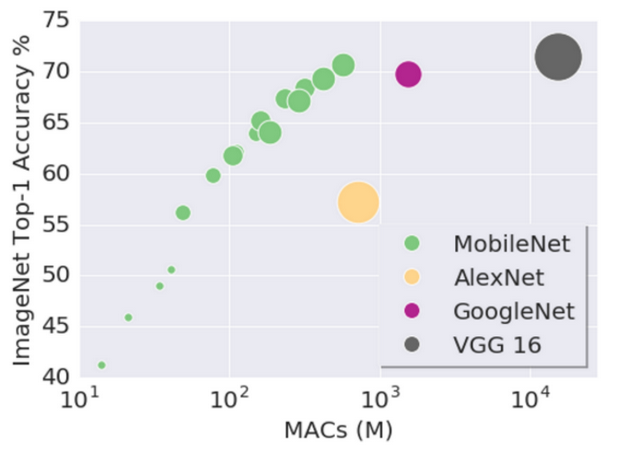
**Mobilenets (Image classification):**

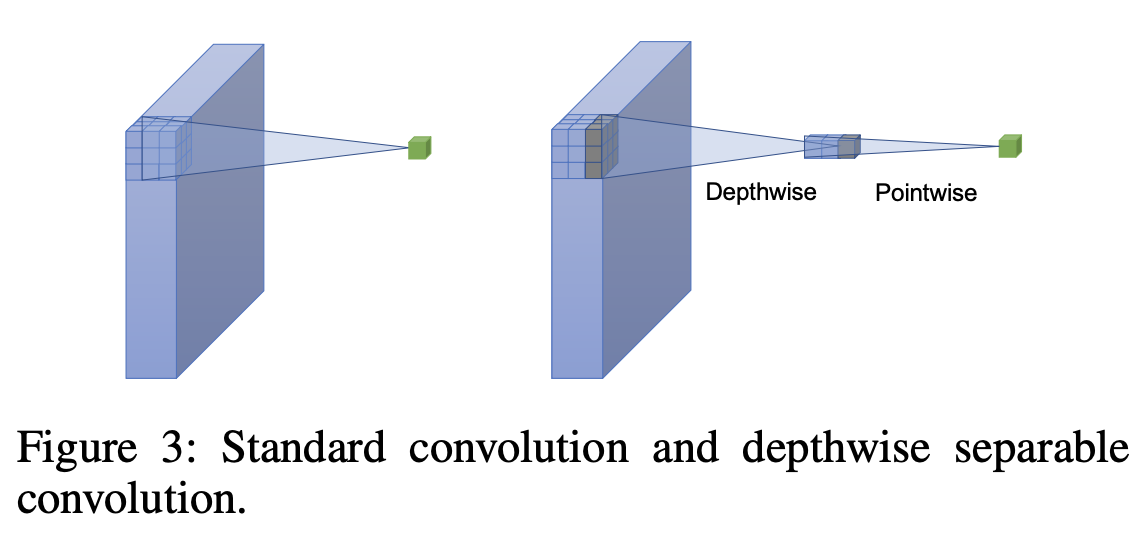
Mobilenets are efficient models that are used for mobile and embedded vision applications. MobileNets are built on a simplified design that creates lightweight deep neural networks using depth-wise separable convolutions. [1] It only requires roughly one-eighth of the computing cost with the depth-wise separable convolution and suffers from a minor accuracy loss. MobileNet also offers two hyper-parameters for the trade-offs of computing cost, size of model parameters, and accuracy. Those are denoted by α and ρ , named width multiplier and resolution multiplier, respectively. Although adjusting α or ρ can significantly reduce the computational cost or parameter size of models, it always reduces accuracy. [2] When deployed on mobile devices, the MobileNet makes use of light-weight depth-wise and point-wise convolution layers to maximize network efficiency.[3] MobileNet might be utilized in object detection, fine grain classification, face recognition, large-scale geo localization etc. Mobilenets are small and low-latency convolutional neural network which also have other advantages like it can reduce the size of the network – 17MB, it can lessen the number of parameters -4.2 million , it is faster in overall performance and are beneficial for cellular applications and so on. Though it has many benefits it is slightly less accurate than other contemporary networks.[4]

MobileNet is a CNN class provided as open source by Google, so it gives us a great starting point for training our classifiers, which are incredibly small and incredibly fast. The speed and power consumption of the network are proportional to the number of MAC (Multiply-Accumulate), which is a measure of the number of merged multiply and add operations.[5]

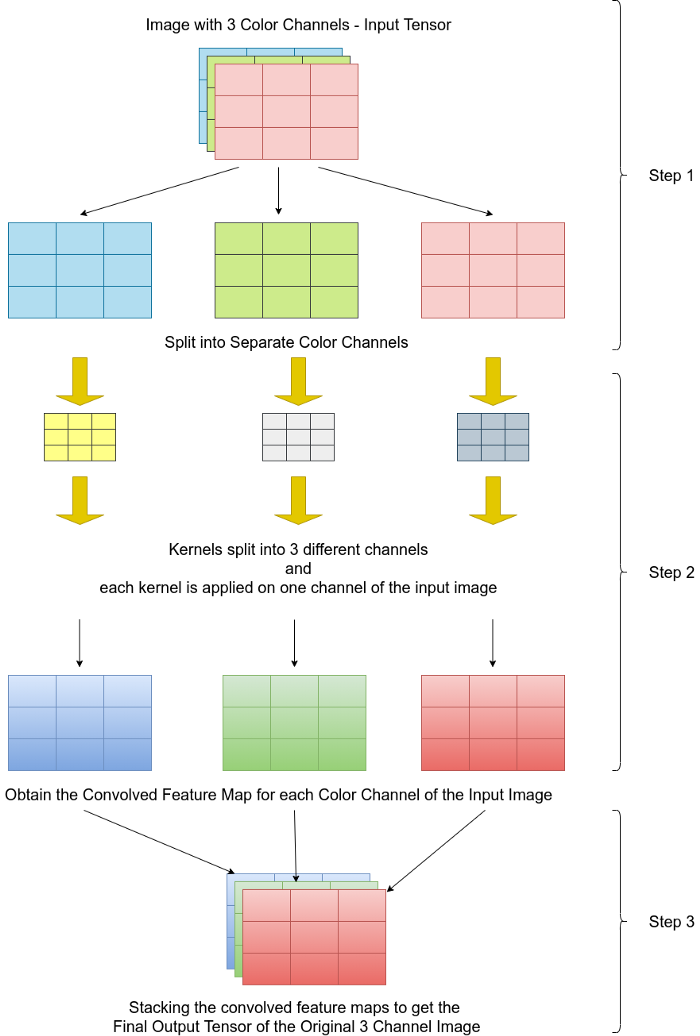


**Depth-wise Convolution:**

While standard convolution performs the channel-wise and spatial-wise computation in a single step, depth-wise separable convolution splits the calculation into two steps: depth convolution applies a single convolution filter per input channel, and pointwise convolution creates a linear combination of the output of the depth-wise convolution.[6] The difference is shown below:

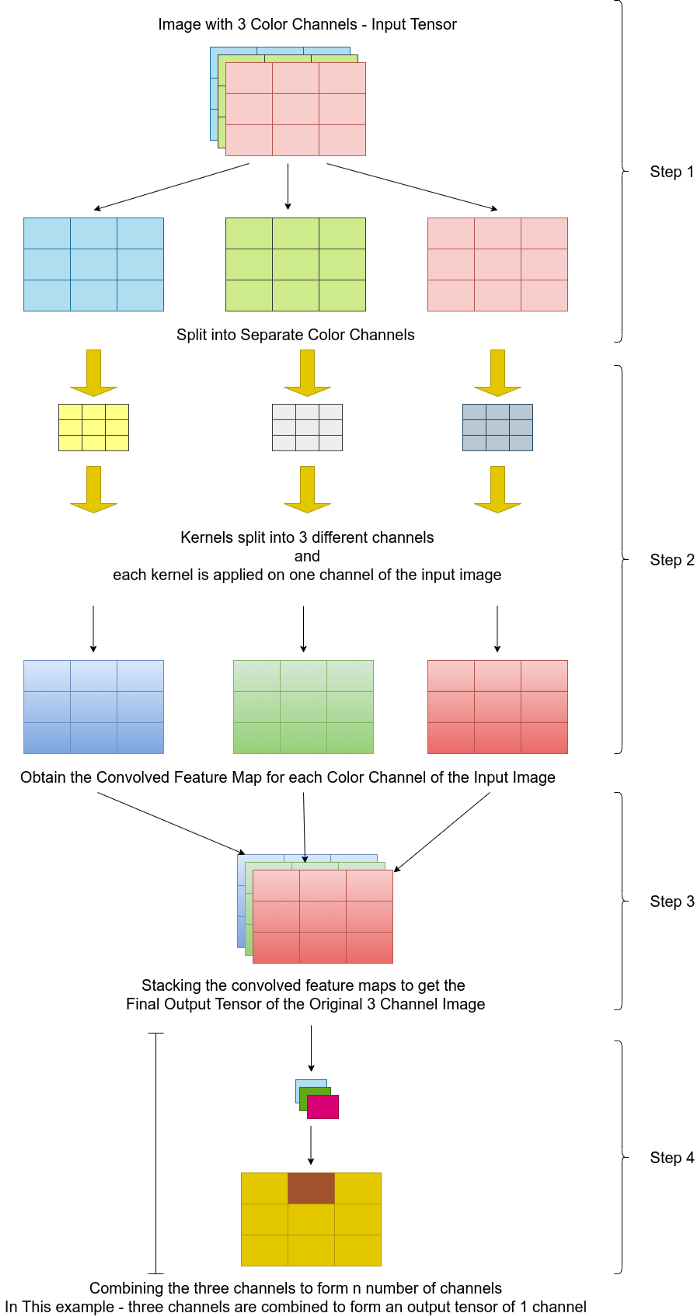


In Depth wise convolution The three-dimensional input tensor is split into separate channels . For each channel, the input is convolved with a (2D) filter. The output of each channel is stacked to get the output over the entire 3D tensor.[7]



**Depth-wise Separable Convolution:**

Depth-wise convolutions are usually applied in combination with another step: the Depth-wise Separable Convolution. This consists of two parts: 1. Filtering (all the above steps) and 2. Blending (combining the 3 color channels to form 'n' channels as desired, in the example below we see how the 3 channels can be blended to them to form a 1-channel output).[7]



**Why is depth-wise convolution so efficient?**

The convolution in depth is -1x1 convolutions on all channels   
  
Suppose we have an input tensor of size — 8x8x3,   
  
AND the desired output tensor is of size — 8x8x256   
  
In 2D convolutions —   
  
Number of multiplications required — ( 8x8) x (5x5x3) x (256) = 1,228,800   
  
In depth-wise separable convolutions —   
  
The number of required multiplications:   
  
a. Filtering: It is divided into individual channels, so a 5x5x1 filter is needed instead of 5x5x3, and since there are three channels, the total number of 5x5x1 filters required is 3, so   
  
(8x8) x (5x5x1) x (3 ) = 3,800   
  
b. Combination: The total number of channels needed is 256, therefore   
,  
(8x8) x (1x1x3) x (256) = 49,152,  
,  
Total number of multiplications = 3,800 + 49,152 = 53,952,  
,  
Therefore, a 2D convolution requires 1,228,800 multiplications, while a depth separable convolution requires only 53,952 multiplications to achieve the same result.   
  
Finally   
  
1,228,800/53,952 = 23x fewer multiplications required   
  
This is why the efficiency of depth-separable convolutions is so high. These are the layers implemented in the MobileNet architecture to reduce the number of computations and make them less power hungry, allowing them to run on mobile/embedded devices that don't have powerful graphics processing units.[7]

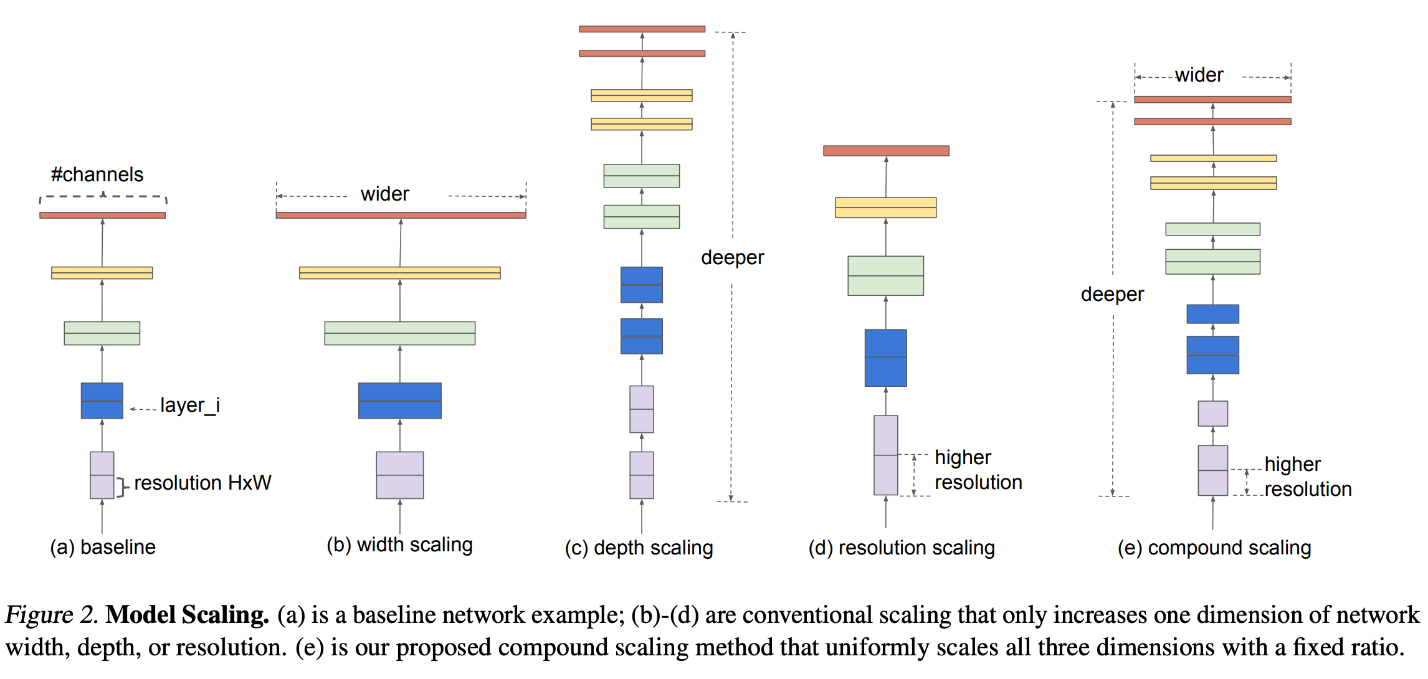
**Efficientnet:**

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all depth/width/resolution dimensions by utilizing a compound coefficient. Contrary to traditional practice, which scales these factors arbitrarily, the EfficientNet scaling method uniformly scales the width, depth, and resolution of the network with a set of fixed scaling coefficients. For example, if we ever want to use times more computational resources, we can simply increase the depth of the network by , the width by , and the size of the image by , where are constant coefficients determined by a small grid search of the original small model. EfficientNet uses a compound coefficient to uniformly scale the network's width, depth, and resolution in a principle-based manner.[8]

**Compound Model Scaling:**

EfficientNet uses a technique called compound coefficient to scale models in a simple but effective way. Rather than randomly scaling width, depth, or resolution, compound scaling uniformly scales each dimension with some fixed set of scaling coefficients. Using the scaling method and AutoML, the efficiency authors developed seven multidimensional models that exceeded the accuracy of most state-of-the-art convolutional neural networks, and with much greater efficiency.[9] We studied the impact of scaling different dimensions of the model in order to understand the effect of scaling the network.While scaling individual dimensions improves model performance, we discovered that balancing all network dimensions—width, depth, and image resolution—against available resources improves overall performance the most.

The compound scaling method begins with a grid search to determine the relationship between different scaling dimensions of the baseline network under a fixed resource constraint (e.g., 2x more FLOPS).This determines the scaling coefficient for each of the previously mentioned dimensions.The coefficients are then used to scale up the baseline network to the desired target model size or computational budget.



When compared to conventional scaling methods, this compound scaling method consistently improves model accuracy and efficiency when scaling up existing models such as MobileNet (+1.4 percent imagenet accuracy) and ResNet (+0.7 percent).[10]

**Performance of EfficientNet:**

In general, the EfficientNet models outperform existing CNNs in terms of accuracy and efficiency, reducing parameter size and FLOPS by an order of magnitude. Though EfficientNets perform well on ImageNet, they should also transfer to other datasets to be most useful.[10]

**Inception V3(Image recognition):**

The Inception-v3 convolutional neural network architecture is from the inception family and uses Label Smoothing, Factorized 7 x 7 convolutions, and the inclusion of an auxiliary classifier to transport label information lower down the network, among other advances (along with the use of batch normalization for layers in the side head). On the ImageNet dataset, it has been demonstrated that the image recognition model Inception v3 can achieve higher than 78.1 percent accuracy. The model is the result of numerous concepts that have been established by various researchers over the years.[11]

Convolutions, average pooling, max pooling, concatenations, dropouts, and fully linked layers are some of the symmetric and asymmetric building components that make up the model itself. The model makes considerable use of batch normalization, which is also applied to the activation inputs. To calculate loss, Softmax is used. [12]

The Inception v3 model, which was introduced in 2015, has 42 layers overall and a reduced mistake rate than its forerunners. Let's examine the various improvements that the Inception V3 model has received.

