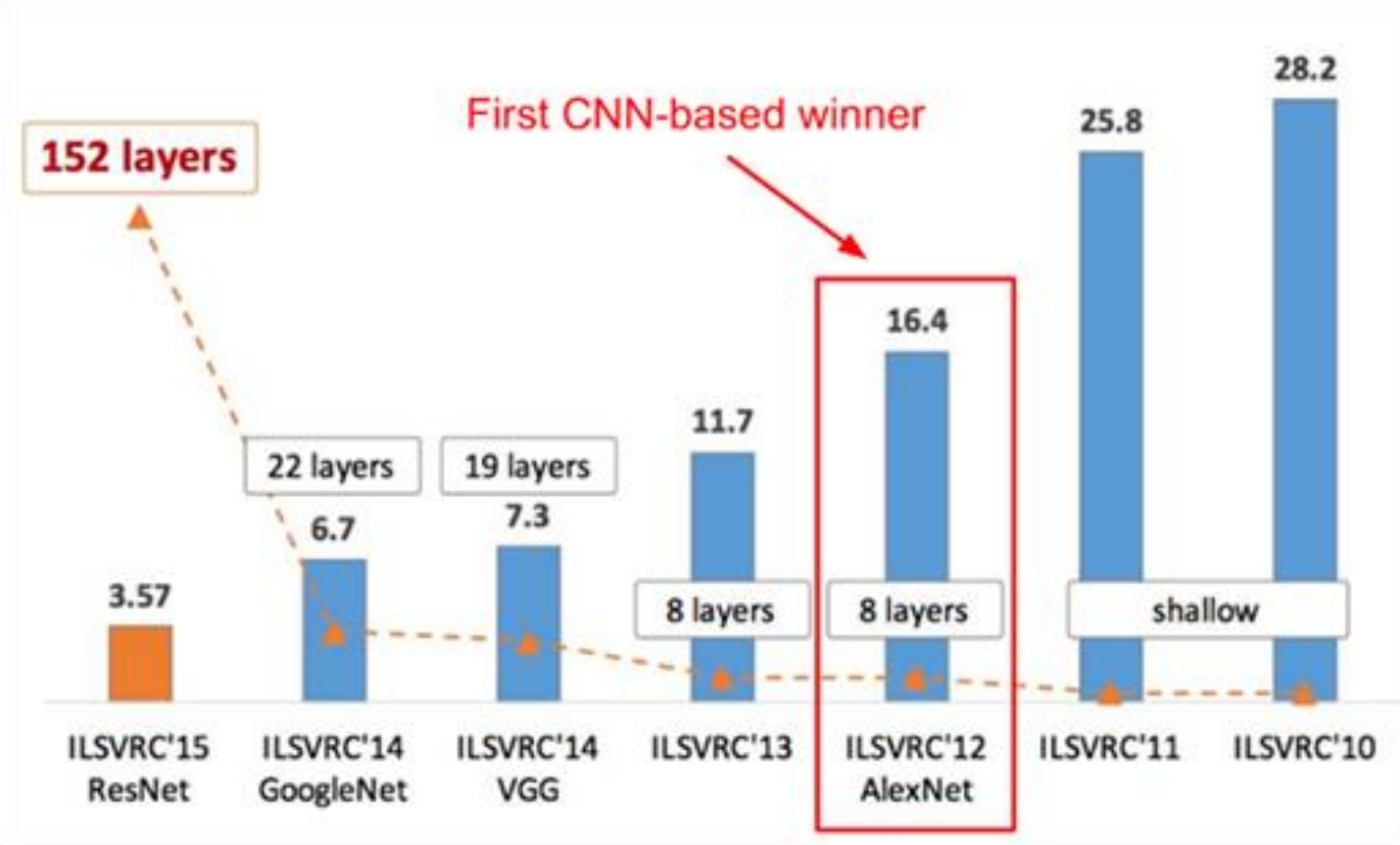
GoogLeNet

Going Deeper with Convolutions

ImagenNet Large Scale Visual Recognition Challenge **(ILSVRC) winners**

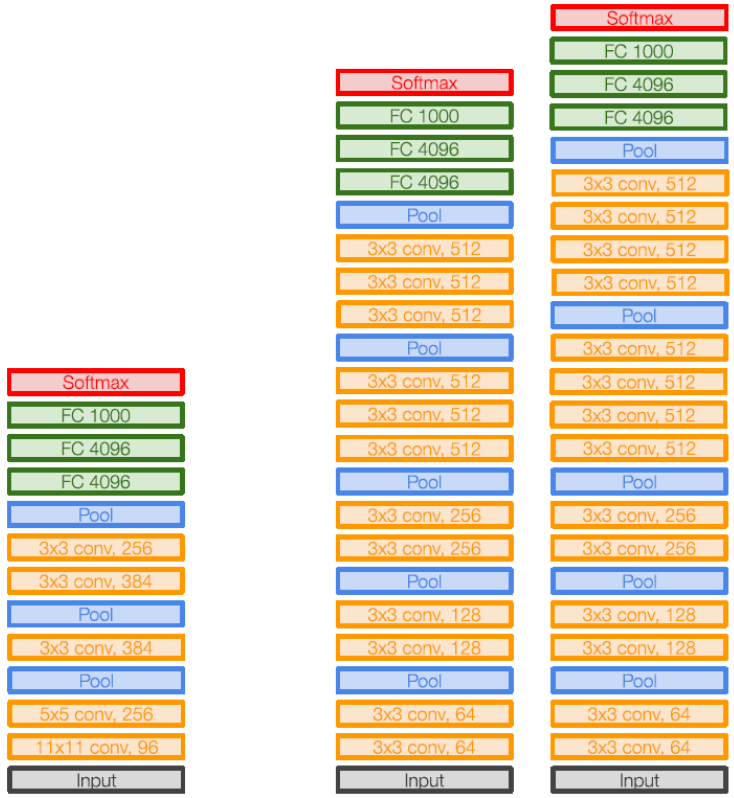


AlexNet: **First CNN-based model**, shallow model이 아닌 Deep Neural Network (8 layers)

VGGNet: **Small filters, Deeper networks**, 3x3 conv filter 사용, 더 적은 파라미터로 복잡한 모델 (19 layers)

CNN 모델의 성능을 향상시키는 방법

(Going Deeper with Convolutions)



