1. **CNN**

Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection), chúng ta hãy cùng tìm hiểu về thuật toán này.

Các lớp cơ bản của thuật toán CNN:

### 1.1 Convolution Layer (Lớp tích chập)

Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vương nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

### 1.2 Bước nhảy - Stride

Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy. Hình dưới là lớp tích chập hoạt động với stride là 2.

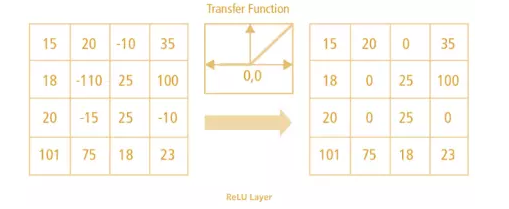
### 1.3 Padding (Đường viền)

Trong trường hợp kernel không tương xứng với hình ảnh đầu vào thì có thể lựa chọn 1 trong 2 cách sau:

* Chèn thêm số 0 vào 4 đường biên hình ảnh
* Cắt bớt những điểm không phù hợp với kernel

### 1.4 Hàm phi tuyến - ReLU

* ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: ƒ (x) = max (0, x).
* Tại sao ReLU lại quan trọng: ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.



Hình 3. Trượt bộ lọc

Hình ảnh này minh họa một bước cơ bản trong xử lý ảnh và mạng nơ-ron tích chập (CNNs), trong đó một bộ lọc được trượt qua ma trận đầu vào để tạo ra một ma trận đầu ra, với mỗi phần tử trong ma trận đầu ra là kết quả của phép toán tích chập giữa bộ lọc và một vùng tương ứng trên ma trận đầu vào. Trong trường hợp này, bộ lọc là một ma trận 3x3 chứa toàn bộ giá trị là 1, dẫn đến việc mỗi phần tử đầu ra là tổng của 9 phần tử lân cận trong ma trận đầu vào.

### 1.5 Lớp gộp - Pooling Layer

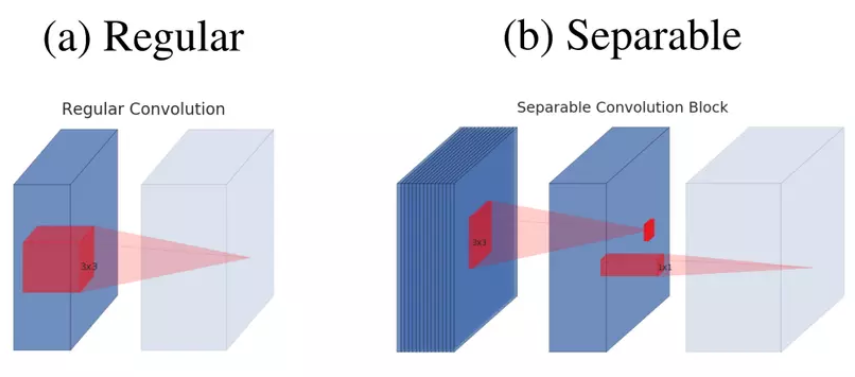
Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau:

* Max Pooling
* Average Pooling
* 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

1. **Mobilenets**

MobileNet là một mô hình kiến trúc của mạng nơ-ron tích chập (CNN) tập trung rõ ràng vào Phân loại hình ảnh cho các ứng dụng di động. Thay vì sử dụng các lớp tích chập tiêu chuẩn, nó sử dụng các lớp tích chập có thể tách theo độ sâu.



Hình ảnh này minh họa sự khác biệt giữa hai loại phép tích chập (convolution) thường dùng trong các mạng nơ-ron tích chập (CNN):

(a) Regular (Tích chập thông thường/Tiêu chuẩn):

* Hình ảnh cho thấy một bộ lọc (filter) duy nhất (khối màu đỏ, ví dụ kích thước 3x3) có chiều sâu bằng với chiều sâu của dữ liệu đầu vào (khối màu xanh lam đậm).
* Bộ lọc này trượt trên toàn bộ dữ liệu đầu vào và thực hiện phép tính tích chập trên cả không gian (chiều rộng, chiều cao) và chiều sâu (số kênh) cùng một lúc để tạo ra một điểm (hoặc một kênh) trong bản đồ đặc trưng đầu ra (khối màu xanh lam nhạt).
* Để tạo ra nhiều kênh ở đầu ra, bạn cần nhiều bộ lọc 3D như vậy.
* Đây là phép tích chập tiêu chuẩn, mạnh mẽ nhưng tốn kém về mặt tính toán và số lượng tham số.