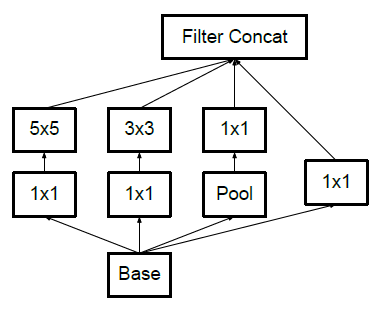
The Inception V3 model has undergone significant changes, including:

* Factorization into Smaller Convolutions
* Spatial Factorization into Asymmetric Convolutions
* Utility of Auxiliary Classifiers
* Efficient Grid Size Reduction

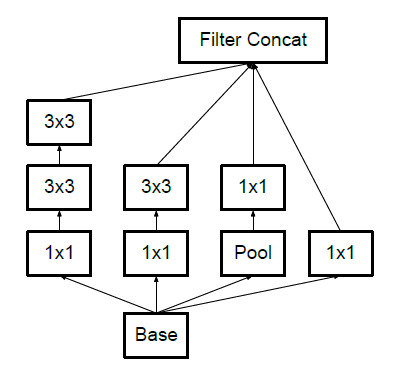
**Factorization into Smaller Convolutions:**

The extensive dimension reduction was one of the Inception V1 model's key advantages. The model's larger Convolutions were factorized into smaller Convolutions to make it even better.

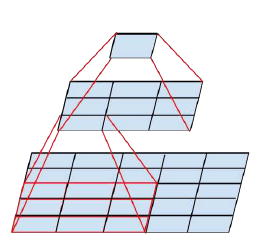
Take the inception V1 module's basic module as an illustration.



It has a 5×5 convolutional layer, which previously said was computationally expensive. The 5×5 convolutional layer was thus replaced with two 3×3 convolutional layers in order to lower the computational cost, as seen below:



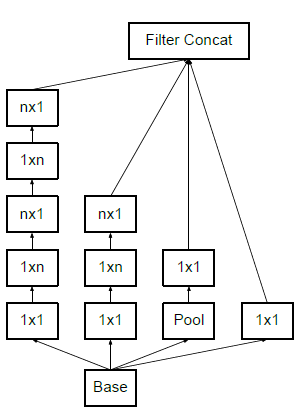
See how the technique of utilizing two 33 convolutions minimizes the amount of parameters to better grasp it.



The computing expenses also decrease as a result of the fewer parameters. There was a relative gain of 28% as a result of factorizing bigger convolutions into smaller convolutions.

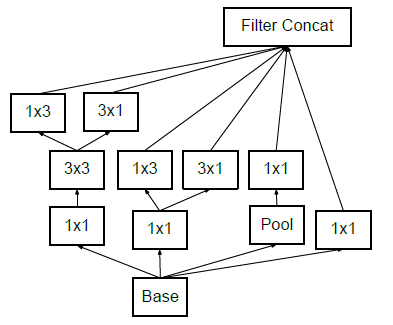
**Spatial Factorization into Asymmetric Convolutions:**

This is done to make the model more efficient. Asymmetric convolutions take the form n×1. They therefore substituted a 1×3 convolution followed by a 3×1 convolution for the 3×3 convolutions. This is equivalent to sliding a two-layer network with a 3×3 convolution's receptive field.



**Module 2**  
Structure of Asymmetric Convolutions

If the number of input and output filters is the same, the two-layer method is 33 percent less expensive for the same number of output filters. This is how the inception module appears following the application of the first two optimization approaches:



**Utility of Auxiliary Classifiers:**

Auxiliary classifiers are used to enhance the convergence of very deep neural networks. In very deep networks, the vanishing gradient problem is mostly solved by the auxiliary classifier.

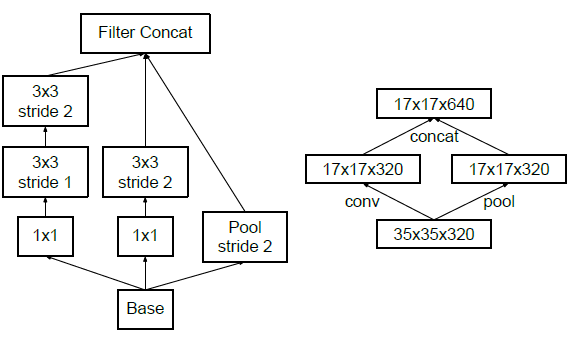
Early on in the training, there was no improvement as a result of the auxiliary classifiers. However, as the experiment progressed, the network with auxiliary classifiers outperformed the network without auxiliary classifiers in terms of accuracy.

As a result, the Inception V3 model architecture's auxiliary classifiers function as a regularizer.

**Efficient Grid Size Reduction:**

Traditionally, the grid size of the feature maps was decreased using average and maximum pooling. The activation dimension of the network filters is increased in the Inception V3 model to more effectively reduce the grid size.

For instance, reduction produces a d/2 × d/2  grid with 2k filters from a d×d  grid with k filters and two parallel blocks of convolution and pooling that were later concatenated are used to accomplish this.



The aforementioned figure demonstrates how the grid size is effectively decreased while the filter banks are expanded.

**Final inception V3 model:**

The final Inception V3 model after all the optimizations seems like this:



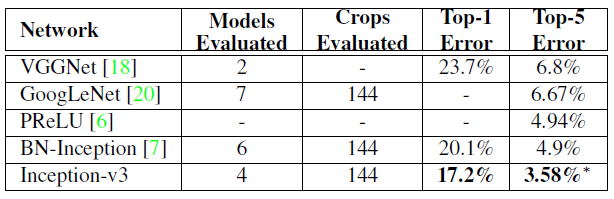
The inception V3 model has 42 layers in total, which is a little more than the inception V1 and V2 models. However, this model's effectiveness is absolutely remarkable.The parts that make up the Inception V3 model are:

| **TYPE** | **PATCH / STRIDE SIZE** | **INPUT SIZE** |
| --- | --- | --- |
| Conv | 3×3/2 | 299×299×3 |
| Conv | 3×3/1 | 149×149×32 |
| Conv padded | 3×3/1 | 147×147×32 |
| Pool | 3×3/2 | 147×147×64 |
| Conv | 3×3/1 | 73×73×64 |
| Conv | 3×3/2 | 71×71×80 |
| Conv | 3×3/1 | 35×35×192 |
| 3 × Inception | Module 1 | 35×35×288 |
| 5 × Inception | Module 2 | 17×17×768 |
| 2 × Inception | Module 3 | 8×8×1280 |
| Pool | 8 × 8 | 8 × 8 × 2048 |
| Linear | Logits | 1 × 1 × 2048 |
| Softmax | Classifier | 1 × 1 × 1000 |

The inception V3 model's general structure is shown in the accompanying table. Each module's output size serves as the subsequent module's input size in this situation.

**Performance of inception V3:**

As anticipated, the Inception V3 version was more accurate and required less computing power than the prior Inception version.

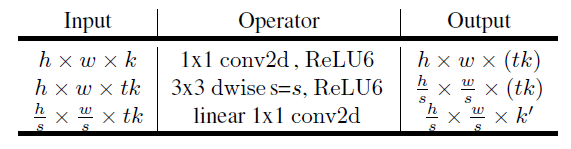


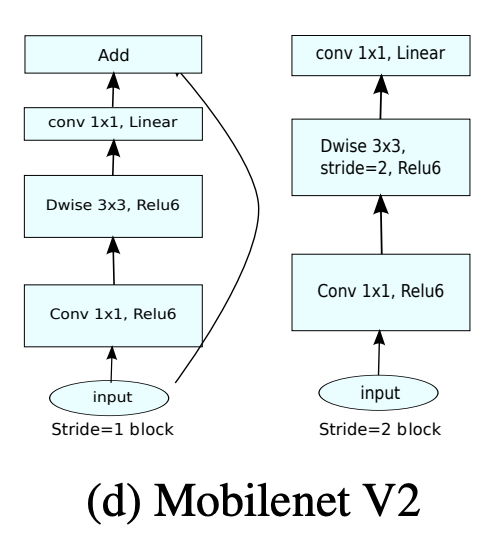
Multi-crop reported results.

