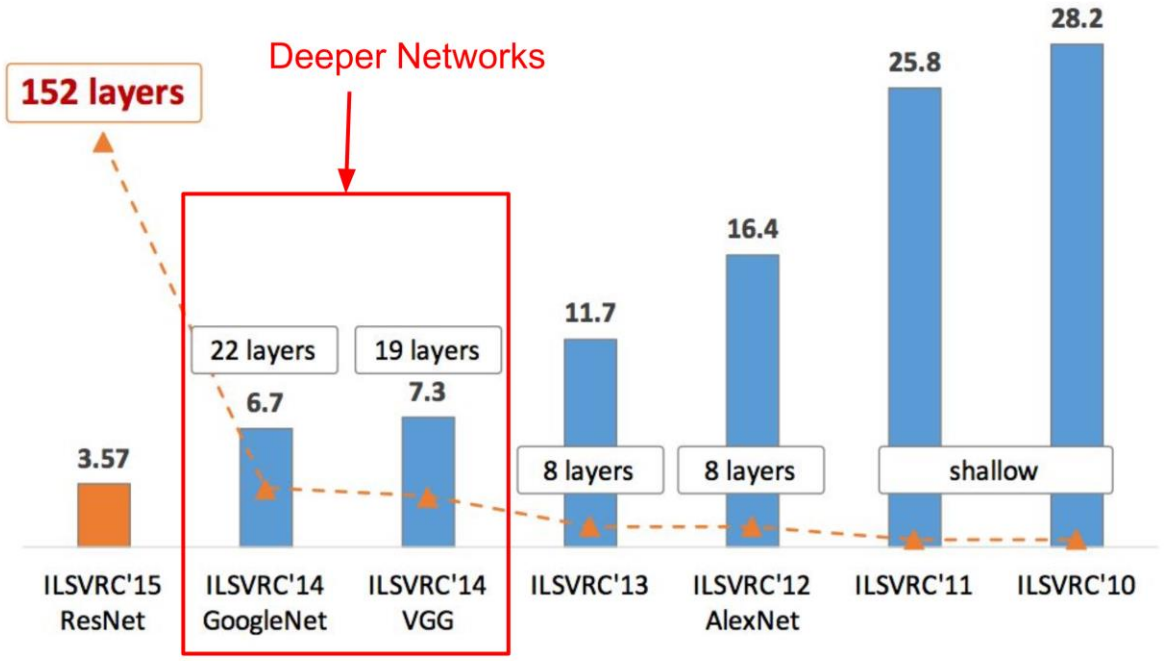
Going deeper with convolutions

2017/09/09

김보섭

What is the GoogLeNet?

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Introduction

Going deeper with convolutions

Christian Szegedy — Google Inc.

Wei Liu — University of North Carolina, Chapel Hill

Yangqing Jia — Google Inc.

Pierre Sermanet — Google Inc.

Scott Reed — University of Michigan

Dragomir Anguelov — Google Inc.

Dumitru Erhan — Google Inc.

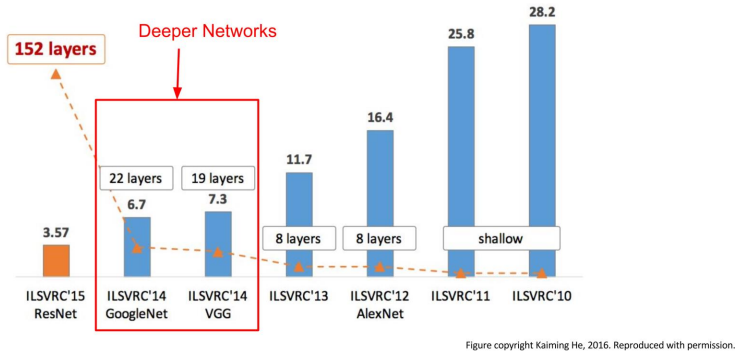
Vincent Vanhoucke — Google Inc.

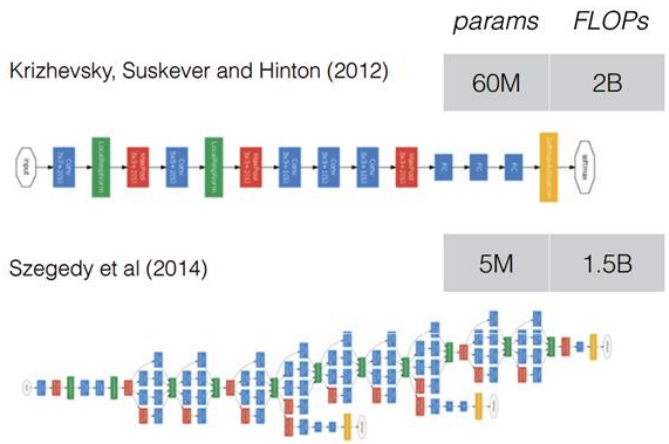
Andrew Rabinovich — Google Inc.

Abstract

We propose a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners





Our **GoogLeNet** submission to ILSVRC 2014 **actually uses 12× fewer parameters than the winning architecture of Krizhevsky et al [9]** from two years ago, while being **significantly more accurate.**

Related work

Network In Network

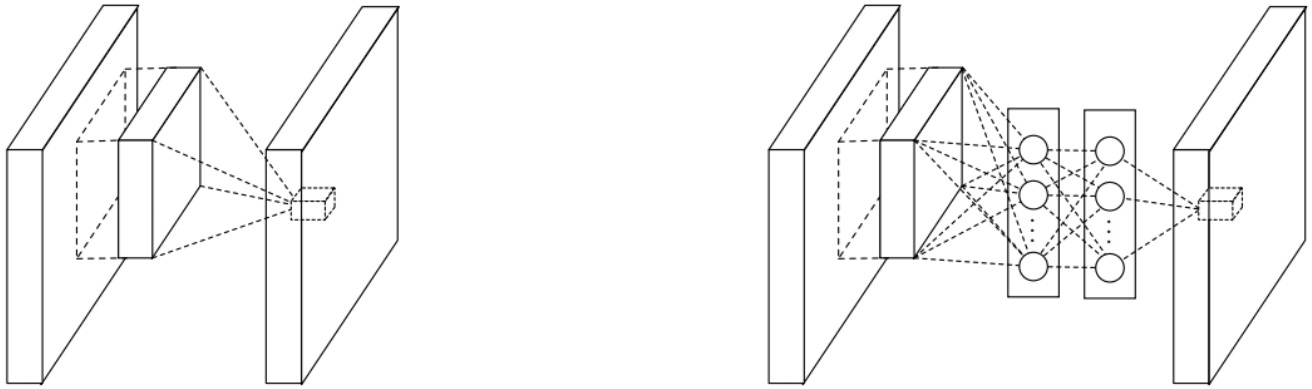
Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]
[1]Graduate School for Integrative Sciences and Engineering
[2]Department of Electronic & Computer Engineering
National University of Singapore, Singapore
{linmin,chenqiang,eleyans}@nus.edu.sg

Abstract

We propose a novel deep network structure called "**N**etwork **I**n **N**etwork"(NIN) to enhance model discriminability for local patches within the receptive field. The conventional convolutional layer uses linear filters followed by a nonlinear activation function to scan the input. Instead, we build micro neural networks with more complex structures to abstract the data within the receptive field. We instantiate the micro neural network with a multilayer perceptron, which is a potent function approximator. The feature maps are obtained by sliding the micro networks over the input in a similar manner as CNN; they are then fed into the next layer. Deep NIN can be implemented by stacking mutiple of the above described structure. With enhanced local modeling via the micro network, we are able to utilize global average pooling over feature maps in the classification layer, which is easier to interpret and less prone to overfitting than traditional fully connected layers. We demonstrated the state-of-the-art classification performances with NIN on CIFAR-10 and CIFAR-100, and reasonable performances on SVHN and MNIST datasets.

Related work

- ✓ **Mlpconv layer**



(a) Linear convolution layer (b) Mlpconv layer

Figure 1: Comparison of linear convolution layer and mlpconv layer. The linear convolution layer includes a linear filter while the mlpconv layer includes a micro network (we choose the multilayer perceptron in this paper). Both layers map the local receptive field to a confidence value of the latent concept.

