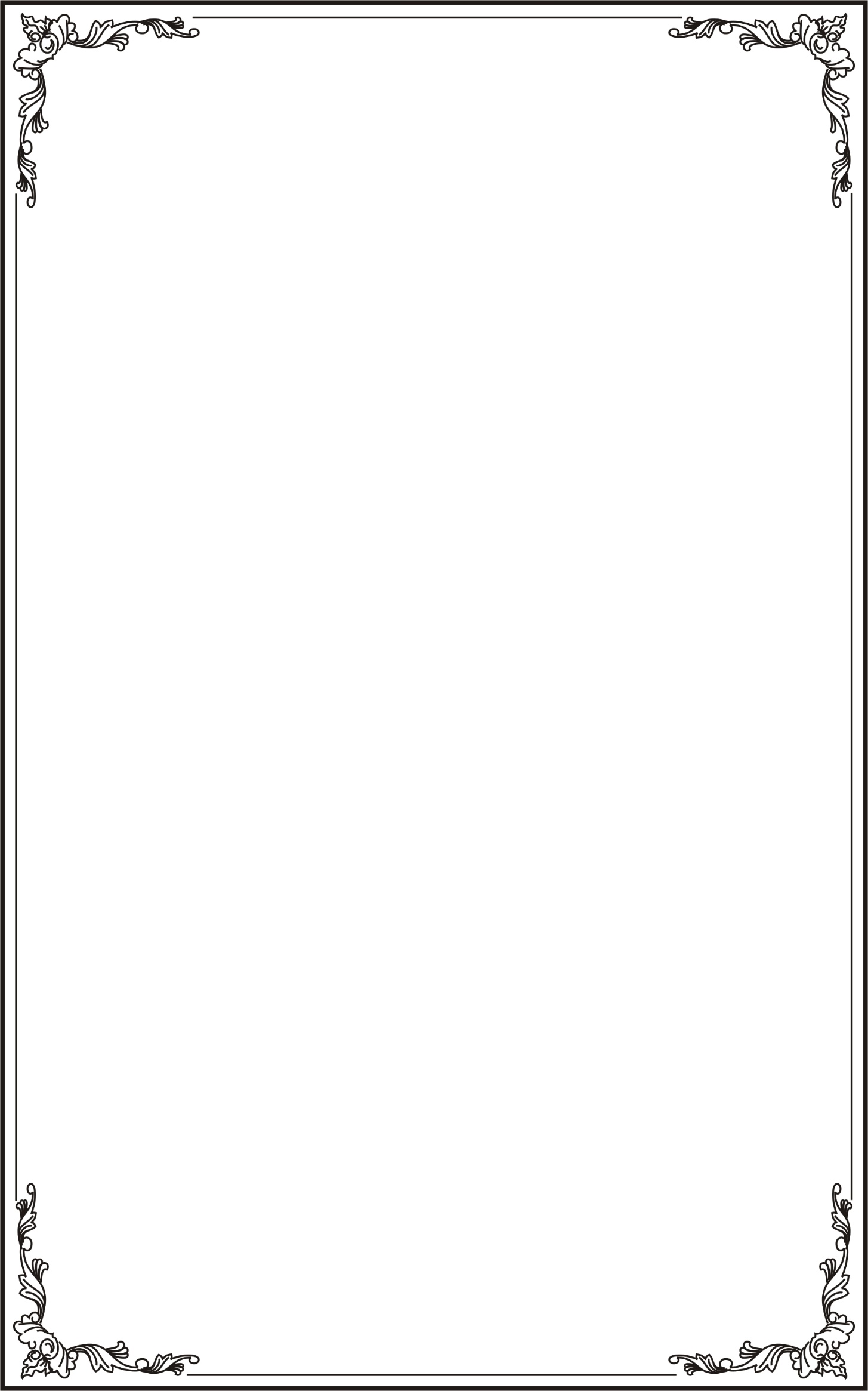
****

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**====** **❁** **❁** **❁====**

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI MÔN MẠNG NƠ RON NHÂN TẠO**

**“Nghiên cứu, xây dựng hệ thống nhận dạng**

**tiền Việt Nam”**

Sinh viên thực hiện :

Nguyễn Quang Huy - 2018605221

Mạc Thành Sơn - 2018601622

Phan Viết Hùng - 2018603428

Phạm Văn Đức - 2018603392

Trịnh Xuân Toàn- 2018604628

MỤC LỤC

[LỜI NÓI ĐẦU 4](#_Toc72366428)

[Tổng quan về đề tài 4](#_Toc72366429)

[Mục tiêu nghiên cứu đề tài 4](#_Toc72366430)

[Nội dung nghiên cứu 4](#_Toc72366431)

[Kết quả đạt được 4](#_Toc72366432)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 5](#_Toc72366433)

[1.1. Mạng nơ-ron nhân tạo 5](#_Toc72366434)

[1.1.1. Khái niệm mô hình mạng nơ-ron nhân tạo 5](#_Toc72366435)

[1.1.2. Các mô hình mạng nơ-ron 5](#_Toc72366436)

[1.1.3. Huấn luyện và ứng dụng mạng nơ-ron 7](#_Toc72366437)

[1.1.4. Ngôn ngữ thực thi mạng nơ-ron 7](#_Toc72366438)

[1.2. Mạng nơ-ron tích chập 8](#_Toc72366439)

[1.2.1. Khái niệm mạng nơ-ron tích chập 8](#_Toc72366440)

[1.2.1.1. Hình ảnh đầu vào (input) 9](#_Toc72366441)

[1.2.1.2. Bộ lọc (kernel) 10](#_Toc72366442)

[1.2.1.3. Padding 12](#_Toc72366443)

[1.2.1.4. Bước nhảy 13](#_Toc72366444)

[1.3. Mạng VGG 16 14](#_Toc72366445)

[CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG NHẬN DIỆN TIỀN VIỆT NAM 16](#_Toc72366446)

[2.1. Mô tả và phân tích bài toán 16](#_Toc72366447)

[2.2. Các giai đoạn 16](#_Toc72366448)

[2.2.1.Tạo dữ liệu 16](#_Toc72366449)

[2.2.2. Xử lý dữ liệu: 17](#_Toc72366450)

[2.2.3. Thiết kế mạng CNN Classify dùng để train: 17](#_Toc72366451)

[2.2.4. Train model CNN Classify 18](#_Toc72366452)

[2.2.5. Thử nghiệm model CNN Classify 18](#_Toc72366453)

[2.2.6. Mô tả quá trình trong hệ thống nhận dạng. 18](#_Toc72366454)

[2.3. Thiết kế mô hình mạng nơ ron nhận dạng tiền việt nam đồng 19](#_Toc72366455)

[2.3.1. Cấu trúc mô hình VGG sẽ sử dụng trong đề tài 19](#_Toc72366456)

[2.3.2. Thực thi mô hình mạng nơ ron nhận dạng tiền Việt Nam Đồng 20](#_Toc72366457)

[2.3.2.1. Chuẩn bị dữ liệu 20](#_Toc72366458)

[2.3.2.2. Huấn luyện 22](#_Toc72366459)

[2.3.2.3. Kết quả 24](#_Toc72366460)

[2.4. Xây dựng giao diện 25](#_Toc72366461)

[2.4.1. Cách sử dụng giao diện 25](#_Toc72366462)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 26](#_Toc72366463)

[TÀI LIỆU THAM KHẢO 26](#_Toc72366464)

# LỜI NÓI ĐẦU

## Tổng quan về đề tài

Nhận dạng là bài toán xuất hiện cách đây khá lâu và vẫn luôn thu hút được nhiều sự quan tâm, nghiên cứu. Đặc biệt là trong vài thập niên gần đây, do sự thúc đẩy của quá trình tin học hoá trong mọi lĩnh vực, bài toán nhận dạng không còn dừng lại ở mức độ nghiên cứu nữa mà nó trở thành một lĩnh vực để áp dụng vào thực tế. Các bài toán nhận dạng đang được ứng dụng trong thực tế hiện nay tập trung vào nhận dạng mẫu, nhận dạng tiếng nói và nhận dạng chữ, nhận dạng tiền mặt, … Trong số này, nhận dạng tiền mặt là bài toán được quan tâm rất nhiều và cũng đã đạt được nhiều thành tựu rực rỡ. Các ứng dụng có ý nghĩa thực tế lớn có thể kể đến như: nhận dạng các loại tiền mặt dùng trong quá trình mua nước ở máy bán nước tự động, …

Xuất phát từ yêu cầu thực tế, đang rất cần có nhưng nghiên cứu về vấn đề này. Chính vì vậy chúng tôi đã chọn đề tài nhận dạng tiền mặt làm đề tài nghiên cứu với mong muốn phần nào áp dụng vào bài toán thực tế.