Comparing the Inception V3 model against its predecessors and its contemporaries, we can observe that it has a remarkably low error rate. [13] However blurry images might result in lower accuracy. [14]

**Mobilenet V2(Image classification):**

A convolutional neural network design called MobileNetV2 aims to function well on mobile devices. It is built on an inverted residual structure where the bottleneck layers are connected by residual connections. Two distinct block types can be seen in MobileNetV2, with a stride of 1, the first is a residual block, the other one is the block with a stride of two for downsizing. For both varieties of blocks, there are three levels. 1×1 convolution with ReLU6 makes up the first layer. The depth wise convolution is the second layer. The third layer is an 1×1 convolution once more, but this time there is no non-linearity. Deep networks are said to only have the power of a linear classifier on the non-zero volume portion of the output domain if ReLU is applied again. Lightweight depth-wise convolutions are used in the intermediate expansion layer as a source of non-linearity to filter features. The architecture of MobileNetV2 includes a 32-filter initial fully convolution layer as well as 19 additional bottleneck layers. MobileNetV2 enhances the state-of-the-art performance of mobile models across a range of different model sizes, tasks, and benchmarks. There is also an expansion factor t and for all main experiments, t=6.The internal output would have  64×t=64×6=384 channels if the input had 64 channels. [15]





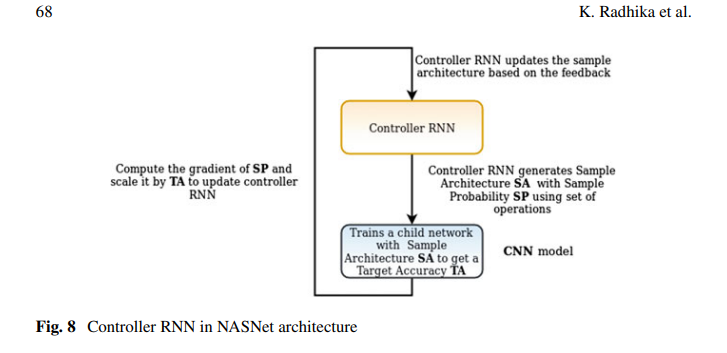
 MobileNetV2 is a significant improvement over MobileNetV1 and pushes the state of the art for mobile visual recognition including classification, object detection and semantic segmentation.[16]

**Mobilenet V2 performance**

With only a slight accuracy loss, MobileNetV2's computational cost is 8 to 9 times lower than that of ordinary convolutions because to its use of k = 3 (3 × 3 depth wise separable convolutions).[15][17] Overall, the MobileNetV2 models are faster across the entire latency spectrum for the same accuracy. The new models, in particular, use 2x fewer operations, require 30% fewer parameters, and are 30-40% faster on a Google Pixel phone than MobileNetV1 models, all while achieving higher accuracy. MobileNetV2 is a powerful feature extractor that can detect and segment objects. For example, when combined with the newly introduced SSD Lite [18] the new model is approximately 35% faster and has the same accuracy as MobileNetV1. The model has been released under the Tensorflow Object Detection API [19].

**Nasnet(Image recognition):**

Two promising concepts, AutoML and Neural Architectural Search (NAS), result in NASNet, a new optimized architecture. Reinforcement Learning (RL) and Evolutionary Algorithms (EAs) are concepts that aid in the optimization task.[20] NASNet, is formed automatically by optimizing the individual cells that make up the network.[21] By introducing the concept of optimized network, the deep neural network has grown to the next generation. This concept was realized by the Google ML group through the concept of NAS. Their strategy relied on reinforcement learning. The parent AI evaluates the performance of the child AI and makes changes to the neural network architecture. To improve network efficiency, several changes were made based on the number of layers, weights, regularization methods, and so on. As shown in Fig. 8, the architecture consists of a Controller Recurrent Neural Network (CRNN) and a CNN to be trained. The NASNet A, B, and C algorithms use the reinforcement learning method to select the best cells.[20] According to Chen et al. [22], the best candidates can be chosen using a reinforced evolutionary algorithm. The tournament selection algorithm is used to eliminate the cell with the lowest performance. The child fitness function is improved, and reinforcement mutations are carried out. This improves the performance of the cell structure even more.



**NASNET architechture:**

The NASNet architecture's operational blocks are listed below:

– Identity

– 1 × 3 then 3 × 1 convolution

– 1 × 7 then 7 × 1 convolution

– 3 × 3 dilated convolution

– 3 × 3 average pooling

– 3 × 3 max pooling

– 5 × 5 max pooling

– 7 × 7 max pooling

– 1 × 1 convolution

– 3 × 3 convolution

– 3 × 3 depth wise-separable—convolution

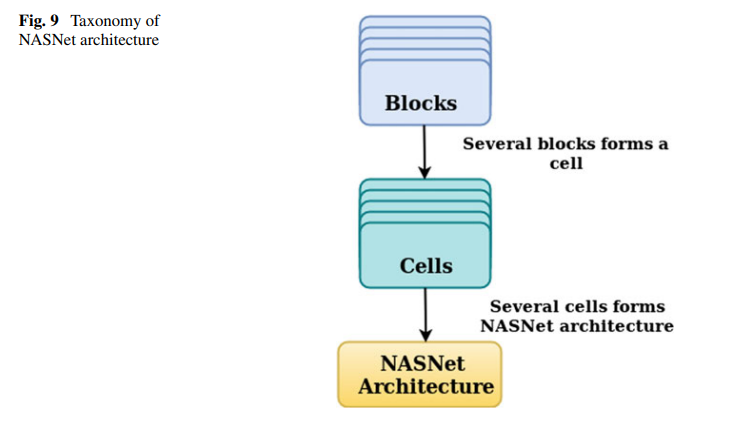
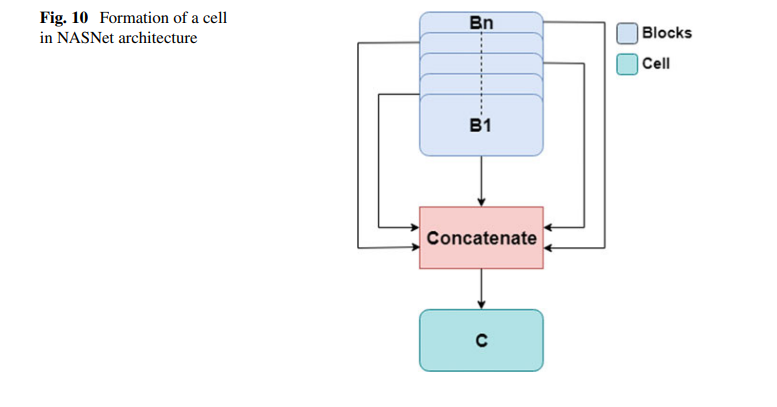
– 5 × 5 depth wise-separable—convolution

– 7 × 7 depth wise-separable—convolution

The NASNet architecture is trained with two types of input images, 331 × 331 and 224 × 224, to produce the NASNetLarge and NASNetMobile architectures. Moving from NASNetMobile to NASNetLarge results in a significant increase in the number of parameters. NASNetMobile has 53,26,716 parameters, while NASNetLarge has 8,89,49,818. This improves the dependability of NASNetMobile.

As shown in Fig. 9, a block is the smallest unit in the NASNet architecture, and a cell is a combination of blocks. As shown in Fig. 10, a cell is formed by concatenating different blocks. NASNet's search space is created by factorizing the network into cells and then further subdividing it into blocks. These cells and blocks are not fixed in terms of number or type, but are optimized for the chosen dataset .

As shown in Fig. 11, a block is an operational module. A block's operations include normal convolutions, separable convolutions, max-pooling, average-pooling, identity mapping, and so on. The blocks convert two inputs (one current and one previous, for example, H0 and H1) to a single output feature map. It necessitates element-by-element addition. If the cell is given a block with a feature map of size H × W and stride of one, the output will be the same size as the feature map. If the stride is two, the size is reduced by two. The cells are combined in the most efficient way possible. The cell structure, the number of cells to be stacked (N), and the number of filters in the first layer all play a role in network development (F). During the search, N and F are initially fixed. Later, N and F in the first layer are adjusted to control the network's depth and width. After the search is completed, models of various sizes are built to fit the datasets. The cells are then linked in an optimized fashion to form the NASNet architecture. Each cell is linked to two input states, known as hidden states.



