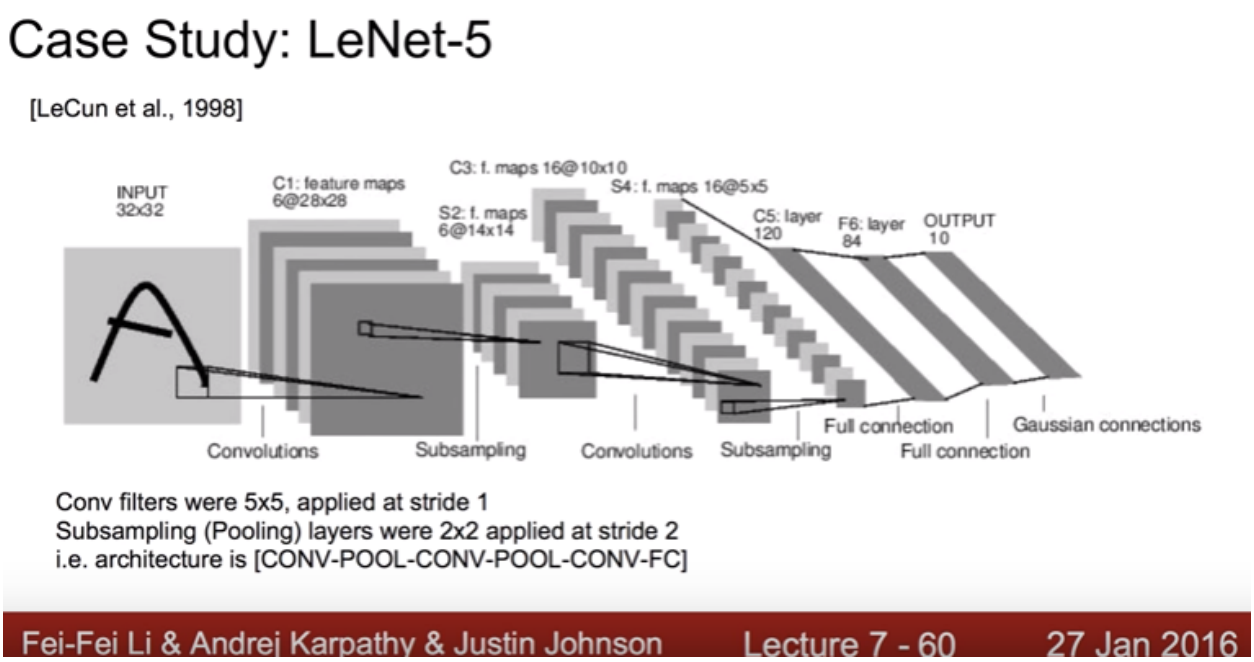
Paper Review on Going Deeper with Convolutions

ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14)에서 1등을 한 **GoogLeNet**을 소개하는 논문이다. GoogLeNet이라는 이름은 논문의 저자 대부분이 Google 직원이라는 점과 LeNet모델에서 따 지어졌다. LeNet은 CNN개념을 처음 개발한 Yann LeCun 이 개발한 구조로 convolution과 pooling layer를 반복한 다음, 마지막에 fully-connected network로 연결해 classification을 수행하는 구조이다. LeCun et al.에서 제시한 Lenet5의 구조는 아래와 같다.



Source: http://cs231n.stanford.edu/slides/2021/lecture_5.pdf

GoogLeNet은 **1x1 convolution layer의 사용**을 통해 그전 layer에서 input된 데이터의 사이즈를 줄이고, **depth를 늘려** 모델의 성능을 개선한다는 점에서 VGGNet과 유사한 점이 있다. 하지만 구조는 VGG와는 다른 독특한 구조인데, 논문에서는 이를 I**nception**이라고 부르며, 2014

년 challenge 당시 image classification과 object detection에서 상당히 뛰어난 성능을 보여주었다. 어째서 이 아키텍처는 Inception이라는 이름을 가지게 된 것일까?

1. Abstract

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.

Abstract에서는 CNN 아키텍처인 Inception의 특징을 간단히 소개한다.

