

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Tp HCM tháng 7 năm 2020



Nhóm sinh viên thực hiện:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Nguyễn Đức Nguyện | 16520843 | | Vũ Văn Hiến | 17520463 | | Lê Võ Ngọc Anh | 18520452 | | Bùi Thanh Tuấn | 18520395 | | Phan Văn Anh Quốc | 16521525 | |  |
|  |  |
|  |  |

GVHD: Nguyễn Đình Hiển

Đề tài: Phương pháp Transfer Learning

Và xây dựng ứng dụng nhận dạng loại hoa

Lớp: CS106.K21

Môn: Trí tuệ nhân tạo

Báo cáo đồ án

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

Mục lục

[1. Giới thiệu đề tài 3](#_Toc44353256)

[1.1 Đề tài: 3](#_Toc44353257)

[1.2 Mô tả dữ liệu 3](#_Toc44353258)

[2. Logistic regression 4](#_Toc44353259)

[3. Support Vector Machine 4](#_Toc44353260)

[3.1 Định nghĩa 4](#_Toc44353261)

[4. Neural Network 6](#_Toc44353262)

[4.1 Neural Network là gì? 6](#_Toc44353263)

[4.2 Mô hình neural 7](#_Toc44353264)

[5. Giới thiệu về xử lý ảnh 7](#_Toc44353265)

[5.1 Hệ màu RGB 7](#_Toc44353266)

[5.2 Điểm ảnh pixel 8](#_Toc44353267)

[5.3 Tensor 8](#_Toc44353268)

[5.4 Ảnh xám 9](#_Toc44353269)

[5.5 Phép tính convolution 9](#_Toc44353270)

[5.5.1 Padding 10](#_Toc44353271)

[5.5.2 Stride 10](#_Toc44353272)

[5.5.3 Ý nghĩa 11](#_Toc44353273)

[6. Convolutional neural network 12](#_Toc44353274)

[6.1 Convolutional layer 12](#_Toc44353275)

[6.2 Pooling layer 13](#_Toc44353276)

[6.3 Fully connected layer 14](#_Toc44353277)

[6.4 Visualise convolutional neural network 14](#_Toc44353278)

[6.5 Mạng VGG 16 15](#_Toc44353279)

[6. Transfer Learning 15](#_Toc44353280)

[6.1 Feature extractor 16](#_Toc44353281)

[6.2 Fine tuning 17](#_Toc44353282)

[6.2.1 Data augmentation 17](#_Toc44353283)

[6.2.2 Thuật toán Fine Tuning 18](#_Toc44353284)

[7. Tài liệu tham khảo 22](#_Toc44353285)

[8. Bảng phân công công việc 23](#_Toc44353286)

# 1. Giới thiệu đề tài

## 1.1 Đề tài:

Sử dụng phương pháp Transfer learning để phát triển mô hình dự đoán tên loại hoa.

## 1.2 Mô tả dữ liệu

Nguồn dữ liệu: <https://www.kaggle.com/rajmehra03/flower-recognition-cnn-keras/data> (lần cuối truy cập 29/6/2020).

Dữ liệu được sử dụng gồm 5 Loại Hoa với 4326 mẫu:

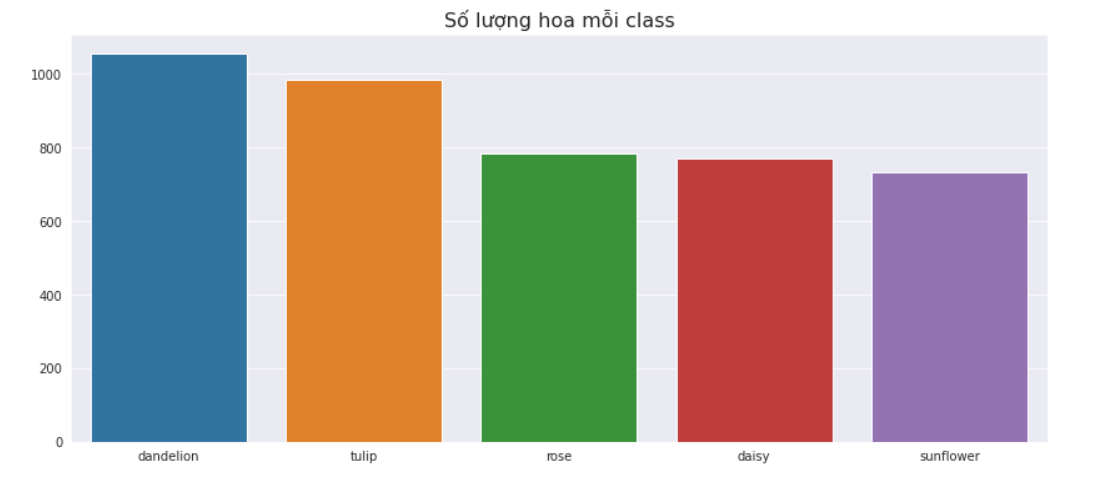
Bồ Công Anh: 1055.

Hoa Tuplip: 984.

Hoa Hồng: 784.

Hoa Cúc:769.

Hoa Hướng Dương: 734.



Kết luận chung:

* Dữ liệu khá ít, chỉ với 4326 mẫu cho 5 lớp.
* Dữ liệu mất sự cân bằng giữa các lớp.

# 2. Logistic regression

Ví dụ: Ngân hàng bạn đang làm có chương trình cho vay ưu đãi cho các đối tượng mua chung cư. Tuy nhiên gần đây có một vài chung cư rất hấp dẫn (giá tốt, vị trí đẹp,…) nên lượng hồ sơ người nộp cho chương trình ưu đãi tăng đáng kể. Bình thường bạn có thể duyệt 10-20 hồ sơ một ngày để quyết định hồ sơ có được cho vay hay không, tuy nhiên gần đây bạn nhận được 1000-2000 hồ sơ mỗi ngày. Bạn không thể xử lý hết hồ sơ và bạn cần có một giải pháp để có thể dự đoán hồ sơ mới là có nên cho vay hay không.

