CSHO23 – DEEP LEARNING

ASSIGNMENT 2

TEAM MEMBERS:

106118001 – BALA ADITYA ANAND

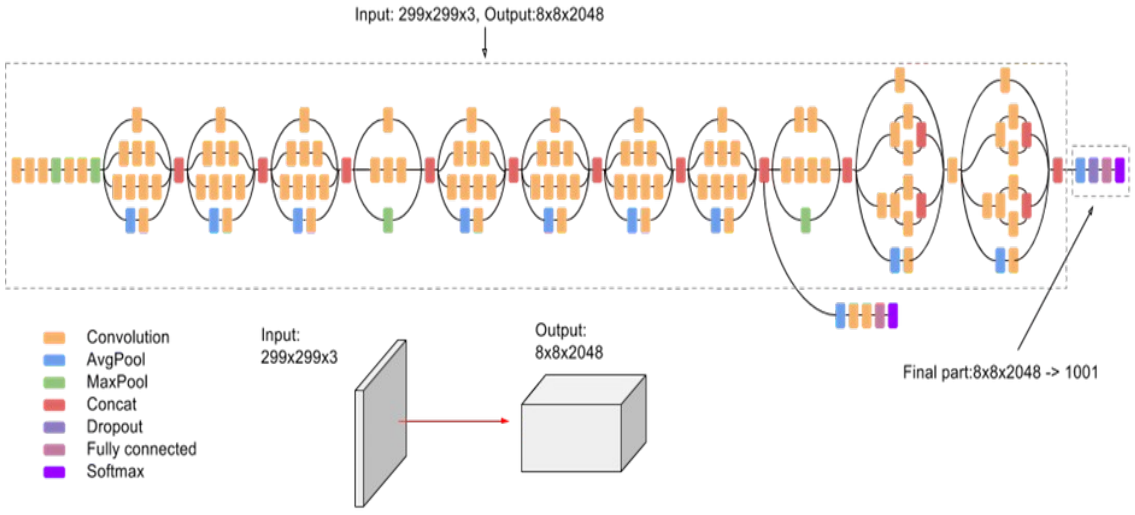
106118019 – BALAJI JAYASHRRI

106118053 – MADHAV AGGARWAL

INCEPTIONV3

A part of the Inception neural network family, InceptionV3 is a convolutional neural network. It helps with object detection and image analysis. While the model is a combination of multiple ideas over the years, its biggest inspiration is the paper "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al, focusing on spending less computational power when compared to previous Inception models. InceptionV3 was trained using a subset of the ImageNet dataset with about 1000 different classes.

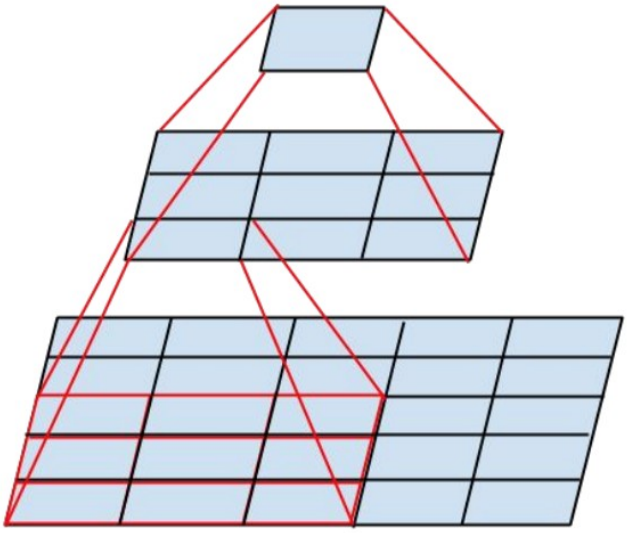
The model has both symmetric and asymmetric blocks. The model uses Batch Normalization extensively and loss is computed using SoftMax. It uses Convolutional layers, Average Pooling, Max Pooling, Dropout layers and Fully Connected Layers. InceptionV3 also uses label smoothening to prevent the largest logit from becoming much larger than all others. A diagram of the network is shown below.



