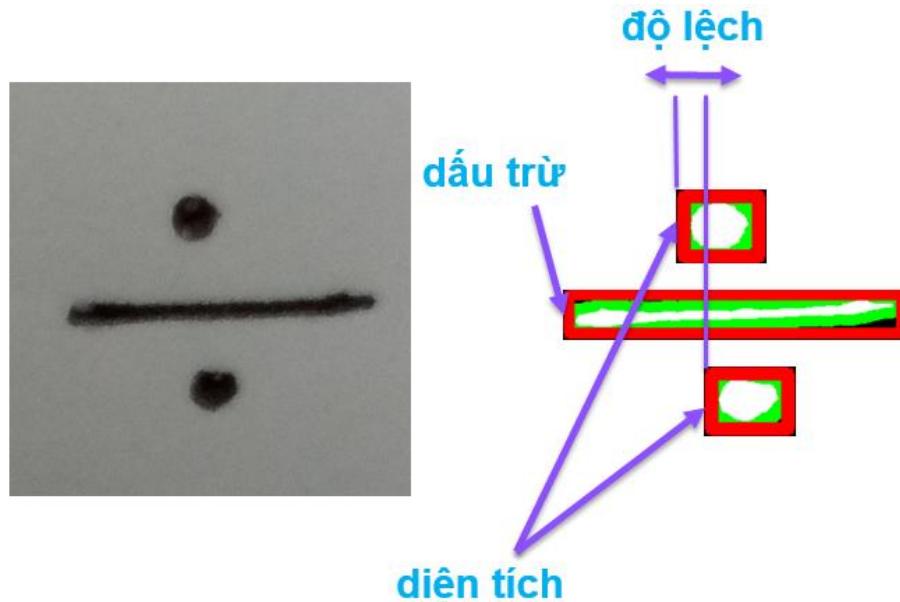
- Khi nhận dạng được ký tự dấu “-“, thì dấu “-“ đó rất có thể là dấu phân số nên ta sẽ xét thêm xem trên và dưới dấu “-“ đó có ký tự nào khác nữa hay không. Nếu có, thì các ký tự nằm ở trên sẽ là tử số, các ký tự bên dưới sẽ là mẫu số, với điều kiện là điểm giữa trên ($x + width / 2$) của ký tự phải nằm trong phạm vi chiều rộng của dấu “-“:



Hình 39: Nhận dạng phân số

- Trong trường hợp, tử số hoặc mẫu số cũng là phân số thì ta áp dụng cách đẽo quy tương tự

- Đặc biệt, với trường hợp dấu chia ở dạng phân số như hình sau:



Hình 40: Nhận dạng dấu chia dạng phân số

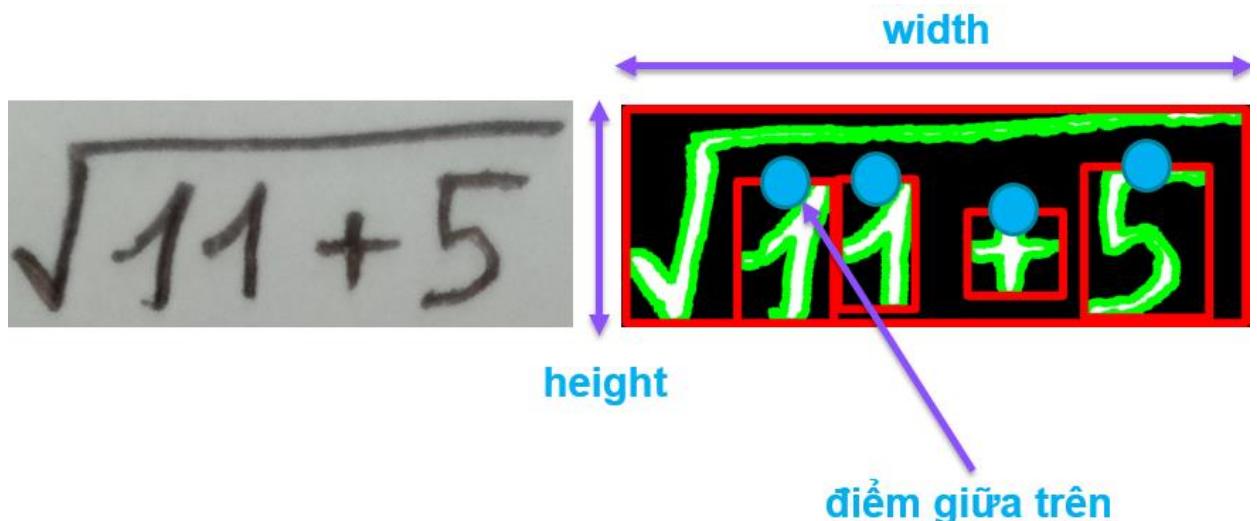
trong trường hợp này, ta sẽ xét như sau: nếu tử số và mẫu số có size là 1 thì xét tiếp diện tích và độ lệch của hai ký tự đó (như đã trình bày trong phần nhận dạng Dấu chia). Nếu thỏa điều kiện về diện tích và độ lệch thì ta suy ra đó là dấu chia.

➤ Căn thức

- Khi đã nhận dạng được dấu căn, ta sẽ tìm các phân tử trong căn dựa vào quy tắc sau: điểm giữa trên của ký tự ($x_{kytu} + width_{kytu} / 2; y_{kytu}$) phải nằm trong hình chữ nhật bao lây căn hay nói cách khác:

$$x_{can} < x_{kytu} + width_{kytu} / 2 < x_{can} + width_{can}$$

$$y_{can} < y_{kytu} < y_{can} + height_{can}$$



Hình 41: Nhận dạng căn bậc hai

➤ Hệ phương trình

- Phương pháp mà chúng tôi dùng để nhận dạng hệ phương trình như sau:

- + Đầu tiên nhận dạng được ký tự “{“
- + Sau đó kẻ một đường thẳng đi ngang qua giữa ký tự “{“
- + Xác định liệu trên đường thẳng đó có chứa dấu “=” hay không
- + Xác định liệu dưới đường thẳng đó có chứa dấu “=” hay không
- + Nếu cả trên và dưới đường thẳng đều có dấu “=” suy ra đây là hệ phương trình và phương trình thứ nhất chính là các ký tự nằm trên đường thẳng,

phương trình thứ hai chính là các ký tự nằm dưới đường thẳng.

$$\left\{ \begin{array}{l} 2x - 3y = 5 + 6x \\ 8y + 6 = -x + 9y \end{array} \right.$$

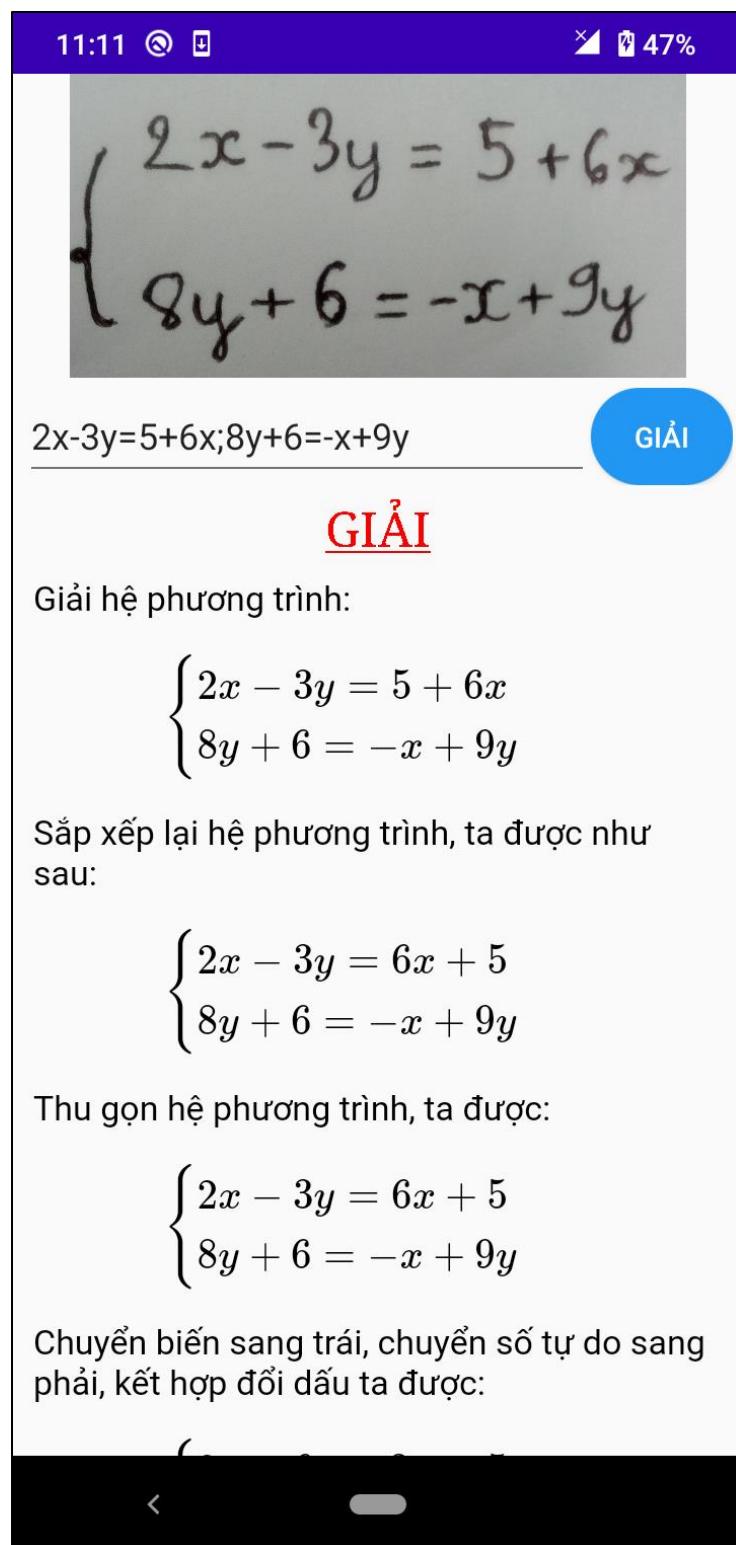
Hình 42: Nhận dạng hệ phương trình

- ❖ **Kết luận:** Đến đây, chúng tôi đã hoàn thành xong quá trình nhận dạng một biểu thức hoàn chỉnh. Tiếp theo, chúng tôi sẽ dựa vào biểu thức đã nhận dạng được để giải và hiện thị kết quả cho người dùng.

