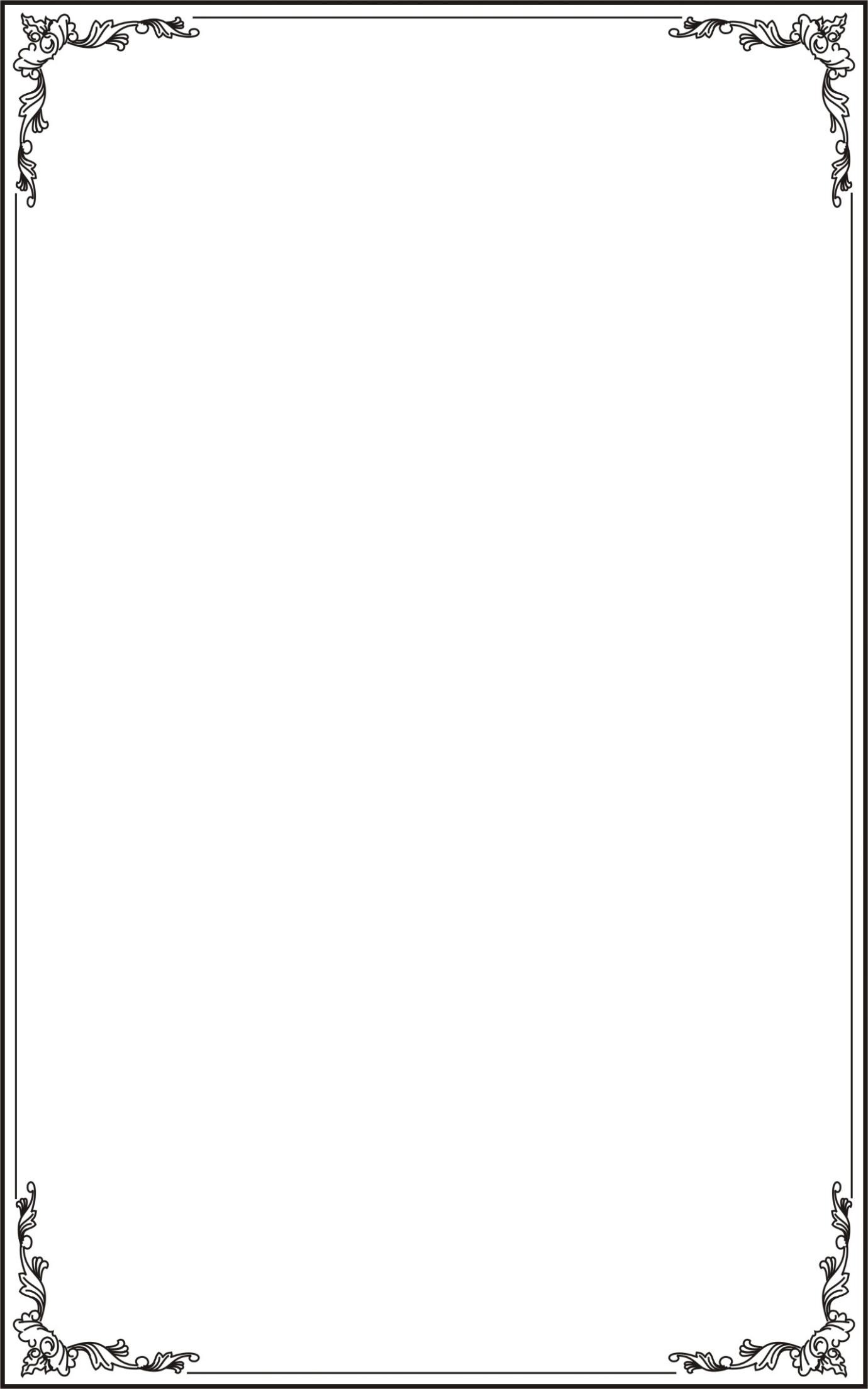
**ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO NHÓM MÔN HỌC TRÍ TUỆ NHÂN TẠO

NGÀNH: CÔNG NGHỆ THÔNG TIN

**ĐỀ TÀI: ỨNG DỤNG MACHINE LEARNING VÀ DEEP LEARNING TRONG NHẬN DIỆN KHUÔN MẶT**

|  |  |
| --- | --- |
| Cán bộ hướng dẫn: | Đoàn Thị Thanh Hằng |
| Sinh viên: |  |
| Lớp: | 72DCHT21 |

**Hà Nội, tháng 12 năm 2023**

Lời nói đầu

Trong thời kỳ chuyển đổi số ngày nay, việc nghiên cứu và phát triển các phương pháp nhận dạng khuôn mặt đã trở thành một lĩnh vực nghiên cứu quan trọng, đặc biệt là trong lĩnh vực an ninh, quản lý thông tin, và trải nghiệm người dùng. Nhận dạng khuôn mặt không chỉ mang lại tiện ích trong việc xác thực cá nhân mà còn đóng vai trò quan trọng trong nhiều ứng dụng như giám sát, điều khiển truy cập, và tìm kiếm thông tin.

Trong bối cảnh này, bài báo cáo này nhằm đề cập và phân tích các phương pháp nhận dạng khuôn mặt hiện đại, với sự tập trung đặc biệt vào các kỹ thuật và mô hình dựa trên trí tuệ nhân tạo. Trong số đó, chúng ta sẽ tập trung vào việc sử dụng Convolutional Neural Networks (CNN), một loại mô hình đặc biệt mạnh mẽ trong việc hiểu và rút trích đặc trưng từ dữ liệu hình ảnh.

Qua quá trình này, chúng em mong muốn đem lại cái nhìn tổng quan về tiến triển và ứng dụng của công nghệ nhận dạng khuôn mặt, cũng như thảo luận về những thách thức và triển vọng trong lĩnh vực này. Bằng cách này, chúng em hy vọng góp phần vào sự hiểu biết và phát triển tiếp theo của cộng đồng nghiên cứu và người làm chính sách liên quan đến an ninh và quản lý thông tin.

**MỤC LỤC**

Trang

**LỜI NÓI ĐẦU**1

**NỘI DUNG**3

**CHƯƠNG I. Tổng quan về Project3**

1. Sơ lược về nhận diện khuôn mặt và trí tuệ nhân tạo3

2. Ứng dụng của nhận diện khuôn mặt trong thực tế 3

**CHƯƠNG II. Xây dựng mô hình 11**

1. Phương pháp giải quyết……………………………………………………….
   1. Ý tưởng thuật toán………………………………………………………
   2. Lựa chọn mô hình…………………………………………………….
   3. Lược đồ mô hình…………………………………………………….
2. Tiền xử lý dữ liệu…………………………………………………………….
   1. . Giới thiệu về PyCharm và thư viện OpenCV…………………………
   2. . Chuẩn bị bộ dữ liệu…………………………………………………..
3. Các thành phần của mô hình………………………………………………….

3.1. Lớp tích chập (Conv2D)………………………………………………..

3.2. Hàm kích hoạt RELU…………………………………………………..

3.3. Lớp Pooling (MaxPooling2D)………………………………………….

3.4. Lớp làm phẳng (Flatten) và Dropout…………………………………..

3.5. Lớp liên kết đầy đủ (Dense)……………………………………………

3.6. Hàm kích hoạt Softmax…………………………………………………

3.7. Thuật toán Adam………………………………………………………

3.8. Hàm mất mát Categorical-Crossentropy………………………………

3.9. Kỹ thuật One-Hot-Encoding…………………………………………..

3.10. Mô hình tổng quan……………………………………………

**CHƯƠNG III. Cài đặt…………………………………………………………**

1. Giới thiệu về Tensorflow và Keras……………………………………..
   1. Tensorflow………………………………………………………….
   2. Keras………………………………………………………………..
2. Cài đặt mô hình…………………………………………………………..

**CHƯƠNG IV. Tổng kết và hướng phát triển……………………………………**

1. Tổng kết…………………………………………………………………….
2. Hướng phát triển…………………………………………………………..

**TÀI LIỆU THAM KHẢO**

**NỘI DUNG**

**CHƯƠNG I. TỔNG QUAN VỀ PROJECT**

1. **Sơ lược về nhận diện khuôn mặt trong trí tuệ nhân tạo**

Công nghệ AI nhận diện khuôn mặt là công nghệ ứng dụng trí tuệ nhân tạo vào việc xác định danh tính thông qua việc phân tích đặc điểm khuôn mặt (có thể là tròng mắt, hình dạng, kích thước, khoảng cách và đặc tính giữa các bộ phận). Ngày nay, với tốc độ phát triển mạnh mẽ của khoa học kỹ thuật, hệ thống AI nhận diện có thể dựa vào các điểm nổi bật trên khuôn mặt như những mô cứng và xương. Từ đó nhìn thấy rõ nhất đường cong của hốc mắt, mũi và cằm để nhận ra đối tượng. Các đặc điểm này với mỗi khuôn mặt là độc nhất và không thay đổi theo thời gian. Hệ thống có thể hoạt động trong nhiều điều kiện thời tiết khác nhau, không ngoại trừ ban ngày hay ban đêm, khi môi trường không đủ điều kiện ánh sáng hoặc trong bóng tối. Công nghệ AI nhận diện khuôn mặt với những tính năng vượt trội đã được áp dụng thành công trong nhiều lĩnh vực, giúp người quản lý dễ dàng kiểm soát mà ít làm ảnh hưởng đến đối tượng bị kiểm soát, tránh họ cảm thấy sự riêng tư bị xâm phạm.

