Deeper Deep Networks

presented by:
Spencer Cappallo

Overview

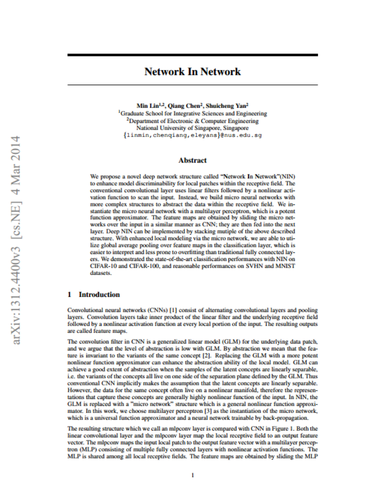
- Three recent papers discussing “deeper” deep networks
- All achieve state-of-the-art results
- High-level overview of new ideas in these networks

Network in Network

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network In Network." *arXiv preprint arXiv:1312.4400* (2013).

Basic Idea:

Uses small “micro networks” as a function approximator to replace conventional convolution operation



arXiv:1312.4400v3 [cs.NE] 4 Mar 2014

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 “Network In Network”(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.

1 Introduction

Convolutional neural networks (CNNs) [1] consist of alternating convolutional layers and pooling layers. Convolution layers take inner product of the linear filter and the underlying receptive field followed by a nonlinear activation function at every local portion of the input. The resulting outputs are called feature maps.

The convolution filter in CNN is a generalized linear model (GLM) for the underlying data patch, and we argue that the level of abstraction is low with GLM. By abstraction we mean that the feature is invariant to the variants of the same concept [2]. Replacing the GLM with a more potent nonlinear function approximator can enhance the abstraction ability of the local model. GLM can achieve a good extent of abstraction when the samples of the latent concepts are linearly separable, i.e. the variants of the concepts all live on one side of the separation plane defined by the GLM. Thus conventional CNN implicitly makes the assumption that the latent concepts are linearly separable. However, the data for the same concept often live on a nonlinear manifold, therefore the representations that capture these concepts are generally highly nonlinear function of the input. In NIN, the GLM is replaced with a "micro network" structure which is a general nonlinear function approximator. In this work, we choose multilayer perceptron [3] as the instantiation of the micro network, which is a universal function approximator and a neural network trainable by back-propagation.

The resulting structure which we call an mlpconv layer is compared with CNN in Figure 1. Both the linear convolutional layer and the mlpconv layer map the local receptive field to an output feature vector. The mlpconv maps the input local patch to the output feature vector with a multilayer perceptron (MLP) consisting of multiple fully connected layers with nonlinear activation functions. The MLP is shared among all local receptive fields. The feature maps are obtained by sliding the MLP

1

Network in Network: Intuition

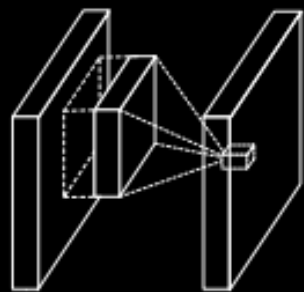
- Activations correspond to latent concepts
- Convolutional filters act as linear binary classifiers for these latent concepts in local patches
- These filters work well when the local latent concepts are linearly separable, but instead often very nonlinear
- NIN instead opts for nonlinear network structure to replace convolutional filters

Network in Network: mlpconv layers

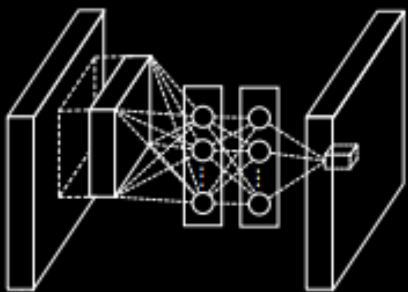
Use multilayer perceptrons as universal function approximator in place of standard convolutions

- Easily incorporated into backpropagated network
- Uses ReLUs
- Shared hidden units

Network in Network: mlpconv layers



(a) Linear convolution layer



(b) Mlpconv layer

Hidden layers are shared between output feature maps

- Cross channel information

Network in Network: Structure

Replace fully connected layers with global average pooling

- More interpretable: direct connection between categories and feature maps
    - Enforces a correspondence between last feature map and category
- Prevents overfitting