3.8. Giải và đưa ra từng bước giải

- Từ chuỗi biểu thức đã nhận dạng được, ví dụ chuỗi có dạng “ $2x^2 - 3x + 1 = 5(x - 1)$ ”. Chúng tôi sẽ tiến hành xử lý chuỗi để giải và đưa ra từng bước giải cho người dùng. Gói `java.util.regex` thật sự hữu ích ở bước này.
- Vì quá trình giải tách biệt với quá trình nhận dạng nên để dễ dàng lập trình và kiểm thử mã, chúng tôi đã sử dụng công cụ Eclipse để lập trình toàn bộ phần giải sau đó xuất thành thư viện `.jar` rồi tích hợp vào ứng dụng Android.

- Cách giải biểu thức của chúng tôi đều dựa trên các cách giải phổ biến được trình bày trong các sách toán.
- Sau khi giải xong biểu thức, công đoạn cuối cùng là chuyển các chuỗi thuần sang dạng LaTeX để hiện thị được biểu thức toán học lên giao diện.
- Một số hình ảnh sau khi hoàn thành xong bước này:



Hình 43: Ví dụ giải hệ phương trình

11:11 ④ 47%

Chuyển biến sang trái, chuyển số tự do sang phải, kết hợp đổi dấu ta được:

$$\begin{cases} 2x - 6x - 3y = 5 \\ x + 8y - 9y = -6 \end{cases}$$

Thu gọn lại, ta được:

$$\begin{cases} -4x - 3y = 5 \\ x - y = -6 \end{cases}$$

Cách giải hệ phương trình:

$$\begin{cases} -4x - 3y = 5 & (1) \\ x - y = -6 & (2) \end{cases}$$

*CÁCH 1: GIẢI HỆ BẰNG PHƯƠNG PHÁP THẾ

Từ phương trình (1), ta suy ra x theo y như sau:

$$x = \frac{5 + 3y}{-4} \quad (3)$$

Thay (3) vào phương trình (2), ta giải phương trình bậc nhất theo biến y như sau:
Giải phương trình:

$$\left(\frac{1}{-4}\right)(5 + 3y) - y = -6$$

< □

Hình 44: Ví dụ giải hệ phương trình

11:11 ④ 47%

Thay (3) vào phương trình (2), ta giải phương trình bậc nhất theo biến y như sau:
 Giải phương trình:

$$\left(\frac{1}{-4}\right)(5 + 3y) - y = -6$$

$$-y - 0,25(5 + 3y) = -6$$

Nhân phân phối với biểu thức trong ngoặc:

$$-y - 0,75y - 1,25 = -6$$

$$-1,75y - 1,25 = -6$$

Chuyển số tự do sang bên phải rồi thu gọn, ta được:

$$-1,75y = -4,75$$

Chia hai vế cho $-1,75$, suy ra:

$$y = 2,714286$$

Thay $y = 2,714286$ vào phương trình (3), suy ra:

$$x = \frac{5 + 3 \cdot 2,714286}{-4}$$

$$x = \frac{5 + 8,142858}{-4}$$
Hình 45: Ví dụ giải hệ phương trình

$x = \frac{13,142858}{-4}$

$$x = -3,285714$$

Vậy hệ phương trình có nghiệm là:
 $(-3,285714 ; 2,714286)$

*CÁCH 2: GIẢI HỆ BẰNG PHƯƠNG PHÁP CỘNG ĐẠI SỐ

Quan sát hệ hai phương trình (1) và (2), ta nhận thấy hệ số a_1 là bội của hệ số a_2 vì:

$$\frac{a_1}{a_2} = \frac{-4}{1} = -4$$

Bằng cách nhân -4 vào hai vế của (2), ta được hệ phương trình mới như sau:

$$\begin{cases} -4x - 3y = 5 & (1) \\ -4x + 4y = 24 & (4) \end{cases}$$

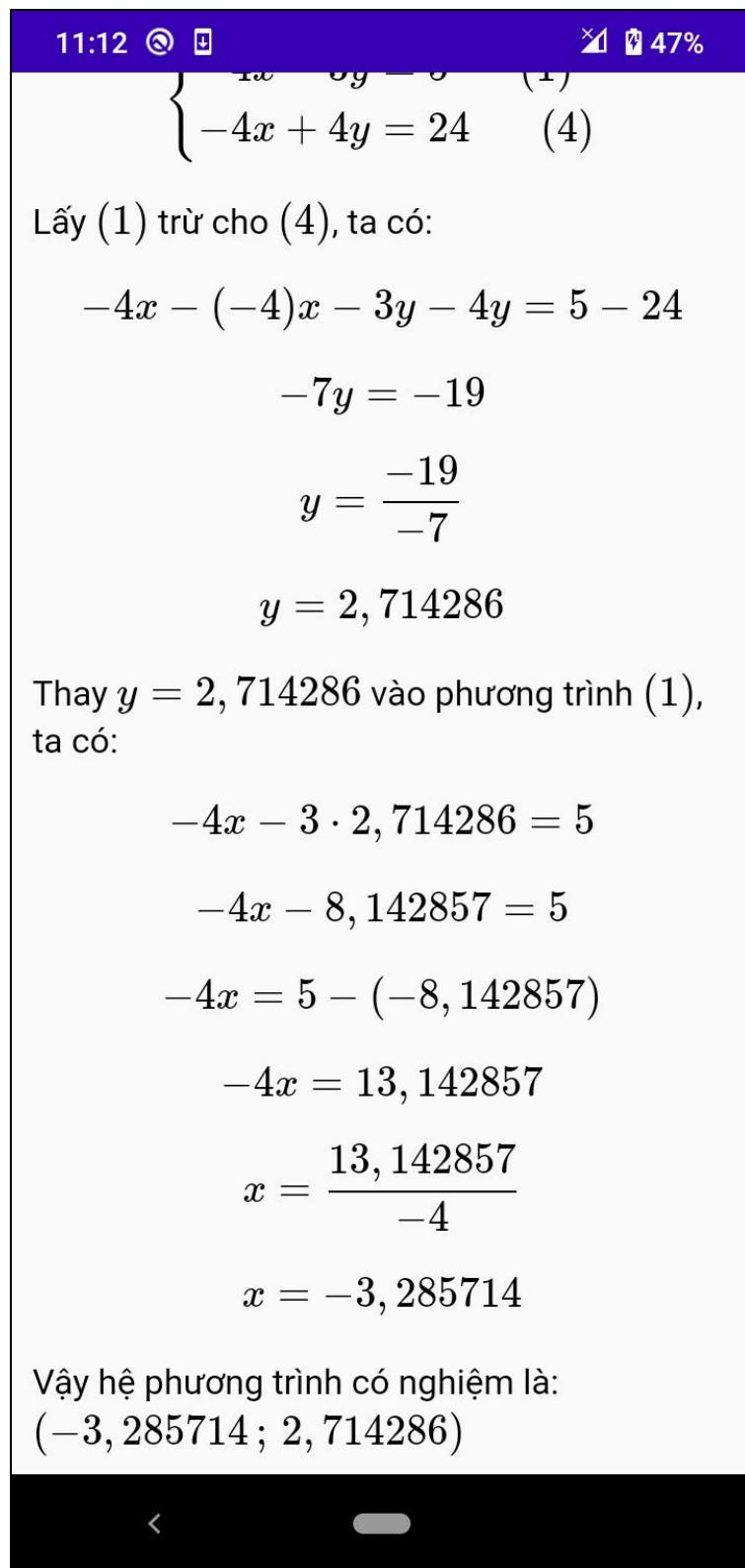
Lấy (1) trừ cho (4), ta có:

$$-4x - (-4)x - 3y - 4y = 5 - 24$$

$$-7y = -19$$

$$y = \frac{-19}{-7}$$

Hình 46: Ví dụ giải hệ phương trình

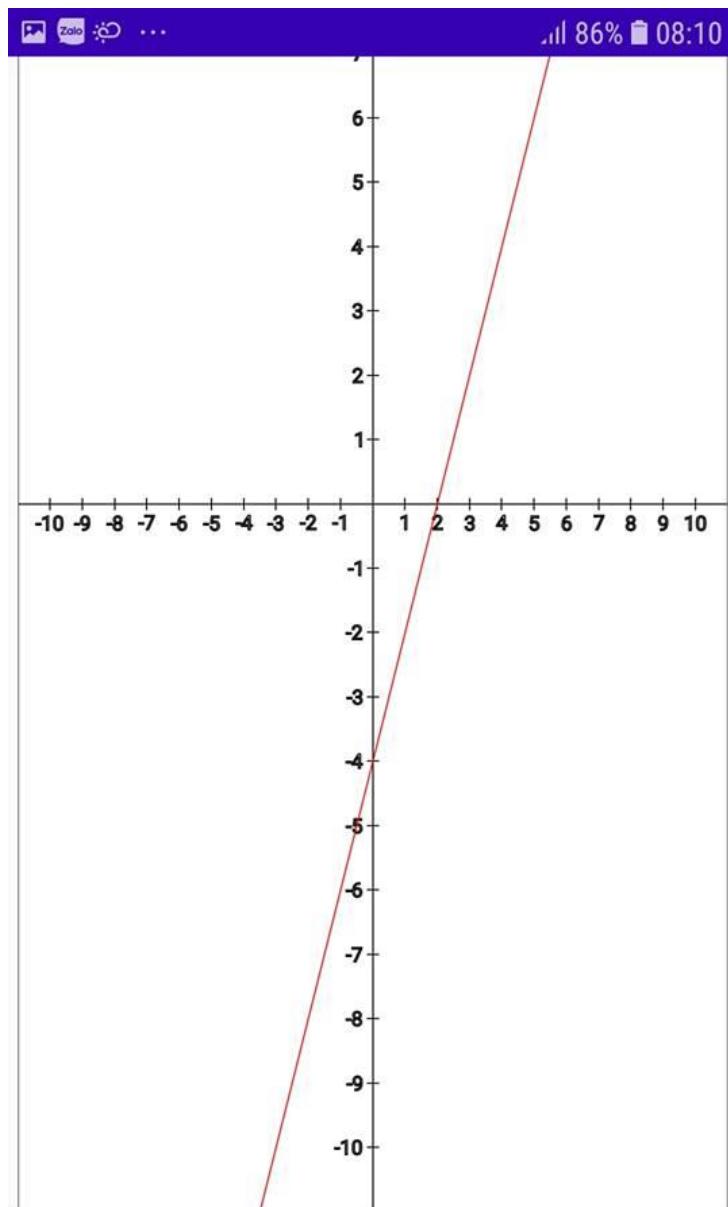


Hình 47: Ví dụ giải hệ phương trình

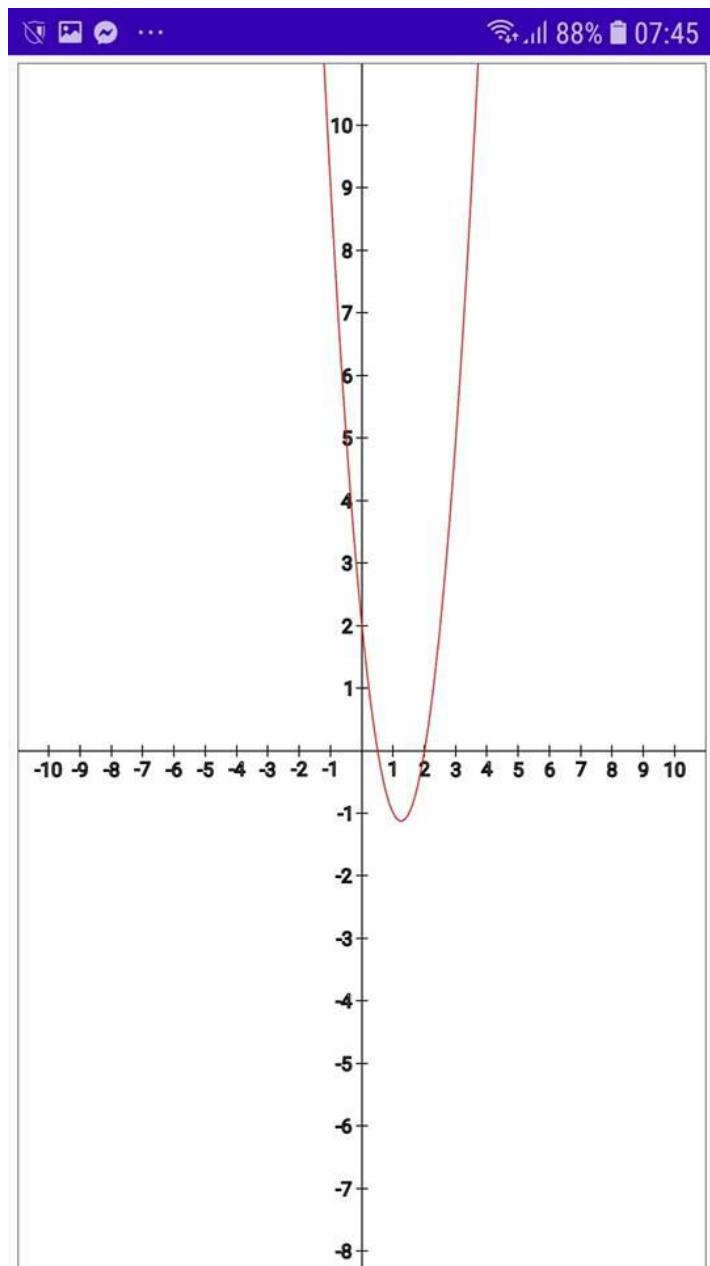
3.9. Bổ sung thêm các chức năng cho ứng dụng

3.9.1. Biểu diễn đồ thị phương trình, hệ phương trình

Bằng cách sử dụng thư viện GraphLib, các phương trình, hệ phương trình sẽ được biểu diễn một cách trực quan đến người dùng. Một số ví dụ:



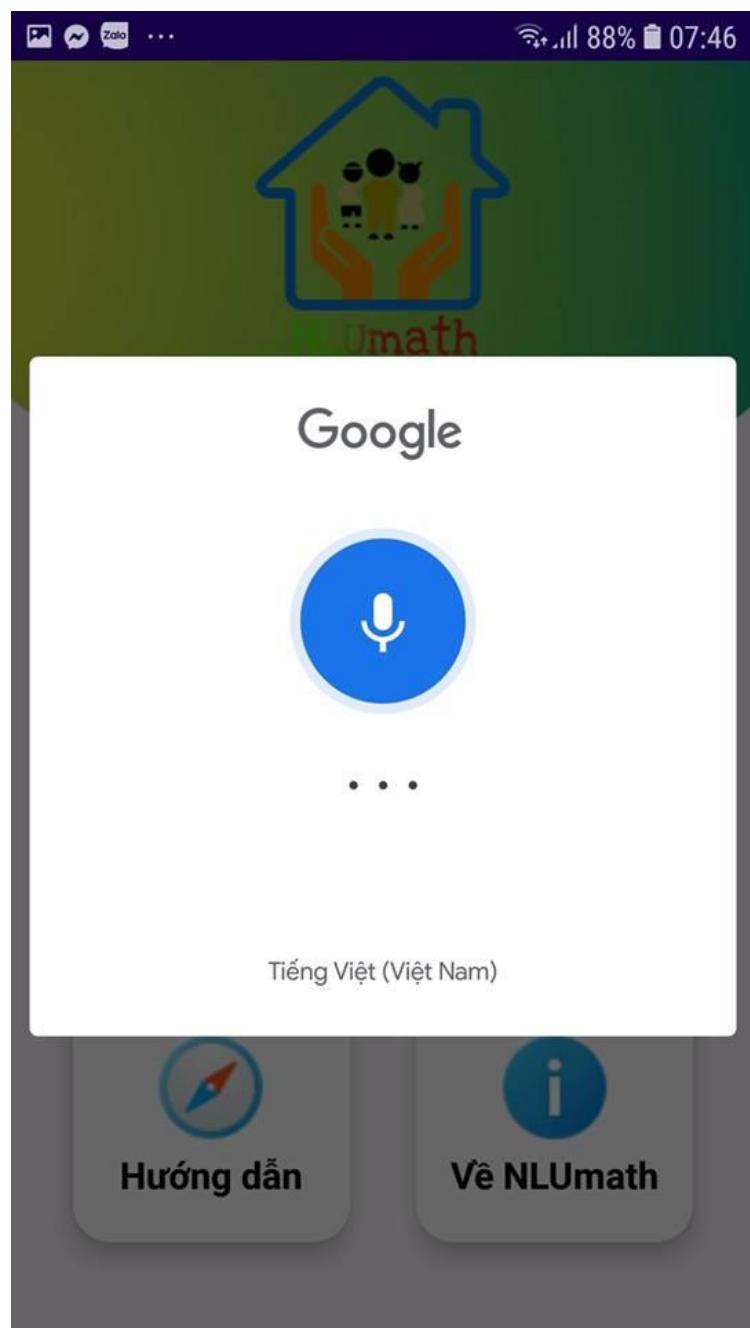
Hình 48: Minh họa đồ thị hàm số bậc nhất



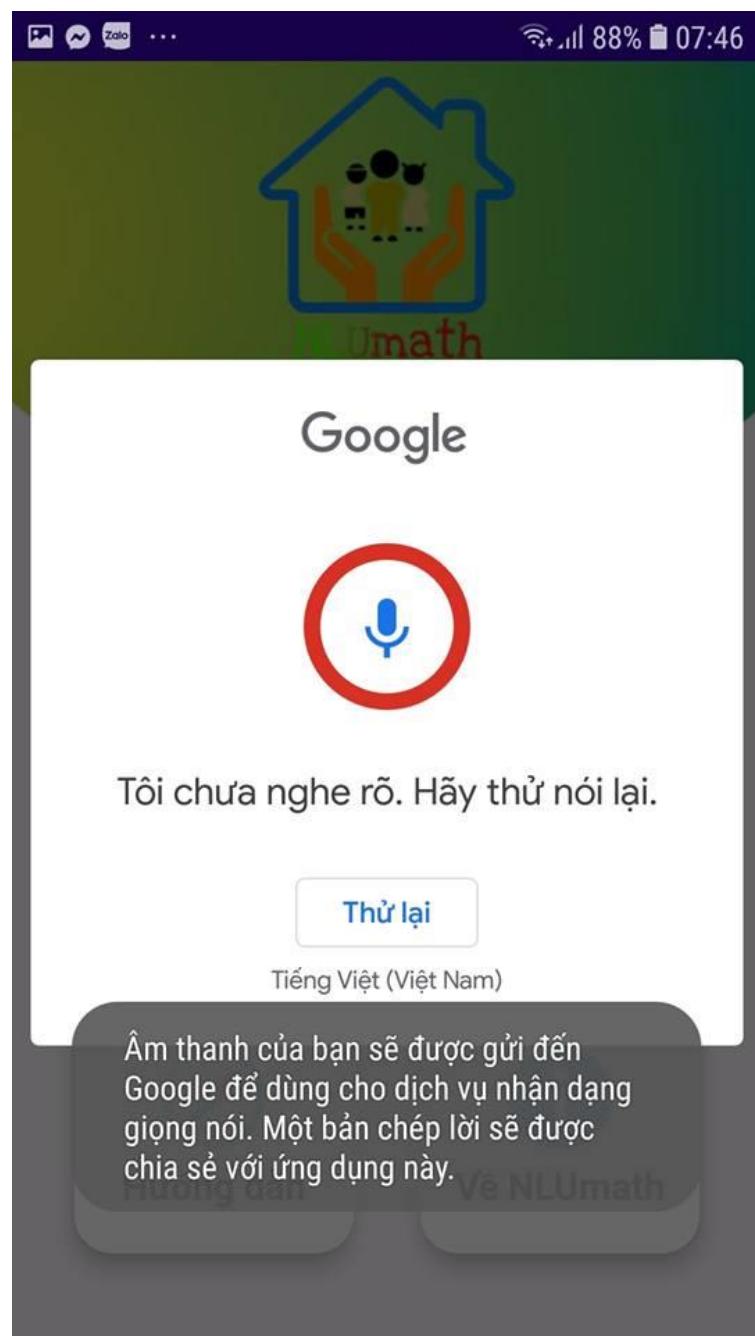
Hình 49: Minh họa đồ thị hàm số bậc hai

3.9.2. Tích hợp nhận dạng giọng nói

Mặc dù chức năng này không nằm trong phạm vi của đề tài, nhưng để người dùng có thể trải nghiệm ứng dụng một cách tiện lợi hơn. Chúng tôi đã tích hợp khả năng nhận dạng giọng nói thông qua Google API. Người dùng chỉ cần nói bài toán và ứng dụng sẽ nhanh chóng hiển thị kết quả giải.