Network-in-Network is an approach proposed by Lin et al. [12] in order **to increase the representational power of neural networks.** When applied to convolutional layers, **the method could be viewed as additional 1×1 convolutional layers followed typically by the rectified linear activation [9]**

Related work

Yann LeCun
2015년 4월 7일 ·

팔로우 •••

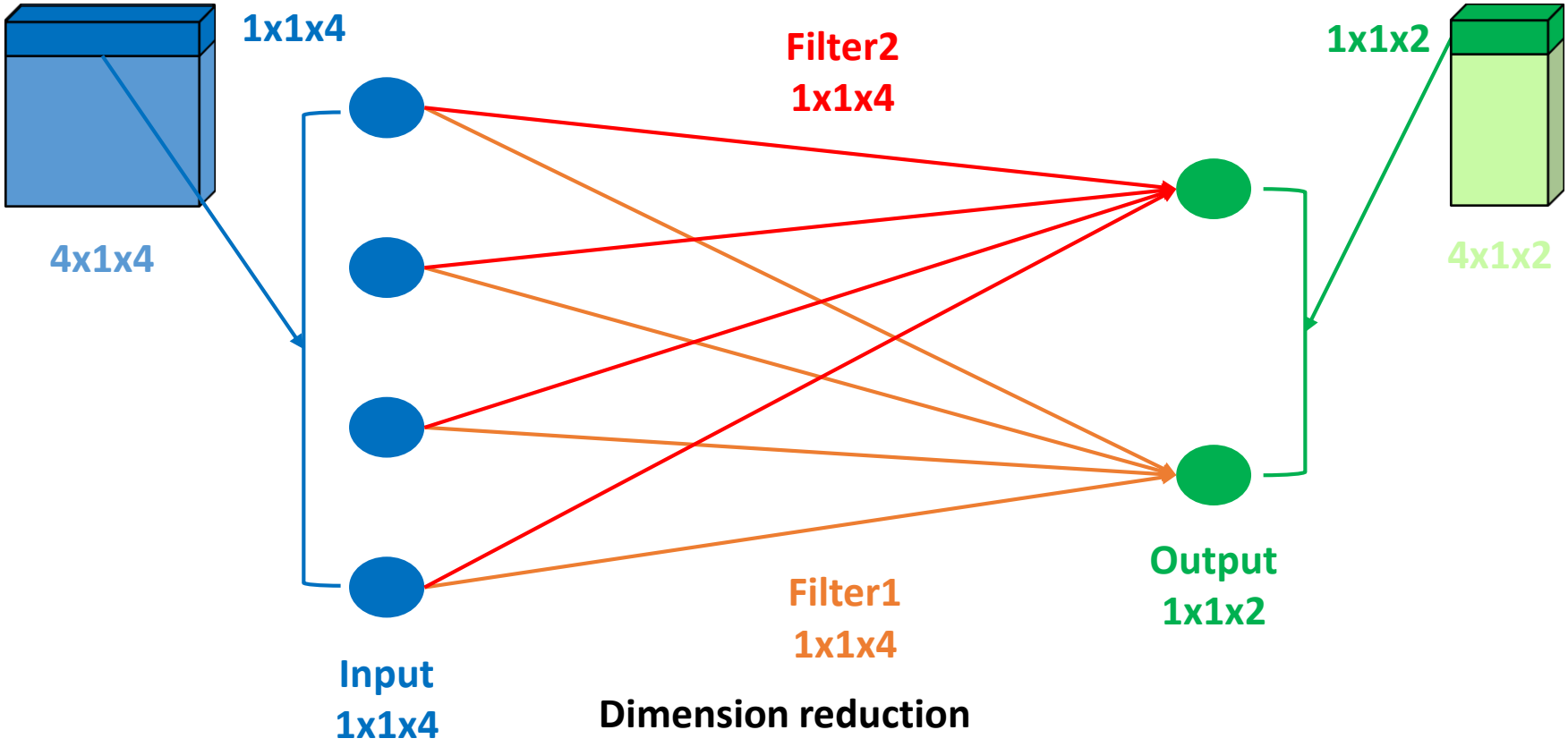
In Convolutional Nets, there is no such thing as "fully-connected layers". There are only convolution layers with 1x1 convolution kernels and a full connection table.

It's a too-rarely-understood fact that ConvNets don't need to have a fixed-size input. You can train them on inputs that happen to produce a single output vector (with no spatial extent), and then apply them to larger images. Instead of a single output vector, you then get a spatial map of output vectors. Each vector sees input windows at different locations on the input.

In that scenario, the "fully connected layers" really act as 1x1 convolutions.

Related work

✓ **1 x 1 Convolution** is equivalent to **1-layer fully-connected neural network**



However, in our setting, **1 × 1 convolutions have dual purpose**: most critically, they are used mainly as **dimension reduction modules to remove computational bottlenecks, that would otherwise limit the size of our networks**. This allows for not just increasing the depth, but also the width of our networks without significant performance penalty.

Related work

- ✓ **Global average pooling**

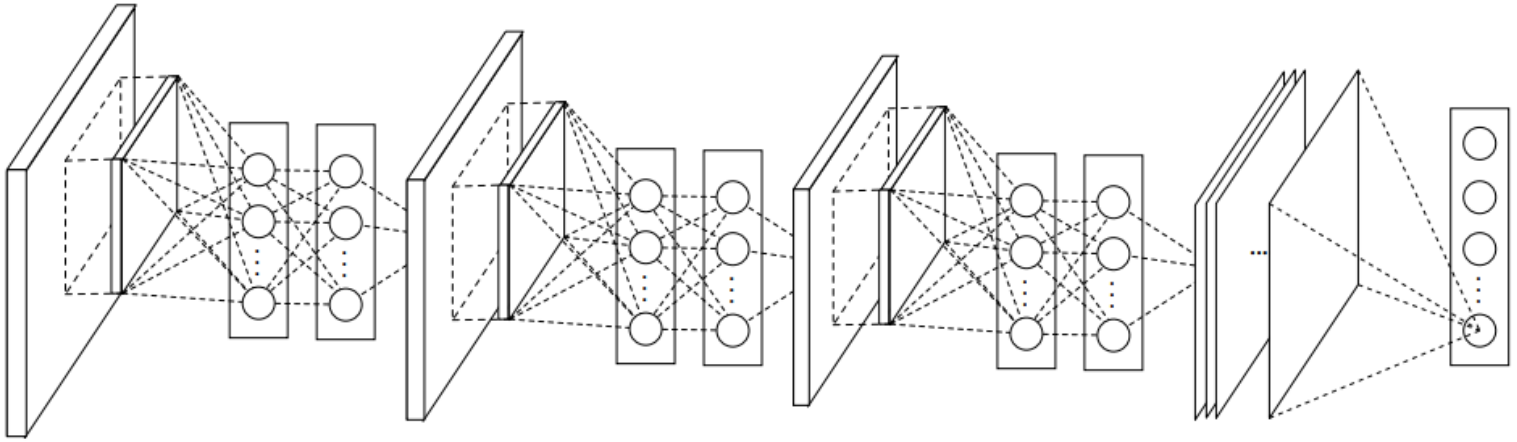


Figure 2: The overall structure of Network In Network. In this paper the NINs include the stacking of three mlpconv layers and one global average pooling layer.

One advantage of global average pooling over the fully connected layers is that it is more native to the convolution structure by enforcing correspondences between feature maps and categories. Thus the feature maps can be easily interpreted as categories confidence maps. **Another advantage is that there is no parameter to optimize in the global average pooling thus overfitting is avoided at this layer. Futhermore, global average pooling sums out the spatial information, thus it is more robust to spatial translations of the input.**

Motivation, Consideration, Architectural Details

✓ **The most straightforward way of improving the performance of deep neural networks is by increasing their size.** This includes both increasing the depth – the number of levels – of the network and its width: the number of units at each level.

This solution comes with two major drawbacks!

1. A large number of parameters

- More prone to **overfitting**
- **Need lots of good data** (expensive)

(a) Siberian husky (b) Eskimo dog

Figure 1: Two distinct classes from the 1000 classes of the ILSVRC 2014 classification challenge.

2. Increased use of computational resources

- For example, in a deep vision network, if two convolutional layers are chained, **any uniform increase in the number of their filters results in a quadratic increase of computation.**

The fundamental way of solving both issues would be **by ultimately moving from fully connected to sparsely connected architectures, even inside the convolutions.**

Motivation, Consideration, Architectural Details

- ✓ Image data is mostly **sparse and clustered**
- ✓ But computation infrastructures are **inefficient for non uniform sparse data structures.**
- ✓ **Clustering sparse matrices into relatively dense submatrices** provide good performance for sparse matrix computations.

Optimal local sparse structure using available dense components

To capture dense clusters : 1x1 convolutions

More spatially spread out clusters captured by 3x3 and 5x5.

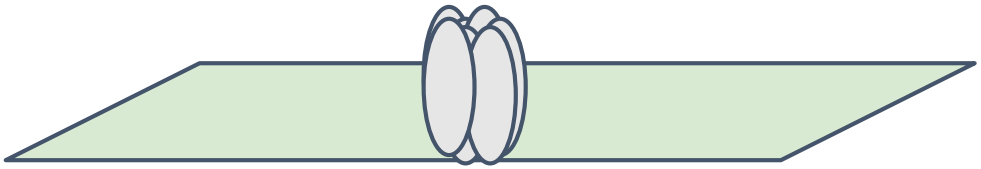
Pooling layer: Generally improves performance.

