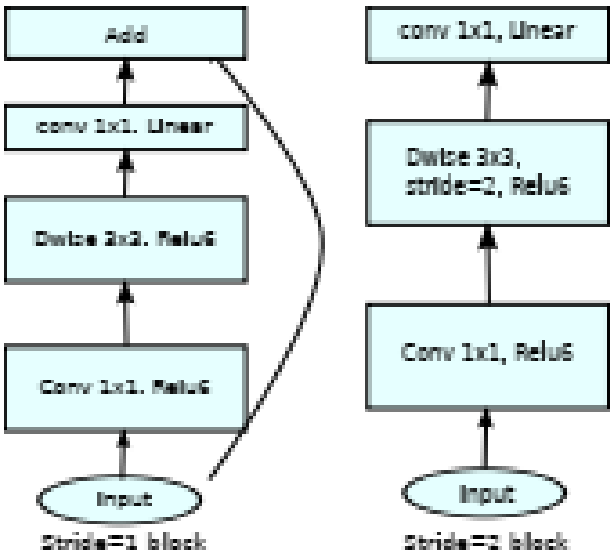
Resnet

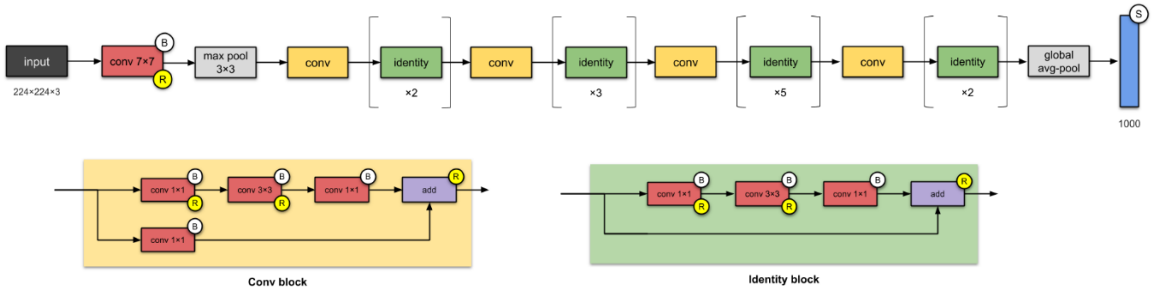
_ introducing skip connections which adds directly the input to the output. The result then goes through relu

_ two types of block: block stride 1, and block stride 2

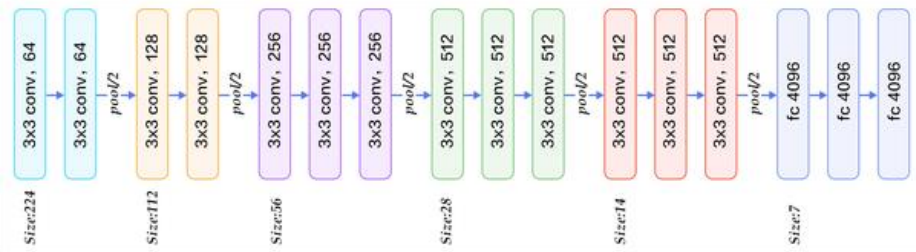
_ block stride 1 has skip connection whereas block stride 2 doesn't.



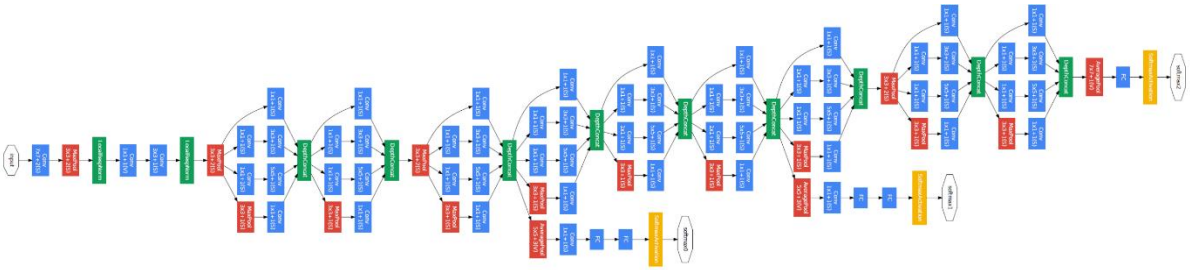
The 1st image is actually from mobilenetv2.

