GoogLeNet (Inception)

<Going Deeper with Convolutions>

서울과학기술대학교 국방인공지능응용학과
이찬호

Contents

Part 1

GoogLeNet 소개

Part 1

GoogLeNet 소개

소개

- 이름은 LeNet으로부터 유래
- Inception이라는 독특한 구조를 사용
- Inception은 논문 NIN과 영화 Inception에서 유래됨
- ILSVRC 2014에서 VGGNet을 제치고 1등을 차지
- 1X1 Conv Layer와 Depth를 늘려 모델 성능 개선 등 VGGNet과 유사
- 연산 시 소모되는 자원 사용 효율 개선이 주요 특징
- 정교한 설계로 인해 네트워크의 Depth와 Width를 늘려도 연산량이 증가하지 않음
- 성능 최적화를 위해 Hebbian Principle (헵의 이론)과 Multi-scale processing을 적용

Fig 1 영화 Inception 대사

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.

Fig 2 논문 Network In Network

Part 2

GoogLeNet 리뷰

Part 2

GoogLeNet 리뷰

Introduction

- 12~15년은 딥러닝의 발전으로 CNN 분야에서 큰 발전이 이루어짐
- 이 발전은 더 좋은 하드웨어, 더 큰 데이터 셋, 더 큰 모델이 아닌 **새로운 아이디어, 알고리즘, 개선된 신경망 구조** 때문
- Google도 소프트웨어 측면에서 개선하기 위해 노력
- GoogLeNet은 AlexNet보다 **파라미터가 12배 적음**에도 불구하고 **더 정확함**
- GoogLeNet은 엄격하게 고정된 구조보다 **유연한 구조**를 가짐
- 추론 시간에 **1.5B 이하의 연산만 수행**하게 함

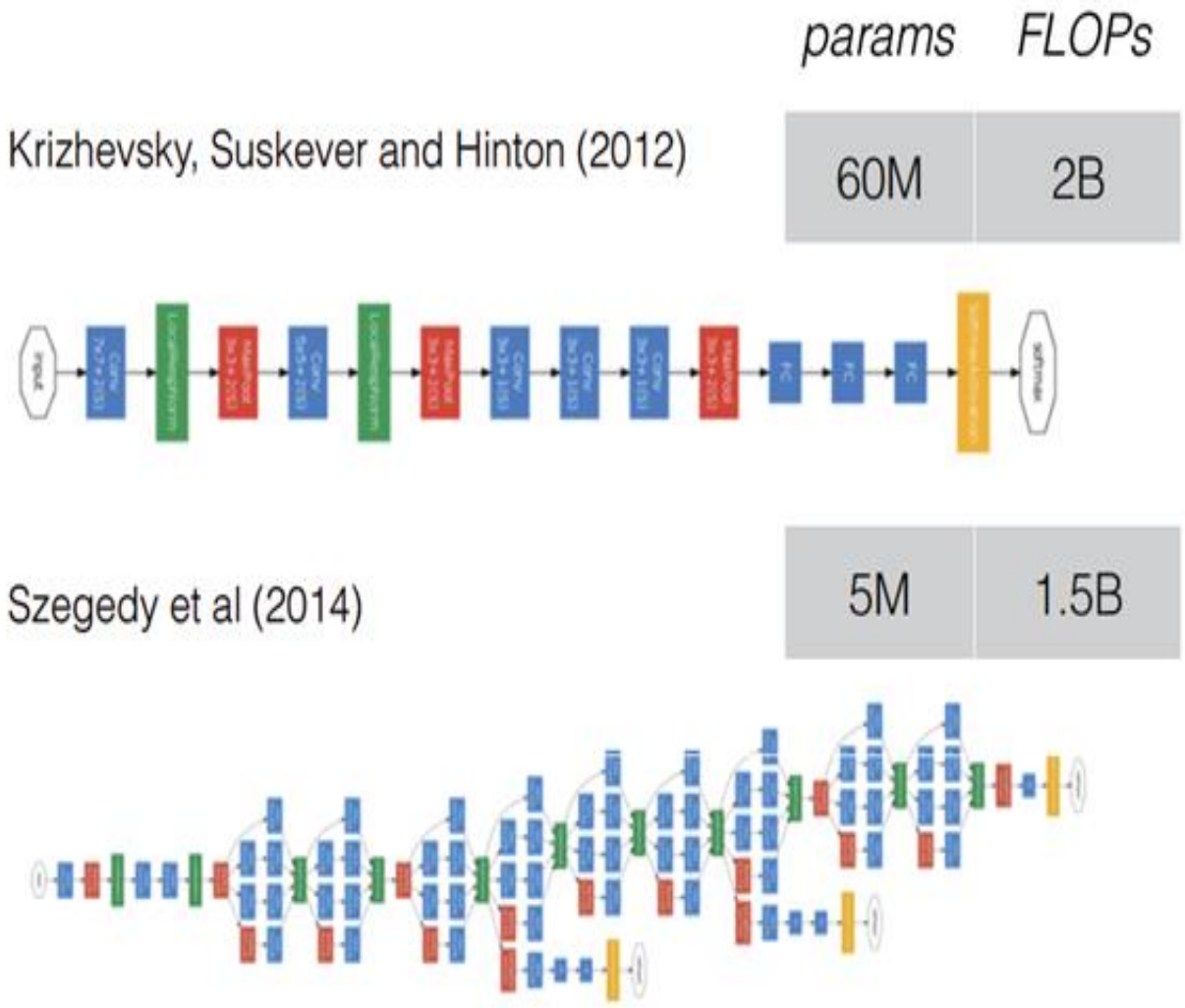

Fig 3 AlexNet(위) & GoogLeNet(아래)

Part 2

GoogLeNet 리뷰

Related Work

- LeNet을 시작으로 CNN은 **Conv layer가 쌓이고 뒤에 FC layer를 쌓는 일반적인 표준 구조**를 가지게 됨
- 'Network In Network'에서 신경망의 표현력을 높이기 위해 **1X1 Conv layer**와 **ReLU 활성화 함수**를 사용
- 1X1 Conv layer는 **병목현상을 제거하기 위한 차원 축소**와 **네트워크 크기 제한**을 **목적**으로 사용
- GoogLeNet 또한 두 가지 목적을 위해 1X1 Conv layer사용
- **LeNet**과 **Network In Network**의 특징들 GoogLeNet에 **많은 영향**을 끼치게 됨

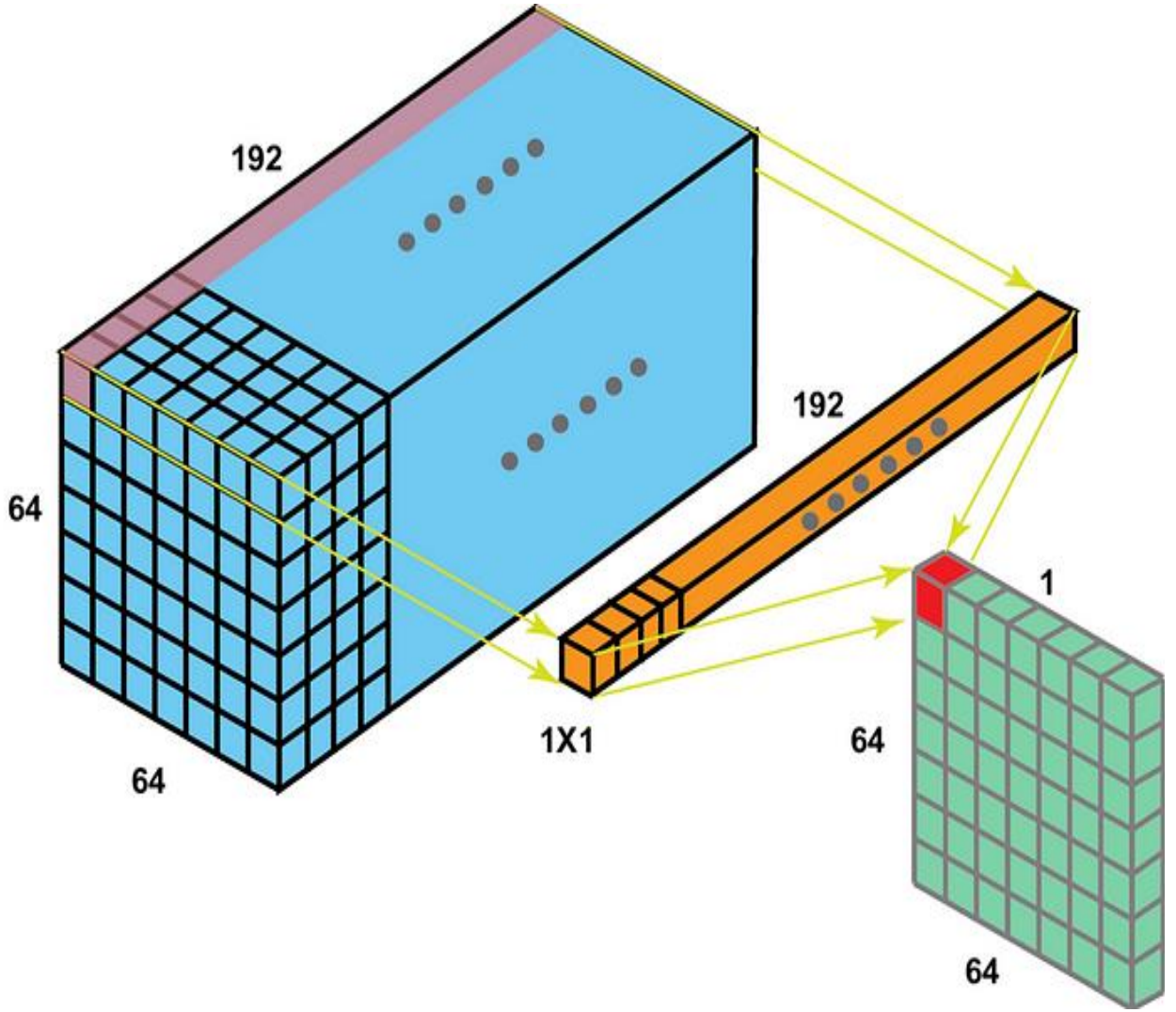

Fig 4 1X1 Conv layer

Part 2

GoogLeNet 리뷰

Motivation and High Level Considerations

- 성능을 개선시키는 가장 **간단한 방법**은 Depth와 Width 증가시켜 신경망의 크기를 키우는 것
- Depth와 Width 증가 시 **두 가지 문제점 발생**
  - 파라미터 수 증가할 경우 적은 데이터로 학습 시 오버피팅 발생으로 인해 **병목 현상**이 발생
  - **컴퓨터 리소스 사용량 증가**
- 해결 방법은 Full Connection 구조에서 **Sparsely Connection 구조**로 변경
- Inception은 유사 Sparse 구조를 시험하기 위해 시작
- Learning Rate, Hyper Parameter 조정, 훈련 방법 개선
- Localization, Object Detection에서 좋은 성능을 냄

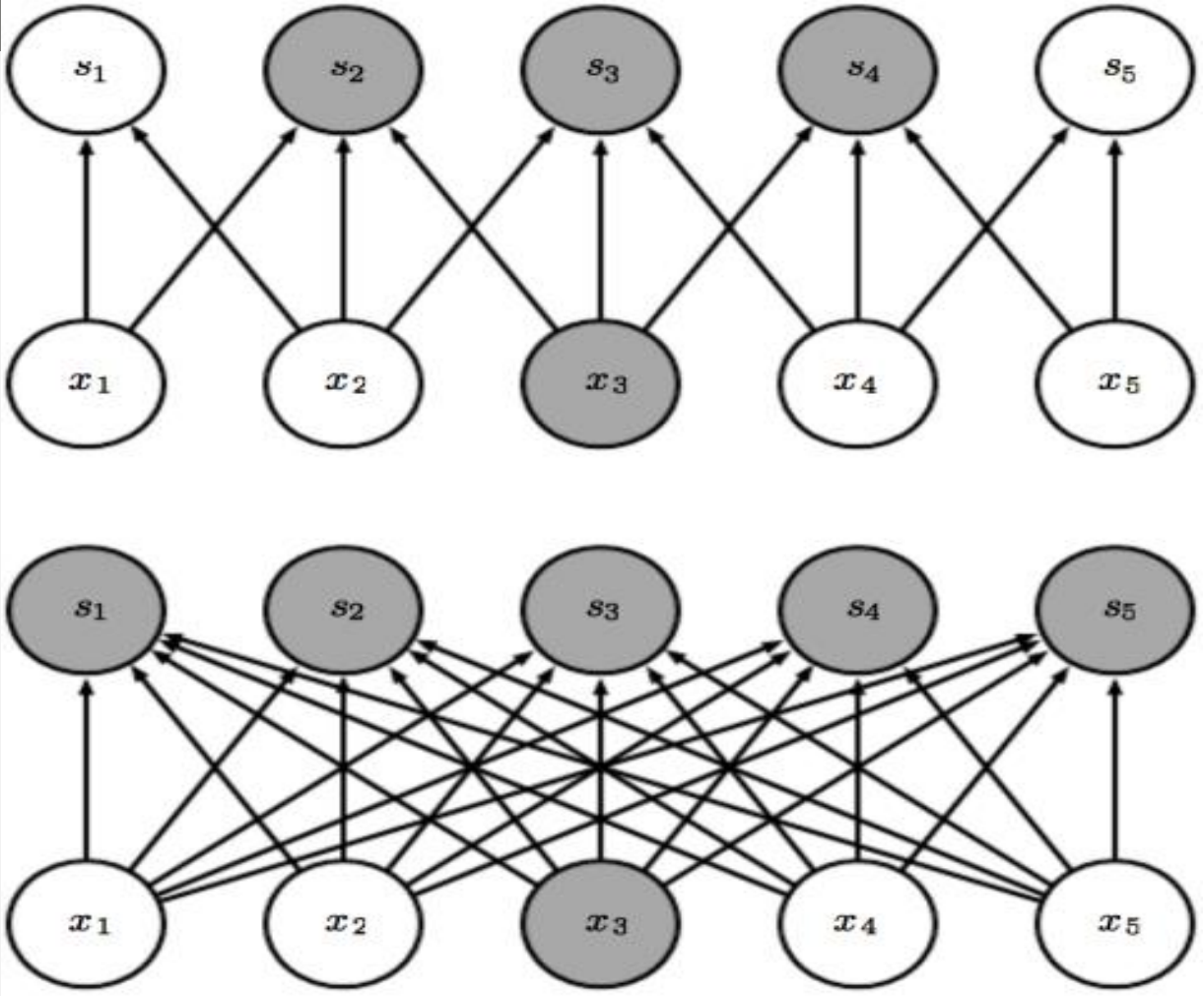

Fig 5 Sparsely Connection (위) & Full Connection 구조 (아래)

Part 2

GoogLeNet 리뷰

Architectural Details - Inception(1)

- Inception 주 아이디어는 CNN에서 각 요소를 **최적의 Local sparce structure로 근사화 후에 Dense component로 바꾸는 것**
- 1X1 Conv를 사용해 input img와 가까운 **Lower layer에서 Correlated unit을 처리**
- **더 넓은 Filter로 Correlated unit의 비율을 처리**하기 위해 **1X1, 3X3, 5X5 Conv를 병렬 수행**
- Higher layer를 향해서 네트워크가 깊어질수록 **3X3, 5X5 Conv 개수 증가**
- 3X3, 5X5 Conv의 수가 많아질수록 **연산량 증가**하여 **계산 시간, 리소스 사용량 증가**하는 문제 발생

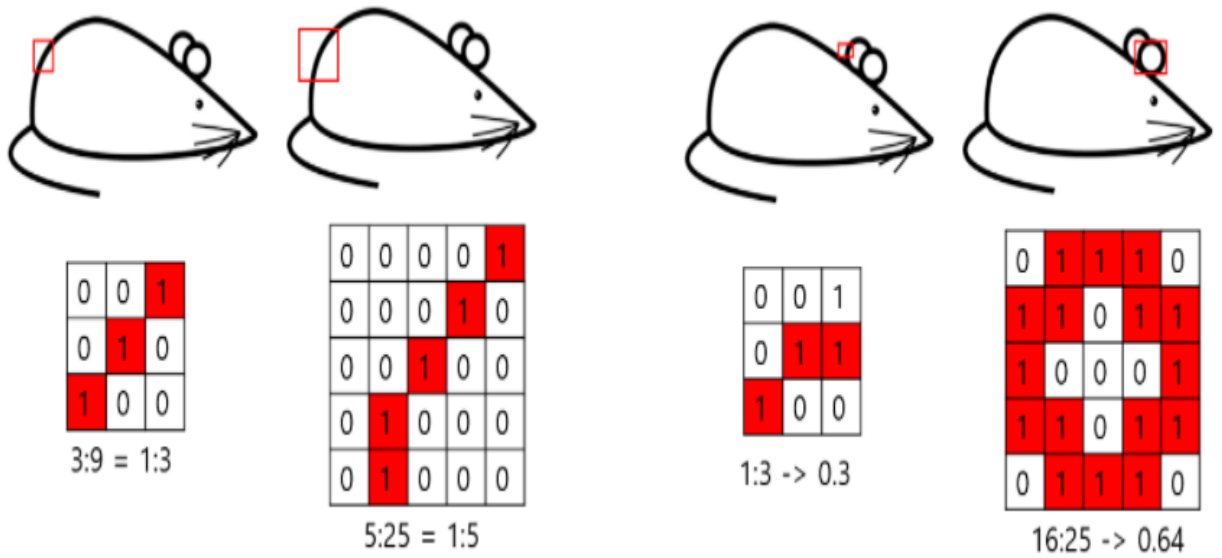

Fig 6 넓은 **filter**로 **Correlated unit** 탐지

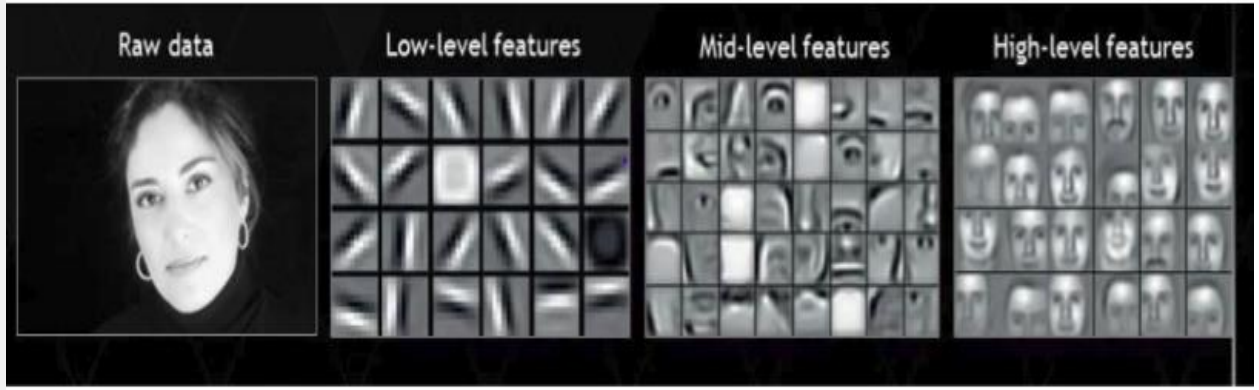

Fig 7 level별 특징 추출

Part 2

GoogLeNet 리뷰

Architectural Details - Inception(2)

- 3X3, 5X5 Conv 앞에 1X1 Conv를 둬서 **차원을 줄여 연산량을 낮추고 ReLU로 비선형성 추가**
- Lower layer에서 기본 CNN 적용, Higher layer에서 Inception module 적용
- Inception 적용 시 효과
  - 연산량 증가 문제 없이 각 layer에서 unit 수 증가
  - 1X1, 3X3, 5X5 Conv 연산으로 다양한 특징 추출

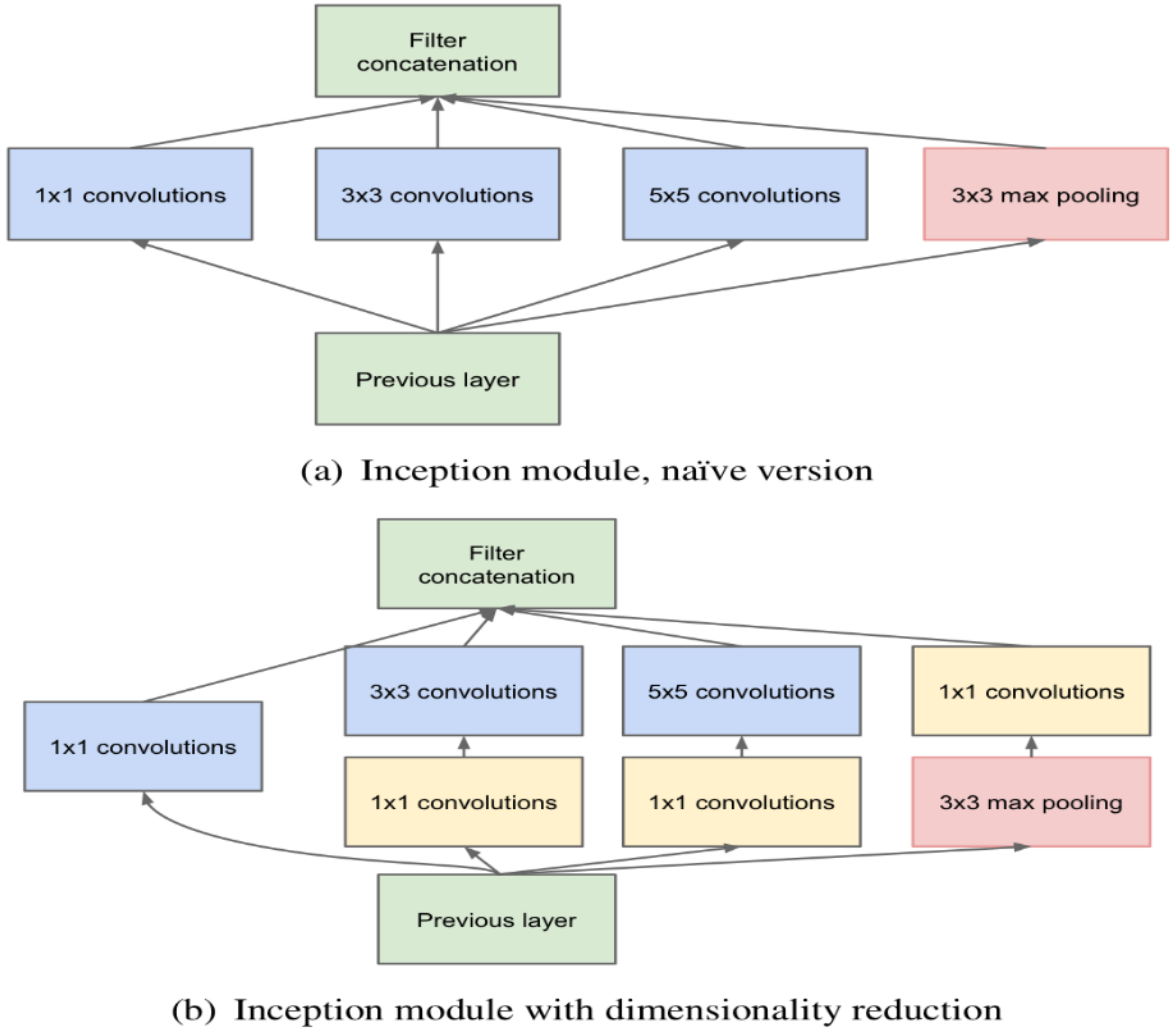

Fig 8 연산량 문제 해결 전 (위) **& 1X1 Conv** 추가로 연산량 해결 (아래)

Part 2

GoogLeNet 리뷰

GoogLeNet - Inception Architecture

- Inception module 내부 포함 모든 Conv layer에 ReLU 적용
- Receptive field 크기는 224X224 RGB이며 mean subtraction 적용
  → 모델 input 값은 224X224 mean subtracted RGB
- #3X3 reduce, #5X5 reduce는 3X3, 5X5 Conv layer 앞에 사용되는 1X1 filter 채널 수
- Pool projection은 max pooling layer 뒤에 1X1 filter 채널 수

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								

Fig 9 Inception architecture

Part 2

GoogLeNet 리뷰

GoogLeNet Architecture – (1)

- Low layer – Fig 10
  - Input img와 Lower layer가 위치
  - 리소스의 효율적인 사용을 위해 Lower layer에서 기본적인 CNN 모델 적용
  - Inception module 적용되지 않음
- Inception module – Fig 11
  - 다양한 특징 추출을 위해 1X1, 3X3, 5X5 Conv layer 병렬로 연산
  - 차원 축소로 연산량을 줄이기 위해 1X1 Conv layer 적용

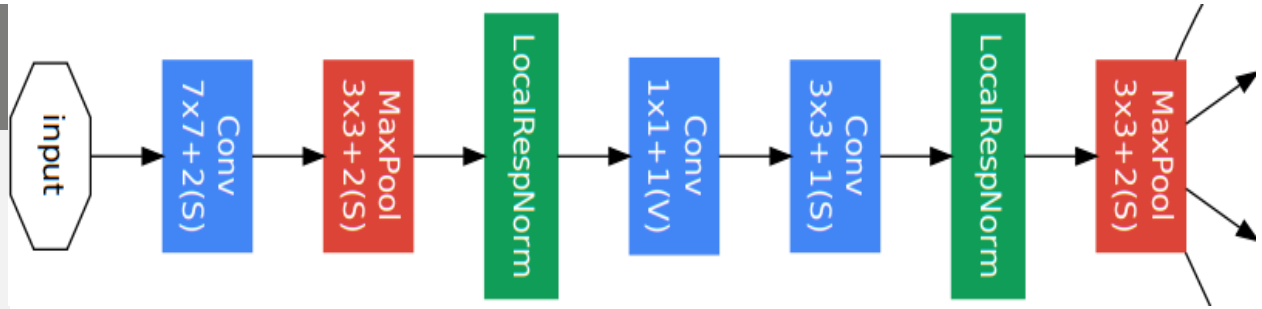

Fig 10 GoogLeNet Architecture – Lower layer

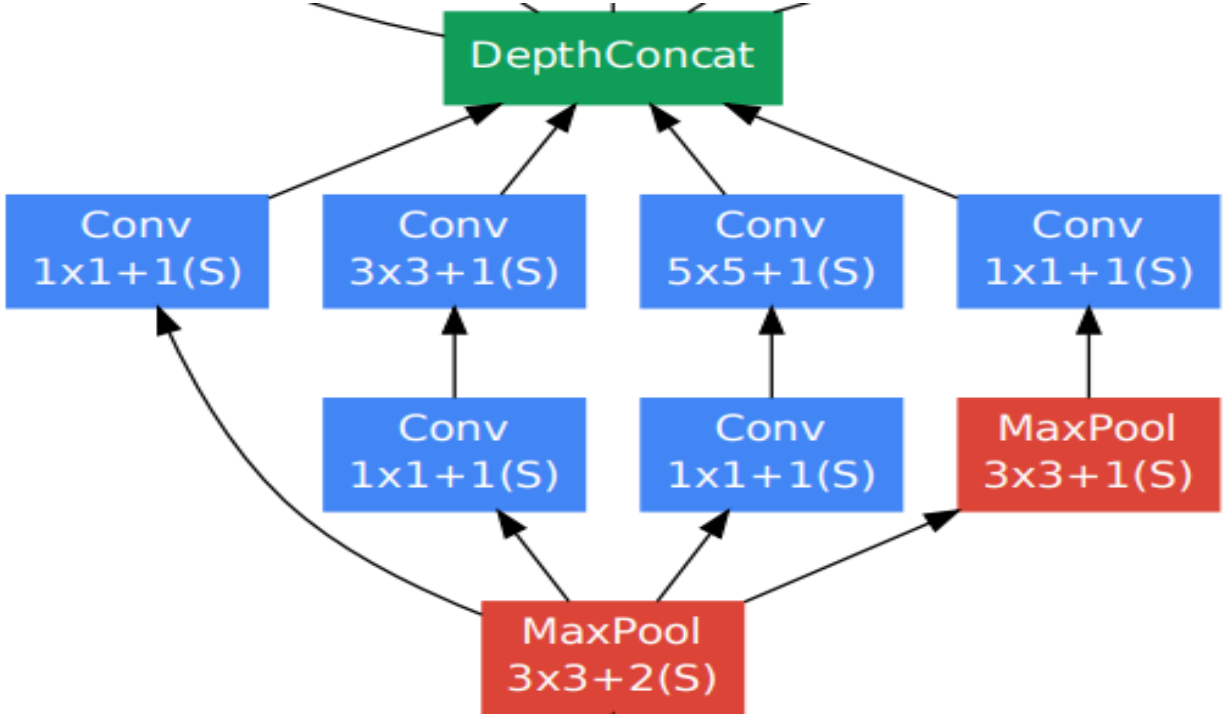

Fig 11 GoogLeNet Architecture – Inception module

Part 2

GoogLeNet 리뷰

GoogLeNet Architecture – (2)

- Auxiliary classifier – Fig 12
  - 모델이 **매우 깊을 경우 Gradient vanishing 문제** 발생
  - Middle layer에 **Auxiliary classifier**를 추가하여 **중간에 결과를 출력**해 **Backpropagation**을 **발생**시켜 **Gradient**를 **전달**
  - 큰 영향력 방지로 **Auxiliary classifier loss**에 **0.3**을 곱하고 **실 테스트 시 Auxiliary classifier 삭제**
- Model end – Fig 13
  - 최종 분류 전 Global Average pooling layer 사용
  - 이전 layer에서 추출된 feature map을 각각 평균 낸 것을 1차원 벡터로 만듦

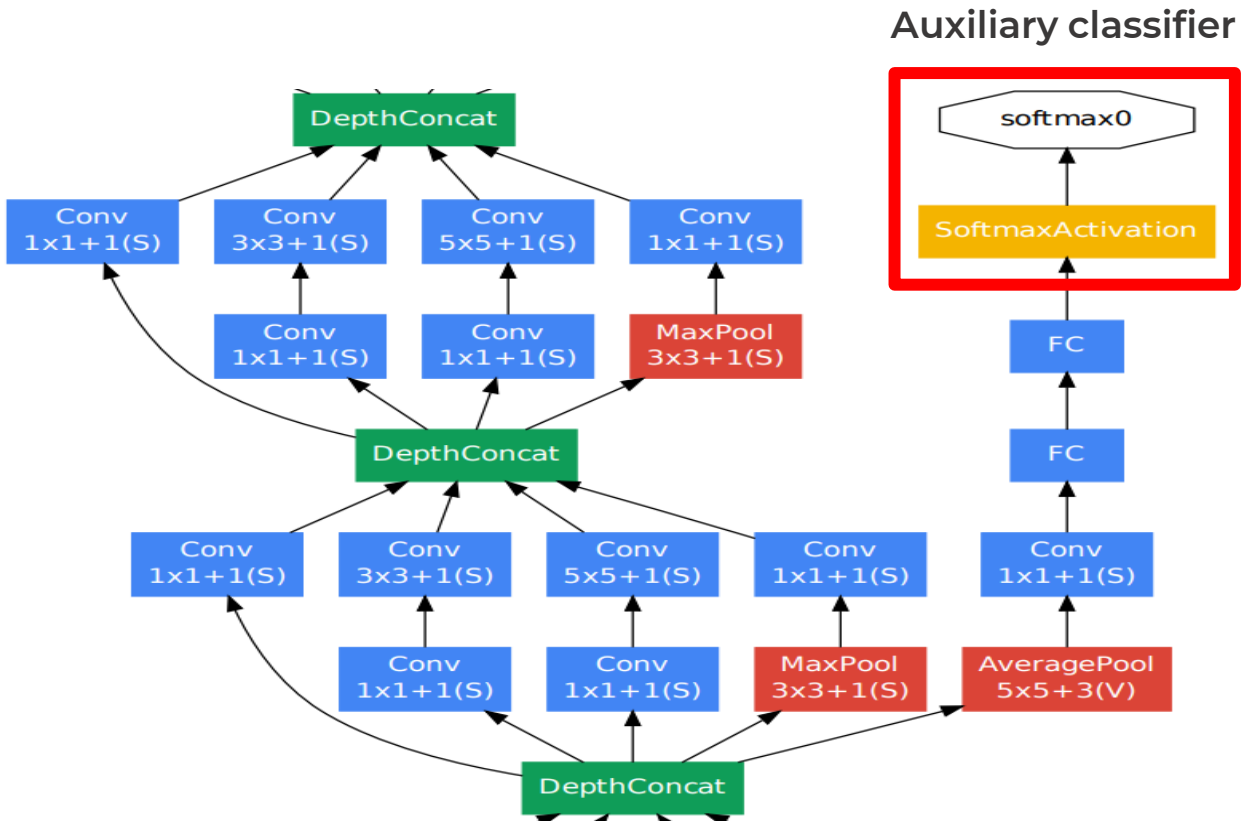

Fig 12 GoogLeNet Architecture – Auxiliary classifier

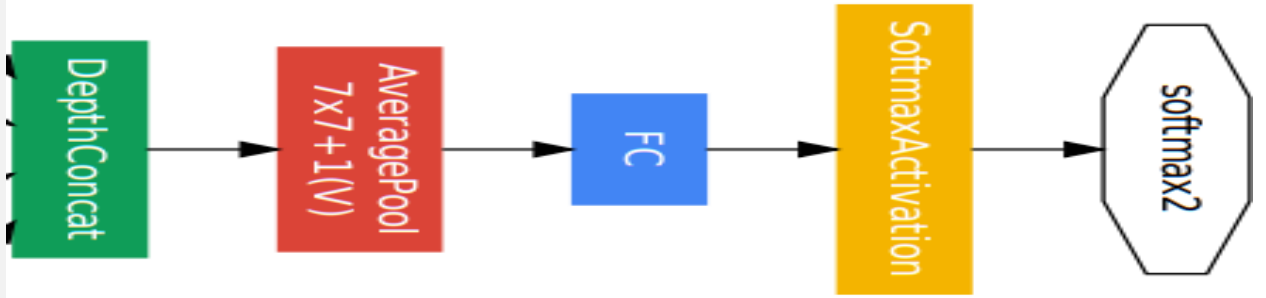

Fig 13 GoogLeNet Architecture – Model end

Part 2

GoogLeNet 리뷰

Training Methodology & Conclusions

- Stochastic gradient descent (0.9 Momentum)
- Learning rate (8 epochs마다 4% 감소)
- 이미지 가로세로 비율 (3 : 4, 4 : 3)
- 이미지 본래 사이즈의 8% ~ 100%가 포함된 다양한 크기의 patch
- Photometric distortions (광도 왜곡)으로 학습 데이터 증가
  - 밝기, 색조, 대비, 채도, 노이즈를 조정하여 동일한 이미지의 더 많은 다양성을 표시
- Inception 구조는 Sparse 구조를 Dense 구조로 근사화 하여 성능을 개선
- 성능은 대폭 상승, 연산량은 약간 증가하는 장점

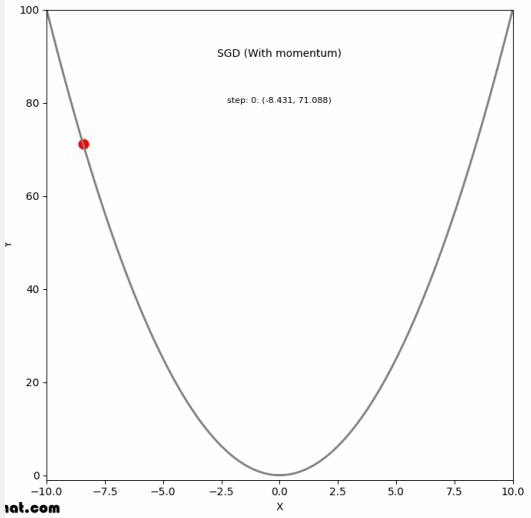

Fig 14
SGD with Momentum

Fig 15 Photometric distortions

THANK YOU FOR YOUR ATTENTION