**ĐẠI HỌC BÁCH KHOA – ĐẠI HỌC ĐÀ NẴNG**

**KHOA ĐIỆN TỬ VIỄN THÔNG – NGÀNH KỸ THUẬT ĐIỆN TỬ VIỄN THÔNG**

--------------------------------------

A picture containing sport

Description automatically generated

*Đà Nẵng,12/2022*

**PBL 4: CHUYÊN ĐỀ HỆ THỐNG MÁY TÍNH**

|  |  |  |  |
| --- | --- | --- | --- |
| **Giáo viên hướng dẫn** | **:** | **TS. Hồ Phước Tiến** |  |
| **Lớp học phần** | **:** | **19N40A** |  |
| **Sinh viên thực hiện** | **:** | **Nguyễn Tiến Dũng** | **106190148** |
|  |  | **Nguyễn Thành Nhơn** | **106190168** |
|  |  | **Đàm Văn Minh** | **106190164** |

***ĐỀ TÀI****:*

Nhận diện dấu vân tay sử dụng mạng nơ – ron tích chập

**THÔNG TIN TÓM TẮT KẾT QUẢ THỰC HIỆN**

**1. Thông tin chung:**

**-** Tên báo cáo: **Báo cáo kết thúc học phần môn PBL4 Chuyên Đề Hệ Thống Máy Tính**

**-** Nhóm sinh viên thực hiện: + Nguyễn Tiến Dũng - MaSV: 106190148

+ Ngô Thành Nhơn - MaSV: 106190168

+ Đàm Văn Minh - MaSV: 106190164

- Thời gian thực hiện: 27/08/2022 – 10/12/2022

**2. Mục tiêu:**

- Hoàn thành các yêu cầu được giao

- Cải thiện kỹ năng sử dụng Python trong xử lý ảnh

- Xây dựng một mô hình nhận diện dấu vân tay sử dụng phương thức mạng nơ-ron tích chập (CNN-Convolution Neural Network)

- Sử dụng module quét vân tay để lấy dấu vân tay từ bên ngoài thực tế để tăng tính trực quan trong xử lý ảnh.

**3. Kết quả:**

- Sử dụng Python với các FrameWork hỗ trợ xử lý ảnh nhằm ứng dụng trong các bước tiền xử lý trước khi mô hình đi tới giai đoạn sử dụng các lớp nơ-ron tích chập

- Hoàn thành dự án “Nhận diện dấu vân tay sử dụng mạng nơ-ron tích chập” và viết báo cáo

**4. Sản phẩm:**

**-** Báo cáo cho dự án

- Mô hình được thiết lập trên máy tính để nhận diện vân tay

**LỜI CẢM ƠN**

Lời đầu tiên, chúng tôi xin gửi lời cảm ơn chân thành tới thầy Hồ Phước Tiến đã tận tình hướng dẫn, chỉ bảo, góp ý và chia sẻ những kiến thức, kinh nghiệm quý báu trong qua trình hướng dẫn cho nhóm chúng tôi nhằm giúp cả nhóm hoàn thành tốt hệ thống mà mình mong muốn.

Chúng tôi xin gửi lời cảm ơn chân thành đến cả những thầy cô trong khoa Điện tử - Viễn thông trường Đại học Bách Khoa – Đại học Đà Nẵng đã tận tình hỗ trợ và giải đáp những thắc mắc từ nhỏ tới lớn trong quá trình chúng tôi thực hiện đề tài. Và đồng thời cũng tạo điều kiện thuận lợi nhất để chúng tôi có thể hoàn thành đề tài một cách trôi chảy.

Chúng tôi xin gửi lời cảm ơn cả những người bạn sinh viên trong khoa Điện tử - Viễn thông đã luôn chia sẻ những kiến thức của bản thân nhằm giải đáp những thắc mắc mỗi khi nhóm chúng tôi gặp rắc rối thông qua đó cũng phần nào giúp nhóm đẩy nhanh hơn tới lúc hoàn thành đề tài.

Trong quá trình tìm hiểu và hoàn thành đề tài, nhóm đã có những lúc thực hiện chưa đạt so với yêu cầu được giao hoặc những sai sót không đáng có trong mô hình, kính mong quý thầy cô bỏ qua và có thể dành những lời góp ý cho nhóm để phát triển hơn trong tương lai.

Một lần cuối chúng tôi xin trân trọng cảm ơn!

**PHÂN CÔNG NHIỆM VỤ**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và Tên** | **Lớp** | **Nhiệm vụ** |
| 1 | Nguyễn Tiến Dũng | 19DTCLC4 | Thu thập dữ liệu  Xử lý dữ liệu  Huấn luyện mô hình  Đánh giá mô hình  Viết báo cáo và làm slide trình bày |
| 2 | Ngô Thành Nhơn | 19DTCLC4 | Thu thập dữ liệu  Xử lý dữ liệu  Huấn luyện mô hình  Đánh giá mô hình  Viết báo cáo và làm slide trình bày |
| 3 | Đàm Văn Minh | 19DTCLC4 | Thu thập dữ liệu  Xử lý dữ liệu  Huấn luyện mô hình  Đánh giá mô hình  Viết báo cáo và làm slide trình bày |

**LỜI MỞ ĐẦU**

Trong những năm trở lại đây kinh tế-xã hội của mỗi một quốc gia thay đổi nhanh với tốc độ chóng mặt. Với việc áp dụng các công nghệ vào mọi lĩnh vực của đời sống làm cho cuộc sống của mỗi chúng ta dễ dàng hơn. Chính vì thế nên đã ra đời những các hoạt động mới phù hợp hơn với thời điểm hiện tại. Nó đòi hỏi các công việc của chúng ta mang tính linh hoạt cao, tính chuyên môn cao mà lại phải dễ dàng thực hiện. Trong số đó, nhận dạng vân tay là một chủ đề cần thiết cho quá trình phát triển trong tương lai, định nghĩa sinh trắc học chắc hẳn không còn xa lạ đối với mỗi người chúng ta. Nhận dạng vân tay đã được sử dụng nhiều trong xác thực điện thoại, an ninh và hơn thế nữa. Nhiều tính năng và thuật toán khác nhau đã được đề xuất để cái thiện việc nhận dạng dấu vân tay. Ở đây chúng tôi đề xuất một mô hình DeepLearning (DL) đó là sử dụng mạng nơ-ron tích chập (Convolution Neural Network) có thể dùng để thực hiện nhận dạng.

Dấu vân tay đã được sử dụng trong nhiều ứng dụng khá nhau như pháp y, xác thực giao dịch, mở khoá điện thoại di động, v. v. Nhiều thuật toán được đề xuất để nhận dạng dấu vân tay là đối xanh dựa trên chi tiếu nhỏ. Các đặc điểm chi tiết thường là các đường vân tay, đường vân nhánh và đường vân ngắn. Dựa vào nhũng hiểu biết của nhóm về Deep Learning chúng ta có thể dựa vào hình dáng của các đường vân để áp dụng cho mạng tích chập nhằm trích xuất những đặc trưng cơ bản của các vân tay trong quá trình nhận diện. Các hình ảnh với độ phân giải tốt sẽ được sử dụng để làm dữ liệu đầu vào, tăng khả năng tách được các đặc trưng thông qua việc sử dụng các lớp Convolution tốt hơn để nhận diện được những hình ảnh có hình dáng vân tay được bóp méo hoặc quay ngược khó hơn.

MỤC LỤC

LỜI MỞ ĐẦU………………………………………………………………………………..4

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT CHUNG 8](#_Toc122638132)

[1.1 Artificial Intelligence(AI), Machine Learning (ML), Deep Learning (DL) 8](#_Toc122638133)

[1.1.1 Artificial Intelligence (AI) : Máy móc với bộ não con người 8](#_Toc122638134)

[1.1.2.Machine Learning (ML):Cách tiếp cận của AI 8](#_Toc122638135)

[1.1.3. Deep Learning(DL): Kỹ thuật của Machine Learning 9](#_Toc122638136)

[1.2. Mạng neural tích chập (Convolutional neural networks - CNN) 9](#_Toc122638137)

[1.2.1. Mạng neural tích chập là gì? 9](#_Toc122638138)

[1.2.2. Convolutional là gì? 9](#_Toc122638139)

[1.2.3. Cấu trúc của mạng CNN 10](#_Toc122638140)

[1.2.3.1.Rút trích đặc trưng 10](#_Toc122638141)

[1.2.3.2.Lớp tích chập (Convolution Layer): 10](#_Toc122638142)

[1.2.3.3.Lớp đường viền (Padding) và bước nhảy (Stride). 11](#_Toc122638143)

[1.2.3.4.Lớp gộp (Pooling Layer): 12](#_Toc122638144)

[1.2.3.5. Optimizer Adam: 13](#_Toc122638145)

[1.2.3.6.Hàm kích hoạt RELU. 14](#_Toc122638146)

[1.2.4.Phân lớp. 15](#_Toc122638147)

[1.2.4.1.Softmax function. 15](#_Toc122638148)

[1.2.4.2.Categorical crossentropy. 16](#_Toc122638149)

[1.2.4.3.Fully Connected Layer. 17](#_Toc122638150)

[CHƯƠNG 2: GIỚI THIỆU TỔNG QUAN MÔ HÌNH 18](#_Toc122638151)

[2.1. Giới thiệu tổng quan: 18](#_Toc122638152)

[2.1.1. Tổng quan về phương thức thực hiện 18](#_Toc122638153)

[2.1.2. Mục tiêu: 20](#_Toc122638154)

[CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH NHẬN DIỆN DẤU VÂN TAY 21](#_Toc122638155)

[3.1. Cài đặt các môi trường thư viện trước khi thực hiện: 21](#_Toc122638156)

[3.1.1. Tạo project bằng cách xử dụng Google Colab 21](#_Toc122638157)

[3.1.2. Chuẩn bị dữ liệu đầu vào: 21](#_Toc122638158)

[3.1.3. Thuật toán sử dụng cho mô hình 21](#_Toc122638159)

[3.2. Triển khai chương trình chấm thuật toán 25](#_Toc122638160)

[3.2.1. Khai báo các thư viện: 25](#_Toc122638161)

[3.2.2. Kết nối với file được sử dụng làm dataset lưu ở Google Driver 25](#_Toc122638162)

[3.2.3. Giai đoạn tiền xử lý 25](#_Toc122638163)

[3.2.4. Sử dụng các lớp tích chập CNN 26](#_Toc122638164)

[3.3 Kết luận chương 35](#_Toc122638165)

[CHUƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ 36](#_Toc122638166)

[4.1. Mô tả dữ liệu 36](#_Toc122638167)

[4.1.1. Kết quả huấn luyện 36](#_Toc122638168)

[4.1.2. Lần train thứ hai: 39](#_Toc122638169)

[4.2. Tiêu chí đánh giá. 40](#_Toc122638170)