InceptionV3 has shown to be more computationally efficient, in both economic costs accrued and the parameters generated when compared to other networks, like the VGGNet. However, the efficiency of the InceptionV3 is due to it being built for certain types of problems without much flexibility. Modifying it for other use cases often results in a loss of efficiency. Ideas have been put forth to modify the model such that the efficiency remains largely intact. These include factored convolutions, regularizations, dimension reduction and parallel computing.

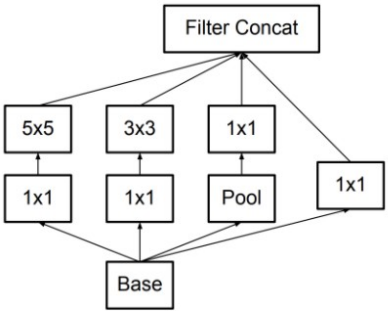
INCEPTIONV3 ARCHITECTURE:

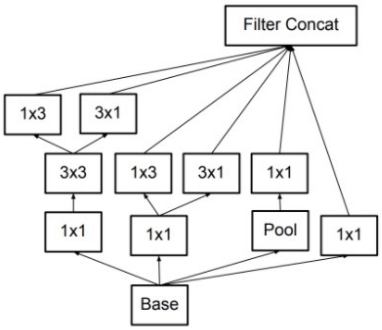
1. Factorized Convolutions – It helps with computational efficiency by reducing the number of parameters involved. It also helps with the efficiency of the network.
2. Smaller Convolutions – A large number of smaller convolutions leads to faster training as opposed to less but larger convolutions.



By using 1 layer of 5×5 filter,
number of parameters = 5×5=25
By using 2 layers of 3×3 filters,
number of parameters = 3×3+3×3=18
There is a considerable decrease in the number of parameters.

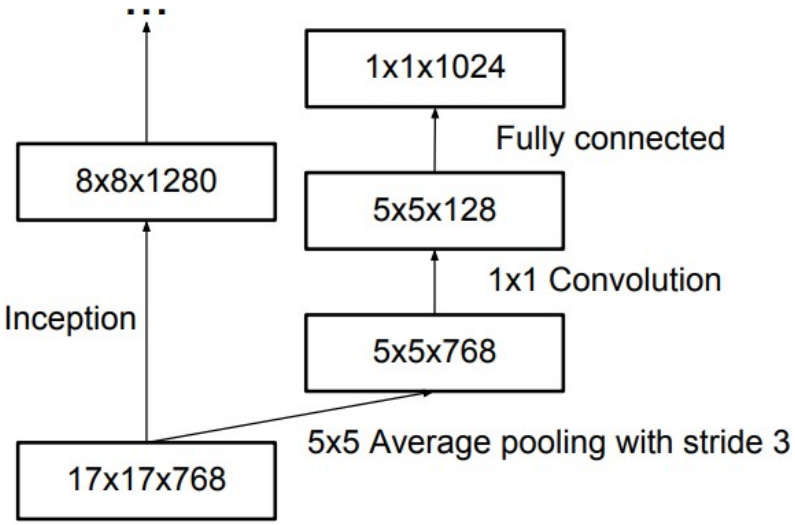
3. Asymmetric Convolutions: Asymmetric convolutions reduce the number of parameters being passed, and hence increases efficiency as compared to symmetric convolutions.