Outputs of all these are concatenated and passed to next layer

Give rise to the (naive) “Inception Module”

Motivation, Consideration, Architectural Details

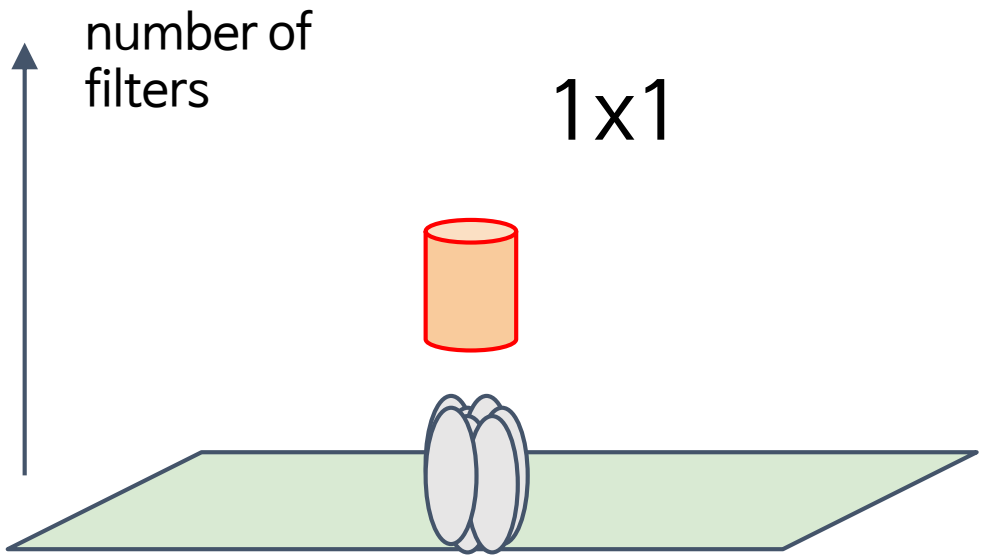
In images, correlations tend to be local



SLIDE CREDIT:GOOGLE INC

Motivation, Consideration, Architectural Details

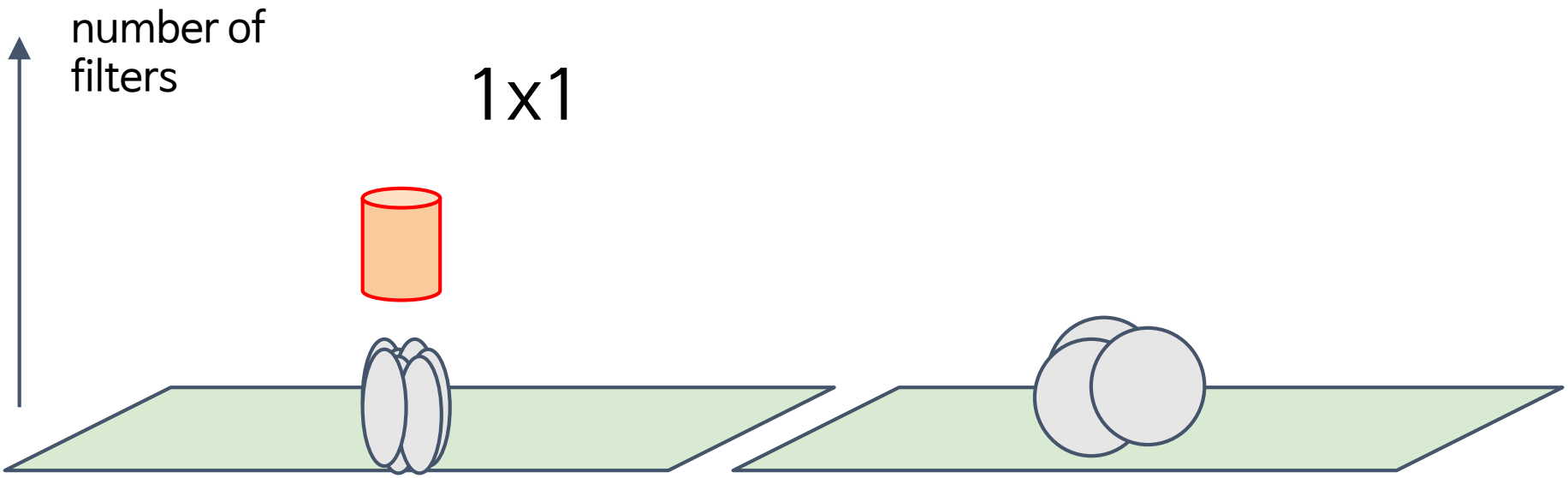
Cover very local clusters by 1x1 convolutions



SLIDE CREDIT:GOOGLE INC

Motivation, Consideration, Architectural Details

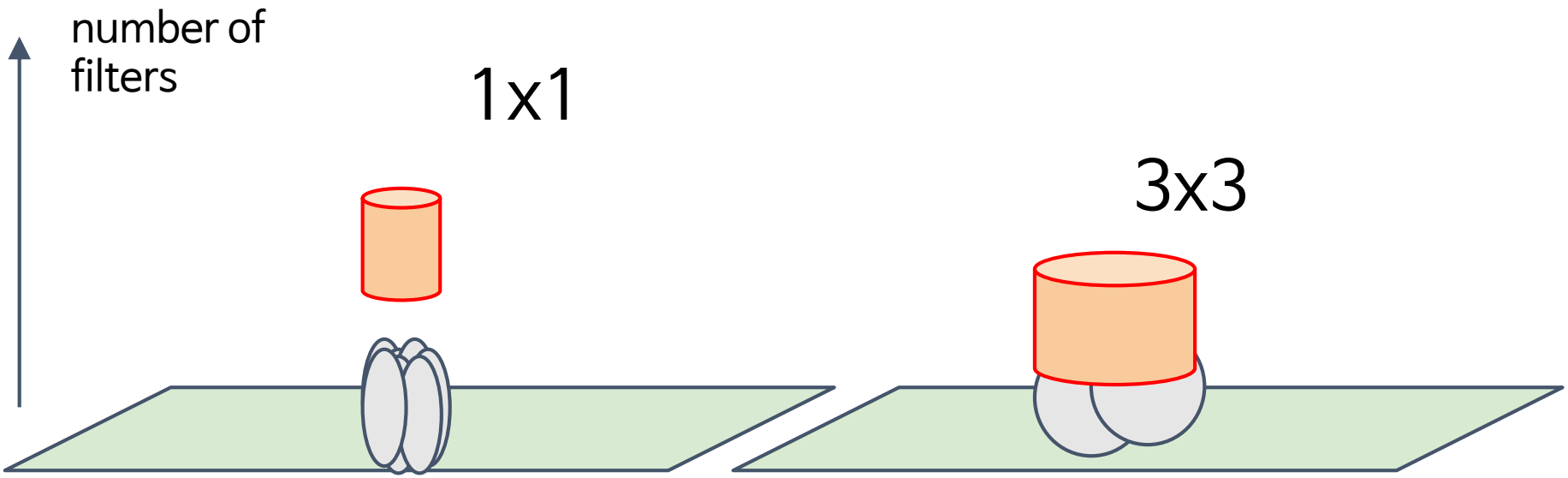
Less spread out correlations



SLIDE CREDIT:GOOGLE INC

Motivation, Consideration, Architectural Details

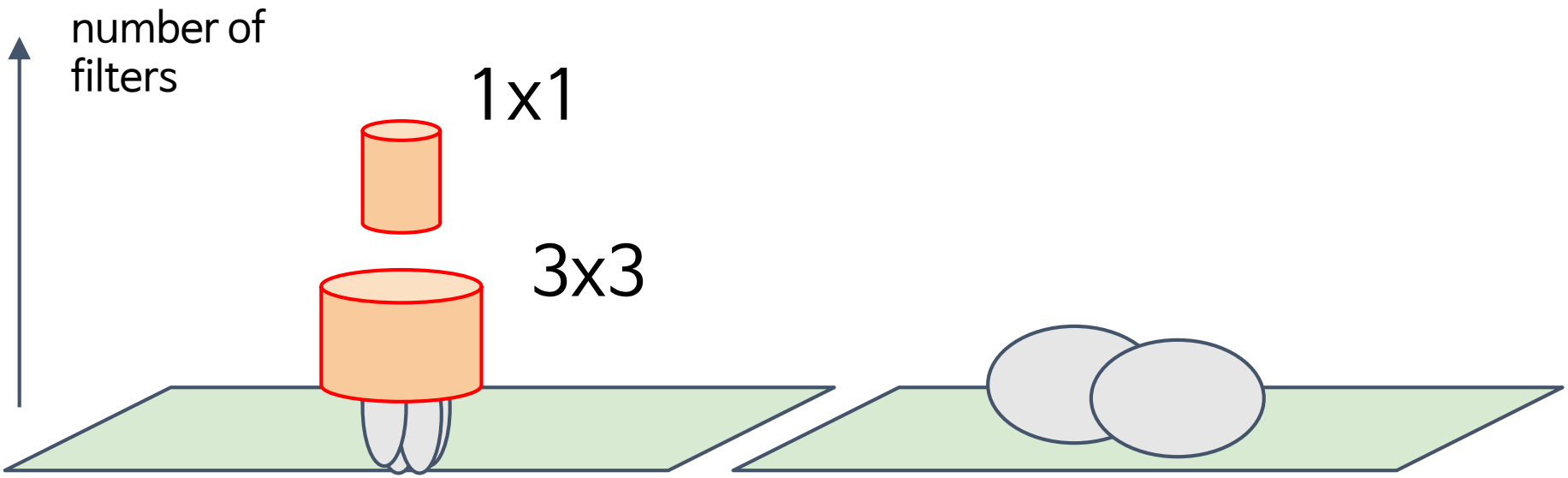
Cover more spread out clusters by 3x3 convolutions



