Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich

...

PRESENTED BY:
Anupama patel (B17EE009), Anshu Priya (B17EE008), Aditi Tiwari (B17ME003)

The paper won the ImageNet Visual Recognition Challenge (in 2014), which is a reputed platform for benchmarking image classification and detection algorithms

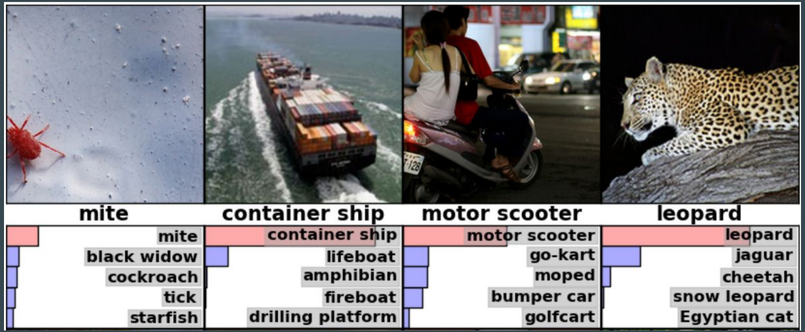
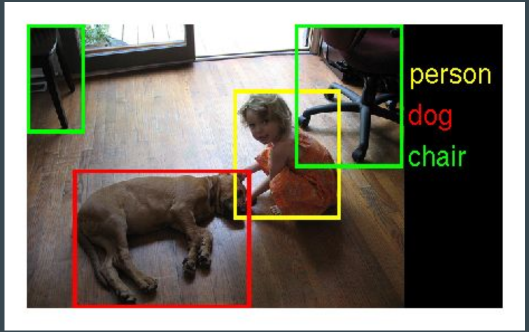


Image classification



object detection

- ImageNet, is a dataset of over 15 millions labeled high-resolution images with around 22,000 categories. ILSVRC uses a subset of ImageNet of around 1000 images in each of 1000 categories. In all, there are **roughly 1.2 million training images, 50,000 validation images and 100,000 testing images.**
- The final network used for submission is called as **GoogleNet** but since inception modules were used in the architecture hence famously known as **inception V1 (version1) network** also.

In the words of the author:

“In this paper, we will focus on an efficient deep neural network architecture for computer vision, codenamed Inception, which derives its name from the Network in network paper in conjunction with the famous “we need to go deeper” internet meme ”

But with less Parameters !!!

GoogLeNet used in these paper actually used 12x fewer parameters than AlexNet used 2 years ago in 2012 competition with improved performance

Problem :

At that time a simple but powerful way of creating better deep learning models was to just make a bigger model, either in terms of deepness, i.e., number of layers, or the number of neurons in each layer i.e width of the network

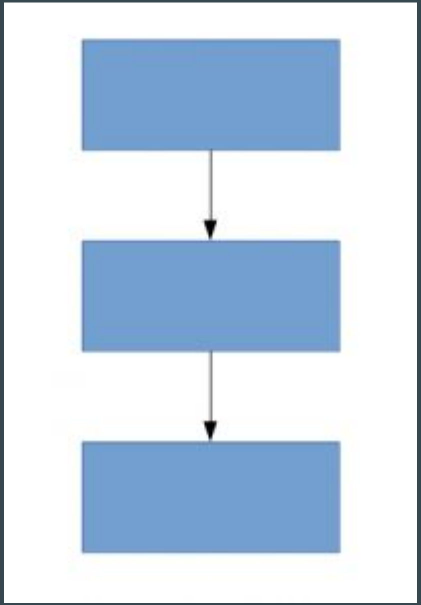
But deeper the network more are the complications