[4.3. Đánh giá và phân tích kết quả. 40](#_Toc122638171)

[4.4. Kết luận chương. 41](#_Toc122638172)

KẾT LUẬN ………………………………………………………………………….……...42

TÀI LIỆU THAM KHẢO…………………………………………………………………..43

**DANH MỤC HÌNH ẢNH**

[Hình 1.1: Cửa sổ trượt trên một ma trận 10](#_Toc122637656)

[Hình 1.2: Mô tả cách tính tích chập cho ảnh 12](#_Toc122637657)

[Hình 1.3: Mô tả các lớp padding được thêm vào ảnh 13](#_Toc122637658)

[Hình 1.4: Mô tả quá trình hoạt động của strike đối với lớp kernel 13](#_Toc122637659)

[Hình 1.5: Mô tả cách hoạt động lớp pooling 14](#_Toc122637660)

[Hình 1.6: Mô tả biểu đồ hàm phi tuyến tính ReLU 15](#_Toc122637661)

[Hình 1.7: Mô tả cách hoạt động hàm mất mát (lost\_function) 17](#_Toc122637662)

[Hình 2.1: Dữ liệu đầu vào được sử dụng để làm dataset 19](#_Toc122637663)

[Hình 2.2: Hình ảnh vân tay được sử dụng làm dữ liệu, một vài vân tay bị làm biến dạng để tăng độ khó cho mô hình 20](#_Toc122637664)

[Hình 3.1: Dữ liệu đầu vào được sử dụng để làm dataset chia là 2 file 22](#_Toc122637665)

[Hình 3.2: Lớp Convolutional và các tham số 23](#_Toc122637666)

[Hình 3.3: Hoạt động của EPOCH khi huấn luyện mô hình 23](#_Toc122637667)

[Hình 3.4: Kiểm tra mô hình bằng một ảnh bất kỳ 24](#_Toc122637668)

[Hình 3.5: Visualization được sử dụng để theo dõi hoạt động của các lớp CNN 25](#_Toc122637669)

[Hình 3.6: Sơ đồ khối miêu tả các hoạt động phía trên 25](#_Toc122637670)

[Hình 4.1: Thông số hiển thị các tham số trong mô hình 37](#_Toc122637671)

[Hình 4.2: Kết quả các đợt huấn luyện cho mô hình 38](#_Toc122637672)

[Hình 4.3: Sơ đồ cho theo dõi quá trình có bị over-fitting hay không 38](#_Toc122637673)

[Hình 4.4: Hình ảnh được đem đi kiểm tra và kết quả dự đoán 39](#_Toc122637674)

[Hình 4.5: Quá trình hoạt động của các lớp CNN 39](#_Toc122637675)

[Hình 4.6: Thử huấn luyện lại mô hình lần hai 40](#_Toc122637676)

[Hình 4.7: Sơ đồ cụ thể cho đợt huấn luyện 41](#_Toc122637677)

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT CHUNG

## 1.1 Artificial Intelligence(AI), Machine Learning (ML), Deep Learning (DL) [1]

- Có thể nói, những thuật ngữ như trí tuệ nhân tạo AI, học máy Machine Learning và học sâu Deep Learning đều là những thuật ngữ phổ biến trong thời đại công nghiệp 4.0. Những thuật ngữ này nghe qua thì chúng có rất nhiều điểm giống nhau nhưng thực tế ba công nghệ này lại hoàn toàn khác biệt nhau.

- Trên thực tế những công nghệ này đều có vai trò rất tích cực tới ngành công nghiệp sản xuất, máy móc nên đôi khi chúng ta thường đánh đồng và có những quan niệm sai lầm về ba khái niệm trên. Nếu xét về thời gian xuất hiện thì trí tuệ nhân tạo AI chính là ý tưởng xuất hiện sớm nhất tiếp theo đó là sự ra đời của học máy machine learning và cuối cùng gần đây nhất chính là Deep learning. Mặc dù xuất hiện muộn nhất nhưng Deep learning lại chính là thứ đang thúc đẩy sự bùng phát của AI hiện nay.

1.1.1 Artificial Intelligence (AI) : Máy móc với bộ não con người[1]

- AI có thể được coi là ngành khoa học của máy tính liên quan đến việc tự động hóa các hành vi thông minh. Với AI, trí tuệ được tạo ra có thể tư duy, suy nghĩ và học hỏi, xử lí dữ liệu ở mức rộng hơn, quy mô hơn, khoa học hơn và nhanh hơn so với con người. AI mang lại những lợi ích to lớn với ngành công nghiệp máy móc, khoa học máy tính tuy vậy chính bản thân AI cũng còn tồn tại rất nhiều hạn chế. Hiện nay những gì chúng ta đang phát triển chỉ dừng ở AI hẹp. Công nghệ này có khả năng thực hiện các nhiệm vụ cụ thể một cách giống hoặc thậm chí tốt hơn con người.

1.1.2. Machine Learning (ML):Cách tiếp cận của AI[1] - Machine Learning là một thuật ngữ rộng để chỉ hành động bạn dạy máy tính cải thiện một nhiệm vụ mà nó đang thực hiện. Khả năng cơ bản nhất của machine learning là sử dụng thuật toán để phân tích những thông tin có sẵn, học hỏi từ nó rồi đưa ra quyết định hoặc dự đoán về một thứ gì đó có liên quan. Thay vì tạo ra một phần mềm với những hành động, hướng dẫn chi tiết để thực hiện một nhiệm vụ cụ thể, máy tính được “huấn luyện” bằng cách sử dụng lượng dữ liệu và các thuật toán để học cách thực hiện nhiệm vụ.

- Machine learning ra đời làm giảm bớt những hạn chế vốn có của AI khi nó mang lại cho máy tính khả năng có thể tìm ra mọi thứ mà không được lập trình rõ ràng. Ngày nay học máy đã chứng tỏ được sự hữu ích của mình qua vô vàn những ứng dụng tích cực.

1.1.3. Deep Learning(DL): Kỹ thuật của Machine Learning[1]

- Deep learning là loại machine learning mà trong đó máy tự đào tạo chính nó. Deep learning đòi hỏi rất nhiều dữ liệu đầu vào và sức mạnh tính toán hơn là machine learning. Công nghệ này cũng đã bắt đầu được triển khai bởi các tập đoàn công nghệ lớn như Facebook, Amazon…

- Có thể nhận thấy đến nay Deep learning đã cho phép ứng dụng nhiều vấn đề thực tế của máy đồng thời mở rộng lĩnh vực tổng thể của trí tuệ nhân tạo. Deep learning phá vỡ các cách thức con người làm việc bằng cách làm cho tất cả các loại máy móc trợ giúp có thể thực hiện được, gần hoặc giống hệt con người. Với sự trợ giúp của Deep learning AI có thể có thể bùng nổ một cách mạnh mẽ.

1.2. Mạng neural tích chập (Convolutional neural networks - CNN)

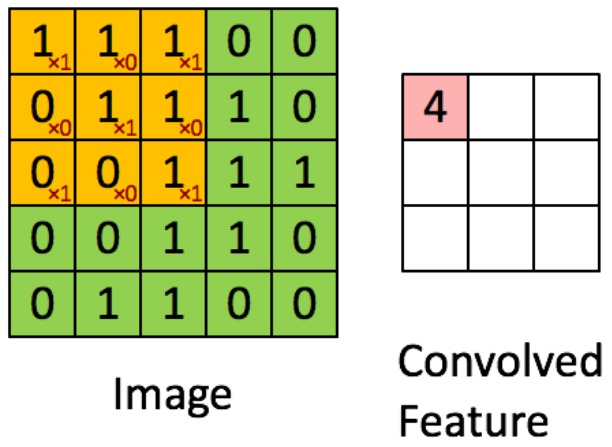
1.2.1. Mạng neural tích chập là gì?

- Convolutional neural network(CNN – Mạng neural tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay[2].

- CNN được sử dụng nhiều trong các bài toán nhận dạng object trong ảnh.

1.2.2. Convolutional là gì?

**-** Là một cửa sổ trượt (sliding windows) trên một ma trận mô tả như hình dưới:



Hình 1.1: Cửa sổ trượt trên một ma trận

- Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.

- Trong hình ảnh ví dụ trên, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước 5×5 và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột.

- Convolution hay tích chập là nhân từng phần tử trong ma trận 3x3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là 3×3.

- Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhận ma trận Filter với ma trận ảnh 5×5 bên trái.

1.2.3. Cấu trúc của mạng CNN

1.2.3.1.Rút trích đặc trưng

- Khi sử dụng thuật toán CNN so sánh hình ảnh theo từng mảnh, mỗi mảnh đó được gọi là Feature. Mỗi Feature được xem như một hình ảnh mini hay gọi là những mảng hai chiều nhỏ. Các Feature được khớp với những khía cạnh chung trong bức ảnh đó (nghĩa là Feature sẽ tương ứng với khía cạnh nào đó của hình ảnh và chúng sẽ khớp lại với nhau) [3].

#### 1.2.3.2.Lớp tích chập (Convolution Layer):

- Tích chập là một khối quan trọng trong CNN. Lớp này sẽ giúp trích xuất được những thông tin quan trọng từ dữ liệu. Thuật ngữ tích chập được dựa trên một phép hợp nhất toán học của hai hàm tạo thành hàm thứ ba. Phép toán này kết hợp hai tập thông tin khác nhau.

- Để dễ hình dung phương pháp tính tích chập cho ảnh, giả sử ta có một hình mang thông tin giống chữ X có kích thước 9 x 9 pixel và cửa sổ tích chập là 3 x 3. Muốn tính toán sự khớp của một feature đối với một mảnh của hình ảnh, ta chỉ cần nhân mỗi điểm ảnh trong feature với giá trị của điểm ảnh tương ứng trong mảnh hình ảnh. Sau đó cộng tổng lại và chia cho số lượng điểm ảnh trong feature. Nếu cả hai điểm ảnh màu trắng (giá trị 1) thì 1 x 1 = 1. Nếu cả hai đều là màu đen, thì (-1) x (-1) = 1. Dù bằng cách nào, mỗi điểm ảnh mà khớp thì đều cho ra kết quả 1. Tương tự như vậy, bất kỳ cái nào không khớp đều ra -1. Nếu tất cả các điểm ảnh trong feature đều khớp, thì cộng tổng lại rồi chia cho số điểm ảnh thì sẽ ra là 1. Tương tự, nếu không có điểm ảnh nào trong feature khớp với mảnh hình ảnh, thì kết quả là -1.Trong hình dưới đây, thành phần kernel/filter (màu xanh lá) trượt trên đầu vào (màu xanh dương) và kết quả được trả về bản đồ đặc trưng (màu đỏ). Kernel/filter có kích thước là 3×3 trong ví dụ này.