1. **Ứng dụng của nhận diện khuôn mặt trong thực tế**

Nhận diện khuôn mặt là một lĩnh vực quan trọng trong công nghệ thông tin và có nhiều ứng dụng thực tế đa dạng. Dưới đây là một số ví dụ về ứng dụng của nhận diện khuôn mặt:

* + 1. Đăng nhập và xác thực: Nhận diện khuôn mặt có thể được sử dụng để đăng nhập và xác thực người dùng trong các hệ thống và thiết bị, chẳng hạn như điện thoại di động, máy tính, cổng vào văn phòng, máy tính bảng, hệ thống an ninh và các ứng dụng di động khác. Thay vì sử dụng mật khẩu hoặc mã PIN, người dùng có thể sử dụng khuôn mặt của họ để xác thực mình.
    2. Quản lý danh sách đen: Nhận diện khuôn mặt có thể được sử dụng để quản lý danh sách đen trong các hệ thống an ninh hoặc kiểm soát truy cập. Hệ thống có thể so sánh khuôn mặt của một người với danh sách đen đã được xác định trước đó để xác định xem người đó có liên quan đến hoạt động không hợp pháp hoặc nguy hiểm hay không.
    3. Theo dõi và nhận dạng khách hàng: Các doanh nghiệp trong lĩnh vực bán lẻ hoặc dịch vụ khách hàng có thể sử dụng nhận diện khuôn mặt để theo dõi và nhận dạng khách hàng. Điều này giúp tạo ra trải nghiệm cá nhân hóa và cải thiện dịch vụ, như gợi ý sản phẩm phù hợp, giới thiệu dịch vụ, quản lý khách hàng trung thành và phục vụ khách hàng tốt hơn.
    4. Giám sát an ninh và giám sát công cộng: Nhận diện khuôn mặt có thể được sử dụng trong các hệ thống giám sát an ninh và giám sát công cộng để phát hiện và nhận dạng các đối tượng nghi ngờ hoặc nguy hiểm. Hệ thống có thể so sánh khuôn mặt của những người xuất hiện trong hình ảnh hoặc video với cơ sở dữ liệu để xác định xem có sự tương đồng hay không.
    5. Quản lý sự kiện và đám đông: Nhận diện khuôn mặt có thể được sử dụng để quản lý sự kiện và đám đông trong các sự kiện lớn.
    6. Giúp trong lĩnh vực an ninh và pháp luật: Nhận diện khuôn mặt có thể hỗ trợ trong việc giám sát và điều tra tội phạm. Hệ thống có thể so sánh khuôn mặt của các đối tượng bị nghi ngờ với cơ sở dữ liệu tội phạm hoặc danh sách tìm kiếm để xác định danh tính của chúng. Điều này có thể giúp trong việc giảm tội phạm và cải thiện an ninh công cộng.
    7. Quản lý nhận dạng trong sân bay và cửa khẩu: Nhận diện khuôn mặt có thể được sử dụng để xác định danh tính của hành khách trong các sân bay và cửa khẩu. Hệ thống có thể so sánh khuôn mặt của hành khách với dữ liệu biểu diễn danh sách hành khách an toàn và cung cấp thông tin cho các cơ quan an ninh để kiểm tra và kiểm soát.
    8. Hỗ trợ trong lĩnh vực y tế: Nhận diện khuôn mặt có thể được sử dụng để xác định và theo dõi bệnh nhân trong các cơ sở y tế. Hệ thống có thể nhận dạng bệnh nhân và liên kết thông tin y tế của họ để đảm bảo sự chính xác trong việc cung cấp dịch vụ y tế và giúp nâng cao chất lượng chăm sóc y tế.
    9. Điều khiển truy cập và an ninh trong tòa nhà và cơ quan: Nhận diện khuôn mặt có thể được sử dụng để điều khiển truy cập vào tòa nhà và cơ quan. Thay vì sử dụng thẻ hoặc mã PIN, nhận diện khuôn mặt có thể cho phép người dùng truy cập vào các khu vực an ninh chỉ bằng cách nhận dạng khuôn mặt của họ.
    10. Ứng dụng trong công nghệ xe tự hành: Nhận diện khuôn mặt có thể được sử dụng trong công nghệ xe tự hành để xác định và nhận dạng người lái. Điều này có thể giúp đảm bảo an toàn và bảo mật trong việc sử dụng và quản lý các dịch vụ liên quan đến xe tự lái.

Tổng hợp lại, nhận diện khuôn mặt có nhiều ứng dụng thực tế trong các lĩnh vực như an ninh, công nghệ, y tế, giao thông v.v…

**CHƯƠNG II: XÂY DỰNG MÔ HÌNH**

## 1. Phương pháp giải quyết

**1.1. Ý tưởng thuật toán**

Để xây dựng một hệ thống trí tuệ nhân tạo nhận diện khuôn mặt chi tiết, bạn có thể áp dụng các bước sau:

1. Xây dựng bộ dữ liệu chi tiết: Thu thập một bộ dữ liệu chứa các hình ảnh khuôn mặt chi tiết, chẳng hạn như mắt, mũi, miệng, và các đặc điểm nhỏ khác trên khuôn mặt. Bạn có thể tạo bộ dữ liệu này bằng cách gắn nhãn (đánh dấu) các điểm chi tiết trên khuôn mặt trong một tập hợp ảnh.
2. Xây dựng mô hình mạng nơ-ron: Sử dụng một kiến trúc mạng nơ-ron, như mạng tích chập (CNN), để học và trích xuất các đặc trưng từ khuôn mặt chi tiết. Bạn có thể sử dụng các lớp tích chập để tìm kiếm các đặc trưng cục bộ và các lớp kích

hoạt phi tuyến như ReLU để giới hạn đầu ra. Các lớp gộp cũng có thể được sử dụng để giảm kích thước dữ liệu và trích xuất các đặc trưng quan trọng.

1. Huấn luyện mạng nơ-ron: Mạng nơ-ron cần được huấn luyện bằng cách cung cấp các hình ảnh khuôn mặt chi tiết đã được gắn nhãn (ví dụ: các điểm chi tiết). Tối ưu hóa hàm mất mát (loss function) để tối đa hóa độ chính xác của việc nhận diện các điểm chi tiết trên khuôn mặt. Các thuật toán tối ưu như gradient descent có thể được sử dụng để điều chỉnh trọng số của mạng.
2. Đánh giá và tinh chỉnh: Đánh giá hiệu suất của mô hình bằng cách sử dụng tập dữ liệu kiểm tra hoặc tập dữ liệu không nhìn thấy trước đó. Các độ đo như độ chính xác (accuracy) và độ chính xác tương đối (precision-recall) có thể được sử dụng để đánh giá hiệu suất của mô hình. Nếu cần, bạn có thể tinh chỉnh mô hình bằng cách thay đổi kiến trúc mạng, siêu tham số (hyperparameters) hoặc kỹ thuật huấn luyện.
3. Sử dụng và triển khai: Sau khi đạt được hiệu suất đáng chấp nhận trên tập dữ liệu kiểm tra, bạn có thể sử dụng mô hình nhận diện khuôn mặt chi tiết để áp dụng cho các ứng dụng thực tế. Các ứng dụng có thể bao gồm nhận diện biểu hiện khuôn mặt, nhận diện dấu vết, nhận diện đặc điểm khuôn mặt để phục vụ cho việc xác thực hoặc giám sát an ninh.
4. Tăng cường dữ liệu (Data augmentation): Để cải thiện khả năng tổng quát hóa và độ chính xác của mô hình, bạn có thể sử dụng kỹ thuật tăng cường dữ liệu. Điều này bao gồm tạo ra các phiên bản biến thể của dữ liệu huấn luyện bằng cách áp dụng các phép biến đổi như xoay, phóng to, thu nhỏ, lật ngang, lật dọc, ánh sáng và sắc tố. Điều này giúp mô hình học được khái quát hơn và giảm thiểu hiện tượng overfitting.
5. Học chuyển tiếp (Transfer learning): Nếu bạn đang làm việc với tập dữ liệu nhỏ, bạn có thể sử dụng kỹ thuật học chuyển tiếp để tận dụng kiến thức đã được học từ mạng nơ-ron đã được huấn luyện trên một tác vụ tương tự. Bằng cách sử dụng các

trọng số và đặc trưng đã được học từ mạng lớn, bạn có thể giảm thời gian và nguồn lực huấn luyện mô hình của mình.

1. Đánh giá và cải thiện: Để đảm bảo hiệu suất tốt của mô hình, bạn nên thường xuyên đánh giá và cải thiện nó. Có thể thử nghiệm các kiến trúc mạng khác nhau, tinh chỉnh siêu tham số và kỹ thuật huấn luyện để tìm ra cấu trúc và cài đặt tốt nhất cho bài toán nhận diện khuôn mặt chi tiết.

Tóm lại, xây dựng một hệ thống trí tuệ nhân tạo nhận diện khuôn mặt chi tiết đòi hỏi việc xây dựng một kiến trúc mạng nơ-ron phù hợp, huấn luyện trên bộ dữ liệu phù hợp, đánh giá và cải thiện hiệu suất của hệ thống.