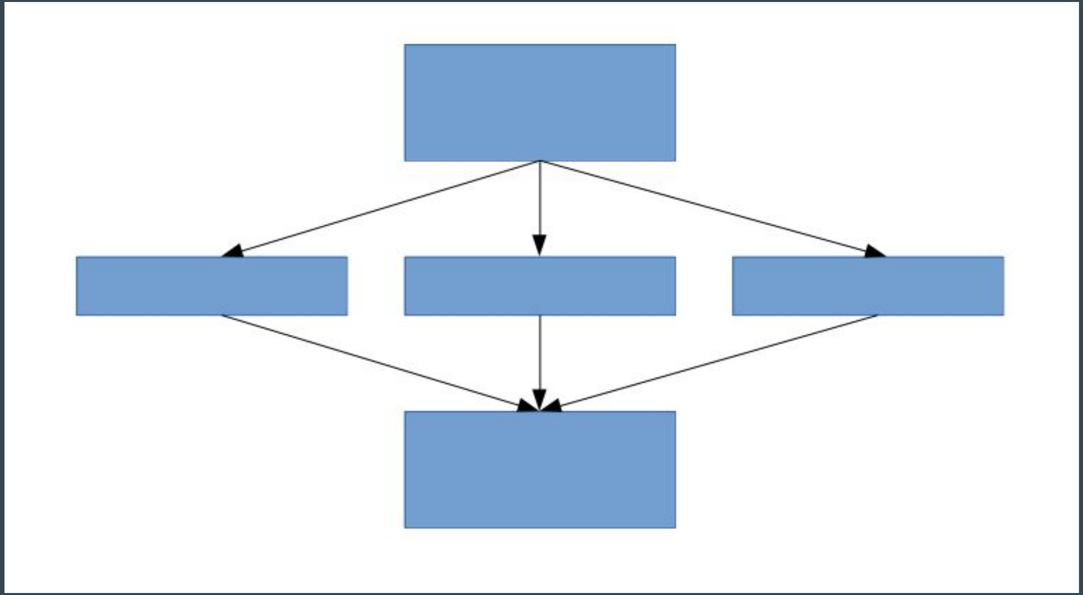
Bigger size of neural networks corresponds to **larger number of parameters, that makes network more prone to overfitting,** especially when labeled training examples are limited.

Increased network size is **increased use of computational resources**. If more convolution layers are chained then there results in more **consumption of computation resources.**

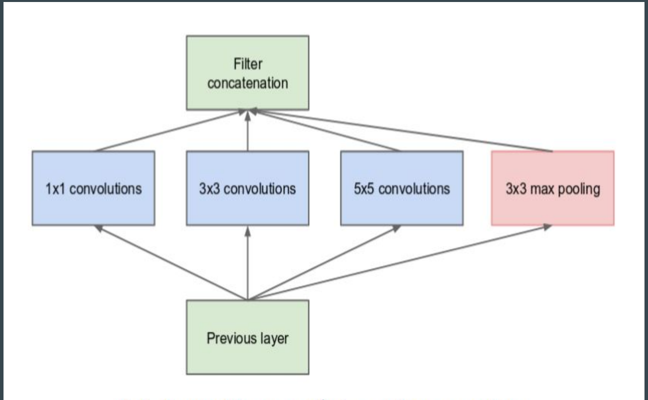
Solution:

form a ‘wider’ network rather than ‘deeper’ which is called as Inception module.


Densely Connected Architecture


Sparsely connected architecture

Inception module



(a) Inception module, naïve version

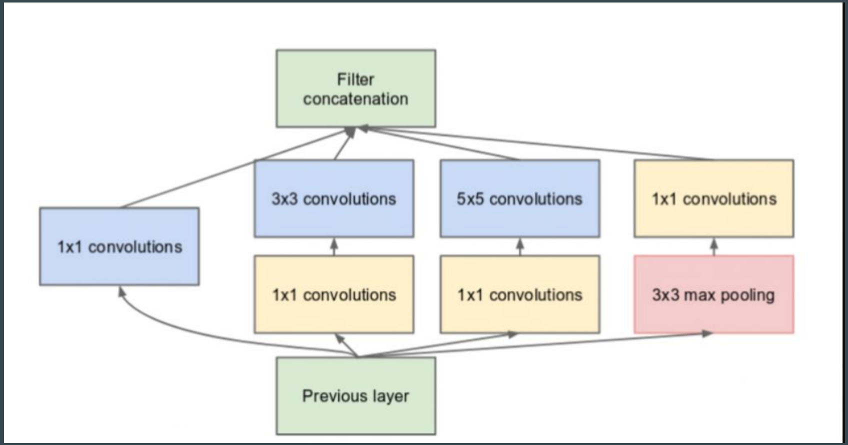
- The ‘naive’ inception module performs convolutions on input from previous layer, with 3 different size of kernels or filters specifically 1×1, 3×3, and 5×5. Outputs are then concatenated and sent to the next inception module.
- In addition to this, max pooling is also performed serves to control overfitting by reducing spatial size.
- Hebbian principle from human learning - says that **“neurons that fire together, wire together”.** The author suggests that **when creating a subsequent layer in a deep learning model, one should pay attention to the learnings of the previous layer.**