Sau khi phân tích thì bạn nhận thấy là hai yếu tố chính quyết định đến việc được vay tiền đó là mức lương và thời gian công tác. Đây là dữ liệu bạn có từ trước đến nay:

|  |  |  |
| --- | --- | --- |
| Lương | Thời gian làm việc | Cho vay |
| 10 | 1 | 1 |
| 5 | 2 | 1 |
| ... | ... | 1 |
| 8 | 0.1 | 0 |
| 7 | 0.15 | 0 |
| ... | ... | 0 |

Tuy nhiên, do ngân hàng đang trong thời kỳ khó khăn nên việc cho vay bị thắt lại, chỉ những hồ sơ nào chắc chắn trên 80% mới được vay.

=> Sử dụng mô hình logistic regression để đưa ra giá trị đầu ra là xác suất cho vay.

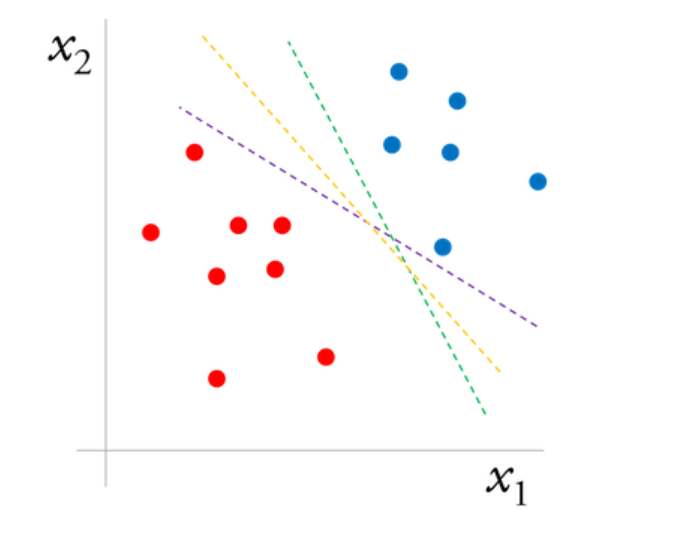
Kết luận:

Mô hình logistic được sử dụng để tính xác suất của một sự kiện nào đó tồn tại như thành công/ thất bại, thắng / thua, sống / chết.

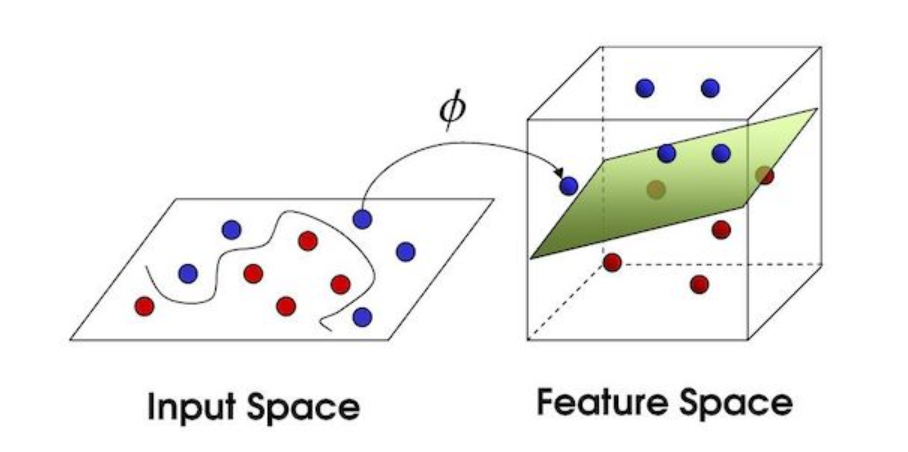
# 3. Support Vector Machine

## 3.1 Định nghĩa

Support Vector Machine (SVM) là một thuật toán thuộc nhóm Supervised Learning (Học có giám sát) dùng để phân chia dữ liệu (Classification) thành các nhóm riêng biệt.



Đối với những bộ dữ liệu phức tạp hơn, ta cần dùng thuật toán để ánh xạ bộ data đó vào không gian nhiều chiều hơn (n chiều), từ đó tìm ra siêu mặt phẳng (hyperplane) để phân chia.

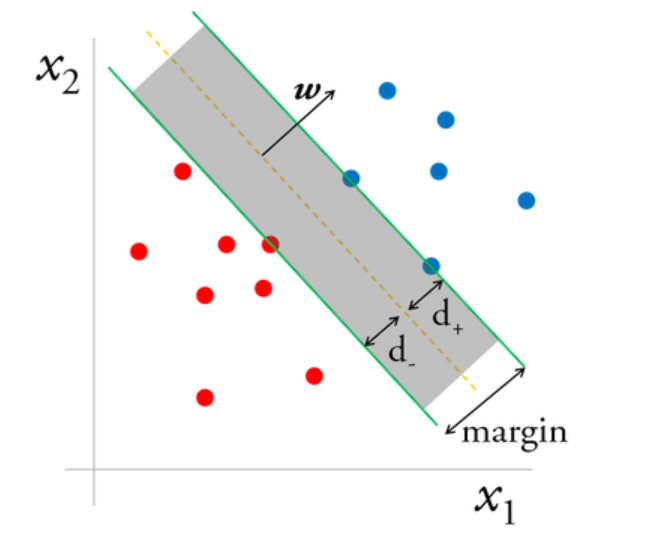


3.2 Tối ưu trong thuật toán SVM

Margin là khoảng cách giữa siêu phẳng đến 2 điểm dữ liệu gần nhất tương ứng với 2 phân lớp.

Tối ưu thuật toán bằng các tìm cách maximize giá trị margin này, từ đó tìm ra siêu phẳng đẹp nhất để phân 2 lớp dữ liệu.

Các điểm xanh, đỏ nằm trên 2 đường biên được gọi là các support vector.