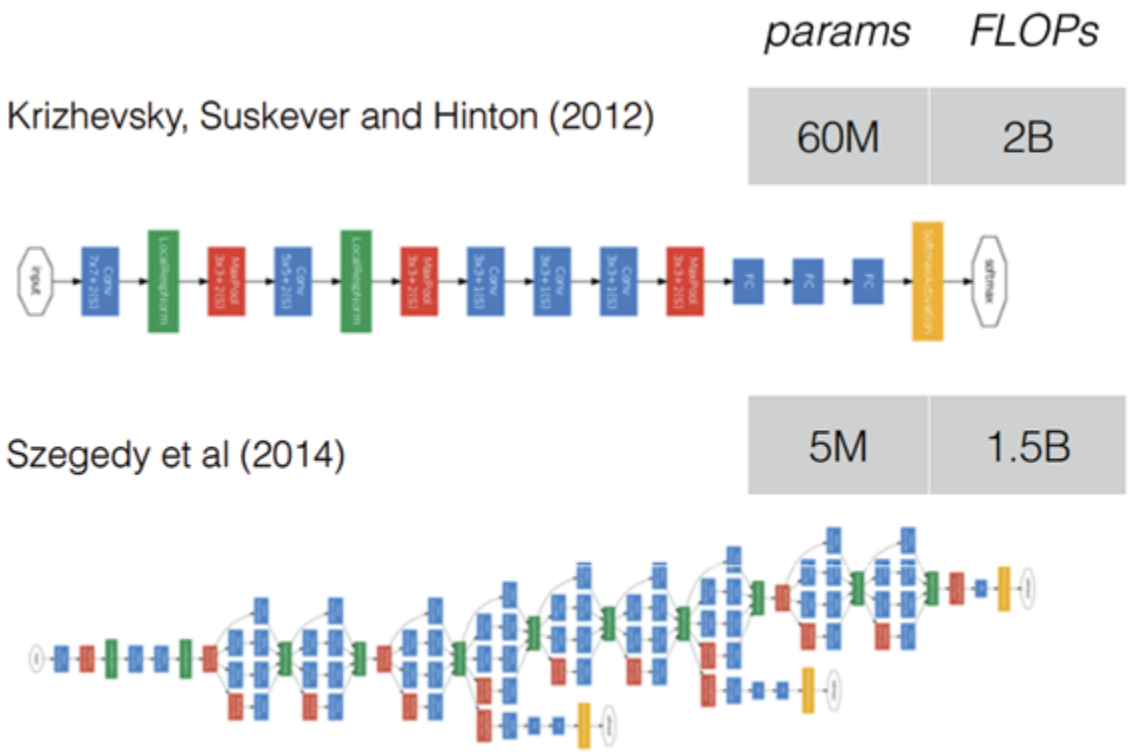
- 이 아키텍처의 주요 특징은 **연산을 하는데 드는 자원의 효율이 개선**된 것이다. 이는 네트워크의 **depth와 width를 늘려도 연산하는 양이 늘어나지 않고 유지**되게 해주는 정교한 설계로 인해 가능해졌다.
- Inception 아키텍처를 ILSVRC14에 제출하기 위해 구현한 하나의 버전이 GoogLeNet이며, 22개의 layer를 가진다.

2. 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].

Introduction에서는 딥러닝의 발전 동향, 그 속에서 나온 GoogLeNet의 설계 주안점, Inception 이름의 의미에 대해 다룬다.

- 지난 3년간(approx. 2012-2014) convolutional network의 발전으로 인해 image recognition과 object detection의 성능이 상당히 좋아졌다. 이는 단순히 하드웨어 성능의 발전, 더 큰 데이터셋, 더 커진 모델에서만 기인한 것이 아니라 **새로운 아이디어, 알고리즘, 그리고 개선된 신경망 구조**에서 온 것이다.
- 그 예시로, GoogLeNet(2014)은 ILSVRC12에서 1등을 한 Krizhevsky et al.(2012)이 제안한 AlexNet보다 12배나 더 적은 수의 parameter를 사용하지만, 성능에 있어서는 훨씬 더 정확했다고 한다. 이러한 개선은 깊은 신경망 구조와 고전 컴퓨터 비전의 시너지 덕분이다 라고 논문은 주장한다.



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.

- 또한 모바일과 임베디드 컴퓨팅이 대두가 되면서 **전력과 메모리 사용의 관점에서 효율적인 알고리즘**이 중요시되고 있다. 그런 측면에서 GoogLeNet이 deep architecture를 구현할 때, accuracy 수치에만 중점을 둔 고정된 모델이 아니라 조금 더 유연하게 전력과 메모리 사용에 있어서도 효율적인 알고리즘이 될 수 있도록 설계한 것이 의미를 가진다.
- 추론시간 동안 1.5 billion이하의 연산만 수행하도록 설계하여 결과가 연구실에만 머무는 것이 아니라 현실에서도 적절한 가격으로 사용될 수 있게 만들어졌다.

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 by Lin et al [12] in conjunction with the famous "we need to go deeper" internet meme [1]. In our case, the word "deep" is used in two different meanings: first of all, in the sense that we introduce a new level of organization in the form of the "Inception module" and also in the more direct sense of increased network depth. In general, one can view the Inception model as a logical culmination of [12] while taking inspiration and guidance from the theoretical work by Arora et al [2]. The benefits of the architecture are experimentally verified on the ILSVRC 2014 classification and detection challenges, on which it significantly outperforms the current state of the art.

- Inception 이름의 의미는 Network in Network이라는 논문과 인셉션의 영화 대사로 만든 인터넷 밈 "we need to go deeper"에서 고안하였다.
- 여기에서 "deep"은 두 가지 의미를 가진다.
  1. "Inception module"이라는 새로운 차원의 구조
  2. 네트워크의 depth과 증가하였다는 직접적인 의미