## Mục tiêu nghiên cứu đề tài

* Nghiên cứu giải pháp nhận dạng tiền polime Việt Nam dựa trên mạng nơ-ron nhân tạo.
* Xây dựng ứng dụng nhận dạng tiền polime Việt Nam.

## Nội dung nghiên cứu

* Tìm hiểu về mạng nơ ron nhân tạo;
* Nghiên cứu các giải pháp nhận dạng hình ảnh và xử lý ảnh dùng mạng nơ-ron;
* Xây dựng ứng dụng nhận dạng tiền polime Việt Nam trên mạng nơ-ron nhân tạo.

## Kết quả đạt được

* Nhận dạng được tiền polime Việt Nam dựa trên mạng nơ-ron nhân tạo.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1.1. Mạng nơ-ron nhân tạo

### 1.1.1. Khái niệm mô hình mạng nơ-ron nhân tạo

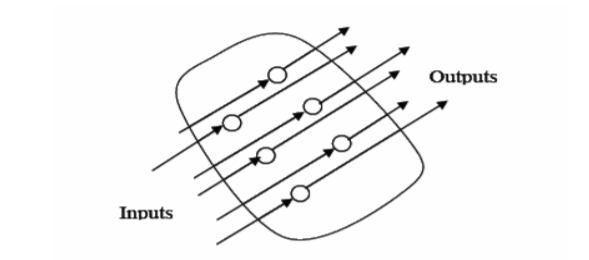
**Mạng Nơron nhân tạo (Artificial Neural Network- ANN) là mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thống thần kinh của sinh vật, bao gồm số lượng lớn các Nơron được gắn kết để xử lý thông tin. ANN giống như bộ não con người, được học bởi kinh nghiệm (thông qua huấn luyện), có khả năng lưu giữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết (unseen data).**

Mạng nơ ron nhân tạo là một chuỗi các thuật toán được sử dụng để tìm ra mối quan hệ của một tập dữ liệu thông qua cơ chế vận hành của bộ não sinh học. Mạng nơ ron nhân tạo thường được huấn luyện qua một tập dữ liệu chuẩn cho trước, từ đó có thể đúc rút được kiến thức từ tập dữ liệu huấn luyện, và áp dụng với các tập dữ liệu khác với độ chính xác cao.

Có hai kiểu mạng nơ ron chính: mạng nơ ron truyền thẳng (feedforward neural network) và mạng nơ ron hồi quy (recurrent neural network).

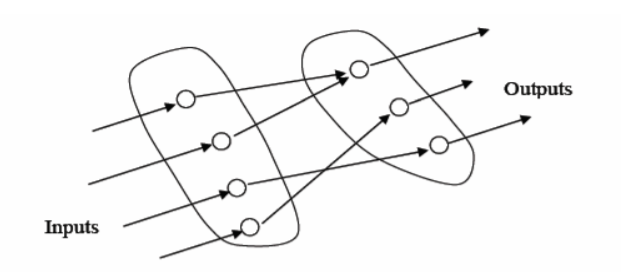
### 1.1.2. Các mô hình mạng nơ-ron

* Tự kết hợp (autoassociative): là mạng có các nơ ron đầu vào cũng là các nơ ron đầu ra.



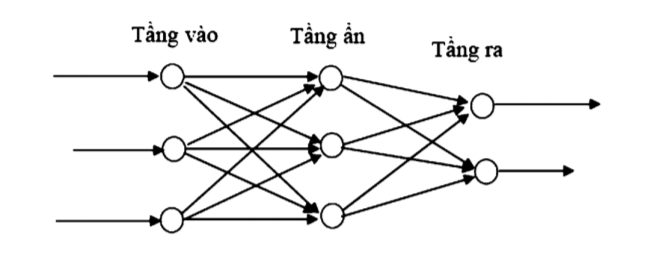
Hình 1. 1 Mô hình mạng nơ ron tự kết hợp

* Kết hợp khác kiểu (heteroassociative): là mạng có tập nơ ron đầu vào và đầu ra riêng biệt. Perceptron, các mạng Perceptron nhiều tầng (MLP: MultiLayer Perceptron) thuộc loại này.



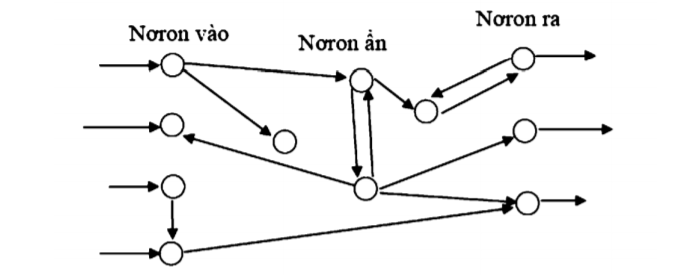
Hình 1. 2 Mô hình mạng nơ ron kết hợp các kiểu

* Kiến trúc truyền thẳng (feedforward architechture): là kiểu kiến trúc mạng không có các kết nối ngược trở lại từ các nơron đầu ra về các nơron đầu vào, mạng không lưu lại các giá trị output trước và các trạng thái kích hoạt của nơron. Các mạng nơron truyền thẳng cho phép tín hiệu di chuyển theo một đường duy nhất, từ đầu vào tới đầu ra, đầu ra của một tầng bất kì sẽ không ảnh hưởng tới tầng đó



Hình 1. 3 Mô hình mạng nơ ron truyền thẳng

* Kiến trúc phản hồi (Feedback architecture): là kiểu kiến trúc mạng có các kết nối từ nơron đầu ra tới nơron đầu vào. Mạng lưu lại các trạng thái trước đó, và trạng thái tiếp theo không chỉ phụ thuộc vào các tín hiệu đầu vào mà còn phụ thuộc vào các trạng thái trước đó của mạng.



Hình 1. 4 Mô hình mạng nơ ron phản hồi

### 1.1.3. Huấn luyện và ứng dụng mạng nơ-ron

Mạng nơ-ron nhận tạođã được sử dụng để giải nhiều bài toán trong hầu hết mọi lĩnh vực của khoa học và công nghệ. Huấn luyện mạng nơ-ron nhân tạo cơ bản bao gồm hai bước:

* Nhận dạng hệ thống
* Huấn luyện

Ta có nhiều cấu trúc huấn luyện sử dụng mạng nơron như:

* Huấn luyện theo vòng hở
* Huấn luyện theo vòng kín
* Huấn luyện với mô hình tham chiếu
* Huấn luyện theo thời gian vượt quá (over time)
* Bộ huấn luyện với quyết định hổ trợ của mạng nơron

Ứng dụng của mạng Nơ ron nhân tạo:

* Nhận dạng đối tượng.
* Điều khiển hện thống.
* Phân tích số học và tính toán.
* So sánh.

### 1.1.4. Ngôn ngữ thực thi mạng nơ-ron

Python là ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, được phát triển bởi ***Guido van Rossum***, nó trở nên phổ biến rất nhanh, chủ yếu bởi có ưu điểm mạnh về dễ đọc, dễ học và dễ nhớ. Python cũng là ngôn ngữ có hình thức rất “sáng sủa”, cấu trúc rõ ràng và thuận tiện cho người mới lập trình.



