

#1. Going deeper with Convolutions

1. 당시 트렌드

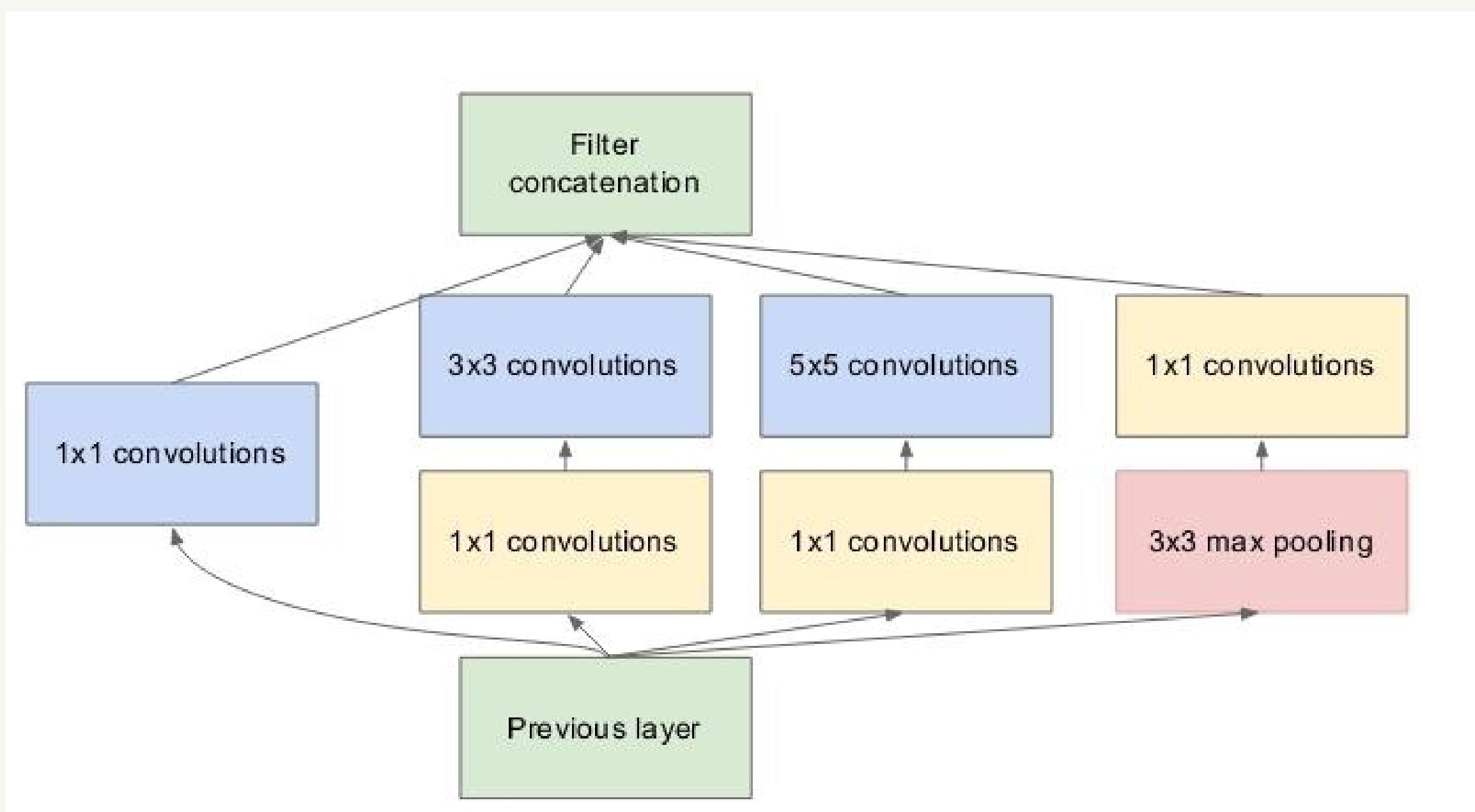
- ① 레이어의 수 & 레이어 사이즈를 늘림
- ② 오버피팅을 방지하기 위해 dropout 사용

⇒ 문제점

- ① parameter \propto \rightarrow overfitting
- ② 계산량은 많아지지만, 계산자원은 제한되어 있음

⇒ Inception module

2. Inception module



- ① Feature map 을 효과적으로 추출하기 위해 $1 \times 1, 3 \times 3, 5 \times 5$ convolution 연산 수행
 - ② 3×3 max pooling은 convolution network의 성공에 필수적
이어서 pooling path를 병렬적으로 추가
- * 문제: $3 \times 3, 5 \times 5$ convolution 연산도 계산량 多