- Salient parts in the image can have extremely large variation in size. For instance, an image with a dog can be either of the following, as shown below. Note that the area occupied by the dog is different in each image.

- Because of this huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough. **A larger kernel is preferred for information that is distributed more globally, and a smaller kernel is preferred for information that is distributed more locally.**
- **inception module allows the internal layers to pick and choose which filter size will be relevant to learn the required information.So even if the size of the face in the image is different as seen in the images, the layer works accordingly to recognize the face. For the first image, it would probably take a higher filter size, while it'll take a lower one for the third image.**

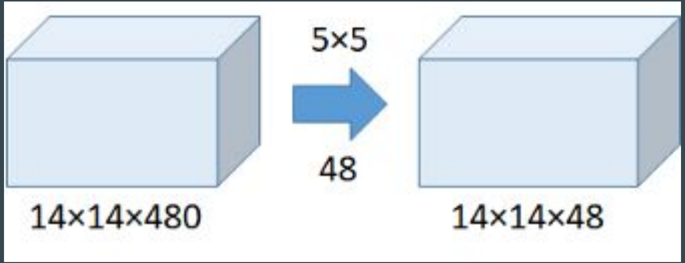
1X1 convolution

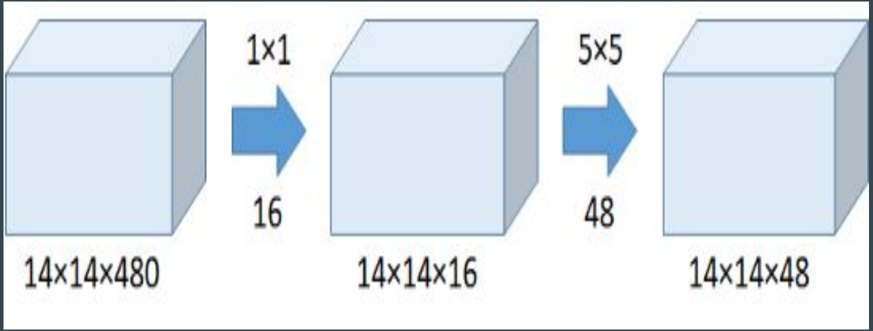
One problem to the 'naive' approach is, even having 5×5 convolutions can lead to require large resource in terms of computations hence the author come up with 1×1 convolution with ReLU



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

example:





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:

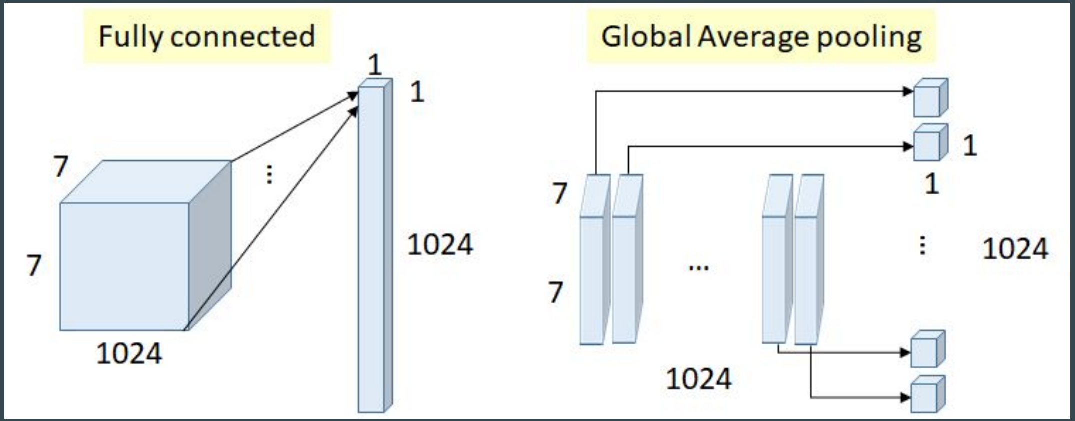
Number of operations for 1×1 = (14×14×16)×(1×1×480) = 1.5M