3. Related Work

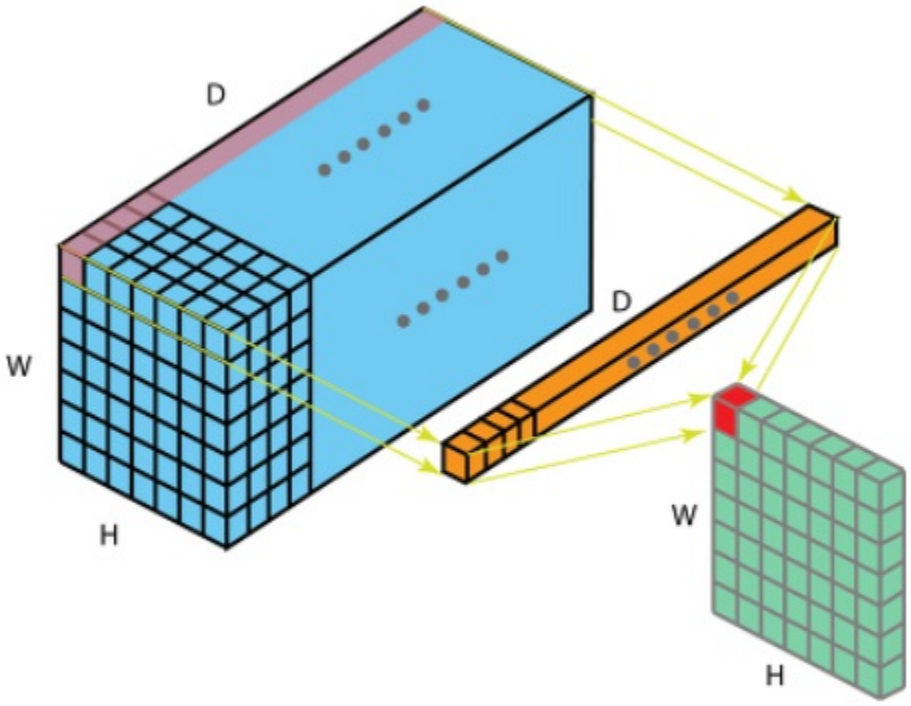
1. LeNet-5

- LeNet-5을 시작으로 CNN은 표준 구조를 가지게 된다. Convolutional layer(옵션으로 contrast normalization이나 max pooling이 따라오기도 함)들이 쌓이고, 그 뒤에 1개 이상의 fully-connected layer가 오는 구조이다.
- 이 basic design의 변형된 것들을 다룬 논문들이 많으며 이 구조는 CIFAR, MNIST, ImageNet challenge등의 classification task에서 성과를 거두었다. ImageNet처럼 큰 데이터셋을 다룰 때의 최근 트렌드는 이 표준 구조를 유지한 채로 layer의 수와 사이즈를 늘리면서 dropout으로 overfitting을 해결하는 것이 되었다.

2. Network-in-Network

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]. This enables it to be easily integrated in the current CNN pipelines. We use this approach heavily in our architecture. 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.

- Network-in-Network는 신경망의 표현력을 높이기 위해 Lin et al.에서 고안한 접근법으로, **기존의 CNN구조에 적용한다면 1x1 convolutional layer(와 ReLU activation)를 추가**하는 방식으로 구현된다.



- GoogLeNet에서는 1x1 convolutional layer를 병목 현상(computational bottleneck)을 제거하기 위한 **차원 축소**(dimensionality reduction) 모듈로 쓴다. 이를 통해 연산 자원의 큰 소모 없이도 네트워크 depth와 width를 늘릴 수 있게 해준다.

3. Motivation and High Level Considerations

- Deep neural network의 성능을 개선할 수 있는 가장 좋은 방법은 신경망의 크기를 늘리는 것으로 이는 depth(layer의 개수를 늘리는 것)와 width(1개의 layer의 unit을 늘리는 것)를 모두 포함한다.
- 많은 양의 labeled training data가 주어졌을 때 가장 안전하고 쉽게 모델의 성능을 개선할 수 있는 방법이다.
- 크기를 늘리는 것의 단점

  1. 신경망의 크기가 커지면 **parameter의 수도 증가**하며, training set이 무한하게 존재하는 것이 아니라 한정적일 때, **overfitting**하기 쉬운 모델이 되어버린다.

     ImageNet처럼 세밀한 분류를 인간 전문가가 해야 할 경우, 고품질의 training data를 만드는 것은 힘들고 비용이 많이 들기 때문에 병목 현상이 될 수 있다.

(a) Siberian husky (b) Eskimo dog

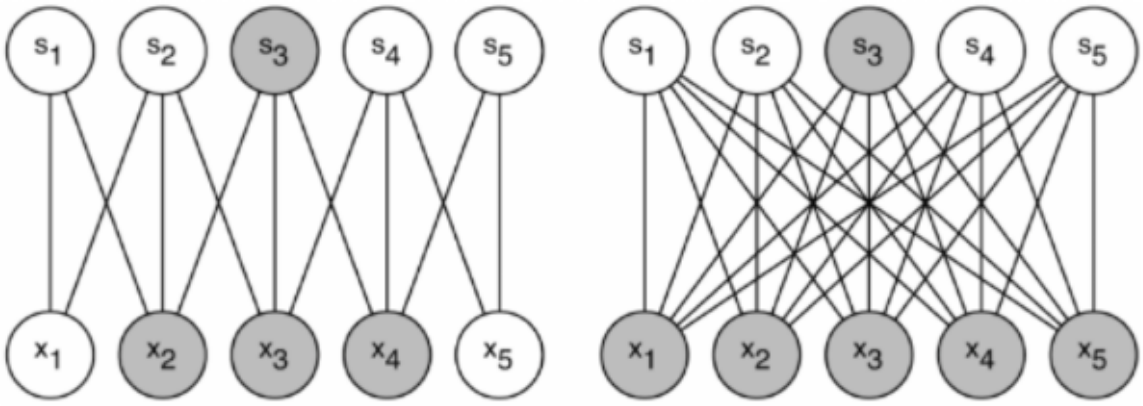
Figure 1: Two distinct classes from the 1000 classes of the ILSVRC 2014 classification challenge.

  2. 신경망의 크기가 커지면 **컴퓨터 자원의 소모량 또한 늘어난다**. 그 예시로 만약 두 개의 conv layer 서로 연결되어 있다면, 필터의 수가 늘어날 때 연산량은 quadratic하게 늘어날 것이다. 컴퓨터 자원은 언제나 한정적이므로 자원을 효율적으로 분배하는 것이 신경망 크기를 무한정 늘리는 것보다 중요하다.

The fundamental way of solving both issues would be by ultimately moving from fully connected to sparsely connected architectures, even inside the convolutions. Besides mimicking biological systems, this would also have the advantage of firmer theoretical underpinnings due to the ground-breaking work of Arora et al. [2]. Their main result states that if the probability distribution of the data-set is representable by a large, very sparse deep neural network, then the optimal network topology can be constructed layer by layer by analyzing the correlation statistics of the activations of the last layer and clustering neurons with highly correlated outputs. Although the strict mathematical proof requires very strong conditions, the fact that this statement resonates with the well known Hebbian principle – neurons that fire together, wire together – suggests that the underlying idea is applicable even under less strict conditions, in practice.