Diagram

Description automatically generated

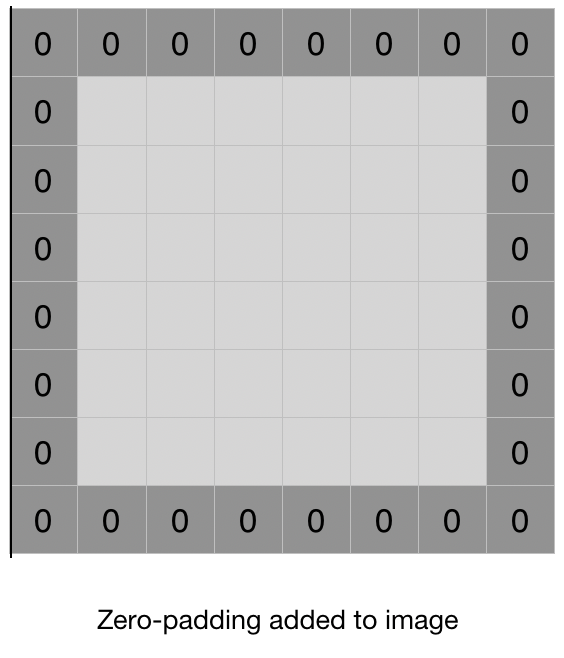
*Hình 1.2: Mô tả cách tính tích chập cho ảnh*

- Để hoàn tất tích chập, chúng ta lặp lại quá trình này, xếp các feature với mọi mảnh hình ảnh có thể. Ta có thể lấy kết quả từ mỗi tích chập và tạo một mảng hai chiều mới từ đó, dựa vào vị trí của mỗi mảnh nằm trong hình ảnh. Bản đồ các phần khớp nhau này cũng là một phiên bản đã được lọc từ hình ảnh gốc. Nó là một bản đồ thể hiện nơi tìm thấy feature trong hình ảnh. Các giá trị gần 1 cho thấy sự khớp mạnh, các giá trị gần -1 cho thấy sự khớp mạnh với âm bản của feature, và các giá trị gần bằng 0 cho thấy không khớp với bất kỳ loại nào[4].

- Bước tiếp theo là lặp lại toàn bộ quá trình chập cho từng feature khác. Kết quả là một tập hợp các hình ảnh đã được lọc, mỗi cái ứng với mỗi filter của chúng ta. Sẽ thuận tiện khi nhìn toàn bộ các thao tác tích chập như là một bước xử lý duy nhất. Trong CNN cái này được gọi là một layer (lớp) tích chập, cho thấy gián tiếp rằng sẽ sớm có các layer khác được thêm vào[4].

#### 1.2.3.3.Lớp đường viền (Padding) và bước nhảy (Stride).

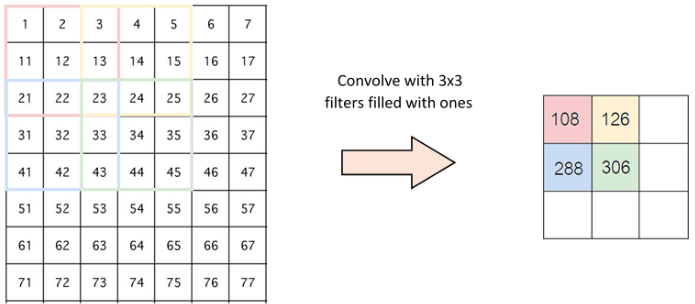
- Nếu muốn ma trận thu được sau phép tính Convolution có kích thước không đổi, chúng ta sẽ thêm các phần tử 0 ở ngoài rìa như hình sau:



Hình 1.3: Mô tả các lớp padding được thêm vào ảnh

- Khi này, ta gọi viền 0 ở ngoài cùng là Padding = 1(Thêm 1 viền 0 ở ngoài cùng của ma trận X). Nếu Padding = 2, 3, ….thì thêm 2,3 viền số 0 ở ngoài.

- Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy.



Hình 1.4: Mô tả quá trình hoạt động của strike đối với lớp kernel

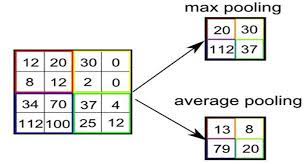
#### 1.2.3.4.Lớp gộp (Pooling Layer):

- Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng[5]. Các pooling có thể có nhiều loại khác nhau, nhưng phổ biến nhất hiện nay có 2 loại là:

+ Max Pooling

+ Average Pooling

- Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, average pooling lấy tổng trung bình của các phần tử trong ma trận đó:



Hình 1.5: Mô tả cách hoạt động lớp pooling

- Như ta thấy ở trên thì max pooling lấy giá trị lớn nhất của ma trận 2x2, sau đó thì dịch sang 2 pixel và tiếp tục. Còn average thì lấy tổng trung bình của các phần tử trong ma trận 2x2 đó.

#### 1.2.3.5. Optimizer Adam:

- Adam optimizer hiện là một thuật toán sử dụng rộng rãi nhất kết hợp kỹ thuật của RMS prop và momentum. Thuật toán sử dụng hai internal states momentum (m) và squared momentum (v) của gradient cho các tham số. Sau mỗi batch huấn luyện, giá trị của m và v được cập nhật lại sử dụng exponential weighted averaging[6].

- Mã giải của việc cập nhật m và v:

𝑚𝑡 = 𝛽1𝑚𝑡−1 + (1 − 𝛽1)𝑔 (1.2.3.1)

𝑣𝑡 = 𝛽2𝑣𝑡−1 + (1 − 𝛽2)𝑔𝑡2 (1.2.3.2)

- Trong đó, 𝛽 được xem như là một siêu tham số.**Công thức cập nhật theta:**

𝜃𝑡 = 𝜃𝑡 − 1 − 𝛼 (1.2.3.3)

- Trong đó, 𝛼 là learning rate, là giá trị được thêm vào để ngăng việc chia cho 0.

- Để việc descent được thực hiện nhanh hơn, thuật toán đã sử dụng hai kỹ thuật:Tính exponential moving average của giá trị đạo hàm lưu vào biến m và sử dụng nó là tử số của việc cập nhật hướng. Với ý nghĩa là nếu m có giá trị lớn, thì việc descent đang đi đúng hướng và chúng ta cần bước nhảy lớn hơn để đi nhanh hơn. Tương tự, nếu giá trị m nhỏ, phần descent có thể không đi về hướng tối tiểu và chúng ta nên đi 1 bước nhỏ để thăm dò. Đây là phần momentum của thuật toán.

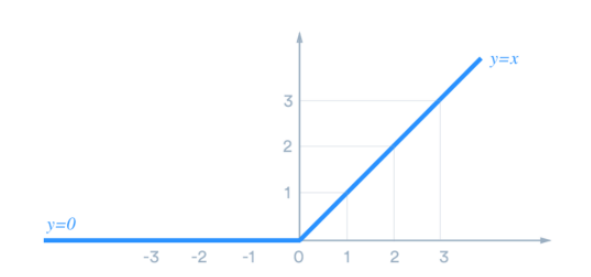
- Tính exponential moving average của bình phương gía trị đạo hàm lưu vào biến v và sử dụng nó là phần mẫu số của việc cập nhật hướng. Với ý nghĩa như sau: Giả sử gradient mang các giá trị dương, âm lẫn lộn, thì khi cộng các giá trị lại theo công thức tính m ta sẽ được giá trị m gần số 0. Do âm dương lẫn lộn nên nó bị triệt tiêu lẫn nhau. Nhưng trong trường hợp này thì v sẽ mang giá trị lớn. Do đó, trong trường hợp này, chúng ta sẽ không hướng tới cực tiểu, chúng ta sẽ không muốn đi theo hướng đạo hàm trong trường hợp này. Chúng ta để v ở phần mẫu vì khi chia cho một giá trị cao, giá trị của các phần cập nhật sẽ nhỏ, và khi v có giá trị thấp, phần cập nhật sẽ lớn. Đây chính là phần tối ưu RMSProp của thuật toán.

- Ở đây, m được xem như là moment thứ nhất, v xem như là moment thứ hai, nên thuật toán có tên là “Adaptive moment estimation”.

- Để lý giải vì sao Adam lại hội tụ nhanh hơn so với SGD, chúng ta có thể giải thích như sau: Exponential weighted averaging cho chúng ta giá trị xấp xỉ gradient mượt hơn qua mỗi lần lặp, dẫn tới tăng tính dừng. Sau đó, việc chia cho căng bậc 2 của giá trị v làm số lước của chúng ta giảm mạnh khi phương sai của giá trị gradient tăng lên. Điều này, như giải thích ở trên, có nghĩa là, khi hướng đi của mô hình chỉ ra không rõ ràng, thuật toán Adam thực hiện các bước đi nhỏ coi như là thăm dò thôi. Và sẽ thực hiện các bước đi lớn, nhanh khi hướng đi rõ ràng.

#### 1.2.3.6.Hàm kích hoạt RELU.

- Hàm ReLU được viết tắt từ Rectified Linear Unit.



Hình 1.6: Mô tả biểu đồ hàm phi tuyến tính ReLU

- Hàm relu kích hoạt tuyến tính được điều chỉnh đơn giản hơn so với sigmoid được thể hiện bằng công thức toán học sau:

y= (1.2.3.4)

Nhận xét:

- Hàm ReLU activation đơn giản để tính => thời gian train model nhanh hơn.

- Đạo hàm là 1 với x >= 0 nên không bị vanishing gradient.

- Tuy nhiên với các node có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0, hiện tượng đấy gọi là “Dying ReLU“. Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ node đấy cũng không được cập nhật với gradient descent.

### 1.2.4.Phân lớp.

- Trong phần phân lớp, ta sử dụng một vài tầng với kết nối đầy đủ để xử lí kết quả của phần tích chập. Vì đầu vào của mạng liên kết đầy đủ là 1 chiều, ta cần làm phẳng đầu vào trước khi phân lớp. Tầng cuối cùng trong mạng CNN là một tầng liên kết đầy đủ, phần này hoạt động tương tự như mạng nơ-ron thông thường.

- Kết quả thu được cuối cùng cũng sẽ là một véc-tơ với các giá trị xác suất cho việc dự đoán như mạng nơ-ron thông thường.

#### 1.2.4.1.Softmax function.

Cách tính hàm softmax[7]:

- Tính hàm lũy thừa số e, với số mũ là những số đã cho

- Tính tổng các lũy thừa đó. Đó sẽ là mẫu số.

- Sử dụng lũy thừa của mỗi số là tử số.

