TRƯỜNG ĐẠI HỌC SƯ PHẠM KĨ THUẬT TP.HCM

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

ĐỒ ÁN CUỐI KÌ

BỘ MÔN: TRÍ TUỆ NHÂN TẠO

XÁC ĐỊNH CÁC HUYỆT ĐẠO MASSAGE TRÊN ĐẦU NGƯỜI

GVHD: PGS.TS Nguyễn Trường Thịnh

Sinh viên thực hiện:

Nguyễn Hoài Nam – 19146219

Thành phố Hồ Chí Minh tháng 6 năm 2022

THÔNG TIN SINH VIÊN THỰC HIỆN

HỌ VÀ TÊN	MSSV	SĐT	EMAIL
Nguyễn Hoài Nam	19146219	0906732772	19146219@student.hcmute.edu.vn

MỤC LỤC

PHỤ LỤC HÌNH ẢNH

LỜI CẢM ƠN

Em xin dành trang viết trân trọng này để gửi lời cảm ơn chân thành nhất đến Ba Mẹ của em. Người mà đã giúp chúng em có điều kiện để đến trường mỗi ngày. Ba mẹ đã giúp chúng em rất nhiều trong việc làm tiểu luận cuối kì không chỉ về mặt tinh thần mà còn chăm lo sức khỏe cho em.

Thứ 2 em xin được phép gửi lời cảm ơn chân thành đến Ban giám hiệu trường Đại Học Sư Phạm Kỹ Thuật TP.HCM cũng như các quý thầy cô trường Đại Học Sư Phạm Kỹ Thuật TP.HCM đã truyền đạt kiến thức từ những môn trước đó để chúng em có kiến thức để làm các phần tính toán truyền động.

Thứ 3 em xin được phép gửi lời cảm ơn chân thành đến thầy Nguyễn Trường Thịnh, người đã luôn nhiệt tình hướng dẫn, truyền đạt, chỉ dẫn chúng em những gì cơ bản nhất về Trí tuệ nhân tạo đến nâng cao giúp cho chúng em có đầy đủ kiến thức để hoàn thành tiểu luận

Em đã vận dụng tốt những kiến đã học, thêm vào đó là sự cố gắng, đam mê hết mình vào tiểu luận tuy nhiên cũng không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và đóng góp ý kiến quý báu từ thầy để bài báo cáo hoàn thiện hơn.

PHẦN 1: MỞ ĐẦU

1.1 Lý do chọn đề tài

Trong một xã hội hiện đại và phát triển không ngừng, con người dần biết chú trọng hơn đến sức khỏe và chất lượng của cuộc sống. Hiện nay, có rất nhiều lựa chọn, giải pháp để chăm sóc, cải thiện và tăng cường sức khỏe với các trang thiết bị và đội ngũ y tế vô cùng hiện đại, thông minh và hiệu quả cao nhưng đổi lại chi phí để được tham gia sử dụng và trải nghiệm lại vô cùng tốn kém. Vì thế, để vẫn mang lại những lợi ích sức khỏe như trên nhưng lại ít tốn kém và đặc biệt là không cần dùng thuốc, con người còn có thể áp dụng một liệu pháp đã có từ rất xa xưa đó chính là tác động vào các "Huyệt Đạo". Bằng các liệu pháp như xoa bóp, bấm huyệt và châm cứu có thể giúp máu huyết lưu thông, điều hòa sự hô hấp, cải thiện tim mạch, giảm đau, xua tan căng thẳng, mệt mỏi, thư giãn, điều trị mất ngủ và ngăn chặn các bệnh mãn tính. Tuy nhiên, hiện nay bản đồ của cơ thể người có tổng cộng 365 huyệt đạo, việc xác định đúng các điểm huyệt đạo không phải việc đơn giản và có thể cần tới các chuyên gia, bác sĩ đông y có kinh nghiệm. Nhận thấy sự khó khăn trên, em đã lên ý tưởng và thiết kế một mô hình mạng neural nhân tạo để nhận diện các điểm huyệt đạo. Do số lượng huyệt đạo trên cơ thể người là khá nhiều, nên em đã giới hạn và chỉ thực hiện việc xác định các huyệt đạo massage chính trên đầu người.

1.2 Mục đích nghiên cứu

Việc xác định các huyệt đạo trên cơ thể người là không hề đơn giản với những người không có chuyên môn sâu và kinh nghiệm về y học cổ truyền. Chính vì vậy, việc xây dựng một mô hình ứng dụng trí tuệ nhân tạo để xác định các huyệt đạo là vô cùng hữu ích và cần thiết. Khi đó, mọi người đều có thể xác định được các điểm huyệt đạo của cơ thể một cách nhanh chóng, dễ dàng và hạn chế chi phí nhất có thể.

Việc tham gia đề tài nghiên cứu "Xác định các huyệt đạo massage trên đầu người" lần này giúp em áp dụng được các kiến thức đã được học trong bộ môn Trí tuệ nhân tạo, đồng thời tích lũy thêm kinh nghiệm, phát triển kỹ năng tự học, và tự nghiên cứu cho bản thân.

1.3 Phạm vi nghiên cứu

Trong chương trình xây dựng cho đề tài này em sẽ áp dụng mạng Neural tích chập CNN (Convolutional Neural Network) là môt trong những mô hình của Deep Learning. CNN cho phép xây dựng các hệ thống thông minh với độ chính xác vô cùng cao và được ứng dụng rất nhiều trong những bài toán nhận dạng vật thể trong ảnh. Bên cạnh đó, em cũng sử dụng thêm các thư viện liên quan đến Deep Learning như Keras và TensorFlow. Thư viện liên quan đến xử lý dữ liệu như Numpy và Pandas.

1.4 Ứng dụng thực tiễn

Mô hình được ứng dụng vào thức tiễn theo một quy trình nhất định, sau khi thu thập dữ liệu hình ảnh đầu người từ camera, hình ảnh sẽ được đưa vào mô hình trải qua các bước xử lý, sau đó so sánh với các hình ảnh đã được gắn nhãn vị trí các điểm huyệt đạo trước đó. Trải qua quá trình đánh giá thông qua các thuật toán phức tạp bên trong, mô hình sẽ tiến hành dự đoán tọa độ các điểm huyệt đao tương ứng với vị trí khuôn mặt của hình ảnh đưa vào và hiển thị cho người dùng biết các vị trí huyệt đạo thông qua trực quan hóa điểm dữ liệu. Bên cạnh đó, mô hình dự đoán huyệt đạo này, còn áp dụng cho các loại máy móc, thiết bị massage trong việc tự tìm ra các điểm huyệt đạo để thực hiện các chức năng của chúng.

1.5 Nội dung báo cáo

Phần 1: Tổng quan đề tài: Nêu lý do, mục đích, phạm vi và ứng dụng của đề tài

Phần 2: Trình bày về cơ sở lý thuyết: Trí tuệ nhân tạo, Học máy, ANN và CNN

Phần 3: Trình bày quy trình xây dựng mô hình: xác định bài toán, thu thập và xử lý dữ liệu, quá trình đào tạo mô hình, kiểm tra độ chính xác và đánh giá mô hình.

Phần 4: Kết luận: đưa ra hạn chế và giải pháp, cải tiến và hướng phát triển trong tương lai

PHẦN 2: CƠ SỞ LÝ THUYẾT

2.1 Trí tuệ nhân tạo

Trí tuệ nhân tạo hay trí thông minh nhân tạo (Artificial intelligence – viết tắt là AI) là một ngành thuộc lĩnh vực khoa học máy tính (Computer science). Là trí tuệ do con người lập trình tạo nên với mục tiêu giúp máy tính có thể tự động hóa các hành vi thông minh như con người. Trí tuệ nhân tạo khác với việc lập trình logic trong các ngôn ngữ lập trình là ở việc ứng dụng các hệ thống học máy (Machine learning) để mô phỏng trí tuệ của con người trong các xử lý mà con người làm tốt hơn máy tính. Trí tuệ nhân tạo được hoạt động và điều khiển bởi các thuật toán có ràng buộc các điều kiện cụ thể. Mục đích của thuật toán này là hỗ trợ kết nối tư duy, nhận thức và hành động với nhau, sau đó sẽ được thể hiện lên các mô hình máy móc. Trí tuệ nhân tạo được chia làm 4 loại chính:

- Công nghệ AI phản ứng (Reactive Machines): Công nghệ AI phản ứng có khả năng phân tích những động thái khả thi nhất của chính mình và của đối thủ, từ đó, đưa ra được giải pháp tối ưu nhất. Một ví dụ điển hình của công nghệ AI phản ứng là Deep Blue. Đây là một chương trình chơi cờ vua tự động, được tạo ra bởi IBM.

- Công nghệ AI với bộ nhớ hạn chế (Limited Memory): Công nghệ AI này thường kết hợp với cảm biến môi trường xung quanh nhằm mục đích dự đoán những trường hợp có thể xảy ra và đưa ra quyết định tốt nhất cho thiết bị. Ví dụ như đối với xe không người lái, nhiều cảm biến được trang bị xung quanh xe và ở đầu xe để tính toán khoảng cách với các xe phía trước, AI sẽ dự đoán khả năng xảy ra va chạm, từ đó điều chỉnh tốc độ xe phù hợp để giữ an toàn cho xe.

- Lý thuyết trí tuệ nhân tạo (Limited Memory): Công nghệ AI này có thể học hỏi cũng như tự suy nghĩ, sau đó áp dụng những gì học được để thực hiện một việc cụ thể.

- Tự nhận thức (Theory of Mind): Công nghệ AI này có khả năng tự nhận thức về bản thân, có ý thức và hành xử như con người. Thậm chí, chúng còn có thể bộc lộ cảm xúc cũng như hiểu được những cảm xúc của con người.

Hiện nay, trí tuệ nhân tạo đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực.

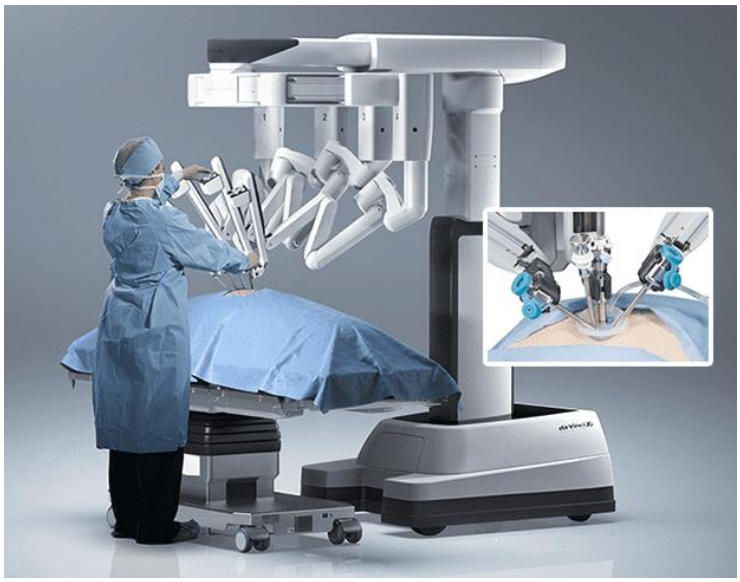
- **Trong ngành vận tải**: Trí tuệ nhân tạo được ứng dụng trên những phương tiện vận tải tự lái, điển hình là ô tô. Sự ứng dụng này góp phần mang lại lợi ích kinh tế cao hơn nhờ khả năng cắt giảm chi phí cũng như hạn chế những tai nạn nguy hiểm đến tính mạng.

Hình 2.1 Ứng dụng AI trong ngành vận tải

- **Trong sản xuất**: Trí tuệ nhân tạo được ứng dụng để xây dựng những quy trình sản xuất tối ưu hơn. Công nghệ AI có khả năng phân tích cao, làm cơ sở định hướng cho việc ra quyết định trong sản xuất.

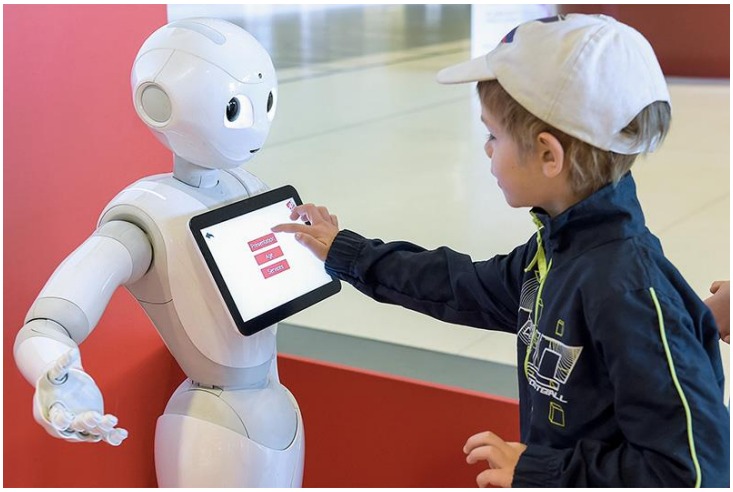
Hình 2.2 Ứng dụng AI trong ngành sản xuất

- **Trong y tế**: Ứng dụng tiêu biểu của trí tuệ nhân tạo trong lĩnh vực y tế là máy bay thiết bị bay không người lái được sử dụng trong những trường hợp cứu hộ khẩn cấp. Thiết bị bay không người lái có tốc độ nhanh hơn xe chuyên dụng đến 40% và vô cùng thích hợp để sử dụng ở những nơi có địa hình hiểm trở.



Hình 2.3 Ứng dụng AI trong ngành y tế

- **Trong giáo dục**: Sự ra đời của trí tuệ nhân tạo giúp tạo ra những thay đổi lớn trong lĩnh vực giáo dục. Các hoạt động giáo dục như chấm điểm hay dạy kèm học sinh có thể được tự động hóa nhờ công nghệ AI. Nhiều trò chơi, phần mềm giáo dục ra đời.



Hình 2.4 Ứng dụng AI trong ngành giáo dục

- **Trong truyền thông:** Đối với lĩnh vực truyền thông, sự phát triển của trí tuệ nhân tạo góp phần làm thay đổi cách thức tiếp cận đối với khách hàng mục tiêu. Nhờ những ưu điểm của công nghệ AI, các công ty có thể cung cấp quảng cáo vào đúng thời điểm, đúng khách hàng tiềm năng, dựa trên việc phân tích các đặc điểm về nhân khẩu học, thói quen hoạt động trực tuyến và những nội dung mà khách hàng thường xem trên quảng cáo.

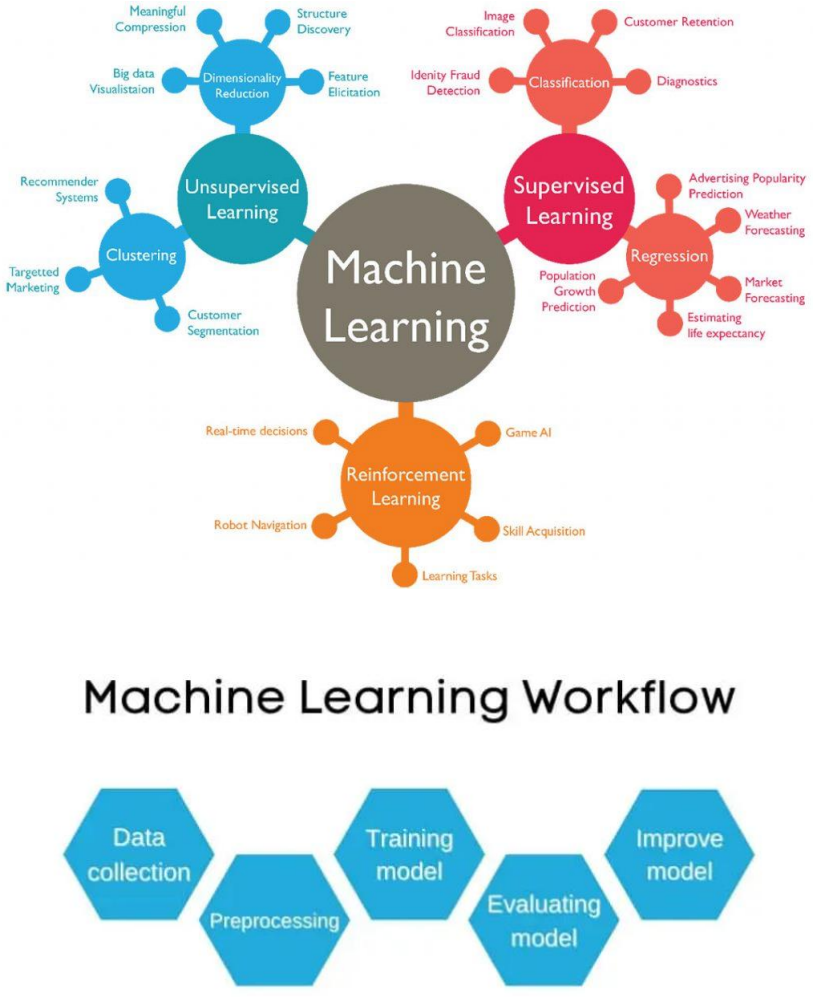
Hình 2.5 Ứng dụng AI trong ngành giáo dục

- **Trong ngành dịch vụ:** Công nghệ AI giúp ngành dịch vụ hoạt động tối ưu hơn và góp phần mang đến những trải nghiệm mới mẻ hơn và tốt hơn cho khách hàng. Thông qua việc thu thập và phân tích dữ liệu, công nghệ AI có thể nắm bắt thông tin về hành vi sử dụng dịch vụ của khách hàng, từ đó mang lại những giải pháp phù hợp với nhu cầu của từng khách hàng.

Hình 2.6 Ứng dụng AI trong truyền thông

2.2 Học máy

Machine learning (máy học) là một nhánh của trí tuệ nhân tạo (AI), là một lĩnh vực nghiên cứu cho phép máy tính có khả năng có thể tự học và ra quyết định, cải thiện chính bản thân chúng dựa trên dữ liệu mẫu (dữ liệu huấn luyện) hoặc dựa vào kinh nghiệm (những gì đã được học). Machine learning có thể tự dự đoán hoặc đưa ra quyết định mà không cần được lập trình cụ thể. Bài toán Machine learning thường được chia làm hai loại là dự đoán (prediction) và phân loại (classification). Các bài toán dự đoán như dự đoán giá nhà, giá xe… Các bài toán phân loại như nhận diện chữ viết tay, nhận diện đồ vật, nhận diện mặt người của Facebook, hệ thống gợi ý bài hát của bài hát, video của Spotify và Youtube…



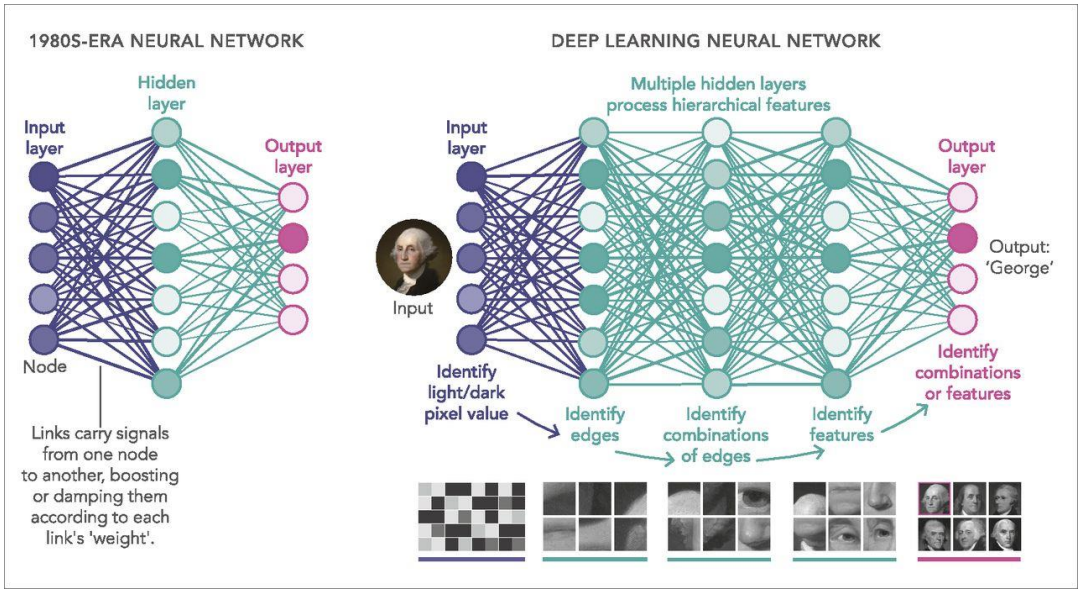
Hình 2. 7 Machine Learning

2.3 Học sâu

Deep Learning là một nhánh của Machine Learning sử dụng mạng lưới thần kinh với nhiều lớp. Deep Learning sử dụng các thuật toán có cấu trúc tương tự như hệ thống nơron thần kinh trong não người (neural network). Một mạng lưới thần kinh sâu phân tích dữ liệu với các biểu diễn đã học tương tự như cách một người nhìn vào một vấn đề.

Trong Machine Learning truyền thống, thuật toán được cung cấp một tập hợp các tính năng có liên quan để phân tích. Tuy nhiên, trong học sâu, thuật toán được cung cấp dữ liệu thô và tự quyết định các tính năng có liên quan. Mạng Deep Learning thường sẽ cải thiện khi tăng lượng dữ liệu được sử dụng để đào tạo chúng.

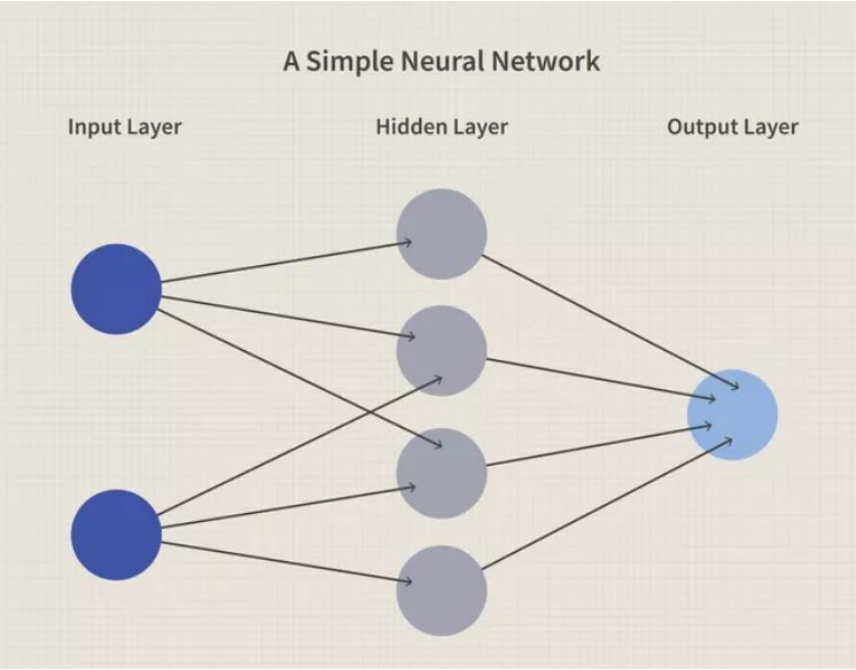
Deep Learning về bản chất là một nhánh của AI cố gắng bắt chước cách thức hoạt động của bộ não con người. Giống như con người học hỏi từ kinh nghiệm, một thuật toán Deep Learning có thể thực hiện một nhiệm vụ nhiều lần, mỗi lần điều chỉnh nó để cải thiện kết quả. Thuật ngữ "Deep Learning" dùng để chỉ các mạng lưới thần kinh có nhiều lớp cho phép học tập. Deep Learning có thể giải quyết bất kỳ vấn đề nào cần 'suy nghĩ' để tìm ra.



Hình 2.8 Deep Learning

2.4 Mạng thần kinh nhân tạo (ANN)

Mạng neuron nhân tạo (Artificial Neural Network gọi tắt là ANN) lấy cảm hứng từ cấu tạo của mạng thần kinh trong não người và động vật. Thông qua một loạt các thuật toán được sắp xếp bên trong mạng, mô hình sẽ tự phân tích đầu vào của dữ liệu, bóc tách đặc trưng giống như các mà não bộ hoạt động.



Hình 2.9 Artificial Neural Network

Ta đề cập đến một mô hình ANN đơn giản nhất với duy nhất 1 lớp đầu vào (input), 1 lớp ẩn (hidden layer) và 1 lớp đầu ra (output). Trong lớp ẩn sẽ có một số lượng neuron nhất định kết nối với các neuron của đầu vào và đầu ra.

Trong ANN, trừ input layer thì tất cả các node thuộc các layer khác đều full-connected với các node thuộc layer trước nó. Mỗi node thuộc hidden layer nhận vào ma trận đầu vào từ layer trước và kết hợp với trọng số để ra được kết quả.

Tuy nhiên, chỉ 1 lớp ẩn (hidden layer) là không đủ để máy tính có thể suy luận và đưa ra kết quả chính xác nên dẫn đến sự ra đời của Multi-layered Preceptron. Multi-layered perceptron là một sự phát triển của Perceptron, lúc này là có nhiều neuron ở giữa là chúng được sắp xếp theo nhiều lớp mà chúng ta không quan tâm đến gọi là nhiều lớp ẩn (Hidden

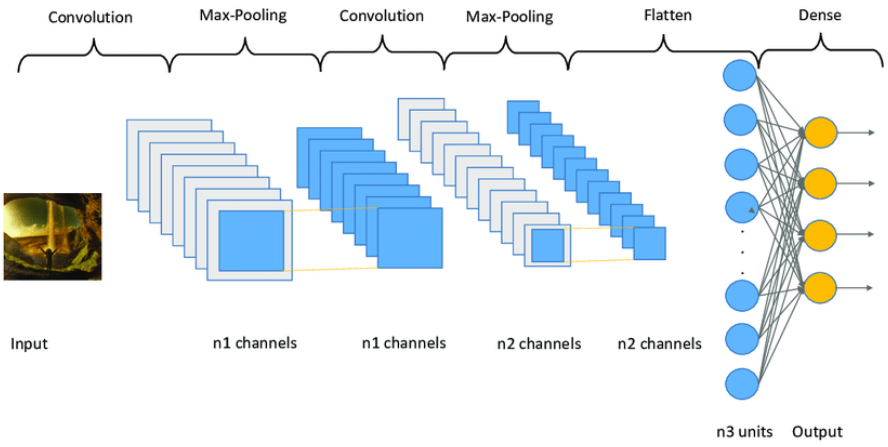
layers). Việc thêm nhiều lớp ẩn sẽ tạo cho model một cách suy luận với nhiều đặc trưng được bóc tách, phát hiện, từ đó giúp cho mô hình có thể cho ra được kết quả chính xác hơn.

2.5 Mạng Neural tích chập (CNN)

Convolutional Neural Network (CNN – mạng neural tích chập) là một trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.

CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected). Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và đưa ra kết quả dựa trên giá trị.



Hình 2.10 Convolution Neural Network Workflow

Cấu tạo cơ bản của một mô hình CNN bao gồm:

- Convolutional layer: Đây là lớp quan trọng nhất của CNN, lớp này có nhiệm vụ thực hiện mọi tính toán. Những yếu tố quan trọng của một convolutional layer là: stride, padding, filter map, feature map.

  + CNN sử dụng các filter để áp dụng vào vùng của hình ảnh. Những filter map này được gọi là ma trận 3 chiều, mà bên trong nó là các con số và chúng là parameter.
  + Stride có nghĩa là khi bạn dịch chuyển filter map theo pixel dựa vào giá trị trừ trái sang phải. Và sự chuyển dịch này chính là Stride.
  + Padding: Là các giá trị 0 được thêm vào với lớp input.
  + Feature map: Nó thể hiện kết quả của mỗi lần filter map quét qua input. Sau mỗi lần quét sẽ xảy ra quá trình tính toán.

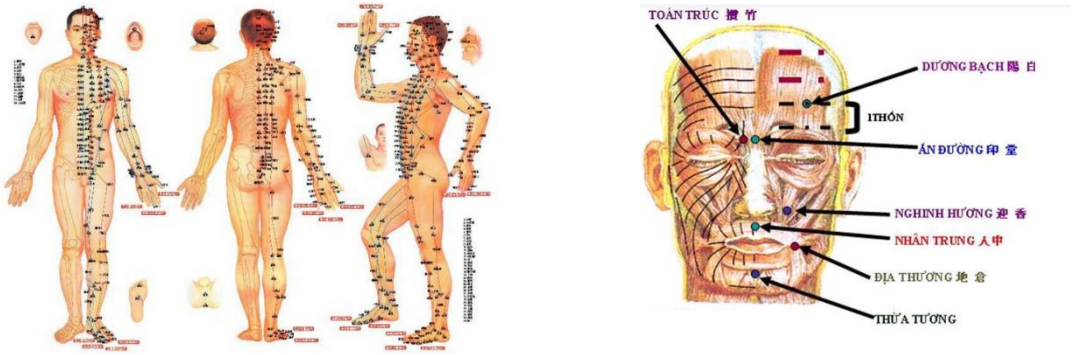
- Relu layers: là hàm kích hoạt trong neural network và hàm này còn được gọi là activation function. Hàm kích hoạt có tác dụng mô phỏng các neuron có tỷ lệ truyền xung qua axon. Trong activation function thì nó còn có hàm nghĩa là: Relu, Leaky, Tanh, Sigmoid, Maxout... Hiện nay, hàm relu được dùng phổ biến và vô cùng thông dụng. Nó được sử dụng nhiều cho các nhu cầu huấn luyện mạng neuron thì relu mang lại rất nhiều ưu điểm nổi bật như: việc tính toán sẽ trở nên nhanh hơn... Quá trình sử dụng relu, chúng ta cần lưu ý đến vấn đề tùy chỉnh các learning rate và theo dõi dead unit. Những lớp relu layer đã được sử dụng sau khi filter map được tính ra và áp dụng hàm relu lên những giá trị của filter map. Hàm ReLU có nhiều biến thể khác như Noisy ReLU, Leaky ReLu, ELUs. Tôi xin phép dừng phần này ở đây vì chưa có ý định đi sâu vào Deep Neural Networks.
- Pooling layers: 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. Các pooling có thể có nhiều loại khác nhau: Max Pooling, Average Pooling và Sum Pooling. Max pooling

lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling.

2.6 Tổng quan về huyệt đạo theo lý thuyết y học cổ truyền

Trong nhiều thế kỷ, con người đã dần hoàn thiện bản đồ huyệt đạo của cơ thể con người một cách chính xác. Bản đồ hoàn thiện nhất thể hiện cơ thể của chúng ta có 365 huyệt đạo trong đó có 108 đại huyệt và 257 tiểu huyệt và mỗi vị trí đều đại diện cho một nguồn năng lượng khác nhau góp phần điều khiển, điều hòa năng lượng bên trong cơ thể giúp chúng ta duy trì cuộc sống khỏe mạnh. Ứng dụng cơ chế dòng chảy năng lượng của các vị trí huyệt đạo, y học cổ truyền Việt Nam đã đưa ra nhiều liệu pháp nhằm hỗ trợ điều trị các vấn đề về sức khỏe của con người.

Tất cả các cơ quan tổ chức của cơ thể người đều nhờ sự nuôi dưỡng của khí huyết mới có thể tồn tại và hoạt động được chức năng riêng của chúng. Nhưng không phải khí huyết tự nó chảy tràn lan trong thân thể, mà nó phải nhờ đến hệ thống kinh lạc làm đường dẫn mới có thể đến được từng cơ quan bộ phận. Kinh lạc là 1 mạng lưới ngang dọc trong cơ thể, giúp nối liền các cơ quan nội tạng bên trong với các bộ phận bên ngoài và trao đổi năng lượng với tự nhiên.Khi bộ phận nội tạng bên trong bị bệnh thì nó sẽ biểu hiện ra các đường kinh lạc tương ứng, ngược lại khi kinh lạc bị các yếu tố bên ngoài gây bệnh thì bệnh từ kinh lạc sẽ truyền vào các bộ phận bên trong. Nhưng đã nói đến kinh lạc thì không thể không nhắc đến các huyệt vị. Một trong những yếu tố người xưa rất coi trọng trong chẩn bệnh chữa bệnh và dưỡng sinh.



Hình 2.11 Bản đồ huyệt đạo trên cơ thể người theo y học cổ truyền Việt Nam

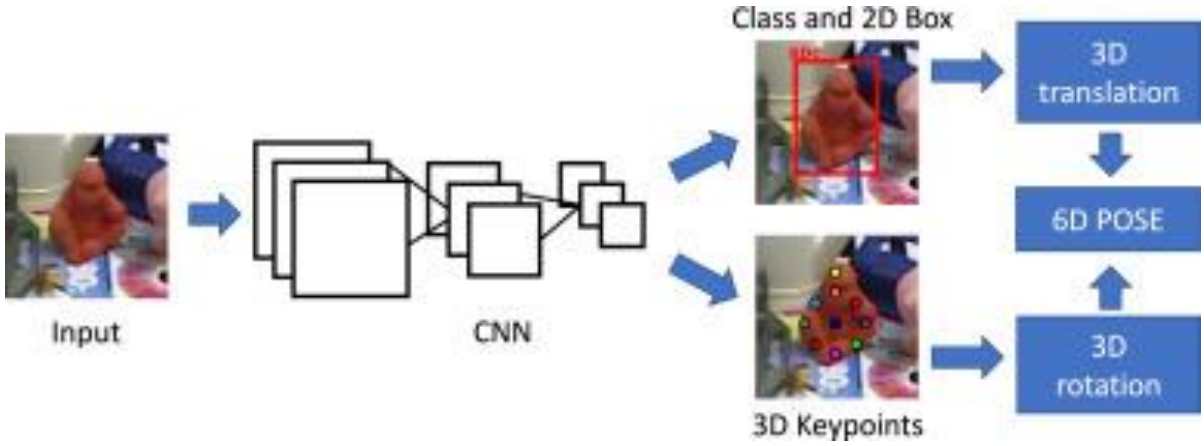
PHẦN 3: QUY TRÌNH TRIỂN KHAI

3.1 Xác định bài toán

Đối với đề tài này, mục đích cuối cùng là xác định đúng được vị trí các huyệt đạo trên đầu người từ hình ảnh camera thu thập được. Vì vậy ta sẽ không sử dụng bài toán phân loại hình ảnh (Object Classification) mà thay vào đó là sử dụng bài toán phát hiện vật thể (Object Detection). Đối với bài toán phát hiện vật thể, ta sẽ áp dụng cụ thể bài toán xác định tọa độ điểm (Keypoint Detection), tùy vào số lượng điểm huyệt đạo trong ảnh đã được gắn nhãn (label) trước mà sẽ cho ra những đầu ra khác nhau, mỗi đầu ra bao gồm 2 yếu tố chính:

3 Tọa độ của điểm (Keypoint coordinates).

4 Tên lớp (Class) mà điểmz đó thuộc vào.

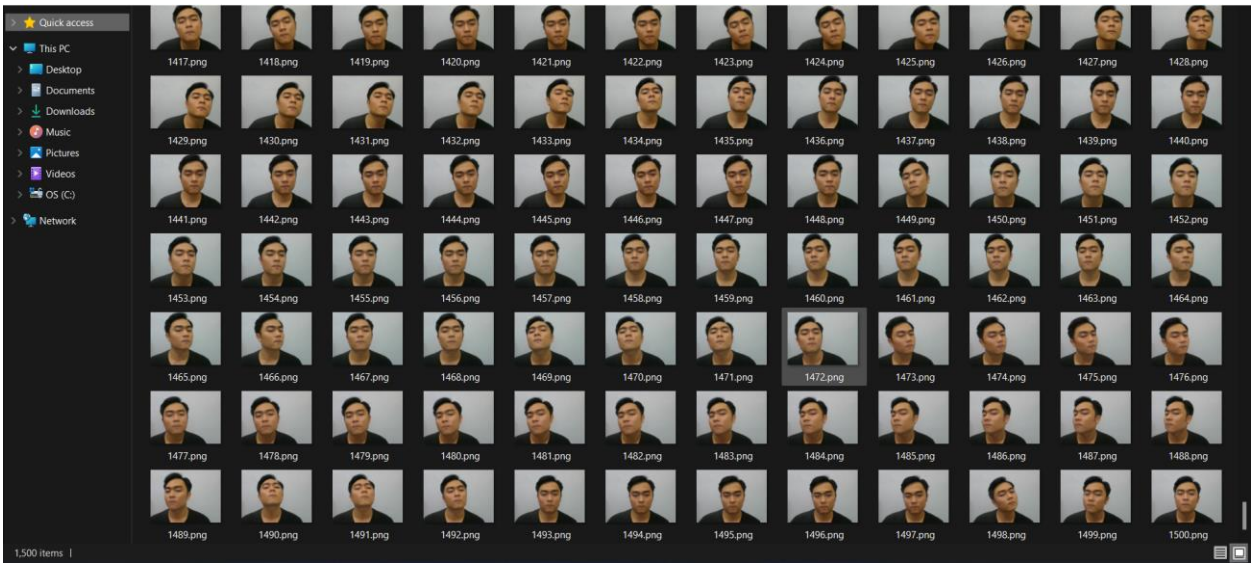


Hình 3.1 Ví dụ về mô hình bài toán phát hiện vật thể và tọa độ điểm

3.2 Thu thập và xử lý dữ liệu

3.2.1. Thu thập dữ liệu

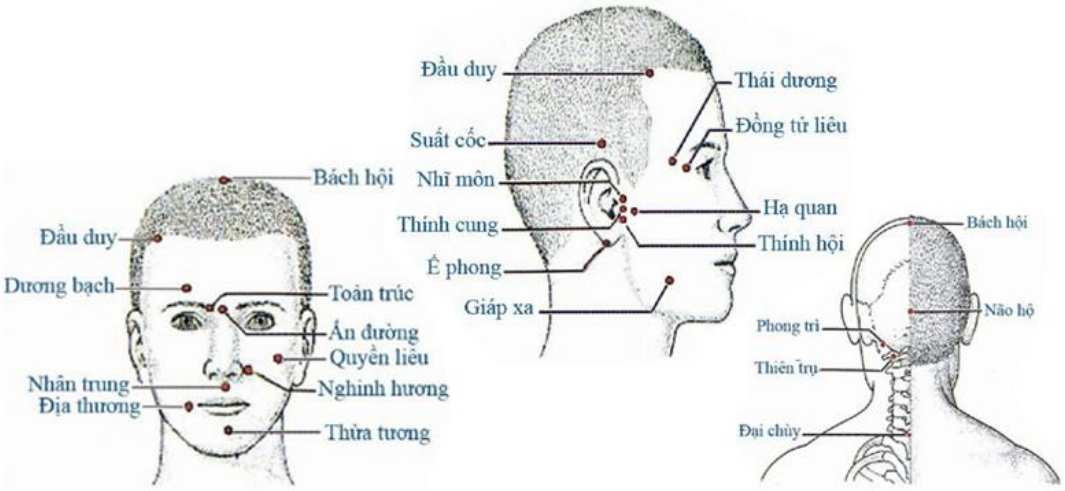
Trước tiên, em lấy dữ liệu hình ảnh tự chụp bản thân, em đã chụp 1500 tấm ảnh với các góc mặt và vị trí đặt mặt khác nhau trong một khung ảnh có kích thước là 640x480 pixel.



Hình 3.3 Tập dữ liệu 1500 tấm ảnh thô

Hình 3.2 Hình ảnh dữ liệu thô

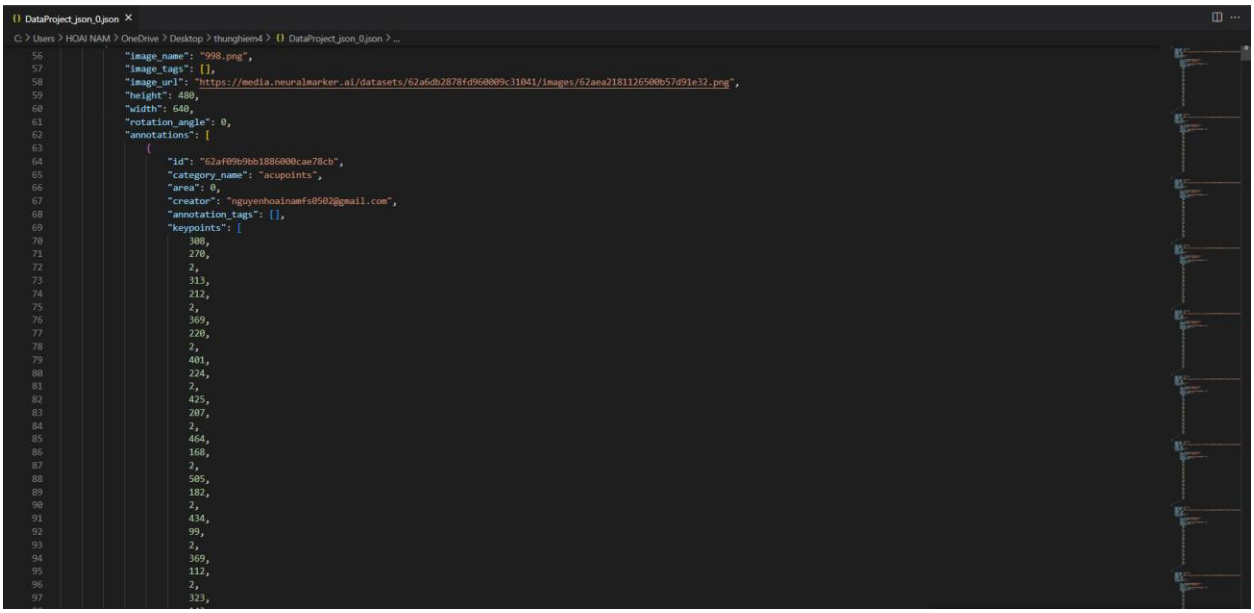
Sau đó, em tiến hành gắn nhãn (label) cho các vị trí huyệt đạo dựa theo lý thuyết y học cổ truyền về các điểm huyệt đạo massage chính trên đầu người. Ở đây em sẽ chọn các huyệt đạo chính áp dụng cho việc massage trên đầu người là: Thái Dương, Dương Bạch, Toản Trúc, Ấn Đường, Đầu Duy và Bách Hội



Hình 3.4 Mô tả các điểm huyệt đạo theo y học cổ truyền Việt Nam

Hình 3.5 Hình ảnh dữ liệu đã gắn nhãn các vị trí điểm huyệt đạo

Sau khi đã gắn nhãn (label) vị trí các điểm huyệt đạo cho 1350 tấm ảnh, em tiến hành xuất các giá trị tọa độ điểm đã được định vị trên từng tấm ảnh, tọa độ điểm của huyệt đạo có hai giá trị là của x (trục hoành) và y (trục tung) trong khung ảnh 640 x 480 pixel. Một tấm ảnh đã được gắn nhãn sẽ có 10 điểm huyệt đạo tương ứng với 20 giá trị tức 10 giá trị tọa độ x và 10 giá trị tọa độ y.



Hình 3.6 Tệp giá trị tọa độ dưới định dạng file json

Do phần mềm em sử dụng để gắn nhãn (Neural Maker) có định dạng xuất file mặc định là json. Nên em tiến hành chuyển đổi lại định dạng thành file csv thông qua trang chuyển đổi định dạng trực tuyến ở google.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	image_nar	image_url	height	width	rotation_a	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio	annotatio
2	1.png	https://me	480	640	0	62aea81fb	acupoints	0	nguyenhoa	274	195	2	314	150	2	364	168	2	397	183	2	420	175
3	10.png	https://me	480	640	0	62aea82cb	acupoints	0	nguyenhoa	243	183	2	271	145	2	318	151	2	349	163	2	375	155
4	100.png	https://me	480	640	0	62aea838l	acupoints	0	nguyenhoa	295	213	2	337	173	2	395	196	2	425	212	2	441	203
5	1000.png	https://me	480	640	0	62aea853l	acupoints	0	nguyenhoa	332	324	2	343	267	2	401	264	2	441	258	2	456	227
6	1001.png	https://me	480	640	0	62aea89ct	acupoints	0	nguyenhoa	343	302	2	360	252	2	417	246	2	456	247	2	473	222
7	1002.png	https://me	480	640	0	62aea8a9l	acupoints	0	nguyenhoa	337	198	2	375	159	2	420	182	2	450	198	2	479	187
8	1003.png	https://me	480	640	0	62aea8b5l	acupoints	0	nguyenhoa	326	179	2	352	147	2	404	166	2	434	186	2	464	181
9	1004.png	https://me	480	640	0	62aea8bel	acupoints	0	nguyenhoa	319	182	2	349	145	2	393	166	2	424	183	2	451	176
10	1005.png	https://me	480	640	0	62aea8c7t	acupoints	0	nguyenhoa	311	188	2	337	147	2	382	164	2	418	181	2	448	173
11	1006.png	https://me	480	640	0	62aea8d4l	acupoints	0	nguyenhoa	309	188	2	335	150	2	384	166	2	416	185	2	444	174
12	1007.png	https://me	480	640	0	62aea8ddl	acupoints	0	nguyenhoa	257	211	2	269	166	2	311	173	2	340	183	2	367	170
13	1008.png	https://me	480	640	0	62aea8e7l	acupoints	0	nguyenhoa	209	238	2	212	193	2	257	188	2	290	187	2	311	167
14	1009.png	https://me	480	640	0	62aea8f1b	acupoints	0	nguyenhoa	201	248	2	200	200	2	252	188	2	284	189	2	308	163
15	101.png	https://me	480	640	0	62aea8feb	acupoints	0	nguyenhoa	293	215	2	337	170	2	396	195	2	423	212	2	449	199
16	1010.png	https://me	480	640	0	62aea909l	acupoints	0	nguyenhoa	184	261	2	183	206	2	231	195	2	264	194	2	281	169
17	1011.png	https://me	480	640	0	62aea916l	acupoints	0	nguyenhoa	228	157	2	268	115	2	304	140	2	329	160	2	365	158
18	1012.png	https://me	480	640	0	62aea921l	acupoints	0	nguyenhoa	236	144	2	267	113	2	307	138	2	334	159	2	364	157
19	1016.png	https://me	480	640	0	62aea92ct	acupoints	0	nguyenhoa	188	139	2	215	97	2	260	111	2	286	127	2	312	119
20	1017.png	https://me	480	640	0	62aea939l	acupoints	0	nguyenhoa	177	138	2	200	98	2	244	111	2	267	121	2	296	114
21	1018.png	https://me	480	640	0	62aea945l	acupoints	0	nguyenhoa	169	146	2	183	103	2	228	107	2	254	117	2	286	108
22	1019.png	https://me	480	640	0	62aea94el	acupoints	0	nguyenhoa	159	155	2	171	107	2	214	112	2	239	122	2	268	107
23	102.png	https://me	480	640	0	62aea966l	acupoints	0	nguyenhoa	291	210	2	331	171	2	391	195	2	420	212	2	444	202
24	1020.png	https://me	480	640	0	62aea970l	acupoints	0	nguyenhoa	147	163	2	158	114	2	201	114	2	228	127	2	253	109

Hình 3.7 Tệp giá trị tọa độ dưới định dạng file csv

3.2.2 Tiền xử lý dữ liệu

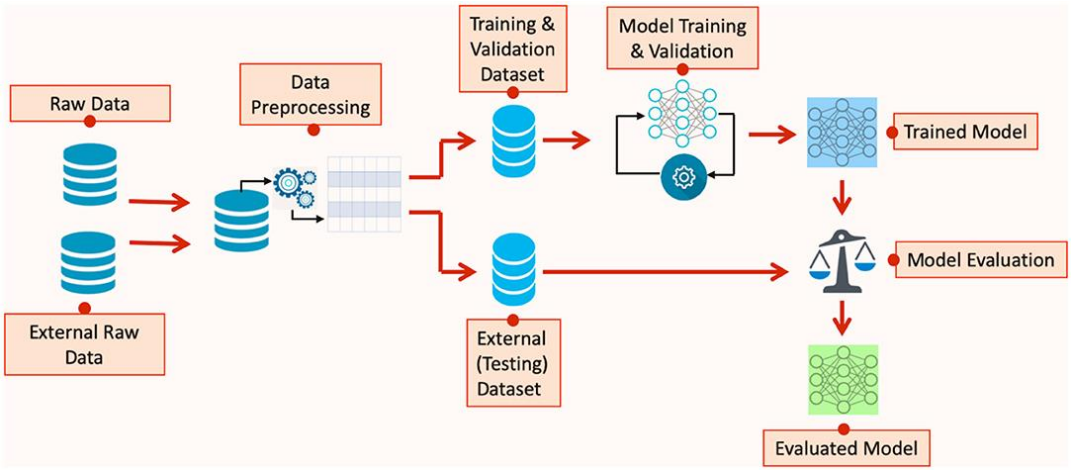
Dữ liệu trước khi đưa vào mô hình huấn luyện cần được xử lý, làm sạch và loại bỏ đi các thông tin không cần thiết. Ở đây, mục tiêu của em là giữ lại 3 thông tin quan trọng nhất:

- Tên bức ảnh
- Tọa độ x (10 điểm huyệt đạo)
- Tọa độ y (10 điểm huyệt đạo)

Để xử lý những dữ liệu quan trọng được đúng cách và nhanh chóng, ta sẽ áp dụng các hàm cơ bản phục vụ cho việc xử lý dữ liệu thuộc hai thư viện Pandas và Numpy để loại bỏ các cột dữ liệu không cần thiết và thay đổi một số thông tin theo mong muốn.

3.2.3 Quy trình sử dụng và phân chia tập dữ liệu

Ban đầu ta đã có dữ liệu ảnh và các điểm tọa độ đã được gắn nhãn tương ứng, sau đó ta tiến hành chia dữ liệu thành 3 tập: tập huấn luyện (training set), tập đánh giá (validation set) và tập kiểm tra (testing set). Khi tiến hành quá trình huấn luyện ta sẽ sử dụng tập huấn luyện để đưa vào mô hình và ở cuối mỗi bước huấn luyện ta sẽ đưa vào tập đánh giá để giám sát và đánh giá mô hình. Sau khi mô hình đã được huấn luyện thì ta dùng tập kiểm tra để ta có thể kiểm tra một cách chính xác nhất hiệu quả của mô hình. Sau cùng, ta sẽ đưa mô hình vào quy trình dự đoán thực tế tức là đầu vào là một tấm ảnh nằm ngoài 3 tập dữ liệu trên chúng không có nhãn, ta sẽ dựa vào mô hình và dự đoán ra nhãn mớ

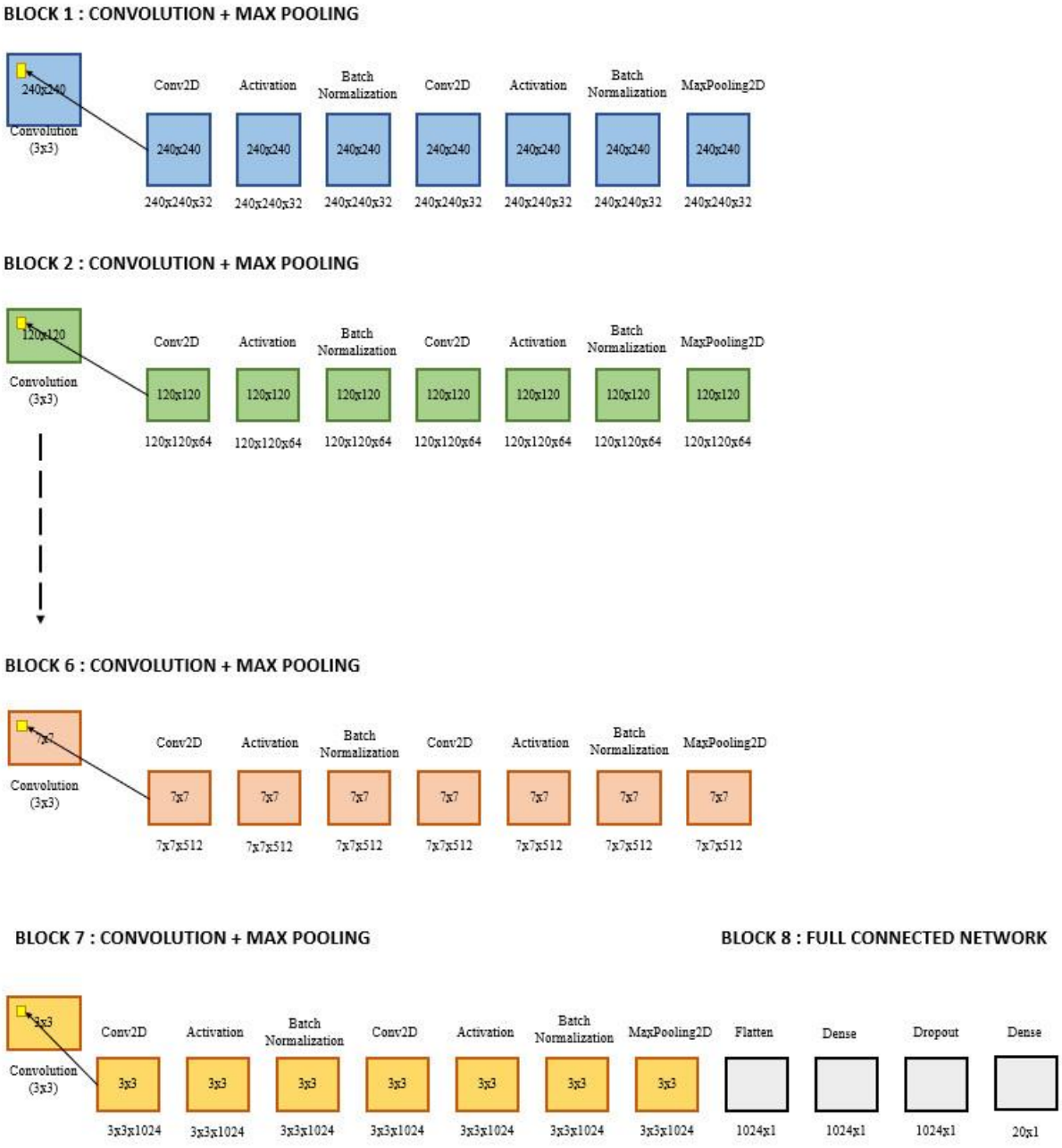


Hình 3.8 Quy trình xử lý dữ liệu

3.3 Triển khai xây dựng mô hình

3.3.1 Xây dựng mô hình

Mô hình CNN mà em xây dựng đã trải qua nhiều lần thử nghiệm, thay đổi các lớp và thông số để có thể tìm ra mô hình có chất lượng tốt nhất.



Hình 3.9 Mô hình CNN tự xây dựng

3.3.2 Trình bày sơ lược về mô hình

- Block 1, có thể thấy ảnh đầu vào với kích thước là 240x240x3 và ta sẽ sử dụng 32 kernel với kích thước là 3x3 để trượt qua từng ảnh gốc để làm nổi bật lên đặc trưng của bức ảnh, sử dụng "padding = same" để giữ lại kích thước gốc ban đầu của ảnh, với "use_bias: False" tức không sử dụng vector thiên vị. Lúc này kích thước ảnh trở thành 32x32 với 32 lớp màu 32x32x32. Sau đó, ta sẽ thêm một lớp Activation LeakyReLU (alpha = 0.1) là một biến thể nổi tiếng ReLU phi tuyến để có thể xấp xỉ được các phân bố phức tạp và hàm này sẽ không làm thay đổi kích thước ban đầu của ảnh đầu vào. Tiếp theo là lớp Batch Normization để phân bố của dữ liệu đồng nhất với dữ liệu đầu vào ban đầu vì qua các lớp như Convolution hay Activation thì phân bố của dữ liệu so với đầu vào là rất lớn. Tiếp tục thêm các lớp tương tự như trên để tăng độ phức tạp cho mô hình. Cuối cùng ta sẽ thêm 1 lớp MaxPooling2D dùng để chọn ra những đặc trưng nổi bật nhất làm tăng độ chính xác đồng thời cũng làm giảm kích thước ảnh.
- Block 2 ta sẽ làm tương tự như Block 1 tuy nhiên ta sẽ giảm kích thước ảnh xuống 120x120 nhưng số lượng lớp màu sẽ tăng lên thành 64.
- Block 3 ta sẽ làm tương tự như Block 2 tuy nhiên ta sẽ giảm kích thước ảnh xuống 60x60 nhưng số lượng lớp màu sẽ tăng lên thành 96.
- Block 4 ta sẽ làm tương tự như Block 3 tuy nhiên ta sẽ giảm kích thước ảnh xuống 30x30 nhưng số lượng lớp màu sẽ tăng lên thành 128.
- Block 5 ta sẽ làm tương tự như Block 4 tuy nhiên ta sẽ giảm kích thước ảnh xuống 15x15 nhưng số lượng lớp màu sẽ tăng lên thành 256.
- Block 6 ta sẽ làm tương tự như Block 5 tuy nhiên ta sẽ giảm kích thước ảnh xuống 7x7 nhưng số lượng lớp màu sẽ tăng lên thành 512.
- Block 7 ta sẽ làm tương tự như Block 6 tuy nhiên ta sẽ giảm kích thước ảnh xuống 3x3 nhưng số lượng lớp màu sẽ tăng lên thành 1024.
- Block 8, sau khi sử dụng các lớp tích chập để trích xuất được các đặc trưng của ảnh thì sẽ qua 1 lớp Flatten để trải phẳng ma trận 2 chiều thành vector 1 chiều trước khi

đưa vào lớp Dense. Tiếp theo là lớp Dropout, thông thường khi tất cả các đặc trưng được kết nối với lớp Dense, nó có thể gây ra quá mức trong bộ dữ liệu đào tạo. Quá tải xảy ra khi một mô hình cụ thể hoạt động rất tốt trên dữ liệu đào tạo gây ra tác động tiêu cực đến hiệu suất của mô hình khi được sử dụng trên dữ liệu mới. Để khắc phục vấn đề này, một lớp Dropout được cần được sử dụng. Cuối cùng, ta sẽ thu nhỏ số lớp sao cho phù hợp với số lượng cần dự đoán. Ở đây có 10 điểm huyệt đạo tức 20 giá trị đầu ra là 10 giá trị tọa độ x và 10 giá trị tọa độ y tương ứng.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 240, 240, 32)	864
leaky_re_lu (LeakyReLU)	(None, 240, 240, 32)	0
batch_normalization (BatchNo	(None, 240, 240, 32)	128
conv2d_1 (Conv2D)	(None, 240, 240, 32)	9216
leaky_re_lu_1 (LeakyReLU)	(None, 240, 240, 32)	0
batch_normalization_1 (Batch	(None, 240, 240, 32)	128
max_pooling2d (MaxPooling2D)	(None, 120, 120, 32)	0
conv2d_2 (Conv2D)	(None, 120, 120, 64)	18432
leaky_re_lu_2 (LeakyReLU)	(None, 120, 120, 64)	0
batch_normalization_2 (Batch	(None, 120, 120, 64)	256
conv2d_3 (Conv2D)	(None, 120, 120, 64)	36864
leaky_re_lu_3 (LeakyReLU)	(None, 120, 120, 64)	0
batch_normalization_3 (Batch	(None, 120, 120, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 60, 60, 64)	0
conv2d_4 (Conv2D)	(None, 60, 60, 96)	55296
leaky_re_lu_4 (LeakyReLU)	(None, 60, 60, 96)	0
batch_normalization_4 (Batch	(None, 60, 60, 96)	384
conv2d_5 (Conv2D)	(None, 60, 60, 96)	82944
leaky_re_lu_5 (LeakyReLU)	(None, 60, 60, 96)	0
batch_normalization_5 (Batch	(None, 60, 60, 96)	384
max_pooling2d_2 (MaxPooling2	(None, 30, 30, 96)	0
conv2d_6 (Conv2D)	(None, 30, 30, 128)	110592
leaky_re_lu_6 (LeakyReLU)	(None, 30, 30, 128)	0
batch_normalization_6 (Batch	(None, 30, 30, 128)	512
conv2d_7 (Conv2D)	(None, 30, 30, 128)	147456
leaky_re_lu_7 (LeakyReLU)	(None, 30, 30, 128)	0
batch_normalization_7 (Batch	(None, 30, 30, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 15, 15, 128)	0
conv2d_8 (Conv2D)	(None, 15, 15, 256)	294912
leaky_re_lu_8 (LeakyReLU)	(None, 15, 15, 256)	0
batch_normalization_8 (Batch	(None, 15, 15, 256)	1024
conv2d_9 (Conv2D)	(None, 15, 15, 256)	589824
leaky_re_lu_9 (LeakyReLU)	(None, 15, 15, 256)	0
batch_normalization_9 (Batch	(None, 15, 15, 256)	1024
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 256)	0
conv2d_10 (Conv2D)	(None, 7, 7, 512)	1179648
leaky_re_lu_10 (LeakyReLU)	(None, 7, 7, 512)	0
batch_normalization_10 (Batc	(None, 7, 7, 512)	2048
conv2d_11 (Conv2D)	(None, 7, 7, 512)	2359296
leaky_re_lu_11 (LeakyReLU)	(None, 7, 7, 512)	0
batch_normalization_11 (Batc	(None, 7, 7, 512)	2048
max_pooling2d_5 (MaxPooling2	(None, 3, 3, 512)	0
conv2d_12 (Conv2D)	(None, 3, 3, 1024)	4718592
leaky_re_lu_12 (LeakyReLU)	(None, 3, 3, 1024)	0
batch_normalization_12 (Batc	(None, 3, 3, 1024)	4096
conv2d_13 (Conv2D)	(None, 3, 3, 1024)	9437184
leaky_re_lu_13 (LeakyReLU)	(None, 3, 3, 1024)	0
batch_normalization_13 (Batc	(None, 3, 3, 1024)	4096
max_pooling2d_6 (MaxPooling2	(None, 1, 1, 1024)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 1024)	1049600
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 20)	20500

Total params: 20,128,116
Trainable params: 20,119,668
Non-trainable params: 8,448

Hình 3.10 Bảng tóm tắt mô hình CNN tự xây dựng

3.3.3 Xây dựng các hàm tối ưu

- Optimizer = Adamax: Là một biến thể của Adam dựa trên tiêu chuẩn vô cực. Adamax đôi khi vượt trội hơn adam, đặc biệt là trong các mô hình có nhúng.
- Batch_size = 4: Số lượng dữ liệu trong 1 lần lấy ra để học là 4.
- Epochs = 500: Số lần huấn luyện là 500 lần.
- Loss = mean_squared_error: hàm mất mát này được sử dụng phổ biến trong các bài toán nhận diện.
- Metrics = accuracy: Thông số accuracy dùng để đánh giá mô hình.

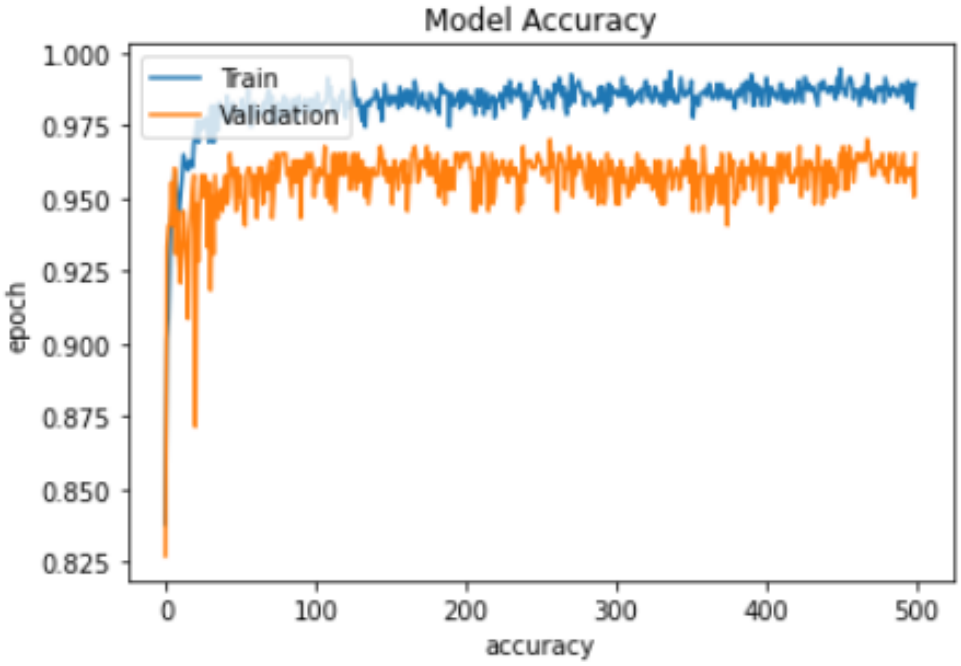
3.4 Tiến hành đào tạo mô hình

```
237/237 [==============================] - 17s 36ms/step - loss: 394.8054 - accuracy: 0.
8381 - val_loss: 1726.0714 - val_accuracy: 0.8272
Epoch 2/500
237/237 [==============================] - 7s 31ms/step - loss: 74.5506 - accuracy: 0.90
16 - val_loss: 93.8279 - val_accuracy: 0.9333
Epoch 3/500
237/237 [==============================] - 7s 31ms/step - loss: 59.5286 - accuracy: 0.90
90 - val_loss: 64.5292 - val_accuracy: 0.9407
Epoch 4/500
237/237 [==============================] - 7s 31ms/step - loss: 54.7616 - accuracy: 0.93
12 - val_loss: 40.3084 - val_accuracy: 0.9407
Epoch 5/500
237/237 [==============================] - 7s 31ms/step - loss: 49.8372 - accuracy: 0.94
39 - val_loss: 45.0108 - val_accuracy: 0.9556
Epoch 6/500
237/237 [==============================] - 7s 31ms/step - loss: 50.0604 - accuracy: 0.94
60 - val_loss: 36.0001 - val_accuracy: 0.9407
Epoch 7/500
237/237 [==============================] - 7s 31ms/step - loss: 44.0149 - accuracy: 0.95
87 - val_loss: 32.5023 - val_accuracy: 0.9605
Epoch 8/500
237/237 [==============================] - 7s 31ms/step - loss: 44.1334 - accuracy: 0.95
56 - val_loss: 42.9266 - val_accuracy: 0.9309
Epoch 9/500
237/237 [==============================] - 7s 31ms/step - loss: 39.1963 - accuracy: 0.95
03 - val_loss: 42.9298 - val_accuracy: 0.9457
Epoch 10/500
```

```
Epoch 491/500
237/237 [==============================] - 7s 32ms/step - loss: 6.1210 - accuracy: 0.989
4 - val_loss: 31.7072 - val_accuracy: 0.9630
Epoch 492/500
237/237 [==============================] - 7s 31ms/step - loss: 6.7350 - accuracy: 0.986
2 - val_loss: 28.8672 - val_accuracy: 0.9654
Epoch 493/500
237/237 [==============================] - 7s 31ms/step - loss: 7.2743 - accuracy: 0.988
4 - val_loss: 22.9244 - val_accuracy: 0.9556
Epoch 494/500
237/237 [==============================] - 7s 30ms/step - loss: 5.9459 - accuracy: 0.983
1 - val_loss: 27.2181 - val_accuracy: 0.9605
Epoch 495/500
237/237 [==============================] - 7s 31ms/step - loss: 6.1287 - accuracy: 0.990
5 - val_loss: 18.8580 - val_accuracy: 0.9580
Epoch 496/500
237/237 [==============================] - 8s 32ms/step - loss: 5.7247 - accuracy: 0.985
2 - val_loss: 33.7961 - val_accuracy: 0.9580
Epoch 497/500
237/237 [==============================] - 7s 31ms/step - loss: 6.4331 - accuracy: 0.989
4 - val_loss: 21.7142 - val_accuracy: 0.9605
Epoch 498/500
237/237 [==============================] - 7s 31ms/step - loss: 5.8672 - accuracy: 0.981
0 - val_loss: 25.1033 - val_accuracy: 0.9605
Epoch 499/500
237/237 [==============================] - 7s 31ms/step - loss: 7.2161 - accuracy: 0.989
4 - val_loss: 23.5442 - val_accuracy: 0.9506
Epoch 500/500
237/237 [==============================] - 7s 30ms/step - loss: 6.2760 - accuracy: 0.989
4 - val_loss: 28.3248 - val_accuracy: 0.9654
```

Hình 3.11 Quá trình đào tạo mô hình

3.5 Đánh giá mô hình



Hình 3.12 Biểu đồ đánh giá độ chính xác của mô hình

Sai số của tập dữ liệu huấn luyện là với 5.7247 độ chính xác là 0.9905.

Sai số của tập dữ liệu kiểm chứng là 33.7961 với độ chính xác là 0.9580.

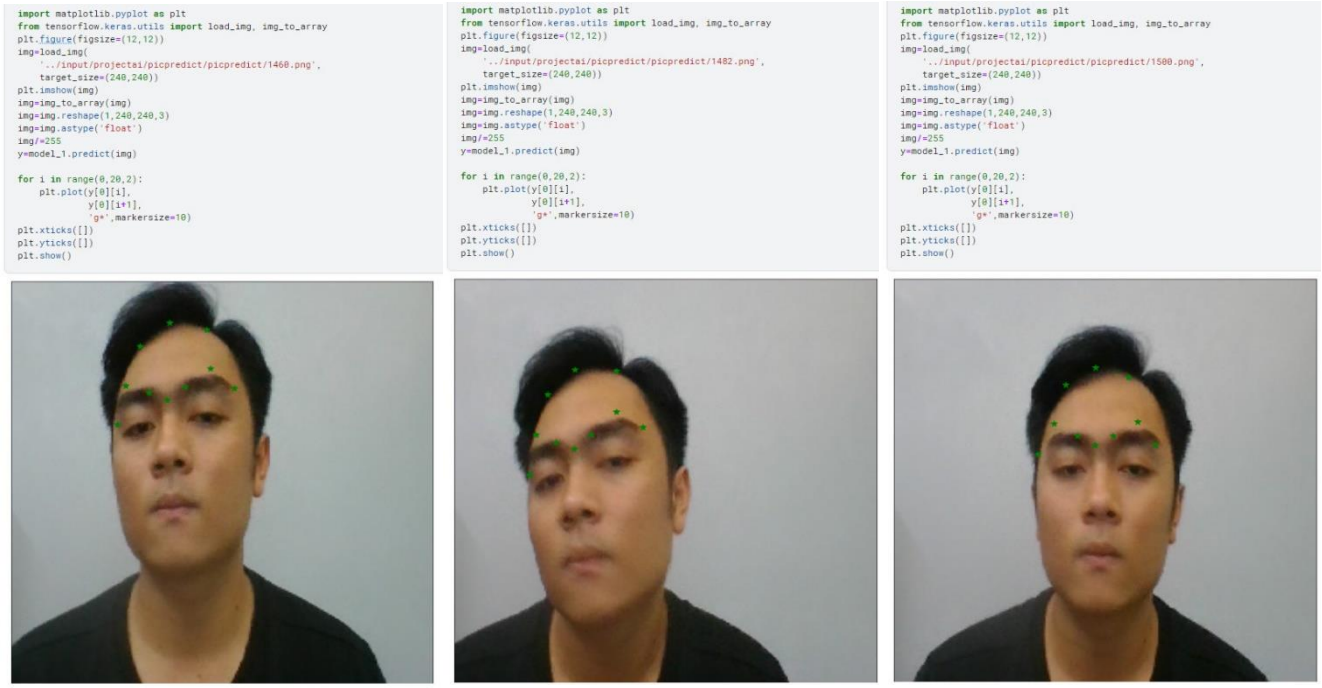
Có thể thấy sai số của tập dữ liệu huấn luyện và dữ liệu kiểm chứng là khá cao. Nhưng độ chính xác của cả hai tập đều trên 90%. Với sai số và độ chính xác như trên mô hình sẽ có khả năng dự đoán đúng trên dưới 75%.

	Huấn luyện	**Kiểm chứng**
Độ chính xác	99.05	95.8

3.6 Kiểm tra hoạt động của mô hình trên tập dữ liệu kiểm tra

Hình 3.13 Dự đoán điểm huyệt đạo trên tập dữ liệu kiểm tra

3.7 Kiểm tra hoạt động của mô hình trên tập dữ liệu thực tế



Hình 3.14 Dự đoán điểm huyệt đạo trên tập dữ liệu thực tế

3.8 Nhận xét và đánh giá mô hình

Tỷ lệ xác định tọa độ các điểm huyệt đạo của mô hình với tập dữ liệu kiểm tra là khá chính xác lên đến 80-90%. Còn tỷ lệ xác định đúng tọa độ các điểm huyệt đạo của mô hình với dữ liệu hình ảnh thực tế bên ngoài là tương đối chính xác khoảng 70-80%. Trừ trường hợp các góc mặt khó, hình ảnh mờ và vị trí góc mặt với số lượng dữ liệu hình ảnh ít dẫn đến việc dự đoán chưa tối ưu và chính xác.

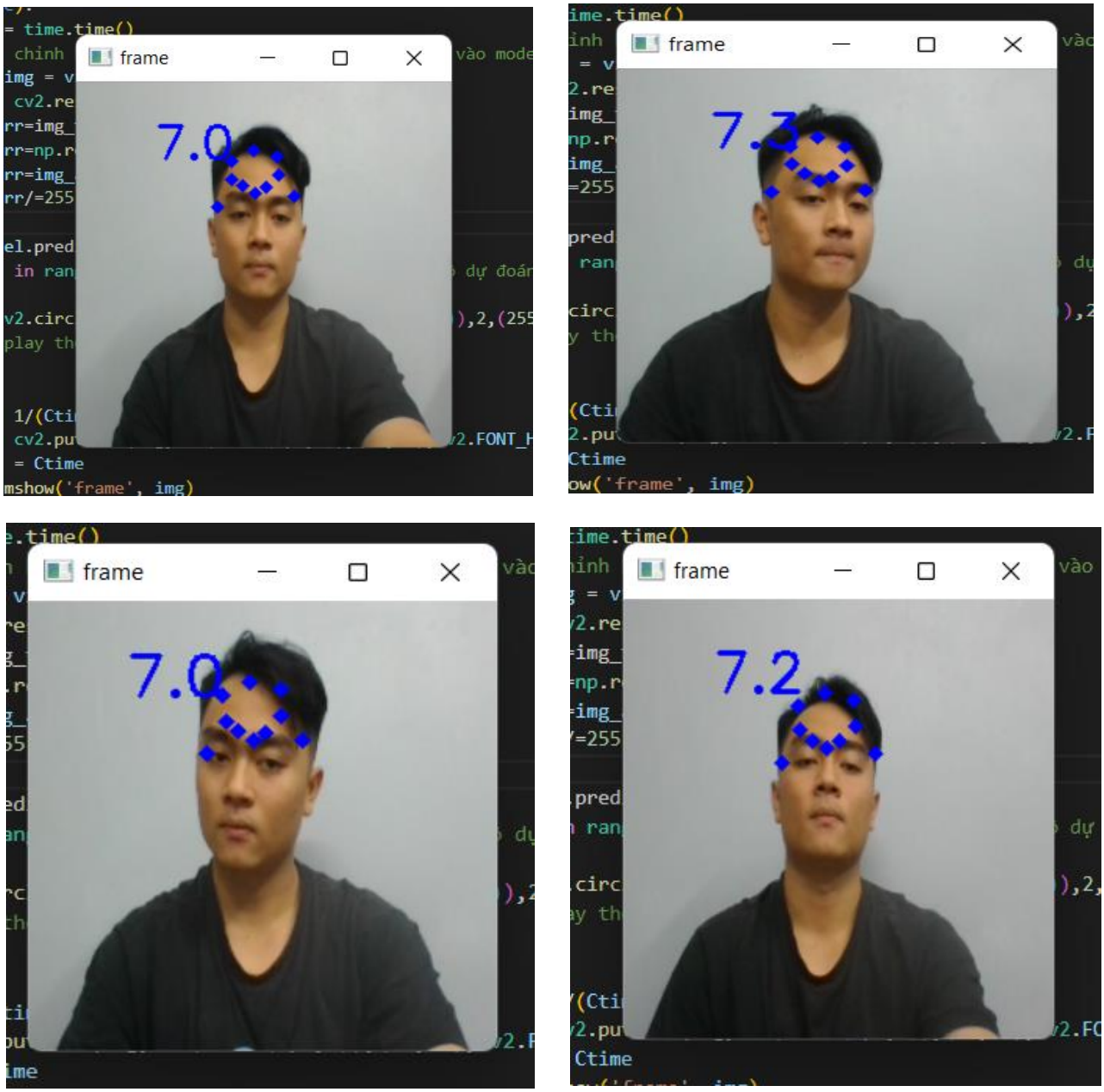
Nhìn chung, mô hình đưa ra kết quả dự đoán là tương đối ổn, chưa có sự chính xác tuyệt đối. Tuy nhiên, với số lượng dữ liệu đầu vào là 1350 tấm ảnh nên khó có thể đòi hỏi mô hình dự đoán với độ chính xác cao. Cần bổ sung thêm nhiều dữ liệu hơn với nhiều góc chụp khác nhau của khuôn mặt để mô hình có thể cải thiện khả năng nhận diện tốt hơn. Đồng thời, nên bổ sung dữ liệu hình ảnh của nhiều người hơn nữa để việc nhận diện huyệt đạo trở nên đa dạng hơn.

Có thể thấy mô hình xây dựng chưa thực sự tốt, ngoài việc bổ sung dữ liệu cũng cần cải thiện các thông số, bổ sung các hàm tối ưu, tối ưu hóa các lớp nơ ron và các thành phần bên trong nhằm cải thiện khả năng dự đoán của mô hình.

PHẦN 4: MÔ PHỎNG THỜI GIAN THỰC VÀ GIAO DIỆN

4.1 Mô phỏng thời gian thực

Chương trình sử dụng thư viện opencv để đọc hình ảnh liên tục từ webcam đồng thời đưa vào mô hình đã được huấn luyện để xử lý và cho ra các giá trị tọa độ điểm huyệt đạo sau đó vẽ các đó điểm lên. Tất cả quá trình từ lúc đọc ảnh tới phát hiện và nhận diện điểm đạt khoảng 7 fps.



Hình 4.1 Mô phỏng thời gian thực

- Nhận xét: Nhìn chung, mô hình nhận diện các điểm huyệt đạo chưa thật sự tốt.
- Đánh giá: Mô hình ở thời gian thực nhận diện điểm huyệt đạo còn khá tệ một phần do nhiều yếu tố như nhiễu ánh sáng, độ phân giải của camera máy tính và phần cốt yếu là do lượng dữ liệu huấn luyện còn quá ít, bên cạnh đó mô hình tự xây dựng vẫn chưa thực sự tố. Có thể thấy tỷ lệ nhận diện chính xác nằm ở khoảng 30-40%.

PHẦN 5: KẾT LUẬN

5.1 Công việc đã hoàn thành

Hoàn thành việc tìm hiểu các tài liệu về cơ sở lý thuyết, các kiến thức trí tuệ nhân tạo có liên quan. Áp dụng lý thuyết đã học trên lớp và xây dựng mô hình xác định các huyệt đạo trên đầu người. Mô hình nhận biết được tương đối các huyệt đạo đã được huấn luyên. Giải quyết được các vấn đề khó khăn phát sinh trong quá trình hoàn thành tiểu luận của mình. Thực hiện được sử dụng thời gian thực.

5.2 Các hạn chế và giải pháp

Hạn chế:

- Mô hình vẫn chưa thực sự đạt được độ chính xác cao do lượng dữ liệu còn quá ít.
- Độ chính xác chưa cao có thể do mô hình xây dựng chưa được tối ưu.
- Còn hạn chế trong việc chỉ sử dụng một người để chụp ảnh làm dữ liệu nhận diện huyệt đạo.
- Còn tham khảo từ nhiều nguồn tài liệu khác nhau.
- Nhận diện các điểm huyệt đạo bằng thời gian thực độ chính xác còn thấp
- Một số công việc trong quá trình làm tiểu luận còn bị chậm tiến độ dẫn đến sẽ có sai sót trong quá trình tính toán.

Giải pháp:

- Nghiên cứu nhiều hơn, đọc nhiều tài liệu hơn về Trí tuệ nhân tạo để tích lũy thêm kiến thức
- Tính toán và xây dựng lại mô hình mạng nơron tốt nhất có thể để tối ưu và cải thiện độ chính xác trong việc xác định các huyệt đạo.
- Siêng năng, tích cực hơn, sắp xếp thời gian biểu một cách hợp lý và cố gắng hoàn thành công việc sớm nhất có thể để có nhiều thời gian cải thiện.

5.3 Cải tiến và hướng phát triển đề tài

Hướng cải tiến:

- Cải thiện chất lượng mô hình bằng cách xây dựng lại mạng nơron phức tạp, dày đặt hơn để mô hình trở nên hoàn thiện.
- Điều chỉnh các thông số, các hàm tối ưu phù hợp và tốt hơn để qua đó độ chính xác của mô hình được cải thiên hơn.
- Tăng số lương dữ liệu từ 1500 tấm ảnh lên 5000 tấm ảnh và đa dạng hóa số lượng người chụp để làm dữ liệu.
- Cải thiện việc áp dụng thời gian thực vào mô hình dự đoán chính xác hơn.
- Phát triển thêm giao diện cho người sử dụng.

Hướng phát triển:

- Mở rộng thêm tập dữ liệu bằng việc tăng số lượng ảnh đầu vào và đồng thời đa dạng hóa hình ảnh với nhiều người khác nhau mà không chỉ riêng em.
- Tăng thêm số lượng huyệt đạo để xác định với nhiều góc mặt và vị trí khác nhau trên đầu.
- Nếu bài toán xác định huyệt đạo trên đầu đã được tối ưu nhất có thể em sẽ mở rộng sang bài toán xác định các huyệt đạo ở vị trí khác trên cơ thể.

5.4 Kết lời

Cuối cùng, em xin chân thành cảm ơn thầy đã cho bản than em một đề tài rất hay và thực tế để đánh giá chất lượng học tập của bản thân cũng như tiếp thu thêm nhiều kiến thức, kỹ năng làm việc, sắp xếp thời gian và lên kết hoạch. Một lần nữa em xin chân thành cảm ơn thầy.

TÀI LIỆU THAM KHẢO

[1] Vũ Hữu Tiệp: Machine Learning cơ bản

[2] Nguyễn Thanh Tuấn, Deep Learning Cơ Bản.

[3] Chris Albon. 2018, Python Machine Learning Cookbook

[4] Wes McKinney. 2017, Python for Data Analysis

[6] Alberto Artasanchez, Prateek Joshi, Arificial Intelligence with Python.

[7] TopDev, Thuật toán CNN – Convolution Neural Network.

(LINK : https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/)

[8] TAXPLUS, Trí tuệ nhân tạo AI là gì. Ứng dụng trong cuộc sống.

(LINK: https://taxplus.vn/tri-tue-nhan-tao-ai-la-gi/)

[9] TPC, Các huyệt đạo trên cơ thể người theo Y học cổ truyền.

(LINK: https://tnpc.edu.vn/cac-huyet-dao-tren-co-the-nguoi-theo-y-hoc-co-truyen.html)

[10] ITNavi, Deep Learning là gì? Khái quát kiến thức về Deep Learning.

(LINK: https://itnavi.com.vn/blog/deep-learning-la-gi)

PHỤ LỤC CODE

CODE CỦA CHƯƠNG TRÌNH CHÍNH

- **Tải tập dữ liệu train và test đã được chia trước**

```
import numpy as np
import pandas as pd
train_df = pd.read_csv('../input/projectai/train.csv')
test_df = pd.read_csv('../input/projectai/test.csv')
train_df.info()
```

- **Lọc và loại bỏ các dữ liệu không cần thiết trong tập train và test**

+ Tập train

```
keypoints_df = train_df.drop(train_df.columns[0:9],axis = 1)
keypoints_df = keypoints_df.drop(keypoints_df.columns[2::3],axis = 1)
y_train_size=train_df.columns[2:4]
y_train = np.array(keypoints_df,dtype='float')
y_train_size=train_df[['height','width']]
y_train_size=np.array(y_train_size,dtype='float')
for i in range(0,y_train.shape[1],2):
    for j in range(y_train.shape[0]):
        y_train[j][i]=y_train[j][i]/y_train_size[j][1]*240
        y_train[j][i+1]=y_train[j][i+1]/y_train_size[j][0]*240
print(y_train.shape)
```

+ Tập test

```
keytest_df = test_df.drop(test_df.columns[0:9],axis = 1)
keytest_df = keytest_df.drop(keytest_df.columns[2::3],axis = 1)
y_test = np.array(keytest_df,dtype='float')
y_test_size=test_df[['height','width']]
y_test_size=np.array(y_test_size,dtype='float')
for i in range(0,y_test.shape[1],2):
    for j in range(y_test.shape[0]):
        y_test[j][i]=y_test[j][i]/y_test_size[j][1]*240
        y_test[j][i+1]=y_test[j][i+1]/y_test_size[j][0]*240
print(y_test.shape)
```

- **Resize và chia tập dữ liệu hình ảnh**

```
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import load_img, img_to_array
import numpy as np
datagen = ImageDataGenerator(rescale=1./255)
data1=datagen.flow_from_directory('../input/projectai/DataProject_unlabell
ed_data_0/datasets',
```

```
                                          target_size=(240,240),batch_size=5,
                                        interpolation="lanczos",shuffle=False)
imageArr = []
for i in range(945):
        img=load_img('../input/projectai/DataProject_unlabelled_data_0/dat
asets/'+data1.filenames[i],
                     target_size=(240,240))
        img=img_to_array(img)
        img=img.reshape(1,240,240,3)
        img=img.astype('float')
        img/=255
        imageArr.append(img)
x_train = np.array(imageArr,dtype='float')
x_train = x_train.reshape(-1,240,240,3)
imageArr_test = []
for i in range(945,1350):
    img1=load_img('../input/projectai/DataProject_unlabelled_data_0/datase
ts/'+data1.filenames[i],
                target_size=(240,240))
    img1=img_to_array(img1)
    img1=img1.reshape(1,240,240,3)
    img1=img1.astype('float')
    img1/=255
    imageArr_test.append(img1)
x_test = np.array(imageArr_test,dtype='float')
x_test = x_test.reshape(-1,240,240,3)
```

- **Trực quan hóa điểm đã gắn nhãn trên dữ liệu hình ảnh của tập train và test**

+ Tập train

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
a=1
img=load_img('../input/projectai/DataProject_unlabelled_data_0/datasets/'+
data1.filenames[a],
                target_size=(240,240))
plt.imshow(img)
for i in range(0,20,2):
    plt.plot(y_train[a][i],
                y_train[a][i+1],
                'b*',markersize=6)
plt.xticks([])
plt.yticks([])
plt.show()
```

+ Tập test

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
```

```
a=20
img=load_img('../input/projectai/DataProject_unlabelled_data_0/datasets/'+
data1.filenames[a+945],
             target_size=(240,240))
plt.imshow(img)
for i in range(0,20,2):
    plt.plot(y_test[a][i],
             y_test[a][i+1],
             'r*',markersize=6)
plt.xticks([])
plt.yticks([])
plt.show()
```

- **Xây dựng mô hình**

```
from keras.models import Sequential, Model
from keras.layers import Activation, Convolution2D,MaxPooling2D,BatchNorma
lization, Flatten, Dense, Dropout,GlobalAveragePooling2D
from keras.layers.advanced_activations import LeakyReLU
from tensorflow.keras.optimizers import RMSprop
model = Sequential()
model.add(Convolution2D(32,(3,3),padding='same',use_bias=False, input_shap
e=(240,240,3)))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())

model.add(Convolution2D(32,(3,3),padding='same',use_bias = False))
model.add(LeakyReLU(alpha=0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Convolution2D(64,(3,3),padding='same',use_bias = False))
model.add(LeakyReLU(alpha=0.1))
model.add(BatchNormalization())
model.add(Convolution2D(64, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Convolution2D(96, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())

model.add(Convolution2D(96, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(128, (3,3),padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
```

```
model.add(Convolution2D(128, (3,3),padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(256, (3,3),padding='same',use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(Convolution2D(256, (3,3),padding='same',use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(512, (3,3),padding='same',use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())

model.add(Convolution2D(512, (3,3),padding='same',use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(1024, (3,3),padding='same',use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())

model.add(Convolution2D(1024, (3,3),padding='same',use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(20))
model.compile(optimizer='Adamax',loss='mean_squared_error',
metrics=['accuracy'])
model.summary()
```

- **Tiến hành huấn luyện mô hình**

```
history=model.fit(x_train,y_train,batch_size=4,epochs=500,validation_data=
(x_test,y_test))
```

- **Vẽ biểu đồ đánh giá độ chính xác của mô hình**

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

```
plt.title('Model Accuracy')
plt.xlabel('accuracy')
plt.ylabel('epoch')
plt.legend(['Train','Validation'],loc='upper left')
plt.show()
```

- **Dự đoán và tiến hành trực quan hóa điểm dữ liệu trên ảnh**

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
img=load_img('../input/projectai/picpredict/picpredict/1462.png',target_si
ze=(240,240))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,240,240,3)
img=img.astype('float')
img/=255
y=model.predict(img)

for i in range(0,20,2):
    plt.plot(y[0][i],
             y[0][i+1],
             'y*',markersize=8)
plt.xticks([])
plt.yticks([])
plt.show()
```

CODE CỦA CHƯƠNG TRÌNH MÔ PHỎNG THỜI GIAN THỰC

```
import cv2
import numpy as np
import tensorflow
import keras
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import load_img, img_to_array
from tensorflow.keras.models import load_model # run thu m
from numpy import reshape
import time
vid = cv2.VideoCapture(0)
model=load_model('ProjectAI.h5')
Ptime  = 0
while(True):
    Ctime= time.time()
    ret, img = vid.read()
    img = cv2.resize(img, (240,240))
    img_arr=img_to_array(img)
    img_arr=np.reshape(img_arr,(1,240,240,3))
    img_arr=img_arr.astype('float')
    img_arr/=255
```

```
    y=model.predict(img_arr)

    for i in range(0,20,2):
        cv2.circle(img, (int(y[0][i]),round(y[0][i+1])),2,(255,0,0),3)
    fps = 1/(Ctime -  Ptime)
    img = cv2.putText(img,str(round(fps,1)),(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0
,0),2 )
    Ptime = Ctime
    cv2.imshow('frame', img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        print(img_arr.shape)
        break
vid.release()
cv2.destroyAllWindows()
```