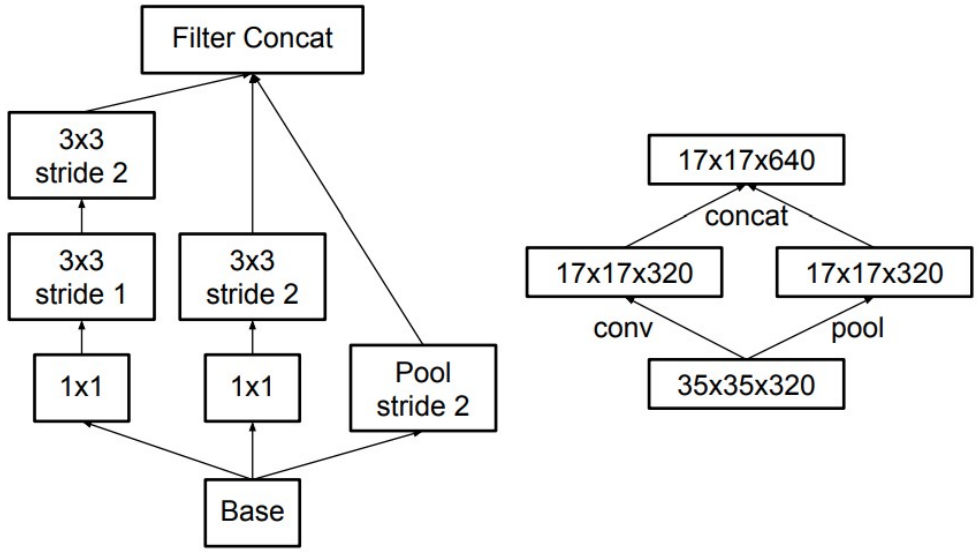


By using a 3×3 filter, number of parameters = 3×3=9
By using 3×1 and 1×3 filters, number of parameters = 3×1+1×3=6
By using two 2×2 filters, number of parameters = 2×2×2 = 8
Using smaller asymmetric filters leads to the least number of parameters.

4. Auxiliary Classifier: An auxiliary classifier is a small CNN inserted between layers that acts as a regularizer in InceptionV3.

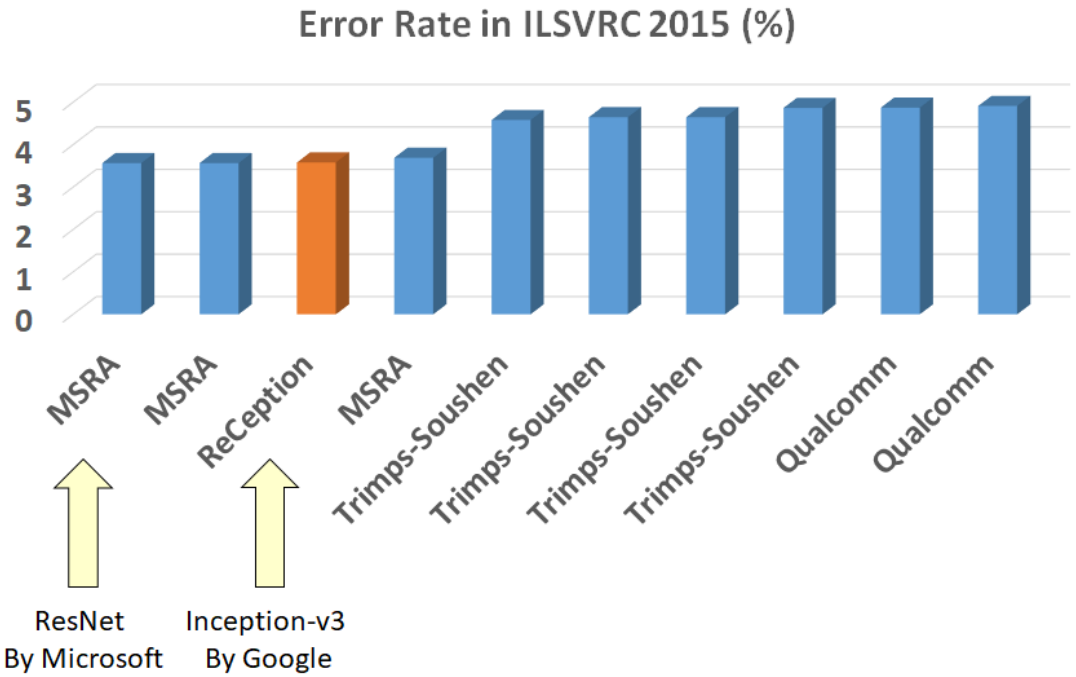


5. Grid Size Reduction – Pooling operations are usually used to reduce the grid size. However, in line with InceptionV3's focus on efficiency, a more efficient size reduction technique is proposed.



INCEPTIONV3 RESULTS:

InceptionV3 secured the first runner up place in the **ILSVRC** (ImageNet Large Scale Visual Recognition Competition) 2015. With a 42 layer deep network and fewer parameters, complexity similar to VGGNet can be achieved.



Due to being augmented with an auxiliary classifier, factorized convolutions and label smoothening, InceptionV3 is able to achieve the lowest error rates as compared to other equivalent models.