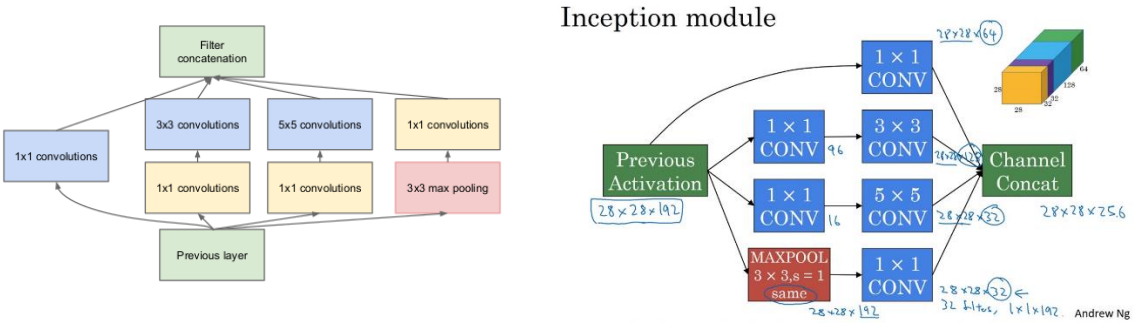


Vgg16 and alexnet don't have a special architecture. They are just deeper version of lenet-5.



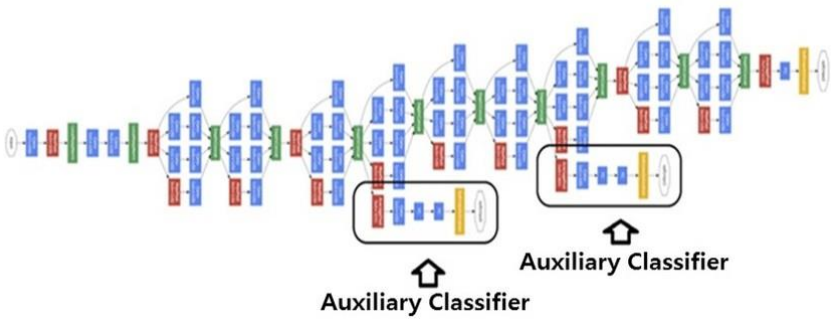
Googlenet is a super huge model which introduces the inception block.

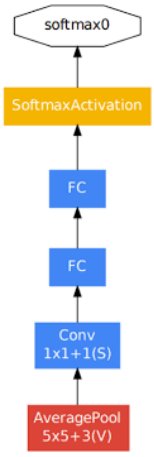




Inception module is said to reduce the dimensions of the input layer for the sake of lowering computational cost. The input is split through many reduced_dim conv blocks and are concatenates back for the next layer.

Because the network is too deep, auxiliary layers are added along to prevent the network from dying out due to vanishing gredients.





The mobilenet family are invented for use in mobile and embedded machines so it works very fast and acquires very small number of parameters compared to other models.

Mobilenetv1 introduces **depth-wise separable convolution (dw).**

Instead of sliding a 3D kernel across an image, dw uses each channel of the kernel to slide on each channel of the image.

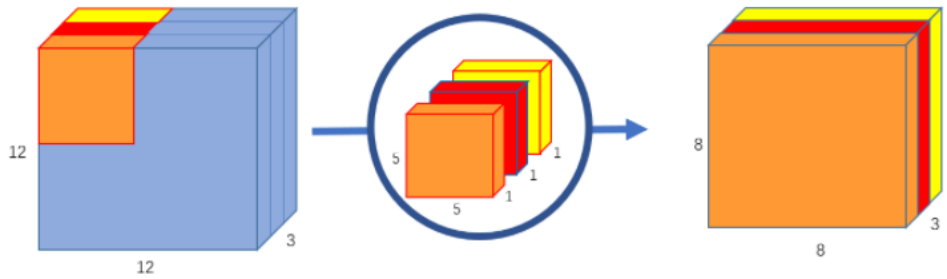


Figure: Depthwise convolution

This method is said to reduce a lot of parameters compared to normal conv and hence computional cost.

For more details, see this webpage: https://hackmd.io/@bouteille/rk-MSuYFU

To be more specific, a depth-wise separable convolution consists of two blocks: **a depthwise block and a pointwise block (pw)**. A pw block is a 1x1 kernel that iterates through every single point.