3. 1×1 convolution의 중요성

$$14 \times 14 \times 480 \xrightarrow[48]{5 \times 5} 14 \times 14 \times 48$$

$$\# \text{ of operation} : (14 \times 14 \times 48) * (5 \times 5 \times 480) = \underbrace{112.9M}_{\text{Red}}$$

$$14 \times 14 \times 480 \xrightarrow[\frac{1}{16}]{} 14 \times 14 \times 16 \xrightarrow[\frac{48}{5}]{} 14 \times 14 \times 48$$

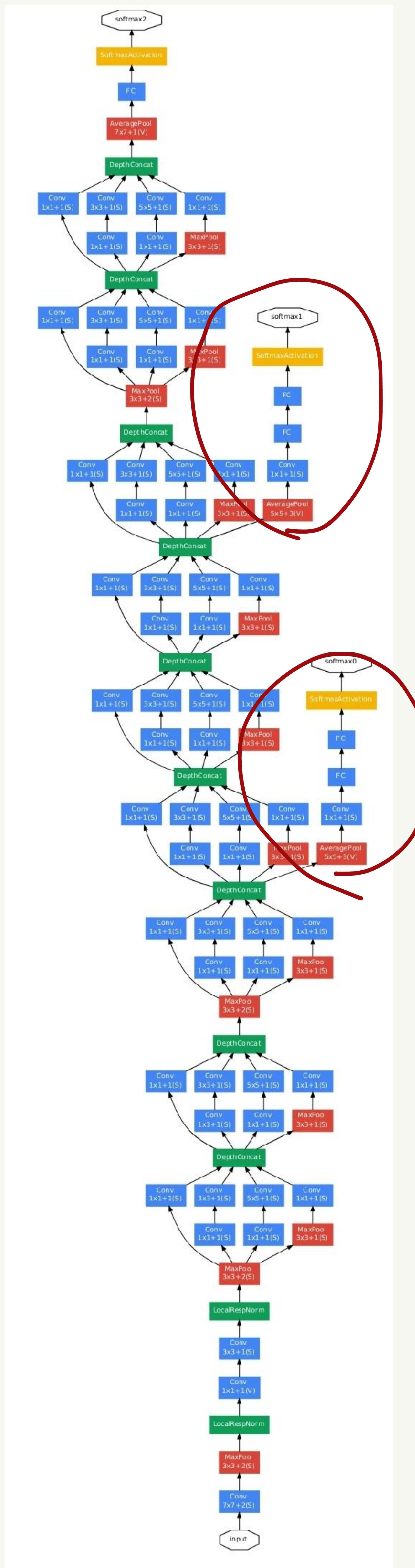
$$\begin{aligned} \# \text{ of operation} &: (14 \times 14 \times 16) * (1 \times 1 \times 480) = 1.5M \\ &+ (14 \times 14 \times 48) * (5 \times 5 \times 16) = 3.8M \\ &\Rightarrow \underbrace{5.3M}_{\text{Red}} \end{aligned}$$

\therefore 계산을 줄이기 위해 차원 축소 모듈로 사용됨.

computation bottleneck을 층임으로써 깊이와 너비를
늘릴 수 있음 병목현상

\Rightarrow 당시 트렌드를 따라감

4. GoogLeNet



① Auxiliary classifier (보조분류기)

- 중간단계에서 예측
- 깊은 네트워크에서 학습이 잘 안될 수 있는 점을 보완 (추가적인 gradient 부여)

② Auxiliary classifier의 구조

- filtersize가 5x5이고 stride가 3인 average pooling layer
- Dimension reduction을 위한 1x1 conv layer 및 ReLU
- FC layer 및 ReLU
- Dropout layer
- Linear layer와 softmax를 사용한 1000-class classifier

③ 성능 높이기 ↴ 학습데이터 순서에 따른 차이

- 동일한 GoogLeNet 모델의 7가지 버전 → ensemble
- cropping 방식 : 이미지 스케일 미러링..
- final prediction은 multiple crop / all the individual classifier에 대한 softmax probability 평균으로

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%

5. 결론

- ① 계산량이 조금 증가하는 것에 비해 성능이 아주 좋아짐
- ② AlexNet보다 2배 적은 parameter를 사용하면서, 더 좋은 성능을 보임

→ 2012년 대회 1등