**1.2. Lựa chọn mô hình**

Có nhiều mô hình học máy được sử dụng để nhận diện khuôn mặt, trong đó CNN (Convolutional Neural Network) là một trong những mô hình quan trọng và phổ biến nhất. Dưới đây là một số mô hình học máy nhận diện khuôn mặt phổ biến, trong đó tập trung vào CNN:

1. LeNet-5: Đây là một trong những mô hình CNN đầu tiên được giới thiệu bởi Yann LeCun. Ban đầu, LeNet-5 được thiết kế để nhận dạng chữ số viết tay, nhưng sau đó cũng được áp dụng vào nhận diện khuôn mặt. Mô hình này sử dụng các lớp convolutional và pooling để trích xuất đặc trưng từ hình ảnh.
2. VGGFace: Mô hình VGGFace được xây dựng trên cơ sở của mạng VGG (Visual Geometry Group). VGGFace có khả năng nhận diện khuôn mặt với độ chính xác cao và đạt được thành công lớn trong cuộc thi nhận diện khuôn mặt ILSVRC 2014. Mô hình này sử dụng một số lớp convolutional và fully connected để học các đặc trưng từ hình ảnh khuôn mặt.
3. FaceNet: FaceNet là một mô hình nhận diện khuôn mặt sử dụng CNN và học biểu diễn khuôn mặt trong không gian vectơ. Nó áp dụng phép toán triplet loss để tạo ra các biểu diễn khuôn mặt gần nhau cho cùng một người và xa nhau cho

những người khác nhau. FaceNet đã đạt được kết quả rất tốt trong các bài toán nhận diện khuôn mặt và xếp hạng.

1. DeepFace: DeepFace là một mô hình nhận diện khuôn mặt phát triển bởi Facebook AI Research. Nó sử dụng một mạng CNN sâu để học biểu diễn khuôn mặt và thực hiện các bước nhận diện khuôn mặt như phát hiện khuôn mặt, phân loại và xác minh. Mô hình này đạt được hiệu suất rất cao trong các bài toán nhận diện khuôn mặt trên tập dữ liệu lớn.

Trong tất cả các mô hình trên, CNN (Convolutional Neural Network) là thành phần chính được sử dụng để học các đặc trưng của khuôn mặt từ dữ liệu ảnh. CNN giúp mô hình có khả năng tự động tổ chức các đặc trưng không gian của khuôn mặt và học các đặc trưng phức tạp thông qua các lớp convolutional. CNN cũng cho phép mô hình tự động học các mức độ độ sâu của đặc trưng, từ các đặc trưng cơ bản như cạnh và góc đến các đặc trưng phức tạp hơn như cấu trúc khuôn mặt và biểu cảm.

Việc sử dụng CNN trong nhận diện khuôn mặt cũng giúp giảm sự phụ thuộc vào việc xác định và trình bày các đặc trưng cụ thể của khuôn mặt. Thay vào đó, CNN có khả năng tự động học các đặc trưng như các đường viền mắt, cung mày, môi, và các đặc trưng khác. Điều này giúp mô hình tổng quát hóa tốt hơn và có khả năng nhận diện khuôn mặt của các người khác nhau và trong các điều kiện khác nhau.

**Ở** **đây** **xin** **được** **phép** **lựa** **chọn** **mô** **hình** **CNN** **của** **TensorFlow** **để** **tạo** **AI** **nhận** **diện** **khuôn** **mặt.**

***Lí do***

CNN (Convolutional Neural Network) là một mô hình neural network đặc biệt được sử dụng rộng rãi trong các ứng dụng liên quan đến xử lý ảnh, bao gồm cả nhận diện khuôn mặt. Có một số lý do chính mà CNN được sử dụng làm AI nhận diện khuôn mặt:

1. Cấu trúc tổ chức: CNN có cấu trúc tổ chức đặc biệt, với các lớp convolutional và pooling, được thiết kế để hiểu và trích xuất thông tin từ các đặc trưng không gian

trong ảnh. Điều này phù hợp với việc nhận diện khuôn mặt, vì khuôn mặt cũng có các đặc trưng không gian như cấu trúc mắt, mũi, miệng, và khuôn khổ tổng thể.

1. Khả năng học đặc trưng tự động: CNN có khả năng tự động học các đặc trưng từ dữ liệu huấn luyện. Thay vì phải xác định và trình bày các đặc trưng cụ thể của khuôn mặt, CNN có thể tự động học các đặc trưng như viền mắt, đường cung mày, hoặc môi. Điều này giúp giảm sự phụ thuộc vào các đặc trưng được định nghĩa bởi con người và giúp mô hình tổng quát hóa tốt hơn.
2. Xử lý đa phân tầng: CNN cho phép xử lý ảnh theo nhiều phân tầng thông qua các lớp convolutional và pooling. Điều này giúp mô hình nhận diện khuôn mặt có khả năng nhìn xuyên qua các mức độ độ sâu của đặc trưng, từ các đặc trưng cơ bản như viền và góc đến các đặc trưng phức tạp hơn như cấu trúc khuôn mặt và biểu cảm.
3. Độ chính xác cao: CNN đã được chứng minh là rất hiệu quả trong việc nhận diện khuôn mặt với độ chính xác cao. Điều này là do khả năng học tự động các đặc trưng phức tạp và khả năng xử lý các biến đổi và biến đổi trong khuôn mặt, bao gồm sự thay đổi về ánh sáng, góc chụp và biểu cảm.

Tóm lại, CNN được sử dụng làm AI nhận diện khuôn mặt vì có cấu trúc tổ chức phù hợp với đặc trưng không gian của khuôn mặt và khả năng học đặc trưng tự động từ dữ liệu huấn luyện. CNN cung cấp khả năng xử lý đa phân tầng, giúp nhìn xuyên qua các mức độ độ sâu của đặc trưng trong khuôn mặt. Nó cũng cho phép xử lý các biến đổi và biến đổi trong khuôn mặt, đồng thời đạt được độ chính xác cao trong việc nhận diện khuôn mặt. Điều này làm cho CNN trở thành lựa chọn phổ biến và hiệu quả trong các ứng dụng thực tế của nhận diện khuôn mặt.

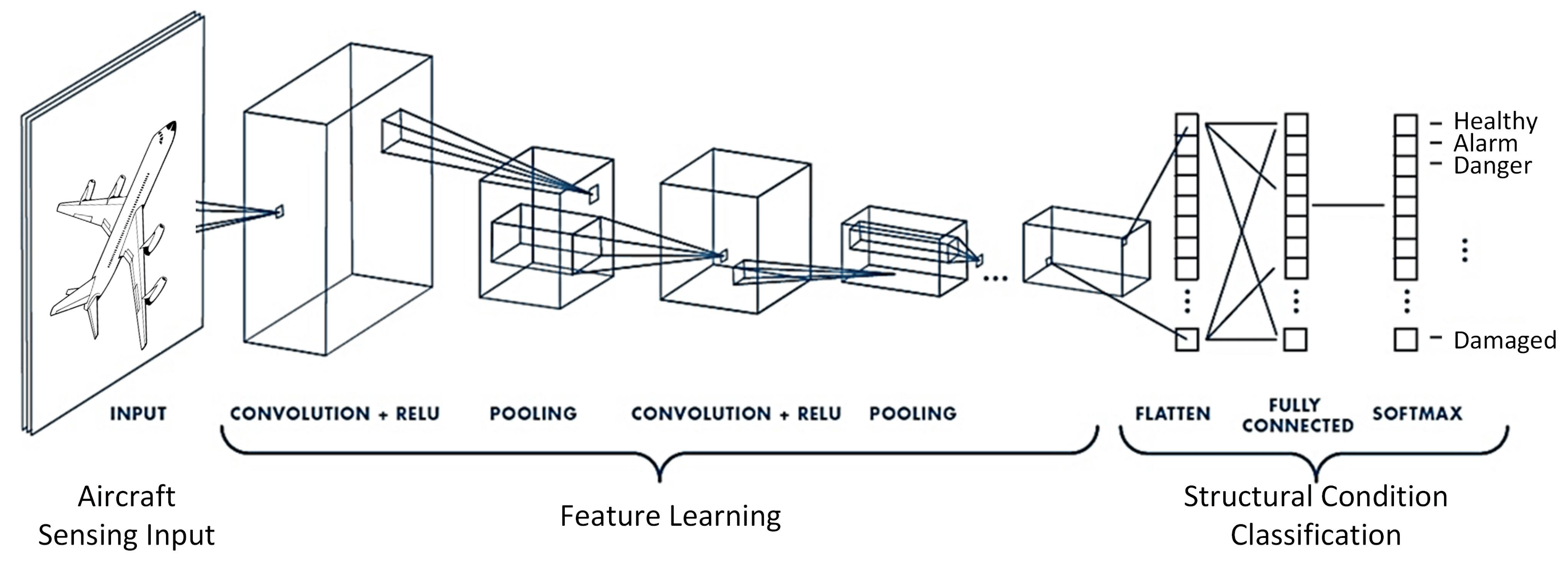
**Có** **một** **số** **ý** **tưởng** **và** **cải** **tiến** **liên** **quan** **đến** **CNN:**