Trong bài toán không gian 2 chiều, ta giả sử đường thẳng phân chia cần tìm có phương trình là: .

Với không gian 2 chiều:

Margin giữa 2 đường thẳng được tính bằng công thức: margin =

Với không gian nhiều chiều:

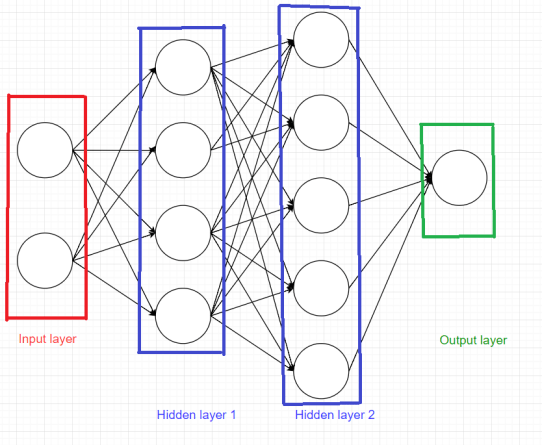
Tổng quát lên không gian nhiều chiều, cần tìm phương trình siêu phẳng có phương trình:  .  
Margin sẽ được tính bằng công thức: margin =

# 4. Neural Network

## 4.1 Neural Network là gì?

Neural là tính từ của neuron (nơ-ron – trong bộ não có chứa các nơ-ron thần kinh liên kết với nhau để xử lý nhưng hoạt động thường ngày), network chỉ cấu trúc đồ thị nên neural network (**NN**) là một hệ thống tính toán lấy cảm hứng từ sự hoạt động của các nơ-ron trong hệ thần kinh.

## 4.2 Mô hình neural



Hình 1 Mô hình reural tổng quát

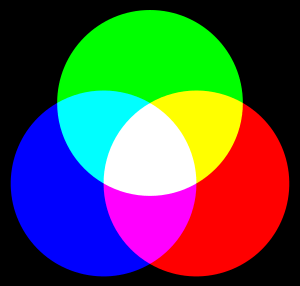
Layer đầu tiên là input layer, các layer ở giữa được gọi là hidden layer, layer cuối cùng được gọi là output layer. Các hình tròn được gọi là node.

Mỗi mô hình luôn có 1 input layer, 1 output layer, có thể có hoặc không các hidden layer. Tổng số layer trong mô hình được quy ước là số layer – 1 (Không tính input layer).

# 5. Giới thiệu về xử lý ảnh

## 5.1 Hệ màu RGB

RGB viết tắt của red (đỏ), green (xanh lục), blue (xanh lam), là ba màu chính của ánh sáng khi tách ra từ lăng kính. Khi trộn ba màu trên theo tỉ lệ nhất định có thể tạo thành các màu khác nhau.

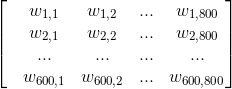


Hình 2 Hệ màu RGB

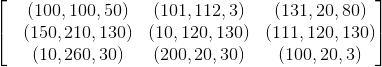
## 5.2 Điểm ảnh pixel

Pixel (hay điểm ảnh, ký hiệu px) là một khối màu rất nhỏ và là đơn vị cơ bản nhất để tạo nên một bức ảnh kỹ thuật số.

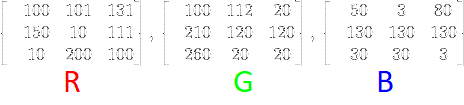
Ví dụ 1 ảnh có kích thước 800\*600px thì sẽ gồm 800\*600=480.000px có thể được biểu diễn dưới dạng ma trận 600\*800 (định nghĩ ma trận là số hàng nhân số cột).



Để biểu diễn 1 màu ta cần 3 thông số (r,g,b):



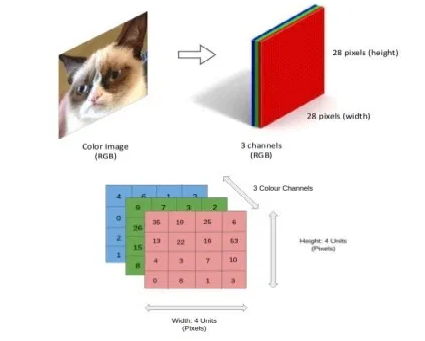
Để tiện lưu trữ và xử lý không thể lưu trong 1 ma trận như thế kia mà sẽ tách mỗi giá trị trong mỗi pixel ra một ma trận riêng.



## 5.3 Tensor

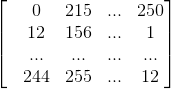
Khi dữ liệu nhiều hơn 2 nhiều thì sẽ được gọi là tensor, ví dụ như dữ liệu có 3 chiều.

Ví dụ biểu diễn ảnh màu kích thước 28\*28, biểu diễn dưới dạng tensor 28\*28\*3.



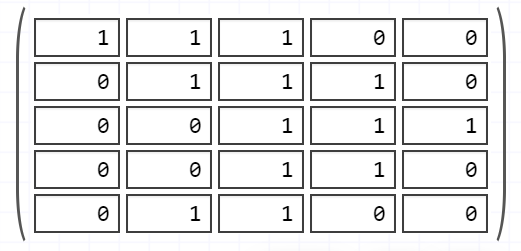
## 5.4 Ảnh xám

Tương tự ảnh màu, ảnh xám cũng có kích thước 800 pixel \* 600 pixel, có thể biểu diễn dưới dạng một ma trận kích thước 600 \* 800 (vì định nghĩa ma trận là số hàng nhân số cột). Tuy nhiên mỗi pixel trong ảnh xám chỉ cần biểu diễn bằng một giá trị nguyên trong khoảng từ [0,255] thay vì (r,g,b) như trong ảnh màu. Do đó khi biểu diễn ảnh xám trong máy tính chỉ cần một ma trận là đủ.