Network	**Top-1** Error	**Top-5** Error	**Cost** Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v3-basic	23.4%	-	3.8
Inception-v3-rmsprop RMSProp	23.1%	6.3	3.8
Inception-v3-smooth Label Smoothing	22.8%	6.1	3.8
Inception-v3-fact Factorized 7×7	21.6%	5.8	4.8
Inception-v3 BN-auxiliary	**21.2%**	**5.6%**	4.8

MOBILENETV2

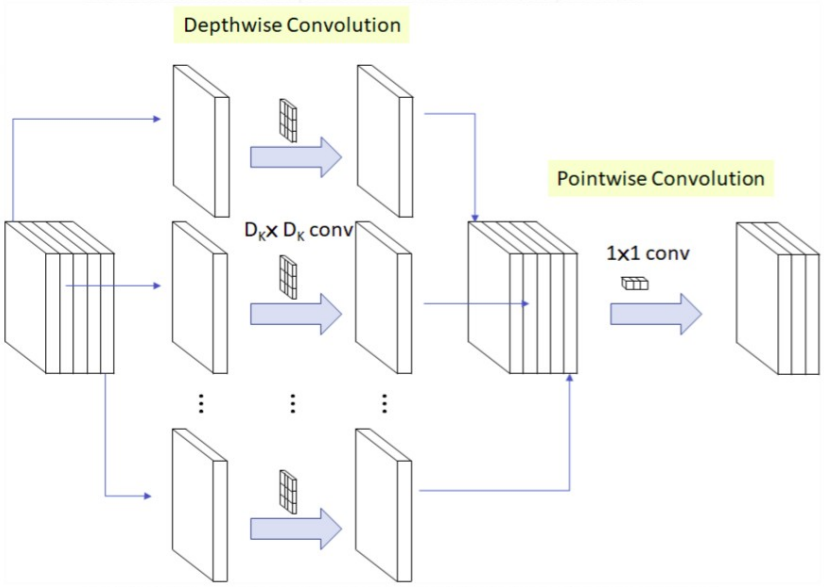
A part of the MobileNet neural network family, MobileNetV2 is a convolutional neural network. It helps with object detection and image analysis using a module called SSDLite. The motivation for the MobileNet was the need to build a scalable CNN which could deliver good performance on mobile devices, hence the name MobileNet. When compared with other similar models, such as the Inception model datasets, MobileNet is a streamlined architecture trying to achieve two fundamental goals in order to build mobile-first computer vision models:

Smaller model size: Fewer number of parameters

Lower complexity: Fewer multiplication and addition operations

Following these principles, MobileNetV1 is a class of small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases with higher accuracy. They are based on a streamlined architecture to build lightweight deep neural networks that can have low latency for mobile and embedded devices.

MobileNetV2 is the second release in the series of MobileNets. A simplified diagram of the network is shown below.



MobileNetV2 built on some of the ideas of its predecessor and incorporated new ideas to optimize the architecture for tasks such as classification, object detection and semantic segmentation. From the architecture standpoint, MobileNetV2 introduced two new features to the architecture:

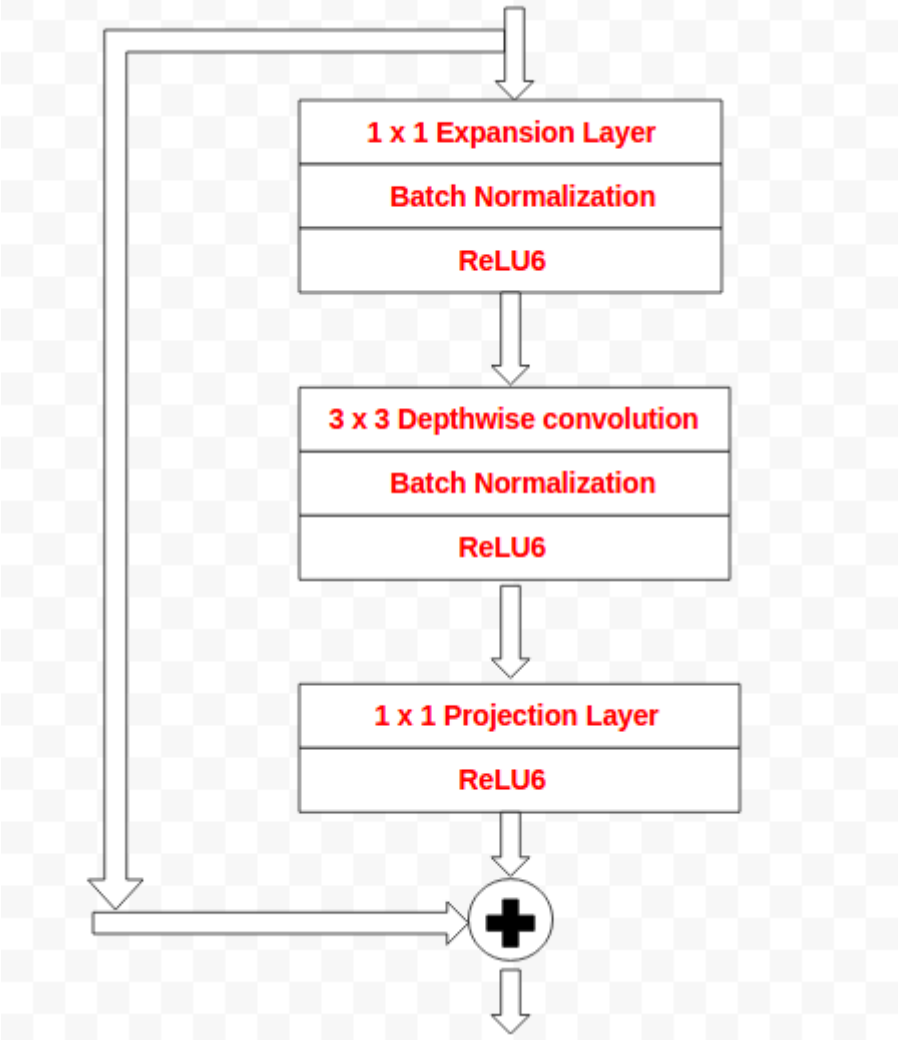
1) Linear bottlenecks between the layers

2) Shortcut connections between the bottlenecks.

The core idea behind MobileNetV2 is that the bottlenecks encode the model's intermediate inputs and outputs while the inner layer encapsulates the model's ability to transform from lower-level concepts such as pixels to higher level descriptors such as image categories. Finally, as with traditional residual connections, shortcuts enable faster training and better accuracy.

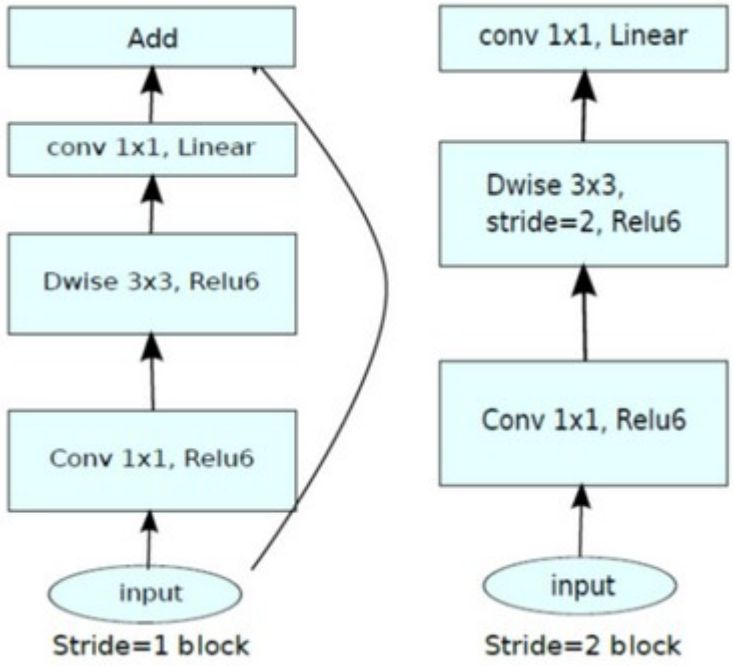
MOBILENETV2 ARCHITECTURE:

1. Inverted Bottleneck Block – MobileNetV2 uses deep separable convolution along with a bottleneck residual block to improve latency.



i. There are 3 convolution layers in the bottleneck residual block, each being a depth wise convolution layer, followed by a 1 x 1 point to point convolution layer, which makes the number of channels smaller. This layer is also called the bottleneck layer since it reduces the amount of data flowing through it.
ii. The first layer is a 1 x 1 expansion layer that increases the number of channels that flow through it. It does the opposite of the projection layer. The data gets expanded based on an expansion factor, the default of which is 6.
iii. ReLU6 is used as an activation function, which is expressed as min(max(x, 0), 6).
iv. Each layer has a batch normalization layer and activation function (ReLU6) except the project layer. The projection layer has only batch normalization because the output from the projection layer is of low dimension.

