Going Deeper with Convolutions

URL	1409.4842v1.pdf https://arxiv.org/abs/1409.4842
DATE	@2024년 7월 21일
Status	Done

Abstract

- **“Inception”: Deep convolutional neural network architecture**
  - Image classification과 detection에서 성능을 크게 향상시킴, ILSVRC14에서 입증됨.
  - 컴퓨터 자원의 효율적 활용, 네트워크의 깊이와 너비를 증가시키면서 계산 예산을 늘리지X
  - 22개의 레이어로 이루어진 GoogleNet 모델
  - Hebbian principle과 multi-scale processing에 기반하여 계산 효율성을 유지하면서 더 나은 quality를 제공함.

Introduction

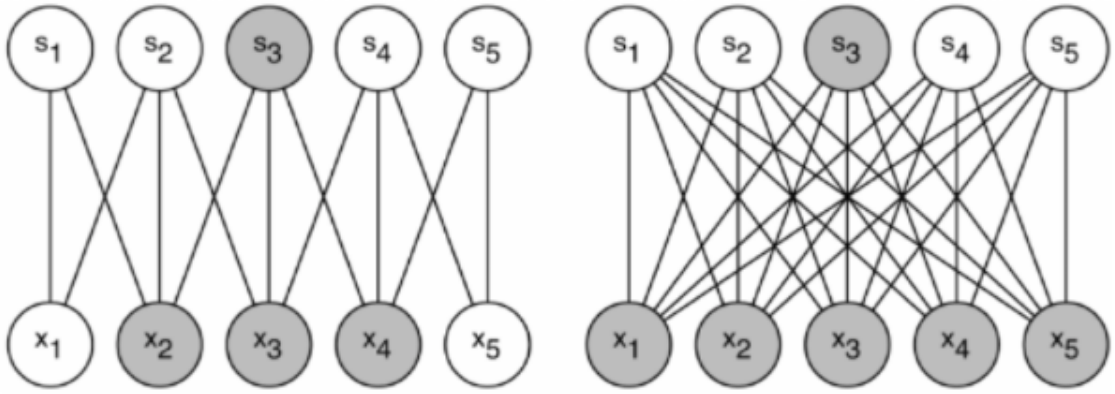
- GoogLeNet은 2년 전 우승한 AlexNet 아키텍처보다 12배 적은 파라미터를 사용하면서 더 높은 정확도를 가짐.
- 모바일과 임베디드 컴퓨팅의 발전으로 알고리즘의 효율성, 특히 전력 및 메모리 사용의 중요성이 커졌다.
- 이 논문에서는 Inception이라는 효율적인 심층 신경망 아키텍처에 대해 다루며, 이 아키텍처의 이점은 ILSVRC 2014 분류 및 탐지 과제에서 실험적으로 검증되었고, 최신 기술을 훨씬 능가하는 성능을 보인다.

Related Work

- LeNet-5를 시작으로 CNN은 일반적으로 여러 convolutional layer를 쌓고, 이후 fully-connected layer를 사용하는 표준 구조를 가짐
    - MNIST, CIFAR, 특히 ImageNet 분류 과제에서 가장 좋은 결과를 얻었다.
- 최근에는 더 많은 레이어와 레이어 크기를 증가시키고 dropout을 사용하여 과적합 문제를 해결하는 경향이 있다.
- Network-in-Network 접근법은 1×1 convolutional layer를 추가하고 그 뒤에 ReLu activation을 추가함 → 신경망의 표현력을 높이는 방법
    - 이 접근법은 주로 차원 축소 모듈로 사용되어 네트워크의 깊이와 너비를 성능 저하 없이 증가시킴.
- 이 논문의 GoogleNet에서도 1×1 convolutional layer를 사용하고 비슷한 pipeline을 채택했으며, 나은 분류를 위해 앙상블 접근법 등 여러 개선점을 탐구했음.

Motivation and High Level Considerations

- Deep neural network의 성능을 개선하는 방법은 network의 depth(level의 수 증가)와 width(각 level의 유닛 수 증가)를 증가시키는 것
    - overfitting과 gradient vanishing이 발생할 수 있음, 컴퓨터 리소스 사용이 급격히 증가함

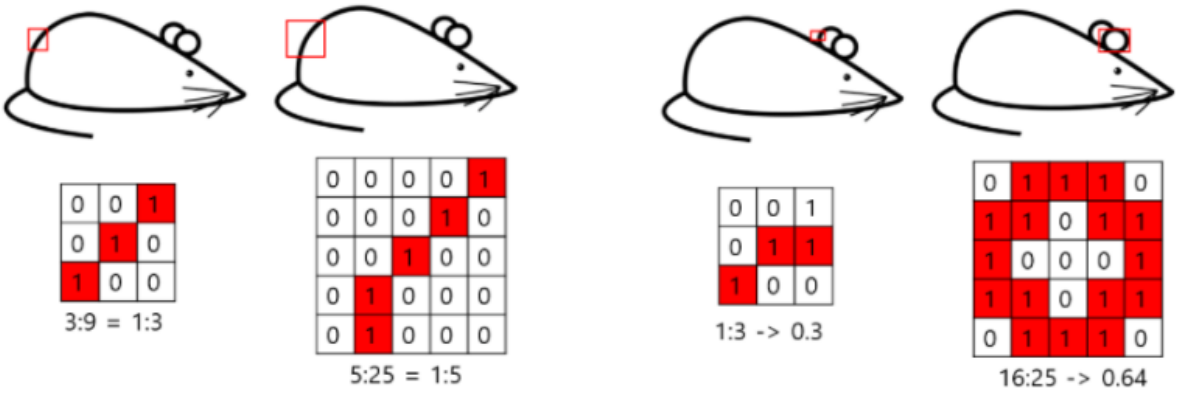


- 위의 2가지 문제를 해결하는 방법: dense한 fully-connected 구조에서 sparesely connected 구조로 바꾸는 것
- 하지만 오늘날의 컴퓨팅 환경은 균일하지 않은 sparse data 구조를 다룰 때 매우 비효율적임.
- 다른 연구에서 sparse matrix를 clustering하여 상대적으로 dense한 submatric를 만드는 것을 제안하였고 좋은 성능을 보였음.

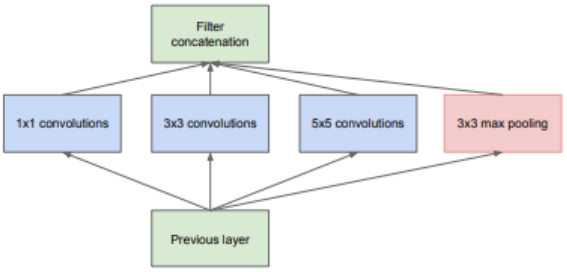
- Inception 구조는 유사 sparse 구조를 시험하기 위해 시작됨, 여러 번의 반복을 거쳐 개선된 성능을 보였으며 localization 및 object detection 분야에서 특히 좋은 성능을 보였다고 함.

Architectural Details

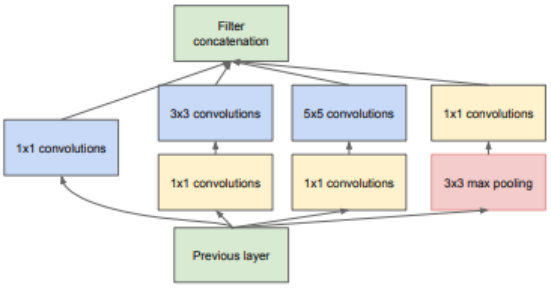
- Inception 구조의 주요 아이디어는 CNN에서 각 요소를 최적의 local sparse structure로 근사화하고, 이를 dense component로 바꾸는 방법을 찾는 것
  - 최적의 local 구성 요소를 찾고 공간적으로 반복 → sparse matrix를 clustering하여 상대적으로 dense한 submatrix를 만드는 것
- 이전 layer의 각 유닛이 입력 이미지의 특정 부분에 해당한다고 가정함
  - 입력 이미지와 가까운 낮은 layer에서는 특정 부분에 correlated unit이 집중돼있음
  - 단일 지역에 많은 cluster들이 집중된다는 뜻이므로 1x1 convolution으로 처리할 수 있음



- 몇몇 위치에서는 더 넓은 영역의 convolutional filter가 있어야 correlated unit의 비율을 높일 수 있는 상황이 나타날 수 있음
  - feature map을 효과적으로 추출할 수 있도록 1x1, 3x3, 5x5 convolution 연산을 병렬적으로 수행
  - CNN에서 pooling layer의 성능은 입증되었으므로 높이와 폭을 맞추기 위해 padding 추가



