**Going deeper with convolutions**

*Christian Szegedy, et al.*

* Source: <https://arxiv.org/pdf/1409.4842.pdf>
* ILSVRC 2014
* The version of the Inception architecture used in the paper is called GoogLeNet
* GoogLeNet has significantly less no. of parameters compared to AlexNet but is more accurate.
* Model was designed to be used in low power/memory devices
* Model uses 1\*1 convolutions to reduce the no. of computations. This allows the model to have more depth and wider layers without increasing computations too much.
* One way to improve a model’s accuracy is to use wider layers and make the model deeper. However, this has two issues:
  + Model becomes more susceptible to overfitting
  + It needs more computational resources and takes longer to train
* So, authors came up with a new CNN architecture – inception
* Inception module:

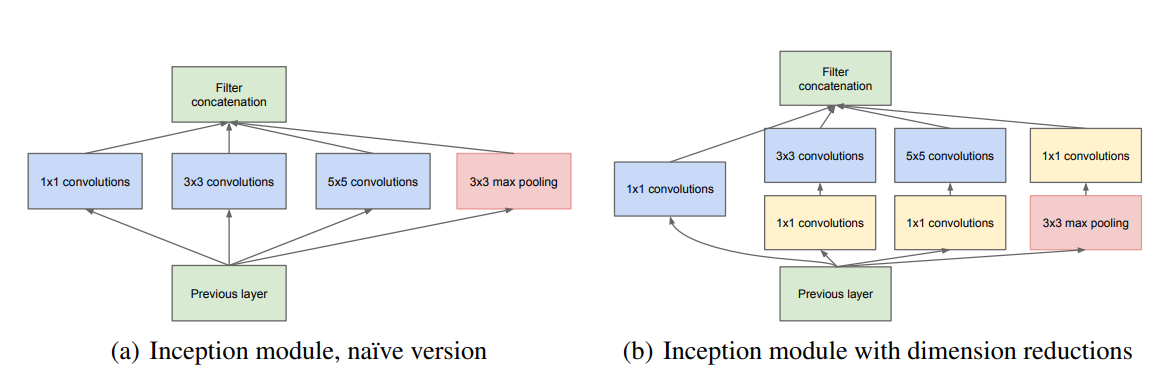
[*https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202*](https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202)

*The object we want to classify may be of any size in an input image. For example, if a picture is taken closely, the object will occupy large portion of the image; on the other hand, if the picture is taken from a far, the object will cover only a small portion of the image.*

*The solution to this is to use filters of different shapes at each stage. This is the fundamental idea of the Inception module.*

The module uses conv layers of various filter shapes and a max-pool layer simultaneously and then combines their outputs along channels. This, then, is passed to the next stage.

This is shown in the below image on the left.



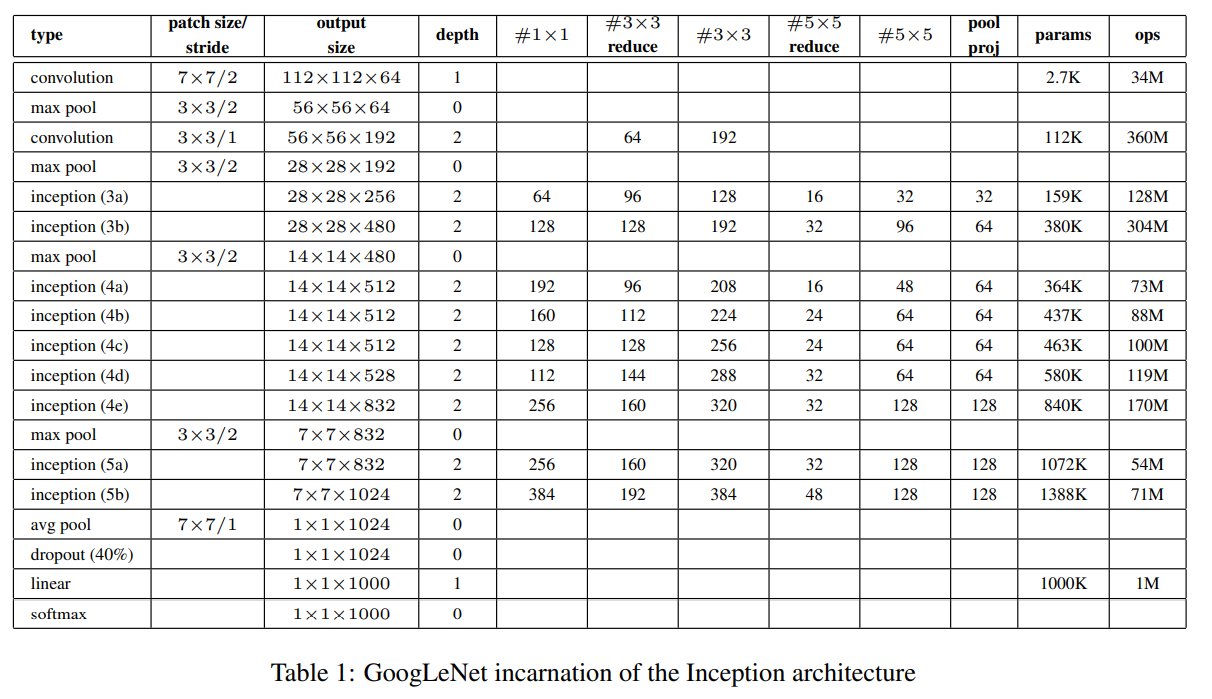
However, using such model leads to extreme increase in the number of computations, even if the no. of filters is kept small.