Hình 50: Nhận dạng giọng nói

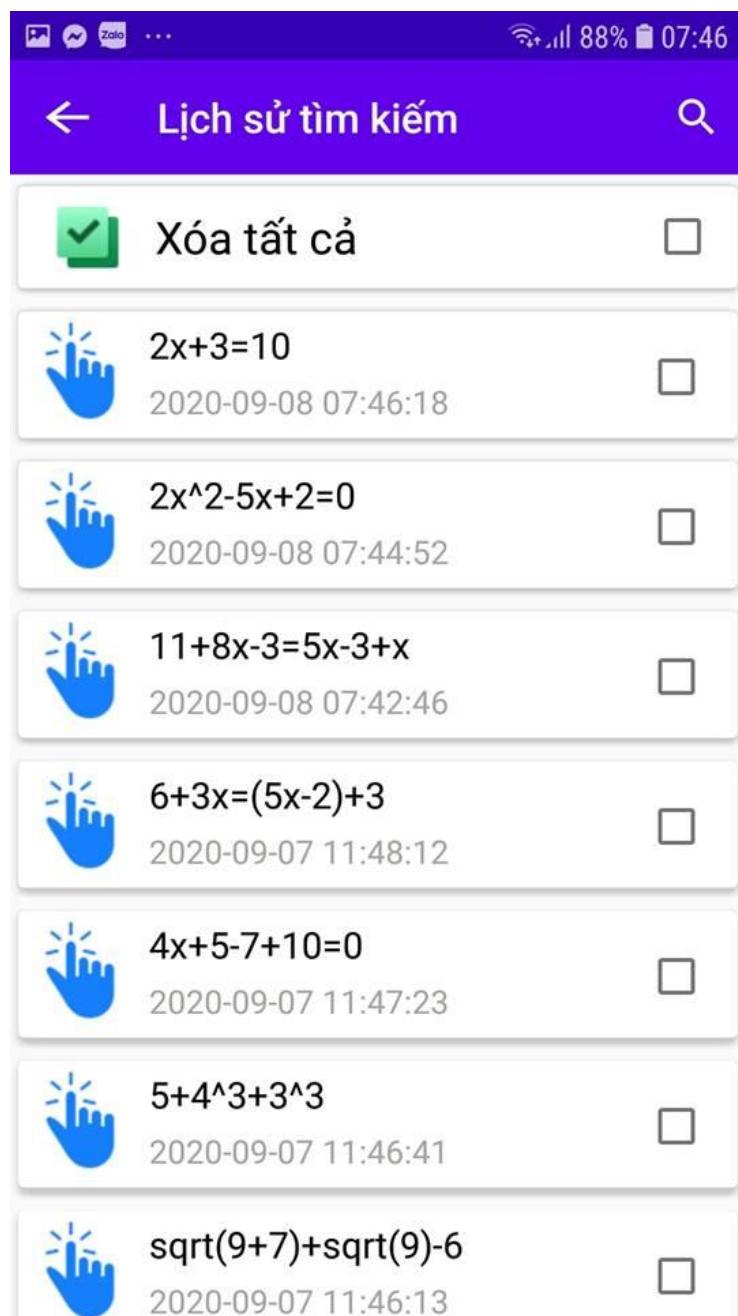


Hình 51: Giọng nói được gửi đến Google để xử lý

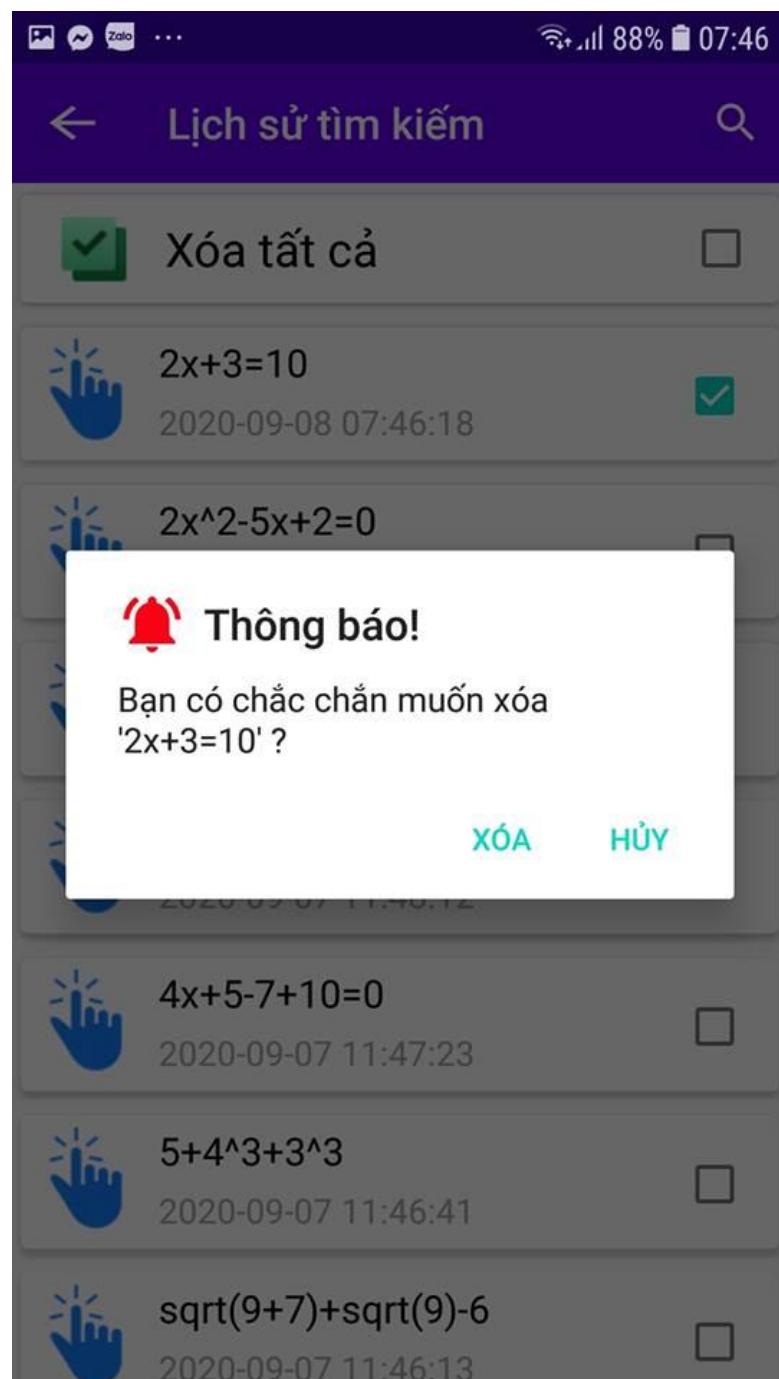
*Hình 52: Giải biểu thức giọng nói*

3.9.3. Lưu lịch sử biểu thức đã giải

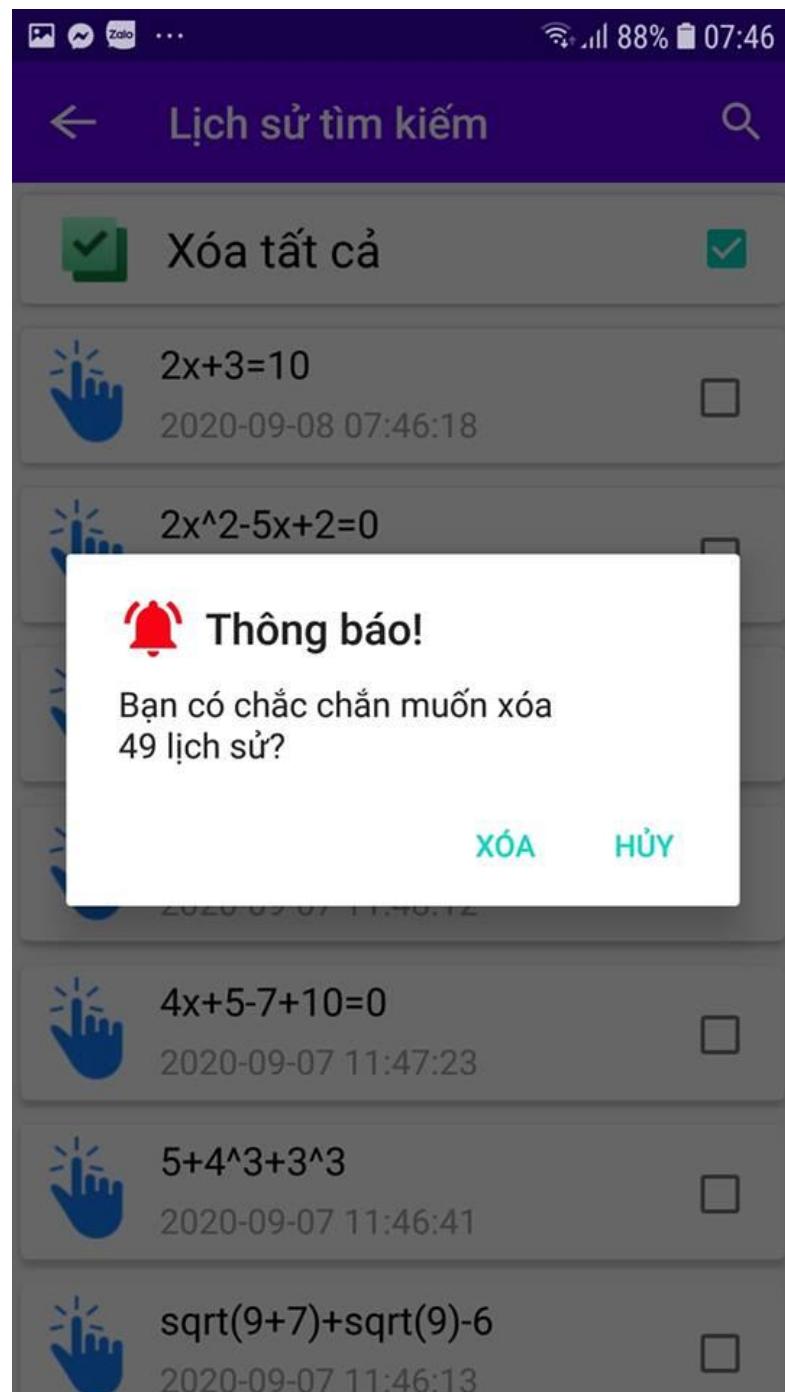
Với cơ sở dữ liệu SQLite, biểu thức sẽ được lưu trữ lại. Từ đó, có thể truy xuất kết quả giải một cách nhanh chóng thông qua chức năng xem Lịch Sử mà không cần phải chụp lại ảnh hay tải ảnh lên từ thư viện.