(b) Separable (Tích chập tách biệt/Depthwise Separable Convolution):

* Đây là loại tích chập được sử dụng trong các kiến trúc hiệu quả như MobileNet. Nó chia phép tích chập thông thường thành hai bước riêng biệt:
* Bước 1: Depthwise Convolution (Tích chập theo chiều sâu): Áp dụng một bộ lọc 2D (ví dụ 3x3, các hình vuông đỏ nhỏ) cho từng kênh đầu vào một cách độc lập. Nếu đầu vào có 10 kênh, bạn sẽ có 10 bộ lọc 2D riêng biệt, mỗi bộ lọc chỉ xử lý 1 kênh. Bước này chỉ lọc về mặt không gian (spatial filtering). Khối màu xanh lam ở giữa đại diện cho kết quả sau bước này, có cùng số kênh với đầu vào.
* Bước 2: Pointwise Convolution (Tích chập điểm): Áp dụng các bộ lọc có kích thước 1x1 (hình chữ nhật đỏ nhỏ, sâu) lên kết quả của bước Depthwise. Các bộ lọc 1x1 này hoạt động trên tất cả các kênh để kết hợp thông tin từ các kênh đã được lọc độc lập ở bước trước, tạo ra các đặc trưng mới ở đầu ra (khối màu xanh lam nhạt). Bước này thực hiện việc kết hợp kênh (channel combination).
  1. **Standard Convolutions**

Standard Convolutions là tích chập tiêu chuẩn thực hiện lọc không gian và lọc kênh đồng thời. Dù MobileNet V1 chủ yếu dùng Depthwise Separable Convolutions để giảm chi phí, Standard Convolutions vẫn được dùng ở lớp đầu tiên và lớp 1x1 cuối để trích xuất đặc trưng ban đầu và kết hợp thông tin giữa các kênh.

### 1.2 Depthwise Separable Convolutions

Depthwise Separable Convolution là một kỹ thuật có thể cải thiện hiệu quả và tốc độ của các mô hình học sâu. Bằng cách chia quá trình tính toán thành hai bước, nó cho phép các mô hình học các tính năng tổng quát hơn, giảm tình trạng quá khớp và sử dụng tài nguyên tốt hơn.

### 1.2.1Deepwise Convolution

Depthwise Convolution là một phép tích chập mà trong đó mỗi kênh đầu vào được lọc bởi một bộ lọc riêng biệt,bộ lọc sâu như đầu vào và cho phép chúng ta tự do trộn các kênh để tạo ra từng phần tử trong đầu ra. Giúp giảm đáng kể chi phí tính toán so với tích chập tiêu chuẩn bằng cách tách biệt quá trình lọc không gian với quá trình kết hợp kênh.

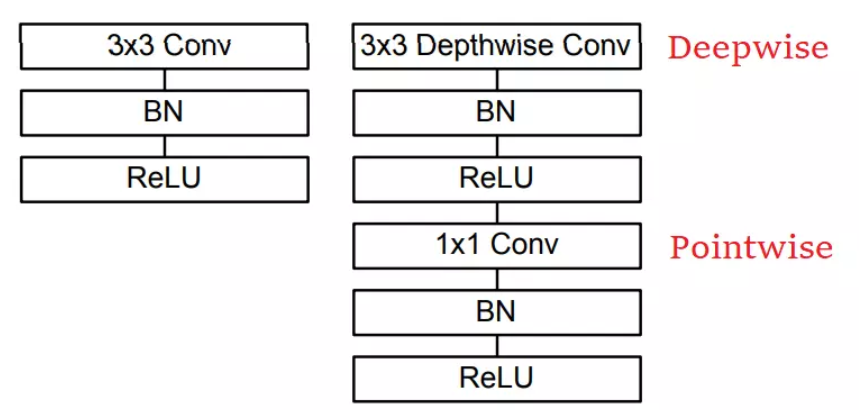
* Rõ hơn: Mỗi filter chỉ áp dụng cho một kênh của đầu vào. Nếu có 32 kênh đầu vào, thì sẽ có 32 filter độc lập, mỗi filter áp dụng lên một kênh duy nhất. Điều này giúp giảm đáng kể chi phí tính toán.