2. There are mainly two types of blocks, one being a bottleneck block with stride 1 and the other with stride 2. There are 3 layers for each of the blocks as mentioned above. If a block of stride 2 is used for depthwise convolution, the bottleneck block would not have a residual connection.



Input = $h*w*k$
By using 1 layer of 1x1 2-D Convolution,
Output = $h*w*tk$

By using 1 layer of 3×3 filter,
Depthwise Convolution with stride = 2
Input = $h*w*k$
Output = $h/s*w/s*tk$

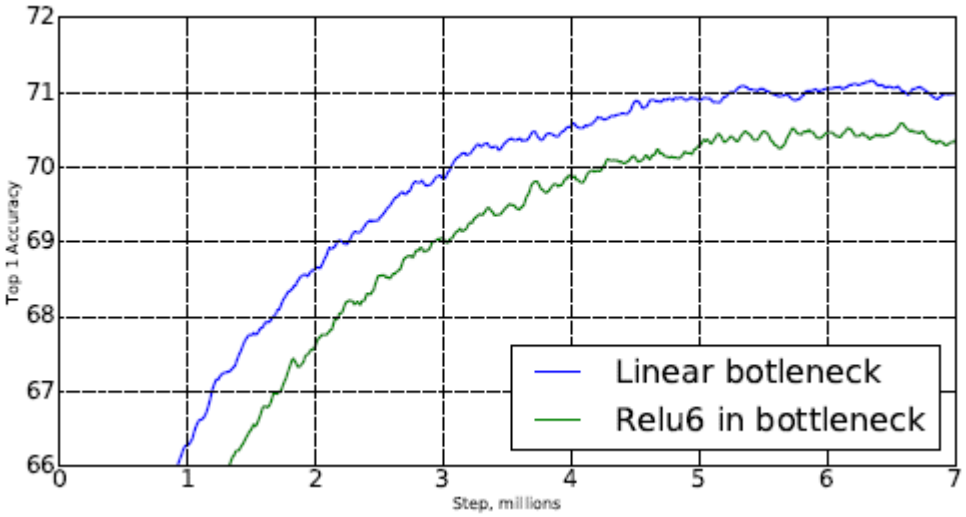
Input = $h/s*w/s*tk$
Linear 1x1 Conv2D layer
Output = $h/s*w/s*k'$

The expansion factor is usually 6. If the input layer has 64 layers, output will have 64*6 = 384.

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	-	

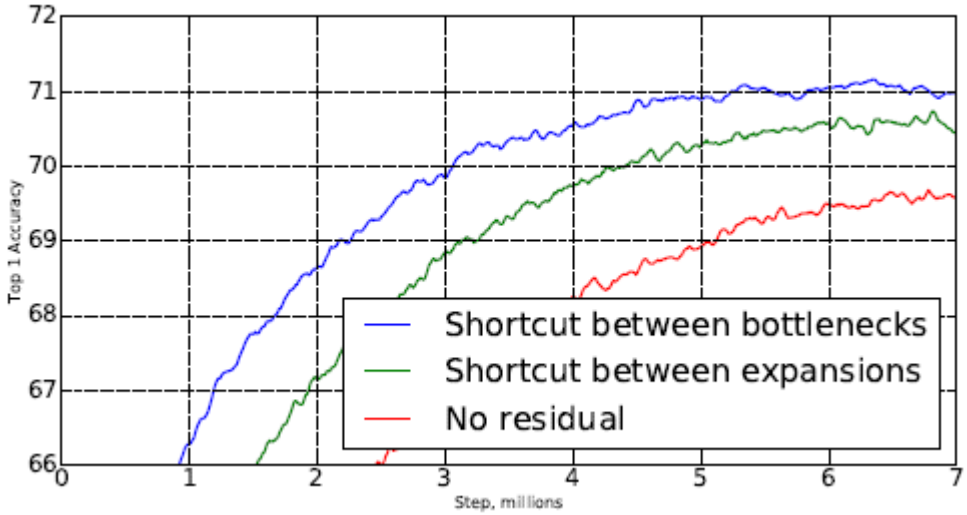
Here **t** is the expansion factor, **c** is the number of output channels, **n** is the repeating number and **s** is the stride.