* + **Đa** **tầng** **tích** **chập:** **Thay** **vì** **sử** **dụng** **chỉ** **một** **lớp** **tích** **chập,** **có** **thể** **sử** **dụng** **nhiều** **lớp** **tích** **chập** **liên** **tiếp** **để** **học** **các** **đặc** **trưng** **phức** **tạp** **hơn.** **Việc** **sử** **dụng** **nhiều** **tầng** **tích** **chập** **có** **thể** **giúp** **mô** **hình** **học** **được** **các** **đặc** **trưng** **từ** **các** **mức** **độ** **trừu** **tượng** **cao** **hơn.**
  + Mạng nơ-ron chồng chéo (Inception network): Mạng Inception sử dụng các module chồng chéo có kích thước khác nhau để học các đặc trưng ở các tỉ lệ không gian khác nhau. Điều này giúp mô hình có khả năng phát hiện đặc trưng ở nhiều mức độ chi tiết và tỉ lệ khác nhau trong ảnh.
  + Học tăng cường (Reinforcement learning): Sử dụng học tăng cường để điều chỉnh cấu trúc của CNN có thể giúp tối ưu hóa hiệu suất. Bằng cách sử dụng phần thưởng và phạt trong quá trình huấn luyện, mô hình có thể tự động điều chỉnh số lượng, vị trí và kích thước của các lớp tích chập trong mạng.
  + Mạng nơ-ron tái cấu trúc (Neural architecture search): Ý tưởng này liên quan đến việc sử dụng thuật toán tìm kiếm tự động để tìm kiếm kiến trúc CNN tối ưu cho một tác vụ nhất định. Thay vì thiết kế kiến trúc mạng thủ công, thuật toán sẽ tìm kiếm và xác định các kiến trúc tốt nhất để đạt được hiệu suất cao nhất.
  + Mạng nơ-ron đa nhiệm (Multi-task learning): CNN có thể được sử dụng để giải quyết nhiều tác vụ cùng một lúc bằng cách chia sẻ các lớp tích chập chung. Điều này cho phép mô hình học được các đặc trưng chung từ các tác vụ khác nhau và cải thiện hiệu suất trên các tác vụ đa dạng.

**Với** **bài** **toán** **nhận** **diện** **khuôn** **mặt** **chúng** **ta** **sẽ** **sử** **dụng** **Mô** **hình** **CNN** **(Đa** **tầng** **tích** **chập)** **để** **tối** **ưu** **hóa** **việc** **huấn** **luyện** **cũng** **như** **phân** **loại** **ảnh** **một** **cách** **dễ** **dàng** **và** **nhanh** **chóng.**

**1.3. Lược đồ mô hình**

Mô hình lọc ảnh qua các lớp đặc biệt để trích xuất những đặc trưng chung nhất của khuôn mặt để lưu vào hệ thống rồi tiến hành liên kết với các nhãn.

******

*Hình 1: Mô* *hình* *tổng* *quan* *CNN* *trong* *TensorFlow*

## 2. Tiền xử lý dữ liệu

**2.1. Giới thiệu về Pycharm và thư viện OpenCV**

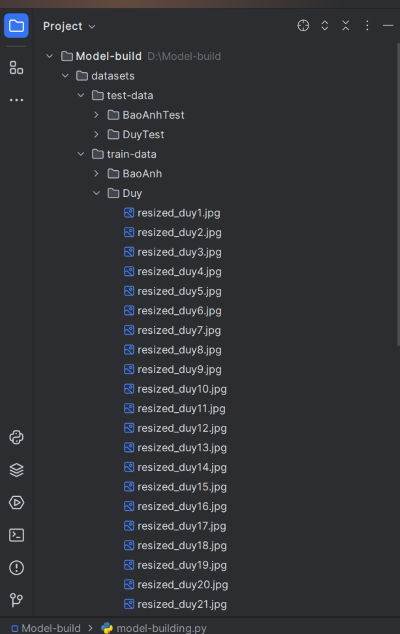
PyCharm là một môi trường phát triển tích hợp (IDE) được sử dụng chủ yếu để lập trình Python. Nó được phát triển bởi JetBrains và cung cấp nhiều tính năng hữu ích giúp lập trình viên nâng cao hiệu suất làm việc. PyCharm cung cấp trình biên dịch Python, hỗ trợ gỡ lỗi, kiểm tra cú pháp và nhiều tính năng khác giúp bạn phát triển và quản lý dự án Python dễ dàng hơn.

Thư viện OpenCV là một thư viện mã nguồn mở cho Python, được sử dụng để nhận diện và nhận dạng khuôn mặt trong ảnh và video. Nó sử dụng công nghệ deep learning để trích xuất các đặc trưng từ khuôn mặt và so sánh chúng với cơ sở dữ liệu khuôn mặt đã được huấn luyện trước. OpenCV hỗ trợ nhiều chức năng như tìm kiếm khuôn mặt trong ảnh, nhận dạng khuôn mặt từ video và cung cấp các thông tin như vị trí khuôn mặt, điểm đặc trưng và khuôn mặt nổi bật.

Khi kết hợp PyCharm với thư viện OpenCV, bạn có thể phát triển ứng dụng nhận diện khuôn mặt trong Python một cách thuận tiện. PyCharm giúp bạn viết mã một cách dễ dàng và tổ chức dự án, trong khi OpenCV cung cấp các công cụ và chức năng để xử lý và phân tích khuôn mặt.

**2.2. Chuẩn bị bộ dữ liệu**

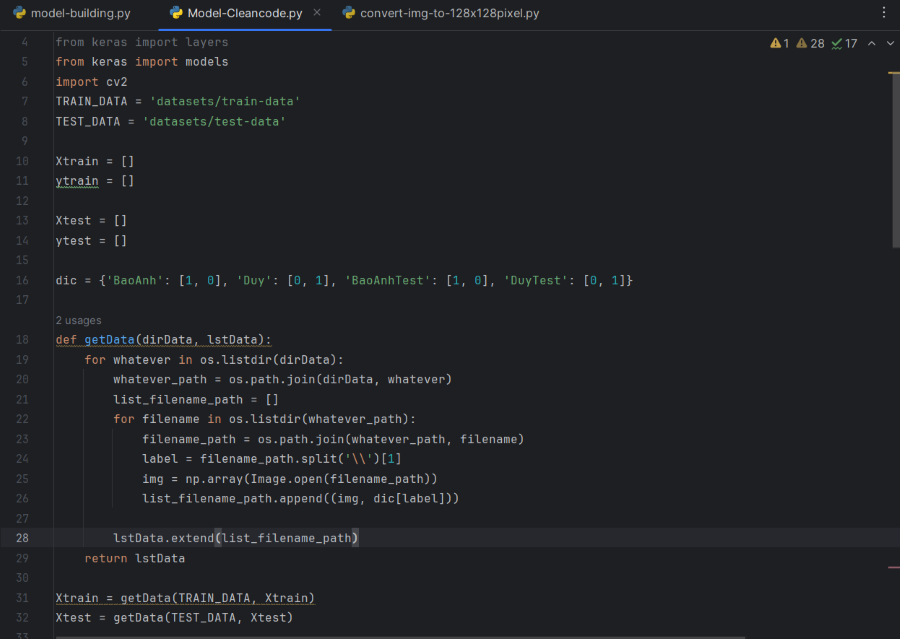
Dữ liệu thô gồm 100 tấm ảnh cá nhân của Duy và Bảo Anh đều có kích cỡ chung là 128px\*128px



*Hình 2: Bộ dữ liệu huấn luyện*

Thư mục train-data bao gồm các hình ảnh được sử dụng cho quá trình đưa vào mô hình để huấn luyện. Thư mục test-data là bộ hình ảnh có nhãn (label) thực tế để mô hình sau khi training sẽ so sánh, huấn luyện và chuẩn đoán để hoàn thiện độ chính xác của mô hình.

Tiếp đến là xử lí dữ liệu thư mục để đổ vào Xtrain, ytrain , Xtest và ytest:



*Hình 3: Đổ bộ dữ liệu vào Xtrain, ytrain, Xtest, ytest*

Dữ liệu sau khi được lấy từ thư mục datasets sẽ được xử lí để lấy được hình ảnh và nhãn của hình ảnh sau đó đổ vào Xtrain, ytrain , Xtest và ytest. Các dữ liệu Xtrain, ytrain , Xtest và ytest sẽ được phục vụ cho việc huấn luyện mô hình và dự đoán.

## 3. Các thành phần của mô hình

Để hiểu được về mô hình ta cần tìm hiểu sơ qua về các lớp, chỉ định hàm mất mát, thuật toán và hàm kích hoạt được sử dụng

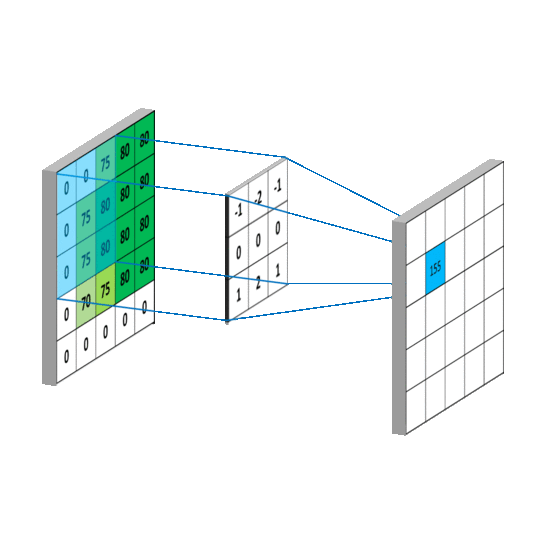
**3.1. Lớp tích chập (Conv2D)**

Lớp tích chập (Convolutional Layer) là một loại lớp quan trọng trong mạng nơ-ron sử dụng cho xử lý hình ảnh và các dạng dữ liệu liên quan đến ma trận. Lớp này thực hiện việc áp dụng một bộ lọc (filter/kernel) trên toàn bộ ảnh đầu vào, với mục đích tìm kiếm các đặc trưng ở mức độ cao hơn.