⇒ depth와 size (width; the number of parameter) 늘리기

기존 CNN 기반 모델의 문제 1: Overfitting

(a) Siberian husky

(b) Eskimo dog

두 이미지를 구분하는 것은 사람도 쉽지 않음

(다양한 클래스로 세분화된 ImageNet과 같은 데이터셋의 bottleneck)

⇒ 모델이 커질수록, 과적합이 발생할 가능성이 높음

기존 CNN 기반 모델의 문제 2: Increase of Computation

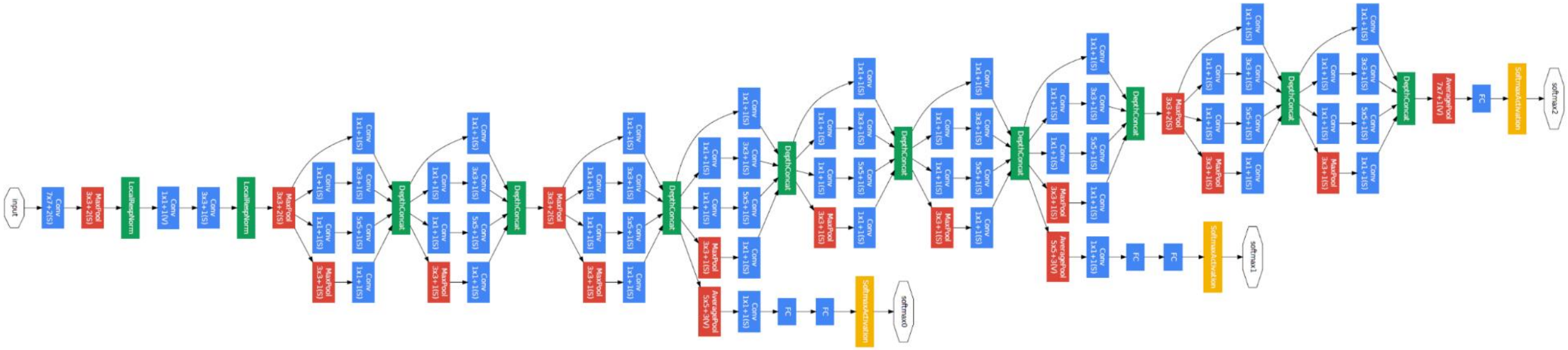
→ e.g. quadratic increase of computation

$$3 \times 3 \times C \rightarrow 3 \times 3 \times C\text{: } C^2 \text{ computations}$$

Conv filter 수가 증가하면 계산량이 제곱으로 증가

⇒ 파라미터 수를 늘리면 computing power가 많이 필요하고, Vanishing 문제 발생

GoogLeNet:
Deeper networks, with computational efficiency



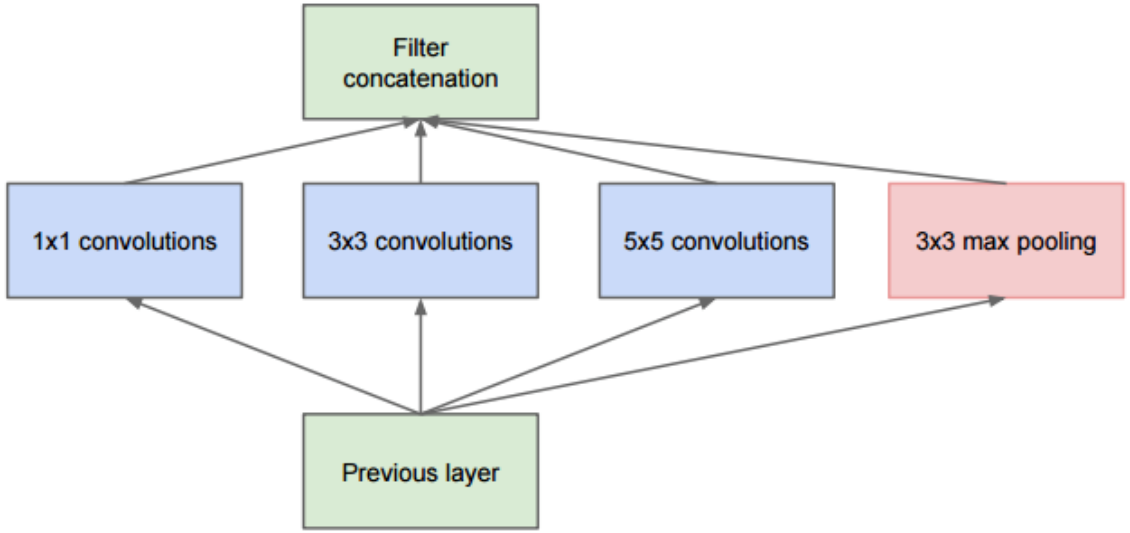
- **Deeper networks** (22 layers)
- **"Inception" module** 첫 도입
- **No FC layers** (Google + LeNet)
- **파라미터 개수 감소** (12x less than AlexNet)

Abstract

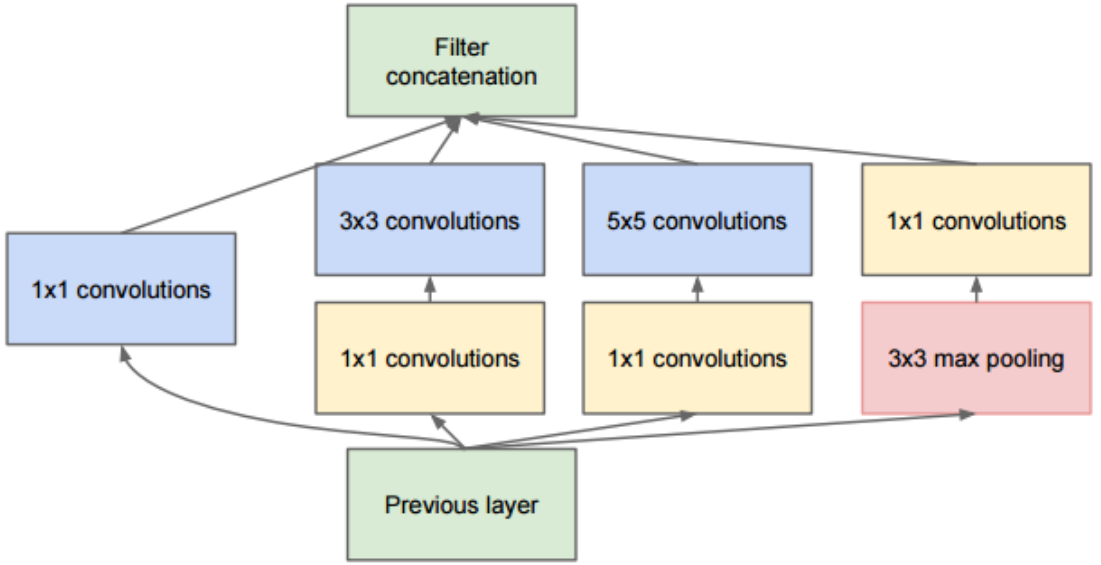
We propose a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved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