Hình 53: Màn hình lịch sử biểu thức đã lưu



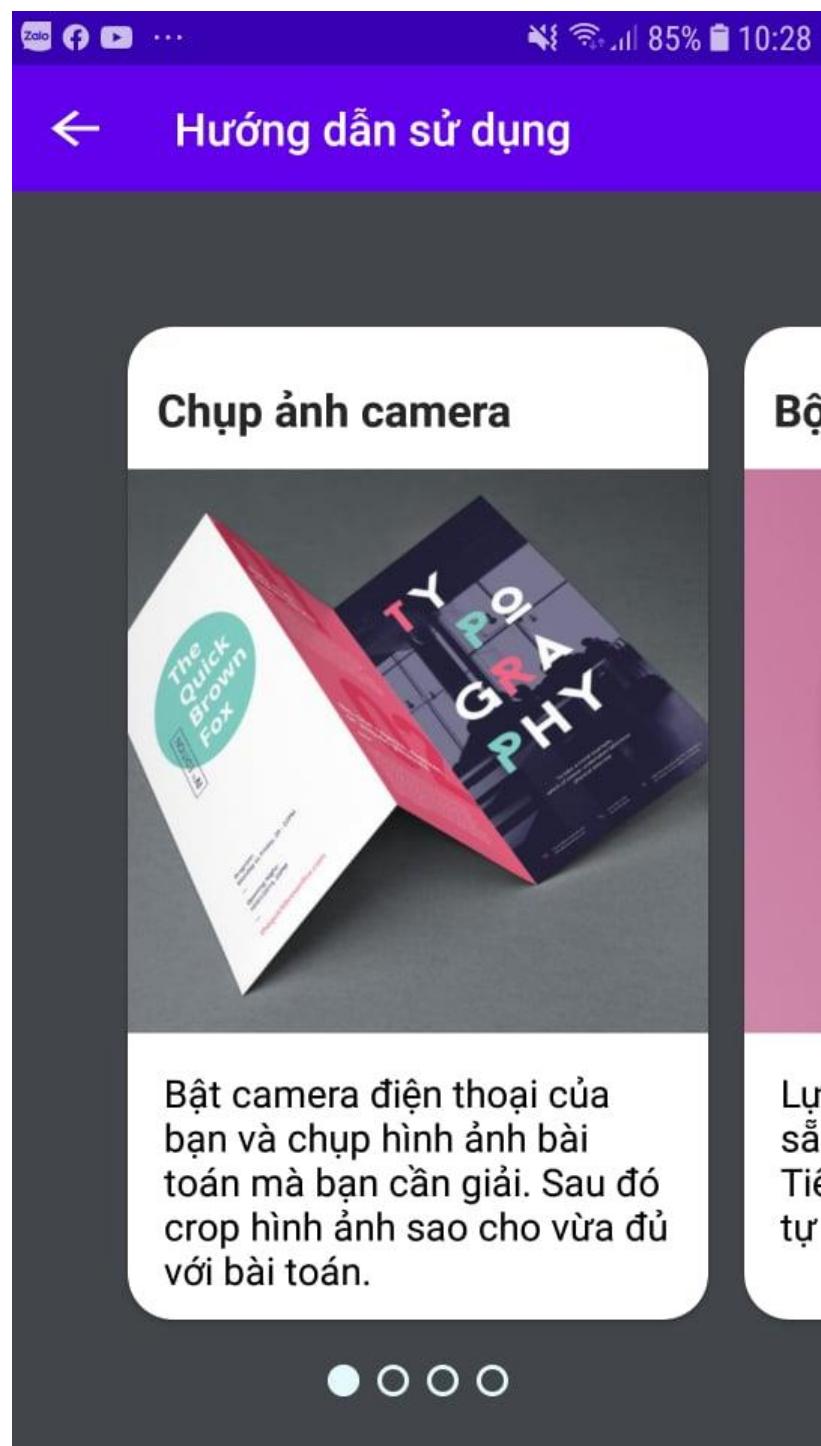
Hình 54: Người dùng hoàn toàn có thể xóa biểu thức đã lưu



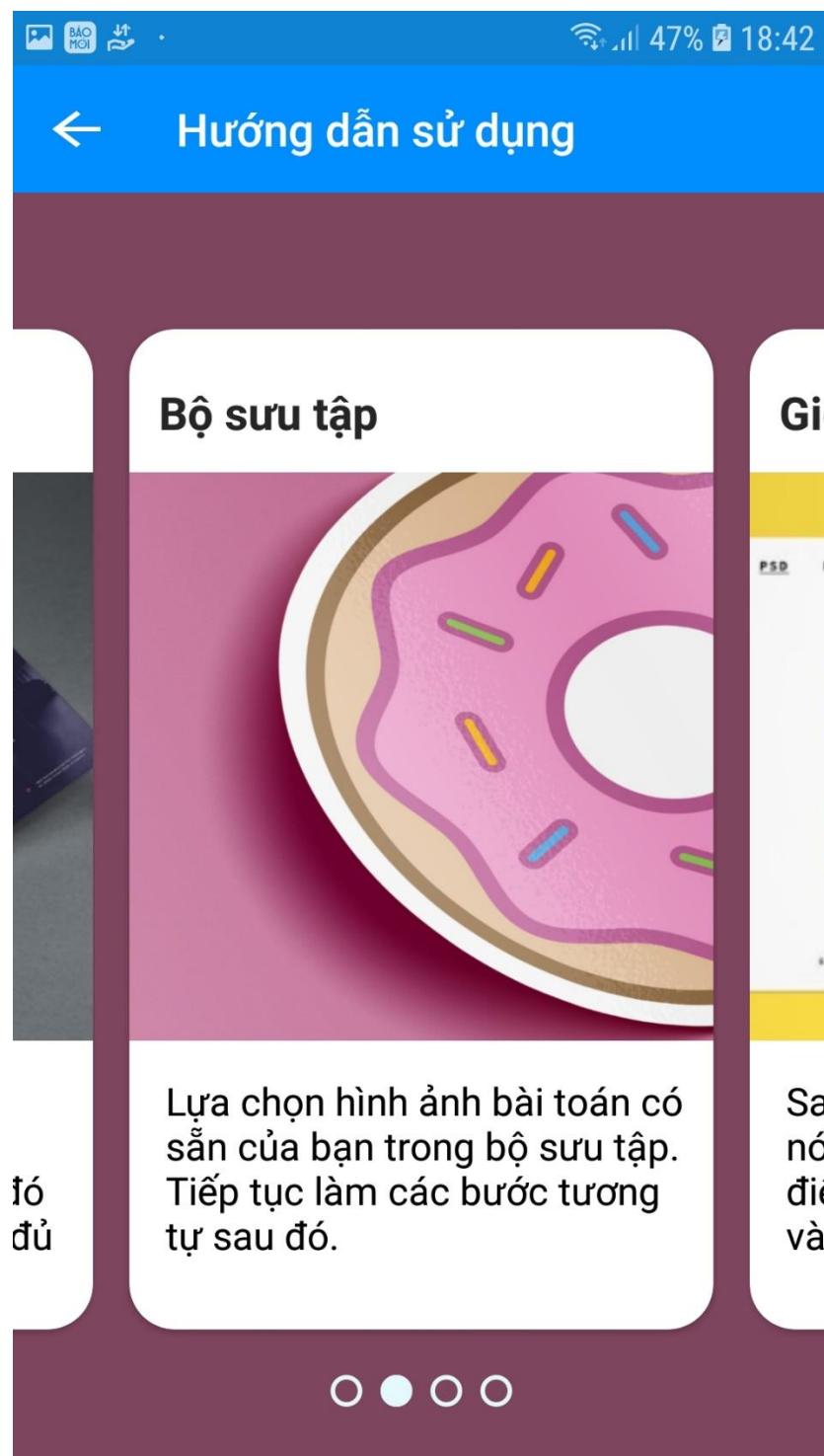
Hình 55: Xóa tất cả biểu thức

3.9.4. Hướng dẫn sử dụng

Chức năng giúp người dùng dễ dàng hiểu cách sử dụng ứng dụng.



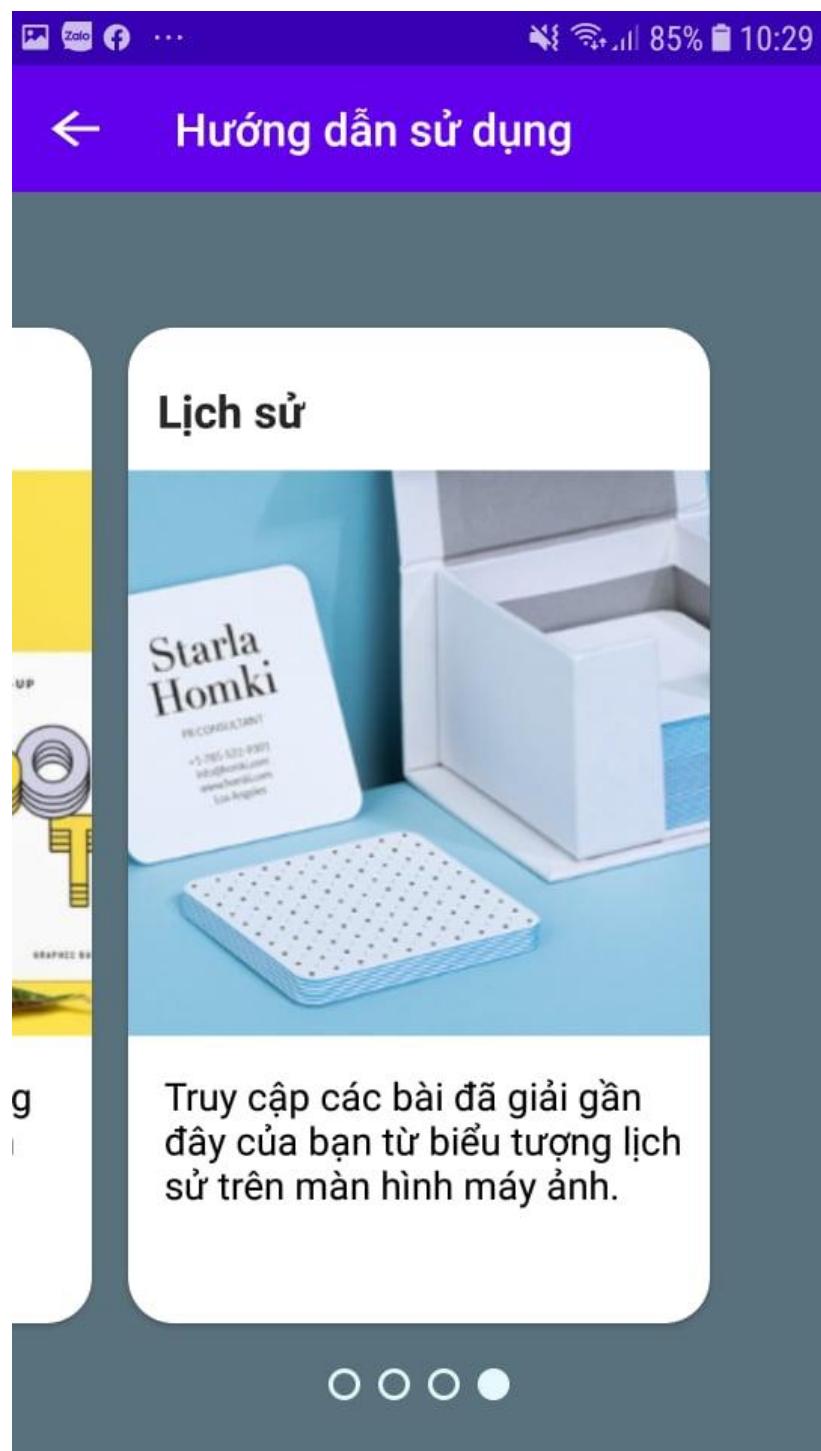
Hình 56: Màn hình hướng dẫn sử dụng



Hình 57: Màn hình hướng dẫn sử dụng



Hình 58: Màn hình hướng dẫn sử dụng



Hình 59: Màn hình hướng dẫn sử dụng

3.9.5. Thông tin ứng dụng

Thông tin về ứng dụng, tác giả cũng như địa chỉ email liên hệ, góp ý cho sản

phẩm.