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 !!!!!!!!!!!!!!!!

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.

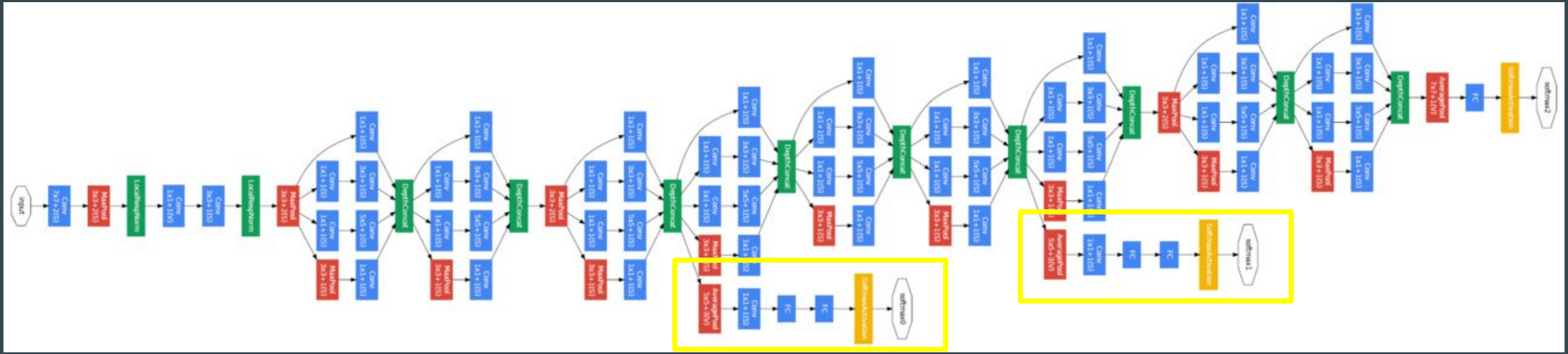
Number of weights = 7×7×1024×1024 = 51.3M

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

It was found that a move from FC layers to average pooling improved accuracy by about 0.6%.

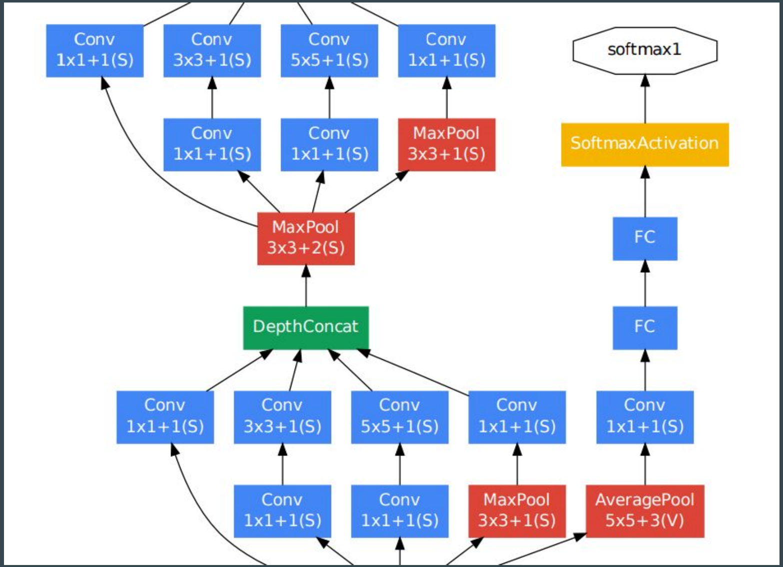
Overall Architecture



- GoogLeNet has 9 such inception module with dimensionality reduction stacked linearly.
- It is 22 layers deep (27, including the pooling layers).
- All the convolutions, including the convolutions inside inception module , uses rectified linear activation
- It uses global average pooling instead of fully connected layer