- Xác suất sẽ là tử số/mẫu số

𝜎= (1.2.4.1)

Trong đó:

|  |  |
| --- | --- |
| 𝑧⃗ | Giá trị vector nhập vào cho hàm softmax, (từ z0 đến zk) |
|  | Tất cả các giá trị z đều là giá trị vector nhập cho hàm softmax. Chúng có thể là bất cứ số thực nào, số dương, số âm hay số 0. Ví dụ một mạng thần kinh nhân tạo có thể có giá trị vector ra là (0.62,8.12,2.53). Đây không phải là phân phối xác suất đúng. Đó là vì sao ta cần đến hàm softmax. |
|  | Hàm lũy thừa tiêu chuẩn được áp dụng cho mỗi giá trị nhập. Nó sẽ đưa ra một giá trị dương lớn hơn 0. Giá trị này sẽ rất nhỏ nếugiá trị nhập là âm, và rất lớn nếu giá trị nhập dương. Tuy nhiên, nó sẽ không cố định trong khoảng (0:1]. Đây là yêu cầu củamột xác suất. |
|  | Dòng phía dưới của công thức là một cụm chuẩn hóa. Nó đảm bảo rằng tổng của cácgiá trị ra sẽ luôn bằng 1 và nằm trong khoảng (0:1].Như vậy, sẽ xuất hiện phân phối xác suất chính xác. |

- Hàm Softmax được sử dụng trong mô hình hồi quy logistic phân loại.

- Trong xây dựng neural network (mạng thần kinh nhân tạo), hàm softmax được sử dụng trong phân nhiều lớp

* Tính chất:

- Xác suất sẽ luôn nằm trong khoảng (0:1].

- Tổng tất cả các xác suất bằng 1.

* Lợi ích:

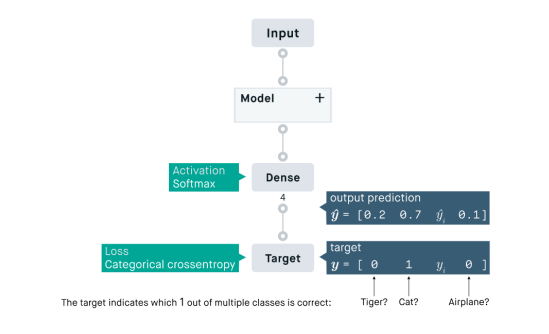
- Hàm softmax là tối ưu khi tính toán xác suất tối đa trong tham số mô hình.

- Tính chất của hàm softmax khiến hàm phù hợp với sự thông dịch xác suất, rất hữu ích trong Machine Learning (Học máy).

- Chuẩn hóa softmax là một cách để giảm thiểu ảnh hưởng của những giá trị cực trị hay dữ liệu ngoại lai trong dữ liệu mà không phải chỉnh sửa dữ liệu ban đầu.

#### 1.2.4.2.Categorical crossentropy.

- Categorical crossentropy là một hàm loss function được sử dụng trong phân loại nhiều lớp, thường được sử dụng với hàm kích hoạt softmax.



Hình 1.7: Mô tả cách hoạt động hàm mất mát (lost\_function)

-Công thức:

(1.2.4.2)

Trong đó:

|  |  |
| --- | --- |
|  | Là giá trị lỗi của mẫu |
| *i* | Là mẫu thử thứ i trong tập |
| *j* | Là chỉ số nhãn/đầu ra |
| 𝑖 | Là giá trị dự đoán |
| *y* | Là giá trị chính xác |
| *output size* | Là số giá trị vô hướng của đầu ra model |

#### 1.2.4.3.Fully Connected Layer.

- Tầng cuối cùng của mô hình CNN trong bài toán phân loại ảnh là tầng fully-connected layer. Tầng này có chức năng chuyển ma trận đặc trưng ở tầng trước thành vector chứa xác suất của các đối tượng cần được dự đoán. Ví dụ, trong bài toán phân loại số viết tay MNIST có 10 lớp tương ứng 10 số từ 0-1, tầng fully connected layer sẽ chuyển ma trận đặc trưng của tầng trước thành vector có 10 chiều thể hiện xác suất của 10 lớp tương ứng.

- Và cuối cùng, quá trình huấn luyện mô hình CNN cho bài toán phân loại ảnh cũng tương tự như huấn luyện các mô hình khác. Chúng ta cần có hàm độ lỗi để tính sai số giữ dự đoán của mô hình và nhãn chính xác, cũng như sử dụng thuật toán backpropagation cho quá trình cập nhật trọng số[8].

# CHƯƠNG 2: GIỚI THIỆU TỔNG QUAN MÔ HÌNH

## 2.1. Giới thiệu tổng quan:

### 2.1.1. Tổng quan về phương thức thực hiện

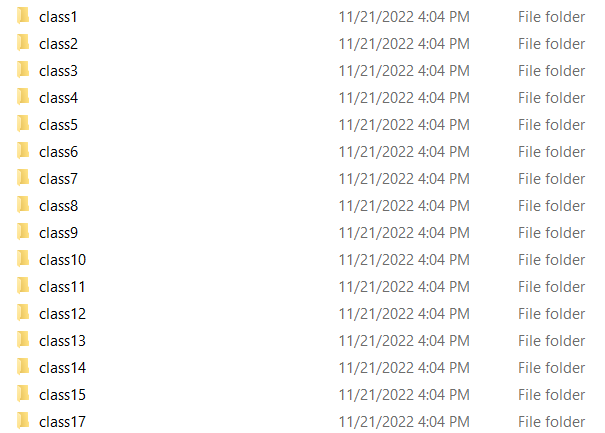
a. Ý tưởng bài toán:

**Đề bài của bài toán:**

- Chúng ta tạo một hệ thống nhận diện dấu vân tay với ý tưởng đầu vào khoảng 20-30 lớp vân tay khác nhau đặc thù cho 20-30 người và trong đó mỗi người chúng ta sẽ lấy khoảng 70-90 vân tay

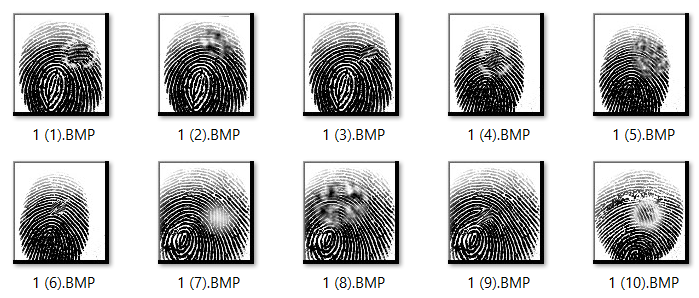
- Tiếp theo đó chúng ta sẽ xây dựng một mô hình sử dụng phương thức CNN, Mạng nơ-ron tích chập để trích xuất đặc trưng cho ảnh đầu vào và xử lý cái đặc trưng đấy bằng những lớp phía sau như Max-Pooling và các Hidden Layer để lưu lại những đặc trưng cho việc Training của mô hình.

- Demo một vào dữ liệu đầu vào được sử dụng trong mô hình



Hình 2.1: Dữ liệu đầu vào được sử dụng để làm dataset

- Chúng tôi đã chia các class, với mỗi class được chia ra để mô hình Train hiểu như là một người cụ thể, nó sẽ được trả về tên class khi chúng ta chọn một hình ảnh để test.

**

Hình 2.2: Hình ảnh vân tay được sử dụng làm dữ liệu, một vài vân tay bị làm biến dạng để tăng độ khó cho mô hình

- Hình ảnh ví dụ cho các vân tay trong class được sử dụng từ một nguồn dataset sẽ được sử dụng để Train. Để tăng khả năng học cho mô hình, có thể thấy một số hình ảnh vân tay đã được làm mờ trên một vùng nhất định hoặc xoay tròn một vùng các đường vân ở vài chỗ khoanh tròn để tăng độ thử thách cho độ chính xác của mô hình.

b. Thuật toán cho bài toán sử dụng mạng tích chập:

- Bước 1: Xử lý ảnh đầu vào với các “class” được đánh số riêng cho mỗi class nhằm phân biệt thủ công theo các nhãn của ảnh theo như ban đầu.

- Bước 2: Tăng cường ảnh bằng cách sử dụng phương pháp Open CV cổ điển để chuyển ảnh về dạng các bit nhị phân 0 và 1 nhằm giảm kích thước cho quá trình tiền xử lý. Đặt cho ảnh một kích thước chung đầu vào cụ thể (Nên là hình vuông) để thuận tiện cho quá trình làm việc của bộ lọc Kernel.

- Bước 3: Thực hiện thiết lập các lớp CNN và các lớp Max Pooling nhằm trích xuất các đặc trung của ảnh và gom chúng lại - ở đây nhóm chúng tôi sẽ sử dụng cụ thể là 3 lớp CNN không nhiều quá cũng không ít quá để có thể lấy hết được các đặc trưng cho ảnh thì việc sử dụng nhiều lớp CNN sẽ tốt hơn trong việc tách được nhiều đặc trưng của ảnh hơn. Theo sau các lớp CNN là các lớp Hidden Layer được sử dụng như là lớp Flatten và Dense.

- Bước 4: Công việc phía sau chỉ đơn thuần là công việc của mô hình CNN làm việc, các lớp Kernel sẽ được tự động lựa chọn bởi mô hình và sẽ chạy hết một lượt EPOCH để Train cho cả 3 lớp Conv2D chúng ta sẽ lặp lại nhiều lần Train – càng train nhiều thì mô hình càng học được nhiều, từ đó chúng ta có thể in ra các giá trị về hàm Loss hay hàm Accuracy để biết tính đúng đắn của mô hình. Train càng nhiều độ chính xác càng tăng lên và độ sai lệch sẽ càng giảm xuống.

- Bước 5: Một loạt hàm được sử dụng để đem một bức ảnh bất kỳ đem tới để đối sánh và cho mô hình nhận diện sau khi đã được học với nhiều lần Train. – Kết quả cho thấy sẽ là vị tri mà ảnh được nhận dạng liệu là đúng hay sai. Một hàm được sử dụng để tạo ra tỉ lệ dự đoán đúng cho bức ảnh đầu vào.

- Bước 6: Quá trình cuối cùng tuỳ vào yêu cầu của chúng ta, chúng ta có thể in ra các layer của mô hình để biết được mô hình CNN mà chúng ta sử dụng đã thực hiện những công việc gì. Ngoài ra, chức năng Visualization có thể được sử dụng để hiển thị các hình ảnh sau mỗi lần lọc qua CNN sẽ nhận được kết quả là gì nhằm giúp chúng ta hiểu kỹ hơn về công việc mà mô hình CNN đang hoạt động.

c. Tiến hành thực hiện dựa trên chương trình cụ thể:

- Sau khi xử lý ảnh để làm dữ liệu cho mô hình và phương pháp thực hiện chúng ta sẽ tiến hành viết chương trình và thực hiện training với test cụ thể để nhận được kết quả mong muốn.

### 2.1.2. Mục tiêu:

- Mục tiêu chính: Đạt được một mô hình nhận diện ảnh ứng dụng công nghệ mạng nơ - ron tích chập đạt kết quả nhận diện cao nhất, chuẩn xác nhất.