⇒ GoogLeNet은 Inception 구조를 통해 depth & width를 늘리면서도 연산량은 유지

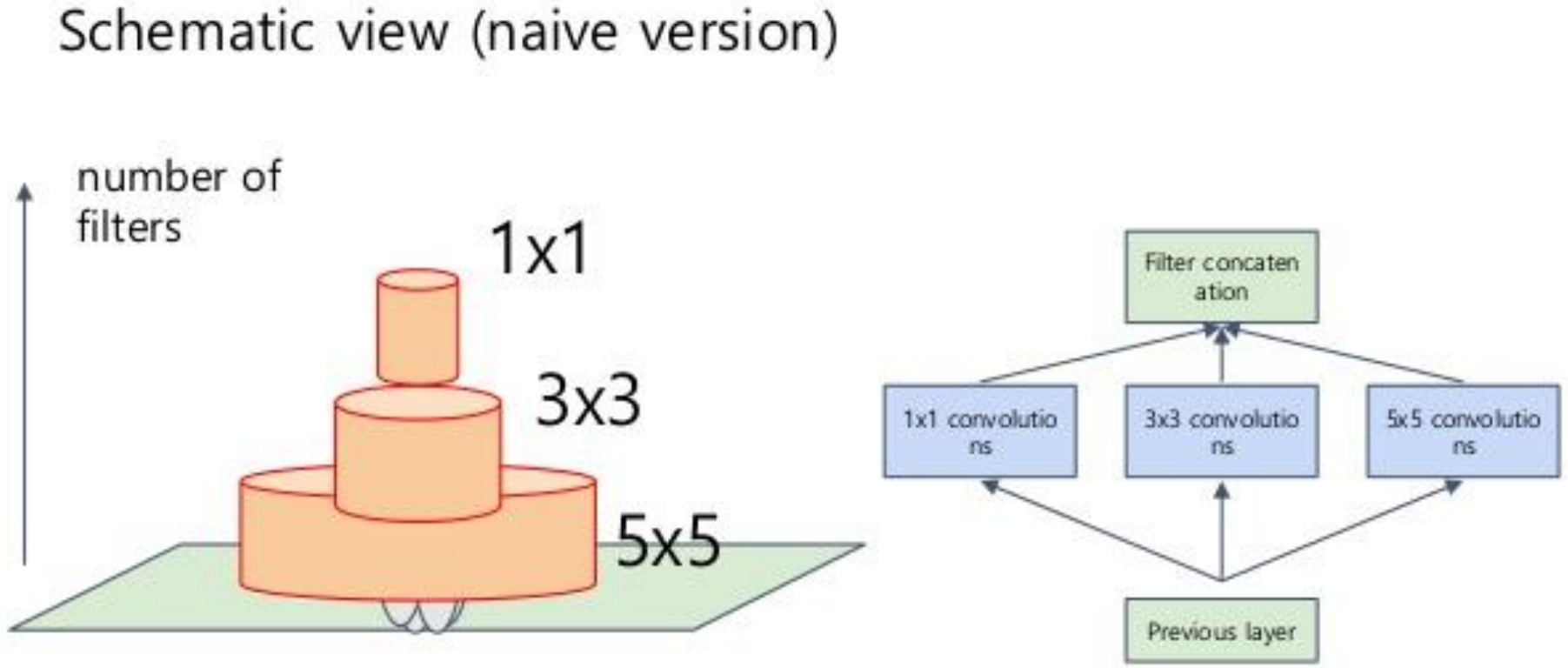
What is Inception module?



(a) Inception module, naïve version



(b) Inception module with dimension reductions

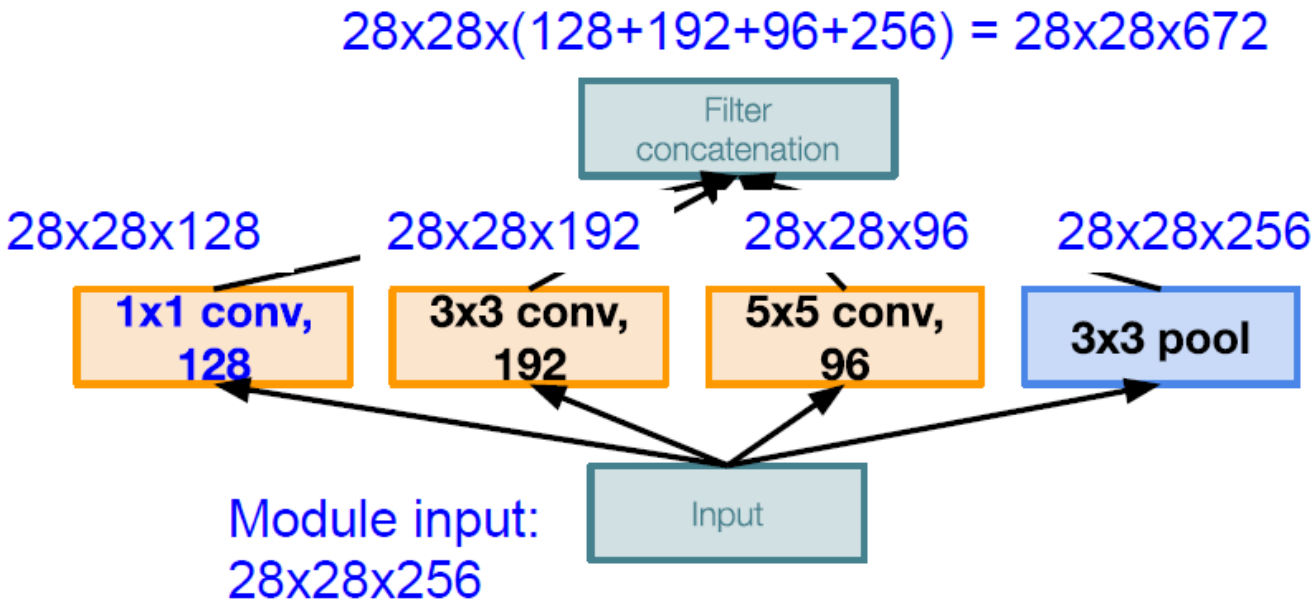


다양한 크기의 Conv Filters (1x1, 3x3, 5x5)를 병렬로 사용하여 다양한 features를 뽑음

좀 더 다양한 resolution을 학습하기 위해 3x3 max pooling 추가 (CNN 구조)

