**Mobile-Net Model**

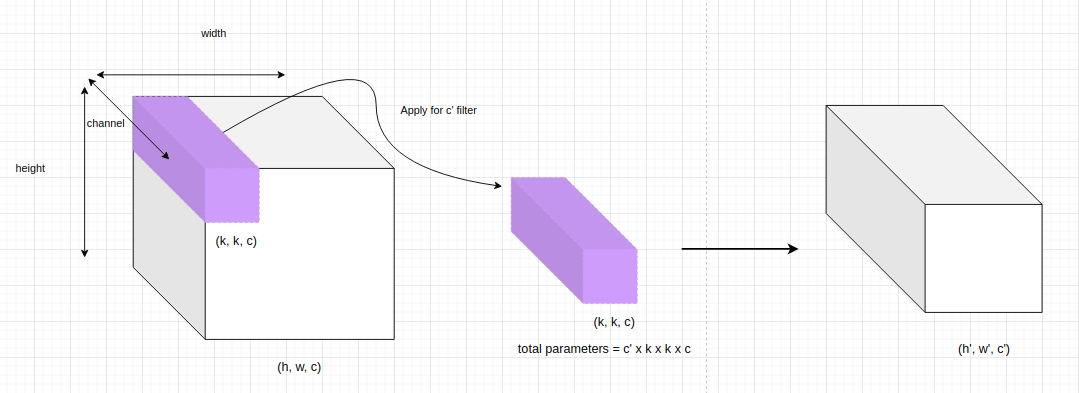
1. **Tóm lượt**

Mô đồ này sẽ hỗ trợ phát triển các ứng dụng về AI trên các thiết vị điện thoại mobile hay các ứng dụng về Iots, nhúng. Mobile-Net là kiến trúc mạng được sắp xếp hợp lý dựa trên các lớp tích chập sâu (Depth-wise convolution) và các lớp tích chập điểm (Point-wise Convolution). Mục đích là xây dựng một mô hình mạng học sâu giảm số lượng parameter cần phải học, giảm bộ nhớ máy tính, giảm độ trễ trong việc thực thi cho phù hợp ứng dụng Iots, nhưng độ chính xác vẫn cao và kiến trúc được gọi là light weight deep neutral network.

1. **Kiến trúc Mobile-Net**
2. *Lớp tích chập sâu và tích chập điểm (Depth-wise separable convolution)*

Depth-wise separable convolution là lớp tích chập chính trong kiến trúc mạng Mobile-Net, khác với lớp tích chập thông thường, thì lớp tích chập này có tác dụng dụng làm giảm đi số lượng parameter tính được, giảm kích thước bộ nhớ nhưng vẫn đảm bảo kích thước đầu ra vẫn giống như kích thước đầu ra của lớp tích chập thông thường.

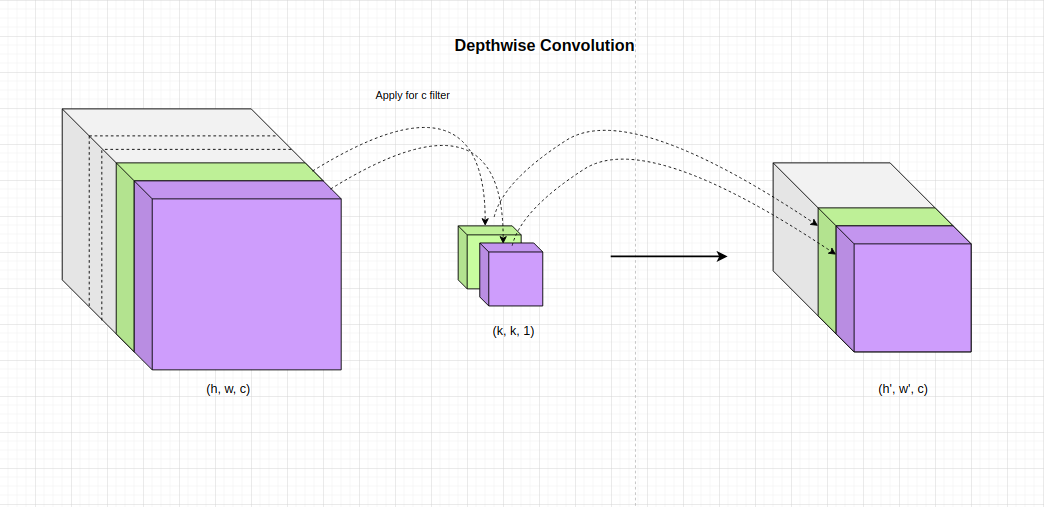
Nhắc lại kiến trúc tích chập thông thường:



Với một data có kích thước (w, h, c) (là viết tắt của weight, height và channel), sẽ được một lớp filter có kích thước (k, k, c) đương nhiên là lớp filter có chiều rộng và chiều cao phải nhỏ hơn kích thước chiều rộng và chiều cao của data nhưng nó phải cùng chiều sâu. Ta cho trượt lớp filter trên data và tính tích chập sau mỗi lần trượt, kết quả mỗi lần tích chập sẽ là một giá trị ở lớp output. Và kết quả output cho một lớp filter có kích thước là (w’, h’, 1) và ta sẽ thực hiện trượt trên data lớp c’ lớp filter nên output cuối cùng sẽ có kích thước là (w’, h’, c’). Số lượng parameter cần phải học là k × k × c × c’.

Nhưng đối lớp kiến trúc Depth-wise separable convolution sẽ chia ra làm lớp tích chập, lớp đầu tiên là tích chập theo chiều sâu (Depth-wise convolution), và lớp còn lại là tích chập điểm.

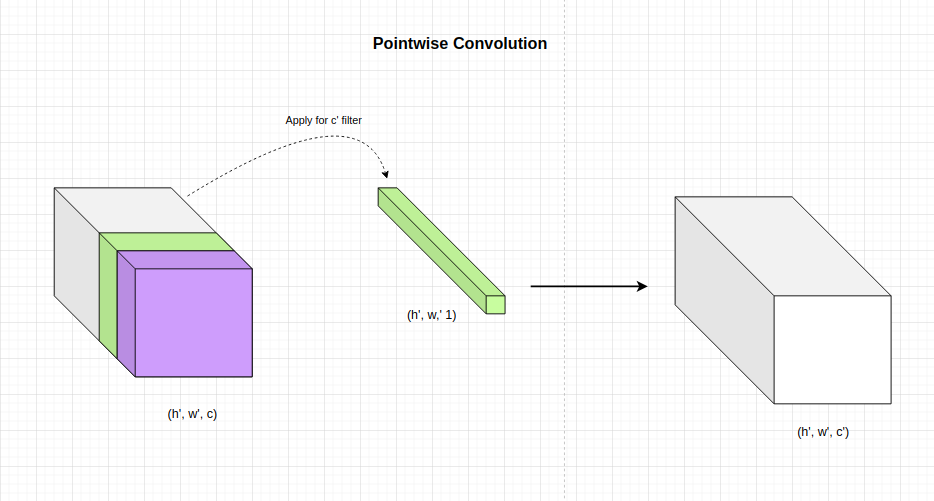
Lớp tích chập theo chiều sâu:



Lớp data đầu vào có kích thước (w, h, c) ta sẽ chia dữ liệu thành các data có c = 1, từ đó ta có c data với kích thước (w, h, 1). với mỗi data như thế này ta sẽ cho nó thực hiện tích chập lớp từng filter khác nhau với vì mỗi data trên có đặc trưng khác nhau cần trích xuất. Sau khi đã tích chập xong các ta được output sẽ là tập hợp các kết quả của từng data sau khi qua từng filter và có kích thước là (w’, h’, c) vì số lượng channel trước và sau khi thực hiện kết quả là bằng nhau.

Ta thấy được rằng với tích chập thông thường để tạo ra được một pixel trên 1 lớp output data cần thực hiện tính k × k × c nhưng tích chập theo chiều sâu ta chỉ cần tính k × k. Hay số lượng parameter cần phải học chỉ còn k × k × c thay vì k × k × c × c’.

Lớp tích chập điểm:



Lớp tích chập điểm là lớp sẽ kết hợp các kết quả của từng data mà lớp trước tạo ra để tạo thành một output hoàn chỉnh hay còn gọi là feature map. Lớp tích chập là một lớp kernel này chỉ có kích thước chiều dài và rộng là (1, 1) mục đích là nhằm không làm thay đổi kích thước chiều dài và chiều rộng của output nhưng có tới c’ lớp kernel như vậy. Để tạo ra một feature map mới với kích thước là (w’, h’, c’) giống với kích thước khi tích thập thông thường.

Ta thấy được rằng ở lớp tích chập điểm này thì phát tính ta cần thực hiện là c × c’. và số lượng parameter cần học là 1 × 1 × c × c’.