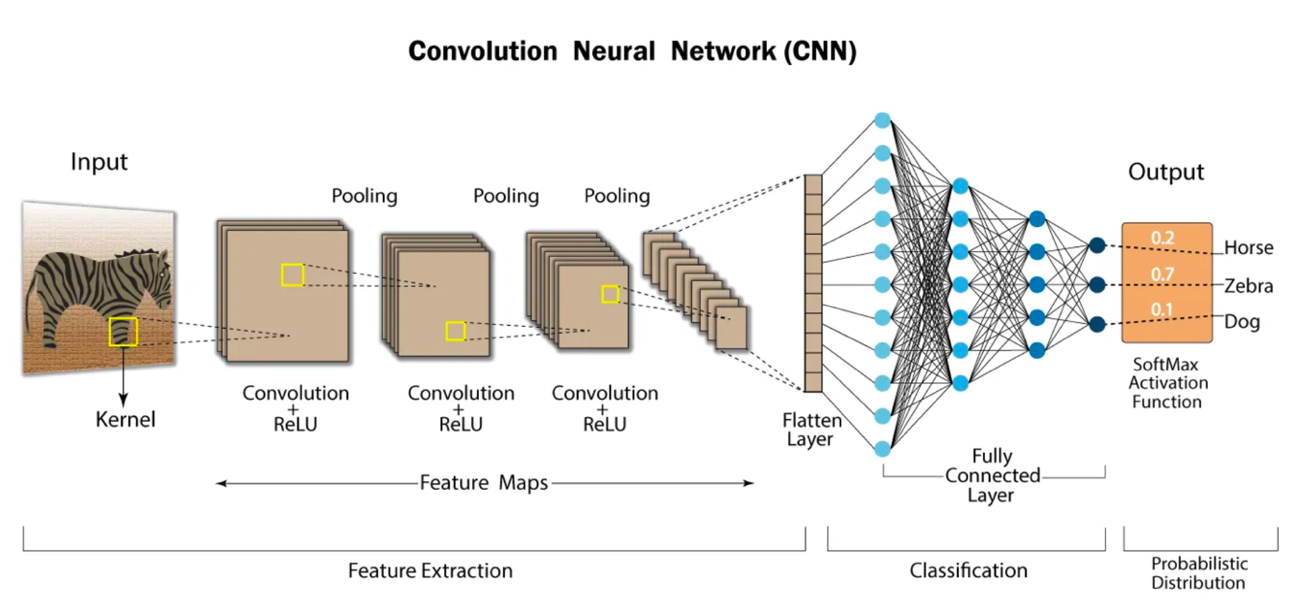
Hình 1. 5 Ngôn ngữ lập trình Python

## ****1.2. Mạng nơ-ron tích chập****

### 1.2.1. Khái niệm mạng nơ-ron tích chập

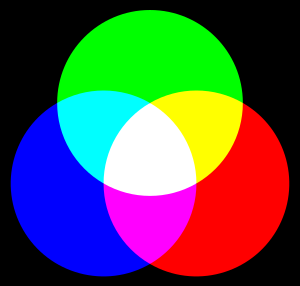
Mạng nơron tích chập (còn gọi là ConvNet / CNN) là một thuật toán Deep Learning có thể lấy hình ảnh đầu vào, gán độ quan trọng (các trọng số - weights và độ lệch - bias có thể học được) cho các đặc trưng/đối tượng khác nhau trong hình ảnh và có thể phân biệt được từng đặc trưng/đối tượng này với nhau. Công việc tiền xử lý được yêu cầu cho mạng nơron tích chập thì ít hơn nhiều so với các thuật toán phân loại khác. Trong các phương thức sơ khai, các bộ lọc được thiết kế bằng tay (hand - engineered), với một quá trình huấn luyện để chọn ra các bộ lọc/đặc trưng phù hợp thì mạng nơron tích chập lại có khả năng tự học để chọn ra các bộ lọc/ đặc trưng tối ưu nhất.

Kiến trúc của nơron tích chập tương tự như mô hình kết nối của các nơron trong bộ não con người và được lấy cảm hứng từ hệ thống vỏ thị giác trong bộ não (visual cortex). Các nơ-ron riêng lẻ chỉ phản ứng với các kích thích trong một khu vực hạn chế của trường thị giác được gọi là Trường tiếp nhận (Receptive Field). Một tập hợp các trường như vậy chồng lên nhau để bao phủ toàn bộ khu vực thị giác.



Hình 1. 6 Mô hình lớp của mạng nơ ron tích chập

#### 1.2.1.1. Hình ảnh đầu vào (input)

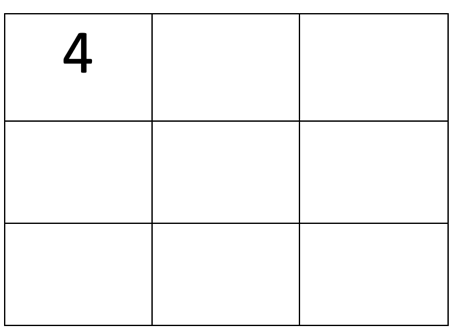
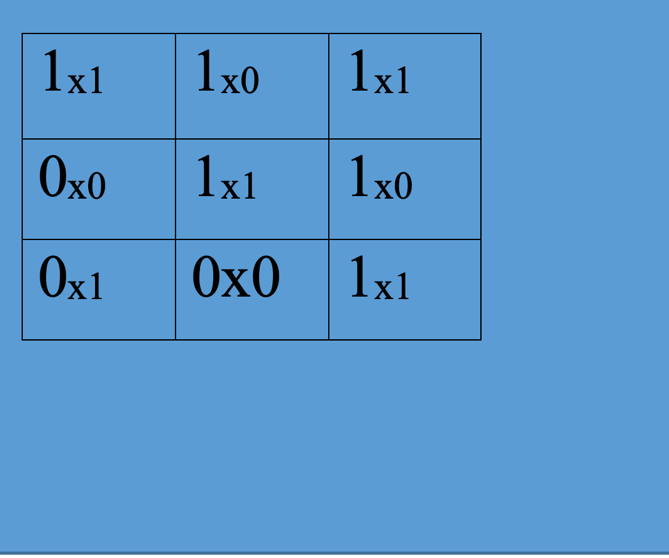
Một hình ảnh RGB thông thường được phân tách theo 3 mặt phẳng (kênh) màu - Đỏ, Xanh lục và Xanh dương. Có một số không gian màu khác cho hình ảnh như Grayscale, RGB, HSV, CMYK, v.v..

Hình 1. 7 Ba dải màu của một hình ảnh thông thường

Ví dụ khi bạn chọn màu ở đây. Khi bạn chọn một màu thì sẽ ra một bộ ba số tương ứng (r, g, b) Với mỗi bộ 3 số r, g, b nguyên trong khoảng [0, 255] sẽ cho ra một màu khác nhau. Do có 256 cách chọn r, 256 cách chọn màu g, 256 cách chọn b => tổng số màu có thể tạo ra bằng hệ màu RGB là: 256 \* 256 \* 256 = 16777216 màu.

Bạn có thể thấy chi phí tính toán sẽ lớn như thế nào khi hình ảnh đạt đến kích thước, giả sử 8K (7680 pixel × 4320 pixel). Vai trò của nơron tích chập là giảm chiều hình ảnh thành một dạng dễ xử lý hơn, mà không đánh mất đi các đặc trưng quan trọng của hình ảnh để có được một dự đoán tốt (good prediction). Điều này rất quan trọng khi chúng ta thiết kế một kiến ​​trúc không chỉ tốt về việc học tập các đặc trưng mà còn có thể tương thích với các bộ dữ liệu có kích thước lớn.

#### 1.2.1.2. Bộ lọc (kernel)



Hình 1. 8 Kết quả của bộ lọc là một ma trận

Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ đặc trưng của hinh ảnh.

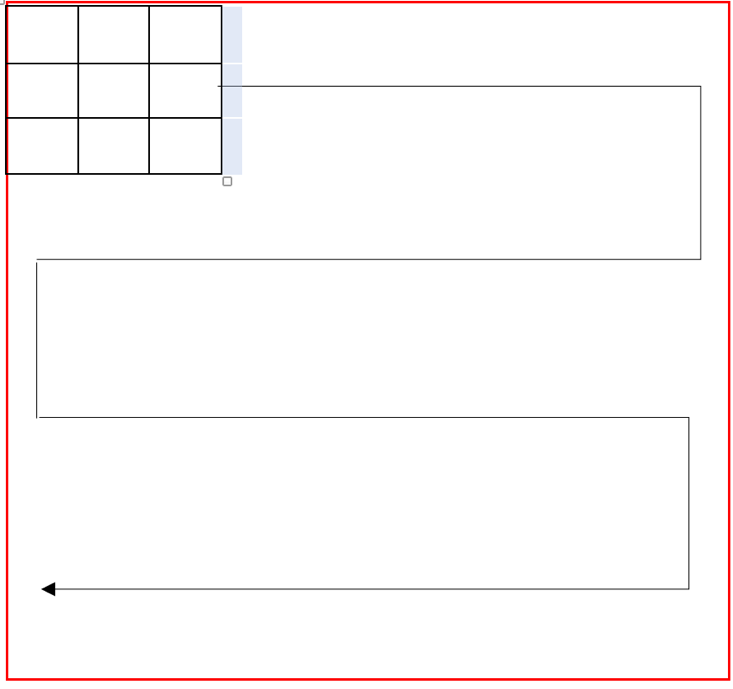
Kích thước hình ảnh = 55 (Chiều cao) x 55 (Chiều rộng) x 11 (Số lượng kênh, ví dụ: RGB).