⇒ 이를 통해, Input features에서 의미 있는 features를 뽑아낼 수 있음

왜 (a) 모델이 아니라 (b) 모델을 사용했는가?

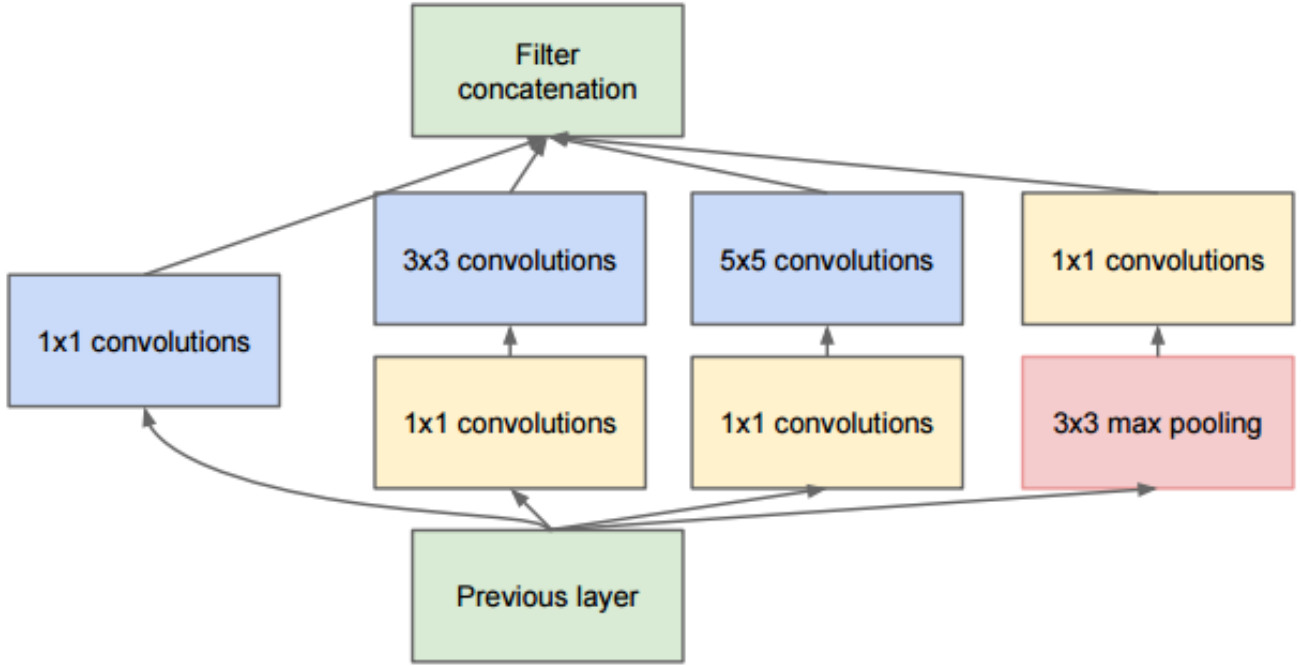


Naive Inception module

Inception module을 거칠수록 depth 증가 (Pooling layer는 depth를 변화시키지 않음)