So, 1\*1 conv. layer was proposed. Just before a 3\*3 conv layer, a 1\*1 conv layer is used. This reduces the no. of channels. Then, the output of this 1\*1 conv layer is passed to the 3\*3 conv. layer.

Similar approach is applied to 5\*5 conv layer.

A 1\*1 conv. layer is used with max-pool layer as well, but while 1\*1 conv. layer comes before 3\*3 and 5\*5 conv. layers, it comes after a max-pool layer.

The modified inception module is given on the right in the above image.

* The overall GoogLeNet model is created by stacking such inception modules with occasional max-pooling layers with stride 2
* GoogLeNet uses the standard conv. layers in the initial layers of the model instead of using inception modules, but you may use inception modules as well.
* The 1\*1 conv. layers allow creating models with more such modules without increasing computations too much.
* Using multiple filters of various shapes and combining their outputs aligns with the intuition that information exists at various scales.
* 

*#3\*3* column specifies the no. of 3\*3 filters in the inception module. Similarly for *#5\*5* column

*#3\*3 reduce* column specifies the no. of 1\*1 filters in the inception module just before the 3\*3 conv layer. Similarly for *#5\*5 reduce* column

* All conv. layers, including those inside the inception module, use ReLU
* The reduction/projection layers also used ReLU
* Input size is 224\*224\*3
* The model is 22 layers deep if counting only the layers having parameters.

The overall number of layers (independent building blocks) used for the construction of the network is about 100.

* Using Average pooling layer, instead of FC layer, just before the output FC layer improved accuracy by 0.6%
* Auxiliary Classifiers:

Because the model is very deep, there was a need to address vanishing/exploding gradients.

To deal with this, authors used auxiliary classifiers on top of two intermediate inception modules, 4a and 4d.

During training the loss from these networks get added to the loss from the rest of the network with some discount weight (0.3 weight)

During testing, these auxiliary networks are discarded.

* Auxiliary network architecture:
  + An average pooling layer with 5×5 filter size and stride 3, resulting in an 4×4×512 output for the (4a), and 4×4×528 for the (4d) stage
  + A 1×1 conv with 128 filters for dimension reduction and ReLU
  + A FC layer with 1024 units and ReLU
  + A dropout layer with 70% ratio of dropped outputs
  + A linear layer with softmax loss as the classifier (predicting the same 1000 classes as the main classifier)
* Training:

SGD with 0.9 momentum, fixed learning rate schedule (decreasing LR by 4% after every 8 epochs)

Sampling patches of various sizes from input images works well.