Qua đó sau 2 lớp trên thì số lượng parameter cần phải học là k × k × c + 1 × 1 × c × c’, trong khi số lượng parameter cần học của lớp tích chập thông thường là k × k × c × c’.

=> = > 0 => số lượng parameter của tích chập 2 lớp là nhỏ hơn. Dẫn đến số lượng phép tính cần tính toán là nhỏ hơn, thời gian thực thi các phép toán sẽ ít hơn.

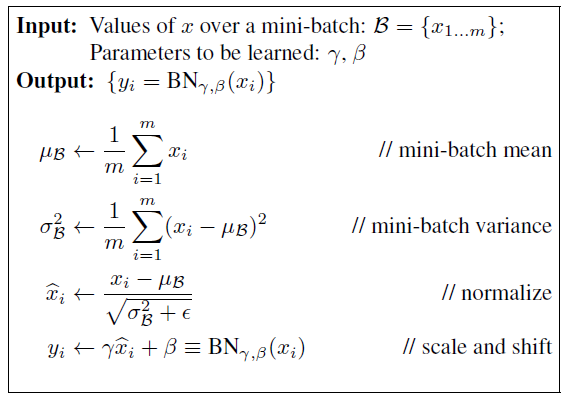
1. *Cấu trúc mạng neutral của Mobile-Net.*

Như đã nói ở trên, Mobile-Net là kiến trúc mạng được xây dựng dựa trên các lớp tích chập depth-wise separable convolution. Bên cạnh đó kiến trúc này luôn đi kèm với các hàm kích hoạt (activation function) là ReLU(Rectified Linear Unit) và BatchNorm.

Như ReLU thì chúng ta đã được học rồi nên bây giờ sẽ giải thích BatchNorm là gì?

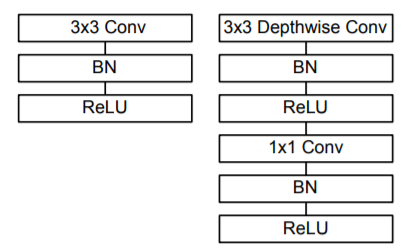
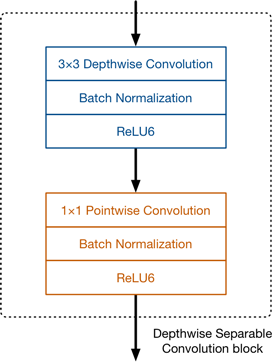
Batch Normalization là một hàm kích hoạt dùng để chuẩn hóa dữ liệu về dạng (0, 1) để việc training dể dàng hơn và tránh vanishing gradients.

Công thực tính batchNorm là:

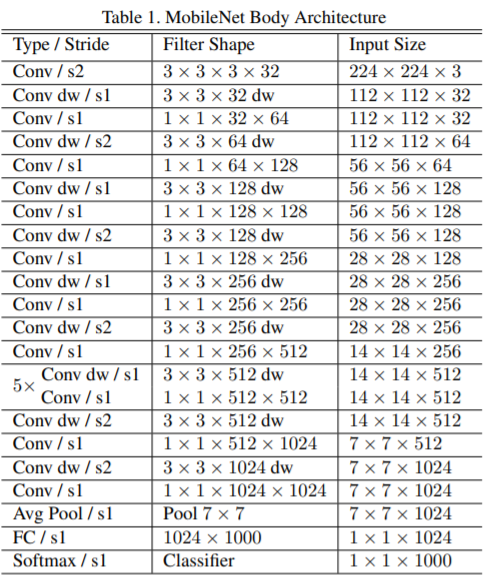


Tham số cần học là gamma và beta được sử dụng lần lượt để học các tham số liên quan tới scale và shift để có một hình dạng và vị trí tốt hơn sau khi thực hiện normalization.