SLIDE CREDIT:GOOGLE INC

Motivation, Consideration, Architectural Details

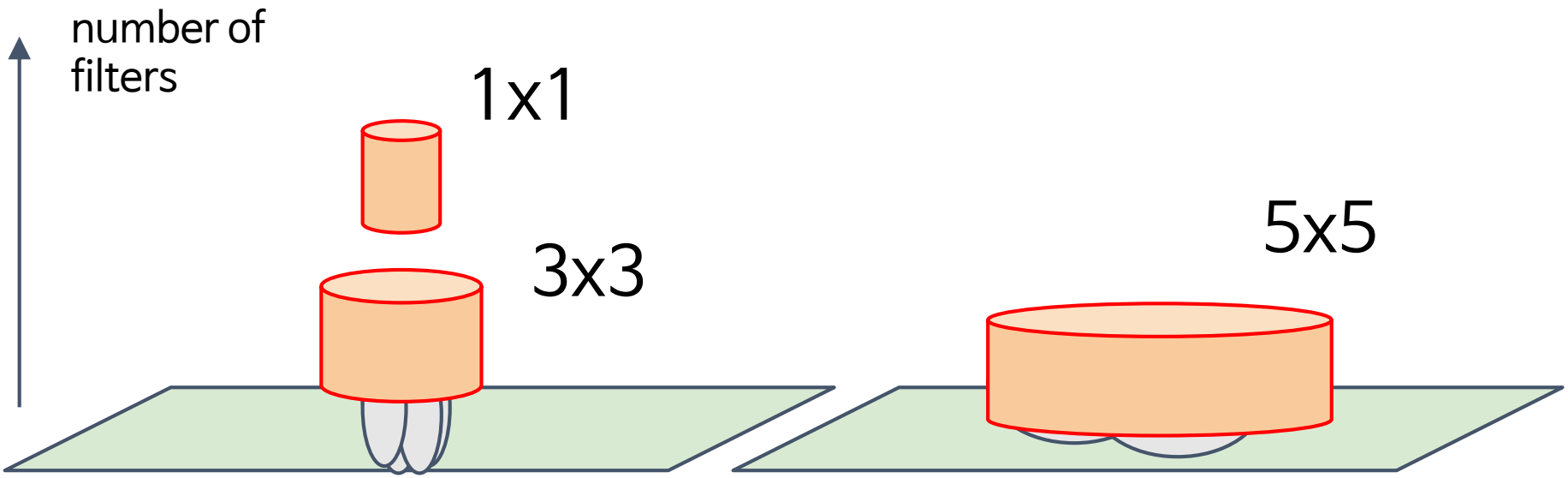
Cover more spread out clusters by 5x5 convolutions



SLIDE CREDIT:GOOGLE INC

Motivation, Consideration, Architectural Details

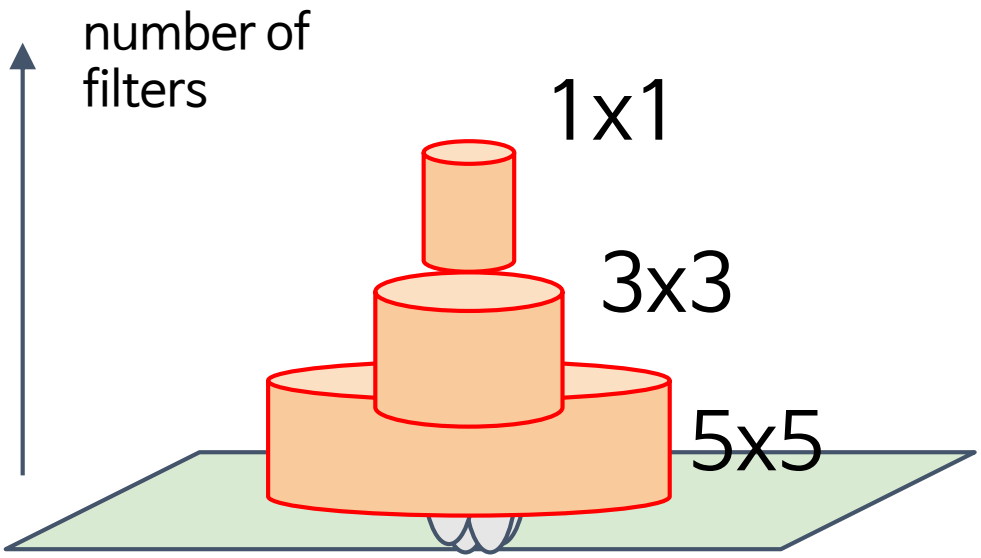
Cover more spread out clusters by 5x5 convolutions



SLIDE CREDIT:GOOGLE INC

Motivation, Consideration, Architectural Details

A heterogeneous set of convolutions

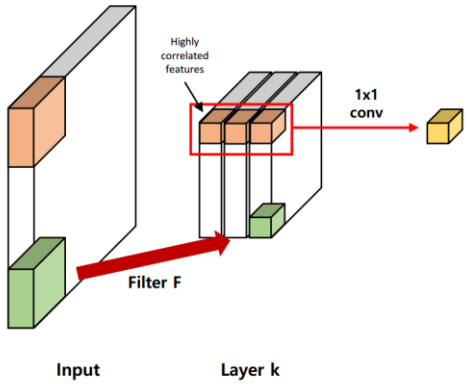


Main idea

- Construct optimal local sparse structure ➡ Repeat it
- "One can should analyze the correlation statistics of the last layer and cluster them into groups of units with high correlation" (Arora et al, 2013)

Assumption

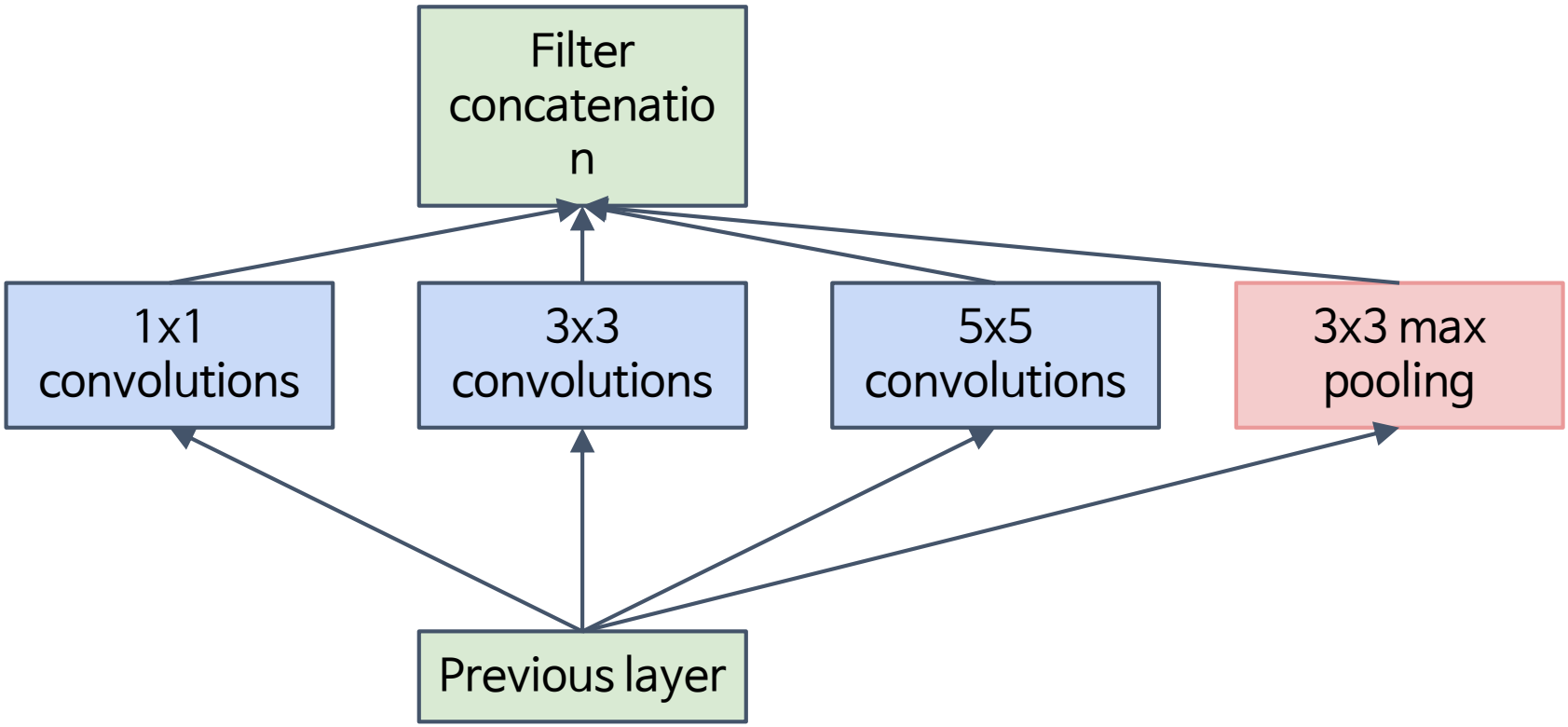
- Each unit from an earlier layer corresponds to some region of the input image
- Units are grouped into filter banks



SLIDE CREDIT:GOOGLE INC

Motivation, Consideration, Architectural Details

Naive idea (does not work!)