Hình 60: Màn hình thông tin ứng dụng

CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC

4.1. Sản phẩm

Nhìn chung, ứng dụng hiện tại đã hiện thực được các mục tiêu đặt ra ban đầu của đề tài: Ứng dụng trên thiết bị di động cho phép người dùng chụp ảnh bài toán sau đó sẽ đưa ra lời giải cũng như chi tiết từng bước và đồ thị cho các phương trình.

Ngoài ra ứng dụng cũng hỗ trợ thêm khả năng giải hệ phương trình bậc nhất hai ẩn và chức năng nhận dạng biểu thức bằng giọng nói.

4.2. Kết quả thực nghiệm

❖ Bảng kết quả nhận dạng một số biểu thức:

- Điều kiện: bút mực viết rõ ràng trên nền giấy A4, ánh sáng tốt, chất lượng ảnh tốt.

Dạng toán	Hình biểu thức	Biểu thức nhận dạng	Biểu thức thực sự	Kết quả nhận dạng
Biểu thức không chứa biến	$\frac{8+1-4+7}{7+5}$	$\frac{8+1-4+7}{7+5}$	$\frac{8+1-4+7}{7+5}$	Đúng
	$\sqrt{9+7} + \sqrt{9}$	$\sqrt{9+7} + \sqrt{9}$	$\sqrt{9+7} + \sqrt{9}$	Đúng
	$5+4^3+3^3$	$5+4^3+3^3$	$5+4^3+3^3$	Đúng
	$\frac{1}{3} + \frac{4}{5} - \frac{3}{6}$	$\frac{1}{x} + \frac{4}{5} - \frac{3}{6}$	$\frac{1}{3} + \frac{4}{5} - \frac{3}{6}$	Sai

	$8^{1+2+3} - \frac{4+5+6}{9-8-7}$	$8^{1+2+3} - \frac{4+5+6}{9-8-7}$	$8^{1+2+3} - \frac{4+5+6}{9-8-7}$	Đúng
	$\sqrt{2} - \frac{3}{4} + 5^2$	$\sqrt{2} - \frac{3}{4} + 5^x$	$\sqrt{2} - \frac{3}{4} + 5^2$	Sai
	$8 \times 9 : 6$	$8 * 9 / 6$	$8 * 9 / 6$	Đúng
	$\sqrt{6^4} + 7^{(-2)} - \frac{88}{2^3}$	$\sqrt{6^4} + 7^{(-2)} - \frac{88}{z^y}$	$\sqrt{6^4} + 7^{(-2)} - \frac{88}{2^3}$	Sai
	$100 \div 5 - (7+3)$	$100 \div 5 - (7+3)$	$100 \div 5 - (7+3)$	Đúng
	$\frac{\frac{1}{2} - \sqrt{2}}{\frac{3}{4} + 6}$	$\frac{\frac{1}{2} - \sqrt{2}}{\frac{3}{4} + 6}$	$\frac{\frac{1}{2} - \sqrt{2}}{\frac{3}{4} + 6}$	Đúng
Phương trình bậc nhất	$6+3x=(5x-2)+3$	$6+3x=(5x-2)+3$	$6+3x=(5x-2)+3$	Đúng
	$(3x-2)(4x+5)=0$	$(3x-2)(4x+5)=0$	$(3x-2)(4x+5)=0$	Đúng
	$4x-2=6$	$4x-z=6$	$4x-2=6$	Sai
	$11+8x-3=5x-3+x$	$11+8x-3=5x-3+$	$11+8x-3=5x-3+x$	Đúng
	$4x+5-7+10=0$	$4x+5-7+10=0$	$4x+5-7+10=0$	Đúng

	$10x + 5 = 62 - 9x$	$104 + 5 = 62 - yx$	$10x + 5 = 62 - 9x$	Sai
	$\frac{1}{2}y + \frac{1}{4}y = \sqrt{2}y + 8^3$	$\frac{1}{2}y + \frac{1}{4}y = \sqrt{2}y + 8^3$	$\frac{1}{2}y + \frac{1}{4}y = \sqrt{2}y + 8^3$	Đúng
	$6x + 7x - 2 = 5 + 3x$	$6x + 7x - 2 = 5 + 3x$	$6x + 7x - 2 = 5 + 3x$	Đúng
	$17x + 2 = 6(x - 4)$	$17x + 2 = 6(x - 4)$	$17x + 2 = 6(x - 4)$	Đúng
	$9^6x + 8^7x = 6x - 55$	$9^6x + 8^7x = 6x - 55$	$9^6x + 8^7x = 6x - 55$	Đúng
Phương trình bậc hai	$2x^2 - 5x + 2 = 0$	$2x^2 - 5x + 2 = 0$	$2x^2 - 5x + 2 = 0$	Đúng
	$(2x - 1)^2 - 25 = 0$	$(2x - 1)^2 - 25 = 0$	$(2x - 1)^2 - 25 = 0$	Đúng
	$x^2 + x - 12 = 0$	$x^2 + x - 12 = 0$	$x^2 + x - 12 = 0$	Đúng
	$(3x - 2)(4x + 5) = 0$	$(3x - 2)(4x + 5) = 0$	$(3x - 2)(4x + 5) = 0$	Đúng
	$2(x+3) - x^2 - 3x = 0$	$2(x+3) - x^2 - 3x = 0$	$2(x+3) - x^2 - 3x = 0$	Đúng
	$z^2 - z = 5$	$7^2 - 2 = 5$	$z^2 - z = 5$	Sai
	$8^3x^2 + 10 = 6x - \sqrt{2}x$	$8^3x^2 + 10 = 6x - \sqrt{2}x$	$8^3x^2 + 10 = 6x - \sqrt{2}x$	Đúng

	$\frac{1}{x^{(-2)}} + 5x + 4 = 0$	$\frac{1}{x^{1-2}} + 5x + y = 0$	$\frac{1}{x^{(-2)}} + 5x + 4 = 0$	Sai
	$2x^2 - 8x + 1 = -3 - 2x$	$2x^2 - 8x + 1 = -3 - 2x$	$2x^2 - 8x + 1 = -3 - 2x$	Đúng
	$4x^2 = (3x + 2)^2$	$4x^2 = (3x + 2)^2$	$4x^2 = (3x + 2)^2$	Đúng
Phương trình bậc ba	$2x^3 - 3x^2 + 6x + 4 = 2x^2 - 3x^3 + x^2 - 10 + x$	$2x^3 - 3x^2 + 6x + 4 = 2x^2 - 3x^3 + x^2 - 10 + x$	$2x^3 - 3x^2 + 6x + 4 = 2x^2 - 3x^3 + x^2 - 10 + x$	Đúng
	$-3z^2 + 6z + 2z^3 + 4 = 6z^2 - 10z^3$	$-3z^2 + 6z + 2z^3 + 4 = 6z^2 - 10z^3$	$-3z^2 + 6z + 2z^3 + 4 = 6z^2 - 10z^3$	Sai
	$2y^3 + 3y^2 - y - 1 = 5 + 2y - 3y^3$	$2y^3 + 3y^2 - y - 1 = 5 + 2y - 3y^3$	$2y^3 + 3y^2 - y - 1 = 5 + 2y - 3y^3$	Đúng
	$-5x^2 + 2x^3 - x^3 - 4x - 1 = 0$	$-3x^2 + 2x^3 - x^3 - 4x - 1 = 0$	$-3x^2 + 2x^3 - x^3 - 4x - 1 = 0$	Đúng
	$x^3 - 3x^2 - 9x - 1 = 2x^2 + 4x + 5$	$x^3 - 3x^2 - 9x - 1 = 3x^2 + 4x + 5$	$x^3 - 3x^2 - 9x - 1 = 2x^2 + 4x + 5$	Sai
	$-3x^3 + 6x^2 - x - 1 = 2x^2 - 3x + 1$	$-3x^3 + 6x^2 - x - 1 = 2x^2 - 3x + 1$	$-3x^3 + 6x^2 - x - 1 = 2x^2 - 3x + 1$	Sai
	$x^3 - 12x + 16 = 0$	$x^3 - 12x + 16 = 0$	$x^3 - 12x + 16 = 0$	Đúng
	$2x^3 + 3x^2 - x - 1 = 0$	$2x^3 + 3x^2 - x - 1 = 0$	$2x^3 + 3x^2 - x - 1 = 0$	Đúng
	$2x^3 - 3x^2 + 6x + 4 = 0$	$2x^3 - 3x^2 + 6x + 4 = 0$	$2x^3 - 3x^2 + 6x + 4 = 0$	Đúng

	$x^3 - 2x^2 - 5x + 6 = 7 - x^2 - x$	$x^3 - 2x^2 - 5x + 6 = 7 - x^2 - 4$	$x^3 - 2x^2 - 5x + 6 = 7 - x^2 - x$	Sai
Hệ phương trình bậc nhất hai ẩn	$\begin{cases} -12y + 8x = -46 \\ 3y - 17x = 10 \end{cases}$	$\begin{cases} -12y + 8x = -46 \\ 3y - 17x = 10 \end{cases}$	$\begin{cases} -12y + 8x = -46 \\ 3y - 17x = 10 \end{cases}$	Đúng
	$\begin{cases} -2y + 4z = 3z - 12 \\ 5z + y = 10y - 7 \end{cases}$	$\begin{cases} -2y + 9z = 32 - 12 \\ 5z + y = 10y - 7 \end{cases}$	$\begin{cases} -2y + 4z = 3z - 12 \\ 5z + y = 10y - 7 \end{cases}$	Sai
	$\begin{cases} -3x + 5y + 6x - 7 = 2y - 3 + 4x - 6 \\ 5y - x + 10 - 9y = -12 + 4x - 2x + 2y - 2y \end{cases}$	$\begin{cases} -3x + 5y + 6x - 7 = 2y - 8 + 4x - 6 \\ 5y - x + 10 - 9y = -12 + 4x - 2x + zy - \end{cases}$	$\begin{cases} -3x + 5y + 6x - 7 = 2y - 3 + 4x - 6 \\ 5y - x + 10 - 9y = -12 + 4x - 2x + 2y - 2y \end{cases}$	Sai
	$\begin{cases} 2x + 3y = 5 - 4x \\ -4y + 6x - 3 = 10 + 2y \end{cases}$	$\begin{cases} 2x + 3y = 5 - 4x \\ -4y + 6x - 3 = 10 + 2y \end{cases}$	$\begin{cases} 2x + 3y = 5 - 4x \\ -4y + 6x - 3 = 10 + 2y \end{cases}$	Đúng
	$\begin{cases} -2z + 1 = 6x - 5 + 10z \\ x - 9 = 11 - x + 3z \end{cases}$	$\begin{cases} -2z + 1 = 6x - 5 + 10z \\ x - 9 = 11 - x + 3z \end{cases}$	$\begin{cases} -2z + 1 = 6x - 5 + 10z \\ x - 9 = 11 - x + 3z \end{cases}$	Đúng
	$\begin{cases} 2x + y = 3 \\ x - y = 6 \end{cases}$	$\begin{cases} 2x + y = 3 \\ x - y = 6 \end{cases}$	$\begin{cases} 2x + y = 3 \\ x - y = 6 \end{cases}$	Đúng
	$\begin{cases} 2x + 3y = 5 \\ 2x - y = 1 \end{cases}$	$\begin{cases} 2x + 3y = 5 \\ 2x - y = 1 \end{cases}$	$\begin{cases} 2x + 3y = 5 \\ 2x - y = 1 \end{cases}$	Đúng
	$\begin{cases} 2x + y = 4 \\ 2x - y = 0 \end{cases}$	$\begin{cases} 2x + y = 4 \\ 2x - y = 0 \end{cases}$	$\begin{cases} 2x + y = 4 \\ 2x - y = 0 \end{cases}$	Đúng