1.2.2 Pointwise Convolution

Pointwise Convolution là một loại hoạt động tích chập trong học sâu sử dụng hạt nhân 1x1. Hạt nhân này lặp qua từng điểm trong dữ liệu đầu vào, thực sự áp dụng phép biến đổi tuyến tính cho từng pixel một cách độc lập trên toàn bộ chiều sâu của đầu vào. Chiều sâu của hạt nhân bằng với số kênh trong hình ảnh đầu vào

* Rõ hơn: Sau bước Depthwise, MobileNet áp dụng convolution 1×1 (tức là mỗi filter có kích thước 1×1) để kết hợp các kênh lại với nhau, tạo ra kênh đầu ra mới.

sự khác biệt giữa standard CNN và kiến trúc MobileNet



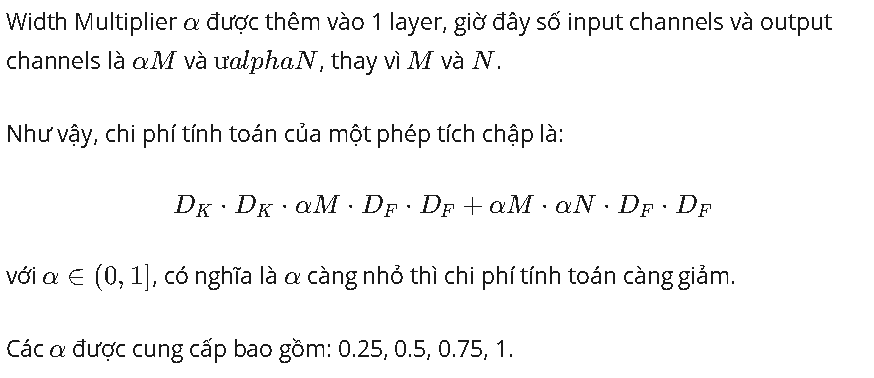
- Khối tích chập tiêu chuẩn thực hiện lọc không gian và lọc kênh đồng thời trong một bước.

- Khối Depthwise Separable Convolution tách quá trình này thành hai bước:

* Depthwise Convolution: Lọc không gian độc lập trên từng kênh.
* Pointwise Convolution: Kết hợp thông tin giữa các kênh.

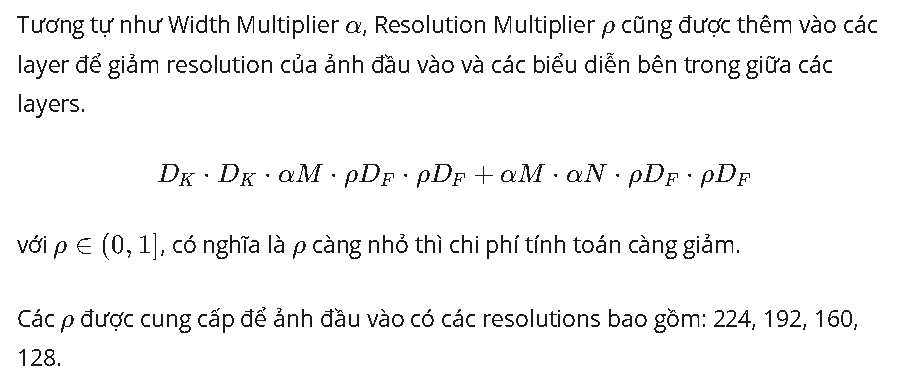
### 1.3 Width Multiplier: Thinner Models

Là một kỹ thuật để tạo ra các phiên bản "mỏng hơn" của một kiến trúc mạng CNN bằng cách nhân số lượng kênh của mỗi lớp với một hệ số α≤1. Điều này dẫn đến các mô hình có ít tham số hơn, chi phí tính toán thấp hơn và kích thước nhỏ hơn, nhưng có thể có độ chính xác thấp hơn so với mô hình gốc.