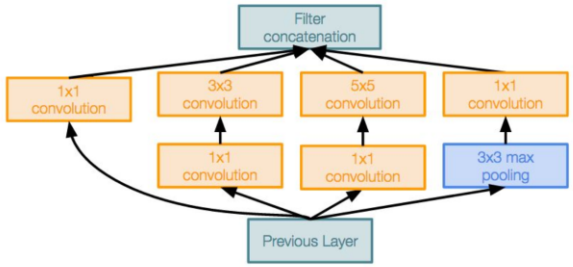
SLIDE CREDIT:GOOGLE INC

Motivation, Consideration, Architectural Details

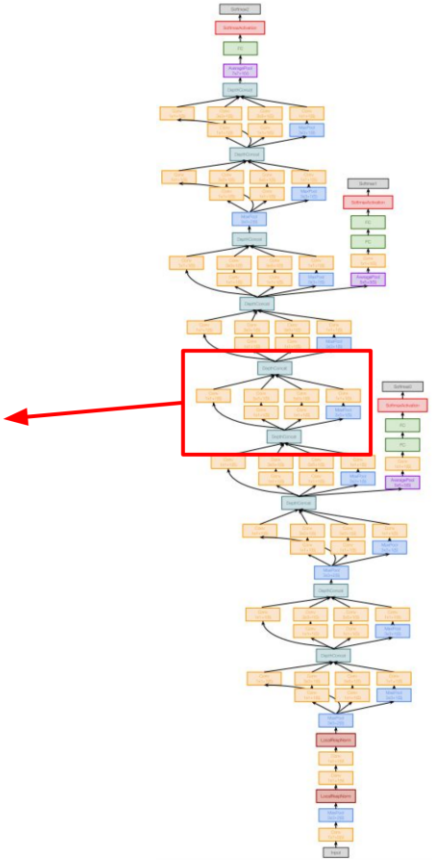
Case Study: GoogLeNet

[Szegedy et al., 2014]

"Inception module": design a good local network topology (network within a network) and then stack these modules on top of each other



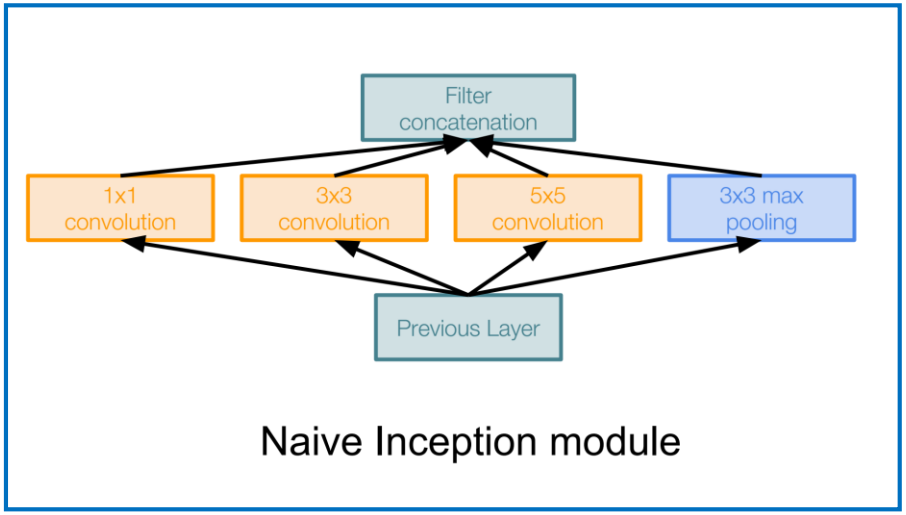
Inception module



Motivation, Consideration, Architectural Details

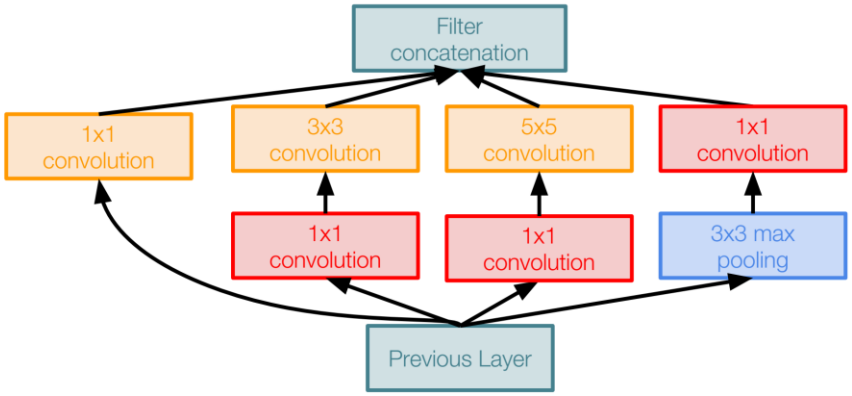
Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

1x1 conv "bottleneck" layers

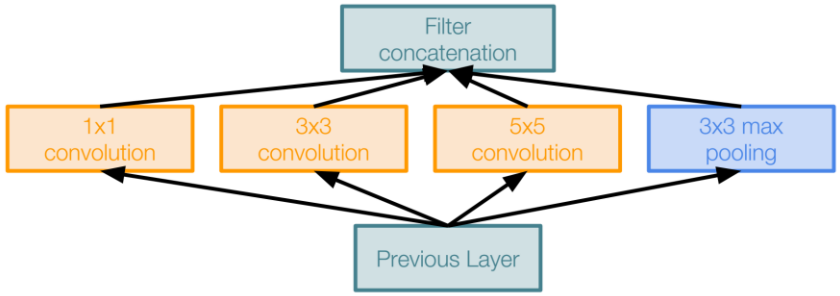


Inception module with dimension reduction

Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

Q: What is the problem with this?
[Hint: Computational complexity]

Motivation, Consideration, Architectural Details

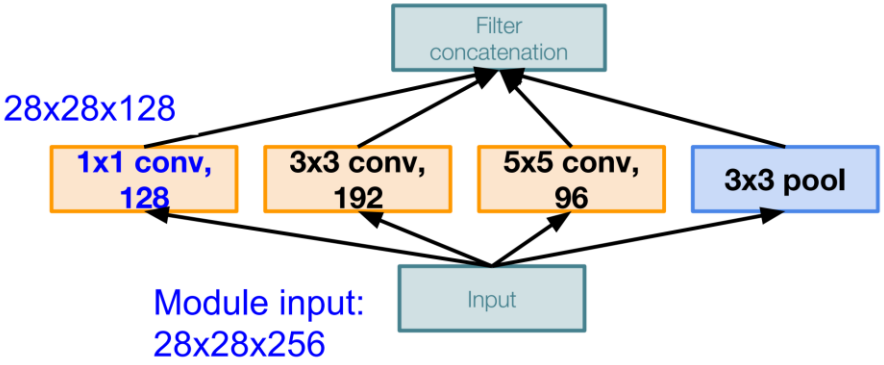
Case Study: GoogLeNet

[Szegedy et al., 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example:

Q1: What is the output size of the 1x1 conv, with 128 filters?



Naive Inception module

Motivation, Consideration, Architectural Details

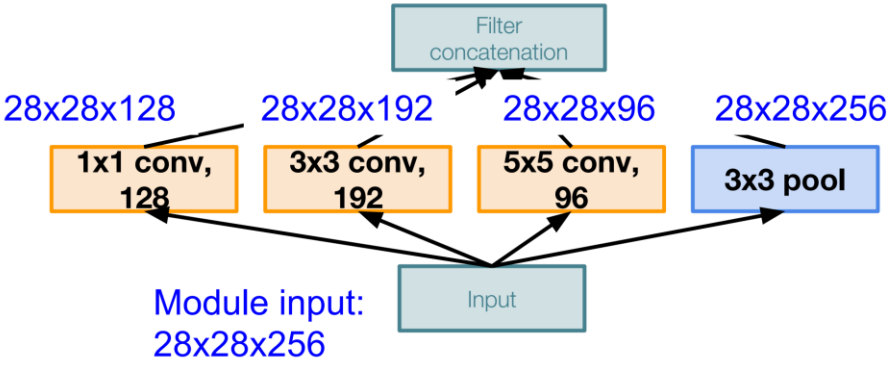
Case Study: GoogLeNet

[Szegedy et al., 2014]

Q: What is the problem with this?
[Hint: Computational complexity]

Example:

Q2: What are the output sizes of all different filter operations?



Naive Inception module

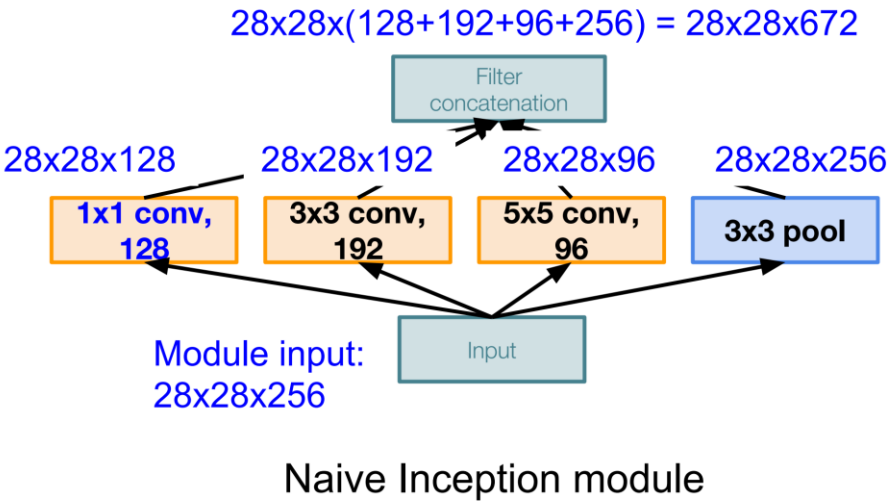
Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3:What is output size after filter concatenation?



Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Very expensive compute

Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

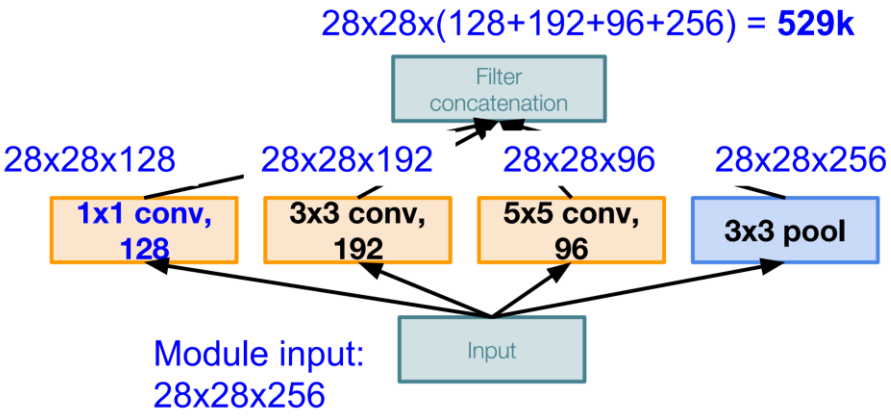
Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3:What is output size after filter concatenation?



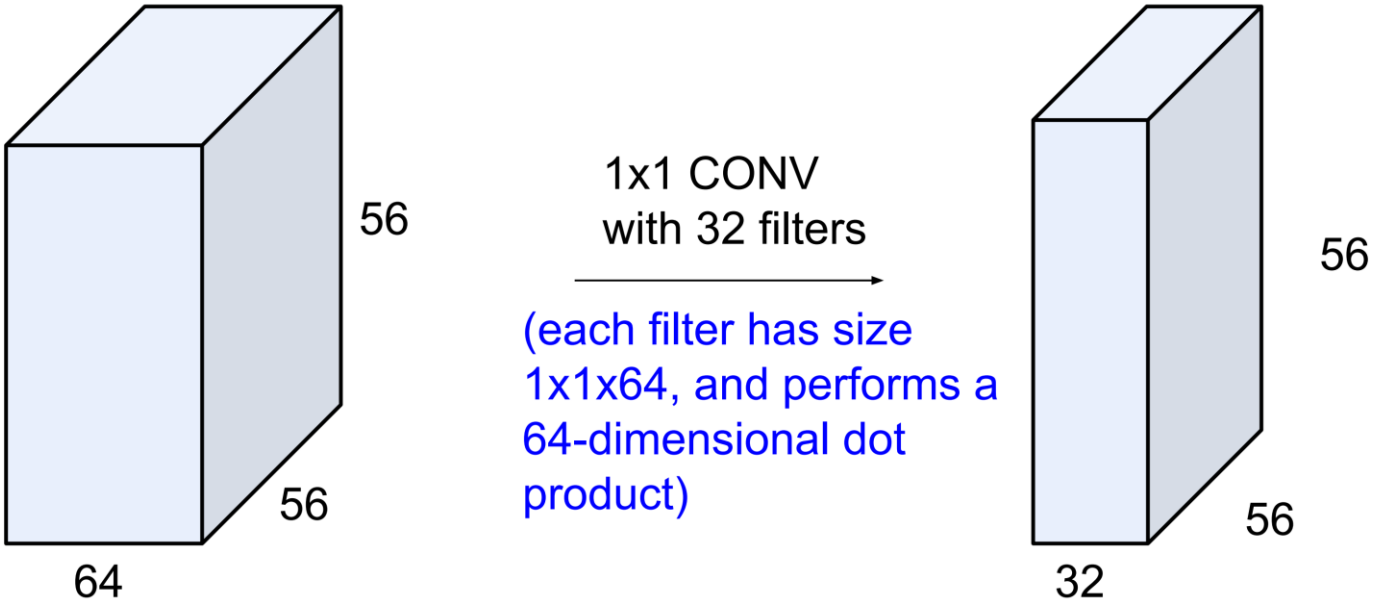
Naive Inception module

Q: What is the problem with this? [Hint: Computational complexity]

Solution: "bottleneck" layers that use 1x1 convolutions to reduce feature depth

Motivation, Consideration, Architectural Details

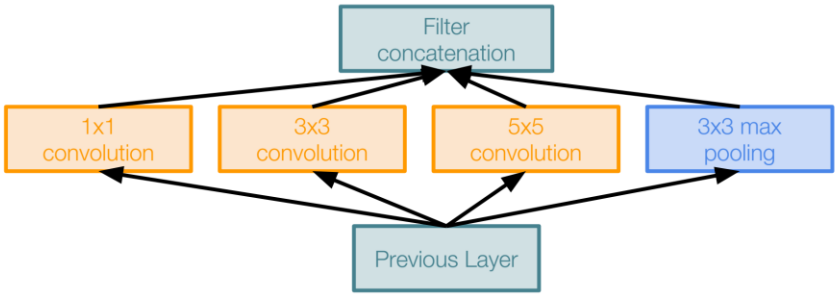
Reminder: 1x1 convolutions



Motivation, Consideration, Architectural Details

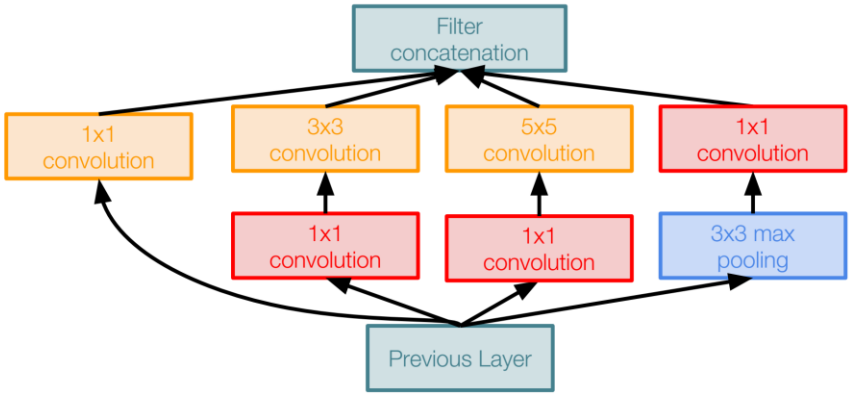
Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

1x1 conv "bottleneck" layers

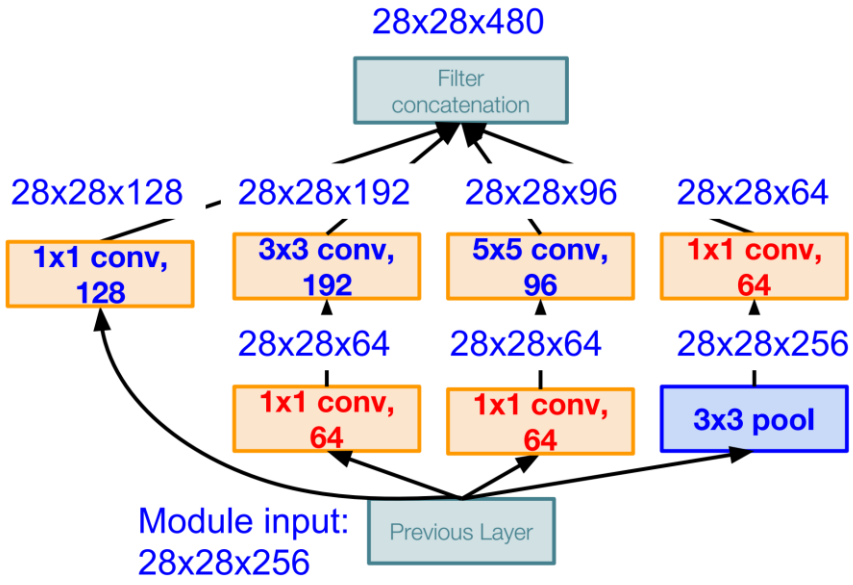


Inception module with dimension reduction

Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]