⇒ **Naïve version은 계산해야 하는 파라미터 수가 854M로 너무 많음**

해결법: 1x1 Conv “Bottleneck” layers 추가

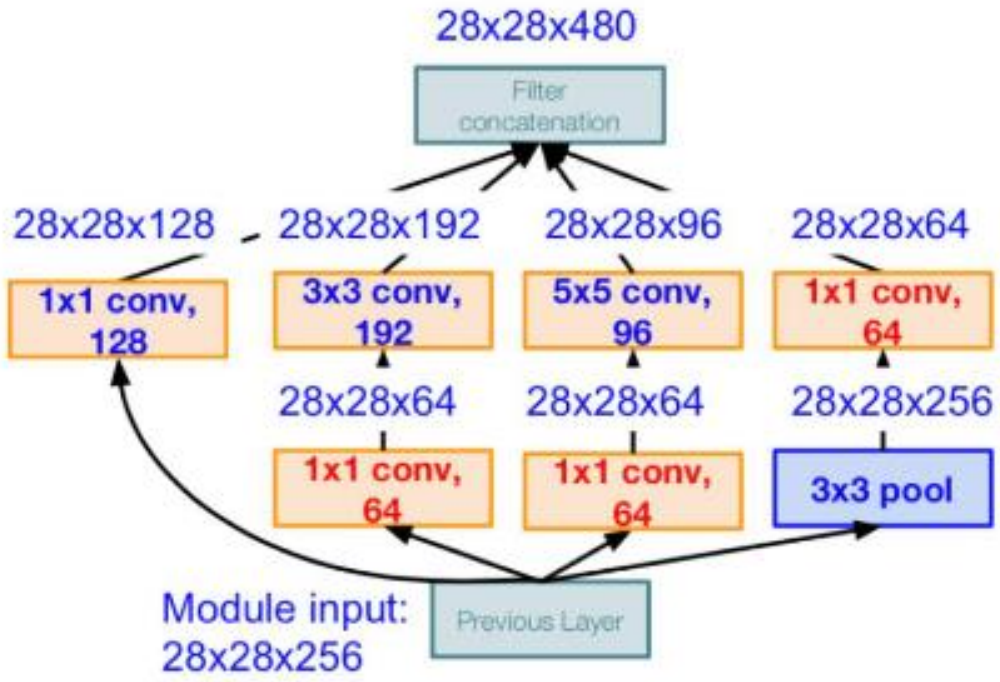


(b) Inception module with dimension reductions

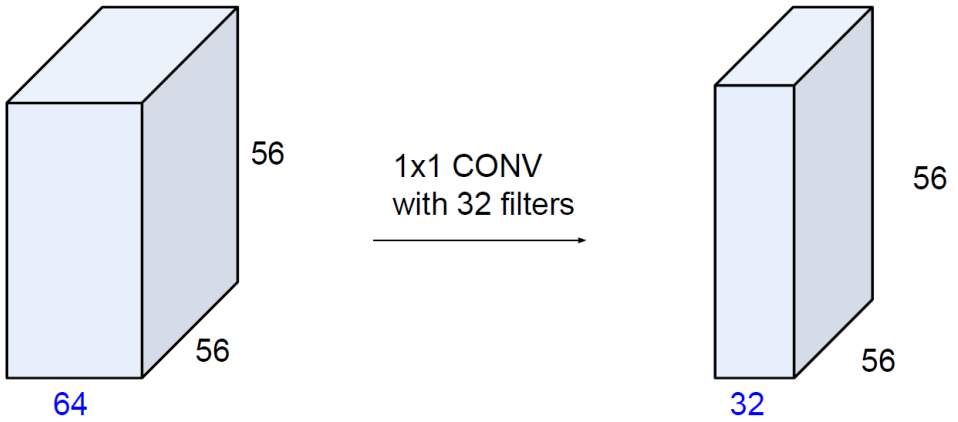
1x1 conv filter를 추가해 channel 수 조절

⇒ **channel reduction과 ReLU activation function을 통해 non-linearity를 부여하는 효과**

1x1 Conv의 효과



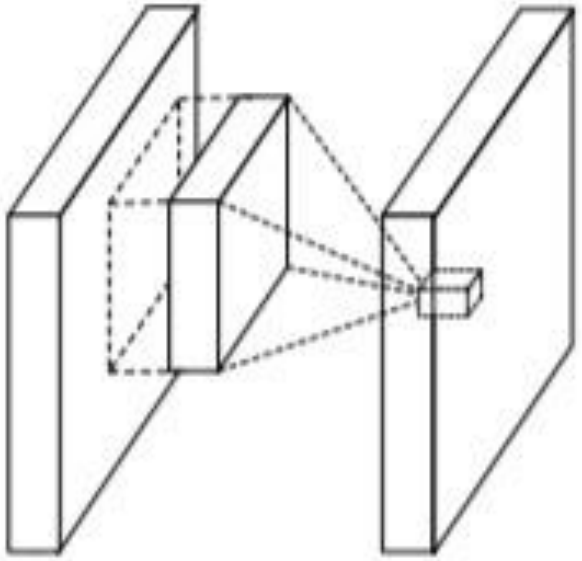
Inception module with dimension reduction



1x1 conv filter는 filter의 x, y는 변경시키지 않지만,

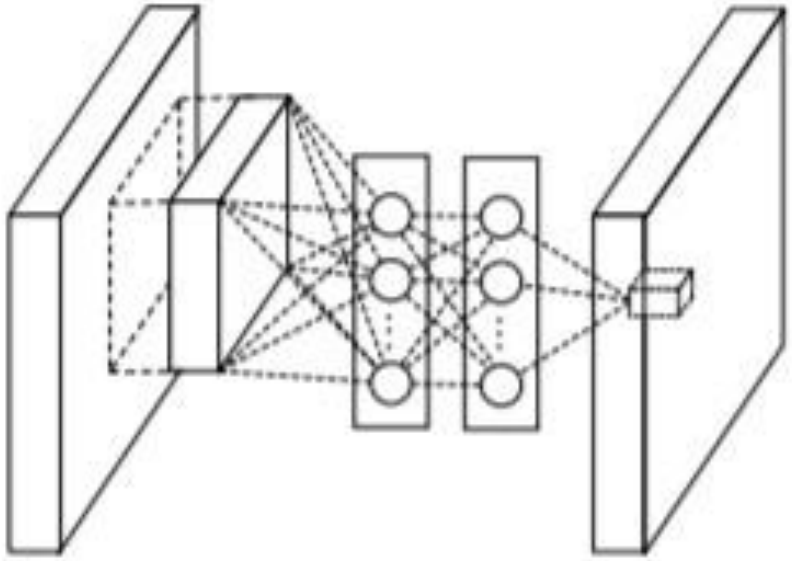
channel를 바꿔줌으로써 연산량을 줄임 **(계산해야 하는 파라미터 수가 358M로 줄어듦)**

관련 개념: Idea of NIN (Network in network)