	$\begin{cases} 2x + 3y = 1 \\ x - y = 3 \end{cases}$	$\begin{cases} 2x + 3y = 1 \\ x - y = 3 \end{cases}$	$\begin{cases} 2x + 3y = 1 \\ x - y = 3 \end{cases}$	Đúng
	$\begin{cases} 7x - 3y = 5 \\ 4x + y = 2 \end{cases}$	$\begin{cases} zx - 3y = 5 \\ 4x + y = 2 \end{cases}$	$\begin{cases} 7x - 3y = 5 \\ 4x + y = 2 \end{cases}$	Sai

Bảng 4: Kết quả nhận dạng một số biểu thức

- Theo kết quả ở bảng trên, ta có tỉ lệ nhận dạng đúng như sau:

- + Biểu thức không chứa biến: 7/10
- + Phương trình bậc nhất: 8/10
- + Phương trình bậc hai: 8/10
- + Phương trình bậc ba: 6/10
- + Hệ phương trình bậc nhất hai ẩn: 7/10

❖ Nhận xét:

- Mặc dù không thể nhận dạng đúng hết tất cả các biểu thức trên nhưng tỉ lệ nhận dạng đúng vẫn ở mức chấp nhận được.

- Ta có thể thấy, biểu thức chưa càng nhiều ký tự thì khả năng nhận diện sai càng cao. Các ký tự “2” và “x”; “7” và “z” thường nhận diện nhầm lẫn cho nhau.

- Số lượng biểu thức bên trên còn khá ít nên mục đích của bảng là chỉ để tham khảo, không thể dùng nó để đánh giá độ chính xác của mô hình vì việc nhận dạng tốt hay không phụ thuộc vào nhiều yếu tố: nét vẽ của ký tự, màu mực, nền giấy, điều kiện ánh sáng, độ phân giải của ảnh,...

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Như vậy, đề tài luận văn đã được hoàn thành. Trong quá trình tìm hiểu đề tài này, chúng tôi phần nào cũng đã giới thiệu cơ bản, khái quát về xử lý ảnh và học sâu, khả năng huấn luyện mô hình Neural Network sử dụng Tensorflow, một thư viện ML mạnh mẽ của Google và thư viện xử lý ảnh nguồn mở vô cùng phổ biến OpenCV.

Thông qua việc ứng dụng đề tài trên thiết bị di động đã giúp chúng tôi phần nào nắm được những kiến thức, phương pháp và kỹ thuật lập trình để xây dựng một ứng dụng chạy trên nền tảng Android bằng ngôn ngữ lập trình Java.

Bên cạnh đó, nhờ sự hướng dẫn và chỉ bảo tận tình của giáo viên hướng dẫn, khả năng tự học cũng như khả năng lập trình được nâng cao rõ rệt. Điều này sẽ giúp cho sản phẩm mỗi ngày được hoàn thiện hơn cho người sử dụng.

Có thể nhận xét rằng, những kiến thức đạt được này là vô cùng phong phú và tạo tiền đề để chúng tôi có thể tiếp tục đào sâu nghiên cứu trên các lĩnh vực này, cũng như là nền tảng cho các dự án sắp tới khi ra trường.

Để ứng dụng này có thể đi vào sử dụng trong thực tế hiệu quả hơn, trong tương lai chúng tôi cần phải hoàn thiện các vấn đề sau đây:

- Tăng độ chính xác khi nhận dạng biểu thức.
- Nhận dạng và giải được nhiều dạng toán hơn.
- Xây dựng chức năng Máy tính bỏ túi.
- Cải thiện giao diện và tăng trải nghiệm người dùng.

TÀI LIỆU THAM KHẢO

Tiếng Việt

- [1]. Đoàn Võ Duy Thanh & Nguyễn Mai Hoàng Long & Lê Khắc Hồng Phúc - Đắm mình vào học sâu. Link: <https://d21.aivivn.com/index.html>
- [2]. Wikipedia - Học sâu. Link: <https://vi.wikipedia.org/>
- [3]. Nguyễn Văn Hiếu – Tensorflow. Link: <https://nguyenvanhieu.vn/thu-vien-tensorflow/>
- [4]. Giới thiệu SQLite. Link: <https://freetuts.net/gioi-thieu-sqlite-sqlite-la-gi-1719.html>

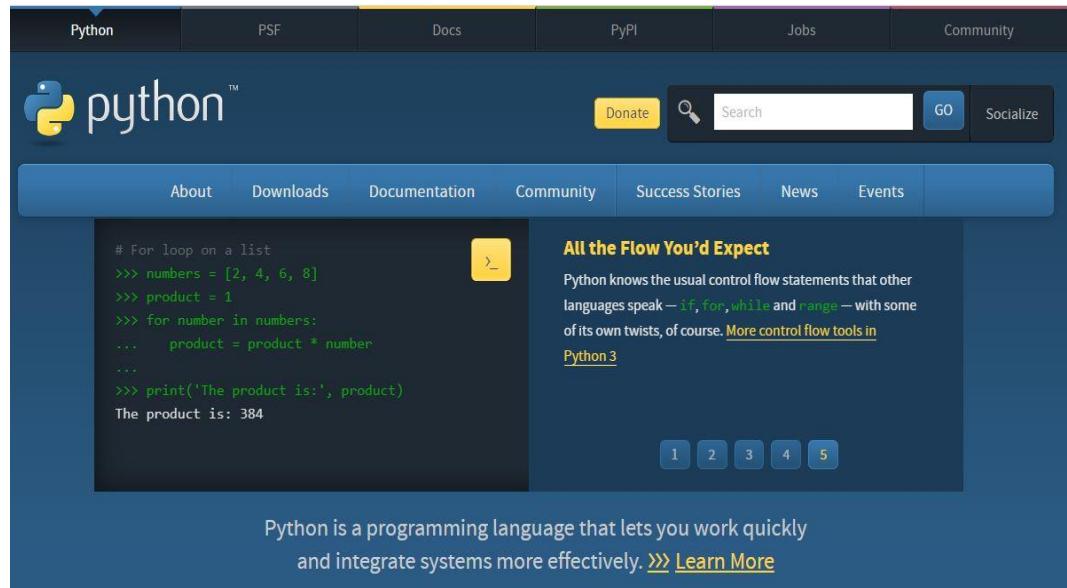
Tiếng Anh

- [5]. ArthurHub - Image Cropping Library for Android. Link: <https://github.com/ArthurHub/Android-Image-Cropper>
- [6]. Taha Emara - Multi-digit-segmentation-and-recognition. Link: <https://github.com/tahaemara/multi-digit-segmentation-and-recognition/blob/master/src/main/java/com/emaraic/digitrecognition/Application.java>
- [7]. Uklomaszewski - EvalEx - Java Expression Evaluator. Link: <https://github.com/uklomaszewski/EvalEx>
- [8]. John I. Moore, Jr. – GraphLib: An open source Android library for graphs. Link: <https://www.infoworld.com/article/3226733/graphlib-an-open-source-android-library-for-graphs.html?page=2>
- [9]. Leslie Lamport - LaTeX, 1984. Link: <https://vi.wikipedia.org/wiki/LaTeX>

PHỤ LỤC

*Cài đặt Python

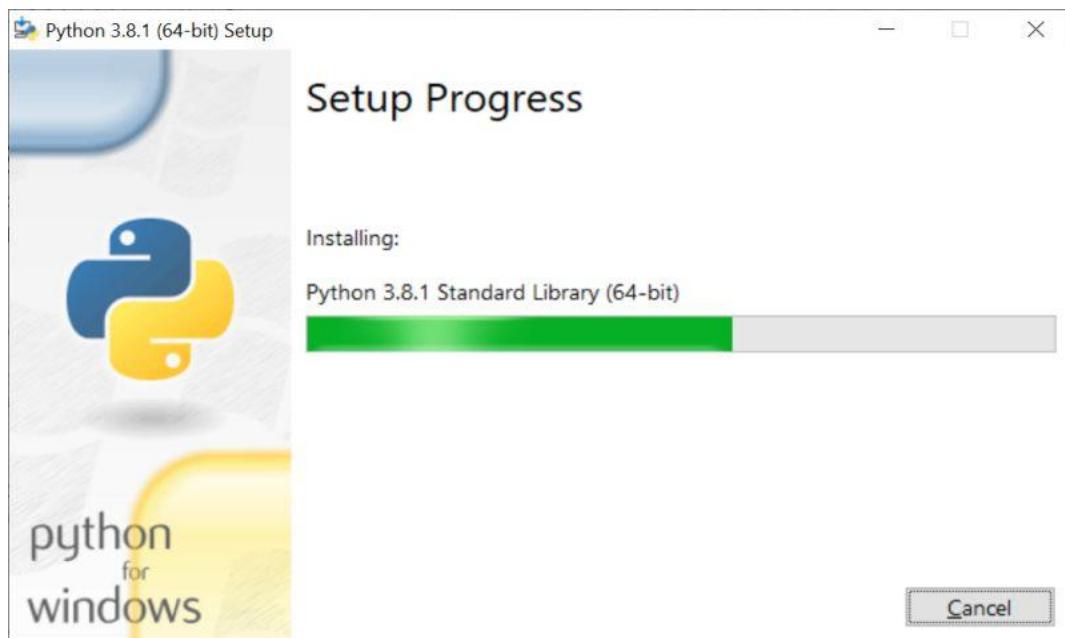
- Download Python từ trang chủ: <https://www.python.org/>



- Chạy file cài đặt sau khi download xong.