- Mục tiêu phụ:

+ Thu thập dữ liệu

+ Xử lý dữ liệu

+ Xây dựng thuật toán

+ Viết chương trình cho thuật toán

+ Tiến hành chạy thử mô hình và đánh giá kết quả

# CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH NHẬN DIỆN DẤU VÂN TAY

# 3.1. Cài đặt các môi trường thư viện trước khi thực hiện:

### 3.1.1. Tạo project bằng cách xử dụng Google Colab

- Tạo một project với tên “fingerprint3”

- Tạo file lưu trữ các dữ liệu được sử dụng làm dataset cho được dẫn tới file này

+ File train được lưu trữ những ảnh thuộc các class được sử dụng để train cho mô hình.

+ File valid được lưu trữ những hỉnh ảnh không nhãn bất kỳ sẽ được sử dụng để đối sánh trong mô hình.

- Các thư viện có thể sẽ được cài đặt:

+ Thư viện keras, tensorflow

+ Thư viện Numpy

+ Thư viện Matplotlib

…….

### 3.1.2. Chuẩn bị dữ liệu đầu vào:

- Dữ liệu đầu vào chính là những bức ảnh được quét ra từ vân tay của một người cụ thể, chúng được sử dụng làm cơ sở để train trực tiếp cho mô hình.

- Một vài bức ảnh sẽ được tăng độ khó bằng cách làm mờ hay làm méo mó các vân tay nhằm gây khó khăn để biết kết quả nhận được từ mô hình.

- Một file ảnh với các bức ảnh không nhãn sẽ được tạo nhằm mục đích cho chương trình chạy nhận dạng dấu vân tay.

### 3.1.3. Thuật toán sử dụng cho mô hình

**- Bước 1:** Xử lý ảnh đầu vào với các “class” được đánh số chia theo như ban đầu. Chúng ta cần một file dataset cho các dữ liệu đầu vào để thuận tiện cho việc học và việc kiểm tra của mô hình. Chúng tôi đã chuẩn bị 2 file nhằm gán nhãn riêng biệt cho từng bức ảnh thuộc mỗi người riêng biệt.



Hình 3.1: Dữ liệu đầu vào được sử dụng để làm dataset chia là 2 file

+ File train được lưu trữ những ảnh thuộc các class được sử dụng để train cho mô hình.

+ File valid được lưu trữ những hỉnh ảnh không nhãn bất kỳ sẽ được sử dụng để đối sánh trong mô hình.

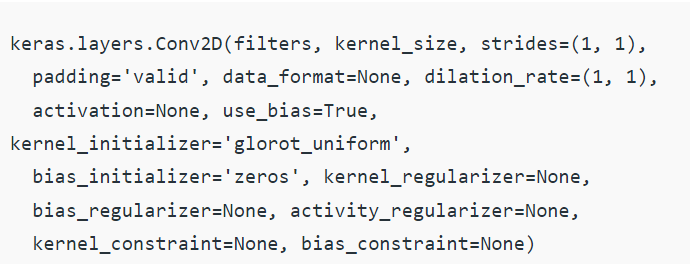
**- Bước 2:** Tăng cường ảnh bằng cách sử dụng phương pháp Open CV cổ điển để chuyển ảnh về dạng các bit nhị phân 0 và 1 nhằm giảm kích thước cho quá trình tiền xử lý. Đặt cho ảnh một kích thước trung đầu vào cụ thể (Nên là hình vuông) để thuận tiện cho quá trình làm việc của bộ lọc Kernel.

+ Chúng tôi đã tăng tường ảnh để huấn luyện đầu vào bằng cách chia nhỏ các bức ảnh thành các ảnh nhị phân, việc phải xử lý ảnh đầu vào là ảnh RGG sẽ là quá lớn cho môi hình.

+ Thêm nữa là chúng tôi đã tiến hành chuyển các ảnh được train thành các bức ảnh có cùng kích thước và trả ảnh về dạng ảnh 2D

**- Bước 3:** Thực hiện thiết lập các lớp CNN và các lớp Max Pooling nhằm trích xuất các đặc trung của ảnh và gom chúng lại - ở đây nhóm chúng tôi sẽ sử dụng cụ thể là 3 lớp CNN, sẽ là không nhiều quá cũng không ít quá để có thể lấy hết được các đặc trưng cho ảnh thì việc sử dụng nhiều lớp CNN sẽ tốt hơn trong việc tách được nhiều đặc trưng của ảnh hơn. Theo sau các lớp CNN là các lớp Hidden Layer được sử dụng như là lớp Flatten và Dense.

+ Các giá trị bộ lọc lần lượt ở các lớp CNN là (32, 64, 128), các giá trị của bộ lọc nên là thuộc luỹ thừa 2 làm giá trị, thông số của các lớp kernel là 3x3, có thểm một vài những thông số phía sau mạng tích chập nhằm tăng cường độ chính xác cho mạng.



Hình 3.2: Lớp Convolutional và các tham số

+ Ngoài ra còn các lớp ẩn với 512 nơ-ron thuộc lớp ẩn và 29 tập dữ liệu được sử dụng là đầu ra tương đương với 29 nhãn.

**- Bước 4:** Công việc phía sau chỉ đơn thuần là công việc của mô hình CNN làm việc, các lớp Kernel sẽ được tự động lựa chọn bởi mô hình và sẽ chạy hết một lượt EPOCH để Train cho cả 3 lớp Conv2D chúng ta sẽ lặp lại nhiều lần Train – càng train nhiều thì mô hình càng học được nhiều, từ đó chúng ta có thể in ra các giá trị về hàm Loss hay hàm Accuracy để biết tính đúng đắn của mô hình. Train càng nhiều độ chính xác càng tăng lên và độ sai lệch sẽ càng giảm xuống.

+ Một hàm model.fit được sử dụng để huấn luyện mô hình, ở đây chúng tôi sẽ cần tới một giá trị EPOCH cho sẵn mà chúng tôi tìm kiếm nhằm đặt được mô hình train cho hệ thống để tránh over-fitting, giá trị EPOCH không nên quá nhiều để làm mô hình trở nên phức tạp dẫn dến sự học một cách không tự nhiên của mô hình

+ Tương tự với EPOCH là giá trị của BATCH, sau mỗi EPOCH mô hình sẽ chạy một lượt BATCH với mỗi lần mô hình sẽ học số ảnh đầu vào là bằng giá trị của ảnh chia đều cho số BATCH



Hình 3.0.3: Hoạt động của EPOCH khi huấn luyện mô hình

**- Bước 5:** Một loạt hàm được sử dụng để đem một bức ảnh bất kỳ đem tới để đối sánh và cho mô hình nhận diện sau khi đã được học với nhiều lần Train. Kết quả cho thấy sẽ là vị tri mà ảnh được nhận dạng liệu là đúng hay sai. Một hàm được sử dụng để tạo ra tỉ lệ dự đoán đúng cho bức ảnh đầu vào.

+ Chọn một bức ảnh bất kỳ mất nhãn, đem đi kiểm tra với mô hình đã học, ảnh được truyền vào nên cùng kích thước với số ảnh đã được học ở đầu mô hình

+ Có thể tạo một hàm dự đoán dựa trên tỉ lệ phần trăm đúng đối với nhãn class đã được train

Graphical user interface

Description automatically generated

Hình 3.4: Kiểm tra mô hình bằng một ảnh bất kỳ

**- Bước 6:** Quá trình cuối cùng tuỳ vào yêu cầu của chúng ta, chúng ta có thể in ra các layer của mô hình để biết được mô hình CNN mà chúng ta sử dụng đã thực hiện những công việc gì. Ngoài ra, chức năng Visualization có thể được sử dụng để hiển thị các hình ảnh sau mỗi lần lọc qua CNN sẽ nhận được kết quả là gì nhằm giúp chúng ta hiểu kỹ hơn về công việc mà mô hình CNN đang hoạt động.

+ Để hiểu rõ hơn về mô hình – việc biết được sau mỗi lớp CNN ảnh sẽ trông như thế nào chúng tôi đã sử dụng một tính năng vượt trội của mạng nơ-ron là Visualization nhằm đem lại những đường nét cụ thể được trích xuất ra từ mô hình

Background pattern

Description automatically generated

Hình 3.5: Visualization được sử dụng để theo dõi hoạt động của các lớp CNN

* **Bước 7:** Cuối cùng là chúng ta cần phải hiển thị được các kết quả đầu ra và kết quả dự đoán của mô hình, các kết quả train, các biểu đồ được thêm vào nhằm tăng tính xác thực cho khả năng học của mô hình

**Sơ đồ khối mô tả phương pháp hoạt động:**

**Diagram

Description automatically generated**

Hình 3.6: Sơ đồ khối miêu tả các hoạt động phía trên

## 3.2. Triển khai chương trình chấm thuật toán

### 3.2.1. Khai báo các thư viện:

A screenshot of a computer

Description automatically generated with medium confidence

### 3.2.2. Kết nối với file được sử dụng làm dataset lưu ở Google Driver

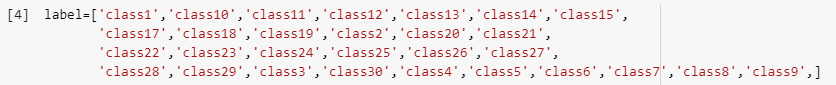


- Tạo một đường dẫn tới file train và file valid

Graphical user interface, text

Description automatically generated

- Gán nhãn cho các class được sử dụng



### 3.2.3. Giai đoạn tiền xử lý

Text

Description automatically generated

- Sử dụng thư viện ImageDataGenerator nhằm tăng cường hình anh trong khi huấn luyện mô hình.

- Để tăng cường cho ảnh chung ta sử dụng hàm để chuyển hình ảnh về dạng bit map với các giá trị thấp hơn vì hình ảnh gốc có thể là ảnh RGB nó sẽ lớn và khiến cho mô hình bị xử lý chậm lại.

Text

Description automatically generated

- Ở đây ta có thể thấy biến train\_data\_gen được nhắc tới qua hàm flow\_from\_directory(), hàm này là một chức năng của lớp ImageDataGenerator, thông qua hàm này sẽ lấy các ảnh sau khi đưa vào được dẫn tạo ra một lô dữ liệu tăng cường.

- Cụ thể hoá nó bằng việc chuyển ảnh về một kích thước cố định 48x48 và sử dụng lệnh class\_mode để chuyển ảnh về dạng 2D

🡪 Giai đoạn tiền xử lý là một bước không thể thiếu cho một mô hình CNN hoạt động tốt, chúng ta sẽ tránh được những rủi do như là ảnh bị khác loại về kích thước, ảnh quá lớn do các pixel ảnh RGB bị lớn sẽ khiến mô hình học bị chậm hơn và với một bức ảnh 2D cũng sẽ dễ dàng xử lý hơn là ảnh 3D.

### 3.2.4. Sử dụng các lớp tích chập CNN

Graphical user interface, text

Description automatically generated

- Như đã dự định từ trước, việc tạo ra nhiều lớp CNN là cần thiết, ở đây chúng tôi đang sử dụng 3 lớp CNN và thử nghiệm trên 4 lớp nhưng kết quả đầu ra không thực sự có thêm sự khác biệt so với ba lớp nên để tăng tốc độ học của mô hình chúng tôi đã làm cho mô hình có 3 lớp CNN đầu vào

- Việc sử dụng một tham số Conv2D sẽ bắt buộc có một số lượng bộ lọc mà các lớp tích chập sẽ học từ đó, việc sử dụng giá trị bộ lọc là ngẫu nhiên nhưng trong các lớp tích chập được khuyến khích sử dụng với giá trị bộ lọc là nằm trong luỹ thừa 2, vì vậy chúng tôi đã sử dụng với 32 lớp bộ lọc ở lớp thứ nhất, lớp thứ hai và ba lần lượt là 64 và 128 bộ lọc. Ngoài ra các tham số bên trong nó như (3, 3) là giá trị kích thước của một lớp kernel, kích thước lớp kernel này sẽ làm tăng thời gian lọc của ảnh nhưng sẽ trích xuất được nhiều giá trị đặc trưng của ảnh hơn trên mỗi lần lọc. Tham số kích hoạt **“activation”**, tham số này đơn giản là một chuỗi tham số thuận tiện cho phép chúng ta cung cấp 1 chuỗi. Bởi vì các mạng nơ – ron là các mạng tuyến tính, việc xếp chồng các lớp CNN sẽ là vô nghĩa bời vì thực chất sau các lớp CNN mạng nơ ron vẫn là các mạng tuyến tính như ban đầu vậy thì khi đó chúng ta sử dụng một hàm kích hoạt tuyến tính để làm tăng tính trực quan cho mô hình mạng nơ - ron. Cuối cùng là **“input\_shape”** được sử dụng ở lớp đầu tiên nhằm tạo một hình dạng ảnh được truyền vào mô hình là bằng với kích thước của ảnh đã được xử lý ttrong quá trình tiền xử lý, và chỉ cần sử dụng cho lớp CNN đầu tiên còn các lớp sau sẽ tự hiểu được kích thước truyền vào từ lớp đầu và sử dụng lại.

- Lớp **max-pooling** kich thước 2x2 được sử dụng để gom những đặc trưng còn lại sau khi đi qua bộ lọc bằng cách lấy những giá trị lớn nhất của ảnh sau khi lọc, kết quả của lớp pooling sẽ là giảm kích thước của ảnh nhưng vẫn giữ được các đặc trưng của ảnh từ đó sẽ giảm thiểu các phép tính trong ảnh. Và nhờ vào phép pooling kích thước ảnh giảm đi, lớp CNN phía sau sẽ đọc được ảnh với kích thước lớn hơn vì vậy mô hình sẽ học được các thuộc tính lớn hơn.

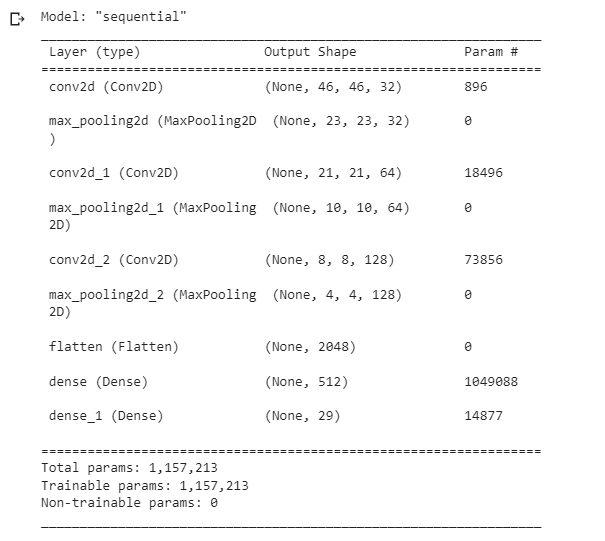
🡪 Có thể nhận thấy khối lượng không gian đầu ra qua mỗi lớp CNN sẽ càng giảm đi bởi vì số bộ lọc tăng lên sẽ khiến cho ảnh ngày càng nhỏ hơn. Nhưng nhờ vào lớp pooling, mô hình cũng có thể học được các thuộc tính lớn hơn.

- Lớp **Flatten()** được dùng để trải các ảnh dầu vào thành một mặt phẳng – ghép các ảnh thành một đường thẳng nhau. Có thể hiểu cụ thể hơn là phương thức Flatten() đang cố loại bỏ tất cả các kích thước ảnh đầu vào và chuyển về một mảng 1 chiều, điều này sẽ làm giảm kích thước cầu đầu vào thành 1 lớp. Việc một hình ảnh kích thước 48x48 được đưa vào sẽ trở thành 48 lớp gồm 48 mạng nơ – ron là không hợp lý, vậy thì chúng ta sẽ làm phẳng nó và biến nó thành 2304x1 sẽ dễ dàng xử lý hơn.

- Lớp **Dense** thứ nhất với 512 nơ – ron lớp ẩn và 29 là số nhãn của tập dữ liệu đầu ra. sử dụng một Args là “**activation=softmax**” được hiểu như một trình phân loại được thêm vào mạng và đầu ra nhận được sẽ là các giá trị dự đoán của mô hình. Sau khi ảnh được chuyển thành các vector 1 chiều qua hàm flatten() thì chúng tôi đã sử dụng hàm softmax(), hàm softmax chuyển đổi một vector giá trị thành một phân phối xác xuất, ảnh đầu vào sẽ được ánh xạ trở thành một giá trị cố định, thông qua đó hàm softmax sẽ đưa ra kết quả dự đoán cho mô hình.



- Dưới đây là các thông số summary của mô hình được in ra để tóm tắt cấu trúc mạng



- Số parameter được tính bằng công thức dưới đây, với ảnh đầu vào kích thước là 48x48x3, sau mỗi lớp CNN ảnh đầu vào sẽ tăng lên thành 32 và 64 là các giá trị nơ – ron của lớp CNN. 1 ở đây là giá trị bias (trọng số kết nối độ lệch)

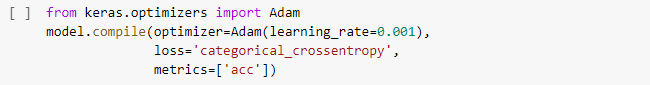


- Còn với lớp Flatten, giá trị parameter sẽ được tính với công thức đầu ra của lớp pooling cuối cùng là



- Parameter thuộc lớp Dense tính theo giá trị từng ảnh tại lớp Flatten()

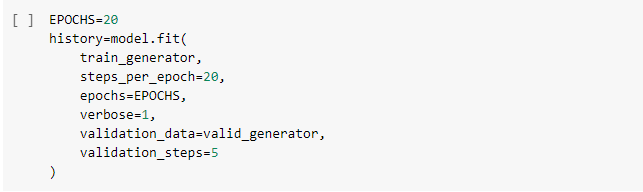




- Tiếp theo là hàm compile, về cơ bản hàm optimizer trong thư viện Adam được xem như một hàm tiền xử lý cho quá trình học của mô hình, trong đó hàm optimizer được xem như một trình tối ưu hoá các chuỗi cho mô hình, nó sẽ lựa chọn với các giá trị tốt nhất cho việc học. Nghĩa là nếu bạn học được kết quả đầu ra là 0.6 nhưng trong một trường hợp mà mô hình bị overfiting các giá trị có thể nhảy loạn xạ, khiến cho mô hình học bị sai lệch với giá trị bị giảm đi chẳng hạn như 0.3 thì hàm optimizer sẽ hoạt động như một cách chọn lọc, nó sẽ loại bỏ các kết quả với giá trị thấp hơn kết quả học phía trước và yêu cầu mô hình học lại cho đến khi kết quả học được tốt hơn và tiếp tục các bước học tiếp theo, điều này khiến cho mô hình học luôn nhận được các kết quả học khả quan và đi đúng hướng học của việc thiết kế mô hình ban đầu.

- Mô hình với fit(), chúng ta chỉ định hàm mất mát (loss), trình tối ưu hoá(optimizer – phải được xem là một chuỗi) và tuỳ chọn, một số chỉ số cần theo dõi. Đầu ra nhận được từ trình tối ưu hoá là các giá trị loss và arc, hai giá trị này được hiển thị qua biểu đồ nhằm phản ánh mức độ ngày càng tốt lên đối với kết quả học (accuracy) và độ sai lệch thông qua hàm mất mát (loss) là ngày càng giảm. Điều đó cho thấy mô hình đang ngày càng trở nên thông minh hơn và đầy đủ hơn.

+ categorical\_accuracy: Dùng cho nhiều class, nếu y\_true == y\_predict (dự đoán) thì sẽ trả về 1 và ngược lại



- Mô hình được huấn luyện bằng cách cho một giá trị EPOCH cố định - ở đây chúng ta có thể hiểu Epoch là số lần lặp của mô hình và trong một lần lặp (hay còn gọi là một epoch) thì mô hình sẽ chia nhỏ các bước để lặp chúng ta gọi nó là batch là việc chia một lần lặp thành nhiều phần để lặp và chúng tôi đã đặt cho batch một giá trị là 20

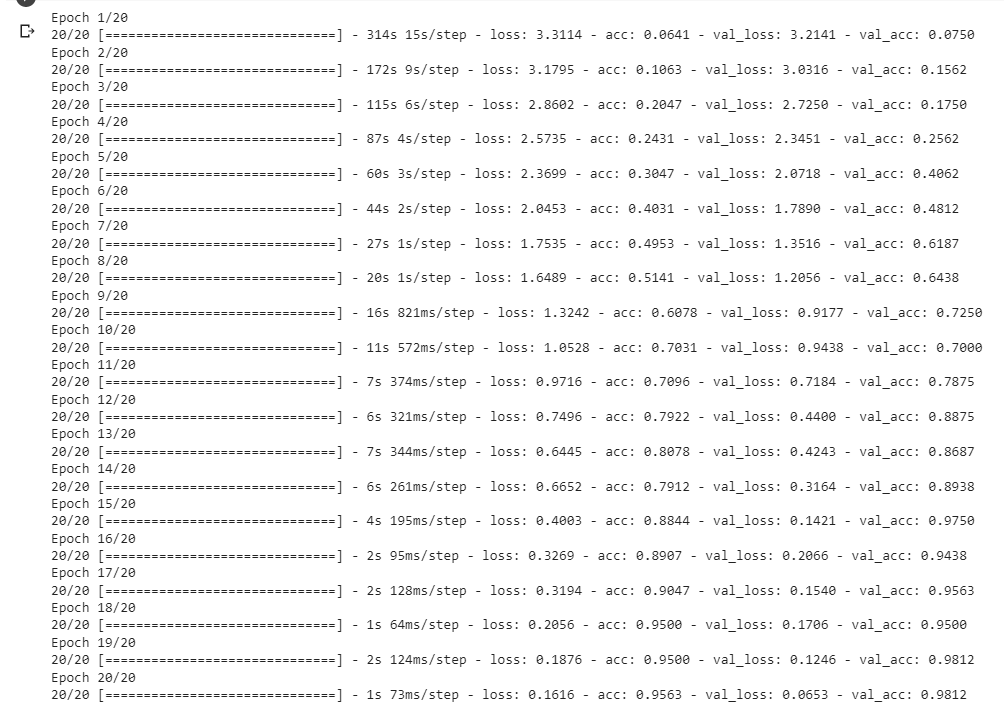
+ steps\_per\_epoch có thể xem như là batch nghĩa là tổng số bước nhảy trước khi train xong một epoch để bắt đầu một epoch mới, với giá trị truyền vào là 20 nghĩa là giả xử nếu dữ liệu đầu vào xử dụng lên tới 1000 ảnh thì giá trị batch mỗi lần sẽ train là 1000/20 là 50 cái ảnh với một lần học.