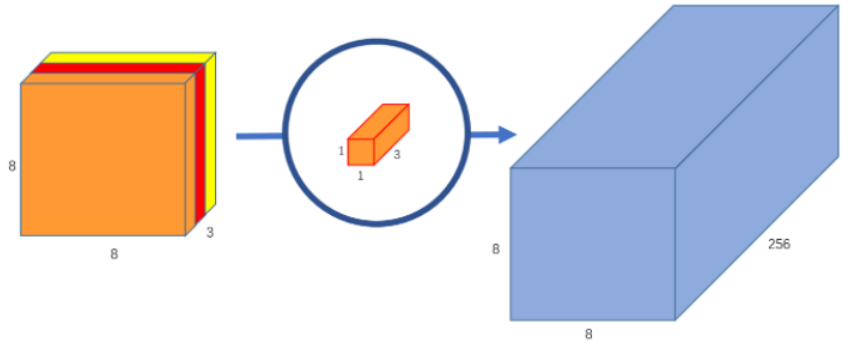


Figure: Pointwise convolution

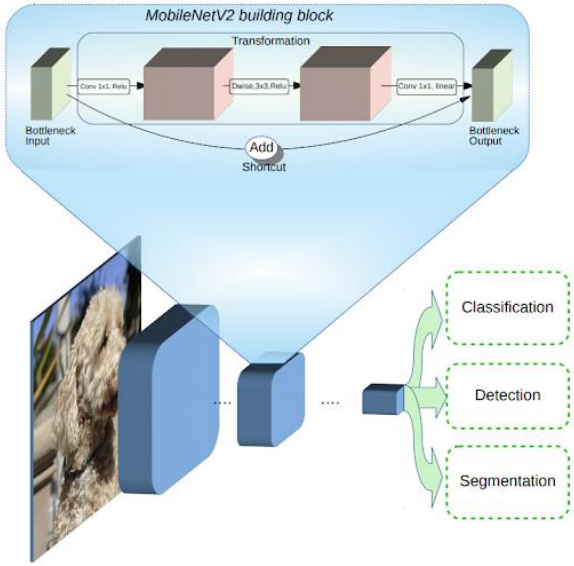
Below is mobilenetv1 architecture:

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

Mobilenetv2 is an updated version of v1 where it introduces a new type of block called **the Inverted Residual or the bottleneck.**

The residual concept is inherited from resnet. Only the inverted part is different.



The Inverted Residual block consists of three blocks: **an expansion block, a depthwise block, and a pointwise block.** They are followed in this order. Only the expansion block is the new one and the other two are inherited from the depthwise separable conv.

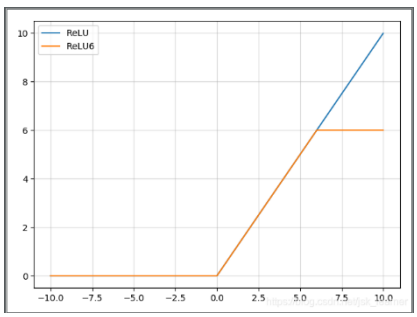
Basically what the expansion block does is to “decompress data to its full form” to extract meaningful features. And in the end, it is converted back to its original form. The “bottleneck” gets its name because the output or the flow of data of each block after going through the final conv layer (pw) is bottleneck-ed.

There are nicely done explanations expressed in here: https://machinethink.net/blog/mobilenet-v2/

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Table 2: MobileNetV2 : Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. All spatial convolutions use 3×3 kernels. The expansion factor t is always applied to the input size as described in Table 1.

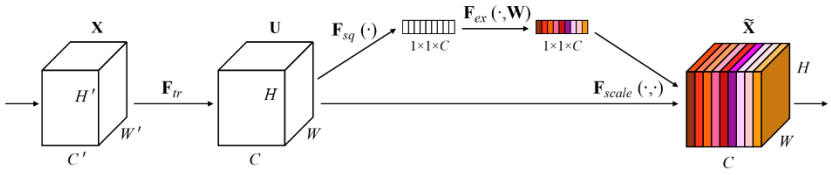
Another point is that v2 uses relu6 as activation function than relu. Relu6 is said to do better than relu because it limits the output to some maximum (i.e: 6).



Other than that, at the end of a depthwise separable conv, v2 does not use activation function (acts) because it is said to preserve the features of an image. Using too many of acts can “wash out” all features.

Mobilenetv3 adds a little more update to v2 in terms of a squeeze and excitation layer (SE layer) and using Hardswish activation function than relu6.