Inception module with dimension reduction

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops

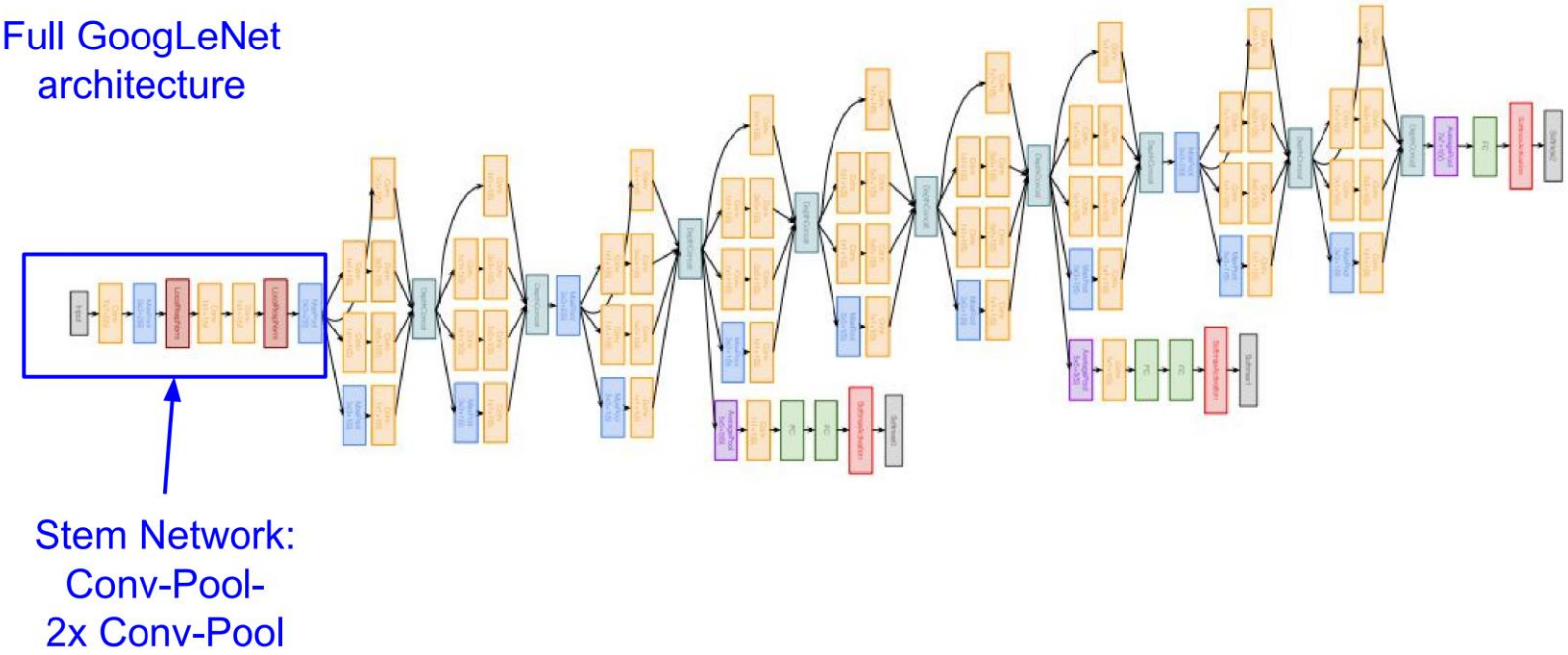
Compared to 854M ops for naive version
Bottleneck can also reduce depth after pooling layer

Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

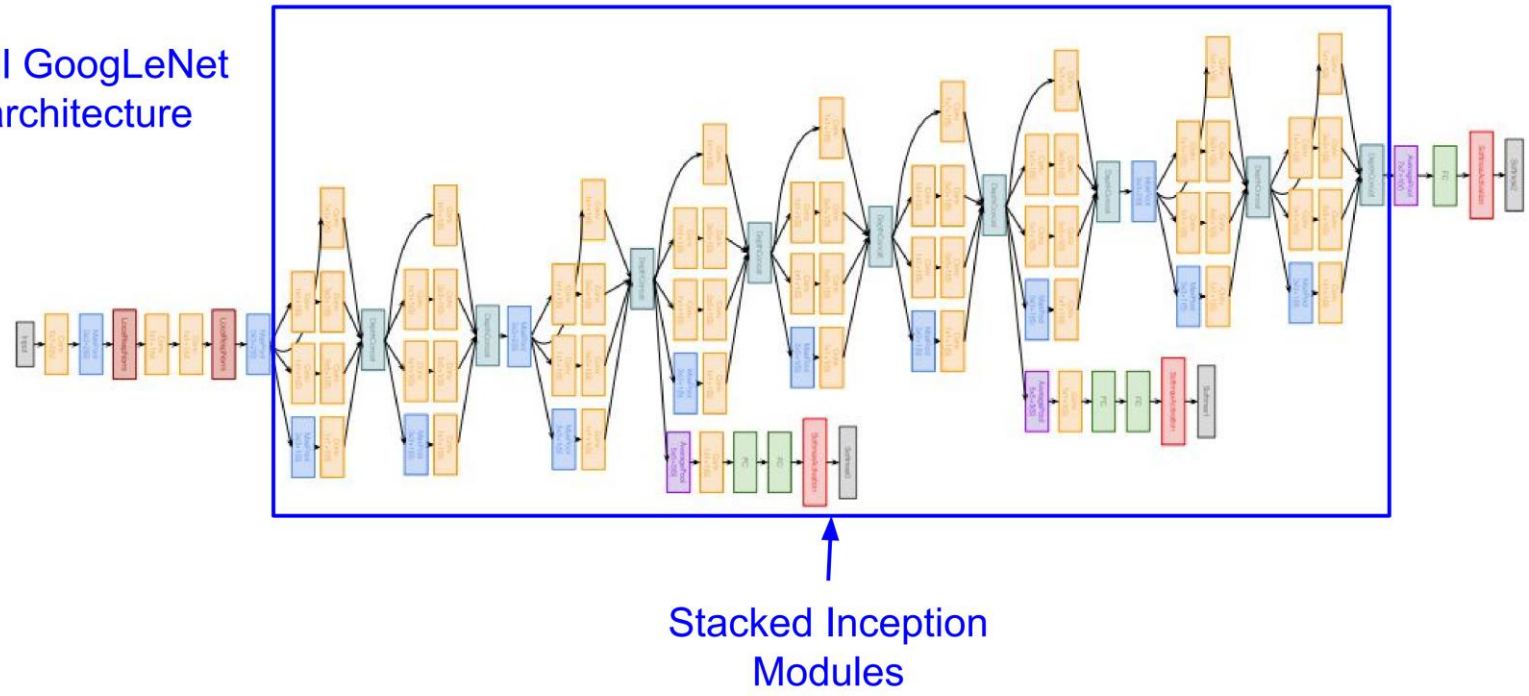


Stem Network: Conv-Pool- 2x Conv-Pool

Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]