* Classification Results:

ILSVRC 2014: 1.2 million training images, 50k validation images, and 100k testing images

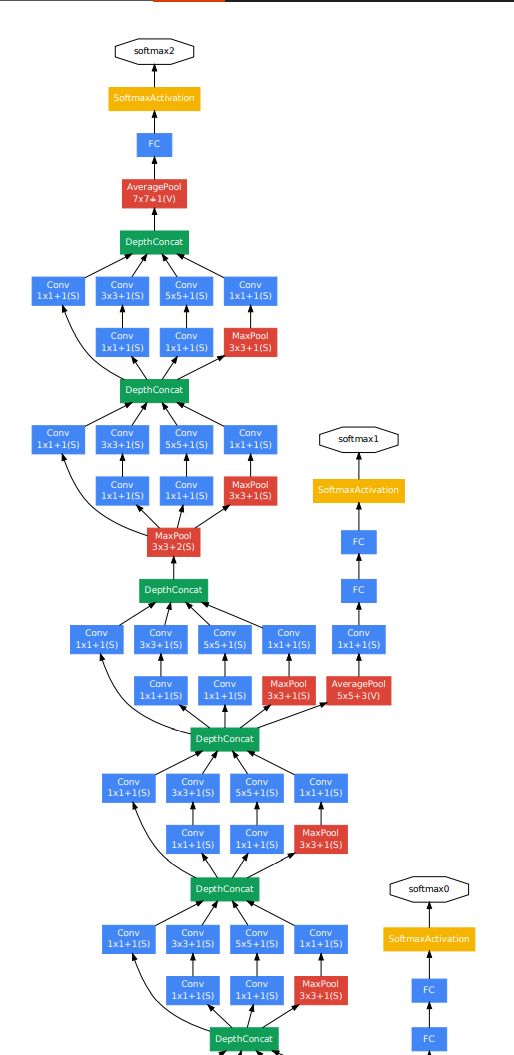
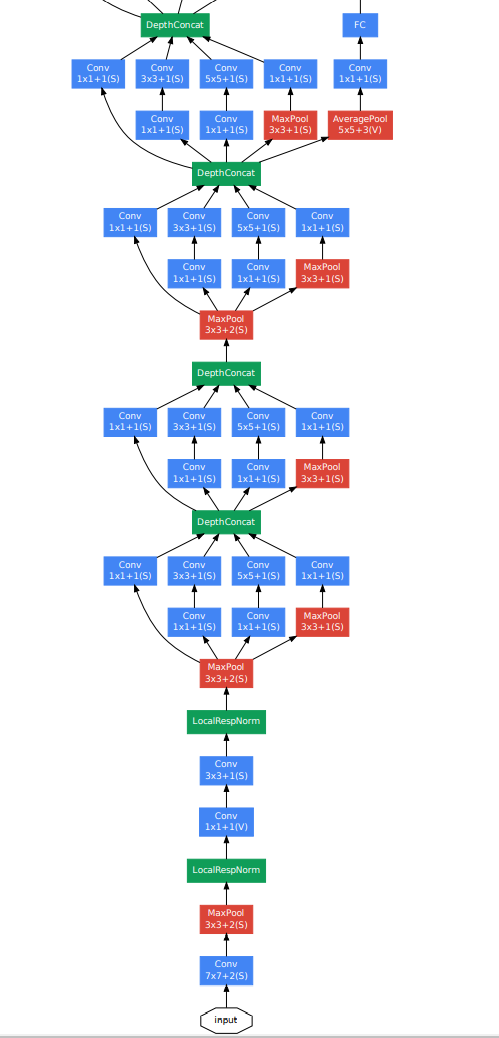
Independently trained 7 versions of GoogLeNet model, which includes one wider version

During testing, each image is resized to 4 scales: 256, 288, 320, and 352. Then, take three square sections of the image (left, centre, and right for landscape images and top, centre, and bottom for portrait images).

For each square, take crops of shape 224\*224 from each corner and the centre. Resize the square to 224\*224 to obtain one more crop. Include image reflections as well of all the crops.

To obtain the final prediction for an image, take average of the outputs over all scales, crops, and classifiers in the ensemble.

* Final submission achieved 6.67% top-5 error, ranking the first in the 2014 competition.



* In the above image the model is split into images. The *“DepthContent”* and *“FC”* layers at the top of the first image are same as the *“DepthContent”* and *“FC”* layers at the bottom of the second image.
* Check the diagram of the complete model architecture in the paper (Page 7)

Results:

