**GoogleNet/Inception(2014)**

It contains **1×1 Convolution** at the middle of the network.

And **global average pooling** is used at the end of the network instead of using fully connected layers.

Another technique, called **inception module**, is to have different sizes/types of convolutions for the same input and stacking all the outputs.

* The 1×1 Convolution
* Inception Module
* Global Average Pooling
* Overall Architecture
* Auxiliary Classifiers for Training
* Testing Details

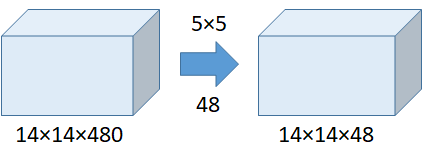
The 1×1 Convolution

1×1 convolution is used with ReLU.

1×1 convolution is used as a dimension reduction module to reduce the computation.

By reducing the computation bottleneck, depth and width can be increased.

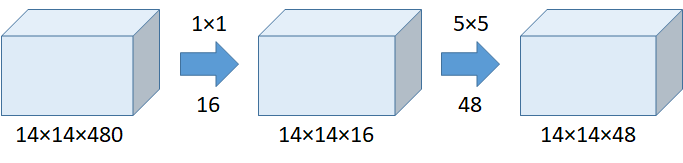
Suppose we need to perform 5×5 convolution without the use of 1×1 convolution as below:



Without the Use of 1×1 Convolution

Number of operations = (14×14×48)×(5×5×480) = 112.9M

With the use of 1×1 convolution:



With the Use of 1×1 Convolution

Number of operations for 1×1 = (14×14×16)×(1×1×480) = 1.5M

**Number of operations for 5×5 = (14×14×48)×(5×5×16) = 3.8M**

**Total number of operations = 1.5M + 3.8M = 5.3M which is much much smaller than 112.9M !!!!!!!!!!!!!!!**

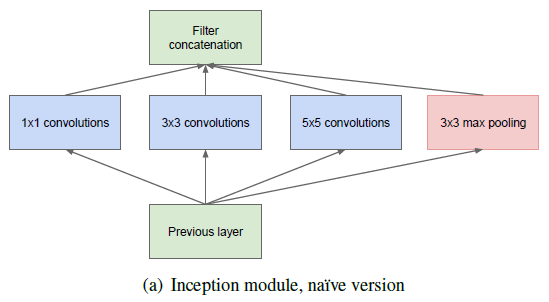
Indeed, the above example is the calculation of **5×5 conv at inception (4a).**

(We may think that, when dimension is reduced, actually we are working on the mapping from high dimension to low dimension in a non-linearity way. In contrast, for PCA, it performs linear dimension reduction.)

Thus, **inception module can be built without increasing the number of operations** **largely!**

Inception Module

The inception module (naive version, without 1×1 convolution) is as below:

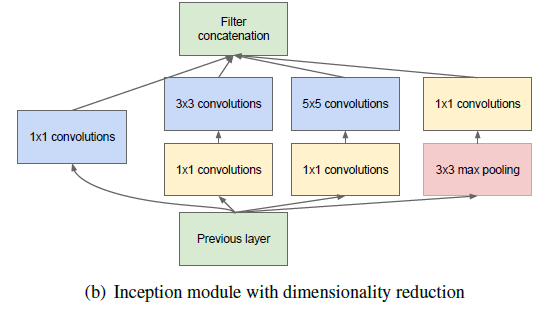


**Inception Module (Without 1×1 Convolution)**

Previously, such as AlexNet, and VGGNet, conv size is fixed for each layer.

Now, 1×1 conv, 3×3 conv, 5×5 conv, and 3×3 max pooling are done altogether for the previous input, and stack together again at output. When image’s coming in, let the network choose the right path.

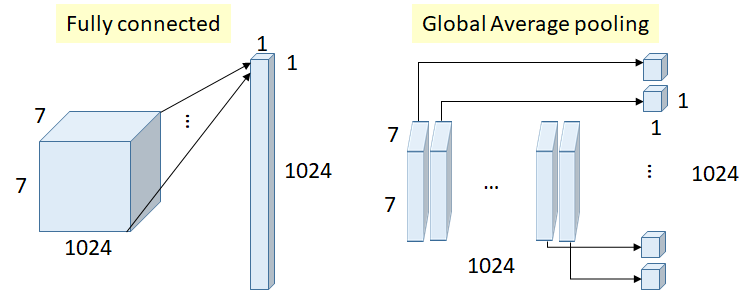
However, without the 1×1 convolution as above, we can imagine how large the number of operation is!



**Inception Module (With 1×1 Convolution)**

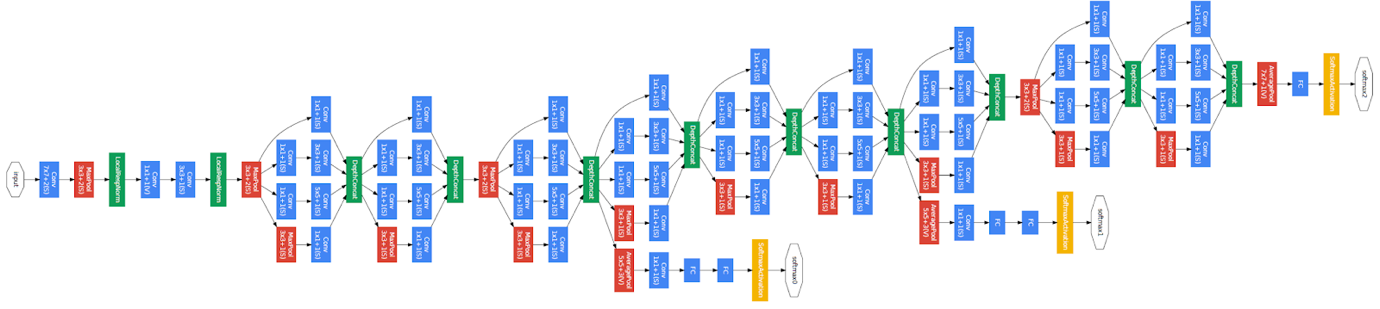
**Thus, 1×1 convolution is inserted into the inception module for dimension reduction!**

Global Average Pooling



**Fully Connected Layer VS Global Average Pooling**

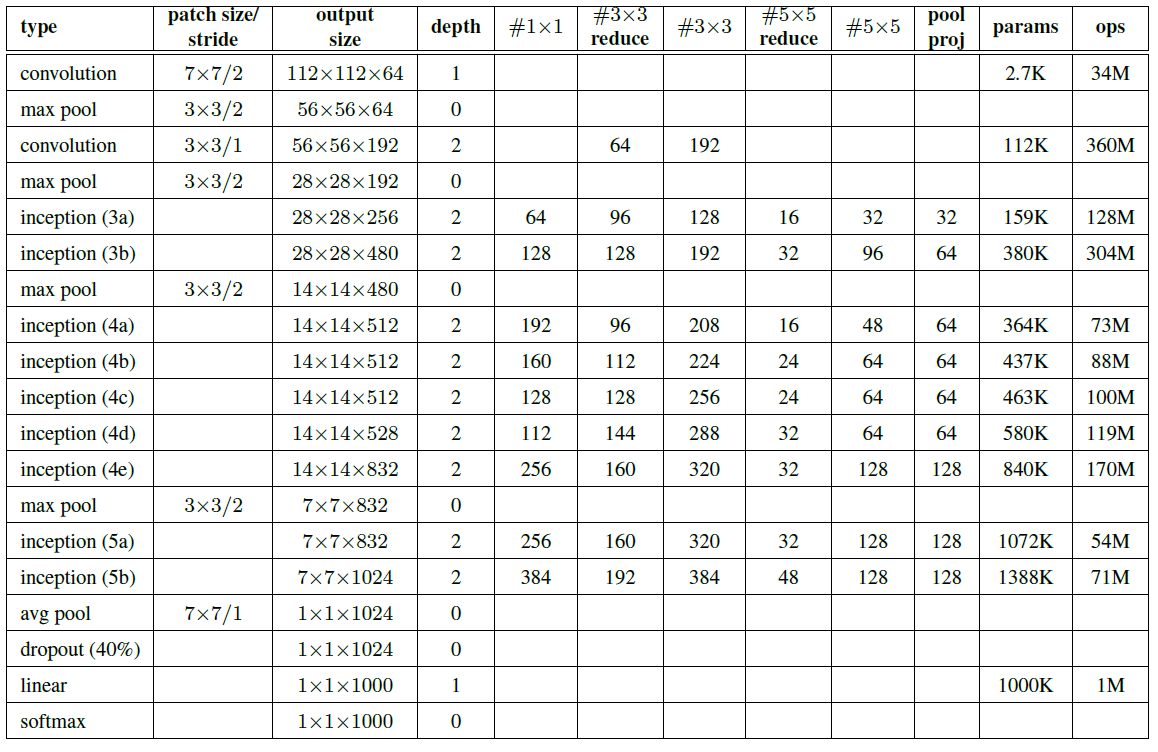
* Previously, fully connected (FC) layers are used at the end of network, such as in AlexNet.
* All inputs are connected to each output.
* Number of weights (connections) above = 7×7×1024×1024 = 51.3M
* In GoogLeNet, global average pooling is used nearly at the end of network by averaging each feature map from 7×7 to 1×1, as in the figure above.
* Number of weights = 0
* And authors found that a move from FC layers to average pooling improved the top-1 accuracy by about 0.6%.
* This is the idea which can be less prone to overfitting.



**There are 22 layers in total**

It is already a very deep model compared with previous AlexNet, ZFNet and VGGNet.And we can see that **there are numerous inception modules connected together to go deeper.** (There are some intermediate softmax branches at the middle, we will describe about them in the next section.)

Below is the details about the parameters if each layer. We actually can extend the example of 1×1 convolution to calculate the number of operations by ourselves. :)



* As we can see there are some intermediate softmax branches at the middle, they are used for training only. These branches are auxiliary classifiers which consist of:
  + 5×5 Average Pooling (Stride 3)
  + 1×1 Conv (128 filters)
  + 1024 FC
  + 1000 FC
  + Softmax
* The loss is added to the total loss, with weight 0.3.
* Authors claim it can be used for combating gradient vanishing problem, also providing regularization.