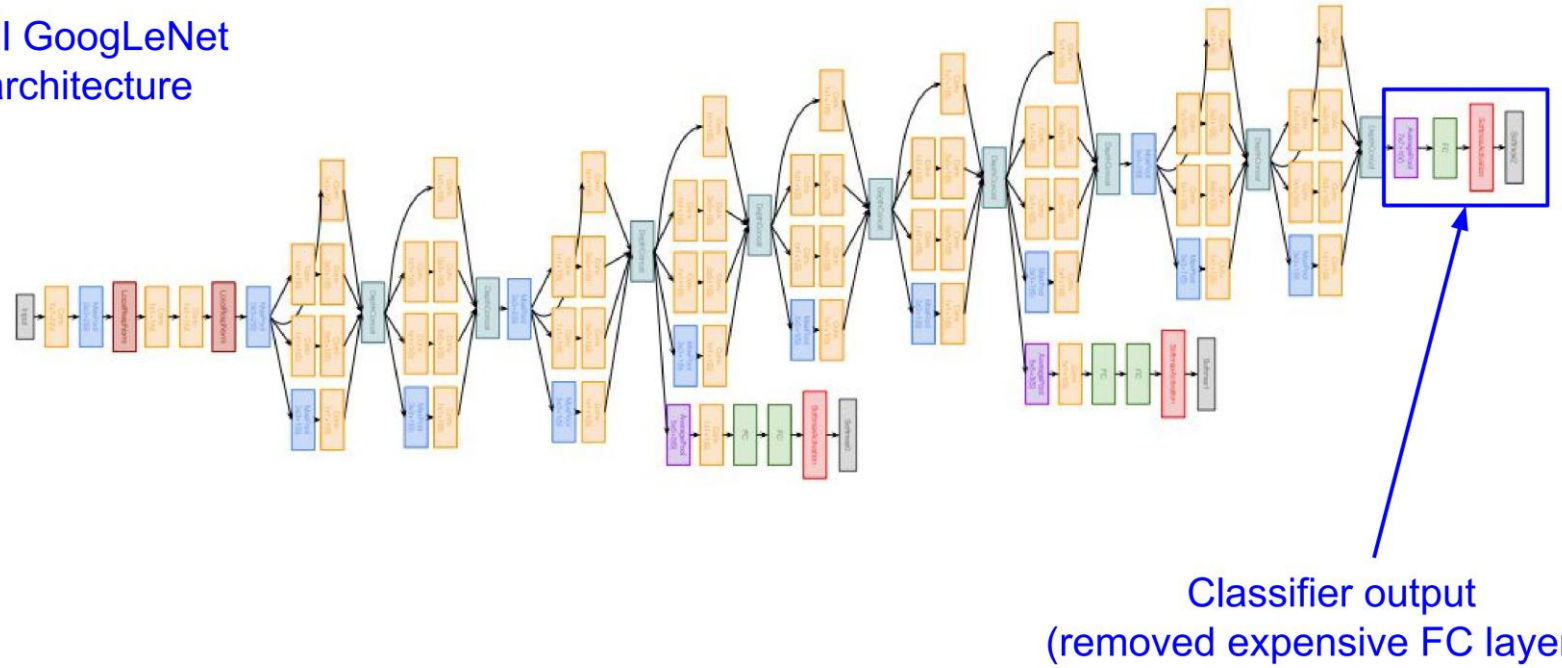


Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



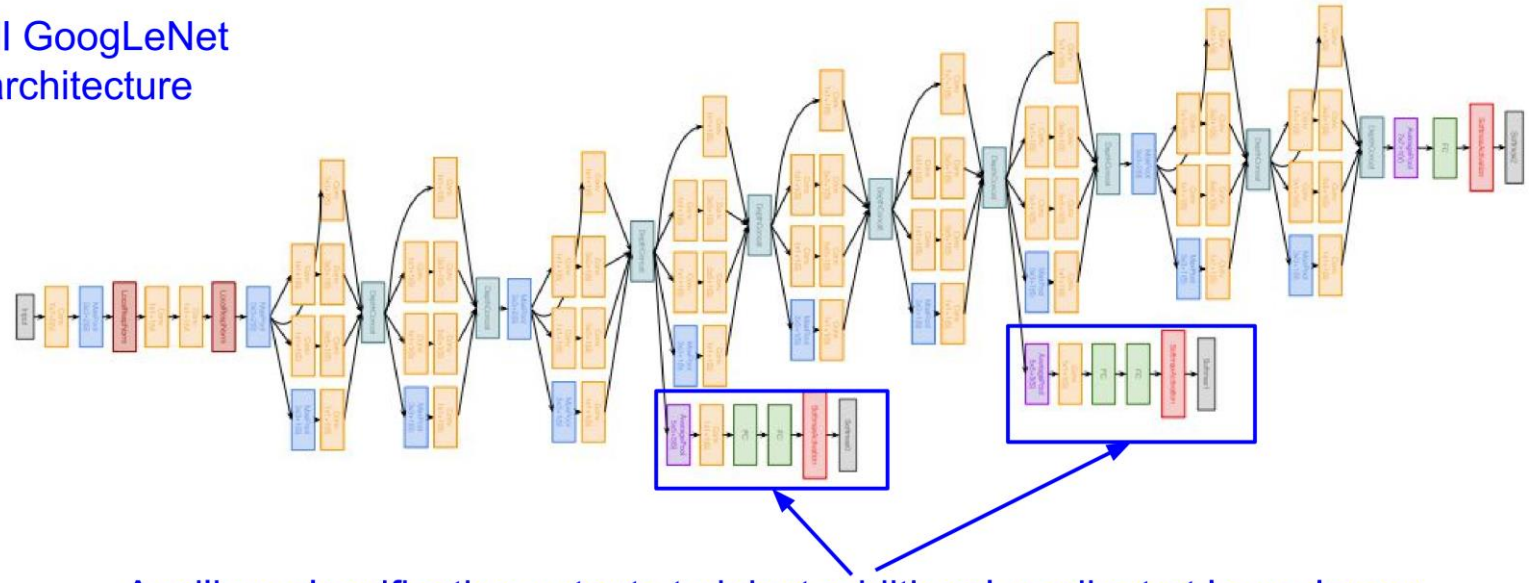
Classifier output
(removed expensive FC layers!)

Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



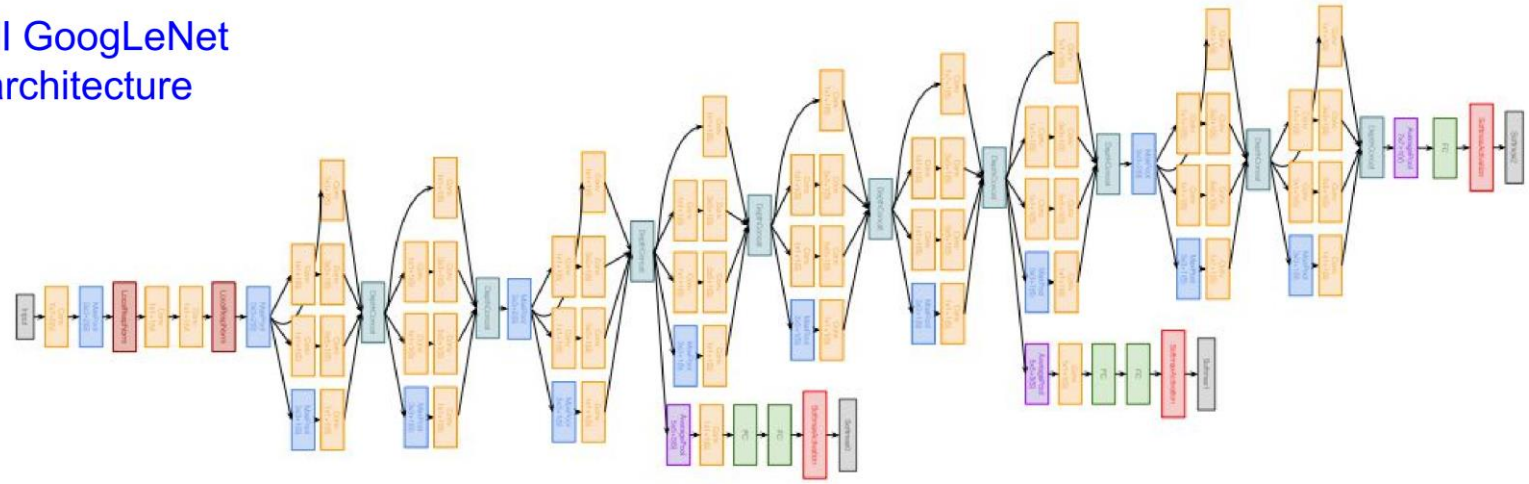
Auxiliary classification outputs to inject additional gradient at lower layers (AvgPool-1x1Conv-FC-FC-Softmax)

Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



22 total layers with weights (including each parallel layer in an Inception module)

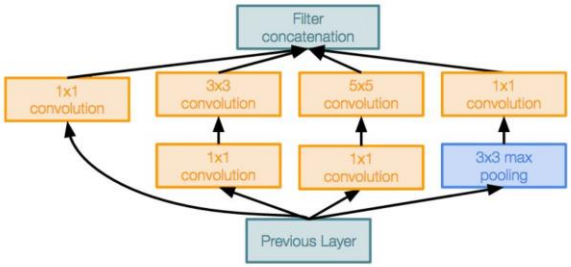
Motivation, Consideration, Architectural Details

Case Study: GoogLeNet

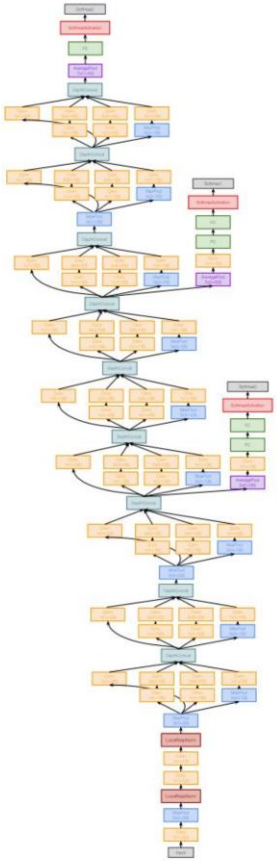
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient "Inception" module
- No FC layers
- 12x less params than AlexNet
- ILSVRC'14 classification winner (6.7% top 5 error)



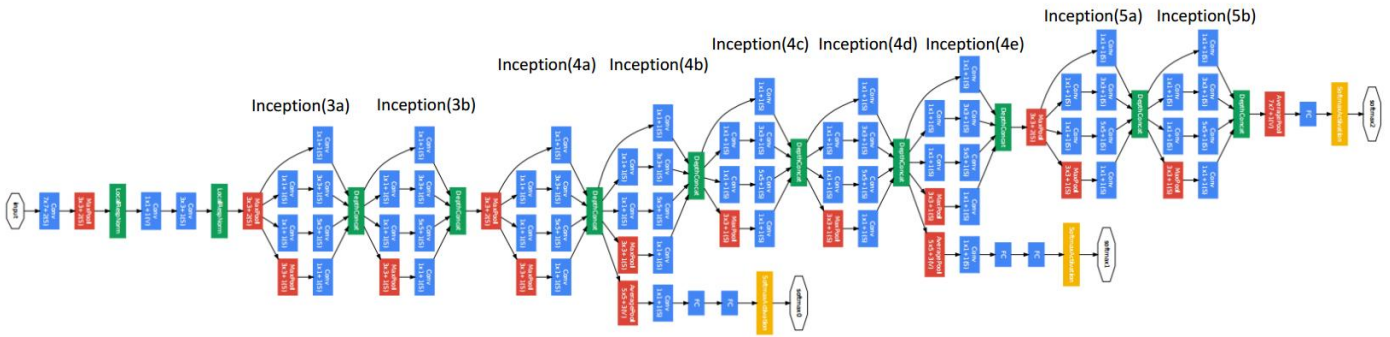
Inception module



Motivation, Consideration, Architectural Details

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture



ILSVRC 2014 Classification Challenge Setup and Results

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Table 2: Classification performance

Number of models	Number of Crops	Cost	Top-5 error	compared to base
1	1	1	10.07%	base
1	10	10	9.15%	-0.92%
1	144	144	7.89%	-2.18%
7	1	7	8.09%	-1.98%
7	10	70	7.62%	-2.45%
7	144	1008	6.67%	-3.45%

Table 3: GoogLeNet classification performance break down

Q&A