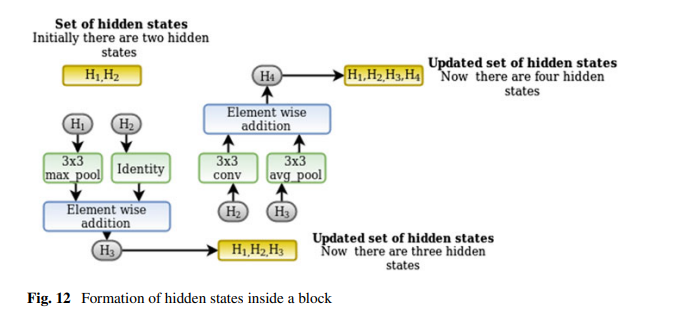
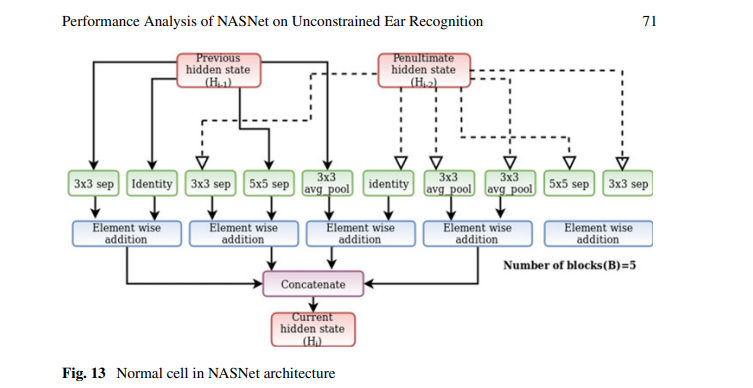
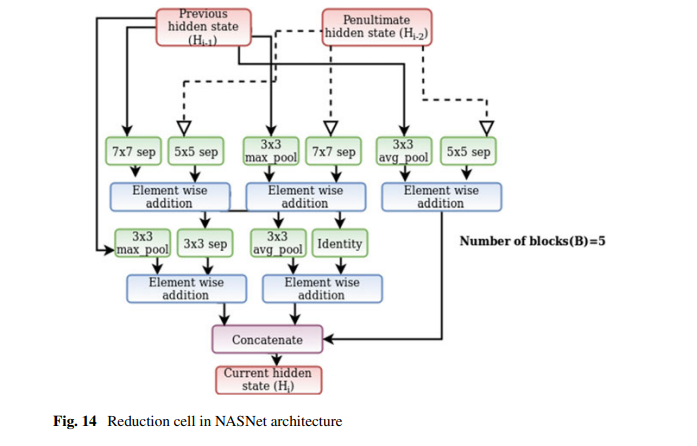
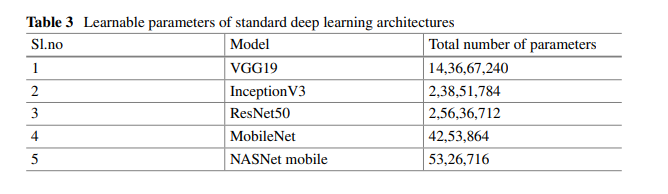
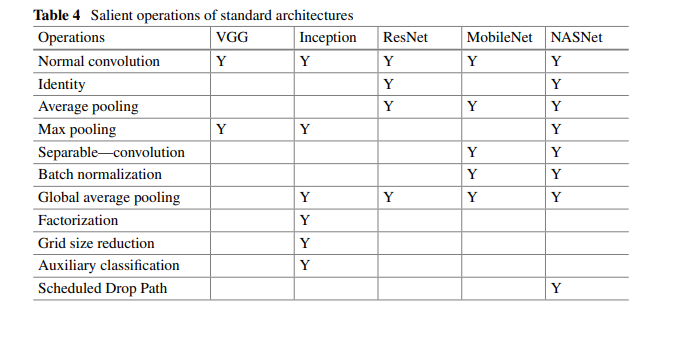


Figure 12 depicts a sample of hidden state formation (4 hidden layers). The hidden layers are created through pairwise combinations and updated through concatenation.Hidden layers can be subjected to a variety of convolution and pooling operations. NASNet architecture selects only the best cells rather than searching all cells. This will speed up the search, allowing for more generalized features to be obtained. As shown in Figs. 13 and 14, the NAS search space has a controller-child structure, with the controller being a normal cell and the child being a reduction cell. NASNetLarge chose N as 6 to provide high accuracy, whereas NASNetMobile's main concern was running with limited resources.The selected set of operations using 5B cells reduces the input size of 224 × 224 × 3 to a size of  7 × 7  at the output for both normal and reduction cells. In NASNet, a new concept known as scheduled droppath is introduced, in which each path in the cell is dropped with a linearly increasing probability as the network's training progresses [20]. The reduction cell in Fig. 14 implies that the Hi is formed by a series of operations on the Hi−1 and Hi−2 cells, culminating in their concatenation. These prediction steps are repeated B times by the RNN. This NASNet structure significantly increased the speed of operation.

**Contribution of different architectures:**

Table 3 shows the total number of learnable parameters in each model. A scenario for reducing the number of learnable parameters was observed while moving on to newer architectures.

Table 4 summarizes the major operations in the architectures discussed above.[21]

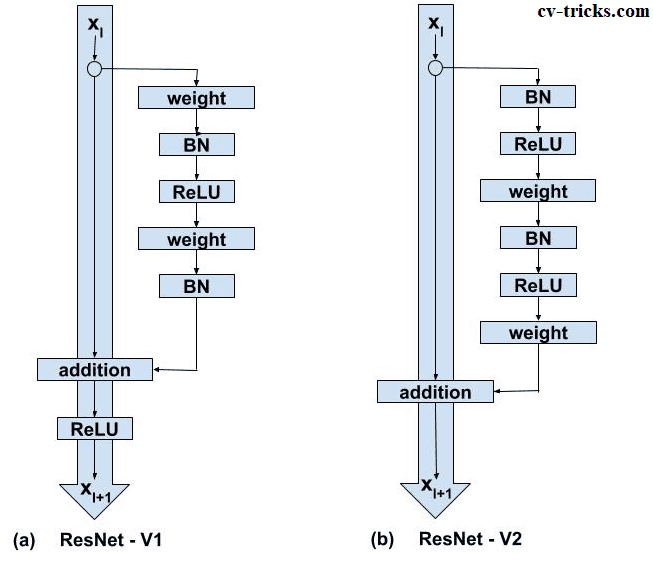


**Nasnet performance:**

The Google Brain team first introduced this deep network in early 2018. They sought to define a building block with high performance in the categorization of a small set of images in its design (CIFAR-10). They then applied the block to a larger data set (ImageNet). This architecture achieves a high classification capacity while using a small number of parameters.[23] Nasnet has superior performance when compared to architectures such as Inception-v2, Inception-v3, Xception, ResNet, and Inception-ResNet-v2[21][23][24]

**Resnet 50 V2(Image classification):**

ResNet50V2 [25] is a modified version of ResNet50 that performs better on the ImageNet dataset than ResNet50 and ResNet101. The propagation formulation of the connections between blocks was changed in ResNet50V2. ResNet50V2 also performs well on the ImageNet dataset. ResNet50 version 2 emphasizes the use of weight layer pre-activation rather than post-activation. The diagram below depicts the basic architecture of ResNet's post-activation (version 1) and pre-activation (version 2) versions.



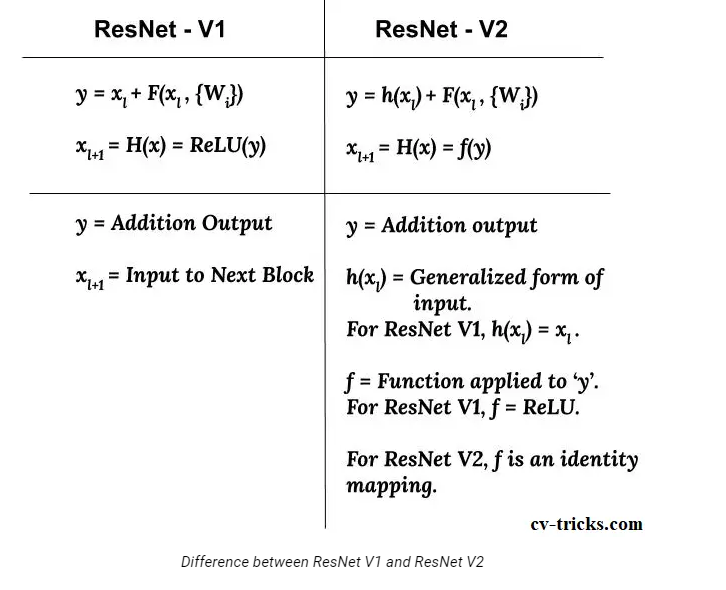
**Difference between ResNet-V1 and ResNet-V2**

After the addition operation is performed between the x and F(x), ResNet V1 adds the second non-linearity . ResNet V2 has removed the last non-linearity, resulting in a clear path from input to output in the form of an identity connection.

Before multiplication with the weight matrix, ResNet V2 applies Batch Normalization and ReLU activation to the input (convolution operation). Convolution is performed by ResNet V1, followed by Batch Normalization and ReLU activation.

The ResNet V2 primarily focuses on treating the second non-linearity as an identity mapping, which means that the output of the addition operation between the identity mapping and the residual mapping should be passed directly to the next block for processing. However, in ResNet V1, the output of the addition operation passes through ReLU activation and is then transferred as the input to the next block.

The signal can be directly propagated between any two units when the function 'f' is an identity function. Furthermore, the gradient value calculated at the output layer can easily reach the initial layer with no signal change.[25][26]



**Resnet performance:**

ResNet is one of the most powerful deep neural networks, and it performed spectacularly in the ILSVRC 2015 classification challenge. ResNet demonstrated excellent generalization performance on other recognition tasks, taking first place in the ILSVRC and COCO 2015 competitions for ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation. There are numerous ResNet architecture variants, which are the same concept but with a different number of layers.ResNet50V2 networks extract deep features as well as or better than other networks.[25][26]

**VGG 16(Object Recognition):**

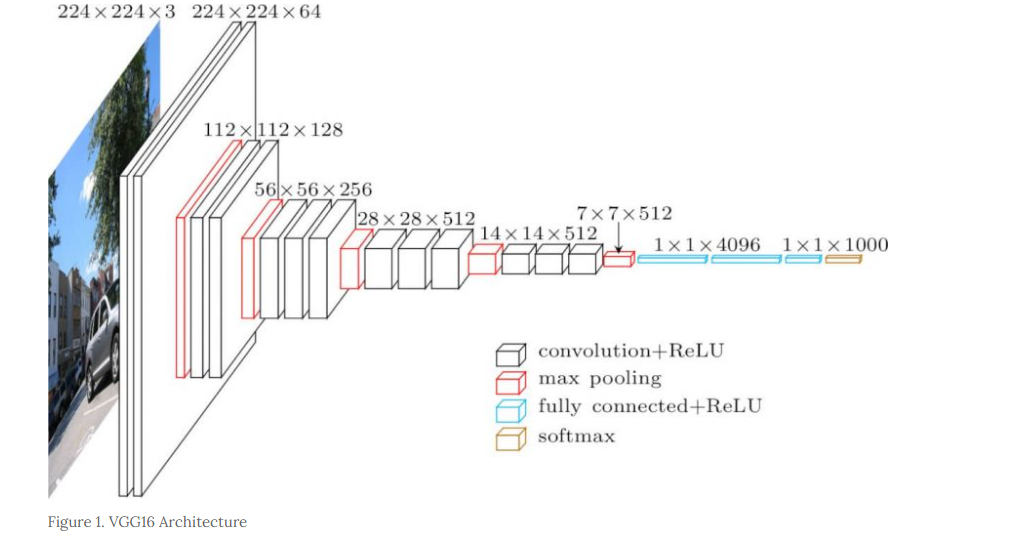
VGG16 is a 16-layer traditional object-recognition model. VGG16, which was designed as a deep CNN, outperforms on a wide range of tasks and data sets other than ImageNet. Even today, VGG16 is one of the most widely used image-recognition architectures.[27] Karen Simonyan and Andrew Zisserman proposed the VGG network idea in 2013, and the actual model based on the idea was submitted in the 2014 ImageNet Challenge. They named it VGG after the Visual Geometry Group department at the University of Oxford to which they belonged. To begin with, in contrast to the large receptive fields in the first convolutional layer, this model proposed using a very small 3 x 3 receptive field (filters) throughout the entire network with a stride of 1 pixel. The concept of using 3 x 3 filters uniformly distinguishes the VGG. Two consecutive 3 x 3 filters produce a 5 x 5 effective receptive field. Similarly, three 3 x 3 filters result in a 7 x 7 receptive field. As a result, a combination of multiple 3 x 3 filters can stand in for a larger receptive area. In addition to the three convolution layers, there are three non-linear activation layers rather than a single one in 7 x 7. As a result, the decision functions become more discriminative. It would give the network the ability to converge faster. Second, it significantly reduces the number of weight parameters in the model. If the input and output of a three-layer 3 x 3 convolutional stack each have C channels, the total number of weight parameters is 3 \* 32 C2 = 27 C2. When compared to a 7 x 7 convolutional layer, 72 C2 = 49 C2 is required, which is nearly twice the number of 3 x 3 layers. This can also be viewed as a regularization of the 7 x 7 convolutional filters, forcing them to decompose through the 3 x 3 filters, with non-linearity added in-between by means of ReLU activations. This would reduce the network's proclivity to over-fit during the training process.

VGG was first proposed, VGG16 continues to pique the interest of data scientists and researchers worldwide. **Here are a few examples of practical applications for VGG16:**

* **VGG16 Image Recognition or Classification**- It can be used to diagnose diseases using medical imaging such as x-rays or MRI. It can also be used to recognize street signs while driving.
* **Image Detection and Localization** - We didn't go over VGG16's detection capabilities earlier, but it can perform admirably in image detection use cases. In fact, it was the 2014 ImageNet detection challenge winner (where it ended up as first runner up for classification challenge)
* **Image Embedding Vectors** - After removing the top output layer, the model can be trained to generate image embedding vectors that can be used for problems such as face verification using VGG16 inside a Siamese network.[28][29]

**VGG 16 Architecture:**

VGG16 is the architecture depicted below.

****

The input to the cov1 layer is a 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, with the filters set to capture the notions of left/right, up/down, and center with a very small receptive field:  3×3 (the smallest size to capture the notions of left/right, up/down, and center). It also employs 1×1 convolution filters in one of the configurations, which can be thought of as a linear transformation of the input channels (followed by non-linearity). The convolution stride is set to 1 pixel, and the spatial padding of conv. layer input is set so that the spatial resolution is preserved after convolution, i.e. 1-pixel padding for  3×3 conv. layers. Five max-pooling layers, which follow some of the conventional layers (not all the conv. layers are followed by max-pooling). Max-pooling is done with stride 2 over a 2×2 pixel window. Following a stack of convolutional layers (with varying depths in different architectures), three Fully-Connected (FC) layers are added: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The soft-max layer is the final layer. In all networks, the configuration of the fully connected layers is the same. The rectification (ReLU) non-linearity is present in all hidden layers. It is also worth noting that none of the networks (except one) use Local Response Normalization (LRN), which does not improve performance on the ILSVRC dataset but increases memory consumption and computation time.[30]

**VGG 16 Performance:**

VGG16 is a convolutional neural network model proposed in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" by K. Simonyan and A. Zisserman of the University of Oxford. In ImageNet, a dataset of over 14 million images classified into 1000 classes, the model achieves 92.7 percent top-5 test accuracy. It was one of the well-known models submitted to the ILSVRC-2014. It outperforms AlexNet by replacing large kernel-sized filters (11 and 5, respectively, in the first and second convolutional layers) with multiple 33 kernel-sized filters one after the other. VGG16 had been training for weeks on NVIDIA Titan Black GPUs.

In the ILSVRC-2012 and ILSVRC-2013 competitions, VGG16 significantly outperforms the previous generation of models. The VGG16 result is competing for the classification task winner (GoogLeNet with 6.7 percent error) and outperforms the ILSVRC-2013 winning submission Clarifai, which achieved 11.2 percent with external training data and 11.7 percent without. In terms of single-net performance, the VGG16 architecture outperforms a single GoogLeNet by 0.9 percent (7.0 percent test error).



VGGNet, unfortunately, has two major drawbacks:

* Training is excruciatingly slow.
* The network architecture weights (in terms of disk/bandwidth) are quite large.