- 위의 두 가지 문제를 동시에 해결하는 방법은 **dense한 fully-connected 구조에서 sparsely connected 구조로 바꾸는 것**이다.
- 만약 dataset의 확률 분포(probability distribution)를 sparse하면서도 큰 심층 신경망으로 표현 가능하다면, (각 layer마다) 그 전 layer의 activation output의 상관 관계를 분석하여 서로 연관성이 높은 뉴런들만 다음 layer에 output함으로써, 최적의 네트워크를 만들 수 있을 것이다.

  (This statement from the paper—*Provable bounds for learning some deep representations(2013)*—would be a theoretical underpinning to the sparsely connected network.)



sparse vs dense

- 하지만, **오늘날의 컴퓨팅 환경은 균일하지 않은 sparse한 데이터 구조를 다룰 때 매우 비효율적**이다. 이러한 격차는 더욱 커졌는데, dense data는 CPU와 GPU의 세세한 하드웨어적 디테일을 활용하여 꾸준히 개선된 라이브러리로 엄청 빠른 연산 dense matrix multiplication이 가능하게 된 반면, Sparse data의 연산은 발전이 미미했다.

- ConvNets는 처음에는 symmetry를 깨고 학습을 발전시키기 위해 sparse structure를 사용해 왔으나, 병렬 컴퓨팅에 더 최적화하기 위해 fully-connected로 구조가 바뀌었다. 더욱이 구조의 균일성, 많은 수의 filter들, 더 큰 batch size가 dense computation을 효율적으로 만들어버렸다.

"This raises the question whether there is any hope for a next, intermediate step: an architecture that makes use of the extra sparsity, even at filter level, as suggested by the theory, but exploits our current hardware by utilizing computations on dense matrices…"

- 이제 문제는 논문에서 제시한 바와 같이 sparsity를 모델 구조에 포함하면서도, 연산 자체는 하드웨어 자원을 최대한 활용할 수 있는 dense matrix 연산을 하는 구조를 만들 수 있는가이다.

"The vast literature on sparse matrix computation suggests that clustering sparse matrices into relatively dense submatrices trends to give state of the art practical performance for sparse matrix multiplication."

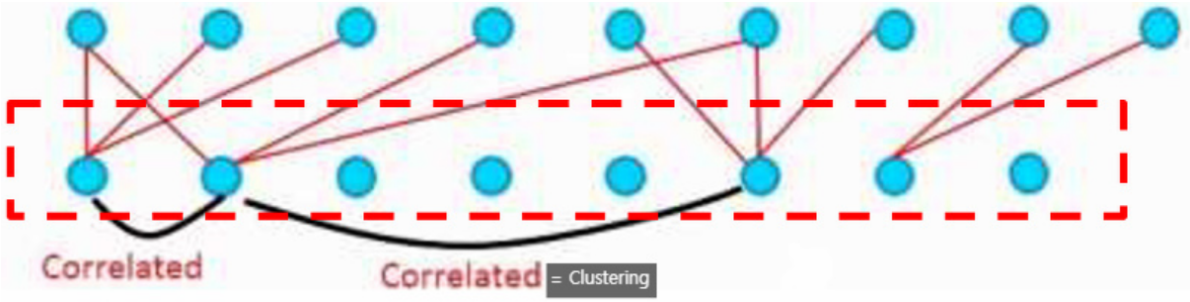
- sparse matrix 연산을 다룬 많은 논문들에서는, sparse matrix multiplication에서 높은 성능을 보이기 위해서 연관성이 높은 sparse matrices를 클러스터링 하여 상대적으로 dense한 submatrices를 만드는 것을 제안하였다.
- Google팀의 Inception 구조는 위에서 말한 유사 Sparse 구조를 시험하기 위한 case study에서 시작했다. 그리고 learning rate와 hyperparameter를 조정하고 훈련 방법 등을 개선한 결과, Localization 및 Object detection 분야에서 특히 좋은 성능을 보였다고 한다.

4. Architectural Details

The main idea of the Inception architecture is based on finding out how an optimal local sparse structure in a convolutional vision network can be approximated and covered by readily available dense components. Note that assuming translation invariance means that our network will be built from convolutional building blocks. All we need is to find the optimal local construction and to repeat it spatially. Arora et al. [2] suggests a layer-by layer construction in which one should analyze the correlation statistics of the last layer and cluster them into groups of units with high correlation. These clusters form the units of the next layer and are connected to the units in the previous layer. We assume that each unit from the earlier layer corresponds to some region of the input image and these units are grouped into filter banks. In the lower layers (the ones close to the input) correlated units would concentrate in local regions. This means, we would end up with a lot of clusters concentrated in a single region and they can be covered by a layer of 1×1 convolutions in the next layer, as suggested in [12]. However, one can also expect that there will be a smaller number of more spatially spread out clusters that can be covered by convolutions over larger patches, and there will be a decreasing number of patches over larger and larger regions. In order to avoid patch-alignment issues, current incarnations of the Inception architecture are restricted to filter sizes 1×1, 3×3 and 5×5, however this decision was based more on convenience rather than necessity. It also means that the suggested architecture is a combination of all those layers with their output filter banks concatenated into a single output vector forming the input of the next stage. Additionally, since pooling operations have been essential for the success in current state of the art convolutional networks, it suggests that adding an alternative parallel pooling path in each such stage should have additional beneficial effect, too (see Figure 2(a)).