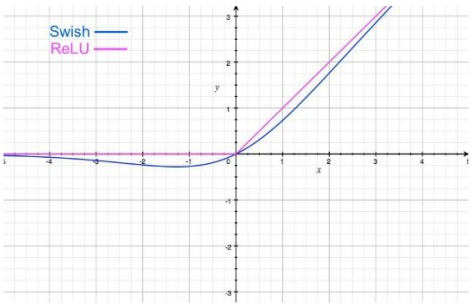
SE layer was adopted from the SE paper with the purpose of giving some attention score to the channels. Since there are channels containing more important features than others and most papers usually focus in spacial architecture than channel-wise like SE paper.

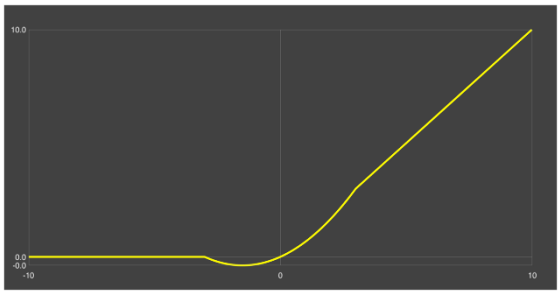


In an SE block, the input feature map first goes through some conv layers (F) and then SE layer before becoming the output. The SE layer contains a squeeze operation which is a **global average pooling operation** that flattens out the spatial dimention (HxW) to 1 and only keeping the channels. The excitation block consists of **two fully connected layers** with activation functions to find the "important channels". The result **is multiplied with the original input** to take back the dimensions.

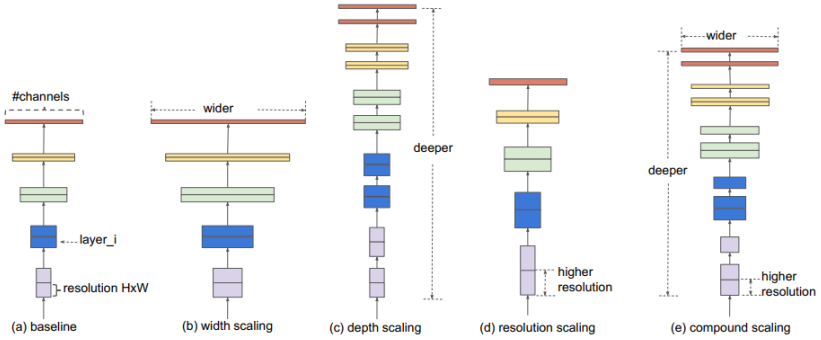
Swish activation function is **the multiplication of the input x with the sigmoid function for x.**

Hard Swish is based on Swish, but **replaces the computationally expensive sigmoid with a piecewise linear analogue**: h-swish (x) = x ReLU6 (x + 3) / 6



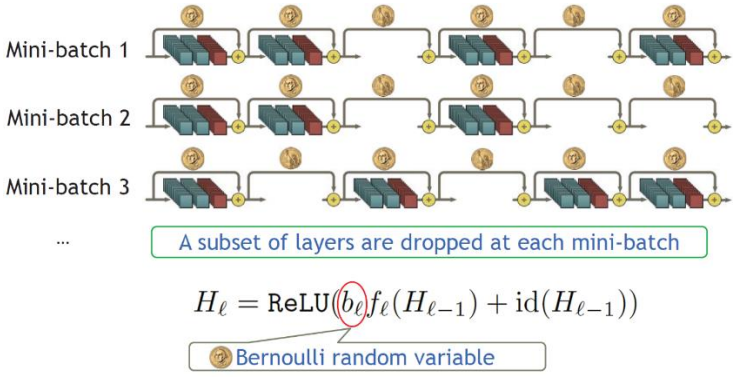


Last but not least, efficientnet combines some from one model and some from other models. It uses the **mobilenet conv block, silu activation function** which is the swish function, **and stochastic depth** from the stochastic depth paper. The focus of efficientnet is to simultaneously scale the width and depth of the network and the resolution of the image.



It is said that most papers usually focus on the architecture rather than tuning the width, the depth, and resolution.

Another important piece of knowledge that was used in efficientnet was **the stochastic depth.** Stochastic depth works like dropout, but instead of reducing the network width stochastic depth **reduces the network depth.** On each minibatch, a layer is randomly dropped but the skip connections are kept. It is said that the layers **are only dropped during training** to lessen the training time and computational cost. And during testing, the full network is used.



Some important terms and definitions:

Back propagation

Batch normalization

Dropout

Inference

Whitening