+ verbose = 1 chúng ta có thể hiểu các giá tị được ánh xạ bởi số trong đó 0 là im lặng, 1 là thanh tiến trình làm việc, 2 là một dòng trên mỗi lần bắt đầu làm việc 🡪 mặc định ở đây là 1 trong hầu hết các trường hợp.

+ validation\_data được gán trực tiếp vào nơi dữ liệu sử dụng để đánh giá cho việc mất mát và bất kỳ chỉ số nào ở cuối mỗi epoch

+ validation\_step là tổng số bước trước khi mô hình dừng lại, thực hiện việc xác thực cuối mỗi epoch. Về cơ bản thì việc tăng cường dữ liệu của keras sẽ luôn sinh ra ảnh để cho epoch chạy, vậy thì chúng ta cần phải tạo điểm dừng cho nó, bằng cách cài đặt cụ thể các bước chạy của mỗi epoch là validation\_step. Ngoài ra thì hàm fit() sẽ luôn test trên validation của nó theo số batch và số batch chạy trên một epoch có thể hiểu là số step, vậy nên nếu số step được cài đặt thấp xuống nó sẽ làm giảm thời gian test nên khuyến khích giá trị của validation\_steps nên thấp xuống.

* Dưới đây là kết quả sau một lần train với giá trị accuracy ngày càng tăng lên và giá trị loss giảm xuống gần về 0.



Text

Description automatically generated

- Hàm save được sử dụng để lưu lại model trong quá trình test và làm việc với model (ghi nhớ những ảnh đã test – giá trị và kết quả model)

- Tiếp sau đó là tạo một biểu đổ sử dụng thư viện Matplot Lib với các thông số sau 20 lần train epoch để tạo thành một biểu đồ trực quan quá trình train của mô hình

🡪 Có thể thấy mô hình đà tăng là accuracy còn giảm là loss thông qua đó kết lận mô hình sẽ học và nhận diện chính xác hơn đối với nhiều lần train hơn.

Chart, line chart

Description automatically generated

Text

Description automatically generated

- Một hàm được sử dụng để chọn một ảnh bất kỳ từ nơi khác đem đi đối xánh với mô hình train thông qua đó phân biện được nhãn của ảnh đó.

+ Hàm imshow sẽ hiển thị ra bức ảnh được chọn

+ Sử dụng một hàm để load ảnh và chuyển ảnh đó về kích thước 48x48 thông qua hàm target\_site=(48, 48).

+ Chuyển các ảnh thành ma trận sử dụng lệnh img\_to\_array() của thư viện keras

+ Thư viện numpy qua hàm expand\_dimps() sẽ mở rộng hình dạng của ảnh ban đâu, và tràn vào một chục mới xuất hiện ở bên hông của hình ảnh nhằm hiển thị kích thước ảnh.

+ Chúng ta có thể cho mô hình dự đoán phần chăm chính xác của mô hình khi chúng ta chọn các bức ảnh bằng cách sử dụng dữ liệu đã đào tạo để chạy thử nghiệm bằng cách dùng hàm predict() ngoài ra chúng ta định nghĩa cụ thể hơn cho hàm bằng cách sử dụng một args đó là batch\_site có thể hiển là số lượng mẫu ảnh được sử dụng mỗi đợt.

Text

Description automatically generated with medium confidence



- Model.layers được dùng để hiển thị ra các lớp CNN đang sử dụng, để người dùng biết quá trình mà hệ thống hoạt động

Text

Description automatically generated

Logo

Description automatically generated with low confidence



- Output\_layers là in ra các thông số được sử dụng thông qua các lớp CNN – nó bao gồm kiểu của các tệp tin, kích thước như được thu nhỏ dần sau mỗi lần thực hiện tích chập.

Text

Description automatically generated

- Cuối cùng chúng ta có thể thực hiện in ra màn hình quá tình mà chúng ta đang làm bằng cách sử dụng visualization. Mô hình sẽ hiển thị một bức ảnh, và tất cả mọi thứ về bức ảnh trong quá trình tích chập.

Graphical user interface, text, application

Description automatically generated

- Đầu tiên chúng ta cần upload ảnh lên sử dụng thư viện upload của colab.

- Tiếp theo chúng ta sử dụng load\_img để load ảnh với kích thước (48,48), sau đó chuyển ảnh thành mảng bằng img\_to\_array().

- Sau đó thay đổi kích thước của ảnh một lần nữa bằng reshape và khởi tạo đối tượng visualization\_model với đầu vào và đầu ra bằng tf.keras.models.Model().

- Tạo đối tượng successive\_feature\_maps để lấy ảnh dự đoán đầu vào và layer\_names để lấy tên các lớp từ model.layers.

- Tạo vòng lặp for:

+ Tạo 1 điều kiện if, nếu điều kiện thoả mãn thì tạo ra 2 đối tượng n\_features để lấy kích thước cuối mảng, size để lấy kích thước thứ 2 của mảng và display\_grid để tạo ra một ma trận mới với kích thước (size, size\*n\_features) với hàm np.zeros.

+ Sau đó ta tạo thêm 1 vòng lặp for trong điều kiện if để biến đổi kích thước ảnh, trong vòng lặp này ta dùng các hàm như mean() để tính giá trị trung bình, std() để tính độ lẹch chuẩn, sau khi tính toán xong thì ta sẽ giới hạn các pixels trong khoảng (0,255) bằng hàm clip().

+ Cuối cùng ta hiển thị các ảnh theo kích cỡ và tên các lớp như conv2d, max\_pooling ở trong layer\_name từ model.layers.

A picture containing timeline

Description automatically generated

## 3.3 Kết luận chương

- Xây dựng hoàn thiện một mô hình nhận diện dấu vân tay với 3 lớp Convolution, 29 class với 2405 đối tượng được sử dụng để train và 290 đối tượng được sử dụng để test.

- Đáp ứng yêu cầu của bài toán đặt ra lúc ban đầu.

- Theo nhận định, còn nhiều vấn đề bất cập, sai lệch trong quá trình nhận diện, tốn thời gian cho việc train bởi vì mô hình nặng về số lớp CNN.

- Mô hình còn nhiều vấn đề cần khắc phục.

# CHUƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

## 4.1. Mô tả dữ liệu

### 4.1.1. Kết quả huấn luyện

- Sau khi hoàn thành mô hình với các tiêu chí đề ra ở chương phía trước, chúng tôi đã tiến hành cho mô hình chạy thử, với dữ liệu đầu vào được kiểm tra chính xác là 2405 bức ảnh chia thành 29 class sẽ được xử lý và huấn luyện trực tiếp trên mô hình

Table

Description automatically generated

Hình 4.1: Thông số hiển thị các tham số trong mô hình

- Có thể thấy bảng thông số từ model.fit() là không đổi nó hiển thị ra thông số trong quá trình tích chập, sau mỗi lớp tích chập thì kích thước của ảnh sẽ giảm đi một nữa và só lớp filter sẽ tăng lêm nhằm giám sát kỹ hơn những đặc trưng bởi vì qua vài lần CNN số lượng và kích thước anh sẽ rất là nhỏ. Sau khi có được bảng thống kê các giá trị para cụ thể mô hình sẽ thực hành huấn luyện trực tiếp

Table

Description automatically generated

Hình 4.2: Kết quả các đợt huấn luyện cho mô hình

- Chúng tôi đã train file datast với 20 epoch và 20 batch, nhìn vào bảng thống kê chúng ta có thể thấy các giá trị accuracy tăng lên và là giá trị tốt giúp nhận diện ảnh chính xác hơn còn giá trị loss giảm gần về không đây là hàm ánh xạ sự sai lệch trong quá trình nhận dạng nghĩa là mô hình đang hoạt động tốt theo như dự tính ban đầu. Hai sơ đồ bên dưới sẽ hiển thị rõ hơn các giá trị bằng hàm kích hoạt ReLU được sử dụng.

Chart, line chart

Description automatically generated

Hình4.3: Sơ đồ cho theo dõi quá trình có bị over-fitting hay không

- Tiếp theo chúng ta sẽ cần một bài kiểm tra cho mô hình, chúng ta sẽ lựa chọn ngẫu nhiên 1 bức ảnh không nhãn trong 290 bức ảnh bất kỳ để kiểm tra mô hình hoạt động tốt hay không.

Graphical user interface

Description automatically generated with medium confidence

Hình 4.4: Hình ảnh được đem đi kiểm tra và kết quả dự đoán

- Một hình ảnh vân tay bất kỳ đã được chọn và cho ra kết quả trong khoảng thời gian là 0,15s và kết quả là hoàn toàn chính xác, chúng ta có thể đem bức ảnh đi đối chiếu với bức ảnh trong class1 và nhận được kết quả là chính xác, vậy mô hình đã hoạt động và kết quả nhận diện là khả quan cho bước đầu kiểm tra.