Network in Network: Structure

The network used in the paper:

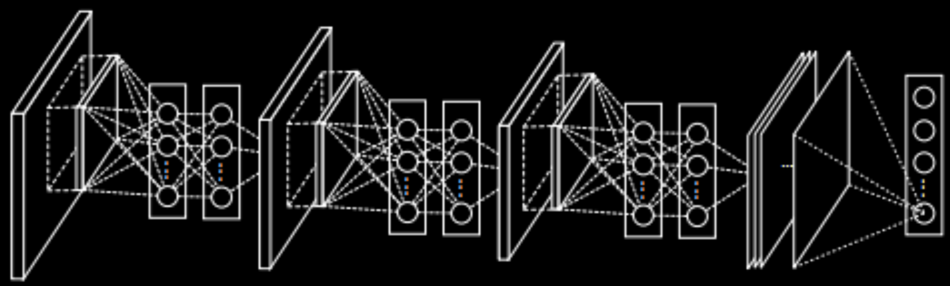


Figure 2: The overall structure of Network In Network. In this paper the NINs include the stacking of three mlpconv layers and one global average pooling layer.

- Three mlpconv layers
- Global average pooling layer
- No FC layers

Network in Network: Results

Comparison of global average pooling to fully connected layers on CIFAR-10:

Method	Testing Error
mlpconv + Fully Connected	11.59%
mlpconv + Fully Connected + Dropout	10.88%
mlpconv + Global Average Pooling	10.41%

Improvement over fully connected layers. This result will also be repeated in the next paper.

Network in Network: Results

Table 1: Test set error rates for CIFAR-10 of various methods.

Method	Test Error
Stochastic Pooling [11]	15.13%
CNN + Spearmint [14]	14.98%
Conv. maxout + Dropout [8]	11.68%
NIN + Dropout	**10.41%**
CNN + Spearmint + Data Augmentation [14]	9.50%
Conv. maxout + Dropout + Data Augmentation [8]	9.38%
DropConnect + 12 networks + Data Augmentation [15]	9.32%
NIN + Dropout + Data Augmentation	**8.81%**

Table 3: Test set error rates for SVHN of various methods.

Method	Test Error
Stochastic Pooling [11]	2.80%
Rectifier + Dropout [18]	2.78%
Rectifier + Dropout + Synthetic Translation [18]	2.68%
Conv. maxout + Dropout [8]	2.47%
NIN + Dropout	2.35%
Multi-digit Number Recognition [19]	2.16%
DropConnect [15]	**1.94%**

Table 2: Test set error rates for CIFAR-100 of various methods.

Method	Test Error
Learned Pooling [16]	43.71%
Stochastic Pooling [11]	42.51%
Conv. maxout + Dropout [8]	38.57%
Tree based priors [17]	36.85%
NIN + Dropout	**35.68%**

Table 4: Test set error rates for MNIST of various methods.

Method	Test Error
2-Layer CNN + 2-Layer NN [11]	0.53%
Stochastic Pooling [11]	0.47%
NIN + Dropout	0.47%
Conv. maxout + Dropout [8]	**0.45%**

* Note that NIN still needs dropout

Network in Network: Take-aways

- Additional non-linearity may significantly improve discriminative abilities of layers

- Replacing fully connected layers with the global average pooling seems to improve performance

- Impressive performance on datasets tested

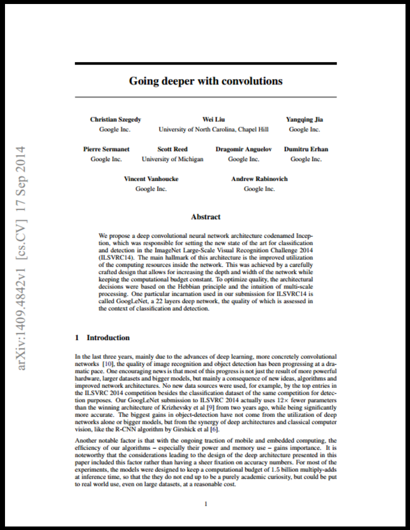
...but how well do these ideas scale up?