(a) Inception module, naïve version



(b) Inception module with dimension reductions

- 3x3, 5x5 convolutional filter도 사용할 경우, 연산량이 많아짐
  - 입력 feature map의 크기가 크거나 5x5 convolutional filte의 수가 많아지면 연산량이 더욱 증가하는 문제 발생
- 이 문제를 해결하기 위해 1x1 convolutional filter를 이용하여 차원을 축소함
  - 3x3과 5x5 앞에 1x1을 두어 차원을 줄이면 여러 scale을 확보하면서 연산량을 줄일 수 있음
  - convolution 연산 이후에 추가되는 ReLU를 통해 비선형적 특징을 추가할 수 있음.
- 효율적인 메모리 사용을 위해 낮은 layer에서는 기본적인 CNN 모델을 적용하고, 높은 layer에서는 Inception module을 사용하는 것이 좋다고 함.
  - Inception module을 사용하면 과도한 연산량 문제업이 각 단계에서 유닛 수를 상당히 증가시킬 수 있다 → 차원 축소를 통해 다음 layer의 input 수를 조절할 수 있기 때문
  - visual 정보가 다양한 scale로 처리되고, 다음 layer는 동시에 서로 다른 layer에서 특징을 추출할 수 있다 → 1x1, 3x3, 5x5 convolution 연산을 통해 다양한 특징을 추출할 수 있기 때문

GoogLeNet

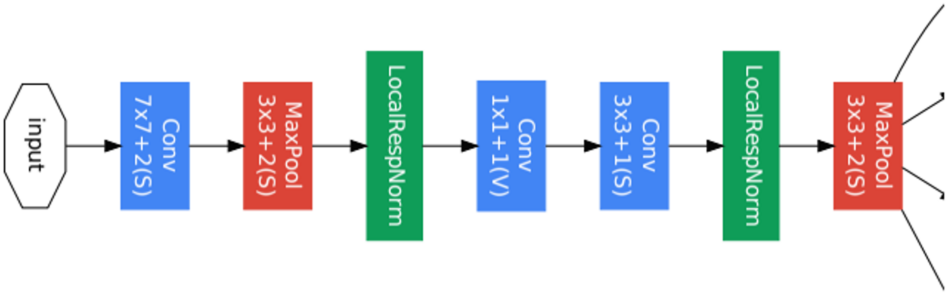
- Inception module이 적용된 전체 GoogleNet의 구조
- GoogleNet이라는 이름은 LeNet으로부터 유래함
- 네트워크 깊이는 22 layers (pooling까지 포함하면 27 layers)
- Inception module 내부를 포함한 모든 convolutinal layer에는 ReLU가 적용돼있음 + receptive field의 크기는 224x224로 RGB 컬러 채널을 가지며, mean subtraction을 적용한다.

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								

Inception 구조에 대한 GoogleNet 성체

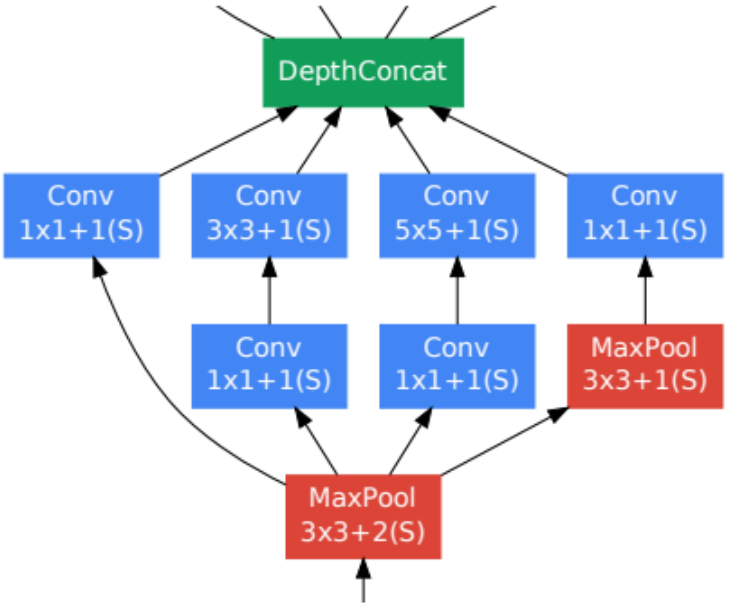
- #3x3 reduce와 #5x5 reduce는 3x3과 5x5 convolutional layer 앞에 사용되는 1x1 필터의 채널 수를 의미
- pool proj는 max pooling layer 뒤에 오는 1x1 필터의 채널 수 의미
- 모든 reduction 및 projection layer에 ReLU가 사용됨

1. 입력 이미지와 가까운 낮은 레이어가 위치해 있는 부분



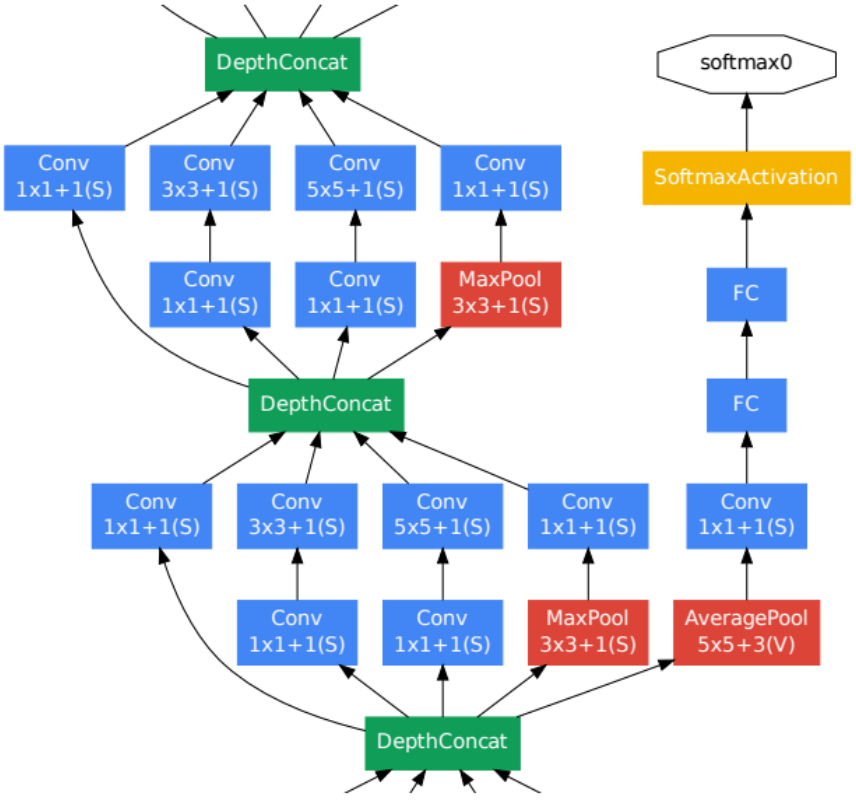
- 효율적인 메모리 사용을 위해 낮은 layer에서는 기본적인 CNN 모델을 적용

2. Inception module



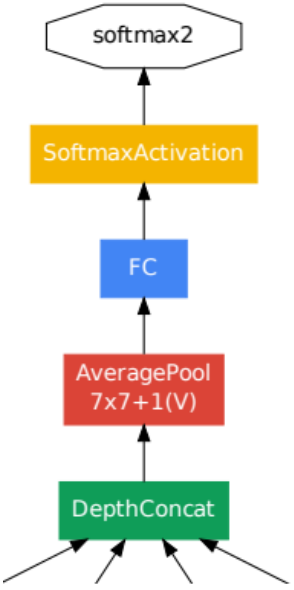
- 다양한 특징을 추출하기 위해 1x1, 3x3, 5x5 convolutional layer가 병렬적으로 연산을 수행
- 차원을 축소해서 연산량을 줄이기 위해 1x1 convolutional layer가 적용돼있음

3. Auxiliary classifier가 적용된 부분 (softmax)

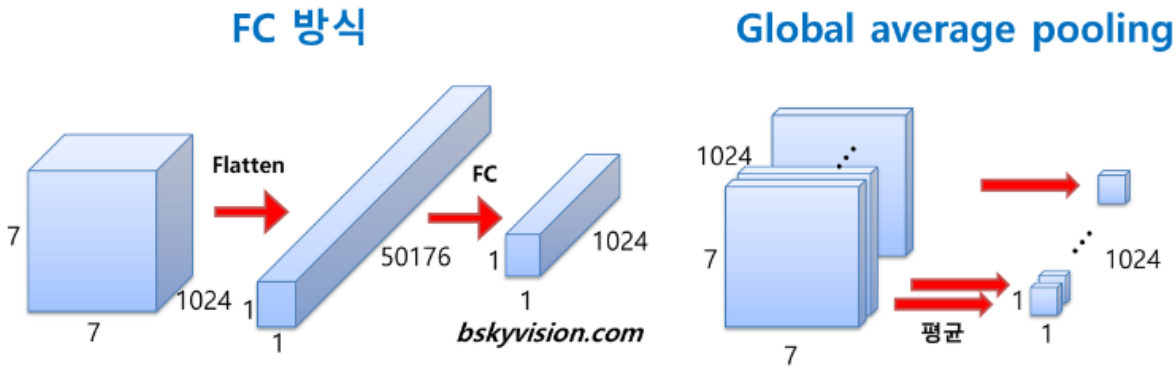


- 모델의 깊이가 매우 깊으면 기울이가 0으로 수렴하는 gradient vanishing 문제가 발생할 수 있음
- 중간 layer에 auxiliary classifier를 추가하여, 중간중간 결과를 출력해 추가적인 역전파를 일으켜 gradient가 전달될 수 있게 하면서 정규화 효과가 나타나도록 함
- 지나치게 영향을 주는 것을 막기 위해 auxiliary classifier의 loss에 0.3을 곱하고, 실제 테스트 시에는 auxiliary classifier를 제거 후, 제일 끝단의 softmax만 사용함.

4. 모델의 끝 부분



- 최종 classifier 이전에 averge pooling layer 사용 → global average pooling이 적용된 것으로 이전 layer에서 추출된 feature map을 각각 평균 낸 것을 이어 1차원 벡터로 만들어줌; 1차원 벡터로 만들어줘야 최종적으로 이미지 분류를 위한 softmax layer와 연결할 수 있기 때문



- 1차원 벡터로 만들면 가중치의 개수를 많이 줄여주는데, FC 방식을 이용하면 가중치 개수가 7x7x1024x1024=51.3M이지만 GAP를 이용하면 단 1개의 가중치도 필요하지 않음 → fine tuning을 하기 쉽다

Training Methodology

- 모델 훈련 방법에 대한 설명
- 0.9 momentum의 stochastic gradient descent를 이용하였고, learning rate는 8 epochs마다 4% 감소시킴
- 이미지의 가로:세로 비율을 3:4와 4:3 사이로 유지하며 본래 사이즈의 8~100%가 포함되도록 다양한 크기의 patch 사용
- photometric distortions를 통해 학습 데이터를 늘림 → overfitting 방지

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

- 2014년 ILSVRC 대회 우승을 하였고, 검증 및 테스트 데이터 모두에서 6.67%의 오차율을 기록함

ILSVRC 2014 Detection Challenge Setup and Results

Team	Year	Place	mAP	external data	ensemble	approach
UvA-Euvision	2013	1st	22.6%	none	?	Fisher vectors
Deep Insight	2014	3rd	40.5%	ImageNet 1k	3	CNN
CUHK DeepID-Net	2014	2nd	40.7%	ImageNet 1k	?	CNN
GoogLeNet	2014	1st	43.9%	ImageNet 1k	6	CNN

Table 4: Detection performance

Team	mAP	Contextual model	Bounding box regression
Trimps-Soushen	31.6%	no	?
Berkeley Vision	34.5%	no	yes
UvA-Euvision	35.4%	?	?
CUHK DeepID-Net2	37.7%	no	?
GoogLeNet	38.02%	no	no
Deep Insight	40.2%	yes	yes

Table 5: Single model performance for detection

- Table 5에서 단일 모델만을 사용한 결과를 비교하는데, 이때 최고 성능의 모델은 Deep Insight이지만, 앙상블을 사용하게 되면 GoogLeNet은 놀라운 성능 향상을 보임.

Conclusions

- Inception 구조는 sparse 구조를 dense 구조로 근사화하여 성능을 개선함
- 기존에 CNN 성능을 높이기 위한 방법과는 다른 새로운 방법이었으며, 성능은 많이 상승하지만 연산량은 약간만 증가한다는 장점이 있음.