A screenshot of a computer

Description automatically generated with medium confidence

Hình 4.5: Quá trình hoạt động của các lớp CNN

- Cuối cùng là quá trình visualization đuợc sử dụng để in ra quá trình vân tay được xử lý và tách các đặc trưng của nó. Bức ảnh nhỏ dần sau mỗi một lớp Conv2D, các đặc trưng ban đầu còn nhìn thấy tương đối rõ, nhưng dần về cuối mô hình chỉ trích xuất ra được những đặc trưng còn lại rất nhỏ và khó có thể hình dung được nếu không có sự hỗ trợ của Viasualization trong mô hình

### 4.1.2. Lần train thứ hai:

- Để xác nhận về vấn đề huấn luyện chúng tôi đã cho mô hình chạy lại một lần nữa để xem mô hình có lưu lại các kết quả huấn luyện từ lần huấn luyện đầu tiên hay không

Table

Description automatically generated

Hình 4.6: Thử huấn luyện lại mô hình lần hai

* Có thể thấy cả từ những EPOCH đầu tiên mô hình đã có thể nhận được kết quả huấn luyện là đúng và hàm mất mát khả quan hơn so với lần huấn luyện thứ nhất. Bản thân chúng tôi trong quá trình huấn luyện trước mô hình cũng nhận thấy sự thay đổi rõ rệt, mô hình đã hoạt động nhanh hơn, chỉ mất 40s để mô hình hoàn thành hết 20 lần lặp, con số này là nhanh hơn rất nhiều so với 30 phút cho lần huấn luyện đầu tiên. Nhưng điều này đã gây ra một vấn đề về kết quả huấn luyện lần 2, dường như mô hình đang học một cách không phải là học từng kết quả một, mà mô hình đang tỏ ra là học vẹt, để hiểu cụ thể hơn thì mô hình đang ghi nhớ những kết quả từ lần huấn luyện 1 và sử dụng cho kết quả lần huấn luyện thứ 2, việc đó có thể khiến cho mô hình không hoạt động đúng với một mô hình nhận dạng vì mô hình sẽ tự động gán cho bức ảnh một cái nhãn mặc dù nhãn đó là sai hoặc đúng.

Chart, line chart

Description automatically generated

Hình 4.7: Sơ đồ cụ thể cho đợt huấn luyện

Có thể thấy sau lần train thứ nhất thì những lần sau giá trị nhận được luôn là tiệm cận là 1 đối với accuracy và 0 đối với loss, nhưng chúng tôi vẫn khuyến khích là không nên train quá nhiều bởi vì mô hình nhận diện đặc tính của nó là tự học vậy nên rất có thể mô hình thay vì học thêm những đặc tính mới sẽ học một cách ghi nhớ tạm thời nên mô hính rất dễ xảy ra lỗi.

## 4.2. Tiêu chí đánh giá.

- Tiêu chí đánh giá mô hình sẽ được sử dụng dựa trên thước đo Accuracy và hàm mất mát Loss-function, việc thước đo Accuracy sẽ nhận được kết quả là phần trăm cho thấy mô hình huấn luyện là tốt hay chưa tốt và với hàm mất mát sẽ đồng thời biết được các giá trị sai phạm có thể xảy ra là bao nhiêu.

## 4.3. Đánh giá và phân tích kết quả.

- Nhìn chung thì chúng tôi đã hoàn thành được mô hình một cách chi tiết và cụ thể, cho thấy các giá trị và biểu đồ tính toán trong quá trình train. Kết quả mô hình nhận được là bảng thông số các parameter cụ thể cho thấy các giá trị biến hoặc ảnh đưa vào mô hình train, một bảng kết quả huấn luyện trực quan với kết quả ban đầu là tương đối thấp với kết quả là 0.037 và sau nhiều lần lặp của EPOCH cuối cùng nhận được kết quả đúng với giá trị acc là 0.95 ngoài ra hàm loss trên tập train là 0.2031, các giá trị mất mát còn tương đối cao bởi vì có thể kết quả là khả quan đối với thước đo Accuracy nhưng bời vì mô hình với lượng ảnh đầu vào lớn thì việc kết quả sai lệch sẽ đem lại những kết quả kiểm tra sai nhiều. Nhóm chúng tôi cũng đã thực hiện kiểm tra đối với nhiều bức hình khác nhau và nhận được độ chính xác là khá cao. Tuy nhiên, với việc mô hình sẽ gặp vài kết quả là sai với những hình ảnh được làm mờ hay làm biến dạng vân tay thì sẽ xảy ra hiện tượng nhận diện sai nghĩa là mô còn gặp nhiều điểm yếu. Ngoài ra, nếu ví dụ như mô hình nếu train ít sẽ nhận được các giá trị accuracy thấp hơn nhưng nếu train nhiều mô hình sẽ sinh ra thói học vẹt bằng cách là tuy mô hình nhận diện là sai nhưng nó vẫn báo là đúng là bởi vì máy đang tự lưu một phần bộ nhớ của máy đã học.

- Ngoài việc mô hình có kết quả loss còn đang chưa thực sự tối ưu thì việc nhận diện nhìn chung là khả quan, về các giá trị của sơ đồ cũng cho thấy là mô hình không bị gặp tình trạng Over fitting bời vì các giá trị nhận về thông số huấn luyện tương đối là chi tiết, không có việc giá trị Accuracy đang ở tầm trung khoảng 0.6 mà nhảy thẳng lên 0.998.

## 4.4. Kết luận chương.

**-** Thông qua chương 4 chung tôi đã trình bày cụ thể hơn cách mà trương trình hoạt động thông qua giải thích code, trình bày các kết quả training, thử trên nhiều ảnh khác nhau và cho chạy nhiều lần

- Song, đôi khi gặp một vài lỗi phát sinh thuộc về sự logic của máy thêm nữa là vấn đề ví dụ như là train quá nhiệu hoặc ảnh khó thì sẽ xảy ra lỗi trong quá trình thực hiện. Ngoài những điều đó thì nhìn chung mô hình đang hoạt động tốt đúng với mục tiêu của nhóm đề ra.

- Qua đó cũng đã thể hiện được điểm mạnh và điểm hạn chế trong mô hình và cả những trường hợp đặc biệt

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Trong dự án này, chúng tôi đã tập trung phát triển dự án theo phương thức CNN, và cho thấy sự hiệu quả nhất định đối với mô hình CNN trong quá trình xử lý các bài toán về dữ liệu lớn. Ảnh đầu vào là ảnh đen trắng đã được chúng tôi xử lý sơ qua để làm ảnh đầu vào, nhìn chung thì quá trình tạo cơ sở dữ liệu luôn là một trong những bước cấp thiết trong những bước đầu tạo một mô hình CNN tốt, chúng tôi đã luôn gặp một vài rắc rối nhỏ về vấn đề đặt tên cho ảnh hay là gán nhãn. Để so sánh với một phương thức gần nhất đối với CNN đó là Yolo, thì có thể thấy việc tạo cơ sở dữ liệu cho CNN sẽ dễ dàng hơn, việc gán nhãn sẽ là tự động không cần phải làm thủ công như Yolo. Tiếp theo là quá trình xây dựng hệ thống, hệ thống CNN mà chúng tôi xây dựng bao gồm 3 lớp mạng nơ – ron, bời vì đầu vào là khá lớn đối với một mô hình nhận dạng, vậy nên chúng tôi đã nâng số lớp sao cho phù hợp với quá trình train để tránh gặp tình trạng over-fitting hay under-fitting, như đã được học thì nếu mô hình quá phức tạp thì sẽ dẫn đến over-fitting còn mô hình quá đơn giản sẽ trở nên under-fitting. Thời gian để máy học không quá lâu, việc xử lý dữ liệu cho lần đầu tiên rơi vào khoảng gần 30 phút không phải quá lâu, bởi vì mô hình đã thực hiện nhiều bước. Về kết quả nhận được, mô hình đã học và kiểm tra được kết quả tương đối chính xác, các giá trị trên thước đo Accuracy và Loss đã phản ảnh được kết quả huấn luyện của mô hình. Tuy nhiên về kết quả thực nghiệm cho thấy mô hình còn gặp nhiều vấn đề, ví dụ như mặc dù kết quả Acc là khả quan nhưng đối với một mô hình dữ liệu đầu vào lớn thì việc sai số dù là nhỏ nhưng số lượng ảnh thực tế bị sai sẽ khiến cho mô hình trở nên không tốt. Cuồi cùng là đánh giá khả năng học của mô hìn, nhóm chúng tôi đã dành nhiều thời gian để cải thiển về vấn đề dữ liệu, tăng dữ liệu lên và giảm dữ liệu xuống và tăng số lần lặp cho mô hình, theo những kết quả cũ thì việc mô hình có nhiều dữ liệu hơn sẽ tăng khả năng sai sót của mô hình nhiều hơn, đối với mô hình có 8-15 class thì kết quả Accuracy của mô hình thật sự là chính xác nhiều hơn so với mô hình hiện tại là 29 class.

Xong, về hướng phát triển tiếp theo, chúng tôi hy vọng rằng có thể cải thiện vấn đề còn tồn đọng trong mô hình hiện tại, khắc phục vấn đề về tăng giảm dữ liệu đầu vào làm ảnh hưởng tới kết quả của mô hình, ngoài ra còn mong muốn có thể xử dụng một vài tính năng tích hợp mà nhóm chưa thể làm trong quá trình thực hiện dự án như là tích hợp module quét vân tay nhằm làm tăng tính thực tế cho mô hình.

**TÀI LIỆU THAM KHẢO**

# [1] Công ty TNHH Tự động hóa và Tin học công nghiệp Bách Khoa (BKAII), “[Sự khác biệt giữa AI, machine learning và deep learning](Sự%20khác%20biệt%20giữa%20AI,%20machine%20learning%20và%20deep%20learning)”, [https://bkaii.com.vn/tin-tuc/385-su-khac-biet-giua-ai-machine-learning-va-deep-learning - 12/2022](https://bkaii.com.vn/tin-tuc/385-su-khac-biet-giua-ai-machine-learning-va-deep-learning%20-%2012/2022)

# [2] TOPDev Blog, “Thuật toán CNN là gì? Cấu trúc mạng Convolutional Neural Network”, <https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/> - 12/2022

[3] TINOGroup, “Convolutional Neural Network là gì? – Cách chọn tham số cho Convolutional Neural Network chuẩn chỉnh”, <https://wiki.tino.org/convolutional-neural-network-la-gi/> - 12/2022

[4] Tan Nguyen (@tannm13), “Mạng nơ-ron tích chập (P1)”, <https://viblo.asia/p/mang-no-ron-tich-chap-p1-DZrGNNjPGVB> - 12/2022

[5] Chung Pham Van (@Pham\_Van\_Chung), “[Deep Learning] Tìm hiểu về mạng tích chập (CNN)”, <https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2> - 12/2022

[6] Phạm Duy Tùng (2021), “Tìm hiểu thuật toán tối ưu hoá Adabeliè Optimizer”, <https://www.phamduytung.com/blog/2021-01-15---adabelief-optimizer/> - 12/2022

[7] Got It VietNam (2021), “Softmax Function là gì? – Tổng quan về Softmax Function”, <https://vn.got-it.ai/blog/softmax-function-la-gi-tong-quan-ve-softmax-function> - 12/2022

[8] Quoc Pham (2019), “Tìm hiểu Convolutional Neural Network cho phân loại ảnh”,<https://pbcquoc.github.io/cnn/> - 12/2022