GoogLeNet

Szegedy, Christian, et al. "Going deeper with convolutions." *arXiv preprint arXiv:1409.4842* (2014).

Submission to ILSVRC2014 challenge

- 1st place Classification
- 1st place Object Detection with additional Training Data



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.

arXiv:1409.4842v1 [cs.CV] 17 Sep 2014

Abstract

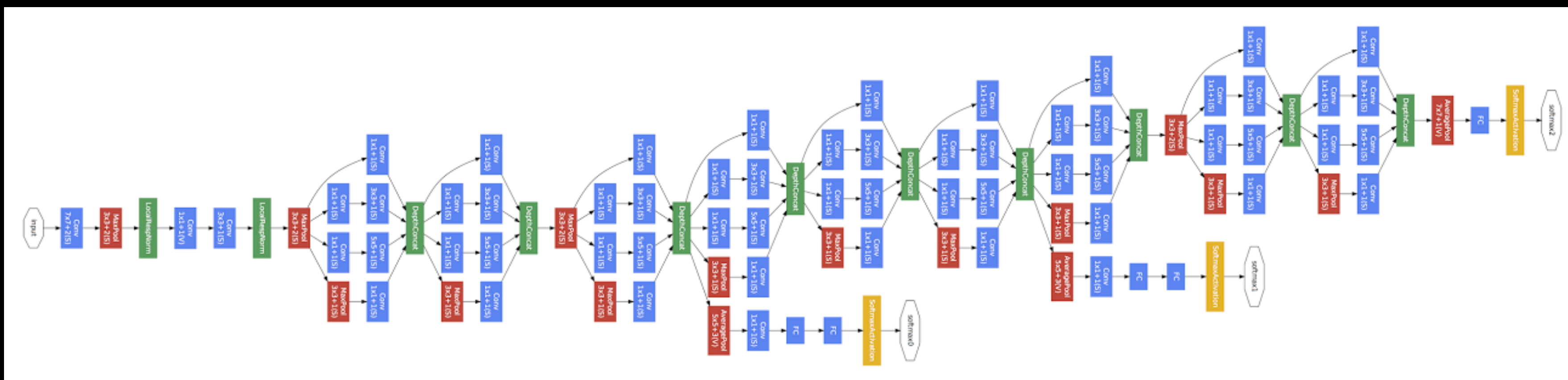
We propose a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved 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.

1 Introduction

In the last three years, mainly due to the advances of deep learning, more concretely convolutional networks [10], the quality of image recognition and object detection has been progressing at a dramatic pace. One encouraging news is that most of this progress is not just the result of more powerful hardware, larger datasets and bigger models, but mainly a consequence of new ideas, algorithms and improved network architectures. No new data sources were used, for example, by the top entries in the ILSVRC 2014 competition besides the classification dataset of the same competition for detection purposes. 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. The biggest gains in object-detection have not come from the utilization of deep networks alone or bigger models, but from the synergy of deep architectures and classical computer vision, like the R-CNN algorithm by Girshick et al [6].

Another notable factor is that with the ongoing traction of mobile and embedded computing, the efficiency of our algorithms – especially their power and memory use – gains importance. It is noteworthy that the considerations leading to the design of the deep architecture presented in this paper included this factor rather than having a sheer fixation on accuracy numbers. For most of the experiments, the models were designed to keep a computational budget of 1.5 billion multiply-adds at inference time, so that the they do not end up to be a purely academic curiosity, but could be put to real world use, even on large datasets, at a reasonable cost.

GoogLeNet



Convolution/FC Max Pooling Softmax Concatenation

Looks like a big, ugly mess.