Vậy nên cứ sau mỗi lớp tích chập lại sẽ có 2 lớp activation function là reLU và BatchNorm, trừ những lớp dense và fully-connected cuối. Như hình mô tả dưới đây:

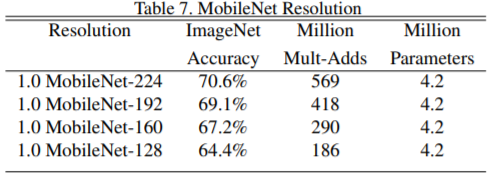
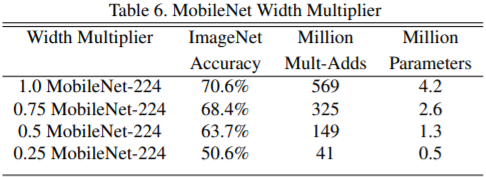
 

Và kiến trúc của model MobileNet sẽ có tổng cộng 30 lớp:



*(trade-off)*

Bên cạnh đó ta cần chú ý thêm 2 thông số áp dụng trong model này để làm giảm sô lượng phép tính toán nhưng không làm giảm quá nhiều về độ chính xác. Thông số là Width Multiplier và Resolution Multiplier. Với Width Multiplier, tại lớp tích chập depth-wise convolution , với ngõ vào data được chia ra thành N ngõ vào với N là số lượng kênh của ngõ vào data, ta có thể giảm đi N để làm giảm đi số lượng phép tính cần thực hiện, ví dụ giảm 1/4 còn 0,75N hay giảm một nữa còn 0,5N. Với Resolution Multiplier, ta sẽ làm giảm kích thức chiều dài và chiều rộng ban đầu của data, từ 224×244 thành 112×112. Sau đây là bảng thông kết cho việc này:



Như ta thầy so với phương pháp Resolution Multiplier thì dùng phương pháp Width Multiplier sẽ có số phép tính toán giảm nhiều hơn, nhưng số lượng parameter học được sẽ giảm dần ảnh hưởng khá lớn đối với độ chính xác ngõ ra. Còn phương pháp Resolution Multiplier thì tuy giảm số lượng phép tính không nhiều nhưng vẫn đảm bảo được số lượng parameters dẫn đến độ chính xác khá ổn định và cao hơn. Hai thông số trên được gọi là Shrinking Hyper-parameters

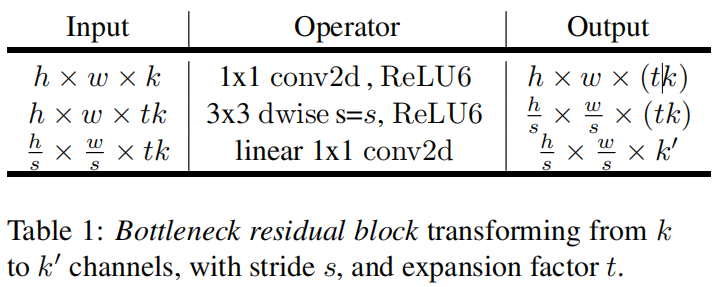
1. **Kiến trúc Mobile Net V2**
2. *Một số khái niệm.*

* Linear Bottlenecks (tắt nghẽn tuyến tính): đây là thuật ngữ được dùng nhiều trong Deep Learning, chỉ việc điều chỉnh độ sâu (số channel) của ngõ vào và ngõ ra được thắt hẹp lại hay nói cách khác là compression/encode, Việc này nhằm làm giảm số lượng parameter khi chúng ta thực hiện convolution(3×3) lúc sau.
* Skip connection, là phương pháp nhằm tránh lỗi “higher training error”, lỗi này xảy ra khi kiến trúc mạng quá lớn, lượng parameter quá nhiều dẫn đến độ chính xác cho ra bị giảm đi. Để thực hiện được điều này ta sẽ thêm vào một khối gọi là residual block, khối này sẽ lấy các output của layer trước đó vào các layer sau. Đối với model MobileNetV2 này thì residual block sẽ là convolution block có kernel là (1×1) để nén ngõ ra.
* Inverted Residuals (đảo ngược phần còn lại): thuật ngữ này bắt đầu bằng việc mở rộng chiều sâu, số channel của input, để việc tích chập có nhiều parameter hơn, sau đó thực hiện depthwise convolution (3×3), cuối cùng là thực hiện thu hẹp lại chiều sâu của đầu ra bằng cách tích chập với lớp kernel (1×1). Kiểu như code sau:

**def inverted\_residual\_block(x, expand=64, squeeze=16):  
 m = Conv2D(expand, (1,1), activation='relu')(x)  
 m = DepthwiseConv2D((3,3), activation='relu')(m)  
 m = Conv2D(squeeze, (1,1), activation='relu')(m)  
 return Add()([m, x])**

1. *Kiến trúc MobileNetV2.*

* Đầu tiên ta sẽ mô tả trước kiến trúc của một khối thắt cổ chai (Bottleneck residual block):



Trong đó:

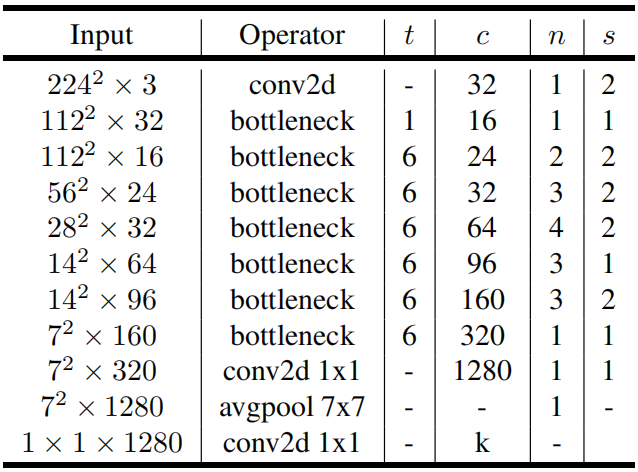
* k là số chiều sâu (số channel) ban đầu của input.
* t là số chiều mở rộng thêm.
* s sẽ là số bước của mỗi lần tích chập.
* k’ sẽ là chiều sau khi nén lại cho output.

Dể hiểu hơn với đoạn code sau đây:

**def bottleneck\_block(x, expand=64, squeeze=16):  
 m = Conv2D(expand, (1,1))(x)  
 m = BatchNormalization()(m)  
 m = Activation('relu6')(m)  
 m = DepthwiseConv2D((3,3))(m)  
 m = BatchNormalization()(m)  
 m = Activation('relu6')(m)  
 m = Conv2D(squeeze, (1,1))(m)  
 m = BatchNormalization()(m)  
 return Add()([m, x])**

Relu6 được thay cho relu: giá trị lớn nhất trả về là 6, nếu có giá trị nào lớn hơn 6 sẽ được trả về bằng 6. Mục đích lấy ngõ ra chỉ có 3 bit…..

* Kiến trúc toàn bộ mạng MobileNetV2:

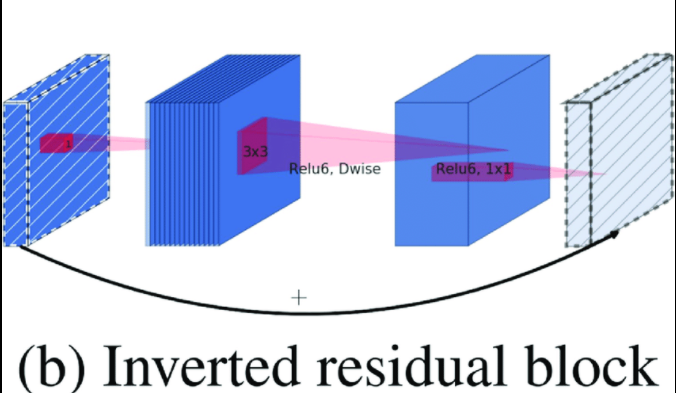


Trong đó:

* c là số lớp đầu ra được co lại sau mỗi bottleneck residual block.
* n là số lần lặp lại của lớp đó.
* s kích thước trượt cho mỗi lần tích chập.
* t là số lớp mở rộng thêm
* Kiến trúc mạng MobileNetV2 ảnh hưởng đến bộ nhớ.

**N**

**A**



**B**

Transformation function: F(x) = [A . N . B]x

A : linear transformation: → (ban đầu channel = k, sau đó nhân t vào thành n channels)

N : Relu6 . depthwise . Relu6: → (qua relu6, tích chập (33), relu6 thay đổi kích thước chiều dài và chiều rộng)

B : linear transformation: → (cuối cùng nén lại channel thành k’)

Từ đó số lượng phép tính cần phải thực thi là |s2k| + |s’2 k’| + O(max(s2, s’2)), đây cũng chính là bộ nhớ cần có của máy tính để có thể thực thi các phép tính. Ta có thể thấy bộ nhớ tính được khá thấp so với các tích chập thông thường cũng như tích chập trong mobileNetV1.

* Ảnh hưởng của Non-linearity trong bottleneck block:

Do lớp input và output có channel nhỏ nên nếu dùng các lớp phi tuyến kèm theo trong việc mở rộng cũng như nén lại ở input và output thì dữ liệu có thể sẽ bị mất làm ảnh hưởng đến độ chính xác đầu ra nên tác giả đã loại bỏ các lớp phi tuyến trong residual block, thay vào là các biến đổi tuyến tính (linear transformation)