- Inception 아키텍처의 주요 아이디어는 CNN에서 최적의 local sparse structure를 근사하고, 이를 dense하게 만드는 법을 찾는 것에 있다.
- 답은 최적의 로컬 구성요소를 찾아 그것을 공간적으로 반복하는 것으로, 직전 layer의 상관관계를 분석하여 상관관계가 높은 것들을 그룹으로 묶어서 다음 layer를 만들며, 그것을 layer-by-layer로 진행하는 것이다.

  즉, s**parse matrix를 서로 묶어 (클러스터링 하여) 상대적으로 dense 한 submatrix를 만드는 것이다.**

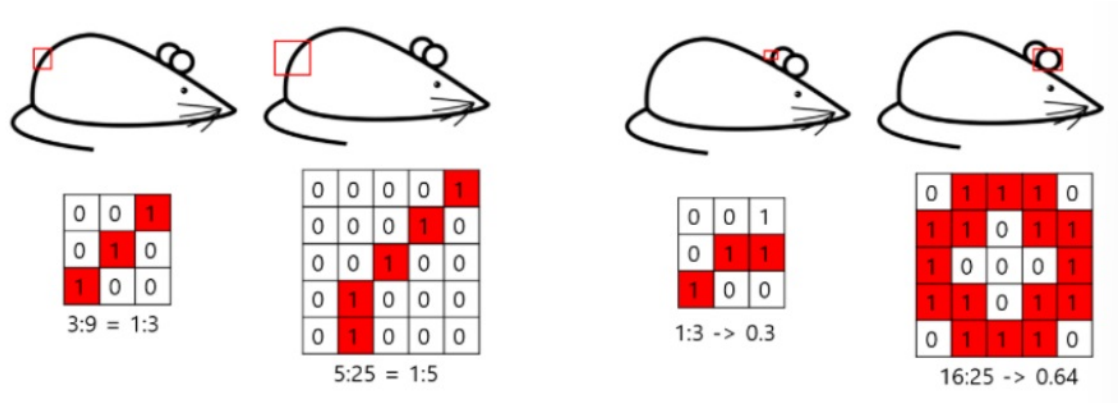


- Inception은 convolution에서 1x1, 3x3, 5x5 사이즈의 필터를 가진다.

  다양한 사이즈의 필터가 필요한 이유:

  우선, 직전 layer의 모든 유닛은 input image의 특정 부분에 해당한다고 가정한다.

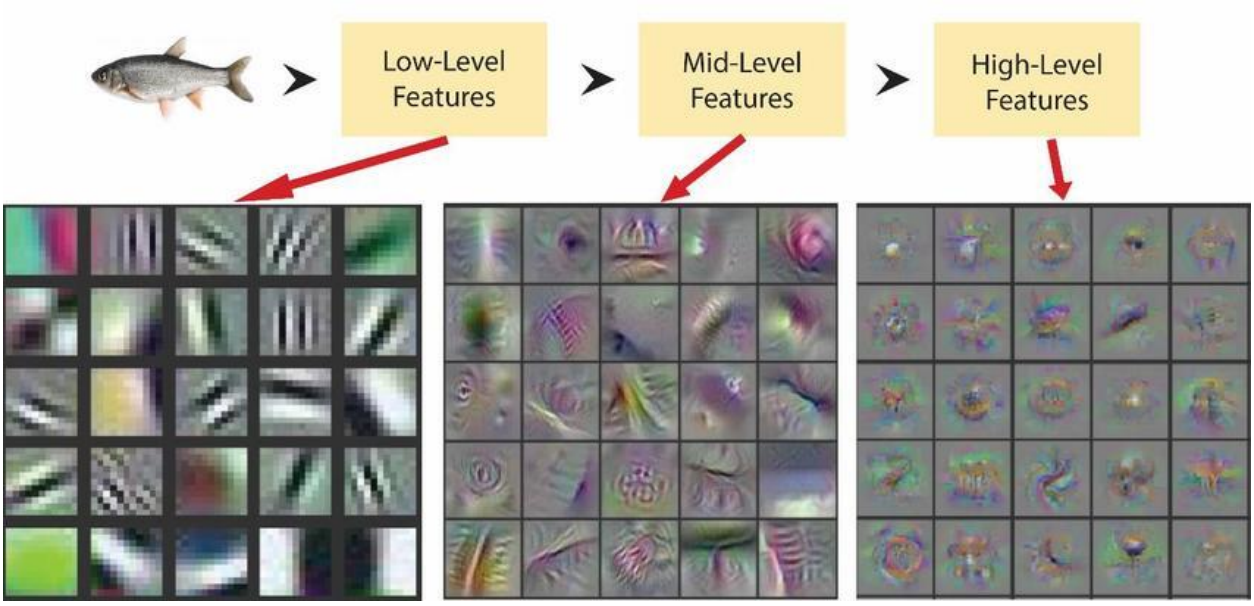
1. 그렇다면 입력에 가까운 낮은 layer에서는 correlated unit들이 특정 부분에 집중되어 있을 것이다. 이는 단일 지역에 대부분의 클러스터들이 모여있다는 뜻이므로, 이어지는 1x1convolution layer로 충분히 처리할 수 있다.
2. 하지만 몇몇 위치에서는 클러스터들이 공간적으로 더 넓게 분포할 수 있고, 이 경우 조금 더 넓은 영역의 conv filter가 있어야 correlated unit의 비율을 높일 수 있다. 따라서 3x3, 5x5 filter를 써서 다양한 scale의 연관된 유닛을 추출한다.



3. 이 논문에서 제시한 Inception Module은 **1 x 1, 3 x 3, 5 x 5 convolution 연산을 병렬적으로 수행하여 그 결과를 concatenate하여 다음 layer의 input을 산출한다.**

As these "Inception modules" are stacked on top of each other, their output correlation statistics are bound to vary: as features of higher abstraction are captured by higher layers, their spatial concentration is expected to decrease suggesting that the ratio of 3×3 and 5×5 convolutions should increase as we move to higher layers.

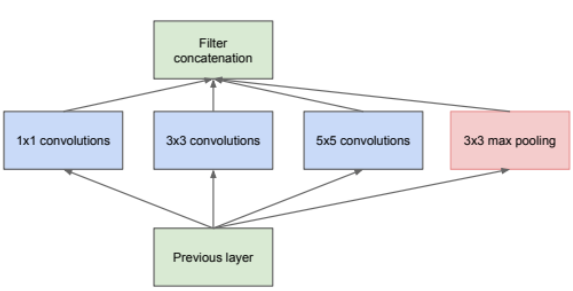
- 1x1, 3x3, 5x5 conv filter의 수는 신경망이 깊어짐에 따라 달라진다. 높은 추상적 개념들은 높은 layer에서만 포착되는데, 이는 서로 상관관계를 가지는 클러스터들의 공간적 집중도가 감소하는 것을 의미한다. 따라서 네트워크가 깊어질수록 3x3, 5x5 filter의 수가 늘어날 것이다.



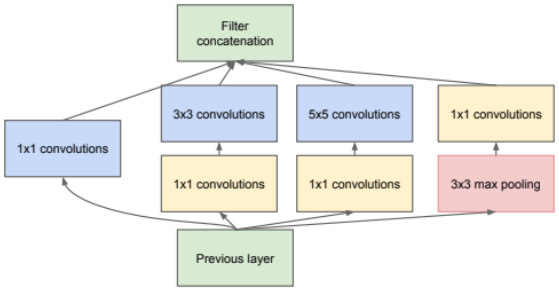
- 3x3, 5x5 conv filter가 늘어나면 생기는 문제

특히 5x5 filter를 사용할 경우 연산량이 많아지는데, 입력 feature map의 크기가 크거나 5 x 5 Convolutional filter의 수가 많아지면 **연산량은 더욱 증가하게 된다.** 이 연산량의 증가는 stage를 거듭함에 따라 계속 누적된다.

따라서 이 아키텍처가 설사 최적의 sparse data 구조를 가질지라도 몇개의 스테이지만에 연산량이 폭발하면서 매우 비효율적으로 동작한다.



(a) Inception module, naïve version



(b) Inception module with dimension reductions

- 따라서 이 문제를 해결하기 위해 **1 x 1 Convolutional filter를 이용하여 차원을 축소**하였다. **3 x 3과 5 x 5 앞에 1 x 1을 두어 차원을 줄이는데, 이를 통해 여러 Scale을 확보하면서도 연산량을 낮출 수 있다.**

추가적으로, Convlution 연산 이후에 추가되는 ReLU를 통해 비선형적 특징을 더하며 추가적인 연산량을 줄였다.

- 또한 inception network는 학습 시 메모리의 효율적 사용을 보장하기 위해 낮은 layer에서는 기본적인 CNN 모델을 사용하고, 높은 layer에서만 Inception module을 사용하는 것을 권장하고 있다

  이 Inception module의 이점은,

  1. **연산 복잡도(computational complexity)의 큰 증가 없이도 각 단계에서 unit 수를 증가**시킬 수 있다. 1x1 conv filter로 차원 축소를 하여 다음 layer의 input 사이즈를 조절할 수 있기 때문이다.
  2. 이미지는 다양한 scale로 처리되고 aggregate되어, 다음 단계에서는 서로 다른 레벨의 feature를 추출할 수 있어야 한다는 intuition과 부합한다.

     즉 1 x 1, 3 x 3, 5 x 5 Convolution 연산을 통해 다양한 scale의 특징을 추출할 수 있기 때문이다.

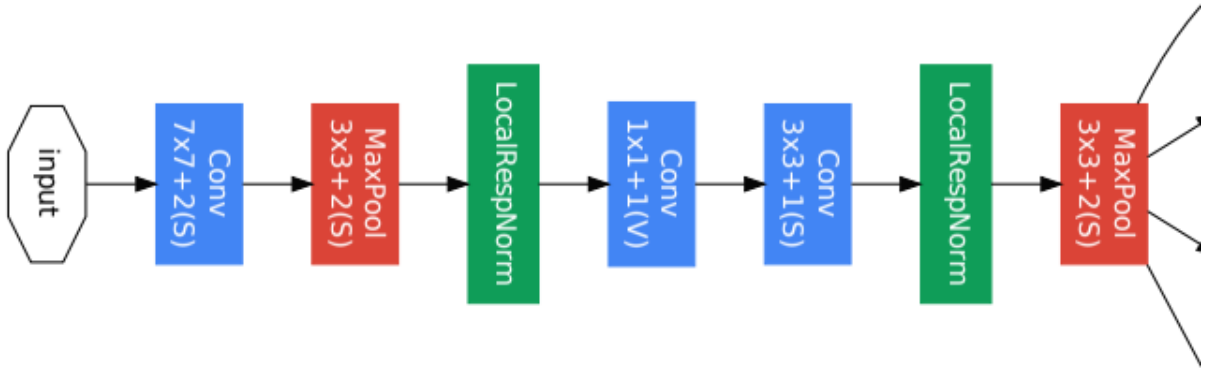
5. GoogLeNet

이 파트는 GoogLeNet의 전체 구조에 대해 다룬다.

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								

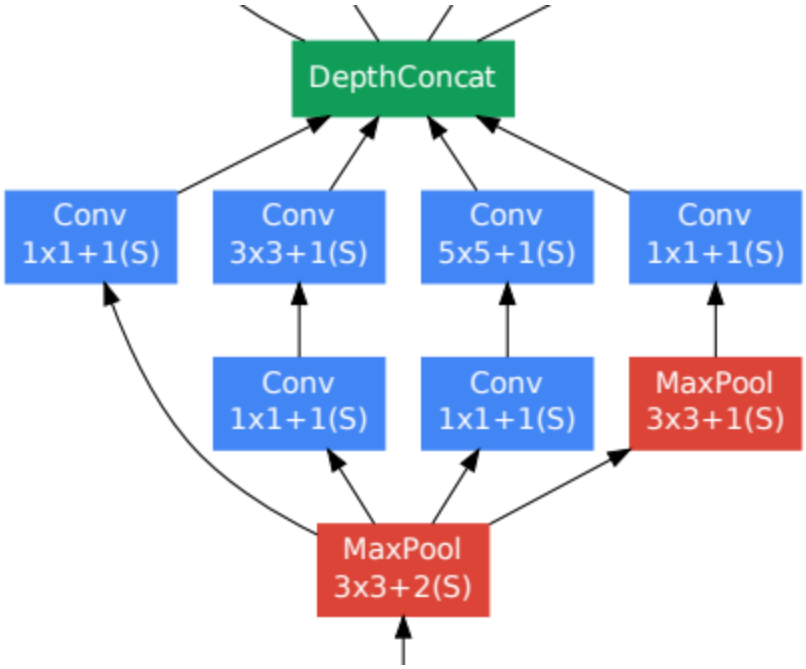
- 위 표에서 **# 3 x 3 reduce**와 **# 5 x 5 reduce**는 **3 x 3과 5 x 5 Convolutional layer 앞에 사용되는 1 x 1 필터의 채널 수를 의미**한다. 그리고 **pool proj**열은 **max pooling layer 뒤에 오는 1 x 1 필터의 채널 수를 의미**한다. 여기서 **모든 reduction 및 projection layer에 ReLU가 사용**된다.
- Inception Module in 4 parts

1. Part 1



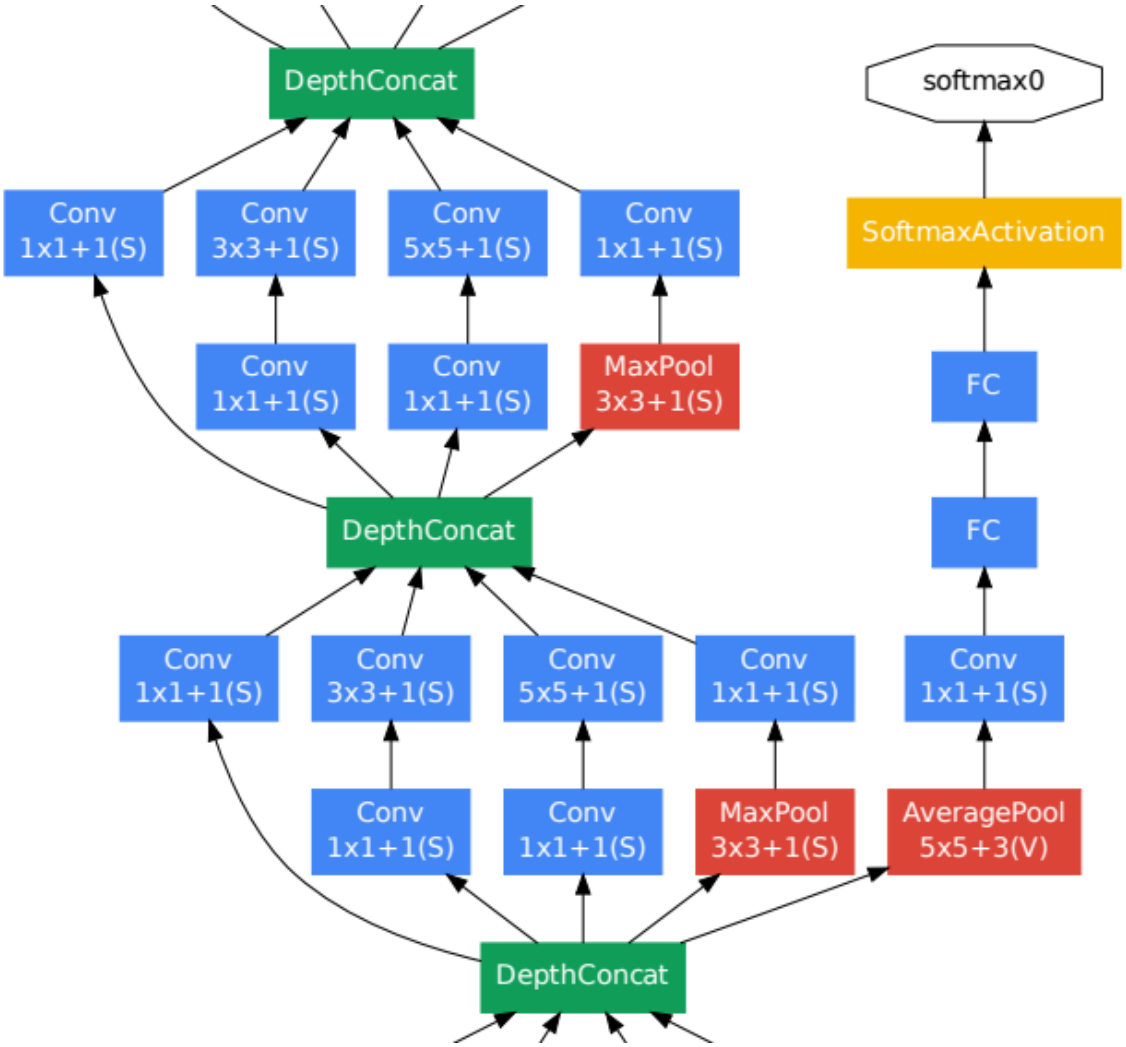
Part 1은 input layer와 가까운 낮은 layer가 있는 부분이다. 효율적인 메모리의 사용을 위해 이 부분에서는 inception을 사용하지 않고 기본적인 CNN모델(traditional convolutional fashion)로 구현되었다.

2. Part 2



Part 2는 **Inception module**로 **다양한 특징을 추출하기 위해 1 x 1, 3 x 3, 5 x 5 Convolutional layer가 병렬적으로 연산을 수행**하고, **차원을 축소하여 연산량을 줄이기 위해 1 x 1 Convolutional layer가 적용**되어 있는 것을 확인할 수 있다

3. Part 3



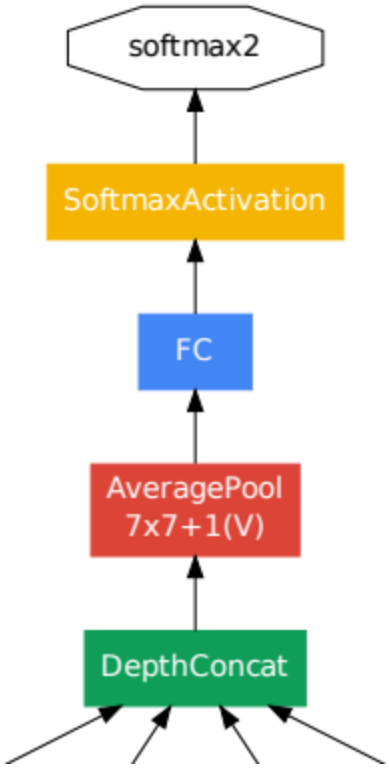
파트 2의 인셉션 모듈은 계속 깊게 쌓여나가는데, 깊은 모델에서는 역전파 될때 기울기가 0으로 수렴하는 gradient vanishing 문제가 발생할 수 있다. 파트 3에서는 인셉션 모델 중간에 추가하여 기울기 소실 문제를 해결할 수 있는 auxiliary classfier를 소개한다.

> “One interesting insight is that the strong performance of relatively shallower networks on this task suggests that **the features produced by the layers in the middle of the network should be very discriminative**. By adding auxiliary classifiers connected to these intermediate layers, we would expect to **encourage discrimination in the lower stages in the classifier, increase the gradient signal that gets propagated back, and provide additional regularization**.”

- 상대적으로 얕은 신경망의 강한 성능을 통해 신경망의 중간 layer에서 생성된 feature가 매우 discriminative 하다고 추정한다. 따라서 이 중간 layer에 auxiliary classifier를 추가하

여, 중간중간에 결과를 출력해 추가적인 역전파를 일으켜 gradient가 전달될 수 있게끔 하면서도, 추가적인 regularization 효과를 노렸다.

4. Part4



- Part 4는 예측 결과가 나오는 모델의 끝 부분이다.

  여기서 최종 Classifier 이전에 average pooling layer를 사용하고 있는데 이는 GAP (Global Average Pooling)가 적용된 것으로 이전 layer에서 추출된 feature map을 각각 평균 낸 것을 이어 1차원 벡터로 만들어 준다. 이를 fully connected layer로 연결한 다음 softmax를 통해 분류를 한다.

  - conv layer를 flatten해서 fc layer로 연결하는 것보다 GAP를 사용하는 편이 가중치의 개수를 상당히 많이 줄일 수 있다.

6. Training Methodology

- 여기서는 모델 training을 어떻게 하였는지에 대해 설명한다.
- Google 팀에서는 0.9 momentum의 Stochastic gradient descent를 이용하였고, learning rate는 8 epochs 마다 4%씩 감소시켰다.

- 또한, 이미지의 가로, 세로 비율을 3 : 4와 4 : 3 사이로 유지하며 본래 사이즈의 8% ~ 100%가 포함되도록 다양한 크기의 patch를 사용하였다. 그리고 photometric distortions를 통해 학습 데이터를 늘렸다고 한다.

9. Conclusion

Our results seem to yield a solid evidence that approximating the expected optimal sparse structure by readily available dense building blocks is a viable method for improving neural networks for computer vision. The main advantage of this method is a significant quality gain at a modest increase of computational requirements compared to shallower and less wide networks. Also note that our detection work was competitive despite of neither utilizing context nor performing bounding box

regression and this fact provides further evidence of the strength of the Inception architecture. Although it is expected that similar quality of result can be achieved by much more expensive networks of similar depth and width, our approach yields solid evidence that moving to sparser architectures is feasible and useful idea in general. This suggest promising future work towards creating sparser and more refined structures in automated ways on the basis of [2].

- 이미 존재하고 있는 dense한 네트워크에서 최적의 sparse structure를 근사하는 것은 컴퓨터비전 기반 신경망을 발전시키는 방법이 될 수 있다. 비슷한 depth와 width를 가진, 연산량이 더 많은 신경망 모델을 통해서도 GoogLeNet과 비슷한 결과를 낼 수 있을지도 모르나, GoogLeNet의 연구결과는 조금 더 sparse한 구조로 신경망을 짜는 것이 유용한 방법이라는 것을 증명했다.

참고 문헌

1. arxiv.org/abs/1409.4842

2.https://phil-baek.tistory.com/entry/3-GoogLeNet-Going-deeper-with-convolutions-논문-리뷰

3.http://cs231n.stanford.edu/slides/2021/lecture_5.pdf