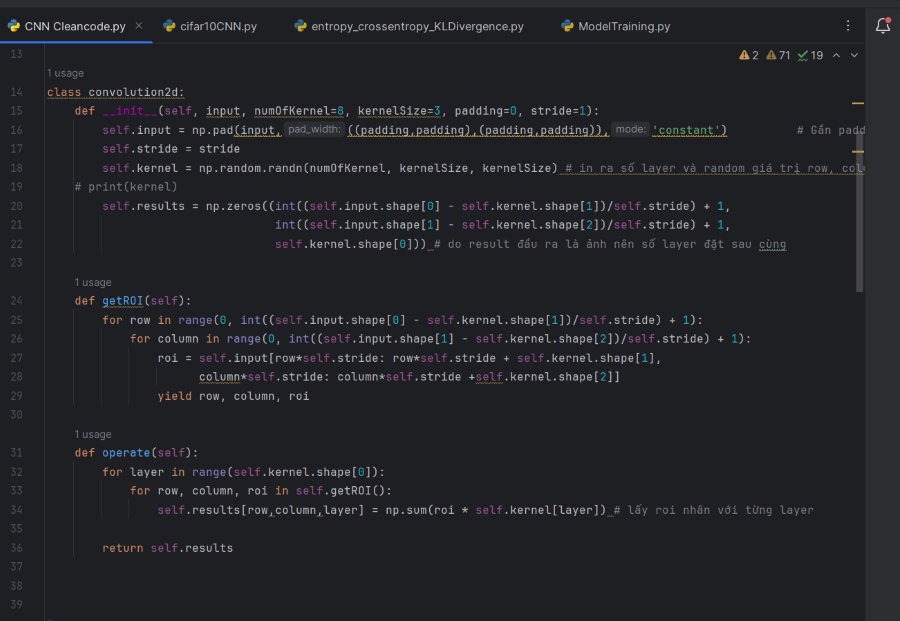
Các tham số quan trọng của lớp tích chập bao gồm:

1. Số lượng filter: Đây là số lượng filter sẽ được sử dụng để áp dụng trên đầu vào. Mỗi filter sẽ tạo ra một feature map (bản đồ đặc trưng), giúp phát hiện các đặc trưng khác nhau trong đầu vào.
2. Kích thước của filter: Đây là kích thước của bộ lọc được áp dụng trên ảnh đầu vào. Kích thước này thường được chọn là số lẻ, ví dụ như 3x3, 5x5, 7x7, v.v.
3. Bước nhảy (stride): Đây là khoảng cách giữa các vị trí áp dụng filter trên ảnh đầu vào. Nếu stride bằng 1, filter sẽ được áp dụng liên tiếp trên từng pixel của ảnh đầu vào. Nếu stride lớn hơn 1, filter sẽ được áp dụng cách đều nhau trên ảnh đầu vào, giảm kích thước của output feature map.
4. Padding: Đây là cách thêm viền xung quanh ảnh đầu vào để tránh mất mát thông tin khi áp dụng filter. Padding có thể được thêm vào ảnh đầu vào bằng cách thêm các giá trị 0 vào các cạnh hoặc trung tâm của ảnh.

Các tham số này sẽ ảnh hưởng đến kích thước và số lượng feature map của output, ảnh hưởng đến quá trình huấn luyện mạng nơ-ron và hiệu suất của mạng. Do đó, việc tinh chỉnh các tham số này là rất quan trọng để đạt được kết quả tốt nhất từ mạng nơ-ron sử dụng lớp tích chập.



*Hình 4: Mô hình convolutional 2D*



*Hình 5: Hàm convolutional 2D được biểu diễn với Python*

**3.2. Hàm kích hoạt RELU ( The Activation Functions RELU)**

Hàm kích hoạt ReLU (Rectified Linear Unit) là một trong những hàm kích hoạt phổ biến trong mạng nơ-ron sử dụng cho bài toán phân loại ảnh. Hàm ReLU được xác định bởi công thức :

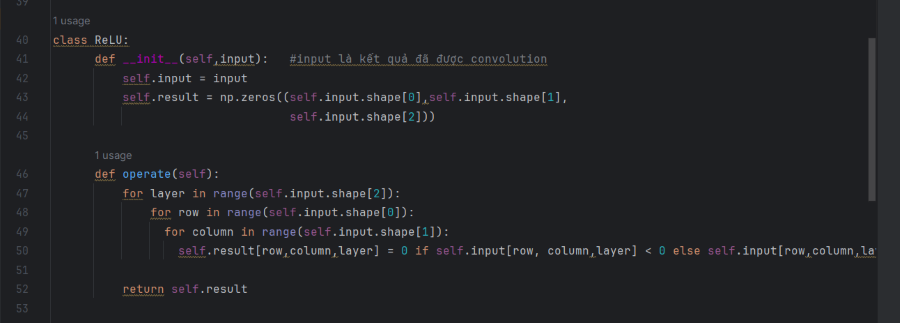
f(x) = max(0, x)

, trong đó x là đầu vào và f(x) là đầu ra.

Vai trò của hàm ReLU trong phân loại ảnh là giúp mạng nơ-ron học được các đặc trưng của ảnh một cách hiệu quả hơn. Khi áp dụng hàm ReLU, các giá trị âm sẽ bị đưa về 0, trong khi các giá trị dương sẽ được giữ nguyên. Điều này giúp mạng nơ- ron tìm ra những đặc trưng quan trọng của ảnh như sự tương phản, đường viền, v.v...

Hơn nữa, hàm ReLU cũng giúp tránh được hiện tượng mất mát gradient (vanishing gradient) trong quá trình huấn luyện mạng nơ-ron. Nếu sử dụng các hàm kích hoạt khác như sigmoid hay tanh, gradient sẽ tiến dần về 0 khi đầu vào xa khỏi trung tâm, làm cho việc cập nhật trọng số trở nên khó khăn. Tuy nhiên, với hàm ReLU, gradient vẫn được giữ nguyên khi đầu vào lớn hơn 0, giúp cải thiện quá trình huấn luyện mạng nơ-ron.

Tóm lại, hàm kích hoạt ReLU giúp mạng nơ-ron học được các đặc trưng quan trọng của ảnh và tránh được hiện tượng mất mát gradient, làm cho quá trình huấn luyện mạng nơ-ron hiệu quả hơn.

****

*Hình 6: Hàm RELU được biểu diễn với Python*

**3.3. Lớp pooling (MaxPooling2D)**

Hàm MaxPooling là một lớp rút trích đặc trưng trong mạng nơ-ron sử dụng cho xử lý hình ảnh. Lớp này thực hiện việc giảm kích thước của feature map bằng cách chọn giá trị lớn nhất từ mỗi vùng không chồng lên nhau của feature map.

Các tham số quan trọng của lớp MaxPooling bao gồm:

1. Kích thước của pooling window: Kích thước của pooling window quy định kích thước của vùng không chồng lên nhau trên feature map mà ta sẽ chọn giá trị lớn nhất để đưa vào output feature map. Kích thước này thường là số lẻ và thường được chọn là 2 hoặc 3.

2.Bước nhảy (stride): Tương tự như lớp tích chập, stride quy định khoảng cách giữa các vị trí áp dụng pooling window trên feature map. Nếu stride bằng kích thước của pooling window, các vùng không chồng lên nhau sẽ được chọn để tính

giá trị lớn nhất. Nếu stride lớn hơn kích thước của pooling window, các vùng không được chọn sẽ bị bỏ qua.

Lớp MaxPooling giúp giảm kích thước của feature map và trích xuất các đặc trưng quan trọng nhất, đồng thời giảm số lượng tham số và tính toán trong mạng nơ-ron. Việc sử dụng lớp

MaxPooling cũng giúp giảm overfitting trong quá trình huấn luyện mạng nơ-ron.



*Hình 7: Hàm Max-Pooling*



*Hình 8: Hàm Max-Pooling được biểu diễn với Python*

**3.4. Lớp làm phẳng (Flatten) và Dropout.**

Lớp Flatten và Dropout là các lớp quan trọng trong các mô hình học sâu được sử dụng để giảm overfitting và cải thiện hiệu suất mô hình.

**Lớp** **Flatten** là một lớp mạng nơ-ron đơn giản, có chức năng chuyển đổi dữ liệu đầu vào từ định dạng ma trận thành định dạng vector. Lớp này được sử dụng để trải phẳng dữ liệu đầu vào trước khi đưa vào các lớp mạng nơ-ron tiếp theo. Ví dụ, nếu dữ liệu đầu vào có kích thước là (batch\_size, 28, 28), thì lớp Flatten sẽ chuyển đổi nó thành một vector có kích thước (batch\_size, 784).

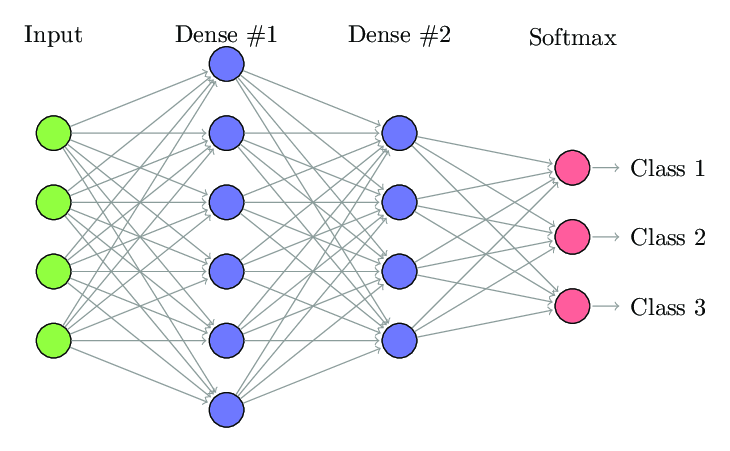
**Lớp** **Dropout** là một kỹ thuật regularization được sử dụng để ngăn chặn overfitting trong các mô hình học sâu. Khi sử dụng lớp Dropout, một phần tỷ lệ các nơ-ron trong mạng nơ-ron sẽ được bỏ qua ngẫu nhiên và không tham gia vào quá trình

huấn luyện. Điều này giúp mô hình học được một cách tổng quát hơn và giảm overfitting.

**3.5. Lớp liên kết đầy đủ (Dense).**

**Lớp** **Dense** trong TensorFlow là một lớp mạng nơ-ron đầy đủ kết nối (fully connected layer), nghĩa là tất cả các đầu vào của lớp trước sẽ được kết nối với tất cả các đầu ra của lớp này.

Lớp Dense chủ yếu được sử dụng trong các mô hình học sâu để ánh xạ đầu vào sang đầu ra bằng cách sử dụng các trọng số và độ lệch (biases). Để sử dụng lớp Dense trong TensorFlow, bạn có thể khởi tạo một thể hiện của lớp này với số lượng đầu ra mong muốn và sau đó truyền đầu vào vào thể hiện của lớp này để tính toán đầu ra.



*Hình 9: Fully Connected Layer*

**3.6. Hàm kích hoạt Softmax.**

Hàm kích hoạt softmax là một hàm kích hoạt phổ biến trong lĩnh vực trí tuệ nhân tạo (AI), đặc biệt trong bài toán phân loại đa lớp. Nó chuyển đổi một tập hợp các giá trị đầu vào thành một phân phối xác suất đồng thời của các lớp khác nhau.

Đầu vào của hàm softmax là một vector có kích thước n và được ký hiệu là z = [z1, z2, ..., zn]. Hàm softmax sẽ tính toán giá trị xác suất của mỗi phần tử trong vector đầu vào, ký hiệu là p = [p1, p2, ..., pn]. Công thức của hàm softmax được biểu diễn như sau:

pi = exp(zi) / (exp(z1) + exp(z2) + ... + exp(zn))

Trong đó, exp(x) là hàm mũ e^x và tổng của tất cả các giá trị exp(zi) đảm bảo rằng tổng của p sẽ bằng 1.

Hàm softmax có tính chất chuyển đổi các giá trị đầu vào thành các giá trị xác suất, với giá trị lớn nhất tương ứng với xác suất cao nhất. Điều này làm cho hàm softmax rất hữu ích trong bài toán phân loại đa lớp, khi ta muốn xác định xác suất của mỗi lớp dựa trên đầu vào.

Hàm kích hoạt softmax thường được sử dụng kết hợp với hàm mất mát như Cross- Entropy để huấn luyện mô hình phân loại đa lớp trong các thuật toán học máy và mạng nơ-ron.

**3.7. Thuật toán Adam.**

Adam (Adaptive Moment Estimation) là một thuật toán tối ưu hóa gradient descent được sử dụng rộng rãi trong huấn luyện mạng nơ-ron và các mô hình học máy khác. Nó kết hợp hai phương pháp tối ưu hóa khác nhau: AdaGrad và RMSProp, để tận dụng lợi thế của cả hai.

Adam duy trì một tập hợp các tham số để theo dõi và điều chỉnh tốc độ học (learning rate) cho từng tham số trong quá trình huấn luyện. Dưới đây là một số thuộc tính chính của thuật toán Adam:

1. Tích hợp moment bậc nhất (first-order moment): Adam tính toán moment bậc nhất của gradient (tức là giá trị trung bình của gradient) để xác định hướng cập nhật của các tham số.
2. Tích hợp moment bậc hai (second-order moment): Adam tính toán moment bậc hai của gradient (tức là giá trị trung bình của bình phương gradient) để ước lượng độ lớn của gradient và điều chỉnh tốc độ học theo gradient đó.
3. Điều chỉnh tốc độ học theo từng tham số: Adam sử dụng các moment đã tính toán để điều chỉnh tốc độ học cho từng tham số riêng biệt. Điều này cho phép Adam tự điều chỉnh tốc độ học phù hợp với mỗi tham số trong quá trình huấn luyện.
4. Bias correction: Trong các vòng lặp đầu tiên, moment bậc nhất và moment bậc hai được khởi tạo với giá trị gần với 0. Điều này có thể gây ra sự nhiễu và ảnh hưởng đến quá trình đầu tư. Adam sử dụng một bước hiệu chỉnh để khắc phục việc này, được gọi là bias correction.

Adam có nhiều ưu điểm, bao gồm:

* + Hiệu suất cao: Adam kết hợp cả moment bậc nhất và moment bậc hai, giúp cải thiện tốc độ hội tụ và đạt được hiệu suất tối ưu tốt hơn trong quá trình huấn luyện.
  + Tính tự điều chỉnh: Adam tự điều chỉnh tốc độ học cho từng tham số, cho phép nhanh chóng tìm ra kết quả tối ưu và giảm khả năng rơi vào vùng cực tiểu cục bộ
  + Hiệu quả với dữ liệu thưa (sparse data): Adam phù hợp với việc làm việc với dữ liệu thưa, trong đó chỉ có một số lượng nhỏ các đặc trưng quan trọng và phần còn lại là giá trị 0.

Tuy nhiên, Adam cũng có một số hạn chế:

* + Tham số tuning: Adam có nhiều tham số cần được điều chỉnh, bao gồm tốc độ học (learning rate), beta1 (momentum) và beta2 (momentum bậc hai). Việc tuning tham số có thể phức tạp và yêu cầu kiến thức và kinh nghiệm trong quá trình huấn luyện mô hình.
  + Bộ nhớ yêu cầu: Adam cần lưu trữ các moment bậc nhất và bậc hai cho từng tham số, điều này yêu cầu bộ nhớ lớn hơn so với một số thuật toán tối ưu hóa khác.
  + Nhạy cảm với scale của gradient: Adam có thể nhạy cảm với scale của gradient, đặc biệt là khi các tham số có gradients rất lớn hoặc rất nhỏ. Điều này có thể ảnh hưởng đến quá trình huấn luyện và cần được xem xét trong quá trình điều chỉnh tham số.

Tổng quan, Adam là một thuật toán tối ưu hóa phổ biến và mạnh mẽ trong lĩnh vực học máy. Nó kết hợp các ưu điểm của AdaGrad và RMSProp để cung cấp một phương pháp hiệu quả cho việc huấn luyện mô hình với tốc độ học tự điều chỉnh và khả năng làm việc với dữ liệu thưa.

**3.8. Hàm mất mát Categorical-Crossentropy.**

Hàm mất mát Categorical Crossentropy (hay Cross-Entropy Loss) là một hàm mất mát phổ biến được sử dụng trong các bài toán phân loại đa lớp (multiclass classification) trong machine learning. Nó được sử dụng để đo lường sự khác biệt giữa các phân phối xác suất thực tế và các dự đoán xác suất của mô hình.

Hàm mất mát Categorical Crossentropy dựa trên nguyên lý thông tin entropy trong lý thuyết thông tin. Hàm này tính toán tổng entropy (sự mất mát thông tin) của các dự đoán xác suất và nhãn thực tế. Mục tiêu của mô hình là giảm mất mát entropy này để tối ưu hóa hiệu suất phân loại.

**3.9.Kỹ thuật One-Hot Encoding.**

Kỹ thuật One-Hot Coding là một phương pháp quan trọng trong xử lý nhãn (labels) của dữ liệu, đặc biệt là trong các bài toán phân loại nhiều lớp.

One-Hot Coding chuyển đổi các nhãn thành các vectơ nhị phân có kích thước bằng số lượng lớp. Mỗi nhãn được biểu diễn dưới dạng một vectơ có giá trị 1 tại vị trí tương ứng với lớp và 0 tại các vị trí còn lại.

One-Hot Coding giúp biểu diễn nhãn một cách rõ ràng và giúp mô hình học được mối quan hệ giữa các lớp.Được sử dụng rộng rãi trong việc chuẩn bị dữ liệu cho các mô hình học máy, đặc biệt là khi sử dụng mạng nơ-ron sâu.

Giả sử ta có biến phân loại "color" với các giá trị {'Red', 'Green', 'Blue'}.