### 1.4 Resolution Multiplier: Reduced Representation

Resolution Multiplier là một kỹ thuật để giảm kích thước không gian của dữ liệu đầu vào và các bản đồ đặc trưng bên trong mạng CNN bằng cách nhân kích thước ảnh đầu vào với một hệ số ρ≤1. Dẫn đến "Reduced Representation" (biểu diễn giảm) của dữ liệu, giúp giảm chi phí tính toán và bộ nhớ, nhưng có thể ảnh hưởng đến độ chính xác của mô hình.



**Kết luận**

MobileNet sử dụng Depthwise Separable Convolutions để giảm số lượng tính toán, giảm số lượng params, đồng thời có thể thực hiện trích xuất đặc trưng một cách tách biệt trên các channel khác nhau.

1. **Mobilenet V2**

MobileNetV2 là kiến trúc mạng nơ-ron tích chập được tối ưu hóa cho các ứng dụng thị giác di động và nhúng. Nó cải thiện MobileNet ban đầu bằng cách đưa vào các khối dư ngược và nút thắt tuyến tính, mang lại độ chính xác và tốc độ cao hơn trong khi vẫn duy trì chi phí tính toán thấp.

MobileNet v2 tiếp tục sử dụng Depthwise Separable Convolutions, ngoài ra còn đề xuất thêm:

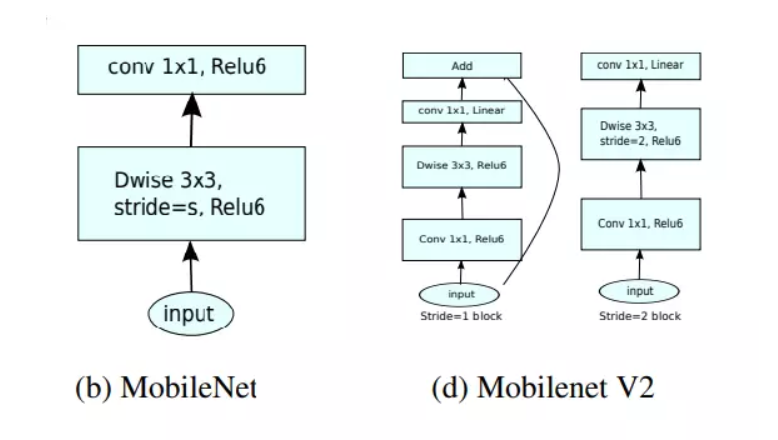
Non – Linear:

Lý do sử dụng hàm kích hoạt Non-Linear trong mạng nơ-ron là vì cái phép nhân nhiều ma trận nó không thể rút gọn thành một phép toán số duy nhất và khi mà mình sử dụng hàm kích hoạt.Vd hàm relu thì nó sẽ loại bỏ đi các gtri <0 ==> thì thấy dc nó loại bỏ đi thì nó sẽ mất đi một phần thông tin ==>họ đưa ra ý tưởng về linear bottleneck. Tức là trong đó cái việc tích chập cuối cùng của khối dư dầu ra, họ sẽ không sử dụng cái hàm phi tuyến tính nữa mà họ sẽ sử dụng hàm tuyến tính là hàm linear thôi

(nói lúc nói phần Linear Bottleneck)

Linear bottlenecks

* Inverted Residual Block (shortcut connections giữa các bottlenecks)
* Nhìn một chút về sự khác nhau giữa v1 và v2



phân tích sự khác biệt trong bức ảnh trên:

* MobileNet gồm sử dụng 1 loại blocks gồm 2 phần, Deepwise và Pointwise.
* MobileNet v2 sử dụng 2 loại blocks, bao gồm: residual block với stride = 1 và block với stride = 2 phục vụ downsizing.

Có 3 phần đối với mỗi block:

* Layer đầu là 1×1 convolution với ReLU6.
* Layer thứ hai, như cũ, là depthwise convolution.
* Layer thứ 3 tiếp tục là 1×1 convolution nhưng không có activation function. Linear được sử dụng thay vì ReLu như bình thường.

Tuy nhiên với block 1, thì phần cuối của nó được nối đầu vào + đầu ra nếu cùng kích thước => đây là inverted residual block một đặc trưng của v2. Đặc biệt, khi ở block 1 nó có 1 con đường tắt (skip connection giúp giữ lại thông tin góc nếu có thể)

1. **CODE**

sau khi train lần 1 thì thấy nó bị overfit nặng ==>

train lần 2, data augmentation và sử dụng l2 và dropout sau đó train lại thì kết quả tình trạng overfit có cải thiện chút ít nhưng ko đáng kể ==> train lần 3 freeze base model và sử dụng thêm ReduceLROnPlateau.

+ freeze model nghĩa là tui đóng băng lại toàn bộ các layers của basemodel (pretrained từ tập Imagnet) để ko cập nhật cho trọng số của chúng và chỉ train lại trên lớp mới thêm vào thôi nhé. Mục đích của là giữ nguyên được các đặc trưng bậc cao đã được học từ pretrained-model (basemodel) mà những đặc trưng này là tốt vì được huấn luyện trên bộ dữ liệu có kích thước lớn hơn và có độ chính xác cao hơn so với khởi tạo hệ số ngẫu nhiên và de giảm độ phức tạp mô hình nhé

về phần dựng web thì thực thiện tạo Flask application và sử dụng restful API để đẩy dữ liệu lên, predict và lưu lại result phần thiết kế thì gồm 3 trang html: index và intro và phần hướng dẫn.

1. **QUY TRÌNH THỰC HIỆN BÀI TOÁN :**
   1. **Mục tiêu bài toán :**  xây dựng mô hình phân loại bệnh về mắt dựa trên hình ảnh chụp võng mạc, thông qua kiến thức học sâu với kiến trúc mobilenetv2. DL được xử lý và huấn luyện bằng TensorFlow/Keras, tận dụng khả năng fine-tuning của mô hình pretrained.
   2. **Quy trình xử lý DL :**

* **Nguồn DL : thu** thập dl từ Kaggle và các datasets trên mạng theo từng loại bệnh.
* **Tiền xử lý** :
* Sử dụng ImageDataGenerator với kỹ thuật preprocess\_input từ Mobilenetv2
* Chia tập DL thành tập huấn luyện (80%) và tập test (20%)
* Áp dụng kỹ thuật shuffle, batch size 32, resize ảnh về kích thước (256, 256).
  1. **Phân tích tập dữ liệu:**

 Đã trực quan hóa số lượng ảnh giữa các tập qua biểu đồ cột.

 Vẽ biểu đồ tròn cho từng tập để quan sát sự phân bố các lớp bệnh.

 Việc này giúp đánh giá sự cân bằng dữ liệu, từ đó xác định cần thiết có tăng cường dữ liệu hay không.

* 1. **Xây dựng mô hình học sâu:**
* Kiến trúc đc sd: MobileNetv2 với trọng số pretrained từ ImageNet, loại bỏ tầng fully connected cuối.
* Các tầng thêm vào:

+ Global Average Pooling 2D

+ Dense(512 neurons) với ReLU

+ Dense (4 lớp) với softmax để phân loại

* Tối uu hoá:

+ Loss function: categorical\_crossentropy

+ Optimizer: Adam

+ Metrics: Accuracy

* 1. **Huấn luyện:**
* Áp dụng 3 callbacks:

+ ModelCheckpoint: lưu mô hình ở độ chính xác validation cao nhất

+ EarlyStopping: dừng sớm khi validation không cải thiện sau 10 epoches.

+ CSVLogger: ghi lại quá trình huấn luyện vào file CSV.