VGG16 is larger than 533MB due to its depth and number of fully connected nodes. This makes VGG deployment time-consuming. Although VGG16 is used in many deep learning image classification problems, smaller network architectures are frequently preferred (such as SqueezeNet, GoogLeNet, etc.). However, because it is simple to implement, it is an excellent learning tool.[30]

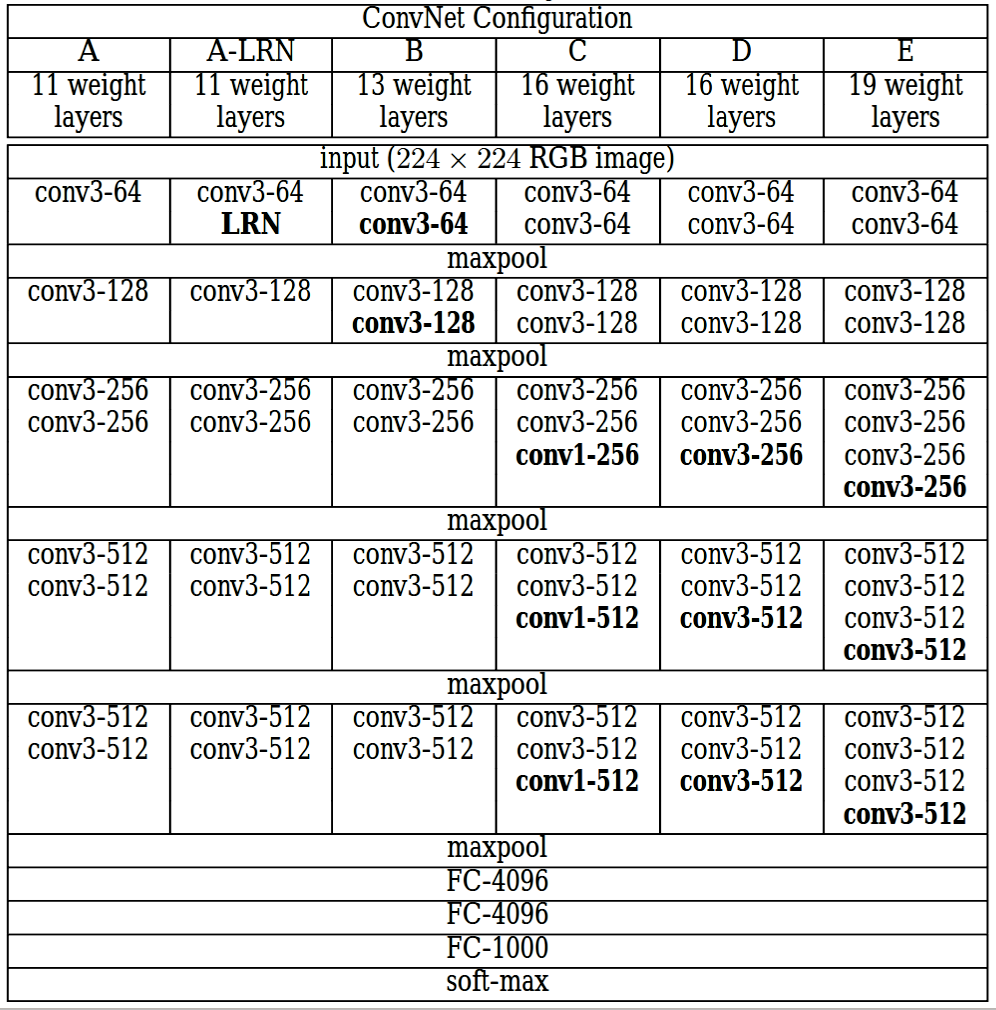
**VGG 19(Image classification):**

VGG19 is a VGG model variant that consists of 19 layers (16 convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 SoftMax layer). Other VGG variants include VGG11, VGG16, and others. VGG19 has a total of 19.6 billion FLOPs. In simpler words, VGG is a deep CNN used to classify images. The following are the layers in the VGG19 model:

* Conv3x3 (64)
* Conv3x3 (64)
* MaxPool
* Conv3x3 (128)
* Conv3x3 (128)
* MaxPool
* Conv3x3 (256)
* Conv3x3 (256)
* Conv3x3 (256)
* Conv3x3 (256)
* MaxPool
* Conv3x3 (512)
* Conv3x3 (512)
* Conv3x3 (512)
* Conv3x3 (512)
* MaxPool
* Conv3x3 (512)
* Conv3x3 (512)
* Conv3x3 (512)
* Conv3x3 (512)
* MaxPool
* Fully Connected (4096)
* Fully Connected (4096)
* Fully Connected (1000)
* SoftMax

**VGG 19** **Architecture:**

* This network was fed a fixed size (224 \* 224) RGB image as input, implying that the matrix was of shape (224,224,3).
* The only preprocessing done was to subtract the mean RGB value from each pixel over the entire training set.
* They used kernels of (3 \* 3) size with a stride size of 1 pixel to cover the entire image concept.
* To keep the image's spatial resolution, spatial padding was used.
* Sride 2 was used to perform max pooling over a 2 \* 2 pixel window.
* This was followed by Rectified linear unit (ReLu) to introduce non-linearity into the model to improve classification and reduce computational time, as previous models used tanh or sigmoid functions and this proved to be far superior to those.
* Three fully connected layers were implemented, the first two of which were 4096 in size, followed by a layer with 1000 channels for 1000-way ILSVRC classification, and the final layer is a softmax function.



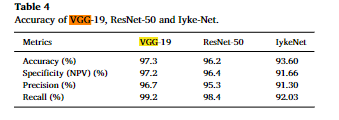
*Table 1 : Actual configuration of the networks, the ReLu layers are not shown for the sake of brevity.*

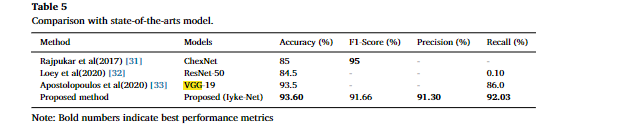
Column E contained 16 layers of CNNs, three fully connected layers, and a final layer for the softmax function; the fully connected layers and final layer will be the same for all network architectures.

* A: Has 8 CNN layers, for a total of 11 layers including the fully connected (FC) layers, and has no internal differences other than the number of layers.
* A-LRN: This is similar to column A, but it has one extra step of Local response normalization (LRN) which implements lateral inhibition in the layer by making a significant peak and thus creating a local maxima which increases the sensory perception which we may want in our CNN but it was observed that for this specific CNN ,In the case of ILSVRC, accuracy was not increasing, and the overall network was taking longer to train.
* C: This architecture has 13 CNN layers and 16 including the FC layers. The authors used a conv filter of (1 \* 1) only to introduce non-linearity and thus better discrimination.
* B and D: These columns simply add extra CNN layers, with 13 and 16 layers, respectively.[31]

**VGG 19 Performance:**

The VGG19 approach [32] outperformed other methods in classifying non-carcinoma and carcinoma histopathology images of breast cancer. In a paper it was found that the overall accuracy of the fully-trained VGG19 model is 90.35 percent (1.35), while the average F1 score is 90.31 percent (1.35)[33] The VGG19 model has recently demonstrated good performance in image classification. The experimental results, however, show that even this model is insufficient for accurate image classification.[34] An experiment showed VGG 19 performed extremely well with 97.3 percent accuracy in identifying all pneumonia images[35]





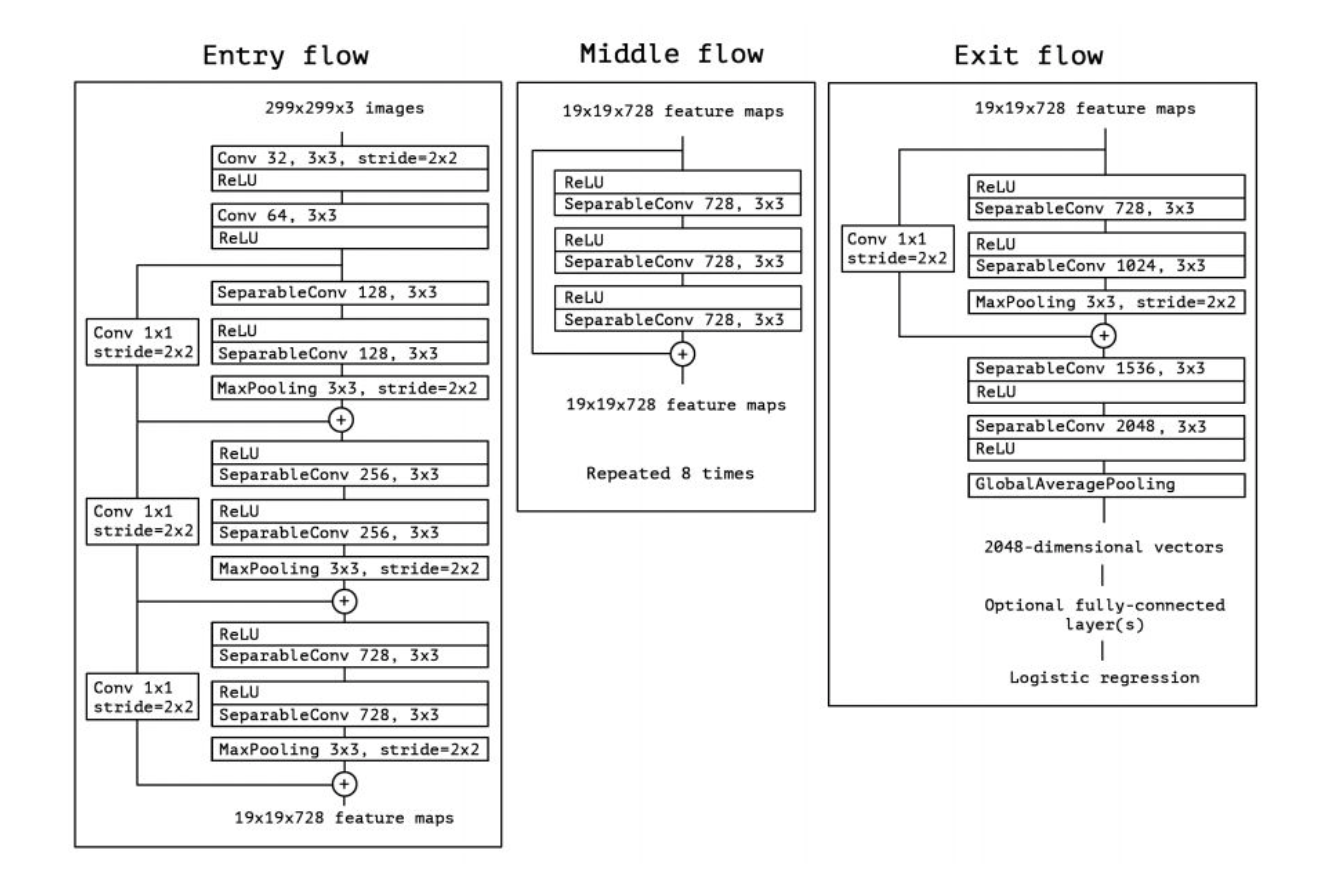
[35]