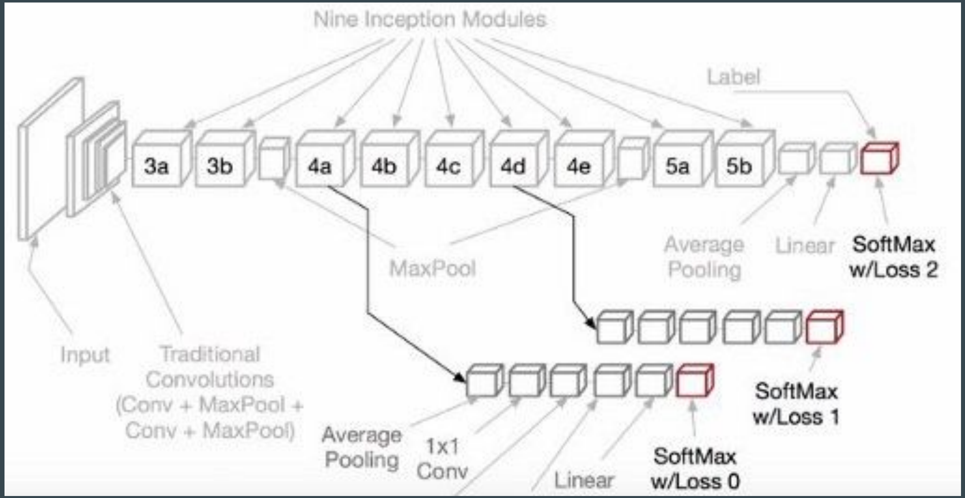
Auxiliary Classifiers for Training

- Like any other deep network, an inception network is a pretty deep network that is subject to the **vanishing gradient problem.**
- To prevent the middle part of the network from "dying out", the authors introduced the **intermediate classifiers** to solve this vanishing gradient problem represented by the yellow blocks in the google Inception network Architecture figure shown in last slide.



Here is a zoomed-out crop from the google inception network to better illustrate these intermediate classifiers.

- These classifiers take the form of smaller convolutional networks put on top of the output of the Inception (4a) and (4d) modules.
- During training, their loss gets added to the total loss of the network with a discount weight (the losses of the auxiliary classifiers were weighted by 0.3).
- At inference time, these auxiliary networks are discarded.



```
   The total loss used by the inception net during training
total_loss = real_loss + 0.3 * aux_loss_1 + 0.3 * aux_loss_
```

Summary

1. 1×1 convolution is used as a dimension reduction module to reduce the computation. 1x1 convolutions before passing it to 3x3 and 5x5 convolutions has proven **efficient and effective.**
2. Different sizes of convolutions (different filter size) done in inception module helps to extract different kinds of features hence **allows for large scaling while minimizing processing bottlenecks.**
3. Replacing the fully-connected layers with a global average pooling which averages out the channel values across the 2D feature map, after the last convolutional layer, drastically **reduces the total number of parameters.** This can be understood from AlexNet, where FC layers contain approx. 90% of parameters.
4. Use of a large network width and depth allows GoogLeNet to remove the FC layers **without affecting the accuracy.**
5. Auxiliary Classifiers i.e. intermediate softmax branches at the middle used at time of Training can be used for **combating gradient vanishing problem in deep networks**

References:

1. going deeper with convolution
2. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202
3. https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch/
4. https://medium.com/ml-cheat-sheet/deep-dive-into-the-google-inception-network-architecture-960f65272314
5. https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/