Ở hình bên, phần màu xanh lục là hình ảnh đầu vào 5\times 5\times 15×5×1 của chúng ta, ta gọi hình ảnh đầu vào này là I*I*

Phần tử liên quan đến việc thực hiện thao tác tích chập trong phần đầu tiên của lớp tích chập được gọi là **Bộ lọc (Kernel / Filter)** , K*K*, được thể hiện bằng màu vàng. Chúng ta chọn K*K* là một ma trận 3 \times 3 \times 3×3×1.

Bộ lọc di chuyển 9 lần vì **độ dài dải trượt (Stride Length)** = 1 (tức không bị trượt), với mỗi lần di chuyển sẽ thực hiện một phép nhân ma trận giữa bộ lọc K*K* và tỉ lệ P*P* của bức ảnh tương ứng với vị trí mà bộ lọc lúc đó đang đi qua.

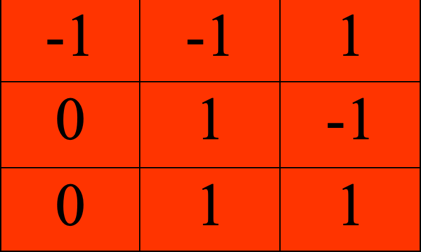
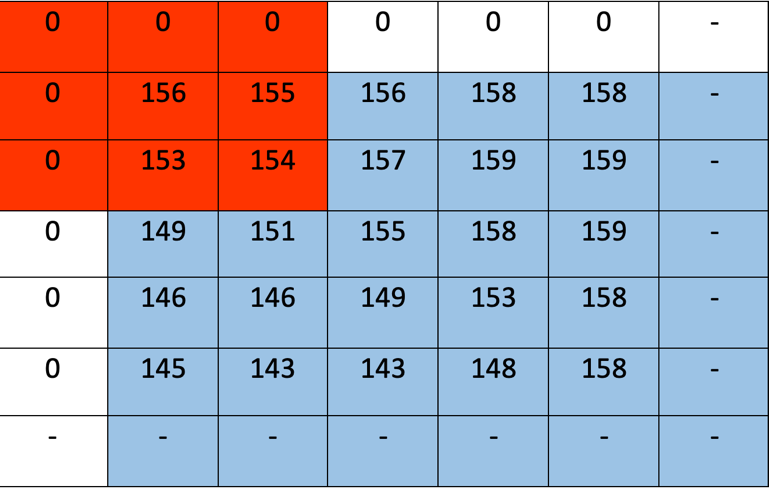
Bộ lọc di chuyển sang phải với Giá trị trượt cố định cho đến khi hoàn thành việc quét theo chiều rộng. Tiếp tục, nó nhảy xuống phía đầu bên trái của hình ảnh với cùng Giá trị trượt và lặp lại quá trình cho đến khi toàn bộ hình ảnh được duyệt qua.



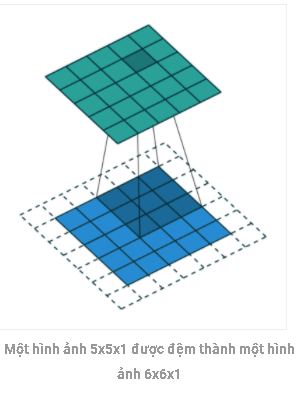
Hình 1. 9 Cách di chuyển của bộ lọc

Trong trường hợp hình ảnh có nhiều kênh (ví dụ: RGB), Bộ lọc (Kernel) có cùng độ sâu với hình ảnh đầu vào. Phép nhân ma trận được thực hiện giữa  Kn*Kn* và In*In* ([K1, I1]; [K2, I2]; [K3, I3])([*K*1,*I*1];[*K*2,*I*2];[*K*3,*I*3]) và tất cả các kết quả được cộng với độ lệch để cung cấp cho chúng ta một kênh một chiều Đầu ra đặc trưng tích chập (Convoluted Feature Output).

Mục tiêu của phép tính tích chập là trích xuất các đặc trưng cấp cao như các cạnh (edges), từ hình ảnh đầu vào. Mạng nơron tích chập không nhất thiết chỉ giới hạn trong một lớp tích chập. Thông thường, lớp tích chập đầu tiên chịu trách nhiệm nắm bắt các đặc trưng cấp thấp như màu sắc (colors), hướng dốc (gradient orientation), v.v. Với các lớp tích chập được thêm vào, mô hình cũng nắm bắt các đặc trưng cấp cao, mang đến cho chúng ta một mạng lưới nơron tích chập có sự hiểu biết toàn diện về hình ảnh trong bộ dữ liệu, tương tự như cách chúng ta - con người hiểu về hình ảnh.



Hình 1. 10 Bộ lọc

Có hai loại kết quả cho phép tính chập - một loại trong đó kết quả tích chập bị giảm chiều so với đầu vào và loại khác trong đó chiều của kết quả tích chập được tăng hoặc giữ nguyên. Điều này được thực hiện bằng cách áp dụng phép **Đệm hợp lệ (Valid** Padding) trong trường hợp trước, hoặc phép Đệm tương tự (Same Padding) trong trường hợp sau.

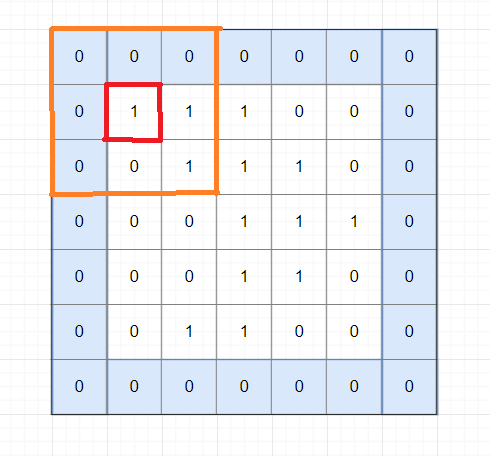
Hình 1. 11 chuyển đổi

Khi chúng ta tăng (đệm) hình ảnh có kích thước 5 ×5×1 thành hình ảnh 6×6×1 và sau đó áp dụng bộ 3×33×1 lên nó, chúng ta thấy rằng ma trận tích chập đầu ra có kích thước 5×5×1. Đó là phép Đệm tương tự (same padding).

Mặt khác, nếu chúng ta thực hiện cùng một thao tác mà không cần đệm, chúng ta sẽ có một ma trận tích chập đầu ra có kích thước 3×3×1bằng chính bộ lọc - Đó là Đệm hợp lệ (valid padding).

#### 1.2.1.3. Padding

Như ở trên thì mỗi lần thực hiện phép tính convolution xong thì kích thước ma trận Y đều nhỏ hơn X. Tuy nhiên giờ ta muốn ma trận Y thu được có kích thước bằng ma trận X => Tìm cách giải quyết cho các phần tử ở viền => Thêm giá trị 0 ở viền ngoài ma trận X. [3]



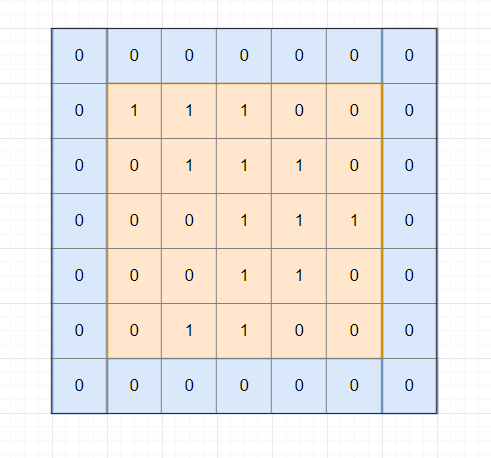
Hình 1. 12 Padding

Rõ ràng là giờ đã giải quyết được vấn đề tìm A cho phần tử x11, và ma trận Y thu được sẽ bằng kích thước ma trận X ban đầu

Phép tính này gọi là convolution với padding=1. Padding=k nghĩa là thêm k vector 0 vào mỗi phía (trên, dưới, trái, phải) của ma trận.

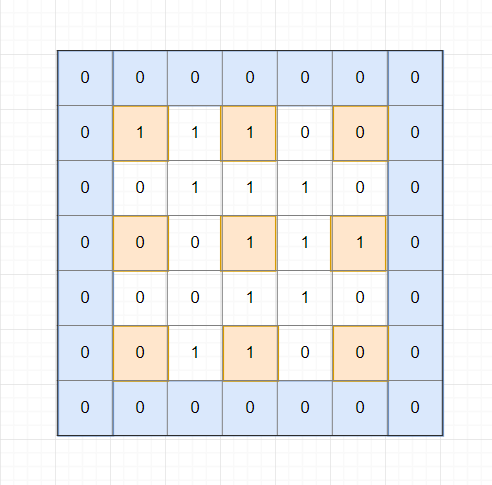
#### 1.2.1.4. Bước nhảy

Như ở trên ta thực hiện tuần tự các phần tử trong ma trận X, thu được ma trận Y cùng kích thước ma trận X [3], ta gọi là stride=1.



Hình 1. 13 Bước nhảy = 1

Tuy nhiên nếu stride=k (k > 1) thì ta chỉ thực hiện phép tính convolution trên các phần tử x1+i∗k;1+j∗k. Ví dụ k = 2.



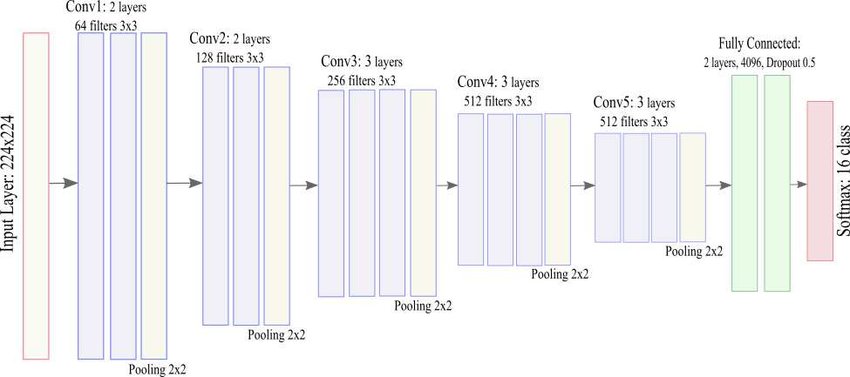
Hình 1. 14 Bước nhảy = 2

Hiểu đơn giản là bắt đầu từ vị trí x11 sau đó nhảy k bước theo chiều dọc và ngang cho đến hết ma trận X. Kích thước của ma trận Y là 3\*3 đã giảm đi đáng kể so với ma trận X. Công thức tổng quát cho phép tính convolution của ma trận X kích thước m\*n với kernel kích thước

Stride thường dùng để giảm kích thước của ma trận sau phép tính convolution.

### 1.3. Mạng VGG 16

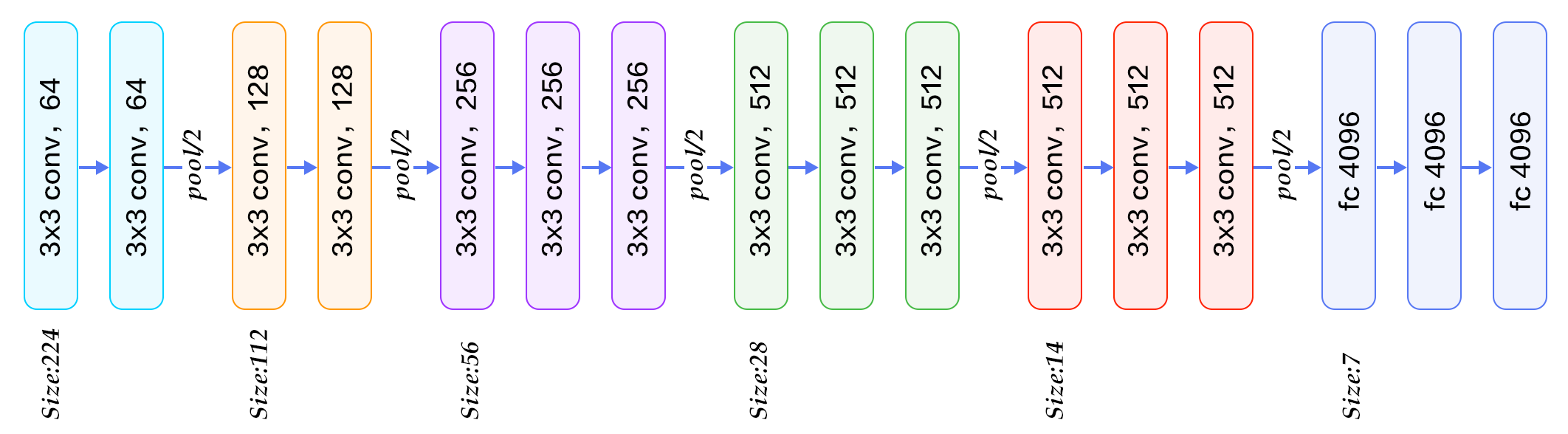
**VGG16** là một mô hình mạng nơ-ron tích **tụ** được đề xuất bởi K. Simonyan và A. Zisserman từ Đại học Oxford trong bài báo “Mạng lưới kết hợp rất sâu để nhận dạng hình ảnh quy mô lớn”. Mô hình đạt được độ chính xác trong bài kiểm tra top 5 là 92,7% trong ImageNet,là một tập dữ liệu của hơn 14 triệu hình ảnh thuộc 1000 lớp. Đây là một trong những mô hình nổi tiếng được nộp cho ILSVRC-2014 . Nó cải tiến hơn AlexNet bằng cách thay thế các bộ lọc có kích thước hạt nhân lớn (11 và 5 trong lớp chập đầu tiên và thứ hai, tương ứng) bằng nhiều bộ lọc kích thước hạt nhân 3 × 3 lần lượt. VGG16 đã được đào tạo trong nhiều tuần và đang sử dụng GPU NVIDIA Titan Black.



Hình 1. 15 Cấu trúc các lớp của mạng VGG16

Với model này, khi ta đưa ảnh 224x224x3 vào , nó sẽ trả ra 1 vector softmax để từ đó ta in ra class của tấm ảnh.

Ngoài ra, nhìn vào đó các bạn sẽ tháy có tất cả 5 convolution block và trong đó có 13 convolution layers (màu xanh blue). Kích thước output của các conv block như sau:

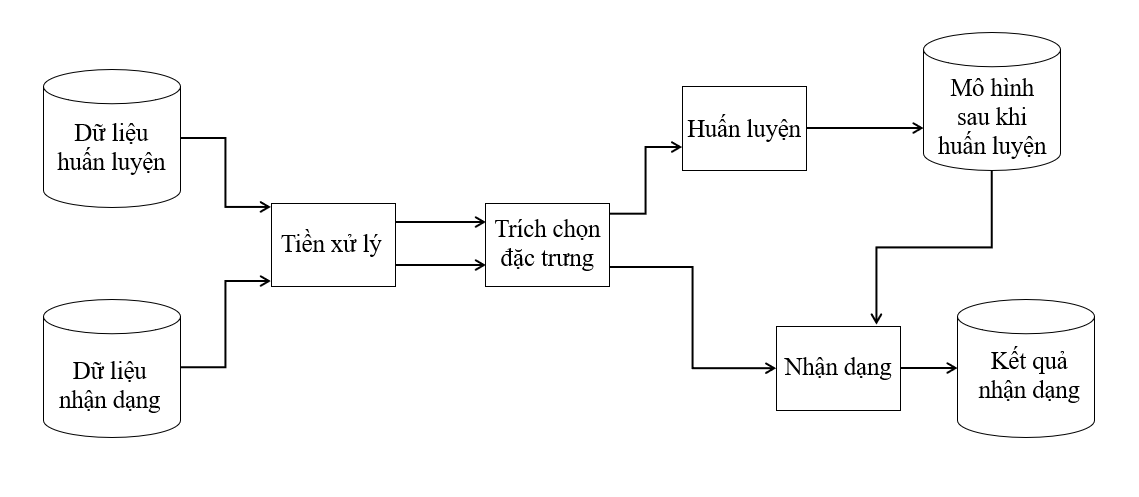


Hình 1. 16 5 khối Convolution

# CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG NHẬN DIỆN TIỀN VIỆT NAM

## 2.1. Mô tả và phân tích bài toán

 Đầu vào là ảnh (128,128,3), đưa vào mạng VGG16 và đầu ra của VGG16 sẽ dùng để đưa vào 1 mạng NN nhỏ kết thúc bằng 1 lớp Dense và hàm softmax.. Đầu ra sẽ là 1 vector softmax chứa các probality p(i) ứng với mỗi class i, chúng ta sẽ in ra giá trị max trong vector đó và chọn đó làm class dự đoán.



Hình 2. 1 Quy trình giải quyết bài toán

## 2.2. Các giai đoạn

+Tạo dữ liệu

+ Xử lý dữ liệu

+Thiết kế mạng CNN Classify dùng để train

+Train model CNN Classify

+Thử nghiệm model CNN Classify

### 2.2.1.Tạo dữ liệu

Cách tạo dữ liệu đơn giản là viết một đoạn python đọc liên tục từ camera và save lại vào các thư mục tương ứng ảnh các tờ tiền.

### 2.2.2. Xử lý dữ liệu:

Các bạn để ý như sau, trong quá trình chúng ta train model, thử nghiệm model thì sẽ phải chạy chương trình rất nhiều lần để debug, sửa lỗi… Như vậy mà mỗi lần chạy lại phải mò hết đống file và thư mục trên thì sẽ khá lâu nên chúng ta sẽ đọc 1 lần và thực hiện:

* Convert nhãn (là cái mớ 00000,10000,20000….) thành one-hot
* Resize ảnh về 128×128 và lưu vào file pickle để lần sau load cho tiện. Sau khi lưu xong thì chúng ta không cần quan tâm đến cái mớ folder data kia nữa mà chỉ quan tâm đến file pix.data mới sinh ra. Nó chứa full cả ảnh và labels rồi.

### 2.2.3. Thiết kế mạng CNN Classify dùng để train:

Ở đây chúng ta sẽ sử dụng mạng VGG 16 để train .  Nhưng classify ở đây là dùng cho mục đích của người ta, ko phải cho bài toán nhận dạng tiền của chúng ta nên chúng ta sẽ bỏ softmax layer của người ta và thay vào đó là FC của mình và output sẽ là 1 vector chứa probality từng class.

# Đóng băng các layers

for layer in model\_vgg16\_conv.layers:

layer.trainable = False

# Tạo model với input là ảnh, lấy output của VGG16 và làm input của các layers FC thêm vào

input = Input(shape=(128, 128, 3), name='image\_input')

output\_vgg16\_conv = model\_vgg16\_conv(input)

# Thêm vào các FC layers

x = Flatten(name='flatten')(output\_vgg16\_conv)

x = Dense(4096, activation='relu', name='fc1')(x)

x = Dropout(0.5)(x)

x = Dense(4096, activation='relu', name='fc2')(x)

x = Dropout(0.5)(x)

x = Dense(4, activation='softmax', name='predictions')(x)

### 2.2.4. Train model CNN Classify

Chúng ta sẽ sử dụng model.fit\_generator . Lý do đơn giản chúng ta sinh ra dữ liệu trong quá trình train:

vgghist=vggmodel.fit\_generator(aug.flow(X\_train,y\_train, batch\_size=64),

epochs=50,

validation\_data=aug.flow(X\_test,y\_test,

batch\_size=len(X\_test)),

callbacks=callbacks\_list)

### 2.2.5. Thử nghiệm model CNN Classify

Chạy chương trình sau đó đưa các tờ tiền lên camera và nhận kết quả.

# Predict

predict = my\_model.predict(image)

print("This picture is: ", class\_name[np.argmax(predict[0])], (predict[0]))

print(np.max(predict[0],axis=0))

if (np.max(predict)>=0.8) and (np.argmax(predict[0])!=0):

# Show image

font = cv2.FONT\_HERSHEY\_SIMPLEX

org = (50, 50)

fontScale = 1.5

color = (0, 255, 0)

thickness = 2

cv2.putText(image\_org, class\_name[np.argmax(predict)], org, font, fontScale, color, thickness, cv2.LINE\_AA)

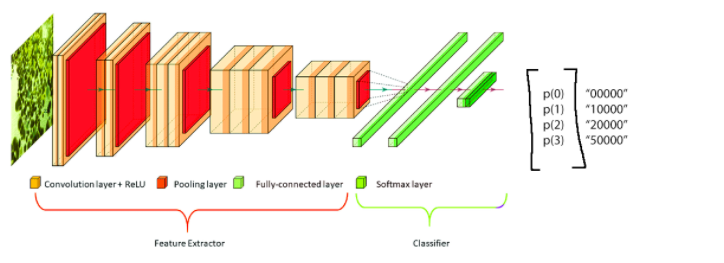
### 2.2.6. Mô tả quá trình trong hệ thống nhận dạng.

CNN sử dụng các khái niệm các filter với các kernel size khác nhau để quét qua tấm ảnh từ trên xuống dưới, từ trái sang phải (lý thuyết thôi nhé, thực tế train qua GPU nó chơi đồng thời luôn ). Sau khi quét xong, nó sẽ tạo ra (bằng cách nhân lần lượt ma trận filter và một phần của tấm ảnh) các feature maps các đặc trưng trích ra được từ tấm ảnh rồi predict ra đúng class của tấm ảnh.

Và features maps chính là những gì mà CNN nhìn thấy từ tấm ảnh.

## 2.3. Thiết kế mô hình mạng nơ ron nhận dạng tiền việt nam đồng

#### 2.3.1. Cấu trúc mô hình VGG sẽ sử dụng trong đề tài



Hình 2. 2 Cấu trúc mạng VGG16

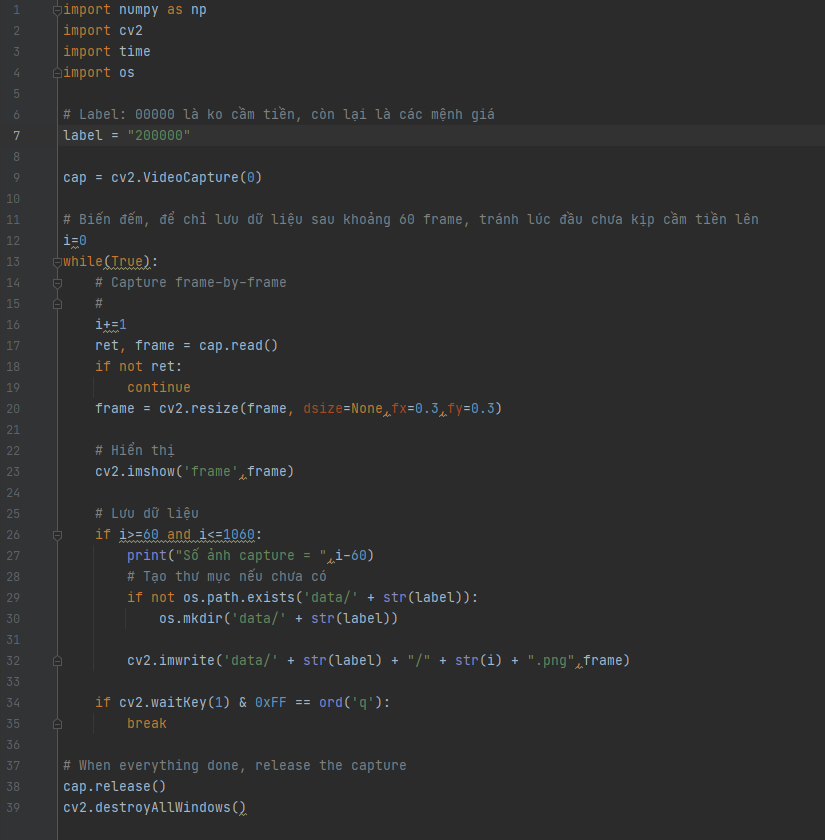
Đầu vào cho lớp cov1 có kích thước cố định 128x128 . Hình ảnh được chuyển qua một chồng các lớp tích hợp (ch.đổi), trong đó các bộ lọc được sử dụng với trường tiếp nhận rất nhỏ: 3 × 3 (là kích thước nhỏ nhất để nắm bắt khái niệm trái / phải, lên / xuống, trung tâm ). Trong một trong các cấu hình, nó cũng sử dụng bộ lọc tích chập 1 × 1, có thể được xem như một phép biến đổi tuyến tính của các kênh đầu vào (theo sau là không tuyến tính). Sải tích chập được cố định thành 1 pixel; phần đệm không gian của lượt chuyển đổi đầu vào lớp sao cho độ phân giải không gian được giữ nguyên sau khi tích chập, tức là phần đệm là 1 pixel cho chuyển đổi 3 × 3. các lớp. Tổng hợp theo không gian được thực hiện bởi năm lớp tổng hợp tối đa, theo sau một số lượt chuyển đổi. các lớp (không phải tất cả các lớp chuyển đổi đều được theo sau bởi tổng hợp tối đa). Tổng hợp tối đa được thực hiện trên cửa sổ 2 × 2 pixel.

Ba lớp được kết nối đầy đủ (FC) tuân theo một chồng các lớp phức hợp (có độ sâu khác nhau trong các kiến ​​trúc khác nhau): hai lớp đầu tiên có 4096 kênh mỗi lớp,  Lớp cuối cùng là lớp soft-max. Ở đây ta sử dụng cấu trúc mạng VGG16 và có sửa đổi đôi chút ở lớp cuối để   output sẽ là 1 vector chứa probality từng class.

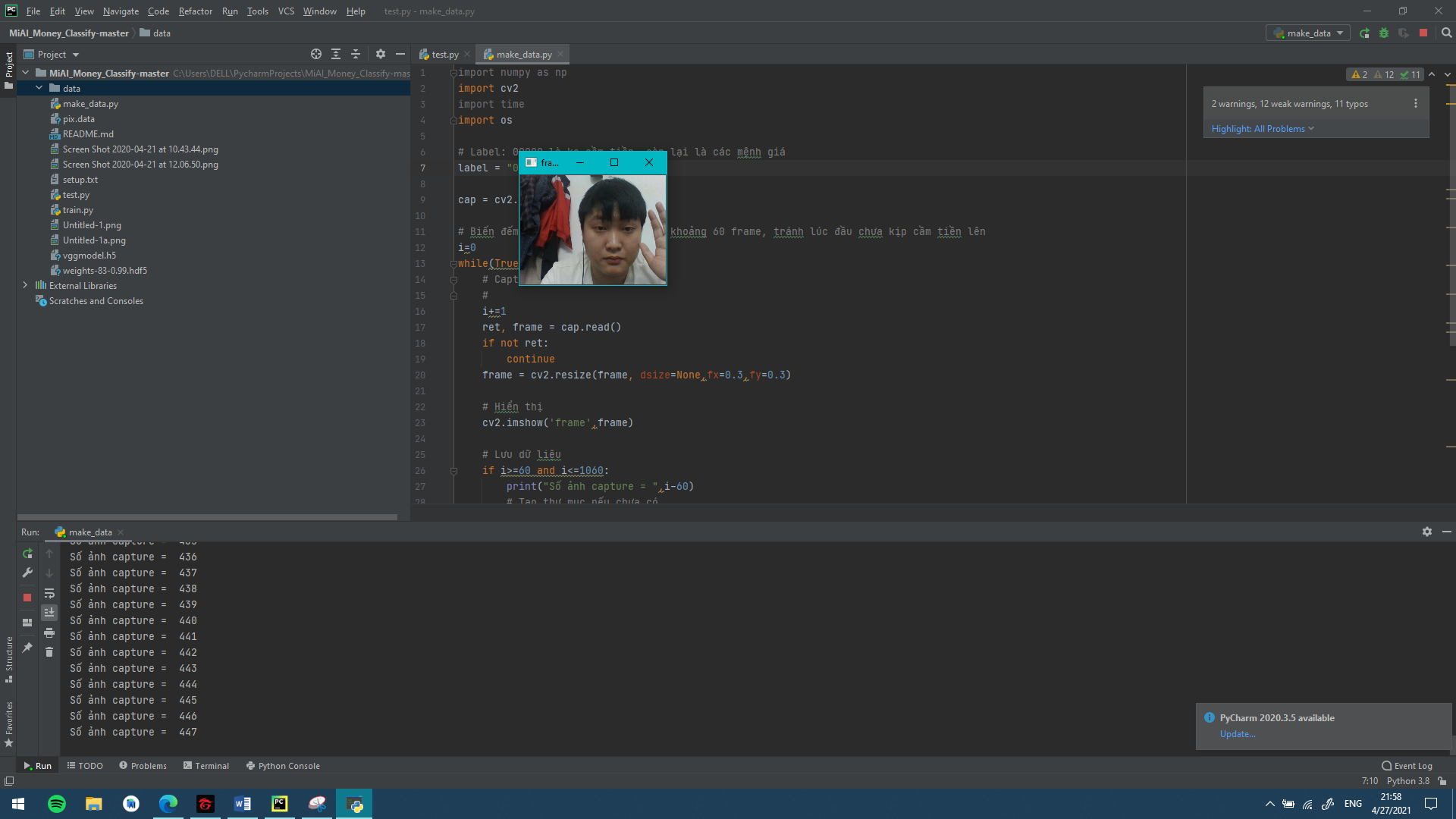
Cấu hình của các lớp được kết nối đầy đủ là giống nhau trong tất cả các mạng.

### 2.3.2. Thực thi mô hình mạng nơ ron nhận dạng tiền Việt Nam Đồng

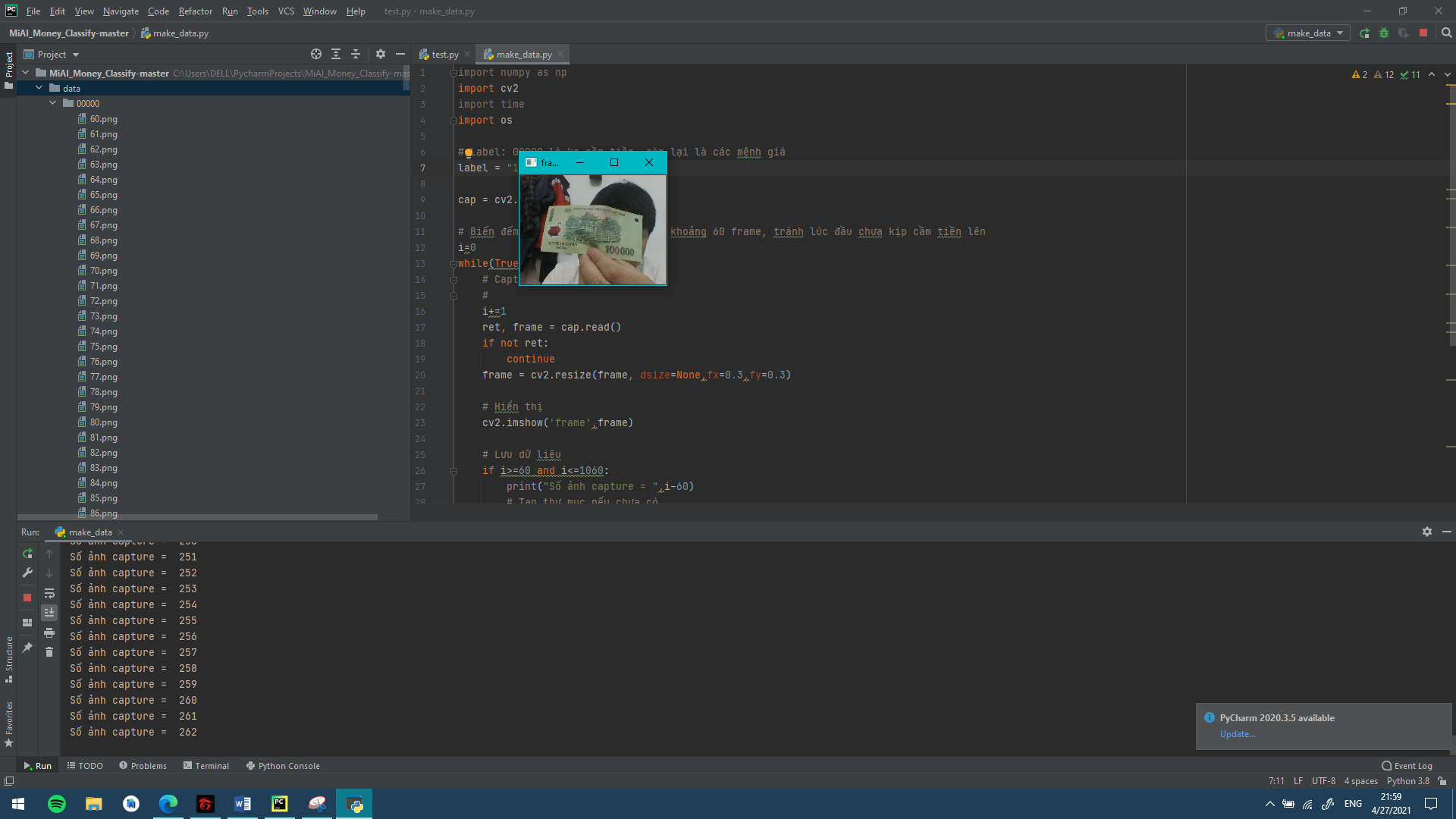
#### 2.3.2.1. Chuẩn bị dữ liệu



Hình 2. 3 Dữ liệu được lấy từ camera thời gian thực

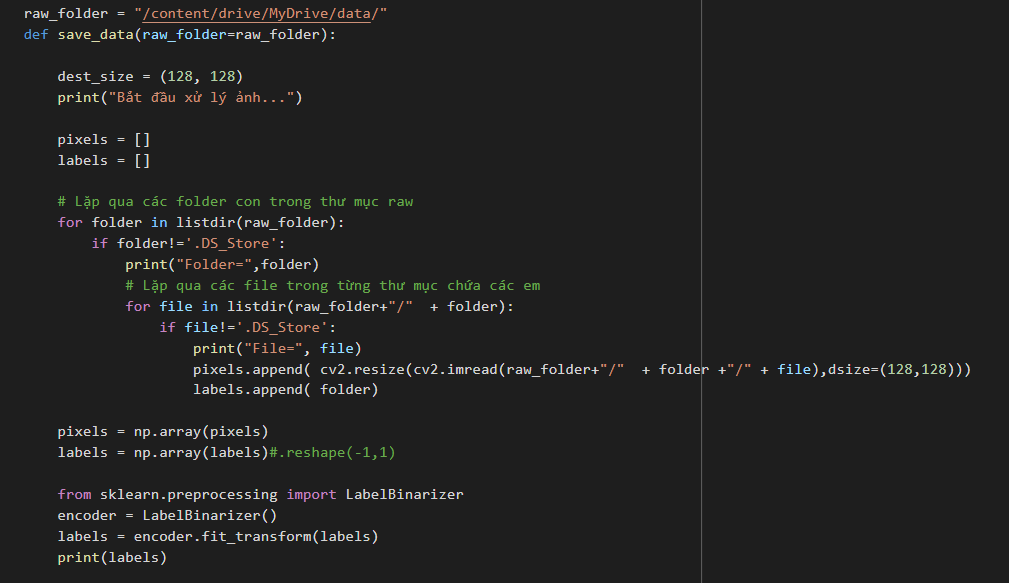


Hình 2. 4 Tạo dữ liệu khi không có tiền

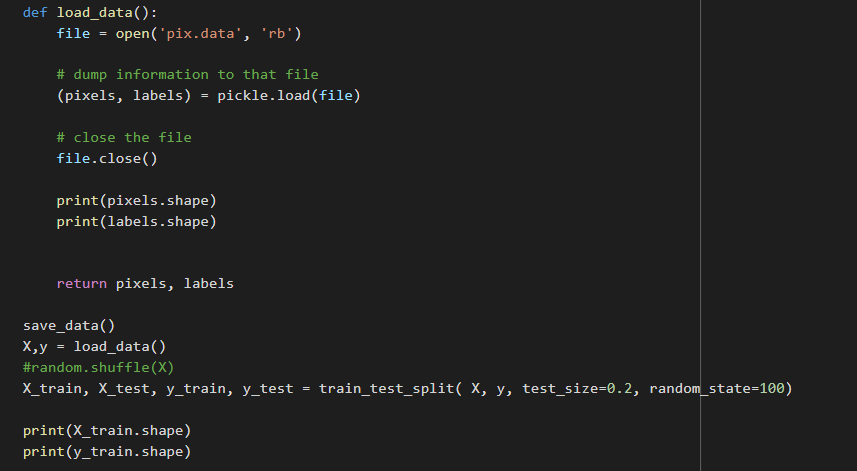


Hình 2.5 Tạo dữ liệu cho tờ tiền 100.000 đồng

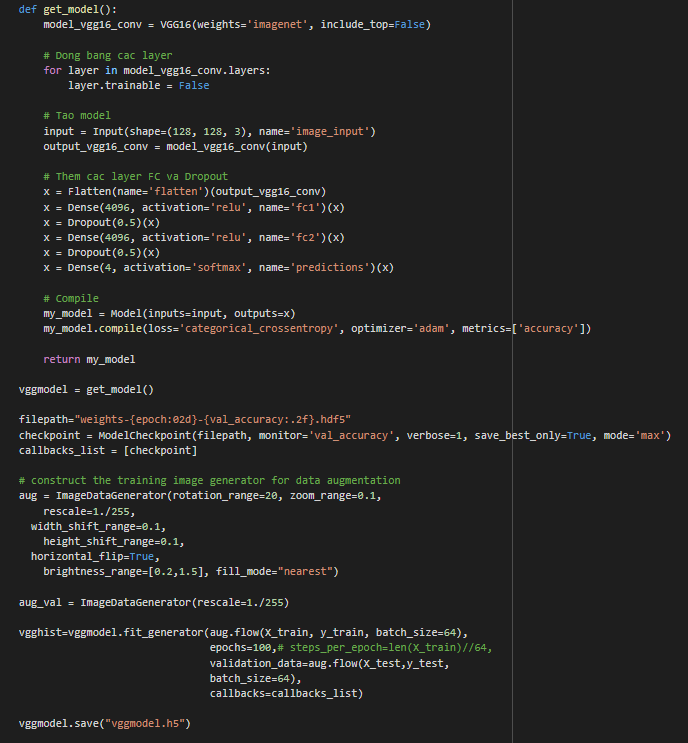
### 2.3.2.2. Huấn luyện



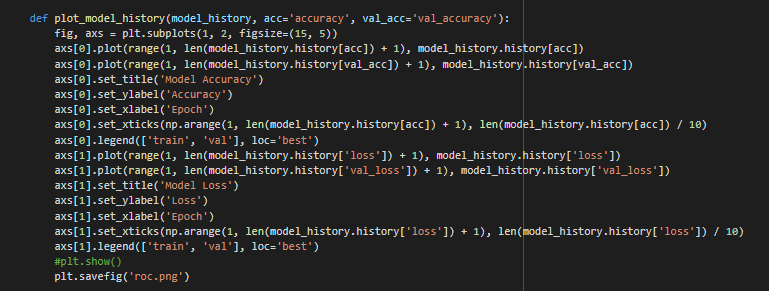
Hình 2. 6 Tạo thư mục lưu dữ liệu và đường dẫn



Hình 2. 7 Load các file data trong thư mục vừa tạo



Hình 2. 8 Khung chương trình tạo các lớp của mạng VGG16



### 2.3.2.3. Kết quả



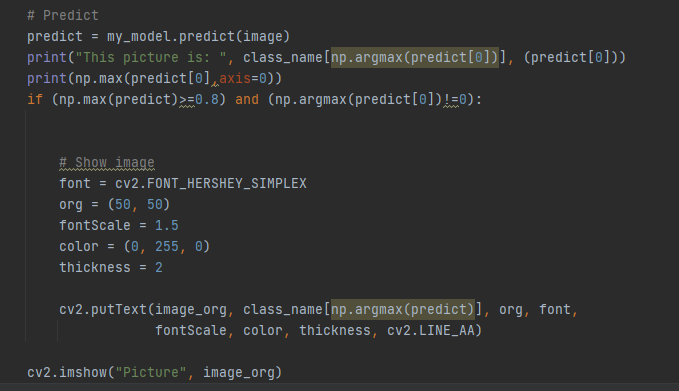
Hình 2. 9 Kết quả khi không cầm tiền



Hình 2. 10 Kết quả khi đưa đồng 10.000 vnd

## 2.4. Xây dựng giao diện

### 2.4.1. Cách sử dụng giao diện



Hình 2. 11 Thiết lập giao diện

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Khi sử dụng mạng VGG 16 cho bài toán trên thì độ chính xác rất cao với những đồng tiền colime nhưng còn hạn chế khi nhận dạng tiền giấy 1000đ,2000đ,5000đ,..

Nếu đưa ra ngoài đời để làm sản phẩm   thì cần nhận diện nhiều góc hơn, nhiều level ánh sáng hơn để tránh nhầm lẫn, predict trong vài frame liên tục và lấy bình quân có trọng số để tăng độ chính xác, tránh nhận nhầm,giới hạn vùng xử lý (ví dụ ở giữa khung hinh) để tăng tốc độ xử lý video…

# TÀI LIỆU THAM KHẢO

<https://www.miai.vn/2020/04/21/nhan-dang-tien-viet-nam-voi-transfer-learning-vgg16-cnn-classify/>

<https://nttuan8.com/bai-6-convolutional-neural-network/>

# https://viblo.asia/p/tim-hieu-ve-convolutional-neural-network-va-lam-mot-vi-du-nho-ve-phan-loai-anh-aWj53WXo56m