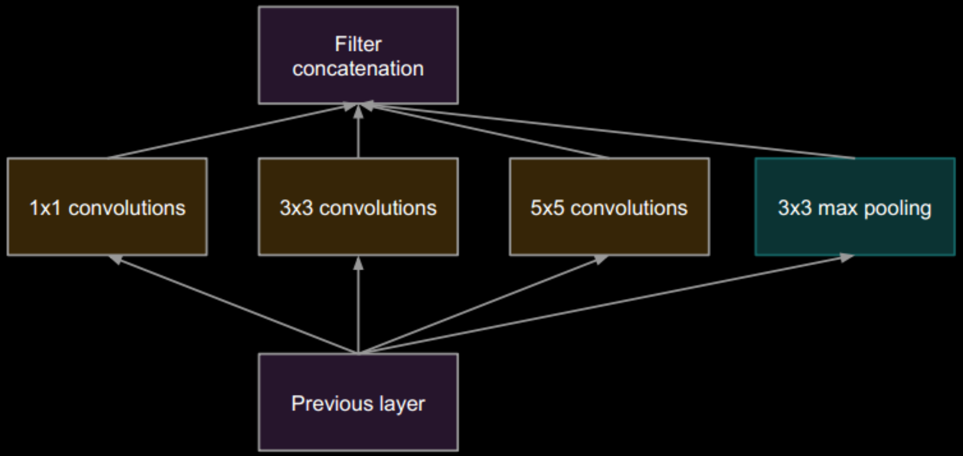
Fortunately, if we break it down a bit it's not *too* bad.

GoogLeNet: Inception Module

Idea 1:

Use 1x1, 3x3, and 5x5 convolutions in parallel to capture a variety of structures

Also add a parallel max pooling path

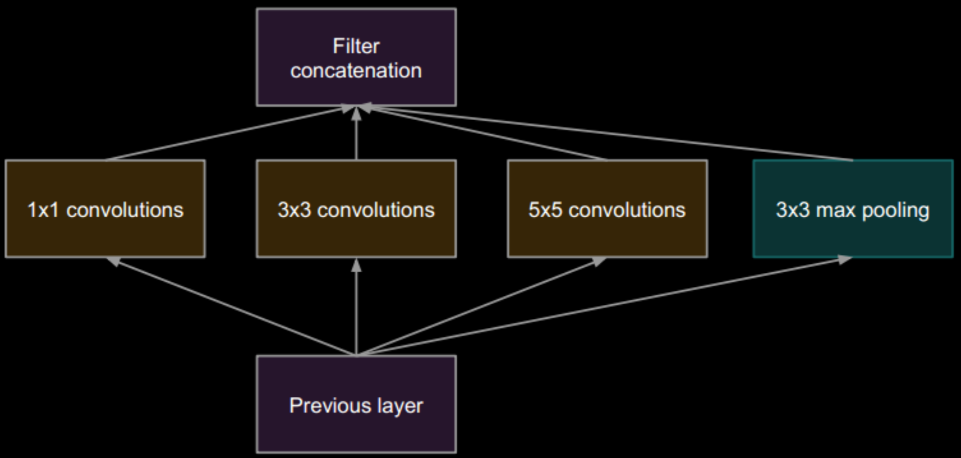


GoogLeNet: Inception Module

Idea 1:

Use 1x1, 3x3, and 5x5 convolutions in parallel to capture a variety of structures

Also add a parallel max pooling path

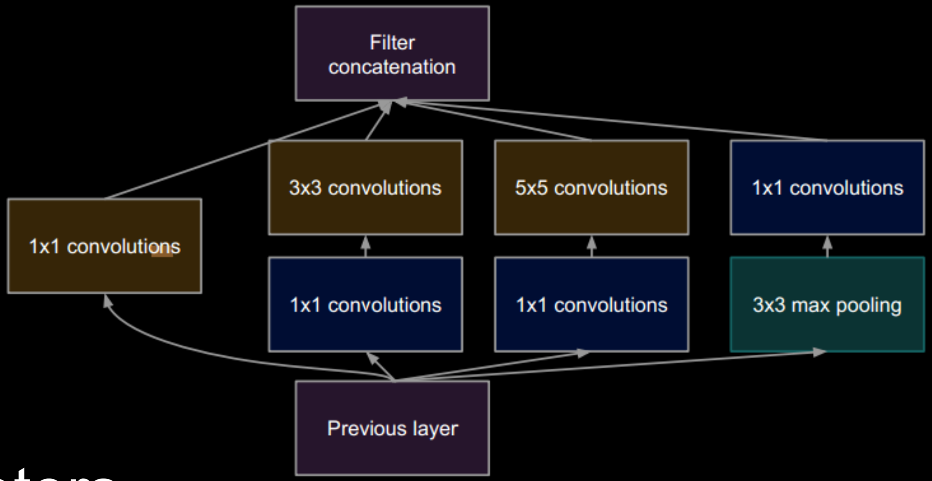


The problem: Computational Expense quickly balloons

GoogLeNet: Inception Module

Idea 2:

Use 1x1 convolutional layers for dimensional reduction.



- Limits computational blow up from increasing parameters

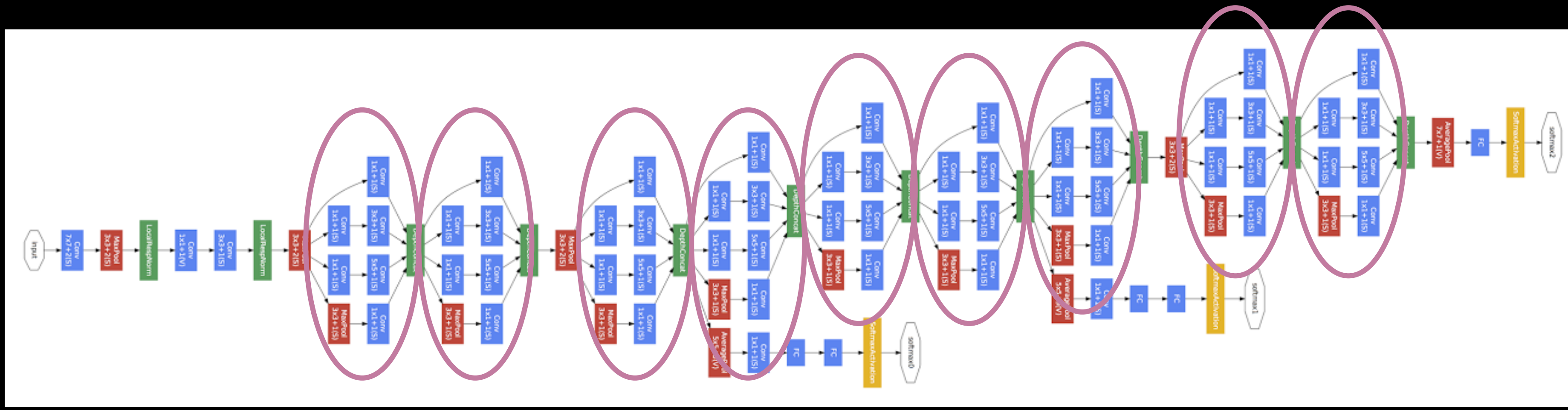
- The 1x1 convolutions also use ReLUs, so provide an added element of non-linearity

GoogLeNet

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								

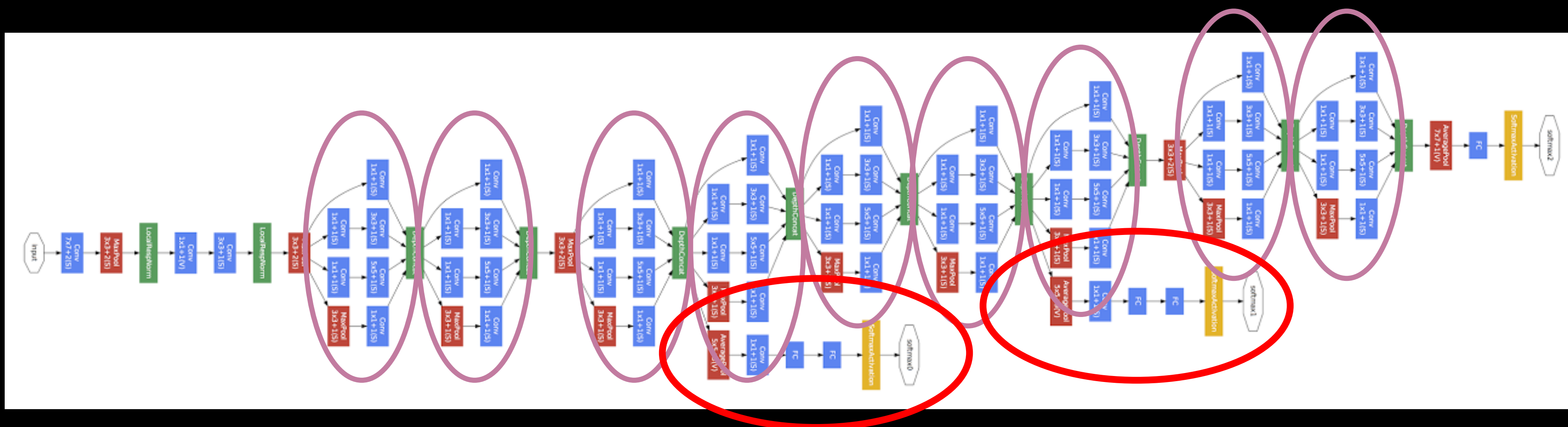
Note, they also replaced FC with avg pool

GoogLeNet



Not so scary after all, just 9 of these “inception modules” stacked on top of each other

GoogLeNet



Not so scary after all, just 9 of these "inception modules" stacked on top of each other

...but wait, what are *these things*?

GoogLeNet: Auxiliary Classifiers

Problem:

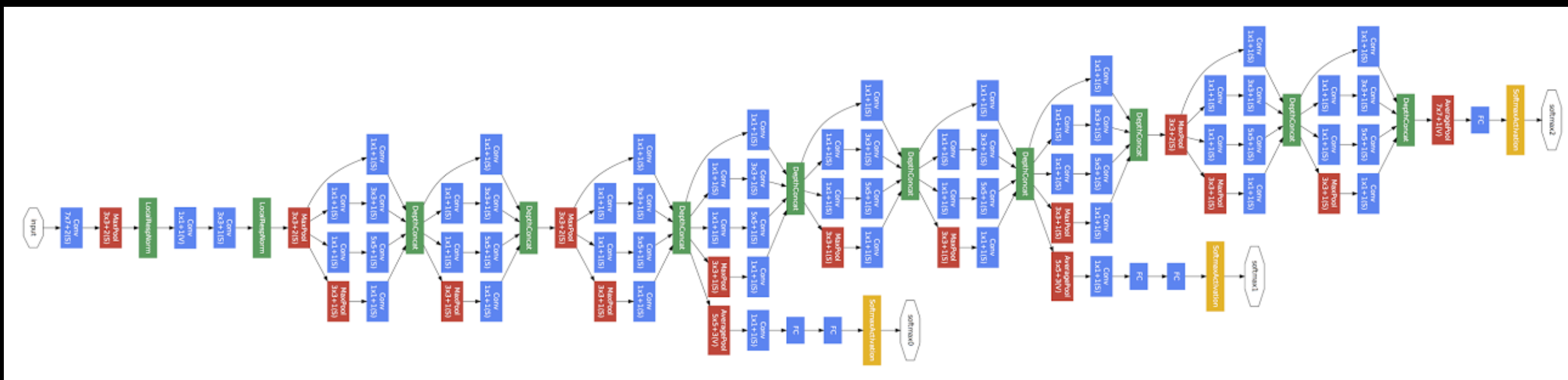
The depth of the network raises concerns about the effectiveness of the backpropagating gradient

Their solution?

Throw on auxiliary classifiers part way through

- Small convnets with a pooling layer, a 1x1 convolution layer, fully connected layers, and softmax loss layer on 1000 classes
- Combined with backprop’d loss with relative weight of 0.3
- (Removed at test time)

GoogLeNet



Now the structure should be less intimidating

GoogLeNet: Stats

12x fewer parameters than AlexNet

- The move away from fully connected layers near the top of the network helps with this

22 Layers deep

~2x more operations than AlexNet

GoogLeNet: 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

GoogLeNet: Take-aways

- Once more we see the replacement of a fully-connected layer with global average pooling

- 1x1 Convolutional filters similar to multilayer perceptrons in Network in Network paper

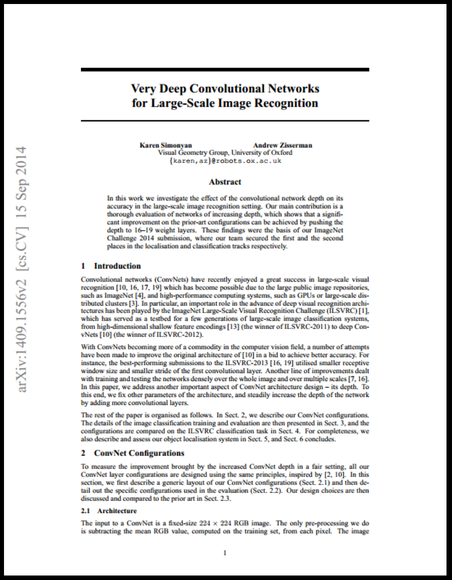
- Concatenation of different size convolutional filters

- Mid-network classification to improve backpropagation signal and increase mid-network discriminant abilities

Very Deep ConvNets

Simonyan, Karen, and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." *arXiv preprint arXiv:1409.1556* (2014).

How much effect does extra depth add?



arXiv:1409.1556v2 [cs.CV] 15 Sep 2014

Very Deep Convolutional Networks for Large-Scale Image Recognition

Karen Simonyan   Andrew Zisserman
Visual Geometry Group, University of Oxford
{karen,az}@robots.ox.ac.uk

Abstract

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively.

1 Introduction

Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale visual recognition [10, 16, 17, 19] which has become possible due to the large public image repositories, such as ImageNet [4], and high-performance computing systems, such as GPUs or large-scale distributed clusters [3]. In particular, an important role in the advance of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [1], which has served as a testbed for a few generations of large-scale image classification systems, from high-dimensional shallow feature encodings [13] (the winner of ILSVRC-2011) to deep ConvNets [10] (the winner of ILSVRC-2012).

With ConvNets becoming more of a commodity in the computer vision field, a number of attempts have been made to improve the original architecture of [10] in a bid to achieve better accuracy. For instance, the best-performing submissions to the ILSVRC-2013 [16, 19] utilised smaller receptive window size and smaller stride of the first convolutional layer. Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales [7, 16]. In this paper, we address another important aspect of ConvNet architecture design – its depth. To this end, we fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers.

The rest of the paper is organised as follows. In Sect. 2, we describe our ConvNet configurations. The details of the image classification training and evaluation are then presented in Sect. 3, and the configurations are compared on the ILSVRC classification task in Sect. 4. For completeness, we also describe and assess our object localisation system in Sect. 5, and Sect. 6 concludes.

2 ConvNet Configurations

To measure the improvement brought by the increased ConvNet depth in a fair setting, all our ConvNet layer configurations are designed using the same principles, inspired by [2, 10]. In this section, we first describe a generic layout of our ConvNet configurations (Sect. 2.1) and then detail out the specific configurations used in the evaluation (Sect. 2.2). Our design choices are then discussed and compared to the prior art in Sect. 2.3.

2.1 Architecture

The input to a ConvNet is a fixed-size 224×224 RGB image. The only pre-processing we do is subtracting the mean RGB value, computed on the training set, from each pixel. The image

1

Very Deep ConvNets

Basic idea:

- Stack a bunch of convolutional layers on top of each other, with occasional max-pooling

- All convolutional layers either 3x3 or 1x1

  - Stacks of 3x3 layers have equivalent receptive fields to larger convolutional filters

  - 1x1 convolutions being used here, again, to introduce extra non-linearity
    (input/output channels' dimensions are equal here)

Very Deep ConvNets

Why stacks of 3x3 Convolutions?

- Added discriminative ability from more ReLU layers
- Effective receptive field equivalent to larger convolutions
- Fewer parameters

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Very Deep ConvNets: Results

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

→ Deeper is better

Very Deep ConvNets: Take-aways

- Again we see deeper nets pushing the state of the art
- Once more, greater non-linearity improving network ability
  - Both through 1x1 convolutions and stacks of 3x3