Bằng cách sử dụng "one-hot encoding," mỗi giá trị sẽ được biểu diễn như sau:

'Red' sẽ được biểu diễn bởi vector [1, 0, 0]

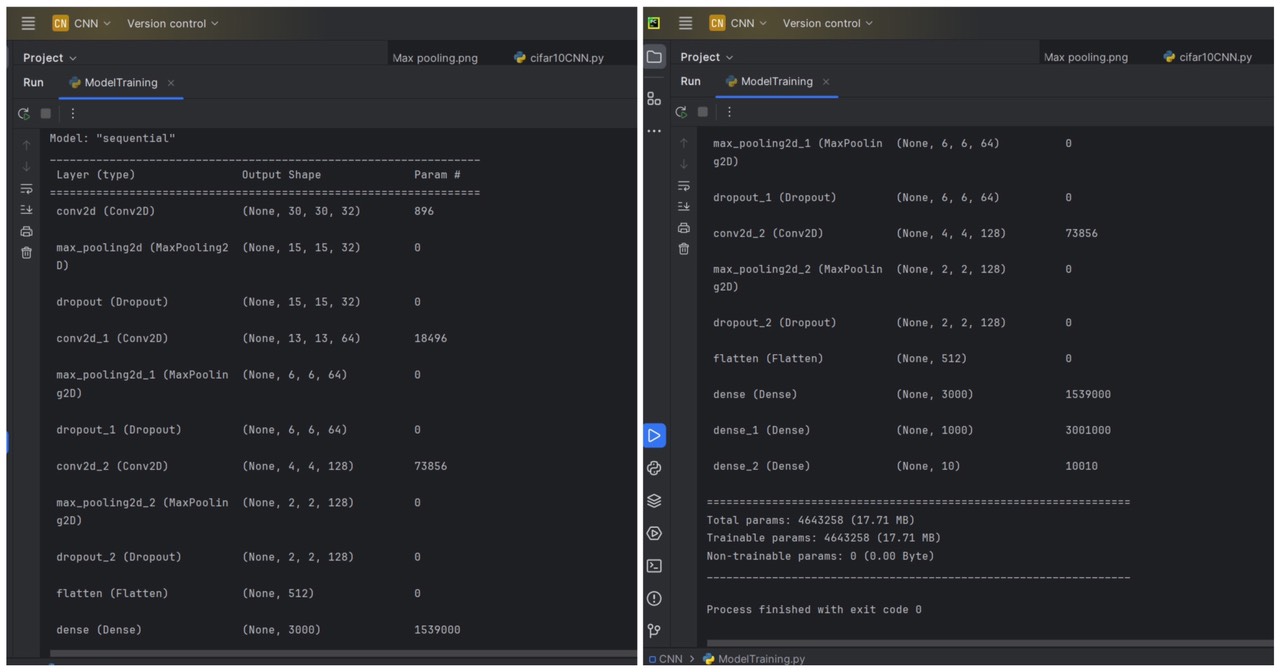
'Green' sẽ được biểu diễn bởi vector [0, 1, 0]

'Blue' sẽ được biểu diễn bởi vector [0, 0, 1]

**3.10. Mô hình tổng quan.**

Bằng cách sử dụng Tensorflow ta có thể dễ dàng học hoặc tạo mạng nơ-ron phân loại ảnh với nhiều lớp lọc đặc trưng của ảnh.

Dưới đây là ảnh mô hình tổng quan mà ta xây dựng.

****

*Hình 10: Tổng quan về kiến trúc của mô hình*

Mô hình này có 3 lớp neuron tích chập, 3 lớp neuron liên kết đầy đủ và tổng tham số là hơn 4 triệu

**CHƯƠNG III: CÀI ĐẶT**

## Giới thiệu về Tensorflow và Keras

**1.1.Tensorflow.**

TensorFlow là một thư viện mã nguồn mở phổ biến trong lĩnh vực machine learning và deep learning. Nó được phát triển bởi Google và cung cấp một nền tảng mạnh mẽ cho việc xây dựng và huấn luyện mô hình học máy.

Dưới đây là một số đặc điểm chính của TensorFlow:

1. Mô hình tính toán dựa trên đồ thị: TensorFlow xây dựng các mô hình tính toán bằng cách sử dụng đồ thị. Các phép tính được biểu diễn dưới dạng các nút (node) và các cung (edge) trong đồ thị, thể hiện quá trình tính toán. Điều này cho phép TensorFlow tận dụng tốt các tài nguyên tính toán và thực hiện song song trên nhiều thiết bị như CPU, GPU hoặc TPU.

Tích hợp rộng rãi với các công cụ phổ biến: TensorFlow tích hợp tốt với các công cụ và thư viện phổ biến như NumPy, scikit-learn và Matplotlib. Điều này giúp người dùng thực hiện các tác vụ phức tạp như xử lý dữ liệu, trực quan hóa và đánh giá mô hình dễ dàng hơn.

1. Hỗ trợ cho deep learning: TensorFlow cung cấp một loạt các lớp và API để xây dựng mô hình deep learning. Bạn có thể xây dựng các mạng neuron đa tầng (multilayer neural networks), mạng nơ-ron tích chập (convolutional neural networks), mạng nơ-ron hồi quy (recurrent neural networks) và nhiều kiến trúc mô hình deep learning khác.
2. Kết nối với TensorFlow Extended (TFX): TensorFlow cung cấp TFX, một nền tảng phát triển và triển khai dựa trên TensorFlow cho việc xây dựng và triển khai các ứng dụng machine learning phức tạp. TFX cung cấp các công cụ và khung làm việc để quản lý dữ liệu, huấn luyện mô hình và triển khai sản phẩm.

TensorFlow đã được sử dụng rộng rãi trong nhiều ứng dụng machine learning và deep learning, từ việc nhận dạng hình ảnh, dịch máy, nhận dạng giọng nói, đến tự động lái xe và phân tích dữ liệu lớn. Với khả năng mạnh mẽ và tích hợp đa dạng, TensorFlow là một công cụ quan trọng trong việc phát triển các ứng dụng nhân tạo và machine learning.

**1.2.Keras.**

Keras là một thư viện mã nguồn mở cung cấp giao diện Python cho mạng nơ-ron nhân tạo. Keras hoạt động như một giao diện cho thư viện TensorFlow. Keras được thiết kế để cho phép thử nghiệm nhanh chóng với mạng nơ-ron sâu, tập trung vào việc dễ sử dụng, linh hoạt và có thể mở rộng. Keras chạy trên các thư viện máy học nguồn mở như TensorFlow, Theano hoặc Cognitive Toolkit (CNTK). Keras được thiết kế để xây dựng mô hình học sâu một cách nhanh chóng. Keras cung cấp các API (Giao diện lập trình ứng dụng) đơn giản và dễ hiểu, giúp người dùng dễ dàng xây dựng và huấn luyện các mô hình học sâu. Keras cũng hỗ trợ nhiều kiến trúc mạng nơ-ron khác nhau, bao gồm cả mạng nơ-ron tích chập (CNN) và mạng nơ-ron hồi quy (RNN).

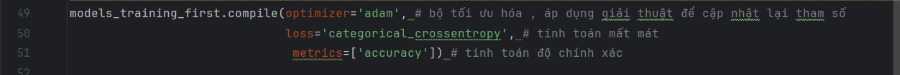
## Cài đặt mô hình

Mô hình được xây dựng theo **CNN** của **TensorFlow** một cách đơn giản với tổng trọng số sử dụng khoảng 4 triệu để có thể phân loại và nhận diện khuôn mặt. Dưới đây là đoạn code cài đặt mô hình :



*Hình 11: Câu lệnh xây dựng mô hình*

Sau đó ta sẽ dùng câu lệnh :



Đoạn code này là phần của mã Python được sử dụng trong quá trình huấn luyện mô hình neural network bằng thư viện TensorFlow

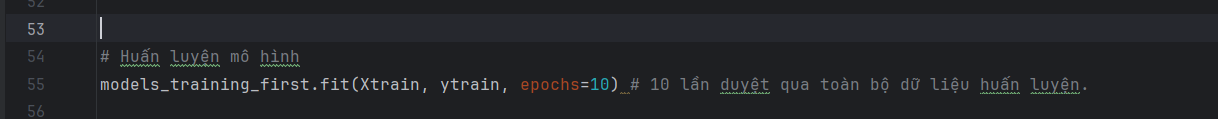
Hàm `compile` được sử dụng để cấu hình mô hình để chuẩn bị cho việc huấn luyện. Nó cần được gọi trước khi bắt đầu huấn luyện mô hình.

Các tham số của hàm `compile` là:

* **`optimizer`**: là tối ưu hóa mà mô hình sử dụng để cập nhật các trọng số trong quá trình huấn luyện. Ở đây, `adam` là một phương pháp tối ưu hóa phổ biến được sử dụng trong deep learning.
* **`loss`**: là hàm mất mát (hay còn gọi là hàm chi phí) được sử dụng để đo lường sự khác biệt giữa giá trị dự đoán và giá trị thực tế. Ở đây, `categorical\_crossentropy` được sử dụng để huấn luyện mô hình phân loại đa lớp.
* **`metrics`**: là các chỉ số được sử dụng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện. Ở đây, chỉ số `accuracy` được sử dụng để đánh giá tỷ lệ dự đoán đúng của mô hình trên tập dữ liệu huấn luyện.

Sau khi được cấu hình bằng hàm `compile`, mô hình sẽ được sử dụng để huấn luyện với dữ liệu đầu vào và tiêu chuẩn mất mát và chỉ số hiệu suất được định nghĩa bởi các tham số trên.