MOBILENETV2 RESULTS:
1. IMPACT OF LINEAR BOTTLENECKS



CONCLUSION Performs better without Relu6 activation.

2. IMPACT OF SHORTER CONNECTION



CONCLUSION Performs better when the shortcut is between bottlenecks.

PERFORMANCE:

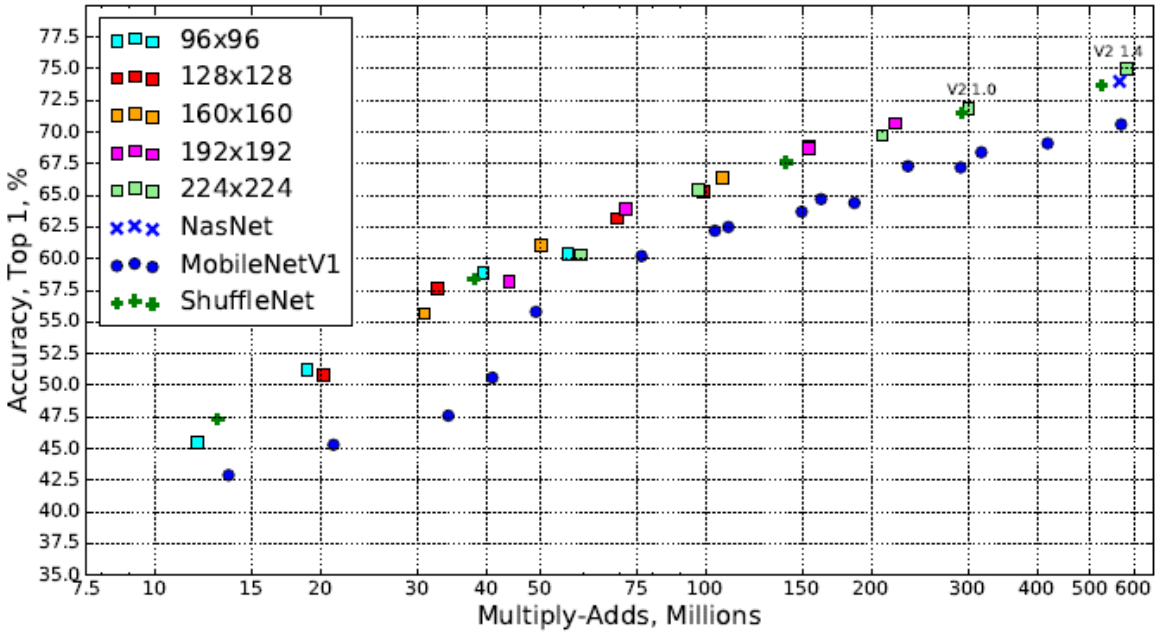
Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

1. MobileNetV2 outperforms MobileNetV1 and ShuffleNet(1.5) with comparable model size and computational cost.

	Params	MAdds
SSD[34]	14.8M	1.25B
SSDLite	**2.1M**	**0.35B**

2. SSDLite is introduced in MobileNetV1 by replacing normal convolutions with depthwise separable convolution. It reduced the parameter count and computational cost.
3. MobileNetV2 + SSDLite achieves competitive accuracy with significantly fewer parameters and smaller computational complexity. Also, the inference time is faster than MobileNetV1.

Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms


96x96
128x128
160x160
192x192
224x224
NasNet
MobileNetV1
ShuffleNet
V2 1.4
V2 1.0
Accuracy, Top 1, %
Multiply-Adds, Millions
77.5
75.0
72.5
70.0
67.5
65.0
62.5
60.0
57.5
55.0
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
7.5
10
15
20
30
40
50
75
100
150
200
300
400
500
600