(a) Linear convolution layer

선형 결합



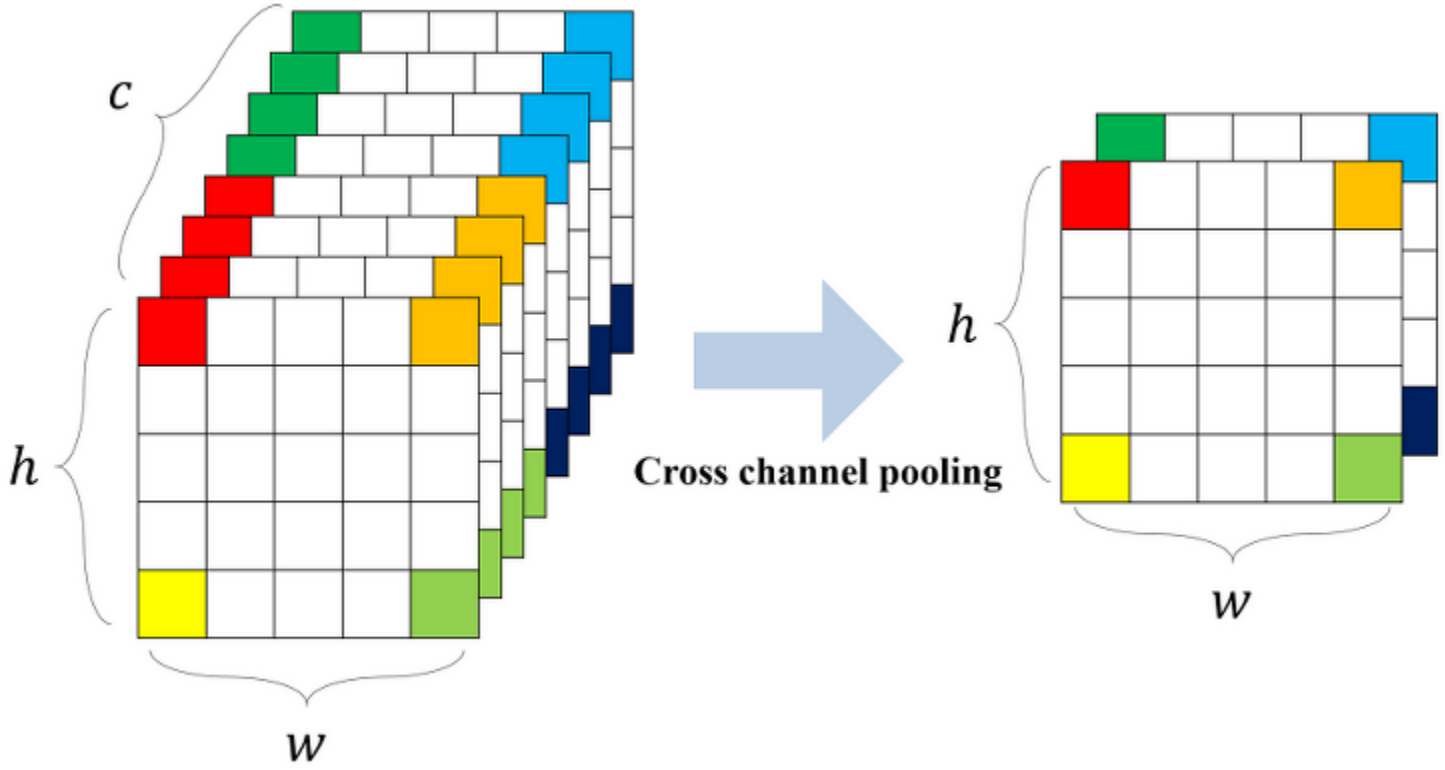
(b) Mlpconv layer

비선형 결합

Conv filter를 이용해 feature를 추출할 때, 중간에 MLP를 더해주어

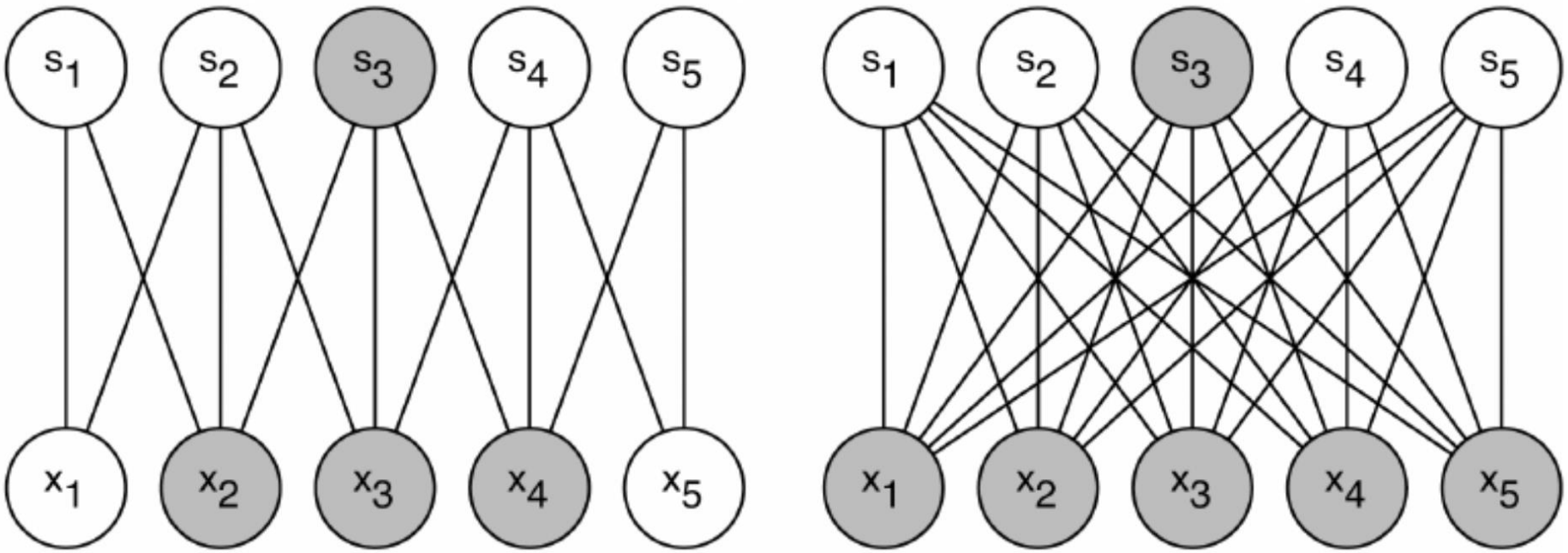
CNN 모델이 non-linearity를 가지게 됨

CCCP (Cascaded Cross Channel Pooling)