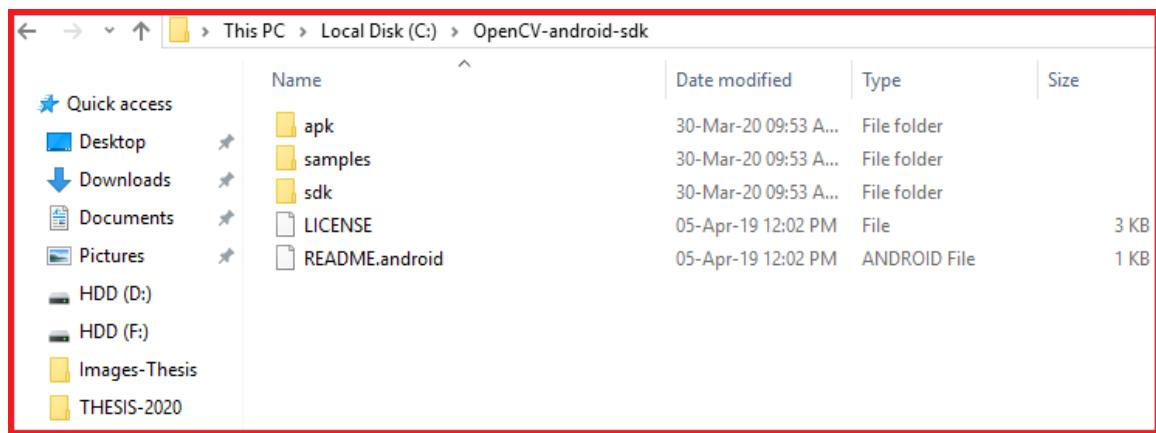
- Check vào mục Add Python 3.8 to PATH để Python được tự động thêm vào biến môi trường, sau đó bấm vào Install Now để bắt đầu cài đặt.



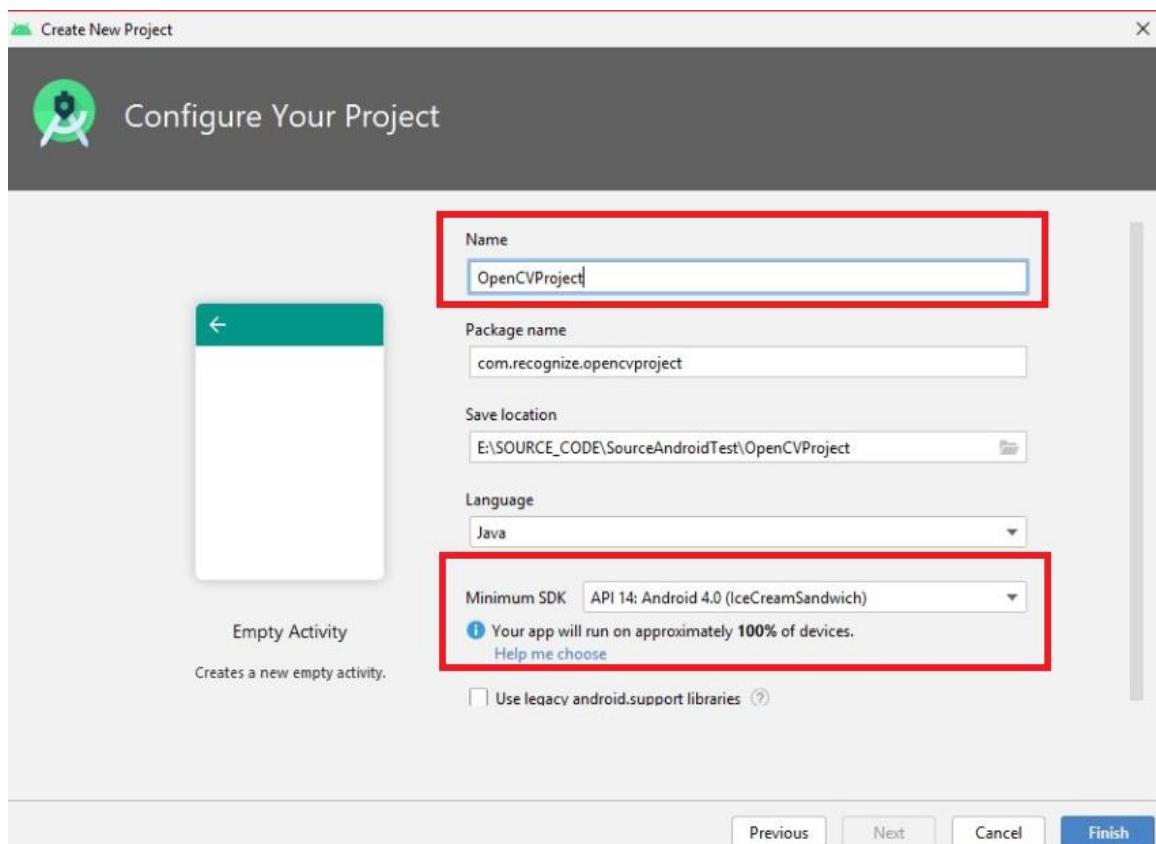
- Sau khi cài đặt hoàn tất, mở command line lên và chạy thử lệnh sau để kiểm tra.

*Tích hợp OpenCV cho Android Studio

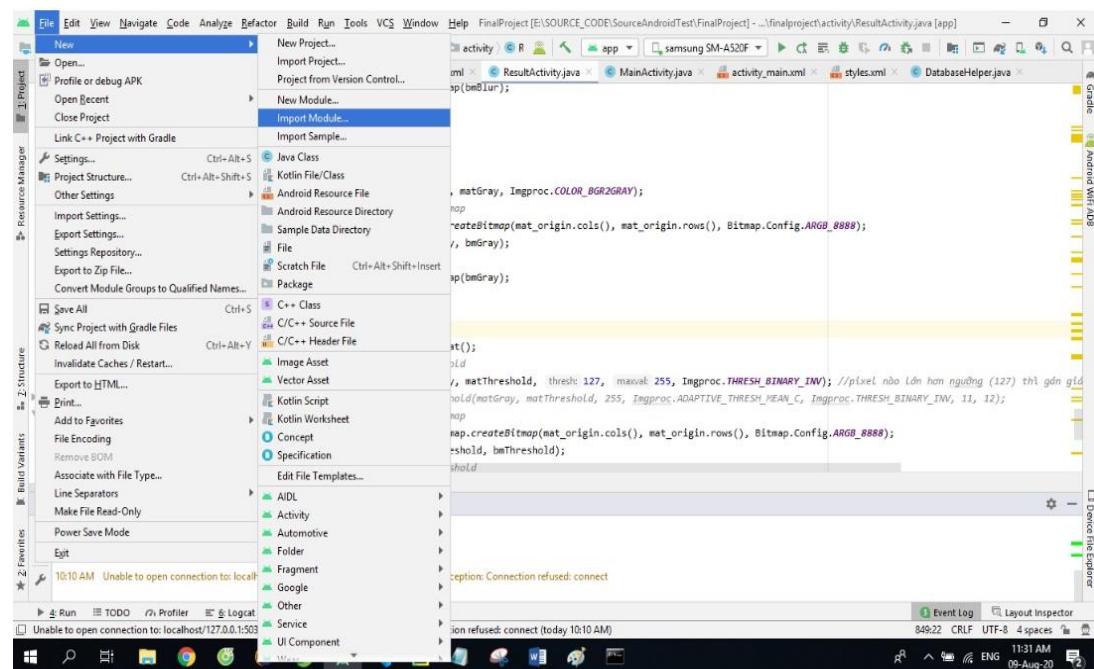
- Truy cập vào link: [OpenCV Library Android](#) vào trang để tải phiên bản mới nhất. Tại thời điểm này phiên bản mới nhất đang là 4.4.0.
- Sau quá trình tải xuống và giải nén file .zip, chúng ta sẽ được một folder như dưới đây:



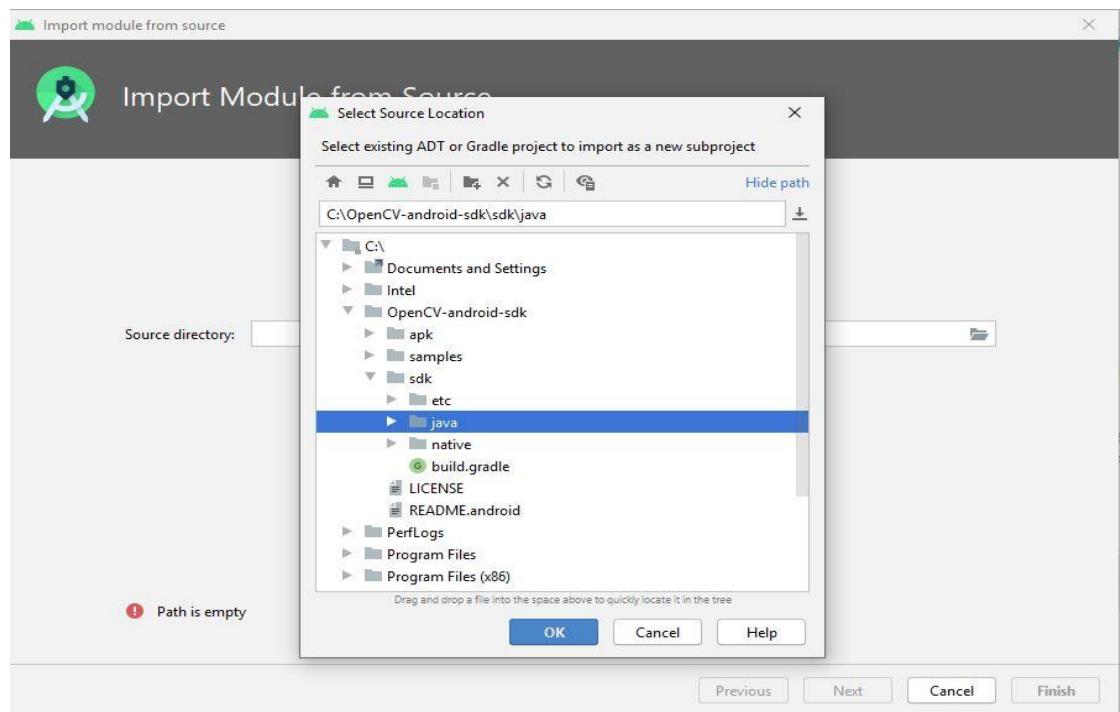
- Tạo mới một dự án



- Sau khi tạo thành công dự án, tiếp theo sẽ tiến hành import OpenCV
- Click: File -> New -> Import Module



- Cửa sổ xuất hiện như hình bên trên bạn có thể chọn đường dẫn tới module mà bạn import.
- Bạn trỏ tới thư mục mà bạn đã giải nén file .zip khi download OpenCV về. Bạn chọn folder SDK tiếp đến là folder Java.



- Sau khi chọn đúng đường dẫn và nhập vào OK, bạn sẽ nhận được một màn hình như hình ảnh bên dưới.

- Nhấn vào Next để đến màn hình tiếp theo. Trên màn hình tiếp theo (hình ảnh bên dưới), bạn nên để các tùy chọn mặc định được chọn và nhấp vào Finish để hoàn tất quá trình nhập mô-đun.