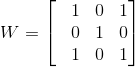


## 5.5 Phép tính convolution

Ví dụ: Cho 1 ảnh xám được biểu diễn dưới dạng ma trận A kích thước m\*n.



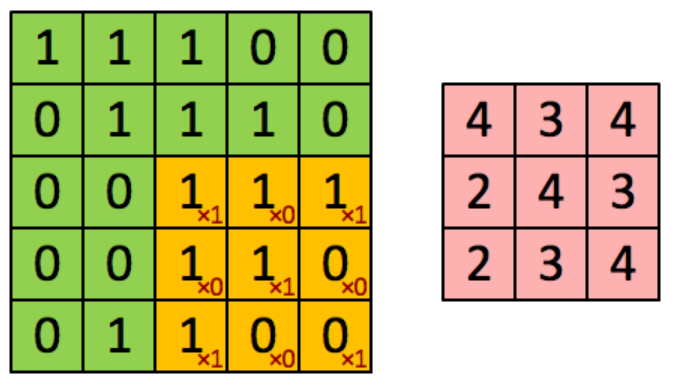
Ta định nghĩa kernel là một ma trận vuông kích thước k\*k trong đó k là số lẻ. k có thể bằng 1, 3, 5, 7, 9,… Ví dụ kernel kích thước 3\*3.



Với mỗi phần tử ​ trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A. Sau đó tính tổng các phần tử của phép tính element-wise của ma trận A và ma trận W, rồi viết vào ma trận kết quả Y.

https://i0.wp.com/nttuan8.com/wp-content/uploads/2019/02/CodeCogsEqn-4.gif?w=1400&ssl=1

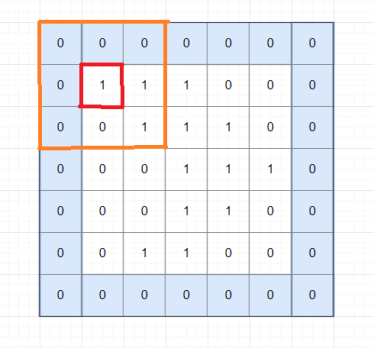
Phép tính element-wise



Kết quả thu được

### 5.5.1 Padding

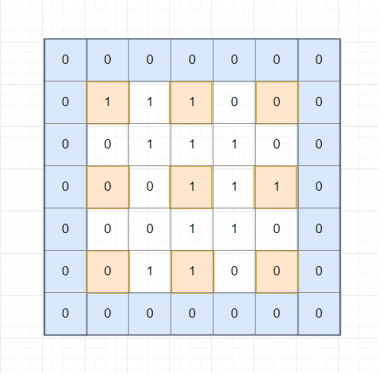
Như ở trên thì mỗi lần thực hiện phép tính convolution xong thì kích thước ma trận Y đều nhỏ hơn X. Tuy nhiên giờ ta muốn ma trận Y thu được có kích thước bằng ma trận X => Tìm cách giải quyết cho các phần tử ở viền => Thêm giá trị 0 ở viền ngoài ma trận X.



Phép tính này gọi là convolution với padding=1. Padding=k nghĩa là thêm k vector 0 vào mỗi phía của ma trận.

### 5.5.2 Stride

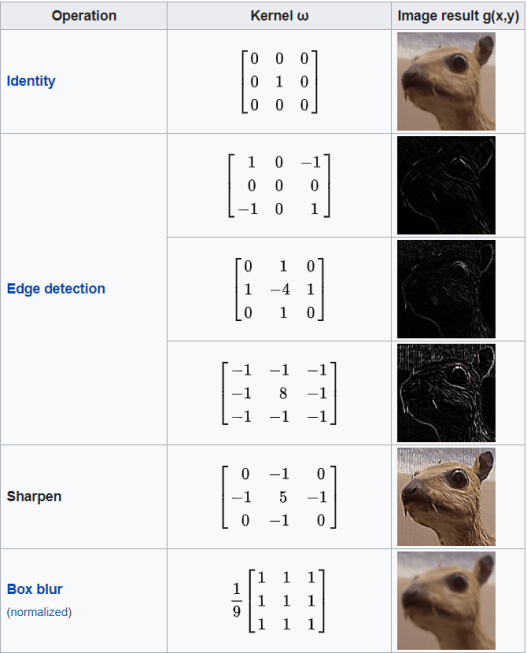
Như ở trên ta thực hiện tuần tự các phần tử trong ma trận X, thu được ma trận Y cùng kích thước ma trận X, ta gọi là stride=1.



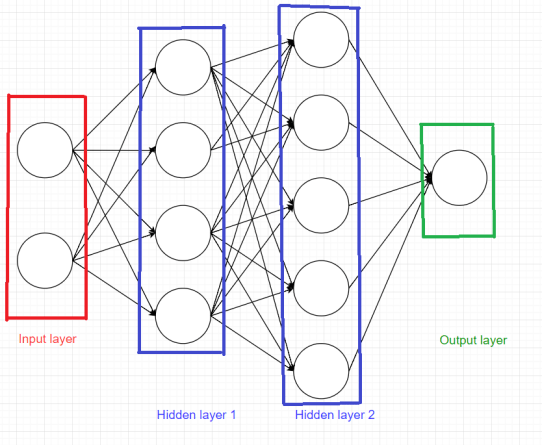
padding=1, stride=2

### 5.5.3 Ý nghĩa

Mục đích của phép tính convolution trên ảnh là làm mở, làm nét ảnh; xác định các đường. Mỗi kernel khác nhau thì sẽ phép tính convolution sẽ có ý nghĩa khác nhau.



# 6. Convolutional neural network



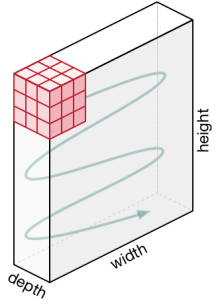
Mô hình reural network

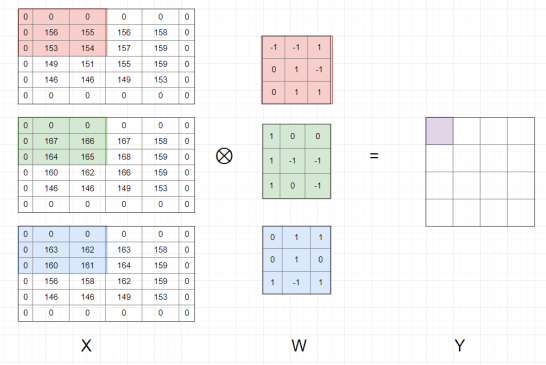
Mỗi hidden layer được gọi là fully connected layer, tên gọi theo đúng ý nghĩa, mỗi node trong hidden layer được kết nối với tất cả các node trong layer trước. Cả mô hình được gọi là fully connected neural network (FCN).

Vấn đề: ảnh màu 64\*64 được biểu diễn dưới dạng 1 tensor 64\*64\*3. Để biểu thị hết nội dung của bức ảnh thì cần truyền vào input layer tất cả các pixel (64\*64\*3 = 12288). Nghĩa là input layer giờ có 12288 nodes. Qua tất cả cái hidden layer thì số lượng node là rất lớn

## 6.1 Convolutional layer

Giái pháp: Sử dụng phép tính convolution. Tuy nhiên ảnh màu có tới 3 channels red, green, blue nên khi biểu diễn ảnh dưới dạng tensor 3 chiều. Nên ta cũng sẽ định nghĩa kernel là 1 tensor 3 chiều kích thước k\*k\*3.





Convolutional layer tổng quát:

Giả sử input của 1 convolutional layer tổng quát là tensor kích thước H \* W \* D.

Kernel có kích thước F \* F \* D (kernel luôn có depth bằng depth của input và F là số lẻ), stride: S, padding: P.

Convolutional layer áp dụng K kernel.

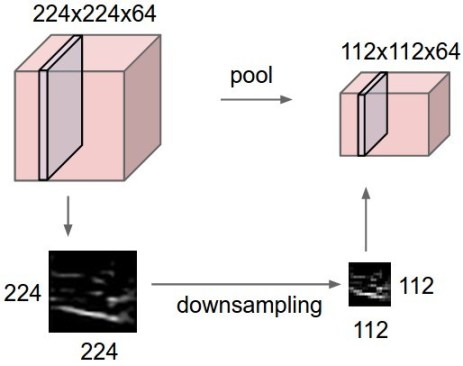
=> Output của layer là tensor 3 chiều có kích thước:

## 6.2 Pooling layer

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model.

Gọi pooling size kích thước K\*K. Input của pooling layer có kích thước H\*W\*D, ta tách ra làm D ma trận kích thước H\*W. Với mỗi ma trận, trên vùng kích thước K\*K trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh.

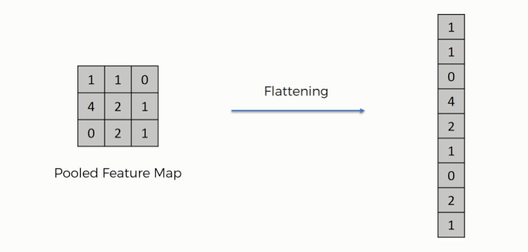
Nhưng hầu hết khi dùng pooling layer thì sẽ dùng size=(2,2), stride=2, padding=0. Khi đó output width và height của dữ liệu giảm đi một nửa, depth thì được giữ nguyên.



Có 2 loại pooling layer phổ biến là: max pooling và average pooling.

## 6.3 Fully connected layer

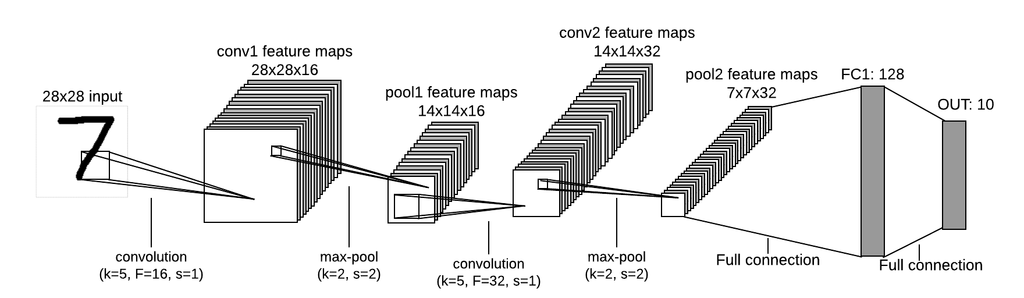
Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,…) thì tensor của output của layer cuối cùng, kích thước H\*W\*D, sẽ được chuyển về 1 vector kích thước (H\*W\*D).



## 6.4 Visualise convolutional neural network

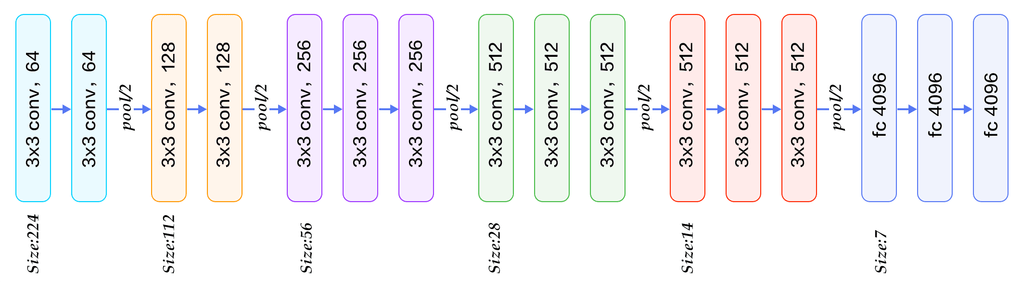
Mô hình convolutional neural network:

Input image -> Convolutional layer (Conv) + Pooling layer (Pool) -> Fully connected layer (FC) -> Output.



## 6.5 Mạng VGG 16

VGG16 là mạng convolutional neural network được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford



Phân tích:

* Convolutional layer: kích thước 3\*3, mặc định stride=1 và padding = 1 để cho output cùng width và height với input.
* Pool/2 : max pooling layer với size 2\*2
* 3\*3 conv, 64: thì 64 là số kernel áp dụng trong layer đấy, hay depth của output của layer đấy.
* Càng các convolutional layer sau thì kích thước width, height càng giảm nhưng depth càng tăng.
* Sau khá nhiều convolutional layer và pooling layer thì dữ liệu được flatten và cho vào fully connected layer.

# 6. Transfer Learning

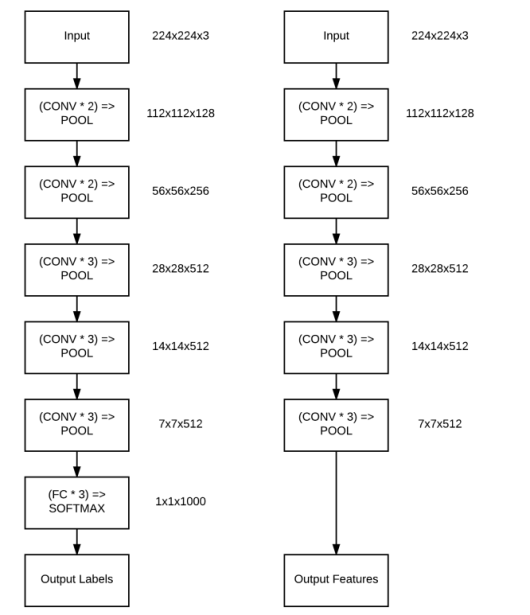
Bài toán: Ta muốn nhận diện ảnh của 17 loài hoa, mỗi loài hoa có khoảng 80 ảnh.

Sử dụng pre-trained model là VGG 16 của ImageNet. Mô hình VGG16 của ImageNet dataset, phân loại ảnh thuộc 1000 thể loại khác nhau. Nên có thể hiểu là nó đủ tổng quát để tách ra các đặc điểm của bức ảnh, cụ thể ở đây là hoa.

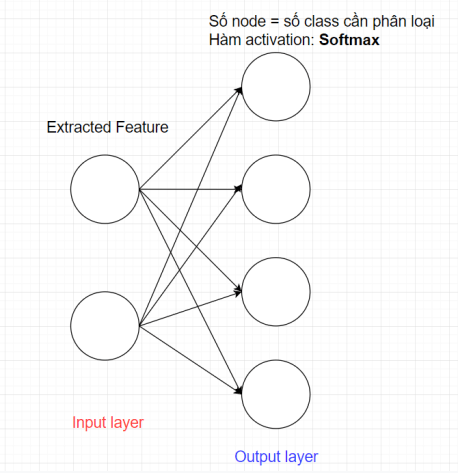
Quá trình sử dụng pre-trained model như trên gọi là transfer learning.

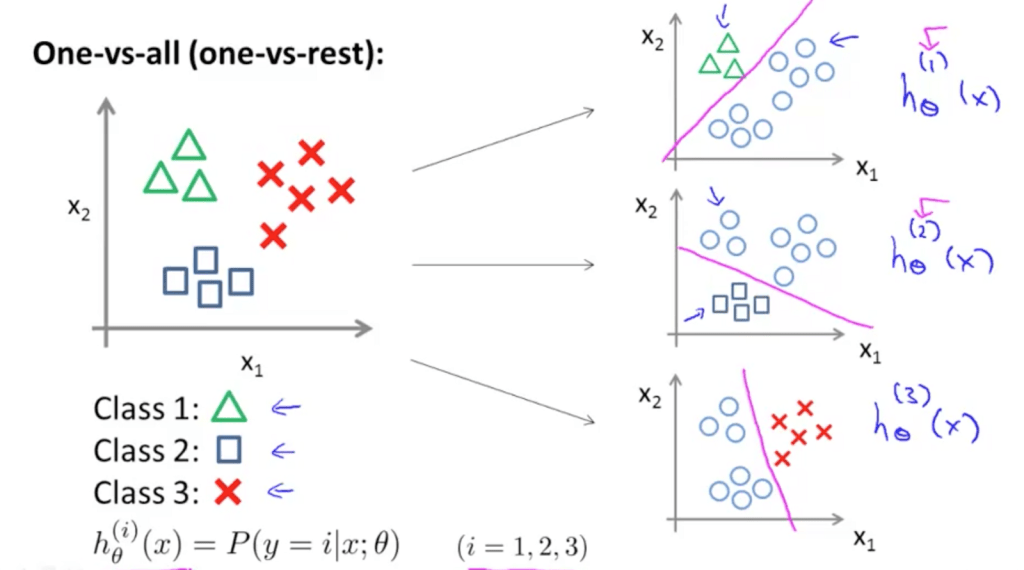
## 6.1 Feature extractor

Ta chỉ giữ lại phần ConvNet trong CNN và bỏ đi FCs. Sau đó dùng output của ConvNet còn lại để làm input cho Logistic Regression với nhiều output.

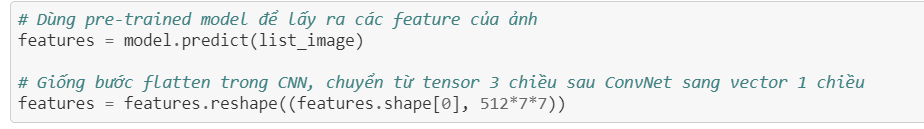


Mô hình logistic regression với nhiều output có 2 dạng:

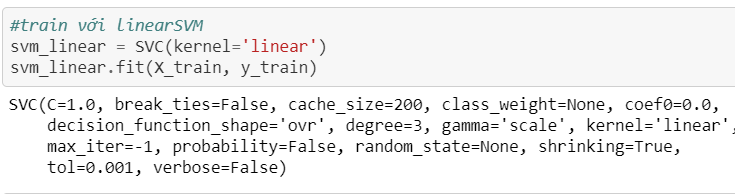
* Dạng thứ nhất là một neural network, không có hidden layer
* Dạng thứ hai giống như bài logistic regression, tức là model chỉ phân loại 2 class. Mỗi lần ta sẽ phân loại 1 class với tất cả các class còn lại.



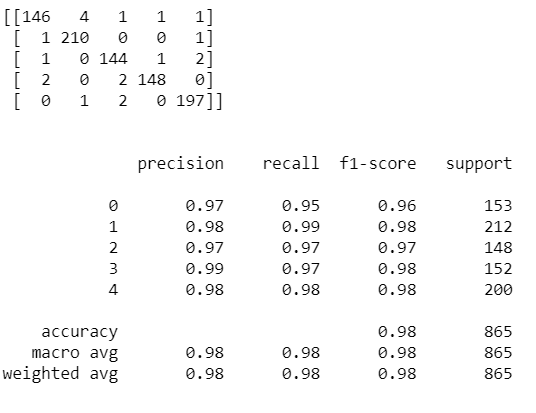
Dùng Pre-train để lấy ra các feature ảnh.



Mô hình sử dụng giải thuật SVM bằng sklearn:



Kết quả:



## 6.2 Fine tuning

### 6.2.1 Data augmentation

Data augmentation là kĩ thuật tạo ra dữ liệu training từ dữ liệu mà ta đang có.  một số kĩ thuật augmentation phổ biến với ảnh

**Flip**: Lật ngược ảnh theo chiều dọc hoặc chiều ngang

**Rotation**: Quay ảnh theo nhiều góc khác nhau

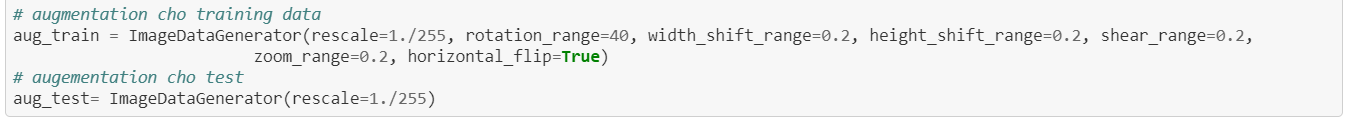
**Scale**: Phóng to hoặc thu nhỏ ảnh

**Crop**: Cắt một vùng ảnh sau đó resize vùng ảnh đấy về kích thước ảnh ban đầu

**Translation**: dịch chuyển ảnh theo chiều x, y.

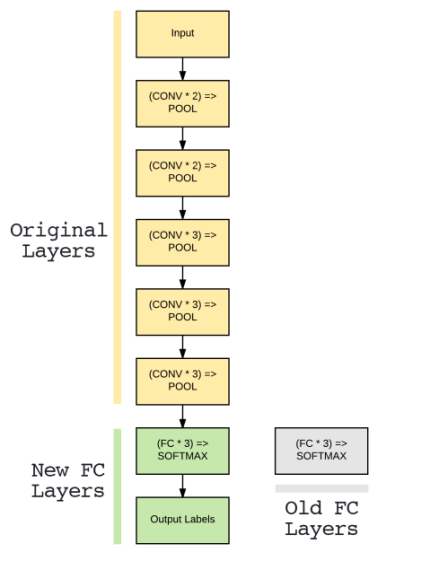
Tuy nhiên khi rotate hoặc translation thì ảnh bị những khoảng đen mà thường ảnh thực tế không có các khoảng đen đấy nên có một số cách để xử lý như: lấy giá trị từ cạnh của ảnh mới để cho các pixel bị đen, gán các giá trị đen bằng giá trị của ảnh đối xứng qua cạnh,…

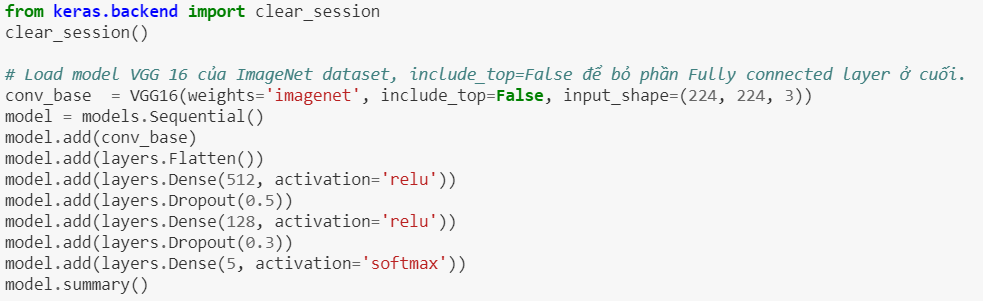
Sử dụng kĩ thuật data augmentation:



### 6.2.2 Thuật toán Fine Tuning

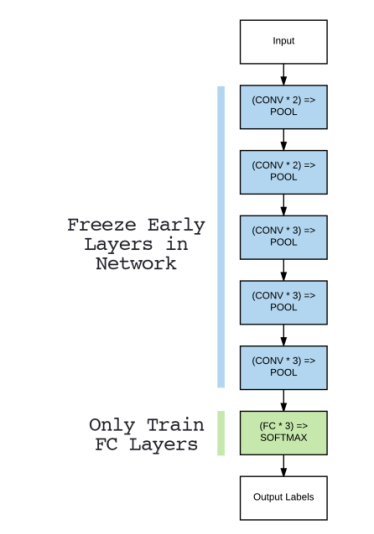
Ta chỉ giữ lại phần ConvNet trong CNN và bỏ đi FCs. Sau đó thêm các Fully Connected layer mới vào output của ConvNet.





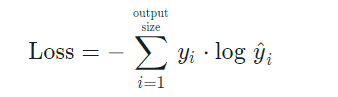
Khi train model ta chia làm 2 giai đoạn

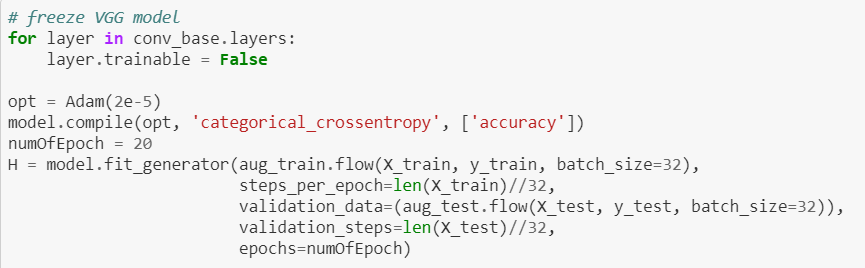
Giai đoạn 1: Vì các fully connected layer ta mới thêm vào có các hệ số được khởi tạo ngẫu nhiên tuy nhiên các layer trong ConvNet của pre-trained model đã được train với ImageNet dataset nên ta sẽ không train (đóng băng/freeze) trên các layer trong ConvNet của model VGG16. Sau khoảng 20-30 epoch thì các hệ số ở các layer mới đã được học từ dữ liệu thì ta chuyển sang giai đoạn 2.



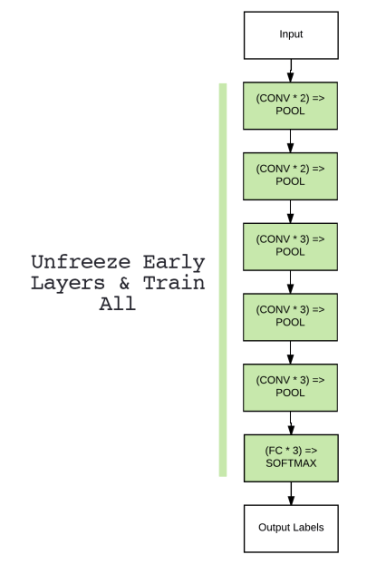
Sử dụng hàm Adam với hệ số (2e-5).

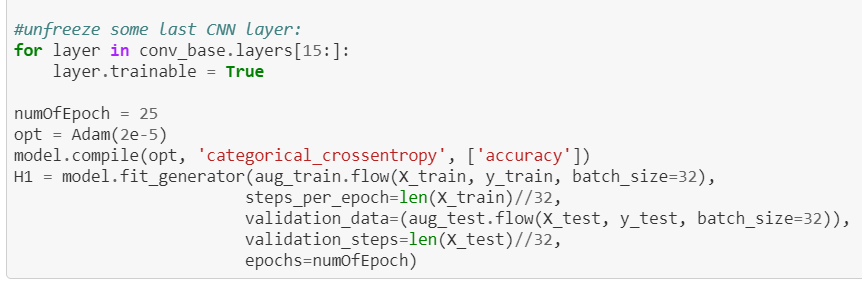
Categorical crossentropy phù hợp với việc phân loại đa lớp.





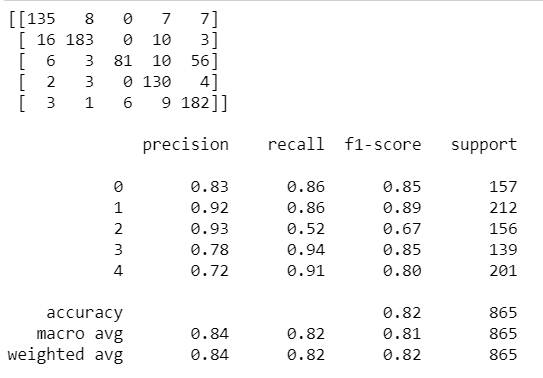
Giai đoạn 2: Ta sẽ unfreeze các layer trên ConvNet của pre-trained model và train trên các layer của ConvNet của pre-trained model và các layer mới. Bạn có thể unfreeze tất cả các layer trong ConvNet của VGG16 hoặc chỉ unfreeze một vài layer cuối tùy vào thời gian và GPU bạn có.





Accuracy của fine-tuning tốt hơn so với feature extractor tuy nhiên thời gian train của fine-tuning cũng lâu hơn rất nhiều do feature extractor chỉ lấy ra đặc điểm chung chung từ pre-trained model của ImageNet dataset cho các loài hoa, nên không được chính xác lắm. Tuy nhiên ở phần fine-tuning ta thêm các layer mới, cũng như train lại 1 số layer ở trong ConvNet của VGG16 nên model giờ học được các thuộc tính, đặc điểm của các loài hoa nên độ chính xác tốt hơn.

Kết quả





# 7. Tài liệu tham khảo

1. <https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7>
2. <https://missinglink.ai/guides/neural-network-concepts/classification-neural-networks-neural-network-right-choice/#:~:text=What%20Is%20Classification%20in%20Machine,into%20one%20of%20several%20categories.>
3. <https://neurohive.io/en/popular-networks/vgg16/>
4. <https://www.sciencedirect.com/science/article/abs/pii/S016636151730502X>

# 8. Bảng phân công công việc

|  |  |
| --- | --- |
| Thành viên | Công việc |
| Nguyễn Đức Nguyện | Tìm hiểu về Transfer Learning, Feature extractor, xây dựng ứng dụng nhận dạng loại hoa |
| Vũ Văn Hiến | Tìm hiểu về CNN, xây dựng ứng dụng nhận dạng loại hoa |
| Lê Võ Ngọc Anh | Tìm hiểu về xử lý ảnh, xây dựng ứng dụng nhận dạng loại hoa |
| Bùi Thanh Tuấn | Tìm hiểu về SVM, xây dựng ứng dụng nhận dạng loại hoa |
| Phan Văn Anh Quốc | Tìm hiểu về Transfer Learning, Fine tuning, xây dựng ứng dụng nhận dạng loại hoa |