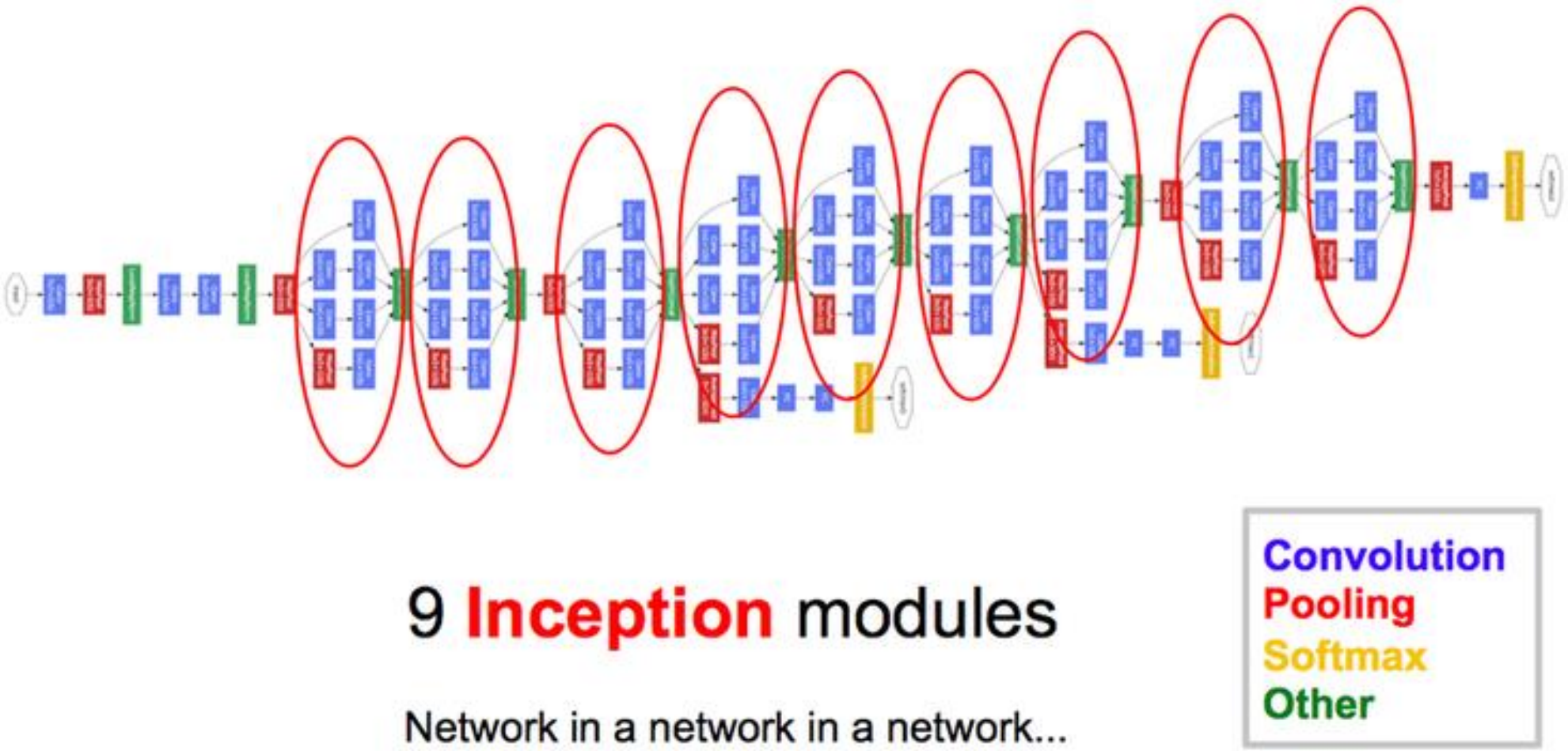


몇 개의 channel을 직렬(cascade) 묶음으로 여러 위치에서 pooling을 하는데, Feature map size는 그대로, channel 수만 줄어든게 한다. 이 아이디어가 1x1 convolution filter로, Inception module에 적용

Sparse Connection



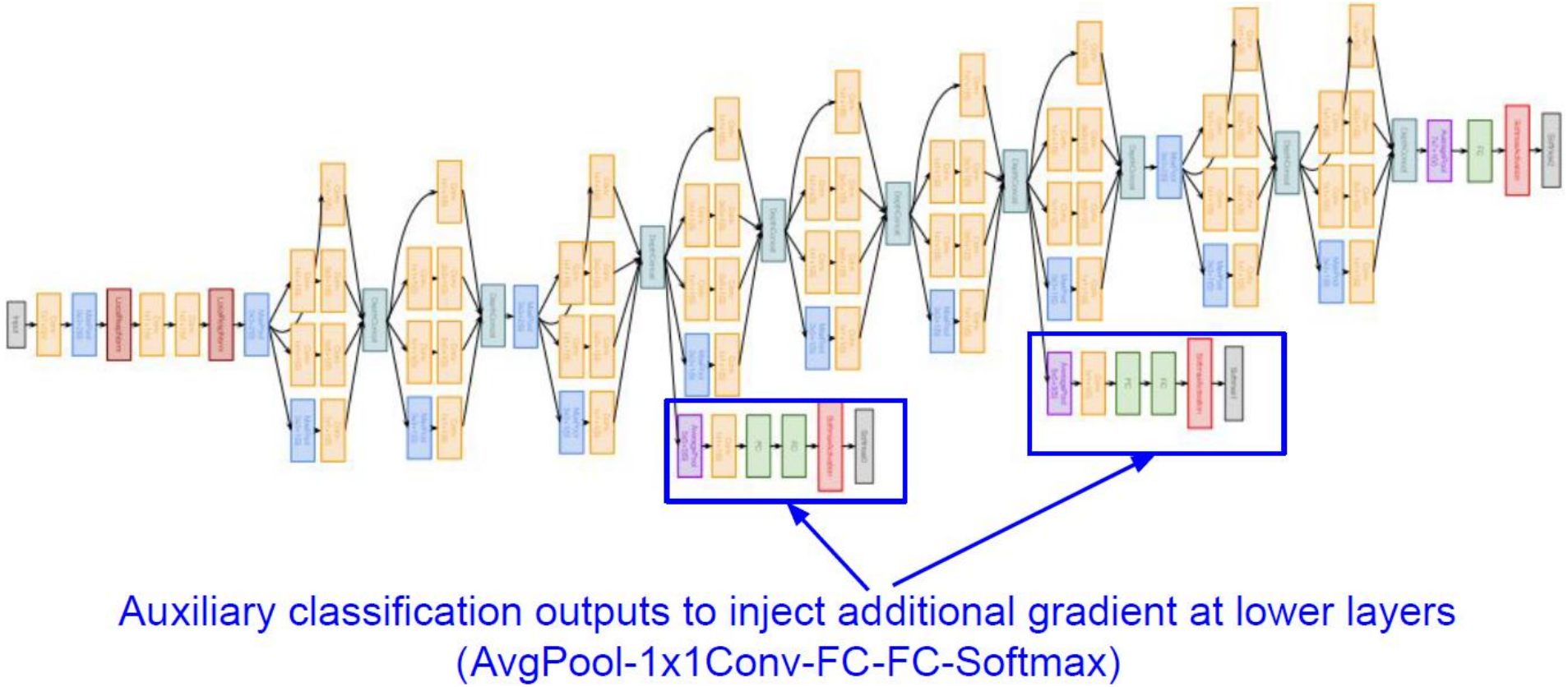
상관성(correlation)이 높은 노드끼리만 연결하는데, 연결을 Sparse하게 바꾸어서 연산량, 파라미터를 줄이고, Computational resource를 적게 사용하는 방법



층 내부에서 상관성(correlation)이 높은 것끼리 clustering

결과적으로 sparse하게 correlated되어 있는 conv filters 통과

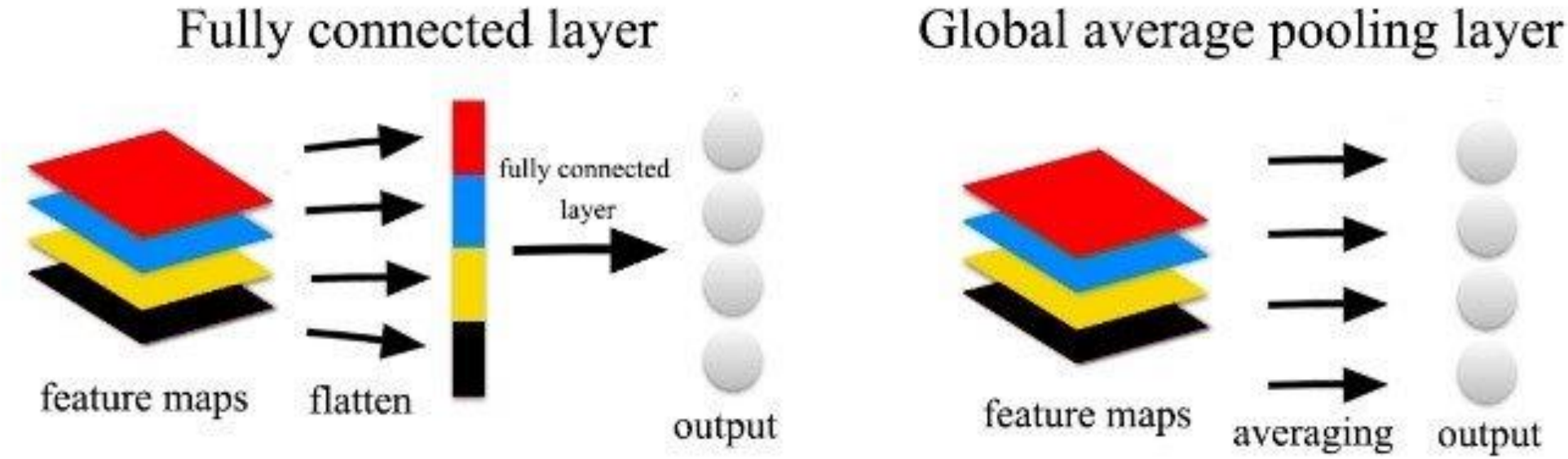
Auxiliary Classifier (보조 분류기)



모델 중간 중간 끊어주며 FC layer와 Softmax로 output 생성, 추가적으로 Gradient 계산 가능

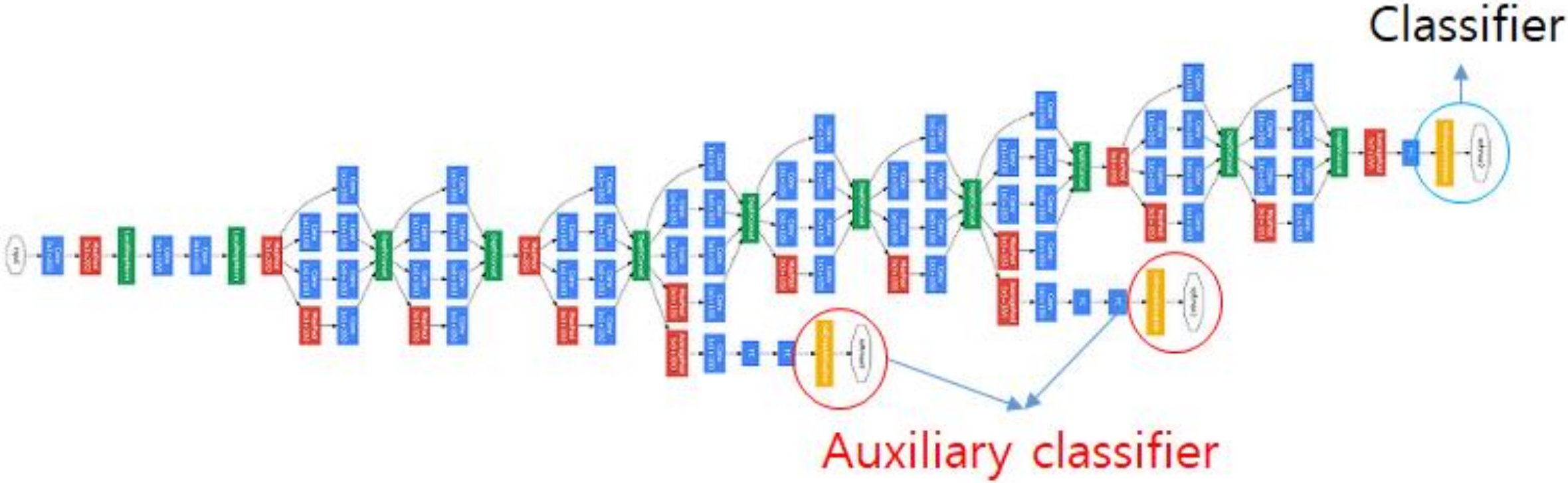
⇒ **깊은 depth로 인해 생기는 Gradient Vanishing 문제 해결**

Global Average Pooling



마지막에 데이터 라벨 개수에 맞는 출력 (분류)을 위해 펴주는 layer, FCN 대신 사용

Avaerage Pooling만으로도 classifer 역할을 할 수 있어 overfitting의 문제 해결


Classifier
Auxiliary classifier

전체 모델 구조

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

Validation: Multi-Crop

- 여러 가지 scale을 사용 (256 외에도 288, 320, 352, 총 4개)
- 각각의 scale에서 3장씩 이미지 선택 (3장)
- 선택된 이미지로부터 4개의 코너 및 센터 2개를 추출 (6장)
- 미러링, 좌우반전 (2장)

⇒ 4 x 3 x 6 x 2 = 144개의 이미지 사용

최종 실험 결과

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

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

⇒ Multi-Crop을 통해 Validation을 수행했을 때, 7개의 모델로 앙상블했을 때 가장 좋은 성능