Ta huấn luyện mô hình bằng câu lệnh sau :

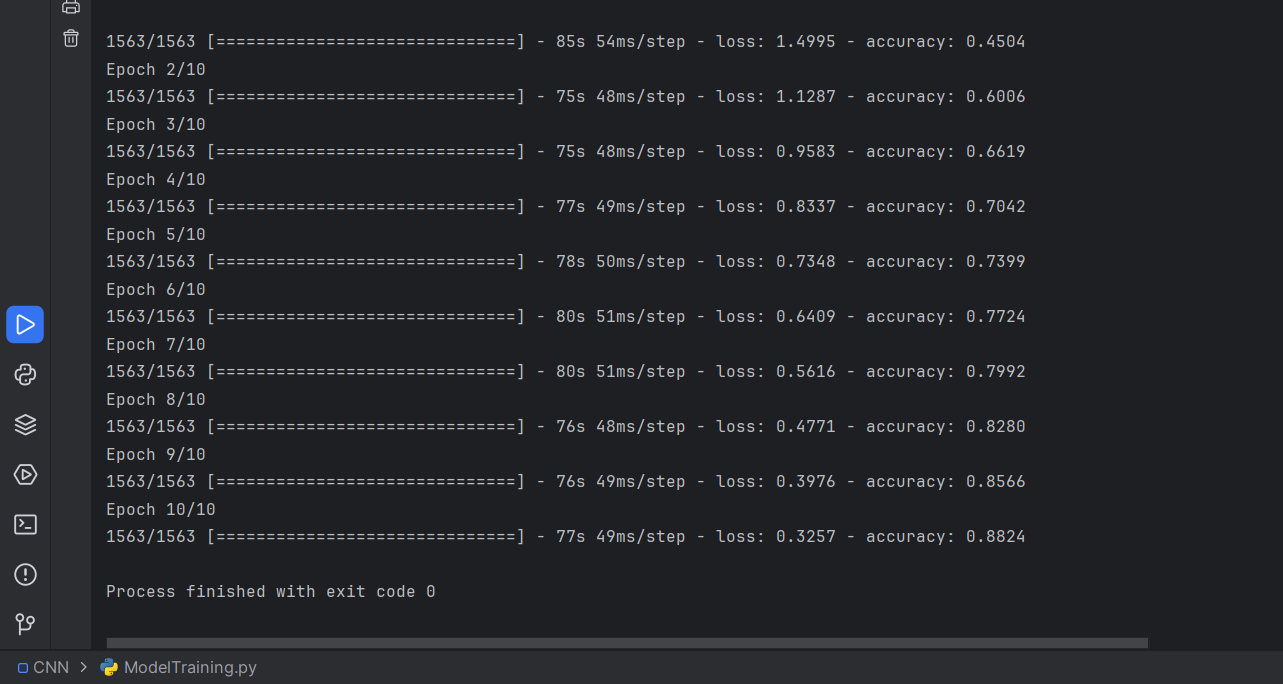


`model` là một đối tượng neural network được định nghĩa trước đó và đã được cấu hình bằng hàm `compile`.

Các tham số của hàm `fit` là:

* `X\_train`: là tập dữ liệu huấn luyện đầu vào (features) được sử dụng để huấn luyện mô hình. Đây là một ma trận numpy.
* `y\_train`: là tập dữ liệu huấn luyện đầu ra (labels) được sử dụng để huấn luyện mô hình. Đây là một ma trận numpy.

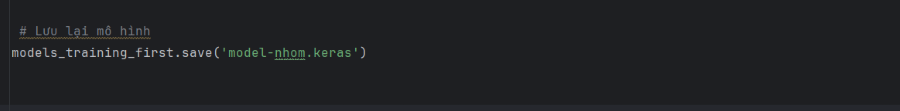
Thông số cuối cùng của mô hình sau khi đã được huấn luyện :



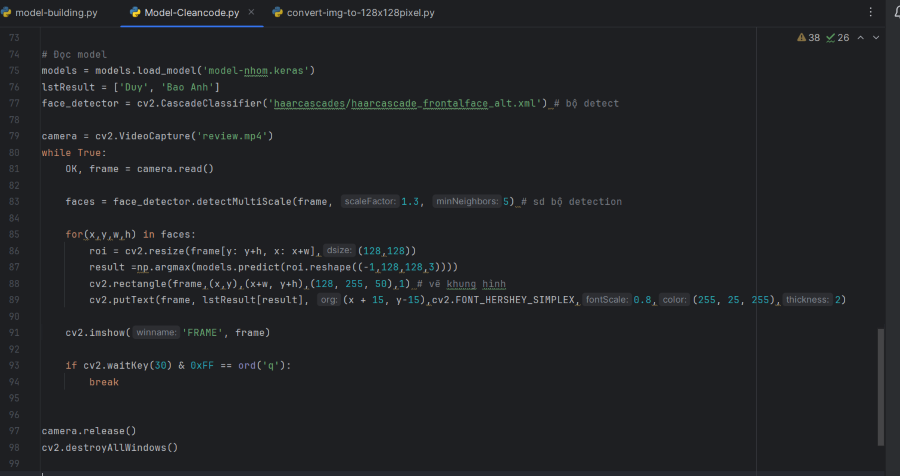
*Hình 12: Mô hình sau khi được huấn luyện*

Ta có thể thấy rằng chỉ số mất mát sẽ giảm đi sau mỗi Epoch và độ chính xác của mô hình đã được cải thiện rất nhiều lần và đến lần cuối mô hình đã có được accuracy = 88% chứng tỏ mô hình đã được điều chỉnh một cách hợp lí để có được thông số chính xác cao như vậy. Nếu ta tăng số lượng epoch lên chắc chắn độ chính xác của mô hình sẽ lên tới 100%.

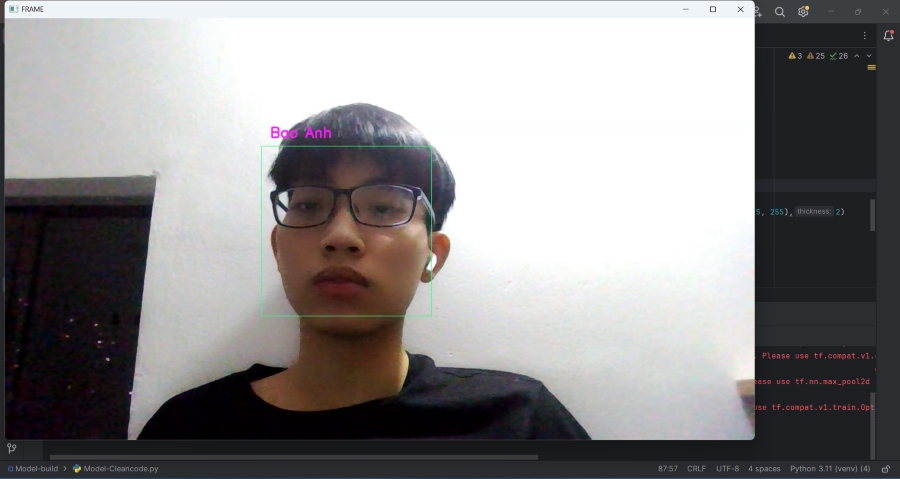
Sau khi có mô hình ta sẽ lưu lại để tái sử dụng :



Cuối cùng là sử dụng mô hình để dự đoán khuôn mặt được xuất hiện trong video :



* Sử dụng hàm ‘load\_model’ để load model đã được lưu
* Sử dụng bộ phát hiện khuôn mặt (detection) của OpenCV có tên là ‘haarcascade-frontalface-alt’ để phát hiện , khoanh vùng chính xác khuôn mặt xuất hiện trong video
* Mở frame , kẻ khung và mô hình sẽ dự đoán được khuôn mặt
* ‘models.predict’ sẽ dự đoán khuôn mặt dựa trên mô hình có sẵn, nó sẽ trả về các xác suất sẽ rơi vào nhóm nào, nếu vị trí có xác suất lớn nhất thì hình sẽ thuộc về nhóm đó
* ‘np.argmax’ sẽ là hàm để trả về vị trí nơi có xác suất lớn nhất, vị trí đó sẽ là tên nhãn của hình ảnh
* Dưới đây là kết quả :



*Hình 12: Kết quả cuối cùng*

**CHƯƠNG IV: TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN**

## Tổng kết

Tuy dữ liệu chưa được hoàn chỉnh và dự đoán của mô hình chưa được tốt nhưng sau project chúng em đã học thêm được rất nhiều điều(mô hình neural network, Tensorflow API,,....) Chúng em sẽ cố gắng tăng cường thêm dữ liệu và cải thiện cấu trúc của mô hình để mô hình chạy được tốt hơn trong tương lai.

## Hướng phát triển

Xây dựng những hệ thống thời gian thực có ứng dụng trong đời sống thực tế như điểm danh bằng khuôn mặt.

**TÀI LIỆU THAM KHẢO**

[1] G. Hinton, “Deep Belief Nets,” IEEE Trans. Neural Networks, vol. 17, no. 6, pp. 1623–1629, 2006, doi: 10.1109/TNN.2006.880582.

[2] S. Gupta, “Deep learning performance breakthrough,” 2018. <https://www.ibm.com/blogs/systems/deep-learning-performance-breakthrough/>.

[3] P. Đ. Thắng, “Các kiến trúc CNN hiện đại,” 2020. <https://phamdinhkhanh.github.io/2020/05/31/CNNHistory.html>.

[4] A. Wasicek, “Artificial Intelligence vs. Machine Learning vs. Deep Learning: 87 What’s the Difference?,” 2018. <https://www.sumologic.com/blog/machinelearning-deep-learning/>.

[5] K. Yan, “Differences between Normalization, Standardization and Regularization,” 2018. <https://maristie.com/2018/02/NormalizationStandardization-and-Regularization>.

[6] S. Marsland, Machine Learning: An Algorithmic Perspective, Second Edition. 86 Stephen Marsland, 2014.