**Xception(Object Recognition):**

Xception is a deep convolutional neural network architecture with Depthwise Separable Convolutions. It was created by Google scientists.[36] The Xception model's architecture is a linear stack of depth-wise separable convolution layers with residual connections that allows the deep network architecture to be easily defined and modified . The Xception is an Inception architecture enhancement that replaces regular Inception modules with distinguishable depth convolutions.[37] Xception stands for "extreme inception," and it takes the principles of Inception to their logical conclusion. In Inception, 1x1 convolutions were used to compress the original input, and we used different types of filters on each depth space from each of those input spaces. This step is simply reversed by Xception. Instead, it first applies the filters to each depth map before compressing the input space with 1X1 convolution across the depth. This method is nearly identical to a depthwise separable convolution, which has been used in neural network design since 2014. One more distinction exists between Inception and Xception. The presence or absence of a nonlinearity following the initial operation. Both operations in the Inception model are followed by a ReLU non-linearity, whereas Xception does not.[38]

**Xception Architecture:**

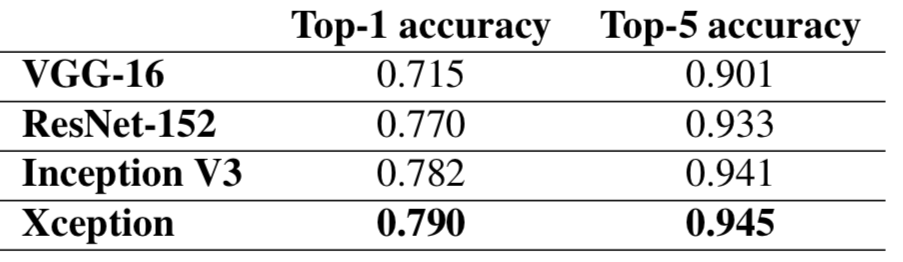
The feature extraction base of the network is formed by 36 convolutional layers in the Xception architecture. Except for the first and last modules, the 36 convolutional layers are organized into 14 modules, each of which is surrounded by linear residual connections. Except for the first and last modules, the 36 convolutional layers are organized into 14 modules, each of which is surrounded by linear residual connections.[39]

[38]

The data is processed after passing through the entry flow. goes through the middle flow (which it repeats 8 times) and finally through the exit flow[38]

**Xception Performance:**

According to the table below, Xception outperforms every model in the ImageNet dataset.[38]

[38]

Validation accuracy is also higher for Xception than inception, and having no non-linearity in between Xception outperforms having any kind of non-linearity.[38]

References:

1. <https://arxiv.org/abs/1704.04861>
2. <https://arxiv.org/ftp/arxiv/papers/1712/1712.04698.pdf>
3. <https://openaccess.thecvf.com/content_CVPR_2020/papers/Phan_Binarizing_MobileNet_via_Evolution-Based_Searching_CVPR_2020_paper.pdf>
4. <https://gogul.dev/software/mobile-net-tensorflow-js>
5. <https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470>
6. <https://paperswithcode.com/method/depthwise-separable-convolution>
7. https://towardsdatascience.com/understanding-depthwise-separable-convolutions-and-the-efficiency-of-mobilenets-6de3d6b62503
8. <https://paperswithcode.com/method/efficientnet>
9. <https://medium.com/mlearning-ai/understanding-efficientnet-the-most-powerful-cnn-architecture-eaeb40386fad>
10. <https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html>
11. <https://arxiv.org/pdf/1512.00567.pdf>
12. <https://cloud.google.com/tpu/docs/inception-v3-advanced>
13. <https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf>
14. <https://iopscience.iop.org/article/10.1088/1742-6596/2010/1/012052/pdf>
15. <https://arxiv.org/pdf/1801.04381v4.pdf>
16. <https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html>
17. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam.
18. [MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications](https://arxiv.org/abs/1704.04861), *Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H, arXiv:1704.04861, 2017.*
19. [Speed/accuracy trade-offs for modern convolutional object detectors](http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_SpeedAccuracy_Trade-Offs_for_CVPR_2017_paper.pdf), *Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K, CVPR 2017.*
20. Zoph, Barret, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 8697–8710.
21. <https://www.researchgate.net/profile/Radhika-K-2/publication/337568463_Performance_Analysis_of_NASNet_on_Unconstrained_Ear_Recognition/links/5df1d53192851c836475f48e/Performance-Analysis-of-NASNet-on-Unconstrained-Ear-Recognition.pdf>
22. Chen, Yukang, Qian Zhang, Chang Huang, Mu Lisen, Gaofeng Meng, and Xinggang Wang. 2018. Reinforced evolutionary neural architecture search. CoRR.
23. F. Martínez, C. Hernández, and F. Martínez, “Evaluation of deep neural network architectures in the identification of bone fissures,” Telkomnika, vol. 18, no. 2, pp. 807–814, 2020.
24. ] A. Rendón, F. Martínez, and C. Hernández, “Deep regression model for predictive control in a vegetable waste carbonization plant,” Contemporary Engineering Sciences, vol. 10, no. 21, pp. 1047–1055, 2017. (<https://core.ac.uk/download/pdf/325990317.pdf>)
25. K. He, X. Zhang, S. Ren, J. Sun

Identity mappings in deep residual networks

European conference on computer vision, Springer (2016), pp. 630-645

(<https://www.sciencedirect.com/science/article/pii/S2352914820302537#:~:text=ResNet50V2%20%5B36%5D%20is%20a%20modified,result%20on%20the%20ImageNet%20dataset>.) (<https://arxiv.org/pdf/1603.05027.pdf>)

1. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: Identity Mappings in Deep Residual Networks (<https://arxiv.org/pdf/1512.03385.pdf>
2. <https://www.liebertpub.com/doi/pdf/10.1089/big.2021.0218>
3. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015. (<https://www.mygreatlearning.com/blog/introduction-to-vgg16/#VGG%2016%20Use%20Cases>)
4. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989 (<https://www.mygreatlearning.com/blog/introduction-to-vgg16/#VGG%2016%20Use%20Cases>)
5. <https://arxiv.org/pdf/1505.06798.pdf> (<https://neurohive.io/en/popular-networks/vgg16/>)
6. <https://iq.opengenus.org/vgg19-architecture/>
7. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
8. <https://www.researchgate.net/publication/343452618_Breast_Cancer_Histopathology_Image_Classification_Using_an_Ensemble_of_Deep_Learning_Models/download>
9. <https://link.springer.com/content/pdf/10.1007/s12652-021-03488-z.pdf>
10. <https://reader.elsevier.com/reader/sd/pii/S2666285X21000558?token=43F9B2AB4B596B280D92B4BCEE125AAE02D6942697558CD3CDF7345EF452534679319756498CA0AAAADCFD62F6F58C07&originRegion=eu-west-1&originCreation=20220802170823>
11. <https://maelfabien.github.io/deeplearning/xception/#>
12. F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 vol. 2017-January, ed: Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1800-1807.( <https://arxiv.org/ftp/arxiv/papers/2003/2003.11055.pdf>)
13. https://arxiv.org/pdf/1610.02357.pdf (<https://iq.opengenus.org/xception-model/>)
14. <https://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf>